**Universidad Mariano Gálvez de Guatemala**

**Ingeniería en Sistemas**

**Integrantes:**

Selvin Gabriel Escobar Zúñiga 1490-19-7106

Navideily Danely García de León 1490-19-4913

Emanuel Fernando Lopez Vargas 1490-19-7106

Jose Aaron Aguirre 1490-16-7811

Astri Suleyma Guox Calel 1490-19-7627

Boris Gonzalo Villagrán Cifuentes 1490-19-11141

**Administración de Tecnologías**

**Mayo 2024**

**Requerimientos de Instalación.**

**Visual Studio Code.**

Visual Studio Code es un editor de código fuente que se puede utilizar con una variedad de lenguajes de programación, incluidos C , C# , C++ , Fortran , Go , Java , JavaScript , Node.js , Python , Rust y Julia . Visual Studio Code emplea el mismo componente de edición (con nombre en código "Monaco") utilizado en Azure DevOps (anteriormente llamado "Visual Studio Online" y "Visual Studio Team Services").



**Python.**

Python es un lenguaje de alto nivel de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma.

Administrado por Python Software Foundation, posee una licencia de código abierto, denominada Python Software Foundation License. Python se clasifica constantemente como uno de los lenguajes de programación más populares.



**Flet.**

Framework de código abierto que facilita desarrollar aplicaciones web, móviles y de escritorio con Python. Flet es basado en Flutter, un framework para el desarrollo de interfaces de usuario multiplataforma de código abierto de Google.



**SQL Server**

Microsoft SQL Server es un sistema de gestión de base de datos relacional, desarrollado por la empresa Microsoft.

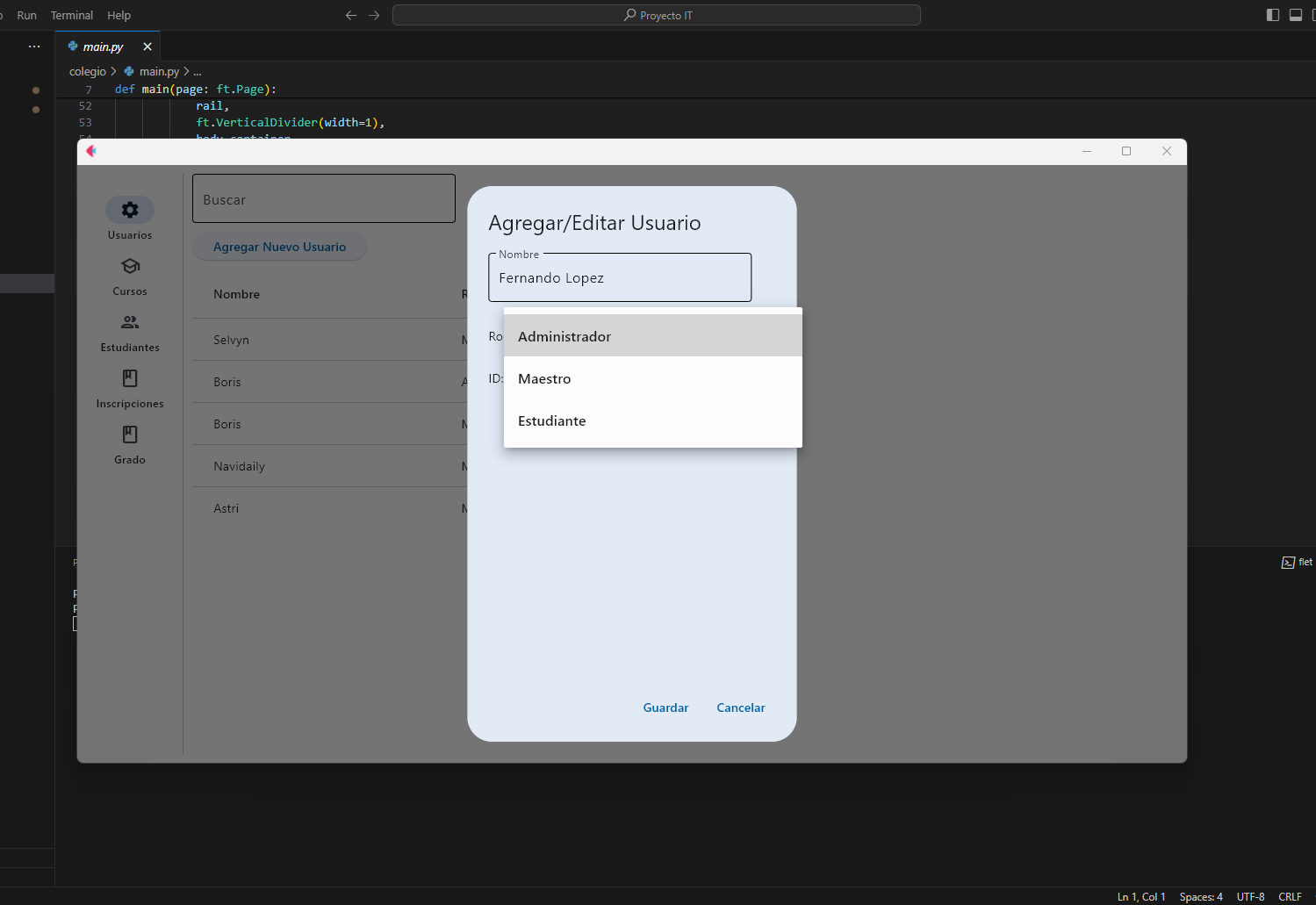
El lenguaje de desarrollo utilizado (por la línea de comandos o mediante la interfaz gráfica de Management Studio) es Transact-SQL (TSQL), una implementación del estándar ANSI del lenguaje SQL, utilizado para manipular y recuperar datos (DML), crear tablas y definir relaciones entre ellas (DDL).



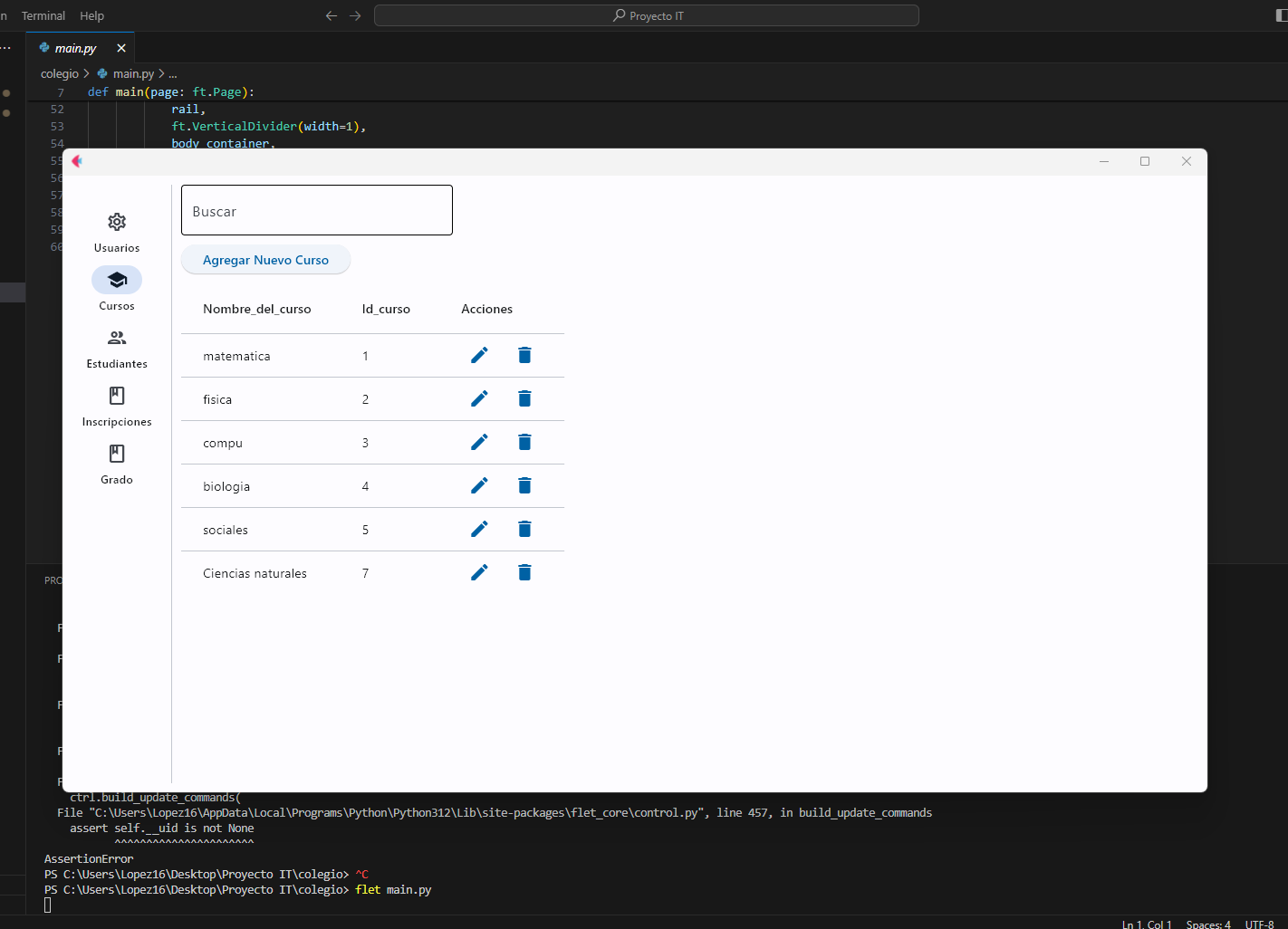
**Sistema.**

**Módulos.**

**Usuarios.** En este modulo podemos crear, editar y eliminar los usuarios que utilizara nuestro sistema, en este caso podemos asignarle los siguientes roles (Administrador, Maestro y Estudiante)



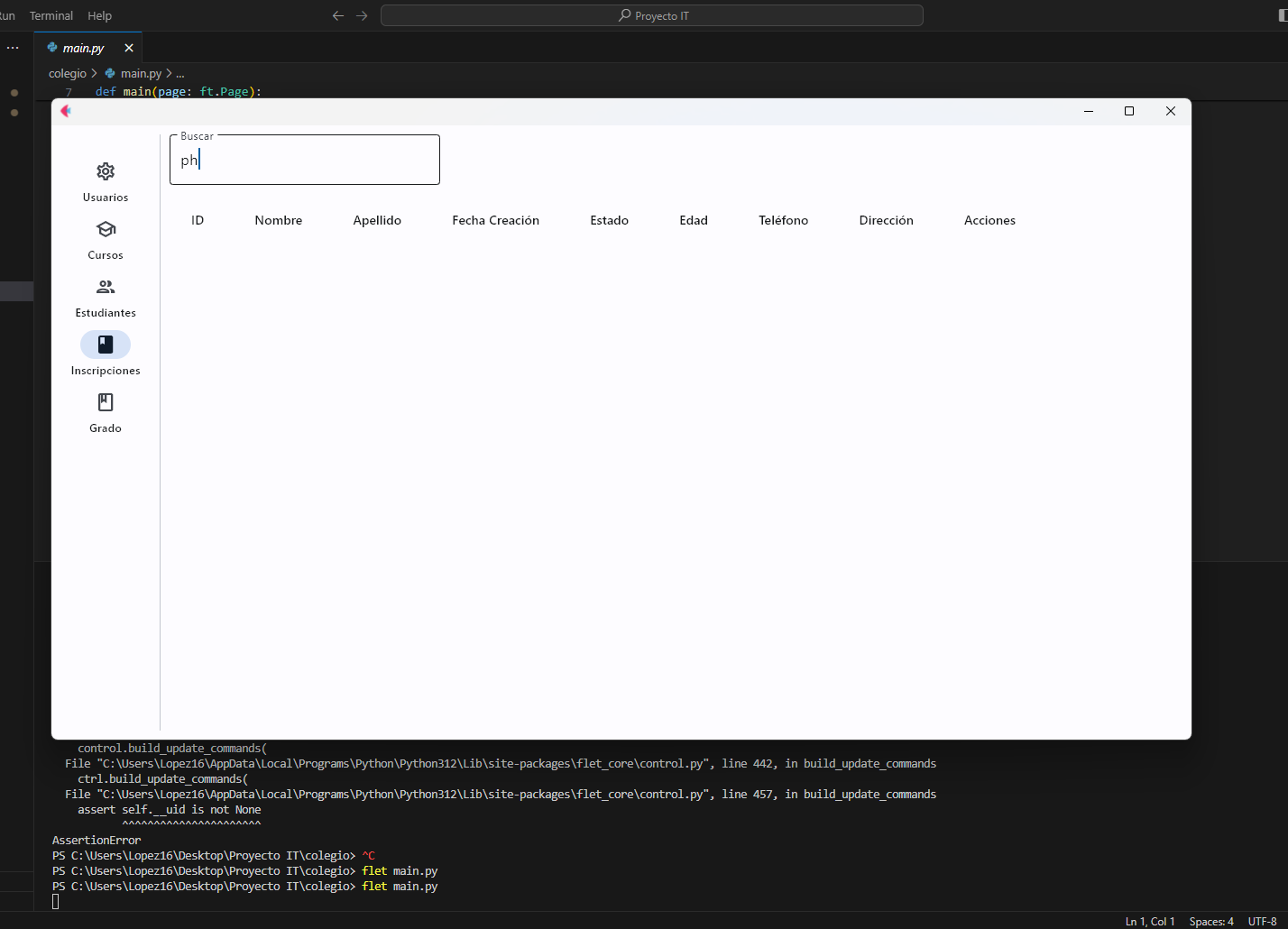
**Cursos.** En este módulo podemos crear, editar y eliminar los cursos o materias que utilice la institución.



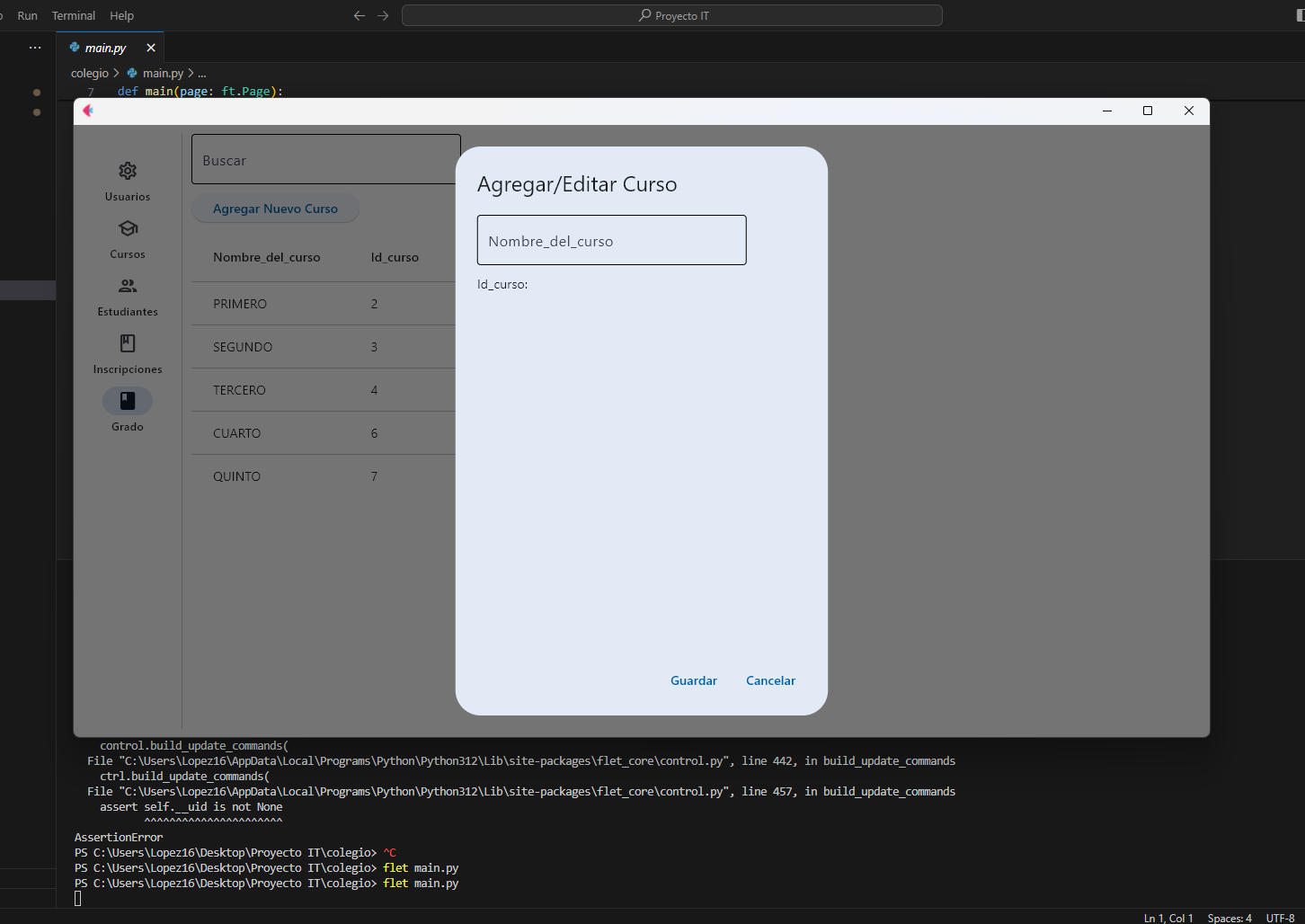
Una captura de pantalla de una computadora

Descripción generada automáticamente**Estudiantes.** En este módulo podemos crear, editar y eliminar los estudiantes del colegio.

**Inscripciones.** En este modulo podemos realizar una búsqueda de los estudiantes que tengamos creados.



**Grado.** En este modulo podemos crear los grados del colegio para agregar a cada estudiante a su respectivo grado.



**Código.**

**Modulo Usuarios**

Función para obtener la conexión a la base de datos.

def obtener\_conexion():

    conn\_str = (

        "Driver={SQL Server};"

        "Server=198.38.94.7;"

        "Database=COLEGIO;"

        "UID=Boris;"

        "PWD=Boris;"

    )

    return pyodbc.connect(conn\_str, timeout=5)

Función para agregar o actualizar la información de la tabla.

    # Función para guardar o actualizar la información del usuario

    def guardar\_usuario():

        if usuario\_seleccionado:

            # Actualizar usuario existente

            sql = "UPDATE Empleados SET Empleado = ?, Rol = ? WHERE ID = ?"

            val = (nombre\_txt.value, rol\_dropdown.value, usuario\_seleccionado['id'])

        else:

            # Insertar nuevo usuario

            ahora = datetime.now()

            formato = ahora.strftime('%Y-%m-%d %H:%M:%S.%f')[:-3]

            sql = "INSERT INTO Empleados (Empleado, Rol) VALUES (?, ?)"

            val = (nombre\_txt.value, rol\_dropdown.value)

        cursor.execute(sql, val)

        conn.commit()

        print(f"Registro {'actualizado' if usuario\_seleccionado else 'insertado'}.")

        load\_data()

        cerrar\_modal()

Función para cargar la vista principal de los usuarios.

# Vista principal de usuarios

def usuarios\_view(container):

    conn = obtener\_conexion()

    cursor = conn.cursor()

    usuario\_seleccionado = {}

    nombre\_txt = ft.TextField(label="Nombre", width=300)

    # Dropdown para seleccionar el rol

    rol\_dropdown = ft.Dropdown(

        width=300,

        options=[

            ft.dropdown.Option("Administrador"),

            ft.dropdown.Option("Maestro"),

            ft.dropdown.Option("Estudiante"),

            # Agrega aquí más opciones según necesites

        ]

    )

    id\_display = ft.Text(value="", width=300)

definición del modal para agregar/editar usuarios.

 # Definición del modal para agregar/editar usuarios

    modal = ft.AlertDialog(

        title=ft.Text("Agregar/Editar Usuario"),

        content=ft.Column([

            nombre\_txt,

            ft.Row([ft.Text("Rol:"), rol\_dropdown]),

            ft.Row([ft.Text("ID:"), id\_display])

        ]),

        actions=[

            ft.TextButton(text="Guardar", on\_click=lambda e: guardar\_usuario()),

            ft.TextButton(text="Cancelar", on\_click=lambda e: cerrar\_modal())

        ],

        open=False

    )

Función para guardar o actualizar la información del usuario

# Función para guardar o actualizar la información del usuario

    def guardar\_usuario():

        if usuario\_seleccionado:

            # Actualizar usuario existente

            sql = "UPDATE Empleados SET Empleado = ?, Rol = ? WHERE ID = ?"

            val = (nombre\_txt.value, rol\_dropdown.value, usuario\_seleccionado['id'])

        else:

            # Insertar nuevo usuario

            ahora = datetime.now()

            formato = ahora.strftime('%Y-%m-%d %H:%M:%S.%f')[:-3]

            sql = "INSERT INTO Empleados (Empleado, Rol) VALUES (?, ?)"

            val = (nombre\_txt.value, rol\_dropdown.value)

        cursor.execute(sql, val)

        conn.commit()

        print(f"Registro {'actualizado' if usuario\_seleccionado else 'insertado'}.")

        load\_data()

        cerrar\_modal()

    def delete\_curso(Id):

        cursor.execute("DELETE FROM Empleados WHERE id = ?", Id)

        conn.commit()

        load\_data()

Función para cargar los datos de la base de datos y actualizar la tabla.

    # Función para cargar los datos de la base de datos y actualizar la tabla

    def load\_data(filter=None):

        filter = filter or ''

        filter = "%" + filter + "%"  # SQL wildcard para búsqueda

        sql = "SELECT Empleado, Rol, ID FROM EMPLEADOS WHERE Empleado LIKE ?"

        cursor.execute(sql, filter)

        result = cursor.fetchall()

        mydt.rows.clear()

        for nombre, rol, id in result:

            row\_actions = ft.Row([

                ft.IconButton(icon=ft.icons.EDIT, on\_click=lambda e, nombre=nombre, rol=rol, id=id: abrir\_modal({'nombre': nombre, 'email': rol, 'id': id})),

                ft.IconButton(icon=ft.icons.DELETE, on\_click=lambda e, Id\_curso=id: delete\_curso(Id\_curso))

            ])

            mydt.rows.append(ft.DataRow(cells=[

                ft.DataCell(ft.Text(nombre)),

                ft.DataCell(ft.Text(rol)),

                ft.DataCell(ft.Text(str(id))),

                ft.DataCell(row\_actions)

            ]))

        container.update()

**Modulo Estudiantes.**

**Vista principal de los estudiantes.**

def estudiantes\_view(container):

    conn = obtener\_conexion()

    cursor = conn.cursor()

    curso\_seleccionado = {}

    id\_display = ft.Text(value="", width=300)

    Nombre\_del\_curso\_txt = ft.TextField(label="Nombre\_del\_curso", width=300)

    apellido\_txt = ft.TextField(label="Apellido", width=300)

    edad\_txt = ft.TextField(label="Edad", width=300)

    telefono\_txt = ft.TextField(label="Teléfono", width=300)

    direccion\_txt = ft.TextField(label="Dirección", width=300)

    search\_txt = ft.TextField(label="Buscar", width=300, autofocus=True)

    mydt = ft.DataTable(

        columns=[

            ft.DataColumn(ft.Text("Nombre")),

            ft.DataColumn(ft.Text("Apellido")),

            ft.DataColumn(ft.Text("Edad")),

            ft.DataColumn(ft.Text("Teléfono")),

            ft.DataColumn(ft.Text("Dirección")),

            ft.DataColumn(ft.Text("Acciones")),

        ],

        rows=[]

    )

**Para abrir el modal que carga los datos de un estudiante nuevo.**

def abrir\_modal(estudiante=None):

        curso\_seleccionado.clear()

        if estudiante:

            curso\_seleccionado.update(estudiante)

            Nombre\_del\_curso\_txt.value = estudiante['Nombre\_del\_curso']

            apellido\_txt.value = estudiante['apellido']

            id\_display.value = str(estudiante['Id\_curso'])

            # Aquí cargas los demás campos del estudiante en el modal

            edad\_txt.value = estudiante['edad']

            telefono\_txt.value = estudiante['telefono']

            direccion\_txt.value = estudiante['direccion']

        else:

            Nombre\_del\_curso\_txt.value = ""

            apellido\_txt.value = ""

            edad\_txt.value = ""

            telefono\_txt.value = ""

            direccion\_txt.value = ""

            id\_display.value = ""

        modal.open = True

        container.update()

    def cerrar\_modal():

        modal.open = False

        container.update()

**Función para cargar la tabla de estudiantes.**

    def load\_data(filter=''):

        filter = "%" + filter + "%"

        sql = "SELECT nombre, apellido, edad, telefono, direccion, id FROM tb\_estudiantes WHERE nombre LIKE ? OR apellido LIKE ?"

        cursor.execute(sql, (filter, filter))

        result = cursor.fetchall()

        mydt.rows.clear()

        for nombre, apellido, edad, telefono, direccion, id in result:

            row\_actions = ft.Row([

                ft.IconButton(icon=ft.icons.EDIT, on\_click=lambda e, Nombre\_del\_curso=nombre, apellido=apellido, Id\_curso=id: abrir\_modal({'Nombre\_del\_curso': Nombre\_del\_curso, 'apellido': apellido, 'Id\_curso': Id\_curso, 'edad': edad, 'telefono': telefono, 'direccion': direccion})),

                ft.IconButton(icon=ft.icons.DELETE, on\_click=lambda e, Id\_curso=id: delete\_curso(Id\_curso))

            ])

            mydt.rows.append(ft.DataRow(cells=[

                ft.DataCell(ft.Text(nombre)),

                ft.DataCell(ft.Text(apellido)),

                ft.DataCell(ft.Text(edad)),

                ft.DataCell(ft.Text(telefono)),

                ft.DataCell(ft.Text(direccion)),

                ft.DataCell(row\_actions)

            ]))

        container.update()

**Modulo de Inscripciones.**

**Función para crear el modal de inscripciones.**

    def abrir\_modal(estudiante=None):

        estudiante\_seleccionado.clear()

        if estudiante:

            estudiante\_seleccionado.update(estudiante)

            nombre\_txt.value = estudiante['nombre']

            apellido\_txt.value = estudiante['apellido']

            edad\_txt.value = estudiante['edad']

            telefono\_txt.value = estudiante['telefono']

            direccion\_txt.value = estudiante['direccion']

            id\_display.value = str(estudiante['id'])

            grado\_dropdown.value = estudiante.get('grado\_id', None)

        else:

            nombre\_txt.value = ""

            apellido\_txt.value = ""

            edad\_txt.value = ""

            telefono\_txt.value = ""

            direccion\_txt.value = ""

            id\_display.value = ""

            grado\_dropdown.value = None

        modal.open = True

        container.update()

**Funcion principal (main) principal.**

import flet as ft

from empleados import usuarios\_view

from cursos import cursos\_view

from estudiantes import estudiantes\_view

from Inscripciones import inscripciones\_view

from grado import grado\_view

def main(page: ft.Page):

    body\_container = ft.Column(expand=True)

    def handle\_nav\_change(e):

        body\_container.controls.clear()

        if e.control.selected\_index == 0:

            usuarios\_view(body\_container)

        elif e.control.selected\_index == 1:

            cursos\_view(body\_container)

        elif e.control.selected\_index == 2:

            estudiantes\_view(body\_container)

        elif e.control.selected\_index == 3:

            inscripciones\_view(body\_container)

        elif e.control.selected\_index == 4:

            grado\_view(body\_container)

        page.update()

    rail = ft.NavigationRail(

        selected\_index=0,

        label\_type=ft.NavigationRailLabelType.ALL,

        min\_width=100,

        min\_extended\_width=400,

        group\_alignment=-0.9,

        destinations=[

            ft.NavigationRailDestination(

                icon=ft.icons.SETTINGS\_OUTLINED, selected\_icon=ft.icons.SETTINGS, label="Usuarios"

            ),

            ft.NavigationRailDestination(

                icon=ft.icons.SCHOOL\_OUTLINED, selected\_icon=ft.icons.SCHOOL, label="Cursos"

            ),

            ft.NavigationRailDestination(

                icon=ft.icons.PEOPLE\_OUTLINED, selected\_icon=ft.icons.PEOPLE, label="Estudiantes"

            ),

            ft.NavigationRailDestination(

                icon=ft.icons.BOOK\_OUTLINED, selected\_icon=ft.icons.BOOK, label="Inscripciones"

            ),

             ft.NavigationRailDestination(

                icon=ft.icons.BOOK\_OUTLINED, selected\_icon=ft.icons.BOOK, label="Grado"

            ),

        ],

        on\_change=handle\_nav\_change

    )

    page.add(

        ft.Row([

            rail,

            ft.VerticalDivider(width=1),

            body\_container,

        ], expand=True)

    )

if \_\_name\_\_ == "\_\_main\_\_":

    ft.app(target=main)

**Enlace para descargar el Proyecto:** <https://drive.google.com/drive/folders/1koxefAbqEOhNZov-s4lGrGlAP4s6YfFR?usp=sharing>