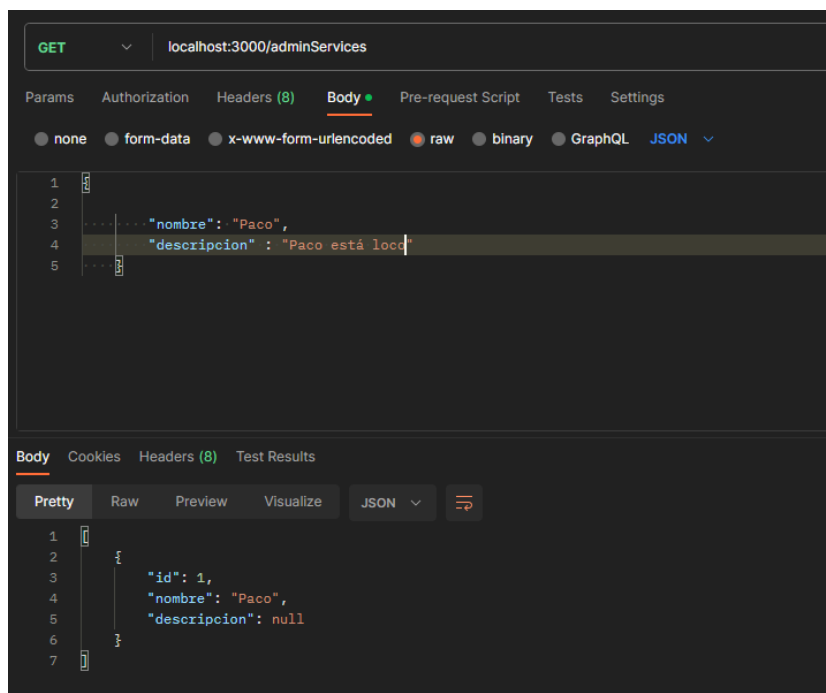


NOMBRE Y APELLIDOS:			FECHA: 02/10/2023		
DOCENTE: MANUEL MACÍAS PÉREZ			NOTA:		
(IFCD0210) DESARROLLO DE APLICACIONES CON TECNOLOGÍAS WEB			N.º CURSO: 22-35/008902		
MF:	0492	UNIDADES DE APRENDIZAJE A LAS QUE RESPONDE:	UA1	Duración:	2 hrs.
UF:	1846				
PRÁCTICA N.º:	Actividad 3				
DENOMINACIÓN: API-REST					
<p><u>DESCRIPCIÓN</u></p> <p>1.- En un supuesto practico, hay que integrar una nueva tabla a una base de datos en sql.</p> <p>Una empresa necesita integrar una nueva categoría de servicios a su base de datos. El nombre de la tabla será "AdminServices" y los campos serán los siguientes: Id, nombre y descripción.</p> <p>El alumno tendrá que realizar dicha integración en la base de datos y la configuración de la API(servidor) para poder probar su funcionamiento desde la aplicación Postman. GET, POST, PUT, DELETE</p> <p>La práctica se realizará de manera individual.</p> <p>Adjuntar pantallazo de las pruebas realizadas y adjuntar los códigos en el servidor.</p> <p><u>MEDIOS PARA SU REALIZACIÓN</u></p> <ul style="list-style-type: none"> - Equipo informático. - Aplicación Visual Code Studio instalada en el equipo. - Navegadores actualizados 					

GET



GET localhost:3000/adminServices

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

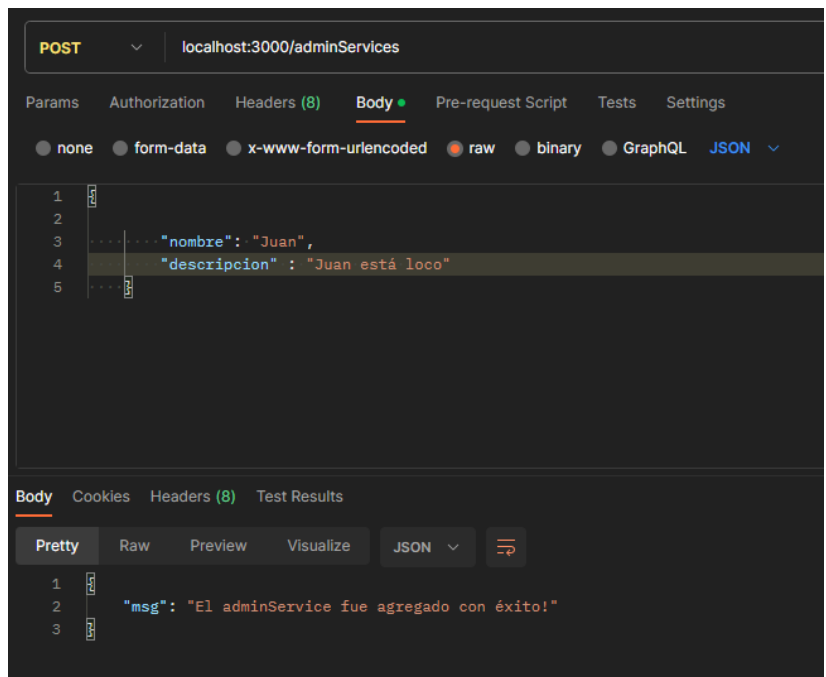
```
1 {
2
3   .... "nombre": "Paco",
4   "descripcion" : "Paco está loco"
5 }
```

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "id": 1,
4     "nombre": "Paco",
5     "descripcion": null
6   }
7 }
```

POST



POST localhost:3000/adminServices

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2
3   .... "nombre": "Juan",
4   "descripcion" : "Juan está loco"
5 }
```

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "msg": "El adminService fue agregado con éxito!"
3 }
```

PUT

PUT localhost:3000/adminServices/2

Params Authorization Headers (8) Body • Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2
3   "nombre": "Juan Melián",
4   "descripcion": "Juan está loco"
5 }
```

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "msg": "El adminService se actualizó con éxito"
3 }
```

DELETE

DELETE localhost:3000/adminServices/1

Params Authorization Headers (8) Body • Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2
3   "nombre": "Juan Melián",
4   "descripcion": "Juan está loco"
5 }
```

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "msg": "El adminService fue eliminado con éxito"
3 }
```

INTERFACES

```
export interface AdminService {
  nombre: string,
  descripcion: string,
}
```

ROUTES

```
import { Router } from "express";
import { deleteAdminService, getAdminService, getAdminServices, postAdminService,
updateAdminService } from "../controllers/AdminServiceController";

const routerAdminService = Router()
routerAdminService.get('/adminServices', getAdminServices);
routerAdminService.get('/adminServices/:id', getAdminService);
routerAdminService.delete('/adminServices/:id', deleteAdminService);
routerAdminService.post('/adminServices', postAdminService);
routerAdminService.put('/adminServices/:id', updateAdminService);

export default routerAdminService;
```

MODELO

```
import { DataTypes } from "sequelize";
import db from '../config/connectdb';

const AdminService = db.define('AdminService',{
  nombre:{
    type: DataTypes.STRING
  },
  descripcion: {
    type : DataTypes.STRING
  },
},{
  createdAt: false,
  updatedAt: false
});

export default AdminService
```

CONTROLLER

```
import { Request, Response } from "express";
import AdminService from "../models/adminServiceModel";

export const getAdminServices = async (req: Request, res: Response) =>{
  const listAdminServices = await AdminService.findAll();
  res.json(listAdminServices);
} //Método GETCURSOS

export const getAdminService = async (req: Request, res: Response) =>{
  const {id} = req.params;
  const adminService = await AdminService.findByPk(id);

  if(adminService){
    res.json(adminService)
  }
}
```

```
}
else{
  res.status(404).json()
  msg: `No existe un adminService con el id: ${id}`
}
}
} //Método GETCURSO

export const deleteAdminService = async (req: Request, res: Response) =>{
  const {id} = req.params;
  const adminService = await AdminService.findByPk(id);

  if(!adminService){
    res.status(404).json({
      msg: `No existe un adminService con el id: ${id}`
    })
  }
  else{
    await adminService.destroy();
    res.json({
      msg: 'El adminService fue eliminado con éxito'
    })
  }
} //Método DELETE

//Crear nuevo registro
export const postAdminService = async (req: Request, res: Response) =>{
  const {body} = req;
  try {
    await AdminService.create(body);
    res.json({
      msg: 'El adminService fue agregado con éxito!'
    })
  } catch (error) {
    console.log(error);
    res.json({
      msg: 'Ha ocurrido un error'
    })
  }
}

export const updateAdminService = async (req: Request, res: Response) =>{
  const {body} = req;
  const {id} = req.params;

  try {
    const adminService = await AdminService.findByPk(id);
    if(adminService){
      await adminService.update(body);
      res.json({
```

```
        msg: 'El adminService se actualizó con éxito'
    })
}
else{
    res.status(404).json()
    msg: `No existe un adminService con el id: &{id}`
}
} catch (error) {
    console.log(error);
    res.json({
        msg: 'Ha ocurrido un error'
    })
}
}
```