```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.";
```

```java
    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.";
    private static final String TITULO_ERROR_USUARIO_INCORRECTO =
            "Usuario incorrecto";
    private static final String MENSAJE_ERROR_USUARIO_INCORRECTO =
            "El usuario introducido no existe en la máquina.";

    private static Mainframe emulador = null;
    private static TasksJob tasks2 = null;
    private static AplicacionVista vista;

    /**
     * Main del programa.
     *
     * @param args
     * @throws IOException
     * @throws InterruptedException
     */
    public static void main(String[] args)
            throws IOException, InterruptedException {
        new Aplicacion();
    }

    /**
     * Constructor de la clase.
     *
     * @throws IOException
     * @throws InterruptedException
     */
    public Aplicacion() throws IOException, InterruptedException {
        vista = new AplicacionVista(this);
        String[] datosInicioSesion = vista.obtenerDatosInicioSesion();

        if (datosInicioSesion != null) {
            falloConexion(datosInicioSesion);
        } else {
            System.exit(0);
        }
    }

    /**
```

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

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

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

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

```
331             vista.notificarMensajeConfirmacion(TITULO_CONFIRMACION_GUARDAR,
332                     MENSAJE_CONFIRMACION_GUARDAR);
333         }
334     }
335 
336     /**
337      * Desconexión de la máquina y cierre de aplicación.
338      *
339      * @param respuesta
340      * @throws IOException
341      * @throws InterruptedException
342      */
343     public void salir(String respuesta)
344             throws IOException, InterruptedException {
345         tasks2.salir(respuesta);
346         emulador.logout();
347         System.exit(0);
348     }
349 }
350 
```