Universidad de Santiago de Chile
Facultad de Ingeniería
Departamento de Ingeniería Informática

Paradigmas de programación

Laboratorio 4

Paradigma dirigido por eventos

Christofer Rodríguez

Profesor: Víctor Flores Sánchez

Santiago – Chile
2-2020

Tabla de contenidos

Introducción

Stack Overflow es un famoso foro de preguntas y respuestas relacionadas, principalmente con la programación e informática; este foro cuenta con diferentes funcionalidades, tales como: preguntas, responder preguntas, votar, entre otras. En base a esto, es necesario hacer una simulación de un sistema de foro, más una interfaz gráfica para interactuar con este, para esto se tomará como referencia las funcionalidades de Stack Overflow y se programará en el lenguaje de programación Java, esto bajo el paradigma dirigido por eventos. En este documento primero se contextualizará con respecto al paradigma utilizado, se analizará el problema en cuestión, luego se explicará la solución creada para resolver este problema, para finalmente con los resultados obtenidos evaluar el grado de alcance de cada requerimiento.

Descripción del problema

Se necesita crear una interfaz gráfica para la simulación de un foro de preguntas y respuestas basado en Stack Overflow, esto debe estar diseñado bajo el paradigma dirigido por eventos y se seleccionó el lenguaje de programación Java para su desarrollo. Esta interfaz debe poseer las características principales de Stack Overflow y permitir al usuario interactuar de manera fácil con este foro. Específicamente para esto se necesita que el usuario sea capaz de registrarse, iniciar sesión, realizar preguntas, responder preguntas, crear nuevas etiquetas para las preguntas, ofrecer recompensas a quien responda una pregunta, aceptar respuestas como las mejores a una pregunta, votar a favor o en contra de una pregunta o respuesta y una manera sencilla de poder visualizar el contenido de este foro.

Descripción del paradigma

El paradigma dirigido por eventos es principalmente utilizado para el desarrollo de interfaces gráficas, en este la cual la ejecución del del código no es totalmente lineal, sino que está determinada por los sucesos que ocurren en el sistema.

Utilizando este estilo de programación es el usuario o el "ente" que este interactuando con el programa quien define el flujo de ejecución del programa y es el trabajo del programador definir qué y como se manejaran los eventos surgidos de las acciones del usuario. Esto ya que una vez se ejecuta el main del código, este inicializa el contenido y luego entra a un loop de espera, se espera que suceda un evento y cuando este ocurra, se ejecutará cierta acción.

Un evento es una acción reconocida por un objeto, los diferentes objetos tienen distintos eventos que puede reconocer, tales como cambio de estado, click del mouse, presionar una tecla, escribir en él, etc.

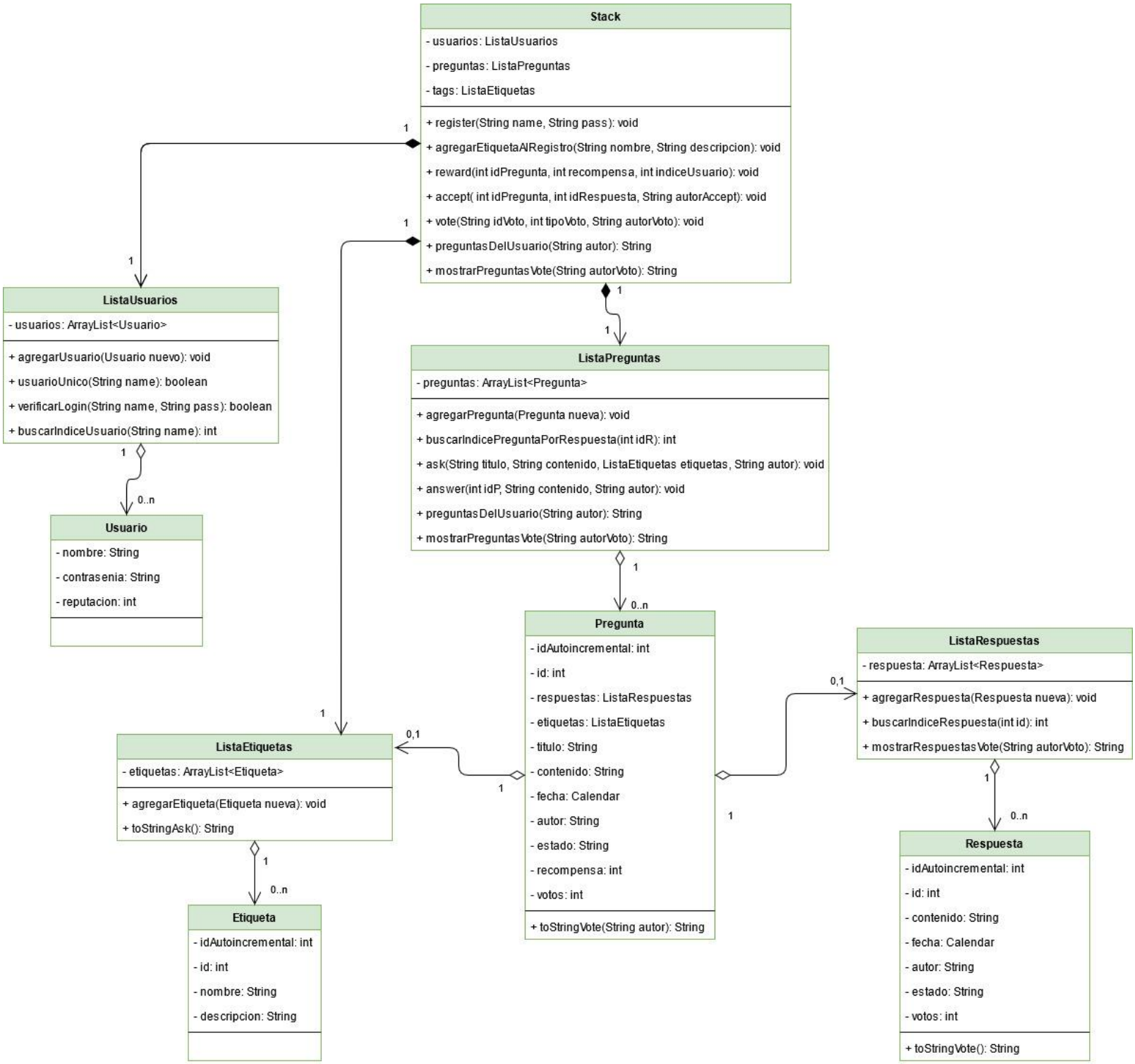
Para poder manejar estos eventos, en Java se hace uso de los **listeners**, estos esperan que ocurra cierto evento y cuando este ocurre, ejecuta una serie de acciones definidas por el programador. Con estos elementos es posible configurar que ocurre en cada ocasión que el usuario interactúe con algún componente de la interfaz gráfica.

Análisis del problema

Ya que bajo el paradigma dirigido por eventos partes de este código no se ejecutan de manera lineal, sino que se ejecutan cuando cierto evento ocurre, será necesario programar una manera de captar dichos eventos y cómo manejarlos.

Si analizamos los requerimientos para este problema, podemos notar que será necesario crear una clase para interactuar de una forma gráfica con cada una de las funcionalidades del foro, se deberá comprobar en cada una de estas ventanas que el usuario ingrese o interactúe correctamente con el programa, una manera de retroalimentar al usuario cuando se realiza un acción correctamente o cuando existe un error, de alguna manera cada ventana debe tener acceso a la información del foro y ser capaz de actualizar su información, además como anteriormente de nombró, programar los diferentes tipos de eventos que pueden ocurrir al interactuar con el programa, estos ya sean pulsar un botón, seleccionar un elemento de una lista, ingresar información, etc. Cada uno de estos eventos deberá ser manejado y en el caso de que no existan errores, comunicarse con la parte del modelo del código para realizar las acciones correspondientes.

Diagrama de análisis



Diseño de la solución

Para llevar a cabo la simulación del foro, por la parte del modelo, de decidió volver a utilizar el código escrito para el proyecto anterior (laboratorio 3 POO), este consiste en las siguientes clases.

1. Usuario: Esta clase será la encargada de representar un usuario registrado dentro del foro, este contendrá un nombre, contraseña y reputación.

2. ListaUsuarios: Esta clase contendrá una lista de usuarios registrados dentro del foro.

3. Pregunta: Representación de una pregunta dentro del foro, almacenará la fecha de creación, su autor, titulo, contenido, etiquetas, identificador, estado, recompensas ofrecidas, votos y una lista de respuestas a esta pregunta.

4. ListaPreguntas: Esta clase contendrá una lista de preguntas creadas dentro del foro.

5. Respuesta: Representación de una respuesta registrada a una pregunta, contendrá el nombre de su autor, fecha de creación, contenido, votos, estado y su respectivo identificador.

6. ListaRespuestas: Esta clase contendrá las respuestas pertenecientes a una pregunta.

7. Etiqueta: Representará una etiqueta de una pregunta, esta solamente contendrá un identificador, su nombre y una breve descripción de ella.

8. ListaEtiquetas: Esta clase contendrá las etiquetas registradas dentro del foro o las etiquetas que representan una pregunta creada.

9. Stack: Esta clase será la representación del foro, “contendrá” a las clases anteriormente mencionadas con una lista de preguntas, una lista de usuarios registrados, una lista de etiquetas registradas en el foro y el nombre del usuario que inicio sesión, si es que se hizo.

Cabe destacar que, si bien este proyecto se diseñó bajo el paradigma dirigido por eventos, también se utilizaros aspectos del paradigma orientado a objetos en este programa.

Para la parte de la vista del programa, se creó al menos una clase (ventana) para cada una de las funcionalidades, en algunos casos se utilizaron 2 para una mejor interacción por parte del usuario.

Para que cada ventana del programa tenga acceso a la información del foro y sea capaz de actualizar su información, cada una de las clases que representa una ventana de la interfaz posee como atributo un objeto de la clase Stack con la información del foro entregada por la ventana anterior, cada vez que una ventana cree la siguiente ventana, esta entregará la información actualizada del foro como argumento de su constructor. De esta manera se logrará mantener una comunicación constantes y actualización del foro.

Al momento de mostrar información del foro al usuario se utilizaron principalmente **textArea** y **list**. El primero de estos se usó para mostrar bloques de texto con información relacionada al foro, mientras que el segundo para presentar opciones de elección al usuario, por ejemplo: seleccionar una pregunta, una respuestas o múltiples etiquetas para una nueva pregunta.

Para el ingreso de información por parte del usuario se utilizaron **textField**, **textPane** y **passwordField**. El primero de estos se utilizó con información "pequeña" como el nombre de usuario, título de una pregunta o recompensa; por su parte **textPane** se usó para que el usuario introdujera información más extensa, tales como contenido de una pregunta, contenido de una respuesta o la descripción de una nueva etiqueta; el último de estos componentes fue utilizado solamente para cuando se introdujera la contraseña de una sesión, de esta manera esta información queda protegida al momento de interactuar con el programa.

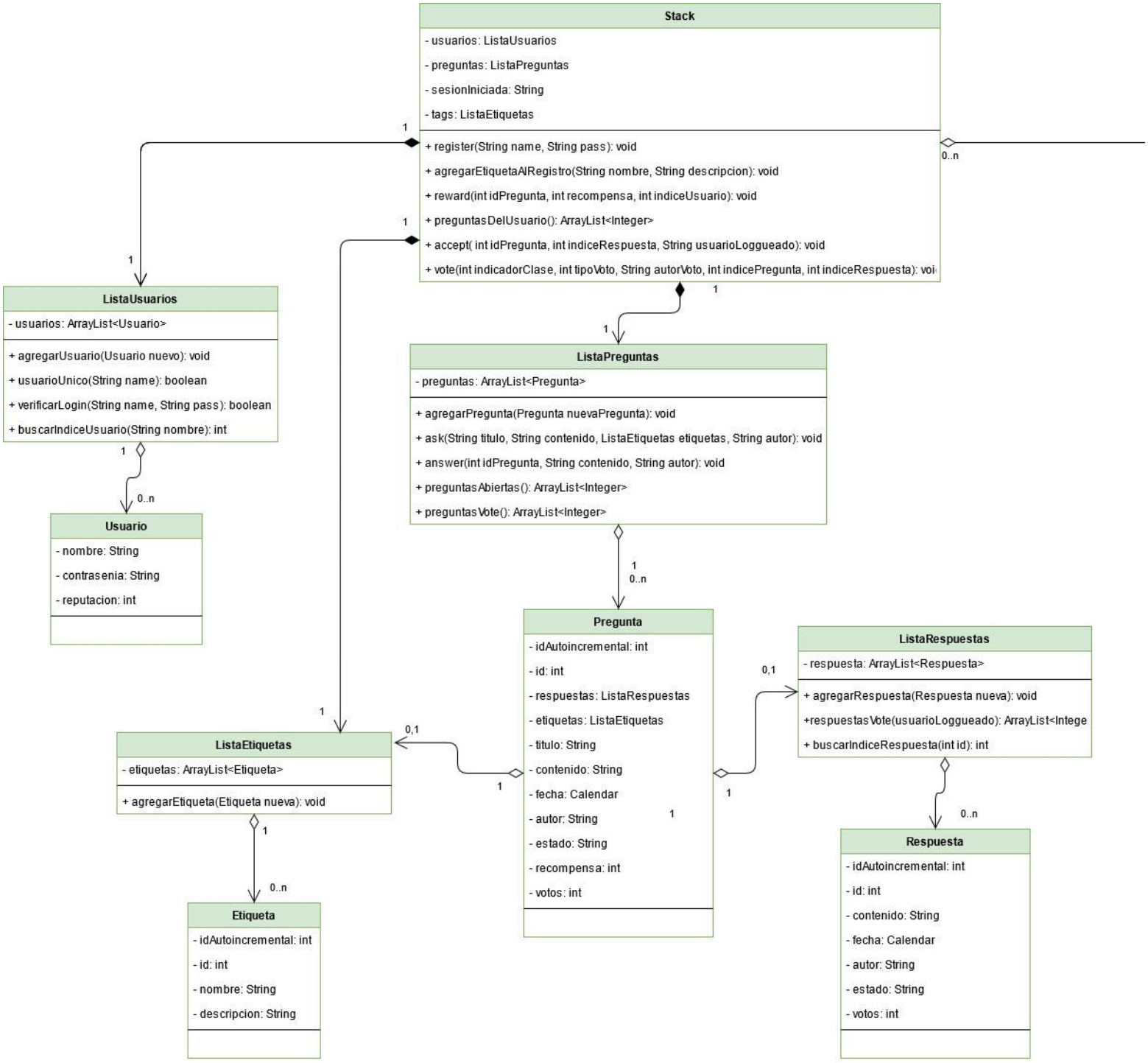
En cuando a los manejadores de eventos se hizo uso de 2 **listener** para estos eventos, el primero de estos es el llamado **actionPerformed**, este **listener** capta cuando se realiza una acción sobre un botón en el caso del diseño de este programa. Al momento de interactuar con estos botones, primero se comprueba que el usuario haya hecho uso del programa correctamente y que no exista información que pueda llegar a afectar las funcionalidades del programa. En caso de que exista un problema, se notificará al usuario mediante una ventana emergente la existencia de este, para ello se utilizó el componente llamado **JOptionPane** y mediante el método **showMessageDialog** se muestra un mensaje con la razón del por qué no se pudo ejecutar el programa correctamente. Si no existen errores, se llamarán los métodos del modelo para modificar la información del foro con la información proporcionada por el usuario mediante selección de elementos o ingreso de texto; luego de actualizar la información, se crea un objeto de la siguiente ventana, se entrega el foro actualizado, se cerrará la ventana actual y se hará visible la siguiente ventana. Al igual que la notificación de error, cuando una acción es ejecutada correctamente, se muestra al usuario un mensaje de que la acción se ejecutó sin problemas.

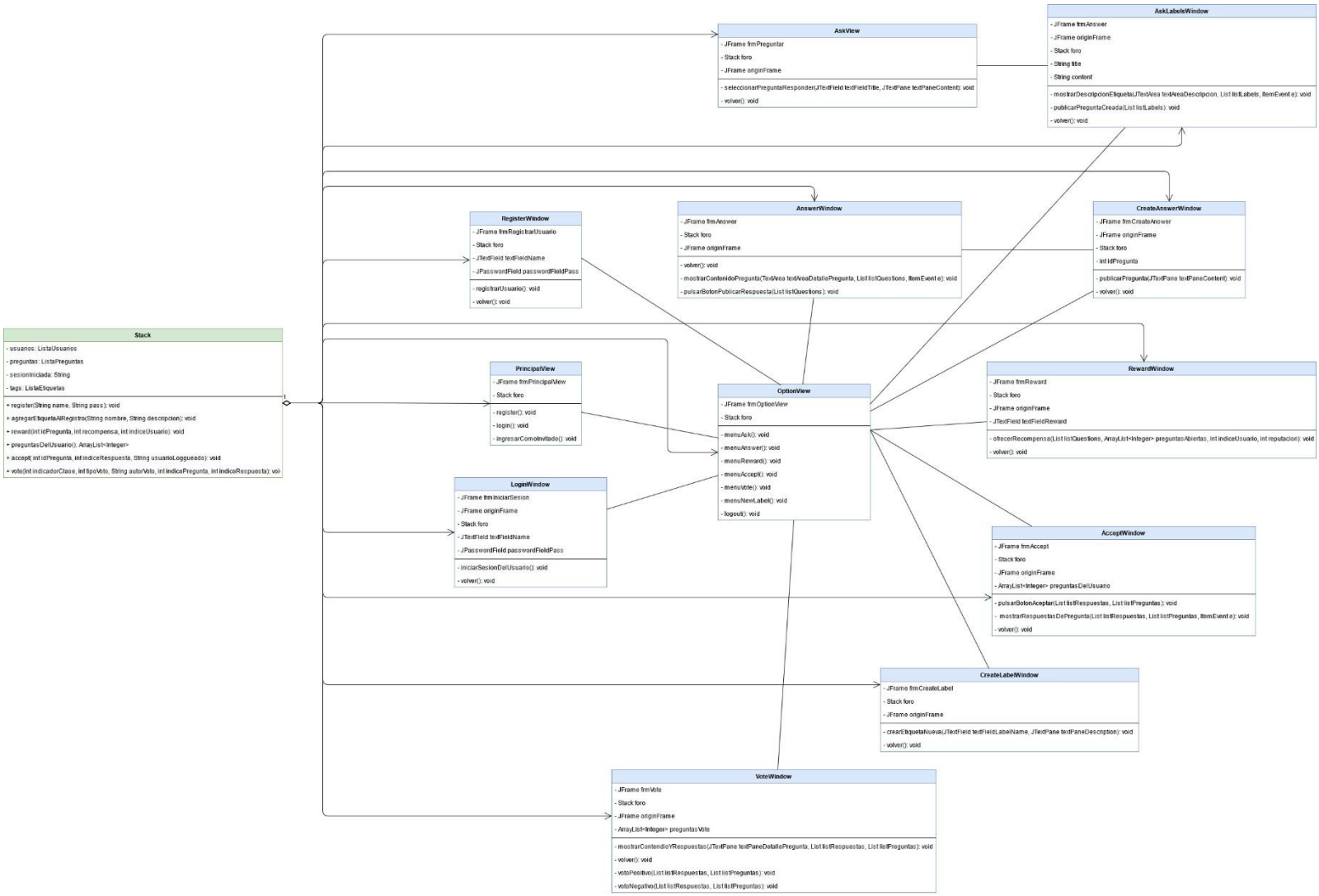
El segundo de los **listener** es el llamado **itemStateChanged**, este capaz de captar cuando un elemento de una lista (en el caso de este programa) es seleccionado o deseleccionado, esto se utilizó para dos casos, uno fue para que cuando se seleccione una pregunta o etiqueta de una lista, dentro de un panel de texto es mostrado el contenido de la pregunta o una breve descripción de la etiqueta según sea el la situación, el segundo caso de uso fue al momento de seleccionar una pregunta de una lista, al ser esta seleccionada otra lista es "llenada" con cada una de las respuestas que se necesiten de esta pregunta.

Para acceder a la información de cada uno de estos componentes se accedía al texto contenido en ellos o se obtenía el índice de el/los elemento/s seleccionado/s de una lista. En la segunda situación, cuando se mostraban todos los elementos de una lista, se aprovechaba el hecho de tanto en el elemento gráfico como dentro del Stack compartían el mismo índice y de esta manera se accedía a su información en el foro o en el caso de que solamente se muestren los elemento que cumplan cierta característica, previamente se creaba un arreglo de enteros que contenga los identificadores numérico de los elementos y cuando se necesitaba acceder a su información, se buscaba el id que se encuentra en la posición del elemento seleccionado dentro del elemento gráfico.

Como una pequeña funcionalidad extra, se decidió agregar la opción de ingresar al foro como un invitado, cuando esto ocurre todos los botones de la ventana que posee el menú principal son deshabilitados, a excepción del botón para cerrar sesión, de esta manera un usuario puede ser capaz de solamente visualizar la información del foro sin necesidad de registrarse o iniciar sesión previamente.

Diagrama de diseño




AskView
- JFrame frmPreguntar
- Stack foro
- JFrame originFrame
- seleccionarPreguntaResponder(JTextField textFieldTitle, JTextPane textPaneContent): void
- volver(): void
AskLabelsWindow
- JFrame frmAnswer
- JFrame originFrame
- Stack foro
- String title
- String content
- mostrarDescripcionEtiqueta(JTextArea textAreaDescripcion, List listLabels, ItemEvent e): void
- publicarPreguntaCreada(List listLabels): void
- volver(): void
RegisterWindow
- JFrame frmRegistrarUsuario
- Stack foro
- JTextField textFieldName
- JPasswordField passwordFieldPass
- registrarUsuario(): void
- volver(): void
AnswerWindow
- JFrame frmAnswer
- Stack foro
- JFrame originFrame
- volver(): void
- mostrarContenidoPregunta(TextArea textAreaDetallePregunta, List listQuestions, ItemEvent e): void
- pulsarBotonPublicarRespuesta(List listQuestions): void
CreateAnswerWindow
- JFrame frmCreateAnswer
- JFrame originFrame
- Stack foro
- int idPregunta
- publicarPregunta(JTextPane textPaneContent): void
- volver(): void
Stack
- usuarios: ListaUsuarios
- preguntas: ListaPreguntas
- sesionIniciada: String
- tags: ListaEtiquetas
+ register(String name, String pass): void
+ agregarEtiquetaARegistro(String nombre, String descripcion): void
+ reward(int idPregunta, int recompensa, int indiceUsuario): void
+ preguntasDelUsuario(): ArrayList<Integer>
+ accept(int idPregunta, int indiceRespuesta, String usuarioLogueado): void
+ vote(int indicadorClase, int tipoVoto, String autorVoto, int indicePregunta, int indiceRespuesta): voi
PrincipalView
- JFrame frmPrincipalView
- Stack foro
- register(): void
- login(): void
- ingresarComoInvitado(): void
OptionView
- JFrame frmOptionView
- Stack foro
- menuAsk(): void
- menuAnswer(): void
- menuReward(): void
- menuAccept(): void
- menuVote(): void
- menuNewLabel(): void
- logout(): void
RewardWindow
- JFrame frmReward
- Stack foro
- JFrame originFrame
- JTextField textFieldReward
- ofrecerRecompensa(List listQuestions, ArrayList<Integer> preguntasAbiertas, int indiceUsuario, int reputacion): void
- volver(): void
LoginWindow
- JFrame frmIniciarSesion
- JFrame originFrame
- Stack foro
- JTextField textFieldName
- JPasswordField passwordFieldPass
- iniciarSesionDelUsuario(): void
- volver(): void
AcceptWindow
- JFrame frmAccept
- Stack foro
- JFrame originFrame
- ArrayList<Integer> preguntasDelUsuario
- pulsarBotonAceptar(List listRespuestas, List listPreguntas): void
- mostrarRespuestasDePregunta(List listRespuestas, List listPreguntas, ItemEvent e): void
- volver(): void
CreateLabelWindow
- JFrame frmCreateLabel
- Stack foro
- JFrame originFrame
- crearEtiquetaNueva(JTextField textFieldLabelName, JTextPane textPaneDescription): void
- volver(): void
VoteWindow
- JFrame frmVote
- Stack foro
- JFrame originFrame
- ArrayList<Integer> preguntasVote
- mostrarContenidoYRespuestas(JTextPane textPaneDetallePregunta, List listRespuestas, List listPreguntas): void
- volver(): void
- votoPositivo(List listRespuestas, List listPreguntas): void
- votoNegativo(List listRespuestas, List listPreguntas): void

Aspectos de implementación

Para la implementación de este proyecto se utilizó Eclipse como intérprete del lenguaje de programación Java utilizando la versión 11 de OpenJDK, solamente fueron utilizadas funcionalidades pertenecientes a la librería estándar de Java, junto con las librerías Swing y AWT para diseñar la interfaz gráfica del programa . Las clases utilizadas para la implementación de este programa fueron divididas en sus correspondientes package, si bien no se siguió de manera total el modelo MVC (modelo-vista-controlador), se creó el package para los modelos y el package para la vista del programa. Además, para la compilación del programa se creó un script de compilación ".bat", que realiza la compilación automática y posterior ejecución del programa.

Instrucciones de uso

Cabe mencionar que el siguiente proceso para compilar y ejecutar el programa fue probado en un computador con sistema operativo Windows, por lo que funciona en este ambiente, pero no se asegura que funcione en otros sistemas operativos.

Para poder utilizar el programa por primera vez es necesario compilar el código, esto se realiza ejecutando el archivo llamado “scriptCompilar.bat” que se encuentra en la carpeta del programa llamada “código fuente”, al ejecutarlo aparecerá una ventana emergente en la que luego de compilar los archivos necesarios, volverá a abrir otra ventana emergente, esta vez será la interfaz gráfica del programa.

En caso de que se quiera volver a utilizar el programa una vez cerrado la primera vez, se debe abrir el programa PowerShell dentro de la carpeta en la que se encuentran los archivos, luego dentro de la ventana emergente que se abrirá, se debe ingresar la siguiente línea de comando “java Main” y presionar la tecla “enter”. Si es que se quiere evitar el procedimiento anterior, igualmente puede volver a ejecutar el archivo “scriptCaompilar.bat”, pero esto volverá a compilar innecesariamente los archivos del programa, aunque funcionando igualmente.

Cabe señalar que la interfaz gráfica se diseñó de una manera intuitiva para el usuario, en las ventanas que requieran una mayor explicación sobre su funcionamiento, se mostrará una pequeña instrucción de cómo utilizar dicha función correctamente. Sumando a lo anterior el programa entregará al usuario una retroalimentación sobre cada acción realizada, esto quiere decir que se indicará cuando una acción se ejecutó correctamente o cuando no se pudo ejecutar correctamente, agregando también la razón general del problema.

Resultados y autoevaluación

Se obtuvieron resultados satisfactorios al poner a prueba cada funcionalidad, para comprobar su correcto funcionamiento se realizaron pruebas con distintos valores, en los casos de ingresar valores validos estas acciones funcionaron correctamente y los valores inválidos fueron detectados correctamente.

Clases y estructuras	Para la parte del foro, se utilizaron las clases del proyecto anterior (laboratorio 3) y se crearon las clases para la vista de una manera que fuera intuitiva para el usuario.
Register / Login / Logout	Se comprueba correctamente que el nombre de usuario sea único al momento de registrarse, que los datos coincidan al momento de abrir sesión y se cierra la sesión satisfactoriamente volviendo a la ventana de ingreso.
Ask	Se logro implementar correctamente esta funcionalidad mediante dos ventanas, una para ingresar los datos y otra para elegir las etiquetas.
Answer	Se logró implementar correctamente con una ventana para elegir la respuesta a responder y otra para escribir el contenido de la respuesta.
Reward	Se ofrece correctamente la recompensa y se le descuenta inmediatamente al usuario. No se retiene temporalmente, ya que no hay forma de que el usuario retire una recompensa ofrecida.
Accept	Se muestran correctamente solo las preguntas del usuario que se encuentren abiertas y con respuestas que puedan ser aceptadas(no pertenecientes al usuario), se cambia el estado de la pregunta y respuesta, para luego aplicar el correspondiente cambio en la reputación de los usuarios involucrados.
Vote	Se muestran al usuario solamente las preguntas y respuestas que no fueron creadas por él. En caso de que exista una pregunta creada por él que tenga una respuesta de otra persona, se muestra dicha pregunta, pero no se puede votar por ella. En ambos casos posteriormente se actualiza la cantidad de votos y se modifica la reputación de los usuarios involucrados.

Conclusión

Ya que se decidió utilizar como base el código del proyecto anterior (laboratorio 3), solo fue necesario crear la interfaz gráfica, por lo que adaptarse a este nuevo paradigma no resultó tan dificultoso como los anteriores cambios de paradigma. A pesar de esto, requirió de trabajo equilibrar código entre diseñar una interfaz gráfica lo más intuitiva y amigable para el usuario, y realizar un diseño que no sobre complique el código solo por una mejora estética. Debido a la poca familiaridad con el desarrollo de interfaces gráficas, existieron ciertas limitaciones al momento de presentar los datos u opciones al usuario, por lo que se tuve que buscar una manera posible de programar y que presentara los datos de manera entendible, pero quizás no tan "estética". El realizar un diagrama de previamente a la escritura del código, ayudo enormemente a tener una idea clara en el flujo del programa y la interacción entre cada ventana de la interfaz. Por lo anteriormente descrito se considera que el proyecto fue realizado de una manera correcta y su desarrollo será de vital importancia para la formación como programador.

Referencias

Campos, O. (2011). Introducción a la programación dirigida por eventos. (Recuperado el 04/03/2021) https://www.genbeta.com/desarrollo/introduccion-a-la-programacion-dirigida-por-eventos

Gonzáles, A. (2015). JavaEventBasedProgramming. (Recuperado el 04/03/2021) http://profesores.elo.utfsm.cl/~agv/elo329/1s15/lectures/Java/JavaEventBasedProgramming.pdf

Universidad nacional de La Rioja. (2020). Paradigma y lenguajes I. (Recuperado el 04/03/2021) https://www.studocu.com/es-ar/document/universidad-nacional-de-la-rioja/paradigmas-y-lenguajes-i/informes/paradigmas-orientados-a-eventos/2821400/view