Mejorando el Encapsulamiento y Pruebas Unitarias en el Sistema de Gestión de Donantes de Sangre de la Cruz Roja



Valentina Montezuma González (2242058)

- (-)

- (-)

- (-)

Universidad del Valle

Mejorando el Encapsulamiento y Pruebas Unitarias en el Sistema de Gestión de Donantes de Sangre de la Cruz Roja

Valentina Montezuma González (2242058)

- (-)

- (-)

- (-)

Profesor

Víctor Andrés Bucheli

Taller Programación Orientada a Objetos

Universidad del Valle

Facultad de Ingeniería

Programación Orientada a Objetos

Santiago de Cali

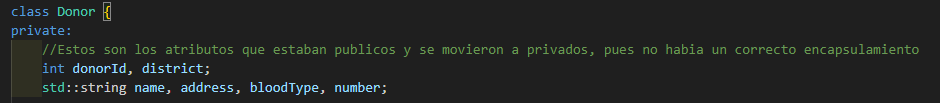
2025

Taller Encapsulamiento y Pruebas Unitarias

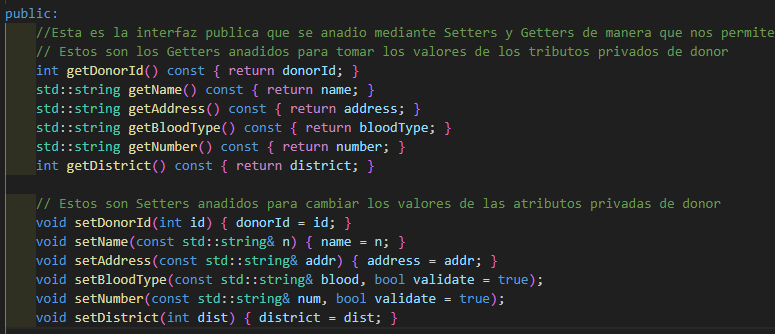
Análisis del diseño de clases, código modificado y problemas resueltos.

Como problema inicial que identificamos fue que los atributos dirección, donorId, name, address, distric, bloodType y number de la clase Donor eran públicos aun tratándose de datos de información delicada, lo que no cumplía con el principio de encapsulamiento, por lo que realizamos los siguientes cambios:

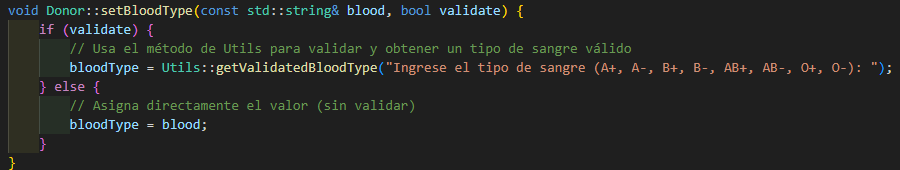
1. Pusimos los atributos privados.

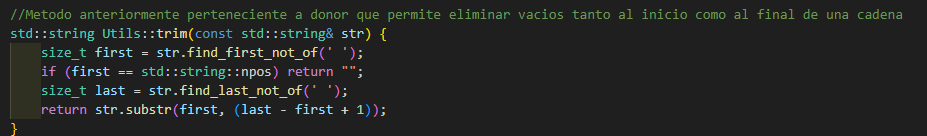


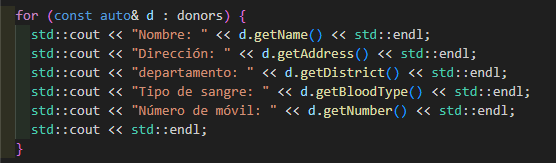
1. Se agregaron setters y getters para acceder a los atributos y modificarlos de una manera controlada y sin acceder directamente a ellos.

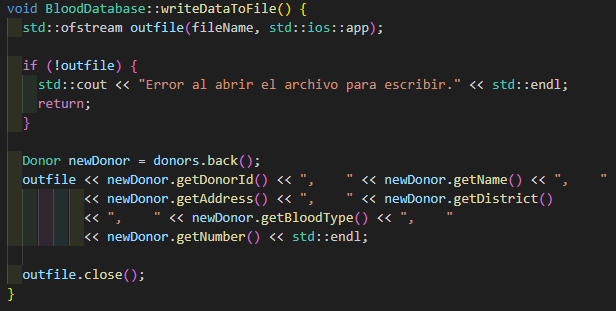


1. Implementamos validaciones para asegurar que los datos sean correctos (El número de teléfono mediante la API y el tipo de sangre usando los tipos existentes). Esto ya que anteriormente había errores al tomar el celular y en el caso de la sangre se podía poner cualquier letra haciendo que el programa no fuera preciso.



1. Movimos el método trim de la clase Donor ya que podía usarse más allá de la misma, por lo que se fue a una clase utilitaria nueva llamada Utils, de la que se hablara a profundidad más adelante. 

Por parte de la clase BlooDatabase, los cambios realizados fueron en función de que anteriormente accedía a los datos privados de Donor, así que se asigno que en su lugar accediera a los getters para evitar errores y respetar el encapsulamiento, a continuación, algunos ejemplos (Ya que es a lo largo de todo el código). 



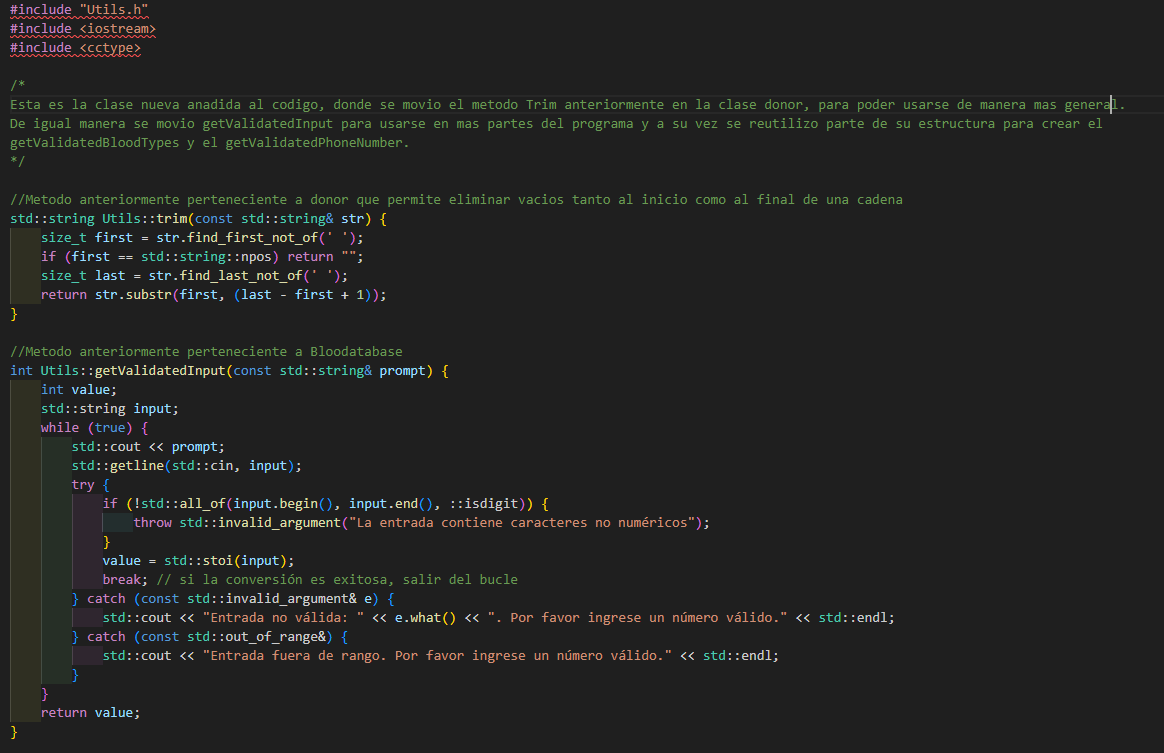
También se agregaron los métodos de:

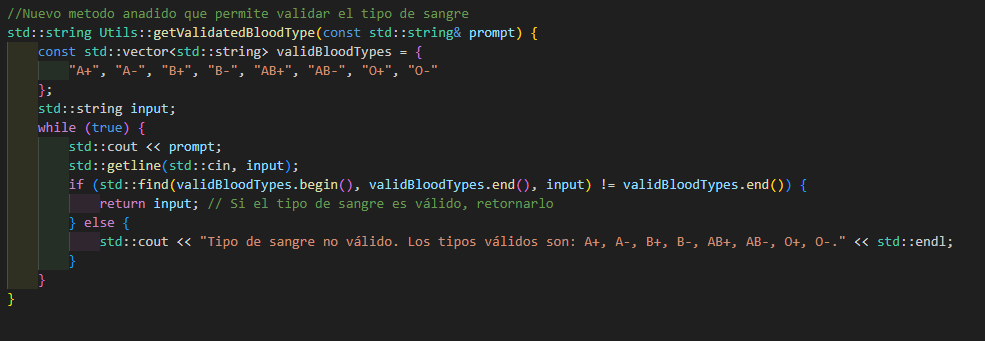
1. Porcentaje de cada tipo de sangre en la base de datos
2. Tipo de sangre más común en la base de datos
3. Porcentaje de la población de cada tipo de sangre
4. Que tipo de sangre tiene la mayoría de la gente

Descripción de la reutilización de código.

El método trim, como se mencionó anteriormente, se movió a una clase totalmente nueva, de manera que se facilita su reutilización en cualquier parte del proyecto y deja de estar limitado para una sola clase.

De igual manera de movió la clase getValidatedInput de BloodDatabase para darle un uso mas general y a su vez se utilizo parte de su código para crear la validación de los tipos de sangre, pues anteriormente no se verificaba si el dato ingresado por el usuario era un tipo de sangre valido, a continuación, se muestra como quedo estructurado en la clase Utils:

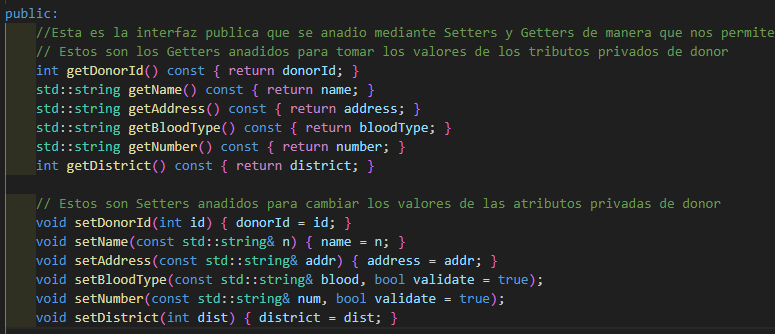




Descripción de la API integrada.

Implementación de interfaces publicas

Los getters y setters se crearon con el propósito de actuar como interfaz publica, pues son los que se usan a lo largo de BloodDatabase para acceder a los a tributos privados de Donor. Es decir, permiten su reutilización y a la vez mantener el encapsulamiento de manera correcta.



Código de las pruebas unitarias y resultados de las mismas