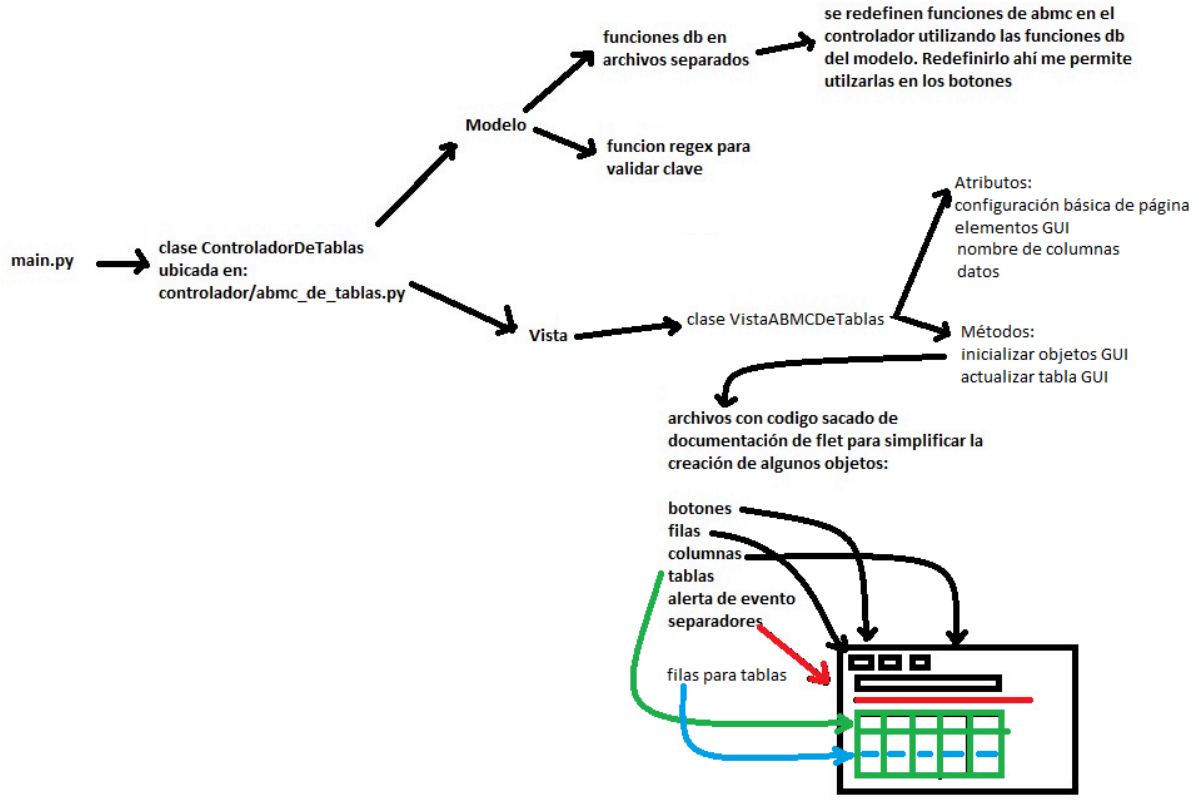
Documentación de Trabajo Práctico Python Nivel Inicial

Federico dos Santos

Entrega (23/06/2025)

Índice

Esquema Modelo/Vista/Controlador



El esquema corresponde al diseño MVC que utilicé para modelar un controlador que pueda manejar las funciones ABMC de una tabla de SQLITE3 y manejar una interfaz gráfica hecha con flet.

Controlador

En el programa main.py se instancia un controlador ABMC de tablas.

Atributos

En un principio pensé en ponerle como atributo la conexión a la base de datos, pero probando se me bloqueaba así que le dejé solamente nombre de base de datos y nombre de tabla para que luego llame a las funciones de sqlite3 y abra y cierre la conexión. También agregué los campos y tipo de dato que se requieren para crear la tabla.

El controlador también tiene una vista de la clase VistaABMCDeTablas. Esta se encargará de crear los objetos GUI y actualizar la tabla mediante métodos.

Métodos

Inicializar()

En el método inicializar se definen las funciones **ABM** haciendo uso de las que ya tenía funcionando previamente en la carpeta modelo. Estas nuevas funciones son necesarias para los botones que necesitan una función sin parámetros.
Entonces por ejemplo:
alta(e) utiliza la función insertar_datos(nombre_base, nombre_tabla,datos) pero la ventaja que tengo es que alta puede ser utilizada en los botones de la GUI.

baja(e) y modificacion(e) hacen lo mismo con sus funciones homólogas.
alerta(e) agrega un objeto de la vista (la alerta de evento) a la página y lo muestra. Para hacer uso previamente se modifica el título del alerta de evento para que tenga sentido mostrar la alerta.

Luego de definir esas funciones ABM y definir la función alerta, se crea la base, se crea la tabla y se consultan los datos tanto como los nombres de las columnas, ya que ambos van a ser de utilidad para la vista, para que construya la tabla GUI.
Después se inicializan los objetos GUI con el método de la vista inicializar_vista() que requiere que previamente estén definidas las funciones ABM y se pasen por parámetro para crear los botones GUI y también que estén los nombres y datos de las columnas para crear y rellenar la tabla GUI.

actualizar_vista()

Consulta los datos nuevamente pisando los que estaban previamente cargados como atributos de la vista.
Llama al método actualizar_tabla() de la vista para que borre las filas de la tabla y las cree de nuevo.

Modelo

SQLITE3

El controlador interactúa con funciones de bases de datos de SQLITE3. Utilicé las que vimos en clase, con unas pequeñas modificaciones para que funcionen de manera genérica, ya que mi objetivo es que luego se pueda interactuar con otras tablas, no solo con una. De forma tal que instanciando nuevos controladores de la clase ControladorDeTablas con otros nombres de tabla/campos pueda manejar otras tablas de SQLITE3.

Si bien el modelo tiene parametrizadas las funciones de base de datos, la baja y modificación requieren que haya una columna llamada ID en la tabla. Entonces puedo crear tablas con distintos nombres y distintas columnas, pero todas deben tener una columna llamada ID.

En cuanto a manejo de errores, en un principio había utilizado manejo de errores en todas las funciones de base de datos, pero luego me di cuenta que las alertas de evento debía manejarlas en el controlador para poder hacer uso de la interfaz gráfica para notificar si hubo un problema. Entonces de algunas funciones de base de datos tuve que quitar el manejo de errores y hacerlo desde la función alta/baja/modificación definidas en el controlador.
Luego vi que si intentaba agregar un registro con id que ya existe, el manejo de excepciones me permite informar que no se puede, pero con baja y modificación tuve que implementar una validación con una consulta a la base de datos para saber si el id a modificar o borrar no existe y poder informarlo.

REGEX

La función regex que creé es simplemente para validar la clave (id) de cada tabla. Usé las siguiente expresión regular: [0-9]{1,10}
La misma es para validar que sea un número entero de al menos 1 dígito y hasta menos de 10 dígitos porque googleando vi que el máximo de un integer es 2.147.483.647 que tiene 10 dígitos pero no los aprovecha hasta el 9.999.999.999.

Vista

En el controlador, archivo /controlador/abmc_de_tablas.py, en la clase ControladorDeTablas defino un atributo llamado vista que instancia un objeto de la clase **VistaABMCDeTablas** definida en **/vista/vista_abmc_tabla.py**

Atributos

Entre los primeros atributos definí la **página** y seteo los atributos de la misma (título, scroll) y actualizo la página.
Después para que funcione pensé que necesito llenar de **datos** la tabla, también conocer los **nombres de las columnas** para la tabla y para el registro modificable. Entonces le puse ambos atributos a la vista.
Después definí los objetos que va a tener visualmente la GUI, los objetos de flet.
Va a tener una **fila de botones**, un **registro modificable**, una **tabla**, un **separador** que separe en parte superior y parte inferior y una alerta de evento que mientras no ocurra ninguno lo inicializo con título vacío.

Métodos

Inicialmente no le había definido métodos, solo tenía atributos con configuración básica de la página y los objetos GUI los creaba desde el método inicializar del controlador. Pero después pensé que si el controlador conoce la vista, le puede pasar las funciones de los botones por parámetro, entonces podía pasar la instanciación esos objetos de flet del controlador a la vista.

Entonces los primeros métodos fueron para inicializar los objetos:

inicializar_tabla

Instancia la tabla GUI llamando a la función que hice generar_tabla con código copiado de *https://github.com/flet-dev/examples/blob/main/python/apps/controls-gallery/examples/layout/datatable/01_basic_datatable.py*

inicializar_botones

Instancia los botones GUI con sus respectivas funciones y después instancia una fila con estos botones guardando esta fila en el atributo botones. Hace uso de la función generar_boton (abstrae la clase OutlinedButton de flet) y de la función generar_filas del archivo /vista/generar_filas.py que copia código de *https://flet.dev/docs/controls/container/*

inicializar_registro_modificable

Instancia campos de texto de la clase TextField que son campos modificables, les asigna el título de las columnas que tiene la tabla sql que están guardados en el atributo de la vista titulos_columnas.
Después genera una fila con estos campos de texto modificable usando la función generar_filas.

inicializar_vista

llama a los métodos:

- inicializar_botones
- inicializar_registro_modificable
- inicializar_tabla

Genera una columna con la fila de botones y la fila de registro modificable. Se guarda en una variable llamada formulario. Hago uso de la función generar_columnas que copia código de *https://flet.dev/docs/controls/column*

Llamo a la función generar_separador que tiene un código que tomé y modifiqué del siguiente enlace
https://github.com/flet-dev/examples/blob/main/python/apps/controls-gallery/examples/layout/divider/02_draggable_divider.py

Finalmente agrego el objeto separador a la página. Este objeto contiene el formulario (una columna con fila de botones y fila de campos de texto modificables) como parte superior y la tabla GUI como parte inferior

actualizar_tabla

Borra las filas de la tabla, después crea nuevas filas (filas de tabla, no filas comunes) usando la función generar_filas del archivo /vista/generar_filas_para_tablas.py
Estas filas de tabla se guardan en el atributo rows de la tabla GUI y luego se actualiza la página.
Como mencioné en el método actualizar_vista del controlador, primero se cargan los datos nuevos en el atributo datos de la vista, después se llama al método actualizar_tabla (de la vista) para que borre las filas de la tabla GUI y cargue nuevas filas de tabla GUI con los datos nuevos.