Análisis Del Proyecto Sistema para Gestión de Proyectos

Proyecto Final Algoritmos Y Estructura De Datos

Nombre Aplicación: *Zoplakotopla Planning*

Integrantes:

- Juan Esteban Gómez Andrade
- Daniel David Sánchez Mendoza
- Deiby Rafael Ospina Triviño

Introducción

EL presente documento tiene como objetivo, el análisis y explicación del proyecto desarrollado. Siendo el mismo la creación de programa de estilo aplicación o página web con el objetivo de la creación, gestión y administración de una gran variedad de proyectos; haciendo uso principal y en su mayoría y/o totalidad el lenguaje de programación C++ junto a las diversas prácticas, algoritmos y estructuras de programación trabajadas en el programa y consultadas autónomamente por los miembros del equipo; esto con el fin de darle una solución óptima y oportuna del proyecto con la cual el usuario pueda satisfacer sus necesidades frente a la planeación de proyectos.

Análisis Solución

Problema / Objetivo:

Desarrollar una aplicación de consola basada en el lenguaje de C++ que permita a los usuarios gestionar proyectos, tomar notas, planificar y organizar sus tareas diarias, semanales o mensuales y asignarles un responsable. Este sistema les permitirá a los usuarios a mantener un registro de sus tareas y compromisos, así como a establecer prioridades y plazos y conocer en cuales proyectos participan cada uno.

Solución

La solución implementada es una aplicación de consola desarrollada en C++ destinada a la gestión de proyectos y tareas. Esta aplicación permite a los usuarios gestionar proyectos planificando y organizar sus tareas diarias, semanales o mensuales, y asignar responsables a cada tarea llevando así un control en el progreso del trabajo. El sistema facilita el mantenimiento de un registro de tareas y compromisos, así como la gestión de prioridades y plazos para mayor comodidad del usuario.

Detalles Técnicos:

- ***Lenguaje de Programación:*** C++
- ***Entorno de Desarrollo:*** Visual Studio Code
- ***Espacio de trabajo / alogamiento del proyecto:*** GitHub
- ***Metodología Empleada:*** Scrum
- ***Bibliotecas Utilizadas:***

  - **"#include <iostream>":** Se usa para poder tener acceso a los dispositivos estándar de entrada y salida de datos.
  - **"#include <string>":** contiene un conjunto de funciones para manipular cadenas de caracteres.
  - **"#include <list>":** Se usa para permitir el acceso y manipulación a la estructura de listas
  - **"#include <set>":** Se usa para permitir el acceso y manipulación a la estructura de conjuntos
  - **"#include <iomanip>":** Añade manipuladores que permiten controlar la forma en que los datos se presentan
  - **"#include <vector>":** Se usa para permitir el acceso y manipulación a la estructura de vectores
  - **"#include <filesystem>":** Ayuda a la manipulación de sistemas de archivos y sus componentes, como archivos y directorios.
  - **"#include <fstream>":** Permite la lectura y escritura en archivos de manera eficiente
  - **"#include <sstream>":** Proporciona clases para trabajar con flujos en memoria, similares a los flujos de entrada y salida estándar
  - **"#include <regex>":** permite realizar operaciones de búsqueda, coincidencia y manipulación de cadenas basadas en patrones.
  - **"#include <algorithm>":** Proporciona una amplia gama de algoritmos para realizar operaciones en contenedores de datos, como vectores, listas, mapas, entre otros.
  - **"#include <queue>":** Se usa para permitir el acceso y manipulación a la estructura de colas
  - **"#include <ctime>":** Permite la implementación de métodos y funciones para el control de variables de tipo tiempo
  - **"#include <chrono>":** Esta librería permite trabajar con variables de tipo tiempo
  - **"#pragma once":** es una directiva de preprocesador utilizada en algunos lenguajes de programación, para evitar la inclusión repetida de un archivo de encabezado en un programa.

Arquitectura

Se implementa un sistema diseñado en base al patrón MVC para el desarrollo del proyecto, el cual consta de los siguientes componentes principales:

- ***Model:*** El paquete Model y todas las clases albergadas en el tienen la función de gestionar los datos y la lógica de los diversos apartados de la aplicación; permitiendo suministrar al sistema métodos para manipular, extraer y recuperar los datos que están almacenados en la aplicación .Incluyendo las clases Usuario, Responsable, Proyecto, Tarea, Nota, Reacción y BD; las cuales poseen todos los parámetros solicitados para la realización de la aplicación y corroborando su existencia única en el sistema.
- ***Controller:*** El paquete Controller permite establecer una relación entre Model y View, ya que recibe las entradas de usuario provenientes de View realizando las operaciones pertinentes para poder actualizarla la información del Model según sea necesario. Incluyendo las clases UsuarioController, ResponsableController, ProyectoController, TareaController, NotaController y FIlesController.; las cuales aplican las diversas funciones destinadas por el equipo de trabajo para la creación, edición o eliminación de datos que son presentados al usuario.
- ***View:*** Representa la interfaz de usuario. Se encarga de mostrar los datos del modelo al usuario y enviar las interacciones o entradas al controlador. Incluyendo UsuarioView, ProyectoView, TareaView y NotaView.
- ***Utils:*** El paquete Utils representa para el sistema un apoyo frente al uso de varias funciones complejas o recursivas de tal forma que no sea requerido una creación constante de las mismas que genere sobre carga en el proyecto. Incluyendo las clases Utils, EstadosProyecto, EstadosTarea, PrioridadesTarea, Reacciones.
- ***Main:*** La clase main es usada para ejecutar todo el proyectó en su totalidad, siendo el principal motivo por el cual el usuario se puede conectar al sistema.

Ahora se expondrán algunos aspectos positivos y negativos del uso de este modelo a la hora de crear el proyecto:

Ventajas:

- ***Separación de responsabilidades:*** Cada componente tiene una función clara y separada, lo que facilita el mantenimiento y la escalabilidad de la aplicación.
- ***Reutilización de código:*** El modelo y las vistas pueden ser reutilizados en diferentes contextos para la realización de nuevas funcionalidades en la aplicación.
- ***Facilita el desarrollo en equipo:*** El equipo scrum de desarrolladores pueden trabajar de manera más eficiente en diferentes componentes (Model, View, Controller) simultáneamente.

Desventajas:

- ***Complejidad inicial:*** La división de la aplicación en tres componentes puede parecer compleja al principio, no obstante, es adaptiva con el tiempo.
- ***Demasiados Archivos:*** El patrón MVC puede resultar en la creación de muchos archivos diferentes, lo cual se presenta en el proyecto con la variedad de clases.

Esto puede hacer que la navegación y la gestión del proyecto sean más complicadas.

- ***Sincronización de Componentes:*** Mantener la sincronización entre el modelo, la vista y el controlador puede ser complicado, a la hora de trabajar en diferentes partes de la aplicación simultáneamente.

Estructuras utilizadas:

- ***Clases:*** Una clase define los atributos y métodos que los objetos creados a partir de ella tendrán. En esencia, una clase encapsula datos y comportamientos relacionados, proporcionando una estructura para organizar y gestionar el código.
- ***Constructor:*** Es un método especial dentro de una clase que se invoca automáticamente cuando se crea una instancia (objeto) de esa clase. Es usada en la mayoría de las clases de la solución.
- ***Métodos:*** Entendemos por métodos ,funciones que están asociadas a una clase específica. Los métodos representan el comportamiento de los objetos de esa clase y pueden acceder y manipular los datos de esos objetos. Siendo ampliamente visible en la solución planteada a ser parte esencial del paquete Model y sus clases.
- ***Listas:*** Es una estructura de datos que almacena una colección de elementos. Estos elementos pueden ser de cualquier tipo y se almacenan de manera contigua en memoria. Siendo visible su uso para el almacenamiento de notas con sus debidos parámetros.
- ***Conjuntos:*** Es una estructura de datos que almacena una colección de elementos únicos, es decir, no permite elementos duplicados. En el código es implementado para almacenar Responsables Con sus debidos aspectos.
- ***Vectores:*** Un vector es una estructura de datos que almacena una colección ordenada de elementos del mismo tipo, pero a diferencia de un array estándar, su tamaño puede cambiar dinámicamente. Por sus características se decidió con el equipo de trabajo implementarlo para el almacenamiento de Tareas , Reacciones , Usuarios y Proyectos.

Principios POO implementados:

- ***Sobre carga:*** La sobrecarga es un principio de programación el cual se encarga de permitirle al desarrollador definir una función o variable la cual puede ejecutarse de diferentes maneras según sea requerido o el objeto que le este llamando. En el código se puede evidenciar en paquete Controller dado que las diferentes clases que albergan poseen funciones para manipular listas, conjuntos, y vectores de datos, pero igualmente manipulan objetos o variables específicas.

- ***Encapsulamiento:*** Este principio permite el almacenamiento de grandes cantidades de datos en objetos, lo cual permite la agrupación de aspectos definidos y solicitados ya sea por el equipo de trabajo o usuario necesarios para el funcionamiento de funciones con el objetivo de operar los datos o interacciones con el usuario. En el código es fácilmente evidenciable este apartado dado que constantemente se usa para facilitar el movimiento de datos entre clases.
- ***Herencia:*** Se puede identificar el uso de herencia en el proyecto al ser evidente las diversas clases que posee el mismo en las cuales heredan los atributos y métodos de otras

Requerimientos Implementados:

- Sistema De login/Register de usuarios en la aplicación
- Creación de un sistema de creación, edición y visualización de proyectos los cuales poseen los siguientes aspectos nombre del proyecto,
- propietario, Estado del proyecto, descripción.
- Creación de un sistema de creación, edición y visualización de Tareas las cuales poseen los siguientes aspectos, nombre de la tarea, Estado,
- prioridad, Responsable, Fecha límite, Resumen (Comentarios).
- Creación de un sistema de creación, edición y visualización de Notas los cuales poseen los siguientes aspectos título de la nota, autor de la nota, descripción y reacciones.
- Implementación de menús interactivos con el usuario para el uso de las funciones y aplicación.
- Capacidad de importar tareas mediante un archivo CVS que proporcione el usuario
- Capacidad de exportar tareas almacenadas en el sistema de la aplicación en formato CVS
- Capacidad de consulta de proyectos por nombre y/o Fecha de creación para una mejor gestión y mayor facilidad de trabajo para el usuario
- Mecanismo De orden en las tareas por prioridad/estado los cuales son resaltados por colores distintivos, además de un sistema para el orden de responsables asignados por sistema alfabético. con el propósito de hacer más fácil la visualización de los aspectos
- Capacidad de consultar tareas por nombre específico o responsable a cargo para hacer un seguimiento detallado de las mismas.

Diagrama De Flujo:

Se presenta un diagrama de flujo en el cual se plasma el algoritmo de la solución implementada, paso a paso y con claridad en los conceptos. (El diagrama de flujo se encuentra adjunto en el espacio destinado para la entrega del proyecto o en su defecto en el repositorio GitHub, esto dado a que no se adaptaba al formato Word)

Metodología Ágil:

Para el desarrollo del proyecto se manejó una metodología tipo scrum la cual se base en el uso de ciclos interactivos (Product Backlog, Sprint Backlog, Increment), los cuales tienen diferentes fases donde se identifican las tareas a realizar para la finalización del proyecto, posteriormente se reparten entre el equipo de trabajo y sin trabajadas en periodos de tiempo definidos para luego hacer una revisión y pasar a repetir el ciclo con nuevas tareas o algunas que no se hayan podido completar hasta acabar el proyecto. De esta forma se repartieron los roles para llevar el control de la metodología, siendo de esta manera:

- **Product Owner:** Tatiana Cabrera
- **Scrum Master**: Deiby Ospina
- **Equipo Scrum:** Juan Gómez, Daniel Sánchez, Deiby Ospina

Llevando un seguimiento del trabajo en un espacio de gestión de proyectos designado por el equipo el cual fue Trello

Link:https://trello.com/invite/b/QEjqVDsE/ATTI1c84a52c9a9ca15a4569d791b239fec256ED28D2/proyecto-final2-sprint

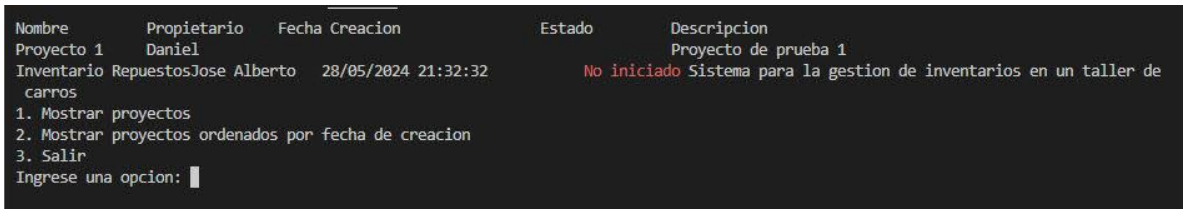
Además de llevar un control de reuniones diarias conocidas como dailys, para la buena gestión y avance del proyecto; Los mismos se encuentran registrados en un archivo Excel el cual se encuentra adjunto en el espacio destinado a la entrega del proyecto o en su defecto en el repositorio de trabajo GitHub.

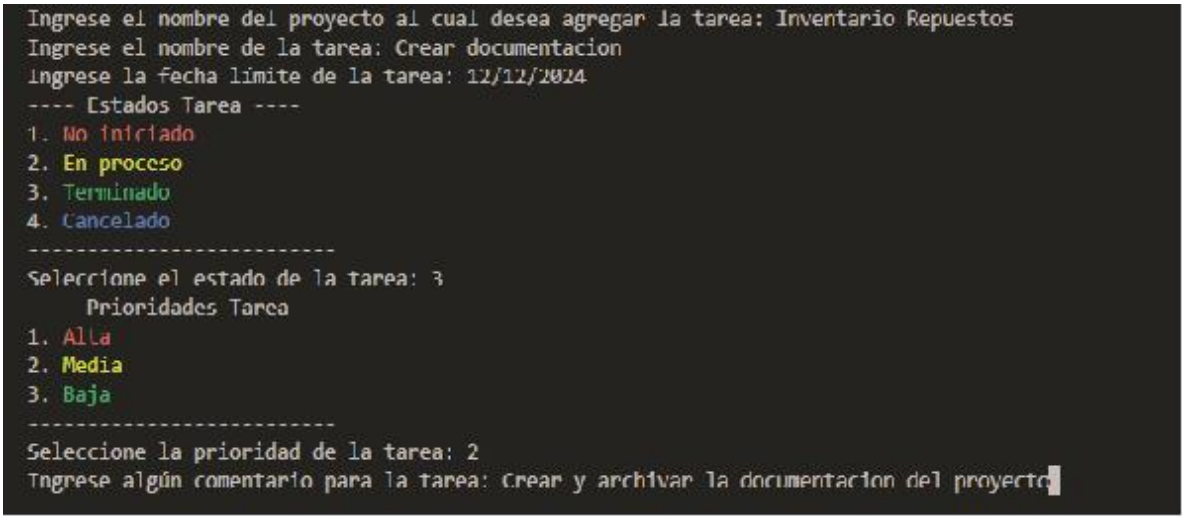
Link Repositorio: https://github.com/DeibyOspina/Proyecto-Final

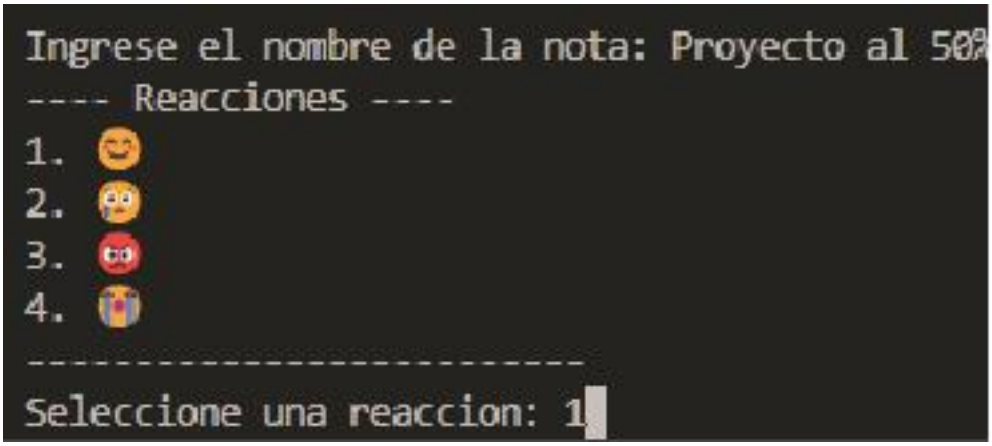
Análisis de resultados:

Se llevaron a cabo varias pruebas técnicas a lo largo del desarrollo del proyecto esto con la intención de entregar la aplicación en un estado funcional, con todos sus apartados en buen funcionamiento. Además, que el hacer las pruebas permitió identificar errores o necesidades no previstas en el código que llevaron a mejorar el sistema.

Ejemplos de las pruebas:







Conclusiones:

Para concluir se puede evidenciar que en su mayoría fue posible cumplir con los requerimientos y metas planteadas para el proyecto. Haciendo uso de las diferentes principios, bases y algoritmos de programación aprendidos por los integrantes

Lecciones aprendidas:

- De la realización del proyecto fue posible el aprendizaje de la implementación de las buenas prácticas de programación, métodos, algoritmos y principios que presenta es campo del conocimiento. Además de reforzar aspectos como el trabajo cooperativo y responsabilidad al ser aplicada la metodología ágil Scrum la cual fue esencial para el oportuno y bien desarrollo del proyecto

Retos y obstáculos

- A lo largo del proyecto se presentaron diversos retos y obstáculos a superar, varios a la hora de la realización de las actividades y algunos otros frente al trabajo con el equipo. Llevando los mismos de aprendizaje para futuros proyectos.

Este sería el análisis del proyecto sistema de gestión de proyectos, nombrado por el equipo como Zoplakotopla Planning con el fin de satisfacer las necesidades del usuario. Reconociendo el trabajo de todos los miembros para terminar el mismo.

Referencias:

Venusmi. (s. f.). Preguntas más frecuentes sobre la biblioteca de C++. Microsoft Learn. https://learn.microsoft.com/es-es/troubleshoot/developer/visualstudio/cpp/libraries/faq-standard-cpp-library

GeeksforGeeks. (2022, 8 noviembre). Implementation of Singleton Class in C. GeeksforGeeks. https://www.geeksforgeeks.org/implementation-of-singleton-class-in-cpp/

Qué es la Programación Orientada a Objetos. (s. f.). Intelequia. https://intelequia.com/es/blog/post/qu%C3%A9-es-la-programaci%C3%B3n-orientada-a-objetos