GC-F-004v.08

Análisis y desarrollo de software

FASE 3 Ejecución – GA8

Diciembre 02 de 2025

@SENAComunica

www.sena.edu.co

Agenda del día

1. Saludo de Bienvenida.
2. Recomendaciones Generales
3. Evidencias a desarrollar

   ✓**Evidencia de desempeño: GA8-220501096-AA1-EV01 desarrollar *software* a partir de la integración de sus módulos componentes.**

4. Resolución de Dudas e Inquietudes
5. Despedida Equipo de Instructores

Actividades de aprendizaje de la competencia 220501096 - Desarrollar la solución de *software* de acuerdo con el diseño y metodologías de desarrollo

3.1.1 Actividad de aprendizaje GA8-220501096-AA1- Integrar módulos

Esta actividad se centra en el estudio de los mecanismos de desarrollo de *software* que permiten construir los componentes que conforman o integran los módulos de una aplicación web.

Duración: 96 horas.
Materiales de formación: para el desarrollo de esta actividad es importante la lectura y análisis del material de formación: “Construcción aplicación web.”.

Evidencias:
A continuación, se describen las acciones y las correspondientes evidencias que conforma la actividad de aprendizaje:

Evidencia de desempeño: GA8-220501096-AA1-EV01 desarrollar *software* a partir de la integración de sus módulos componentes.

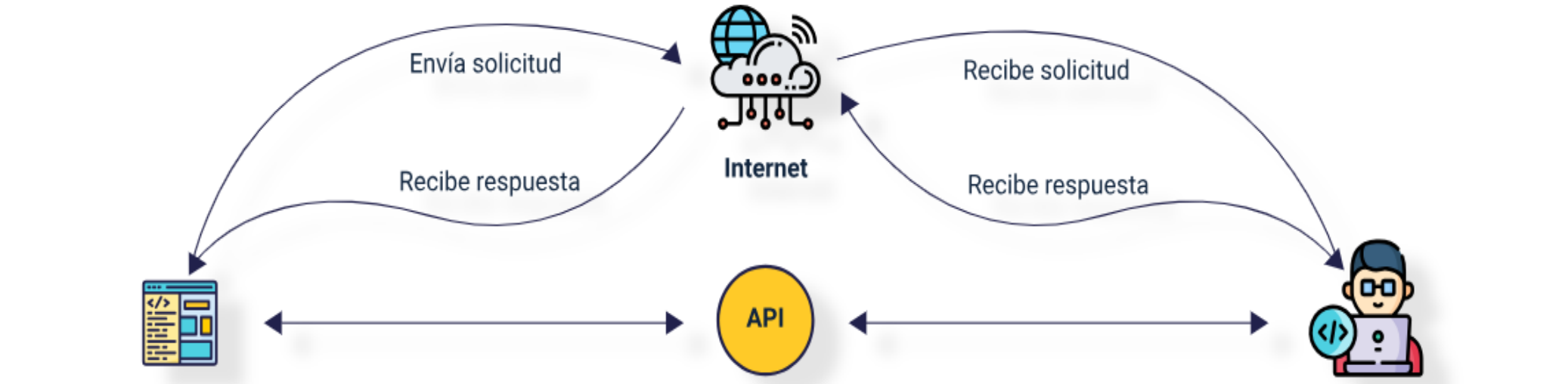
Construcción aplicación web

Introducción



En este componente formativo se abordarán los conceptos y técnicas que apoyarán el desarrollo de una aplicación web de manera completa, es decir, se realizará el Backend y Front-end de la aplicación, de esta forma se podrá observar la comunicación, intercambio y procesamiento de la información de estos dos componentes, por lo que se iniciará con la instalación de herramientas y recursos del lado del servidor, que gestionará la base o plataforma para el despliegue de la aplicación del lado del cliente. En la segunda parte de este documento se da alcance a la construcción de la aplicación web Responsive, que consumirá la API REST ya realizada en el servidor.

FrontEnd – BackEnd



FrontEnd

Programas del lado del cliente

JavaScript
Usando varias tecnologías

Comunicación FrontEnd / BackEnd

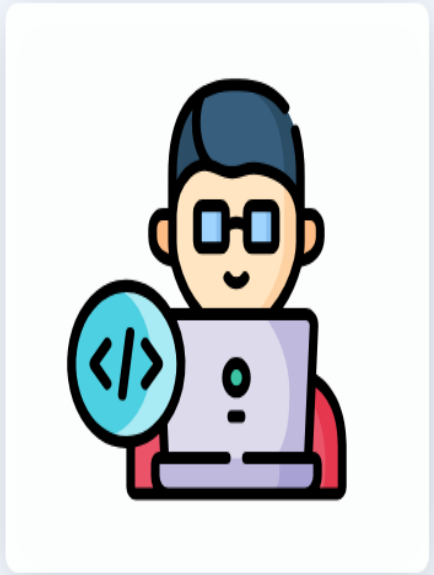
A Través de API

BacktEnd

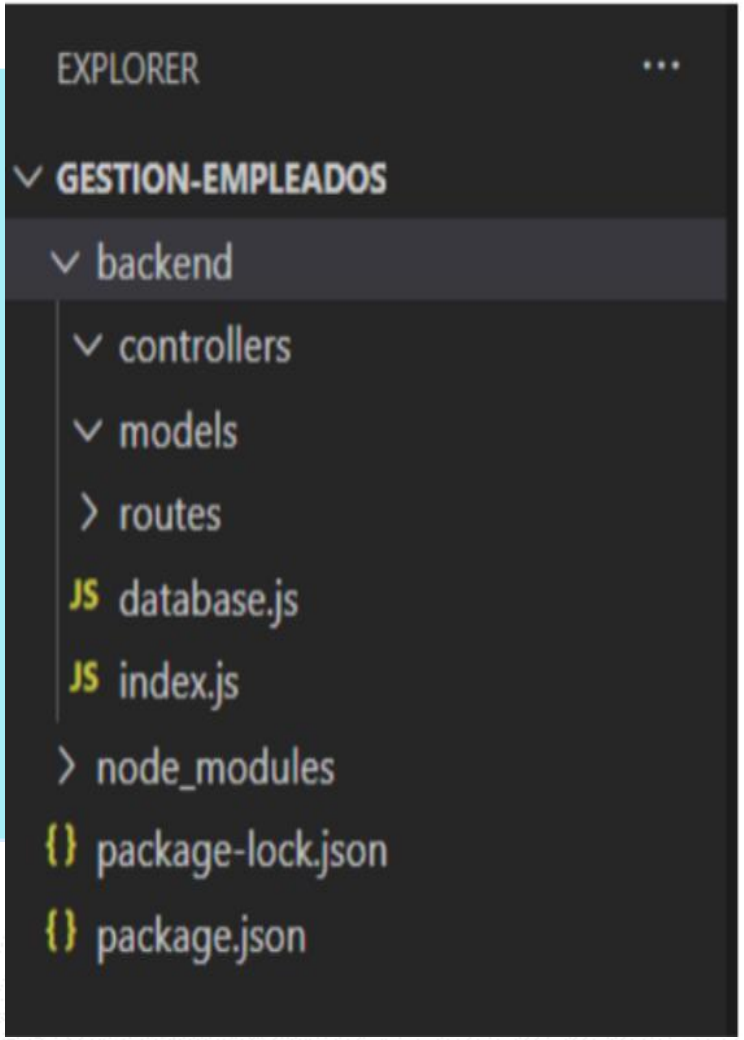
Programas del lado del servidor

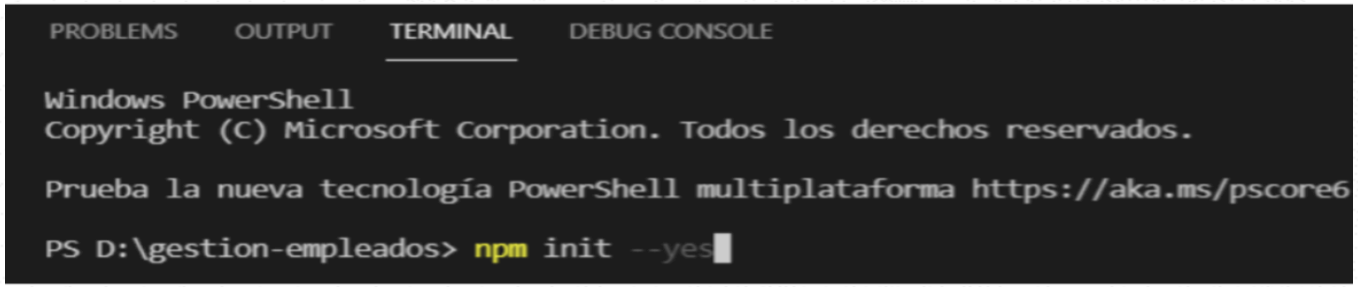
Varios Lenguajes
Usando varias plataformas

1 Construcción del Backend de la aplicación web



Se inicia con una estructura de proyecto MEAN (Mongo, Express, Angular y Node.js) para el proyecto que abarca tanto el Front-end como el Backend. Para la instalación de las primeras herramientas como Node.js y Express en la sección material complementario de este documento encontrará la guía de instalación de Node.js y de Express. A continuación, se incluye una figura que representa de manera general el proceso de codificación de *software*.





Una vez creadas las carpetas y archivos como lo son *controllers, models, routes*, database.js e index.js, se abre la terminal del VS Code y se digita el siguiente comando para crear un proyecto Node.js. **npm init --yes**

Una vez creado el proyecto se requiere todas las bondades del *framework* de Node.js, para lo cual se hace necesario Express, que va a generar una serie de componentes que ayudarán mucho al desarrollo de esta aplicación; por lo tanto, se instala Express con la siguiente línea de comandos desde el terminal del VS Code: **npm install express**

1.2 Desarrollo del index.js

```
const express = require('express')
const morgan = require('morgan');
const cors = require('cors');
const app = express(); // la constante app tendrá ahora todo el funcionamiento del servidor
const { mongoose } = require('./database'); // no se quiere todo el archivo sino la conexión
/** * Se crea una REST API, es la manera de decirle al servidor que reciba y envíe datos */
// Configuraciones
app.set('port', process.env.PORT || 3000);
app.use(morgan('dev'));
app.use(express.json()); // método que ayuda a convertir el código para que el servidor pueda entender lo que viene del cliente.
app.use(cors({origin: 'http://localhost:4200'})); // método para comunicar con el cliente
// rutas de nuestro servidor
app.use('/api/empleados',require('./routes/empleado.routes'));
// Iniciando el servidor
app.listen(app.get('port'), () => {// esta es una mejor manera de configurar el puerto
console.log('server activo en el puerto', app.get('port'));
});
```

En la figura anterior se aprecia el código del archivo index.js para el proyecto, por lo cual en la medida del avance de este componente se irá explicando cada línea de código que se requiera ejecutar. En ese sentido, se inicia con el requerimiento de Express y se almacena en una constante en JavaScript llamada Express. De esta forma se tendrá el acceso a todo el funcionamiento del servidor con la línea de código:

const app = express();

Una vez creada la constante app, se pasa a la configuración del puerto por donde va a escuchar y a recibir el servidor todo tipo de solicitudes por parte del cliente, según se muestra en la siguiente línea:

app.set('port', process.env.PORT || 3000)

Se instala Nodemon, como una utilidad que monitorea de manera constante los cambios en el código fuente que se está desarrollando y de manera automática reinicia el servidor. Por lo tanto, la línea de código utilizada en la terminal de VS Code es: **npm install nodemon --D**

Se configura el archivo package.json según se muestra en la siguiente figura (puede descargar el contenido de estas instrucciones en el archivo package.json disponible en la carpeta), y se inicia a ejecutar el servidor con la siguiente línea de comando: **npm run dev**

Estructura package.json

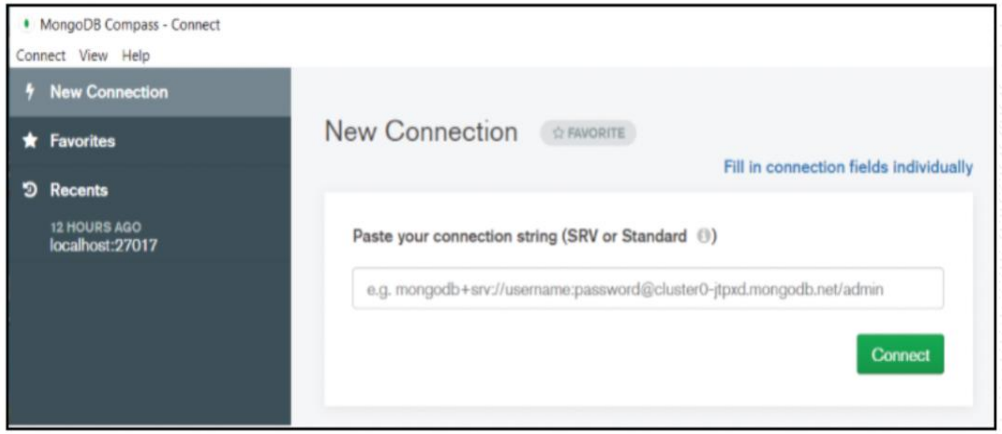
```
{
  "name": "gestion-empleados",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "dev": "nodemon BackEnd/index.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^4.17.1",
    "mongoose": "^5.12.14",
    "Morgan": "^1.10.0"
  },
  "devDependencies": {
    "nodemon": "^2.0.7"
  }
}
```

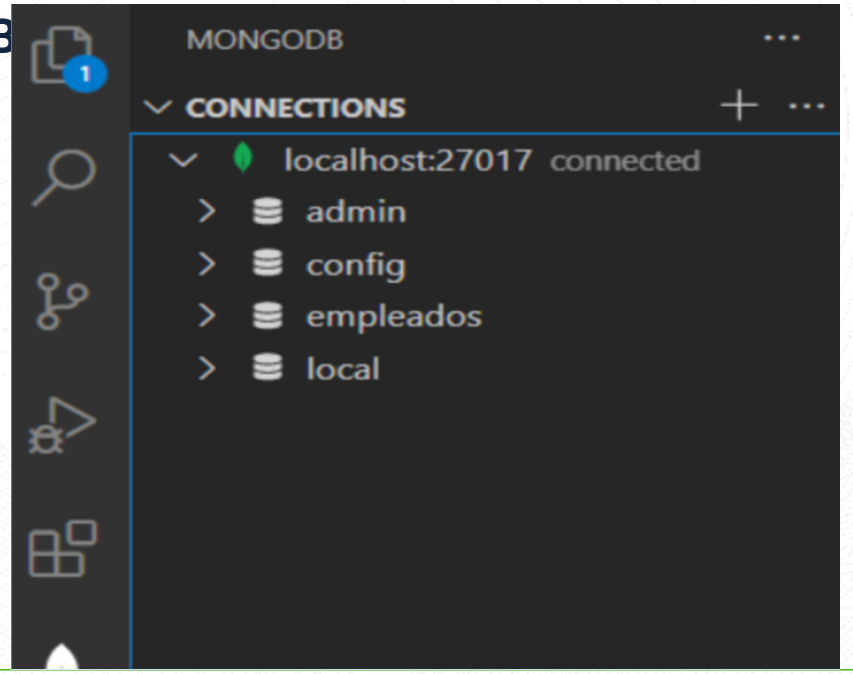
Se construyen las Middleware dentro del archivo de index.js como se puede apreciar en la figura anterior, encargadas de procesar los datos, es decir, cuando se solicitan o envían al servidor. Al respecto, el servidor debe entender los datos que le llegan del cliente (AngularJS toma los datos del cliente y los envía en formato JSON) y es por eso que se necesita algún tipo de conversión, por lo que se requiere del uso del formato JSON ejecutando la siguiente instrucción:

app.use(express.json());

Se tiene una herramienta útil llamada *morgan*, que ayuda a mirar en consola lo que el usuario está solicitando. Esta herramienta es opcional para el desarrollo de la aplicación, no obstante se deja la línea de comando para su instalación y utilización: **npm install Morgan** **app.use(morgan('dev'));**

1.3 Configuración e instalación de la base de datos MongoDB





Se instala la herramienta Mongoose con la siguiente línea de comando desde la terminal de VS Code:

npm install mongoose

1.4 Configuración de database.js

npm run dev

```
const mongoose = require('mongoose');
  const URI = 'mongodb://localhost/empleados';
  mongoose.connect(URI)
    .then(db => console.log('DB is connected'))
    .catch(err => console.error(err));
module.exports = mongoose;
```

1.5 Configuración de las rutas del proyecto con Node.js

El archivo index.js, en el cual se establece la ruta por donde se van a recibir o enviar los datos desde el servidor que para el caso de estudio será **http://localhost:3000/api/empleados**, de tal forma se observa que para la definición de esta ruta se hace necesario requerir el archivo empleado.routes.js ejecutando la siguiente instrucción:

app.use('/api/empleados',require('./routes/empleado.routes'));

```
/**
* Vamos a crear rutas del servidor
* creamos un módulo por eso utilizamos express
* vamos a utilizar como nuestra rest api para
* enviar y recibir datos en formato json
*/
const express = require('express');
const router = express.Router();
/* generamos un ejemplo cuando le soliciten
algo al servidor por el método GET **/
router.get('/', (req, res) => {

   res.json({
     status: 'API REST funcionando'
   });
})
module.exports = router;
```

Estructura empleado.js

```
const mongoose = require('mongoose');
const {Schema} = mongoose;

const EmpleadoSchema = new Schema({
    name: {type:String, require:true},
    position: {type:String, require:true},
    office: {type:String, require:true},
    salary: {type:Number, require:true},
});

module.exports = mongoose.model('Empleado', EmpleadoSchema);
```

Desarrollo de controladores para la API REST

```
* Se coloca el controlador como un objeto y luego se exporta como
* se requiere primero el modelo empleado
*/

const Empleado = require('../models/empleado');
const empleadoCtrl = {};

/**
* DEFINO LOS MÉTODOS */

//Obtener todos los empleados
empleadoCtrl.getEmpleados = async (req, res) => {
    const empleados = await Empleado.find();
    res.json(empleados);
}

// Crear empleados

empleadoCtrl.createEmpleados = async (req, res) => {
    const empleado = new Empleado(req.body);
    await empleado.save();
    res.json({
      'status': 'Empleado guardado'
    });
```

Código de la ruta empleado

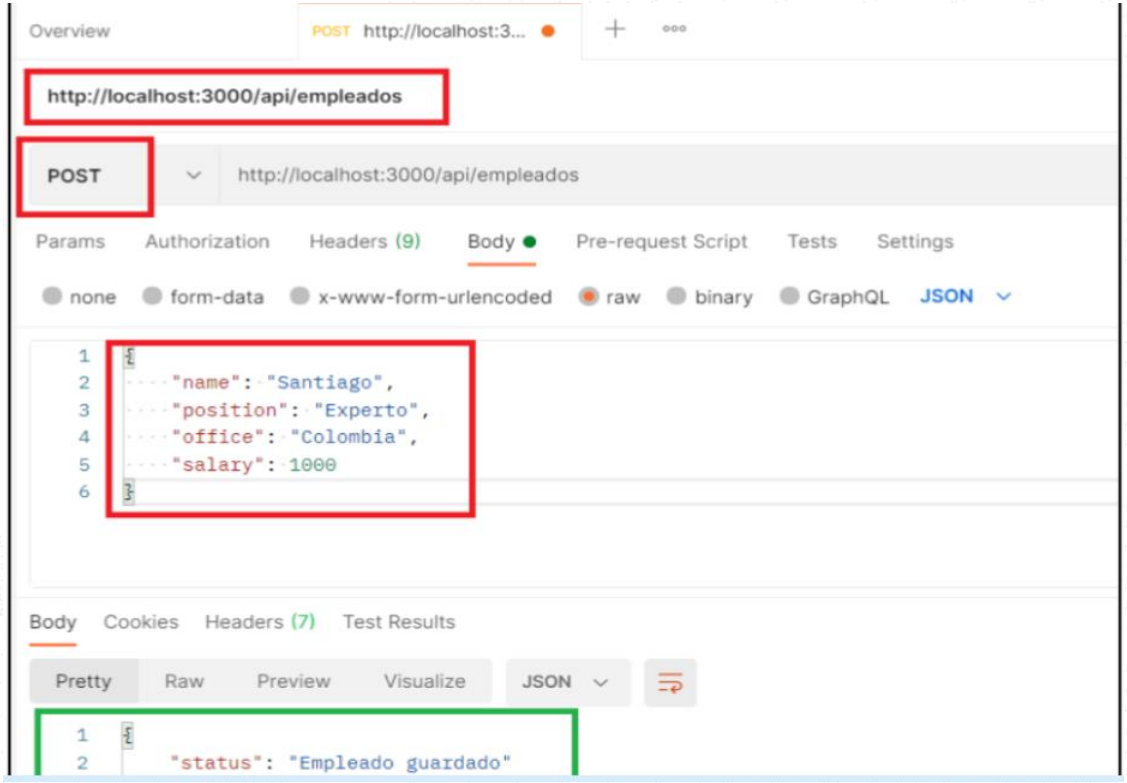
```
/**
*
* creamos un módulo por eso utilizamos express
*/

const express = require('express');
const router = express.Router();
const empleadoCtrl = require('../controllers/empleado.controller');

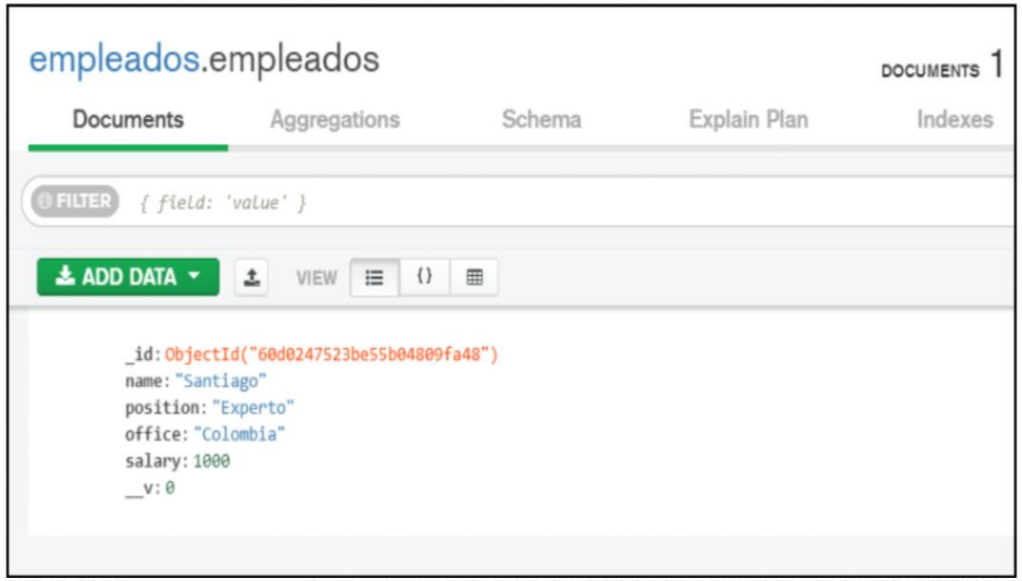
router.get('/', empleadoCtrl.getEmpleados); // Rutas más limpias (obtener empleados)
router.post('/', empleadoCtrl.createEmpleados);//guardar
router.get('/:id', empleadoCtrl.getUnicoEmpleado);// obtiene un único empleado
router.put('/:id',empleadoCtrl.editarEmpleado); //Actualizar datos (uno a la vez)
router.delete('/:id', empleadoCtrl.eliminarEmpleado);

module.exports = router;
```

Realización de pruebas de la API REST con Postman



Registro de empleado almacenado en MongoDB



● Evidencia de desempeño: GA8-220501096-AA1-EV01 desarrollar *software* a partir de la integración de sus módulos componentes

En esta evidencia el estudiante debe codificar los módulos del proyecto con base en los requerimientos del sistema para una aplicación orientada a la web.

Elementos a tener en cuenta:

- Estudiar detenidamente los conceptos y características definidas en el componente formativo.
- Se deben conocer los requerimientos del sistema.
- Manejar los documentos de casos de uso o historias de usuario.
- Conocer el funcionamiento del IDE de desarrollo.
- Se debe conocer el diagrama de clases.
- Se debe conocer el diagrama de paquetes.
- Se debe conocer el diagrama de componentes.
- Se debe conocer los mecanismos de seguridad que requiere la aplicación.
- Identificar las capas en donde se ubican los componentes.
- Conocer la metodología de desarrollo de *software*.
- Conocer el mapa de navegación de la aplicación.
- Codificar cada módulo en el lenguaje seleccionado.
- Manejar un repositorio de control de versiones (GIT, SVN, Otros)
- Determinar las librerías necesarias en cada capa de la aplicación.

● Evidencia de desempeño: GA8-220501096-AA1-EV01 desarrollar *software* a partir de la integración de sus módulos componentes

● Determinar los frameworks a en cada capa de la aplicación.
● Se recomienda dividir el módulo a desarrollar en componentes que pueden ser reutilizados.
● Aplicar la codificación siguiendo las buenas prácticas de escritura de código.
● Se recomienda dividir el código fuente de cada componente que corresponde a un módulo en paquetes con nombres de fácil entendimiento.
● Aplicar los patrones de diseño de acuerdo con la arquitectura del *software* por componente.
● Se deben realizar las pruebas unitarias de cada módulo.
● Configuraciones de servidores y de bases de datos.
● Documentar los ambientes de desarrollo y pruebas.

Lineamientos generales para la entrega de la evidencia:
o **Producto para entregar:** código fuente, archivos compilados, documentos, url
o **Formato:** PDF
o **Extensión:** libre.
o Para hacer el envío de la evidencia remítase al área de la actividad correspondiente y acceda al espacio: **desarrollar *software* a partir de la integración de sus módulos componentes GA8-220501096-AA1-EV01**.

Criterios de evaluación
• Integra los módulos del *software* de acuerdo con los propósitos del sistema.

2. LISTA DE CHEQUEO

No.	VARIABLES/INDICADORES DE LOGRO	CUMPLE		Observaciones
		SI	NO	
	Sección 1			
1	Codifica los módulos de acuerdo con los requerimientos del sistema en el lenguaje seleccionado.		✓	40%
2	Aplica buenas prácticas de codificación.			15%
3	Maneja de manera adecuada el IDE de desarrollo.			15%
4	Aplica el control de versiones del código escrito.			15%
5	Aplica pruebas unitarias.			15%

Plazo Actividades:

https://www.youtube.com/watch?v=pDky0NPvzZ8&ab_channel=DamianSireDesarrollo

02 al 14 de Diciembre



G R A C I A S

Línea de atención al ciudadano: 01 8000 910270
Línea de atención al empresario: 01 8000 910682

@SENAComunica

www.sena.edu.co