|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **NOMBRE Y APELLIDOS: Javier Díaz Machado** | | | **FECHA: 28/09/2023** | | |
| **DOCENTE: MANUEL MACÍAS PÉREZ** | | | **NOTA:** | | |
| **(IFCD0210) DESARROLLO DE APLICACIONES CON TECNOLOGÍAS WEB** | | | **Nº CURSO: 22-35/008902** | | |
| MF: | 0492 | UNIDADES DE APRENDIZAJE A LAS QUE RESPONDE: | UA1 | Duración: | 3 hrs. |
| UF: | 1846 |
| PRÁCTICA Nº: | Actividad 1 |
| DENOMINACIÓN: **API-REST** | | | | | |
| **DESCRIPCIÓN**  1.- Se propone la creación de un servidor para trabajar con bases de datos SQL. Adjuntar Códigos  Crear Modelos, Controller y Routes de:   * Cursos * Usuarios * Mensajes * Tareas       **MensajeModel:**  import { DataTypes } from 'sequelize';  import db from '../config/connectdb';  const Mensaje = db.define('Mensaje',{      sender: {          type: DataTypes.STRING      },      email: {          type: DataTypes.STRING      },      title: {          type: DataTypes.STRING      },      content: {          type: DataTypes.STRING      },      sendDate: {          type: DataTypes.DATE      }  }, {      createdAt: false,      updatedAt: false  });  export default Mensaje;  **MensajeRoutes:**  import { Router } from 'express'  import { deleteMensaje, getMensaje, getMensajes, postMensaje, updateMensaje } from '../controllers/MensajeController'  const routerMensajes = Router();  routerMensajes.get('/mensajes/', getMensajes);  routerMensajes.get('/mensajes/:id', getMensaje);  routerMensajes.delete('/mensajes/:id', deleteMensaje);  routerMensajes.post('/mensajes/', postMensaje);  routerMensajes.put('/mensajes/:id', updateMensaje);  export default routerMensajes;  **MensajeController:**  import {Request, Response} from 'express';  import Mensaje from '../models/MensajeModel';  export const getMensajes = async (req: Request, res: Response) =>{      const listMensajes = await Mensaje.findAll();      res.json(listMensajes);  }  export const getMensaje = async (req: Request, res: Response) => {      const {id} = req.params;      const mensaje = await Mensaje.findByPk(id);      if (mensaje) {          res.json(mensaje)      } else {          res.status(404).json({              msg: `No existe un mensaje con ese ${id}`          })      }  }  export const deleteMensaje = async (req: Request, res: Response) => {      const {id} = req.params;      const mensaje = await Mensaje.findByPk(id);      if (!mensaje) {          res.status(404).json({              msg: `No existe un mensaje con ese ${id}`          })      } else {          await mensaje.destroy();          res.json ({              msg: 'El mensaje fue eliminado con exito'          })      }  }  export const postMensaje = async (req: Request, res: Response) => {      const {body} = req;      try{          await Mensaje.create(body);          res.json({              msg: '¡El mensaje fue agregado con exito!'          })      } catch (error){          console.log(error);          res.json({              msg: 'Ha ocurrido un error'          })      }  }  export const updateMensaje = async (req: Request, res: Response) => {      const {body} = req;      const {id} = req.params;      try{          const mensaje = await Mensaje.findByPk(id);          if(mensaje){              await mensaje.update(body);              res.json({                  msg: 'El mensaje fue actualizado con exito'              })          } else {              res.status(404).json({                  msg: `No existe un mensaje con ese ${id}`              })          }      } catch (error){          console.log(error);          res.json({              msg: 'Ha ocurrido un error'          })      }  }  La práctica se realizará de manera individual.  **MEDIOS PARA SU REALIZACIÓN**   * Equipo informático. * Aplicación Visual Code Studio instalada en el equipo. * Navegadores actualizados | | | | | |