Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas

Programación Avanzada 2MV7

Practica 5

Autor:
Barrios Mendez Jose Alberto
Boleta: 2022640111

Profesor:
Cruz Mora Jose Luis

Ing. Mecatrónica

20 de septiembre de 2024

Índice

1. Objetivo.

Crear aplicaciones con interfaces gráficas de usuario (GUI) utilizando los controles avanzados.

2. Introduccion.

Para esta practica numero 5, nos proponen un reto, el cual consiste en reutilizar la practica de la tienda de mascotas para crear una interfaz de usuario en la cual se lleve a cabo la implementacion para la venta de las mascotas requeridas en la practica anteriormente solicitada. Sin embargo el reto aqui es desarrollar la practica en una interfaz de usuario (GUI) con la ayuda de PyQt6, podremos hacer el diseño de la interfaz de una forma mas amigable, recordando que PyQt6 tiene como objetivo el diseño de interfaces. Una vez que hayamos elegido el diseño de nuestra interfaz es neccesario llevar a cabo el back-end, para que nuestra interfaz de usuario funcione de una forma optima

3. Desarrollo.

3.1. Creacion de la interfaz

Con ayuda del designer de PyQt6, creamos la siguiente interfaz grafica en el entorno

Figura 1: Interfaz diseñada en PyQt6-Designer

En la imagen podemos observar distintos elementos, empezando por los elementos encerrados en el recuadro de color rojo, los cuales son elementos de tipo Pushbuttom, tienen como funcionalidad o proposito el cambiar entre distintas ventanas de la interfaz grafica, esto lo trataremos mas a detalle mas adelante. Continuando con los elementos dentro de esta primera interfaz tambien notamos elementos de tipo QLael, los cuales me sirven como contenerdores o como elementos para poder representar o ubicar una imagen dentro de estos, ademas de poder colocar texto en estos elementos de la interfaz. Es importante mencionar que dentro de tod o el menu de color parecido al morado, este va a ser un

menu desplegable, el cual se activara al presionar el simbolo de menu, o algunas de los elementos del menu, para poder llevar a cabo esto, es necesario hacer lo siguiente:

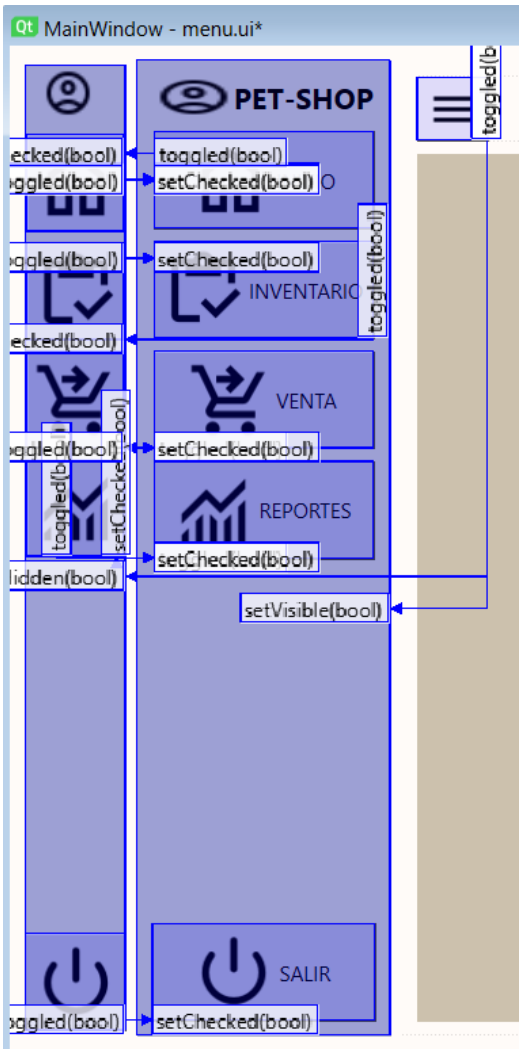

Figura 2: Conexion entre elementos dentro de la interfaz grafica

Aqui tenemos algunas funciones a destacar

setCheckable(bool): Este método se utiliza para especificar si un elemento del menú es seleccionable o no. Si se establece como True, el elemento será seleccionable y tendrá un estado de verificación, lo que significa que el usuario puede marcar o desmarcar el elemento. Si se establece como False, el elemento no será seleccionable.

setChecked(bool): Este método se utiliza para establecer el estado de verificación de un elemento del menú. Si el elemento es checkable (seleccionable), setChecked(True) lo marcará, y setChecked(False) lo desmarcará.

setVisible(bool): Este método se utiliza para controlar la visibilidad de un elemento del menú. Si se establece como True, el elemento será visible en el menú desplegable. Si se establece como False, el elemento estará oculto y no será visible para el usuario en el menú.

En el caso de las diferentes ventanas dentro de la interfaz es suficiente con usar un stack widget. Un "stacked widget.es útil cuando se quiere mostrar diferentes vistas o pantallas en una misma área

de la interfaz de usuario, pero solo quieres que una de ellas sea visible a la vez. Por ejemplo, podría tener un formulario de registro con varias páginas. Para nuestra interfaz nos sevira para poder tener las paginas de cada uno de los botones del menu mostrado anteriormente:

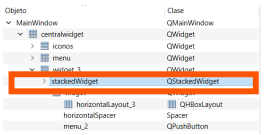

Figura 3: elemento para poder almacenar varias paginas

Aunque la parte del diseño juega un papel crucial dentro del desarollo de la practica, es importante conocer el como estan implementados los elementos dentro del codigo, o mejor dicho, saber el como funcionara interiormente. Aunque esta primera pagina no requiere mucha ciencia para describir el funcionamiento, ya que todos los elementos son estaticos y no van a cambiar, es importante conocer como funcionan la parte del menu, veamos como esta implementado

```
        self.setupUi(self)


        self.inicio1.clicked.connect(self.activar_inicio1)
        self.inicio2.toggled.connect(self.activar_inicio1)
        self.inventario1.clicked.connect(self.activar_inventario1)
        self.inventario2.toggled.connect(self.activar_inventario1)
        self.comprar1.clicked.connect(self.activar_comprar1)
        self.comprar2.clicked.connect(self.activar_comprar1)
        self.acpetar_cambios.clicked.connect(self.activar_cambios)
        self.reporte1.toggled.connect(self.activar_reporte1)
        self.reporte2.toggled.connect(self.activar_reporte1)
        self.salir1.clicked.connect(self.close)
        self.totala=0
```

Figura 4: Conexion de la señales çlickedçon una funcion

Para empezar conectamos cada señal que puedan hacer los botones, es decir, al momento que se detecte un click sobre estos botones, vamos a mandar a llamar una a una funcion que se encargara de realizar la accion, para esta primera parte solo nos activara alguna pagina del stack widget :

```
def activar_inicio1(self, checked):
    if checked:
        self.stackedWidget.setCurrentIndex(0)
```

Figura 5: Funcion para activar una pagina del stacked widget

Aqui podemos ver que al llevar acabo el evento se manda a llamar la respectiva funcion de cada boton, aqui entra en juego el concepto de indice del stacked widget, el cual empieza con cero, entonces la pagina numero 1, tendra el indice cero

Ahora continuemos con nuestra segunda ventana de la interfaz grafica:

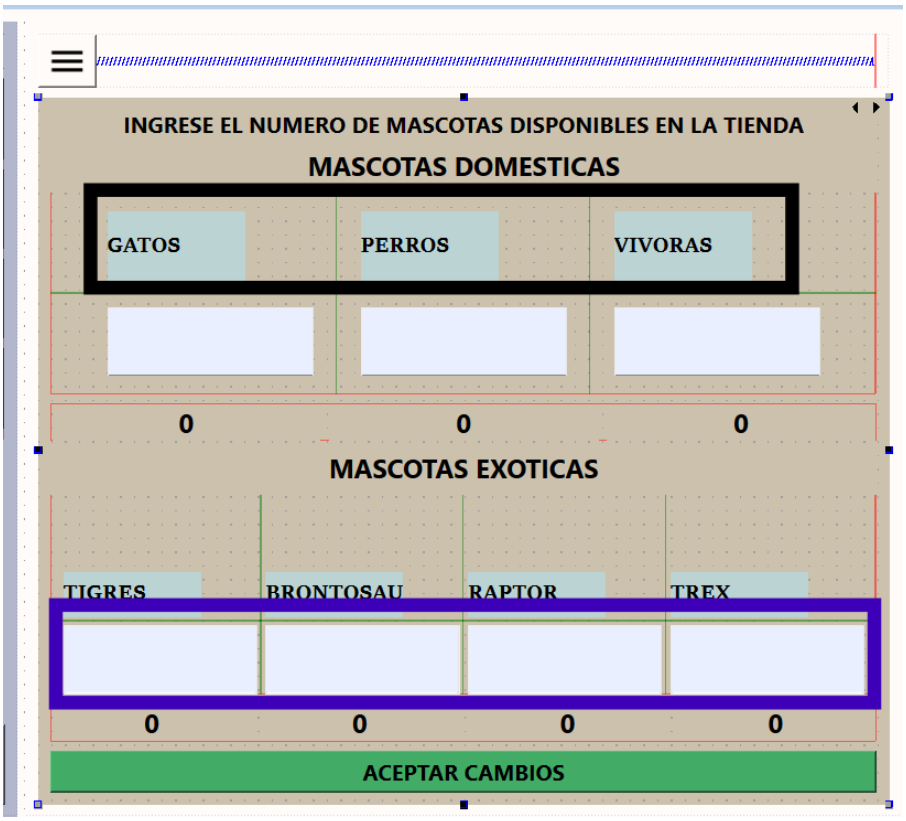

Figura 6: Interfaz para poder configurar el inventario

En este caso podemos observar que los elementos encerrados en el recuadro de color negro son elementos de tipo QLabel, se mantendran estaticos y no se actualizaran, tambien hay algunos elementos en blanco, los cuales son de tipo EditText, en estos recuadros el usuario ingresara el numero de mascatos de cada tipo que tiene, ademas los valores que en un inicio estan marcados con cero, se actualizaran al numero ingresado cuando se presione el boton de color verde .ªCEPTAR CAMBIOS"
Veamos que hay en la implementacion del codigo para esta ventanda, recordando que la señal clicked ya esta conectada a la funcion siguiente:

```
def activar_cambios(self):
    self.total_perros.setText(self.num_perros.toPlainText())
    self.total_gatos.setText(self.num_gatos.toPlainText())
    self.total_tigres.setText(self.num_tigres.toPlainText())
    self.total_vivoras.setText(self.num_vivoras.toPlainText())
    self.total_bronto.setText(self.num_bronto.toPlainText())
    self.total_raptor.setText(self.num_raptor.toPlainText())
    self.total_rex.setText(self.num_rex.toPlainText())
```

Figura 7: Funcion de aceptar cambios

Para esta funcion es algo sencillo, solo actualizar valores dentro de los QLabels self.totalperros.setText(self.nump
Esto actualiza el texto del widget totalperros con el contenido del widget de entrada de texto numperros.

De forma similar aplica para el resto de QLabels

Ademas de este metodo, se agrego otro metodo el cual tiene como funcion ser un filtro, donde analizara la entrada del teclado y en caso de ingresar una tecla diferente a un numero, mandara un mensaje de error, la funcion de filtro es la siguiente:

```
147     def eventFilter(self, obj, event):
148         if event.type() == QEvent.Type.KeyPress:
149             key = event.key()
150             if key < Qt.Key.Key_0 or key > Qt.Key.Key_9:
151                 QMessageBox.warning(self, "Advertencia", "Por favor, ingrese solo números.")
152                 return True
153         return super().eventFilter(obj, event)
```

Figura 8: Metodo para filtrar la entrada de teclado

Una vez que hemos hecho las primeras 2 paginas, pasaremos a la tercera la cual se muestra a continuacion:

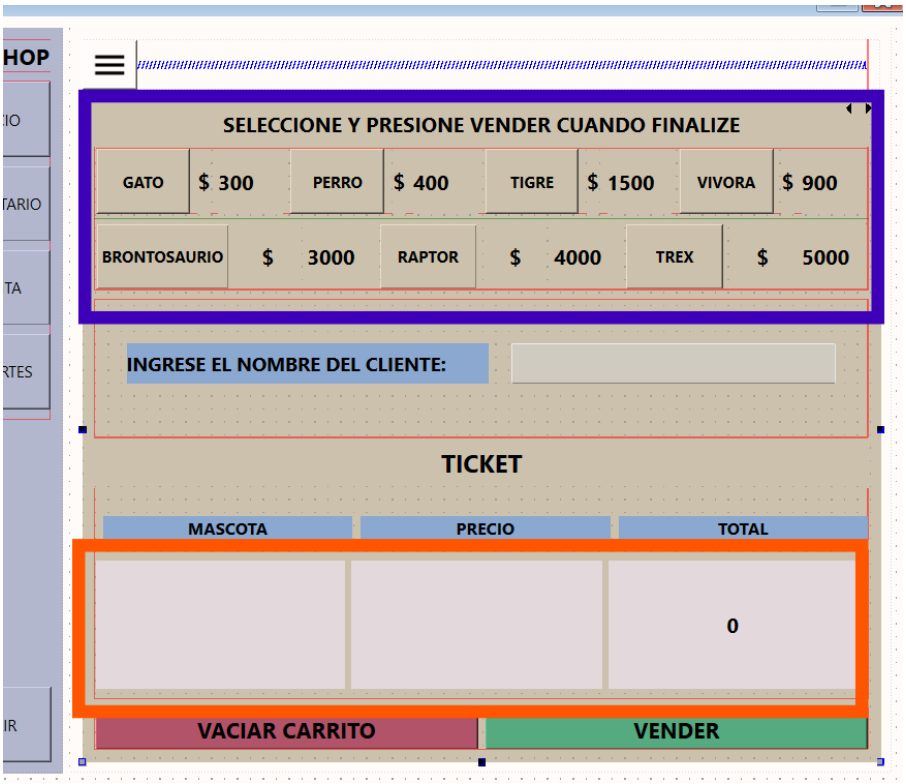

Figura 9: Ventana para llevar acabo las ventas de la tienda

En esta ventana observemos que tenemos las mascotas disponibles en la parte superior(recuadro de color azul), en la parte inferior (recuadro rojo) tenemos 3 elementos que se iran actualizando conforme se agreguen los elementos al carrito, los cuales simularan una especie de ticked, que posteriormente se imprimira.
En la parte de la implementacion del codigo tendremos lo siguiente:

```python
def agregar_text_al_ticket(self,nueva_mascota):
    cadena=self.ticket_mascota.text()
    nueva_cadena=f"{cadena}\n{nueva_mascota}"
    self.ticket_mascota.setText(nueva_cadena)

def agregar_mascota_al_ticket(self,precio_mascota):
    texto_actual_precio = self.ticket_precio.text()
    nuevo_texto_precio = f"{texto_actual_precio}\n{precio_mascota}"
    self.ticket_precio.setText(nuevo_texto_precio)
```

Figura 10: Actualizar los elementos en el carrito

agregartextalticket(self, nuevamascota): Esta función agrega un nuevo texto al widget ticketmascota. Toma el texto actual del widget, concatena el texto de la nueva mascota al final, y luego actualiza el texto del widget ticketmascota con la nueva cadena.

cadena = self.ticketmascota.text(): Obtiene el texto actual del widget ticketmascota.

nuevacadena = f"cadenanuevamascota": Crea una nueva cadena que consiste en el texto original del widget ticketmascota seguido de un salto de línea y el texto de la nueva mascota.

self.ticketmascota.setText(nuevacadena): Establece el texto del widget ticketmascota como la nueva cadena creada.

agregarmascotaalticket(self, preciomascota): Esta función agrega un nuevo precio al widget ticketprecio. Similar a la función anterior, toma el texto actual del widget, concatena el precio de la nueva mascota al final, y luego actualiza el texto del widget ticketprecio con la nueva cadena. textoactualprecio = self.ticketprecio.text(): Obtiene el texto actual del widget ticketprecio.

nuevotextoprecio = f"textoactualpreciopreciomascota": Crea una nueva cadena que consiste en el texto original del widget ticketprecio seguido de un salto de línea y el precio de la nueva mascota.

self.ticketprecio.setText(nuevotextoprecio): Establece el texto del widget ticketprecio como la nueva cadena creada.

Para el metodo que se activara cuando se presione un boton con el nombre de la mascota tendremos funciones parecidas a la siguiente con cada boton

```python
def venta_gato(self):

    self.totala=float(self.total.text())
    self.agregar_mascota_al_ticket(self.gato_precio.text())
    self.agregar_text_al_ticket(self.gato_venta.text())
    self.total.setText(str(self.totala+float(self.gato_precio.text())))
```

Figura 11: Metodo para venta de mascotas

self.totala = float(self.total.text()): Obtiene el texto actual del widget total, lo convierte a un valor flotante y lo asigna a la variable self.totala.

self.agregarmascotaalticket(self.gatoprecio.text()): Llama a la función agregarmascotaalticket (que explicamos anteriormente) para agregar el precio del gato al ticket. Toma el texto actual del widget gatoprecio y lo pasa como argumento a la función.

self.agregartextalticket(self.gatoventa.text()): Llama a la función agregartextalticket (también explicada anteriormente) para agregar el texto relacionado con la venta del gato al ticket. Toma el texto actual del widget gatoventa y lo pasa como argumento a la función.

self.total.setText(str(self.totala + float(self.gatoprecio.text()))): Actualiza el texto del widget total sumando el precio del gato al valor previamente almacenado en self.totala. Primero convierte el texto

actual del widget gatoprecio a un valor flotante, luego suma este valor al valor previamente almacenado en self.totala, y finalmente convierte el resultado de vuelta a una cadena antes de establecerlo como el nuevo texto del widget total.

Una vez que hemos explicado el funcionamiento de las ventanasd e la interfaz grafica, procederemos a vizualizar los resultados

4. Resultados.

En esta parte de resultados vamos a ejecutar nuestro archivo .py, con las implementaciones en el codigo, anteriormente mencionadas

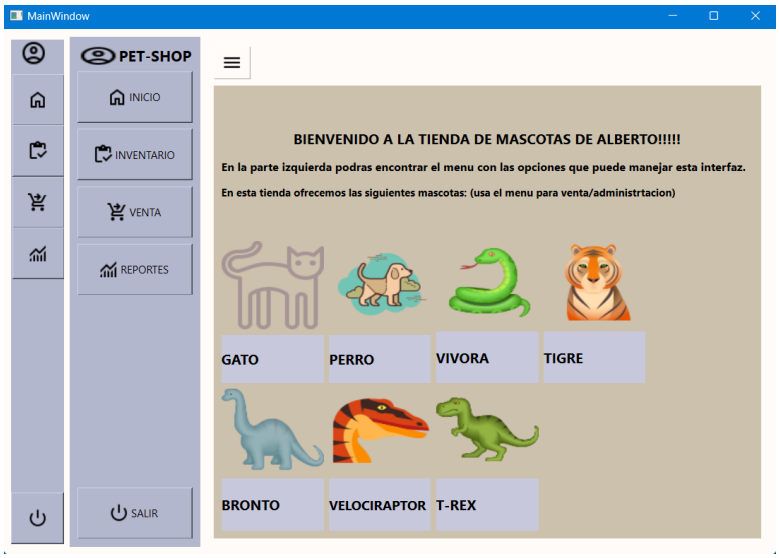

Figura 12: Ventana de inicio de la interfaz

Figura 13: click en el simbolo de menu

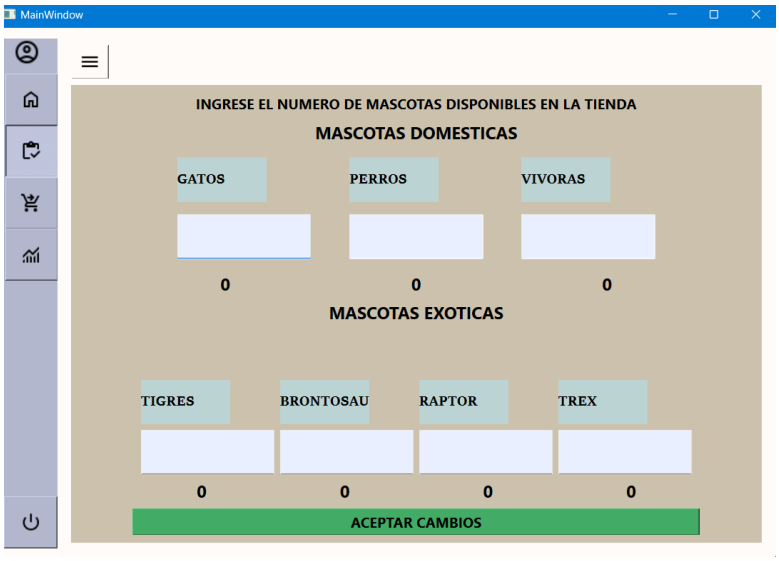

Figura 14: Ventana para modificar el inventario.

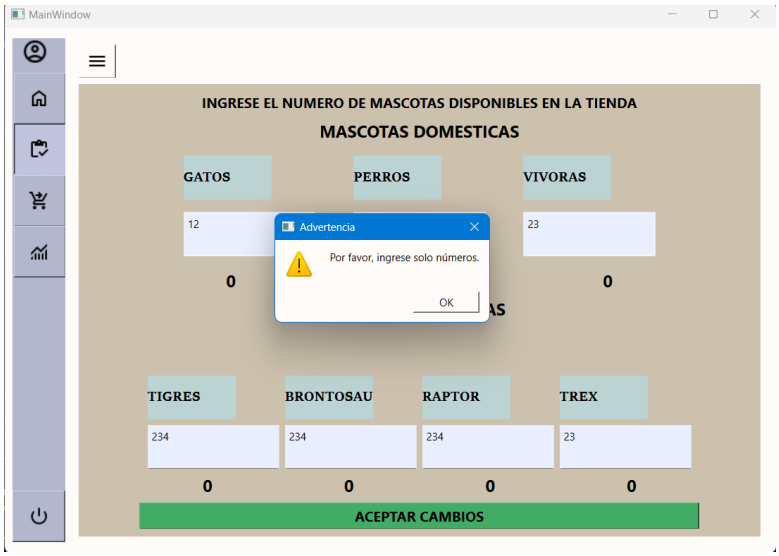

Figura 15: Intento de igresar dato erroneo

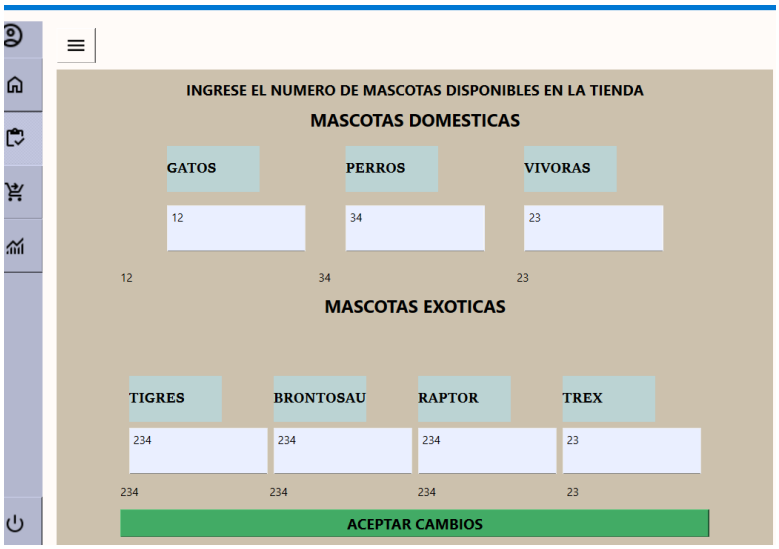

Figura 16: Actualizacion en el numero de animales, una vez presionado ACEPTAR CAMBIOS

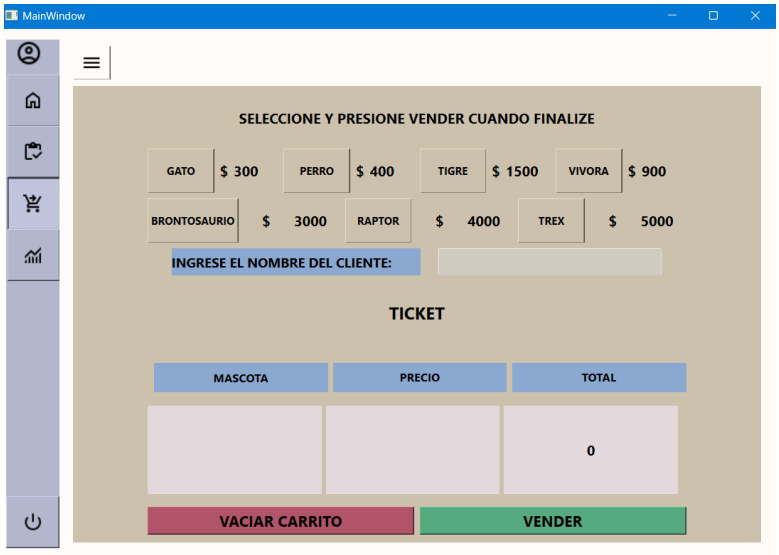

Figura 17: Ventana para la opcion de vender mascotas

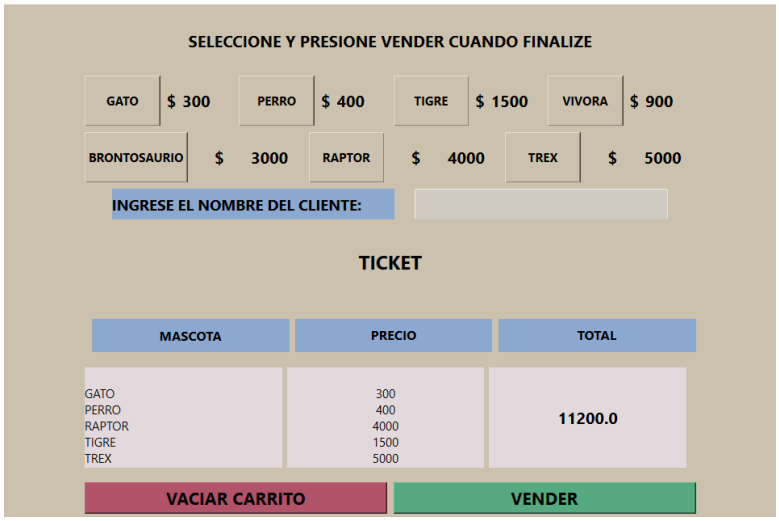

Figura 18: Agregar mascotas al carrito

Figura 19: Intentar presionar el boton de vender, sin ingresar el nombre del cliente.

Figura 20: Una vez ingreado nos muestra el ticket de venta

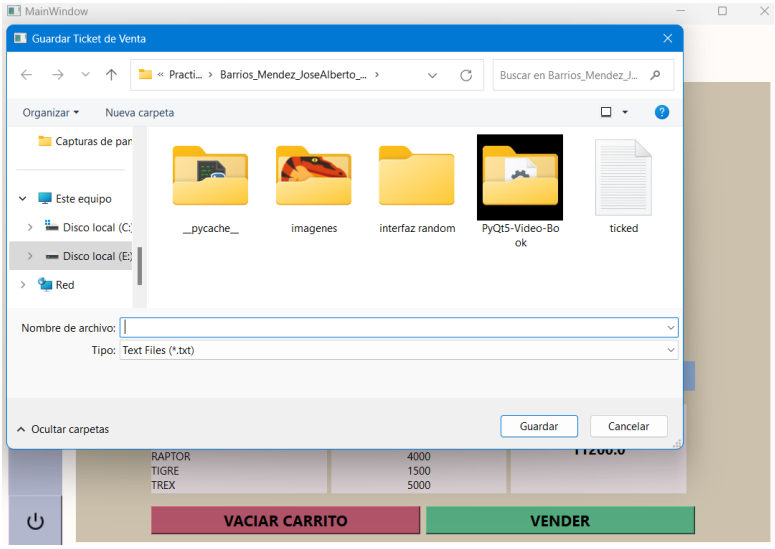

Figura 21: Al presionar OK, nos manda a una ventana para poder guardar el ticket como archivo .txt

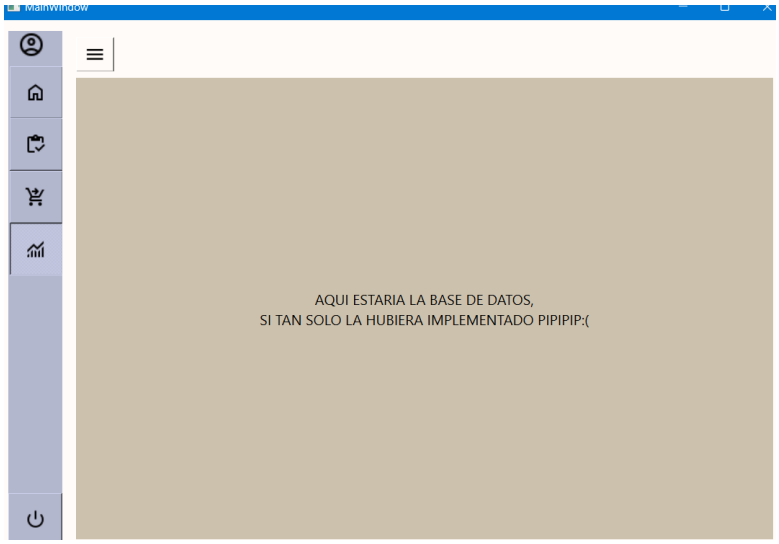

Figura 22: No medi bien mi tiempo profe:(

5. Conclusiones.

Al realizar esta practica me e dado cuenta que la forma de realizar las interfaces graficas es un proceso sumamente complicado, en el ambito de lograr obtener un buen diseño, ya que es necesario considerar o verlo desde un punto de vista a nivel de usuario, es decir, imaginar que tu eres el usuario para poder plaenear de una mejor forma la interfaz, y de esta forma lograr obtener una buena interfaz. De esta forma se llega a la clonclusion que lo mas complicado es hacer el diseño, de igual forma, conforme estas avanzando con la interfaz te das cuenta de que mas cosas se le puede agregar, o al menos en mi caso, fue sobre la marcha donde me iba dando cuenta que necesitaba algun otro elemento para poder hacer mas facil la implementacion del codigo, sin embargo por falta de administracion de tiempo, eh dejado imcompleta esta practica, pero aun asi me quedo con el conocimiento obtenido para lograr los avances en la interfaz