Universidad de los Andes
ISIS 1126 – Sección 3
Diseño y Programación Orientada a Objetos
Proyecto 3 – Entrega 1

DOCUMENTO DE DISEÑO

Como se realizó para entregas anteriores, la aplicación se estructura en dos partes: interfaz y modelo. **Nota:** Para consultar los diagramas de clases, por favor diríjase a los archivos llamados "Diagrama Modelo.pdf" y "Diagrama Interfaz.pdf" en la carpeta de la Entrega 1.

El modelo implementa el patrón de fachada, la cual está representada por la clase llamada CoordinadorProyecto. Esta clase contiene todos los métodos que la aplicación le ofrece al usuario. Además, existen otras clases poco sorprendentes, como lo son Participante, ReporteParticipante (calcula estadísticas), Proyecto y Actividad. Para este Proyecto 3, también se definieron las clases PaqueteDeTrabajo y Tarea.

Como se tenía en entregas pasadas, la clase Proyecto es abstracta, pues solo tiene implementados un método para agregar participantes y getters. Los métodos que manipulan directamente el registro de actividades se encuentran implementados en la clase GestorActividades, la cual hereda de Proyecto. Esta relación de herencia se realizó para orientar el diseño al principio de *Single Responsibility* de SOLID.

En las anteriores entregas, la clase Actividad contenía directamente la información que se debe conocer de una actividad en la aplicación. No obstante, como para este Proyecto 3 se piden requerimientos adicionales que, en principio, incluirían que las tareas conocieran a las actividades que las componen, se decidió usar el principio proxy para evitar almacenar las actividades duplicadamente. De esta forma, la clase Actividad es una interfaz implementada por dos clases: ProxyRegistro y Registro. La clase Registro es la que ahora contiene toda la información relevante a la actividad, mientras que ProxyRegistro conoce a su Registro correspondiente mediante un ID. Así, las actividades estarán ahora almacenadas en un único lugar: la clase AlmacenActividades, y cada instancia de ProxyRegistro es la única que sabe cómo acceder a su respectiva actividad en el AlmacenActividades. Entonces, las clases Proyecto y Tarea tendrán como "actividades" a instancias de ProxyRegistro, los cuales sirven de punteros para acceder a las actividades reales.

Finalmente, las clases ArchivadorProyecto y ArchivoUsuarios se mantienen inalteradas de entregas pasadas, pues se siguen encargando de cargar y guardar toda la información que necesite ser consultada.

Otras consideraciones importantes:

- El tipo de una actividad será, obligatoriamente, igual al tipo de la tarea que está desarrollando.
- Todos los tiempos serán medidos en minutos.
- Lo que un Proyecto conoce como su WBS es una instancia de PaqueteDeTrabajo. Esta instancia corresponde a la raíz del WBS.

DOCUMENTACIÓN DE MANEJO DE ERRORES

El presente apartado tiene como objetivo principal el mostrar los posibles errores que pueden surgir durante la ejecución del programa, así como también explicar los diferentes mecanismos de manejo de errores empleados para garantizar el correcto flujo de la aplicación.

Para facilitar el identificar donde se presenta cada error, cada categoría presentada representa un menú de la interfaz gráfica con la que interactúa el usuario

1.) INGRESO A LA APLICACIÓN:

Usuario no registrado: Si el usuario intenta iniciar sesión sin haberse registrado previamente, el programa arrojará un aviso indicándole al usuario que el login ingresado no se encuentra registrado en el sistema.

Para saber si el usuario esta registrado, la clase MenuLogin obtiene acceso a la clase ArchivoUsuarios por medio de la clase padre VentanaAplicacion. Posteriormente, se invoca el método getParticipante() con el login ingresado como parámetro, el cual verifica si el usuario está presente en el HashMap InfoUsuarios. En caso de que el retorno sea nulo, la clase MenuLogin llamará el método UserNotFound() de la clase PanelLogin1. Finalmente, este último método actualizara el panel para mostrar el error.

Campos por llenar: Si el usuario trata de registrarse sin suministrar todos los datos solicitados, es decir si no ingresa el login o el nombre, el programa le impedirá ingresar al sistema y le notificará para que diligencie los campos faltantes.

Para cerciorarse que todos los campos de texto hayan sido diligenciados, cuando se presiona el botón de entrar, la clase PanelLogin2 revisa si algunas de las cadenas recibidas (login y nombre) es igual a la cadena vacía, en caso de ser así, se actualizará el panel en cuestión con el mensaje del error.

Usuario ya existente: Si al momento de registrarse el usuario ingresa un login que ya fue registrado en el sistema, la aplicación le avisara al usuario para que pruebe con un login distinto.

Para registrar si un login ya fue usado con anterioridad, la clase MenuLogin obtiene acceso a la clase ArchivoUsuarios por medio de la clase padre VentanaAplicacion. Posteriormente, se invoca el método getParticipante() con el login ingresado como parámetro, el cual verifica si el usuario está presente en el HashMap InfoUsuarios. Si el retorno no es nulo, significa que ya existe una cuente con el login en cuestión, tras lo cual se llamará el método userExistent () de la clase PanelLogin2, método el cual actualizará el panel para mostrar el error.

Botón Continuar: Si por alguna razón desconocida las medidas descritas no surten efecto, existe un mecanismo de respaldo para garantizar el correcto funcionamiento de la aplicación. Para avanzar al siguiente menú es necesario presionar el botón “continuar”, si alguno de los casos de error se presenta, el programa mantendrá el botón inhabilitado hasta recibir un input valido.

Para la aplicación de este mecanismo, el programa fue diseñado para que por defecto el botón este deshabilitado. La única forma de habilitarlo dentro del flujo normal del sistema es llamando el método enableBotonContinuar() de la clase VentanaAplicacion. Cabe aclarar que la mecánica anterior no es exclusiva de este menú, sino que aplica para todas las ventanas que forman parte de la interfaz, siempre que se quiera avanzar a la siguiente.

2.) ELECCION DEL PROYECTO

Selección del proyecto: Para evitar que el usuario elija un proyecto no existente, se muestra una lista desplegable con los diferentes proyectos disponibles a los que pertenece el usuario. Si bien tal decisión no soluciona ningún error directamente, sí que previene algunos errores con su implementación.

La clase PanelEleccion1 es la encargada de mostrar la lista desplegable y de agregar nuevos proyectos a esa lista cuando sean creados. Para garantizar que solo aparezcan los proyectos correspondientes al usuario, La clase MenuEleccionProyecto obtiene acceso a la clase ArchivoUsuarios por medio de la clase padre VentanaAplicacion.

Posteriormente, se invoca el método getProyectosUsuario() con el login ingresado como parámetro. Este método retorna todos los proyectos de los que forma parte el usuario, si el retorno es Null, se desactivará la lista desplegable llamando el método disableFields() de la clase PanelEleccion1. Por otro lado, si el valor de retorno es diferente a Null, se recorrerá el ArrayList con los proyectos para irlos agregando a la lista desplegable con el método addProyectoDesplegable() de la clase PanelEleccion1.

Numero de tipos de actividades: En el misma ventana también se utiliza otra lista desplegable para limitar el número de posibles inputs del usuario. Cuando se desea crear un nuevo proyecto, para ingresar el número de tipos de actividad se utiliza el mismo mecanismo, esto para evitar que si se coloca un número muy grande sea imposible representar gráficamente la ventana emergente que pide al usuario los tipos de actividad.

En este caso las opciones de la lista desplegable siempre serán las mismas, por lo que la única que se encarga de su implementación es la clase PanelEleccion2 dentro de su constructor.

Para no ser tan reiterativo con este error, su manejo de omitirá de aquí en adelante, ya que la lógica detrás de su implementación es muy similar para todos los casos.

Campos por llenar 2: Si al momento de crear un proyecto algún campo no fue diligenciado, el programa notificara al usuario para completar el campo faltante.

Para cerciorarse que todos los campos de texto hayan sido diligenciados, cuando se presiona el botón de entrar, la clase PanelEleccion2 revisa si algunas de las cadenas recibidas (login y nombre) es igual a la cadena vacía, en caso de ser así, se actualizará el panel en cuestión con el mensaje del error.

Para no ser tan reiterativo con este error, su manejo de omitirá de aquí en adelante, ya que la lógica detrás de su implementación es muy similar para todos los casos.

3.) PROYECTO

Agregar un participante repetido: A la hora de agregar un participante, en caso de ingresar como login el del usuario que está haciendo uso de la aplicación, el programa no lo vuelve a agregar a la lista de participantes y cierra la ventana emergente.

Al seleccionar la opción agregar participante, la clase PanelEleccion2 invoca el método newParticipantSettings de la clase padre MenuProyecto. Luego se crea una instancia de la clase DialogAgregarPartcipante, cuando el botón “aceptar” se presione, se llamará al método continuar(). Este invocara a su vez al método loginRegistrado() de la clase padre, el cual retorna True si el login del participante ya está registrado en ArchivoUsuarios y False en caso contrario. Finalmente, si el valor retornado fue True se imprimirá un mensaje por consola indicado que el usuario ya está registrado.