Universidade do Minho
Mestrado Integrado em Engenharia Informática
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Lectivo de 2017/2018

Centro Hípico Quintas Ferreira

Diana Barbosa A78679

Fábio Gonçalves A78793

Gonçalo Camaz A76861

José Oliveira A78806

Janeiro 2018

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Centro Hípico Quintas Ferreira

Diana Barbosa A78679

Fábio Gonçalves A78793

Gonçalo Camaz A76861

José Oliveira A78806

Janeiro 2018

Resumo

No ano letivo 2017/2018 foi proposto, no âmbito da unidade curricular de Bases de Dados, desenvolver um sistema de base de dados de um tema à nossa escolha. O grupo optou então por fazer uma base de dados relativa a um centro hípico na qual se registam todos os seus elementos organizados pela modalidade que praticam. No primeiro trabalho prático realizado no âmbito desta unidade curricular, desenvolvemos uma Base de Dados Relacional para gerir o centro. Agora, iremos apresentar uma alternativa, uma Base de Dados não Relacional com o mesmo intuito de poder gerir a informação relativa a alunos, cavalos, instrutores, inscrições, etc.

Primeiramente é apresentada a contextualização do nosso problema. De seguida, uma fundamentação para a alteração da base de dados no contexto do centro hípico. Mostramos também como chegamos ao modelo da nova base de dados que vamos usar, assim como foi efetuado a migração de dados existentes na base de dados antiga para a nova. Por fim, são apresentados queries equivalentes às queries desenvolvidas no trabalho anterior.

Área de Aplicação: Desenho e arquitetura de um Sistemas de Bases de Dados não relacional orientada a documentos no âmbito de uma aplicação responsável por registar todos os elementos do centro hípico como também as modalidades que estes praticam.

Palavras-Chave: Base de Dados, Sistema de Gestão de Base de Dados, Base de Dados não Relacional, SQL, NoSQL, MongoDB, JSON.

Índice

1. Introdução

1.1. Contextualização

O Centro Hípico Quintas Ferreira, fundado em 1997 pelo Sr. Quintas possui neste momento vários alunos inscritos e disponibiliza diversas modalidades para estes aprenderem. Cada modalidade é lecionada por instrutores que se dedicam apenas a ensinar essa mesma, e possui também cavalos apropriados e apenas usados para uma modalidade específica.

Para guardar estas informações relativas a alunos, cavalos, instrutores, modalidades e inscrições, de modo a melhorar a logística do dia-a-dia do centro hípico, o Sr. Quintas recorreu a uma equipa de programadores que lhe sugeriram uma aplicação que guardasse toda esta informação numa base de dados, e que lhe permitisse responder a queries úteis (de modo a cumprir os requisitos com ele acordados).

O projeto foi aprovado e implementado, e trouxe muitos benefícios para o negócio, o que deixou o cliente bastante satisfeito.

1.2. Fundamentação da implementação da nova Base de Dados

Com o passar do tempo, o centro hípico foi sendo gerido pela base de dados relacional e o Sr. Quintas foi observando a sua evolução. Reparou que quando ocorria alguma alteração nas informações de uma modalidade, dos cavalos, dos alunos ou dos instrutores, como por exemplo um cavalo morrer, ou um instrutor passar a lecionar outra modalidade, perdia-se o registo destas informações, pois ou os registos deles eram apagados da tabela, ou as modificações efetuadas e não havia maneira de aceder à informação original.

Isto acontecia pelo facto de a informação ser guardada separadamente em tabelas, por ser uma base de dados relacional. Portanto, torna-se necessário fazer alterações que possam resolver este problema. A solução encontrada pela equipa de programação, passa por mudar a base de dados para uma não relacional orientada a documentos (nomeadamente MongoDB), na qual os documentos são unidades independentes e portanto podemos modifica-los sem afetar outros.

1.3. Análise da viabilidade do processo

A viabilidade do projeto pode ser justificada pelos seguintes itens:

- A base de dados implementada não é muito complexa, o que facilita a migração, não sendo necessários muitos recursos para completar o processo.
- O Sistema de Bases de Dados MongoDB é open-source, e, portanto, a sua gestão acarreta custos reduzidos.
- O facto de o Sr. Quintas agora poder ter registos de informação sem ser apenas a mais atual, permite uma melhor e mais eficiente gestão do seu negócio.

1.4. Estrutura do Relatório

No resto deste relatório, apresentamos uma explicação de bases de dados não relacionais e uma justificação pelo uso de MongoDB no capítulo 2.

No capítulo 3 mostramos como foi efetuado a migração dos dados do mySQL para o MongoDB, assim como as coleções resultantes.

Por fim, no capítulo 4 apresentamos queries no Mongo, equivalentes às queries desenvolvidas na primeira parte do trabalho em mySQL.

2. Sistema de Bases de Dados não Relacional

2.1. Opção por Sistemas NoSQL

Este trabalho consiste na implementação de um sistema de base de dados não relacional em MongoDB, aplicada a uma base de dados já existente. Assim, requer a migração dos dados da primeira base de dados, bem como o planeamento da estrutura da nova, e a criação de queries equivalentes, tendo sempre em atenção as mudanças necessárias e os cuidados a ter que advêm da mudança de paradigma de relacional para não relacional.

Um sistema de base de dados relacional permite preservar a integridade, consistência, durabilidade e isolamento dos dados, em parte devido à obrigatoriedade de definição prévia de uma estrutura que é rígida e assim, aquando da inserção, é feita uma verificação dos dados e, portanto, estes não são inseridos se não estiverem de acordo com os parâmetros definidos.

Estes sistemas são extremamente eficientes no armazenamento de dados estruturados, o que levou, ao longo dos anos, os SBDR a uma posição de predominância no mercado. Contudo isto não impediu o aparecimento de outras soluções diferentes, especialmente para alguns problemas que se opunham ao modelo relacional. Um exemplo na atualidade é o Facebook que, como sabemos, lida com uma enorme quantidade de dados variados, o que torna o manuseamento de uma estrutura relacional extremamente trabalhoso, demorado e dificultoso, daí a procura de uma nova solução.

Em suma, os SBDR por serem bastante rígidos podem dificultar e atrasar algumas alterações que se pretenda efetuar às estruturas. Isto poderia ir contra com o objetivo de implementações de grandes volumes de informação, onde a escalabilidade é cada vez mais procurada. Assim, as bases de dados não relacionais, vêm tentar resolver este problema, uma vez que apresentam uma maior flexibilidade neste ponto. De uma forma geral, podemos afirmar que os sistemas NoSQL(Not only SQL) tentam oferecer soluções a alguns problemas encontrados no modelo relacional. Estas bases de dados, em vez de guardarem informação em tabelas, utilizam outras estruturas, como por exemplo grafos(Neo4j), documentos(Mongo), etc.

2.2. Escolha do MongoDB

Para este projeto, tendo em conta a nossa abordagem NoSQL decidimos usar o MongoDB, um dos modelos mais utilizados na atualidade, o qual, segundo o site www.db-engines.com se encontra em 5º lugar no ranking mundial em janeiro de 2018.

Este, é um sistema não relacional de bases de dados orientado a documentos, muito utilizado por ser open-source, mas também pelas suas vantagens infracitadas. Os dados são armazenados em documentos JSON, com uma organização schema-free, isto é, a cada entrada não temos que nos preocupar em seguir um esquema (estrutura de dados, número e tipo de campos, …) previamente definido, como acontecia em MySQL. Aliás, os documentos de uma dada coleção não precisam de ter todos os mesmos campos, e podemos a qualquer momento acrescentar ou apagar algum.

Por outro lado, podemos estabelecer analogias entre alguns conceitos de MySQL e MongoDB. Por exemplo, uma tabela a uma coleção, uma linha a um documento, uma coluna a um campo e um JOIN a embedded documents.

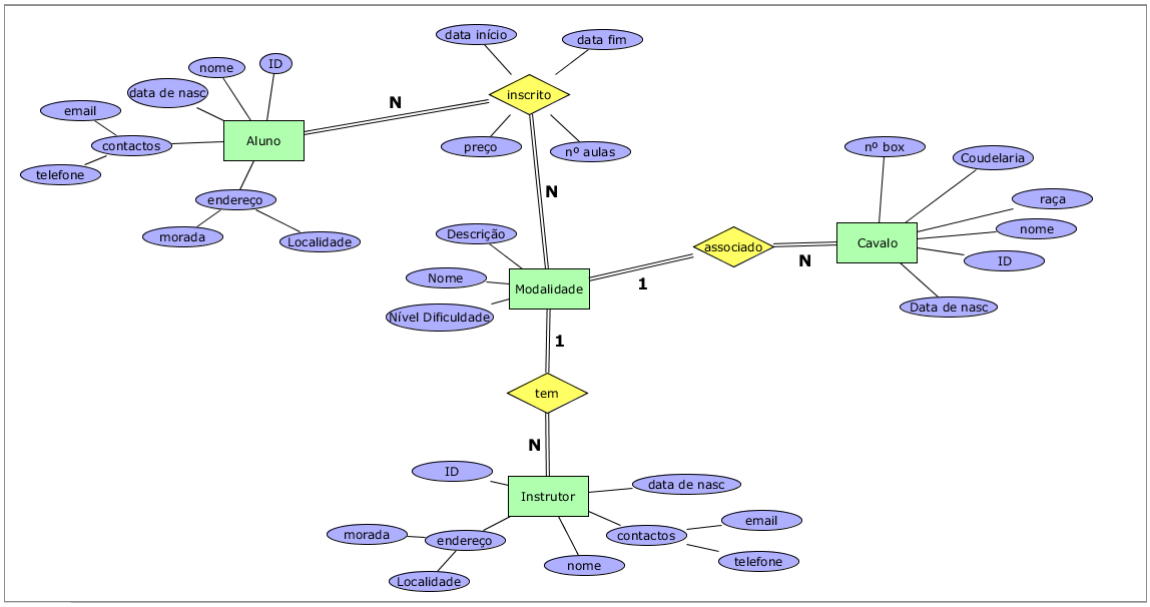
- **Vantagens**
    - **Alta performance:** Tal como a maioria das Bases de Dados NoSQL, o MongoDB está construído de modo a ser rápido e ter uma elevada taxa de transferência (throughput). Isto deve-se em parte ao facto de ser possível distribuir os documentos por vários clusters (máquinas). Por outro lado, inserções também são mais rápidas uma vez que como não há um esquema definido, e os documentos são independentes, não temos que validar o input, e, por conseguinte, a inserção é imediata.
    - **Facilidade nas operações:** Além da facilidade nas inserções referida no ponto 1, também as consultas e transações são mais simples uma vez que em mongo não existem joins, transações e relacionamentos como acontece em SQL.
    - **Facilidade na escalabilidade:** O crescimento da Base de Dados acarreta custos reduzidos (pois é open-source), e como os documentos podem ser distribuídos por várias máquinas (como referido no ponto 1), conseguimos uma maior escalabilidade.

- **Desvantagens**
    - **Redundância de dados:** Por não haverem relacionamentos, temos muitos dados repetidos, o que leva a redundância de dados e consequente desnecessária ocupação de memória.
    - **Inconsistência de dados:** Uma vez que não há um esquema definido não há controlo nas inserções e portanto a consistência dos dados não está garantida, tendo que ser assegurada pelo utilizador.

3. Migração

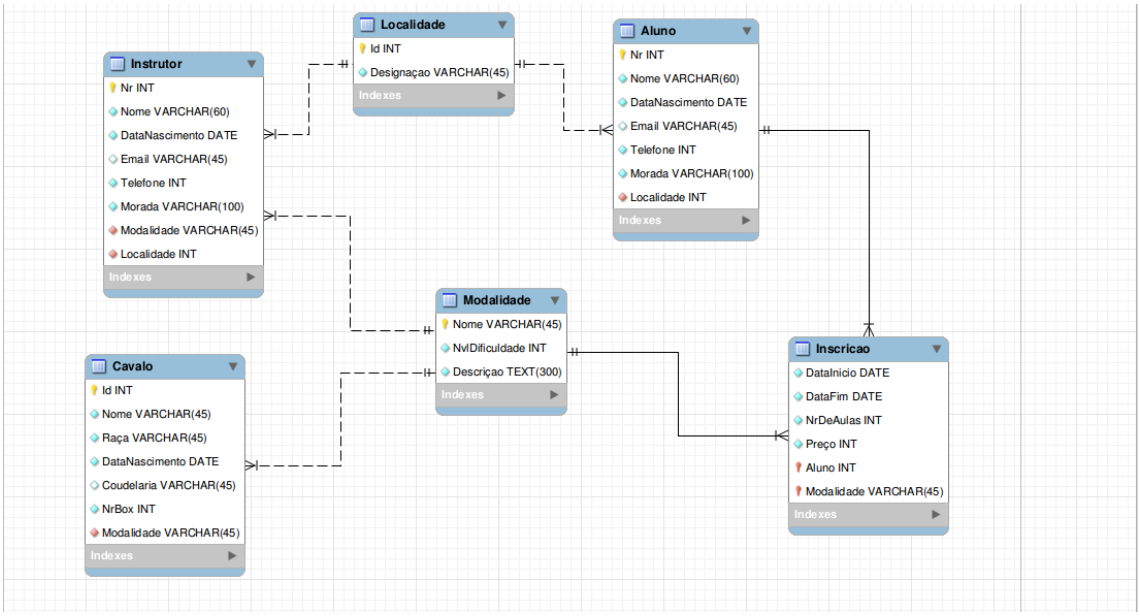
3.1. Diagrama ER Concetual

O diagrama ER do modelo concetual permanece inalterado, tendo em conta que as entidades, atributos e relacionamentos do problema se mantêm as mesmas.



3.2. Diagrama do Modelo Lógico

O diagrama do modelo lógico utilizado para criar a base de dados em SQL serve-nos de base para a criação das coleções de documentos em MongoDB, uma vez que a informação que consta nas relações terá que também estar presente na nova base de dados, pelo que será feita uma migração dos dados da base de dados original para a nova.



3.3. Migração dos Dados

Para conseguirmos migrar os dados, da base de dados relacional para a nova base de dados em MongoDB, inicialmente recorremos ao Mongify, um programa que faz automaticamente a migração tendo em contas as analogias que podem ser estabelecidas entre estes Sistemas de Bases de Dados. Assim, transformou as tabelas em coleções, as linhas em documentos, entre várias outras alterações necessárias.

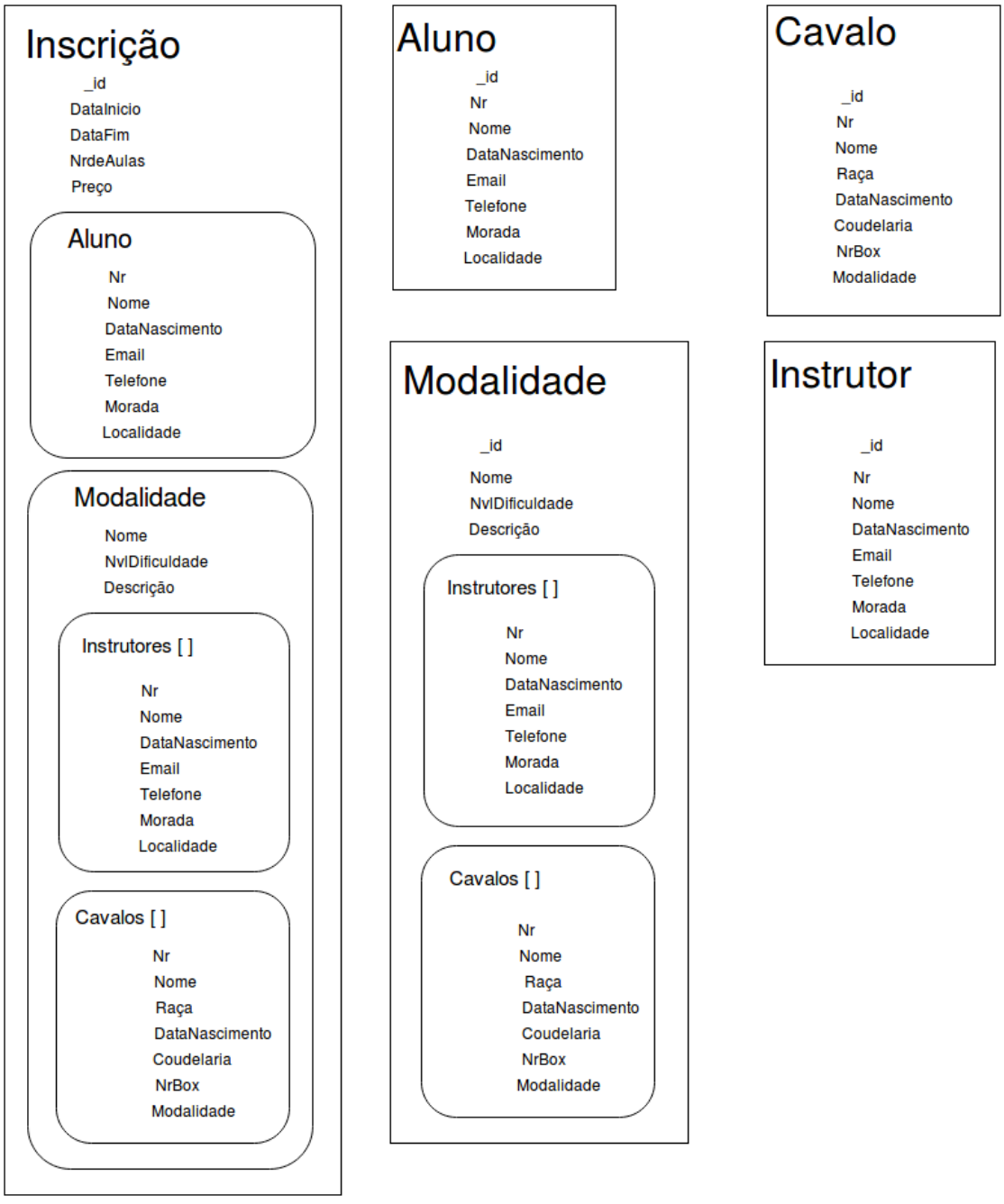
No entanto, não obtivemos o esquema que tínhamos idealizado para as coleções, já que certas coisas que faziam sentido na BDR, não são apropriadas para MongoDB. Por exemplo, em SQL optamos pela criação de uma tabela para a localidade em que a Primary Key é um inteiro, que é uma Foreign Key nas tabelas Aluno e Instrutor, já que no contexto do problema os valores iriam-se repetir muitas vezes e desta forma pouparíamos memória. Em MongoDB isto não se aplica, e, por conseguinte, queremos que na localidade apareça o nome em questão.

Para resolver este e outros problemas, optamos pela criação de um programa em Java que fizesse a migração de dados de modo a obtermos as coleções e documentos da maneira que achamos mais apropriada. O script do programa encontra-se nos Anexos deste relatório.

3.4. Coleções

No que concerne às coleções de documentos do nosso projeto, optamos pela criação de 5: Aluno, Instrutor, Cavalo, Modalidade e Inscrição.

Na de Aluno guardaremos os documentos com as informações relativas aos alunos que frequentam atualmente o centro hípico. Na de Instrutor, guardaremos os documentos com as informações relativas aos instrutores que lecionam modalidades atualmente no centro hípico. Na de Modalidade, guardaremos os documentos com as informações relativas às modalidades lecionadas atualmente no centro hípico, bem como os instrutores que as lecionam e os cavalos disponíveis para as aulas dessas mesmas modalidades. Por fim, na coleção Inscrição guardaremos os documentos com as informações referentes às inscrições dos alunos que frequentam/já frequentaram o centro hípico, bem como os dados do aluno e da modalidade à qual de inscreveu.



3.5. Documentos

```
COLEÇÃO: Aluno
{

    "_id": id atribuído pelo MongoDB
    "Nr": número de identificação do aluno
    "Nome": nome do aluno
    "DataNascimento": data de nascimento do aluno
    "Email": email do aluno
    "Telefone": telefone do aluno
    "Morada": morada do aluno
    "Localidade": localidade do aluno

}

COLEÇÃO: Cavalo
{
    "_id": id atribuído pelo MongoDB
    "Nr": número de identificação do cavalo
    "Nome": nome do cavalo
    "Raça": raça do cavalo
    "DataNascimento": data de nascimento do cavalo
    "Coudelaria": coudelaria do cavalo
    "NrBox": número da box do cavalo
    "Modalidade": modalidade do cavalo
}

COLEÇÃO: Instrutor
{
    "_id": id atribuído pelo MongoDB
    "Nr": número de identificação do instrutor
    "Nome": nome do instrutor
    "DataNascimento": data de nascimento do instrutor
    "Email": email do instrutor
    "Telefone": telefone do instrutor
    "Morada": morada do instrutor
    "Localidade": localidade do instrutor
}
```

```
COLEÇÃO: Modalidade
{
    "_id": id atribuído pelo MongoDB
    "Nome": nome da modalidade
    "NvlDificuldade": nível de dificuldade da modalidade
    "Descrição": descrição da modalidade
    "Instrutores": [
        {
            "Nr": número de identificação do instrutor
            "Nome": nome do instrutor
            "DataNascimento": data de nascimento do instrutor
            "Email": email do instrutor
            "Telefone": telefone do instrutor
            "Morada": morada do instrutor
            "Localidade": localidade do instrutor
        }
    ]
    "Cavalos": [
        {
            "Nr": número de identificação do cavalo
            "Nome": nome do cavalo
            "Raça": raça do cavalo
            "DataNascimento": data de nascimento do cavalo
            "Coudelaria": coudelaria do cavalo
            "NrBox": número da box do cavalo
            "Modalidade": modalidade do cavalo
        }
    ]
}
```

```
COLEÇÃO: Inscrição
{
    "_id": id atribuído pelo MongoDB
    "DataInicio": data da inscrição
    "DataFim": prazo de validade da inscrição
    "NrdeAulas": número de aulas da inscrição
    "Preço": preço da inscrição
    "Aluno":
    {
        "Nr": número de identificação do aluno
        "Nome": nome do aluno
        "DataNascimento": data de nascimento do aluno
        "Email": email do aluno
        "Telefone": telefone do aluno
        "Morada": morada do aluno
        "Localidade": localidade do aluno
    }

    COLEÇÃO: Modalidade
    {
        "_id": id atribuído pelo MongoDB
        "Nome": nome da modalidade
        "NvlDificuldade": nível de dificuldade da modalidade
        "Descrição": descrição da modalidade
        "Instrutores": [
            {
                "Nr": número de identificação do instrutor
                "Nome": nome do instrutor
                "DataNascimento": data de nascimento do instrutor
                "Email": email do instrutor
                "Telefone": telefone do instrutor
                "Morada": morada do instrutor
                "Localidade": localidade do instrutor
            }
        ]
        "Cavalos": [
            {
                "Nr": número de identificação do cavalo
                "Nome": nome do cavalo
                "Raça": raça do cavalo
                "DataNascimento": data de nascimento do cavalo
                "Coudelaria": coudelaria do cavalo
                "NrBox": número da box do cavalo
                "Modalidade": modalidade do cavalo
            }
        ]
    }
}
```

4. Queries

4.1. Query nº1

Esta query devolve todas as informações relativas aos cavalos associados a uma modalidade (neste caso “Iniciação”). É de notar, que no mySQl, os resultados são ordenados por nome de cavalo, enquanto que no Mongo não é possível fazer a ordenação por String, encontrando-se, assim, desordenado.

```
1  USE CentroHipico;
2
3  /* Saber todas as informações relativas aos cavalos associados a determinada
6  DROP PROCEDURE IF EXISTS info_cavalos_modalidade
7  DELIMITER $$
8  CREATE PROCEDURE info_cavalos_modalidade (nome_modalidade VARCHAR(45))
9  BEGIN
10   SELECT * FROM Cavalo
11     WHERE Modalidade = nome_modalidade
12     ORDER BY Nome;
13 END $$
14 DELIMITER ;
15
16 CALL info_cavalos_modalidade('Iniciação');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

#	Id	Nome	Raca	DataNascimento	Coudelaria	NrBox	Modalidade
1	29	Amado	Cruzado Português	2004-03-12	Coudelaria Quintas Ferreira	30	Iniciação
2	25	Animado	Cruzado Português	2008-11-12	Coudelaria Nacional	25	Iniciação
3	28	Benâncio	Cruzado Português	2002-01-12	Coudelaria Quintas Ferreira	28	Iniciação
4	30	Bonito	Cruzado Português	1996-04-10	Coudelaria Nacional	29	Iniciação
5	43	Dinarte	Garrano	2004-10-20	Coudelaria José Bernardo	43	Iniciação
6	42	Duquesa	Garrano	2004-10-20	Coudelaria José Bernardo	42	Iniciação
7	27	Queimado	Cruzado Português	2000-06-12	Coudelaria José Quim	27	Iniciação
8	26	Tristonho	Cruzado Português	2000-01-15	Coudelaria José Fernando	26	Iniciação

```
> db.Modalidade.find({"Nome":"Iniciação"},{Cavalos: 1}).pretty()
{
        "_id" : ObjectId("5a5e314c23efb950f80a7fe6"),
        "Cavalos" : [
                {
                        "Numero" : 25,
                        "Nome" : "Animado",
                        "Raca" : "Cruzado Português",
                        "DataNascimento" : "2008-11-12",
                        "Coudelaria" : "Coudelaria Nacional",
                        "NrBox" : 25
                },
                {
                        "Numero" : 26,
                        "Nome" : "Tristonho",
                        "Raca" : "Cruzado Português",
                        "DataNascimento" : "2000-01-15",
                        "Coudelaria" : "Coudelaria José Fernando",
                        "NrBox" : 26
                },
                {
                        "Numero" : 27,
                        "Nome" : "Queimado",
                        "Raca" : "Cruzado Português",
                        "DataNascimento" : "2000-06-12",
                        "Coudelaria" : "Coudelaria José Quim",
                        "NrBox" : 27
                },
                {
                        "Numero" : 28,
                        "Nome" : "Benâncio",
                        "Raca" : "Cruzado Português",
                        "DataNascimento" : "2002-01-12",
                        "Coudelaria" : "Coudelaria Quintas Ferreira",
                        "NrBox" : 28
                },
                {
                         "Numero" : 29,
                         "Nome" : "Amado",
                         "Raca" : "Cruzado Portugês",
                         "DataNascimento" : "2004-03-12",
                         "Coudelaria" : "Coudelaria Quintas Ferreira",
                         "NrBox" : 30
                },
                {
                         "Numero" : 30,
                         "Nome" : "Bonito",
                         "Raca" : "Cruzado Português",
                         "DataNascimento" : "1996-04-10",
                         "Coudelaria" : "Coudelaria Nacional",
                         "NrBox" : 29
                },
                {
                         "Numero" : 42,
                         "Nome" : "Duquesa",
                         "Raca" : "Garrano",
                         "DataNascimento" : "2004-10-20",
                         "Coudelaria" : "Coudelaria José Bernardo",
                         "NrBox" : 42
                },
                {
                         "Numero" : 43,
                         "Nome" : "Dinarte",
                         "Raca" : "Garrano",
                         "DataNascimento" : "2004-10-20",
                         "Coudelaria" : "Coudelaria José Bernardo",
                         "NrBox" : 43
                }
        ]
```

4.2. Query nº2

Nesta query, pretendemos descobrir, para uma determinada modalidade (neste caso "Dressage"), quais são os instrutores que a lecionam.

```
4   DROP PROCEDURE IF EXISTS instrutores_de_modalidade
5
6   DELIMITER $$
7   CREATE PROCEDURE instrutores_de_modalidade (nome_modalidade VARCHAR(45))
8   BEGIN
9   SELECT Nome FROM Instrutor
10        WHERE Modalidade = nome_modalidade;
11  END $$
12  DELIMITER ;
13
14  CALL instrutores_de_modalidade('Dressage');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

#	Nome
1	Miguel Bastos Nogueira

```
> db.Modalidade.find({"Nome":"Dressage"},{Cavalos:0}).pretty();
{
        "_id" : ObjectId("5a5e314c23efb950f80a7fe3"),
        "Nome" : "Dressage",
        "Descricao" : "Exercícios de alta escola a cavalo",
        "NvlDificuldade" : 3,
        "Instrutores" : [
                {
                        "Numero" : 3,
                        "Nome" : "Miguel Bastos Nogueira",
                        "DataNascimento" : "1980-12-12",
                        "Email" : "slbforever1994@gmail.com",
                        "Telefone" : 933121678,
                        "Morada" : "Rua dos Bacalheiros,nº411,1ºesq",
                        "Localidade" : "Maia"
                }
        ]
}
```

4.3. Query nº3

Nesta query, queremos saber a box a que cada cavalo está alocado.

```
3
4 •  SELECT Nome,NrBox FROM Cavalo
5          ORDER BY NrBox;
6
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

#	Nome	NrBox
1	Portugal	1
2	Brasil	2
3	Fidalgo	3
4	Amadeus	4
5	Janota	5
6	Quinado	6
7	Iman	7
8	Quina	8
9	Thunder	9
10	McDonalds	10
11	Moncherie	11
12	Rocher	12
13	Fire	13
14	Beatles	14
15	Eiffel	15

```
> db.Cavalo.find({}, {Nome:1 , "NrBox":1, _id:0}).pretty()
{ "Nome" : "Portugal", "NrBox" : 1 }
{ "Nome" : "Brasil", "NrBox" : 2 }
{ "Nome" : "Amadeus", "NrBox" : 4 }
{ "Nome" : "Fidalgo", "NrBox" : 3 }
{ "Nome" : "Janota", "NrBox" : 5 }
{ "Nome" : "Iman", "NrBox" : 7 }
{ "Nome" : "Quinado", "NrBox" : 6 }
{ "Nome" : "Quina", "NrBox" : 8 }
{ "Nome" : "Thunder", "NrBox" : 9 }
{ "Nome" : "McDonalds", "NrBox" : 10 }
{ "Nome" : "Fire", "NrBox" : 13 }
{ "Nome" : "Moncherie", "NrBox" : 11 }
{ "Nome" : "Rocher", "NrBox" : 12 }
{ "Nome" : "Eiffel", "NrBox" : 15 }
{ "Nome" : "Pierre", "NrBox" : 17 }
{ "Nome" : "Antoniette", "NrBox" : 16 }
{ "Nome" : "Marie", "NrBox" : 18 }
{ "Nome" : "Moncherie", "NrBox" : 19 }
{ "Nome" : "Jack", "NrBox" : 20 }
{ "Nome" : "Rose", "NrBox" : 21 }
Type "it" for more
> it
{ "Nome" : "Titanic", "NrBox" : 22 }
{ "Nome" : "Darth Vader", "NrBox" : 23 }
{ "Nome" : "Skywalker", "NrBox" : 24 }
{ "Nome" : "Beatles", "NrBox" : 14 }
{ "Nome" : "Animado", "NrBox" : 25 }
{ "Nome" : "Tristonho", "NrBox" : 26 }
{ "Nome" : "Queimado", "NrBox" : 27 }
{ "Nome" : "Benâncio", "NrBox" : 28 }
{ "Nome" : "Amado", "NrBox" : 30 }
{ "Nome" : "Bonito", "NrBox" : 29 }
{ "Nome" : "Molhado", "NrBox" : 31 }
{ "Nome" : "Farah", "NrBox" : 32 }
{ "Nome" : "Farid", "NrBox" : 33 }
{ "Nome" : "Fakir", "NrBox" : 34 }
{ "Nome" : "Atrevido", "NrBox" : 36 }
{ "Nome" : "Otica", "NrBox" : 35 }
{ "Nome" : "Parda", "NrBox" : 37 }
{ "Nome" : "Verissima", "NrBox" : 38 }
{ "Nome" : "Tormenta", "NrBox" : 39 }
{ "Nome" : "Duquesa", "NrBox" : 40 }
Type "it" for more
> it
{ "Nome" : "Dinamarca", "NrBox" : 41 }
{ "Nome" : "Duquesa", "NrBox" : 42 }
{ "Nome" : "Dinarte", "NrBox" : 43 }
{ "Nome" : "Cavalinho", "NrBox" : 44 }
{ "Nome" : "Eguazinha", "NrBox" : 45 }
```

4.4. Query nº4

Nesta query, queremos saber as modalidades a que um aluno (neste caso “Gonçalo Camaz Amoreira”) está inscrito e as datas de início e fim dessas inscrições.

```
1  USE CentroHipico;
2
3  /* Saber a que modalidades um aluno está inscrito e as datas de inicio e fim dessas inscrições */
4
5  DROP PROCEDURE IF EXISTS modalidades_de_aluno;
6  DELIMITER $$
7  CREATE PROCEDURE modalidades_de_aluno(nome_aluno VARCHAR(60))
8  BEGIN
9
10     SELECT Inscricao.Modalidade, Inscricao.DataInicio, Inscricao.DataFim FROM Inscricao
11         INNER JOIN Aluno AS A ON A.Nr = Inscricao.Aluno
12         WHERE A.Nome = nome_aluno;
13 END$$
14 DELIMITER ;
15
16 CALL modalidades_de_aluno('Gonçalo Camaz Amoreira');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

#	Modalidade	DataInicio	DataFim
1	Dressage	2017-05-01	2020-05-01
2	Equitação de Trabalho	2017-05-01	2019-05-01

```
> db.Inscricao.find({"Aluno.Nome": "Gonçalo Camaz Amoreira"},{_id: 0, DataInicio:1, DataFim:1, "Modalidade.Nome":1}).pretty()
{
        "DataInicio" : "2017-05-01",
        "DataFim" : "2020-05-01",
        "Modalidade" : {
                "Nome" : "Dressage"
        }
}
{
        "DataInicio" : "2017-05-01",
        "DataFim" : "2019-05-01",
        "Modalidade" : {
                "Nome" : "Equitação de trabalho"
        }
}
```

4.5. Query nº5

Nesta query, pretendemos saber a morada e o contacto telefónico dos alunos que estão inscritos numa modalidade e que moram numa determinada localidade. Neste caso utilizamos a modalidade “Salto” e a localidade “Braga”.

```
 6  DROP PROCEDURE IF EXISTS morada_e_contacto ;
 7  DELIMITER $$
 8  CREATE PROCEDURE morada_e_contacto (modalidade VARCHAR(45), nome_localidade VARCHAR(45))
 9  BEGIN
10      SELECT Aluno.Nome, Aluno.Morada, Aluno.Telefone FROM Aluno
11        INNER JOIN Inscricao AS I ON I.Aluno = Aluno.Nr
12        INNER JOIN Modalidade AS M ON M.Nome = I.Modalidade
13        INNER JOIN Localidade AS L ON L.Id = Aluno.Localidade
14           WHERE I.Modalidade = modalidade AND L.Designacao = nome_localidade;
15  END
16  $$
17  DELIMITER ;
18  CALL morada_e_contacto('Salto','Braga');
```

Result Grid Filter Rows: Export: Wrap Cell Content:

#	Nome	Morada	Telefone
1	Fábio Gonçalves Quintas	Rua D.Sebastião,nº 33, 1ºdir	253681212
2	Daniel Fernandes Amorim	Rua das Palmeirinhas, nº313	923341415

```
> db.Inscricao.find({"Modalidade.Nome": "Salto", "Aluno.Localidade": "Braga"},{_id: 0, "Aluno.Nome":1, "Aluno.Morada":1,"Aluno.Telefone":1}).pre
tty()
{
        "Aluno" : {
                "Nome" : "Fábio Gonçalves Quintas",
                "Morada" : "Rua D.Sebastião,nº 33, 1ºdir",
                "Telefone" : 253681212
        }
}
{
        "Aluno" : {
                "Nome" : "Daniel Fernandes Amorim",
                "Morada" : "Rua das Palmeirinhas, nº313",
                "Telefone" : 923341415
        }
}
```

4.6. Query nº6

Nesta query, pretendemos saber quantos alunos estão inscritos numa modalidade, neste caso “Corrida”.

```
1  Use CentroHipico;
2  /* Quantos alunos estão inscritos numa determinada modalidade */
3  DROP PROCEDURE IF EXISTS total_alunos_inscritos;
4  DELIMITER $$
5  CREATE PROCEDURE total_alunos_inscritos (modalidade VARCHAR(45))
6  BEGIN
7      SELECT Count(Aluno.Nome) AS Total_Alunos,I.Modalidade FROM Aluno
8          INNER JOIN Inscricao AS I ON I.Aluno = Aluno.Nr
9          INNER JOIN Modalidade AS M ON M.Nome = I.Modalidade
10             GROUP BY I.Modalidade
11             HAVING I.Modalidade = modalidade;
12 END$$
13 DELIMITER ;
14
15 CALL total_alunos_inscritos ('Corrida');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

#	Total_Alunos	Modalidade
1	4	Corrida

```
> db.Inscricao.find({"Modalidade.Nome": "Corrida"}).count()
4
```

4.7. Query nº 7

Nesta query, pretendemos saber o top 3 de modalidades com mais inscrições.

É de se notar que na nossa base de dados temos mais do que uma modalidade com 4 inscrições, daí serem apresentados dois resultados diferentes.

```
1  Use CentroHipico;
2
3  /* Top 3 das modalidades com mais inscrições */
4
5  SELECT Count(Aluno.Nome) AS Total_Alunos,I.Modalidade FROM Aluno
6     INNER JOIN Inscricao AS I ON I.Aluno = Aluno.Nr
7     INNER JOIN Modalidade AS M ON M.Nome = I.Modalidade
8           GROUP BY I.Modalidade
9           ORDER BY Total_Alunos DESC
10          LIMIT 3
11    ;
12
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

#	Total_Alunos	Modalidade
1	15	Iniciação
2	6	Salto
3	4	Equitação de Trabalho

```
> db.Inscricao.aggregate([{"$group":{_id: "$Modalidade.Nome", "Número de Inscrições":{$sum:1}}},{$sort:{"Número de Inscrições":-1}},{$limit: 3}])
{ "_id" : "Iniciação", "Número de Inscrições" : 15 }
{ "_id" : "Salto", "Número de Inscrições" : 6 }
{ "_id" : "Corrida", "Número de Inscrições" : 4 }
```

4.8. Query nº8

Com esta query pretendemos saber quantos alunos e instrutores pertencem a uma certa Localidade (neste caso “Vila do Conde”).

É importante notar que a query em SQL original (apresentada no relatório do primeiro trabalho) não estava correta. Contudo apresentamos agora uma correção onde, face ao povoamento da base de dados, é apresentado o resultado pretendido, o que vai de acordo com o obtido utilizando o Mongo.

```
1  USE CentroHipico;
2
3  /* Quantos alunos e instrutores registados moram em Vila do Conde */
4
5  SELECT count(*) FROM (
6  select I.Nome as Pessoa from Instrutor as I
7  where I.Localidade = 3
8  UNION ALL
9  select A.Nome as Pessoa from Aluno as A
10 where A.Localidade = 3) As Result;
11
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

#	count(*)
1	6

```
> db.Aluno.find({"Localidade":"Vila do Conde"},{}).count() + db.Instrutor.find({"Localidade":"Vila do Conde"}, {}).count()
6
```

5. Conclusão

O nosso grupo assumiu como objetivo transformar uma base de dados relacional, numa base de dados NoSQL orientada a documentos (MongoDB), com o objetivo de esta passar a guardar informação que na base de dados original se perderia. Para tal, reunimos várias vezes de modo a analisar o problema, delinear os passos a seguir, e implementar o sistema.

Conseguimos com sucesso fazer a migração dos dados, organizar as coleções da forma que nos pareceu mais apropriada, bem como resolver o problema que o Sr Quintas tinha e que fundamenta este projeto.

Durante o processo encontramos uma falha na base de dados original. O erro diz respeito à tabela Inscrição cuja chave primária era apenas composta por Aluno e Modalidade, mas que deveria incluir também data de início já que um aluno pode se inscrever na mesma modalidade mais do que uma vez. No entanto, este erro não se aplica a MongoDB já que neste o conceito de PK não existe.

Poderíamos num futuro próximo expandir o nosso sistema para que este registasse todas aulas do Centro Hípico, guardando um registo dos alunos e cavalos que nelas participam garantindo que o uso dos cavalos é bem gerido visto que estes animais são sensíveis e com a idade necessitam de repousar entre as aulas. Para isto, criaríamos uma coleção Aula, na qual registaríamos a data (incluindo hora), a lista de alunos que a frequentou e a modalidade (incluindo o instrutor que a lecionou a lista dos cavalos que participaram).

Por fim, este projeto permitiu-nos adquirir conhecimentos importantes no manuseio de um sistema não relacional de bases de dados nomeadamente MongoDB, e na migração de dados entre este BD relacionais e este sistema. Chegamos acima de tudo à conclusão de que, um esquema que pode ser bom em SQL poderá não ser o mais adequado noutro SBD, e, portanto, ao realizar este tipo de procedimentos deve-se pensar não apenas na migração de dados, mas na forma mais adequada de a realizar.

Referências

1. “Importance of Business Record Keeping.” ESalesTrack, www.esalestrack.com.
2. “DB-Engines Ranking.” DB-Engines, db-engines.com.
3. “MongoDB and MySQL Compared.” MongoDB, www.mongodb.com.
4. “Getting Your Data out of SQL and into MongoDB with Ease.” Mongify, mongify.com/.
5. “Databases, Documents and Collections in MongoDB.” w3resource, www.w3resource.com.

Lista de Siglas e Acrónimos

SBD Sistema de Base de Dados

SBDR Sistema de Base de Dados Relacional

NoSQL *Not only SQL*

Anexos

- **Anexo 1 - Script java para a migração de dados.**

```java
package migracao;

import com.mongodb.MongoClient;
import com.mongodb.MongoClientOptions;
import com.mongodb.client.MongoDatabase;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.PreparedStatement;
import java.sql.ResultSet;


public class CentroHipico
{
    public static void main(String[] args) throws ClassNotFoundException
    {
        MongoClient mongo = new MongoClient("localhost",27017);
        MongoDatabase db = mongo.getDatabase("CentroHipico");

        Connection con = null;

        Migracao m = new Migracao(con, db);

                m.drop();
        m.migrateAluno();
                m.migrateInstrutor();
                m.migrateCavalo();
                m.migrateModalidade();
                m.migrateInscricao();
    }
}
```

```java
package migracao;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class Connect {

    private static final String URL = "localhost";
    private static final String DB = "CentroHipico";
    private static final String USERNAME = "root";
    private static final String PASSWORD = "******";

    public static Connection connect() throws SQLException, ClassNotFoundException {
        Class.forName("com.mysql.jdbc.Driver");

        return DriverManager.getConnection("jdbc:mysql://"+URL+"/"+DB+"?user="+USERNAME+"&password="+PASSWORD+"&autoReconnect=true&useSSL=false");
    }

    public static void close(Connection c) {
        try {
            if(c!=null && !c.isClosed()) {
                c.close();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```java
package migracao;

import com.mongodb.BasicDBList;
import com.mongodb.BasicDBObject;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class Migracao {

    private Connection con;
    private MongoDatabase db;

    public Migracao(Connection con, MongoDatabase db) {
        this.db = db;
        this.con = con;
    }
```

```java
    public void migrateAluno() throws ClassNotFoundException {
        String nome = null, morada = null, email = null, localidade = null;
        int telemovel = 0;
        int nr = 0;
        String data = null;

        try {

            this.con = (Connection) Connect.connect();
            String sql = ""+ "Select A.Nome, A.Morada, A.DataNascimento, A.Email, A.Telefone, L.Designacao FROM Aluno AS A \n"
                    + "INNER JOIN Localidade AS L\n"
                    + "ON A.Localidade = L.Id\n";
            PreparedStatement ps = this.con.prepareStatement(sql);
            ResultSet rs = ps.executeQuery();
            MongoCollection<BasicDBObject> collection = this.db.getCollection("Aluno", BasicDBObject.class);

            while (rs.next()) {

                nome = rs.getString("A.Nome");
                morada = rs.getString("A.Morada");
                data = rs.getString("A.DataNascimento");
                email = rs.getString("A.Email");
                telemovel = rs.getInt("A.Telefone");
                localidade = rs.getString("L.Designacao");

                BasicDBObject document = new BasicDBObject();

                document.put("Nome", nome);
                document.put("Morada", morada);
                document.put("DataNascimento", data);
                document.put("Email", email);
                document.put("Telefone", telemovel);
                document.put("Localidade", localidade);
                collection.insertOne(document);
            }
        } catch (SQLException e) {
            System.out.printf(e.getMessage());
        } finally {
            try {
                Connect.close((java.sql.Connection) con);
            } catch (Exception e) {
                System.out.printf(e.getMessage());
            }
        }
    }
```

```java
    public void migrateCavalo() throws ClassNotFoundException{
        String nome = null, raça = null, coudelaria = null;
        int nrbox = 0;
        String data = null;

        try {

            this.con = (Connection) Connect.connect();
            String sql = ""+ "Select C.Nome, C.Raca, C.DataNascimento, C.Coudelaria, C.NrBox FROM Cavalo AS C \n";
            PreparedStatement ps = this.con.prepareStatement(sql);
            ResultSet rs = ps.executeQuery();
            MongoCollection<BasicDBObject> collection = this.db.getCollection("Cavalo", BasicDBObject.class);

            while (rs.next()) {

                nome = rs.getString("C.Nome");
                raça = rs.getString("C.Raca");
                data = rs.getString("C.DataNascimento");
                coudelaria = rs.getString("C.Coudelaria");
                nrbox = rs.getInt("C.NrBox");


                BasicDBObject document = new BasicDBObject();

                document.put("Nome", nome);
                document.put("Raca", raça);
                document.put("DataNascimento", data);
                document.put("Coudelaria", coudelaria);
                document.put("NrBox", nrbox);

                collection.insertOne(document);
            }
        } catch (SQLException e) {
            System.out.printf(e.getMessage());
        } finally {
            try {
                Connect.close((java.sql.Connection) con);
            } catch (Exception e) {
                System.out.printf(e.getMessage());
            }
        }
    }
```

```java
public void migrateInstrutor() throws ClassNotFoundException{
    String nome = null, morada = null, email = null, localidade = null;
    int telemovel = 0;
    String data = null;

    try {

        this.con = (Connection) Connect.connect();
        String sql = ""+ "Select I.Nome, I.Morada, I.DataNascimento, I.Email, I.Telefone, L.Designacao from Instrutor AS I \n"
                + "INNER JOIN Localidade AS L\n"
                + "ON I.Localidade = L.Id\n";
        PreparedStatement ps = this.con.prepareStatement(sql);
        ResultSet rs = ps.executeQuery();
        MongoCollection<BasicDBObject> collection = this.db.getCollection("Instrutor", BasicDBObject.class);

        while (rs.next()) {

            nome = rs.getString("I.Nome");
            morada = rs.getString("I.Morada");
            data = rs.getString("I.DataNascimento");
            email = rs.getString("I.Email");
            telemovel = rs.getInt("I.Telefone");
            localidade = rs.getString("L.Designacao");

            BasicDBObject document = new BasicDBObject();

            document.put("Nome", nome);
            document.put("Morada", morada);
            document.put("DataNascimento", data);
            document.put("Email", email);
            document.put("Telefone", telemovel);
            document.put("Localidade", localidade);
            collection.insertOne(document);
        }
    } catch (SQLException e) {
        System.out.printf(e.getMessage());
    } finally {
        try {
            Connect.close((java.sql.Connection) con);
        } catch (Exception e) {
            System.out.printf(e.getMessage());
        }
    }
}
```

```java
public void migrateInscricao() throws ClassNotFoundException
{
    String dataInicio = null, dataFim = null;
    int nrAulas = 0, preco = 0;
    int idAluno = 0;

    String nome = null, morada = null, email = null, localidade = null;
    int telemovel = 0;
    String data = null;

    String modalidade = null;
    int nivel = 0;
    String descricao = null;

    String datanascimento = null, nomeInst = null, emailInst = null, moradaInst = null, localidadeInst = null;
    int telefone;

    String nomeCav = null, raça = null, coudelaria = null, dataCav = null;
    int nrbox = 0;
```

```java
    try{
            con = Connect.connect();
            PreparedStatement ps = con.prepareStatement("" + "SELECT * FROM Inscricao\n");
            ResultSet rs = ps.executeQuery();
            MongoCollection<BasicDBObject> collection = this.db.getCollection("Inscricao",BasicDBObject.class);

            while(rs.next())
            {

                dataInicio = rs.getString("DataInicio");
                dataFim = rs.getString("DataFim");
                nrAulas = rs.getInt("NrDeAulas");
                preco = rs.getInt("Preco");

                idAluno = rs.getInt("Aluno");
                BasicDBObject doc = new BasicDBObject();

                doc.put("DataInicio",dataInicio);
                doc.put("DataFim",dataFim);
                doc.put("NrAulas",nrAulas);
                doc.put("Preco", preco);


                PreparedStatement ps2 = con.prepareStatement(""
                   + "select A.Nome, A.DataNascimento, A.Email, A.Morada, A.Telefone, L.Designacao, I.Modalidade from Aluno As A \n"
                   + "inner join Inscricao as I \n"
                   + "inner join Localidade as L\n"
                   + "where I.Aluno = A.Nr and A.Localidade = L.Id and I.Aluno = ?");


                   ps2.setInt(1, idAluno);
                   ResultSet rs2 = ps2.executeQuery();
```

```
                while(rs2.next())
                {
                    nome = rs2.getString("A.Nome");
                    morada = rs2.getString("A.Morada");
                    data = rs2.getString("A.DataNascimento");
                    email = rs2.getString("A.Email");
                    telemovel = rs2.getInt("A.Telefone");
                    localidade = rs2.getString("L.Designacao");
                    modalidade = rs2.getString("I.Modalidade");

                    BasicDBObject docAluno = new BasicDBObject();

                    docAluno.put("Nome", nome);
                    docAluno.put("Morada", morada);
                    docAluno.put("DataNascimento", data);
                    docAluno.put("Email", email);
                    docAluno.put("Telefone", telemovel);
                    docAluno.put("Localidade", localidade);

                    doc.put("Aluno",docAluno);
                }

                    PreparedStatement ps3 = con.prepareStatement(""
                    + "Select M.Nome, M.NvlDificuldade, M.Descricao from Modalidade As M\n"
                    +"where M.Nome = ?");
                    ps3.setString(1,modalidade);
                    ResultSet rs3 = ps3.executeQuery();
                    BasicDBObject mod = new BasicDBObject();
```

```
                while(rs3.next())
                {
                    nivel = rs3.getInt("M.NvlDificuldade");
                    descricao = rs3.getString("M.Descricao");

                    mod.put("Nome", modalidade);
                    mod.put("Nivel", nivel);
                    mod.put("Descricao", descricao);

                    doc.put("Modalidade",mod);

                }


            PreparedStatement ps4 = con.prepareStatement(""
            + "SELECT I.Nome, I.Morada, I.DataNascimento, I.Email, I.Telefone, L.Designacao FROM Instrutor AS I \n"
            + "INNER JOIN Localidade AS L \n"
            + "ON I.Localidade = L.Id \n"
            + "WHERE Modalidade = ?");


            ps4.setString(1, modalidade);
            ResultSet rs4 = ps4.executeQuery();
            BasicDBList lista = new BasicDBList();
```

```
378                 while(rs4.next()){
379                     nomeInst = rs4.getString("I.Nome");
380                     datanascimento = rs4.getString("I.DataNascimento");
381                     emailInst = rs4.getString("I.Email");
382                     telefone = rs4.getInt("I.Telefone");
383                     moradaInst = rs4.getString("I.Morada");
384                     localidadeInst = rs4.getString("L.Designacao");
385 
386                     BasicDBObject docInstrutor = new BasicDBObject ();
387 
388                     docInstrutor.put("Nome",nomeInst);
389                     docInstrutor.put("DataNascimento",datanascimento);
390                     docInstrutor.put("Email", emailInst);
391                     docInstrutor.put("Telefone", telefone);
392                     docInstrutor.put("Morada", moradaInst);
393                     docInstrutor.put("Localidade",localidadeInst);
394 
395                     lista.add(docInstrutor);
396 
397 
398                 }
399 
400                 doc.put("Instrutores",lista);
401 
402                 PreparedStatement ps5 = con.prepareStatement(""
403                 + "SELECT * FROM Cavalo AS C WHERE Modalidade = ?");
404 
405                 ps5.setString(1, modalidade);
406                 ResultSet rs5 = ps5.executeQuery();
407                 BasicDBList listaCavalos = new BasicDBList();
408 
```

```
409                     while(rs5.next())
410                     {
411                         nomeCav = rs5.getString("C.Nome");
412                         raça = rs5.getString("C.Raca");
413                         dataCav = rs5.getString("C.DataNascimento");
414                         coudelaria = rs5.getString("C.Coudelaria");
415                         nrbox = rs5.getInt("C.NrBox");
416 
417                         BasicDBObject docCavalos = new BasicDBObject();
418 
419                         docCavalos.put("Nome", nomeCav);
420                         docCavalos.put("Raca", raça);
421                         docCavalos.put("DataNascimento",dataCav);
422                         docCavalos.put("Coudelaria", coudelaria);
423                         docCavalos.put("NrBox", nrbox);
424                         listaCavalos.add(docCavalos);
425                     }
426 
427                     doc.put("Cavalos",listaCavalos);
428                     collection.insertOne(doc);
429 
430                 }
431 
432             }
433             catch(SQLException e)
434             {
435                 System.out.printf(e.getMessage());
436             }
437             finally
438             {
439                 try
440                 {
441                     Connect.close(con);
442                 }
443                 catch(Exception e)
444                 {
445                     System.out.printf(e.getMessage());
446                 }
447             }
448         }
```

```
450     public void drop ()
451     {
452         MongoCollection<BasicDBObject> collection = this.db.getCollection("Instrutor",BasicDBObject.class);
453         collection.drop();
454         collection = this.db.getCollection("Aluno", BasicDBObject.class);
455         collection.drop();
456         collection = this.db.getCollection("Cavalo", BasicDBObject.class);
457         collection.drop();
458         collection = this.db.getCollection("Modalidade", BasicDBObject.class);
459         collection.drop();
460         collection = this.db.getCollection("Inscricao", BasicDBObject.class);
461         collection.drop();
462     }
463 
464 }
```

Sublime Text

```java
public void migrateModalidade() throws ClassNotFoundException{

    String nomeMod = null, descriçao = null;
    int nvldificuldade;

    String datanascimento = null, nomeInst = null, email = null, morada = null, localidade = null;
    int telefone;

    String nomeCav = null, raça = null, coudelaria = null, dataCav = null;
    int nrbox = 0;


    try{
        con = Connect.connect();
        PreparedStatement ps = con.prepareStatement("" + "SELECT * FROM Modalidade\n");
        ResultSet rs = ps.executeQuery();
        MongoCollection<BasicDBObject> collection = this.db.getCollection("Modalidade",BasicDBObject.class);

        while(rs.next()){

            nomeMod = rs.getString("Nome");
            descriçao = rs.getString("Descricao");
            nvldificuldade = rs.getInt("NvlDificuldade");
            BasicDBObject doc = new BasicDBObject();

            doc.put("Nome",nomeMod);
            doc.put("Descricao",descriçao);
            doc.put("NvlDificuldade",nvldificuldade);

            PreparedStatement ps2 = con.prepareStatement(""
            + "SELECT I.Nome, I.Morada, I.DataNascimento, I.Email, I.Telefone, L.Designacao FROM Instrutor AS I \n"
            + "INNER JOIN Localidade AS L \n"
            + "ON I.Localidade = L.Id \n"
            + "WHERE Modalidade = ?");


            ps2.setString(1, nomeMod);
            ResultSet rs2 = ps2.executeQuery();
            BasicDBList lista = new BasicDBList();
```

```java
            while(rs2.next()){
                nomeInst = rs2.getString("I.Nome");
                datanascimento = rs2.getString("I.DataNascimento");
                email = rs2.getString("I.Email");
                telefone = rs2.getInt("I.Telefone");
                morada = rs2.getString("I.Morada");
                localidade = rs2.getString("L.Designacao");

                BasicDBObject docInstrutor = new BasicDBObject ();

                docInstrutor.put("Nome",nomeInst);
                docInstrutor.put("DataNascimento",datanascimento);
                docInstrutor.put("Email", email);
                docInstrutor.put("Telefone", telefone);
                docInstrutor.put("Morada", morada);
                docInstrutor.put("Localidade",localidade);

                lista.add(docInstrutor);


            }

            doc.put("Instrutores",lista);

            PreparedStatement ps3 = con.prepareStatement(""
            + "SELECT * FROM Cavalo AS C WHERE Modalidade = ?");

            ps3.setString(1, nomeMod);
            ResultSet rs3 = ps3.executeQuery();
            BasicDBList listaCavalos = new BasicDBList();
```

```
232                 while(rs3.next()){
233                     nomeCav = rs3.getString("C.Nome");
234                     raça = rs3.getString("C.Raca");
235                     dataCav = rs3.getString("C.DataNascimento");
236                     coudelaria = rs3.getString("C.Coudelaria");
237                     nrbox = rs3.getInt("C.NrBox");
LibreOffice Writer
240
241                     BasicDBObject docCavalos = new BasicDBObject();
242
243                     docCavalos.put("Nome", nomeCav);
244                     docCavalos.put("Raca", raça);
245                     docCavalos.put("DataNascimento",dataCav);
246                     docCavalos.put("Coudelaria", coudelaria);
247                     docCavalos.put("NrBox", nrbox);
248                     listaCavalos.add(docCavalos);
249                 }
250
251                 doc.put("Cavalos",listaCavalos);
252                 collection.insertOne(doc);
253
254             }
255         }
256         catch(SQLException e)
257         {
258             System.out.printf(e.getMessage());
259         }
260         finally
261         {
262             try{
263                 Connect.close(con);
264             }
265             catch(Exception e)
266             {
267                 System.out.printf(e.getMessage());
268             }
269         }

    }
```