Inicio da aula

Criar uma pasta nova chamada JWT

Explicar que é o JWT, autenticação de usuários

Entrar dentro da pasta com o VS CODE e abrir o terminal

No terminal inicializar o projeto → `npm init -y`

Instalar a dependencia (pacotes) bcrypt → `npm install bcrypt` → Criptografar e Descriptografar a senha

Instalar a dependencia dotenv → `npm install dotenv` → Configuração das variaveis de ambiente

Instalar a dependencia express → `npm install express` → Configuração das rotas da API

Instalar a dependencia jsonwebtoken → `npm install jsonwebtoken` → Gerenciar os tokens de acesso

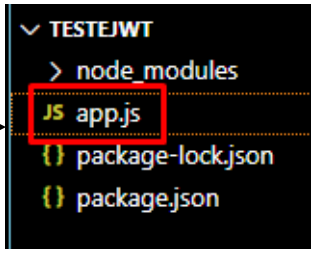
Instalar a dependencia mongoose → `npm install mongoose` → Conexão com o banco de dados

Instalar a dependencia nodemon → `npm install nodemon` → Atualiza automaticamente as modificações

no PACKAGE.JS, alterar os comandos de script para inicializar o servidor

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "dev": "nodemon ./app.js",
  "start": "node ./app.js"
},
```

Criar o arquivo APP.JS na raiz do projeto



no terminal dar o comando para subir o servidor local no modo desenvolvimento

```
PS D:\Professor_Vini\Experts\TesteJWT> npm run dev

> testejwt@1.0.0 dev
> nodemon ./app.js

[nodemon] 2.0.16
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./app.js`
[nodemon] clean exit - waiting for changes before restart
```

no APP.JS, programar para configurar os pacotes e servidor

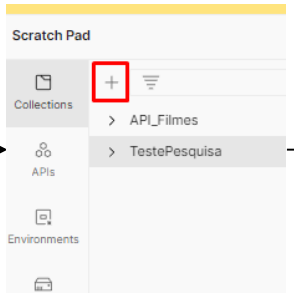
```
//Importar os pacotes
require('dotenv').config()
const express = require('express')
const mongoose = require('mongoose')
const bcrypt = require('bcrypt')
const jwt = require('jsonwebtoken')


//Chamar o pacote express
const app = express()

//Configurar o tipo de envio e recebimento de dados em JSON
app.use(express.json())

//Configurar a porta do servidor
app.listen(3000)
```

no APP.JS, montar a primeira rota publica

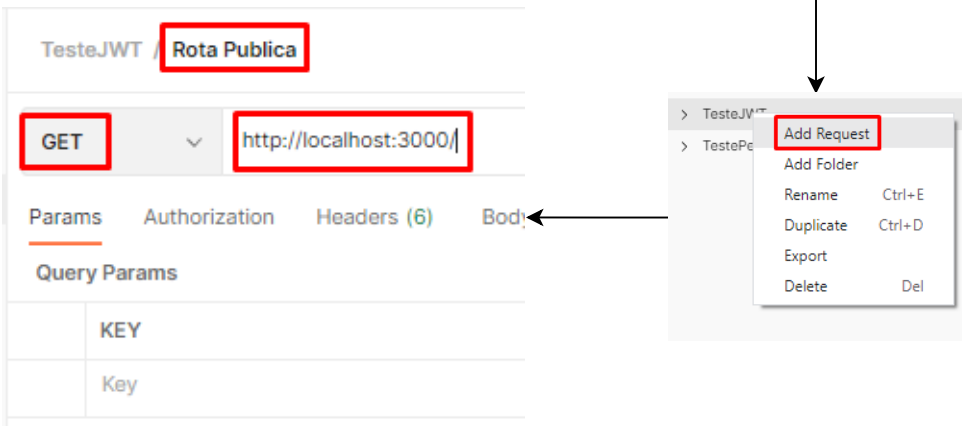
```
//Criar uma Rota Publica (Sem segurança)
app.get('/', (req, res) => {
    res.status(200).json({msg: "Bem Vindo ao Sistema de Login"})
})
```

Testar no Postman

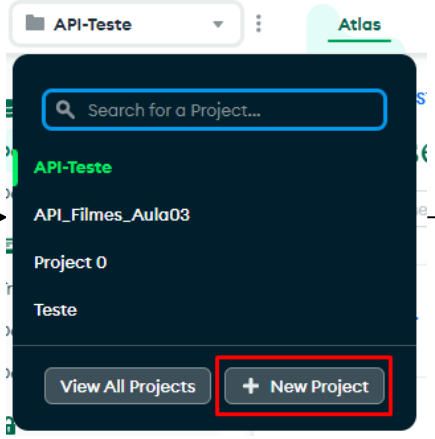
Criar uma nova colection

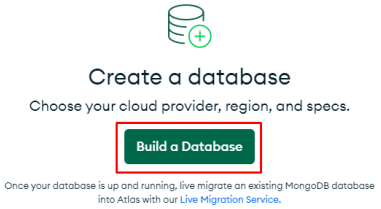


Criar uma nova request



Entrar no mongoDB e criar um novo projeto



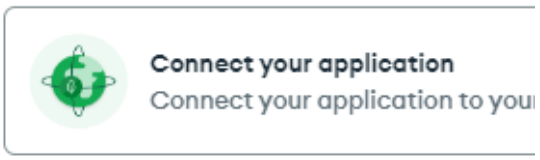


Escolher a versão free e criar o cluster sem modificar nada na configuração

Criar o usuário com a senha super123

Configurar o IP de acesso para geral 0.0.0.0/0

no mongoDB, pegar o link de conexão



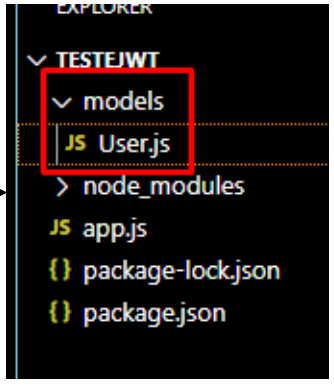

```
mongodb+srv://vinicius:<password>@cluster0.rr1ys.mongodb.net/?
retryWrites=true&w=majority
```

no APP.JS, programar a conexão com o banco de dados

Lembrar de add a senha no link de conexão

```
//Configurar a porta do servidor
app.listen(3000)

//Conectar com o Mongo DB
mongoose.connect("mongodb+srv://vinicius:super123@cluster0.rr1ys.mongodb.net/?retryW
    .then(() => {
        console.log("Conectado ao MongoDB")
    })
    .catch((erro) => {
        console.log(erro)
    })
```

criar a pasta models e a entidade User.js



em USER.JS, configurar a entidade

```
const mongoose = require('mongoose')

const User = mongoose.model('User', {
    name: String,
    email: String,
    password: String,
})

module.exports = User
```

no APP.JS, importar o model user

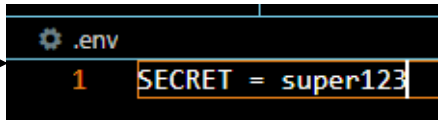
```
//Importar os pacotes
require('dotenv').config
const express = require('express')
const mongoose = require('mongoose')
const bcrypt = require('bcrypt')
const jwt = require('jsonwebtoken')

//Models
const User = require('./models/User')
```

no APP.JS, criar a rota de criação de usuário

```
//Criar a rota de Cadastro de Usuário
app.post('/auth/register', async(req,res) => {
    const {name, email, password} = req.body

    //verificar se o usuário já existe no banco
    const userExists = await User.findOne({email: email})
    if (userExists) {
        return res.status(422).json({message: "Usuário já existe! Escolha outro email."})
    }

    //Criar o password Criptografado
    const salt = await bcrypt.genSalt(12)
    const passwordHash = await bcrypt.hash(password, salt)

    //criar o usuário
    const user = new User({
        name,
        email,
        password: passwordHash
    })
    try{
        await user.save()
        res.status(201).json({msg: "usuário criado com sucesso"})
    } catch (erro){
        res.status(500).json({msg: "Erro ao cadastrar"})
    }
})
```

Testas no Postman e verificar no mongo DB

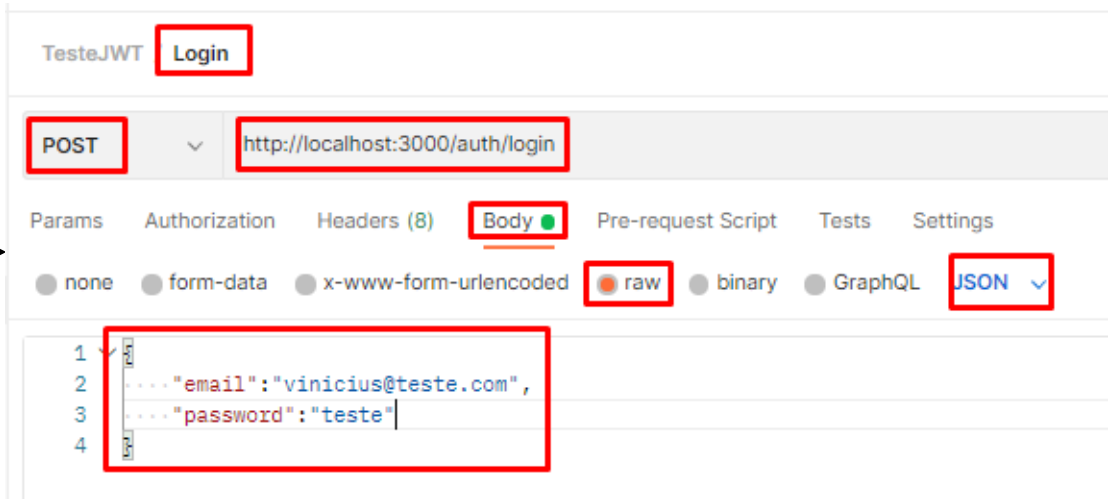
no projeto criar um arquivo .env com o segredo para o token



no arquivo APP.JS, criar a rota de login

```
//Rota de Login
app.post("/auth/login", async(req, res) => {
    const {email, password} = req.body

    //Verificar se o usuário existe
    const user = await User.findOne({email: email})

    if (!user) {
        return res.status(404).json({msg: "Usuário não encontrado !!!"})
    }

    //Verificar a senha
    const checkPassword = await bcrypt.compare(password, user.password)

    if (!checkPassword){
        return res.status(422).json({msg: "Senha Inválida !!!"})
    }

    //Fazer a autenticação do usuário atraves do Token
    try{
        const secret = process.env.SECRET
        const token = jwt.sign(
            {
                id: user._id,
            },
            secret,
        )

        res.status(200).json({msg: "Autenticação realizada com sucesso", token})

    } catch (erro){
        res.status(500).json({msg:"Aconteceu um erro na autenticação!!!"})
    }
})
```

Testar a rota no postman



no APP.JS, criar uma função para verificar o token

```
function checkToken(req, res, next){
    const authHeader = req.headers['authorization']
    const token = authHeader && authHeader.split(" ")[1]

    if(!token){
        return res.status(401).json({msg: "Acesso negado!"})
    }

    try{
        const secret = process.env.SECRET

        jwt.verify(token, secret)

        next()

    } catch(erro){
        res.status(400).json({msg: "Token inválido !"})
    }
}
```

no APP.JS, criar uma rota privada que só é utilizado depois de ter feito o login

```
//Criar uma Rota Privada (Só é acessada quando o login é feito)
app.get('/user/:id', checkToken, async(req, res) => {
    const id = req.params.id

    //Verificar se o usuário existe
    const user = await User.findById(id, '-password')
    if (!user) {
        return res.status(404).json({msg: "Usuário não encontrado"})
    }

    res.status(200).json({user})
})
```

Testar no postman, colocando o token no header da autorização.