Proyecto programado # 2

SISTEMA DE RESERVAS CALENDAR-PRO

En este proyecto el estudiantado utilizará herencia y polimorfismo para la resolución de un problema aplicando Programación Orientada a Objetos con C++.

El diseño técnico de la solución depende de usted. Puede realizar la tarea individualmente o en parejas. Se recomienda que trabaje con alguien diferente cada vez.

Descripción del problema

La aplicación de calendario CalendarPro le permite a las y los usuarios de una organización crear, manipular y seguir todo tipo de reuniones y eventos, y hasta compartirlos con otras personas. La funcionalidad principal es que **permite personalizar un calendario por persona**, estableciendo también vínculos a calendarios de otras, y permitiendo así compartir información y organizar su día.

Como podrá intuirse fácilmente**, los calendarios son simplemente conjuntos de reservaciones, pero las acciones son diferentes** de acuerdo con los permisos con que cuenta quien lo ve, y de acuerdo con el tipo de reservaciones en cada caso. Por ello, debe implementar una estructura que utilice herencia y polimorfismo para generar los calendarios, accesos y acciones correctas en cada caso.

La aplicación mantiene los siguientes elementos fundamentales: usuarios y reservaciones.

Usuarios[1]. La organización tiene usuarios en varios niveles de jerarquía.

- **Individual Contributor:** Es una persona que no tiene nadie a cargo, solo es responsable de sí mismo.
- **Manager:** Es una persona que tiene subalternos a su cargo, adicional a sus responsabilidades regulares.

Todo registro de usuario en el sistema tiene las características típicas de una persona empleada, como un nombre, cédula de identidad, y puesto de trabajo. También, posee un nombre de usuario que la identifica de forma única, y una contraseña de acceso al sistema.

Al abrir la aplicación de CalendarPro, lo primero que debe suceder es un **inicio de sesión** donde pregunta a la persona el nombre de usuario y su contraseña. A partir de ahí se debe generar una validación para ver si las credenciales son correctas. Una vez se haya ingresado al sistema, **todos los movimientos que se realicen van a ser desde la perspectiva de la o el usuario ingresado**. También deberá mostrarse una opción de cerrar sesión, con lo cual volvería a la opción de inicio de sesión.

[1] El término "Usuario" se usa para generalizar una entidad de personas que usan el sistema, pero se aclara que pueden ser hombres o mujeres.

Reservaciones. La aplicación permite generar distintos tipos de reservaciones, cada uno son sus características. Todas las reservas por su naturaleza tienen fecha y tiempo de inicio y finalización, además de compartir otros rasgos como una breve descripción.

- **Reunión**: Cuando una persona – la organizadora – invita a varias personas a un encuentro sincrónico. Por lo tanto, posee como mínimo una lista de asistentes y un lugar de reunión. En este caso, se envía invitación al calendario de todas las personas involucradas.
- **Cita personal**: Cuando la persona necesita bloquear su calendario por un evento personal. No posee detalles como el sitio de reunión, ni incluye a otras personas.
- **Actividad social**: El uso de una zona común de la oficina para un evento de varias personas, pero en donde todas las personas involucradas participan de igual forma como organizadoras.
- **Evento diario**: Permita a la persona registrar en su calendario eventos de todo un día completo, tales como cuando se va a estar de vacaciones.

El sistema debe proveer **un menú** – una vez se inició sesión – que le permita a quien usa el programa ver y manipular su calendario y el de otras personas. La relación entre usuarios y calendarios puede ser importante por los niveles de acceso que tiene cada persona. De forma general se le debe permitir a la persona:

- **Ver su calendario** de eventos aceptados
- **Ver y aceptar o rechazar** invitaciones de eventos que vienen de otras personas y que tiene pendientes
- **Crear una nueva reservación** en su calendario, con todas sus características, para lo cual deberá especificarse el tipo. Al crearse, si esta implica a otras personas, debe contemplarse todas las acciones necesarias sobre las demás.
- **Cancelar una reservación** o su participación. Al cancelar una participación, si la persona era la organizadora, o bien si es la única involucrada, el evento se cancela para todas las personas involucradas y se les elimina de sus calendarios.
- **Modificar una reservación** en su calendario, para lo cual debe ser la organizadora o la única persona involucrada en el evento. Al cambiar algo, el cambio debe también notificarse a todas las involucradas.
- **Ver calendario de otras personas**. En este caso, si la persona es subalterna podrá ver todo el calendario con los nombres de las actividades, mientras que si no es subalterno solo podrá ver la disponibilidad de la persona, sin conocer detalles.

Entonces, el sistema es el mismo para todas las personas. La información de las personas usuarias deberá permanecer almacenada en un archivo.

Idealmente, también sus calendarios, de forma tal que al iniciar el programa se carguen los datos. Pero si no, simplemente se asumirá que al iniciar el programa todos los calendarios están vacíos.

Interacción con las personas usuarias

Debe contener un inicio de sesión, un menú principal y en cada caso solicitar los datos que amerite.

Para mostrar un calendario, cuenta con libertad creativa. Puede mostrarlo en una matriz, o bien mostrarlo como una lista de elementos, o incluso usando colores. Pero en cada caso debe estar siempre ordenada por fecha y hora. Opcionalmente se sugiere implementar una funcionalidad de búsqueda que le permita ver el calendario de un día específico.

Manejo de excepciones

Utilice manejo de excepciones si lo requiere, para evitar que su ejecución caiga en cualquier momento.

Debe diseñar una excepción si la persona acepta una reservación que causa un conflicto con otra aceptada. Esta excepción podría heredar de exception, de la librería estándar, o bien crearse como un tipo de dato nuevo.

Usuarios del sistema

Diseñe un formato de archivo de texto que permita establecer todos los datos de cada usuario. Al iniciar el programa, debe cargarlos de ahí. Para agregar o eliminar usuarios, puede manejarse directamente modificando este archivo. No es necesario que su programa implemente un módulo para gestionar cuentas.

Suposiciones de programación

Realice un archivo README.md, donde debe especificar suposiciones técnicas y funcionales que hizo a la hora de desarrollar su programa. Por ejemplo, una aclaración válida podría ser cómo se mantiene la lista de usuarios del sistema. Los supuestos que no estén claros en este documento de enunciado, deben ir en este archivo.

En este mismo archivo, especifique la forma en que se debe compilar todo el proyecto. También, defina especifique qué características debe cumplirse para agregar nuevos tipos de eventos al sistema en un futuro.

Formato de entrega

Todo el desarrollo debe mantenerse desde el inicio en un **repositorio GIT**, y cada persona es responsable de realizar los cambios individuales sobre el código. La **carga de desarrollo individual** documentada en el control de versiones **podrá ser tomada en cuenta para la ponderación de la nota** de acuerdo con los aportes que realizó cada persona a la solución.

El código fuente debe entregarse como un link a un repositorio de acceso público, antes de la hora límite establecida en el aula virtual del curso.

Restricciones técnicas

- Debe entregar un diagrama de clases UML con el diseño de la solución.
- El código entregado debe compilar.
- El código entregado debe seguir **buenas prácticas de programación** en la organización de archivos de proyecto: archivos de encabezado, archivos de implementación, y un main que incluya solo un llamado inicial a otro método.
- En general, debe utilizar el paradigma de **Programación Orientada a Objetos**, herencia y polimorfismo.
- Utilice clases abstractas para tipos de datos creados que no tienen sentido por sí solos.
- Solo puede utilizar temas vistos en clase.
- Debe utilizar **estructuras de datos dinámicas** de elaboración propia, para los elementos que pueden crecer o decrecer en ejecución.
- Puede incluir y utilizar **iostream**, **fstream**, **string**, y librerías que le permitan jugar con **fechas**.
- **Su programa debe ser fácilmente escalable, por lo que en el mismo código establece requisitos para la creación de nuevos tipos de reservas en el calendario o de nuevos tipos de usuario por medio de clases abstractas.**

Evaluación

Solo se obtiene calificación si cumple **todas** las restricciones técnicas establecidas. Caso contrario, se evalúa como cero. Si cumple las restricciones, entonces se sigue la siguiente rúbrica de evaluación.

Rubro	**Peso**
Uso de encapsulamiento POO – constructores, destructores, miembros de clase, valores estáticos, niveles de acceso. El main solo hace un llamado inicial.	10%
Manejo de usuarios de sistema, inicio y cierre de sesión	15%
Uso correcto de herencia – clases base, clases derivadas, niveles de acceso	20%
Uso correcto de polimorfismo – para determinar acciones diferentes según tipo de dato	25%
Interacción ordenada con las personas usuarias y menú	5%
Estructuras de datos para la construcción y manipulación de calendarios – uso responsable de memoria	10%
Manejo de excepciones	10%
Archivo README con información solicitada e inclusión de diagrama UML	5%

Desarrollo opcional

Opcionalmente, puede programar alguna de estas 2 mejoras, por un 5% adicional en la nota de la tarea:

- Por un 3% adicional, diseñe un sistema de encriptación para las contraseñas de las personas, de forma que en el archivo no se escriban tal como son.
- Por un 7% adicional, haga que los calendarios se guarden en archivos de texto y se carguen al correr el programa.