```java
/**
 * TasksJobWrapperVista.java
 * <p>
 * Clase que representa la vista de la aplicación implementada.
 * <p>
 * Radu Constantin Robu y Alberto Pérez
 */
package P3.Vista;

import javax.swing.*;
import javax.swing.border.Border;
import javax.swing.border.EmptyBorder;
import javax.swing.border.EtchedBorder;
import javax.swing.text.SimpleAttributeSet;
import javax.swing.text.StyleConstants;
import javax.swing.text.StyledDocument;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;

import P3.Control.OyenteVista;
import P3.Modelo.Tupla;

public class TasksJobWrapperVista implements ActionListener,
        PropertyChangeListener {
    private static final int ANCHO = 600;
    private static final int ALTO = 800;
    private static final int ANCHO_PANEL_BOTONES = 200;
    private static final int ALTO_PANEL_BOTONES = 200;
    private static final int ANCHO_PANEL_TAREAS = 400;
    private static final int ALTO_PANEL_TAREAS = 200;
    private static final int INDENT_TAREAS = 70;
    private static final int TAMANYO_NOMBRE_TAREA = 16;
    private static final int TAMANYO_CAMPOS_INICIO_SESION = 16;
    private static final int TAMANYO_DESCRIPCION_TAREA = 32;
    private static final int TAMANYO_CAMPOS_TEXTO = 10;
    private static final int DELAY_TIMER = 1000;

    private static final String FORMATO_FECHA = "^[0-9]{2} [0-9]{2} [0-9]{4}$";
    private static final String PATRON_CADENA_DIGITOS = "^[0-9]+$";
    private static final String RESPUESTA_SI = "y";
    private static final String RESPUESTA_NO = "n";
    private static final String MENSAJE_ESPERA = "Procesando. Por favor " +
            "espere" +
            ".\nTiempo transcurrido: ";
    private static final String TITULO_VENTANA_ESPERA = "Procesando";

    private OyenteVista oyenteVista;
    private static final String NUEVO_FICHERO = "Nuevo fichero";
    private static final String ANYADIR_TAREA = "Añadir tarea";
    private static final String ELIMINAR_TAREA = "Eliminar tarea";
    private static final String BUSCAR_TAREA = "Buscar tareas";
    private static final String LISTAR_TAREAS = "Listar tareas";
```

```
56      private static final String GUARDAR_TAREAS = "Guardar tareas";
57      private static final String SALIR = "Salir";
58      private static final String TASKSJOB_WRAPPER = "Wrapper Tasks2.job";
59      private static final String ACCION_NUEVO_FICHERO = "NUEVO_FICHERO";
60      private static final String ACCION_ANYADIR_TAREA = "ANYADIR_TAREA";
61      private static final String ACCION_ELIMINAR_TAREA = "ELIMINAR_TAREA";
62      private static final String ACCION_BUSCAR_TAREAS = "BUCAR_TAREAS";
63      private static final String ACCION_LISTAR_TAREAS = "LISTAR_TAREAS";
64      private static final String ACCION_GUARDAR_TAREAS = "GUARDAR_TAREAS";
65      private static final String ACCION_SALIR = "SALIR";
66      private static final String TITULO_GUARDAR_SALIR = "Guardar y salir";
67      private static final String MENSAJE_GUARDAR_CAMBIOS =
68              "¿Desea guardar los cambios?";
69
70      private static final String TITULO_VENTANA_INICIAR_SESION =
71              "Inicio sesión";
72      private static final String[] CAMPOS_INICIAR_SESION =
73              {"Host", "Usuario", "Contraseña"};
74      private static final String[] OPCIONES_INICIAR_SESION =
75              {"Iniciar sesión", "Cancelar"};
76
77      private static final String TITULO_VENTANA_ANYADIR_TAREA =
78              "Añadir una tarea";
79      private static final String[] CAMPOS_ANYADIR_TAREA =
80              {"Id", "Nombre", "Descripción", "Fecha"};
81      private static final String[] OPCIONES_ANYADIR_TAREA =
82              {"Añadir", "Cancelar"};
83
84      private static final String TITULO_VENTANA_BUSCAR_TAREAS = "Buscar tareas";
85      private static final String[] CAMPOS_BUSCAR_TAREAS = {"Fecha"};
86      private static final String[] OPCIONES_BUSCAR_TAREAS =
87              {"Buscar", "Cancelar"};
88
89      private static final String TITULO_VENTANA_ELIMINAR_TAREA =
90              "Eliminar tarea";
91      private static final String[] CAMPOS_ELIMINAR_TAREA = {"Id"};
92      private static final String[] OPCIONES_ELIMINAR_TAREA =
93              {"Eliminar", "Cancelar"};
94
95      private static final String ETIQUETA_VENTANA_ERROR_FECHA =
96              "Fecha incorrecta";
97      private static final String MENSAJE_FECHA_INCORRECTA =
98              "El formato de la fecha introducida no es correcto.\n" +
99                      "Formato aceptado:\n" + "DD MM AAAA\n";
100
101     private static final String ETIQUETA_VENTANA_ID_INCORRECTO =
102             "ID incorrecto";
103     private static final String MENSAJE_ID_INCORRECTO =
104             "El id de una tarea debe contener únicamente números.";
105
106     private static final String ETIQUETA_VENTANA_NOMBRE_INCORRECTO =
107             "Nombre incorrecto";
108     private static final String MENSAJE_NOMBRE_INCORRECTO =
109             "El nombre de la tarea debe tener menos de " +
110                     TAMANYO_NOMBRE_TAREA + " caracteres.";
```

```
111
112     private static final String ETIQUETA_VENTANA_DESCRIPCION_INCORRECTA =
113             "Descripción incorrecta";
114     private static final String MENSAJE_DESCRIPCION_INCORRECTA =
115             "La descripción de la tarea debe tener menos de " +
116                     TAMANYO_DESCRIPCION_TAREA + " caracteres.";
117
118     private String[] datosInicioSesion;
119     private int tiempoTranscurrido = 0;
120
121     private JButton botonNuevoFichero;
122     private JButton botonAnyadirTarea;
123     private JButton botonEliminarTarea;
124     private JButton botonBuscarTarea;
125     private JButton botonListarTareas;
126     private JButton botonGuardarTareas;
127     private JButton botonSalir;
128     private JPanel panelTareas;
129     private JTextPane areaTextoTareas;
130     private JOptionPane ventanaEspera;
131     private JDialog dialogoEspera;
132     private Timer temporizador;
133
134     private enum CodigoRespuesta {
135         OK, ERROR_ID_TAREA, ERROR_LONG_NOMBRE, DATOS_VACIOS,
136         ERROR_FECHA_INCORRECTA, ERROR_LONG_DESCRIPCION
137     }
138
139     /**
140      * Constructor de la clase.
141      *
142      * @param oyenteVista
143      */
144     public TasksJobWrapperVista(OyenteVista oyenteVista) {
145         datosInicioSesion = iniciarSesion();
146         this.oyenteVista = oyenteVista;
147     }
148
149     /**
150      * Devuelve los datos de inicio de sesión.
151      *
152      * @return
153      */
154     public String[] obtenerDatosInicioSesion() {
155         return datosInicioSesion;
156     }
157
158     /**
159      * Crea la ventana principal de la interfaz y sus elementos.
160      */
161     public void crearElementosVentanaPrincipal() {
162         JFrame ventanaPrincipal = new JFrame(TASKSJOB_WRAPPER);
163         ventanaPrincipal.setSize(ANCHO, ALTO);
164         ventanaPrincipal.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
165         ventanaPrincipal.setLocationRelativeTo(null);
```

```
166         ventanaPrincipal.setLayout(new BorderLayout());
167         ventanaPrincipal.add(crearPanelBotones());
168         ventanaPrincipal.add(crearPanelMostrarTareas(), BorderLayout.EAST);
169         ventanaPrincipal.setVisible(true);
170 
171     }
172 
173     /**
174      * Crea el panel de tareas de la ventana principal.
175      *
176      * @return
177      */
178     public JPanel crearPanelMostrarTareas() {
179         panelTareas = new JPanel(new BorderLayout());
180         Border bordePanelTareas =
181                 BorderFactory.createEtchedBorder(EtchedBorder.LOWERED);
182         panelTareas.setBorder(bordePanelTareas);
183         panelTareas.setPreferredSize(new Dimension(ANCHO_PANEL_TAREAS,
184                 ALTO_PANEL_TAREAS));
185 
186         JScrollPane panelDeslizable = new JScrollPane();
187         areaTextoTareas = new JTextPane();
188         areaTextoTareas.setEditable(false);
189         areaTextoTareas.setAlignmentX(70);
190         panelDeslizable.setViewportView(areaTextoTareas);
191         panelTareas.add(panelDeslizable);
192 
193         StyledDocument doc = areaTextoTareas.getStyledDocument();
194 
195         SimpleAttributeSet center = new SimpleAttributeSet();
196         StyleConstants.setLeftIndent(center, INDENT_TAREAS);
197         doc.setParagraphAttributes(0, doc.getLength(), center, false);
198 
199         return panelTareas;
200     }
201 
202 
203     /**
204      * Crea el panel de botones de la ventana principal.
205      *
206      * @return
207      */
208     private JPanel crearPanelBotones() {
209         crearBotones();
210         JPanel panelBotones = new JPanel(new BorderLayout());
211         panelBotones.setBorder(new EmptyBorder(2, 3, 2, 3));
212 
213         JPanel layoutBotones = new JPanel(new GridBagLayout());
214         layoutBotones.setBorder(new EmptyBorder(5, 5, 5, 5));
215         Border bordePanelBotones =
216                 BorderFactory.createEtchedBorder(EtchedBorder.LOWERED);
217         layoutBotones.setBorder(bordePanelBotones);
218 
219         JPanel contenedorBotones =
220                 new JPanel(new GridLayout(10, 1, 10, 20));
```

```
221
222         contenedorBotones.add(botonNuevoFichero);
223         contenedorBotones.add(botonAnyadirTarea);
224         contenedorBotones.add(botonEliminarTarea);
225         contenedorBotones.add(botonBuscarTarea);
226         contenedorBotones.add(botonListarTareas);
227         contenedorBotones.add(botonGuardarTareas);
228         contenedorBotones.add(botonSalir);
229         layoutBotones.add(contenedorBotones);
230         layoutBotones.setPreferredSize(new Dimension(ANCHO_PANEL_BOTONES,
231                 ALTO_PANEL_BOTONES));
232         panelBotones.setPreferredSize(new Dimension(ANCHO_PANEL_BOTONES,
233                 ALTO_PANEL_BOTONES));
234         contenedorBotones.setPreferredSize(new Dimension(ANCHO_PANEL_BOTONES,
235                 ALTO_PANEL_BOTONES));
236         panelBotones.add(layoutBotones, BorderLayout.WEST);
237
238         return panelBotones;
239     }
240
241     /**
242      * Crea los botones del panel de botones.
243      */
244     private void crearBotones() {
245         botonNuevoFichero = new JButton(NUEVO_FICHERO);
246         botonNuevoFichero.addActionListener(this);
247         botonNuevoFichero.setActionCommand(ACCION_NUEVO_FICHERO);
248
249         botonAnyadirTarea = new JButton(ANYADIR_TAREA);
250         botonAnyadirTarea.addActionListener(this);
251         botonAnyadirTarea.setActionCommand(ACCION_ANYADIR_TAREA);
252
253         botonEliminarTarea = new JButton(ELIMINAR_TAREA);
254         botonEliminarTarea.addActionListener(this);
255         botonEliminarTarea.setActionCommand(ACCION_ELIMINAR_TAREA);
256
257         botonBuscarTarea = new JButton(BUSCAR_TAREA);
258         botonBuscarTarea.addActionListener(this);
259         botonBuscarTarea.setActionCommand(ACCION_BUSCAR_TAREAS);
260
261         botonListarTareas = new JButton(LISTAR_TAREAS);
262         botonListarTareas.addActionListener(this);
263         botonListarTareas.setActionCommand(ACCION_LISTAR_TAREAS);
264
265         botonGuardarTareas = new JButton(GUARDAR_TAREAS);
266         botonGuardarTareas.addActionListener(this);
267         botonGuardarTareas.setActionCommand(ACCION_GUARDAR_TAREAS);
268
269         botonSalir = new JButton(SALIR);
270         botonSalir.addActionListener(this);
271         botonSalir.setActionCommand(ACCION_SALIR);
272     }
273
274     /**
275      * Método de tratamiento de eventos.
```

```
276      *
277      * @param e
278      */
279     @Override
280     public void actionPerformed(ActionEvent e) {
281         switch (e.getActionCommand()) {
282             case ACCION_NUEVO_FICHERO:
283                 oyenteVista.eventoProducido(OyenteVista.Evento.NUEVO_FICHERO,
284                         null);
285                 break;
286 
287             case ACCION_ANYADIR_TAREA:
288                 String[] datosAnyadirTarea = anyadirTarea();
289                 if (datosAnyadirTarea != null) {
290                     Tupla<Tupla, Tupla> tupla = new Tupla(
291                         new Tupla(datosAnyadirTarea[0], datosAnyadirTarea[1]),
292                         new Tupla(datosAnyadirTarea[2], datosAnyadirTarea[3]));
293                     oyenteVista.eventoProducido(
294                             OyenteVista.Evento.ANYADIR_TAREA, tupla);
295                 }
296                 break;
297 
298             case ACCION_BUSCAR_TAREAS:
299                 String[] datosBuscarTareas = buscarTareas();
300                 if (datosBuscarTareas != null) {
301                     oyenteVista.eventoProducido(OyenteVista.Evento.BUSCAR_TAREA,
302                             datosBuscarTareas[0]);
303                 }
304                 break;
305 
306             case ACCION_SALIR:
307                 int resp = JOptionPane.showConfirmDialog(null,
308                         MENSAJE_GUARDAR_CAMBIOS,
309                         TITULO_GUARDAR_SALIR, JOptionPane.YES_NO_OPTION);
310                 if (resp == JOptionPane.YES_OPTION) {
311                     oyenteVista.eventoProducido(OyenteVista.Evento.SALIR,
312                             RESPUESTA_SI);
313                 } else if (resp == JOptionPane.NO_OPTION) {
314                     oyenteVista.eventoProducido(OyenteVista.Evento.SALIR,
315                             RESPUESTA_NO);
316                 }
317                 break;
318 
319             case ACCION_ELIMINAR_TAREA:
320                 String[] datosEliminarTarea = obtenerDatosEliminarTarea();
321                 if (datosEliminarTarea != null) {
322                     oyenteVista.eventoProducido(
323                             OyenteVista.Evento.ELIMINAR_TAREA,
324                             datosEliminarTarea[0]);
325                 }
326                 break;
327 
328             case ACCION_LISTAR_TAREAS:
329                 oyenteVista.eventoProducido(OyenteVista.Evento.LISTAR_TAREAS,
330                         null);
```

```
331                 break;
332
333             case ACCION_GUARDAR_TAREAS:
334                 oyenteVista.eventoProducido(OyenteVista.Evento.GUARDAR_TAREAS,
335                         null);
336                 break;
337         }
338     }
339
340     /**
341      * Método para mostrar tareas en un mainframe.
342      *
343      * @param tareas
344      */
345     public void mostrarTareas(String tareas) {
346         areaTextoTareas.setText("");
347         areaTextoTareas.setText(tareas);
348     }
349
350     /**
351      * Método para iniciar sesión en un mainframe.
352      *
353      * @return
354      */
355     private String[] iniciarSesion() {
356         ComplexDialoguePanel ventanaIniciarSesion =
357                 new ComplexDialoguePanel(TITULO_VENTANA_INICIAR_SESION,
358                         CAMPOS_INICIAR_SESION, TAMANYO_CAMPOS_INICIO_SESION);
359         String[] datosInicioSesion =
360                 ventanaIniciarSesion.obtenerTextoCampos(
361                         OPCIONES_INICIAR_SESION);
362
363         return datosInicioSesion;
364     }
365
366     /**
367      * Método para añadir una tarea en el mainframe.
368      *
369      * @return
370      */
371     private String[] anyadirTarea() {
372         String[] datosAnyadirTarea = null;
373
374         ComplexDialoguePanel ventanaAnyadirTarea = new ComplexDialoguePanel(
375                 TITULO_VENTANA_ANYADIR_TAREA, CAMPOS_ANYADIR_TAREA,
376                 TAMANYO_DESCRIPCION_TAREA);
377         datosAnyadirTarea = ventanaAnyadirTarea.obtenerTextoCampos(
378                 OPCIONES_ANYADIR_TAREA);
379
380         CodigoRespuesta codigo = verificarDatosTarea(datosAnyadirTarea);
381
382         switch (codigo) {
383             case OK:
384                 return datosAnyadirTarea;
385
```

```
386             case ERROR_FECHA_INCORRECTA:
387                 JOptionPane.showMessageDialog(new JFrame(),
388                         MENSAJE_FECHA_INCORRECTA, ETIQUETA_VENTANA_ERROR_FECHA,
389                         JOptionPane.ERROR_MESSAGE);
390                 break;
391 
392             case ERROR_ID_TAREA:
393                 JOptionPane.showMessageDialog(new JFrame(),
394                         MENSAJE_ID_INCORRECTO, ETIQUETA_VENTANA_ID_INCORRECTO,
395                         JOptionPane.ERROR_MESSAGE);
396                 break;
397 
398             case ERROR_LONG_NOMBRE:
399                 JOptionPane.showMessageDialog(new JFrame(),
400                         MENSAJE_NOMBRE_INCORRECTO,
401                         ETIQUETA_VENTANA_NOMBRE_INCORRECTO,
402                         JOptionPane.ERROR_MESSAGE);
403                 break;
404 
405             case ERROR_LONG_DESCRIPCION:
406                 JOptionPane.showMessageDialog(new JFrame(),
407                         MENSAJE_DESCRIPCION_INCORRECTA,
408                         ETIQUETA_VENTANA_DESCRIPCION_INCORRECTA,
409                         JOptionPane.ERROR_MESSAGE);
410                 break;
411         }
412         return null;
413     }
414 
415     /**
416      * Comprueba los datos que el usuario ha introducido.
417      *
418      * @param datosTarea
419      * @return
420      */
421     private CodigoRespuesta verificarDatosTarea(String[] datosTarea) {
422         if (datosTarea != null) {
423             if (datosTarea[0].matches(PATRON_CADENA_DIGITOS)) {
424                 if ((datosTarea[1].length() > 0) &&
425                         (datosTarea[1].length() <= TAMANYO_NOMBRE_TAREA)) {
426                     if ((datosTarea[2].length() > 0) &&
427                             (datosTarea[2].length() <=
428                                     TAMANYO_DESCRIPCION_TAREA)) {
429                         if (datosTarea[3].matches(FORMATO_FECHA)) {
430                             return CodigoRespuesta.OK;
431                         } else {
432                             return CodigoRespuesta.ERROR_FECHA_INCORRECTA;
433                         }
434                     }
435                     return CodigoRespuesta.ERROR_LONG_DESCRIPCION;
436                 } else {
437                     return CodigoRespuesta.ERROR_LONG_NOMBRE;
438                 }
439             } else {
440                 return CodigoRespuesta.ERROR_ID_TAREA;
```

```
441             }
442         } else {
443             return CodigoRespuesta.DATOS_VACIOS;
444         }
445     }
446 
447     /**
448      * Método para buscar tareas en el mainframe por fecha.
449      *
450      * @return
451      */
452     private String[] buscarTareas() {
453         ComplexDialoguePanel ventanaBuscarTareas =
454                 new ComplexDialoguePanel(TITULO_VENTANA_BUSCAR_TAREAS,
455                         CAMPOS_BUSCAR_TAREAS, TAMANYO_CAMPOS_TEXTO);
456         String[] datosBuscarTareas =
457                 ventanaBuscarTareas.obtenerTextoCampos(OPCIONES_BUSCAR_TAREAS);
458 
459         return datosBuscarTareas;
460     }
461 
462     /**
463      * Método para eliminar una tarea.
464      *
465      * @return
466      */
467     private String[] obtenerDatosEliminarTarea() {
468         ComplexDialoguePanel ventanaEliminarTareas =
469                 new ComplexDialoguePanel(TITULO_VENTANA_ELIMINAR_TAREA,
470                         CAMPOS_ELIMINAR_TAREA, TAMANYO_CAMPOS_TEXTO);
471         String[] datoseliminarTareas =
472                 ventanaEliminarTareas.obtenerTextoCampos(
473                         OPCIONES_ELIMINAR_TAREA);
474 
475         return datoseliminarTareas;
476     }
477 
478     /**
479      * Método para mostrar una ventana de espera al usuario.
480      */
481     public void mostrarVentanaEspera() {
482         tiempoTranscurrido = 0;
483         ventanaEspera = new JOptionPane(
484                 MENSAJE_ESPERA + tiempoTranscurrido + "s",
485                         JOptionPane.INFORMATION_MESSAGE,
486                         JOptionPane.DEFAULT_OPTION, null, new Object[]{},
487                         null);
488         ventanaEspera.revalidate();
489         dialogoEspera = new JDialog();
490         dialogoEspera.setModal(false);
491         dialogoEspera.setLocationRelativeTo(null);
492         dialogoEspera.setTitle(TITULO_VENTANA_ESPERA);
493         dialogoEspera.setContentPane(ventanaEspera);
494         dialogoEspera.setDefaultCloseOperation(JDialog.DO_NOTHING_ON_CLOSE);
495         dialogoEspera.pack();
```

```
496
497         temporizador = new Timer(DELAY_TIMER, new AbstractAction() {
498             @Override
499             public void actionPerformed(ActionEvent e) {
500                 tiempoTranscurrido++;
501                 ventanaEspera
502                         .setMessage(MENSAJE_ESPERA + tiempoTranscurrido + "s");
503             }
504         });
505         temporizador.start();
506         dialogoEspera.setVisible(true);
507     }
508
509     /**
510      * Método para destruir la ventana de espera.
511      */
512     public void cerrarVentanaEspera() {
513         ventanaEspera.setVisible(false);
514         temporizador.stop();
515         dialogoEspera.dispose();
516     }
517
518     /**
519      * Muestra un diálogo con el error que se ha producido.
520      *
521      * @param titulo
522      * @param mensaje
523      */
524     public void notificarMensajeError(String titulo, String mensaje) {
525         cerrarVentanaEspera();
526         JOptionPane.showMessageDialog(new JFrame(), mensaje, titulo,
527                 JOptionPane.ERROR_MESSAGE);
528     }
529
530     /**
531      * Muestra un diálogo con la confirmación de la tarea.
532      *
533      * @param titulo
534      * @param mensaje
535      */
536     public void notificarMensajeConfirmacion(String titulo, String mensaje) {
537         cerrarVentanaEspera();
538         JOptionPane.showMessageDialog(new JFrame(), mensaje, titulo,
539                 JOptionPane.INFORMATION_MESSAGE);
540     }
541
542     /**
543      * Método de cambios de propiedades de los elementos de la ventana.
544      *
545      * @param evt
546      */
547     @Override
548     public void propertyChange(PropertyChangeEvent evt) {
549     }
550 }
```