Tasky-chan

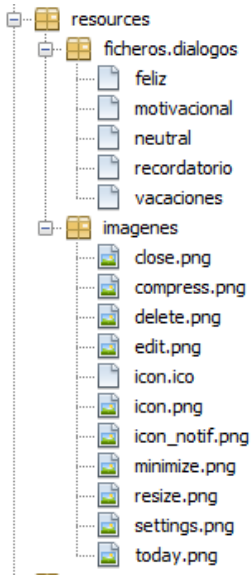
Tasky-chan
Manual de Sistema

Versión 1.0

22/05/18

Descripción y ubicación de los archivos

El estado de Tasky es guardado en un archivo binario llamado *tasky.bin*. Este archivo puede encontrarse, en el caso de Windows, en el directorio %appdata%. En el caso de Linux, en /home/. Los demás archivos, cuya información es estática, se encuentran en el paquete *resources*.



dialogos

En este paquete hay cinco archivos de texto, los cuales contienen las líneas que Tasky dirá durante su ejecución, dependiendo de su estado. En *feliz* se encuentran las líneas cuando la puntuación es mayor a 10. En *motivacional* cuando esta es menor a -10. En *neutral*, las interacciones para cuando esté en ese rango. En *recordatorio* se encuentran las líneas que Tasky dirá para recordarle al usuario que tiene alguna tarea urgente. En *vacaciones*, las líneas que dirá cuando se encuentre en dicho modo.

imagenes

Se encuentran todas las imágenes que se muestran en el programa, como botones e íconos.

Clases principales

Las clases principales están dentro del paquete *tasky*. Más adelante se describirán las clases que estos contienen a fondo.

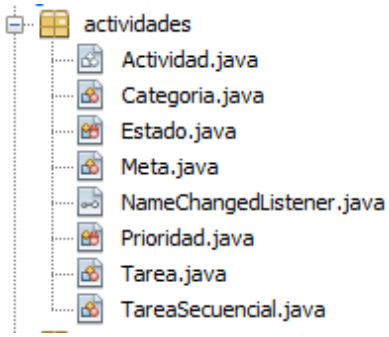
- ***Tasky*.java:** la clase principal de todo el programa. Se usó el patrón de diseño "singleton," para asegurar que solo exista una instancia de la clase en todo el programa. Tiene como atributos: una lista con categorías, el usuario, los puntos, un lógico que indica si está en vacaciones o no, y una cadena de texto con el estado actual. En esta

```
private final DoubleLinkedList<Categoria> categorias;
private final Usuario usuario;
private int puntos;
private boolean enVacaciones;
private String estado;
```

  clase se encuentra la lógica necesaria para serializar a Tasky y guardar su estado en un archivo binario.
- **Usuario.java:** contiene la información del usuario. Actualmente solo se guarda el nombre de este.

actividades

en este paquete se encuentran las clases que describen la lógica de las actividades como las metas, las tareas y las tareas secuenciales. Consta de las siguientes clases:



Actividad.java

Define una actividad abstracta, la cual tiene un nombre, una descripción, una fecha de fin, un estado y una prioridad. Además, contiene tres listeners, de los cuales dos tienen el modificador *transient* (no se serializan). Estos son: *ActivityFinishedListener*, el cual se produce cuando la actividad llama a su método "terminar", *NameChangedListener*, que se produce cuando la actividad cambia su nombre, y *StatusChangedListener,* cuando la actividad cambia su estado.

Categoria.java

Define una categoría, la cual tiene un nombre, unas actividades, un lógico que indica si es eliminable o no, y un *NameChangedListener.*

Estado.java

Es una enumeración que tiene como literales a los tres estados posibles de una actividad: TERMINADA, PENDIENTE o VENCIDA.

Meta.java

Una actividad concreta, que describe una actividad la cual tiene un contador el cual hay que completar para que la actividad cuente como completada. Tiene un *ValueChangedListener*¸ que informa cuando el valor de su contador cambia.

Prioridad.java

Una enumeración que tiene como literales a las distintas prioridades posibles que puede tener una actividad. Cada una de estas prioridades tiene asignado un valor que se usará para calcular los puntos cuando se complete (o no se complete) la actividad.

Tarea.java

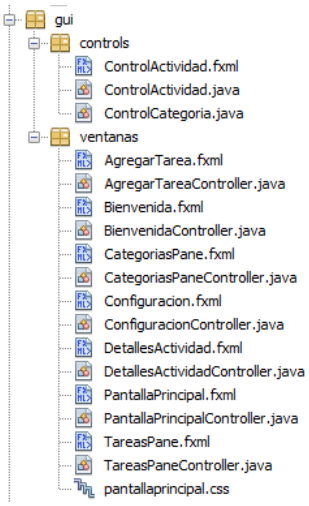
Una tarea concreta. Actualmente es idéntica a Actividad.

TareaSecuencial.java

Una tarea que consta de una serie de pasos que tienen que ser completados en una secuencia específica. Solo se termina cuando todos sus pasos son completados.

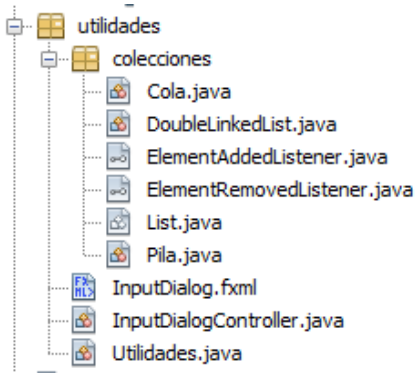
gui

Aquí se encuentra toda la lógica que tiene que ver con la parte gráfica de la aplicación: las ventanas y los controles personalizados, usados para representar tanto categorías como actividades en general.



utilidades

Como su nombre lo indica, contiene clases auxiliares para facilitar ciertos procesos a lo largo del programa, como abrir ventanas, archivos, comprobar si una fecha está en cierto rango u operaciones con cadenas.



- **colecciones:** este paquete contiene las estructuras de datos que se usan en el programa. Cada una de estas tiene asociadas un *ElementAddedListener*, que se llama cuando un elemento se añade, y *ElementRemovedListener*, que se llama cuando un elemento es removido.
    - **Cola.java:** representa una cola.
    - **DoubleLinkedList.java:** representa una lista doblemente enlazada.
    - **List.java:** representa una lista abstracta.
    - **Pila.java:** representa una pila
- **InputDialogController.java** e **InputDialog.fxml:** representan una ventana que pide datos al usuario.
- **Utilidades.java:** tiene métodos estáticos que pueden ser de utilidad más adelante.