MANUAL TÉCNICO

Descripción

Aplicación solución para el proyecto 1 del curso “Sistemas de Bases de Datos 1”. El problema para resolver estaba ligado a las votaciones nacionales en Guatemala. Se contaba con una serie de archivos csv con toda la información relacionada. La tarea era cargar esa información a un motor de base de datos y realizar una serie de consultas con esos datos.

La aplicación esta hecha en express usando como DBMS a MySQL. Es una API que contiene once endpoints para cada una de las consultas que pedían en los requerimientos y tres endpoints para crear el modelo, bórralo y llenarlo.

Requisitos

- Node 18.17.1
- MySQL
- Un tester para probar la API (Postman, Thunder o un navegador)
- Sistema operativo libre

Instalación

1. Nos ubicamos en la carpeta donde se encuentre ubicada la aplicación.
2. Abrimos una terminal en la carpeta “Aplicación” donde se encuentra el package.json
3. Ingresamos el comando “npm install”.

Configuración

Para que la aplicación funcione, debemos de crear un archivo “.env” en la carpeta donde se encuentra alojado el archivo “app.js”. En ese archivo, debemos añadir las siguientes variables de entorno con la información de nuestro usuario y la base de datos MySQL.

1. DB_HOST: Usualmente localhost.
2. DB_PORT: Puerto que hayamos configurado en la instalación de MySQL. Usualmente, el 3306.
3. DB_USER: Nombre de usuario. Puede ser root.
4. DB_PASSWORD: Contraseña asignada al usuario.

5. DB_DATABASE: Nombre de la base de datos en la que vayamos a trabajar. Debe de ser creada previamente a la ejecución.

Uso

- Ejecución:

1. Nos dirigimos a la carpeta donde esta el proyecto y abrimos la carpeta Aplciacion.
2. Abrimos una terminal en la carpeta app
3. Ingresamos el comando “node app.js”
4. Abrimos cualquier navegador o tester e ingresamos la ruta “localhost:3000/”.

- Uso:

Todas las peticiones de la API son de tipo GET. Se puede consumir la api incluso desde un navegador ingresando “localhost:3000/” seguido de la ruta a la que se quiere acceder. La API devolverá un archivo JSON con los resultados de la tarea.

Rutas de la API

1. “/crearmodelo” : La aplicación cuenta con el script dentro de sus archivos. Este endpoint ejecuta el script para crear las tablas dentro de la base de datos local (o depende de la configuración de la conexión.

2. “/eliminarmodelo” : Ejecuta el script que elimina las tablas creadas en la base de datos.

3. “/cargartabtemp” : Carga masiva de datos en las tablas del modelo. La información se encuentra dentro de los archivos de la aplicación en la carpeta “data”. Se hace uso de tablas temporales para distribuir los datos en todo el modelo.

4. “/consulta1” : Muestra los candidatos a presidente / vicepresidente por partido.

5. “/consulta2” : Muestra el numero de candidatos a diputado por partido.

6. “/consulta3” : Muestra los candidatos a alcalde por partido.

7. “/consulta4” : Muestra el número de candidatos por partido.

8. “/consulta5” : Muestra el número de votos por departamento.

9. “/consulta6” : Muestrael número de votos nulos.

10. “/consulta7” : Muestra el top 10 de edades que votaron.

11. “/consulta8” : Muestra el top 10 de candidatos a presidente / vicepresidente que recibieron más votos.

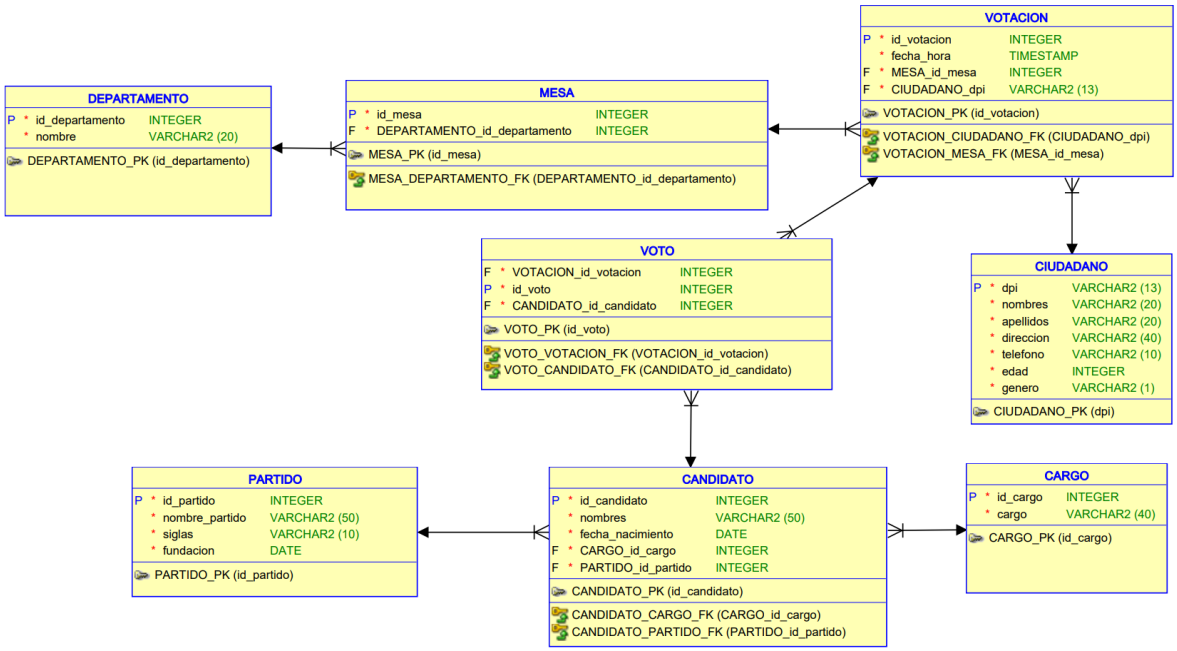
12. “/consulta9” : Muestra el top 5 de mesas más frecuentadas.

13. “/consulta10” : Muestra el top 5 de horas en las que más fueron a votar.

14. “/consulta11” : Muestra la cantidad de votos por sexo.

Base de Datos

- Modelo Físico



- Cada una de las tablas del modelo son una idéntica representación de las entidades descritas en los archivos con los datos a trabajar. Para hacer más eficiente las referencias a cada uno de sus campos, se opto por cambiar cosas como id del cargo o id de la mesa a poner directamente mesa o cargo como nombre para cada uno de los campos.

- Para el nombre de las tablas, siguiendo el esquema de Barker, se colocaron el nombre de las entidades en mayúsculas y cada una de las entidades en singular.

- Para los campos, se aprovecho que todos los identificadores fueron números y se colocaron la mayoría con enteros (INTEGER). La excepción a esto es la llave primaria de ciudadano que es una cadena de exactamente trece caracteres. Se trato de contar y determinar un estimado de la media de caracteres que usaban los datos con texto para evitar asignar espacio de más. Por lo general, se dejó como el múltiplo de cinco más cercano a la media estimada.

- La única tabla que se partió en dos fue para el archivo “votaciones.csv”. Esta dio origen a las tablas votos y votación. Esto se debe a que la información del votante (fecha, dpi, número de voto) se repetían en casi todas las filas, cambiando el candidato al que votaron nada más. Se mando la información del voto a la tabla votación y solo se dejo en voto el identificador del mismo y el candidato por el que votaron. A esta tabla se le asigno una nueva llave primaria, ya que existe la posibilidad de votar nulo para todos los candidatos. Si se tomaba el id del candidato y el candidato como una llave compuesta, lo más seguro es que se generaría conflicto al tener filas con la misma combinación de valores como llave primaria.

- Script:

```sql
-- Tabla Cargo
CREATE TABLE IF NOT EXISTS cargo (
    id INT PRIMARY KEY,
    cargo VARCHAR(40) NOT NULL
);

-- Tabla Partido
CREATE TABLE IF NOT EXISTS partido (
  id INT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  siglas VARCHAR(10) NOT NULL,
  fundacion DATE NOT NULL
);

-- Tabla Ciudadanos
CREATE TABLE IF NOT EXISTS ciudadano (
  dpi VARCHAR(13) PRIMARY KEY,
  nombres VARCHAR(20) NOT NULL,
  apellidos VARCHAR(20) NOT NULL,
  direccion VARCHAR(40) NOT NULL,
  telefono VARCHAR(10) NOT NULL,
  edad INT NOT NULL,
  genero VARCHAR(1) NOT NULL
);

-- Tabla Departamentos
CREATE TABLE IF NOT EXISTS departamento (
    id INT PRIMARY KEY,
    nombre VARCHAR(20) NOT NULL
);

-- Tabla Mesas
CREATE TABLE IF NOT EXISTS mesa (
    id INT PRIMARY KEY,
    departamento INT NOT NULL,
    FOREIGN KEY (departamento) REFERENCES departamento(id)
);

-- Tabla Candidatos
CREATE TABLE IF NOT EXISTS candidato (
  id INT PRIMARY KEY,
  nombres VARCHAR(50)  NOT NULL,
  fecha_nacimiento DATE NOT NULL,
  partido INT NOT NULL,
```

```
    cargo INT NOT NULL,
    FOREIGN KEY (partido) REFERENCES partido(id),
    FOREIGN KEY (cargo) REFERENCES cargo(id)
);

-- Tabla Votacioens
CREATE TABLE IF NOT EXISTS votacion (
    id INT PRIMARY KEY,
    fecha_hora TIMESTAMP NOT NULL,
    dpi VARCHAR(13) NOT NULL,
    mesa INT NOT NULL,
    FOREIGN KEY (dpi) REFERENCES ciudadano(dpi),
    FOREIGN KEY (mesa) REFERENCES mesa(id)
);

-- Tabla Voto
CREATE TABLE IF NOT EXISTS voto (
    id INT AUTO_INCREMENT PRIMARY KEY,
    voto INT NOT NULL,
    candidato INT NOT NULL,
    FOREIGN KEY (voto) REFERENCES votacion(id),
    FOREIGN KEY (candidato) REFERENCES candidato(id)
);

```