### **Título del Proyecto: Social Structure - Implementación de estructuras no lineales**

### **1. Resumen Ejecutivo**

Este proyecto se centra en la aplicación de conocimientos avanzados de Estructuras de Datos para desarrollar una aplicación de escritorio en C++ que permita gestionar relaciones de amistad, recomendaciones de amigos y protección de datos mediante tecnologías de seguridad avanzada. Uno de los problemas principales que aborda el proyecto es la gestión y organización eficiente de grandes volúmenes de datos de usuarios y sus relaciones, así como la seguridad de dicha información. Para ello, los usuarios se almacenarán en una tabla hash, optimizando el acceso y la búsqueda de información.

La solución propuesta integra estructuras de datos no lineales, como grafos para representar redes de amistad y árboles de Merkle para asegurar la integridad de los datos. Además, la aplicación implementará compresión de datos mediante el algoritmo de Huffman y utilizará tecnología blockchain para garantizar la protección de la información en un entorno seguro y confiable. Esta combinación de herramientas y técnicas avanzadas busca ofrecer una plataforma eficiente, escalable y segura para el manejo de relaciones sociales, priorizando la privacidad y la integridad de la información del usuario.

### **3. Objetivos del Aprendizaje**

#### **3.1 Objetivo General**

El estudiante será capaz de diseñar y desarrollar una aplicación de escritorio en C++ que gestione relaciones de amistad y recomendaciones, utilizando estructuras de datos como tablas hash y árboles de Merkle para la protección de datos y la eficiencia en la búsqueda.

#### **3.2 Objetivos Específicos**

Al finalizar el proyecto, los estudiantes deberán ser capaces de:

1. Desarrollar un sistema funcional que almacene usuarios en una tabla hash y gestione relaciones de amistad.
2. Documentar correctamente el desarrollo del sistema, incluyendo manuales de usuario y guías de instalación.
3. Demostrar habilidades de resolución de problemas y realizar pruebas exhaustivas del sistema.

### **4. Enunciado del Proyecto**

El proyecto consiste en el diseño y desarrollo de una aplicación de escritorio en C++ que permita gestionar relaciones de amistad entre usuarios y proporcionar recomendaciones basadas en esas relaciones. El problema a resolver es la falta de una herramienta eficiente que facilite la administración de conexiones sociales, permitiendo a los usuarios agregar, eliminar y visualizar amigos de manera sencilla, así como recibir sugerencias para expandir su red de contactos.

**Alcance del Proyecto**

**Funcionalidades Principales:**

* Registro de usuarios y almacenamiento en una tabla hash para un acceso rápido y eficiente.
* Creación, eliminación y visualización de relaciones de amistad.
* Implementación de un sistema de recomendaciones que sugiera nuevos amigos basado en relaciones existentes.
* Seguridad en el manejo de datos utilizando un árbol de Merkle para verificar la integridad de las relaciones almacenadas.

**Entregables Esperados:**

* Una aplicación de escritorio funcional escrita en C++ que cumpla con los requisitos mencionados.
* Documentación técnica que incluya manuales de usuario, guías de instalación y diagramas de arquitectura del sistema.
* Un informe que detalle el proceso de desarrollo, incluyendo pruebas realizadas y resultados obtenidos.

Este proyecto busca no solo abordar un problema práctico en la gestión de relaciones sociales, sino también proporcionar a los estudiantes una experiencia enriquecedora en el desarrollo de software, aplicando conceptos aprendidos en el curso.

**4.1 Descripción del problema a resolver**En el ámbito de la gestión de relaciones sociales, los usuarios a menudo enfrentan dificultades para administrar y mantener sus conexiones de amistad. La falta de una herramienta eficiente puede llevar a que se pierdan interacciones valiosas y a que los usuarios no sepan cómo expandir su red de contactos de manera efectiva. El objetivo de este proyecto es diseñar y desarrollar una aplicación de escritorio que facilite la gestión de amistades, permitiendo a los usuarios agregar, eliminar y visualizar fácilmente sus relaciones. Además, la aplicación proporcionará recomendaciones personalizadas de nuevos amigos basadas en las conexiones existentes, promoviendo así una interacción social más rica y significativa.

**4.2 Alcance del proyecto**Establece los límites del proyecto para que los estudiantes sepan hasta dónde deben llegar. Esto incluye las funcionalidades esenciales que deben desarrollarse, como también las que son opcionales o recomendadas.

**Ejemplo**:

* **Alcance obligatorio:** La aplicación debe permitir a los usuarios gestionar sus amistades, incluyendo las funcionalidades de agregar, eliminar y visualizar conexiones de amistad. Además, la aplicación deberá ofrecer un sistema de recomendaciones personalizadas para nuevos amigos basado en las relaciones existentes, utilizando una tabla hash para almacenar la información de los usuarios.
* **Alcance opcional:** Se recomienda implementar funcionalidades adicionales como la integración de un sistema de recordatorios para mantener el contacto con amigos, o la posibilidad de categorizar amistades según diferentes criterios (por ejemplo, conocidos, amigos cercanos, etc.). También se podría considerar la inclusión de un módulo de análisis que brinde estadísticas sobre las interacciones del usuario con sus amigos.

**4.3 Requerimientos técnicos**Los estudiantes deberán utilizar C++ como lenguaje de programación principal para el desarrollo del proyecto. Se espera que integren Qt como plataforma para crear una interfaz gráfica de usuario (GUI), lo que permitirá una experiencia de usuario intuitiva y efectiva. Para el almacenamiento de datos, se implementará una tabla hash para gestionar eficientemente la información de los usuarios y sus relaciones.

Adicionalmente, se recomienda el uso de herramientas de control de versiones como Git para mantener un historial del desarrollo del proyecto y facilitar la documentación técnica.

**4.4 Entregables**Los estudiantes deberán entregar los siguientes elementos en un repositorio de GitHub:

* **Prototipo funcional de la aplicación:** Se debe entregar un archivo ejecutable (por ejemplo, un archivo .exe) del sistema desarrollado, junto con el código fuente.
* **Informe Técnico:** Un informe de no más de 10 páginas que describa el proceso de desarrollo, los retos encontrados y cómo se resolvieron.
* **Documentación del usuario:** Manual de uso del sistema con capturas de pantalla.

### **5. Metodología**

1. Los estudiantes deberán seguir los siguientes pasos y procesos para desarrollar el proyecto:
2. Investigación preliminar: Los estudiantes deben realizar una investigación sobre las tecnologías y herramientas que se utilizarán en el desarrollo de la aplicación, incluyendo C++ y Qt.
3. Diseño del sistema: Crear diagramas de flujo o esquemas de la aplicación que representen la estructura y funcionamiento del sistema, incluyendo la interacción con la tabla hash para el almacenamiento de usuarios.
4. Desarrollo: Implementar la solución en fases, asegurándose de cumplir con los requerimientos técnicos descritos, y enfocándose en la creación de una interfaz gráfica intuitiva.
5. Pruebas y ajustes: Realizar pruebas para verificar que la aplicación funciona correctamente en diferentes condiciones. Documentar las pruebas realizadas y sus resultados para garantizar el correcto funcionamiento del sistema.
6. Documentación: Elaborar la documentación técnica y del usuario, asegurando que incluya manuales y guías claras sobre el uso del sistema.

### **6. Desarrollo de Habilidades Blandas**

Para complementar el desarrollo técnico, esta sección se centra en las habilidades blandas que los estudiantes deberán mejorar a lo largo del proyecto individual.

**6.1 Proyectos Individuales**

Los proyectos individuales permiten a los estudiantes desarrollar autonomía y responsabilidad sobre su propio trabajo. En este tipo de proyectos, cada estudiante debe gestionar su tiempo, investigar, planificar y ejecutar todas las fases del proyecto de manera independiente.

**6.1.1 Autogestión del Tiempo**

Los estudiantes deben crear un cronograma personal para cumplir con los plazos establecidos. Esto les ayuda a mejorar su disciplina y capacidad de priorización, habilidades esenciales en cualquier entorno profesional.

**6.1.2 Responsabilidad y Compromiso**

En un proyecto individual, el estudiante asume la totalidad de las responsabilidades, desde la investigación hasta la entrega final. Esto fomenta el sentido de compromiso y permite una mayor personalización en la solución del problema planteado.

**6.1.3 Resolución de Problemas**

Trabajar de manera independiente impulsa a los estudiantes a buscar soluciones de forma creativa y a enfrentarse a los desafíos sin depender del apoyo constante de otros. Esto fortalece su capacidad para resolver problemas de manera autónoma.

**6.1.4 Reflexión Personal**

Al concluir el proyecto, el estudiante realiza una autoevaluación, reflexionando sobre sus decisiones, lo aprendido y las áreas en las que podría mejorar. Esta práctica promueve el desarrollo continuo y el autoaprendizaje.

### **7. Cronograma**

| Tipo | Fecha Inicio | Fecha Fin |
| --- | --- | --- |
| Asignación de Proyecto | Semana 9 | Semana 9 |
| Elaboración | Semana 9 | Semana 13 |
| Calificación | Semana 14 | Semana 14 |

Este **cronograma** es un ejemplo general y puede ser ajustado según las necesidades y el contexto del proyecto.

### **8. Evaluación**

### **Resumen de Puntuaciones**

| **Área** | **Puntos Totales** | **Puntos Obtenidos** |
| --- | --- | --- |
| Aplicación | 15 |  |
| Usuarios | 10 |  |
| Blockchain | 15 |  |
| Compresión de Datos | 15 |  |
| Reportes | 25 |  |
| Persistencia de Datos | 10 |  |
| Documentación | 10 |  |
| **Total** | **100** |  |

### **Detalle de la Calificaciòn**

| **Criterio** | **Descripción** | **Puntos Máximos** | **Puntuación Obtenida** |
| --- | --- | --- | --- |
| **1. Aplicación** |  | **15** |  |
| 1.1 Funcionalidad | Grado en que la aplicación cumple con los requisitos básicos y funciona correctamente. |  |  |
| 1.2 Interfaz de Usuario | Calidad de la interfaz de usuario, facilidad de uso, y experiencia general del usuario. |  |  |
|  |  |  |  |
| **2. Usuarios** |  | **10** |  |
| 2.1 Gestión de Usuarios | Implementación de funcionalidades de registro, inicio de sesión y control de usuarios en la aplicación. |  |  |
| 2.2 Experiencia de Usuario | Calidad y efectividad en la interacción del usuario con el sistema. |  |  |
|  |  |  |  |
| **3. Blockchain** |  | **15** |  |
| 3.1 Seguridad | Integración de tecnología blockchain para asegurar la información de los usuarios. |  |  |
| 3.2 Registro de Transacciones | Implementación y correcto funcionamiento del registro de transacciones en la blockchain. |  |  |
|  |  |  |  |
| **4. Compresión de Datos** |  | **15** |  |
| 4.1 Implementación | Uso de algoritmos de compresión para optimizar el almacenamiento y transmisión de datos. |  |  |
| 4.2 Eficiencia | Grado de compresión logrado y su impacto en el rendimiento general de la aplicación. |  |  |
|  |  |  |  |
| **5. Reportes** |  | **25** |  |
| 5.1 Generación de Reportes | Funcionalidad y efectividad en la generación de reportes de actividad o uso dentro de la aplicación. |  |  |
| 5.2 Precisión y Claridad | Claridad en la presentación de los datos y precisión en la información presentada en los reportes. |  |  |
|  |  |  |  |
| **6. Persistencia de Datos** |  | **10** |  |
| 6.1 Guardado de Información | Implementación de métodos de almacenamiento y recuperación de datos. |  |  |
| 6.2 Recuperación de Datos | Eficiencia y rapidez en la recuperación de datos almacenados en el sistema. |  |  |
|  |  |  |  |
| **7. Documentación** |  | **10** |  |
| 7.1 Manual de Usuario | Claridad y detalle en el manual de usuario, con guías de uso y resolución de problemas comunes. |  |  |
| 7.2 Manual Técnico | Calidad y detalle en la documentación técnica, incluyendo diagramas y explicación de la arquitectura. |  |  |

### **Comentarios Generales:**