MANUAL TECNICO
PROYECTO #1

Luis Manuel Chay Marroquín

Huehuetenango, 17 de septiembre de 2023

DATOS TECNICOS

LENGUAJE UTILIZADO

Python

IDE UTILIZADO

Visual Studio Code - Insiders

SISTEMA OPERATIVO

Windows

HERRRAMIENTAS DE ADMINISTRACION DE BASE DE DATOS

Se uso la base de datos MySQL junto a MySQL WorkBench para llevar un control de la base de datos local

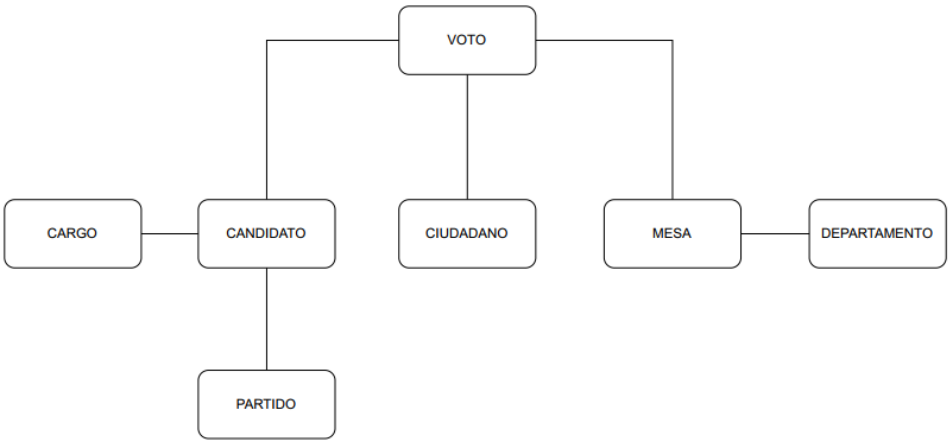
VISUALIZACION DE API

Postman

Lógica del programa

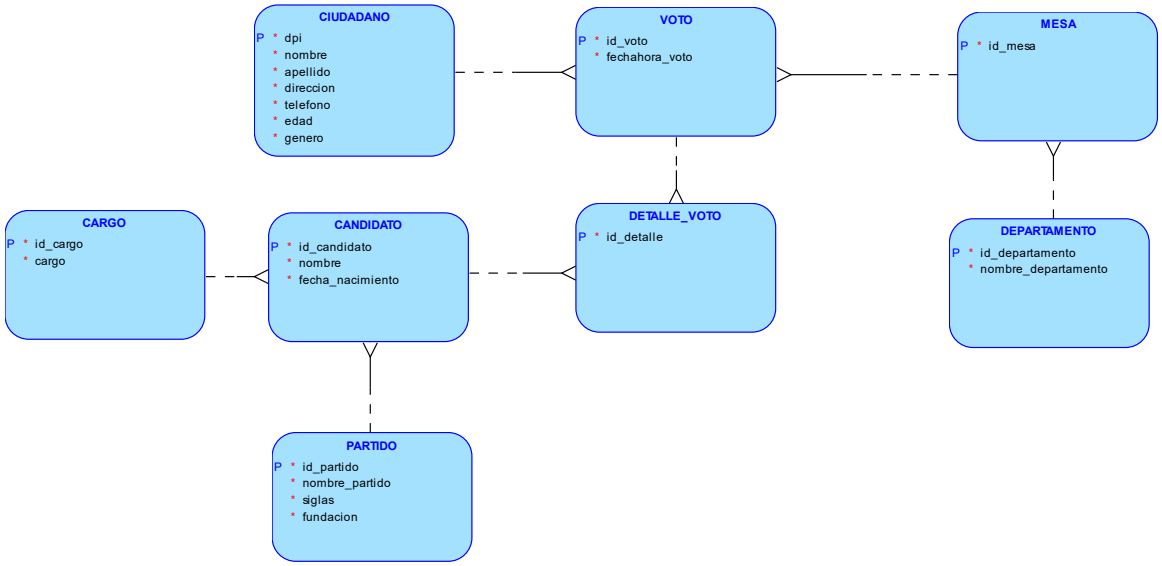
Modelos de la base de datos

Modelo Conceptual



En este modelo hacemos una representación básica y sin muchos detalles acerca de los datos sobre el modelo a usar en el proyecto, definimos las tablas que mejor abstraen la información del problema dado y establecemos relaciones

Modelo Lógico



En este modelo hacemos una representación acerca de los atributos que contara cada una de las tablas y establecemos las llaves primarias de cada una de ellas, así como la relación entre tablas, teniendo que todas las relaciones cumplen con una relación uno a todos,

Es una relación uno a todos de cargo y partido hacia candidato ya que solo pueden pertenecer a un cargo y partido a la vez, pero pueden ser n cantidad de candidatos

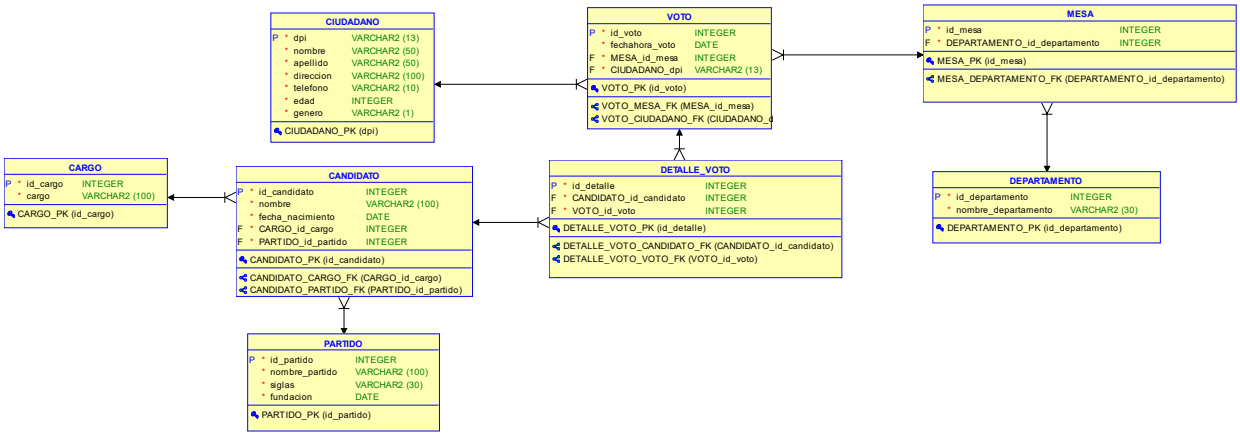
Es una relación uno a todos de departamento hacia mesa ya que solo pueden pertenecer una mesa a un departamento sin repetirse en ninguno otro

Es una relación uno a todos de mesa hacia voto ya que solo se puede registrar un voto desde una sola mesa, no existe el mismo voto de dos mesas diferentes

Es una relación uno a todos de ciudadano hacia voto ya que un ciudadano puede votar una única vez por cada una de las votaciones existentes (binomio presidencial, alcaldías, diputados, etc.)

Es una relación uno a todos de candidato y voto hacia detalle_voto ya que solo un candidato puede ser votado, valga la redundancia, por medio de un voto valido

Modelo Físico



En este modelo hacemos una representación acerca de los atributos que contara cada una de las tablas y establecemos las llaves primarias de cada una de ellas, así como la relación entre tablas.

Las tablas cargo y partido cumplen una relación de uno a todos hacia la tabla candidato, permitiendo que sus llaves primarias sean atributos de la tabla candidato, así como es solicitada la información por el TSE, nos interesa que sean llaves foráneas de la tabla candidato porque con eso podremos relacionar mejor la información y devolverla con mayor detalle

La tabla departamento cumplen una relación de uno a todos hacia la tabla mesa, permitiendo que sus llaves primarias sean atributos de la tabla mesa, así como es solicitada la información por el TSE, nos interesa que sean llaves foráneas de la tabla mesa porque con eso podremos relacionar mejor la información y devolverla con mayor detalle

La tabla mesa cumplen una relación de uno a todos hacia la tabla voto, permitiendo que sus llaves primarias sean atributos de la tabla voto, así como es solicitada la información por el TSE, nos interesa que sean llaves foráneas de la tabla voto porque con eso podremos relacionar mejor la información y devolverla con mayor detalle

La tabla ciudadano cumplen una relación de uno a todos hacia la tabla voto, permitiendo que sus llaves primarias sean atributos de la tabla voto, así como es solicitada la información por el TSE, nos interesa que sean llaves foráneas de la tabla voto porque con eso podremos relacionar mejor la información y devolverla con mayor detalle

Las tablas candidato y voto cumplen una relación de uno a todos hacia la tabla detalle_voto, permitiendo que sus llaves primarias sean atributos de la tabla candidato, así como es solicitada la información por el TSE, nos interesa que sean llaves foráneas de la tabla candidato porque con eso podremos relacionar mejor la información y devolverla con mayor detalle

La existencia de una tabla detalle_voto es para que no haya una redundancia de datos y se puedan filtrar mejor para cada una de las consultas

Código de la aplicación

Creacion API

```python
app = Flask(__name__)
CORS(app)
conexion = MySQL(app)

#Rutas iniciales

@app.route('/', methods=['GET'])
def rutaInicial():
    return("Ruta inicial")
```

```python
if __name__ == "__main__":
    app.config.from_object(config['development'])
    app.run(host="0.0.0.0", port=4000, debug=True)
```

Endpoint crearmodelo

En este endpoint creamos las tablas del modelo, cabe recalcar que la base de datos ha sido creada anteriormente

```python
#TABLAS DEL MODELO
cursor = conexion.connection.cursor()
cursor.execute("""
CREATE TABLE IF NOT EXISTS cargo (
id_cargo INT NOT NULL PRIMARY KEY,
cargo VARCHAR(100) NOT NULL
);
""")
cursor.execute("""
CREATE TABLE IF NOT EXISTS departamento (
id_departamento INT NOT NULL  PRIMARY KEY,
nombre_departamento VARCHAR(30) NOT NULL
);
""")
cursor.execute("""
CREATE TABLE IF NOT EXISTS mesa (
id_mesa INT NOT NULL  PRIMARY KEY,
id_departamento INT NOT NULL,
FOREIGN KEY (id_departamento)  REFERENCES departamento(id_departamento)
);
""")
cursor.execute("""
CREATE TABLE IF NOT EXISTS ciudadano (
dpi VARCHAR(13) NOT NULL PRIMARY KEY,
nombre VARCHAR(50) NOT NULL,
apellido VARCHAR(50) NOT NULL,
direccion VARCHAR(100) NOT NULL,
telefono VARCHAR(10) NOT NULL,
edad INT NOT NULL,
genero VARCHAR(1) NOT NULL
);
""")
cursor.execute("""
CREATE TABLE IF NOT EXISTS partido (
id_partido INT NOT NULL  PRIMARY KEY,
nombre_partido VARCHAR(100) NOT NULL,
siglas VARCHAR(30) NOT NULL,
fundacion DATE NOT NULL
);
""")
```

```
    cursor.execute("""
    CREATE TABLE IF NOT EXISTS candidato (
    id_candidato INT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    fecha_nacimiento DATE NOT NULL,
    id_cargo INT NOT NULL,
    id_partido INT NOT NULL,
    FOREIGN KEY (id_cargo) REFERENCES cargo(id_cargo),
    FOREIGN KEY (id_partido) REFERENCES partido(id_partido)
    );
    """)

    cursor.execute("""
    CREATE TABLE IF NOT EXISTS voto (
    id_voto INT NOT NULL  PRIMARY KEY,
    id_candidato INT NOT NULL,
    dpi VARCHAR(13) NOT NULL,
    id_mesa INT NOT NULL,
    fechahora_voto DATETIME NOT NULL,
    FOREIGN KEY (dpi) REFERENCES ciudadano(dpi),
    FOREIGN KEY (id_mesa) REFERENCES mesa(id_mesa)
    );
    """)
    cursor.execute("""
    CREATE TABLE IF NOT EXISTS detalle_voto (
    id_detalle INT AUTO_INCREMENT NOT NULL  PRIMARY KEY,
    id_voto INT NOT NULL,
    id_candidato INT NOT NULL,
    FOREIGN KEY (id_voto)  REFERENCES voto(id_voto),
    FOREIGN KEY (id_candidato) REFERENCES candidato(id_candidato)
    );
    """)
    conexion.connection.commit()
    return("Modelo creado")
except Exception as e:
        print("Error al insertar:", str(e))
        return("Error al insertar")
```

Endpoint eliminarmodelo

En este endpoint eliminamos las tablas del modelo

```python
@app.route('/eliminarmodelo', methods=['GET'])
def eliminarmodelo():
    try:
        cursor = conexion.connection.cursor()
        cursor.execute("DROP TABLE IF EXISTS cargo, ciudadano, candidato, partido, voto, detalle_voto, mesa, departamento;")
        return("Modelo eliminado")
    except Exception as e:
            print("Error al insertar:", str(e))
            return("Error al insertar")
```

Endpoint cargartabtemp

En este endpoint creamos las tablas temporales, luego la llenamos con la información proporcionada por el TSE en archivos csv y realizamos el llenado de las tablas del modelo

```python
try:
    cursor.execute("""
    CREATE TEMPORARY TABLE cargotemp (
    id_cargo INT NOT NULL PRIMARY KEY,
    cargo VARCHAR(100) NOT NULL
    );
    """)
    cursor.execute("""
    CREATE TEMPORARY TABLE departamentotemp (
    id_departamento INT NOT NULL  PRIMARY KEY,
    nombre_departamento VARCHAR(30) NOT NULL
    );
    """)
    cursor.execute("""
    CREATE TEMPORARY TABLE mesatemp (
    id_mesa INT NOT NULL  PRIMARY KEY,
    id_departamento INT NOT NULL
    );
    """)
    cursor.execute("""
    CREATE TEMPORARY TABLE  ciudadanotemp (
    dpi VARCHAR(13) NOT NULL PRIMARY KEY,
    nombre VARCHAR(50) NOT NULL,
    apellido VARCHAR(50) NOT NULL,
    direccion VARCHAR(100) NOT NULL,
    telefono VARCHAR(10) NOT NULL,
    edad INT NOT NULL,
    genero VARCHAR(1) NOT NULL
    );
    """)
    cursor.execute("""
    CREATE TEMPORARY TABLE partidotemp (
    id_partido INT NOT NULL  PRIMARY KEY,
    nombre_partido VARCHAR(100) NOT NULL,
    siglas VARCHAR(30) NOT NULL,
    fundacion DATE NOT NULL
    );
    """)
    cursor.execute("""
    CREATE TEMPORARY TABLE candidatotemp (
    id_candidato INT NOT NULL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    fecha_nacimiento DATE NOT NULL,
    id_cargo INT NOT NULL,
    id_partido INT NOT NULL
    );
    """)
```

```
    cursor.execute("""
    CREATE TEMPORARY TABLE vototemp (
    id_voto INT NOT NULL,
    id_candidato INT NOT NULL,
    dpi VARCHAR(13) NOT NULL,
    id_mesa INT NOT NULL,
    fechahora_voto DATETIME NOT NULL
    );
    """)

    depfile = pd.read_csv('csv/departamentos.csv',encoding="utf8")
    for index, row in depfile.iterrows():
        cursor.execute("INSERT INTO departamentotemp (id_departamento, nombre_departamento) VALUES (%s,%s);", (int(row[0]), row[1]))

    cursor.execute("INSERT INTO departamento (id_departamento, nombre_departamento) SELECT id_departamento, nombre_departamento FROM departamentotemp;")
    conexion.connection.commit()

    cargosfile = pd.read_csv('csv/cargos.csv',encoding="utf8")
    for index, row in cargosfile.iterrows():
        cursor.execute("INSERT INTO cargotemp (id_cargo, cargo) VALUES (%s,%s);", (int(row[0]), row[1]))

    cursor.execute("INSERT INTO cargo (id_cargo, cargo) SELECT id_cargo, cargo FROM cargotemp;")
    conexion.connection.commit()


    partidosfile = pd.read_csv('csv/partidos.csv',encoding="utf8")
    for index, row in partidosfile.iterrows():
        cursor.execute("INSERT INTO partidotemp (id_partido, nombre_partido, siglas, fundacion) VALUES (%s,%s,%s,%s);", (int(row[0]), row[1], row[2], datetime.datetime.strptime(row[3], "%d/%m/%Y").date()))

    cursor.execute("INSERT INTO partido (id_partido, nombre_partido, siglas, fundacion) SELECT id_partido, nombre_partido, siglas, fundacion FROM partidotemp;")
    conexion.connection.commit()

    mesasfile = pd.read_csv('csv/mesas.csv',encoding="utf8")
    for index, row in mesasfile.iterrows():
        cursor.execute("INSERT INTO mesatemp (id_mesa, id_departamento) VALUES (%s,%s);", (int(row[0]), row[1]))

    cursor.execute("INSERT INTO mesa (id_mesa, id_departamento) SELECT id_mesa, id_departamento FROM mesatemp;")
    conexion.connection.commit()

    ciudafile = pd.read_csv('csv/ciudadanos.csv',encoding="utf8")
    for index, row in ciudafile.iterrows():
        cursor.execute("INSERT INTO ciudadanotemp (dpi, nombre, apellido, direccion, telefono, edad, genero) VALUES (%s,%s,%s,%s,%s,%s,%s);", (row[0], row[1], row[2], row[3], row[4], int(row[5]), row[6]))

    cursor.execute("INSERT INTO ciudadano (dpi, nombre, apellido, direccion, telefono, edad, genero) SELECT dpi, nombre, apellido, direccion, telefono, edad, genero FROM ciudadanotemp;")
    conexion.connection.commit()
```

```
        candifile = pd.read_csv('csv/candidatos.csv',encoding="utf8")
        for index, row in candifile.iterrows():
            cursor.execute("INSERT INTO candidatotemp (id_candidato, nombre, fecha_nacimiento, id_partido, id_cargo) VALUES (%s,%s,%s,%s,%s);", (int(row[0]), row[1], datetime.datetime.strptime(row[2], "%d/%m/%Y").date

        cursor.execute("INSERT INTO candidato (id_candidato, nombre, fecha_nacimiento, id_cargo, id_partido) SELECT id_candidato, nombre, fecha_nacimiento, id_cargo, id_partido FROM candidatotemp;")
        conexion.connection.commit()




        votafile = pd.read_csv('csv/votaciones.csv',encoding="utf8")
        for index, row in votafile.iterrows():
            cursor.execute("INSERT INTO vototemp (id_voto, id_candidato, dpi, id_mesa, fechahora_voto) VALUES (%s,%s,%s,%s,%s);", (int(row[0]), int(row[1]), row[2], int(row[3]), datetime.datetime.strptime(row[4],  "%

        cursor.execute("SELECT * FROM vototemp;")
        resultadoquery = cursor.fetchall()
        for x in resultadoquery:
            cursor.execute("SELECT * FROM voto WHERE id_voto = %s;", (int(x[0]),))
            aux = cursor.fetchall()
            if len(aux) == 0:
                cursor.execute("INSERT INTO voto (id_voto, id_candidato, dpi, id_mesa, fechahora_voto) VALUES (%s,%s,%s,%s,%s);", (int(x[0]), int(x[1]), x[2], int(x[3]), x[4]   ) )
            cursor.execute("INSERT INTO detalle_voto (id_voto, id_candidato) VALUES (%s,%s);", (int(x[0]), int(x[1])  ) )
        conexion.connection.commit()
        return("Datos cargados")



    except Exception as e:
            print("Error al insertar:", str(e))
            return("Error al insertar")
```

Endpoint consulta1

En este endpoint se obtiene los datos de la base de datos y se devuelve el nombre de los candidatos a presidentes y vicepresidentes por partido

```python
@app.route('/consulta1', methods=['GET'])
def consulta1():
    try:
        cursor = conexion.connection.cursor()
        try:
            cursor.execute("""SELECT
                P.nombre_partido AS partido,
                (SELECT C1.nombre FROM candidato C1 WHERE C1.id_partido = P.id_partido AND C1.id_cargo = 1) AS nombre_presidente,
                (SELECT C2.nombre FROM candidato C2 WHERE C2.id_partido = P.id_partido AND C2.id_cargo = 2) AS nombre_vicepresidente
            FROM partido P
            WHERE P.nombre_partido <> 'NULO';
                        """)
            res = cursor.fetchall()
            resjson = []
            for x in res:
                resjson.append({"Partido": x[0], "Presidente": x[1], "Vicepresidente": x[2]})
            return jsonify(resjson)
        except Exception as e:
                print("Error al consultar:", str(e))

        res = cursor.fetchall()
        resjson = []
        for x in res:
            resjson.append({"Candidato": x[0], "Partido": x[1]})
        return jsonify(resjson)
    except:
        return("Error en consulta 1")
```

Endpoint consulta2

En este endpoint se obtiene los datos de la base de datos y se devuelve numero de candidatos a diputados

```python
@app.route('/consulta2', methods=['GET'])
def consulta2():
    try:
        cursor = conexion.connection.cursor()
        cursor.execute("""SELECT COUNT(*) AS cuenta, p.nombre_partido AS partido
        FROM candidato c
        inner join partido p on c.id_partido = p.id_partido
        inner join cargo ca on c.id_cargo = ca.id_cargo
        WHERE c.id_cargo IN (3, 4, 5)
        GROUP BY c.id_partido;""")
        cantidad = cursor.fetchall()
        resjson = []

        for x in cantidad:
            resjson.append({"Cantidad de candidatos a diputados": x[0], "Partido": x[1]})

        return jsonify(resjson)
    except:
        return("Error en consulta 2")
```

Endpoint consulta3

En este endpoint se obtiene los datos de la base de datos y se devuelve el nombre de los candidatos a alcalde por partido

```python
@app.route('/consulta3', methods=['GET'])
def consulta3():
    try:
        cursor = conexion.connection.cursor()
        cursor.execute("""SELECT p.nombre_partido AS partido, c.nombre AS candidato
        FROM candidato c
        INNER JOIN partido p ON c.id_partido = p.id_partido
        WHERE c.id_cargo = 6;""")
        res = cursor.fetchall()
        resjson = []
        for x in res:
            resjson.append({"Partido": x[0], "Candidato": x[1]})


        return jsonify(resjson)
    except:
        return("Error en consulta 3")
```

Endpoint consulta4

En este endpoint se obtiene los datos de la base de datos y se devuelve la cantidad de candidatos por partido

```python
@app.route('/consulta4', methods=['GET'])
def consulta4():
    try:
        cursor = conexion.connection.cursor()
        cursor.execute("""SELECT COUNT(c.nombre) as cuenta, p.nombre_partido AS partido
                    FROM candidato c
                    inner join partido p on c.id_partido = p.id_partido
                    inner join cargo ca on c.id_cargo = ca.id_cargo
                    WHERE c.id_cargo IN (1, 2,3,4,5,6)
                    GROUP BY c.id_partido;""")
        res = cursor.fetchall()

        resjson = []
        for x in res:
            resjson.append({"Cantidad de candidatos": x[0], "Partido": x[1]})
        return jsonify(resjson)
    except:
        return("Error en consulta 4")
```

Endpoint consulta5

En este endpoint se obtiene los datos de la base de datos y se devuelve la cantidad de votaciones por departamento

```
@app.route('/consulta5', methods=['GET'])
def consulta5():
    try:
        #INNER JOIN voto v ON dv.id_voto = v.id_voto
        cursor = conexion.connection.cursor()
        cursor.execute("""SELECT COUNT(*) AS numero_de_registros, d.nombre_departamento AS departamento
FROM voto dv
INNER JOIN mesa m ON dv.id_mesa = m.id_mesa
INNER JOIN departamento d ON m.id_departamento = d.id_departamento
GROUP BY d.nombre_departamento;""")
        cantidad = cursor.fetchall()
        resjson = []
        for x in cantidad:
            resjson.append({"Numero de votos": x[0], "Departamento": x[1]})

        return jsonify(resjson)
    except:
        return("Error en consulta 5")
```

Endpoint consulta6

En este endpoint se obtiene los datos de la base de datos y se devuelve la cantidad de votos nulos

```
@app.route('/consulta6', methods=['GET'])
def consulta6():
    try:
        cursor = conexion.connection.cursor()
        cursor.execute("""SELECT COUNT(*) AS numero_de_registros
        FROM voto
        WHERE id_candidato = -1;""")
        cantidad = cursor.fetchall()
        return jsonify({"Numero de votos nulos": cantidad[0][0]})
    except:
        return("Error en consulta 6")
```

Endpoint consulta7

En este endpoint se obtiene los datos de la base de datos y se devuelve el top 10 de edad de ciudadanos que realizaron su voto

```python
@app.route('/consulta7', methods=['GET'])
def consulta7():
    try:
        cursor = conexion.connection.cursor()
        cursor.execute("""SELECT c.edad, COUNT(*) AS frecuencia
FROM voto v
INNER JOIN ciudadano c ON v.dpi = c.dpi
GROUP BY c.edad
ORDER BY COUNT(*) DESC
LIMIT 10;""")
        cantidad = cursor.fetchall()

        resjson = []
        for x in cantidad:
            resjson.append({"Edad": x[0], "Frecuencia": x[1]} )

        return jsonify(resjson)
    except:
        return("Error en consulta 7")
```

Endpoint consulta8

En este endpoint se obtiene los datos de la base de datos y se devuelve el top 10 de candidatos mas votados para presidente y vicepresidente

```python
def consulta8():
    try:
        cursor = conexion.connection.cursor()
        cursor.execute("""
        SELECT c1.nombre AS nombre_presidente, c2.nombre AS nombre_vicepresidente, COUNT(*) AS cantidad_de_votos
        FROM candidato as c1
        INNER JOIN candidato as c2 ON c1.id_partido = c2.id_partido AND (c1.id_cargo = 1 AND c2.id_cargo = 2)
        INNER JOIN detalle_voto as dv ON c1.id_candidato = dv.id_candidato
        GROUP BY nombre_presidente, nombre_vicepresidente
        ORDER BY cantidad_de_votos DESC
        LIMIT 10;
        """)
        cantidad = cursor.fetchall()
        resjson = []
        for x in cantidad:
            resjson.append({"Presidente": x[0], "Vicepresidente": x[1], "Cantidad de votos": x[2]} )


        return jsonify(resjson)
    except Exception as e:
        print("Error al insertar:", str(e))
        return("Error en consulta 8")
```

Endpoint consulta9

En este endpoint se obtiene los datos de la base de datos y se devuelve el top 5 de mesas mas frecuentadas por departamento

```python
@app.route('/consulta9', methods=['GET'])
def consulta9():
    #INNER JOIN voto v ON dv.id_voto = v.id_voto
    try:
        cursor = conexion.connection.cursor()
        cursor.execute("""SELECT m.id_mesa AS numero_de_mesa, d.nombre_departamento AS departamento, COUNT(*) AS cantidad_de_votos
FROM voto dv
INNER JOIN mesa m ON dv.id_mesa = m.id_mesa
INNER JOIN departamento d ON m.id_departamento = d.id_departamento
GROUP BY m.id_mesa, d.nombre_departamento
ORDER BY cantidad_de_votos DESC
LIMIT 5;""")
        cantidad = cursor.fetchall()

        resjson = []
        for x in cantidad:
            resjson.append({"Numero de mesa": x[0], "Departamento": x[1], "Cantidad de votos": x[2]} )


        return jsonify(resjson)
    except Exception as e:
        print("Error al insertar:", str(e))
        return("Error en consulta 9")
```

Endpoint consulta10

En este endpoint se obtiene los datos de la base de datos y se devuelve el top 5 de horas mas concurridas en los ciudadanos que fueron a votar

```python
@app.route('/consulta10', methods=['GET'])
def consulta10():
    #INNER JOIN voto v ON dv.id_voto = v.id_voto
    try:
        cursor = conexion.connection.cursor()
        cursor.execute("""SELECT
    DATE_FORMAT(dv.fechahora_voto, '%H:%i') AS hora,
    COUNT(*) AS cantidad_de_votos
FROM voto dv
GROUP BY DATE_FORMAT(dv.fechahora_voto, '%H:%i')
ORDER BY cantidad_de_votos DESC
LIMIT 5;""")
        cantidad = cursor.fetchall()
        resjson = []
        for x in cantidad:
            resjson.append({"Hora": x[0], "Cantidad de votos": x[1]} )


        return jsonify(resjson)
    except Exception as e:
        print("Error al insertar:", str(e))
        return("Error en consulta 10")
```

Endpoint consulta11

En este endpoint se obtiene los datos de la base de datos y se devuelve la cantidad de votos por genero

```python
@app.route('/consulta11', methods=['GET'])
def consulta12():
    #INNER JOIN voto v ON dv.id_voto = v.id_voto
    try:
        cursor = conexion.connection.cursor()
        cursor.execute("""SELECT c.genero AS genero, COUNT(*) AS cantidad_de_votos
FROM voto dv
INNER JOIN ciudadano c ON c.dpi = dv.dpi
GROUP BY c.genero;""")
        cantidad = cursor.fetchall()
        resjson = []
        for x in cantidad:
            resjson.append({"Genero": x[0], "Cantidad de votos": x[1]})

        return jsonify(resjson)
    except Exception as e:
        print("Error al insertar:", str(e))

        return("Error en consulta 11")
```