Universidad de San Carlos de Guatemala
Facultad de ingeniería
Escuela de Ciencias y Sistemas
Sistemas de Bases de Datos 1
Segundo semestre 2023

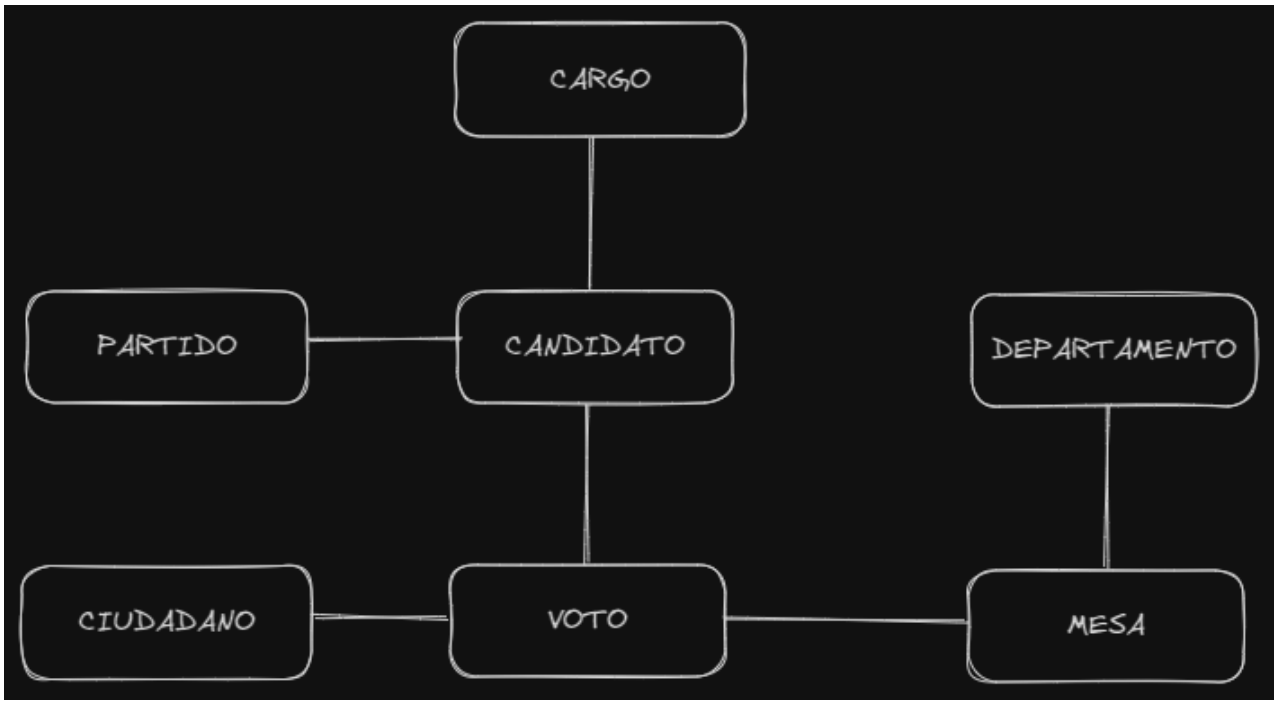
Manual Técnico

Jhonatan Josué Tzunun Yax

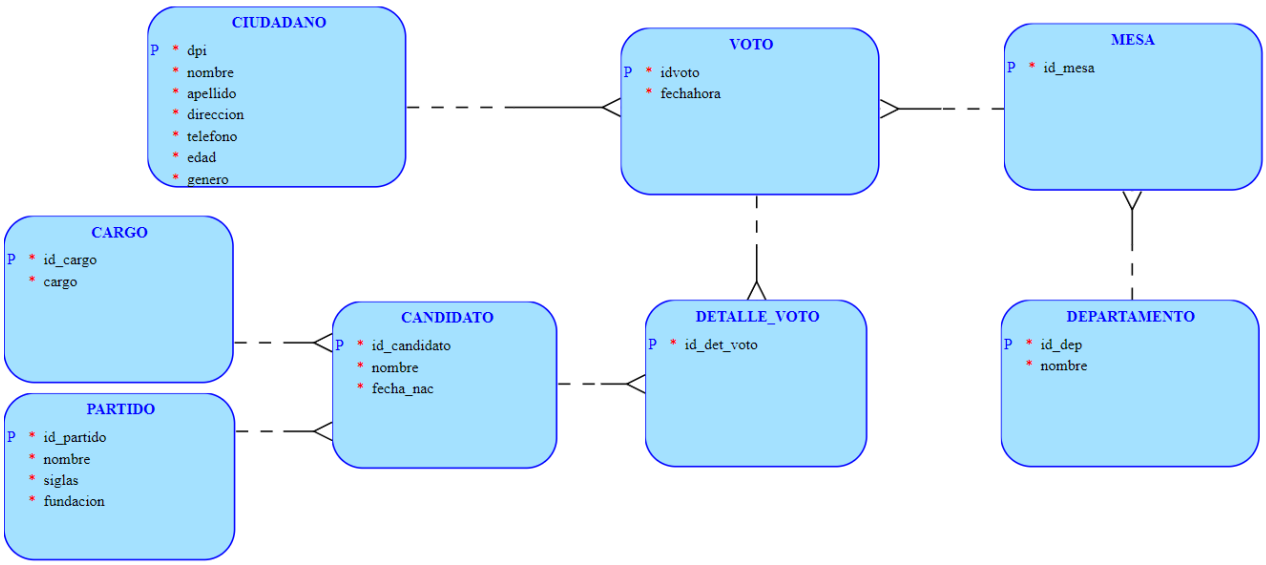
Introducción

En el siguiente manual se describe a detalle los endpoint y funciones, así como el contenido de cada una y porque fue necesaria la creación de esta, para que se usó cada función y endpoint, explicando de forma breve la lógica aplicada en este proyecto.

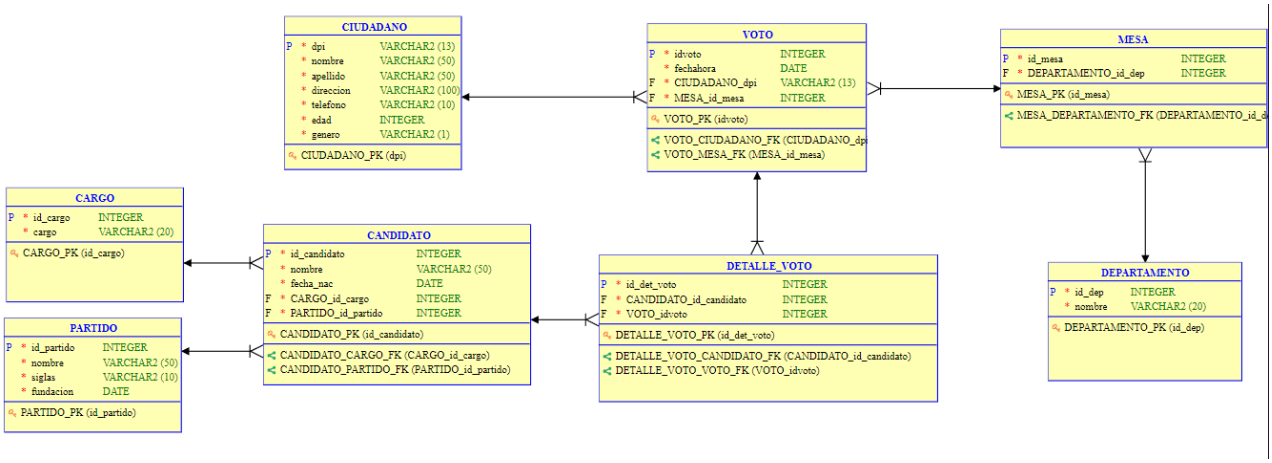
Modelos Conceptual



Modelo lógico



Modelo físico (ER)



Creación de modelo

```sql
-- CREAR BASE DE DATOS

CREATE SCHEMA IF NOT EXISTS bd1_p1;

--  TABLA PARTIDO

CREATE TABLE IF NOT EXISTS bd1_p1.PARTIDO (
  id INTEGER NOT NULL,
  nombre VARCHAR(50) NOT NULL,
  siglas VARCHAR(10) NOT NULL,
  fundacion DATE NOT NULL,
  PRIMARY KEY (id)
);

-- TABLA CARGO

CREATE TABLE IF NOT EXISTS bd1_p1.CARGO (
  id INTEGER NOT NULL,
  cargo VARCHAR(50) NOT NULL,
  PRIMARY KEY (id)
);

-- TABLA CANDIDATO

CREATE TABLE IF NOT EXISTS bd1_p1.CANDIDATO (
  id INTEGER NOT NULL,
  nombre VARCHAR(50) NOT NULL,
  fecha_nac DATE NOT NULL,
  id_partido INTEGER NOT NULL,
  id_cargo INTEGER NOT NULL,
```

```
  PRIMARY KEY (id),
  FOREIGN KEY (id_partido) REFERENCES bd1_p1.PARTIDO (id),
  FOREIGN KEY (id_cargo) REFERENCES bd1_p1.CARGO (id)
);


-- TABLA CIUDADANO

CREATE TABLE IF NOT EXISTS bd1_p1.CIUDADANO (
  dpi VARCHAR(13) NOT NULL,
  nombre VARCHAR(50) NOT NULL,
  apellido VARCHAR(50) NOT NULL,
  direccion VARCHAR(100) NOT NULL,
  telefono VARCHAR(10) NOT NULL,
  edad INTEGER NOT NULL,
  genero VARCHAR(1) NOT NULL,
  PRIMARY KEY (dpi)
);

-- TABLA DEPARTAMENTO

CREATE TABLE IF NOT EXISTS bd1_p1.DEPARTAMENTO (
  id INTEGER NOT NULL AUTO_INCREMENT,
  nombre VARCHAR(20) NOT NULL,
  PRIMARY KEY (id)
);

-- TABLA MESA

CREATE TABLE IF NOT EXISTS bd1_p1.MESA (
  id INTEGER NOT NULL AUTO_INCREMENT,
  id_dep INTEGER NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (id_dep) REFERENCES bd1_p1.DEPARTAMENTO (id)
);

-- TABLA VOTO

CREATE TABLE IF NOT EXISTS bd1_p1.VOTO (
  id INTEGER NOT NULL AUTO_INCREMENT,
  dpi VARCHAR(13) NOT NULL,
  id_mesa INTEGER NOT NULL,
  fechahora DATETIME NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (dpi) REFERENCES bd1_p1.CIUDADANO (dpi),
  FOREIGN KEY (id_mesa) REFERENCES bd1_p1.MESA (id)
```

```
);

-- TABLA DETALLE VOTO

CREATE TABLE IF NOT EXISTS bd1_p1.DETALLE_VOTO (
  id_voto INTEGER NOT NULL,
  id_candidato INTEGER NOT NULL,
  FOREIGN KEY (id_voto) REFERENCES bd1_p1.VOTO (id),
  FOREIGN KEY (id_candidato) REFERENCES bd1_p1.CANDIDATO (id)
);
```

Se crea la base de datos si no existe, al igual que las tablas. Ningún dato puede ser nulo. Además, se añaden las llaves primarias y foráneas.

createmodel.js

El archivo createmodel utilizara el anterior script, antes de ejecutar la consulta se eliminan los comentarios del script con una expresión regular, luego se hace un Split con el símbolo “;” esto para separar cada consulta sql y se guarda en una variable.

```
for (let i=0; i<sqlCmds.length; i++) {
        let sql = sqlCmds[i];
        if (sql.length === 0) {
           continue;
        }
        await db.query(sql, []);
     }
```

Se recorre la variable y se ejecuta cada comando sql por separado. Si todo sale bien retorna un mensaje de éxito, de lo contrario muestra el error.

deletemodel.js

Funciona igual que créatemodel, pero ejecuta el siguiente script:

```
-- BORRAR BASE DE DATOS
DROP DATABASE IF EXISTS bd1_p1;
-- BORRAR TABLA PARTIDO
DROP TABLE IF EXISTS bd1_p1.PARTIDO;
-- BORRAR TABLA CARGO
DROP TABLE IF EXISTS bd1_p1.CARGO;
-- BORRAR TABLA CANDIDATO
DROP TABLE IF EXISTS bd1_p1.CANDIDATO;
-- BORRAR TABLA CIUDADANO
DROP TABLE IF EXISTS bd1_p1.CIUDADANO;
-- BORRAR TABLA DEPARTAMENTO
DROP TABLE IF EXISTS bd1_p1.DEPARTAMENTO;
-- BORRAR TABLA MESA
DROP TABLE IF EXISTS bd1_p1.MESA;
```

bulk_load_tmp.js

Funciones

- dateFormat: convierte la cadena enviada al formato YYY-MM-DD y la retorna.
- notacionCientifica: convierte la cadena o numero enviado a número entero y lo retorna.
- Créate_tmp_table: Primero crea las tablas temporales luego las llena, después de ese proceso añade el contenido de las tablas temporales a las tablas normales.

```
// Carga de datos de csv a tabla temporal
     let data = fs.readFileSync(filepath, 'utf8');
     let lines = data.split('\n');
     let nulo;
     //Recorrer el arreglo
     for (let i = 1; i < lines.length; i++) {
        const fields = lines[i].split(/,(?=(?:(?:[^"]*"){2})*[^"]*$)/);
        const fields_sanitized = fields.map(field => field.replace(/['"]+/g,
'').trim());
        // Insertar los datos en la tabla temporal
        if (fields_sanitized.length > 1) {
           await db.queryWithoutClose(connection, `INSERT INTO bd1_p1.PARTIDO_TMP
(id, nombre, siglas, fundacion) VALUES (?, ?, ?, ?)`, [fields_sanitized[0],
fields_sanitized[1], fields_sanitized[2], dateFormat(fields_sanitized[3])]);
        }
     }
     await db.queryWithoutClose(connection, `INSERT INTO bd1_p1.PARTIDO (id,
nombre, siglas, fundacion) SELECT id, nombre, siglas, fundacion FROM
bd1_p1.PARTIDO_TMP`, []);
```

Primero abre y lee el archivo y lo separa por saltos de línea y recorre ese vector. En este punto inserta los datos en la tabla temporal, al terminar ingresa esos datos a la tabla normal