**Documentación y guía paso a paso de Sistema ABM Librería**

Tabla de contenido

1. Introduccion
   1. Alcance
   2. Comportamiento
   3. Maqueta (Previsualización)
2. Reproducción paso a paso del sistema
   1. Introducción de herramientas a utilizar
   2. Configuración del área de trabajo
   3. Creación de la estructura MVC
   4. Desarrollo de apartado visual (vista.py)
      1. Sector Principal
      2. Listado de registros
      3. Sector de búsqueda/filtro
3. **Introduccion**

1.1 Alcance

La finalidad de la aplicación a desarrollar es la de mantener un control de los libros e historietas presentes dentro de una librería.

Para esto se pensó una aplicación en donde se ingrese:

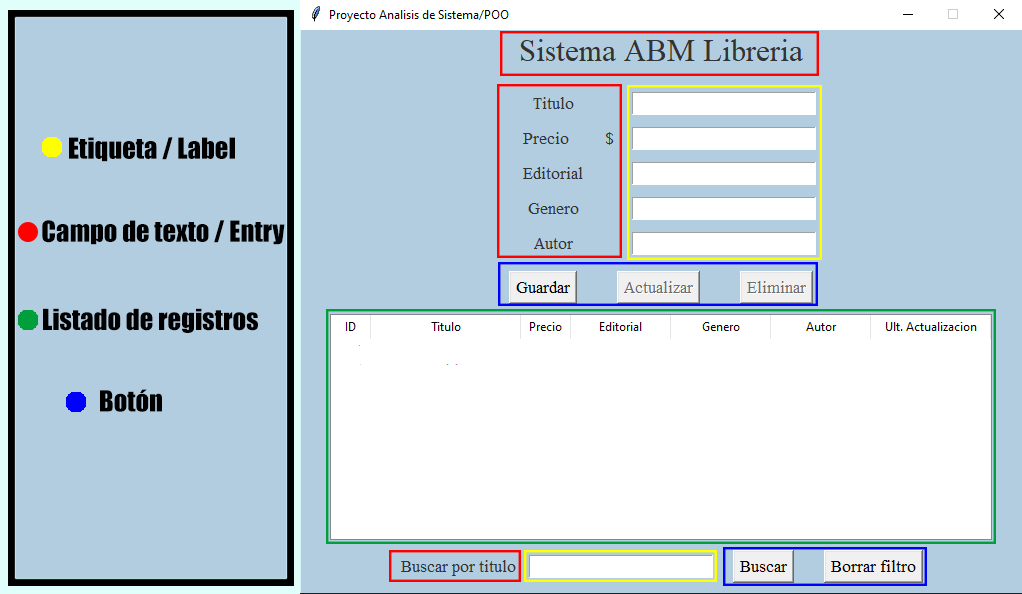
* Titulo: Titulo comercial del libro
* Precio: Valor en pesos del producto
* Editorial: La empresa que produjo ese libro
* Genero: Genero literario del libro
* Autor: Escritor del libro

1.2 Comportamiento

La aplicación se compone de seis campos de ingreso de texto junto a seis “labels” o etiquetas que se encargan de describir cada uno de estos campos, a su vez, la aplicación cuenta con un listado en donde se exhiben todos los registros guardados y una serie de cinco botones que se comportan según se describe a continuación:

* Guardar: Almacena los datos ingresados en los campos de entrada dentro de la Base de Datos.
* Actualizar: Edita y guarda un registro seleccionado desde la lista de libros.
* Eliminar: Elimina desde la Base de Datos al registro seleccionado en el listado.
* Buscar: Muestra un listado de registros que tengan un titulo parecido o igual al ingresado dentro del campo de texto “Buscar por titulo”.
* Borrar filtro: Muestra el listado completo de registros y limpia el campo de texto “Buscar por titulo”

1.3 Maqueta (Previsualización)



1. **Reproducción paso a paso del Sistema**

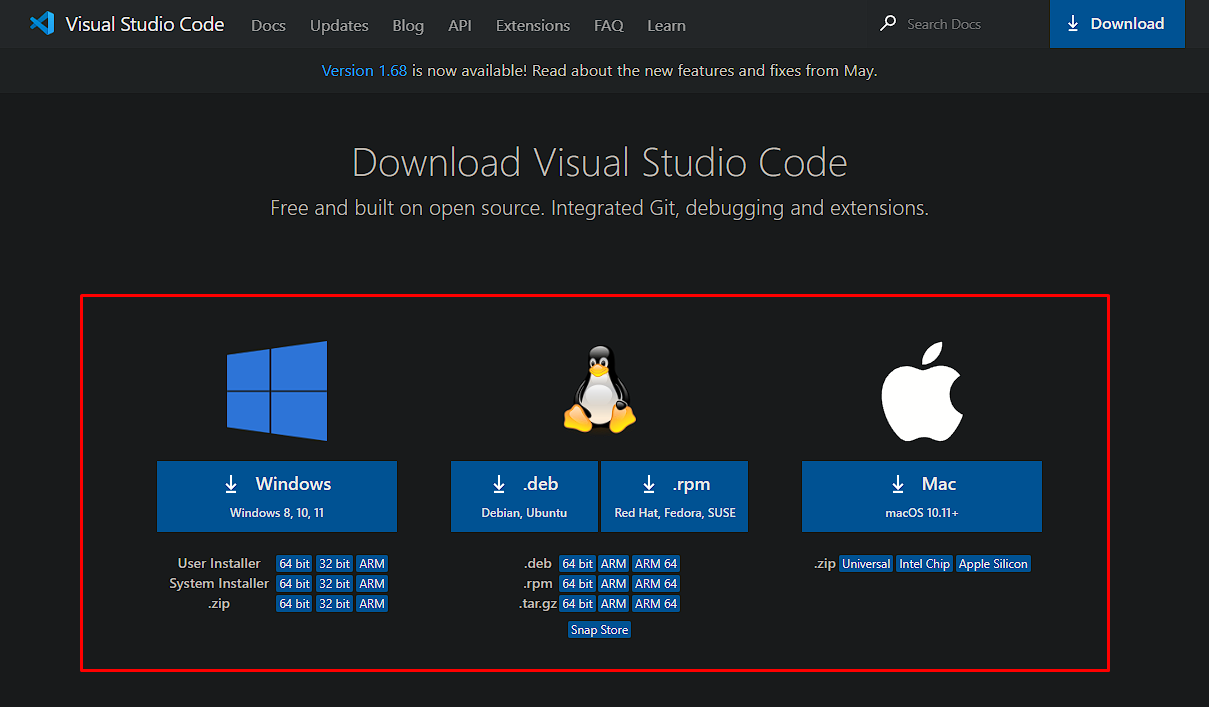
2.1 Introducción de herramientas a utilizar

Esta es una breve introducción a las herramientas que vamos a utilizar a lo largo del desarrollo de este sistema:

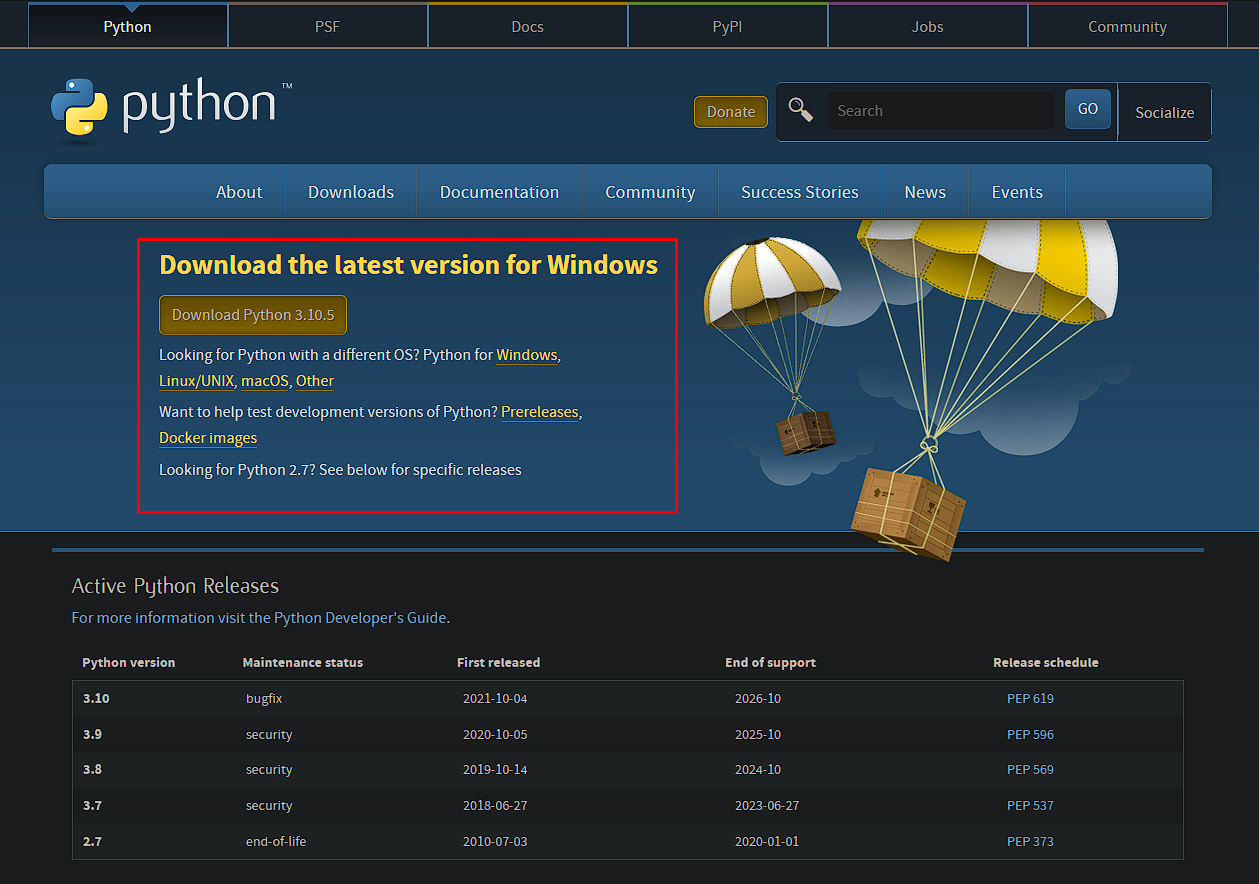
* Vamos a utilizar Visual Studio Code como nuestro editor de texto en donde vamos a crear nuestro programa utilizando el lenguaje de programación Python.
* Dentro de Python vamos a utilizar las siguientes librerías:
* Tkinter: va a ser nuestra interfaz grafica por la cual el usuario se va a poder comunicar con el sistema
* SQLite3: motor de bases de datos SQL ligero donde almacenaremos los datos ingresados
* re: módulo que proporciona operaciones de coincidencia de expresiones regulares (utilizado en las validaciones de ingreso de datos)
* Datetime: módulo que proporciona clases para manipular fechas y horas

2.2 Configuración del área de trabajo

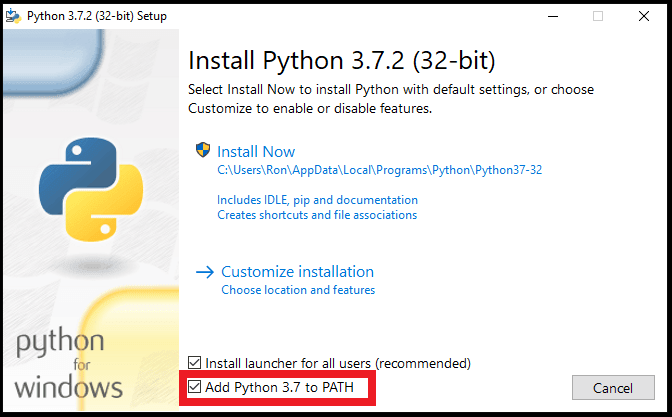
El primer paso va a ser el de descargar Visual Studio Code, para eso vamos a dirigirnos a la siguiente pagina https://code.visualstudio.com/download en donde debemos hacer click en el enlace de descarga correspondiente a nuestro sistema operativo:



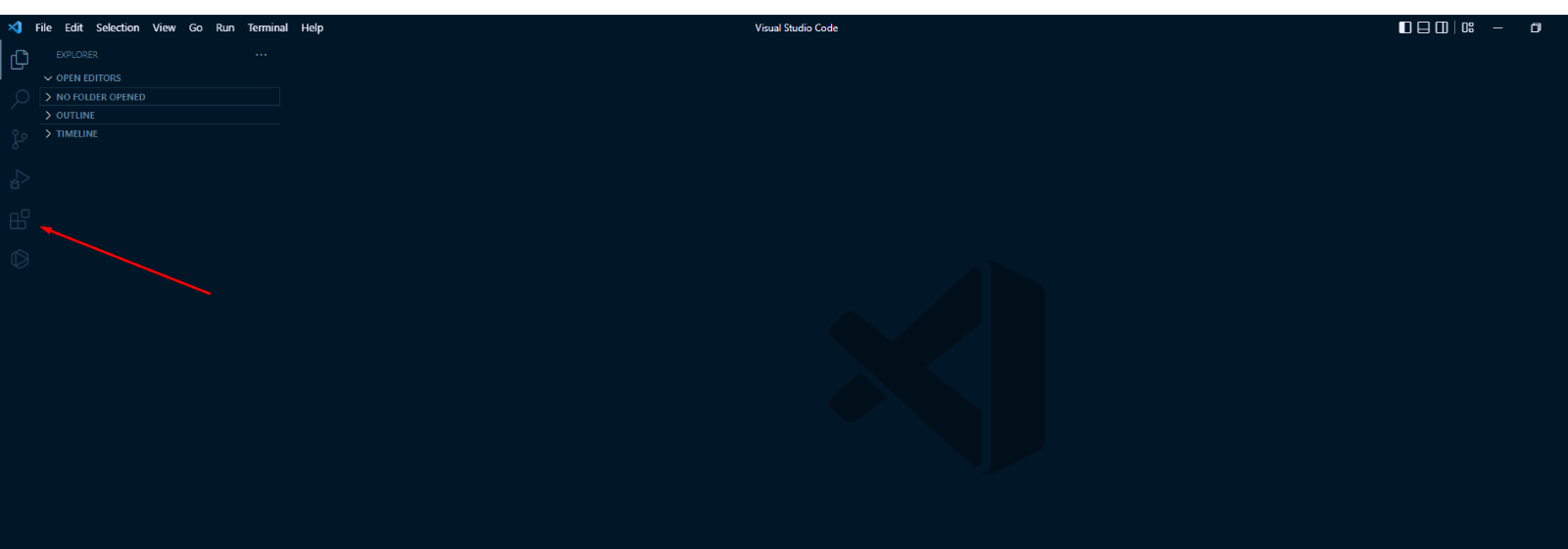
El siguiente paso es el de descargar Python por lo que vamos a ir al siguiente enlace: <https://www.python.org/downloads/> y vamos a hacer click en el botón de descarga de la versión más reciente



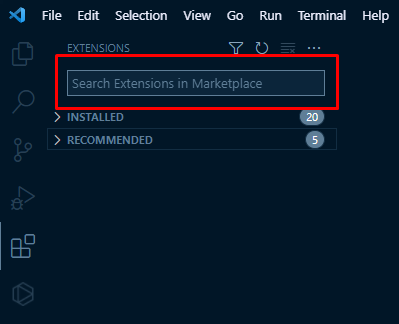
Mientras estamos haciendo la instalación de Python es muy importante que se tilde la opción de “Add Python 3.X to PATH” como muestra la imagen a continuación. Si no realizamos este paso, es muy probable que tengamos problemas de compatibilidad a futuro



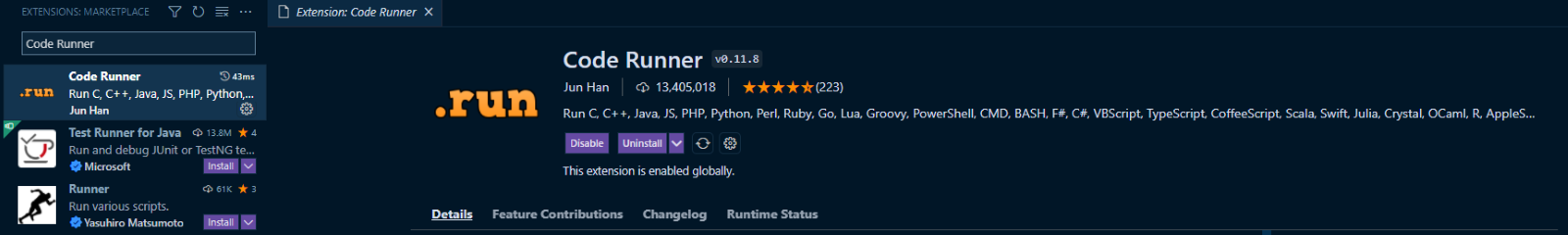
El último paso a la hora de crear nuestro espacio de trabajo es el de poder utilizar Python dentro de Visual Studio Code. Para eso vamos a abrir VS Code y nos vamos a dirigir a la seccion de extensiones



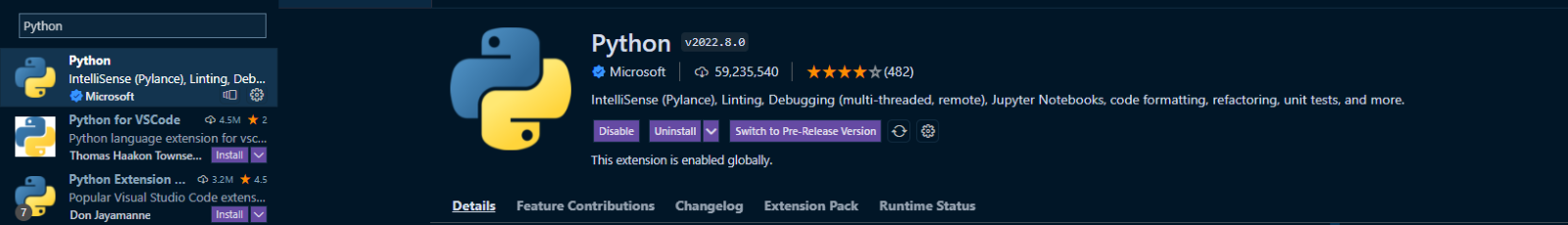
Y dentro de esta sección vamos a utilizar el buscador indicado a continuación para encontrar e instalar las extensiones que dejo más adelante



- Code Runner



- Python



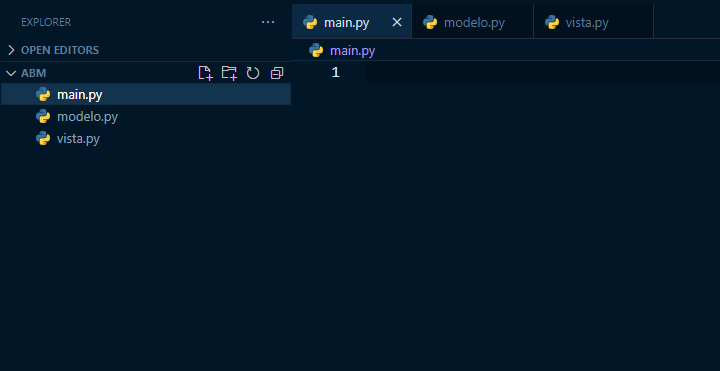
2.3 Creación de la estructura MVC

Luego de seguir los anteriores pasos ya deberíamos contar tanto con VS Code como con Python instalados y funcionando en nuestra computadora así que nuestro siguiente paso es el diseñar la base o arquitectura de nuestro programa.

En este caso la arquitectura del sistema va a estar basada en el patrón MVC (Modelo-Vista-Controlador) en el cual contamos con un archivo “Vista” que es por donde el usuario se va a comunicar con el sistema, un archivo “Modelo” que es el que contiene gran parte de la lógica de nuestro sistema y un archivo “Controlador” que se encarga de comunicar a la Vista con el Modelo.

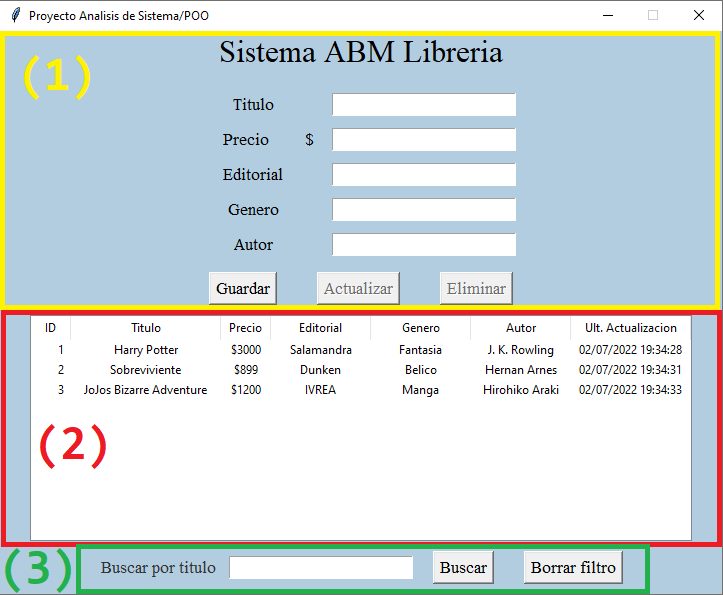
Lo primero que debemos hacer dentro del Visual Studio Code es crear tres archivos que compondrán a nuestro modelo MVC y los vamos a nombrar de la siguiente manera:

* “main.py” como nuestro Controlador
* “modelo.py” como nuestro Modelo
* “Vista.py” como nuestra Vista



2.4 Desarrollo del apartado visual (vista.py)

Voy a separar la parte visual del programa en tres partes para una mejor comprensión del código. Esta división se encuentra graficada a continuación:

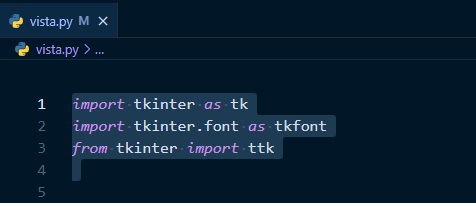


1. Sector principal
2. Listado de Registros
3. Sector de búsqueda/filtro

2.4.1 Sector Principal

Nos vamos a dirigir “vista.py” en donde vamos a importar tres librerías que traen varias funcionalidades para el desarrollo del apartado visual del sistema. Las librerías a importar son las siguientes:

* Tkinter: es el paquete más utilizado para crear interfaces gráficas en Python, es la herramienta que nos permite crear las etiquetas, botones, campos de texto y hasta la misma ventana donde el usuario interactúa con el sistema.
* Tkfont: utilizada para modificar la fuente del texto dentro de los elementos de Tkinter.
* Tkinter.Ttk: extensión del paquete Tkinter que nos permite crear el listado registros



Luego vamos a crear una clase con nombre “Panel” que tendrá un metodo de instancia (metodo que se llama al momento de instanciar a la clase) que recibe los siguientes parámetros:

* self: Hace referencia a la instancia del objeto
* ventana: Hace referencia a un objeto “tk” que va a servir como la ventana donde vamos a posicionar nuestros elementos visuales
* objeto\_modelo: Hace referencia a un objeto de la clase modelo que vamos a crear más adelante. Nos permite interactuar con la parte lógica del programa

*import* tkinter *as* tk

*import* tkinter.font *as* tkfont

*from* tkinter *import* ttk

#*Clase que se encarga del apartado visual del programa*

class Panel():

# *Metodo de instacia de la clase Panel*

def \_\_init\_\_(self, ventana, objeto\_modelo):

Y ahora vamos a realizar las configuraciones básicas de nuestra ventana y las declaraciones de las variables que se van a utilizar dentro de la misma:

#*Clase que se encarga del apartado visual del programa*

class Panel():

# *Metodo de instacia de la clase Panel*

def \_\_init\_\_(self, ventana, objeto\_modelo):

# *Creo una variable de instancia root y la cargo con nuestro objeto tk "ventana" que determina la ventana sobore la que estamos trabajando*

self.root = ventana

# *Configuro el titulo de la ventana sobre la que trabajo*

self.root.title("Proyecto Analisis de Sistema/POO")

# *Este metodo hace que el tamaño de la ventana no se pueda modificar en ninguno de los dos ejes*

self.root.resizable(False, False)

# *Declaro y cargo una variable para almacenar el color de fondo del sistema*

self.color\_fondo = tk.StringVar()

self.color\_fondo.set("#b3cde0")

# *Configuro el color de fondo de la ventana utilizando la variable*

self.root.configure(bg=self.color\_fondo.get())

# *Declaracion de variables a utilizar dentro del sistema*

# *Cada una de estas variables va a ser referenciada dentro de un campo de texto/Entry*

self.var\_titulo = tk.StringVar()

self.var\_precio = tk.StringVar()

self.var\_editorial = tk.StringVar()

self.var\_genero = tk.StringVar()

self.var\_autor = tk.StringVar()

self.var\_id = tk.StringVar()

self.var\_busqueda = tk.StringVar()

Luego vamos a declarar los “tk.Frame”. Este objeto representa un contenedor con un área rectangular en donde se pueden agrupar y organizar nuestros elementos dentro de la ventana del sistema

# *Declaro los frames donde van a ir ubicado los widgets del sistema*

self.frame\_central = tk.Frame(

self.root, bg=self.color\_fondo.get())

self.frame\_entry = tk.Frame(

self.frame\_central, bg=self.color\_fondo.get())

self.frame\_entryprecio = tk.Frame(

self.frame\_entry, bg=self.color\_fondo.get())

self.frame\_botones = tk.Frame(

self.frame\_central, bg=self.color\_fondo.get())

self.frame\_tree = tk.Frame(

self.frame\_central, bg=self.color\_fondo.get())

self.frame\_buscar = tk.Frame(

self.frame\_central, bg=self.color\_fondo.get())

Ahora pasamos a declarar nuestras Label o Etiquetas principales que sirven como guía para el usuario a la hora de poder ingresar información.

# *Declaración de las etiquetas que componen al sector inicial de nuestro programa*

# *El primer parametro es la ubicación en donde se va a posicionar el elemento*

# *El parametro "bg" determina el color de fondo del elemento*

# *El parametro "text" determina el texto que va a tener nuestro elemento*

# *El parametro "font" indica la fuente de este elemento*

self.sistema\_label = tk.Label(

self.root,

bg=self.color\_fondo.get(),

text="Sistema ABM Libreria",

font=tkfont.Font(family="Times", size=23),

)

self.titulo\_label = tk.Label(

self.frame\_entry,

bg=self.color\_fondo.get(),

text="Titulo",

font=tkfont.Font(family="Times", size=13),

)

self.editorial\_label = tk.Label(

self.frame\_entry,

bg=self.color\_fondo.get(),

text="Editorial",

font=tkfont.Font(family="Times", size=13),

)

self.simbolopeso\_label = tk.Label(

self.frame\_entryprecio,

bg=self.color\_fondo.get(),

text="$",

font=tkfont.Font(family="Times", size=13),

)

self.precio\_label = tk.Label(

self.frame\_entryprecio,

bg=self.color\_fondo.get(),

text="Precio",

font=tkfont.Font(family="Times", size=13),

)

self.genero\_label = tk.Label(

self.frame\_entry,

bg=self.color\_fondo.get(),

text="Genero",

font=tkfont.Font(family="Times", size=13),

)

self.autor\_label = tk.Label(

self.frame\_entry,

bg=self.color\_fondo.get(),

text="Autor",

font=tkfont.Font(family="Times", size=13),

)

Creamos nuestros campos de entrada que le permiten ingresar datos en el sistema al usuario y van a estar ubicados al lado de sus etiquetas/labels correspondientes

# *Declaración de las campos de texto que componen al sector inicial de nuestro programa*

# *El primer parametro es la ubicacion en donde se va a pocisionar el elemento*

# *El parametro "textvariable" determina la variable en donde se va a*  *almacenar la informacion de nuestro Entry*

# *El parametro "font" indica la fuente de este elemento*

self.titulo\_entry = tk.Entry(

self.frame\_entry,

textvariable=self.var\_titulo,

font=tkfont.Font(family="Times", size=13),

)

self.precio\_entry = tk.Entry(

self.frame\_entry,

textvariable=self.var\_precio,

font=tkfont.Font(family="Times", size=13),

)

self.editorial\_entry = tk.Entry(

self.frame\_entry,

textvariable=self.var\_editorial,

font=tkfont.Font(family="Times", size=13),

)

self.gen\_entry = tk.Entry(

self.frame\_entry,

textvariable=self.var\_genero,

font=tkfont.Font(family="Times", size=13),

)

self.autor\_entry = tk.Entry(

self.frame\_entry,

textvariable=self.var\_autor,

font=tkfont.Font(family="Times", size=13),

)

Luego de crear los campos de entrada, vamos a declarar los botones principales del programa que son “Guardar”, “Actualizar” y “Eliminar”. Cada uno de estos botones va a llamar a una función que vamos a declarar más adelante en el modulo “Modelo” donde se encuentra la logica del programa

# *Botones del sector principal*

#*El primer parametro de cada boton determina su ubicacion*

#*El parametro "state" determina si el boton es clickeable o no*

#*Boton Eliminar*

#*Llama a la funcion "funcion\_baja" que crearemos mas adelante en el modulo "modelo"*

self.borrar\_boton = tk.Button(

self.frame\_botones,

command=lambda: objeto\_modelo.funcion\_baja(

self.var\_id.get(),

self.upd\_boton,

self.borrar\_boton,

self.tree,

self.var\_titulo,

self.var\_precio,

self.var\_editorial,

self.var\_genero,

self.var\_autor,

self.var\_busqueda,

),

font=tkfont.Font(family="Times", size=13),

text="Eliminar",

state="disabled",

)

# *Boton guardar*

# *Llama a la funcion "funcion\_alta" que crearemos mas adelante en el modulo "modelo"*

self.g\_boton = tk.Button(

self.frame\_botones,

command=lambda: objeto\_modelo.funcion\_alta(

self.var\_titulo,

self.var\_precio,

self.var\_editorial,

self.var\_genero,

self.var\_autor,

self.upd\_boton,

self.borrar\_boton,

self.var\_busqueda,

self.tree,

),

font=tkfont.Font(family="Times", size=13),

text="Guardar",

)

# *Boton guardar*

# *Llama a la funcion "funcion\_alta" que crearemos mas adelante en el modulo "modelo"*

self.upd\_boton = tk.Button(

self.frame\_botones,

command=lambda: objeto\_modelo.funcion\_modificar(

self.var\_id,

self.var\_titulo,

self.var\_precio,

self.var\_editorial,

self.var\_genero,

self.var\_autor,

self.upd\_boton,

self.borrar\_boton,

self.var\_busqueda,

self.tree,

),

font=tkfont.Font(family="Times", size=13),

text="Actualizar",

state="disabled",

)

Y el último paso para dentro del sector en donde vamos a hacer nuestro ingreso de datos es el de ubicar los elementos que creamos. Para eso vamos a utilizar el método “grid” que nos permite ubicar cada elemento dentro de una grilla imaginaria dentro de la ventana

self.frame\_central.grid(row=1, column=0)

self.frame\_entry.grid(row=0, column=0)

self.frame\_botones.grid(row=1, column=0)

self.frame\_tree.grid(row=2, column=0)

self.frame\_buscar.grid(row=3, column=0)

self.sistema\_label.grid(row=0, column=0)

self.titulo\_label.grid(row=1, column=0, padx=15, pady=(20, 5))

self.titulo\_entry.grid(row=1, column=1, padx=15, pady=(20, 5))

self.frame\_entryprecio.grid(row=2, column=0, padx=(30, 0))

self.precio\_label.grid(row=0, column=0, pady=5, padx=(0, 30))

self.simbolopeso\_label.grid(row=0, column=1)

self.precio\_entry.grid(row=2, column=1)

self.editorial\_label.grid(row=3, column=0, pady=5)

self.editorial\_entry.grid(row=3, column=1)

self.genero\_label.grid(row=4, column=0, pady=5)

self.gen\_entry.grid(row=4, column=1)

self.autor\_label.grid(row=5, column=0, pady=5)

self.autor\_entry.grid(row=5, column=1)

self.g\_boton.grid(row=0, column=0, padx=30, pady=10)

self.upd\_boton.grid(row=0, column=1, padx=10, pady=10)

self.borrar\_boton.grid(row=0, column=2, padx=30, pady=10)

2.4.2 Listado de registros

Para la creación del listado vamos a usar el objeto “Treeview” perteneciente a la librería “ttk”.

Un dato a aclarar es que los treeview vienen con una columna por default (normalmente llamada “ghost column” o “columna fantasma”) y vamos a utilizar esa columna para almacenar nuestro dato de “ID”, por eso no se declara la columna “ID” dentro de nuestro array de columnas “self.columnas”

#*Declaro un array con nombres para identificar a cada columna*

self.columnas = ("titulo", "precio", "editorial",

"genero", "autor", "fecha\_upd")

#*Creo el listado treeview y le paso como parametros la ubicacion en donde va a estar pocisionado*

#*y el listado de las columnas*

self.tree = ttk.Treeview(self.frame\_tree, columns=self.columnas)

#*Declaro los heading o encabezados de la lista y le asigno un nombre*

self.tree.heading("#0", text="ID")

#*Modifico la configuracion de la columna correspondiente a cada encabezado*

#*-minwidth determina el ancho minimo que puede tener la columna*

#*-width determina el ancho predeterminado de la columna*

#*-anchor determina como ubico el texto dentro de la columna, en este caso el texto estaria centrado*

self.tree.column("#0", minwidth=0, width=40, anchor="center")

self.tree.heading("titulo", text="Titulo")

self.tree.column("titulo", minwidth=0, width=150, anchor="center")

self.tree.heading("precio", text="Precio")

self.tree.column("precio", minwidth=0, width=50, anchor="center")

self.tree.heading("editorial", text="Editorial")

self.tree.column("editorial", minwidth=0, width=100, anchor="center")

self.tree.heading("genero", text="Genero")

self.tree.column("genero", minwidth=0, width=100, anchor="center")

self.tree.heading("autor", text="Autor")

self.tree.column("autor", minwidth=0, width=100, anchor="center")

self.tree.heading("fecha\_upd", text="Ult. Actualizacion")

self.tree.column("fecha\_upd", minwidth=0, width=120, anchor="center")

#*Pocisiono el listado tree dentro de la grilla*

self.tree.grid(row=0, column=0, padx=30)

Luego creamos un evento utilizando el método “bind” que nos permite ejecutar una función cada vez que el usuario hace click en un registro del listado.

# *Evento que se acciona cuando hago click en un item del treeview*

self.tree.bind(

"<<TreeviewSelect>>",

lambda event: objeto\_modelo.select\_item(

self.tree.item(self.tree.focus()),

self.var\_id,

self.var\_titulo,

self.var\_precio,

self.var\_editorial,

self.var\_genero,

self.var\_autor,

self.borrar\_boton,

self.upd\_boton,

),

)

Y por ultimo llamamos a un metodo que vamos a crear mas adelante y que nos permite popular la lista con los registros guardados en la base de datos. La declaracion de esta funcion se va a hacer mas adelante en el modulo “metodo”.

# *Llamo a funcion para popular la lista*

# *Esta funcion solo se llama una vez*

objeto\_modelo.cargar\_listado(self.tree, self.var\_busqueda)

2.4.3 Sector de búsqueda/filtro

El último sector a desarrollar en el módulo de vista es el de búsqueda/filtro. Este sector está compuesto por:

* Etiqueta/Label: identificador del campo de entrada .
* Campo de texto: espacio en donde el usuario puede ingresar el título del producto a buscar.
* Botón “Buscar”: ejecuta un función“cargar\_listado” del módulo “modelo” que modifica la información del listado treeview dependiendo de los datos ingresados en el campo de texto.
* Botón “Borrar Filtro”: ejecuta la misma función que el botón “Buscar” pero aplicando una lógica distinta que permite vaciar los campos de entrada y mostrar el listado de registros completos sin aplicar filtros.

#*Label "Buscar por titulo"*

self.buscar\_label = tk.Label(

self.frame\_buscar,

bg=self.color\_fondo.get(),

text="Buscar por titulo",

font=tkfont.Font(family="Times", size=13),

)

#*Campo de texto para ingresar titulo utilizado en el filtro*

self.buscar\_entry = tk.Entry(

self.frame\_buscar,

textvariable=self.var\_busqueda,

font=tkfont.Font(family="Times", size=13),

)

#*Boton de busqueda: muestra solo los registros que contengan el texto del entry "buscar\_entry" dentro del listado de registros*

self.busq\_boton = tk.Button(

self.frame\_buscar,

command=lambda: objeto\_modelo.cargar\_listado(

self.tree,

self.var\_busqueda,

),

font=tkfont.Font(family="Times", size=13),

text="Buscar",

)

#*Boton borrar filtro: vacia campos de texto y reinicia el listado de registros*

self.b\_filtro\_boton = tk.Button(

self.frame\_buscar,

command=lambda: objeto\_modelo.cargar\_listado(

self.tree,

self.var\_busqueda,

True,

),

font=tkfont.Font(family="Times", size=13),

text="Borrar filtro",

)

#*Pocisionamiento de la etiquetas, campos de entrada y los botones del sector busqueda*

self.buscar\_label.grid(row=8, column=0, padx=10, pady=10)

self.buscar\_entry.grid(row=8, column=1, pady=10)

self.busq\_boton.grid(row=8, column=2, padx=20, pady=10)

self.b\_filtro\_boton.grid(row=8, column=3, padx=10, pady=10)