1. El desarrollo debe estar alojado e implementado sobre alguno de los siguientes servicios IaaS en la Nube: AWS, Google Cloud Firebase, Azure, Heroku, para ellos pueden utilizar la cuenta de correo institucional
2. Aparte del CRUD BÁSICO, normalmente generado automáticamente por todos los Framework Web, el desarrollo debe tener las siguientes funciones:
   1. Enviar correo
   2. Presentar Gráficos estadísticos
   3. Generar informes en PDF
   4. Iniciar sesión
   5. Ser Responsive
3. Cada alumno debe entregar un documento en el cual explique muy bien y de forma ilustrativa los elementos mas importantes de su desarrollo, tales como:
   1. Descripción del entorno de trabajo
      1. Hardware
      2. Software
         1. SO
         2. IDE
         3. Motor de BD
         4. Cliente FTP
         5. Servidor Web
         6. Fremework Base
         7. Librerías y/o plantillas adicionales
   2. Modelo de BD
   3. Modelo de clases Base o Modelo Conceptual
   4. Información sobre la instalación y configuración básica e inicial del Framework y cada librería que utilicen en el desarrollo.
   5. Información sobre la estructura de carpeta y archivos del proyecto
   6. Script con fragmentos de código que contengan las instrucciones mas importantes del desarrollo, y su respectiva explicación destallada, por ejemplo:
      1. integración de las Vistas con el Controlador y el Modelo
      2. Instrucciones donde agrega, Busca, edita, elimina y lista
      3. Instrucciones donde envía el correo, genera el gráfico estadístico y el reporte en PDF
   7. Un video breve subido a youtube con explicación del buen funcionamiento del producto final

Como todo documento académico, este debe contener:

* Presentación
* introducción
* Objetivos
* Justificación

**INFORME FINAL PROYECTO**

CONTEO CIUDADANO

**RAFAEL ORTIZ VERGARA**

Código 7501810003

**TUTOR**

JOHN ARRIETA



**UNIVERSIDAD DE CARTAGENA**

**PROGRAMA DE INGENIERÍA DE SOFTWARE**

**DESARROLLO DE SOFTWARE WEB**

**CARTAGENA D.T. y C.**

**JULIO 2020**

**TABLA DE CONTENIDO**

[**INTRODUCCIÓN**](#_5h0bs4besq8n)3

[**TÍTULO**](#_k1zb3xvy0h0p)4

[**OBJETIVOS**](#_kj08dqzg2ubo)5

[**DESCRIPCIÓN DEL PROBLEMA**](#_yq30z7pvdiu0)6

[**DESARROLLO**](#_5dkw8tcvzj6j)7

[Descripción del entorno de trabajo](#_40eogg7wedkf) 7

[Navegadores sobre el cual hicieron las pruebas](#_qc63h172nhiv) 9

[Google Chrome v84](#_hsqdzlqnrza8) 9

[Modelo de BD](#_qc63h172nhiv) 9

[Modelo de clases Base o Modelo Conceptual](#_x95o7v4ypurv) 10

**Procedimiento para correr la app 139**

[**CONCLUSIONES**](#_5czz14s3r3x4)138

# INTRODUCCIÓN

El presente escrito busca presentar un informe acerca de la puesta en práctica de todo lo aprendido en las asignaturas Desarrollo web a través de un proyecto real en el que se pueda demostrar las habilidades adquiridas por medio de lo relacionado con el diseño e implementación de bases de datos para aplicaciones de software.

# 

# 

# 

# TÍTULO

Diseño, Implementación y conexión de Conteo Ciudadano App utilizando las tecnologías del PERN stack: PostgreSQL + Express + React + NodeJS.

# JUSTIFICACIÓN

La razón u objetivos de realizar este proyecto es con el único fin de comprender la naturaleza en ámbitos de desarrollo web, implementar protocolos de acciones que permitan realizar conexiones con servidores web, realizar aplicativos full stack que interactúen el front con el backend, implementando estrategias que permitan la viabilidad y confiabilidad del aplicativo desarrollado durante el proceso de aprendizaje y futuros diseños laborales.

# 

# 

# OBJETIVOS

* Comprender desde un punto de vista práctico los conceptos explicados en las unidades didácticas teóricas.
* Disponer de un modelo de referencia para emprender proyectos de Desarrollo Web
* Adquirir el criterio suficiente para identificar las actividades clave y tomar decisiones en un proyecto que implique el uso de Desarrollo Web.
* Diseñar una app que controle el censo de los ciudadanos de una población
* Mostrar los resultados obtenidos durante la elaboración del proyecto.
* Aprender la forma de usar algunas librería para realizar reportes PDF.

# 

# DESCRIPCIÓN DEL PROBLEMA

Para realizar el diseño de la Base de Datos del censo de población se dispone de la siguiente especificación de requerimientos:

* Se almacenarán datos de personas y datos de lugares.
* Cada persona tiene un nombre, apellidos, fecha de nacimiento, lugar de nacimiento, edad, estatura, sexo y nivel de estudios. Para los hombres también interesa conocer su situación militar. Para las personas mayores de 16 años también se debe considerar el DNI.
* Cada lugar corresponde a un municipio, teniendo como datos propios el nombre y el código (número secuencial del municipio según orden alfabético en cada provincia). Por ejemplo, Ciudad Real capital es el municipio “13034”, ya que el código de la provincia es el “13”.
* Todas las personas están censadas (población de derecho) en un municipio en una dirección (calle y número). Interesa conocer la fecha en que se registró dicha situación.
* Además, cada persona es residente (población de hecho) en un municipio en una dirección o en un país extranjero. En ambos casos queremos conocer también la fecha de inicio.
* Una persona puede ser residente en el mismo municipio.
* De cada municipio y de cada provincia interesa conocer su población de derecho (censada) y de hecho (residente realmente).

**Se pide:**

1. Diseñar el sistema de bases de datos para el caso anterior.
2. Proponer una forma lo más sencilla posible de transformar el anterior diagrama para que la dirección (calle y no) donde está censada una persona, nos permita conocer el distrito al que pertenece dentro del municipio. El distrito es una subdivisión del municipio compuesta por un conjunto de manzanas (una manzana no puede dividirse para pertenecer a dos distritos). Una manzana es un conjunto de casas físicamente unidas, separadas de las demás por calles. Cada lado de la manzana incluye una serie de números consecutivos (pares o impares) de una calle.

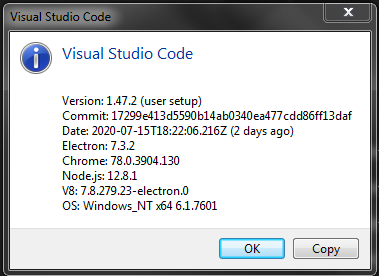
# 

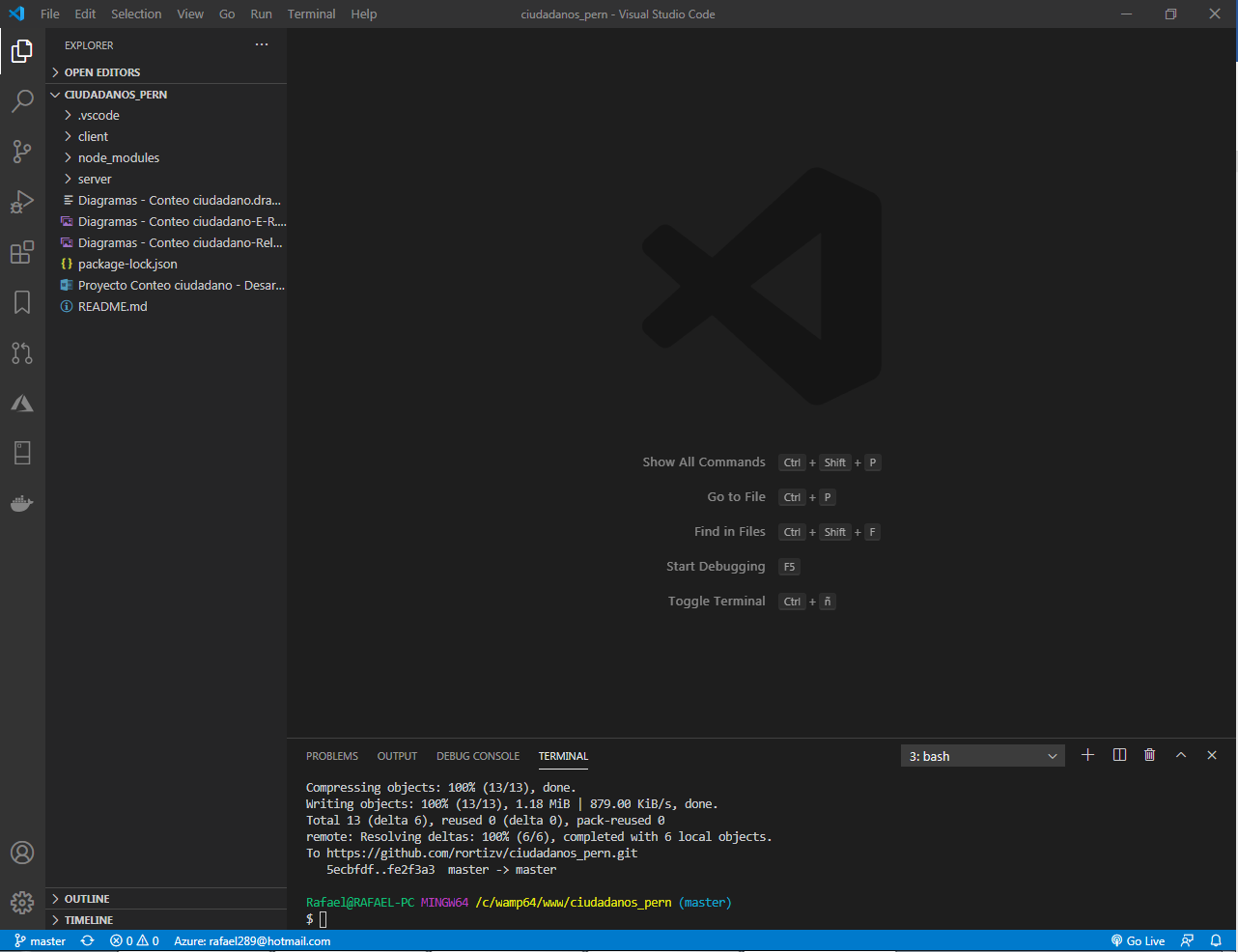
# 

# DESARROLLO

## Descripción del entorno de trabajo

* + 1. Editor de código: Visual Studio Code v1.47.2





* + 1. Motor de BD: Postgre SQL 12



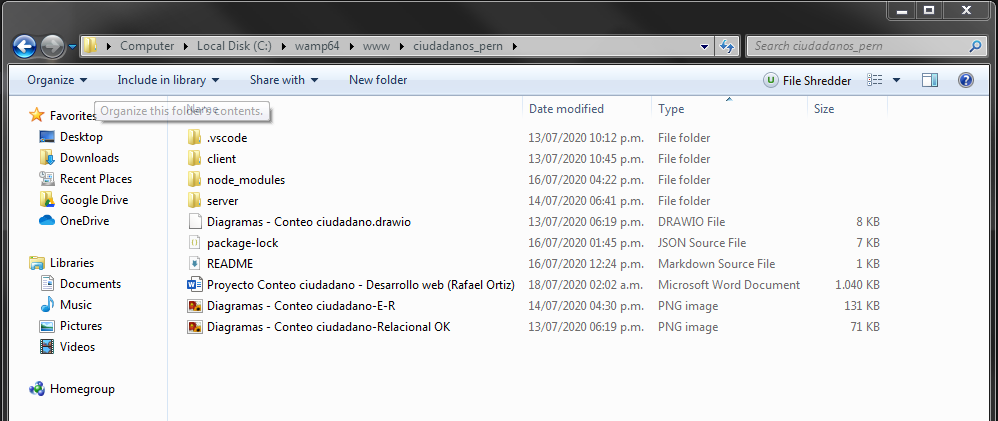
# Navegadores sobre el cual hicieron las pruebas

## Google Chrome v84

# Modelo de BD

# Modelo de clases Base o Modelo Conceptual

Información sobre la estructura de carpeta y archivos del proyecto



# ADMINISTRACIÓN DE LA BASE DE DATOS CON PGADMIN4

# 

# 

# Vista (carpeta): A continuación la estructura de la carpeta con el modelo de componentes manejado por React:

* Client
  + Public
    - Favicon.ico
    - Index.html
    - Logo192.png
    - Logo512.png
    - Manifest.json
    - Robots.txt
    - Src
      * Components
        + Censo\_derecho

EditCensoDerecho.js

InputCensoDerecho.js

ListCensoDerecho.js

* + - * + Censo\_distrital\_derecho

EditCensoDistritalDerecho.js

InputCensoDistritalDerecho.js

ListCensoDistritalDerecho.js

* + - * + Censo\_distrital\_hecho

EditCensoDistritalHecho.js

InputCensoDistritalHecho.js

ListCensoDistritlHecho.js

* + - * + Censo\_hecho

EditCensoHecho.js

InputCensoHecho.js

ListCensoHecho.js

* + - * + Home

Home.js

* + - * + Municipio

EditMunicipio.js

InputMunicipio.js

ListMunicipio.js

* + - * + Navbar

Navbar.js

* + - * + Persona

EditPersona.js

InputPersona.js

ListPersona.js

* + - * + Provincia

EditProvincia.js

InputProvincia.js

ListProvincia.js

* + - * App.css
      * App.js
      * Index.css
      * Index.js
* Server
  + Db.js
  + Index.js
  + Package.json
  + Package-lock.json

# SERVER

Dentro de esta carpeta, el archivo index.js comprende la API REST que controla el backend.

**Instrucciones donde agrega, Busca, edita, elimina y lista**

Estas instrucciones son llevadas a cabo en el archivo index.js, donde tenemos todas los comandos de los diferentes *crud’s.*

const express = require("express");

const app = express();

const cors = require("cors");

const pool = require("./db");

//middleware

app.use(cors());

app.use(express.json()); //req.body

app.use(express.urlencoded({extended: false}));

//\*\*\*\*\*\* ROUTES \*\*\*\*\*\*\*//

//\*\*\*PROVINCIA\*\*\*

//create a provincia

app.post("/provincia", async (req, res) => {

  try {

    const { nombre\_provincia } = req.body;

    const newProvincia = await pool.query(

      "INSERT INTO provincia (nombre\_provincia) VALUES($1) RETURNING \*",

      [nombre\_provincia]

    );

    res.json(newProvincia.rows[0]);

  } catch (err) {

    console.error(err.message);

  }

});

//get all provincia

app.get("/provincia", async (req, res) => {

  try {

    const allProvincia = await pool.query("SELECT \* FROM provincia");

    res.json(allProvincia.rows);

  } catch (err) {

    console.error(err.message);

  }

});

//get a provincia

app.get("/provincia/:id", async (req, res) => {

  try {

    const { id } = req.params;

    const provincia = await pool.query("SELECT \* FROM provincia WHERE cod\_provincia = $1", [

      id

    ]);

    res.json(provincia.rows[0]);

  } catch (err) {

    console.error(err.message);

  }

});

//update a provincia

app.put("/provincia/:id", async (req, res) => {

  try {

    const { id } = req.params;

    const { nombre\_provincia } = req.body;

    const updateProvincia = await pool.query(

      "UPDATE provincia SET nombre\_provincia = $1 WHERE cod\_provincia = $2",

      [nombre\_provincia, id]

    );

    res.json("Provincia actualizada!");

  } catch (err) {

    console.error(err.message);

  }

});

//delete a provincia

app.delete("/provincia/:id", async (req, res) => {

  try {

    const { id } = req.params;

    const deleteProvincia = await pool.query("DELETE FROM provincia WHERE cod\_provincia = $1", [

      id

    ]);

    res.json("Provincia eliminada!");

  } catch (err) {

    console.log(err.message);

  }

});

//\*\*\*PERSONA\*\*\*

//create a persona

app.post("/persona", async (req, res) => {

  try {

    const { numero\_doc, nombre, apellidos, fecha\_nacimiento, tipo\_doc, edad, estatura, situacion\_militar, sexo, nivel\_de\_estudios, fk\_persona\_cod\_municipio } = req.body;

    const newPersona = await pool.query(

      "INSERT INTO persona (numero\_doc, nombre, apellidos, fecha\_nacimiento, tipo\_doc, edad, estatura, situacion\_militar, sexo, nivel\_de\_estudios, fk\_persona\_cod\_municipio) VALUES($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11) RETURNING \*",

      [numero\_doc, nombre, apellidos, fecha\_nacimiento, tipo\_doc, edad, estatura, situacion\_militar, sexo, nivel\_de\_estudios, fk\_persona\_cod\_municipio]

    );

    res.json(newPersona.rows[0]);

  } catch (err) {

    console.error(err.message);

  }

});

//get all persona

app.get("/persona", async (req, res) => {

  try {

    const allPersona = await pool.query("SELECT \* FROM persona");

    res.json(allPersona.rows);

  } catch (err) {

    console.error(err.message);

  }

});

//get a persona

app.get("/persona/:id", async (req, res) => {

  try {

    const { id } = req.params;

    const persona = await pool.query("SELECT \* FROM persona WHERE numero\_doc = $1", [

      id

    ]);

    res.json(persona.rows[0]);

  } catch (err) {

    console.error(err.message);

  }

});

//update a persona

app.put("/persona/:id", async (req, res) => {

  try {

    const { id } = req.params;

    const { nombre, apellidos, fecha\_nacimiento, tipo\_doc, edad, estatura, situacion\_militar, sexo, nivel\_de\_estudios } = req.body;

    const updatePersona = await pool.query(

      'UPDATE persona SET nombre=$1, apellidos=$2, fecha\_nacimiento=$3, tipo\_doc=$4, edad=$5, estatura=$6, situacion\_militar=$7, sexo=$8, nivel\_de\_estudios=$9 WHERE numero\_doc = $10',

      [nombre, apellidos, fecha\_nacimiento, tipo\_doc, edad, estatura, situacion\_militar, sexo, nivel\_de\_estudios, id]

    );

    res.json("Persona actualizada!");

  } catch (err) {

    console.error(err.message);

  }

});

//delete a persona

app.delete("/persona/:id", async (req, res) => {

  try {

    const { id } = req.params;

    const deletePersona = await pool.query("DELETE FROM persona WHERE numero\_doc = $1", [id]);

    res.json("Persona eliminada!");

  } catch (err) {

    console.log(err.message);

  }

});

//\*\*\*MUNICIPIO\*\*\*

//create a municipio

app.post("/municipio", async (req, res) => {

  try {

    const { cod\_municipio, nombre\_municipio, fk\_cod\_provincia } = req.body;

    const newMunicipio = await pool.query(

      "INSERT INTO municipio (cod\_municipio, nombre\_municipio, fk\_cod\_provincia) VALUES($1, $2, $3) RETURNING \*",

      [cod\_municipio, nombre\_municipio, fk\_cod\_provincia]

    );

    res.json(newMunicipio.rows[0]);

  } catch (err) {

    console.error(err.message);

  }

});

//get all municipio

app.get("/municipio", async (req, res) => {

  try {

    const allMunicipio = await pool.query("SELECT \* FROM municipio");

    res.json(allMunicipio.rows);

  } catch (err) {

    console.error(err.message);

  }

});

//get a municipio

app.get("/municipio/:id", async (req, res) => {

  try {

    const { id } = req.params;

    const municipio = await pool.query("SELECT \* FROM municipio WHERE cod\_municipio = $1", [

      id

    ]);

    res.json(municipio.rows[0]);

  } catch (err) {

    console.error(err.message);

  }

});

//update a municipio

app.put("/municipio/:id", async (req, res) => {

  try {

    const { id } = req.params;

    const { nombre\_municipio, fk\_cod\_provincia } = req.body; //

    const updateMunicipio = await pool.query(

      'UPDATE municipio SET nombre\_municipio=$1, fk\_cod\_provincia=$2 WHERE cod\_municipio = $3',

      [nombre\_municipio,fk\_cod\_provincia, id]

    );

    res.json("Municipio actualizado!");

  } catch (err) {

    console.error(err.message);

  }

});

//delete a municipio

app.delete("/municipio/:id", async (req, res) => {

  try {

    const { id } = req.params;

    const deleteMunicipio = await pool.query("DELETE FROM municipio WHERE cod\_municipio = $1", [id]);

    res.json("Municipio eliminado!");

  } catch (err) {

    console.log(err.message);

  }

});

//\*\*\*CENSO DERECHO\*\*\*

//create a censo\_derecho

app.post("/censo\_derecho", async (req, res) => {

  try {

    const { id\_censo\_derecho, fecha\_registro, direccion } = req.body;

    const newCensoDerecho = await pool.query(

      "INSERT INTO censo\_derecho (id\_censo\_derecho, fecha\_registro, direccion) VALUES($1, $2, $3) RETURNING \*",

      [id\_censo\_derecho, fecha\_registro, direccion]

    );

    res.json(newCensoDerecho.rows[0]);

  } catch (err) {

    console.error(err.message);

  }

});

//get all censo\_derecho

app.get("/censo\_derecho", async (req, res) => {

  try {

    const allCensoDerecho = await pool.query("SELECT \* FROM censo\_derecho");

    res.json(allCensoDerecho.rows);

  } catch (err) {

    console.error(err.message);

  }

});

//get a censo\_derecho

app.get("/censo\_derecho/:id", async (req, res) => {

  try {

    const { id } = req.params;

    const censo\_derecho = await pool.query("SELECT \* FROM censo\_derecho WHERE id\_censo\_derecho = $1", [

      id

    ]);

    res.json(censo\_derecho.rows[0]);

  } catch (err) {

    console.error(err.message);

  }

});

//update a censo\_derecho

app.put("/censo\_derecho/:id", async (req, res) => {

  try {

    const { id } = req.params;

    const { fecha\_registro, direccion } = req.body; //

    const updateCensoDerecho = await pool.query(

      'UPDATE censo\_derecho SET fecha\_registro=$1, direccion=$2 WHERE id\_censo\_derecho = $3',

      [fecha\_registro,direccion, id]

    );

    res.json("Censo Derecho actualizado!");

  } catch (err) {

    console.error(err.message);

  }

});

//delete a censo\_derecho

app.delete("/censo\_derecho/:id", async (req, res) => {

  try {

    const { id } = req.params;

    const deleteCensoDerecho = await pool.query("DELETE FROM censo\_derecho WHERE id\_censo\_derecho = $1", [id]);

    res.json("Censo Derecho eliminado!");

  } catch (err) {

    console.log(err.message);

  }

});

//\*\*\*CENSO HECHO\*\*\*

//create a censo\_hecho

app.post("/censo\_hecho", async (req, res) => {

  try {

    const { id\_censo\_hecho, fecha\_inicio\_residencia, direccion, pais\_residencia } = req.body;

    const newCensoHecho = await pool.query(

      "INSERT INTO censo\_hecho (id\_censo\_hecho, fecha\_inicio\_residencia, direccion, pais\_residencia) VALUES($1, $2, $3, $4) RETURNING \*",

      [id\_censo\_hecho, fecha\_inicio\_residencia, direccion, pais\_residencia]

    );

    res.json(newCensoHecho.rows[0]);

  } catch (err) {

    console.error(err.message);

  }

});

//get all censo\_hecho

app.get("/censo\_hecho", async (req, res) => {

  try {

    const allCensoHecho = await pool.query("SELECT \* FROM censo\_hecho");

    res.json(allCensoHecho.rows);

  } catch (err) {

    console.error(err.message);

  }

});

//get a censo\_hecho

app.get("/censo\_hecho/:id", async (req, res) => {

  try {

    const { id } = req.params;

    const censo\_hecho = await pool.query("SELECT \* FROM censo\_hecho WHERE id\_censo\_hecho = $1", [

      id

    ]);

    res.json(censo\_hecho.rows[0]);

  } catch (err) {

    console.error(err.message);

  }

});

//update a censo\_hecho

app.put("/censo\_hecho/:id", async (req, res) => {

  try {

    const { id } = req.params;

    const { fecha\_inicio\_residencia, direccion, pais\_residencia } = req.body; //

    const updateCensoHecho = await pool.query(

      'UPDATE censo\_hecho SET fecha\_inicio\_residencia=$1, direccion=$2, pais\_residencia=$3 WHERE id\_censo\_hecho = $4',

      [fecha\_inicio\_residencia, direccion, pais\_residencia, id]

    );

    res.json("Censo Hecho actualizado!");

  } catch (err) {

    console.error(err.message);

  }

});

//delete a censo\_hecho

app.delete("/censo\_hecho/:id", async (req, res) => {

  try {

    const { id } = req.params;

    const deleteCensoHecho = await pool.query("DELETE FROM censo\_hecho WHERE id\_censo\_hecho = $1", [id]);

    res.json("Censo Hecho eliminado!");

  } catch (err) {

    console.log(err.message);

  }

});

//\*\*\*DISTRITO\*\*\*

//create a distrito

app.post("/distrito", async (req, res) => {

  try {

    const { cod\_distrito, nombre\_distrito, fk\_cod\_municipio, fk\_cod\_provincia } = req.body;

    const newDistrito = await pool.query(

      "INSERT INTO distrito (cod\_distrito, nombre\_distrito, fk\_cod\_municipio, fk\_cod\_provincia) VALUES($1, $2, $3, $4) RETURNING \*",

      [cod\_distrito, nombre\_distrito, fk\_cod\_municipio, fk\_cod\_provincia]

    );

    res.json(newDistrito.rows[0]);

  } catch (err) {

    console.error(err.message);

  }

});

//get all distrito

app.get("/distrito", async (req, res) => {

  try {

    const allDistrito = await pool.query("SELECT \* FROM distrito");

    res.json(allDistrito.rows);

  } catch (err) {

    console.error(err.message);

  }

});

//get a distrito

app.get("/distrito/:id", async (req, res) => {

  try {

    const { id } = req.params;

    const distrito = await pool.query("SELECT \* FROM distrito WHERE cod\_distrito = $1", [

      id

    ]);

    res.json(distrito.rows[0]);

  } catch (err) {

    console.error(err.message);

  }

});

//update a distrito

app.put("/distrito/:id", async (req, res) => {

  try {

    const { id } = req.params;

    const { nombre\_distrito, fk\_cod\_municipio, fk\_cod\_provincia } = req.body; //

    const updateDistrito = await pool.query(

      'UPDATE distrito SET nombre\_distrito=$1, fk\_cod\_municipio=$2, fk\_cod\_provincia=$3 WHERE cod\_distrito = $4',

      [nombre\_distrito,fk\_cod\_municipio,fk\_cod\_provincia, id]

    );

    res.json("Distrito actualizado!");

  } catch (err) {

    console.error(err.message);

  }

});

//delete a distrito

app.delete("/distrito/:id", async (req, res) => {

  try {

    const { id } = req.params;

    const deleteDistrito = await pool.query("DELETE FROM distrito WHERE cod\_distrito = $1", [id]);

    res.json("Distrito eliminado!");

  } catch (err) {

    console.log(err.message);

  }

});

//\*\*\*CENSO DISTRITAL DERECHO\*\*\*

//create a censo\_distrital\_derecho

app.post("/censo\_distrital\_derecho", async (req, res) => {

  try {

    const { id\_censo\_derecho\_dis, id\_fecha\_registro, fk\_id\_censo\_derecho } = req.body;

    const newCensoDistritalDerecho = await pool.query(

      "INSERT INTO censo\_distrital\_derecho (id\_censo\_derecho\_dis, id\_fecha\_registro, fk\_id\_censo\_derecho) VALUES($1, $2, $3) RETURNING \*",

      [id\_censo\_derecho\_dis, id\_fecha\_registro, fk\_id\_censo\_derecho]

    );

    res.json(newCensoDistritalDerecho.rows[0]);

  } catch (err) {

    console.error(err.message);

  }

});

//get all censo\_distrital\_derecho

app.get("/censo\_distrital\_derecho", async (req, res) => {

  try {

    const allCensoDistritalDerecho = await pool.query("SELECT \* FROM censo\_distrital\_derecho");

    res.json(allCensoDistritalDerecho.rows);

  } catch (err) {

    console.error(err.message);

  }

});

//get a censo\_distrital\_derecho

app.get("/censo\_distrital\_derecho/:id", async (req, res) => {

  try {

    const { id } = req.params;

    const censo\_distrital\_derecho = await pool.query("SELECT \* FROM censo\_distrital\_derecho WHERE id\_censo\_derecho\_dis = $1", [

      id

    ]);

    res.json(censo\_distrital\_derecho.rows[0]);

  } catch (err) {

    console.error(err.message);

  }

});

//update a censo\_distrital\_derecho

app.put("/censo\_distrital\_derecho/:id", async (req, res) => {

  try {

    const { id } = req.params;

    const { id\_fecha\_registro, fk\_id\_censo\_derecho } = req.body; //

    const updateCensoDistritalDerechoo = await pool.query(

      'UPDATE censo\_distrital\_derecho SET id\_fecha\_registro=$1, fk\_id\_censo\_derecho=$2 WHERE id\_censo\_derecho\_dis = $3',

      [fecha\_registro, fk\_id\_censo\_derecho, id]

    );

    res.json("Censo Distrital Derecho actualizado!");

  } catch (err) {

    console.error(err.message);

  }

});

//delete a censo\_distrital\_derecho

app.delete("/censo\_distrital\_derecho/:id", async (req, res) => {

  try {

    const { id } = req.params;

    const deleteCensoDistritalDerecho = await pool.query("DELETE FROM censo\_distrital\_derecho WHERE id\_censo\_derecho\_dis = $1", [id]);

    res.json("Censo Distrital Derecho eliminado!");

  } catch (err) {

    console.log(err.message);

  }

});

//\*\*\*CENSO DISTRITAL HECHO\*\*\*

//create a censo\_distrital\_hecho

app.post("/censo\_distrital\_hecho", async (req, res) => {

  try {

    const { id\_censo\_hecho\_dis, id\_fecha\_registro\_hecho, fk\_id\_censo\_hecho } = req.body;

    const newCensoDistritalHecho = await pool.query(

      "INSERT INTO censo\_distrital\_hecho (id\_censo\_hecho\_dis, id\_fecha\_registro\_hecho, fk\_id\_censo\_hecho) VALUES($1, $2, $3) RETURNING \*",

      [id\_censo\_hecho\_dis, id\_fecha\_registro\_hecho, fk\_id\_censo\_hecho]

    );

    res.json(newCensoDistritalHecho.rows[0]);

  } catch (err) {

    console.error(err.message);

  }

});

//get all censo\_distrital\_hecho

app.get("/censo\_distrital\_hecho", async (req, res) => {

  try {

    const allCensoDistritalHecho = await pool.query("SELECT \* FROM censo\_distrital\_hecho");

    res.json(allCensoDistritalHecho.rows);

  } catch (err) {

    console.error(err.message);

  }

});

//get a censo\_distrital\_hecho

app.get("/censo\_distrital\_hecho/:id", async (req, res) => {

  try {

    const { id } = req.params;

    const censo\_distrital\_hecho = await pool.query("SELECT \* FROM censo\_distrital\_hecho WHERE id\_censo\_hecho\_dis = $1", [

      id

    ]);

    res.json(censo\_distrital\_hecho.rows[0]);

  } catch (err) {

    console.error(err.message);

  }

});

//update a censo\_distrital\_hecho

app.put("/censo\_distrital\_hecho/:id", async (req, res) => {

  try {

    const { id } = req.params;

    const { id\_censo\_hecho\_dis, id\_fecha\_registro\_hecho, fk\_id\_censo\_hecho } = req.body; //

    const updateCensoDistritalHechoo = await pool.query(

      'UPDATE censo\_distrital\_hecho SET id\_fecha\_registro\_hecho=$1, fk\_id\_censo\_hecho=$2 WHERE id\_censo\_hecho\_dis = $3',

      [id\_fecha\_registro\_hecho, fk\_id\_censo\_hecho, id]

    );

    res.json("Censo Distrital Hecho actualizado!");

  } catch (err) {

    console.error(err.message);

  }

});

//delete a censo\_distrital\_hecho

app.delete("/censo\_distrital\_hecho/:id", async (req, res) => {

  try {

    const { id } = req.params;

    const deleteCensoDistritalHecho = await pool.query("DELETE FROM censo\_distrital\_hecho WHERE id\_censo\_hecho\_dis = $1", [id]);

    res.json("Censo Distrital Hecho eliminado!");

  } catch (err) {

    console.log(err.message);

  }

});

// Server running

app.listen(5000, () => {

  console.log("Servidor corriendo en puerto 5000");

});

# Librerías o bibliotecas adicionales necesarias

Para el desarrollo de este aplicativo fue necesario usar un Framework de FrontEnd el cual fue Bootstrap en su versión 4.5, para realizar algunos diseños a las tablas, vistas y alertas entre otro. 

**Script con fragmentos de código que contengan las instrucciones más importantes del desarrollo, y su respectiva explicación detallada**

Entre fragmentos, tenemos el siguiente, en el que definimos la conexión a nuestra base de datos local PostgreSQL.

Nombre base de datos: **conteo\_ciudadano\_bd**

Usuario base de datos: **postgres**

Contraseña base de datos: **(en blanco)**

Enlace base de datos:

const Pool = require("pg").Pool;

const pool = new Pool({

    user: "postgres",

    password: "Montero.84",

    host: "localhost",

    port: 5432,

    database: "conteo\_ciudadano\_bd"

});

module.exports = pool;

**Integración de las Vistas con el Controlador y el Modelo**

Este trabajo es realizado en la renderización de los componentes de React.

Estos archivos son los que visualizan la información de cada una de las tablas que procesa la base de datos.

**Components**

/provincia/ListProvincia.js

import React, { Fragment, useEffect, useState } from "react";

import EditProvincia from "./EditProvincia";

import InputProvincia from "./InputProvincia";

const ListProvincia = () => {

  const [provincia, setProvincia] = useState([]);

  //delete provincia function

  const deleteProvincia = async id => {

    try {

      const deleteProvincia = await fetch(`http://localhost:5000/provincia/${id}`, {

        method: "DELETE"

      });

      setProvincia(provincia.filter(provincia => provincia.cod\_provincia !== id));

    } catch (err) {

      console.error(err.message);

    }

  };

  const getProvincia = async () => {

    try {

      const response = await fetch("http://localhost:5000/provincia");

      const jsonData = await response.json();

      setProvincia(jsonData);

    } catch (err) {

      console.error(err.message);

    }

  };

  useEffect(() => {

    getProvincia();

  }, []);

  console.log(provincia);

  return (

    <Fragment>

      {" "}

      <InputProvincia />

      {/\* <h1 className="text-center mt-5">PROVINCIA</h1> \*/}

      <table className="table mt-5 text-center">

        <thead>

          <tr>

            <th>Nombre Provincia</th>

            <th>Editar</th>

            <th>Eliminar</th>

          </tr>

        </thead>

        <tbody>

          {/\*<tr>

            <td>John</td>

            <td>Doe</td>

            <td>john@example.com</td>

          </tr> \*/}

          {provincia.map(provincia => (

            <tr key={provincia.cod\_provincia}>

              <td>{provincia.nombre\_provincia}</td>

              <td>

                <EditProvincia provincia={provincia} />

              </td>

              <td>

                <button

                  className="btn btn-danger"

                  onClick={() => deleteProvincia(provincia.cod\_provincia)}

                >

                  Eliminar

                </button>

              </td>

            </tr>

          ))}

        </tbody>

      </table>

    </Fragment>

  );

};

export default ListProvincia;

/provincia/InputProvincia.js

import React, { Fragment, useState } from "react";

const InputProvincia = () => {

  const [nombre\_provincia, setNombreProvincia] = useState("");

  const onSubmitForm = async e => {

    e.preventDefault();

    try {

      const body = { nombre\_provincia };

      const response = await fetch("http://localhost:5000/provincia", {

        method: "POST",

        headers: { "Content-Type": "application/json" },

        body: JSON.stringify(body)

      });

      window.location = "/provincia";

    } catch (err) {

      console.error(err.message);

    }

  };

  return (

    <Fragment>

      <h1 className="text-center mt-5">PROVINCIA</h1>

      <form className="d-flex mt-3" onSubmit={onSubmitForm}>

        <input

          type="text"

          className="form-control ml-5 mr-2"

          value={nombre\_provincia}

          placeholder="Nombre PROVINCIA a agregar..."

          onChange={e => setNombreProvincia(e.target.value)}

        />

        <button className="btn btn-success ml-3 mr-5">Agregar</button>

      </form>

    </Fragment>

  );

};

export default InputProvincia;

/provincial/EditProvincia.js

import React, { Fragment, useState } from "react";

const EditProvincia = ({ provincia }) => {

  const [nombre\_provincia, setNombreProvincia] = useState(provincia.nombre\_provincia);

  //edit nombre\_provincia function

  const updateNombreProvincia = async e => {

    e.preventDefault();

    try {

      const body = { nombre\_provincia };

      const response = await fetch(

        `http://localhost:5000/provincia/${provincia.cod\_provincia}`,

        {

          method: "PUT",

          headers: { "Content-Type": "application/json" },

          body: JSON.stringify(body)

        }

      );

      window.location = "/provincia";

    } catch (err) {

      console.error(err.message);

    }

  };

  return (

    <Fragment>

      <button

        type="button"

        className="btn btn-warning"

        data-toggle="modal"

        data-target={`#id${provincia.cod\_provincia}`}

      >

        Edit

      </button>

      {/\*

        id = id10

      \*/}

      <div

        className="modal"

        id={`id${provincia.cod\_provincia}`}

        onClick={() => setNombreProvincia(provincia.nombre\_provincia)}

      >

        <div className="modal-dialog">

          <div className="modal-content">

            <div className="modal-header">

              <h4 className="modal-title">Editar Provincia</h4>

              <button

                type="button"

                className="close"

                data-dismiss="modal"

                onClick={() => setNombreProvincia(provincia.nombre\_provincia)}

              >

                &times;

              </button>

            </div>

            <div className="modal-body">

              <input

                type="text"

                className="form-control"

                value={nombre\_provincia}

                onChange={e => setNombreProvincia(e.target.value)}

              />

            </div>

            <div className="modal-footer">

              <button

                type="button"

                className="btn btn-warning"

                data-dismiss="modal"

                onClick={e => updateNombreProvincia(e)}

              >

                Edit

              </button>

              <button

                type="button"

                className="btn btn-danger"

                data-dismiss="modal"

                onClick={() => setNombreProvincia(provincia.nombre\_provincia)}

              >

                Close

              </button>

            </div>

          </div>

        </div>

      </div>

    </Fragment>

  );

};

export default EditProvincia;

/persona/ListPersona.js

import React, { Fragment, useEffect, useState } from "react";

import EditPersona from "./EditPersona";

import InputPersona from "./InputPersona";

const ListPersona = () => {

  const [persona, setPersona] = useState([]);

  //delete persona function

  const deletePersona = async id => {

    try {

      const deletePersona = await fetch(`http://localhost:5000/persona/${id}`, {

        method: "DELETE"

      });

      setPersona(persona.filter(persona => persona.numero\_doc !== id));

    } catch (err) {

      console.error(err.message);

    }

  };

  const getPersona = async () => {

    try {

      const response = await fetch("http://localhost:5000/persona");

      const jsonData = await response.json();

      setPersona(jsonData);

    } catch (err) {

      console.error(err.message);

    }

  };

  useEffect(() => {

    getPersona();

  }, []);

  console.log(persona);

  return (

    <Fragment>

      {" "}

      <InputPersona />

      <table className="table mt-5 text-center">

        <thead>

          <tr>

            <th>Numero doc</th>

            <th>Nombre</th>

            <th>Apellidos</th>

            <th>Fecha nacim</th>

            <th>Tipo doc</th>

            <th>Edad</th>

            <th>Estatura</th>

            <th>Situacion milit</th>

            <th>Sexo</th>

            <th>Nivel estudios</th>

          </tr>

        </thead>

        <tbody>

          {/\*<tr>

            <td>John</td>

            <td>Doe</td>

            <td>john@example.com</td>

          </tr> \*/}

          {persona.map(persona => (

            <tr key={persona.numero\_doc}>

              <td>{persona.numero\_doc}</td>

              <td>{persona.nombre}</td>

              <td>{persona.apellidos}</td>

              <td>{persona.fecha\_nacimiento}</td>

              <td>{persona.tipo\_doc}</td>

              <td>{persona.edad}</td>

              <td>{persona.estatura}</td>

              <td>{persona.situacion\_militar}</td>

              <td>{persona.sexo}</td>

              <td>{persona.nivel\_de\_estudios}</td>

              <td>

                <EditPersona persona={persona} />

              </td>

              <td>

                <button

                  className="btn btn-danger"

                  onClick={() => deletePersona(persona.numero\_doc)}

                >

                  Eliminar

                </button>

              </td>

            </tr>

          ))}

        </tbody>

      </table>

    </Fragment>

  );

};

export default ListPersona;

/personaEditPersona.js

import React, { Fragment, useState } from "react";

const EditPersona = ({ persona }) => {

  const [nombre, setNombrePersona] = useState(persona.nombre);

  const [apellidos, setApellidosPersona] = useState(persona.apellidos);

  const [fecha\_nacimiento, setFechaNacimientoPersona] = useState(persona.fecha\_nacimiento);

  const [tipo\_doc, setTipoDocPersona] = useState(persona.tipo\_doc);

  const [edad, setEdadPersona] = useState(persona.edad);

  const [estatura, setEstaturaPersona] = useState(persona.estatura);

  const [situacion\_militar, setSituacionMilitarPersona] = useState(persona.situacion\_militar);

  const [sexo, setSexoPersona] = useState(persona.sexo);

  const [nivel\_de\_estudios, setNivelEstudiosPersona] = useState(persona.nivel\_de\_estudios);

  const [fk\_persona\_cod\_municipio, setFkPersonaCodMunicipio] = useState(persona.fk\_persona\_cod\_municipio);

  //edit nombre\_persona function

  const updateNombrePersona = async e => {

    e.preventDefault();

    try {

      const body = { nombre, apellidos, fecha\_nacimiento, tipo\_doc, edad, estatura, situacion\_militar, sexo, nivel\_de\_estudios, fk\_persona\_cod\_municipio };

      const response = await fetch(

        `http://localhost:5000/persona/${persona.numero\_doc}`,

        `http://localhost:5000/persona/${persona.nombre}`,

        `http://localhost:5000/persona/${persona.apellidos}`,

        `http://localhost:5000/persona/${persona.fecha\_nacimiento}`,

        `http://localhost:5000/persona/${persona.tipo\_doc}`,

        `http://localhost:5000/persona/${persona.edad}`,

        `http://localhost:5000/persona/${persona.estatura}`,

        `http://localhost:5000/persona/${persona.sexo}`,

        `http://localhost:5000/persona/${persona.nivel\_de\_estudios}`,

        `http://localhost:5000/persona/${persona.fk\_persona\_cod\_municipio}`,

        {

          method: "PUT",

          headers: { "Content-Type": "application/json" },

          body: JSON.stringify(body)

        }

      );

      window.location = "/persona";

    } catch (err) {

      console.error(err.message);

    }

  };

  return (

    <Fragment>

      <button

        type="button"

        className="btn btn-warning"

        data-toggle="modal"

        data-target={`#id${persona.numero\_doc}`}

      >

        Edit

      </button>

      <div

        className="modal"

        id={`id${persona.numero\_doc}`}

        onClick={() => setNombrePersona(persona.nombre)}

      >

        <div className="modal-dialog">

          <div className="modal-content">

            <div className="modal-header">

              <h4 className="modal-title">Editar Persona</h4>

              <button

                type="button"

                className="close"

                data-dismiss="modal"

                onClick={() => setNombrePersona(persona.nombre)}

              />

                <button

                type="button"

                className="close"

                data-dismiss="modal"

                onClick={() => setApellidosPersona(persona.apellidos)}

              />

                <button

                type="button"

                className="close"

                data-dismiss="modal"

                onClick={() => setFechaNacimientoPersona(persona.fecha\_nacimiento)}

              />

                <button

                type="button"

                className="close"

                data-dismiss="modal"

                onClick={() => setTipoDocPersona(persona.tipo\_doc)}

              />

                <button

                type="button"

                className="close"

                data-dismiss="modal"

                onClick={() => setEdadPersona(persona.edad)}

              />

                <button

                type="button"

                className="close"

                data-dismiss="modal"

                onClick={() => setEstaturaPersona(persona.estatura)}

              />

              <button

                type="button"

                className="close"

                data-dismiss="modal"

                onClick={() => setSituacionMilitarPersona(persona.situacion\_militar)}

              />

              <button

                type="button"

                className="close"

                data-dismiss="modal"

                onClick={() => setSexoPersona(persona.sexo)}

              />

              <button

                type="button"

                className="close"

                data-dismiss="modal"

                onClick={() => setNivelEstudiosPersona(persona.nivel\_de\_estudios)}

              />

              <button

                type="button"

                className="close"

                data-dismiss="modal"

                onClick={() => setFkPersonaCodMunicipio(persona.fk\_persona\_cod\_municipio)}

              />

                &times;

            </div>

            <div className="modal-body">

              Nombre:

              <input

                type="text"

                className="form-control"

                value={nombre}

                onChange={e => setNombrePersona(e.target.value)}

              />

              Apellidos:

              <input

                type="text"

                className="form-control"

                value={apellidos}

                onChange={e => setApellidosPersona(e.target.value)}

              />

              Fecha Nacimiento:

              <input

                type="text"

                className="form-control"

                value={fecha\_nacimiento}

                onChange={e => setFechaNacimientoPersona(e.target.value)}

              />

              Tipo documento:

              <input

                type="text"

                className="form-control"

                value={tipo\_doc}

                onChange={e => setTipoDocPersona(e.target.value)}

              />

              Edad:

              <input

                type="text"

                className="form-control"

                value={edad}

                onChange={e => setEdadPersona(e.target.value)}

              />

              Estatura:

              <input

                type="text"

                className="form-control"

                value={estatura}

                onChange={e => setEstaturaPersona(e.target.value)}

              />

              Situacion militar:

              <input

                type="text"

                className="form-control"

                value={situacion\_militar}

                onChange={e => setSituacionMilitarPersona(e.target.value)}

              />

              Sexo:

              <input

                type="text"

                className="form-control"

                value={sexo}

                onChange={e => setSexoPersona(e.target.value)}

              />

              Nivel de estudios:

              <input

                type="text"

                className="form-control"

                value={nivel\_de\_estudios}

                onChange={e => setNivelEstudiosPersona(e.target.value)}

              />

              Municipio:

              <input

                type="text"

                className="form-control"

                value={fk\_persona\_cod\_municipio}

                onChange={e => setFkPersonaCodMunicipio(e.target.value)}

              />

            </div>

            <div className="modal-footer">

              <button

                type="button"

                className="btn btn-warning"

                data-dismiss="modal"

                onClick={e => updateNombrePersona(e)}

              >

                Edit

              </button>

              <button

                type="button"

                className="btn btn-danger"

                data-dismiss="modal"

                onClick={() => setNombrePersona(persona.numero\_doc)}

              >

                Close

              </button>

            </div>

          </div>

        </div>

      </div>

    </Fragment>

  );

};

export default EditPersona;

/persona/InputPersona.js

import React, { Fragment, useState } from "react";

const InputPersona = () => {

  const [nombre\_provincia, setNombreProvincia] = useState("");

  const [nombre, setNombrePersona] = useState("");

  const [apellidos, setApellidosPersona] = useState("");

  const [fecha\_nacimiento, setFechaNacimientoPersona] = useState("");

  const [tipo\_doc, setTipoDocPersona] = useState("");

  const [edad, setEdadPersona] = useState("");

  const [estatura, setEstaturaPersona] = useState("");

  const [situacion\_militar, setSituacionMilitarPersona] = useState("");

  const [sexo, setSexoPersona] = useState("");

  const [nivel\_de\_estudios, setNivelEstudiosPersona] = useState("");

  const [fk\_persona\_cod\_municipio, setFkPersonaCodMunicipio] = useState("");

  const onSubmitForm = async e => {

    e.preventDefault();

    try {

      const body = { nombre, apellidos, fecha\_nacimiento, tipo\_doc, edad, estatura, situacion\_militar, sexo, nivel\_de\_estudios, fk\_persona\_cod\_municipio };

      const response = await fetch("http://localhost:5000/persona", {

        method: "POST",

        headers: { "Content-Type": "application/json" },

        body: JSON.stringify(body)

      });

      window.location = "/persona";

    } catch (err) {

      console.error(err.message);

    }

  };

  return (

    <Fragment>

      <h1 className="text-center mt-5">PERSONA</h1>

      <form className="dmt-3 form-group" onSubmit={onSubmitForm}>

        <input

          type="text"

          className="form-control m-2"

          value={nombre}

          placeholder="Nombre..."

          onChange={e => setNombrePersona(e.target.value)}

        />

        <input

          type="text"

          className="form-control m-2"

          value={apellidos}

          placeholder="Apellidos..."

          onChange={e => setApellidosPersona(e.target.value)}

        />

        <input

          type="text"

          className="form-control m-2"

          value={fecha\_nacimiento}

          placeholder="Fecha de nacimiento..."

          onChange={e => setFechaNacimientoPersona(e.target.value)}

        />

        <input

          type="text"

          className="form-control m-2"

          value={tipo\_doc}

          placeholder="Tipo de documento..."

          onChange={e => setTipoDocPersona(e.target.value)}

        />

        <input

          type="text"

          className="form-control m-2"

          value={edad}

          placeholder="Edad..."

          onChange={e => setEdadPersona(e.target.value)}

        />

        <input

          type="text"

          className="form-control m-2"

          value={estatura}

          placeholder="Estatura..."

          onChange={e => setEstaturaPersona(e.target.value)}

        />

        <input

          type="text"

          className="form-control m-2"

          value={situacion\_militar}

          placeholder="Situación militar..."

          onChange={e => setSituacionMilitarPersona(e.target.value)}

        />

        <input

          type="text"

          className="form-control m-2"

          value={sexo}

          placeholder="Sexo..."

          onChange={e => setSexoPersona(e.target.value)}

        />

        <input

          type="text"

          className="form-control m-2"

          value={nivel\_de\_estudios}

          placeholder="Nivel de estudios..."

          onChange={e => setNivelEstudiosPersona(e.target.value)}

        />

        <input

          type="text"

          className="form-control m-2"

          value={fk\_persona\_cod\_municipio}

          placeholder="Municipio..."

          onChange={e => setFkPersonaCodMunicipio(e.target.value)}

        />

        <button className="btn btn-success ml-3 mr-5">Agregar</button>

      </form>

    </Fragment>

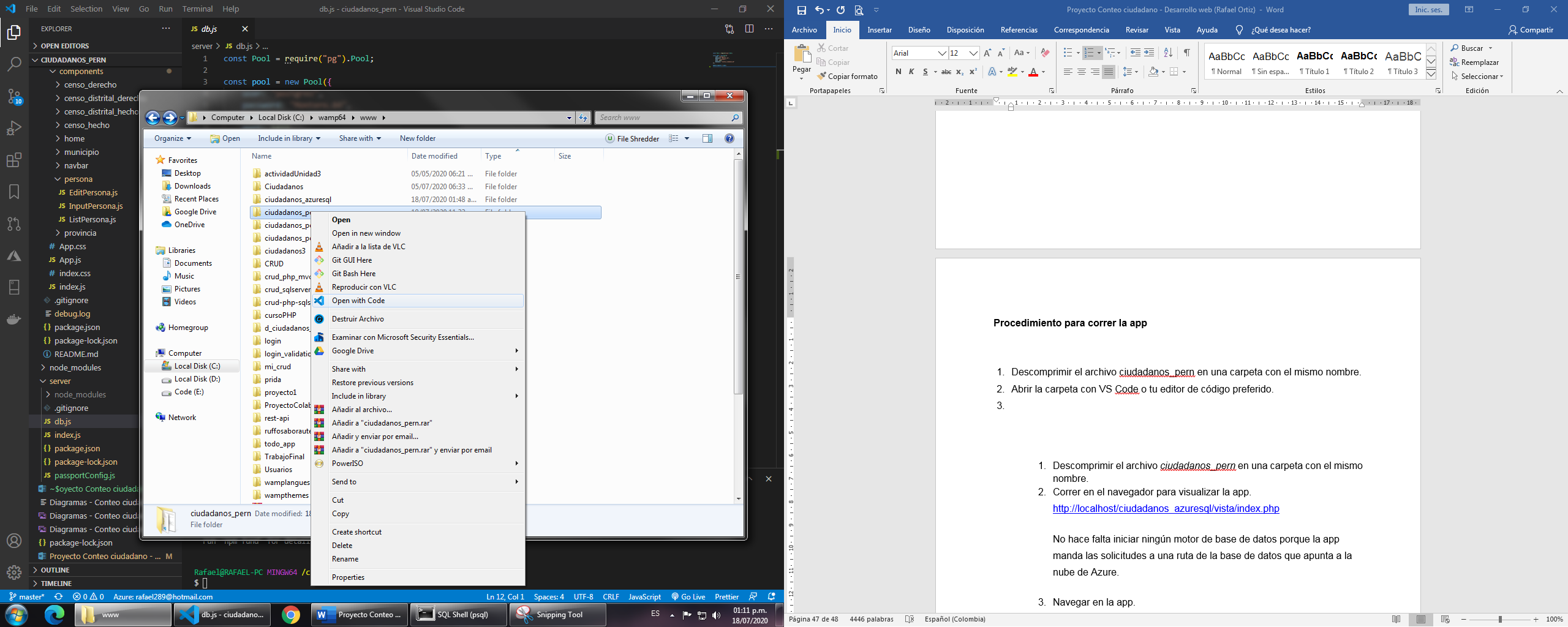
  );

};

export default InputPersona;

# Procedimiento para correr la app

1. Descomprimir el archivo ciudadanos\_pern en una carpeta con el mismo nombre.
2. Abrir la carpeta con VS Code o tu editor de código preferido.



# Descomprimir el archivo *ciudadanos\_pern* en una carpeta con el mismo nombre.

1. Correr en el navegador para visualizar la app. <http://localhost/ciudadanos_azuresql/vista/index.php>

No hace falta iniciar ningún motor de base de datos porque la app manda las solicitudes a una ruta de la base de datos que apunta a la nube de Azure.

1. Navegar en la app.

# 

# CONCLUSIONES

* Se aprendió a conectar la web app en lenguaje PHP para trabajar con una base de datos SQL de la nube de Azure.
* Se fusionó también lo aprendido en la asignatura Desarrollo de software web con esta, de manera que la web app realizará CRUD’s en la base de datos.
* Se aprendió a gestionar y administrar la base de datos a través del mismo dashboard de Azure y también con la herramienta Microsoft SQL Server Management Studio.
* Se aprendió a utilizar funciones y triggers para darle más dinamismo a la base de datos.
* Se aprendió el proceso de realizar backup y restauraciones de base de datos en el motor en mención.
* Se aprendió a conectar el proyecto local en una base de datos Online, Azure.