

1.- Instalacion de node version 10

- windows

<https://nodejs.org/dist/latest-v10.x/>

- linux

```
sudo apt install curl
curl -sL https://deb.nodesource.com/setup_10.x | sudo -E bash
sudo apt install nodejs
```

2.- Checar version de node en terminal

```
node -version
```

3.- Instalación de Visual Studio Code

<https://code.visualstudio.com/Download>

4.- Instalación de framework para creacion de API

```
npm install -g swagger
```

5.- Creacion de Rest API para proyecto

```
swagger project create data-rest-api
```

6.- Abrir proyecto con Visual Studio Code

7.- Probar proyecto desde terminal de Vscod, en modo MOCK

```
swagger project start -m
```

8.- Arrancar proyecto en modo normal y probar el editor de swagger

Terminal 1

```
swagger project start
```

Terminal 2

```
swagger project edit
```

9.- Instalar postgres

<https://www.enterprisedb.com/downloads/postgres-postgresql-downloads>

10.- crear base de datos data-science-xxxx

11.- Instalación de sequelize en proyecto swagger, abrir proyecto swagger y a nivel de la raíz del proyecto (package.json), ejecutar los siguiente comandos para instalar sequelize.

```
$ sudo npm install -g sequelize-cli
```

```
$ sudo npm install --save sequelize
```

```
$ sudo npm install --save pg pg-hstore
```

12.- Configuración, cambiarse a la carpeta api del proyecto swagger

```
$ cd api
```

- windows

```
$ notepad .sequelizerc
```

- linux

```
$ touch .sequelizerc
```

```
const path = require('path');
module.exports = {
  "config": path.resolve('./config', 'config.json'),
  "models-path": path.resolve('./models'),
  "seeders-path": path.resolve('./seeders'),
  "migrations-path": path.resolve('./migrations')
};
```

\$ sequelize init

\$ editar config/config.json

```
{
  "development": {
    "username": "postgres",
    "password": "xxxxxx",
    "database": "data-science-xxxx",
    "host": "xx.xx.xx.xx",
    "dialect": "postgres"
  },
  "test": {
    "username": "root",
    "password": "dj@mw@r3",
    "database": "node_sequelize",
    "host": "127.0.0.1",
    "dialect": "postgres"
  },
  "production": {
    "username": "postgres",
    "password": "xxxx",
    "database": "data-science-xxxx",
    "host": "xx.xx.xx.xx",
    "dialect": "postgres"
  }
}
```

13.- Creación de Modelo Denues

```
$ sequelize model:create --name Denues --attributes  
nombre:string,tipo:string,descripcion:string,calle:string,numero:string,colonia:string,cp:string,ideestado:string,estado:string,idmunicipio:string,municipio:string,lat:string,lng:string
```

- Realizar Migration

```
$ sequelize db:migrate
```

- quitar restriccion de not null a campos createdAt y updatedAt

- importar **denues.csv** desde pgAdmin

14.- Creación de Modelo Censos

```
$ sequelize model:create --name Censos --attributes  
ideestado:string,estado:string,idmunicipio:string,municipio:string,actividad:string,UE:decimal,H001A:decimal,H010A:decimal,A111A:decimal,A211A:decimal,M091A:decimal,H010D:decimal,H020A:decimal,I000A:decimal,I100A:decimal,I200A:decimal,K000A:decimal,K020A:decimal,K311A:decimal,K040A:decimal,K041A:decimal,K050A:decimal,K620A:decimal,K060A:decimal,K810A:decimal,K090A:decimal,A700A:decimal,M000A:decimal,M020A:decimal,M090A:decimal,A800A:decimal
```

- Realizar Migration

```
$ sequelize db:migrate
```

- quitar restriccion de not null a campos createdAt y updatedAt

- importar **censos.csv** desde pgAdmin

15.- Creación de Modelo Estados

```
$ sequelize model:create --name Estados --attributes ideestado:string,estado:string
```

- Realizar Migration

```
$ sequelize db:migrate
```

- quitar restriccion de not null a campos createdAt y updatedAt

- importar **estados.csv** desde pgAdmin

16.- Creación de Modelo Municipios

```
$ sequelize model:create --name Municipios --attributes  
ideestado:string,idmunicipio:string,municipio:string
```

- Realizar Migration

```
$ sequelize db:migrate
```

- quitar restriccion de not null a campos createdAt y updatedAt
- importar **municipios.csv** desde pgAdmin

17.- Creación de Modelo Bancos

```
$ sequelize model:create --name Bancos --attributes tipo:string,descripcion:string
```

- Realizar Migration

```
$ sequelize db:migrate
```

- quitar restriccion de not null a campos createdAt y updatedAt
- importar **bancos.csv** desde pgAdmin

18.- Creación de operaciones en controller, crear archivo /controllers/data.controller.js

```
const { Estados } = require('../models'); // Sequelize
const { Municipios } = require('../models'); // Sequelize
const { Bancos } = require('../models'); // Sequelize
const { Censos } = require('../models'); // Sequelize
const { Denues } = require('../models'); // Sequelize

const MODULE_NAME = '[data Controller]';

function getEstados(req, res) {

  try {
    console.log("Estados...");
    console.log(Estados);
    Estados.findAll({
    })
    .then((estados) => {
      console.log(estados);
      res.status(200).send(estados);
    }, (error) => {
      res.status(500).send(error);
    });
  } catch (error) {
    controllerHelper.handleErrorResponse(MODULE_NAME, getEstados.name, error, res);
  }
}
```

```

}
function getMunicipios(req, res) {
  try {
    var params = {
      entidad: req.swagger.params.entidad.value
    };
    console.log("municipios..." + params);
    console.log(Municipios);
    Municipios.findAll(
      {
        where: {
          idestado : params.entidad
        }
      }
    )
    .then((municipios) => {
      console.log(municipios);
      res.status(200).send(municipios);
    }, (error) => {
      console.log("error : " + error);
      res.status(500).send(error);
    });
  } catch (error) {
    controllerHelper.handleErrorResponse(MODULE_NAME, getGameSystems.name, error, res);
  }
}
}

```

```

function getBancos(req, res) {
  try {
    console.log("Bancos...");
    console.log(Bancos);
    Bancos.findAll(
      {
      }
    )
    .then((bancos) => {
      console.log(bancos);
      res.status(200).send(bancos);
    }, (error) => {
      console.log("error : " + error);
      res.status(500).send(error);
    });
  } catch (error) {
    controllerHelper.handleErrorResponse(MODULE_NAME, getBancos.name, error, res);
  }
}
}

```

```

function getCensosByMun(req, res) {

```

```

try {
var params = {
entidad: req.swagger.params.entidad.value,
municipio: req.swagger.params.municipio.value
};

console.log("censos..." + params);
console.log(Censos);
Censos.findAll({
where: {
idestado : params.entidad,
idmunicipio : params.municipio,
}
})
.then((censos) => {
res.status(200).send(censos);
}, (error) => {
console.log("error : " + error);
res.status(500).send(error);
});
} catch (error) {
controllerHelper.handleErrorResponse(MODULE_NAME, getCensosByMun.name, error, res);
}
}

```

```

function getCensosByEdo(req, res) {
try {
var params = {
entidad: req.swagger.params.entidad.value
};

console.log("censos..." + params);
console.log(Censos);
Censos.findAll({
where: {
idestado : params.entidad,
actividad: 'Total municipal'
}
})
.then((censos) => {
res.status(200).send(censos);
}, (error) => {
console.log("error : " + error);
res.status(500).send(error);
}

```

```

});
} catch (error) {
controllerHelper.handleErrorResponse(MODULE_NAME, getCensosByEdo.name, error, res);
}
}

```

```

function getDenues(req, res) {
try {
var params = {
entidad: req.swagger.params.entidad.value,
municipio: req.swagger.params.municipio.value,
tipo: req.swagger.params.tipo.value

```

```

};

```

```

console.log("Denues..." + params);
console.log(Denues);
Denues.findAll({
where: {
idestado : params.entidad,
idmunicipio : params.municipio,
tipo : params.tipo
}
})
.then((denues) => {
res.status(200).send(denues);
}, (error) => {
console.log("error : " + error);
res.status(500).send(error);
});
} catch (error) {
controllerHelper.handleErrorResponse(MODULE_NAME, getDenues.name, error, res);
}

```

```

}

```

```

module.exports = {
getMunicipios,
getEstados,
getBancos,
getCensosByMun,
getCensosByEdo,
getDenues,
MODULE_NAME
}

```


19.- Edicion del archivo swagger.yaml

```
swagger: "2.0"
info:
  version: "0.0.1"
  title: Hello World App
  # during dev, should point to your local machine
  host: localhost:10010
  # basePath prefixes all resource paths
  basePath: /
  #
  schemes:
    # tip: remove http to make production-grade
    - http
    - https
    # format of bodies a client can send (Content-Type)
  consumes:
    - application/json
    # format of the responses to the client (Accepts)
  produces:
    - application/json
  paths:
    /hello:
      x-swagger-router-controller: hello_world
      get:
        description: Returns 'Hello' to the caller
        # used as the method name of the controller
        operationId: hello
        parameters:
          - name: name
            in: query
            description: The name of the person to whom to say hello
            required: false
            type: string
        responses:
          "200":
            description: Success
            schema:
              # a pointer to a definition
              $ref: "#/definitions/HelloWorldResponse"
              # responses may fall through to errors
            default:
              description: Error
```

```
schema:
  $ref: "#/definitions/ErrorResponse"
/entidades:
  x-swagger-router-controller: data.controller
  get:
    description: get the game system list
    operationId: getEstados
```

```
responses:
  "200":
    description: Success
    schema:
      $ref: "#/definitions/GetEntidadListResponse"
    default:
      description: Error
      schema:
        $ref: "#/definitions/ErrorResponse"
```

```
/municipios:
  x-swagger-router-controller: data.controller
  get:
    description: get the game system list
    operationId: getMunicipios
    parameters:
      - name: entidad
        in: query
        type: string
        required: true
    responses:
      "200":
        description: Success
        schema:
          $ref: "#/definitions/GetMunicipioListResponse"
        default:
          description: Error
          schema:
            $ref: "#/definitions/ErrorResponse"
```

```
/bancos:
  x-swagger-router-controller: data.controller
  get:
    description: get the game system list
    operationId: getBancos
    responses:
      "200":
        description: Success
        schema:
```

```
$ref: "#/definitions/GetBancoListResponse"  
default:  
description: Error  
schema:  
$ref: "#/definitions/ErrorResponse"
```

```
/censobymun:  
x-swagger-router-controller: data.controller  
get:  
description: get the game system list  
operationId: getCensosByMun  
parameters:  
- name: entidad  
in: query  
type: string  
required: true  
- name: municipio  
in: query  
type: string  
required: true  
responses:  
"200":  
description: Success  
schema:  
$ref: "#/definitions/GetCensoListResponse"  
default:  
description: Error  
schema:  
$ref: "#/definitions/ErrorResponse"
```

```
/censobyedo:  
x-swagger-router-controller: data.controller  
get:  
description: get the game system list  
operationId: getCensosByEdo  
parameters:  
- name: entidad  
in: query  
type: string  
required: true  
responses:  
"200":  
description: Success  
schema:  
$ref: "#/definitions/GetCensoListResponse"  
default:  
description: Error
```

```
schema:
$ref: "#/definitions/ErrorResponse"

/denues:
x-swagger-router-controller: data.controller
get:
description: get the game system list
operationId: getDenues
parameters:
- name: entidad
in: query
type: string
required: true
- name: municipio
in: query
type: string
required: true
- name: tipo
in: query
type: string
required: true
responses:
"200":
description: Success
schema:
$ref: "#/definitions/GetDenuelistResponse"
default:
description: Error
schema:
$ref: "#/definitions/ErrorResponse"
```

```
/swagger:
x-swagger-pipe: swagger_raw
# complex objects have schema definitions
definitions:
HelloWorldResponse:
required:
- message
properties:
message:
type: string
ErrorResponse:
required:
- message
properties:
message:
```

type: string

GetMunicipioResponse:

type: object

properties:

id:

type: string

description: identifier

idmunicipio:

type: string

description: Name of the game

municipio:

type: string

description: Developer of the game

GetMunicipioListResponse:

required:

- videogames

properties:

videogames:

type: array

items:

\$ref: "#/definitions/GetMunicipioResponse"

GetEntidadResponse:

type: object

properties:

id:

type: string

description: identifier

idestado:

type: string

description: Name of the game

estado:

type: string

description: Developer of the game

GetEntidadListResponse:

required:

- videogames

properties:

videogames:

type: array

items:

\$ref: "#/definitions/GetEntidadResponse"

GetBancoResponse:

type: object

properties:

id:
type: string
description: identifier
tipo:
type: string
description: Name of the game
descripcion:
type: string
description: Developer of the game
GetBancoListResponse:
required:
- videogames
properties:
videogames:
type: array
items:
\$ref: "#/definitions/GetBancoResponse"

GetCensoResponse:
type: object
properties:
idestado:
type: string
description: Name of the Game System
idmunicipio:
type: string
description: Description of the Game System
GetCensoListResponse:
required:
- gamesystems
properties:
gamesystems:
type: array
items:
\$ref: "#/definitions/GetCensoResponse"

GetDenueResponse:
type: object
properties:
idestado:
type: string
description: Name of the Game System
idmunicipio:
type: string
description: Description of the Game System
GetDenueListResponse:
required:

```
- gamesystems
properties:
gamesystems:
type: array
items:
$ref: "#/definitions/GetDenueResponse"
```

20.- Test

swagger project start

swagger project edit

21.- Agregar mas capas de datos como universidades, hospitales, comercios, entre otros.

<https://www.inegi.org.mx/app/mapa/denue/>

Referencias

<https://medium.com/@diegopm2000/creando-un-api-rest-con-swagger-node-c880bdac04a5>

<https://www.djamware.com/post/5b56a6cc80aca707dd4f65a9/nodejs-expressjs-sequelizejs-and-postgresql-restful-api>