UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
INTRODUCCION A LA PROGRAMACIÓN Y COMPUTACIÓN 1
CATEDRÁTICO: ING. NEFTALI DE JESUS CALDERON MENDEZ
TUTOR ACADÉMICO: DOUGLAS ALEXANDER SOCH CATALÁN

MANUAL TÉCNICO

IPC QUIMIK

MELVIN GEOVANNI GARCÍA SUMALÁ
CARNÉ: 202300712
SECCIÓN: E

GUATEMALA, 03 DE SEPTIEMBRE DEL 2,024

índice

INTRODUCCIÓN

El sistema integra funcionalidades clave, como la autenticación de usuarios, la gestión de investigadores y muestras, y la comparación de patrones. También se implementaron reportes automáticos en formato HTML y la serialización de datos para asegurar su persistencia. La documentación técnica, que incluye manuales y un diagrama de clases, se desarrolló para garantizar la correcta utilización y mantenimiento del sistema.

OBJETIVOS DEL SISTEMA

GENERAL

- Familiarizar al estudiante con el lenguaje de programación Java.
- Aplicar los conocimientos adquiridos en programación y computación para resolver una problemática real.
- Desarrollar la lógica necesaria para crear un sistema de gestión de muestras y experimentos en un laboratorio químico.

ESPECÍFICOS

- Utilizar Java para el desarrollo de software orientado a objetos.
- Implementar una interfaz gráfica de usuario con AWT y Swing.
- Desarrollar funcionalidades de control, manejo de arreglos, matrices y administración de datos.
- Crear herramientas administrativas que faciliten la gestión de investigadores y muestras.
- Implementar algoritmos de encriptación y serialización para la seguridad y persistencia de datos.

REQUISITOS DEL SISTEMA

- Lenguaje de Programación**:** Java (uso de AWT y Swing).
- IDE Recomendado**:** NetBeans.
- AWT, Swing, LinkedList, ArrayList, JFreeChart, java.io, java.lang.Math.
- Persistencia de Datos**:** Serialización en archivos binarios.
- Interfaz Gráfica**:** Desarrollo con AWT/Swing, se permite el uso de Drag and Drop en IDE.
- Repositorio: Privado en GitHub.
- Documentación: Manual Técnico, Manual de Usuario, Diagrama de Clases y Diagramas de Flujo.

INFORMACIÓN DEL SISTEMA

Investigador

Descripción: Modelo de datos que representa a un investigador, permitiendo la serialización para almacenamiento en archivos binarios.

Atributos:

`codigo`: Código único del investigador.

`nombre`: Nombre del investigador.

`genero`: Género del investigador.

`experimento`: Número de experimentos en los que ha participado.

`contrasenia`: Contraseña del investigador.

Métodos Clave:

Constructores para inicializar un investigador con o sin datos.

Métodos `get` y `set` para acceder y modificar los atributos.

frmCrearInvestigador

Descripción: Interfaz gráfica (GUI) que permite la creación de un nuevo investigador.

Atributos:

`menuAdmin`: Referencia al menú de administración.

`txtCodigo`, `txtNombre`, `txtGenero`, `txtContrasenia`: Campos de texto para ingresar los datos del investigador.

`jButton1`: Botón para ejecutar la acción de creación.

Métodos Clave:

`frmCrearInvestigador()`: Constructores que inicializan la interfaz.

`jButton1ActionPerformed`: Maneja el evento de creación, verificando códigos duplicados y almacenando datos.

`isCodigoDuplicado`: Verifica si el código del investigador ya existe.

frmActualizarInvestigador

Descripción: Interfaz gráfica (GUI) que permite actualizar la información de un investigador existente.

Atributos:

`menuAdmin`: Referencia al menú de administración.

`txtCodigo`, `txtContrasenia`: Campos de texto para el código y la contraseña del investigador.

`jButton1`: Botón para ejecutar la acción de actualización.

Métodos Clave:

`frmActualizarInvestigador()`: Constructores que inicializan la interfaz.

frmEliminarInvestigador

Descripción: Interfaz gráfica (GUI) que permite eliminar un investigador de un archivo binario.

Atributos:

`txtCodigo`: Campo de texto para ingresar el código del investigador a eliminar.

`jButton1`: Botón para ejecutar la acción de eliminación.

`menuAdmin`: Referencia al menú de administración.

Métodos Clave:

`initComponents()`: Inicializa los componentes de la interfaz gráfica.

`jButton1ActionPerformed`: Maneja el evento de eliminación, eliminando al investigador del archivo binario, mostrando un mensaje de confirmación y actualizando la interfaz de administración.

`main(String args[])`: Método principal que inicia la aplicación y muestra la interfaz gráfica.

Estas clases colaboran para gestionar la creación, actualización, y eliminación de investigadores dentro de la aplicación, así como para manejar el almacenamiento y la interfaz gráfica.

ManejadorArchivoBinarioInvestigador

Descripción: Esta clase se encarga de manejar operaciones de entrada y salida de archivos binarios que contienen objetos de tipo `Investigador`. Proporciona métodos para obtener, imprimir, borrar el contenido del archivo, y generar una gráfica con los tres investigadores que tienen más experimentos registrados.

Muestra

Descripción: Modelo de datos que representa una muestra en la aplicación. Implementa la interfaz Serializable para facilitar su almacenamiento y recuperación desde archivos binarios.

Atributos:

`matrix`: Lista de listas de enteros que representa la matriz de datos de la muestra.

`codigo`: Código único de la muestra.

`descripcion`: Descripción de la muestra.

`estado`: Estado de la muestra.

`codigoCSV`: Código CSV asociado a la muestra.

Métodos Clave:

Constructor para inicializar una nueva muestra con los valores proporcionados.

Métodos `get` y `set` para acceder y modificar la matriz, código, descripción, estado y código CSV de la muestra.

frmCrearMuestra

Descripción: Interfaz gráfica de usuario (GUI) que permite crear una nueva muestra. Convierte una matriz de cadenas a una matriz de enteros, genera una representación CSV de la matriz y guarda la nueva muestra en un archivo binario.

Atributos:

`matrix`: Matriz de cadenas que se convierte a una matriz de enteros.

`intMatrix`: Matriz de enteros resultante de la conversión de la matriz de cadenas.

`csvContent`: Contenido CSV generado a partir de la matriz de cadenas.

`muestras`: Lista de muestras que se actualiza con la nueva muestra.

`file`: Archivo donde se guardan las muestras.

Métodos Clave:

`main(String args[])`: Método principal que inicia la aplicación y muestra la interfaz gráfica.

Conversión de matriz de cadenas a matriz de enteros, manejando posibles excepciones de formato.

Generación de contenido CSV a partir de la matriz de cadenas.

Agregar nueva muestra a la lista de muestras y guardar la lista actualizada en un archivo binario.

Patron

Descripción: Modelo de datos que representa un patrón en la aplicación. Implementa la interfaz `Serializable` para permitir la serialización de sus instancias, facilitando su almacenamiento y recuperación desde archivos binarios.

Atributos:

`codigo`: Cadena que almacena el código único del patrón.

`nombre`: Cadena que almacena el nombre del patrón.

`csvContent`: Cadena que almacena el contenido del patrón en formato CSV.

Métodos Clave:

Constructor que inicializa una nueva instancia de `Patron` con los valores proporcionados.

Métodos `get` y `set` para acceder y modificar el código, nombre, y contenido CSV del patrón.

frmCrearPatron

Descripción: Interfaz gráfica de usuario (GUI) que permite crear un nuevo patrón a partir de un archivo CSV. Maneja la carga del archivo CSV, la conversión de su contenido a una matriz, y el almacenamiento de la información en un archivo binario. Además, actualiza una tabla en la interfaz de administración con los datos del patrón.

Atributos:

`selectedFile`: Archivo seleccionado por el usuario a través del `JFileChooser`.

`codigoField`: Campo de texto para ingresar el código del patrón.

`nombreField`: Campo de texto para ingresar el nombre del patrón.

`jButton1`: Botón para cargar el archivo CSV.

`jButton2`: Botón para guardar el patrón.

`menuAdmin`: Referencia a la instancia del menú de administración (`frmMenuAdministracion`).

Métodos Clave:

`frmCrearPatron(frmMenuAdministracion menuAdmin)`: Constructor que inicializa la interfaz gráfica y carga los datos desde el archivo binario al abrir la ventana.

`initComponents()`: Método que inicializa los componentes de la interfaz gráfica.

`jButton1ActionPerformed(ActionEvent evt)`: Maneja el evento de clic en el botón "Cargar CSV".

`jButton2ActionPerformed(ActionEvent evt)`: Maneja el evento de clic en el botón "Guardar".

`readCSV(File file)`: Lee un archivo CSV y lo convierte en una matriz de cadenas.

`saveMatrixToBin(List<List<String>> matrix, String codigo, String nombre)`: Guarda la matriz de cadenas en un archivo binario junto con el código y nombre del patrón.

`updateTable(List<List<String>> matrix)`: Actualiza la tabla en la interfaz de administración con los datos de la matriz.

`updateTableFromBin()`: Carga los datos desde el archivo binario y actualiza la tabla en la interfaz de administración.

frmEliminarPatron

Descripción: Interfaz gráfica de usuario (GUI) que permite eliminar un patrón de un archivo binario. Utiliza componentes de Swing para la interfaz y se integra con otras clases para manejar la lógica de negocio, como la eliminación de datos en un archivo binario y la actualización de la interfaz de administración.

Atributos:

`txtCodigo`: Campo de texto para ingresar el código del patrón a eliminar.

`jButton1`: Botón para ejecutar la acción de eliminación.

`jLabel1`: Etiqueta que muestra el título "Eliminar Patron".

`jLabel2`: Etiqueta que muestra el texto "Codigo".

Métodos Clave:

`initComponents()`: Método que inicializa los componentes de la interfaz gráfica.

`jButton1ActionPerformed(java.awt.event.ActionEvent evt)`: Maneja el evento de clic en el botón "Eliminar".

`main(String args[])`: Método principal que inicia la aplicación y muestra la interfaz gráfica.

ManejadorArchivoBinarioPatron

Descripción: Esta clase gestiona operaciones de entrada y salida de archivos binarios que contienen objetos de tipo `Patron`. Proporciona métodos para agregar, eliminar, obtener, imprimir y borrar patrones en el archivo.

Resultado

Descripción: Modelo de datos que representa el resultado de una comparación entre una muestra y un patrón en la aplicación. Implementa la interfaz `Serializable` para permitir la serialización de sus instancias, facilitando su almacenamiento y recuperación desde archivos binarios.

Atributos:

`codigoMuestra`: Cadena que almacena el código de la muestra asociada al resultado.

`codigoPatron`: Cadena que almacena el código del patrón asociado al resultado.

`fecha`: Cadena que almacena la fecha en la que se realizó la comparación.

`hora`: Cadena que almacena la hora en la que se realizó la comparación.

`resultado`: Cadena que almacena el resultado de la comparación.

Métodos Clave:

Métodos `get` y `set` para acceder y modificar el código de la muestra, el código del patrón, la fecha, la hora, y el resultado de la comparación.

ManejadorArchivoBinarioResultado

Descripción: Clase encargada de manejar operaciones de entrada y salida de archivos binarios que contienen objetos de tipo `Resultado`. Proporciona métodos para obtener, imprimir y borrar resultados en el archivo.

Atributos:

`respuesta`: `ArrayList<Resultado>` - Utilizada para almacenar y devolver la lista de resultados leída del archivo.

`archivo`: `File` - Representa el archivo desde el cual se leerán o escribirán los datos.

`resultados`: `ArrayList<Resultado>` - Utilizada para almacenar la lista de resultados obtenida del archivo.

`listadoResultados`: `List<Resultado>` - Utilizada para crear una lista vacía al borrar el contenido del archivo.

Métodos Clave:

`obtenerResultados(String rutaArchivo)`: Lee y devuelve una lista de objetos `Resultado` desde un archivo binario.

`imprimirResultados(String rutaArchivo)`: Imprime en consola los detalles de cada resultado contenido en el archivo.

`borrarContenido(String rutaArchivo)`: Borra el contenido del archivo, escribiendo una lista vacía de resultados.

LoginFrame

Descripción: `LoginFrame` es una interfaz gráfica de usuario (GUI) diseñada para permitir a los investigadores iniciar sesión en la aplicación. Utiliza componentes de Swing y maneja la autenticación mediante la verificación de códigos y contraseñas almacenados en un archivo binario.

Atributos:

`codigoField`: Campo de texto para ingresar el código del investigador.

`passwordField`: Campo de texto para ingresar la contraseña del investigador.

`investigadores`: Lista de objetos `Investigador` obtenidos del archivo binario.

`codigosExistentes`: Conjunto de códigos de investigadores existentes para verificación rápida.

Métodos Clave:

`LoginFrame()`: Constructor que configura el frame, inicializa los componentes de la interfaz gráfica y carga los datos de los investigadores desde un archivo binario.

`initComponents()`: Método implícito en el constructor que inicializa y configura los componentes de la interfaz gráfica.

`main(String args[])`: Método principal que inicia la aplicación y muestra la interfaz gráfica de inicio de sesión.

Nota: `LoginFrame` decide a qué menú redirigir al usuario tras la autenticación exitosa, ya sea `frmMenuAdministracion` para administradores o `frmMenuUsuario` para usuarios estándar.

frmMenuAdministracion

Descripción: `frmMenuAdministracion` es una interfaz gráfica de usuario (GUI) que sirve como menú principal para los administradores en la aplicación. Facilita el acceso a funciones clave como la creación, eliminación y gestión de investigadores, muestras y patrones.

Atributos:

`jButton1`: Botón para acceder a la funcionalidad de crear investigadores.

`jButton2`: Botón para acceder a la funcionalidad de eliminar investigadores.

`jButton3`: Botón para acceder a la funcionalidad de crear muestras.

`jButton4`: Botón para acceder a la funcionalidad de eliminar muestras.

`jButton5`: Botón para acceder a la funcionalidad de crear patrones.

`jButton6`: Botón para acceder a la funcionalidad de eliminar patrones.

`jLabel1`: Etiqueta que muestra el título del menú de administración.

Métodos Clave:

`initComponents()`: Método que inicializa los componentes de la interfaz gráfica.

Métodos `jButton1ActionPerformed`, `jButton2ActionPerformed`, etc.: Manejan los eventos de clic en los botones correspondientes para abrir las ventanas de creación o eliminación de investigadores, muestras y patrones.

frmMenuUsuario

Descripción: Interfaz gráfica de usuario (GUI) que sirve como menú principal para los usuarios en la aplicación. Permite a los usuarios acceder a diferentes funcionalidades como la visualización de investigadores, muestras y patrones, así como la gestión de sus propios datos.

Atributos:

`jButton1`: Botón para acceder a la funcionalidad de ver investigadores.

`jButton2`: Botón para acceder a la funcionalidad de ver muestras.

`jButton3`: Botón para acceder a la funcionalidad de ver patrones.

`jButton4`: Botón para acceder a la funcionalidad de gestionar datos del usuario.

`jLabel1`: Etiqueta que muestra el título del menú de usuario.

Métodos Clave:

`initComponents()`: Método que inicializa los componentes de la interfaz gráfica.

`jButton1ActionPerformed(java.awt.event.ActionEvent evt)`: Maneja el evento de clic en el botón "Ver Investigadores".

`jButton2ActionPerformed(java.awt.event.ActionEvent evt)`: Maneja el evento de clic en el botón "Ver Muestras".

`jButton3ActionPerformed(java.awt.event.ActionEvent evt)`: Maneja el evento de clic en el botón "Ver Patrones".

`jButton4ActionPerformed(java.awt.event.ActionEvent evt)`: Maneja el evento de clic en el botón "Gestionar Datos".

Asignacion

Descripción: `Asignacion` es un modelo de datos que representa la asignación de un investigador a una muestra. Implementa la interfaz `Serializable` para permitir la serialización de sus instancias, facilitando su almacenamiento y recuperación desde archivos binarios.

Atributos:

`codigoInvestigador`: Cadena que almacena el código del investigador asignado.

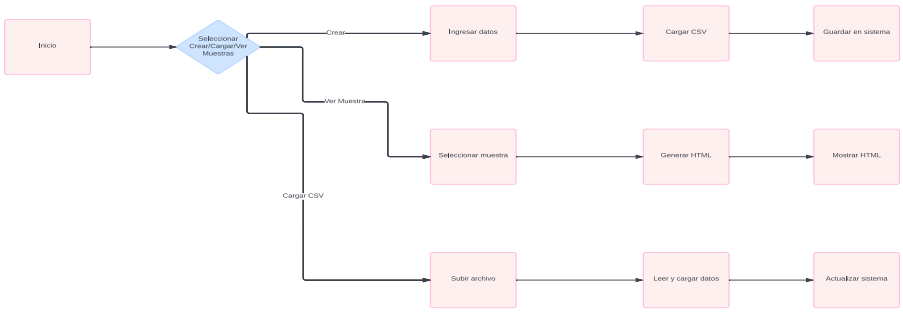
`codigoMuestra`: Cadena que almacena el código de la muestra asignada.

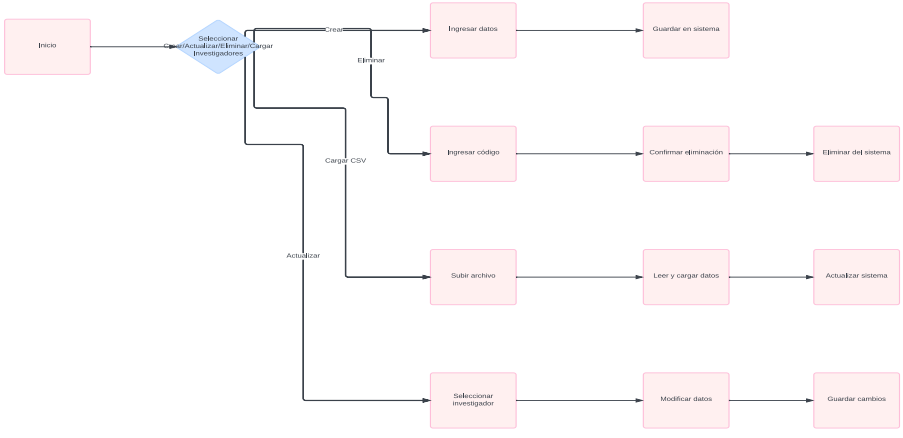
Métodos Clave:

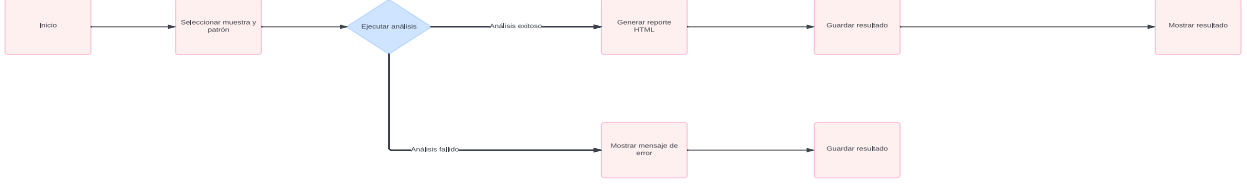
`Asignacion(String codigoInvestigador, String codigoMuestra)`: Constructor que inicializa una nueva instancia de `Asignacion` con los códigos del investigador y la muestra.

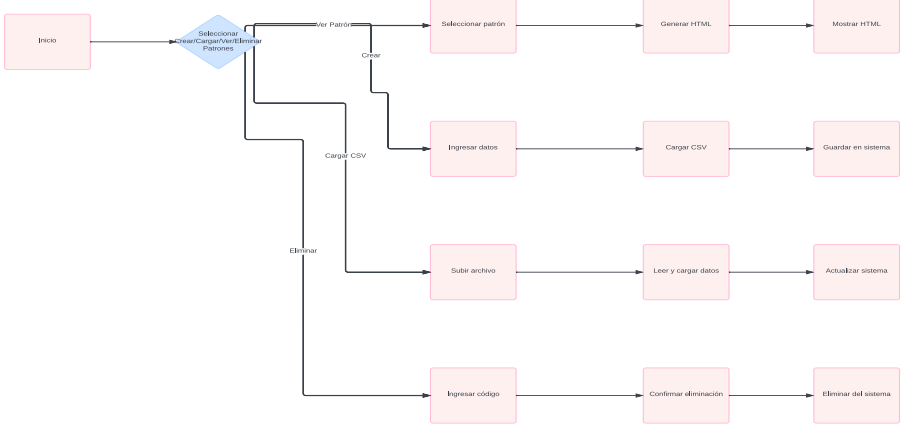
Métodos `get` y `set` para acceder y modificar los códigos del investigador y la muestra asignada.

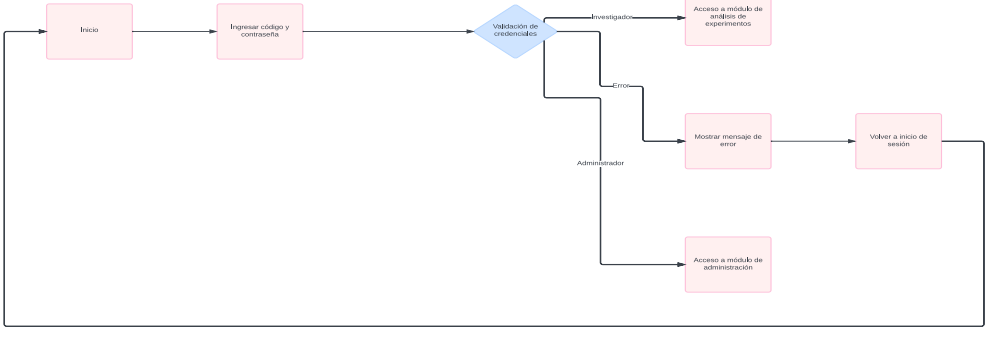
Nota: La clase `Asignacion` es clave en la funcionalidad del `frmMenuUsuario`, donde los usuarios pueden gestionar y revisar sus asignaciones y resultados.


Inicio
Seleccionar Crear/Cargar/Ver Muestras
Crear
Ingresar datos
Cargar CSV
Guardar en sistema
Ver Muestras
Seleccionar muestra
Generar HTML
Mostrar HTML
Cargar CSV
Subir archivo
Leer y cargar datos
Actualizar sistema


Inicio
Seleccionar Crear/Actualizar/Eliminar/Cargar Investigadores
Crear
Ingresar datos
Guardar en sistema
Eliminar
Ingresar código
Confirmar eliminación
Eliminar del sistema
Cargar CSV
Subir archivo
Leer y cargar datos
Actualizar sistema
Actualizar
Seleccionar investigador
Modificar datos
Guardar cambios


Inicio
Seleccionar muestra y patrón
Ejecutar análisis
Análisis exitoso
Generar reporte HTML
Guardar resultado
Mostrar resultado
Análisis fallido
Mostrar mensaje de error
Guardar resultado


Inicio
Seleccionar Crear/Cargar/Ver/Eliminar Patrones
Ver Patrón
Seleccionar patrón
Generar HTML
Mostrar HTML
Crear
Ingresar datos
Cargar CSV
Guardar en sistema
Cargar CSV
Subir archivo
Leer y cargar datos
Actualizar sistema
Eliminar
Ingresar código
Confirmar eliminación
Eliminar del sistema


Inicio
Ingresar código y contraseña
Validación de credenciales
Investigador
Acceso a módulo de análisis de experimentos
Error
Mostrar mensaje de error
Volver a inicio de sesión
Administrador
Acceso a módulo de administración