NoSQL Databases

PL03 – Introduction to the Document Model

Teacher: Cristiana Neto
Email: cristiana.neto@algoritmi.uminho.pt

Office hours:
Friday 10h-11h

Summary

1 MongoDB - Architecture | Database | Collections | Physical model

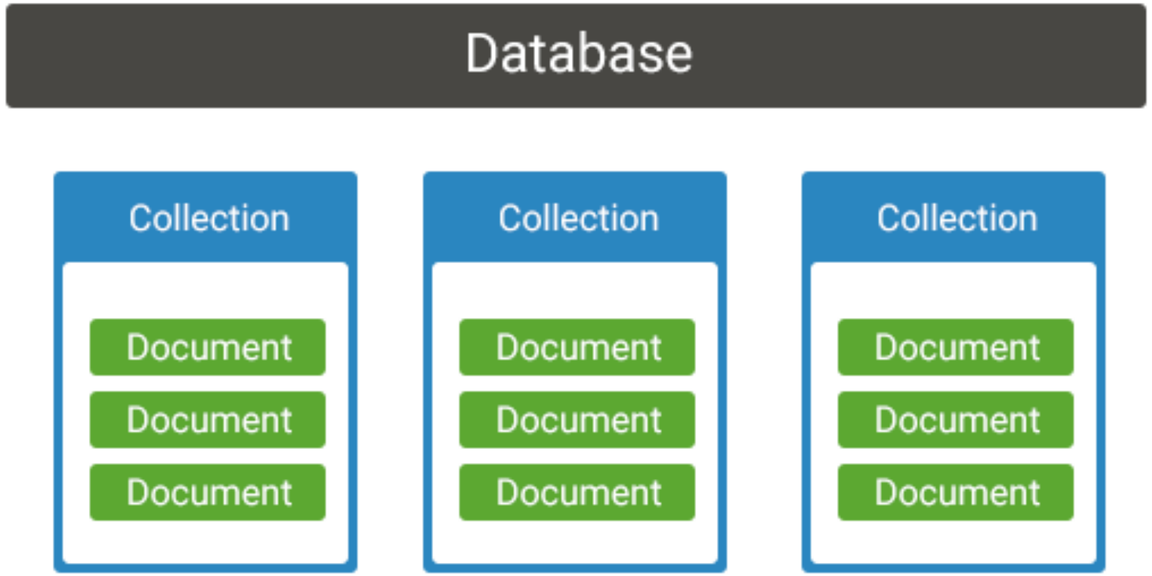
2 MongoShell e MongoDB - Tutorial

3 FEO3 – Worksheet 3

mongoDB

Hu**mongo**us DB

- Open-source
- Document-oriented
- JSON alike
- Embedded documents



MongoDB

➡ Main features

➜ **High Performance:** Indexing is possible in any field of the documents;

➜ **High Availability and Replication:** The process of replication, in a simplified way, is the ability to distribute data across multiple servers and keep everything synchronized. Existence of primary node and secondary nodes;

➜ **Absence of Schema:** There may be different documents within a database, not all with the same structure;

➜ **Scalability:** Possibility to add several nodes to contain a huge number of documents.

MongoDB

Architecture

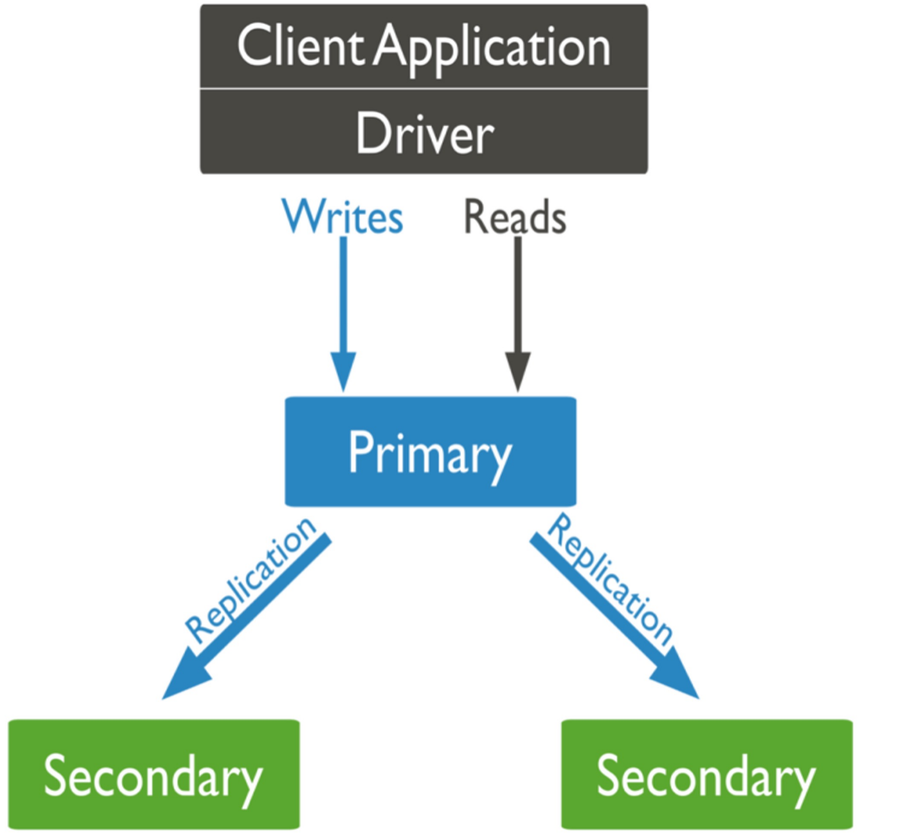
Replication

- Fault tolerance and redundancy;
- High availability;
- Invisible to the client application;

Existence of 1 primary node and 1..n secondary nodes:

Primary node: read and write operations;

Secondary nodes: asynchronous replication;



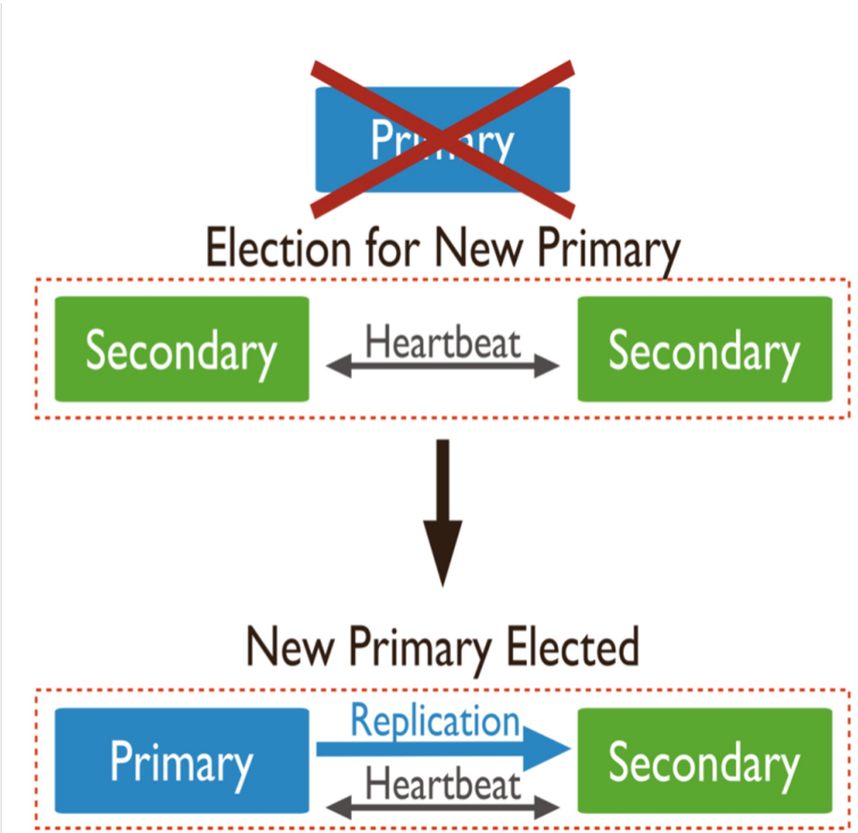
MongoDB

Architecture

Replication

In case of failure:

- If the primary fails to communicate for a certain period of time;
- Election of the new primary;
- Guarantee of the existence of a heartbeat;
- During the election of a new primary node, no transactions can take place.

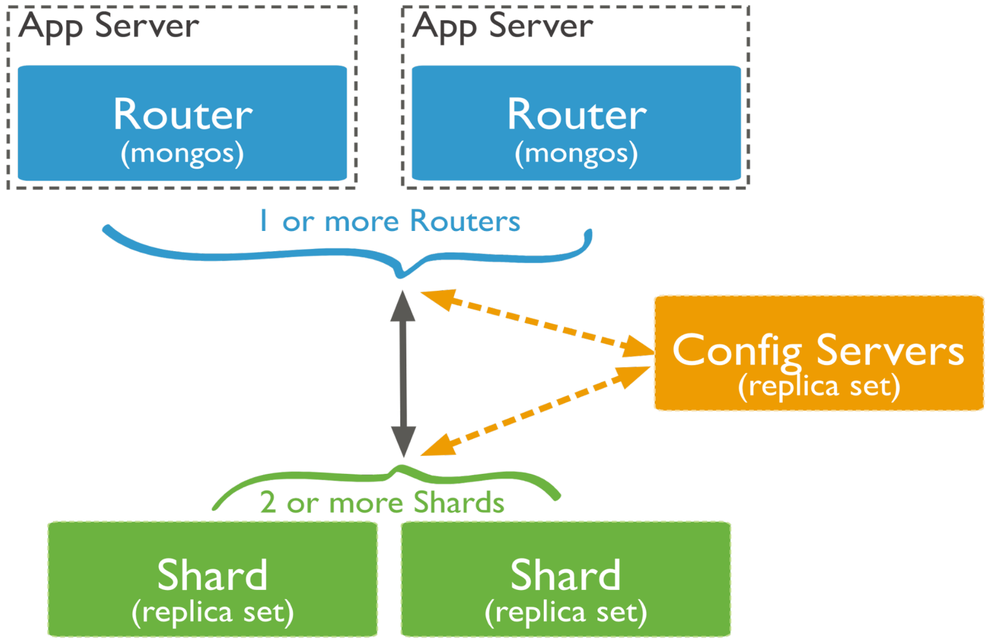


MongoDB

Architecture

Sharding

Method for distributing data across multiple servers. MongoDB uses this technique to allow the storage of huge datasets and ensure that complex operations are performed on them.



MongoDB

Architecture

Sharding

- shard: Each shard contains a subset of the totality of the data. They can be defined as replica sets.
- mongos: Mongos act as a router for the queries to be executed. They provide an interface between client applications and the cluster.
- config servers: Each configuration server contains metadata and server settings.

MongoDB

Architecture

Model

- Each document has a maximum of 16mb;
- The format of the documents is BSON;
- Each document is stored in a collection;
- Collections
  - Common indexes;
  - They are similar to tables in relational databases;
  - Documents contained in a collection may not have a uniform structure.

MongoDB

Architecture

JSON – "JavaScript Object Notation"

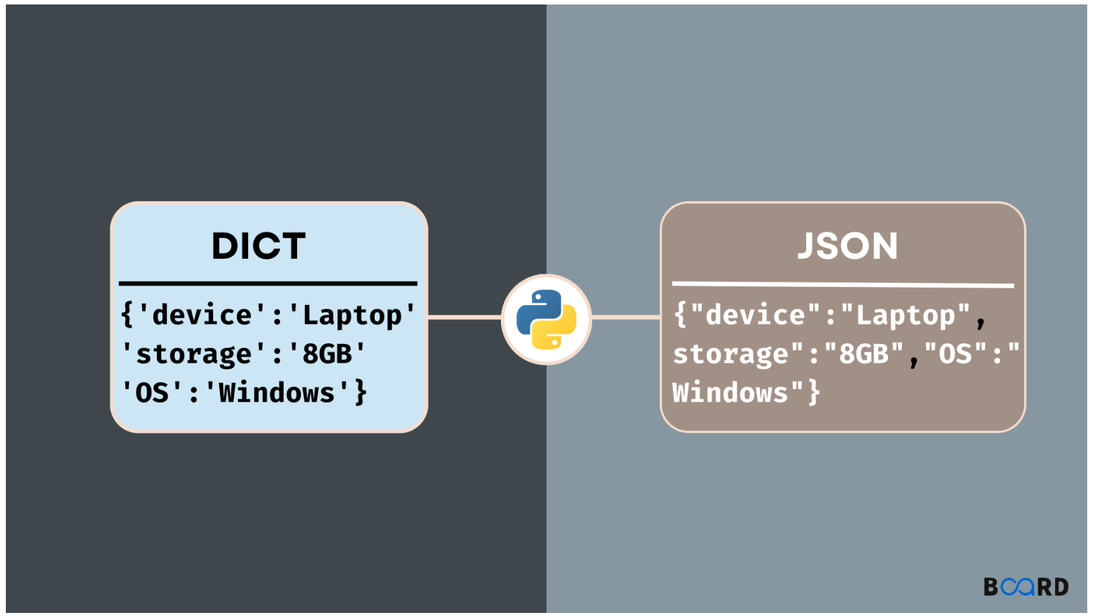
- Easy for humans to read and write;
- Easy for machines to process and manage;
- Structure:
- Name/value set;
- Ordered list of values (array);

BSON – "Binary JSON"

- Binary serialization of JSON documents;
- The main goal is efficiency;

MongoDB

Architecture



Python	JSON Equivalent
True	true
False	false
float	Number
int	Number
None	null
dict	Object
list	Array
tuple	Array

MongoDB

Architecture

Exemplo JSON:

```
{
  "_id" :     "37010"
  "city" :    "ADAMS",
  "pop" :     2660,
  "state" :   "TN",
  "councilman" : {
                  name: "John Smith"
                  address: "13 Scenic Way"
                }
}
```

MongoDB

Architecture

Campo _id:

- By default, all documents contain a field _id:
- Serves as the primary key for the document within the collection;
- It is unique and cannot be changed or transformed;
- The default data type is ObjetId:
  - Small
  - Unique
  - Easy to generate,
  - Easy to sort (similar to sorting by creation date)

MongoDB

Architecture

Tipo	Descrição
Double	used to store floating point values.
String	BSON strings are UTF-8. It is most commonly used datatype.
Object	used for embedded documents.
Array	used to store multiples of values and data types.
Binary data	used to store the binary data.
Undefined	used to store the undefined values.
Object id	used to store the document's id.
Boolean	used to store a boolean value.
Date	used to store the current date or time in UNIX time format.
Null	used to store a Null value.
Regular Expression	used to store regular expression.
JavaScript	used to store the javascript data without a scope.
Symbol	It is similar to the string data type, and not supported by a shell.
JavaScript (with scope)	It stores javascript data with a scope.
Timestamp	used to store a timestamp.
Min key	Min key compares the value of the lowest BSON element.
Max key	Max key compares the value against the highest BSON element.

MongoDB

Architecture

RDBMS		MongoDB
Database	→	Database
Table	→	Collection
Row	→	Document
Index	→	Index
Join	→	Embedded Document
Foreign Key	→	Reference

Mongosh

MongoSH allows users to perform various MongoDB database operations, such as queries, updates, and management of databases, collections, and documents using the command line.

Executar shell do container mongodb:

```
# docker exec -it <container-id> bash
# mongosh
```

```
root@mongo:/# mongosh
Current Mongosh Log ID: 65ea0f84d9d4a38b49776107
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.1.4
Using MongoDB:          7.0.5
Using Mongosh:          2.1.4

For mongosh info see: https://docs.mongodb.com/mongodb-shell/


To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

------
   The server generated these startup warnings when booting
   2024-03-07T19:02:26.289+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
   2024-03-07T19:02:27.352+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
   2024-03-07T19:02:27.352+00:00: vm.max_map_count is too low
------

test>
```

MongoDB – Tutorial

- Create Database:

```
> use nosql
```

If the database name used already exists, the existing database will be used!

- Check current DB:

```
> db
```

- List all databases:

```
> show dbs
```

The created database does not appear, as we have to save the first document!

- Create Collection and Store First Document:

```
> db.pl.insertOne({
   nome: "Cristiana Neto",
   age: "28"
})
```

```
// Verify DB creation:
> show dbs
```

```
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
nosql   0.000GB
```

MongoDB – Tutorial

- Delete created database:

```
// View existing DBs
> show dbs

// Enter the desired database
> use nosql

//Delete the Database
> db.dropDatabase()

// Verify that the db has been deleted
> show dbs
```

MongoDB – Tutorial

- Create Collection:

1. Create and store first document:

```
// Select the database
> use music

// Insert document into the desired collection (songs)
> db.songs.insertOne({
    name: "Alive",
    band: "Pearl Jam",
    year: 1990
})

// List all documents in the collection
> db.songs.find()
```

2. Criar sem gerar primeiro documento:

```
> use music

> db.createCollection("albums")
```

MongoDB – Tutorial

- Delete Collection:

```
// View existing DBs
> show dbs

// Enter the desired database
> use music

// View existing collections
> show collections

// Delete the collection you want
> db.albums.drop()

// View existing collections
> show collections
```

MongoDB – Tutorial

- Insert document:

Sintaxe:

```
> db.collection_name.insertOne()
```

```
// Create From Document
> db.songs.insertOne({
   name: "Jeremy",
   band: "Pearl Jam",
   year: 1990,
   crew: [{vocals: "Eddie Vedder",drums: "Dave Abbruzzese"}]
})
```

```
// Verify Document Creation
db.collection_name.find()
```

MongoDB – Tutorial

- Insert multiple documents:

```
var musicas= [{
            "name" : "Riders on the Storm",
    "band" : "The Doors",
    "year": 1971
},
{
            "name" : "Roadhouse Blues",
    "band" : "The Doors",
    "year": 1969
}
];
```

```
//Insert Documents.                    // Validate Creation                    // Validate JSON
db.songs.insert(musicas);    db.songs.find()      db.songs.find().pretty()
```

MongoDB - Tutorial

- Search documents:

```
db.songs.find({name:"Alive"}).pretty()
```

- Greater than:

```
db.songs.find({year:{$gt:1980}}).pretty()
```

- Greater than or equal to:

```
db.songs.find({year:{$gte:1980}}).pretty()
```

-Less than

:

```
db.songs.find({year:{$lt:1980}}).pretty()
```

- Less than or equal to

```
db.songs.find({year:{$lte:1980}}).pretty()
```

- Not equal to:

```
db.songs.find({band:{$ne:"Pearl Jam"}}).pretty()
```

MongoDB – Tutorial

- Update Document:

```
db.songs.updateOne({name: "Alive"}, {$set: {year: 1995}})
```

```
db.songs.updateOne({band: "Pearl Jam"}, {$set: {band: "PJ"}})
```

- Update multiple documents:

```
db.songs.updateMany({band: "Pearl Jam"}, {$set: {band: "PJ"}})
```

- Create in case it doesn't exist:

```
db.songs.update({band: "Nirvana"}, {$set: {foundation: 1990}},
{upsert:true})
```

MongoDB – Tutorial

– Delete document:

```
db.songs.removeMany({band: "Nirvana"})
```

Unlike the update, the remove turns off all entries!!

– Delete only 1 document (first you find it):

```
db.songs.removeOne({band: "Nirvana"})
```

MongoDB – Tutorial

Projections:

- List only the band name, excluding the _id:

```
db.songs.find({},{_id: 0, band: 1}).pretty();
```

- List the band's name (id appears):

```
db.songs.find({},{band: 1, year:1}).pretty();
```

Limit:

- List only the first 2 results:

```
db.songs.find({}).limit(2).pretty();
```

- Skip the first 2 results and list the next

```
db.songs.find({}).skip(2).limit(2).pretty();
```

MongoDB – Tutorial

Order:

- List songs by descending release date:

```
db.songs.find({}).sort({year: -1}).pretty();
```

- List songs by upstream release date:

```
db.songs.find({}).sort({year: 1}).pretty();
```

FE03 – Introduction to MongoDB

Universidade do Minho
Departamento de Informática

FE03

Curso: Mestrado Integrado em Informática
U.C.: Bases de Dados NoSQL

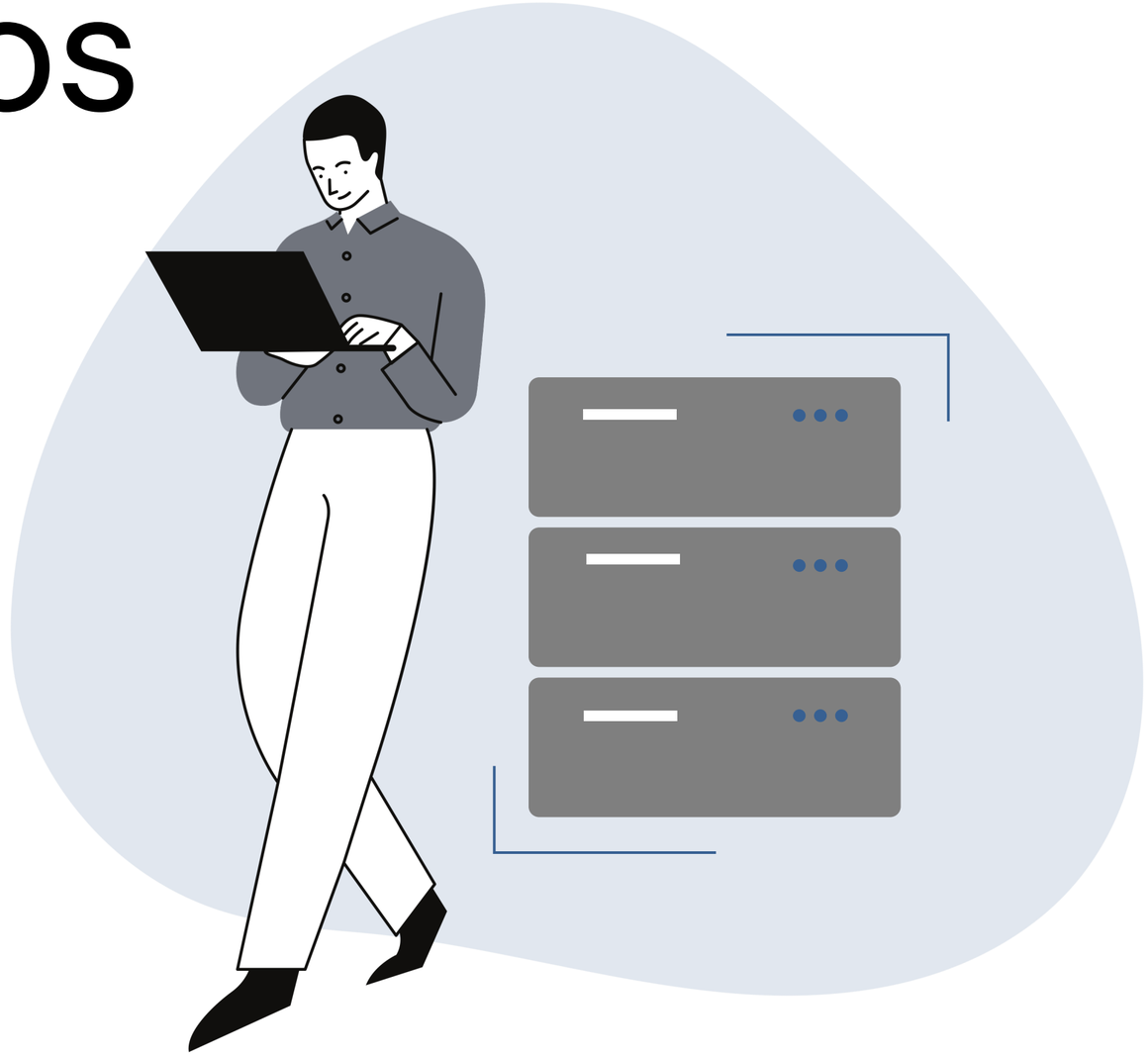
Ficha de Exercícios 03 - PL04	
Docente:	António Abelha / Cristiana Neto
Tema:	Introdução ao MongoDB
Turma:	Prática Laboratorial
Ano Letivo:	2023-2024 – 2º Semestre
Duração da aula:	2 horas

Universidade do Minho

Ano Letivo: 2023/24

Bases de Dados NoSQL

PL03 – Introdução ao Modelo Documental



Docente: Cristiana Neto
Email: cristiana.neto@algoritmi.uminho.pt

Horário de Atendimento:
6ª feira 10h-11h