

Aula 4 – Conexão com Banco de dados

PROFESSOR: HARLEY MACÊDO DE MELLO

Roteiro

- Conceitos
- MongoDB
- Relacionamentos
- Mongoose

Conceitos

- Maior demanda por armazenamento de dados
- Necessidade de escalonamento rápido
- Maior velocidade nas consultas
- Permissão para armazenar dados sem estrutura fixa
- Dados mais próximos do formato JSON

Conceitos

- Banco de dados não relacional
- Sem tabelas e relacionamentos
- Registros independentes
- Estrutura livre, ou 'schema free'

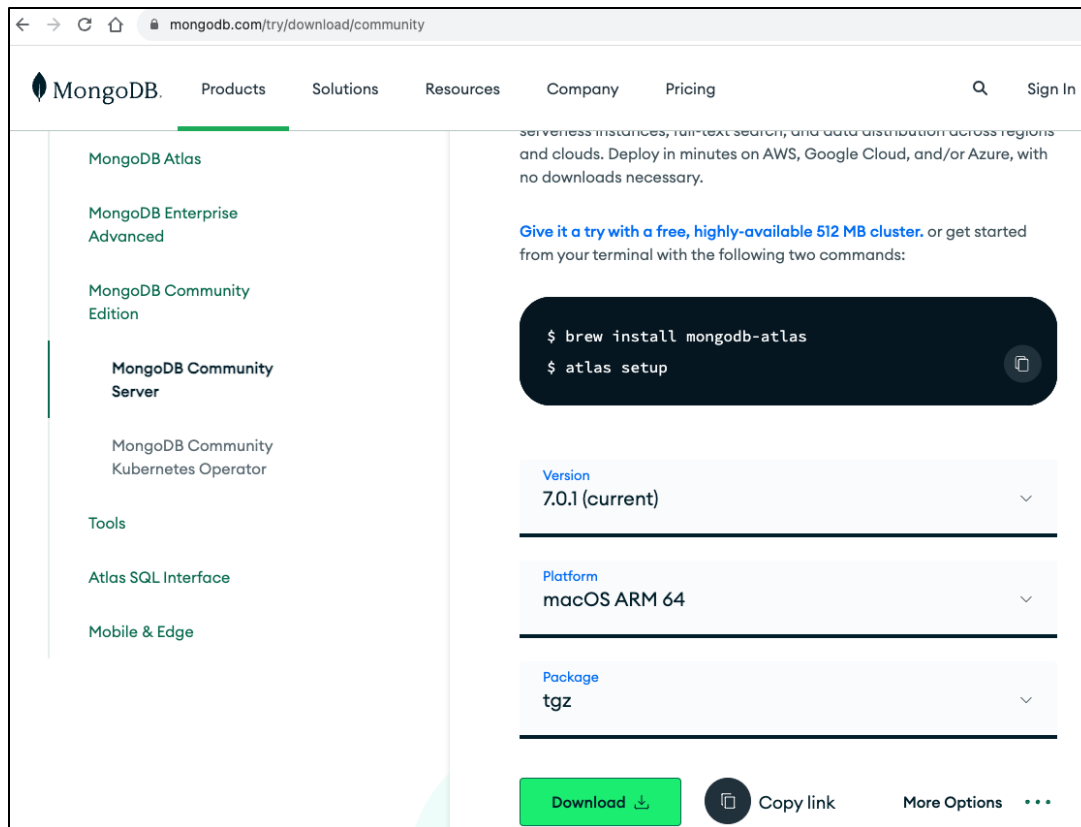
MongoDB

- Banco NoSQL e open source
- Muito utilizado atualmente
- Segundo pesquisa do StackOverflow, usado por mais de 25% do programadores
- Orientado à documentos
- Usa dados em formato JSON
- Software como serviço(SaaS) disponível
- Escalabilidade, flexibilidade, disponibilidade, desempenho

MongoDB

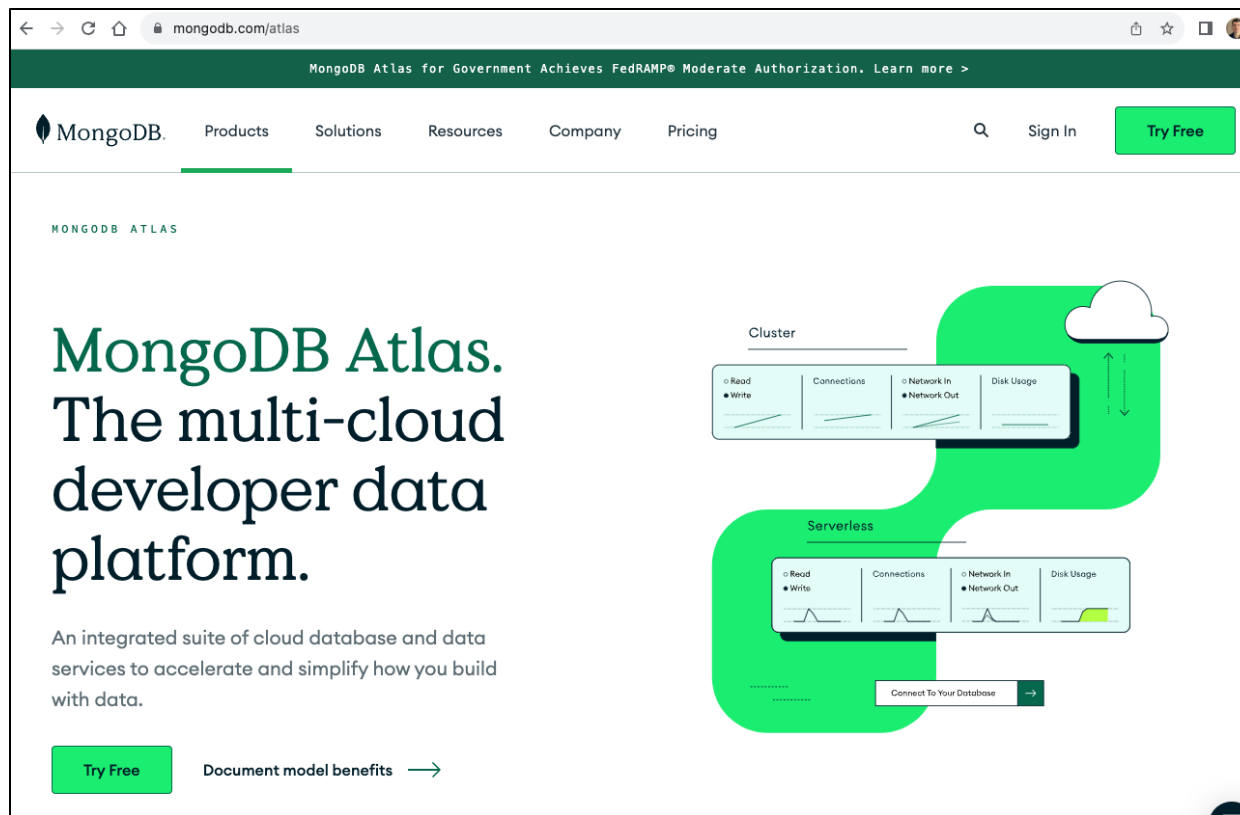
- Download no site mongodb.com/try/download/community
- Usar como serviço em mongodb.com/atlas
- Serviço atlas disponibiliza 500 MB no plano grátis
- Painel que auxilia a construção do DB
- Painel para ajudar na configuração e monitoramento do DB

MongoDB



Tela de download do MongoDB para uso local.

MongoDB



Tela para logar e acessar o serviço atlas MongoDB, para ter um DB em nuvem.

MongoDB

- Organização
 - É a empresa, órgão ou grupo que você está associado
 - Para projetos individuais, é representada pela sua pessoa
- Projeto
 - Uma unidade de trabalho, que pode ter 1 ou mais DBs(Cluster) associados
 - Pode ter 1 time com vários usuários trabalhando no projeto

MongoDB

- Cluster
 - Representa um Database, com seus documentos e relacionamentos
 - Podem ter associados usuários específicos, com permissões definidas
- Documentos
 - Representam os registros
 - Semelhantes às linhas em uma tabela

MongoDB

The screenshot shows the MongoDB Atlas interface for an organization named 'Harley's Org'. The left sidebar contains navigation links: ORGANIZATION, Projects (highlighted), Alerts, Activity Feed, Settings, Integrations, Access Manager, Billing, Support, and Live Migration. The main content area is titled 'Projects' and includes a search bar 'Find a project...'. Below the search bar is a table with columns: Project Name, Database Deployments, Users, Teams, Alerts, and Actions. The table lists two projects: 'Project 0' and 'aula2'. Each project row has a 'More Actions' menu (three dots) and a 'Delete' icon (trash can). A 'New Project' button is located in the top right corner of the main content area.

Project Name	Database Deployments	Users	Teams	Alerts	Actions
Project 0	1 Deployment	1 User	0 Teams	0 Alerts	...
aula2	1 Deployment	1 User	0 Teams	0 Alerts	...

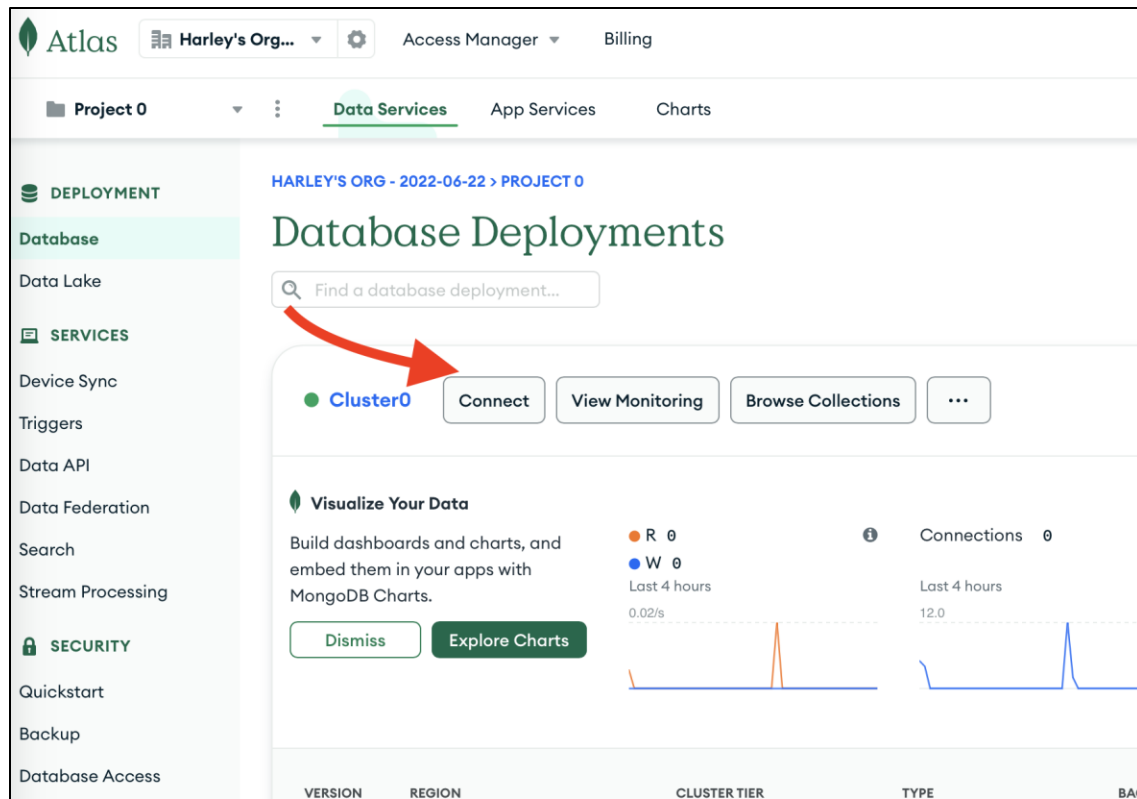
Tela raiz do Atlas MongoDB, na qual podemos visualizar projetos, usuários, times, alertas e etc.

MongoDB

The screenshot displays the MongoDB Atlas web interface. At the top, the 'Atlas' logo is on the left, and navigation links for 'Harley's Org...', 'Access Manager', and 'Billing' are in the center. On the right, there are links for 'All Clusters', 'Get Help', and a user profile 'Harley'. Below this, a secondary navigation bar shows 'Project 0' and tabs for 'Data Services', 'App Services', and 'Charts'. The left sidebar contains a 'DEPLOYMENT' section with 'Database' selected, and a 'SERVICES' section with various options like 'Data Lake', 'Device Sync', etc. The main content area is titled 'HARLEY'S ORG - 2022-06-22 > PROJECT 0 > DATABASES' and shows a 'Cluster0' overview. The 'Collections' tab is active, displaying a list of collections under the 'test' namespace: 'alunos', 'cursos', 'disciplinas', and 'grupos'. The 'alunos' collection is selected, showing its details: 'STORAGE SIZE: 20KB', 'LOGICAL DATA SIZE: 422B', 'TOTAL DOCUMENTS: 3', and 'INDEXES TOTAL SIZE: 20KB'. Below this, there are tabs for 'Find', 'Indexes', 'Schema Anti-Patterns', 'Aggregation', and 'Search Indexes'. The 'Find' tab is active, showing a query filter bar with the text 'Type a query: { field: 'value' }' and buttons for 'Filter', 'Reset', 'Apply', and 'More Options'. Below the query bar, it says 'QUERY RESULTS: 1-3 OF 3' and displays a single document with fields: '_id: ObjectId('643ad64e84f04a54977b2eb0')', 'matricul...: "98374348092"', 'nom...: "Eliza Maria"', and 'emai...: "eliza.maria@aluno.ifce.edu.br"'.

Painel que exhibe as collections, como alunos e cursos.

MongoDB



Para obter a String de conexão, clique no botão 'Connect', em seguida em 'Drive'.

MongoDB

Connect to Cluster0

Set up connection security Choose a connection method **3** Connect

Connecting with MongoDB Driver

1. Select your driver and version

We recommend installing and using the latest driver version.

Driver: Node.js Version: 5.5 or later

2. Install your driver

Run the following on the command line

```
npm install mongodb
```

[View MongoDB Node.js Driver installation instructions.](#)

3. Add your connection string into your application code

☐ View full code sample

```
mongodb+srv://<username>:<password>@cluster0.a05vfop.mongodb.net/?  
retryWrites=true&w=majority
```

Replace **<password>** with the password for the **<username>** user. Ensure any option params are [URL encoded](#).

Obtendo string de
conexão para seu App
acessar o DB.

MongoDB



```
.env
Aula4ConexaoBancoDados > DatabaseComRelacao > .env
1 BD_STR_CON=mongodb+srv://profharley:d[REDACTED]:3@cluster0.a05vfop.mongodb.net/
2
```

Arquivo .env em seu projeto local, com dados para acessar o MongoDB Atlas.

Relacionamentos

- Modelos mais usados
 - Modelo de documento embutido: Documento pai ter uma coleção de documentos filhos
 - Modelo de referência: Documentos filhos têm uma referência ao documento pai

Relacionamentos

```
1  const mongoose = require('mongoose')
2
3  const aluno = mongoose.Schema({
4    matricula: String,
5    nome: String,
6    email: String,
7    cursoNome: String
8  })
9
10 module.exports = mongoose.model('Aluno', aluno)
```

Esquema aluno com o atributo 'cursoNome', que irá relacionar este documento com curso.

Relacionamentos

```
1  const mongoose = require('mongoose')
2
3  const curso = mongoose.Schema({
4    nome: String,
5    tipo: String,
6    duracao: Number
7  })
8
9  module.exports = mongoose.model('Curso', curso)
```

Esquema curso, que não faz referência a aluno.

Relacionamentos

The screenshot shows a REST client interface with a GET request to `localhost:3000/aluno/curso/SI`. The response is a JSON array of two student objects, each associated with the 'SI' course.

Request: GET `localhost:3000/aluno/curso/SI`

Response Status: 200 OK, Size: 310 Bytes, Time: 83 ms

Response Body:

```
{
  "alunos": [
    {
      "_id": "643ad64e84f04a54977b2eb0",
      "matricula": "98374348092",
      "nome": "Eliza Maria",
      "email": "eliza.maria@aluno.ifce.edu.br",
      "cursoNome": "SI",
      "__v": 0
    },
    {
      "_id": "643ad66984f04a54977b2eb2",
      "matricula": "98374348093",
      "nome": "Tulio Machado",
      "email": "tulio.machado@aluno.ifce.edu.br",
      "cursoNome": "SI",
      "__v": 0
    }
  ]
}
```

Request para recuperar dados de relacionamento, trazendo apenas alunos que pertencem ao curso de SI.

Mongoose

- Modelagem de dados de objeto(ODM)
- Biblioteca que abstrai o uso do MongoDB
- Tradutor de objetos do sistema para objetos no DB
- Auxilia a projeção de esquemas para o sistema

Mongoose

- Função `mongoose.Schema({schema})`
 - Define uma entidade do sistema
 - O esquema não obriga que todos os objetos tenham a mesma estrutura
- Função `mongoose.connect(string_con)`
 - Cria uma nova conexão com um cluster
 - Após um período sem uso, a conexão é fechada pelo servidor

Mongoose

- Comandos para manipulação de dados
 - Create
 - Criação de um novo registro(document)
 - `Model.create({atributo: valor})`
 - Find
 - Busca de registros
 - `Model.find({atributo: valor})`
 - Os parâmetros servem como filtro da busca

Mongoose

- Comandos para manipulação de dados
 - DeleteOne
 - Exclui um documento da coleção
 - `Model.DeleteOne({filtro})`
 - FindOneAndUpdate
 - Atualiza um documento da coleção
 - `Model.FindOneandUpdate({filtro}, {novosDados})`

Mongoose

```
5 //Rota GET para obter todos os alunos
6 alunoRouter.get('/aluno/todos', async (req, res) => {
7   try {
8     await mongoose.connect(process.env.BD_STR_CON)
9     const alunosBuscados = await aluno.find()
10    res.json({alunos: alunosBuscados})
11  } catch (error) {
12    res.json({erro: true, mensagem: 'Erro durante consulta'})
13  }
14 })
15
16 //Rota GET para obter 1 alunos, pela matrícula
17 alunoRouter.get('/aluno/matricula/:matricula', async (req, res) => {
18   try {
19     await mongoose.connect(process.env.BD_STR_CON)
20     const alunoBuscado = await aluno.findOne({matricula: req.params.matricula})
21     res.json({aluno: alunoBuscado})
22   } catch (error) {
23     res.json({erro: true, mensagem: 'Erro durante consulta'})
24   }
25 })
```

Funções find e findOne em uso.

Mongoose

```
57 //Rota PUT para alterar um aluno, pela matrícula
58 alunoRouter.put('/aluno', async (req, res) => {
59     try {
60         await mongoose.connect(process.env.BD_STR_CON)
61         await aluno.findOneAndUpdate(
62             {
63                 "matricula": req.body.matricula
64             },
65             {
66                 "matricula": req.body.matricula,
67                 "nome": req.body.nome,
68                 "email": req.body.email,
69                 "cursoNome": req.body.cursoNome
70             }
71         )
72         res.json({mensagem: 'Aluno atualizado com sucesso'})
73     } catch (error) {
74         res.json({erro: true, mensagem: 'Erro durante consulta'})
75     }
76 })
```

Função put em uso.

Mongoose

```
78 //Rota DELETE para excluir um aluno, pela matrícula
79 alunoRouter.delete('/aluno', async (req, res) => {
80     try {
81         await mongoose.connect(process.env.BD_STR_CON)
82         await aluno.deleteOne({matricula: req.body.matricula})
83         res.json({mensagem: 'Aluno removido com sucesso'})
84     } catch (error) {
85         res.json({erro: true, mensagem: 'Erro durante consulta'})
86     }
87 })
88
89 module.exports = alunoRouter
```

Função delete em uso.