UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERIA

ESCUELA DE CIENCIAS Y SISTEMAS

ANALISIS Y DISEÑO DE SISTEMAS 1

ING. JOSÉ MANUEL RUIZ

TUTOR CESAR SAZO

MANUAL TECNICO

Lunes 07 de junio 2021

Objetivo

- Alcances del sistema
- Explicación técnica
- Guía de uso
- Sección de solución de problemas

Requerimientos mínimos para la app

- Windows 10, Linux 16.04 o Linux 20.04
- Navegador Firefox, Microsoft Edge, brave, Google Chrome, opera.
- Base de datos (mysql v 08.0)
- Procesador Procesador de x86 o x64 bits de doble núcleo de 1,9 gigahercios (GHz) o más con el conjunto de instrucciones SSE2
- 2 gb de RAM
- Servidor nodejs
- Para el desarrollo de nodejs tener instalado visual studio y la carga de trabajo de desarrollo de nodejs

Servidor (Back-end)

En este apartado se desarrolla la conexión cliente servidor, donde el cliente se comunica con el api de nodejs y este se comunica con la base de datos que es mysql.

Node.js utiliza un modelo de entrada y salida sin bloqueo controlado por eventos que lo hace ligero y eficiente. Puede referirse a cualquier operación, desde leer o escribir archivos de cualquier tipo hasta hacer una solicitud http.

Instalar nodejs

- Verificacion de la versión de ***node -v*** si no está instalado usar el siguiente comando ***install nodejs***
- Verificación versión de ***npm -v*** si no está instalado usar el siguiente comando para instalar ***npm install -g npm*** este comando sirve para instalar globalmente
- Crear un directorio donde desea ubicar el proyecto de nodejs
- Instalar librerías necesarias para la creación del proyecto
    - ***Npm install express –save***
    - ***Npm install body-parser –save***
    - ***Npm install nodemon –save***
- Abre el Proyecto y continua desarrollando como se explica a continuación.

Comunicación de node.js con la base de datos (mysql)

```
or > src > database > TS database.ts > ...
import mysql from 'promise-mysql';
import config from './config';
const pool = mysql.createPool(config.database);

pool.getConnection()
    .then(connection =>{
        pool.releaseConnection(connection);
        console.log('Conectado a la base de datos');
    });
export default pool;
```

Para utilizar el código *import mysql from 'promise-mysql'* es necesario descargar e instalar algunos módulos de mysql desde nodejs, este comando ***npm install mysql*** se ejecuta en la cosola (cmd o consola visual stuido).

Crear conexión

Ahora se realiza la conexión a la base de datos, utilizando el nombre y la contraseña de la base de datos de MySql.

```
export default{
    database:{
        host: 'localhost',
        user: 'root',
        password: 'password',
        database: 'practica1'
    }
}
```

Se debe guardar estas líneas de comando en algún archivo y ejecutar el mismo ***node nameFile.js*** lo que resultado daría ***"conectado a la base de datos"***

Consulta a la base de datos

Utilizando sentencias SQL para crear, actualizar, insertar y eliminar se utilizaron los siguientes comandos desde un archivo creado en nodejs. Como se muestra en la siguiente figura.

```
/**########## READ ######## */
public async read(req:Request,res:Response){
    const games = await pool.query('SELECT * FROM empleado');
    res.json(games);

}


/**########## CREATE ######## */
public async create(req:Request,res:Response): Promise<void>{
    await pool.query('INSERT INTO empleado set ?',[req.body]);
    res.json({message: 'Empleado Guardado'});
}
```

El objetivo de conexión creado en la figura anterior tiene un método para consultar a la base de datos como read() y créate().

Cliente (Front-end)

Endpoints

Se utilizó un servicio para consumir los endpoints desde angular.

Read: muestra todos los empleados registrados en la base de datos.

```
/**READ EMPLEADOS */
read_employes(  (property) EmpleadoService.http: HttpClient
  return this.http.get(`${this.ruta}/practica1/empleados`);
}
```

CREATE**:** Registra los empleados en la base de datos.

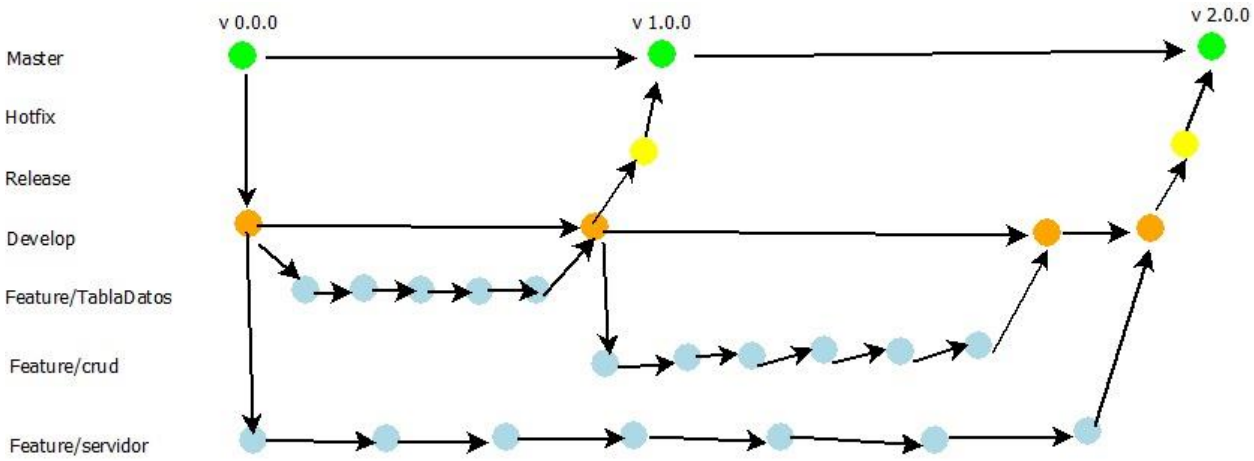
```
create_employes(empleado:Empleado){
  return this.http.post(`${this.ruta}/practica1/empleados`, empleado);
}
```

UPDATE: Actualiza los datos de un empleado registrado anteriormente a partir del id.

```
update_employes(empleado:Empleado){
  return this.http.put(`${this.ruta}/practica1/empleados/${empleado.id}`, empleado);
}
```

Delete: Elimina un empleado de la base de datos a partir del id.

Flujo de trabajo



En la gráfica se puede notar 3 versiones de la rama Master.

En la versión 0.0.0 se hizo la creación del proyecto y la aplicación.

A partir de ella se creó una rama **develop** para el desarrollo de nuevas características, de ella se generaron otras tres ramas las cuales son los feature tales como: **feature/TablaDatos, feature crud y feature/servidor.**

Cada uno de estos Feature fueron trabajados como por ejemplo el Feature/TablaDatos donde cada integrante hizo un **checkout** para agregar sus datos, posteriormente hacer un commit, finalizando dicho feature se mezcla con la rama develop.

El feature/crud se utilizó en el desarrollo del front end, donde cada integrante desarrollo lo que le coresponde del C-R-U-D.

El feature/servidor se utilizó en el desarrollo del backend donde se encarga de la conexión del servidor cliente y con la base de datos.

Al finalizar estos últimos dos features se mezclaron (respectivo merge) con la rama develop posteriormente este se mezcla con el realice y este con el master.