**Desafío II**

Nicolás Valencia

Karen Mazo

Universidad de Antioquia

2598521: Informática II

Aníbal Guerra

1. de octubre de 2024

**Análisis del problema y consideraciones para la alternativa de solución propuesta.**

Se está desarrollando un programa para gestionar una red nacional de gasolineras. Este sistema debe manejar varias estaciones, cada una con su infraestructura específica, que incluye tanques para almacenar combustible, surtidores para despachar el combustible y ventas que son registradas por cada surtidor. El objetivo es crear un diseño adecuado para modelar estos elementos y las relaciones entre ellos.

La solución del problema se llevará a cabo por medio de clases, dentro de las cuales se ha decido definir; red nacional que representa el sistema que agrupa todas las estaciones de servicio siendo está el nivel más alto en la jerarquía de clases, su función es gestionar las operaciones de todas las estaciones; también se definió la clase estación de servicio que está conectada a un tanque que almacena el combustible, su función es gestionar las ventas, controlar las operaciones de los surtidores y gestionar el combustible; además, se definió la clase tanque, ya que cada estación tiene un tanque central que contiene los tipos de combustible y su capacidad puede variar entre estaciones, su función es llevar un inventario del combustible que abastece a los surtidores; también se definió la clase surtidor que está conectado al tanque central y registra las ventas diarias, su función es realizar el despacho del combustible y registrar las ventas; por último se definió la clase ventas que se asocian a la cantidad de combustible vendido, permite la gestión de cada venta para llevar control de lo vendido en cada surtidor; esto permitirá estructurar el sistema de forma modular con relaciones entre las clases y sus respectivos atributos.

Además, se usará arreglos dinámicos para almacenar la información de las estaciones, surtidores y ventas buscando la optimización de la memoria, se optó por esta alternativa ya que no se debe usar STL, y la información de las estaciones, surtidores y ventas es cambiante, es decir, puede aumentar o disminuir.

**Diagrama de clases de la solución planteada. Y descripción de la lógica de las tareas.**

Las clases identificadas para la solución del problema:

* RedNacional: Administra el conjunto de estaciones, por lo cual la clase estación se relaciona con esta de la forma “*es parte de*” con una cardinalidad 1 a n, es decir, la red nacional puede contener muchas estaciones.
* Estación: contiene al tanque y a los surtidores, por lo que la clase tanque y surtidores se relacionan con la clase estación de la forma “*es parte de*” con una cardinalidad con respecto al tanque de 1 a 1, es decir, una estación tiene un tanque central y con respecto a surtidores tiene una cardinalidad de 1 a n, es decir, una estación puede tener varios surtidores.
* Tanque: Maneja la cantidad de combustible disponible de cada categoría
* Surtidor: Realiza ventas y registra cada una por lo que la clase ventas se relaciona con la clase surtidor de la forma “*es parte de*” ya que las ventas solo pueden estar asociadas a un surtidor especifico, con una cardinalidad de 1 a n, ya que un surtidor puede registrar muchas ventas.
* Ventas: Almacena la información de una venta individual.

En este apartado, se ofrece una visión general de la lógica y estructura de las tareas más importantes dentro del programa, facilitando la comprensión de cómo se deben resolver las funcionalidades del programa, se explica de forma general que hace cada sección.

Dentro de la clase de red nacional se implementan las siguientes tareas:

* Agregar estación: inserta una nueva estación en la red, con datos como el nombre, la región, las coordenadas. Se debe verificar que no haya conflictos con códigos repetidos.
* Eliminar estación: permite eliminar una estación de la red solo si no tiene surtidores activos. verifica que todos los surtidores estén inactivos antes de proceder con la eliminación.
* Calcular el monto total de ventas: recorre todas las estaciones y suma las ventas totales separadas por tipo de combustible (Regular, Premium, EcoExtra).
* Fijar los precios del combustible: actualiza los precios de cada tipo de combustible en todas las estaciones, teniendo en cuenta la región a la que pertenece cada estación.

Dentro de la clase de estaciones se implementan las siguientes tareas:

* Agregar surtidor: inserta un nuevo surtidor, verifica que no esté repetido el código, agrega datos como el código y el modelo. El código lo construye dependiendo de la estación y de la nave a la que pertenece el surtidor.
* Eliminar surtidor: permite eliminar un surtidor y las ventas asociadas a ese surtidor.
* Activar surtidor: permite marcar un surtidor como activo si esta desactivado si ya está activo no cambia su estado, se hace con el fin de manejar su estado operativo.
* Desactivar surtidor: permite marcar un surtidor como inactivo si está activo si ya está inactivo no cambia su estado, se hace con el fin de manejar su estado operativo.
* Consultar histórico: permite acceder al historial de ventas de un surtidor especifico.
* Reportar litros vendidos: recorre las ventas de la estación y separa los litros vendidos por tipo de combustible.
* Simular venta: asigna aleatoriamente un surtidor que debe estar activo, selecciona una cantidad aleatoria de 3-20 litros, muestra los datos de la venta.
* Asignar capacidad del tanque: asigna una cantidad aleatoria de 100-200 litros a cada tipo de combustible del tanque manteniendo las capacidades separadas.
* Verificación de fugas: compara lo vendido y lo almacenado con respecto a la capacidad original del tanque.

**algoritmos implementados.**

En este apartado se dan los detalles técnicos de forma general de los algoritmos más importantes que se implementaron en el desarrollo de la solución del problema, además, en el código se incluyen comentarios para aclarar el propósito de cada bloque y dar una visión más completa de los algoritmos al complementarse con las siguientes explicaciones.

Los algoritmos que se implementan en la solución del problema son los siguientes:

* Agregar estación: primero se verifica el código para evitar que haya códigos duplicados, luego se ajusta el tamaño del arreglo de estaciones si es necesario para agregar la estación, asigna cada dato de la estación en la posición correcta del arreglo.
* Eliminar una estación: recorre los surtidores y verifica que todos estén inactivos ya que solo se puede eliminar si esto ocurre, reajusta el tamaño del arreglo de estaciones si es necesario.
* Fijar los precios del combustible: Dependiendo de la región donde está la estación se define un precio a cada tipo de combustible.
* Agregar surtidor: se pide el código de la estación y de la nave a la cual desea que pertenezca el surtidor, verifica que el código del surtidor no exista, lo agrega ajustando el tamaño del arreglo de surtidores si es necesario.
* Eliminar surtidor: elimina el surtidor reajustando el tamaño del arreglo, además, elimina las ventas que están asociadas a él.
* Consultar el histórico: se debe seleccionar la estación de la cual desea ver el historial de las ventas, el cual es un arreglo donde se almacena cada venta con sus respectivos detalles, se recorre dicho arreglo y se muestran los datos de manera ordenada por cada surtidor de la estación.
* Reportar la cantidad de litros vendida: recorre el arreglo de ventas, y para cada surtidor de la estación suma las ventas discriminadas por tipo de combustible, almacena el resultado de las ventas por tipo en variables.
* Asignar la capacidad del tanque: se usa una función para generar un valor aleatorio entre 100 y 200 litros para cada tipo de combustible y estos valores se almacenan en la capacidad de cada categoría del tanque.
* Simulación de ventas: se usa una función para seleccionar un surtidor activo de forma aleatoria, además, de forma aleatoria también se determina la cantidad de litros vendidos, reduce la cantidad de combustible de la categoría correspondiente a la venta, actualiza el registro de ventas del surtidor
* verificación de fugas: se suma el combustible vendido por cada surtidor de la estación y se compara con lo que queda en el tanque.

**Problemas de desarrollo.**

En este apartado se aborda algunos de los problemas que se afrontaron en el desarrollo del programa, los cuales tienen que ver con el manejo de la información y de la relación entre las clases, ya que se debe identificar, cual es la forma correcta de enlazar la información de cada una. Además, otro desafío dentro del programa se relaciona con la gestión del crecimiento y la escalabilidad del sistema, dado que el número de estaciones, surtidores y ventas puede variar en el tiempo. Teniendo en cuenta que se necesita un enfoque que permita almacenar y recuperar datos relacionados (ventas, por ejemplo) sin consumir demasiada memoria ni complicar la estructura. En conclusion en el desarrollo de la solución al problema se presentaron una serie de desafíos técnicos relacionados con la memoria dinámica, el manejo de punteros, la gestión de relaciones entre clases.

**e. Evolución de la solución y consideraciones para tener en cuenta en la implementación.**

La evolución del desarrollo requirió de un análisis que fue progresando en la comprensión del problema hasta la implementación, donde se tomaron decisiones estratégicas, además, de una constante revisión a lo propuesto para poder garantizar una solución mantenible.

Como primera etapa se hizo el análisis de lo que se estaba pidiendo y lo necesario para llevarlo a cabo, ya que el problema involucra a varias entidades, se optó por un enfoque basado en el diseño modular, con el fin de dividir el problema y hacer más sencilla la solución, también para manejar la información de forma más sencilla se hizo uso de arreglos dinámicos, en esta primera etapa se definieron las clases necesarias para representar las entidades que están involucradas, luego en una segunda etapa se diseñó la relación y la cardinalidad que tendrían cada una de las clases entre sí, también se desarrollaron métodos y atributos de cada clase, luego en una tercera etapa se hizo la implementación de los algoritmos, por último se hizo una serie de pruebas para asegurar que el programa funcionara de la forma esperada, en caso de que no lo hiciera se hicieron los ajustes donde fuera necesario. La solución evolucionó a partir del análisis hecho hacia la implementación que cumple con los requisitos planteados.

Dentro de las consideraciones para la implementación se tuvo en cuenta que a medida que la red nacional crezca, la solución debe ser capaz de manejar una mayor cantidad de estaciones, surtidores y ventas sin perder rendimiento. Esto implica optimizar el uso de arreglos dinámicos y la gestión de memoria; además, a medida que se dificulte el sistema, se debe prever posibles errores o excepciones que puedan surgir, como fallos en los registros o inconsistencias en los datos. Para superar estos retos es crucial adoptar buenas prácticas de programación, como la buena gestión de la memoria, el uso de estructuras de datos eficientes y la división de tareas complejas en subproblemas más fáciles de manejar, tales como el diseño de un sistema modular.

En conclusión, el desarrollo de la solución al problema requirió de varias fases como análisis, diseño, implementación y pruebas, además, del diseño modular, la gestión de la memoria y la optimización de los algoritmos a medida que se fueron presentando problemas buscando la correcta ejecución del programa.