Laboratório de Desenvolvimento Web

Prof. Ricardo Maroquio

Roteiro da Aula de 18/10/2022

1. Abrir VSCode em uma pasta de trabalho vazia
2. No terminal:
    a. npm init -y
3. Em package.json
    a. "type": "module"
    b. "start": "nodemon index.js"
4. No terminal:
    a. npm i express mongoose dotenv
5. Criar arquivo *index.js* na raiz:

```
import express from "express";
import dotenv from "dotenv";
const app = express();
dotenv.config();
app.listen(8080, () => {
    console.log("Servidor rodando na porta 8080.");
});
```

6. No terminal:
    a. npm i -g nodemon
    b. npm run start
7. Entrar em https://www.mongodb.com/
    a. Criar conta e logar
    b. Create -> Shared (Free) -> AWS -> São Paulo -> Name: ClusterIfes -> Create Cluster
    c. Security -> Database Access -> Add New Database User -> User: ifes -> Password : ifes123456 -> Read and write to any database -> Add User
    d. Database -> Browse Collections -> Create Database -> Database: academia, Collection: aluno -> create
    e. Database -> Connect -> Connect your application -> copiar URL
    f. Security -> Network Access -> Adicionar IP 0.0.0.0/0 (acesso irrestrito), caso não tenha
8. Criar arquivo *.env* na raiz e colar URL copiada do Atlas, substituindo itens em negrito:
    a. MONGODB_ATLAS=mongodb+srv://**ifes**:**ifes123456**@clusterifes.gljraae.mongodb.net/**academia**?retryWrites=true&w=majority
9. Criar arquivo *db.js* na pasta *config*:

```
import mongoose from "mongoose";
const connectDatabase = async () => {

    try {
        await mongoose.connect(process.env.MONGODB_ATLAS);
    } catch (error) {
        throw error;
    }
};
mongoose.connection.on("disconnected", () => {
    console.log("Desconectado do MongoDB.");
});
mongoose.connection.on("connected", () => {
    console.log("Conectado ao MongoDB.");
});
mongoose.connection.on("error", (error) => {
    console.log(`Erro no MongoDB:\n${error}`);
});
export default connectDatabase;
```

10. Adicionar ao arquivo .env a string para conexão ao MongoDB local (usar quando necessário):
    a. MONGODB_LOCAL= mongodb://localhost:27017/academia

11. Modificar index.js

```
import express from "express";
import dotenv from "dotenv";
import connectDatabase from "./config/db.js";
const app = express();
dotenv.config();
app.listen(8080, () => {
    connectDatabase();
    console.log("Servidor rodando na porta 8080.");
});
```

12. Criar arquivo *alunoRoutes.js* na pasta *routes*:

```
import express from "express";
const router = express.Router();
router.post("/", (req, res) => {
    res.send("Rota de cadastro de aluno.");
});
router.put("/:id", (req, res) => {
    res.send("Rota de atualização de aluno.");
});
router.delete("/:id", (req, res) => {
    res.send("Rota de remoção de aluno.");
});
router.get("/:id", (req, res) => {
    res.send("Rota de busca de aluno.");
});
router.get("/", (req, res) => {
    res.send("Rota de listagem de alunos.");
});
export default router;
```

13. Modificar *index.js*

```
import express from "express";
import dotenv from "dotenv";
import connectDatabase from "./config/db.js";
import alunoRoutes from "./routes/alunoRoutes.js";
const app = express();
dotenv.config();
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use("/api/alunos", alunoRoutes);
app.listen(8080, () => {
    connectDatabase();
    console.log("Servidor rodando na porta 8080.");
});
```

14. Testar rotas de alunos com Thunder Client
    a. GET http://localhost:8080/api/alunos/
    b. POST http://localhost:8080/api/alunos/
    c. PUT http://localhost:8080/api/alunos/ID_EXISTENTE
    d. DELETE http://localhost:8080/api/alunos/ID_EXISTENTE
    e. GET http://localhost:8080/api/alunos/ID_EXISTENTE

15. Criar arquivo *Aluno.js* na pasta *models*
https://mongoosejs.com/docs/schematypes.html

```
import mongoose from "mongoose";
const { Schema } = mongoose;
const AlunoSchema = new Schema(
    {
        nome: {
            type: String,
            required: true,
            trim: true,
        },
        dataNascimento: {
            type: Date,
            required: true,
        },
        sexo: {
            type: String,
            required: true,
            trim: true,
            enum: ["M", "F"],
        },
        email: {
            type: String,
            required: true,
            trim: true,
            unique: true,
            index: true,
        },
        senha: {
            type: String,
            required: true,
        },
        ativo: {
            type: Boolean,
            default: false,
        },
    },
    { timestamps: true }
);
export default mongoose.model("Aluno", AlunoSchema);
```

16. Modificar *alunoRoutes.js* da pasta *routes*:

```
import express from "express";
import Aluno from "../models/Aluno.js";
const router = express.Router();
router.post("/", async (req, res) => {
    const aluno = new Aluno(req.body);
    try {
        const novoAluno = await aluno.save();
        res.status(201).json(novoAluno);
    } catch (error) {
        throw error;
    }
});
...
```

17. Testar a rota do POST de alunos e observar retorno:
    a. POST http://localhost:8080/api/alunos/

       BODY:

       ```
       {
        "nome": "Aluno Padrão",
        "dataNascimento": "2000-10-25",
        "sexo": "M",
        "email": "aluno@email.com",
        "senha": "123456",
        "ativo": true
       }
       ```

18. Criar arquivo *error.js* na pasta *utils* para conter o módulo de tratamento de exceção:

```
const createError = (statusCode, message) => {
    const error = new Error(message);
    error.status = statusCode;
    return error;
};
const errorHandling = (err, req, res, next) => {
    const errorStatus = err.status || 500;
    const errorMessage = err.message || "Erro interno do servidor.";
    res.status(errorStatus).json({
        success: false,
        status: errorStatus,
        message: errorMessage,
        stack: err.stack,
    });
};
export { createError, errorHandling };
```

19. Modificar *index.js*:

```
import express from "express";
import dotenv from "dotenv";
import connectDatabase from "./config/db.js";
import alunoRoutes from "./routes/alunoRoutes.js";
import { errorHandling } from "./utils/error.js";
const app = express();
dotenv.config();
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(errorHandling);
app.use("/api/alunos", alunoRoutes);
...
```

20. Criar arquivo *alunoController.js* na pasta *controllers*:

```
import Aluno from "../models/Aluno.js";
export const createAluno = async (req, res, next) => {
    const aluno = new Aluno(req.body);
    try {
        const createdAluno = await aluno.save();
        res.status(201).json(createdAluno);
    } catch (error) {
        next(error);
    }
};
export const updateAluno = async (req, res, next) => {
    try {
        const updatedAluno = await Aluno.findByIdAndUpdate(req.params.id, { $set:
req.body }, { new: true });
        res.status(200).json(updatedAluno);
    } catch (error) {
        next(error);
    }
};
export const deleteAluno = async (req, res, next) => {
    try {
        await Aluno.findByIdAndDelete(req.params.id);
        res.status(200).json({ message: "Aluno excluído com sucesso." });
    } catch (error) {
        next(error);
    }
};
export const getAluno = async (req, res, next) => {
    try {
        const aluno = await Aluno.findById(req.params.id);
        res.status(200).json(aluno);
    } catch (error) {
        next(error);
    }
};
export const getAlunos = async (req, res, next) => {
    try {
        const alunos = await Aluno.find();
        res.status(200).json(alunos);
    } catch (error) {
        next(error);
    }
};
```

21. Modificar arquivo *alunoRoutes.js*:

```
import express from "express";
import { createAluno, updateAluno, deleteAluno, getAluno, getAlunos } from
"../controllers/AlunoController.js";
const router = express.Router();
router.post("/", createAluno);
router.put("/:id", updateAluno);
router.delete("/:id", deleteAluno);
router.get("/:id", getAluno);
router.get("/", getAlunos);
export default router;
```

22. Testar rotas como no passo 14 usando *id*s de documentos existentes.