🧐

Manual Técnico

Requerimientos

1. NodeJS v14.17.2
2. Docker
3. Postman
4. DBeaver (u otro cliente de base de datos)
5. Datamodeler
6. Git

Configuración

Repositorio

Todo el código fuente del sistema está pubilcado en un repositorio de GitHub. Para obtener el código fuente es necesario ejecutar el siguiente comando:

```
git clone https://github.com/pereznator/bases1-proyecto1.git
```

Una vez descargado el código fuente del programa, se deben descargar todas las dependencias que se utilizan. Como el sistema está desarrollado en NodeJS, es necesario ejecutar el siguiente comando en la raíz del repositorio antes de poder utilizar el sistema:

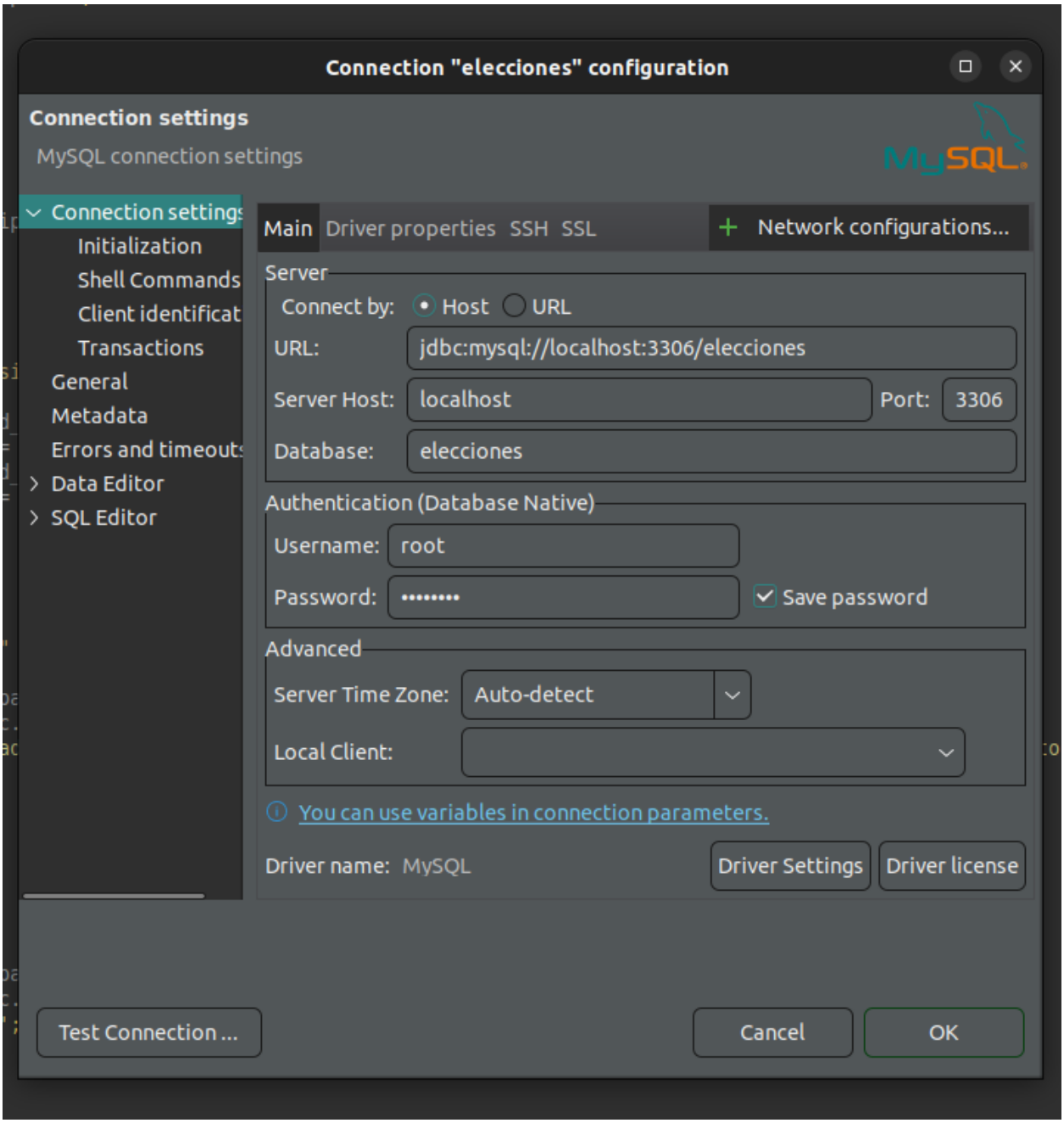
```
npm install
```

MySQL

El sistema utiliza una base de datos de MySQL ejecutandose en un contenedor de Docker. Dentro del repositorio del proyecto, hay una carpeta llamada “Docker”, en donde se encuentra el archivo docker-compose.yaml. En la consola de comandos, entrar a esa carpeta y ejecutar el siguiente comando para levantar una instancia de MySQL con Docker:

```
sudo docker-compose up -d
```

Para poder visualizar el contenido de la base de datos, se puede utilizar DBeaver. Solamente se debe crear una nueva conexión e ingresar las credenciales de la base de datos que se encuentran en el archivo docker-comose.yaml.



Iniciar el Programa

En la raíz del repositorio ejecutar los siguientes comandos para poder empezar a utilizar el API y poder desarrollar nuevos cambios.

Como se está utilizando TypeScript, para poder compilar los cambios es necesario ejecutar el siguiente comando:

```
npm run watcht-s
```

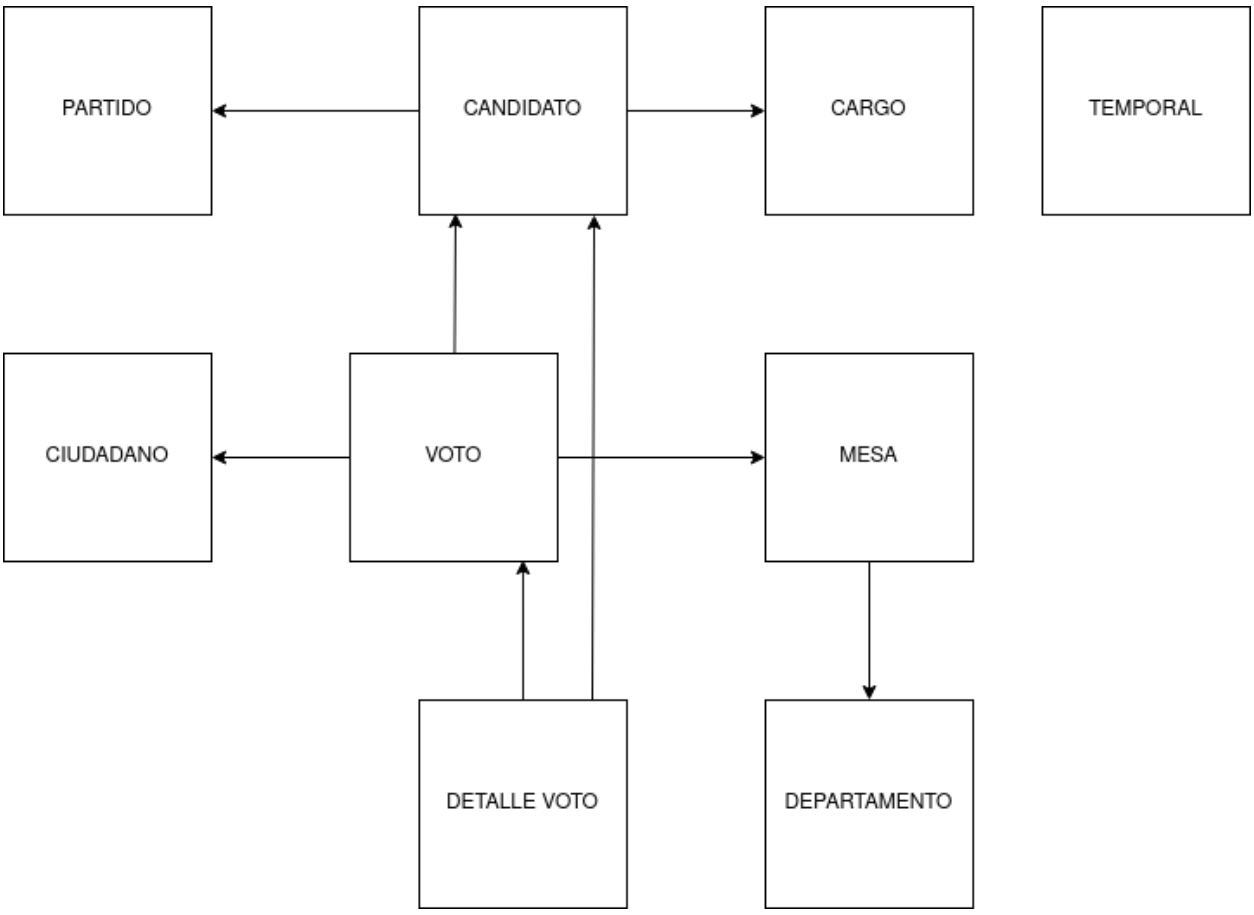
Para poder iniciar el sistema, se debe ejecutar el siguiente comando:

```
npm run dev
```

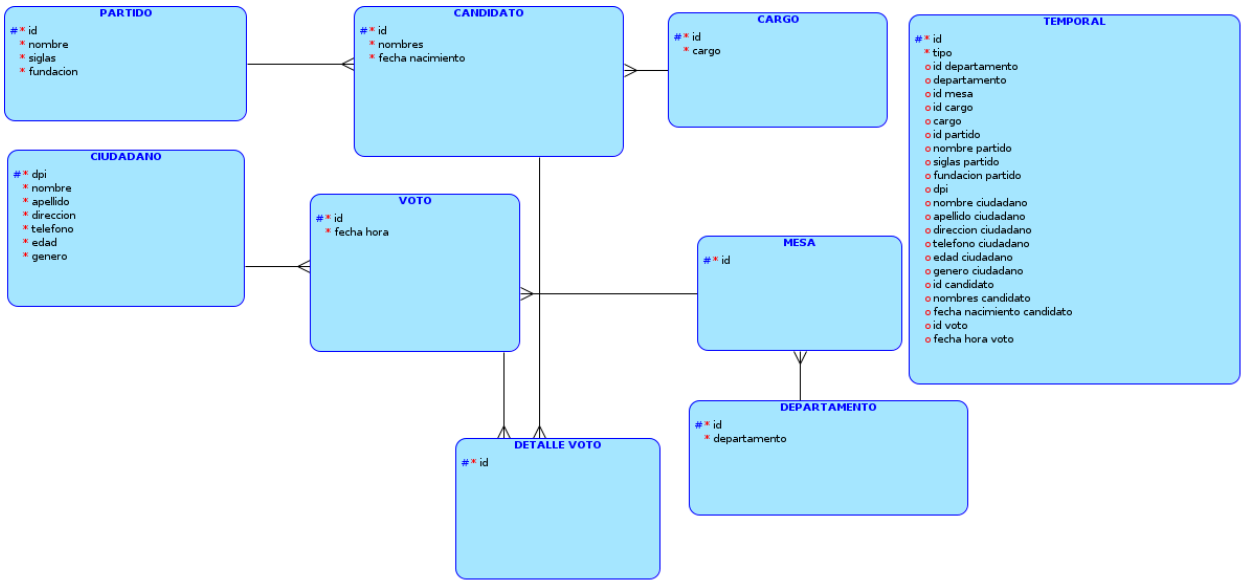
Modelos

La base de datos sigue una estructura especialmente diseñada para las necesidades y requerimientos del sistema. Los siguientes modelos dan una mejor idea de como estan distribuidas las tablas y sus atributos para un mejor desempeño.

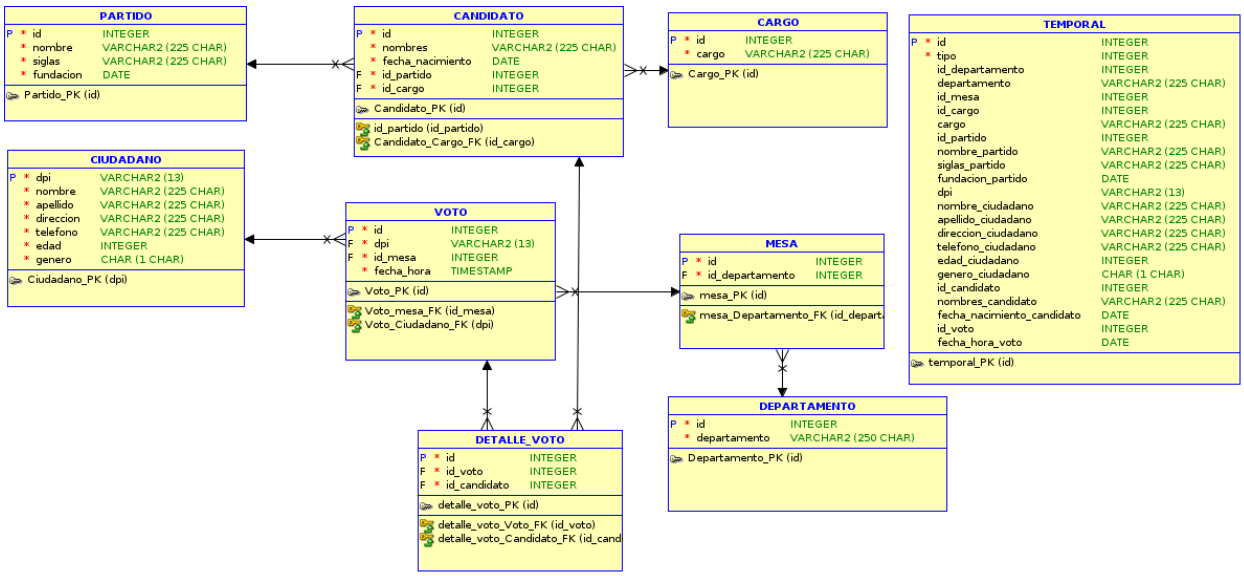
Modelo Coceptual



Modelo Lógico



Modelo Físico



Consultas

El sistema utiliza el lenguaje SQL para hacer consultas a la base de datos. Consultas para la creación de la estructura de la base de datos y consultas para hacer reportes para luego obtener información importante sobre toda la información almacenada en la misma.

Creación del Modelo

```
CREATE TABLE candidato (
    id               INTEGER NOT NULL,
    nombres          VARCHAR(225) NOT NULL,
    fecha_nacimiento DATETIME NOT NULL,
    id_partido       INTEGER NOT NULL,
    id_cargo         INTEGER NOT NULL
);

ALTER TABLE candidato ADD CONSTRAINT candidato_pk PRIMARY KEY ( id );

CREATE TABLE cargo (
    id    INTEGER NOT NULL,
    cargo VARCHAR(225) NOT NULL
);

ALTER TABLE cargo ADD CONSTRAINT cargo_pk PRIMARY KEY ( id );

CREATE TABLE ciudadano (
    dpi       VARCHAR(13) NOT NULL,
    nombre    VARCHAR(225) NOT NULL,
    apellido  VARCHAR(225) NOT NULL,
    direccion VARCHAR(225) NOT NULL,
    telefono  VARCHAR(225) NOT NULL,
    edad      INTEGER NOT NULL,
    genero    CHAR(1) NOT NULL
);

ALTER TABLE ciudadano ADD CONSTRAINT ciudadano_pk PRIMARY KEY ( dpi );

CREATE TABLE departamento (
    id           INTEGER NOT NULL,
    departamento VARCHAR(250) NOT NULL
);

ALTER TABLE departamento ADD CONSTRAINT departamento_pk PRIMARY KEY ( id );

CREATE TABLE detalle_voto (
    id           INTEGER NOT NULL,
    id_voto      INTEGER NOT NULL,
    id_candidato INTEGER NOT NULL
);

ALTER TABLE detalle_voto ADD CONSTRAINT detalle_voto_pk PRIMARY KEY ( id );

CREATE TABLE mesa (
    id              INTEGER NOT NULL,
    id_departamento INTEGER NOT NULL
);

ALTER TABLE mesa ADD CONSTRAINT mesa_pk PRIMARY KEY ( id );

CREATE TABLE partido (
    id        INTEGER NOT NULL,
    nombre    VARCHAR(225) NOT NULL,
    siglas    VARCHAR(225) NOT NULL,
    fundacion DATETIME NOT NULL
);
```

```
ALTER TABLE partido ADD CONSTRAINT partido_pk PRIMARY KEY ( id );

CREATE TABLE temporal (
    id                          INTEGER NOT NULL,
    tipo                        INTEGER NOT NULL,
    id_departamento             INTEGER,
    departamento                VARCHAR(225),
    id_mesa                     INTEGER,
    id_cargo                    INTEGER,
    cargo                       VARCHAR(225),
    id_partido                  INTEGER,
    nombre_partido              VARCHAR(225),
    siglas_partido              VARCHAR(225),
    fundacion_partido           DATETIME,
    dpi                         VARCHAR(13),
    nombre_ciudadano            VARCHAR(225),
    apellido_ciudadano          VARCHAR(225),
    direccion_ciudadano         VARCHAR(225),
    telefono_ciudadano          VARCHAR(225),
    edad_ciudadano              INTEGER,
    genero_ciudadano            CHAR(1),
    id_candidato                INTEGER,
    nombres_candidato           VARCHAR(225),
    fecha_nacimiento_candidato DATETIME,
    id_voto                     INTEGER,
    fecha_hora_voto             DATETIME
);

ALTER TABLE temporal ADD CONSTRAINT temporal_pk PRIMARY KEY ( id );

CREATE TABLE voto (
    id         INTEGER NOT NULL,
    dpi        VARCHAR(13) NOT NULL,
    id_mesa    INTEGER NOT NULL,
    fecha_hora DATETIME(6) NOT NULL
);

ALTER TABLE voto ADD CONSTRAINT voto_pk PRIMARY KEY ( id );

ALTER TABLE candidato
    ADD CONSTRAINT candidato_cargo_fk FOREIGN KEY ( id_cargo )
        REFERENCES cargo ( id )
            ON DELETE CASCADE;

ALTER TABLE detalle_voto
    ADD CONSTRAINT detalle_voto_candidato_fk FOREIGN KEY ( id_candidato )
        REFERENCES candidato ( id )
            ON DELETE CASCADE;

ALTER TABLE detalle_voto
    ADD CONSTRAINT detalle_voto_voto_fk FOREIGN KEY ( id_voto )
        REFERENCES voto ( id )
            ON DELETE CASCADE;

ALTER TABLE candidato
    ADD CONSTRAINT id_partido FOREIGN KEY ( id_partido )
        REFERENCES partido ( id )
```

```
            ON DELETE CASCADE;

ALTER TABLE mesa
    ADD CONSTRAINT mesa_departamento_fk FOREIGN KEY ( id_departamento )
        REFERENCES departamento ( id )
            ON DELETE CASCADE;

ALTER TABLE voto
    ADD CONSTRAINT voto_ciudadano_fk FOREIGN KEY ( dpi )
        REFERENCES ciudadano ( dpi )
            ON DELETE CASCADE;

ALTER TABLE voto
    ADD CONSTRAINT voto_mesa_fk FOREIGN KEY ( id_mesa )
        REFERENCES mesa ( id )
            ON DELETE CASCADE;

ALTER TABLE elecciones.temporal MODIFY COLUMN id int auto_increment NOT NULL;

ALTER TABLE elecciones.detalle_voto MODIFY COLUMN id int auto_increment NOT NULL;
```

Eliminación de Tablas

```
SET @tableName = '${nombreTabla}';
SET @schemaName = 'elecciones';

-- Verificar si la tabla existe
SELECT COUNT(*)
INTO @tableExists
FROM information_schema.tables
WHERE table_schema = @schemaName
AND table_name = @tableName;

-- Eliminar la tabla si existe
SET @dropTableSQL = IF(@tableExists = 1, CONCAT('DROP TABLE ', @schemaName, '.', @tableName), NULL);
PREPARE stmt FROM @dropTableSQL;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;
```

Listar Presidentes y Vicepresidentes Por Partido

```
select
c1.nombres as "Presidente",
c2.nombres as "Vicepresidente",
p.nombre as "Partido"
from partido p
join candidato c1 on c1.id_partido = p.id
join cargo ca1 on ca1.id = c1.id_cargo and ca1.cargo = 'presidente'
join candidato c2 on c2.id_partido = p.id
join cargo ca2 on ca2.id = c2.id_cargo and ca2.cargo = 'vicepresidente'
;
```

Listar Numero de Candidatos a Diputados Por Partido

```
select
p.nombre as "Partido",
count(c.id) as "Diputados"
from partido p
join candidato c on c.id_partido = p.id
join cargo c2 on c2.id = c.id_cargo
where c2.cargo in ('diputado congreso lista nacional',
'diputado congreso distrito electoral',
'diputado parlamento centroamericano')
group by p.nombre
```

Listar Nombre de Candidatos a Alcalde Por Partido

```
select
p.nombre as "Partido",
c.nombres as "Alcalde"
from partido p
join candidato c ON c.id_partido = p.id
join cargo ca on ca.id = c.id_cargo
where ca.cargo = 'alcalde';
```

Listar Numero de Candidatos Por Partido

```
select
p.nombre as "Partido",
count(c.id) as "Candidatos"
from partido p
join candidato c on c.id_partido = p.id
group by p.nombre
;
```

Listar Cantidad de Votos Por Departamento

```
select
d.departamento as "Departamento",
count(v.id) as "Votos"
from departamento d
join mesa m on m.id_departamento = d.id
join voto v on v.id_mesa = m.id
group by d.departamento
```

Cantidad de Votos Nulos

```
select COUNT(*) as "VotosNulos"
from voto v
join detalle_voto dv ON dv.id_voto = v.id
where dv.id_candidato = -1
```

Listar Top 10 Edades de Ciudadanos en Votar

```
select
c.edad as "Edad",
count(c.dpi) as "Cantidad"
from ciudadano c
join voto v on v.dpi = c.dpi
group by c.edad
order by count(c.dpi) DESC
limit 10;
```

Listar Top 10 Candidatos Más Votados Para Presidente Y VicePresidente

```
select
presidente.nombres as "Presidente",
vicepresidente.nombres as "Vicepresidente",
p.partido as "Partido",
count(v.id) as "Votos"
from voto v
join detalle_voto dv on dv.id_voto = v.id
join candidato presidente on presidente.id = dv.id_candidato
join cargo cargoPresidente on cargoPresidente.id = presidente.id_cargo
  and cargoPresidente.cargo = 'presidente'
join partido p on p.id = presidente.id_partido
join candidato vicepresidente on vicepresidente.id_partido = p.id
join cargo cargoVicepresidente on cargoVicepresidente.id = vicepresidente.id_cargo
  and cargoVicepresidente.cargo = 'vicepresidente'
group by presidente.nombres, vicepresidente.nombres
order by count(v.id) DESC
limit 10;
```

Listart Top 5 Mesas Más Frecuentadas

```
select
m.id as "Mesa",
d.departamento as "Departamento",
count(v.id) as "Votos"
from mesa m
join voto v on v.id_mesa = m.id
join departamento d on d.id = m.id_departamento
group by m.id
```

```
order by count(v.id) desc
limit 5;
```

Listar Top 5 Horas Más Concurridas Para Votar

```
SELECT
    TIME(fecha_hora) AS Hora,
    COUNT(*) AS Votos
FROM voto
GROUP BY Hora
ORDER BY Votos DESC
LIMIT 5;
```

Listar Votos Por Género

```
select
c.genero as Genero,
count(v.id) as Votos
from ciudadano c
join voto v on v.dpi = c.dpi
group by Genero;
```