Documentación de Examen

# Introducción

El siguiente es la documentación examen práctico que trata sobre una aplicación web que permite gestionar una base de datos de una empresa y mostrar/insertar estos datos a través de dicha aplicación web. La base de datos es de MySQL, la cual es consultada por el frontend de la app web que está escrita en C# que utiliza el framework ASP.NET Core y utiliza la arquitectura Model-View-Controller (MVC). Este frontend se conecta a la base de datos a través de una REST API escrita en Node.js.

# Estructura del proyecto

**Componentes principales:**

* API en Node.js:
  + Maneja la conexión del frontend a la base de datos MySQL y proporciona los endpoints necesarios para las consultas
* App Web en ASP.NET Core (C#)
  + Es el frontend que interactúa con la API para mandar las consultas a la base de datos. Este contiene los botones necesarios para las consultas, así como los campos tipo formulario para agregar nuevos registros a las tablas, así como los componentes necesarios para la visualización de los resultados de estas consulta.
* Base de datos MySQL
  + Está llamada “empresa” y contiene dos tablas:
    - departamento:
      * código\_departamento – Integer, Primary Key
      * nombre\_depto – Varchar(75)
      * presupuesto – Decimal
    - empleado:
      * dpi – Integer, Primary Key
      * nombre – Varchar(100)
      * apellidos – Varchar(125)
      * cod\_depto – Integer, Foreign Key que referencia a departamento.codigo\_departamento

## Requisitos de sistema

**Software requerido:**

* MySQL versión 8.0 o superior
* Node.js versión 14 o superior
* .NET SDK versión 6
* Dependencias de Node:
  + mysql
  + cors
  + bodyparser
  + express
* Dependencias C#:
  + WebApp

**Instalación:**

* Para crear la base de datos llamada “empresa”, con sus tablas (“empleado” y “departamento”), se deben ejecutar los queries que están en el archivo “queries.sql”.
* Para configurar la REST API de Node se debe:
  + Correr el comando “npm install” en el directorio base del proyecto de Node.js para instalar las dependencias.
  + Editar la configuración de la base de datos en el archivo app.js
  + Correr el comando “node app.js” para iniciar el servidor.

## Funcionalidades

**Visualización de los datos:**

Visualiza por medio de una tabla las consultas que se hacen a través de los botones de la interfaz gráfica.

**Gestión de departamentos:**

Permite registrar datos a la tabla de departamento registrando su código, nombre y el presupuesto con el que este departamento cuenta.

**Gestión de empleados:**

Permite registrar datos a la tabla de empelados registrando su DPI, nombre, apellidos y el código del departamento al que pertenece.

## Endpoints disponibles

|  |  |  |
| --- | --- | --- |
| Metodo | Endpoint | Descripcion |
| GET | /consulta1 | Ejecuta el query de la consulta 1 |
| GET | /consulta2 | Ejecuta el query de la consulta 2 |
| GET | /consulta3 | Ejecuta el query de la consulta 3 |
| GET | /consulta4 | Ejecuta el query de la consulta 4 |
| GET | /consulta5 | Ejecuta el query de la consulta 5 |
| GET | /consulta6 | Ejecuta el query de la consulta 6 |
| GET | /consulta7 | Ejecuta el query de la consulta 7 |
| GET | /consulta8 | Ejecuta el query de la consulta 8 |
| GET | /consulta9 | Ejecuta el query de la consulta 9 |
| GET | /consulta10 | Ejecuta el query de la consulta 10 |
| GET | /consulta11 | Ejecuta el query de la consulta 11 |
| GET | /consulta12 | Ejecuta el query de la consulta 12 |
| GET | /consulta13 | Ejecuta el query de la consulta 13 |
| GET | /consulta14 | Ejecuta el query de la consulta 14 |
| POST | /departamento | Agrega datos a la tabla departamento para la consulta 15 |
| POST | /empleado | Agrega datos a la tabla empleado para la consulta 15 |