Taller I

Se solicita desarrollar un programa para manejo de software que cumpla con los siguientes requerimientos:

* El usuario debe ser capaz de agregar o eliminar software de su biblioteca.
* Los softwares se deben dividir en tipos. (juegos, ofimática, producción, Navegadores, Seguridad, Comunicación, etc.)
* La biblioteca es compartida en un computador donde hay múltiples usuarios. Por lo tanto, si más de un usuario utiliza un software, se debe tener el permiso de todos los usuarios para borrar el software.
* Los softwares tienen restricciones de edad y niveles de seguridad para acceder a ellos.
* Se debe poder acceder al “log” de cada software (se necesita un nivel de seguridad mínimo para poder acceder al log) para conocer el uso que le están dando los usuarios.

Tener en cuenta:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Atributos | | | | |
| Usuarios | Usuario | Contraseña | Edad | Correo | Acceder log |
| Administrador | o | o | ? | o | o |
| Usuario normal | o | o | >=18 | o | x |
| Niño | o | o | <18 | x | x |

Los softwares comparten la siguiente lista de atributos:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre | Developer | Clasificación de edad | Lista de usuarios | Precio |

Y se dividen en los siguientes tipos:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Juegos | Ofimática | Producción | Navegador | Seguridad | Social |

Cada uno de los tipos de software tienen atributos únicos:

|  |  |
| --- | --- |
| Juegos | Genero (FPS, MMO, PUZZLE, ETC -> mínimo 10 géneros) |
| Ofimática | Cantidad de archivos creado por usuario |
| Producción | Tipo de solución (video, música, streaming, fotos) |
| Navegador | historial de navegación (ultimas 10 páginas visitadas) |
| Seguridad | Tipo de Malware (Ransomware, Spyware, botnets, rootkits, gusanos, troyanos) |
| Social | Usuarios (Cuantos "amigos" tiene un usuario agregado) |

*\*Utilizar la información que está en los paréntesis como información única, por ejemplo, para juegos utilizar los 4 géneros que hay descritos ya agregar 6 más para completar los 10. En caso de los softwares de seguridad, mantener los 6 descritos y no agregar más.*

**Cada software debe tener métodos que sean capaces de manejar los atributos únicos antes mencionados, por ejemplo:**

Ofimática -> Método para crear un archivo o borrar un archivo (int x+1 o x-1 donde x es la cantidad de archivos).

Social -> Agregar o eliminar usuarios (los “usuarios” deben ser los mismos usuarios que están inscritos en el computador, piensen que son sistemas sociales internos).

Condiciones de entrega

* Dentro de la función main, se debe iniciar poblando la “base de datos” con un mínimo de 15 usuarios (1 administrador, 4 niños y 10 usuarios normales).
* Dentro de la función main, se debe iniciar poblando la “base de datos” con un mínimo de 20 juegos (2 de cada género y mínimo 5 para 18+).
* Dentro de la función main, se debe iniciar poblando la “base de datos” con un mínimo de 5 software de ofimática.
* Dentro de la función main, se debe iniciar poblando la “base de datos” con un mínimo de 4 software de producción.
* Dentro de la función main, se debe iniciar poblando la “base de datos” con un mínimo de 2 navegadores.
* Dentro de la función main, se debe iniciar poblando la “base de datos” con un mínimo de 6 software de seguridad (1 de cada tipo).
* Dentro de la función main, se debe iniciar poblando la “base de datos” con un mínimo de 2 software social (mínimo 1 tiene que ser 18+)
* Los softwares de tipo producción son 18+
* Los softwares de tipo seguridad solo son accesibles por administrador.
* Los softwares de social deben ser inicializados con minimo 2 “amigos” por usuario. (los “niños” solo pueden tener a otros niños como amigos)
* Se debe crear un menú contextual por consola que permita realizar las acciones antes mencionadas. (debe contar en un inicio con un login y debo ser capaz de moverme por el menú ingresando comandos de texto).
* Al realizar logout, se deben entregar 2 opciones. Logear con otro usuario o terminar el programa.

Entregables

* Se debe subir un link de github con el proyecto en visualización publica a campus virtual con un plazo máximo hasta el 06/10/2023.
* El programa debe ser capaz de compilar con g++ (buscar información sobre mingw o utilizar codespace de github para realizar el proyecto). Se ocupará el comando “g++ -o main main.cpp” para compilar el proyecto.
* Dentro del README del repositorio se debe incluir toda la información sobre los objetos que se crean en el main durante el inicio del programa. Por ejemplo:

*admin1: { usuario:”Jose”; contraseña:”123123”; edad:”30”; correo:”admin@admin.cl”; log:”true”}.*

* Se debe utilizar programación orientada a objetos para cumplir los requerimientos (ver pauta).

**Pauta evaluación**

|  |  |  |
| --- | --- | --- |
|  | Pts totales | Pts obtenidos |
| **Diseño del Programa (50 puntos)** |  |  |
| **Definición de Clases (10 puntos)** |  |  |
| Se han definido clases relevantes para el problema. | 5 |  |
| Se han establecido atributos adecuados en las clases. | 5 |  |
| **Herencia y Polimorfismo (10 puntos)** |  |  |
| Se ha utilizado la herencia de manera adecuada para modelar las relaciones entre las clases. | 5 |  |
| Se ha implementado el polimorfismo de manera efectiva. | 5 |  |
| **Encapsulamiento (10 puntos)** |  |  |
| Se han declarado atributos privados en las clases. | 5 |  |
| Se han proporcionado métodos públicos para acceder y modificar los atributos. | 5 |  |
| **Abstracción (10 puntos)** |  |  |
| Se han creado clases abstractas o interfaces cuando sea necesario. | 5 |  |
| Las clases derivadas han implementado los métodos requeridos de manera adecuada. | 5 |  |
| **Diseño de Menú (10 puntos)** |  |  |
| Se ha diseñado un menú interactivo por consola con opciones claras y comprensibles. | 5 |  |
| Las opciones del menú están relacionadas con las funcionalidades del programa. | 5 |  |
|  |  |  |
| **Implementación del Programa (35 puntos)** |  |  |
| **Funcionalidad General (10 puntos)** |  |  |
| El programa se ejecuta sin errores graves. | 5 |  |
| Los usuarios pueden agregar, eliminar y buscar software en la biblioteca. | 5 |  |
| **Funcionalidades Específicas (15 puntos)** |  |  |
| Las funcionalidades específicas, como agregar amigos a software social, funcionan correctamente. | 10 |  |
| Se han implementado al menos tres funcionalidades adicionales más allá de las básicas. | 5 |  |
| **Documentación (10 puntos)** |  |  |
| Se ha proporcionado documentación clara y legible en el código. | 5 |  |
| Se incluyen comentarios que explican las secciones críticas del programa. | 5 |  |
|  |  |  |
| **Usabilidad y Presentación (20 puntos)** |  |  |
| **Interfaz de Usuario (10 puntos)** |  |  |
| La interfaz de usuario es amigable y fácil de usar. | 5 |  |
| El menú interactivo es intuitivo. | 5 |  |
| **Organización del Código (10 puntos)** |  |  |
| El código está organizado de manera ordenada y legible. | 5 |  |
| Se siguen buenas prácticas de codificación. | 5 |  |

* Fecha de entrega 06/10 23.59 a través de campus virtual. Consultas -> [jose.veas@ce.ucn.cl](mailto:jose.veas@ce.ucn.cl)
* Se evaluará con nota 1 en caso de:
  + Copia
  + No entregar documentación
  + Programa no compila
  + No entregar en fechas indicadas

\*El profesor se reserva el derecho de solicitar revisión con alumno en caso de encontrarse con situaciones sospechosas que podrían modificar la evaluación.