```java
/**
 * Aplicación.java
 * <p>
 * Aplicación con interfaz gráfica para emplear una aplicación legada sobre
 * una máquina mainframe.
 * <p>
 * Radu Constantin Robu y Alberto Pérez
 */
package P3.Control;

import P3.Control.MainframeAPI.RESPUESTAS_INICIO_SESION;
import P3.Modelo.Tarea;
import P3.Modelo.Tupla;
import P3.Vista.TasksJobWrapperVista;

import java.io.IOException;
import java.util.List;

public class TasksJobWrapper implements OyenteVista {
    private static final String TITULO_ERROR_FICHERO_TAREAS =
            "Error fichero tareas";
    private static final String MENSAJE_ERROR_FICHERO_TAREAS =
            "No se ha creado el fichero de tareas.";
    private static final String TITULO_CONFIRMACION_FICHERO_TAREAS =
            "Nuevo fichero creado";
    private static final String MENSAJE_CONFIRMACION_FICHERO_TAREAS =
            "Fichero de tareas creado correctamente.";
    private static final String TITULO_ERROR =
            "ERROR";
    private static final String MENSAJE_ERROR_INESPERADO =
            "Error inesperado, cierre la app.";
    private static final String TITULO_ERROR_ELIMINAR_TAREA =
            "Error eliminar tarea";
    private static final String MENSAJE_ERROR_ELIMINAR_TAREA =
            "Id de la tarea no existe.";
    private static final String TITULO_CONFIRMACION_ELIMINAR_TAREA =
            "Tarea eliminada";
    private static final String MENSAJE_CONFIRMACION_ELIMINAR_TAREA =
            "Tarea eliminada correctamente.";
    private static final String TITULO_ERROR_ANYADIR_TAREA =
            "ID incorrecto";
    private static final String MENSAJE_ERROR_ID_INCORRECTO =
            "El ID introducido no es correcto.";
    private static final String TITULO_CONFIRMACION_ANYADIR_TAREA =
            "Tarea añadida";
    private static final String MENSAJE_CONFIRMACION_TAREA_ANYADIDA =
            "Tarea añadida correctamente.";
    private static final String TITULO_NO_HAY_TAREAS =
            "No hay tareas";
    private static final String MENSAJE_LISTA_TAREAS_VACIA =
            "La lista de tareas está vacía.";
    private static final String TITULO_TAREAS_NO_ENCONTRADAS =
            "No hay tareas";
    private static final String MENSAJE_TAREAS_NO_ENCONTRADAS =
            "No se han encontrado tareas para la fecha introducida.";
```

```
56      private static final String TITULO_ERROR_GUARDAR =
57              "Error guardar tareas";
58      private static final String MENSAJE_ERROR_GUARDAR =
59              "No se han guardado las tareas.";
60      private static final String TITULO_CONFIRMACION_GUARDAR =
61              "Tareas guardadas";
62      private static final String MENSAJE_CONFIRMACION_GUARDAR =
63              "Tareas guardadas correctamente.";
64      private static final String TITULO_ERROR_CONTRASENYA_INCORRECTA =
65              "Contraseña incorrecta";
66      private static final String MENSAJE_ERROR_CONTRASENYA_INCORRECTA =
67              "La contraseña introducida es incorrecta.";
68      private static final String TITULO_ERROR_USUARIO_EN_USO =
69              "Usuario en uso";
70      private static final String MENSAJE_ERROR_USUARIO_EN_USO =
71              "El usuario introducido ya está conectado a la máquina.";
72      private static final String TITULO_ERROR_USUARIO_INCORRECTO =
73              "Usuario incorrecto";
74      private static final String MENSAJE_ERROR_USUARIO_INCORRECTO =
75              "El usuario introducido no existe en la máquina.";
76
77      private static Mainframe emulador = null;
78      private static TasksJob tasks2 = null;
79      private static TasksJobWrapperVista vista;
80
81      /**
82       * Main del programa.
83       *
84       * @param args
85       * @throws IOException
86       * @throws InterruptedException
87       */
88      public static void main(String[] args)
89              throws IOException, InterruptedException {
90          new TasksJobWrapper();
91      }
92
93      /**
94       * Constructor de la clase.
95       *
96       * @throws IOException
97       * @throws InterruptedException
98       */
99      public TasksJobWrapper() throws IOException, InterruptedException {
100         vista = new TasksJobWrapperVista(this);
101         String[] datosInicioSesion = vista.obtenerDatosInicioSesion();
102
103         if (datosInicioSesion != null) {
104             vista.mostrarVentanaEspera();
105             falloConexion(datosInicioSesion);
106             vista.cerrarVentanaEspera();
107         } else {
108             System.exit(0);
109         }
110     }
```

```
111
112     /**
113      * Verifica la conexión con la máquina.
114      *
115      * @param datos
116      * @throws IOException
117      * @throws InterruptedException
118      */
119     private void falloConexion(String[] datos)
120             throws IOException, InterruptedException {
121         emulador = Mainframe.getInstance();
122         RESPUESTAS_INICIO_SESION resultadoConexion =
123                 emulador.conexion(datos[0], datos[1], datos[2]);
124
125         switch (resultadoConexion) {
126             case OK:
127                 tasks2 = new TasksJob(emulador);
128                 vista.crearElementosVentanaPrincipal();
129                 break;
130             case CONTRASENYA_INCORRECTA:
131                 vista.notificarMensajeError(TITULO_ERROR_CONTRASENYA_INCORRECTA,
132                         MENSAJE_ERROR_CONTRASENYA_INCORRECTA);
133                 System.exit(0);
134                 break;
135             case USUARIO_EN_USO:
136                 vista.notificarMensajeError(TITULO_ERROR_USUARIO_EN_USO,
137                         MENSAJE_ERROR_USUARIO_EN_USO);
138                 System.exit(0);
139                 break;
140             case USUARIO_INCORRECTO:
141                 vista.notificarMensajeError(TITULO_ERROR_USUARIO_INCORRECTO,
142                         MENSAJE_ERROR_USUARIO_INCORRECTO);
143                 System.exit(0);
144                 break;
145             case NOK:
146                 vista.notificarMensajeError(TITULO_ERROR,
147                         MENSAJE_ERROR_INESPERADO);
148                 System.exit(0);
149         }
150     }
151
152     /**
153      * Método sobreescrito para tratamiento de errores.
154      *
155      * @param evento
156      * @param obj
157      */
158     @Override
159     public void eventoProducido(Evento evento, Object obj) {
160         try {
161             switch (evento) {
162                 case NUEVO_FICHERO:
163                     nuevoFicheroTareas();
164                     break;
165
```

```
166                 case ANYADIR_TAREA:
167                     anyadirTarea(obj);
168                     break;
169 
170                 case ELIMINAR_TAREA:
171                     String idTarea = (String) obj;
172                     eliminarTarea(idTarea);
173                     break;
174 
175                 case LISTAR_TAREAS:
176                     listarTareas();
177                     break;
178 
179                 case BUSCAR_TAREA:
180                     String fecha = (String) obj;
181                     buscarTarea(fecha);
182                     break;
183 
184                 case GUARDAR_TAREAS:
185                     guardarTareas();
186                     break;
187 
188                 case SALIR:
189                     String guardarCambios = (String) obj;
190                     salir(guardarCambios);
191 
192             }
193         } catch (IOException e) {
194             e.printStackTrace();
195         } catch (InterruptedException e) {
196             e.printStackTrace();
197         }
198     }
199 
200     /**
201      * Crear un nuevo fichero de tareas en la aplicación legada.
202      *
203      * @throws IOException
204      * @throws InterruptedException
205      */
206     public void nuevoFicheroTareas() throws IOException, InterruptedException {
207         vista.mostrarVentanaEspera();
208         if (!tasks2.nuevoFicheroTareas()) {
209             vista.notificarMensajeError(TITULO_ERROR_FICHERO_TAREAS,
210                     MENSAJE_ERROR_FICHERO_TAREAS);
211         } else {
212             vista.notificarMensajeConfirmacion(
213                     TITULO_CONFIRMACION_FICHERO_TAREAS,
214                     MENSAJE_CONFIRMACION_FICHERO_TAREAS);
215         }
216     }
217 
218     /**
219      * Añade una nueva tarea en la aplicación legada.
220      *
```

```
221      * @param obj
222      * @throws IOException
223      * @throws InterruptedException
224      */
225     public void anyadirTarea(Object obj)
226             throws IOException, InterruptedException {
227         vista.mostrarVentanaEspera();
228         Tupla<Tupla, Tupla> tuplaTarea = (Tupla<Tupla, Tupla>) obj;
229         Tupla<String, String> tuplaIdNombre = tuplaTarea.a;
230         Tupla<String, String> tuplaDescFecha = tuplaTarea.b;
231 
232         TasksAPI.CODIGO_ERROR codigoAnyadir =
233                 tasks2.anyadirTarea(tuplaIdNombre.a, tuplaIdNombre.b,
234                         tuplaDescFecha.a, tuplaDescFecha.b);
235         switch (codigoAnyadir) {
236             case NOK:
237                 vista.notificarMensajeError(TITULO_ERROR,
238                         MENSAJE_ERROR_INESPERADO);
239                 break;
240             case IDTAREA_INCORRECTO:
241                 vista.notificarMensajeError(TITULO_ERROR_ANYADIR_TAREA,
242                         MENSAJE_ERROR_ID_INCORRECTO);
243                 break;
244             case OK:
245                 vista.notificarMensajeConfirmacion(
246                         TITULO_CONFIRMACION_ANYADIR_TAREA,
247                         MENSAJE_CONFIRMACION_TAREA_ANYADIDA);
248                 break;
249         }
250     }
251 
252     /**
253      * Elimina una tarea de la aplicación legada.
254      *
255      * @param idTarea
256      * @throws IOException
257      * @throws InterruptedException
258      */
259     public void eliminarTarea(String idTarea)
260             throws IOException, InterruptedException {
261         vista.mostrarVentanaEspera();
262         TasksAPI.CODIGO_ERROR codigoEliminar = tasks2.eliminarTarea(idTarea);
263 
264         switch (codigoEliminar) {
265             case NOK:
266                 vista.notificarMensajeError(TITULO_ERROR,
267                         MENSAJE_ERROR_INESPERADO);
268                 break;
269             case IDTAREA_INCORRECTO:
270                 vista.notificarMensajeError(TITULO_ERROR_ELIMINAR_TAREA,
271                         MENSAJE_ERROR_ELIMINAR_TAREA);
272                 break;
273             case OK:
274                 vista.notificarMensajeConfirmacion(
275                         TITULO_CONFIRMACION_ELIMINAR_TAREA,
```

```
276                         MENSAJE_CONFIRMACION_ELIMINAR_TAREA);
277                 break;
278         }
279     }
280 
281     /**
282      * Lista las tareas de la aplicación legada.
283      *
284      * @throws IOException
285      * @throws InterruptedException
286      */
287     public void listarTareas() throws IOException, InterruptedException {
288         vista.mostrarVentanaEspera();
289         List<Tarea> tareasListar = tasks2.listarTareas();
290         String cadenaTareasListar = "";
291         if (tareasListar.size() == 0) {
292             vista.notificarMensajeError(TITULO_NO_HAY_TAREAS,
293                     MENSAJE_LISTA_TAREAS_VACIA);
294         } else {
295             for (Tarea tarea : tareasListar) {
296                 cadenaTareasListar += tarea.toString();
297             }
298             vista.mostrarTareas(cadenaTareasListar);
299             vista.cerrarVentanaEspera();
300         }
301     }
302 
303     /**
304      * Busca tareas en la aplicación legada.
305      *
306      * @param fecha
307      * @throws IOException
308      * @throws InterruptedException
309      */
310     public void buscarTarea(String fecha)
311             throws IOException, InterruptedException {
312         vista.mostrarVentanaEspera();
313         String cadenaTareasBuscar = "";
314         List<Tarea> tareasBuscar = tasks2.buscarTareas(fecha);
315         if (tareasBuscar.size() == 0) {
316             vista.notificarMensajeError(TITULO_TAREAS_NO_ENCONTRADAS,
317                     MENSAJE_TAREAS_NO_ENCONTRADAS);
318         } else {
319             for (Tarea tarea : tareasBuscar) {
320                 cadenaTareasBuscar += tarea.toString();
321             }
322             vista.mostrarTareas(cadenaTareasBuscar);
323             vista.cerrarVentanaEspera();
324         }
325     }
326 
327     /**
328      * Guarda las tareas de la aplicación legada.
329      *
330      * @throws IOException
```

```
331      * @throws InterruptedException
332      */
333     public void guardarTareas() throws IOException, InterruptedException {
334         vista.mostrarVentanaEspera();
335         if (!tasks2.guardarTareas()) {
336             vista.notificarMensajeError(TITULO_ERROR_GUARDAR,
337                     MENSAJE_ERROR_GUARDAR);
338         } else {
339             vista.notificarMensajeConfirmacion(TITULO_CONFIRMACION_GUARDAR,
340                     MENSAJE_CONFIRMACION_GUARDAR);
341         }
342     }
343 
344     /**
345      * Desconexión de la máquina y cierre de aplicación.
346      *
347      * @param respuesta
348      * @throws IOException
349      * @throws InterruptedException
350      */
351     public void salir(String respuesta)
352             throws IOException, InterruptedException {
353         vista.mostrarVentanaEspera();
354         tasks2.salir(respuesta);
355         emulador.logout();
356         System.exit(0);
357     }
358 }
359 
```