```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.AplicacionVista;

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

public class Aplicacion 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.";
    private static final String TITULO_ERROR_GUARDAR = "Error guardar tareas";
    private static final String MENSAJE_ERROR_GUARDAR = "No se han guardado las tareas.";
    private static final String TITULO_CONFIRMACION_GUARDAR = "Tareas guardadas";
    private static final String MENSAJE_CONFIRMACION_GUARDAR = "Tareas guardadas correctamente.";
    private static final String TITULO_ERROR_CONTRASENYA_INCORRECTA = "Contraseña incorrecta";
    private static final String MENSAJE_ERROR_CONTRASENYA_INCORRECTA = "La contraseña introducida es incorrecta.";
    private static final String TITULO_ERROR_USUARIO_EN_USO = "Usuario en uso";
    private static final String MENSAJE_ERROR_USUARIO_EN_USO = "El usuario introducido ya está conectado a la máquina.";
```

```
45      private static final String TITULO_ERROR_USUARIO_INCORRECTO = "Usuario incorrecto";
46      private static final String MENSAJE_ERROR_USUARIO_INCORRECTO = "El usuario
    introducido no existe en la máquina.";
47
48      private static Mainframe emulador = null;
49      private static TasksJob tasks2 = null;
50      private static AplicacionVista vista;
51
52      /**
53       * Main del programa.
54       *
55       * @param args
56       * @throws IOException
57       * @throws InterruptedException
58       */
59      public static void main(String[] args) throws IOException, InterruptedException {
60          new Aplicacion();
61      }
62
63      /**
64       * Constructor de la clase.
65       *
66       * @throws IOException
67       * @throws InterruptedException
68       */
69      public Aplicacion() throws IOException, InterruptedException {
70          vista = new AplicacionVista(this);
71          String[] datosInicioSesion = vista.obtenerDatosInicioSesion();
72
73          if (datosInicioSesion != null) {
74              falloConexion(datosInicioSesion);
75          } else {
76              System.exit(0);
77          }
78      }
79
80      /**
81       * Verifica la conexión con la máquina.
82       *
83       * @param datos
84       * @throws IOException
85       * @throws InterruptedException
86       */
87      private void falloConexion(String[] datos) throws IOException, InterruptedException
     {
88          emulador = Mainframe.getInstance();
89          RESPUESTAS_INICIO_SESION resultadoConexion = emulador.
90                  conexion(datos[0], datos[1], datos[2]);
91
92          switch (resultadoConexion) {
93              case OK:
94                  tasks2 = new TasksJob(emulador);
95                  vista.crearElementosVentanaPrincipal();
96                  break;
97              case CONTRASENYA_INCORRECTA:
98                  vista.notificarMensajeError(TITULO_ERROR_CONTRASENYA_INCORRECTA,
    MENSAJE_ERROR_CONTRASENYA_INCORRECTA);
99                  System.exit(0);
100                 break;
```

```
101             case USUARIO_EN_USO:
102                 vista.notificarMensajeError(TITULO_ERROR_USUARIO_EN_USO,
    MENSAJE_ERROR_USUARIO_EN_USO);
103                 System.exit(0);
104                 break;
105             case USUARIO_INCORRECTO:
106                 vista.notificarMensajeError(TITULO_ERROR_USUARIO_INCORRECTO,
    MENSAJE_ERROR_USUARIO_INCORRECTO);
107                 System.exit(0);
108                 break;
109             case NOK:
110                 vista.notificarMensajeError(TITULO_ERROR, MENSAJE_ERROR_INESPERADO);
111                 System.exit(0);
112         }
113     }
114 
115     /**
116      * Método sobreescrito para tratamiento de errores.
117      *
118      * @param evento
119      * @param obj
120      */
121     @Override
122     public void eventoProducido(Evento evento, Object obj) {
123         try {
124             switch (evento) {
125                 case NUEVO_FICHERO:
126                     nuevoFicheroTareas();
127                     break;
128 
129                 case ANYADIR_TAREA:
130                     anyadirTarea(obj);
131                     break;
132 
133                 case ELIMINAR_TAREA:
134                     String idTarea = (String) obj;
135                     eliminarTarea(idTarea);
136                     break;
137 
138                 case LISTAR_TAREAS:
139                     listarTareas();
140                     break;
141 
142                 case BUSCAR_TAREA:
143                     String fecha = (String) obj;
144                     buscarTarea(fecha);
145                     break;
146 
147                 case GUARDAR_TAREAS:
148                     guardarTareas();
149                     break;
150 
151                 case SALIR:
152                     String guardarCambios = (String) obj;
153                     salir(guardarCambios);
154 
155             }
156         } catch (IOException e) {
157             e.printStackTrace();
```

```
158         } catch (InterruptedException e) {
159             e.printStackTrace();
160         }
161     }
162 
163     /**
164      * Crear un nuevo fichero de tareas en la aplicación legada.
165      *
166      * @throws IOException
167      * @throws InterruptedException
168      */
169     public void nuevoFicheroTareas() throws IOException, InterruptedException {
170         if (!tasks2.nuevoFicheroTareas()) {
171             vista.notificarMensajeError(TITULO_ERROR_FICHERO_TAREAS,
    MENSAJE_ERROR_FICHERO_TAREAS);
172         } else {
173             vista.notificarMensajeConfirmacion(TITULO_CONFIRMACION_FICHERO_TAREAS,
174                     MENSAJE_CONFIRMACION_FICHERO_TAREAS);
175         }
176     }
177 
178     /**
179      * Añade una nueva tarea en la aplicación legada.
180      *
181      * @param obj
182      * @throws IOException
183      * @throws InterruptedException
184      */
185     public void anyadirTarea(Object obj) throws IOException, InterruptedException {
186         Tupla<Tupla, Tupla> tuplaTarea = (Tupla<Tupla, Tupla>) obj;
187         Tupla<String, String> tuplaIdNombre = tuplaTarea.a;
188         Tupla<String, String> tuplaDescFecha = tuplaTarea.b;
189 
190         TasksAPI.CODIGO_ERROR codigoAnyadir = tasks2.anyadirTarea(tuplaIdNombre.a,
    tuplaIdNombre.b,
191                 tuplaDescFecha.a, tuplaDescFecha.b);
192 
193         switch (codigoAnyadir) {
194             case NOK:
195                 vista.notificarMensajeError(TITULO_ERROR, MENSAJE_ERROR_INESPERADO);
196                 break;
197             case IDTAREA_INCORRECTO:
198                 vista.notificarMensajeError(TITULO_ERROR_ANYADIR_TAREA,
    MENSAJE_ERROR_ID_INCORRECTO);
199                 break;
200             case OK:
201                 vista.notificarMensajeConfirmacion(TITULO_CONFIRMACION_ANYADIR_TAREA,
202                         MENSAJE_CONFIRMACION_TAREA_ANYADIDA);
203                 break;
204         }
205     }
206 
207     /**
208      * Elimina una tarea de la aplicación legada.
209      *
210      * @param idTarea
211      * @throws IOException
212      * @throws InterruptedException
213      */
```

```
214     public void eliminarTarea(String idTarea) throws IOException, InterruptedException {
215         TasksAPI.CODIGO_ERROR codigoEliminar = tasks2.eliminarTarea(idTarea);
216 
217         switch (codigoEliminar) {
218             case NOK:
219                 vista.notificarMensajeError(TITULO_ERROR, MENSAJE_ERROR_INESPERADO);
220                 break;
221             case IDTAREA_INCORRECTO:
222                 vista.notificarMensajeError(TITULO_ERROR_ELIMINAR_TAREA,
    MENSAJE_ERROR_ELIMINAR_TAREA);
223                 break;
224             case OK:
225                 vista.notificarMensajeConfirmacion(TITULO_CONFIRMACION_ELIMINAR_TAREA,
226                         MENSAJE_CONFIRMACION_ELIMINAR_TAREA);
227                 break;
228         }
229     }
230 
231     /**
232      * Lista las tareas de la aplicación legada.
233      *
234      * @throws IOException
235      * @throws InterruptedException
236      */
237     public void listarTareas() throws IOException, InterruptedException {
238         List<Tarea> tareasListar = tasks2.listarTareas();
239         String cadenaTareasListar = "";
240         if (tareasListar.size() == 0) {
241             vista.notificarMensajeError(TITULO_NO_HAY_TAREAS, MENSAJE_LISTA_TAREAS_VACIA
    );
242         } else {
243             for (Tarea tarea : tareasListar) {
244                 cadenaTareasListar += tarea.toString();
245             }
246             vista.mostrarTareas(cadenaTareasListar);
247         }
248     }
249 
250     /**
251      * Busca tareas en la aplicación legada.
252      *
253      * @param fecha
254      * @throws IOException
255      * @throws InterruptedException
256      */
257     public void buscarTarea(String fecha) throws IOException, InterruptedException {
258         String cadenaTareasBuscar = "";
259         List<Tarea> tareasBuscar = tasks2.buscarTareas(fecha);
260         if (tareasBuscar.size() == 0) {
261             vista.notificarMensajeError(TITULO_TAREAS_NO_ENCONTRADAS,
    MENSAJE_TAREAS_NO_ENCONTRADAS);
262         } else {
263             for (Tarea tarea : tareasBuscar) {
264                 cadenaTareasBuscar += tarea.toString();
265             }
266             vista.mostrarTareas(cadenaTareasBuscar);
267 
268         }
269     }
```

```
270
271     /**
272      * Guarda las tareas de la aplicación legada.
273      *
274      * @throws IOException
275      * @throws InterruptedException
276      */
277     public void guardarTareas() throws IOException, InterruptedException {
278         if (!tasks2.guardarTareas()) {
279             vista.notificarMensajeError(TITULO_ERROR_GUARDAR, MENSAJE_ERROR_GUARDAR);
280         } else {
281             vista.notificarMensajeConfirmacion(TITULO_CONFIRMACION_GUARDAR,
    MENSAJE_CONFIRMACION_GUARDAR);
282         }
283     }
284
285     /**
286      * Desconexión de la máquina y cierre de aplicación.
287      *
288      * @param respuesta
289      * @throws IOException
290      * @throws InterruptedException
291      */
292     public void salir(String respuesta) throws IOException, InterruptedException {
293         tasks2.salir(respuesta);
294         emulador.logout();
295         System.exit(0);
296     }
297 }
298
```