# **Análisis**

La empresa TerMax solicita crear un sistema de comercialización de combustible que nos permita llevar la gestión eficiente de las estaciones de servicio, los surtidores y las transacciones de venta de combustible. Del documento de especificación se obtiene lo siguiente:

**Estaciones de servicio:** la cual tiene un nombre, un código identificador, un gerente, una región y una ubicación geográfica es el objeto principal dentro del modelado de la solución a plantear, cumple la función de contener un tanque que a su vez va a almacenar el combustible. Este combustible puede ser de tres tipos, que son: Regular, Premium y EcoExtra y que pueden ser distribuidos por un numero de máquinas surtidoras de entre 2 y 12 según el documento de especificación.

**Máquina surtidora:** la cual tienen un código identificador y un modelo de máquina, cumple la función de suministrar individualmente el combustible al cliente, es decir realizar las ventas del día. La venta debe contener la siguiente información: la fecha, la hora, la cantidad de combustible, la categoría de combustible, documento del cliente, cantidad de dinero de la compra y método de pago (Efectivo, TDebito, TCrédito). Además, cada vez que se vende combustible se debe actualizar la disponibilidad del mismo en el tanque de almacenamiento y en caso de tener menos combustible del requerido por el cliente se vende la cantidad disponible y se cobra dicha cantidad.

Como requerimientos adicionales se solicitan crear un menú de red nacional y otro de gestión para cada estación de servicio.

Las funcionalidades para el **menú de red nacional:**

* Agregar estaciones de servicio.
* Eliminar una E/S de la red nacional (sólo si no posee surtidores activos). Importante tener presente si los surtidores están activos.
* Calcular el monto total de las ventas en cada E/S del país, discriminado por categoría de combustible.
* Fijar los precios del combustible para toda la red.

**Menú de gestión de estación de servicio:**

* Agregar/eliminar un surtidor a una E/S.
* Activar/desactivar un surtidor de una E/S.
* Consultar el histórico de transacciones de cada surtidor de la E/S.
* Reportar la cantidad de litros vendida según cada categoría de combustible.
* Simular una venta de combustible.
* Asignar la capacidad del tanque de suministro, con un valor aleatorio entre 100 y 200 litros para cada una de las categorías.

## **Consideraciones adicionales para tener en cuenta:**

1. **Registro e historial de una estación de servicio:**

consideramos usar archivos (.txt) para manejar y registrar información local y global de las estaciones de servicios, como lo es las ventas, el historial independiente de las operaciones de cada surtidor y la disponibilidad de un tanque en su categoría.

1. **Precios de combustible por región:**

La diferencia de precios por región tendrá las siguientes propiedades; el precio fijado para las estaciones de servicio en el área centro será considerado el pilar para determinar los precios del área norte y sur. Por lo cual planteamos que el precio para el área norte será 10% mayor a la del área centro y para el área sur será 10% menor a la del área centro. (solo necesitaremos fijar el precio del área centro)

1. **Disponibilidad de combustible categoría y combustible menor al solicitado:**

En base a una venta se hará descuento a la cantidad disponible en el tanque a su categoría al disponer de menor cantidad a la requerida por un cliente se venderá y cobrará esa cantidad. Para el próximo cliente se hará venta normal con la disponibilidad máxima de combustible de la categoría de ese tanque (dado que no se menciona la cantidad de llenado y el momento de llenado)

1. **Verificación de fugas:**

Dado que se nos especifica que será considerado fuga siempre y cuando la venta y la capacidad del tanque corresponda a menos de 95%. Haremos esta comparación cada vez que el tanque es vaciado. entonces debemos comparar esta relación de capacidad y venta con una franja de diferencia del 5% respecto al precio fijado por su región.

# **Diagrama de clases:**

A continuación, se muestra la imagen correspondiente al diagrama de clases en su etapa previa a iniciar el desarrollo de la solución, para mayor claridad también se adjunta como imagen al repositorio:

Diagrama

Descripción generada automáticamente

Imagen 1. Diagrama de clases.

Para dar mayor claridad sobre el diseño de la solución plasmado en el diagrama de clases se presenta una explicación detalla de cada una de las clases a continuación:

**1. Clase TermaxGestionEstacion**

* **Descripción general**: Clase que gestiona las estaciones de servicio a nivel nacional.
* **Atributos**:
  + estaciones: EstacionServicio\* - Lista de estaciones que gestiona a nivel nacional y es de tipo EstacionServicio, clase que modela una estación. Adicional, se usará memoria dinámica, pues se tiene en cuenta que se podrían crear varias y se permite el control de la vida útil de la estructura de datos dentro del programa.
* **Métodos**:
  + eliminarEstacion(idEstacion: string): string - Elimina una estación del sistema usando su ID que es un sting único para cada estación. Devuelve un mensaje que indica si se realizo la eliminación o no.
  + agregarEstacion(EstacionServicio estacion): string - Agrega una estación al sistema. Recibe como parámetro una instancia de la clase EstacionServicio y la agrega a la estructura de datos que contiene las estaciones y devuelve el respectivo mensaje de confirmación.

**2. Clase EstacionServicio**

* **Descripción general**: Clase que representa una estación de servicio.
* **Atributos**:
  + idEstacion: string - ID único de la estación. Tipo string, ya que no se va a usar para hacer cálculos.
  + nombre: string - Nombre de la estación.
  + gerente: string - Nombre del gerente asignado a la estación.
  + region: Region - Región a la que pertenece la estación. Es de tipo Region, que es la clase que modela una región para el sistema.
  + latitud: string - Latitud geográfica de la estación.
  + longitud: string - Longitud geográfica de la estación.
  + tanque: Tanque - Tanque de almacenamiento de combustibles de la estación. De tipo Tanque, clase que modela el tanque de almacenamiento de combustible de la estación.
  + islas: Isla\* - Conjunto de islas de servicio en la estación. Corresponde a las subdivisiones físicas de la estación de servicio.
* **Métodos**:
  + Métodos como getIdEstacion, getNombre, getRegion, getGerente, etc., permiten obtener los valores correspondientes a sus atributos descritos en el nombre de los métodos.
  + Métodos como setIdEstacion, setNombre, setRegion, setGerente, etc., permiten modificar los valores correspondientes a sus atributos descritos en el nombre de los métodos.
  + setTanque(Tanque tanque) - Asigna un tanque de almacenamiento a la estación. Recibe una instancia de la clase Tanque que modela el tanque de almacenamiento.

**3. Clase Region**

* **Descripción general**: Clase que representa una región geográfica dónde va a estar ubicada la estación y adicional es importante porque dependiendo está se va a asignar el precio del combustible.
* **Atributos**:
  + nombre: string - Nombre de la región.
  + tiposCombustibles: Combustible[3] - Tipos de combustibles disponibles en la región. De los requerimientos se obtiene que se vana a manejar tres tipos de combustible, por lo que es un arreglo de tres posiciones cada una para una categoría de combustible.
* **Métodos**:
  + Métodos como getNombre, getTiposCombustible, permiten obtener los valores correspondientes a los nombres de cada atributo teniendo en cuenta el nombre del método.
  + setNombre(string nombre) - Asigna el nombre a la región.
  + definirCombustible(short índice, Combustible combustible) - Define un combustible en la lista de tipos de combustibles con sus características que son el tipo y el precio.

**4. Clase Isla**

* **Descripción general**: Clase que representa una isla de servicio dentro de una estación.
* **Atributos**:
  + idIsla: string - ID de la isla.
  + surtidores: Surtidor\* - Conjunto de surtidores disponibles en la isla.
* **Métodos**:
  + Métodos como getIdIsla, setIdIsla, agregarSurtidor(Surtidor surtidor) permiten gestionar los surtidores de la isla. El último mencionado permite agregar un surtidor al arreglo por lo que recibe una instancia de un surtidor.

**5. Clase Surtidor**

* **Descripción general**: Clase que representa un surtidor de combustible en una isla.
* **Atributos**:
  + codigo: string - Código único del surtidor.
  + modeloMaquina: string - Modelo del surtidor.
  + estado: bool - Estado del surtidor (operativo o no). Y será indispensable para la eliminación de un surtidor.
  + ventas: Venta\* - Conjunto de ventas realizadas en el surtidor.
* **Métodos**:
  + Métodos como getCodigo, getModeloMaquina, setEstadoSurtidor(bool estado), permiten gestionar los atributos del surtidor.
  + consultarTransacionSurtidor, este método devuelve todas las ventas registradas en el surtidor.
  + calcularVentasCategoria, devuelve el total de dinero de las ventas registradas en el surtidor discriminadas por categoría.
  + calcularVentasLitrosCategoria, devuelve el total de litros vendidos por cada categoría.
  + registrarVenta(Venta venta), recibe una instancia de venta para hacer su registro del combustible suministrado al cliente.

**6. Clase Tanque**

* **Descripción general**: Clase que representa un tanque de almacenamiento de combustible en una estación.
* **Atributos**:
  + capacidadRegular: float[2] - Capacidad para almacenar combustible Regular. Es un arreglo de dos posiciones porque la primera representa la cantidad de combustible Regular que contine y la segunda posición la cantidad de combustible vendido.
  + capacidadPremium: float[2] - Capacidad para almacenar combustible Premium. Es un arreglo de dos posiciones porque la primera representa la cantidad de combustible Premium que contine y la segunda posición la cantidad de combustible vendido.
  + capacidadExtra: float[2] - Capacidad para almacenar combustible Extra. Es un arreglo de dos posiciones porque la primera representa la cantidad de combustible Extra que contine y la segunda posición la cantidad de combustible vendido.
* **Métodos**:
  + getCapacidadRegular(), getCapacidadPremium(), getCapacidadExtra() permiten obtener las capacidades de los diferentes tipos de combustibles.

**7. Clase Combustible**

* **Descripción general**: Clase que representa un tipo de combustible.
* **Atributos**:
  + tipoCombustible: string - Tipo de combustible (Regular, Premium, etc.).
  + precioCombustible: float - Precio del combustible. De tipo float, ya que un precio puede contener decimales, máximo 2 decimales.
* **Métodos**:
  + getTipoCombustible(), getPrecioCombustible() permiten obtener los valores de los atributos.
  + setPrecioCombustible(float precio) permite establecer el precio del combustible.

**8. Clase Venta**

* **Descripción general**: Clase que representa una venta realizada en un surtidor.
* **Atributos**:
  + fechaHora: tm\* - Fecha y hora de la venta.
  + tipoCombustible: Combustible - Tipo de combustible vendido.
  + cantidadCombustible: float - Cantidad de combustible vendido. Float porque puede contener decimales la cantidad, máximo 2 decimales.
  + metodoPago: short - Método de pago utilizado. Es un short que corresponde a la posición de un arreglo constante que contiene los tres tipos de pago (Efectivo, TDebito, TCrédito), esto facilita el manejo.
  + idCliente: string - ID del cliente.
  + dinero: float - Cantidad de dinero de la venta. Se obtiene a partir de multiplicar el precio del combustible por la cantidad de combustible vendido.
* **Métodos**:
  + Métodos como getFechaHora(), getTipoCombustible(), getMetodoPago(), permiten obtener los detalles de la venta.
  + setMetodoPago(string metodo) permite establecer el método de pago.
  + simularTransacion(), método encargado de simular una venta y que se registrará posteriormente en el surtidor.

**Relaciones:**

1. **Relación entre TermaxGestionEstacion y EstacionServicio:**
   * Relación de uno a muchos (1..\*). La clase TermaxGestionEstacion puede gestionar múltiples instancias de EstacionServicio. Es una relación de composición, pues **TermaxGestionEstacion** se compone de estaciones y si desaparece **TermaxGestionEstacion** las estaciones asociadas también lo deben hacer.
2. **Relación entre EstacionServicio y Isla:**
   * Relación de uno a muchos (1..\*). Cada EstacionServicio puede tener múltiples islas asociadas. Es una relación de composición, pues **EstacionServicio** se compone de islas y si desaparece **EstacionServicio** las islas asociadas también lo deben hacer.
3. **Relación entre EstacionServicio y Region:**
   * Relación de muchos a uno (\*..1). Cada EstacionServicio pertenece a una Region. Es una relación de asociación un tipo de relación más débil, si la Estación desparece la región no necesariamente debe desaparecer.
4. **Relación entre EstacionServicio y Tanque:**
   * Relación de uno a uno (1..1). Cada estación tiene un tanque de almacenamiento. Es una relación de composición un tipo de relación fuerte, ya que si la estación desaparece el tanque también lo debe hacer.
5. **Relación entre Isla y Surtidor:**
   * Relación de uno a muchos (1..\*). Cada isla puede tener múltiples surtidores. Es una relación de agregación, un tipo de relación más débil. Donde una Isla puede contener surtidores, pero los surtidores no dependen de la isla para existir, se podrían mover a otra Isla.
6. **Relación entre Tanque y Surtidor:**
   * Relación de uno a uno (2..12). Cada Tanque tiene dos o máximo 12 surtidores conectados. Es una relación de asociación.
7. **Relación entre Surtidor y Venta:**
   * Relación de uno a muchos (1..\*). Cada surtidor puede registrar múltiples ventas. Es una relación de asociación.
8. **Relación entre Región y Combustible:**
   * Relación de uno a muchos (1..\*). Cada región tiene varios tipos de combustibles. Es una relación de asociación.
9. **Relación entre Venta y Combustible:**
   * Relación de muchos a uno (\*..1). Cada venta se asocia a un tipo de combustible específico. Es una relación de asociación