Instituto Tecnológico de Costa Rica
Profesor: William Mata Rodríguez
Curso: Taller de Programación – CI

Programa 3: ReTeVe

Estudiantes: Jose Mario Jiménez Vargas - 2023102334
John Sebastián Ceciliano Piedra - 2023267790
Fecha de entrega: lunes 19 de junio del 2023

TEC | Tecnológico de Costa Rica

Contenido

TEC | Tecnológico de Costa Rica

Enunciado del proyecto

El proyecto consiste en desarrollar un programa para administrar el proceso de revisión técnica de vehículos en una estación especializada. El programa tendrá las siguientes funciones:

Registro de citas de revisión:

Permite registrar las citas de revisión de los vehículos, incluyendo la información relevante como fecha, hora y datos del propietario del vehículo.

Se podrán asignar citas de forma manual o automática, considerando la disponibilidad de fechas y horas. Los datos registrados incluirán información del vehículo y del propietario. El programa utilizará un Árbol Binario de Búsqueda para almacenar las citas asignadas. Se realizarán validaciones para evitar citas duplicadas en un mismo día y se enviará un correo electrónico al solicitante con la fecha y hora asignadas. El objetivo es facilitar la gestión de las citas de revisión técnica de vehículos de manera eficiente.

Cancelar citas:

La opción de cancelar una cita permite al usuario ingresar el número de la cita y el número de placa para cancelar una cita previamente registrada. El estado de la revisión se actualizará a "CANCELADA". Se realizan validaciones para asegurar que la cita esté en estado "PENDIENTE", que la placa coincida y que la cita exista. No es posible cancelar una cita si el vehículo ya se encuentra en la "cola de revisión". Si el vehículo está en la "cola de espera", se eliminará de allí. Antes de cancelar una cita, se deben realizar todas las validaciones correspondientes.

Ingreso de vehículos:

La opción de ingreso a la cita permite al usuario presentarse en la estación. Se requiere el número de cita y la placa del vehículo. El programa muestra información del vehículo y calcula el costo de la revisión. Se realizan validaciones como la fecha y hora de la cita, el estado de la revisión y la disponibilidad en las colas de revisión. Si pasa las validaciones, el vehículo se envía a la cola de espera en la línea de trabajo menos ocupada.

Tablero de revisión:

En la opción del Tablero de revisión, se muestra una tabla que representa los vehículos en cada puesto de trabajo. Se pueden ejecutar comandos para mover los vehículos entre los puestos y realizar acciones específicas. Los comandos pueden ser de avance, asignación de fallas, finalización del proceso de revisión y regreso al menú anterior. Se realizan validaciones para asegurar que los vehículos estén en los lugares correctos y se refleja la actualización en tiempo real en el tablero. También se generan documentos PDF con el resultado de la revisión y el certificado de tránsito en caso de aprobación. 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

Lista de posibles fallas:

Se implementa un diccionario para mantener una lista de fallas actualizada. Cada falla se identifica con un número y contiene una descripción y un tipo (leve o grave). Las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) se utilizan para administrar esta lista. Se puede agregar, consultar, modificar y eliminar fallas según sea necesario. Esto garantiza la integridad y la actualización de la información en el diccionario de fallas.

Configuración del sistema:

En la configuración inicial de la aplicación se establecen parámetros como la cantidad de líneas de trabajo (6), el horario de la estación (de 6:00 AM a 9:00 PM), la duración de cada cita de revisión (20 minutos), el máximo de días para la reinspección (30), la cantidad de fallas graves para sacar un vehículo de circulación (4), la cantidad de meses para mostrar citas disponibles (1), el porcentaje del IVA (13.0%), y una tabla de tarifas con diferentes precios para cada tipo de vehículo. Estos parámetros pueden ser modificados posteriormente, con algunas restricciones, y los cambios se aplican de inmediato.

Ayuda:

Esta opción desplegará el manual del usuario (o video).

Acerca de:

Esta opción la usaremos para desplegar información “Acerca del programa” donde pondremos al menos los datos del nombre del programa, la versión, la fecha de creación y autor.

Salir:

Esta opción se usa para salir del programa.

Utilidades:

El programa utilizará archivos para almacenar la información de manera persistente, asegurando que los datos estén disponibles en cada ejecución.

La interfaz gráfica del programa se implementará mediante botones que permitirán acceder a las diferentes funciones, manteniendo la información registrada y ofreciendo la posibilidad de regresar al menú principal.

El objetivo del proyecto es facilitar la administración de la revisión técnica de vehículos, mejorando la eficiencia y precisión del proceso, y brindando una herramienta intuitiva y funcional para los operadores de la estación especializada.

TEC | Tecnológico de Costa Rica

Temas investigados

Software de control de versiones y de trabajo colaborativo

¿De qué trata?	¿Cómo se usa?
El software colaborativo o groupware es una herramienta que asiste a los usuarios en las actividades diarias como la gestión de calendarios, agendas de contactos, correo electrónico, foros, listas de distribución, mensajería instantánea y otras funciones. El software de colaboración actúa como una plataforma desde la cual los usuarios pueden crear y modificar documentos con control de versiones, administrar contenido en línea, compartir recursos como calendarios y bandejas de entrada, y comunicarse a través de funciones de chat y mensajería. Es importante ya que es una herramienta esencial para optimizar la colaboración y el flujo de trabajo de equipos, especialmente aquellos que están geográficamente dispersos o trabajan de forma remota. Permite a los equipos compartir contenido en diferentes formatos, fomentando una comunicación eficiente y rápida a través de diálogos interactivos en lugar de correos electrónicos. Además, facilita la programación de reuniones diarias, la planificación de tareas y la obtención de comentarios e informes en tiempo real, lo que beneficia especialmente a los equipos remotos. Algunas herramientas para el trabajo colaborativo en el desarrollo de software: • GitHub • GitLab • Bitbucket • Jira Software • Azure DevOps	Para este proyecto utilizamos el software colaborativo GitHub. ¿Qué es GitHub? GitHub es una plataforma de desarrollo colaborativo que ofrece un control de versiones con Git, permitiendo gestionar repositorios de código. Facilita la colaboración entre equipos, el seguimiento de problemas, y ofrece integraciones y automatizaciones. Además, fomenta la colaboración comunitaria, permitiendo descubrir proyectos y contribuir a ellos. En resumen, GitHub es una herramienta esencial para trabajar en equipo, gestionar proyectos y compartir código de manera eficiente. Pasos para el uso de GitHub: 1. Descarga e instala la aplicación GitHub Desktop. 2. Inicia sesión en tu cuenta de GitHub en la aplicación. 3. Clona un repositorio seleccionando "Clone a repository" y elige el repositorio deseado. 4. Configura la ubicación del repositorio clonado en tu máquina local. 5. Realiza cambios en los archivos utilizando tu editor de código. 6. En la aplicación, selecciona los archivos modificados y proporciona un mensaje descriptivo en el campo "Summary". 7. Crea una confirmación (commit) para guardar los cambios de manera local en el repositorio clonado. 8. Sincroniza los cambios con el repositorio remoto seleccionando "Push origin".

Arboles Binarios de Búsqueda.

¿De qué trata?	¿Cómo se usa?
Es un Tipo de Datos Abstracto o TDA (Abstract Data Type), conformado por un grupo de elementos llamados nodos relacionados de forma jerárquica. La estructura inicia en un nodo (elemento) llamado raíz, y a partir de este nacen o se le asocian otros nodos, a su vez cada nodo en la estructura puede tener asociados otros nodos. Todas las relaciones entre los nodos son jerárquicas, es decir, un nodo o varios dependen de otro nodo.	Primero se debe inicializar el árbol con una lista vacía o None, esto para poder agregar la raíz al mismo. Para agregar nodos, incluyendo la propia raíz, se debe hacer que la función establecida para esto reconozca la lista vacía o None, ya que así podrá agregar un nuevo nodo en ese campo. Al agregar un nodo, se le agrega al propio nodo sus dos hijos (izquierdo y derecho). Para ordenar los nodos dentro del árbol, se debe hacer una comparación entre nodos con algún dato que esté incluido en ellos, en nuestro caso utilizamos la fecha y hora de la cita. Lo mismo para buscarlos, al buscarlos, el programa va comparando el dato comparador entre nodos para lograr encontrar el nodo deseado. En Python, al ser un TDA, es muy sencillo modificar un nodo, ya que al ser en realidad una lista, se podría cambiar un dato sin afectar el resto del árbol y sin muchos problemas.

Archivos.dat.

¿De qué trata?	¿Cómo se usa?
Estos archivos de datos con extensión “.dat” pueden contener diferentes tipos de información dependiendo del programa o sistema que los genere. La estructura y contenido de los archivos ".dat" varían y su interpretación y uso específico dependen del software, en nuestro caso estos archivos son utilizados para guardar información importante del proyecto, como el árbol de citas, lista de fallas, algunos valores para verificaciones etc.	1. **Apertura de archivos:** Utilizando la función open(), se puede abrir un archivo en modo de lectura, escritura o apendizaje. También se puede especificar el formato de codificación del archivo, como 'utf-8' para archivos de texto. 2. **Lectura de archivos:** Se pueden leer los contenidos de un archivo utilizando métodos como read(),

En Python, los archivos son utilizados para leer y escribir información en diferentes formatos, como texto, binario o datos estructurados. El manejo de archivos en Python permite realizar diversas tareas, como leer datos de entrada, almacenar resultados, interactuar con archivos de configuración y procesar grandes conjuntos de datos almacenados en archivos.	readline() o readlines(). Estos métodos permiten leer todo el archivo, una línea o todas las líneas, respectivamente. 3. **Escritura en archivos:** Los datos se pueden escribir en un archivo utilizando el método write() o writelines(). El método write() permite escribir una cadena de caracteres, mientras que writelines() puede escribir una lista de cadenas. 4. **Cierre de archivos:** Después de realizar operaciones en un archivo, es importante cerrarlo utilizando el método close() para liberar los recursos asociados al archivo y asegurarse de que los cambios se guarden correctamente.

Módulos.

¿De qué trata?	¿Cómo se usa?
Los módulos son archivos con extensión .py que contienen definiciones de funciones, clases y variables que se pueden reutilizar en otros programas. Los módulos se utilizan para organizar y estructurar el código, promoviendo la modularidad y reutilización. Se importan en los programas principales utilizando las declaraciones import o from. Los módulos pueden ser parte de la biblioteca estándar de Python o pueden ser creados por los propios desarrolladores para encapsular y reutilizar su propio código.	Aquí tienes un resumen de los pasos para instalar librerías en el CMD utilizando pip: 1. Abre una ventana de CMD. 2. Verifica que tienes Python instalado. 3. Opcionalmente, actualiza pip con el comando python -m pip install --upgrade pip. 4. Instala una librería específica con el comando pip install nombre_de_libreria. 5. Verifica la instalación importando la librería en tu programa Python. Aquí tienes un resumen de cómo utilizar una librería en tu programa Python: 1. Importa la librería al principio de tu programa utilizando la declaración import. 2. Utiliza las funciones, clases o métodos proporcionados por la librería según tus necesidades. 3. Explora la documentación de la librería para conocer todas las opciones disponibles y cómo utilizarlas correctamente. 4. Continúa desarrollando tu programa utilizando las funcionalidades de la librería importada.

Módulos importantes usados para el proyecto:

from tkinter import *
La librería tkinter en Python proporciona herramientas para crear interfaces gráficas de usuario (GUI). Se basa en la biblioteca Tcl/Tk y permite a los programadores desarrollar aplicaciones con ventanas, botones, cuadros de texto y otros elementos interactivos. tkinter es una librería estándar de Python, lo que significa que está disponible sin necesidad de instalaciones adicionales. Es ampliamente utilizada, multiplataforma y ofrece una forma sencilla de crear aplicaciones con una interfaz gráfica de usuario.

from validate_email_address import validate_email
La librería "validate-email-address" en Python proporciona funciones para validar direcciones de correo electrónico. Su objetivo es verificar si una dirección de correo electrónico tiene un formato válido y si existe. Utilizando la función "validate_email", se puede comprobar si una dirección de correo electrónico dada es válida o no y además si existe.

import smtplib
La librería smtplib en Python se utiliza para enviar correos electrónicos a través del protocolo SMTP. Proporciona funciones para establecer una conexión con un servidor SMTP, autenticarse si es necesario, componer y enviar mensajes de correo electrónico. Se utiliza junto con la librería email para crear y formatear el contenido del mensaje. En resumen, smtplib facilita el envío de correos electrónicos programáticamente desde una aplicación Python.

from docx import Document
La línea de código from docx import Document importa la librería python-docx en Python. Esta librería permite crear, modificar y leer archivos de documentos de Microsoft Word (.docx). Con python-docx, puedes realizar tareas como crear nuevos documentos, agregar contenido como párrafos y tablas, aplicar formatos y estilos, insertar imágenes y guardar los documentos en diferentes formatos. Es una herramienta útil para trabajar con documentos de Word utilizando Python de manera programática.

import win32com.client as win32
La librería win32com, que permite la comunicación con aplicaciones de Windows utilizando la tecnología COM. Con win32com.client, puedes automatizar aplicaciones de Microsoft Office, interactuar con programas de correo electrónico, controlar aplicaciones de Windows y automatizar tareas en aplicaciones empresariales, en el caso del proyecto se utiliza para transformar archivos DOCX en PDF. Esta librería es específica para entornos Windows y requiere las bibliotecas pywin32 correspondientes.

Diseño de la solución.

Estructuras utilizadas en el proyecto

Estructura	Implementación
Colas	En este proyecto se utilizaron las colas para implementar la lista de espera del tablero de revisión, ya que de esta manera los vehículos entrarían al tablero en orden de entrada a la lista. Esta cola contiene todas las placas (vehículos) que van a ingresar a la estación, de forma ordenada como una cola, es decir, en orden de izquierda a derecha conforme entran los vehículos. Esta estructura interactúa directamente con la matriz, ya que la matriz lee esta cola para mostrar el primer elemento en pantalla.

Estructura	Implementación
Diccionarios	En este proyecto se utilizaron los diccionarios para llevar un registro de valores introducidos durante el uso del programa, se utilizó para registrar: Fallas agregadas, el número de cita que se lleva por el momento, los números de falla que se han utilizado y las placas que están en un proceso de cita actualmente. Todas las llaves de estos diccionarios están ordenadas de izquierda a derecha, en orden a como se van agregando los datos. Las fallas agregadas contienen la descripción de la falla y la gravedad de esta dentro de una tupla. El número de cita contiene el número de citas que se han registrado hasta el momento, este se cambia siempre que se programa una nueva cita. Los números de falla utilizadas contienen los números de falla que se han registrado, el mismo se actualiza cada vez que sucede un cambio dentro de la opción de fallas dentro del programa. El registro de citas contiene las placas que están en un proceso de revisión en ese

	momento, esto para que una placa no se registre dos veces a la vez. La placa estará disponible para registrarse otra vez cuando la misma salga del tablero de revisión. Todas tienen conexión con todos los botones del programa, ya que funcionan para validar datos y que se tenga fluidez en el programa.

Estructura	Implementación
Matrices	Las matrices se utilizaron en este proyecto para la creación del tablero de revisión. En este caso, la matriz es de labels, los cuáles dependiendo del índice de fila y columna, leen el archivo de las líneas de trabajo, para obtener las placas que estén registradas en los puestos. Están ordenados dependiendo de la configuración, ya que, dependiendo de las líneas de trabajo, esa misma cantidad se le asignará para las filas de la matriz y las columnas siempre son 5 puestos de trabajo, pero nosotros le asignamos una más para mostrar también la placa que sigue en la lista de espera. Tienen una conexión con la lista de líneas de trabajo, ya que esta matriz lee esa lista para poder imprimirse correctamente en pantalla.

Estructura	Implementación
	Las listas se utilizaron en este proyecto para guardar datos básicos y no muy cambiantes, como lo son la configuración, la lista de colas, y el registro de fallas. También se utilizaron para guardar los nodos con los datos de las citas dentro de ellas. La configuración contiene los datos de la configuración y están ordenados conforme se pide en el programa, la misma cambia cuando el usuario cambia las configuraciones dentro del programa y se relaciona con todo el programa, ya que las configuraciones se usan para determinar ciertas funciones del programa.

Listas	La lista de colas contiene n sublistas, dependiendo de la configuración, cada sublista contiene tres datos: El número de línea de trabajo, la cola de espera, y una lista con 5 listas que cada una representa un puesto de trabajo, esta estructura está ordenada de menor a mayor en cuanto a las listas de trabajo. Este se relaciona directamente con la matriz, ya que la matriz lee esta lista para representarse en pantalla. El registro de fallas contiene las fallas que se están utilizando actualmente, esto es para validar que se pueda borrar una falla, no tienen un orden específico ya que a como se van asignando las fallas, se van agregando a la lista. Esta tiene una conexión con el botón lista de fallas, para validar que se pueda borrar una. Los nodos contienen todos los datos de la cita programada, están ordenados a como se ingresan en el programa, luego de esto, está listo para ingresarse en el ABB, por lo mismo tiene una conexión directa con el Árbol de búsqueda binaria.

Estructura	**Implementación**
Árbol de búsqueda binaria (ABB)	Esta estructura de ordenamiento se utilizó en el programa para guardar las citas que se van asignando en el programa de forma ordenada, ya que el programa lo pide de esta manera en la implementación del guardado de citas. El árbol contiene todas las citas que se han registrado, cuando hay un cambio en algún nodo, éste debe contener ese cambio para usos posteriores. El orden del árbol se basa en las fechas y horas de las citas registradas, es decir, al registrar una cita, el árbol va comparando la fecha y hora de las citas, si la cita asignada es menor o igual que el nodo actual, se va comparando hacia la izquierda, de lo contrario se compara hacia la derecha, esto lo hace con todos los nodos que se encuentra, hasta encontrarse con un nodo de valor None, el None quiere

	decir que ese hijo está listo para que se asigne una cita en él. El árbol tiene una relación directa con todo el programa, ya que el mismo necesita acceder a las citas programadas y generar cambios sobre ellas, a la vez, se necesita la información de las citas para generar certificados, por ejemplo, o para hacer validaciones en las demás estructuras, como los diccionarios y listas.

Archivos utilizados en el proyecto

Archivo	Implementación
arbol_cita.dat	Este archivo sirve para guardar los datos de las citas registradas (ABB). Contiene todas las citas que se han registrado, ordenadas por fecha y hora. Tiene relación con todos los archivos, ya que el programa, al ser en torno al ABB, interacciona con todas las funciones.

Archivo	Implementación
registro_arbol.dat	Este archivo sirve para tener un registro de los vehículos que están siendo revisados o por revisar, para que no se generen dos citas con la misma placa a la vez. Contiene los vehículos que están siendo revisados o están por revisar. Interacciona con el árbol, ya que contiene una parte pequeña de información de este.

Archivo	Implementación
lista_colas.dat	Se utiliza para guardar las líneas de trabajo con su cola de espera y sus puestos de trabajo. Contiene las líneas de trabajo, las colas de espera y los puestos de revisión, si fuese el caso, también contiene las placas que se encuentren en cada dato. Interacciona con la configuración del sistema, para saber cuántas líneas debe generar.

Archivo	Implementación
registro_fallas.dat	Se utiliza para saber cuales fallas están siendo utilizadas en ese momento. Contiene todas las fallas que se están usando en la tabla de revisión. Interactúa con el tablero para acceder a esta información, y también con la lista de fallas, para saber cuáles se pueden eliminar.

Archivo	Implementación
configuración_reteve.dat	Se utiliza para guardar las configuraciones del sistema. Contiene todos los datos de la configuración en orden de como se piden en el programa. Interactúa con todo el programa, ya que dependiendo de la configuración que se ponga, ciertas ventanas deben cambiar la información que muestran.

Archivo	Implementación
registros.dat	Se utiliza para registrar información de citas. Contiene el número de citas que se lleva por el momento y las fallas que se han utilizado, esto para validar las fallas que se pueden usar y también para asignar el número de cita a las mismas. Interactúa con el árbol de citas, para asignar el número de cita y también con la lista de fallas, para saber en cuáles se puede tomar acción.

Archivo	Implementación
lista_fallas.dat	Sirve para guardar en un diccionario las fallas que se han creado en el programa. Contiene todas las fallas registradas hasta el momento, este archivo cambia conforme las decisiones que tome el usuario sobre esta, cada llave contiene la descripción de la falla y su gravedad, la llave es el número de falla. Interactúa con los registros para saber cuáles números de fallas hay y con la lista de colas, para asignar fallas a las placas.

Conclusiones del trabajo

Problemas encontrados

Problemas encontrados	Solución al problema
Al crear el árbol, teníamos el problema de que costaba que se guardara bien el árbol completo.	Tuvo solución al crear un return en el programa, justo debajo de las condicionales de hijos, este retorna el árbol a como se lleva por el momento.
Teníamos problemas para buscar nodos dentro del árbol, esto porque dependemos de una información que no siempre se tiene (la fecha y la hora de la cita).	Tuvimos que crear un archivo que actúe como registro de citas que están en proceso actualmente, que guarda la placa, el número de cita, su fecha y su hora, esto para facilitar la búsqueda de nodos.
Se necesitaba algo para poder saber cuáles fallas se estaban utilizando sin tener que depender solamente del ABB.	Se creó el archivo registro_fallas.dat para poder validar esta opción.
No sabíamos cómo actualizar los datos de la matriz del tablero de revisión, para que el usuario pueda ver los resultados de los comandos que va ejecutando.	Se hizo que los labels de la matriz lean el archivo de lista de colas, para que representen la información en ellos, de esta forma, se cierra y vuelve a abrir la ventana del tablero, para que se muestren los cambios realizados.
Había que realizar muchos cambios en los nodos del árbol, ya que había tres opciones que lo requieren.	Se decidió hacer una función que ejecute una acción diferente dependiendo de lo que se necesite, ya que, si se incluyese todo en uno mismo, podría generar problemas de ejecución en la función.
El implementar que la configuración empiece a regir desde el momento que se cambian las cosas.	En lugar de que se cambien todos los datos desde el momento del cambio, el programa valida ciertas configuraciones al inicio del programa. Por lo que, al guardar las configuraciones cambiadas, se cierra el programa y cuando se abre de nuevo, se inicia con las configuraciones realizadas.
Implementar el cambiar de color una placa en el tablero cuando tiene una falla grave asignada.	Cuando se genera la matriz de labels, se valida que la lista de puestos de revisión sólo tenga un elemento, ya que al tener una falla grave le agregué otro elemento (“!”), que actúa como comprobación para saber que la placa tiene una falla grave.

Aprendizajes obtenidos

1) Aprendimos sobre la implementación de Árboles Binarios de Búsqueda, lo cual nos ha ayudado a organizar y gestionar eficientemente conjuntos de datos de manera jerárquica. Esta estructura de datos nos ha permitido realizar operaciones de búsqueda, inserción y eliminación de manera rápida y eficiente. Además, hemos aprendido a aplicar algoritmos de recorrido en el árbol, como el recorrido en orden, preorden y postorden, lo que nos ha brindado mayor flexibilidad y control sobre los datos almacenados. Estos conocimientos nos han proporcionado una herramienta poderosa para resolver una amplia gama de problemas y nos han permitido optimizar el rendimiento del proyecto RiTeVe. En general, la implementación de Árboles Binarios de Búsqueda ha ampliado nuestra comprensión de las estructuras de datos y nos ha brindado una base sólida para seguir aprendiendo y creciendo en el ámbito de la programación.

2) Hemos mejorado en la estética de este proyecto haciendo que la personalización de ventanas, botones, cuadros de texto, etiquetas y otros elementos de la interfaz de usuario sean más agradable a la vista. Hemos utilizado colores, fuentes y estilos para crear una apariencia coherente y atractiva en toda la aplicación.

3) Se ha logrado una mejora significativa en la programación de las funciones CRUD (Crear, Leer, Actualizar y Eliminar) gracias a la continua practica en tareas anteriores y en este proyecto RiTeVe. Durante el desarrollo de este proyecto, nos hemos enfocado en perfeccionar cada una de estas funciones para lograr un rendimiento óptimo y una gestión eficiente de los datos.

4) Se ha perfeccionado el uso de archivos .dat en la elaboración de este proyecto, lo que ha resultado en una gestión más eficiente y efectiva de los datos. A lo largo del desarrollo, hemos dedicado tiempo a optimizar y refinar la manipulación de estos archivos, lo cual ha mejorado la calidad y confiabilidad de nuestros procesos de lectura y escritura de datos.

Hemos implementado técnicas para garantizar la integridad de los archivos .dat, evitando errores de lectura o escritura que podrían comprometer la integridad de los datos almacenados. Esto incluye la implementación de validaciones y verificaciones para asegurar que los datos sean consistentes y estén en el formato esperado.

5) Se ha perfeccionado el manejo de documentos en este proyecto, ya que hemos logrado implementar la funcionalidad de transformar estos documentos en otros formatos y enviarlos por correo electrónico. Esta mejora nos ha brindado una mayor flexibilidad y capacidad para compartir información de manera eficiente.

Gracias a la implementación de algoritmos y herramientas adecuadas, hemos logrado convertir los documentos requeridos de DOCX a PDF, según las necesidades del usuario. Esto nos ha permitido adaptar los documentos a diferentes plataformas y requisitos de visualización.

Conclusión:
Este proyecto ha tenido un impacto altamente positivo en nuestra adquisición de conocimientos y comprensión de la programación. Nos sentimos muy satisfechos con los logros obtenidos, dado que dedicamos una cantidad considerable de tiempo y esfuerzo a su desarrollo. A lo largo de este proceso, hemos adquirido valiosas habilidades que serán fundamentales en proyectos venideros, lo que nos hace sentir aún más preparados para enfrentarlos exitosamente. En el panorama tecnológico actual, la programación es una habilidad crucial, y contar con una base sólida en sus fundamentos resulta sumamente valioso. Nuestra dedicación y perseverancia en este proyecto demuestran nuestra capacidad para enfrentar desafíos y alcanzar el éxito como programadores. Además de la escritura de código, hemos comprendido que la programación implica habilidades como el pensamiento crítico, la resolución de problemas y la creatividad. Al seguir desarrollando estas habilidades, estaremos mejor equipados para afrontar futuros desafíos en el campo de la tecnología y continuar nuestro aprendizaje y crecimiento en esta apasionante área. Como equipo, hemos demostrado una excelente colaboración y complementación, aprovechando nuestras fortalezas individuales para lograr nuestros objetivos comunes. Nos enorgullece el trabajo conjunto que hemos realizado y nos emociona anticipar las futuras oportunidades que nos esperan.

Estadística de tiempos

Actividades Realizadas	Horas
Análisis del problema	8
Diseño de algoritmos	20
Configuración	12
ABB Recursivo	19
Despliegue tablero	3
Ejecución comandos	13
PDF y envío	3
Programación	57
Pruebas	9
Elaboración del manual de usuario	3
Elaboración de documentación del proyecto	5
TOTAL	**152**

TEC | Tecnológico de Costa Rica

Lista de revisión del proyecto

Concepto	Puntos originales	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	
Funcionalidades adicionales				Opción consultar en CRUD de fallas más amigable con el usuario. Creación de los certificados muy realistas con base en los originales. Personalización del tablero más ordenado y amigable con el usuario. Ayuda al usuario mediante messageboxes que le indican qué está colocando mal y lo guían.

Referencias

Atico Grupo. (2020). Software Colaborativo. https://protecciondatos-lopd.com/empresas/software-colaborativo-groupware/#:~:text=El%20software%20colaborativo%20o%20groupware,correo%2C%20mensajería%20instantánea%20y%20más.

Github. (2023). Installing and configuring gthub desktop. https://docs.github.com/es/desktop/installing-and-configuring-github-desktop/overview/getting-started-with-github-desktop

Stack overflow. (2022). Importación de módulos en python. https://es.stackoverflow.com/questions/544677/importación-de-módulos-en-python

Mata Rodríguez W. (2023). Árboles. Material brindado en clase