Universidad Nacional de Colombia

Sede Medellín

Objetos – 3004599-1

Trabajo Práctico 2- Valor: 20%

Manejo de interfaces graficas de usuario y Excepciones

Profesor Jaime A. Guzmán

Responsable de Prácticas: Jaime Guzmán
Fecha de entrega: 29 de junio del 2022.
Fecha de Recibo y Sustentación: 29 de junio al 1de julio de 2022.

Objetivo

Con la realización de este problema el alumno se familiarizará con los conceptos vistos en clase y los temas específicos a investigar sobre el manejo de Interfaces Gráficas de Usuario-GUI utilizando para ello la funcionalidad provista por las clases de la API Tkinter de Python.

INTRODUCCIÓN

En esta práctica se retomará el código correspondiente a la práctica 1 y se deberá implementar el desarrollo de su interfaz gráfica (todo en Python) que permita realizar las funcionalidades del sistema de dicha primera práctica. Este trabajo consta de las siguientes partes:

PRIMERA PARTE (CARACTERISTICAS DE LA INTERFAZ) (Valor 60 %)

La aplicación deberá permitir que el usuario utilice su respectiva interfaz gráfica para editar y consultar la información manejada por el sistema.

La aplicación tendrá los siguientes componentes básicos:

- Ventana de inicio
  - Permite brindar información del sistema y saludo de bienvenida
- Ventana Principal del usuario. En esta se manejan las funcionalidades del sistema planteadas para este trabajo de práctica.

A continuación, se detallará cada ventana.

1. Ventana de inicio (valor 20%**)**

Esta ventana deberá permitir:

- Brindar un saludo de bienvenida al sistema (Izquierda superior: P3)
- Breve hoja de vida de cada desarrollador. Cada hoja de vida cambia por click del ratón sobre la región del texto de la hoja de vida. (Derecha superior: P5)
- Cada vez que se cambie la hoja de vida de cada desarrollador como se describe en el apartado anterior, se deberá mostrar en P6 4 fotos de cada uno de los desarrolladores haciendo uso del posicionamiento Grid dentro de P6.
- Permitir el ingreso al sistema por medio de un botón que al dar click ira a la siguiente ventana (Ventana Principal de cualquier usuario) (Deberá ocupar la parte inferior de P4)
- Imágenes asociadas al sistema. Se podrán cambiar por un evento del ratón al pasar sobre la misma región de la foto. Presentar 5 imágenes. (La imagen deberá ocupar la parte superior de P4)

Los anteriores elementos deberán estar distribuidos en esta ventana. Esta Ventana se debe implementar usando *Frames* anidados. Su distribución será así:

Figura 1. Ventana Principal de inicio

Para su implementación utilizar los controladores Label, Entry, Button y Text. Para el manejo de imágenes usar los elementos asociados a tk.PhotoImage.
Esta ventana manejará un **menú** (como el visto en clase) llamado ***Inicio*** quien tendrá las siguientes opciones de menú:

- Salir de la aplicación
- Descripción del sistema (con esta aparecerá en alguna parte de la ventana de inicio una breve descripción de lo que hace el sistema)

2. Ventana principal del usuario (Valor 40%**)**

2.1. Distribución ventana del usuario:

La ventana principal del Usuario deberá estar distribuida como lo indica la figura 2. Debe implementarse usando *Frames anidados* y no mediante posiciones fijas. Esta ventana deberá llevar como título de la aplicación (zona 0 en rojo de la figura1).
La ventana tendrá una zona de menús que permitirán implementar los menús y funcionalidades desarrolladas en la práctica 1 (zona 1 en rojo de la figura 1). La otra zona es donde se implementarán los componentes de interfaz que manejan la información necesaria de los procesos o consultas que implementen las funcionalidades del sistema asociada a cada usuario (zona 2 en rojo de la figura 2).

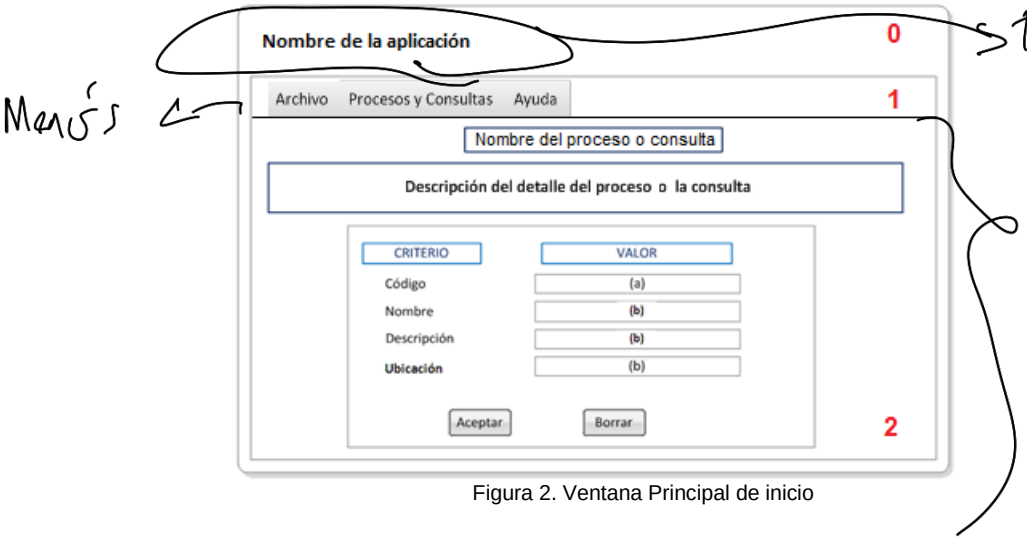


Figura 2. Ventana Principal de inicio

En las dos siguientes secciones se describen las zonas 1 y 2 de la interfaz.

2.2. Menú superior (Zona 1 de la interfaz)
Su estructura será la siguiente:

- Archivo

- Aplicación: Se despliega una ventana de diálogo con la información básica de lo que hace la aplicación.
- Salir: retorna a la Ventana de Inicio del programa.

- Procesos y Consultas
  - Listara todos los procesos y consultas que permite la aplicación (incluida las 5 funcionalidades solicitadas) acorde a la primera práctica.
- Ayuda
  - Acerca de: muestra una ventana de diálogo con los nombres de los autores de la aplicación. Formato libre.

2.3. Zona de interacción usuario (Zona 2 de la interfaz)
La parte inferior de la interfaz mostrará tanto los diálogos de texto (solicitudes de información al usuario) como los resultados de procesos y/o consultas.

2.3.1. Los diálogos de texto:
En algunas etapas de la aplicación se emplean diálogos que contienen formularios parecidos al mostrado en la parte inferior de sección 2 de la figura 1. Es decir, una sucesión de campos donde se introducen los valores solicitados.

Para tal fin, se pide implementar este tipo de formulario como un componente genérico. El componente creado debe cumplir los siguientes requisitos que se describen a continuación.

Se deberá implementar un componente ***FieldFrame*** que herede de ***Frame*** para visualizar y gestionar listas de atributo-valor. La interface pública deberá ser como mínimo:

```
class FieldFrame(Frame):
/**
crea un nuevo objeto de tipo FieldFrame
@arg tituloCriterios titulo para la columna "Criterio"
@arg criterios array con los nombres de los criterios
@arg tituloValores titulo para la columna "valor"
@arg valores array con los valores iniciales; Si 'None', no hay valores iniciales
@arg habilitado array con los campos no-editables por el usuario; Si 'None', todos son editables
*/
def __init__(tituloCriterios, criterios, tituloValores, valores, habilitado):
        ...

/**
@arg criterio el criterio cuyo valor se quiere obtener
@return el valor del criterio cuyo nombre es 'criterio'
*/
Def getValue(self, criterio):
        ...
```

En el constructor el componente deberá crear un Entry y Label por campo introducido y deberá alinearlos dinámicamente utilizando un posicionamiento Grid.

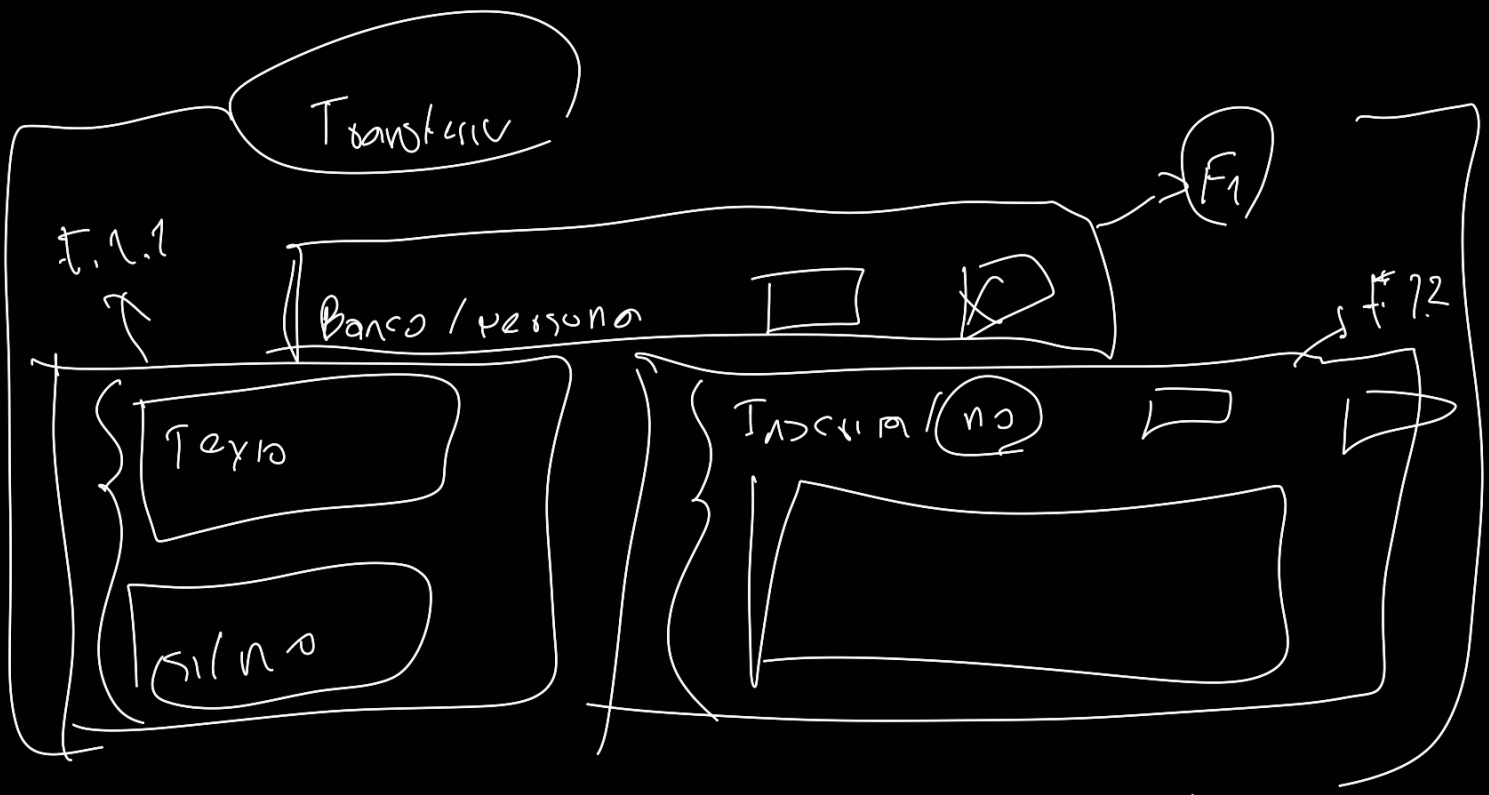
Un ejemplo de uso del componente es el siguiente fragmento de código, que generaría la parte superior del diálogo anterior. Los botones podrían formar parte de un panel adicional situado bajo el *FieldFrame*:

```
criterios = [Codigo", "Nombre", "Descripción", "Ubicación"]
fp = FieldFrame ("Criterio", criterios, "Valor", null)
```

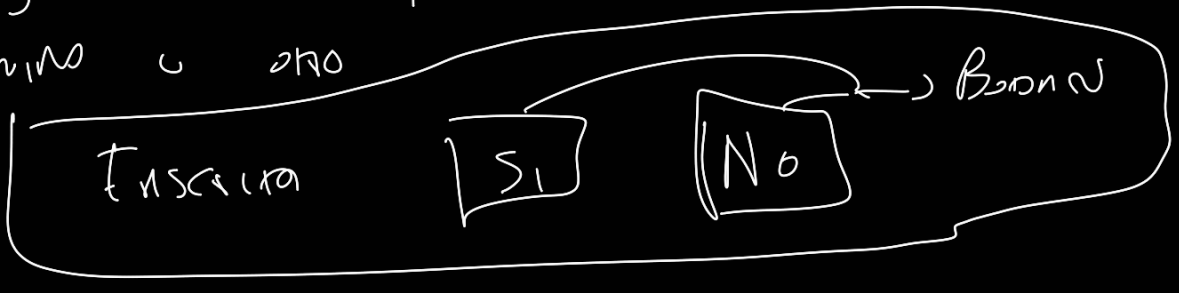
Este componente se debe utilizar en todos los diálogos generados por el menú.

Notas:

- En general se debe tener en cuenta que si existe por algún motivo el uso de códigos de cualquier clase (Identificador de un objeto) deberán ser generados automáticamente por el sistema y estos no podrán ser editados por ningún usuario (ver caso a en la figura 2). En cambio, para el caso de Nombre, Descripción (entre otros) este sí podrá ser editado por el Usuario (ver



Se puede crear otra clase "Field Option" que tenga cobrado y dos botones para decidir un camino u otro



caso b en la figura 2). Basado en esto modifique su código original de la práctica 1 para que haga esto.

- Mire que en el campo de los valores de Texto en el caso de tipo (b) se deberá manejar una rutina donde verifique que el dato que se entra si existe previamente en los objetos de su clase correspondiente asociadas a ellas. En caso de que no exista deberá lanzar una **excepción** creada por usted donde le indique al usuario que no existe el tipo de dato que acaba de entrar y que por favor entre nuevamente el valor (utilice para esto una ventana emergente nueva de advertencia).

Para el caso de los botones, el Aceptar se emplea para guardar los datos. Este botón verificará que todos los textos tengan un valor, y en caso de que falte alguno se disparará una **excepción** en la que por medio de una ventana de advertencia se le indicará al usuario cuales campos le faltan por llenar. El Botón Borrar permite borrar todos los campos de los cuadros de Texto.

2.3.2. Muestra de resultados de procesos y/o consultas:

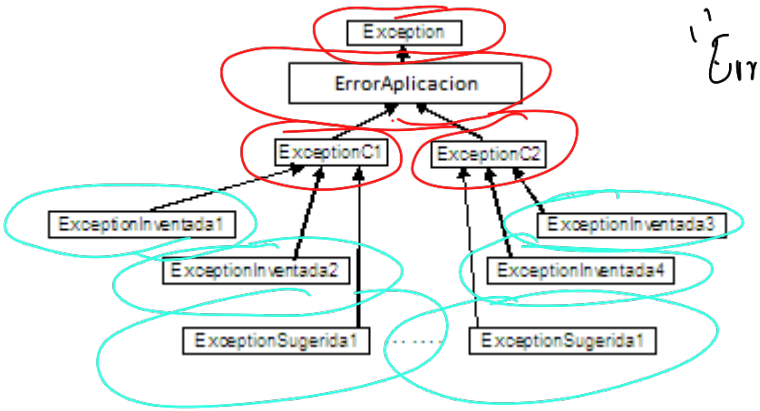
La parte inferior de la ventana de la aplicación mostrará los resultados de los procesos y las consultas. La lista (que es de tipo texto) deberá aumentar de tamaño ocupando todo el espacio disponible cuando se redimensiona la ventana. Se debe considerar además que existen algunos procesos que pueden necesitar de otras ventanas para representar la actividad. Este tipo de procesos y/o consultas deben considerar en su nueva ventana, una estructura similar a la ya planteada, donde se presente un menú superior y una lista de resultados.

Nota sobre la zona ~~de componentes:~~ al ingresar a esta ventana por primera vez, se deberá mostrar ~~la~~ ventana anterior donde el formulario para el ingreso de la información es reemplazado por una "**Interfaz de inicio**" que tiene un formato libre y deberá darle información al usuario de cómo usar esta aplicación y que se puede hacer. Una vez que se ejecute uno de los menús de procesos y consultas desaparece la anterior interfaz de inicio y aparece lo mostrado en la sección 2 de la figura 2**.**

SEGUNDA PARTE (REQUERIMIENTOS ADICIONALES) (Valor 20 %)

1. Manejo de Errores

Para el manejo de errores se deberá hacer una jerarquía de errores empezando por una clase de error llamada **ErrorAplicacion** que es hija de la clase ***Exception***. Además de la clase **ErrorAplicacion** se implementarán de manera original otras dos clases de errores que agrupen 4 tipos de errores (2 por cada tipo) que se deberán inventar (originales) y que serán lanzados desde algún método. Esta jerarquía deberá tener en cuenta lo siguiente: Cuando se lance una excepción la clase ErrorAplicacion aportará la primera parte del mensaje de error: "Manejo de errores de la Aplicación:" y a continuación la clase particular que pertenecerá a las ramas del árbol de errores adicionará el respectivo complemento del mensaje de error. Cada una de estas excepciones inventadas hará una rutina deferente de manejo de errores. Por último, se deberán implementar 2 Excepciones de las sugeridas en la **nota final** del numeral **2.3.1.** Estas últimas excepciones deberán hacer parte de alguna de las dos categorías de errores propuestas por usted. El árbol de errores quedará así:



Exception 1 = Excepciones ~~Sistema~~ Aplicación

Exception 2 = Excepciones Controladas

Error de Aplicación

ExceptionEjecución

1. Error de llenado en Field Frame
2. Error de tipo
3. Error de extracción de cuenta con # cuenta y tipo cuenta

(Hay que hacer un Try-Except al intentar extraer la cuenta de la lista cuentas del banco)

ExceptionLogic

1. No tiene crédito
2. No tiene saldo
3. La cuenta interior ya no está disponible en la listacuentas y por lo tanto no se DEBE realizar la transferencia
4. Error de ingreso de cédula en inserción

2. Requerimientos Faltantes
Hasta aquí están relacionados todos los requerimientos que se desean para esta aplicación. Si existen requerimientos que a su juicio hacen falta o existen inconsistencias que son vitales para el buen funcionamiento del sistema, estos deberán ser consultados en los horarios de consulta con el profesor titular de la materia quien luego los discutirá a nivel del grupo durante las clases teóricas.

Es necesario para este trabajo que completen el trabajo de la práctica 1 y si es necesario realicen algunas adaptaciones para que se acoplen a estas especificaciones. Por lo anterior los requerimientos de la práctica 1 son también material evaluable en esta segunda práctica.

TERCERA PARTE (Manual de usuario) (Valor 10 %)

Se debe retomar el manual de usuario de la practica 1 y modificar el manual de usuario con las interfaces gráficas, describir las funcionalidades complejas junto con sus interfaces y el modo de usarse. En general se debe actualizar la memoria escrita en base a esta entrega.

IMPORTANTE!!

FORMA DE EVALUACIÓN

- Aplicación (Primera y Segunda parte del trabajo): 80%
- Manual de usuario explicando que hace cada ventana y consultas (Tercera Parte): 10%
- Sustentación individual: 10%

NOTA 1: La sustentación del Trabajo se realizará en la fecha indicada y se realizará en forma individual mediante un examen oral o escrito acerca de la aplicación desarrollada por el grupo.
NOTA 2: Para este trabajo, la conformación de los equipos es la misma de la primera práctica.

ANEXO 1
INSTRUCCIONES GENERALES PARA LA PRESENTACIÓN DE LOS TRABAJOS

Normas generales

- Los trabajos serán presentados en los mismos grupos conformados para la práctica 1
- **No se admiten trabajos similares, en caso de presentarse programas con códigos similares serán anulados**. La totalidad del trabajo solicitado al alumno, en esta evaluación práctica deberá ser original y propio del autor que lo presenta. El estudiante es responsable de evitar que su material evaluable (código, solución al problema, memorias, etc.) sea accesible a estudiantes de otros grupos. En caso de que se detecten copias por incumplimiento de estas reglas, por acción o inacción, la sanción afectará a todos los estudiantes involucrados: quienes copien y quienes hayan sido copiados.
- Se hará un seguimiento y control individualizado de la realización de los trabajos el día de la sustentación. En este sentido, los integrantes de cada grupo deberán ser capaces de explicar y responder preguntas sobre el trabajo realizado. Recuerden la asistencia puntual al horario de la evaluación.
- El plazo de entrega de los trabajos será estricto. No se admiten trabajos después de esta fecha. **Si el código no compila correctamente ó no se puede ejecutar durante el momento de su evaluación, el trabajo será considerado como NO ENTREGADO**.

Normas del Material a Entregar

- Se deberá entregar en **Google Classroom** un archivo ZIP (con el siguiente nombre del archivo: **practica2-grupoXX-EquipoYY.zip, las xx representan el número del grupo y las yy el equipo**) que contenga lo siguiente:
  - Una carpeta src/: que contiene todas las fuentes (los .py), en su propia estructura de directorios. En los archivos fuentes se incluirá la siguiente documentación:
    - Cabecera del archivo: funcionalidad del módulo, autores, componentes del módulo, etc.
    - Cabeceras en las clases, explicando su finalidad y describiendo las estructuras de datos definidas cuando sean relevantes.
    - Cabeceras en los métodos, comentando su propósito y describiendo los parámetros de entrada/salida.
    - Comentarios en líneas de código de relevante interés o importancia.
    - Otros aspectos de interés a tener en cuenta por el profesor.
  - Desarrollar un archivo de ejecución (start.bat) que permita su ejecución por consola.
  - Archivo con la Memoria escrita del trabajo
- Implementar su código en GITHUB y compartirlo con el Profesor y el Monitor. Para esto es posible qué durante el tiempo del desarrollo de esta práctica, se programe una cita para solicitar una breve explicación de cómo adelantan su desarrollo y evaluar su trabajo en equipo.

  Deben trabajar sobre el repositorio que fue creado para la práctica 1, generando una carpeta con el nombre Python en donde será almacenado todo lo referente a la práctica 2.