UCT Informatic Industries

Home Contenido Contacto

UNIVERSIDAD CATÓLICA DE TEMUCO

17-10-2025

RESTAURANT

BASTIAN LIEMPI
MARTIN LOPEZ
VICENTE SANTIN
PROFESOR: GUIDO MELLADO

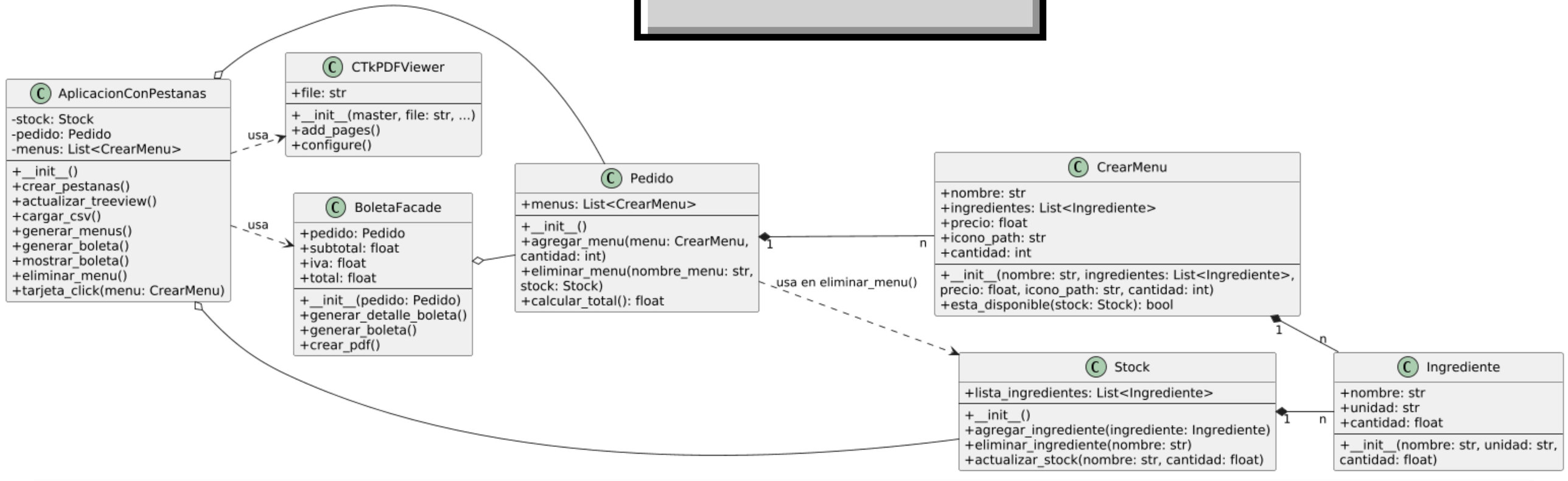
Start

INTRODUCCIÓN

La problemática principal es la ausencia de un sistema central unificado que gestione las operaciones de un Restaurante, por lo que, en consecuencia, se desarrolla un sistema que contiene:

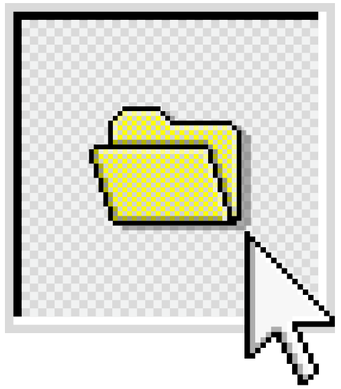
- Arquitectura POO
- Gestión de inventario dinámico de disponibilidad
- Interfaz grafica para integrar todas las funcionalidades
- Generación de documentos mediante pátron de diseño Facade

MAPA UML


AplicacionConPestanas
-stock: Stock
-pedido: Pedido
-menus: List<CrearMenu>
+__init__()
+crear_pestanas()
+actualizar_treeview()
+cargar_csv()
+generar_menus()
+generar_boleta()
+mostrar_boleta()
+eliminar_menu()
+tarjeta_click(menu: CrearMenu)
usa
CTkPDFViewer
+file: str
+__init__(master, file: str, ...)
+add_pages()
+configure()
usa
BoletaFacade
+pedido: Pedido
+subtotal: float
+iva: float
+total: float
+__init__(pedido: Pedido)
+generar_detalle_boleta()
+generar_boleta()
+crear_pdf()
Pedido
+menus: List<CrearMenu>
+__init__()
+agregar_menu(menu: CrearMenu, cantidad: int)
+eliminar_menu(nombre_menu: str, stock: Stock)
+calcular_total(): float
1
n
usa en eliminar_menu()
CrearMenu
+nombre: str
+ingredientes: List<Ingrediente>
+precio: float
+icono_path: str
+cantidad: int
+__init__(nombre: str, ingredientes: List<Ingrediente>, precio: float, icono_path: str, cantidad: int)
+esta_disponible(stock: Stock): bool
1
n
Stock
+lista_ingredientes: List<Ingrediente>
+__init__()
+agregar_ingrediente(ingrediente: Ingrediente)
+eliminar_ingrediente(nombre: str)
+actualizar_stock(nombre: str, cantidad: float)
1
n
Ingrediente
+nombre: str
+unidad: str
+cantidad: float
+__init__(nombre: str, unidad: str, cantidad: float)

CARGA CSV

Restaurante.py



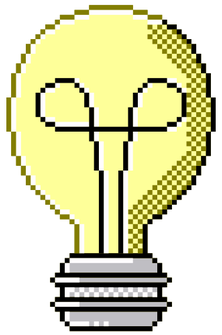
```
ruta_csv = filedialog.askopenfilename(
  title="Seleccione el archivo CSV",
  filetypes=(("CSV", "*.csv"), ("todos los archivos", "*.*"))
)
if ruta_csv:
  try:
    self.df_csv = pd.read_csv(ruta_csv)
    self.mostrar_dataframe_en_tabla(self.df_csv)
    self.boton_agregar_stock.configure(command=self.agregar_csv_al_stock)
  except Exception as e:
    CTkMessagebox(title="Error", message=f"No se pudo cargar el archivo CSV.\n{e}", icon="warning")
```

UCT Informatic Industries

Home **Contenido** Contacto

Restaurante.py

```
def ingresar_ingrediente(self):
    nombre = self.entry_nombre.get()
    nombre = nombre.title()
    unidad = self.combo_unidad.get()
    cantidad = self.entry_cantidad.get()

          if   not   self.validar_nombre(nombre)   or   not
self.validar_cantidad(cantidad):
      return

             ingrediente   =   Ingrediente(nombre=nombre,
unidad=unidad, cantidad=cantidad)
    self.stock.agregar_ingrediente(ingrediente)
    self.actualizar_treeview()
```



Stock.py

```
def agregar_ingrediente(self, ingrediente):
    nombre_ing = ingrediente.nombre.lower()
    for ing in self.lista_ingredientes:
      nombre_ing_existente = ing.nombre.lower()
      if nombre_ing_existente == nombre_ing:
        ing.cantidad += float(ingrediente.cantidad)
        return
    self.lista_ingredientes.append(ingrediente)
```

Restaurante.py

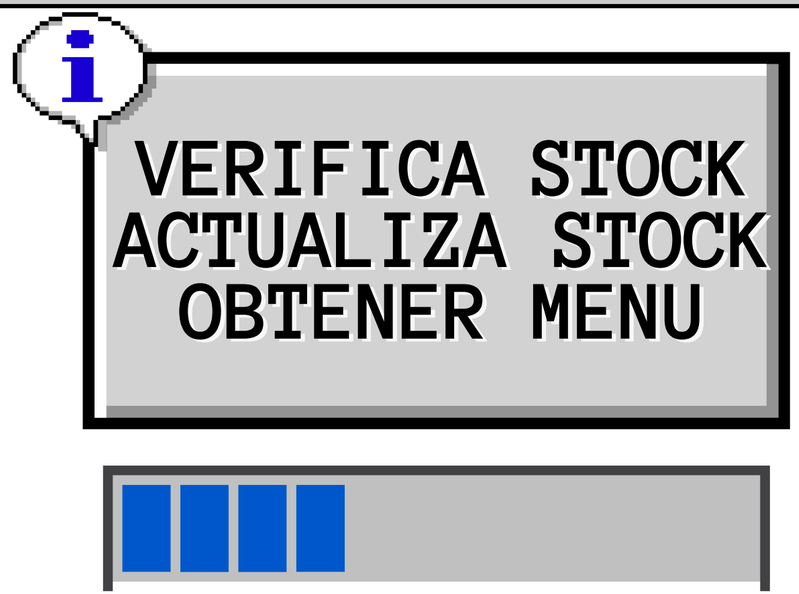
```
def eliminar_ingrediente(self):
    seleccionado = self.tree.selection()
    if not seleccionado:
        CTkMessagebox(title="Error", message="Por favor, selecciona un ingrediente
para eliminar.", icon="warning")
      return
    # proceso de borrar el item seleccionado
    item = self.tree.item(seleccionado)
    nombre_ingrediente = item['values'][0]
    exito = self.stock.eliminar_ingrediente(nombre_ingrediente)

    if exito:
      self.actualizar_treeview()
            CTkMessagebox(title="Ingrediente Eliminado", message=f"El ingrediente
'{nombre_ingrediente}' ha sido eliminado del stock.", icon="info")
    else:
        CTkMessagebox(title="Error", message=f"No se pudo encontrar el ingrediente
'{nombre_ingrediente}' en el stock.", icon="warning")
```

Stock.py

```
def eliminar_ingrediente(self, nombre_ingrediente):
    nombre_ingrediente = nombre_ingrediente.lower()
    for ing in self.lista_ingredientes:
      if ing.nombre.lower() == nombre_ingrediente:
        self.lista_ingredientes.remove(ing)
        return True
    return False
```

UCT Informatic Industries

Home **Contenido** Contacto



Stock.py

```
def verificar_stock(self):
    ingredientes_disponibles = []
    for ing in self.lista_ingredientes:
      if ing.cantidad > 0:
        ingredientes_disponibles.append(ing)
    return ingredientes_disponibles

  def actualizar_stock(self, nombre_ingrediente, nueva_cantidad):
    nombre_ingrediente = nombre_ingrediente.lower()
    for ing in self.lista_ingredientes:
      if ing.nombre.lower() == nombre_ingrediente:
        ing.cantidad = float(nueva_cantidad)
        return True
    return False

  def obtener_elementos_menu(self):
    elementos_menu = []     for ing in self.lista_ingredientes:
      if ing.cantidad > 0:
        elementos_menu.append(ing.nombre)
    return elementos_menu
```


```
def generar_y_mostrar_carta_pdf(self):
    try:
      menus_disponibles = [m for m in self.menus if self.menu_disponible(m)]

      if not menus_disponibles:
        CTkMessagebox(title="Carta vacía", message="No hay menús con ingredientes suficientes en stock.", icon="warning")
        return

      pdf_path = "carta.pdf"

      # crea el PDF con los menús disponibles
      create_menu_pdf(
        menus_disponibles,
        pdf_path,
        titulo_negocio="Restaurante",
        subtitulo="Carta Primavera 2025",
        moneda="$"
      )

      if self.pdf_viewer_carta is not None:
        try:
          self.pdf_viewer_carta.pack_forget()
          self.pdf_viewer_carta.destroy()
        except Exception:
          pass
        self.pdf_viewer_carta = None

      abs_pdf = os.path.abspath(pdf_path) # se convierte la ruta relativa a absoluta
      self.pdf_viewer_carta = CTkPDFViewer(self.pdf_frame_carta, file=abs_pdf)
      self.pdf_viewer_carta.pack(expand=True, fill="both")
```

Restaurante.py

UCT Informatic Industries

Home **Contenido** Contacto

Restaurante.py

```
def actualizar_treeview(self):
    for item in self.tree.get_children():
      self.tree.delete(item)

    for ingrediente in self.stock.lista_ingredientes:
       self.tree.insert("", "end", values=(ingrediente.nombre,
ingrediente.unidad, ingrediente.cantidad))
```

Restaurante.py

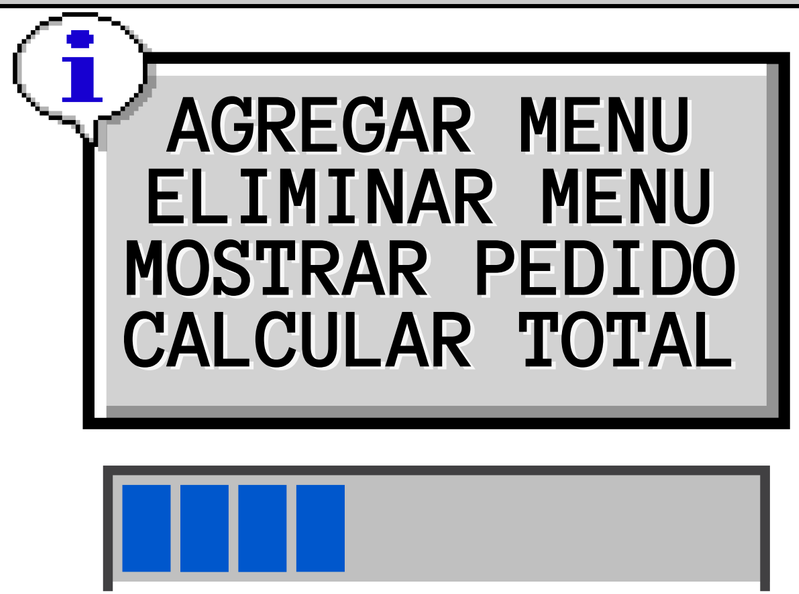

```
def generar_menus(self):
    for widget in self.tarjetas_frame.winfo_children():
      widget.destroy()

    self.menus_creados = []
    menus_disponibles = [m for m in self.menus if self.menu_disponible(m)]

    if not menus_disponibles:
      CTkMessagebox(title="Carta vacía", message="No hay menús con ingredientes suficientes en stock.", icon="warning")
      return

    for menu in menus_disponibles:
      self.crear_tarjeta(menu)

    CTkMessagebox(title="Carta generada", message="Los menús disponibles se han generado correctamente.", icon="info")

  def menu_disponible(self, menu):
    if not getattr(menu, "ingredientes", None):
      return False
    for ingr_req in menu.ingredientes:
      encontrado = False
      for ingr_stock in self.stock.lista_ingredientes:
        if ingr_req.nombre == ingr_stock.nombre:
          try:
            if int(ingr_stock.cantidad) >= int(ingr_req.cantidad):
              encontrado = True
              break
          except Exception:
            return False
      if not encontrado:
        return False
    return True
```

Restaurante.py


```
def eliminar_menu(self):
    seleccionado = self.treeview_menu.selection()
    if not seleccionado:
      CTkMessagebox(title="Error", message="Por favor, selecciona un menú para eliminar.", icon="warning")
      return

    item = self.treeview_menu.item(seleccionado)
    nombre_menu = item['values'][0]
    cantidad_menu = int(item['values'][1]) # Obtener la cantidad del menú a eliminar

    menu_a_eliminar = next((menu for menu in self.pedido.menus if menu.nombre == nombre_menu), None)

    if not menu_a_eliminar:
      CTkMessagebox(title="Error", message="No se encontró el menú seleccionado en el pedido.", icon="warning")
      return

    for ingrediente_necesario in menu_a_eliminar.ingredientes:
      for _ in range(cantidad_menu): # Iterar según la cantidad del menú
        for ingrediente_stock in self.stock.lista_ingredientes:
          if ingrediente_necesario.nombre == ingrediente_stock.nombre:
            ingrediente_stock.cantidad = str(int(ingrediente_stock.cantidad) + int(ingrediente_necesario.cantidad))
            break # Salir del bucle interno una vez que se ha encontrado y actualizado el ingrediente

    exito = self.pedido.eliminar_menu(nombre_menu)

    if exito:
      self.actualizar_treeview_pedido()
      total = self.pedido.calcular_total()
      self.label_total.configure(text=f"Total: ${total:.2f}")
      CTkMessagebox(title="Menú Eliminado", message=f"El menú '{nombre_menu}' ha sido eliminado del pedido y los ingredientes devueltos al stock.", icon="info")
    else:
      CTkMessagebox(title="Error", message=f"No se pudo eliminar el menú '{nombre_menu}' del pedido.", icon="warning")
```

UCT Informatic Industries

Home **Contenido** Contacto



Pedido.py

```
def agregar_menu(self, menu: CrearMenu,cantidad: int = 1):
    for item in self.menus:
      if item.nombre == menu.nombre:
        item.cantidad += cantidad
        return
    menu.cantidad = cantidad
    self.menus.append(menu)

  def eliminar_menu(self, nombre_menu: str):
    for menu in self.menus:
      if menu.nombre == nombre_menu:
        self.menus.remove(menu)
        return True
    return False

  def mostrar_pedido(self):
    return [(m.nombre, m.cantidad, m.precio, m.precio * m.cantidad) for m in self.menus]

  def calcular_total(self) -> float:
    return sum(m.precio * m.cantidad for m in self.menus)
```

UCT Informatic Industries

Home **Contenido** Contacto

Restaurante.py

```
def generar_boleta(self):
    if not self.pedido.menus:
      CTkMessagebox(title="Error", message="No hay menús en el pedido para generar una boleta.", icon="warning")
      return

    try:
      boleta = BoletaFacade(self.pedido)
      mensaje = boleta.generar_boleta()
      CTkMessagebox(title="Boleta Generada", message=mensaje, icon="info")
    except Exception as e:
      CTkMessagebox(title="Error", message=f"No se pudo generar la boleta.\n{e}", icon="warning")
```

UCT Informatic Industries

Home **Contenido** Contacto

Restaurante.py

```
def mostrar_boleta(self):
    try:
      pdf_path = "boleta.pdf"
      if not os.path.exists(pdf_path):
        CTkMessagebox(title="Error", message="Primero debes generar una boleta.", icon="warning")
        return

      if self.pdf_viewer_boleta is not None:
        try:
          self.pdf_viewer_boleta.pack_forget()
          self.pdf_viewer_boleta.destroy()
        except Exception:
          pass
        self.pdf_viewer_boleta = None

      abs_pdf = os.path.abspath(pdf_path)
      self.pdf_viewer_boleta = CTkPDFViewer(self.pdf_frame_boleta, file=abs_pdf)
      self.pdf_viewer_boleta.pack(expand=True, fill="both")

    except Exception as e:
      CTkMessagebox(title="Error", message=f"No se pudo mostrar la boleta.\n{e}", icon="warning")
```