FACULTAD DE INGENIERÍA

INGENIERIA DE SISTEMAS

Manual de Creación

ARQUITECTURA DE DATOS

**AUTORES**

German Darío Sánchez Roa

Milton Fabian Ovalle García

**DOCENTE**

Juan Carlos Martínez Diaz

**NRC:**

10 – 3825

**Introducción**

La gestión eficiente de la información censal es crucial para la planificación de políticas públicas. Con este objetivo, se propone desarrollar una aplicación que administre los censos de los municipios de todo el país, registrando datos como el nombre del municipio, población total, cantidad de hombres, edad promedio, ingreso promedio y temperatura media del departamento.

Este proyecto está conformado por los integrantes German Darío Sánchez Roa y Milton Fabian Ovalle García; donde en cada registro de censo actualizará automáticamente los totales a nivel departamental y nacional, garantizando información precisa y actualizada. La aplicación permitirá agregar, editar y eliminar municipios, así como buscar información específica sobre departamentos y consultar datos censales según criterios de ingresos y población.

Con esta herramienta, se busca facilitar el acceso a datos relevantes que apoyen la toma de decisiones informadas y contribuyan al desarrollo social y económico del país.

**Desarrollo del Proyecto**

El proyecto se basa en una arquitectura de software que sigue el patrón Modelo-Vista-Controlador (MVC) y un enfoque de arquitectura cliente-servidor. A continuación, se describe detalladamente cómo se aplican estos modelos en el proyecto:

**1. Modelo-Vista-Controlador (MVC)**

**El patrón MVC divide el proyecto en tres componentes principales, lo que facilita la organización del código y su mantenimiento:**

* **Modelo (Model):** Este componente representa la lógica de datos y la comunicación con la base de datos. En el proyecto, los modelos se corresponden con las tablas **'Municipios' y 'Departamentos',** las cuales son gestionadas a través de funciones definidas en el archivo db\_connection.py. Estas funciones permiten realizar operaciones como guardar, modificar y eliminar datos, así como ejecutar consultas SQL para interactuar con la base de datos MySQL.
* **Vista (View):** Es la interfaz de usuario del proyecto. En este caso, la interfaz gráfica de **Tkinter** actúa como la vista, proporcionando al usuario un medio para interactuar con el sistema. A través de esta interfaz, los usuarios pueden llevar a cabo acciones como registrar municipios, modificar datos y eliminarlos. La vista se encarga de mostrar formularios, mensajes y otros elementos visuales para facilitar la interacción con el sistema**.**
* **Controlador (Controller):** Este componente coordina las acciones entre la vista y el modelo. En el proyecto, el controlador está compuesto por funciones que procesan las acciones del usuario en la interfaz gráfica y llaman a los modelos para ejecutar las operaciones correspondientes en la base de datos. Por ejemplo, las funciones **guardar\_municipio(), modificar\_municipio() y eliminar\_municipio()** actúan como controladores, gestionando las solicitudes del usuario y garantizando que las operaciones se realicen correctamente en la base de datos.

**2. Arquitectura Cliente-Servidor**

**Además del patrón MVC, el proyecto sigue una arquitectura cliente-servidor:**

* **Cliente:** La interfaz gráfica de **Tkinter** se ejecuta en el cliente, permitiendo al usuario realizar acciones como registrar municipios y visualizar resultados. Esta interfaz envía datos a los modelos, que luego interactúan con la base de datos para realizar las operaciones necesarias.
* **Servidor:** La lógica del servidor se basa en **Flask**, el cual se encarga de gestionar las operaciones relacionadas con la base de datos cuando el proyecto se implementa como una aplicación distribuida. **Flask** actúa como intermediario entre el cliente (la interfaz gráfica) y la base de datos. En un entorno local, tanto el cliente como el servidor se ejecutan en la misma máquina, pero en escenarios más avanzados, el servidor puede estar alojado en una máquina remota, permitiendo así una mayor escalabilidad.

**Cómo funciona en conjunto**

1. El usuario interactúa con la vista (la interfaz de Tkinter).
2. La vista llama a un controlador para llevar a cabo una acción (por ejemplo, guardar un municipio).
3. El controlador utiliza los modelos (que interactúan con MySQL) para realizar la operación solicitada.
4. Los resultados (como un mensaje de éxito o un error) se devuelven a la vista, que los muestra al usuario.

**Ventajas del enfoque**

* **Separación de responsabilidades:** La lógica de la base de datos, la lógica de negocio y la interfaz de usuario están claramente separadas, lo que facilita el desarrollo y mantenimiento del proyecto.
* **Facilidad de mantenimiento:** Las modificaciones en la interfaz gráfica no afectan a los modelos ni a los controladores, y viceversa,
* **Escalabilidad:** La arquitectura cliente-servidor permite trasladar el servidor a una máquina remota si es necesario en el futuro, mejorando la escalabilidad del sistema.

En resumen, el proyecto combina el patrón MVC para una organización interna eficiente y el enfoque cliente-servidor para una comunicación fluida entre componentes, resultando en un sistema modular, mantenible y escalable.

**Creación de la base de datos**

La implementación de la base de datos para la gestión de información censal de los municipios comienza con la creación de una base de datos llamada **CensoMunicipal**. Este paso es fundamental, ya que establece el entorno donde se almacenarán todos los datos relevantes.

Una vez creada la base de datos, se definen las tablas necesarias. La primera tabla es la de **Departamentos**, que almacena información clave sobre cada departamento, como su nombre, población total, cantidad de hombres y mujeres, edad promedio, ingreso promedio y temperatura media. Esta tabla es esencial para organizar los datos a nivel departamental.

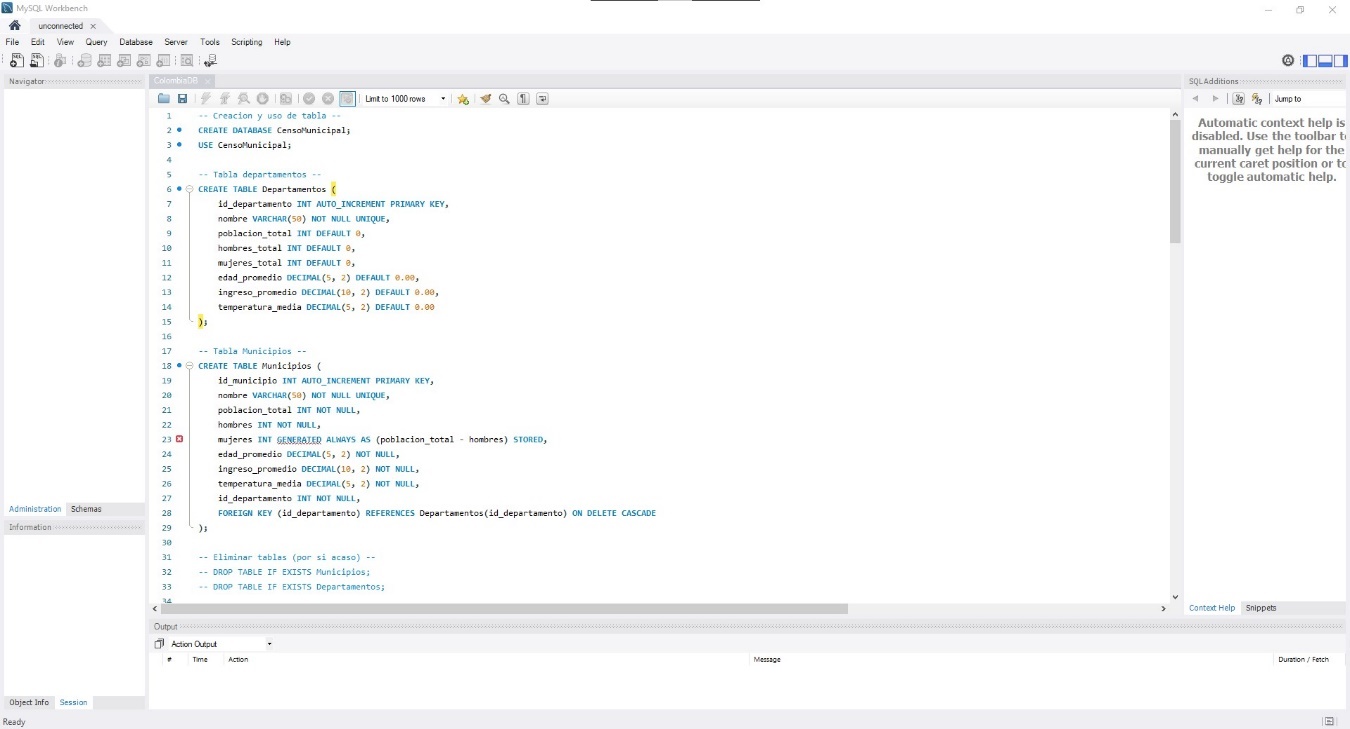
La segunda tabla es la de **Municipios**, que contiene datos específicos de cada municipio, incluyendo su nombre, población total, cantidad de hombres y un cálculo automático de la cantidad de mujeres. Esta tabla también se relaciona con la tabla de departamentos a través de una clave foránea, lo que permite mantener la integridad de los datos y facilitar las consultas.

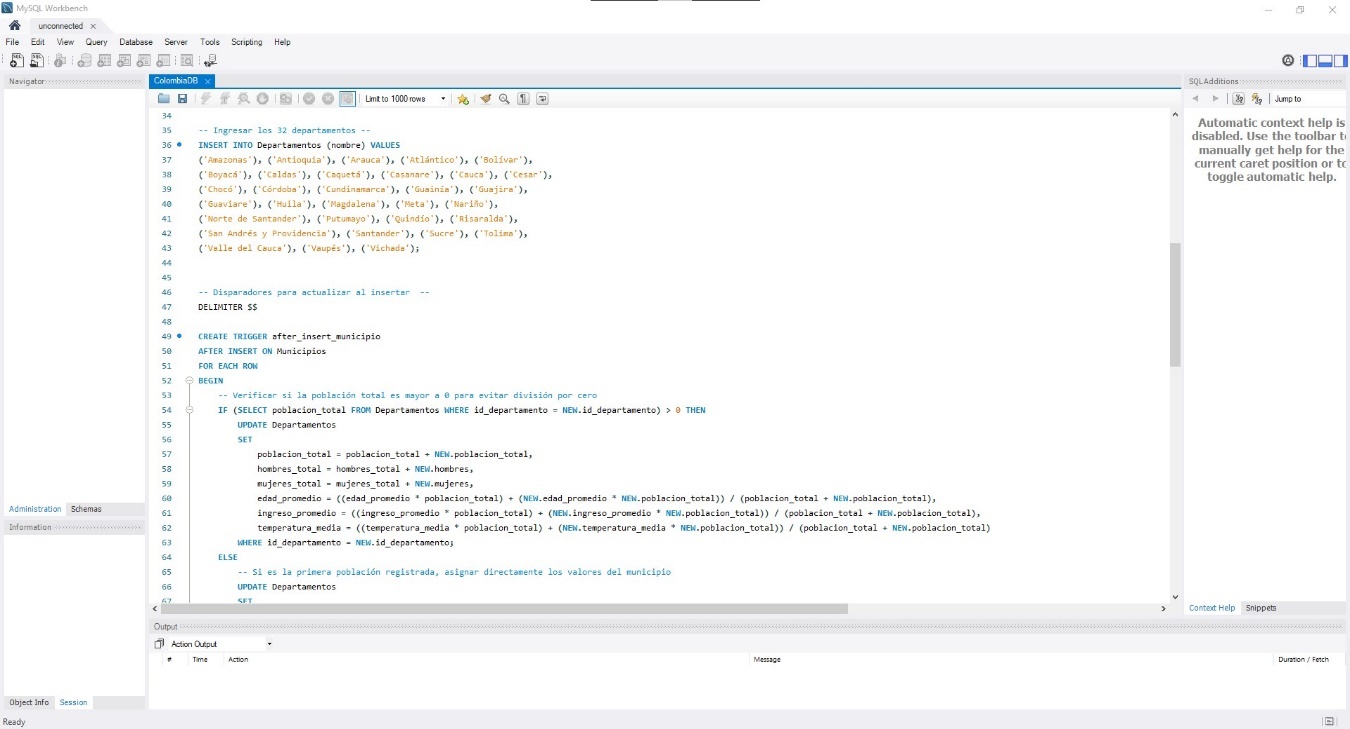
Para poblar la tabla de **Departamentos**, se insertan los nombres de los 32 departamentos del país. Este paso es crucial para tener un conjunto de datos inicial sobre el cual se podrán realizar análisis y consultas posteriores.

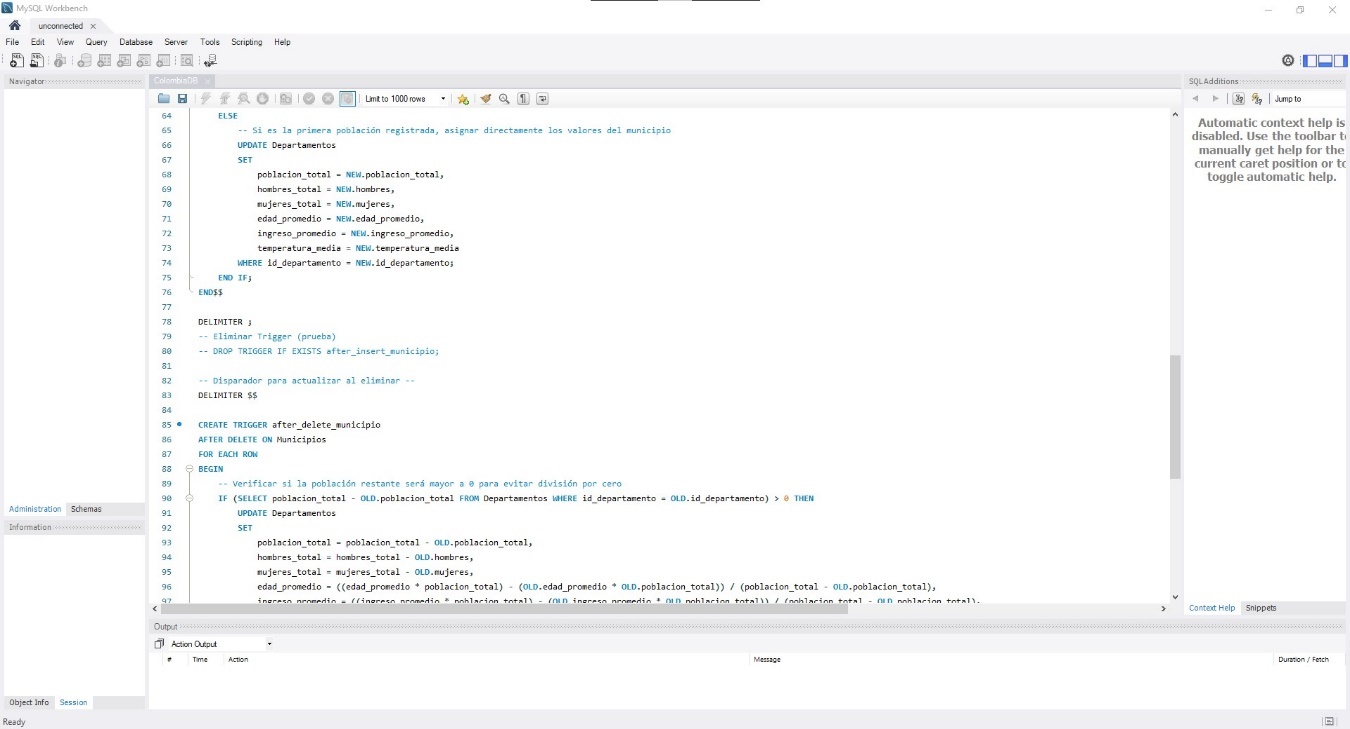
Además, se implementan **triggers** que permiten la actualización automática de la información en la tabla de **Departamentos** cada vez que se inserta o elimina un municipio. Esto asegura que los totales de población y otros datos se mantengan actualizados sin necesidad de realizar consultas manuales, lo que optimiza la gestión de la base de datos.

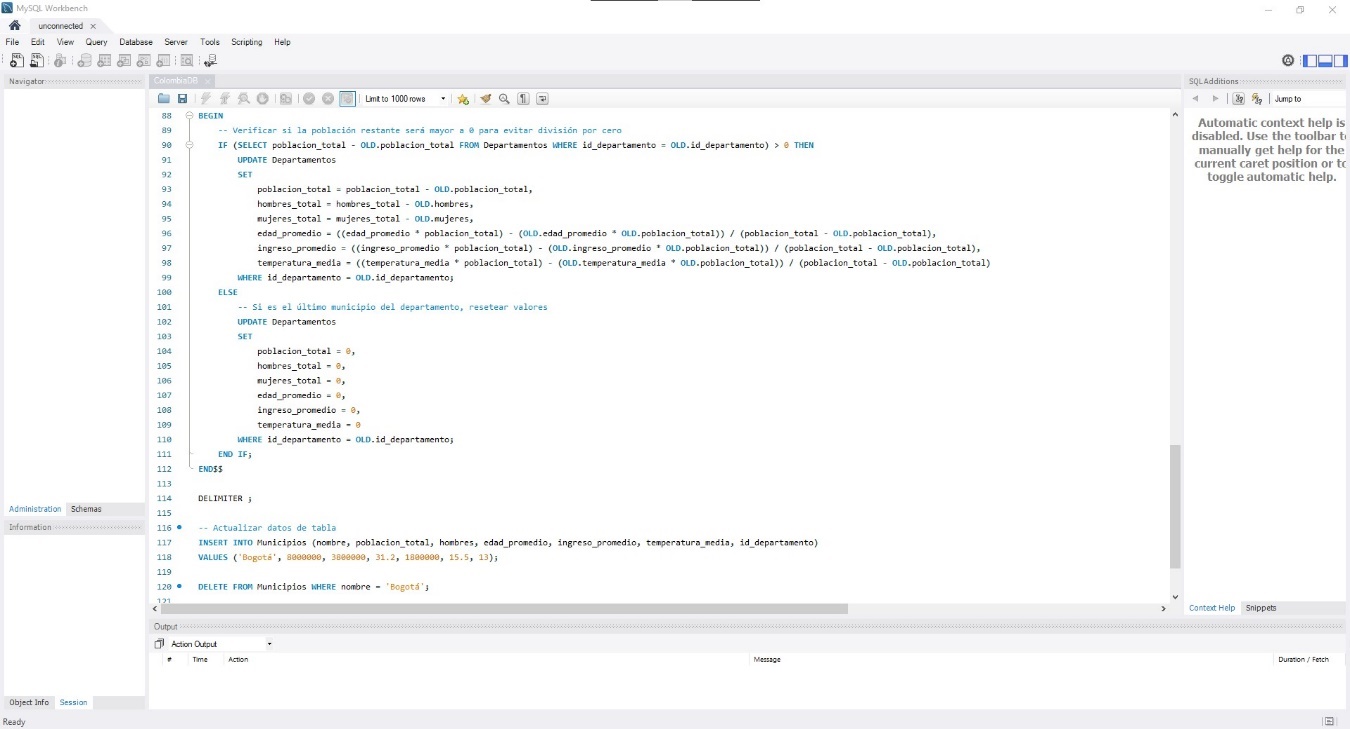
En resumen, la estructura de la base de datos está diseñada para facilitar la gestión de datos censales, permitiendo un acceso eficiente y actualizaciones automáticas que reflejan los cambios en la población de los municipios y departamentos. Esto contribuye a una mejor planificación y análisis de políticas públicas.

**Código de la BD**





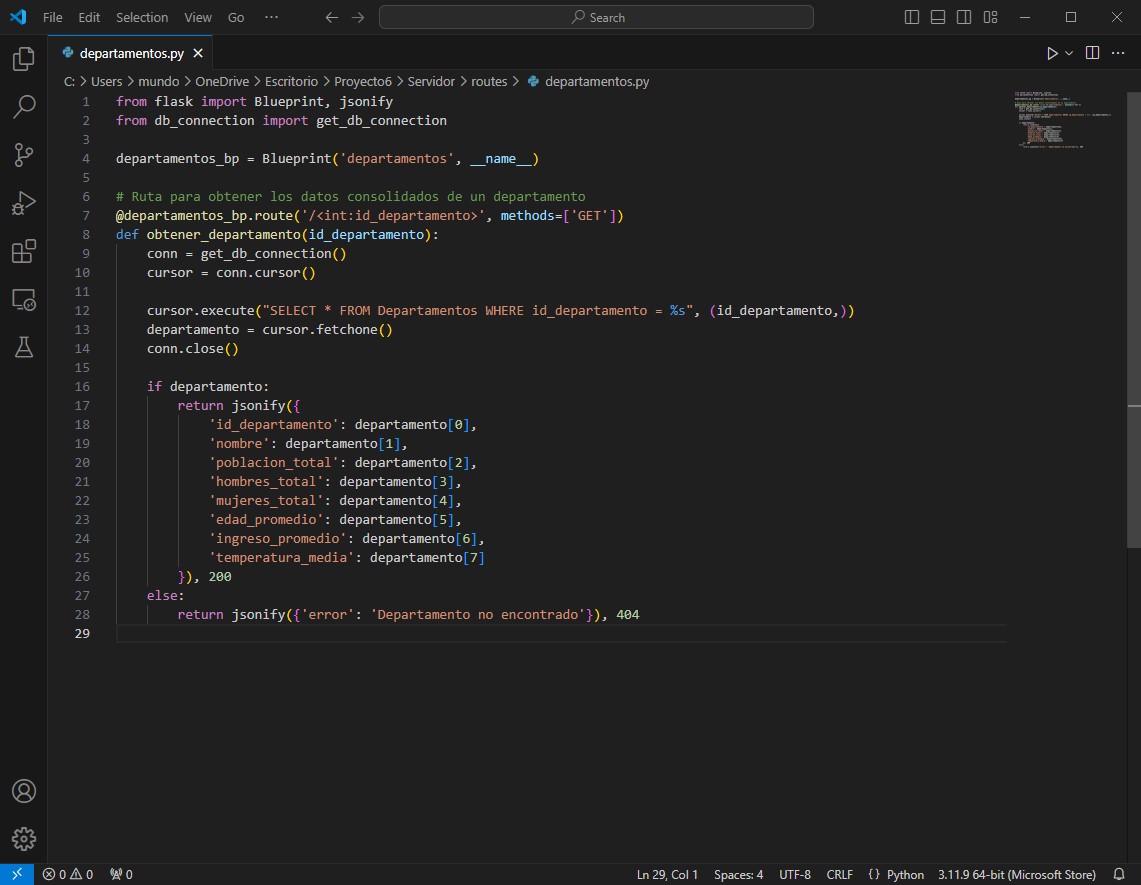




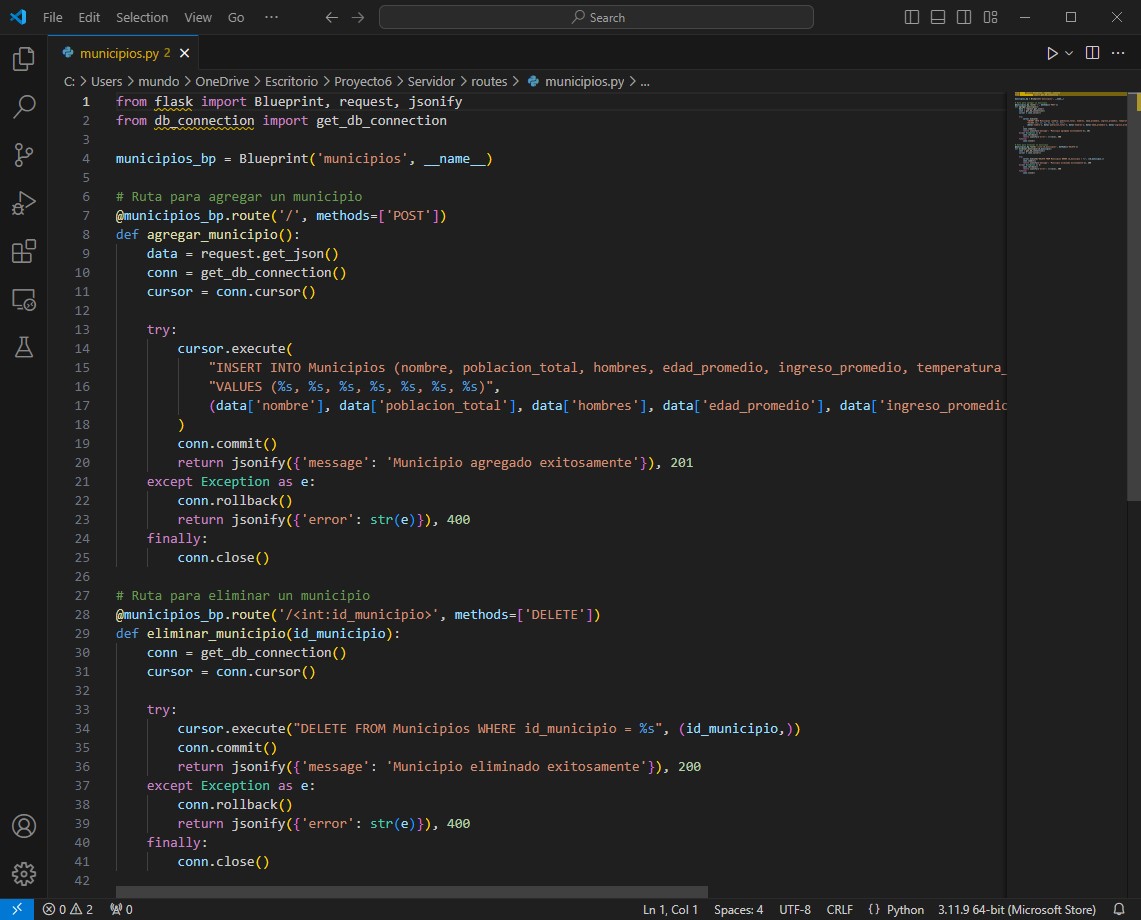
**Código fuente del servidor en Python**

Aquí tenemos dos archivos en el servidor

**Departamentos.py**

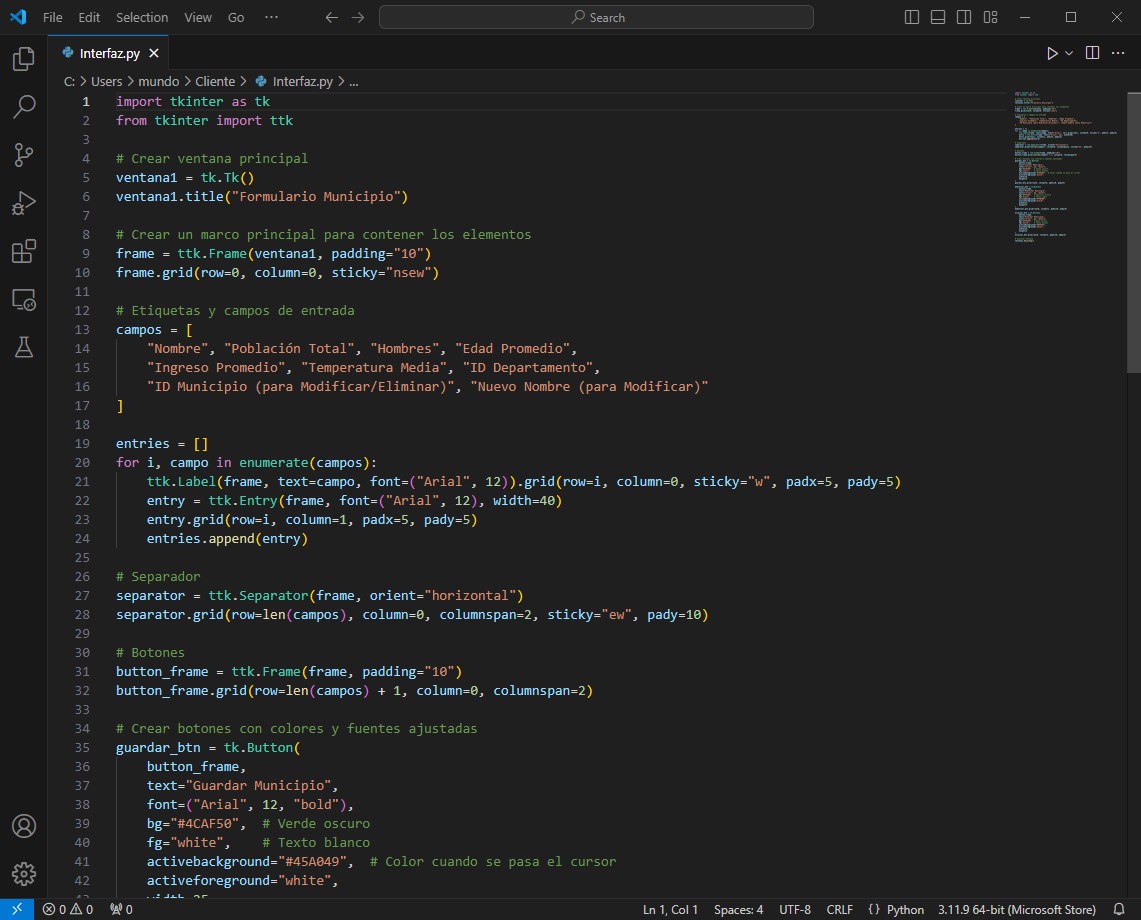
****

**Municipios.py**

****

**Código fuente del Cliente en Python**

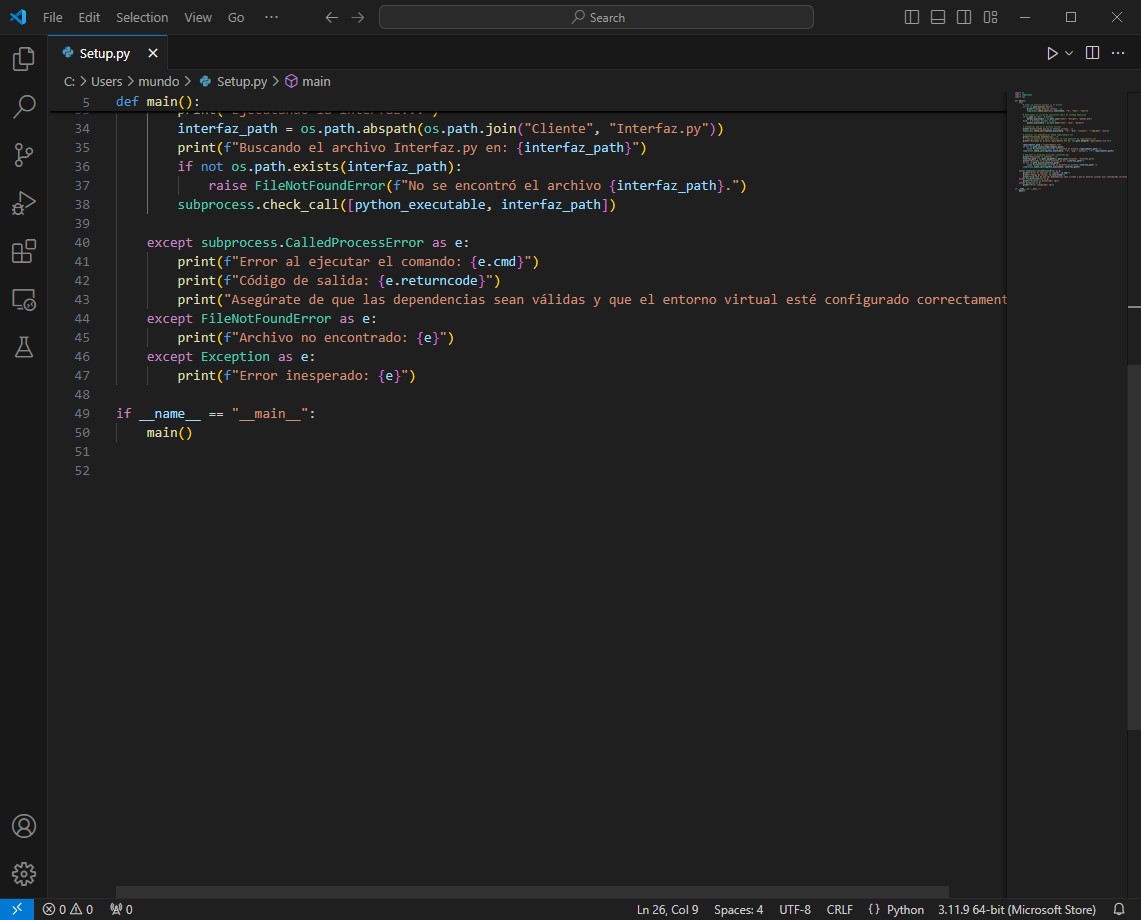
**Interfaz.py**

****

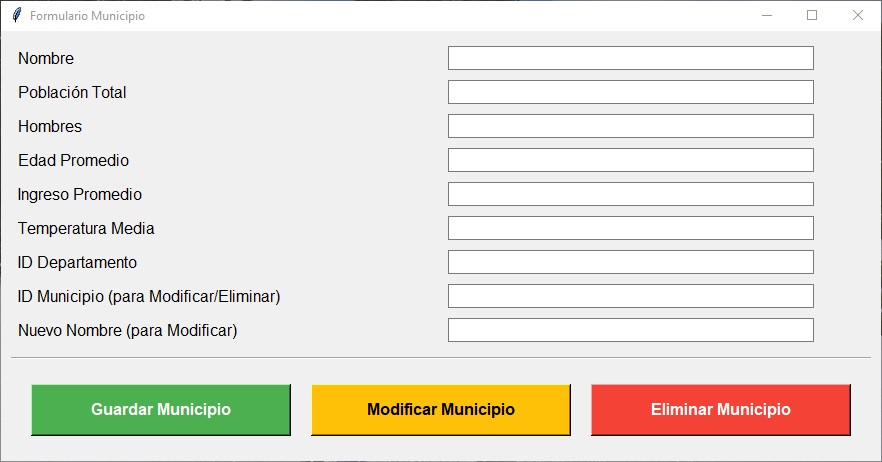
**Archivo Setup.py**

**Texto

Descripción generada automáticamente**



**Interfaz final**

****