INSTITUTO TECNOLÓGICO DE COSTA RICA

TALLER DE PROGRAMACIÓN

GRUPO #3 y 4

Proyecto 3: documentación externa

DANIEL ALPIZAR BATISTA
CARNÉ: 2023063268

LEDVIN LEIVA MATA
CARNÉ: 2023071280

FECHA DE ENTREGA: 19 de junio, 7:30am

SEMESTRE #1
2023

NOMBRE DEL PROFESOR: WILLIAM MATA RODRÍGUEZ

Contenido:

- Temas investigados
- Conclusiones del trabajo
- Diseño de la solución
- Estadística de tiempos
- Lista de revisión del proyecto

Enunciado del proyecto: Revisión técnica vehicular

DEFINICIÓN DEL PROYECTO:

Desarrollar un programa para administrar una parte de un proceso de revisión técnica de vehículos llevada a cabo en una estación especializada. La revisión consiste en examinar vehículos para obtener un resultado de la revisión y si el vehículo es apto para circular se emite un certificado de tránsito. De no ser apto para circular este certificado no se emite, en su lugar el vehículo necesitará reinspección o podría ser que la revisión determine que el vehículo debe sacarse de circulación. El programa tendrá funciones para el registro de las citas de revisión de los vehículos, el ingreso de vehículos a la estación, un tablero que muestra en qué parte de la revisión está el vehículo dentro de la estación, el obtener los resultados de la revisión, una lista de posibles fallas y datos de configuración del sistema. Para efectos de esta especificación, cuando se considere necesario se hará referencia a los valores mostrados actualmente en la sección de configuración, por ejemplo, cuando se hable de cantidad de líneas de trabajo en la estación será 6 que es su valor actual. Tenemos entonces que en la estación hay 6 líneas de trabajo: una línea de trabajo es una cola (primero que entra, primero que sale) donde están los vehículos a revisar. En cada línea de trabajo el vehículo debe pasar secuencialmente por 5 puestos de revisión: - Puesto 1: revisiones generales (luces, llantas, vidrios, etc.) - Puesto 2: banco de amortiguadores - Puesto 3: sistema de frenado - Puesto 4: detector de holguras - Puesto 5: emisión de contaminantes En cada puesto puede haber solo un vehículo, y cuando sale del puesto puede estar sin fallas o con fallas leves o graves.
Al final de la línea de trabajo, después del puesto 5, un vehículo aprueba o no aprueba la revisión técnica. “Cola de espera”: hay una por cada línea de trabajo, antes del puesto 1 los vehículos están en esta cola esperando la entrada a ese puesto de revisión. “Cola de revisión”: hay una por cada línea de trabajo, cuando los vehículos entran a la revisión, a partir del puesto 1, pasan de la “cola de espera” a la “cola de revisión”. Esto causa una salida de la “cola de espera” y una entrada en la “cola de revisión”. Toda la información del sistema debe ser persistente, es decir, se debe preservar la información actualizada de tal manera que pueda estar disponible en cada corrida del programa, para ello puede usar archivos. El programa tendrá un menú principal implementado con botones (widget Button) desde el cual se accederán las funciones del programa. Cada opción debe ofrecer la posibilidad de regresar al menú anterior preservando la información registrada. Usted define la

interfaz gráfica siempre y cuando cumpla con los requerimientos funcionales del programa indicados seguidamente.

Temas investigados:

La herramienta utilizada a lo largo de este proyecto para la creación del manual de usuario fue tkinter. Tkinter es una librería de python que se utiliza muy frecuentemente para crear interfaces de usuario. Esta librería ya viene con python por lo que no es necesario instalarla. Sobre esta librería se utilizaron varios métodos para diferentes cosas. Como son los siguientes.

1) Método más básico de tkinter: Tk. Este método se utiliza para crear una ventana donde se pueden poner el resto de métodos como los botones o los label. Algunos otros métodos utilizados para editar este son: title, que es un método que permite cambiarle el título a la ventana y geometry que es un método que permite cambiar el tamaño de la ventana.

2) El segundo método utilizado es el de Label. Este método permite poner dentro de una ventana texto básico. A este se le pueden cambiar el texto, el tipo de fuente, el tamaño de letra y el color de las letras.

3) También existen métodos generales utilizados para posicionar todos los objetos dentro de las ventanas como pueden ser el de pack, grid, place entre otros. Los dos más utilizados durante este programa son el de grid y el de place. El método de place permite dar una x y una y dentro de la ventana donde el objeto al que se lo apliquemos estará siendo posicionado. Por otro lado, el método de grid divide el espacio en filas y columnas básicamente como si fuera una cuadrícula. Para este método se elige el número de fila y el número de columna en el que se quiere posicionar el objeto.

4) El siguiente método es el de button. Este tipo de objeto es un objeto al que el usuario puede clicar lo que le permite realizar una función cada vez que es clicado. Este método fue utilizado para que el usuario pudiera realizar ciertas interacciones como meterse en una ventana, devolverse, darle aceptar, entre otras.
5) Para las tablas se utiliza una forma de organizar los objetos bastante conveniente llamada frame. Este método permite crear algo parecido a otra ventana dentro de la ventana ya que se le pueden asignar objetos dentro y fue mayormente usada para más facilidad a la hora de organizar la interfaz gráfica.

6) Otro tipo de dato muy usado es el Entry que permite que el usuario pueda escribir algo como respuesta y esto se usó mucho para cuando era necesario obtener un dato concreto del usuario. Otros tipos de datos para el mismo fin menos utilizados son: Listbox y Checkbutton. También para obtener las

respuestas se utilizó el .get y para cambiar alguna especificación de algún objeto se utilizó el .config.

7) La librería SMTP en Python proporciona funcionalidades para enviar correos electrónicos utilizando el protocolo SMTP (Simple Mail Transfer Protocol). SMTP es un protocolo estándar utilizado para el envío de correos electrónicos a través de la red.La librería SMTP de Python se encuentra en el módulo smtplib. Proporciona una interfaz sencilla para establecer una conexión con un servidor SMTP, autenticarse (si es necesario), enviar correos electrónicos y gestionar los errores relacionados con el envío de los correos.

8) ReportLab es una biblioteca de Python que te permite generar archivos PDF de manera programática. Proporciona una amplia gama de funciones y herramientas para crear documentos PDF personalizados y profesionales.Con ReportLab, puedes crear documentos PDF desde cero o modificar y personalizar documentos PDF existentes. Puedes agregar texto, imágenes, gráficos, tablas y otros elementos a tus documentos PDF, y ajustar su diseño y formato según tus necesidades.

9) Para tratar los árboles se utilizó la programación orientada a objetos o POO. Las clases proporcionan una forma poderosa de organizar y estructurar el código, permitiendo la reutilización de código y la creación de abstracciones de alto nivel. Además, facilitan la creación de relaciones entre objetos y la aplicación de conceptos como herencia, encapsulación y polimorfismo, que permiten un diseño modular y flexible del código.

10) Para almacenar los datos de manera indefinida se utilizó el tipo de dato JSON (JavaScript Object Notation) es un formato ligero y de intercambio de datos ampliamente utilizado en el ámbito de la programación. Se basa en una sintaxis de objetos y arrays que es legible tanto para los humanos como para las máquinas. JSON es independiente del lenguaje, lo que significa que puede ser utilizado por una variedad de lenguajes de programación.
11) Variables de uso global, en programación, una variable de uso global es aquella que se declara fuera de cualquier función o bloque de código y puede ser accedida y modificada desde cualquier parte del programa, ya sea dentro de funciones, clases u otros bloques de código. Cuando se declara una variable como global, significa que su ámbito de visibilidad abarca todo el programa y no está limitada a un solo bloque o función. Esto permite que la variable sea compartida y utilizada por múltiples partes del programa sin necesidad de pasarla como argumento entre funciones o utilizar técnicas de alcance local. Es importante tener en cuenta que el uso de variables globales puede tener implicaciones en la legibilidad y mantenibilidad del código, ya que cualquier parte del programa puede modificar su valor. Se recomienda utilizar variables globales con moderación y en situaciones donde su uso sea

realmente necesario, optando en su lugar por el uso de variables locales dentro de funciones y el paso explícito de parámetros entre ellas.

12) El módulo os en Python es una biblioteca estándar que proporciona funciones para interactuar con el sistema operativo subyacente. Permite realizar diversas operaciones relacionadas con la gestión de archivos, directorios, variables de entorno, procesos y otras funcionalidades del sistema. Permite interactuar con el sistema operativo de manera portátil, lo que significa que las funciones funcionarán en diferentes sistemas operativos, como Windows, macOS y Linux, sin tener que preocuparse por las diferencias específicas de cada sistema. El módulo os también proporciona acceso a otras funcionalidades avanzadas del sistema operativo, como la manipulación de variables de entorno, la ejecución de comandos del sistema y la gestión de procesos. Esto hace que sea una herramienta poderosa para la automatización de tareas, la manipulación de archivos y la gestión del sistema en general.
13) El módulo time en Python es una biblioteca estándar que proporciona funciones relacionadas con la manipulación y medición del tiempo. Permite trabajar con operaciones de tiempo, como obtener la hora actual, medir el tiempo de ejecución de un programa y realizar pausas o retrasos en la ejecución de un programa. El módulo time se utiliza para interactuar con el reloj del sistema y proporciona varias funciones útiles. En resumen, el módulo time proporciona una forma conveniente de trabajar con el tiempo en Python, permitiendo realizar tareas relacionadas con la medición del tiempo, pausas en la ejecución y conversiones entre formatos de tiempo.
14) El uso de command=lambda en Python se refiere a la creación de funciones anónimas (también conocidas como funciones lambda) que se asignan como comandos a ejecutar en respuesta a eventos, como hacer clic en un botón en una interfaz gráfica o presionar una tecla. La función lambda toma uno o más argumentos separados por comas y devuelve el resultado de la expresión evaluada. Estas funciones son llamadas "anónimas" porque no se definen con un nombre como las funciones regulares. En el contexto de la interfaz gráfica, el uso de command=lambda permite asociar una función lambda como el comando a ejecutar cuando ocurre un evento específico, como hacer clic en un botón. El código dentro de la función lambda se ejecutará cuando se active el evento correspondiente.

Diseño de la solución:

El proyecto cuenta con una estructura de árbol binario implementada mediante clases para facilitar su manejo. Cada nodo del árbol contiene un diccionario que almacena los datos de cada cita, los cuales son fundamentales en todo el proyecto. Este árbol se convierte en la piedra angular del proyecto, ya que las demás estructuras, como listas y diccionarios predefinidos, giran en torno a él.

El árbol binario se utiliza principalmente para almacenar todos los datos relacionados con las citas. Estos datos son manipulados tanto de forma conjunta como individual a lo largo del proyecto mediante una estrategia de recursividad implementada al inicio, cuando se diseñó la clase del árbol binario.

Para asegurar un almacenamiento persistente de los datos, se utilizan varios archivos JSON en el proyecto. Estos archivos incluyen:

- cola_de_espera.JSON: Almacena los datos de la cola de espera, que consisten en los vehículos que están esperando para ser atendidos.
- cola_de_revision.JSON: Almacena los datos de la cola de revisión, que incluye los vehículos que están siendo revisados en ese momento.
- config.JSON: Contiene todas las configuraciones de la aplicación, como la cantidad de líneas, las horas de inicio y fin, y otros parámetros personalizables.
- fallas.JSON: Almacena todas las posibles fallas de los vehículos en la aplicación. Este archivo contiene una estructura de datos que mapea los números de falla con su descripción y tipo (leve o grave).
- citas.JSON: Almacena una estructura iterable del árbol binario, lo que permite guardar y cargar el estado del árbol en cualquier momento.

El uso de archivos JSON proporciona una forma flexible y escalable de almacenar los datos del proyecto de forma indefinida. Cada archivo JSON se utiliza para un propósito específico y contribuye al funcionamiento general del proyecto al permitir el almacenamiento y la recuperación de los datos necesarios en diferentes etapas del flujo de trabajo.

Conclusiones del trabajo:

La mayor parte del proyecto fue bastante sencilla no se nos complicó mucho. Uno de los problemas que encontramos al realizar este proyecto fue el botón de volver al menú ya que el orden de las funciones era muy importante a la hora de realizar esto y no podíamos averiguar una forma de hacerlo. La solución que encontramos fue poner la creación de la ventana del menú en el main y la creación de los botones en una función de esta forma no importaba el orden y todo funcionaba. Otro problema que tuvimos fue utilizar los árboles ya que no habíamos utilizado mucho estos y se nos complicó aprender a usarlos. En este programa utilizamos una forma de dividir las tareas del programa que se usa profesionalmente para de esta forma poder tener un primer contacto a como sería realizar un proyecto para una empresa de verdad. Esta forma de división es la de front-end y back-end. Esto se refiere a la parte gráfica y la parte lógica del programa y fue la división inicial que se hizo empezar este proyecto.

Estadística de tiempos:

Actividad Realizada	Horas
Análisis del problema	4
Investigación	4
Realización de interfaz gráfica	10
Realización validaciones	2
Funciones de los botones	3
Elaboración de tablero	4
Organización de datos	5
Documentación interna	1
Pruebas	5
Elaboración del manual de usuario	1
Elaboración de documentación del proyecto	1
Total	40 horas

LISTA DE REVISIÓN DEL PROYECTO

Concepto	Puntos	Avance 100/%/0	Puntos obtenidos	Análisis de resultados
Programación de citas implementando ABB con recursión	25	100%	25	
Asignación manual de citas	2	100%	2	
Asignación automática de citas	5	100%	5	
Cancelar citas	2	100%	2	
Ingreso de vehículos a la estación	5	100%	5	
Despliegue del tablero de revisión	10	100%	10	
Ejecución de los comandos del tablero	12	100%	12	
Registro de fallas por cita de revisión	5	100%	5	
Resultado de la revisión en PDF	5	100%	5	
Certificado de tránsito en PDF	3	100%	3	
Envío de correos electrónicos (cita, resultado de la revisión)	5	100%	5	
Lista de fallas (CRUD)	10	100%	10	
Configuración del sistema	6	100%	6	
Ayuda: manual de usuario en pantalla	5	100%	5	
TOTAL	100	100%	100	

Referencias:

• tkinter-Python interface to Tcl/Tk (n.d.). Python Documentation. https://docs.python.org/3/library/tkinter.html
• Python Software Foundation. (2021). Tkinter - Python interface to Tcl/Tk. Recuperado de https://docs.python.org/3/library/tkinter.html
• Python Software Foundation. (2021). 7. More Control Flow Tools. Recuperado de https://docs.python.org/3/tutorial/controlflow.html#lambda-expressions
• Python Software Foundation. (2021). time — Time access and conversions. Recuperado de https://docs.python.org/3/library/time.html
• Python Software Foundation. (2021). os — Miscellaneous operating system interfaces. Recuperado de https://docs.python.org/3/library/os.html