Criando uma aplicação – CRUD com banco de dados MongoDB

Crie as pastas 2oanosi e dentro dela projeto1

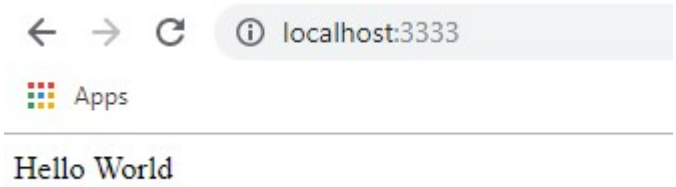
a) Entre na pasta projeto1 e digite o comando yarn init –y para criar o projeto
b) Instalar o express, digitando yarn add express
c) Instale o nodemon digitando yarn add nodemon -D
d) Observe no package.json que criou a dependência do express, a pasta node_modules e o arquivo yarn.lock que é um arquivo de cache
e) Crie a pasta src e dentro dela o arquivo index.js. Digite o código abaixo neste arquivo

```
const express = require('express')

const app = express()

app.get('/', (req, res) => {
    res.send('Hello World')
})

app.listen(3333)
```

Para facilitar a inicialização da aplicação, edite o arquivo backend/package.json e inclua as linhas 6, 7 e 8 conforme print abaixo.

```
1  {
2    "name": "backend",
3    "version": "1.0.0",
4    "main": "index.js",
5    "license": "MIT",
6    "scripts": {
7      "dev":  "nodemon src/index.js"
8    },
9    "dependencies": {
10     "express": "^4.17.1"
11   },
12   "devDependencies": {
13     "nodemon": "^1.19.1"
14   }
15 }
```

f) Inicie a aplicação digitando **yarn dev**
g) Abra o navegador digitando localhost:3333 e será exibida a janela abaixo.

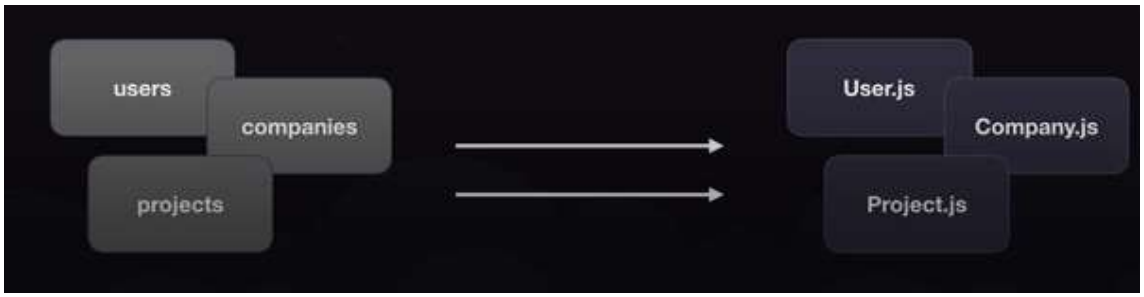


Mongoose

É um ORM (Object Relation Mapper) para NodeJS trabalhar banco de dados não relacionais.

O que é um ORM

- **Abstração do banco de dados:** É basicamente a forma de abstrair o banco de dados, mudar a forma como o banco de dados funciona e como a aplicação se comunica com ele.
- **Tabelas viram models:** Numa arquitetura MVC (Model – View e Controller) as tabelas viram Models, por exemplo, para as tabelas Users, Companies e Projects você terá arquivos JS para representa-las. Veja o print abaixo.



Manipulando os dados

Para manipular um dado do banco de dados, fazendo um insert, update ou delete não será utilizado SQL e sim código JavaScript e o Mongoose será responsável por gravar as informações diretamente no banco de dados MongoDB.

Exemplo 1:

- **Inserindo o nome e e-mail utilizando o SQL**

```
INSERT INTO users (name, email)
        VALUES (
                "Luiz Claudio",
                "lclsouza@hotmail.com"
        )
```

- **Inserindo nome e e-mail utilizando o Mongoose. O User é um model**

```
User.create({
        name: "Luiz Claudio",
        email: "lclsouza@hotmail.com",
})
```

Exemplo 2:

- **Pesquisando o email utilizando o SQL**

  Select *
  From Users
  Where email = lclsouza@hotmail.com
  Limit 1

- **Pesquisando o email utilizando o Mongoose. O User é um model**

```
User.findOne({
        where: {
                email: “lclsouza@hotmail.com”
        }
})
```

Para instá-lo digite **yarn add mongoose**

Edite o arquivo src/index.js conforme print abaixo. Insira as linhas 2, 6, 7 e 8. A linha 6 conecta no servidor onde está instalado o mongodb na porta 27017 e no banco de dados bdescola. O comando useNewUrlParser: true deve ser informado para você utilizar url de conexão no banco de dados.

index.js

```
1  const express = require('express')
2  const mongoose = require('mongoose')
3
4  const app = express()
5
6  mongoose.connect(`mongodb://localhost:27017/bdescola`, {
7    useNewUrlParser: true
8  })
9
10 app.get('/', (req, res) => {
11     res.send('Hello World')
12 })
13
14 app.listen(3333), () => {
15   console.log("Servidor rodando")
16 }
```

Criando as rotas da aplicação

Crie o arquivo src/routes.js

routes.js

```
1  const { Router } = require('express')
2
3  const routes = new Router();
4
5  routes.get('/', (req, res) => {
6    res.send('Hello World')
7  })
8
9  module.exports = routes
```

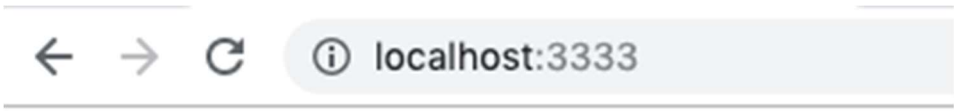
Importando as rotas no arquivo src/index.js

Insira a linha 10 e deixe o seu arquivo conforme o print abaixo.

index.js

```
1  const express = require('express')
2  const mongoose = require('mongoose')
3
4  const app = express()
5
6  mongoose.connect(`mongodb://localhost:27017/bdescola`, {
7    useNewUrlParser: true
8  })
9
10 app.use(require('./routes'))
11
12 app.listen(3333), () => {
13   console.log("Servidor rodando")
14 }
```

Abra o navegador e digite http://localhost:3333 e será exibida a página abaixo.



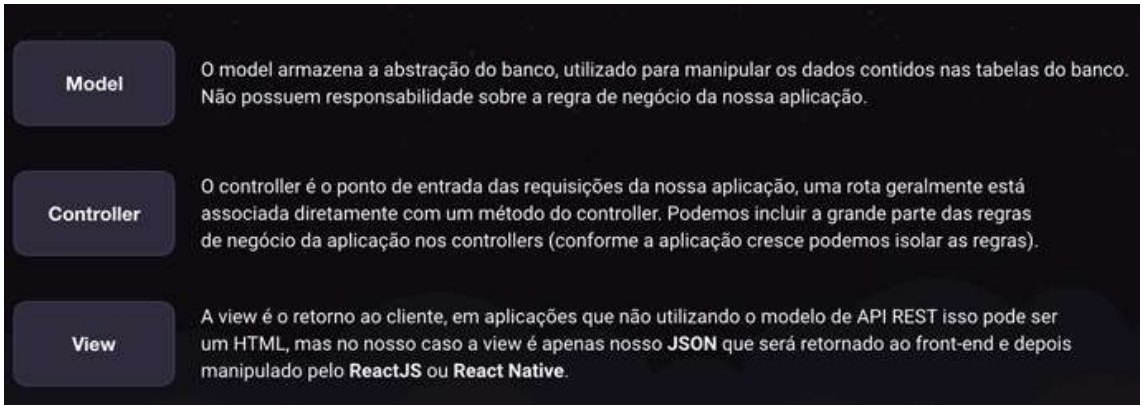
Hello World

Criando outras pastas do projeto

Dentro da pasta src, crie as pastas models (que terá como objetivo definir as tabelas se for um banco de dados relacionais ou schema para banco de dados não relacionais), controllers (onde serão definidas todas as regras de negócio da aplicação)

Arquitetura MVC

Ela consiste numa forma de estruturar as pastas e arquivos da aplicação a fim de separar as responsabilidades de cada tipo de arquivo. São três agentes principais: Model, View e Controller.



A face de um controller

O controller é uma **classe** e **sempre vai retornar um JSON** (resposta ou error). Ele nunca vai chamar outro método do controller. Ele é criado quando tiver uma nova entidade na aplicação. Entidade não é a mesma coisa que Model. Pode ser que você tenha controller e não tenha model, por exemplo, uma aplicação de autenticação, você tem um usuário que se autentica, A sessão será um controller e não terá um model, pois ela é referente ao usuário. Então será criado por exemplo, uma SessionController.

Um controller deve ter no máximo cinco métodos:

```
class UserController {

  index() { } // Listagem de usuários

  show() { } // Exibir um único usuário

  store() { } // Cadastrar usuário

  update() { } // Alterar usuário

  delete() { } // Remover usuário
```

Criando o Model

Crie o arquivo src/models/Users.js conforme print abaixo.

- Na linha 1 carrega o módulo do mongoose
- Na linha 3 cria um novo schema com os campos nome, email e senha e define o nome de UserSchema
- A linha 8 define que será criado os campos createdAt e updatedAt para armazenar a data de criação e alteração dos registros.
- A linha 11 exporta o UserSchema com o nome de Users

Users.js

```
1  const mongoose = require('mongoose')
2
3  const UserSchema = new mongoose.Schema({
4    nome: String,
5    email: String,
6    senha: String,
7  }, {
8    timestamps: true
9  })
10
11 module.exports = mongoose.model('Users', UserSchema)
```

Criando o Controller

Crie o arquivo src/controllers/UserController.js, com o objetivo de listar todos os usuários (método index), listar somente um usuário (método show), criar, alterar e deletar usuário (métodos store, update e delete), conforme print abaixo:

UserController.js

```
1  const User = require('../models/Users')
2
3  module.exports = {
4
5    async index(req, res) {
6      return res.json({ message: 'Listar todos usuários'})
7    },
8
9    async show(req, res) {
10     return res.json({ message: 'Listar um usuário'})
11   },
12
13   store(req, res) {
14     return res.json({ message: 'Gravar um usuário'})
15   },
16
17   async update(req, res) {
18     return res.json({ message: 'Alterar um usuário'})
19   },
```

UserController.js

```
20
21   async delete(req, res) {
22     return res.json({ message: 'Excluir um usuário'})
23   }
24 }
```

Criando as rotas para da aplicação

Edite o arquivo src/routes.js, conforme print abaixo:

- Na linha 2 importa o controller UserController.
- A linha 6 cria a rota para listar todos os usuários e executa o método index do UserController
- A linha 7 cria a rota para listar um único usuário e executa o método show do UserController
- A linha 8 cria a rota para gravar um usuário e executa o método store do UserController
- A linha 9 cria a rota para alterar um usuário e executa o método update do UserController
- A linha 10 cria a rota para listar excluir um usuário e executa o método delete do UserController

routes.js

```
1  const { Router } = require('express')
2  const UserController = require('./controllers/UserController')
3
4  const routes = new Router();
5
6  routes.get('/users', UserController.index)
7  routes.get('/users/:id', UserController.show)
8  routes.post('/users', UserController.store)
9  routes.put('/users/:id', UserController.update)
10 routes.delete('/users/:id', UserController.delete)
11
12 module.exports = routes
```

Utilizando o insomnia - Exercício

- Crie um workspace chamado 2oanoSI_crud_mongoDB.
- Crie uma pasta chamada usuários
- Crie uma requisição chamada **Listagem de todos os Usuários com o método GET** para acessar o seguinte endereço: http://localhost:3333/users
- Crie uma requisição chamada Listagem de um Usuário com o método GET para acessar o seguinte endereço: http://localhost:3333/users/1
- Crie uma requisição chamada **Criar Usuário com o método POST** para acessar o seguinte endereço: http://localhost:3333/users
- Crie uma requisição chamada **Alterar Usuário com o método PUT** para acessar o seguinte endereço: http://localhost:3333/users/1
- Crie uma requisição chamada **Excluir Usuário com o método DELETE** para acessar o seguinte endereço: http://localhost:3333/users/1

Gravando um usuário no banco de dados

Edite o arquivo src/index.js e insira a linha 4 para aceitar o formato json na requisição, conforme print a seguir

```
1  const express = require('express')
2  const mongoose = require('mongoose')
3  const app = express()
4  app.use(express.json())
5
6  mongoose.connect(`mongodb://localhost:27017/bdescola`, {
7      useNewUrlParser: true
8  })
9
```

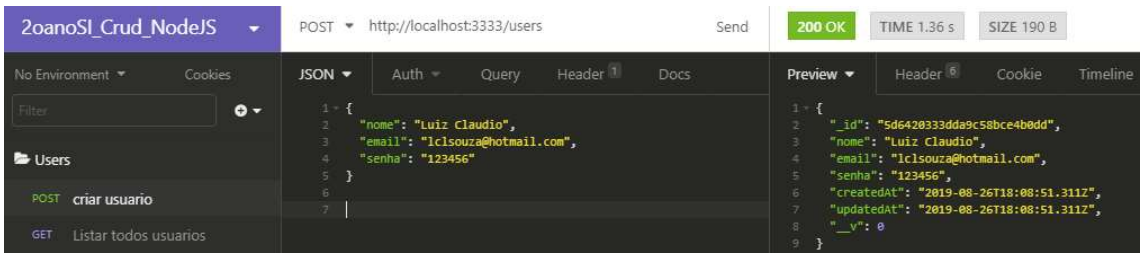
Edite o arquivo src/controllers/UserController.js,

No método store, insira da linha 13 até 23, conforme print a seguir

- A linha 14 cria uma desestruturação para receber os campos do formulário (req.body)
- A linha 16 cria um objeto usuarios, o mesmo será gravado no banco de dados com o comando create e o comando await faz com que o processo aguarde até ser finalizada a gravação.
- Na linha 22 retorna um json com os dados gravados no banco de dados.

```
9     async show(req, res) {
10        res.json( { message: 'Listar um usuário' })
11    },
12
13    async store(req, res) {
14        const { nome, email, senha } = req.body
15
16        const usuarios = await User.create({
17            nome,
18            email,
19            senha
20        })
21
22        return res.json(usuarios)
23    },
24
25    async update(req, res) {
```

Abra o insomnia e na requisição **Criar usuário**, crie um json conforme print a seguir e clique no botão Send para gravar no banco de dados

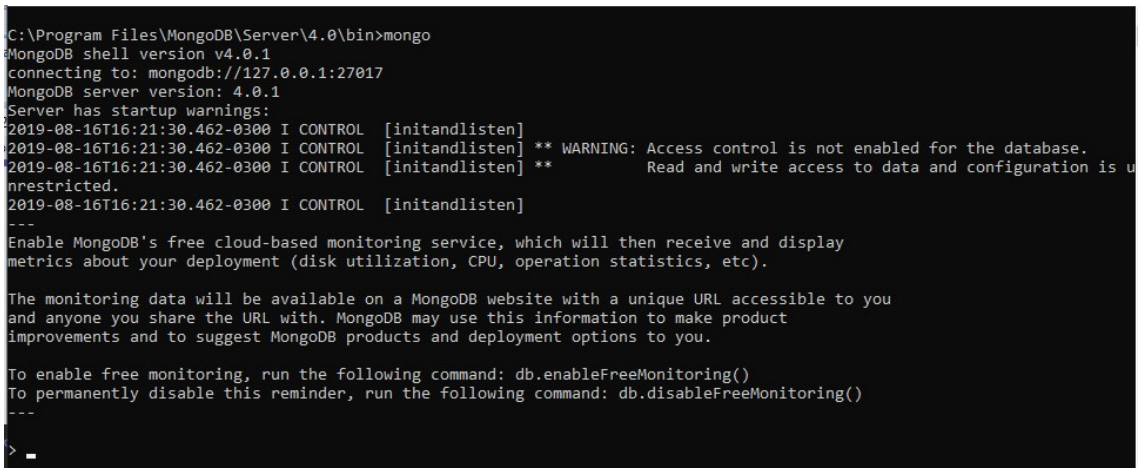


Observe no print acima, que o registro foi gravado no banco de dados bdescola do mongoDB na collection (tabela comparando com o banco de dados relacional) Users, pois foi gerado um id e criado os campos createdAt e updatedAt.

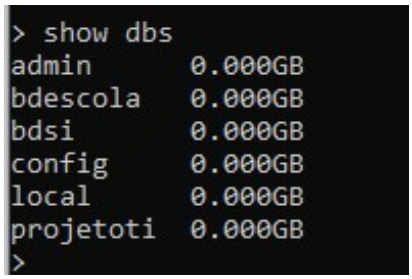
Visualizando os dados no banco de dados bdescola

Abra o seguinte caminho no explorer C:\Program Files\MongoDB\Server\4.0\bin e depois digite o comando cmd para abrir o prompt de comando.

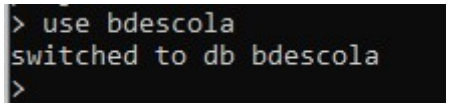
Digite o comando **mongo** para abrir o cliente do mongoDB, conforme print a seguir.



Para visualizar os bancos de dados criados, digite o comando **show dbs** conforme print a seguir.

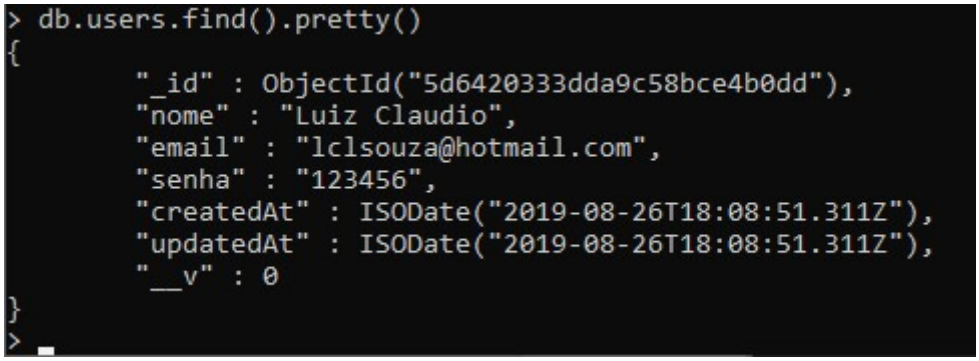


Para abrir o banco de dados bdescola, digite o comando **use bdescola**, conforme print a seguir.



Para visualizar as collections (tabelas) do banco de dados bdescola, digite o comando **show collections**

Para visualizar os registros da collections users, digite o comando **db.users.find().pretty()**



Insira mais 3 usuários no banco de dados.