**Método de la ingeniería.**

**Contexto del problema.**

Un gran banco requiere de un software que sea capaz de manejar el funcionamiento de una de sus sedes con mayor flujo de personas.

**Fase no.1: Identificación del problema.**

**Identificación de necesidades y síntomas.**

* La sede de este banco requiere de un software que le permite manejar de manera eficiente el gran flujo de clientes.
* No existe un software en esta sede capaz de manejar la gran cantidad de clientes.
* Se requiere de algoritmos de búsqueda eficientes para el software.
* Se requiere de un manejo adecuado de los turnos para los clientes.
* Se requiere de un manejo eficiente de los clientes registrados y de las acciones que estos decidan realizar.
* Se requiere que la información de un usuario pueda ser visualizada.
* Se requiere de un botón en el software para deshacer cambios realizados.
* Se requiere que el software posea una interfaz gráfica de usuario.

**Definición del problema.**

La sede de este banco requiere de un software para el manejo de grandes cantidades de clientes.

**Fase no.2: Recopilación de información.**

Con el fin de dar una solución optima al problema, se ha recopilado información necesaria que permita el entendimiento de este. Para ello, se ha recopilado los requerimientos funcionales y no funcionales que podrían dar solución al problema.

**Requerimientos funcionales.**

Requerimiento no.1: Registrar un usuario en el programa. Permite registrar un usuario a través de su nombre y su número de identidad.

Requerimiento no.2: Modificar datos de un usuario registrado. Permite modificar el nombre, número de identidad, cuenta bancaria, tarjetas de débito o crédito, fechas de pago de las tarjetas y la fecha en la que se incorporó al banco el usuario.

Requerimiento no.3: Mostrar datos de un usuario registrado. Permite visualizar los datos de un usuario, los datos que deben mostrarse son el nombre, número de identidad, cuenta bancaria, tarjetas de débito o crédito, fechas de pago de las tarjetas y la fecha de incorporación al banco.

Requerimiento no.4: Asignar turno a un usuario. Permite asignar un turno al usuario una vez que este ha sido registrado en el programa.

Requerimiento no.5: Atender turno de usuario. Permite atender un turno a un usuario para que realice operaciones en el programa.

Requerimiento no.6: Buscar un usuario registrado. Permite buscar un usuario por medio de un número de identificación.

Requerimiento no.7: Modificar monto de dinero de un usuario. Permite modificar el monto de dinero de un usuario si este decide retirar o consignar dinero en su cuenta.

Requerimiento no.8: Cancelar la cuenta de un usuario. Permite cancelar la cuenta de un usuario registrado en el programa.

Requerimiento no.9: Registrar exusuarios que han cancelado su cuenta. Permite registrar a los usuarios que han eliminado su cuenta del programa. Además, se almacenará la fecha de la cancelación y el motivo por el cual se canceló la cuenta.

**Requerimientos no funcionales.**

Requerimiento no.1: Realizar un botón de “Undo” o “Deshacer”. El programa deberá contar con un botón para deshacer cambios o modificaciones realizadas durante el uso del programa.

Requerimiento no.2: Implementar algoritmos de ordenamiento. El programa debe contar con al menos un algoritmo de ordenamiento cuya complejidad temporal promedio sea de O(n2) y tres algoritmos de ordenamiento mucho más eficientes.

**Fase no.3: Búsqueda de soluciones creativas.**

Con el propósito de hallar la mejor solución para este problema, decidimos hacer uso de una de las técnicas operacionales más comunes para generar ideas creativas. En este caso, elegimos una Lluvia de ideas con el objetivo de reunir las mejores alternativas posibles pensando en el uso de estructuras de datos y algoritmos de ordenamiento.

**Posibles alternativas de solución.**

* ArrayList.
* Lista enlazada.
* Lista doblemente enlazada.
* Hash table.
* Cola.
* Pila.
* Array.
* Árbol binario de búsqueda.
* Algoritmo de ordenamiento tipo burbuja.
* Algoritmo de ordenamiento tipo inserción.
* Algoritmo de ordenamiento tipo selección.
* Algoritmo de ordenamiento tipo Montones.
* Algoritmo de ordenamiento tipo Conteo.
* Algoritmo de ordenamiento tipo Mezclas.
* Algoritmo de ordenamiento tipo rápido.

**Fase no.4: Transición de las ideas a los diseños preliminares.**

En este punto, descartamos las alternativas que no nos eran viables para la solución del problema. Entonces, descartamos los algoritmos de ordenamiento cuya complejidad temporal promedio fuese de O(n2), a excepción de uno, ya que era permitido usar al menos uno de estos. Por otro lado, para las estructuras de datos, decidimos descartar