



Regional Distrito Capital
Centro de Gestión de Mercados, Logística y
Tecnologías de la Información


Aprendiz: Yimi Ruiz

Instructor: Jesús Ropero Barbosa

JAVASCRIPT

API TAREAS (BACKEND)

Área Teleinformática
Bogotá, marzo 2020

 Sistema de Gestión de la Calidad	Regional Distrito Capital Centro Gestión de Mercados, Logística y Tecnologías de la Información PROGRAMA DE FORMACION Construcción del diseño de la estructura de la red de datos	Fecha: Versión: 1 Página 2 de 6
---	--	---

DESARROLLO API

Para la instalación de las librerías se tiene que realizar los siguientes comandos:

nos ubicamos en backend y ejecutamos

1) Npm init

Esto activara la inicialización del proyecto

NPM : Es un gesto de paquetes para JavaScript, sirve para gestionar versiones de paquetes y librerías

Después procedemos a crear el archivo index.js

1) npm install express

express es un marco de aplicación web, se utiliza para diseñar y crear aplicaciones web de forma rápida

Se instala la librería js para el mapeo de la bd mongo

2) Npm install mongoose

Se instala el sistema de autenticación json web token

3) Npm install jsonwebtoken


Una vez terminada la instalación de librerías, realizamos los siguientes pasos:

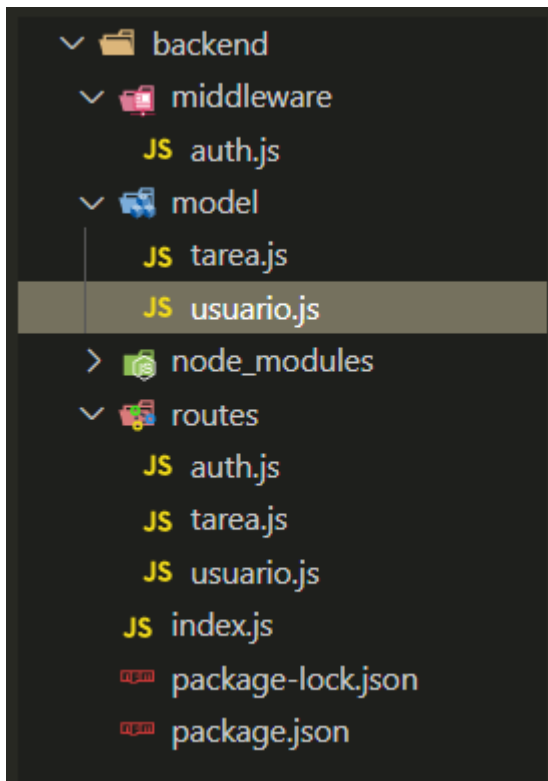
Dentro de la carpeta backend crearemos las siguientes carpetas

Model

Routes

En estas almacenaremos los scripts js para rutas y modelos

 Sistema de Gestión de la Calidad	Regional Distrito Capital Centro Gestión de Mercados, Logística y Tecnologías de la Información PROGRAMA DE FORMACION Construcción del diseño de la estructura de la red de datos	Fecha: Versión: 1 Página 3 de 6
---	--	---



Explicación de modulo usuario


Hacemos uso de mongoose y jsonwebtoken como módulos internos

```
//modulos internos Modelo usuario

/* */
const mongoose = require("mongoose");
const jwt = require("jsonwebtoken");
```

Creamos el esquema de la colección usuario

```
const esquemaUsuario = new mongoose.Schema({
  nombre: String,
  correo: String,
  contraseña: String,
});
```

 Sistema de Gestión de la Calidad	Regional Distrito Capital Centro Gestión de Mercados, Logística y Tecnologías de la Información PROGRAMA DE FORMACION Construcción del diseño de la estructura de la red de datos	Fecha: Versión: 1 Página 4 de 6
---	--	---

Generamos el jsonwebtoken con el esquema usuario

```
esquemaUsuario.methods.generateJWT = function(){
  return jwt.sign({
    /*NOTAS:
    mongo nos crea un id por defecto*/
    _id:this._id,
    nombre:this.nombre,
    correo:this.correo,
  }, "clave");
};
```

se realizan los exports

```
const Usuario = mongoose.model("usuario",esquemaUsuario);
module.exports.Usuario = Usuario;
module.exports.esquemaUsuario = esquemaUsuario;
```


EXPLICACION DE LA RUTA USUARIO

Se hace uso de los módulos internos

```
const express = require("express");
const router = express.Router();
```

Creamos los módulos propios, se define la clase usuario

```
const { Usuario } = require("../model/usuario");
```

 Sistema de Gestión de la Calidad	Regional Distrito Capital Centro Gestión de Mercados, Logística y Tecnologías de la Información PROGRAMA DE FORMACION Construcción del diseño de la estructura de la red de datos	Fecha: Versión: 1 Página 5 de 6
---	--	---

Creamos las rutas necesarias

```
router.post("/", async (req, res) => {
  //preguntamos cuando el usuario exista
  let usuario = await Usuario.findOne({ correo: req.body.correo });
  //si el usuario existe en la base de datos
  if (usuario) return res.status(400).send("El usuario existe en la BD");
  //si el usuario no existe continua
  usuario = new Usuario({
    //lo que venga del cuerpo del json y tenga tal propiedad
    nombre: req.body.nombre,
    correo: req.body.correo,
    contrasena: req.body.contrasena,
  });
  //vamos a guardar el usuario en la BD y generamos el jwt
  const result = await usuario.save();
  const jwtToken = usuario.generateJWT();
  res.status(200).send({ jwtToken });
});
```

Por último, los exports

```
module.exports = router;
```


EXPLICACION DEL ARCHIVO INDEX.JS

Uso de módulos internos

```
const express = require("express");
const mongoose = require("mongoose");
```

Uso de módulos propios

```
//consumimos las rutas
const usuario = require("../routes/usuario");
```

 Sistema de Gestión de la Calidad	Regional Distrito Capital Centro Gestión de Mercados, Logística y Tecnologías de la Información PROGRAMA DE FORMACION Construcción del diseño de la estructura de la red de datos	Fecha: Versión: 1 Página 6 de 6
---	--	---

Se define la ruta

```

// como
app.use("/api/usuario",usuario);

```

Puerto de ejecución de nuestro api

```

const port = process.env.port || 3004;
app.listen(port,() => console.log("escuchando el puerto: " +port));

//definir funcion mongoose le decimos cual es nuestro punto de conexion (base de datos)
mongoose.connect("mongodb://localhost/344tareasbd",{
  useNewUrlParser :true,
  useFindAndModify :false,
  useCreateIndex :true,
  useUnifiedTopology: true,
})

.then(() => console.log("conexion con mongo Ok!!"))
.catch((error) => console.log("Fallo la conexión!!" + error));

```

Para comprobar la conexión con mogodb ejecutamos el siguiente comando en una terminal o consola

Node index.js

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
2: node
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\xampp\htdocs\proyectos\js\4_trimestre\appTareas> cd backend
PS C:\xampp\htdocs\proyectos\js\4_trimestre\appTareas\backend> node index.js
escuchando el puerto: 3004
conexion con mongo Ok!!

```