Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación.

Curso: Taller de programación.

Grupo: 04.

PROGRAMA 3: PARQUEO.

Estudiante: Dina Monge Sandoval – 2022117025.

Profesor: William Mata Rodríguez.

Fecha de entrega: 21 de junio de 2022.

Semestre: I Semestre.

CONTENIDO

OBJETIVOS DEL PROYECTO

1. Continuar aplicando el ciclo completo de la metodología general de desarrollo de programas.
2. Aplicar y reforzar aspectos del lenguaje Python 3.
3. Aplicar buenas prácticas de programación básicas: documentación interna y externa del programa, reutilización de software, nombres significativos.
4. Validar los datos de entrada.
5. Usar archivos de datos.
6. Usar algún software de control de versiones de software, por ejemplo, Git o algún otro que usted decida.
7. Fomentar algunas habilidades investigativas (lectura-escritura, procesos cognitivos, trabajo colaborativo, socialización del conocimiento, obtención de información, análisis, motivación, conciencia del autoaprendizaje.

ENUNCIADO DEL PROYECTO.

REQUERIMIENTOS DE INFORMACIÓN

Iniciar con un menú principal implementado con botones para estas funcionalidades:

Configuración
Cargar cajero
Saldo del cajero
Ingresos de dinero
Entrada de vehículo
Cajero del parqueo
Salida de vehículo
Ayuda

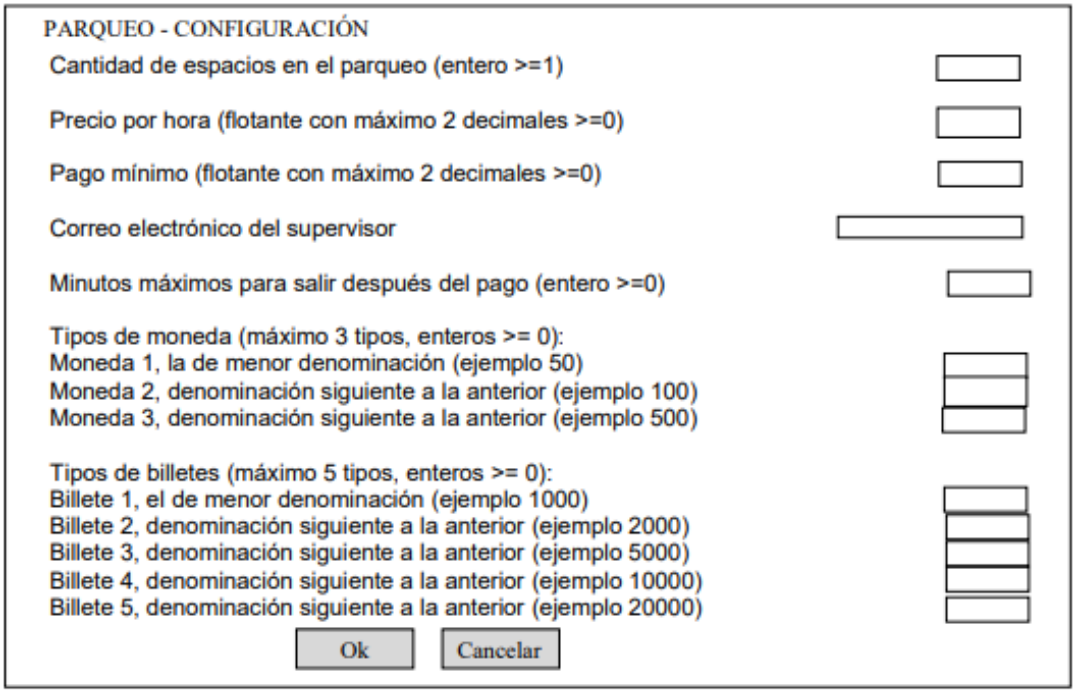
Usted puede agregar otras funcionalidades que vayan a mejorar el producto.

Se puede salir del programa usando la opción de cerrar "X" en la GUI.

Estando en cualquier ventana del programa se puede volver al menú principal usando la opción de cerrar "X" en la GUI.

Seguidamente se explica el detalle de las funcionalidades.

A) CONFIGURACIÓN



El correo electrónico es un string con un correo existente, debe verificar que este correo realmente exista ya que se usará para que el programa pueda enviarle mensajes al supervisor administrativo.

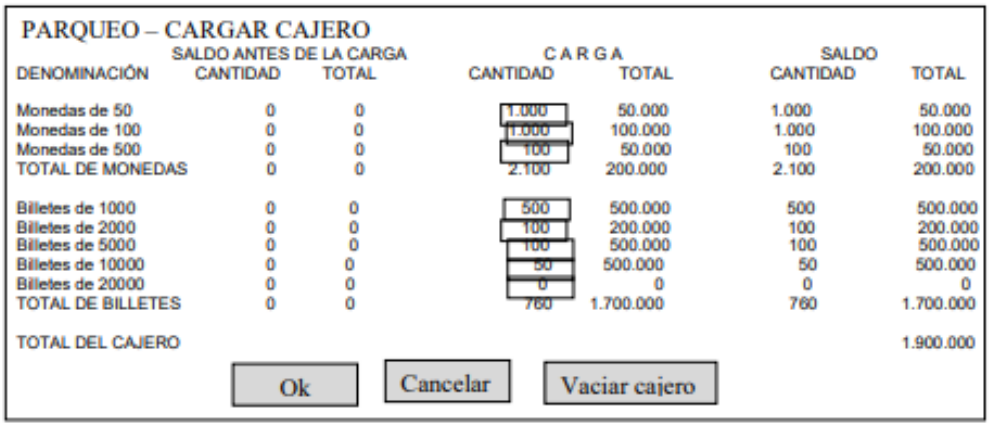
Botón **Ok**: los valores dados en esta ventana se deben mantener durante la ejecución del programa, sustituyen valores previos que tengan las variables de configuración.
Botón **Cancelar**: se cancelan, es decir, no se hacen cambios en los valores de las variables de configuración, ellas siguen manteniendo los valores que tenían antes de ingresar a esta ventana.
Luego regresa al menú principal.

Validaciones:

- Las denominaciones deben seguir las reglas indicadas: por ejemplo, si la moneda 1 es 50, la siguiente moneda debe ser > 50 o 0.
- Cuando un tipo de moneda sea 0 las siguientes también serán 0.
- Cuanto un tipo de billete sea 0 los siguientes también serán 0.

- Las denominaciones se pueden cambiar solamente cuando el cajero este vacío, es decir, saldos en cero para todas las denominaciones.
- Usted debe asegurarse que los datos registrados en la configuración permitan que los cambios (vueltos) que se deben dar al usuario se puedan hacer con esa configuración. Por ejemplo, no podría aceptar un pago mínimo de 75 pesos con una moneda de 100 sino tiene monedas de 25 en las denominaciones.
- Pueden hacerse cambios a la configuración solamente cuando el parqueo está vacío.

B) CARGAR CAJERO



Cuando el programa inicia el cajero empieza con cero en todos los saldos de cantidades de las diferentes denominaciones. Los saldos se van modificando con los pagos de los clientes y sus respectivos cambios, además de las cargas manuales que se hacen en esta función. Las cargas se pueden hacer en cualquier momento y cuantas veces se necesiten
La sección SALDO ANTES DE LA CARGA debe reflejar los valores que tienen las variables al momento de ingresar a esta función.
Este proceso actualiza los montos de dinero en el cajero para las diferentes denominaciones. En la sección CARGA se va registrando la CANTIDAD que se está cargando en el cajero por cada denominación. Conforme se da la cantidad se va actualizando la columna TOTAL junto con las columnas CANTIDAD y TOTAL de la sección SALDO.
Botón **Ok**: estable las cantidades de la sección SALDO como las cantidades actuales de cada denominación en el cajero, por tanto debe actualizar las variables que usa para tales efectos.
Botón **Cancelar**: las cantidades de las denominaciones no son modificadas con los valores registrados en la ventana, en su lugar se mantienen con los valores que tenían antes de ingresar a esta función.
Botón **Vaciar cajero**: todas las cantidades y totales se ponen en cero, esta ventana debe reflejar este hecho, es decir todas las columnas deben verse en cero. Luego esta opción debe desaparecer de la lista de opciones y vuelven a quedar disponibles las otras para que el usuario pueda escogerlas nuevamente.
Luego regresa al menú principal.

Validaciones:
- Cantidad (entero >=0).

C) SALDO DEL CAJERO

PARQUEO – SALDO DEL CAJERO

	ENTRADAS		SALIDAS		SALDO	
DENOMINACIÓN	CANTIDAD	TOTAL	CANTIDAD	TOTAL	CANTIDAD	TOTAL
Monedas de 50	1.000	50.000	0	0	1.000	50.000
Monedas de 100	1.009	100.900	4	400	1.005	100.500
Monedas de 500	100	50.000	0	0	100	50.000
TOTAL DE MONEDAS	2.109	200.900	4	400	2.105	200.500
Billetes de 1000	501	501.000	2	2.000	499	499.000
Billetes de 2000	100	200.000	0	0	100	200.000
Billetes de 5000	101	505.000	0	0	101	505.000
Billetes de 10000	50	500.000	0	0	50	500.000
Billetes de 20000	10	200.000	0	0	10	200.000
TOTAL DE BILLETES	762	1.906.000	2	2.000	760	1.904.000

Ok

Esta función se usa para consultar el dinero que tiene el cajero. La sección ENTRADAS contiene las cantidades que han entrado al cajero por los conceptos de cargas y pagos. La sección SALIDAS contiene las cantidades que han salido por concepto de cambios (vueltos). La sección SALDO contiene ENTRADAS – SALIDAS.
Botón **Ok**: regresa al menú principal.

D) INGRESOS DE DINERO

PARQUEO – INGRESOS DE DINERO
Del día dd/mm/aaaa
Al día dd/mm/aaaa

TOTAL DE INGRESOS EN EFECTIVO xxx.xxx.xxx
TOTAL DE INGRESOS POR TARJETA DE CRÉDITO xxx.xxx.xxx
TOTAL DE INGRESOS xxx.xxx.xxx

ESTIMADO DE INGRESOS POR RECIBIR xxx.xxx.xxx

Ok

Con esta función se calculan los ingresos reales y estimados durante un período de tiempo.
El usuario registra el período y el programa comienza con los cálculos.
Los ingresos de dinero (en efectivo y tarjeta) se calculan tomando todos los pagos que se han realizado en el período según el diccionario **parqueo** y la lista **detalle_de_uso**.
Los vehículos que no han salido físicamente pero que ya han realizado el pago están en el diccionario **parqueo**.
Los pagos de los vehículos que han salido físicamente están en la lista **detalle_de_uso.**

Para calcular el estimado de ingresos hay que considerar los vehículos que no han pagado: estos se encuentran en el diccionario **parqueo** y no tienen datos de pago. Los cálculos se hacen tomando el tiempo transcurrido desde su entrada hasta la fecha y hora que tenga en el momento el sistema.

Botón **Ok**: regresa al menú principal.

E) ENTRADA DE VEHÍCULO

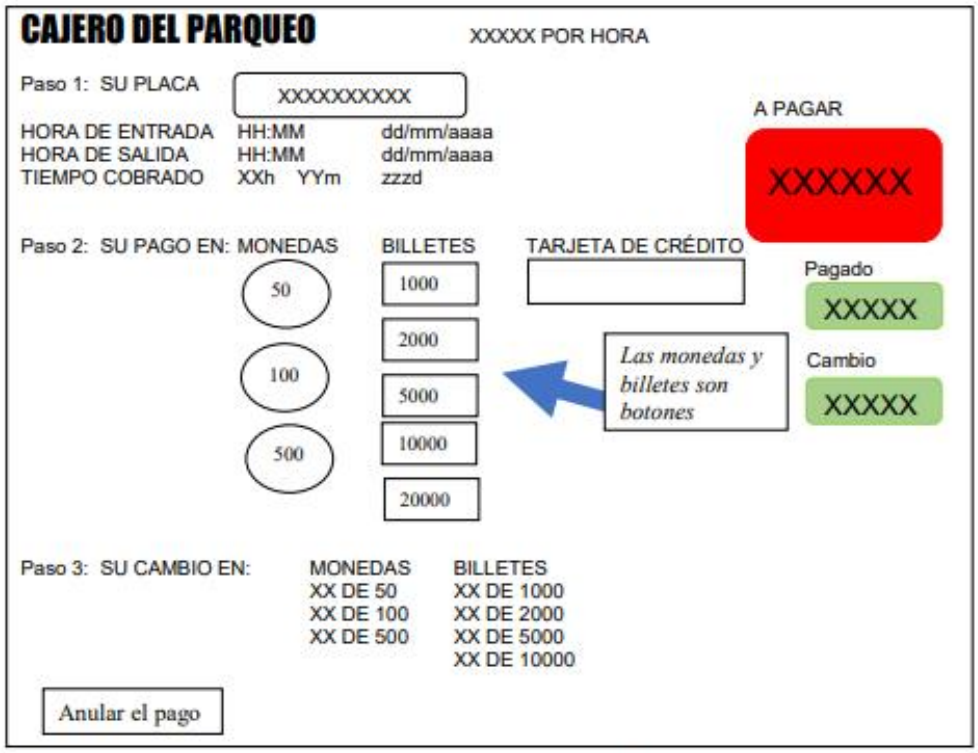
Puede entrar al parqueo si hay espacios disponibles. La hora y fecha de entrada se toma del sistema.

Botón **Ok**: reserva espacio. Los datos se registran en el diccionario **parqueo**.
Botón **Cancelar**: no reserva espacio.
Luego regresa a pedir otra placa.

Validaciones:

- La placa no debe estar en el diccionario **parqueo**, esto significaría que el vehículo ya está dentro del mismo.
- Calcular los espacios disponibles. Sino hay espacios desplegar el mensaje **NO HAY ESPACIO** (color rojo y letras que sobresalgan).

F) CAJERO DEL PARQUEO

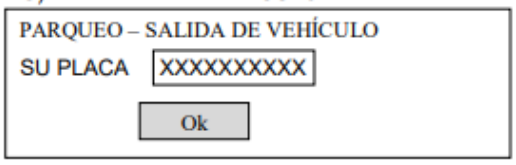


Esta operación simula "un cajero automático de parqueo".
Debe desplegar una ventana con los datos indicados aquí.
Se pide la placa para obtener datos de entrada y salida para cobro.
El cobro debe calcularse según la configuración y se despliega en la casilla A PAGAR.
El pago se puede hacer con monedas, billetes y tarjeta de crédito.
Para simular el pago en lugar de dar dinero físicamente se presionan los botones respectivos de MONEDAS, BILLETES y TARJETA DE CRÉDITO del paso 2. Por ejemplo, si va a pagar 350 pesos con 3 monedas de 100 y una de 50 debe presionar 3 veces el botón de la moneda de 100 y 1 vez el botón de la moneda de 50. Cada vez que se presiona un botón de dinero se acumula el monto correspondiente en la casilla **Pagado**.
El usuario puede presionar estos botones mientras lo **Pagado** sea menor que **A PAGAR**. En el momento que la casilla **Pagado** sea igual o mayor a la casilla **A PAGAR** ya no debe aceptar más pagos. En su lugar debe calcular y desplegar el cambio (vuelto) junto con su desglose de moneda (paso 3). También actualiza datos del pago en diccionario **parqueo**.
En caso de pagar con una tarjeta de crédito debe posicionar el cursor en esa casilla y dar un número natural de 10 dígitos exactos. De la tarjeta se toma la diferencia que exista entre **A**

PAGAR y **Pagado**. No hay validación de aceptación de la tarjeta de crédito, vamos a asumir que siempre se hace.
Al finalizar el pago se debe crear un recibo que incluya los datos del paso 1 (formato PDF).
Botón **Anular el pago:** se puede presionar mientras no se haya hecho el pago total. Sirve para anular el pago devolviendo el dinero que ha pagado el usuario.
Después de registrar o anular el pago el programa vuelve al paso 1 para otro pago.

Validaciones:
- El cálculo del cambio con sus respectivas denominaciones (desglose de moneda) debe considerar todos los casos posibles. Siempre debe dar la mínima cantidad de denominaciones según los saldos que maneja el cajero. Considere diferentes casos, entre ellos:
  - Caso 1: No se puede dar el cambio. Ejemplo: A pagar 1.650, pago con 2.000, no hay monedas de 50 (saldo de estas monedas es 0): el cambio de 350 requiere 1 moneda de 50 pero no se puede porque no hay. En estos casos el pago no se acepta y debe devolver el pago que ha hecho el cliente exactamente con las mismas denominaciones. Se envía un correo electrónico al supervisor con un mensaje sobre la situación presentada para que se presente a cargar el cajero.
  - Caso 2: Dar el cambio pero no representa la mínima cantidad de denominaciones. Puede pasar cuando el saldo de algunas denominaciones no es suficiente. Por ejemplo: A pagar 1.700 con un billete de 2.000, en caso de haber solo 1 moneda de 100 los otros 200 de vuelto se pueden completar con 4 monedas de 50 si las hubiera.
- Luego de realizar un pago el programa debe enviar un correo al supervisor con las denominaciones que tengan una cantidad menor a 5.
- Considere que hay un pago mínimo.

G) SALIDA DE VEHÍCULO



El espacio ocupado por el vehículo se libera hasta que dicho vehículo haga la salida física. Esto es debido a que puede hacer el pago pero permanecer en el parqueo. Por eso mientras el vehículo no haga este proceso, se asume que sigue permaneciendo en el parqueo, ocupando el mismo espacio.
Botón **Ok**: actualiza diccionario **parqueo** (elimina elemento) y lista **detalle_de_uso** (se agrega elemento) según se mencionó anteriormente. Además, si no puede hacer la salida porque excedió el tiempo para ello, automáticamente al diccionario **parqueo** le agrega un elemento donde la fecha y hora de entrada se toman de la fecha y hora de pago del elemento eliminado, reflejando que es una nueva entrada del vehículo para que pueda hacer el pago en el momento que considere oportuno. Luego regresa al dato de placa para otra salida.
Validaciones:
- La placa debe estar en el diccionario **parqueo** y haber pagado.

- Tiempo transcurrido entre el pago y la salida física: si es mayor a lo permitido según la configuración, no se permite la salida del vehículo, en cuyo caso debe dar la siguiente información:

No puede salir porque excedió el tiempo permitido para ello.
Tiempo máximo para salir luego del pago xxxxxxxxxx
Tiempo que usted ha tardado xxxxxxxxxx
Debe regresar al cajero a pagar la diferencia.

H) AYUDA
Esta opción la usaremos para que el usuario pueda ver el Manual de Usuario directamente en la computadora (despliega el pdf respectivo).

ACERCA DE
Puede poner esta opción para desplegar información "Acerca del programa" donde pondremos al menos los datos del nombre del programa, la versión, la fecha de creación y el autor.

SALIR
Puede poner esta opción para salir del programa. También se puede salir con la opción de cerrar "X" en la GUI.

USO DE ARCHIVOS

Cada vez que inicie la ejecución del programa debe leer los datos que están en los archivos para asignarlos a las estructuras en memoria. La primera vez que se corre el programa los archivos están vacíos, luego se van actualizando con las operaciones que haga el usuario. Cada vez que termina la ejecución del programa debe grabar los datos en los archivos.

ARCHIVO **configuración.dat**
Contiene los datos de configuración.
Archivo tipo string por líneas: cada dato de configuración se debe poner en una línea.

ARCHIVO **parqueo.dat**
Contiene el diccionario parqueo.
Archivo binario usado con el módulo pickle.

ARCHIVO **detalle_de_uso.dat**
Contiene la lista detalle_de_uso.
Archivo binario usado con el módulo pickle.

ARCHIVO **cajero.dat**
Debe contener los datos que usted necesita para manejar el cajero. Usted define esos datos y el tipo de archivo que va a utilizar.

ASPECTOS A INVESTIGAR:

Como en todo trabajo o proyecto, se sabe que mucha de la información requerida para poder llevarlo a cabo completo y correctamente requiere que se destine algún tiempo para poder investigar aspectos que se hacen necesarios, así como desde luego revisar el material de apoyo del mismo curso. En esta ocasión a pesar de que se emplea un programa ya utilizado en repetidas actividades, se da importancia a indagar un poquito en los siguientes temas:

- Software de control de versiones.

Un sistema de control de versiones (o VCS, por sus siglas en inglés), también conocido como sistema de control de revisiones o de fuentes, es una herramienta de software que monitoriza y gestiona cambios en un sistema de archivos.

Asimismo, un VCS ofrece herramientas de colaboración para compartir e integrar dichos cambios en otros usuarios del VCS.

Al operar al nivel del sistema de archivos, un VCS monitorizará las acciones de adición, eliminación y modificación aplicadas a archivos y directorios.

Un repositorio es un término del VCS que describe cuando un VCS está monitorizando un sistema de archivos.

En el alcance los archivos individuales de códigos fuente, un VCS monitorizará las adiciones, eliminaciones y modificaciones de las líneas de texto que contiene ese archivo. Entre las opciones populares de VCS del sector de software, se incluyen Git, Mercurial, SVN y preforce.

¿Por qué necesito un software de control de versiones?

VCS es una valiosa herramienta con numerosos beneficios para un flujo de trabajo de equipos de software de colaboración. Cualquier proyecto de software que tiene más de un desarrollador manteniendo archivos de código fuente debe, sin duda, usar un VCS. Además, los proyectos mantenidos por una sola persona se beneficiarán enormemente del uso de un VCS. Se puede decir que no hay una razón válida para privarse del uso de un VCS en cualquier proyecto moderno de desarrollo de software.

GitHub

Github es un portal creado para alojar el código de las aplicaciones de cualquier desarrollador, y que fue comprada por Microsoft en junio del 2018. La plataforma está creada para que los desarrolladores suban el código de sus aplicaciones y herramientas, y que como usuario no sólo puedas descargarte la aplicación, sino también entrar a su perfil para leer sobre ella o colaborar con su desarrollo.

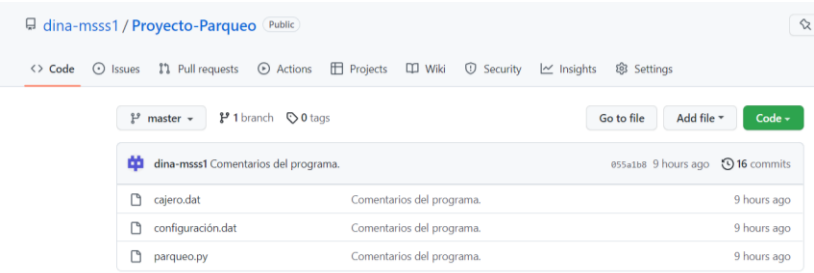
Github permite que los desarrolladores alojen proyectos creando repositorios de forma gratuita. Pero hay que tener una cosa en mente, y es que para poder subir gratis los proyectos deberán ser de código abierto. Y no quieres que tu aplicación sea de código abierto, la plataforma también tiene una versión de pago para alojar proyectos de forma privada.

Como te hemos mencionado más arriba, en Github también puedes entrar a los proyectos de los demás y colaborar para mejorarlos. Esto quiere decir que los usuarios pueden opinar, dejar sus comentarios sobre el código, colaborar y contribuir mejorando el código. También pueden reportar errores para que los desarrolladores lo mejoren.

Github también ofrece una serie de herramientas propias con las que complementar las ventajas que ya tiene el sistema Git de por sí solo. Por ejemplo, puedes crear una Wiki para cada proyecto, de forma que puedas ofrecer toda la información sobre él y anotar todos los cambios de las diferentes versiones.

También tiene un sistema de seguimiento de problemas, para que otras personas puedan hacer mejoras, sugerencias y optimizaciones en los proyectos. Ofrece también una herramienta de revisión de código, de forma que no sólo se pueda mirar el código fuente de una herramienta, sino que también se pueden dejar anotaciones para que su creador o tú mismo después si es tu proyecto las podáis revisar. Se pueden crear discusiones también alrededor de estas anotaciones para mejorar y optimizar el código.

A su vez, se pueden encontrar gráficos para ver cómo trabajan los desarrolladores en sus proyectos y bifurcaciones del proyecto, viendo las actualizaciones realizadas a partir de la primera versión o los cambios que se han realizado. Y por último también se incluyen características de redes sociales, como un sistema para seguir a tus creadores favoritos y ni perderte sus actualizaciones.



- Envío de correos electrónicos.

PyWin32 es un potente API que permite acceder a todo el sistema Windows: info del sistema, interface de usuario, sistema de archivos, registro, …

El módulo que nos permitirá controlar la interface gráfica de windows es win32gui. En este caso, vamos a ver un ejemplo sencillo y recuperar las ventanas de aplicaciones en ejecución.

Win32 es una librería para Python que nos permite interactuar y manipular funciones de sistema operativo Windows. Tiene muchos métodos y funciones que debes conocer. Win32 tiene funciones para manipular archivos, conexiones de redes, seguridad del sistema y muchas otras más.

- Nuevas características de tkinter exploradas en este proyecto.

1. Entry.

El widget Entry se utiliza para aceptar cadenas de texto de una sola línea de un usuario. Si desea mostrar varias líneas de texto que se pueden editar, debe usar el widget Text. Si desea mostrar una o más líneas de texto que el usuario no puede modificar, debe usar el widget Label.

```
self.label = Label(self.canvas_configuracion, text='Cantidad de espacios en el parqueo (entero >=1)')
self.label.place(x=10,y=30)
self.entrada_espacio = Entry(self.canvas_configuracion)
self.entrada_espacio.place(x=440, y=30, width=50)

self.label = Label(self.canvas_configuracion, text='Precio por hora (flotante con máximo 2 decimales >=0)')
self.label.place(x=10,y=70)
self.entrada_precio = Entry(self.canvas_configuracion)
self.entrada_precio.place(x=440, y=70, width=50)

self.label = Label(self.canvas_configuracion, text='Pago mínimo (flotante con máximo 2 decimales >=0)')
self.label.place(x=10,y=110)
self.entrada_pago = Entry(self.canvas_configuracion)
self.entrada_pago.place(x=440, y=110, width=50)
```

2. Grid.

El método grid nos permite posicionar los widgets en una celda en específico, indicamos la celda usando el índice de fila y columna correspondiente, el ancho y la altura de cada celda son configurables, además un widget puede ocupar varias celdas si lo deseamos, usando grid podemos crear fácilmente interfaces gráficas de usuario tipo formulario.

```
self.label_cargar = Label(self.canvas_cargar, text='DENOMINACIÓN')
self.label_cargar.grid(row=2,column=0)
self.label_denominacion_monedast1 = Label(self.canvas_cargar, text='Monedas de '+conf[5])
self.label_denominacion_monedast1.grid(row=3,column=0)
self.label_denominacion_monedast2 = Label(self.canvas_cargar, text='Monedas de '+conf[6])
self.label_denominacion_monedast2.grid(row=4,column=0)
self.label_denominacion_monedast3 = Label(self.canvas_cargar, text='Monedas de '+conf[7])
self.label_denominacion_monedast3.grid(row=5,column=0)
```

3. Button.

El botón es típicamente el control más común en una aplicación con interfaz gráfica. Un botón es un recuadro con un texto y/o una imagen que puede ser presionado por el usuario para ejecutar una operación. En Tk está representado por las clases tk.Button y ttk.Button.

```
self.boton = Button(self.canvas_menu, text='Configuración', command=self.ventana_cambiar_configuracion)
self.boton.place(x=175,y=20, width=150)

self.boton = Button(self.canvas_menu, text='Cargar cajero', command=self.ventana_cambiar_cargar_cajero)
self.boton.place(x=175,y=60, width=150)

self.boton = Button(self.canvas_menu, text='Saldo del cajero', command=self.ventana_cambiar_saldo_cajero)
self.boton.place(x=175,y=100, width=150)

self.boton = Button(self.canvas_menu, text='Ingresos de dinero', command=self.ventana_cambiar_ingresos_dinero)
self.boton.place(x=175,y=140, width=150)

self.boton = Button(self.canvas_menu, text='Entrada de vehiculo', command=self.ventana_cambiar_entrada_vehiculo)
self.boton.place(x=175,y=180, width=150)

self.boton = Button(self.canvas_menu, text='Cajero del parqueo', command=self.ventana_cambiar_cajero_parqueo)
self.boton.place(x=175,y=220, width=150)

self.boton = Button(self.canvas_menu, text='Salida de vehículo', command=self.ventana_cambiar_salida_vehiculo)
self.boton.place(x=175,y=260, width=150)

self.boton = Button(self.canvas_menu, text='Ayuda', command=self.ventana_cambiar_ayuda)
self.boton.place(x=175,y=300, width=150)

self.boton = Button(self.canvas_menu, text='Acerca de', command=self.ventana_cambiar_acerca_de)
self.boton.place(x=175,y=340, width=150)

self.boton = Button(self.canvas_menu, text='Salir', command=self.ventana_cambiar_salir)
self.boton.place(x=175,y=380, width=150)
```

CONCLUSIONES

- Al igual que en el proyecto anterior es fácil concluir que cuando se desea llegar a un buen término un proyecto es necesario aplicar el ciclo completo de la metodología general, y en este caso desde el poder comprender que se requiere hacer, hasta el seguir con la secuencia fue vital; sin dejar de lado el tema de la planificación y buen uso del tiempo, que en esta ocasión no me fue suficiente.

- De igual modo se logró aplicar nuevos aspectos del lenguaje Python en este caso los conceptos relativos al GUI, de tal modo que se interiorizó su funcionalidad para poder desarrollar un programa como el de parqueo, ya que para poder programar la maquina se requiere emplear la interfaz gráfica y desde luego la cantidad de módulos que este conlleva, muchos de ellos utilizados anteriormente.

- Por otro lado, se debe reconocer que nuevamente la documentación interna y externa del programa hace que cualquier otra persona ajena a nuestro proyecto sea capaz de poder correrlo e interpretar cada paso, de igual modo como se manifestó en el proyecto anterior la documentación que se debe producir en este documento es clave para poder reunir todo lo que se debe lograr en el proyecto en especial lo que se pretende del programa al ejecutarse, así como los conceptos o la información que refuerzan los aprendizajes adquiridos no solo en clase sino como el complemento tras la constante investigación.

- Gracias a la validación de los datos de entrada, es que se logró poder correr el programa de forma completa sin presentar ningún inconveniente al garantizar que los datos que se solicitan o requieren fuesen los correctos. Se podría decir que es como el filtro de entrada a la información correcta.

- Se usaron en mi caso tres tipos diferentes de archivos, uno orientado al manual de usuario el que se adjunta y se abre como un pdf. Los otros dos son archivos .dat utilizados para guardar la información de la configuración y otro para el cajero.

- Finalmente, y no menos importante que se debe dejar claro es que por lo novedoso del proyecto al incluir todo lo que implica la parte gráfica, se requirió de buscar mucho apoyo para poder lograr que cada parte del programa se pudiera concretar y que al final corriera sin ningún inconveniente, desde luego esto significó, leer mucho, ver tutoriales y hasta preguntar a persona que podrían orientarnos acerca de cuál camino ir tomando.

- El usar la aplicación GitHub la cual sirvió para el control de versiones del programa, fue de gran importancia, ya que se mantiene un mayor orden y se puede llegar a observar los avances lo cual también a uno lo motiva a seguir, esta enseñanza va a ser de gran uso a futuro, para los próximos cursos de la carrera.

ESTADÍSTICAS DE TIEMPO

Actividad Realizada	Horas
Análisis del proyecto Leer y entender el proyecto Etc.	5
Diseño de algoritmos	10
Investigación de …	5
Programación	35
Documentación interna	2
Pruebas	15
Elaboración del manual de usuario	2
Elaboración de documentación del proyecto	4
TOTAL	78

LISTA DE REVISIÓN DEL PROYECTO

Concepto	Puntos originales	Avance 100/%/0	Puntos obtenidos	Análisis de resultados
Menú principal	2	100	2	
Configuración	10	100	10	
Cargar cajero	8	100	8	
Saldo del cajero	4	75	3	No se hizo la parte de salidas
Ingresos: Reales Estimados	 8 8	31,25	5	Se hizo solo la interfaz
Entrada de vehículo	5	40	2	Se hizo solo la interfaz
Cajero del parqueo paso 1	5	40	2	Se hizo solo la interfaz
Cajero del parqueo paso 2: Registrar pago Devolver el pago Enviar correos Recibo del pago	 5 3 5 4	29,41	5	Se hizo solo la interfaz
Cajero del parqueo paso 3	5	40	2	Se hizo solo la interfaz
Salida de vehículo: Eliminar elemento de parqueo Agregar elemento a detalle_de_usoEntrada automática por exceder tiempo de salida	 3 3 3	44,44	4	Se hizo solo la interfaz
Validación de datos	8	100	8	
Ayuda (manual de usuario desplegado en el programa)	5	100	5	
Archivos: Leer archivos Grabar archivos	 3 3	100	6	
TOTAL	**100**		62	
Funciones desarrolladas adicionalmente				