Proyecto Certificador de Desarrollo de Software 1

Tema Nº11:

Fase de Implementación – Sprint 3

Indicador de logro Nº11:Desarrolla los entregables del producto de software en el Sprint 3, aplicando la Fase de Implementación del Framework Scrum.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº11:**

FASE DE IMPLEMENTACIÓN – SPRINT 1 - 2 – 3

**Sprint 1 – Preparación e Inicio**

**Objetivo:** Configurar entorno y definir primeros entregables. **Actividades:**

* Instalación y configuración de Python + Tkinter (Tkinter ya viene incluido en la mayoría de distribuciones de Python).
* Diseño preliminar de la base de datos lógica (tablas: Cliente, Producto, Ventas).
* Elaboración del **Product Backlog inicial** con historias de usuario.
* Ceremonias Scrum: Daily Scrum y Sprint Review para validar backlog inicial. **Entregables:**
* Entorno de desarrollo listo.
* Backlog inicial documentado.
* Plantillas de artefactos Scrum

## **Sprint 2 – Desarrollo inicial de módulos**

**Objetivo:** Construir el primer CRUD funcional con Tkinter. **Actividades:**

* Desarrollo del **CRUD de Clientes**:
  + Ventana Tkinter con campos: Código, Nombres, Apellidos, Dirección, Teléfono, Email.
  + Botones: Insertar, Actualizar, Eliminar, Buscar, Limpiar.
  + Controlador (ClienteController) con lógica CRUD en memoria (diccionario o lista).
* Pruebas funcionales: validación de inserción, actualización y eliminación.
* Refinamiento del backlog según feedback.

## **Sprint 3 – Expansión y Consolidación**

**Objetivo:** Ampliar el sistema con nuevos módulos y consolidar la integración. **Actividades:**

* Desarrollo del **CRUD de Productos**:
  + Ventana Tkinter con campos: Código, Nombre, Precio, Stock.
  + Botones: Insertar, Actualizar, Eliminar, Buscar, Limpiar.
  + Controlador (ProductoController) con lógica CRUD en memoria.
* Integración de artefactos Scrum: creación de la **Guía Artefactos Scrum 09**.
* Documentación técnica de mantenimientos realizados.
* Validación de atributos de calidad: usabilidad, mantenibilidad, seguridad básica.
* Ceremonias: Daily Scrum, Sprint Review mostrando Clientes + Productos, Sprint Retrospective. **Entregables:**
* Módulo de Productos operativo en Tkinter.
* Artefactos Scrum integrados y documentados.
* Incremento de software listo para stakeholders.

**Reuniones Scrum Diarias o Daily Meetings**

Las reuniones diarias son una de las ceremonias más importantes de Scrum.

* **Duración**: máximo 15 minutos.
* **Frecuencia:** todos los días del Sprint.
* **Objetivo:** mantener al equipo sincronizado, identificar problemas y asegurar que todos avanzan hacia el Sprint Goal.

Cada persona debe de responder las siguientes preguntas:

* ¿Qué hiciste ayer?

**Axel (Gestor):**Ayer avancé en la interfaz de Clientes con Tkinter.

**Kevin**: **(Analista/Desarrollador):**Ayer organicé las historias de usuario

**Eduardo (Programador):**Ayer probé el botón Insertar en Clientes. Hoy haré pruebas de actualización.

* ¿En qué trabajarás hoy?

**Axel:** Hoy trabajaré en la documentación del backlog.

**Kevin**: Hoy implementaré el CRUD de Producto

**Eduardo:** Hoy haré pruebas de actualización.

* ¿Qué obstáculos han surgido?

**Axel:** El obstáculo es que aún falta definir los campos de Producto.

**Kevin:** . El obstáculo es que necesito ejemplos de validación de datos.

**Ceremonia: Sprint Review**

* El equipo de desarrollo comenta posteriormente qué ha ocurrido durante el Sprint 1 y 2 , los impedimentos que se han encontrado, así como soluciones tomadas y actualizan a los stakeholders con la situación del equipo.

Ceremonia: Reunión diaria (Daily Scrum Meeting).

Las reuniones diarias son una de las ceremonias más importantes de Scrum.

* **Duración:** máximo 15 minutos.
* **Frecuencia:** todos los días del Sprint.
* **Objetivo:** mantener al equipo sincronizado, identificar problemas y asegurar que todos avanzan hacia el Sprint Goal.

### 3ª ceremonia: Sprint Review

El Sprint Review es la reunión que ocurre al final del Sprint, generalmente el último viernes del Sprint, donde el product owner y el Develpment Team presentan a los stakeholders el incremento terminado para su inspección y adaptación correspondientes. En esta reunión organizada por el product owner se estudia cuál es la situación y se actualiza el Product Backlog con las nuevas condiciones que puedan afectar al negocio.

Por un lado, se revisará el incremento terminado. Se mostrará el software funcionando en producción y los stakeholders tendrán la oportunidad de hacer cuantas preguntas estimen oportunas sobre el mismo. El software funcionando ha sido validado previamente por el product owner, que se ha encargado de trabajar con el equipo durante el Sprint para asegurarse que cumple con la Definition of Done y, efectivamente, hace que el Sprint Goal sea válido. Si no existe software funcionando, el Sprint Review carece de sentido, aunque en ciertas ocasiones y oportunidades se sigue manteniendo.

El Development Team tiene que tener un papel importante en esta reunión. Muchas veces no es el product owner quien demuestra el incremento producido, sino que son los propios miembros del Development Team quienes lo hacen. Es una buena práctica no sólo el que lo lleven a cabo, sino también el que lo hagan de forma rotatoria y, tras varios Sprints, hayan participado todos.

El equipo de desarrollo comenta posteriormente qué ha ocurrido durante el Sprint, los impedimentos que se han encontrado, así como soluciones tomadas y actualizan a los stakeholders con la situación del equipo. Por último, el product owner actualiza -con la información de negocio recibida en esta reunión- el Product Backlog para el siguiente Sprint.

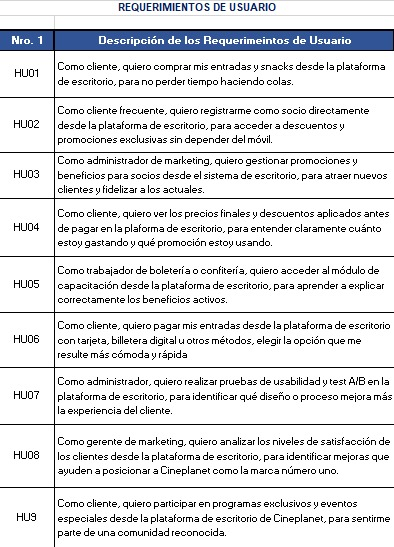
En contra de lo que comúnmente se cree, el Sprint Review no se trata de una demo para un cliente o para los stakeholders o incluso para el product owner, ni es tampoco una reunión para felicitar al Equipo de Desarrollo. Es una reunión de trabajo, una de las más importantes porque sirve para marcar la estrategia de negocio. La duración estimada en el estándar para un Sprint Review es de 8 horas para un Sprint de 4 semanas, aunque habitualmente estas reuniones se ejecutan en un entorno de entre 2 y 3 horas.

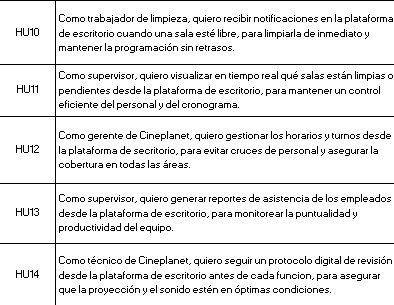
**Entregables del producto**

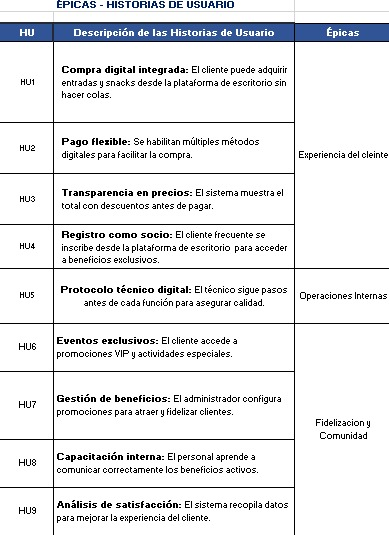
* + Creación de plantilla de todos los artefactos Scrum.
  + Uso de la arquitectura y patrones de diseño de software.
  + Uso de buenas prácticas de análisis y diseño de desarrollo de software.
  + Uso de buenas prácticas de codificación del software.
  + Atributos de calidad de software.
  + Integración de todos los artefactos Scrum en una sola plantilla Scrum

Desarrollo de la Guia\_Artefactos Scrum\_09

* Caso : Ventas
* Sistema para Cineplanet



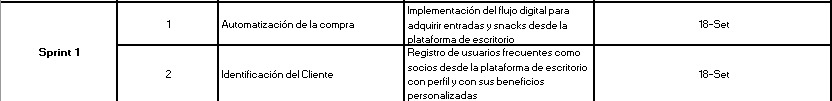


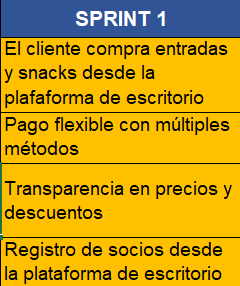


Product Backlog Priorizado

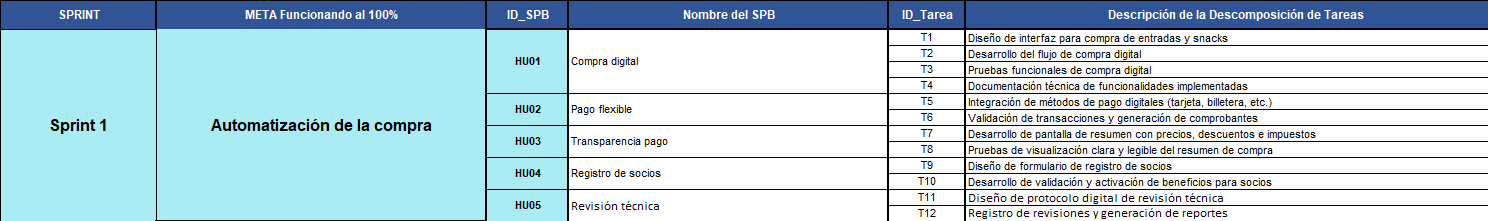


DETERMINACIÓN DE LA LONGITUD DEL SPRINT 1

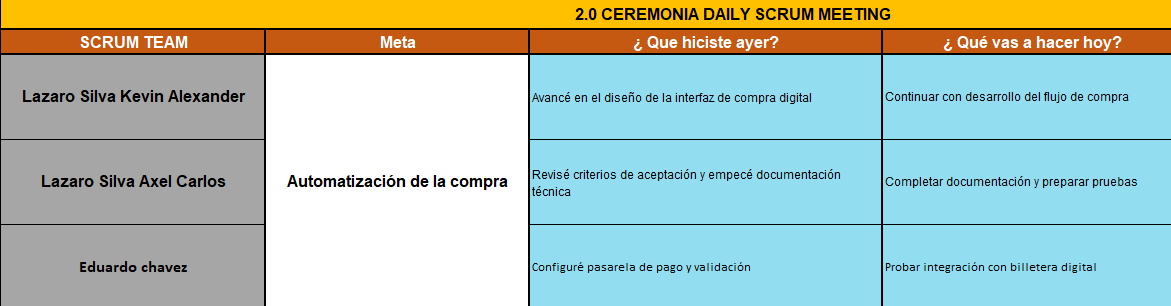


Sprint Backlog

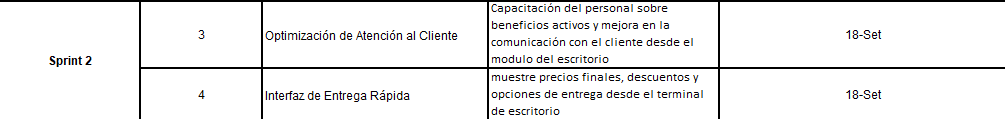
Desarrollo del Sprint 1



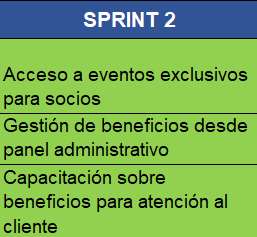
Ceremonias Daily Scrum Meeting-Sprint 1

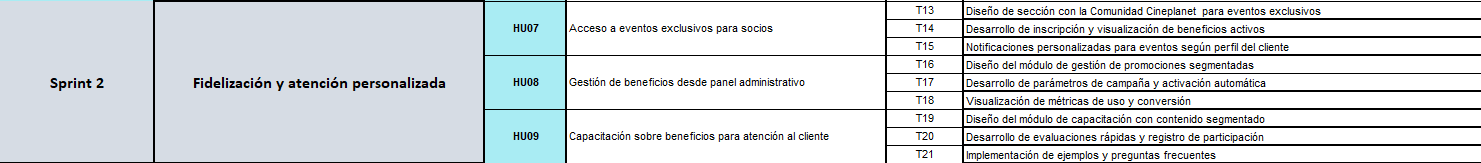


DETERMINACIÓN DE LA LONGITUD DEL SPRINT 2

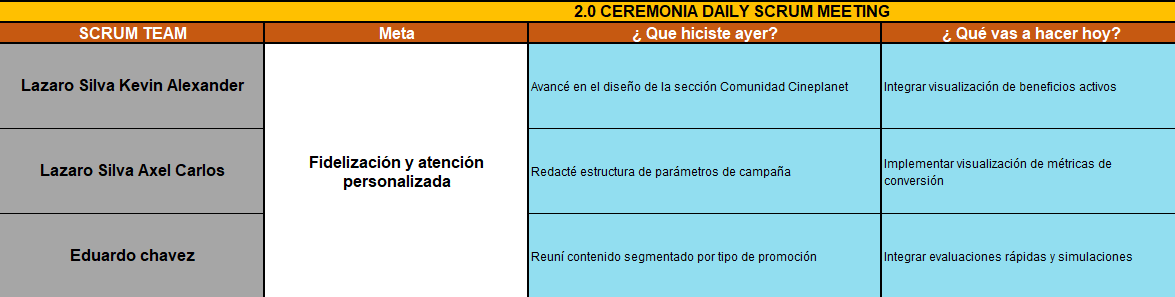


Sprint Backlog



****Desarrollo del Sprint 2

Ceremonias Daily Scrum Meeting-Sprint 2

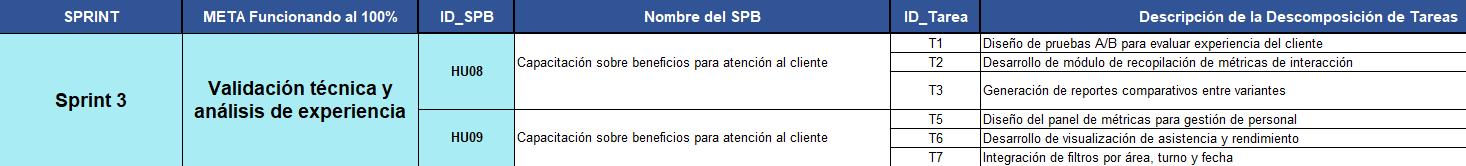
****

DETERMINACIÓN DE LA LONGITUD DEL SPRINT 3

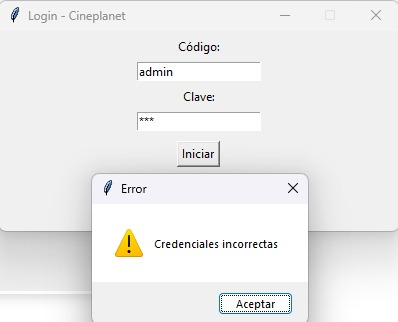
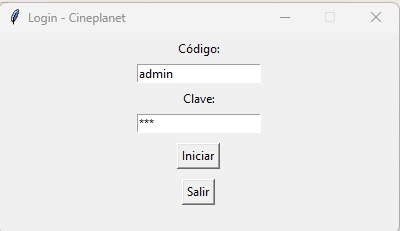
****

Sprint Backlog

****

****Desarrollo del Sprint

****Ceremonias Daily Scrum Meeting-Spr





import tkinter as tk

from tkinter import ttk, messagebox

from datetime import datetime

# -------------------- LOGIN --------------------

class Login(tk.Tk):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.title("Login - Cineplanet")

        self.geometry("400x200")

        self.resizable(False, False)

        tk.Label(self, text="Código:").pack(pady=5)

        self.txtCodigo = tk.Entry(self)

        self.txtCodigo.pack()

        tk.Label(self, text="Clave:").pack(pady=5)

        self.txtClave = tk.Entry(self, show="\*")

        self.txtClave.pack()

        tk.Button(self, text="Iniciar", command=self.validar).pack(pady=10)

        tk.Button(self, text="Salir", command=self.destroy).pack()

    def validar(self):

        if self.txtCodigo.get() == "admin" and self.txtClave.get() == "1234":

            self.destroy()

            VentanaPrincipal().mainloop()

        else:

            messagebox.showwarning("Error", "Credenciales incorrectas")

Este código crea una ventana de inicio de sesión con **Tkinter**. La clase Login define la interfaz con campos para código y clave. El campo de clave usa show="\*" para ocultar los caracteres escritos. Hay dos botones: uno para iniciar sesión y otro para salir. El método validar comprueba si las credenciales son "admin" y "1234". Si son correctas abre la ventana principal, si no muestra un mensaje de error.



# -------------------- VENTANA PRINCIPAL --------------------

class VentanaPrincipal(tk.Tk):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.title("Sistema Cineplanet")

        self.geometry("800x600")

        self.resizable(True, True)  # ✅ Ahora sí puedes ajustar con el mouse

        notebook = ttk.Notebook(self)

        notebook.pack(expand=True, fill="both")

        notebook.add(TabLimpieza(notebook), text="Limpieza")

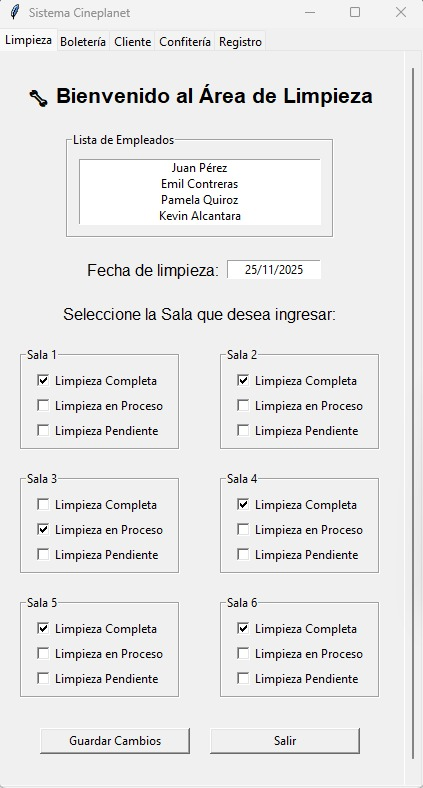
        notebook.add(TabBoleteria(notebook), text="Boletería")

        notebook.add(TabCliente(notebook), text="Cliente")

        notebook.add(TabConfiteria(notebook), text="Confitería")

        notebook.add(TabRegistro(notebook), text="Registro")

La clase VentanaPrincipal hereda de y define la ventana principal del sistema Cineplanet. Se configura el título de la ventana y un tamaño inicial de 800x600 píxeles. La opción permite que el usuario ajuste la ventana libremente con el mouse. Dentro de la ventana se crea un contenedor de , que organiza la interfaz en pestañas. El método asegura que el ocupe todo el espacio disponible. Posteriormente se agregan las pestañas correspondientes a los módulos del sistema. Estas pestañas representan Limpieza, Boletería, Cliente, Confitería y Registro, facilitando la navegación entre funcionalidades







# -------------------- LIMPIEZA --------------------

import tkinter as tk

from tkinter import messagebox

from datetime import datetime

class TabLimpieza(tk.Frame):

    def \_\_init\_\_(self, parent):

        super().\_\_init\_\_(parent)

        # Canvas con scrollbar (solo para esta pestaña)

        canvas = tk.Canvas(self, highlightthickness=0)

        scrollbar = tk.Scrollbar(self, orient="vertical", command=canvas.yview)

        canvas.configure(yscrollcommand=scrollbar.set)

        canvas.pack(side="left", fill="both", expand=True)

        scrollbar.pack(side="right", fill="y")

        # Frame interno para scroll

        scroll\_frame = tk.Frame(canvas)

        canvas.create\_window((0, 0), window=scroll\_frame, anchor="n")

        scroll\_frame.bind("<Configure>", lambda e: canvas.configure(scrollregion=canvas.bbox("all")))

        # Contenedor principal

        contenedor = tk.Frame(scroll\_frame)

        contenedor.pack(pady=20)

        # Encabezado

        tk.Label(contenedor, text="🔧 Bienvenido al Área de Limpieza", font=("Arial", 16, "bold")).pack(pady=10)

        # Lista de empleados

        empleados\_frame = tk.LabelFrame(contenedor, text="Lista de Empleados", padx=10, pady=10)

        empleados\_frame.pack(pady=10)

        self.listaEmpleados = tk.Listbox(empleados\_frame, height=4, width=40, justify="center")

        for nombre in ["Juan Pérez", "Emil Contreras", "Pamela Quiroz", "Kevin Alcantara"]:

            self.listaEmpleados.insert(tk.END, nombre)

        self.listaEmpleados.pack()

        # Fecha

        fecha\_frame = tk.Frame(contenedor)

        fecha\_frame.pack(pady=10)

        tk.Label(fecha\_frame, text="Fecha de limpieza:", font=("Arial", 12)).pack(side="left", padx=5)

        self.fechaLimpieza = tk.Entry(fecha\_frame, justify="center", width=15)

        self.fechaLimpieza.insert(0, datetime.now().strftime("%d/%m/%Y"))

        self.fechaLimpieza.pack(side="left")

        # Salas distribuidas en 3 filas de 2

        tk.Label(contenedor, text="Seleccione la Sala que desea ingresar:", font=("Arial", 12)).pack(pady=10)

        self.estadoSalas = {}

        salas\_frame = tk.Frame(contenedor)

        salas\_frame.pack()

        salas = ["Sala 1", "Sala 2", "Sala 3", "Sala 4", "Sala 5", "Sala 6"]

        for i, sala in enumerate(salas):

            fila = i // 2

            columna = i % 2

            frame = tk.LabelFrame(salas\_frame, text=sala, padx=10, pady=5)

            frame.grid(row=fila, column=columna, padx=20, pady=10)

            varCompleta = tk.IntVar()

            varProceso = tk.IntVar()

            varPendiente = tk.IntVar()

            tk.Checkbutton(frame, text="Limpieza Completa", variable=varCompleta).pack(anchor="w")

            tk.Checkbutton(frame, text="Limpieza en Proceso", variable=varProceso).pack(anchor="w")

            tk.Checkbutton(frame, text="Limpieza Pendiente", variable=varPendiente).pack(anchor="w")

            self.estadoSalas[sala] = (varCompleta, varProceso, varPendiente)

        # Botones finales

        botones = tk.Frame(contenedor)

        botones.pack(pady=20)

        tk.Button(botones, text="Guardar Cambios", command=self.guardar, width=20).pack(side="left", padx=10)

        tk.Button(botones, text="Salir", command=self.salir, width=20).pack(side="right", padx=10)

    def guardar(self):

        resumen = []

        for sala, (completa, proceso, pendiente) in self.estadoSalas.items():

            estado = []

            if completa.get(): estado.append("Completa")

            if proceso.get(): estado.append("En Proceso")

            if pendiente.get(): estado.append("Pendiente")

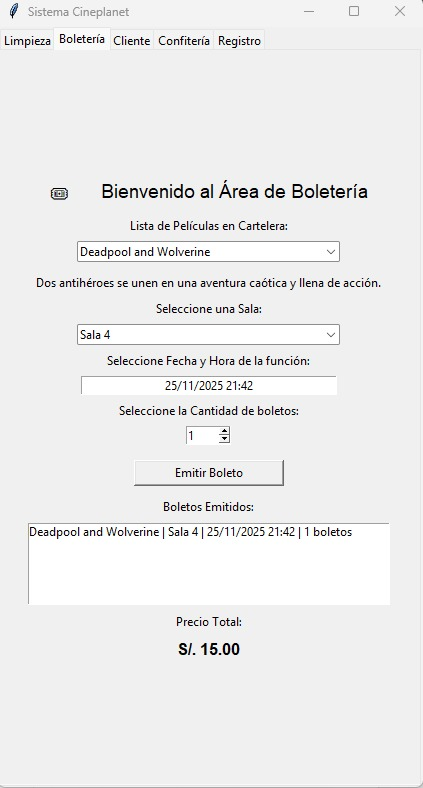
            resumen.append(f"{sala}: {', '.join(estado) if estado else 'Sin estado'}")

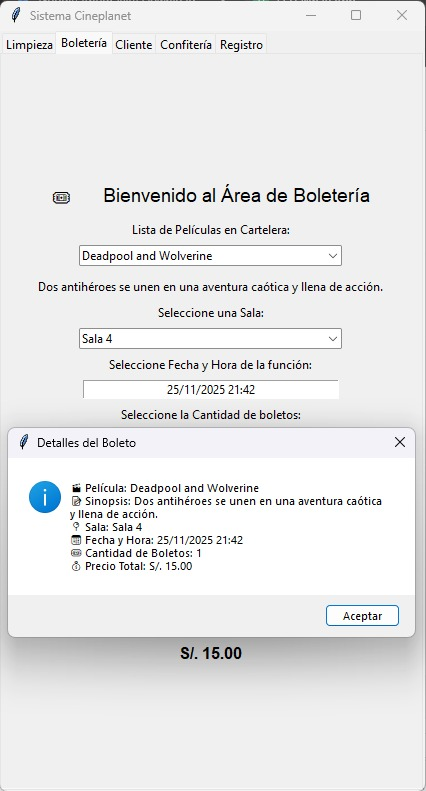
        messagebox.showinfo("Guardado", "\n".join(resumen))

    def salir(self):

        self.master.quit()

Este código define la pestaña **Limpieza** dentro del sistema Cineplanet. La clase TabLimpieza hereda de tk.Frame y organiza la interfaz con scroll. Incluye un encabezado y una lista de empleados asignados al área de limpieza. Se muestra la fecha actual en un campo de entrada editable. Las salas de cine se presentan en cuadros con opciones de estado: completa, en proceso o pendiente. El botón **Guardar Cambios** genera un resumen de los estados seleccionados y lo muestra en un mensaje. El botón **Salir** cierra la aplicación desde esta pestaña.







# -------------------- BOLETERÍA --------------------

class TabBoleteria(tk.Frame):

    def \_\_init\_\_(self, parent):

        super().\_\_init\_\_(parent)

        contenedor = tk.Frame(self)

        contenedor.pack(expand=True)

        self.peliculas = {

            "Deadpool and Wolverine": "Dos antihéroes se unen en una aventura caótica y llena de acción.",

            "Inside Out 2": "Las emociones de Riley enfrentan nuevos desafíos en su adolescencia.",

            "Dune: Parte Dos": "Paul Atreides lidera la rebelión en el planeta Arrakis.",

            "Kung Fu Panda 4": "Po regresa para entrenar a un nuevo guerrero dragón.",

            "Oppenheimer": "La historia del físico detrás de la bomba atómica.",

            "Barbie": "Una muñeca descubre el mundo real y su identidad.",

        }

        tk.Label(contenedor, text="🎟️ Bienvenido al Área de Boletería", font=("Arial", 14)).pack(pady=10)

        tk.Label(contenedor, text="Lista de Películas en Cartelera:").pack()

        self.comboPeliculas = ttk.Combobox(contenedor, values=list(self.peliculas.keys()), width=40)

        self.comboPeliculas.set("Deadpool and Wolverine")

        self.comboPeliculas.pack(pady=5)

        self.comboPeliculas.bind("<<ComboboxSelected>>", self.mostrar\_sinopsis)

        self.labelSinopsis = tk.Label(contenedor, text=self.peliculas["Deadpool and Wolverine"], wraplength=500, justify="left")

        self.labelSinopsis.pack(pady=5)

        tk.Label(contenedor, text="Seleccione una Sala:").pack()

        self.comboSala = ttk.Combobox(contenedor, values=["Sala 1", "Sala 2", "Sala 3","Sala 4","Sala 5","Sala 6"], width=40)

        self.comboSala.set("Sala 1")

        self.comboSala.pack(pady=5)

        tk.Label(contenedor, text="Seleccione Fecha y Hora de la función:").pack()

        self.dateTimeFuncion = tk.Entry(contenedor, width=42, justify="center")

        self.dateTimeFuncion.insert(0, datetime.now().strftime("%d/%m/%Y %H:%M"))

        self.dateTimeFuncion.pack(pady=5)

        tk.Label(contenedor, text="Seleccione la Cantidad de boletos:").pack()

        self.spinCantidad = tk.Spinbox(contenedor, from\_=1, to=20, width=5)

        self.spinCantidad.pack(pady=5)

        tk.Button(contenedor, text="Emitir Boleto", command=self.emitir, width=20).pack(pady=10)

        tk.Label(contenedor, text="Boletos Emitidos:").pack()

        self.listaBoletos = tk.Listbox(contenedor, width=60, height=5)

        self.listaBoletos.pack(pady=5)

        tk.Label(contenedor, text="Precio Total:").pack()

        self.labelPrecio = tk.Label(contenedor, text="S/. 0.00", font=("Arial", 12, "bold"))

        self.labelPrecio.pack(pady=5)

    def mostrar\_sinopsis(self, event=None):

        pelicula = self.comboPeliculas.get()

        sinopsis = self.peliculas.get(pelicula, "Sinopsis no disponible.")

        self.labelSinopsis.config(text=sinopsis)

    def emitir(self):

        pelicula = self.comboPeliculas.get()

        sinopsis = self.peliculas.get(pelicula, "Sinopsis no disponible.")

        sala = self.comboSala.get()

        fecha\_hora = self.dateTimeFuncion.get()

        cantidad = int(self.spinCantidad.get())

        total = cantidad \* 15.00

        self.labelPrecio.config(text=f"S/. {total:.2f}")

        self.listaBoletos.insert(tk.END, f"{pelicula} | {sala} | {fecha\_hora} | {cantidad} boletos")

        detalles = (

            f"🎬 Película: {pelicula}\n"

            f"📝 Sinopsis: {sinopsis}\n"

            f"📍 Sala: {sala}\n"

            f"📅 Fecha y Hora: {fecha\_hora}\n"

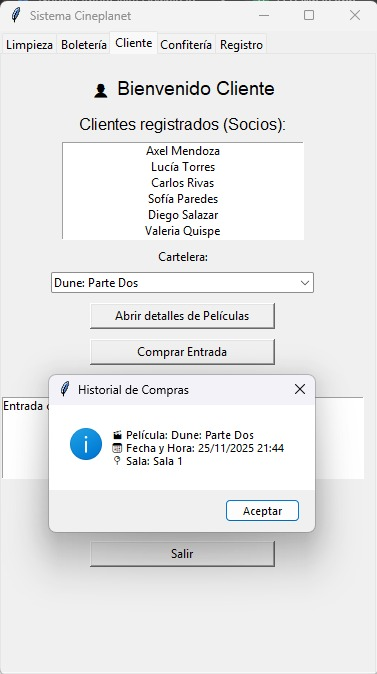
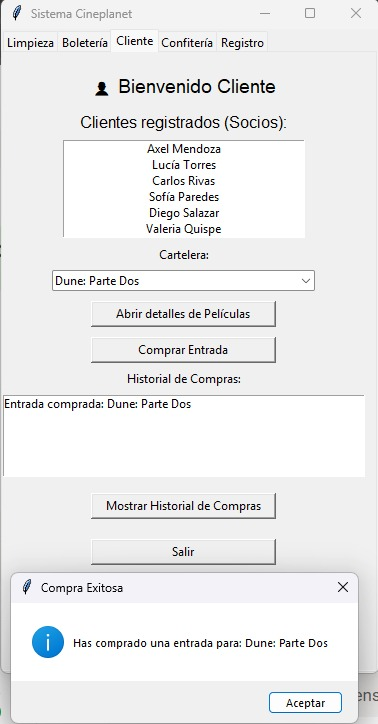
            f"🎟️ Cantidad de Boletos: {cantidad}\n"

            f"💰 Precio Total: S/. {total:.2f}"

        )

        messagebox.showinfo("Detalles del Boleto", detalles)

Este código define la pestaña **Boletería** dentro del sistema Cineplanet. La clase TabBoleteria muestra una lista de películas en cartelera con su sinopsis. El usuario puede seleccionar la sala, la fecha y la hora de la función. También se elige la cantidad de boletos mediante un control Spinbox. Al presionar **Emitir Boleto**, se calcula el precio total y se registra en la lista. Finalmente, se muestra un mensaje con los detalles completos de la compra.





# -------------------- CLIENTE --------------------

class TabCliente(tk.Frame):

    def \_\_init\_\_(self, parent):

        super().\_\_init\_\_(parent)

        canvas = tk.Canvas(self, highlightthickness=0)

        scrollbar = tk.Scrollbar(self, orient="vertical", command=canvas.yview)

        canvas.configure(yscrollcommand=scrollbar.set)

        canvas.pack(side="left", fill="both", expand=True)

        scrollbar.pack(side="right", fill="y")

        scroll\_frame = tk.Frame(canvas)

        canvas.create\_window((0, 0), window=scroll\_frame, anchor="n")

        scroll\_frame.bind("<Configure>", lambda e: canvas.configure(scrollregion=canvas.bbox("all")))

        contenedor = tk.Frame(scroll\_frame)

        contenedor.pack(pady=10)

        self.peliculas = {

            "Deadpool and Wolverine": {"descripcion": "Dos antihéroes se unen en una aventura caótica.", "genero": "Acción / Comedia"},

            "Inside Out 2": {"descripcion": "Las emociones de Riley enfrentan nuevos desafíos.", "genero": "Animación / Familiar"},

            "Dune: Parte Dos": {"descripcion": "Paul Atreides lidera la rebelión en Arrakis.", "genero": "Ciencia Ficción / Drama"},

            "Barbie": {"descripcion": "Una muñeca descubre el mundo real y su identidad.", "genero": "Comedia / Fantasía"},

            "Kung Fu Panda 4": "Po regresa para entrenar a un nuevo guerrero dragón.", "genero": "Acción / Comedia",

            "Oppenheimer": "La historia del físico detrás de la bomba atómica.", "genero": "Acción / suspenso",

        }

        self.historial = []

        tk.Label(contenedor, text="👤 Bienvenido Cliente", font=("Arial", 14)).pack(pady=10)

        tk.Label(contenedor, text="Clientes registrados (Socios):", font=("Arial", 12)).pack()

        self.listaSocios = tk.Listbox(contenedor, height=6, width=40, justify="center")

        for nombre in ["Axel Mendoza", "Lucía Torres", "Carlos Rivas", "Sofía Paredes", "Diego Salazar", "Valeria Quispe"]:

            self.listaSocios.insert(tk.END, nombre)

        self.listaSocios.pack(pady=5)

        tk.Label(contenedor, text="Cartelera:").pack()

        self.comboCartelera = ttk.Combobox(contenedor, values=list(self.peliculas.keys()), width=40)

        self.comboCartelera.set("Inside Out 2")

        self.comboCartelera.pack(pady=5)

        tk.Button(contenedor, text="Abrir detalles de Películas", command=self.abrir\_detalles, width=25).pack(pady=5)

        tk.Button(contenedor, text="Comprar Entrada", command=self.comprar, width=25).pack(pady=5)

        tk.Label(contenedor, text="Historial de Compras:").pack()

        self.listaHistorial = tk.Listbox(contenedor, width=60, height=5)

        self.listaHistorial.pack(pady=5)

        tk.Button(contenedor, text="Mostrar Historial de Compras", command=self.mostrar\_historial, width=25).pack(pady=10)

        tk.Button(contenedor, text="Salir", command=self.salir, width=25).pack(pady=10)

    def abrir\_detalles(self):

        pelicula = self.comboCartelera.get()

        detalles = self.peliculas.get(pelicula, {})

        descripcion = detalles.get("descripcion", "Sin descripción.")

        genero = detalles.get("genero", "Sin género.")

        messagebox.showinfo("Detalles de Película", f"🎬 {pelicula}\n📝 {descripcion}\n🎞️ Género: {genero}")

    def comprar(self):

        pelicula = self.comboCartelera.get()

        fecha\_hora = datetime.now().strftime("%d/%m/%Y %H:%M")

        sala = "Sala 1"

        self.historial.append({

            "pelicula": pelicula,

            "fecha\_hora": fecha\_hora,

            "sala": sala

        })

        self.listaHistorial.insert(tk.END, f"Entrada comprada: {pelicula}")

        messagebox.showinfo("Compra Exitosa", f"Has comprado una entrada para: {pelicula}")

    def mostrar\_historial(self):

        if not self.historial:

            messagebox.showinfo("Historial", "No hay compras registradas.")

            return

        resumen = ""

        for item in self.historial:

            resumen += (

                f"🎬 Película: {item['pelicula']}\n"

                f"📅 Fecha y Hora: {item['fecha\_hora']}\n"

                f"📍 Sala: {item['sala']}\n\n"

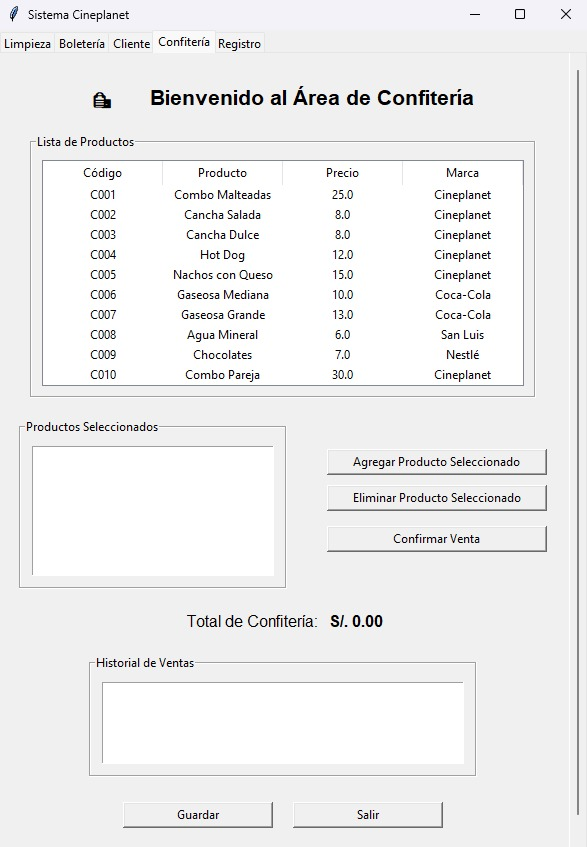
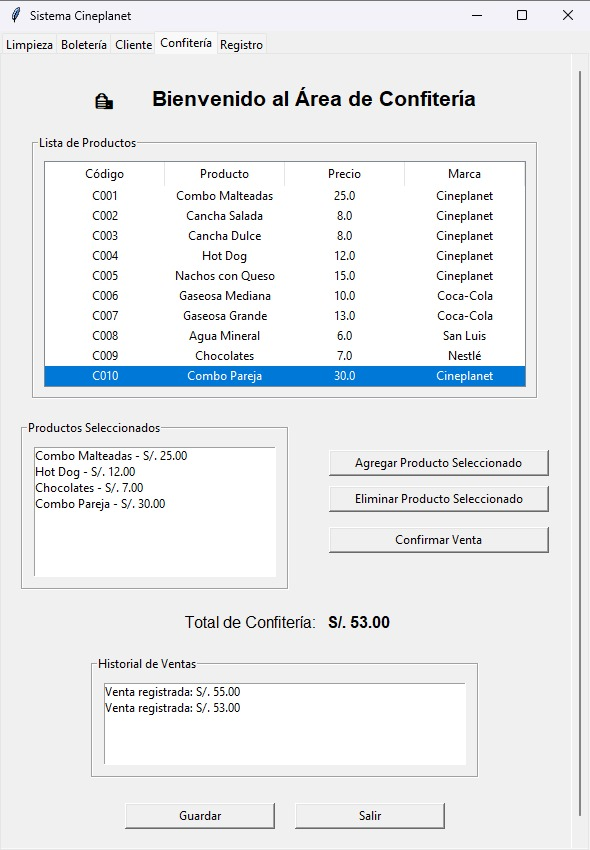
            )

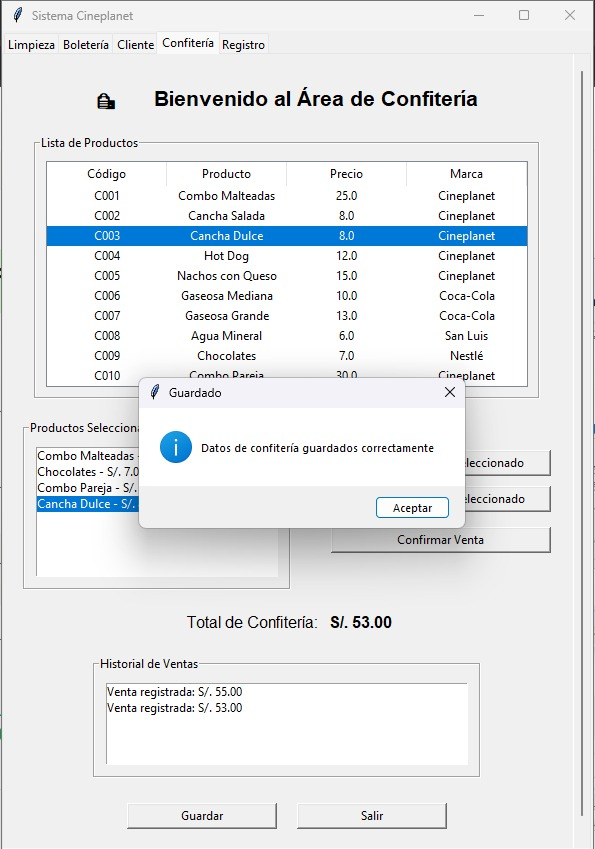
        messagebox.showinfo("Historial de Compras", resumen.strip())

    def salir(self):

        self.master.quit()

Este código define la pestaña **Cliente** dentro del sistema Cineplanet. La clase TabCliente organiza la interfaz con scroll y un contenedor principal. Se muestra una lista de socios registrados y un combo con películas en cartelera. El botón **Abrir detalles** permite ver la descripción y género de la película seleccionada. El botón **Comprar Entrada** registra la compra en un historial y muestra un mensaje de confirmación. El historial de compras se visualiza en una lista y puede mostrarse en detalle con otro botón. Finalmente, el botón **Salir** cierra la aplicación desde esta pestaña.







# -------------------- CONFITERÍA --------------------

class TabConfiteria(tk.Frame):

    def \_\_init\_\_(self, parent):

        super().\_\_init\_\_(parent)

        # Canvas con scrollbar

        canvas = tk.Canvas(self, highlightthickness=0)

        scrollbar = tk.Scrollbar(self, orient="vertical", command=canvas.yview)

        canvas.configure(yscrollcommand=scrollbar.set)

        canvas.pack(side="left", fill="both", expand=True)

        scrollbar.pack(side="right", fill="y")

        scroll\_frame = tk.Frame(canvas)

        canvas.create\_window((0, 0), window=scroll\_frame, anchor="n")

        scroll\_frame.bind("<Configure>", lambda e: canvas.configure(scrollregion=canvas.bbox("all")))

        contenedor = tk.Frame(scroll\_frame)

        contenedor.pack(pady=20)

        tk.Label(contenedor, text="🛍️ Bienvenido al Área de Confitería", font=("Arial", 16, "bold")).pack(pady=10)

        # Sección de productos

        productos\_frame = tk.LabelFrame(contenedor, text="Lista de Productos", padx=10, pady=10)

        productos\_frame.pack(pady=10)

        self.tablaProductos = ttk.Treeview(productos\_frame, columns=("Código", "Producto", "Precio", "Marca"), show="headings", height=10)

        for col in self.tablaProductos["columns"]:

            self.tablaProductos.heading(col, text=col)

            self.tablaProductos.column(col, anchor="center", width=120)

        self.tablaProductos.pack()

        productos = [

            ("C001", "Combo Malteadas", 25.0, "Cineplanet"),

            ("C002", "Cancha Salada", 8.0, "Cineplanet"),

            ("C003", "Cancha Dulce", 8.0, "Cineplanet"),

            ("C004", "Hot Dog", 12.0, "Cineplanet"),

            ("C005", "Nachos con Queso", 15.0, "Cineplanet"),

            ("C006", "Gaseosa Mediana", 10.0, "Coca-Cola"),

            ("C007", "Gaseosa Grande", 13.0, "Coca-Cola"),

            ("C008", "Agua Mineral", 6.0, "San Luis"),

            ("C009", "Chocolates", 7.0, "Nestlé"),

            ("C010", "Combo Pareja", 30.0, "Cineplanet"),

        ]

        for item in productos:

            self.tablaProductos.insert("", "end", values=item)

        # Sección de selección y acciones

        acciones\_frame = tk.Frame(contenedor)

        acciones\_frame.pack(pady=10)

        seleccionados\_frame = tk.LabelFrame(acciones\_frame, text="Productos Seleccionados", padx=10, pady=10)

        seleccionados\_frame.grid(row=0, column=0, padx=20)

        self.listaSeleccionados = tk.Listbox(seleccionados\_frame, width=40, height=8)

        self.listaSeleccionados.pack()

        botones\_frame = tk.Frame(acciones\_frame)

        botones\_frame.grid(row=0, column=1, padx=20)

        tk.Button(botones\_frame, text="Agregar Producto Seleccionado", command=self.agregar\_producto, width=30).pack(pady=5)

        tk.Button(botones\_frame, text="Eliminar Producto Seleccionado", command=self.eliminar\_producto, width=30).pack(pady=5)

        tk.Button(botones\_frame, text="Confirmar Venta", command=self.confirmar\_venta, width=30).pack(pady=10)

        # Total y historial

        resumen\_frame = tk.Frame(contenedor)

        resumen\_frame.pack(pady=10)

        total\_frame = tk.Frame(resumen\_frame)

        total\_frame.pack()

        tk.Label(total\_frame, text="Total de Confitería:", font=("Arial", 12)).pack(side="left", padx=5)

        self.labelTotal = tk.Label(total\_frame, text="S/. 0.00", font=("Arial", 12, "bold"))

        self.labelTotal.pack(side="left")

        historial\_frame = tk.LabelFrame(contenedor, text="Historial de Ventas", padx=10, pady=10)

        historial\_frame.pack(pady=10)

        self.listaHistorial = tk.Listbox(historial\_frame, width=60, height=5)

        self.listaHistorial.pack()

        # Botones finales

        botones\_finales = tk.Frame(contenedor)

        botones\_finales.pack(pady=15)

        tk.Button(botones\_finales, text="Guardar", command=self.guardar, width=20).pack(side="left", padx=10)

        tk.Button(botones\_finales, text="Salir", command=self.salir, width=20).pack(side="right", padx=10)

    def agregar\_producto(self):

        selected = self.tablaProductos.selection()

        for item in selected:

            valores = self.tablaProductos.item(item)["values"]

        try:

            precio = float(valores[2])

            texto = f"{valores[1]} - S/. {precio:.2f}"

            self.listaSeleccionados.insert(tk.END, texto)

        except ValueError:

            messagebox.showerror("Error", "El precio no es válido.")

    def eliminar\_producto(self):

        seleccion = self.listaSeleccionados.curselection()

        for i in reversed(seleccion):

            self.listaSeleccionados.delete(i)

    def confirmar\_venta(self):

        total = 0

        for i in range(self.listaSeleccionados.size()):

            texto = self.listaSeleccionados.get(i)

            try:

                precio = float(texto.split("S/.")[1])

                total += precio

            except:

                pass

        self.labelTotal.config(text=f"S/. {total:.2f}")

        self.listaHistorial.insert(tk.END, f"Venta registrada: S/. {total:.2f}")

        self.listaSeleccionados.delete(0, tk.END)

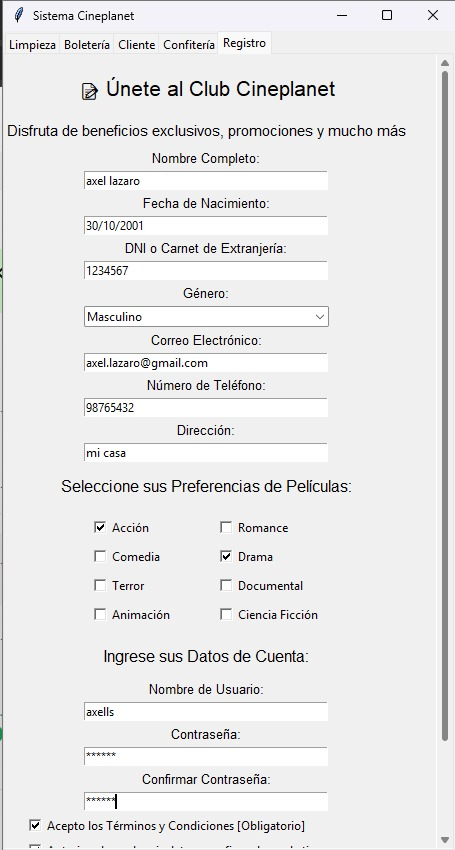
    def guardar(self):

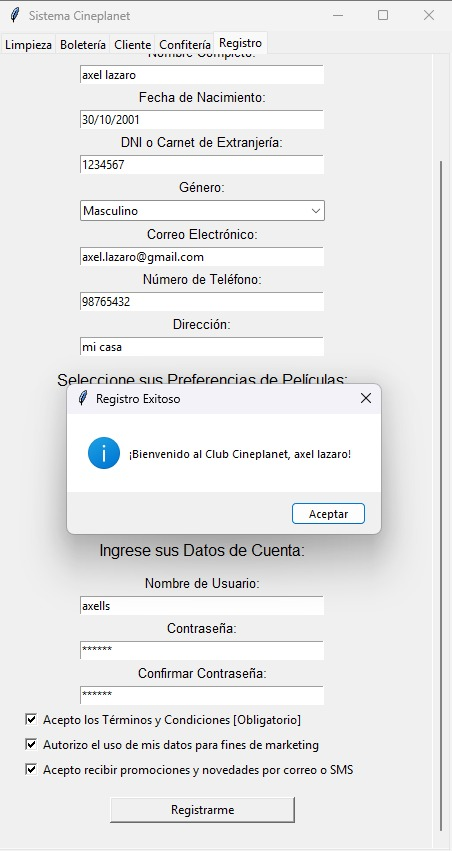
        messagebox.showinfo("Guardado", "Datos de confitería guardados correctamente")

    def salir(self):

        self.master.quit()

Este código define la pestaña **Confitería** dentro del sistema Cineplanet. La clase TabConfiteria organiza la interfaz con scroll y un contenedor principal. Se muestra una tabla de productos con código, nombre, precio y marca. El usuario puede agregar productos seleccionados a una lista y eliminarlos si es necesario. El botón **Confirmar Venta** calcula el total y registra la venta en el historial. También se incluye un botón **Guardar** para confirmar datos y otro **Salir** para cerrar la aplicación.







# -------------------- REGISTRO --------------------

class TabRegistro(tk.Frame):

    def \_\_init\_\_(self, parent):

        super().\_\_init\_\_(parent)

        canvas = tk.Canvas(self, highlightthickness=0)

        scrollbar = tk.Scrollbar(self, orient="vertical", command=canvas.yview)

        canvas.configure(yscrollcommand=scrollbar.set)

        canvas.pack(side="left", fill="both", expand=True)

        scrollbar.pack(side="right", fill="y")

        scroll\_frame = tk.Frame(canvas)

        canvas.create\_window((0, 0), window=scroll\_frame, anchor="n")

        scroll\_frame.bind("<Configure>", lambda e: canvas.configure(scrollregion=canvas.bbox("all")))

        contenedor = tk.Frame(scroll\_frame)

        contenedor.pack(pady=10)

        tk.Label(contenedor, text="📝 Únete al Club Cineplanet", font=("Arial", 16)).pack(pady=10)

        tk.Label(contenedor, text="Disfruta de beneficios exclusivos, promociones y mucho más", font=("Arial", 11)).pack(pady=5)

        self.txtNombre = tk.Entry(contenedor, width=40)

        self.txtNacimiento = tk.Entry(contenedor, width=40)

        self.txtDNI = tk.Entry(contenedor, width=40)

        self.comboGenero = ttk.Combobox(contenedor, values=["Masculino", "Femenino", "Otro"], width=37)

        self.txtCorreo = tk.Entry(contenedor, width=40)

        self.txtTelefono = tk.Entry(contenedor, width=40)

        self.txtDireccion = tk.Entry(contenedor, width=40)

        for label, widget in [("Nombre Completo:", self.txtNombre),

                              ("Fecha de Nacimiento:", self.txtNacimiento),

                              ("DNI o Carnet de Extranjería:", self.txtDNI),

                              ("Género:", self.comboGenero),

                              ("Correo Electrónico:", self.txtCorreo),

                              ("Número de Teléfono:", self.txtTelefono),

                              ("Dirección:", self.txtDireccion)]:

            tk.Label(contenedor, text=label, font=("Arial", 10)).pack()

            widget.pack(pady=2)

        tk.Label(contenedor, text="Seleccione sus Preferencias de Películas:", font=("Arial", 12)).pack(pady=10)

        framePreferencias = tk.Frame(contenedor)

        framePreferencias.pack(pady=5)

        col1 = tk.Frame(framePreferencias)

        col1.pack(side="left", padx=20)

        col2 = tk.Frame(framePreferencias)

        col2.pack(side="right", padx=20)

        self.prefVars = {}

        generos = ["Acción", "Comedia", "Terror", "Animación", "Romance", "Drama", "Documental", "Ciencia Ficción"]

        for i, genero in enumerate(generos):

            var = tk.IntVar()

            self.prefVars[genero] = var

            target\_col = col1 if i < 4 else col2

            tk.Checkbutton(target\_col, text=genero, variable=var, anchor="w").pack(anchor="w", pady=2)

        tk.Label(contenedor, text="Ingrese sus Datos de Cuenta:", font=("Arial", 12)).pack(pady=10)

        self.txtUsuario = tk.Entry(contenedor, width=40)

        self.txtClave = tk.Entry(contenedor, show="\*", width=40)

        self.txtConfirmar = tk.Entry(contenedor, show="\*", width=40)

        for label, widget in [("Nombre de Usuario:", self.txtUsuario),

                              ("Contraseña:", self.txtClave),

                              ("Confirmar Contraseña:", self.txtConfirmar)]:

            tk.Label(contenedor, text=label, font=("Arial", 10)).pack()

            widget.pack(pady=2)

        self.checkTerminos = tk.IntVar()

        self.checkMarketing = tk.IntVar()

        self.checkPromociones = tk.IntVar()

        tk.Checkbutton(contenedor, text="Acepto los Términos y Condiciones [Obligatorio]", variable=self.checkTerminos).pack(anchor="w", padx=20)

        tk.Checkbutton(contenedor, text="Autorizo el uso de mis datos para fines de marketing", variable=self.checkMarketing).pack(anchor="w", padx=20)

        tk.Checkbutton(contenedor, text="Acepto recibir promociones y novedades por correo o SMS", variable=self.checkPromociones).pack(anchor="w", padx=20)

        tk.Button(contenedor, text="Registrarme", command=self.registrar, width=25).pack(pady=15)

    def registrar(self):

        if not self.checkTerminos.get():

            messagebox.showwarning("Error", "Debes aceptar los términos y condiciones")

            return

        if self.txtClave.get() != self.txtConfirmar.get():

            messagebox.showwarning("Error", "Las contraseñas no coinciden")

            return

        nombre = self.txtNombre.get()

        messagebox.showinfo("Registro Exitoso", f"¡Bienvenido al Club Cineplanet, {nombre}!")

# -------------------- EJECUCIÓN --------------------

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

    Login().mainloop()

La clase TabRegistro crea la pestaña de registro del sistema Cineplanet. Usa un Canvas con Scrollbar para permitir desplazamiento en el formulario. Dentro del contenedor se organizan etiquetas y campos de entrada para datos personales y de contacto. También incluye una sección de preferencias de películas con casillas de verificación. Se añaden campos de cuenta como usuario y contraseña con validación de coincidencia. El formulario incorpora opciones de consentimiento y marketing mediante checkbuttons. Finalmente, el botón Registrarme ejecuta la función registrar, que valida condiciones y muestra un mensaje de bienvenida.