**Universidad Peruana de Ciencias Aplicadas**



Competencia de Pensamiento Innovador

CURSO DE ALGORITMOS Y

ESTRUCTURAS DE DATOS - CC182

Carrera de Ciencias de la Computación

Sección: SS33

Alumno: Barrionuevo Gutiérrez, Daniel Ulises

Código: u201922128

Junio 2021

**CONTENIDO**

1. Uso del conocimiento
2. Identificación del problema
3. Idea de solución al problema
4. Propuesta
5. **USO DEL CONOCIMIENTO**

Para el desarrollo de este proyecto ha sido necesario tener conocimientos previos acerca de diversos temas, principalmente de programación orientada a objetos, recursividad, lambdas y templates para implementar estructuras de datos que se usaron tales como arboles binarios, listas, hash tables y vectores. Tales conocimientos previos nos permitieron realizar las investigaciones necesarias para implementar nuestras propias estructuras de datos en adición a las ya creadas en clase como el “Map” o el “Set”, siendo el 1ro un tipo de diccionario el cual mediante una llave se puede acceder a un valor, y el 2do guardando datos que solo se almacenaran una vez y los ordena (haciendo uso del TimSort que también fue investigado).

Gracias al conocimiento acerca de la complejidad algorítmica y el big o pudimos realizar las comparaciones necesarias de diferentes métodos (agregar, buscar, insertar, etc.) entre las estructuras de datos tanto las creadas en clase como las ya definidas por stl y las que fueron investigadas, así como la comparación de los diversos métodos de ordenamiento que se explicaron en clase tanto como el investigado, todo con el fin de poder tener una mejor eficiencia del programa.

1. **IDENTIFICA EL PROBLEMA**

Uno de los principales problemas fue lidiar con las restricciones computacionales (tiempo vs espacio) ya que al tener una inmensa cantidad de usuarios que se registraran cada uno con sus respectivos datos e información personal el espacio necesario era gigante. Otro problema que se tuvo en el desarrollo de este trabajo fue el tiempo que era requerido al tener que encontrar un usuario en específico, así como almacenar las diferentes opciones y que estas aparezcan al finalizar y abrir el programa.

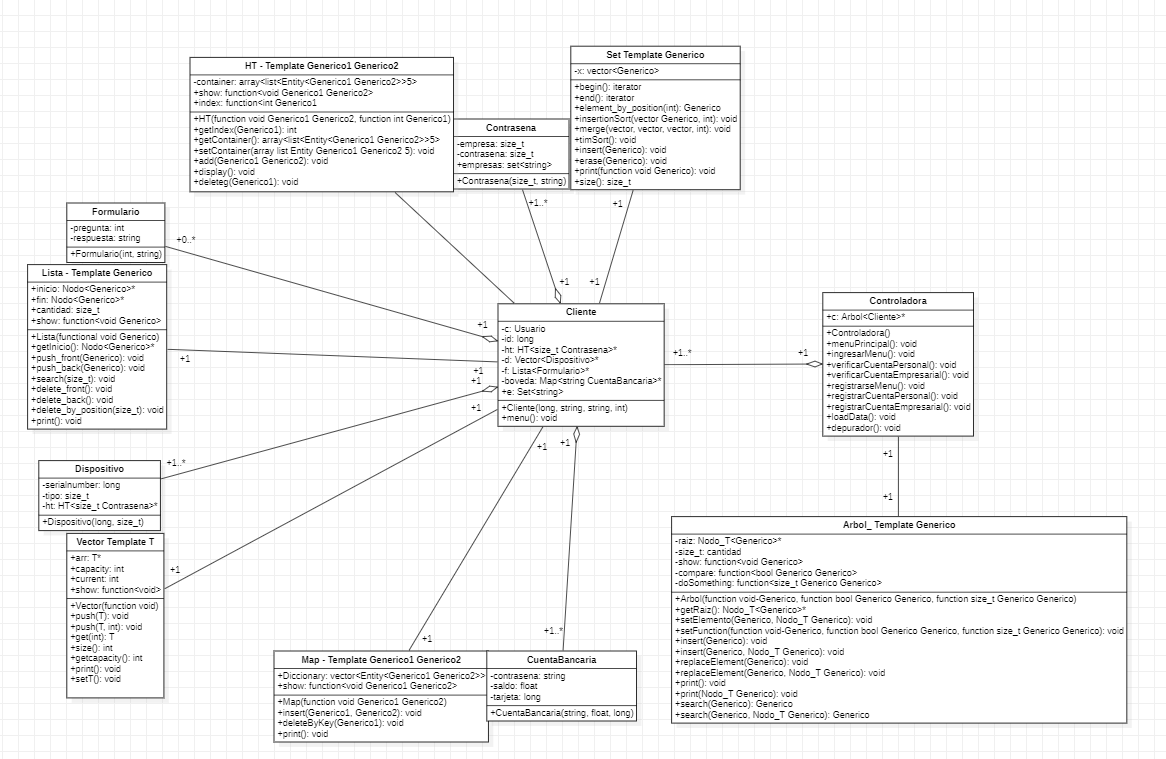
1. **IDEAR**

Basándonos en los problemas mencionados anteriormente, se llegaron a varias posibles soluciones como hallar una estructura de datos especifica dependiendo del tipo de entidad o método que se necesite, una inserción rápida de o(n), una búsqueda eficaz, o la capacidad de almacenar gigantes cantidades de datos (arboles binarios) así como insertar una indefinida cantidad de elementos(listas). Del mismo modo usar archivos para almacenar las diversas opciones y que al abrir o cerrar el programa estas sigan apareciendo independientemente del usuario que se inicie sesión.

1. **PROPUESTA**

Tomando en cuenta las soluciones anteriores, finalmente se optó por usar arboles binarios en el caso de guardar los clientes ya que al ser una cantidad inmensa y almacenarlos por su id facilita la búsqueda de uno al momento de querer iniciar sesión ya que este cuenta con una o(log(n)), brindándonos una considerable mejora cuando se habla de velocidad. Al momento de guardar las contraseñas se implemento el hastable ya que este podía usar como llave la empresa a la que está asociada y almacenar en cada posición del arreglo de listas una empresa con sus respectivas contraseñas, aprovechando también su rápida inserción de o(1), como en el caso anterior se mejora considerablemente la velocidad, sin embargo el espacio utilizado no es aprovechado al 100%. Mientras que en con los formularios hicimos usos de las listas ya que al tener un método de inserción con o(i) nos ayuda a mejorar el tiempo empleado. En el caso de las empresas a guardar las almacenamos en un archivo de texto y usamos la estructura set ya que esta permite almacenar valores no repetidos para no redundar las empresas que podrían agregarse evitando utilizar espacio de más. En cuanto a los dispositivos se uso el vector ya que este nos da un fácil acceso o(1) e inserción de una cantidad controlada de elementos. Finalmente se usó el map para poder guardar 2 tipos de datos en conjunto siendo en nuestro caso la contraseña y la cuenta bancaria evitándonos crear variables de más y asociarlas, ahorrando así el espacio que se pudo haber usado creando variables de más que se tiene que tomar en cuenta debido a la inmensa cantidad de usuarios que se registraran.

Estas soluciones se llegaron a tomar gracias a que cada integrante del grupo primero avanzo cada uno por su cuenta ciertas asignaciones que se acordaron previamente para luego verificar los avances y seguir codificando el programa en conjunto simultáneamente.



Para concluir, los resultados obtenidos mediante las pruebas del dataset que se implementó en la versión final se pudo confirmar que el árbol binario si resultó efectivo al momento de almacenar tales cantidades de datos (100000), creando, eliminando y guardando diferentes tipos de elementos en casa objeto usuario de manera eficaz y rápida. En adición a esto las empresas que los usuarios agregaban seguían permaneciendo gracias al archivo creado y ahorrando espacio al usar el set evitando redundar información. Sin embargo, una de las dificultades que se encontraron al momento de implementar la solución fue acceder a los elementos específicos de la tabla hash por lo que la solución que implementamos fue al momento de que el usuario quiera realizar modificaciones tendrá que hacerlo con todas las contraseñas relacionadas a una empresa (llave del hash).