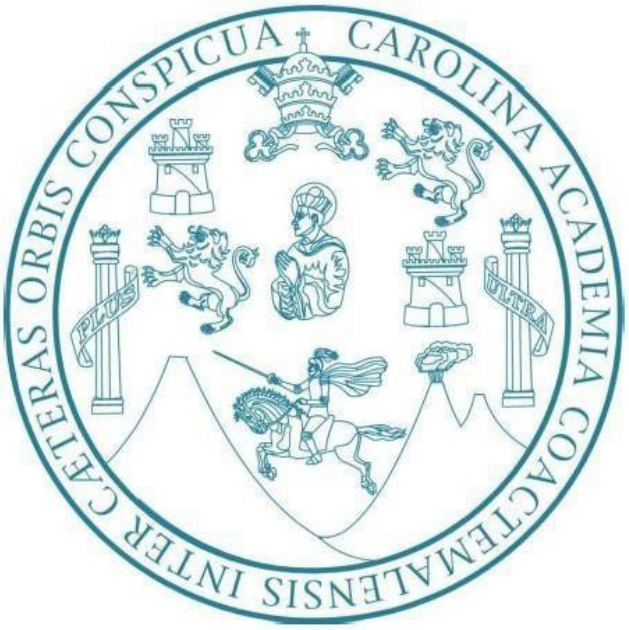
Universidad de San Carlos de Guatemala
División de Ciencias de la Ingeniería
Introducción a la Programación y Computación 2



DOCUMENTACIÓN PRÁCTICA 1
GESTOR DE BILBIOTECA

Estudiantes:
Carné: 201831260
Nombre: Michael Kristopher Marín Reyes
Carné: 202132299
Nombre: Gérman Alex Ramirez Límatuj

Sección: A

GESTOR DE BIBLIOTECA

La siguiente documentación muestra la estructura de código en la cual se construyo la aplicación, se conforma de la siguiente manera:

Paquete backend/importaciondedatos

Clase: ImportarDatos

Descripción: Esta clase proporciona métodos para importar datos desde un archivo de texto. Los datos importados pueden ser libros, estudiantes y préstamos. Los libros y estudiantes importados se clasifican y agregan a la aplicación. Los préstamos se registran si tanto el libro como el estudiante asociado existen en la aplicación.

Paquete: backend.importaciondedatos

Métodos:

1. `public void abrirArchivo(String ruta)`
    - Descripción: Abre el archivo especificado y procesa cada línea para importar los datos según su tipo.
    - Parámetros:
        - `ruta`: La ruta del archivo a abrir.
    - Excepciones:
        - `IOException`: Si ocurre un error durante la lectura del archivo.
2. `private void importarLibro(BufferedReader br) throws IOException`
    - Descripción: Importa un libro desde el archivo proporcionado y lo agrega a la aplicación si no está repetido.
    - Parámetros:
        - `br`: BufferedReader para leer las líneas del archivo.
    - Excepciones:
        - `IOException`: Si ocurre un error durante la lectura del archivo.
3. `private void importarEstudiante(BufferedReader br) throws IOException`
    - Descripción: Importa un estudiante desde el archivo proporcionado y lo agrega a la aplicación si no está repetido.
    - Parámetros:
        - `br`: BufferedReader para leer las líneas del archivo.
    - Excepciones:
        - `IOException`: Si ocurre un error durante la lectura del archivo.
4. `private void importarPrestamo(BufferedReader br) throws IOException`
    - Descripción: Importa un préstamo desde el archivo proporcionado y lo registra si tanto el libro como el estudiante asociado existen en la aplicación.

- Parámetros:
    - `br`: BufferedReader para leer las líneas del archivo.
- Excepciones:
    - `IOException`: Si ocurre un error durante la lectura del archivo.

Atributos:

1. `private FuncionamientoAplicacion clasificar`
    - Descripción: Instancia de la clase FuncionamientoAplicacion utilizada para clasificar y gestionar los datos importados.
2. `private String path`
    - Descripción: Ruta del archivo que se está importando.

Dependencias Externas:

- `backend.principal.FuncionamientoAplicacion`: Se utiliza para clasificar y gestionar los datos importados.

Autor: Michael.

Paquete backend/principal

Clase: Estudiante

Descripción: Esta clase representa a un estudiante en el sistema de gestión de préstamos de libros. Contiene información como el carnet, nombre, código de carrera, fecha de nacimiento y la cantidad de libros prestados actualmente.

Paquete: backend.principal

Métodos:

1. `public Estudiante(int carnet, String nombre, int codigoCarrera, LocalDate fechaNacimiento)`
    - Descripción: Constructor de la clase Estudiante.
    - Parámetros:
        - `carnet`: Número de carnet del estudiante.
        - `nombre`: Nombre del estudiante.
        - `codigoCarrera`: Código de la carrera del estudiante.
        - `fechaNacimiento`: Fecha de nacimiento del estudiante.
2. `public int getCarnet()`
    - Descripción: Obtiene el número de carnet del estudiante.
    - Retorna: El número de carnet.
3. `public void setCarnet(int carnet)`

- Descripción: Establece el número de carnet del estudiante.
- Parámetros:
    - `carnet`: El número de carnet a establecer.

4. `public String getNombre()`
    - Descripción: Obtiene el nombre del estudiante.
    - Retorna: El nombre del estudiante.
5. `public void setNombre(String nombre)`
    - Descripción: Establece el nombre del estudiante.
    - Parámetros:
        - `nombre`: El nombre a establecer.
6. `public int getCodigoCarrera()`
    - Descripción: Obtiene el código de carrera del estudiante.
    - Retorna: El código de carrera.
7. `public void setCodigoCarrera(int codigoCarrera)`
    - Descripción: Establece el código de carrera del estudiante.
    - Parámetros:
        - `codigoCarrera`: El código de carrera a establecer.
8. `public LocalDate getFechaNacimiento()`
    - Descripción: Obtiene la fecha de nacimiento del estudiante.
    - Retorna: La fecha de nacimiento.
9. `public void setFechaNacimiento(LocalDate fechaNacimiento)`
    - Descripción: Establece la fecha de nacimiento del estudiante.
    - Parámetros:
        - `fechaNacimiento`: La fecha de nacimiento a establecer.
10. `public int getLibrosPrestados()`
    - Descripción: Obtiene la cantidad de libros prestados al estudiante.
    - Retorna: La cantidad de libros prestados.
11. `public void setLibrosPrestados(int librosPrestados)`
    - Descripción: Establece la cantidad de libros prestados al estudiante.
    - Parámetros:
        - `librosPrestados`: La cantidad de libros prestados a establecer.
12. `public boolean puedePrestarLibros()`
    - Descripción: Verifica si el estudiante puede tomar prestados más libros (máximo 3).
    - Retorna: `true` si el estudiante puede tomar prestados más libros; `false` en caso contrario.

Atributos:

1. `private int carnet`
    - Descripción: Número de carnet del estudiante.
2. `private String nombre`
    - Descripción: Nombre del estudiante.
3. `private int codigoCarrera`
    - Descripción: Código de la carrera del estudiante.
4. `private LocalDate fechaNacimiento`
    - Descripción: Fecha de nacimiento del estudiante.
5. `private int librosPrestados`
    - Descripción: Cantidad de libros prestados actualmente al estudiante.

Autor: Michael

Clase: FuncionamientoAplicacion

Descripción: Esta clase maneja el funcionamiento general de la aplicación de gestión de préstamos de libros. Se encarga de administrar las listas de préstamos, devoluciones, libros y estudiantes, así como de llevar a cabo operaciones como agregar nuevos libros o estudiantes, prestar libros, devolver libros, entre otras.

Paquete: backend.principal

Métodos:

1. `public FuncionamientoAplicacion()`
    - Descripción: Constructor de la clase FuncionamientoAplicacion. Se encarga de abrir los archivos serializables al inicializar la aplicación.
2. `public void agregarNuevoLibro(String codigo, String autor, String titulo, int cantidadCopias, LocalDate fechaPublicación, String editorial)`
    - Descripción: Agrega un nuevo libro a la base de datos.
    - Parámetros:
        - `codigo`: Código del libro.
        - `autor`: Autor del libro.
        - `titulo`: Título del libro.
        - `cantidadCopias`: Cantidad de copias disponibles del libro.
        - `fechaPublicación`: Fecha de publicación del libro.
        - `editorial`: Editorial del libro.
3. `public void actualizarLibro(Libro libroActualizado)`
    - Descripción: Actualiza los datos de un libro específico en la base de datos.

- Parámetros:
    - `libroActualizado`: Libro con los datos actualizados.

4. `public void agregarNuevoEstudiante(int carnet, String nombre, int codigoCarrera, LocalDate fechaNacimiento)`
    - Descripción: Agrega un nuevo estudiante a la base de datos.
    - Parámetros:
        - `carnet`: Número de carnet del estudiante.
        - `nombre`: Nombre del estudiante.
        - `codigoCarrera`: Código de la carrera del estudiante.
        - `fechaNacimiento`: Fecha de nacimiento del estudiante.
5. `public void prestarLibro(Libro codigoLibro, Estudiante estudiante, LocalDate fechaPrestamo, LocalDate fechaDevolucion, int monto)`
    - Descripción: Gestiona el préstamo de un libro a un estudiante.
    - Parámetros:
        - `codigoLibro`: Libro que se va a prestar.
        - `estudiante`: Estudiante que toma prestado el libro.
        - `fechaPrestamo`: Fecha en que se realiza el préstamo.
        - `fechaDevolucion`: Fecha de devolución prevista del libro.
        - `monto`: Monto asociado al préstamo.
6. `public void incrementarLibrosPrestadosPorEstudiante(String carnetEstudiante)`
    - Descripción: Incrementa el contador de libros prestados por estudiante.
    - Parámetros:
        - `carnetEstudiante`: Carnet del estudiante al que se le incrementa el contador.
7. `public int getLibrosPrestadosPorEstudiante(int carnetEstudiante)`
    - Descripción: Obtiene la cantidad de libros prestados a un estudiante.
    - Parámetros:
        - `carnetEstudiante`: Carnet del estudiante.
    - Retorna: La cantidad de libros prestados al estudiante.
8. `public void ordenarLibros()`
    - Descripción: Ordena la lista de libros por su código en orden ascendente.
9. `public void agregarDevoluciones(Prestamo prestamo)`
    - Descripción: Agrega un préstamo a la lista de devoluciones.
    - Parámetros:
        - `prestamo`: Préstamo que se agrega a la lista de devoluciones.
10. `public boolean validarEstudiantesRepetidos(String carnet)`

- Descripción: Valida si un estudiante con el carnet especificado ya existe en la base de datos.
- Parámetros:
  - `carnet`: Carnet del estudiante a validar.
- Retorna: `true` si el estudiante ya existe, `false` en caso contrario.

11. `public boolean validarLibroRepetido(String codigo)`

- Descripción: Valida si un libro con el código especificado ya existe en la base de datos.
- Parámetros:
  - `codigo`: Código del libro a validar.
- Retorna: `true` si el libro ya existe, `false` en caso contrario.

12. `public Libro buscarLibroPorCodigo(String codigoLibro)`

- Descripción: Busca un libro en la base de datos por su código.
- Parámetros:
  - `codigoLibro`: Código del libro a buscar.
- Retorna: El libro encontrado o `null` si no se encuentra ningún libro con ese código.

13. `public Estudiante buscarEstudiantePorCarnet(int carnetEstudiante)`

- Descripción: Busca un estudiante en la base de datos por su carnet.
- Parámetros:
  - `carnetEstudiante`: Carnet del estudiante a buscar.
- Retorna: El estudiante encontrado o `null` si no se encuentra ningún estudiante con ese carnet.

14. `public void actualizarEstudiante(Estudiante estudianteActualizado)`

- Descripción: Actualiza los datos de un estudiante en la base de datos.
- Parámetros:
  - `estudianteActualizado`: Estudiante con los datos actualizados.

15. `public LocalDate obtenerFechaActual()`

- Descripción: Obtiene la fecha actual del sistema.
- Retorna: La fecha actual.

16. `public void restarUnaCopia(Libro libro)`

- Descripción: Decrementa en uno la cantidad de copias disponibles de un libro.
- Parámetros:
  - `libro`: Libro al que se le resta una copia.

17. `public void sumarUnaCopia(Libro libro)`

- Descripción: Incrementa en uno la cantidad de copias disponibles de un libro.
- Parámetros:
  - `libro`: Libro al que se le suma una copia.

18. `public void devolucionDeLibro(String codigoLibro, int carnetEstudiante)`
    - Descripción: Gestiona la devolución de un libro por parte de un estudiante.
    - Parámetros:
        - `codigoLibro`: Código del libro que se devuelve.
        - `carnetEstudiante`: Carnet del estudiante que devuelve el libro.
19. `public Prestamo obtenerPrestamo(String codigoLibro, int carnetEstudiante)`
    - Descripción: Obtiene un préstamo específico según el código del libro y el carnet del estudiante.
    - Parámetros:
        - `codigoLibro`: Código del libro del préstamo a buscar.
        - `carnetEstudiante`: Carnet del estudiante del préstamo a buscar.
    - Retorna: El préstamo encontrado o `null` si no se encuentra ningún préstamo.

Atributos:

1. `public static ArrayList<Prestamo> listaPrestamos`
    - Descripción: Lista que almacena todos los préstamos realizados.
2. `public static ArrayList<Prestamo> listaDevoluciones`
    - Descripción: Lista que almacena todas las devoluciones realizadas.
3. `public static ArrayList<Libro> listaLibros`
    - Descripción: Lista que almacena todos los libros registrados en el sistema.
4. `public static ArrayList<Estudiante> listaEstudiantes`
    - Descripción: Lista que almacena todos los estudiantes registrados en el sistema.
5. `private static String pathCarpeta`
    - Descripción: Ruta de la carpeta donde se almacenan los archivos serializables.
6. `private Map<String, Integer> librosPrestadosPorEstudiante`
    - Descripción: Mapa que almacena la cantidad de libros prestados por cada estudiante.

Autores: Michael y Alex.

Clase: Libro

Descripción: Esta clase representa un libro en el sistema de gestión de préstamos de la biblioteca. Almacena información como título, autor, código, cantidad de copias disponibles, fecha de publicación y editorial del libro.

Clase: Main

Descripción: Esta clase es la clase principal de la aplicación. Se encarga de iniciar la interfaz de usuario de la aplicación llamando al método `setVisible(true)` en una instancia de la clase `Principal`.

Paquete: backend.principal

Métodos:

1. `public static void main(String[] args)`
    - Descripción: Método principal de la aplicación. Crea una instancia de la clase `Principal` y la hace visible.
    - Parámetros:
        - `args`: Argumentos de línea de comandos (no utilizados).

Atributos:

- No hay atributos en esta clase.

Dependencias Externas:

- frontend.Principal: Se requiere la clase `Principal` del paquete `frontend` para iniciar la interfaz de usuario.

Autor: Michael.

Clase: Prestamo

Descripción: Esta clase representa un préstamo realizado en el sistema de gestión de préstamos de la biblioteca. Almacena información sobre el libro prestado, el estudiante que realiza el préstamo, las fechas de préstamo y devolución, así como detalles sobre moras y costos asociados.

Paquete: backend.principal

Métodos:

1. `public Prestamo(Libro libro, Estudiante estudiante, LocalDate fechaPrestamo, LocalDate fechaDevolucion, int monto)`
    - Descripción: Constructor de la clase Prestamo. Inicializa un nuevo préstamo con los parámetros especificados.
    - Parámetros:
        - `libro`: Libro prestado.
        - `estudiante`: Estudiante que realiza el préstamo.
        - `fechaPrestamo`: Fecha en que se realiza el préstamo.
        - `fechaDevolucion`: Fecha prevista de devolución del libro.
        - `monto`: Monto asociado al préstamo.
2. `public Libro getLibro()`

- Descripción: Obtiene el libro prestado.
- Retorna: El libro prestado.

3. `public void setLibro(Libro libro)`
    - Descripción: Establece el libro prestado.
    - Parámetros:
        - `libro`: Nuevo libro prestado.
4. `public Estudiante getEstudiante()`
    - Descripción: Obtiene el estudiante que realiza el préstamo.
    - Retorna: El estudiante que realiza el préstamo.
5. `public void setEstudiante(Estudiante estudiante)`
    - Descripción: Establece el estudiante que realiza el préstamo.
    - Parámetros:
        - `estudiante`: Nuevo estudiante que realiza el préstamo.
6. `public LocalDate getFechaPrestamo()`
    - Descripción: Obtiene la fecha de préstamo.
    - Retorna: La fecha de préstamo.
7. `public void setFechaPrestamo(LocalDate fechaPrestamo)`
    - Descripción: Establece la fecha de préstamo.
    - Parámetros:
        - `fechaPrestamo`: Nueva fecha de préstamo.
8. `public LocalDate getFechaDevolucion()`
    - Descripción: Obtiene la fecha prevista de devolución.
    - Retorna: La fecha prevista de devolución.
9. `public void setFechaDevolucion(LocalDate fechaDevolucion)`
    - Descripción: Establece la fecha prevista de devolución.
    - Parámetros:
        - `fechaDevolucion`: Nueva fecha prevista de devolución.
10. `public int getDiasConMora()`
    - Descripción: Obtiene la cantidad de días de mora en el préstamo.
    - Retorna: La cantidad de días de mora.
11. `public void setDiasConMora(int diasConMora)`
    - Descripción: Establece la cantidad de días de mora en el préstamo.
    - Parámetros:
        - `diasConMora`: Nueva cantidad de días de mora.
12. `public int getCostoPorMora()`
    - Descripción: Obtiene el costo por mora asociado al préstamo.
    - Retorna: El costo por mora.

13. `public void setCostoPorMora(int costoPorMora)`
    - Descripción: Establece el costo por mora asociado al préstamo.
    - Parámetros:
        - `costoPorMora`: Nuevo costo por mora.
14. `public int getMonto()`
    - Descripción: Obtiene el monto asociado al préstamo.
    - Retorna: El monto asociado al préstamo.
15. `public void setMonto(int monto)`
    - Descripción: Establece el monto asociado al préstamo.
    - Parámetros:
        - `monto`: Nuevo monto asociado al préstamo.

Atributos:

1. `private Libro libro`
    - Descripción: Libro prestado.
2. `private Estudiante estudiante`
    - Descripción: Estudiante que realiza el préstamo.
3. `private LocalDate fechaPrestamo`
    - Descripción: Fecha de préstamo.
4. `private LocalDate fechaDevolucion`
    - Descripción: Fecha prevista de devolución.
5. `private int diasConMora`
    - Descripción: Cantidad de días de mora en el préstamo.
6. `private int costoPorMora`
    - Descripción: Costo por mora asociado al préstamo.
7. `private int monto`
    - Descripción: Monto asociado al préstamo.

Dependencias Externas:

- Libro: Se requiere la clase `Libro` del mismo paquete para representar el libro prestado.
- Estudiante: Se requiere la clase `Estudiante` del mismo paquete para representar al estudiante que realiza el préstamo.

Autor: Michael.

Paquete backend/reportes

Clase: CalculoGanancia

Descripción: Esta clase representa un cálculo de ganancia asociado a un período de tiempo, definido por una fecha de inicio y una fecha de entrega. Se utiliza para generar informes relacionados con las ganancias durante un determinado intervalo de tiempo.

Paquete: backend.reportes

Métodos:

1. `public CalculoGanancia(LocalDate fechaInicio, LocalDate fechaEntrega)`
    - Descripción: Constructor de la clase CalculoGanancia. Inicializa un nuevo cálculo de ganancia con las fechas de inicio y entrega especificadas.
    - Parámetros:
        - `fechaInicio`: Fecha de inicio del período de cálculo.
        - `fechaEntrega`: Fecha de entrega del período de cálculo.
2. `public LocalDate getFechaInicio()`
    - Descripción: Obtiene la fecha de inicio del período de cálculo.
    - Retorna: La fecha de inicio del período de cálculo.
3. `public void setFechaInicio(LocalDate fechaInicio)`
    - Descripción: Establece la fecha de inicio del período de cálculo.
    - Parámetros:
        - `fechaInicio`: Nueva fecha de inicio del período de cálculo.
4. `public LocalDate getFechaEntrega()`
    - Descripción: Obtiene la fecha de entrega del período de cálculo.
    - Retorna: La fecha de entrega del período de cálculo.
5. `public void setFechaEntrega(LocalDate fechaEntrega)`
    - Descripción: Establece la fecha de entrega del período de cálculo.
    - Parámetros:
        - `fechaEntrega`: Nueva fecha de entrega del período de cálculo.

Atributos:

1. `LocalDate fechaInicio`
    - Descripción: Fecha de inicio del período de cálculo.
2. `LocalDate fechaEntrega`
    - Descripción: Fecha de entrega del período de cálculo.

Autor: Michael.

Clase: MoraPorDevolucionTarde

Descripción: Esta clase representa el cálculo de mora asociado a la devolución tardía de préstamos, definido por una fecha de inicio y una fecha de entrega. Se utiliza para generar informes relacionados con las moras por devolución tardía durante un período de tiempo determinado.

Paquete: backend.reportes

Métodos:

1. `public MoraPorDevolucionTarde(LocalDate fechaInicio, LocalDate fechaEntrega)`
    - Descripción: Constructor de la clase MoraPorDevolucionTarde. Inicializa un nuevo cálculo de mora por devolución tardía con las fechas de inicio y entrega especificadas.
    - Parámetros:
        - `fechaInicio`: Fecha de inicio del período de cálculo.
        - `fechaEntrega`: Fecha de entrega del período de cálculo.
2. `public LocalDate getFechaInicio()`
    - Descripción: Obtiene la fecha de inicio del período de cálculo.
    - Retorna: La fecha de inicio del período de cálculo.
3. `public void setFechaInicio(LocalDate fechaInicio)`
    - Descripción: Establece la fecha de inicio del período de cálculo.
    - Parámetros:
        - `fechaInicio`: Nueva fecha de inicio del período de cálculo.
4. `public LocalDate getFechaEntrega()`
    - Descripción: Obtiene la fecha de entrega del período de cálculo.
    - Retorna: La fecha de entrega del período de cálculo.
5. `public void setFechaEntrega(LocalDate fechaEntrega)`
    - Descripción: Establece la fecha de entrega del período de cálculo.
    - Parámetros:
        - `fechaEntrega`: Nueva fecha de entrega del período de cálculo.

Atributos:

1. `LocalDate fechaInicio`
    - Descripción: Fecha de inicio del período de cálculo.
2. `LocalDate fechaEntrega`
    - Descripción: Fecha de entrega del período de cálculo.

Autor: Michael

Paquete frontend/principal

Clase: Principal

Descripción: La clase Principal representa la interfaz principal de la aplicación de gestión de la biblioteca. Esta clase contiene funcionalidades para mostrar menús, gestionar eventos de usuario y controlar el flujo de la aplicación. A continuación, se detallan los aspectos técnicos de esta clase:

Atributos:

1. imagenFondo: Una imagen que se utiliza como fondo para la interfaz de usuario.
2. relojLabel: Etiqueta para mostrar la hora actual.
3. contenedorPanel: Panel principal que contiene los distintos componentes de la interfaz.
4. estudianteNuevo: Instancia de la clase RegistroEstudianteNuevo para gestionar el registro de nuevos estudiantes.
5. libroNuevo: Instancia de la clase RegistroLibroNuevo para gestionar el registro de nuevos libros.
6. newStudents: Instancia de la clase RegistroEstudianteNuevo para gestionar el registro de nuevos estudiantes.
7. newPrestamo: Instancia de la clase HacerPrestamo para gestionar el préstamo de libros.
8. newEditarLibro: Instancia de la clase EditarLibro para gestionar la edición de información de libros.
9. newEditarEstudiante: Instancia de la clase EditarEstudiante para gestionar la edición de información de estudiantes.
10. prestamosVigPorestudiante: Instancia de la clase PrestamosVigentesPorEstudiante para mostrar los préstamos vigentes por estudiante.
11. devolucionesConcluidas: Instancia de la clase DevolucionesConcluidas para mostrar las devoluciones concluidas.
12. relojActivo: Booleano que indica si el reloj está activo o no.
13. tamañoPanelFondo: Tamaño del panel de fondo de la interfaz.
14. titulo: Título de la ventana de la aplicación.

Métodos:

1. initUI(): Método privado que inicializa la interfaz de usuario.
2. addMenus(JMenuBar menuPrincipal): Método privado que agrega los distintos menús a la barra de menús.
3. pintarPanel(Component panel): Método privado que cambia el panel principal de la interfaz.
4. addMessageLabel(JMenuBar menuPrincipal): Método privado que agrega una etiqueta de mensaje a la barra de menús.
5. addClockLabel(JMenuBar menuPrincipal): Método privado que agrega una etiqueta de reloj a la barra de menús.
6. iniciarReloj(): Método privado que inicia un hilo para actualizar el reloj en la interfaz.
7. actualizarReloj(): Método privado que actualiza la etiqueta de reloj con la hora actual.
8. initComponents(): Método generado automáticamente que inicializa los componentes de la interfaz.

Constructores:

1. Principal(): Constructor que inicializa la interfaz de usuario y configura la ventana principal de la aplicación.

Otros:

La clase también contiene los métodos de gestión de eventos para los distintos elementos de menú, como la importación de registros, la creación de nuevos estudiantes y libros, la edición de información, la realización de préstamos, etc.

Autor: Alex

Paquete frontend/registrosinformacionnueva/DevolucionLibro

Clase: DevolucionLibro

Descripción: la clase DevolucionLibro representa un panel en la interfaz de usuario utilizado para realizar la devolución de libros prestados por estudiantes. Esta clase permite verificar la información de un préstamo, calcular multas por devolución tardía y finalizar la transacción de devolución. A continuación, se detallan los aspectos técnicos de esta clase:

Atributos:

1. app: Instancia de la clase FuncionamientoAplicacion para interactuar con la lógica de la aplicación.
2. libro: Instancia de la clase Libro para representar el libro asociado al préstamo.
3. estudiante: Instancia de la clase Estudiante para representar el estudiante asociado al préstamo.
4. carnetLabel: Etiqueta para ingresar el carné del estudiante.
5. codigoLibroLabel: Etiqueta para ingresar el código del libro.
6. fechaDevolucionLabel: Etiqueta para ingresar la fecha de devolución.
7. informacionLabel: Etiqueta para mostrar la información del préstamo.
8. carnetText: Campo de texto para ingresar el carné del estudiante.
9. codigoLibroText: Campo de texto para ingresar el código del libro.
10. fechaDevolucionText: Campo de texto para ingresar la fecha de devolución.
11. informacionTextArea: Área de texto para mostrar la información del préstamo.
12. verificarPrestamoButton: Botón para verificar la existencia del préstamo.
13. finalizarTransaccionButton: Botón para finalizar la transacción de devolución.
14. codigoLibro: Código del libro asociado al préstamo.

Métodos:

1. initComponents(): Método privado que inicializa los componentes de la interfaz de usuario.
2. verificarPrestamo(): Método privado que verifica la existencia del préstamo y muestra su información.
3. finalizarTransaccion(): Método privado que finaliza la transacción de devolución y agrega detalles de devolución.
4. limpiarCampos(): Método privado que limpia los campos de entrada y el área de texto.
5. setFontStyle(): Método privado que establece el estilo de fuente para los componentes de la interfaz.

Constructores:

1. DevolucionLibro(): Constructor que inicializa el panel y sus componentes.

Autor: Alex

Paquete frontend/registrosinformacionnueva/EditarEstudiante

Clase: EditarEstudiante

Descripción: la clase EditarEstudiante representa un panel en la interfaz de usuario utilizado para editar la información de un estudiante almacenado en el sistema. Esta clase permite buscar un estudiante por su código, mostrar su información, realizar cambios en su información y guardar los cambios realizados. A continuación, se detallan los aspectos técnicos de esta clase:

Atributos:

1. app: Instancia de la clase FuncionamientoAplicacion para interactuar con la lógica de la aplicación.
2. codigoEstudianteText: Campo de texto para ingresar el código del estudiante.
3. informacionEstudianteTextArea: Área de texto para mostrar la información del estudiante.
4. jLabel1: Etiqueta para el título de la sección.
5. jLabel2: Etiqueta para indicar el campo de código del estudiante.
6. jScrollPane1: Panel de desplazamiento para el área de texto.

Métodos:

1. initComponents(): Método privado que inicializa los componentes de la interfaz de usuario.
2. buscarEstudianteButtonActionPerformed(ActionEvent evt): Método privado que maneja el evento de búsqueda de un estudiante.
3. cambiosButtonActionPerformed(ActionEvent evt): Método privado que maneja el evento de realizar cambios en la información del estudiante.
4. guardarCambiosConfirmarButtonActionPerformed(ActionEvent evt): Método privado que maneja el evento de guardar los cambios realizados.
5. limpiarCampos(): Método privado que limpia los campos de entrada y el área de texto.

Constructores:

1. EditarEstudiante(): Constructor que inicializa el panel y sus componentes.

Autor: Alex

Paquete frontend/registrosinformacionnueva/EditarLibro

Clase: EditarLibro

Descripción: la clase EditarLibro representa un panel en la interfaz de usuario utilizado para editar la información de un libro almacenado en el sistema. Esta clase permite buscar un libro por su código,

mostrar su información, realizar cambios en su información y guardar los cambios realizados. A continuación, se detallan los aspectos técnicos de esta clase:

Atributos:

1. app: Instancia de la clase FuncionamientoAplicacion para interactuar con la lógica de la aplicación.
2. codigoLibroText: Campo de texto para ingresar el código del libro.
3. informacionLibroTextArea: Área de texto para mostrar la información del libro.
4. jLabel1: Etiqueta para el título de la sección.
5. jLabel2: Etiqueta para indicar el campo de código del libro.
6. jScrollPane1: Panel de desplazamiento para el área de texto.

Métodos:

1. initComponents(): Método privado que inicializa los componentes de la interfaz de usuario.
2. buscarLibroButtonActionPerformed(ActionEvent evt): Método privado que maneja el evento de búsqueda de un libro.
3. cambiosButtonActionPerformed(ActionEvent evt): Método privado que maneja el evento de realizar cambios en la información del libro.
4. guardarCambiosConfirmarButtonActionPerformed(ActionEvent evt): Método privado que maneja el evento de guardar los cambios realizados.
5. limpiarCampos(): Método privado que limpia los campos de entrada y el área de texto.

Constructores:

1. EditarLibro(): Constructor que inicializa el panel y sus componentes.

Autor: Alex

Paquete frontend/registrosinformacionnueva/HacerPrestamo

Clase: HacerPrestamo
Descripción: la clase HacerPrestamo representa un panel en la interfaz de usuario utilizado para registrar un nuevo préstamo de un libro a un estudiante. Esta clase permite buscar un libro por su código, mostrar su información, ingresar el carné del estudiante, seleccionar la fecha de devolución y realizar el préstamo. A continuación, se detallan los aspectos técnicos de esta clase:

Atributos:

1. app: Instancia de la clase FuncionamientoAplicacion para interactuar con la lógica de la aplicación.
2. libro: Objeto Libro que representa el libro seleccionado para el préstamo.
3. fechaDevolucionBox: ComboBox que permite seleccionar la fecha de devolución del libro.
4. monto: Variable que almacena el monto a pagar por el préstamo del libro.

Métodos:

1. initComponents(): Método privado que inicializa los componentes de la interfaz de usuario.

2. buscarLibroButtonActionPerformed(ActionEvent evt): Método privado que maneja el evento de búsqueda de un libro.
3. realizarPrestamoButtonActionPerformed(ActionEvent evt): Método privado que maneja el evento de realizar el préstamo.
4. agregarFechasLimiteComboBox(): Método privado que agrega las fechas límite al ComboBox de selección de fecha de devolución.
5. limpiarCampos(): Método privado que limpia los campos de entrada y el área de texto.

Constructores:

1. HacerPrestamo(): Constructor que inicializa el panel y sus componentes.

Autor: Alex

Paquete frontend/registrosinformacionnueva/ListadoEstudiantes

Clase: ListadoEstudiantes

Descripción: la clase ListadoEstudiantes es un panel de la interfaz de usuario que muestra una tabla con la lista de estudiantes registrados en el sistema. Permite buscar estudiantes por su nombre o carné, filtrando los resultados en tiempo real, y ajusta automáticamente el ancho de las columnas de la tabla para adaptarse al contenido. A continuación, se detallan los aspectos técnicos de esta clase:

Atributos:

1. textFieldBusqueda: Campo de texto utilizado para ingresar el criterio de búsqueda.
2. rowSorter: Clasificador de filas utilizado para filtrar la tabla de estudiantes.

Métodos:

1. initComponents(): Método privado que inicializa los componentes de la interfaz de usuario.
2. agregarCampoBusqueda(): Método privado que agrega el campo de búsqueda a la parte superior del panel.
3. filtrarTabla(String texto): Método privado que filtra la tabla de estudiantes según el texto ingresado en el campo de búsqueda.
4. actualizarTablaEstudiantes(): Método público que actualiza la tabla de estudiantes con la información actualizada del sistema.
5. ajustarColumnaTexto(): Método privado que ajusta el ancho de las columnas de la tabla para adaptarse al contenido.
6. nombreCarrera(int codigoCarrera): Método privado que devuelve el nombre de la carrera según su código.

Constructores:

1. ListadoEstudiantes(): Constructor que inicializa el panel y sus componentes, y carga la información de los estudiantes en la tabla.

Autor: Michael y Alex

Paquete frontend/registrosinformacionnueva/ListadoLibros

Clase: ListadoLibros

Descripcion: la clase ListadoLibros es un panel de la interfaz de usuario que muestra una tabla con la lista de libros registrados en el sistema. Permite buscar libros por su título, autor, editorial o código, filtrando los resultados en tiempo real, y ajusta automáticamente el ancho de las columnas de la tabla para adaptarse al contenido.

Atributos:

1. textFieldBusqueda: Campo de texto utilizado para ingresar el criterio de búsqueda.
2. rowSorter: Clasificador de filas utilizado para filtrar la tabla de libros.

Métodos:

1. initComponents(): Método privado que inicializa los componentes de la interfaz de usuario.
2. agregarCampoBusqueda(): Método privado que agrega el campo de búsqueda a la parte superior del panel.
3. filtrarTabla(String texto): Método privado que filtra la tabla de libros según el texto ingresado en el campo de búsqueda.
4. actualizarTablaLibros(): Método público que actualiza la tabla de libros con la información actualizada del sistema.
5. ajustarColumnaTexto(): Método privado que ajusta el ancho de las columnas de la tabla para adaptarse al contenido.

Constructores:

1. ListadoLibros(): Constructor que inicializa el panel y sus componentes, y carga la información de los libros en la tabla.

Autor: Michael y Alex

Paquete frontend/registrosinformacionnueva/ListadoPrestamos

Clase: ListadoPrestamos

Descripción: la clase ListadoPrestamos es un panel de la interfaz de usuario que muestra una tabla con la lista de préstamos registrados en el sistema. Permite buscar préstamos por el código del libro, carné del estudiante, título del libro, nombre del estudiante, filtrando los resultados en tiempo real, y ajusta automáticamente el ancho de las columnas de la tabla para adaptarse al contenido.

Atributos:

1. textFieldBusqueda: Campo de texto utilizado para ingresar el criterio de búsqueda.
2. rowSorter: Clasificador de filas utilizado para filtrar la tabla de préstamos.

Métodos:

1. initComponents(): Método privado que inicializa los componentes de la interfaz de usuario.
2. agregarCampoBusqueda(): Método privado que agrega el campo de búsqueda a la parte superior del panel.
3. filtrarTabla(String texto): Método privado que filtra la tabla de préstamos según el texto ingresado en el campo de búsqueda.
4. actualizarTablaPrestamos(): Método público que actualiza la tabla de préstamos con la información actualizada del sistema.
5. ajustarColumnaTexto(): Método privado que ajusta el ancho de las columnas de la tabla para adaptarse al contenido.

Constructores:

1. ListadoPrestamos(): Constructor que inicializa el panel y sus componentes, y carga la información de los préstamos en la tabla.

Autor: Alex

Paquete frontend/registrosinformacionnueva/RegistroEstudianteNuevo

Clase: RegistroEstudianteNuevo

Descripción: la clase RegistroEstudianteNuevo es un panel de la interfaz de usuario utilizado para registrar un nuevo estudiante en el sistema. Permite ingresar el carné, nombre, carrera y fecha de nacimiento del estudiante, verificando los campos obligatorios y el formato de la fecha.

Atributos:

1. app: Instancia de la clase FuncionamientoAplicacion para interactuar con el funcionamiento del sistema.

Métodos:

1. initComponents(): Método privado que inicializa los componentes de la interfaz de usuario y establece su estilo.
2. guardarEstudianteBotonActionPerformed(java.awt.event.ActionEvent evt): Método privado que se activa cuando se presiona el botón "Guardar Estudiante", realiza la validación de los campos y guarda la información del estudiante en el sistema.
3. verificarCamposObligatorios(String carnet, String nombre, String codigoCarrera): Método privado que verifica si los campos obligatorios (carné, nombre y código de carrera) están llenos.
4. verificarFormatoFecha(String texto): Método privado que verifica si la fecha tiene el formato correcto (yyyy-MM-dd).
5. limpiarCampos(): Método privado que limpia los campos de entrada después de guardar un estudiante.

Constructores:

1. RegistroEstudianteNuevo(): Constructor que inicializa el panel y sus componentes, establece su estilo y agrega los eventos necesarios.

Variables de declaración:

1. carnetText: Campo de texto para ingresar el carné del estudiante.
2. codigoCarreraComboBox: ComboBox para seleccionar la carrera del estudiante.
3. fechaText: Campo de texto para ingresar la fecha de nacimiento del estudiante.
4. guardarEstudianteBoton: Botón para guardar la información del estudiante.
5. jLabel1, jLabel2, jLabel3, jLabel4, jLabel5: Etiquetas para mostrar texto informativo en la interfaz de usuario.
6. nombreText: Campo de texto para ingresar el nombre del estudiante.

Autor: Alex

Paquete frontend/registrosinformacionnueva/RegistroLibroNuevo

Clase: RegistroLibroNuevo

Descripcion: la clase RegistroLibroNuevo es un panel de la interfaz de usuario utilizado para registrar un nuevo libro en el sistema. Permite ingresar el código, autor, título, cantidad de copias, fecha de publicación y editorial del libro, verificando los campos obligatorios y el formato de la fecha y el código.

Atributos:

1. app: Instancia de la clase FuncionamientoAplicacion para interactuar con el funcionamiento del sistema.

Métodos:

1. initComponents(): Método privado que inicializa los componentes de la interfaz de usuario y establece su estilo.
2. guardarLibroBotonActionPerformed(java.awt.event.ActionEvent evt): Método privado que se activa cuando se presiona el botón "Guardar Libro", realiza la validación de los campos y guarda la información del libro en el sistema.
3. verificarCamposObligatorios(): Método privado que verifica si todos los campos obligatorios están llenos.
4. validarCodigo(String codigo): Método privado que valida el formato del código del libro.
5. validarNumero(String texto): Método privado que valida si un texto ingresado es un número.
6. verificarFormatoFecha(String texto): Método privado que verifica si el texto ingresado tiene el formato correcto de fecha.
7. limpiarCampos(): Método privado que limpia los campos de entrada después de guardar un libro.
8. obtenerFecha(String fechaEnString): Método privado que convierte un string en formato de fecha yyyy-MM-dd a un objeto LocalDate.

Variables de declaración:

1. autorLibroText, codigoLibroText, copiasText, editorialText, fechaText, guardarLibroBoton, tituloLibroText: Campos de texto y botón para ingresar la información del libro.
2. jLabel1, jLabel2, jLabel3, jLabel4, jLabel5, jLabel6, jLabel7: Etiquetas para mostrar texto informativo en la interfaz de usuario.

Autor: Alex

Paquete frontend/reportes

Clase: DevolucionDiaEnCurso

Descripción: Esta clase representa un panel en la interfaz de usuario que muestra las devoluciones programadas para el día en curso. Se actualiza dinámicamente con los préstamos registrados para el día actual.

Paquete: frontend.reportes

Constructores:

1. `public DevolucionDiaEnCurso()`
    - Descripción: Constructor de la clase DevolucionDiaEnCurso. Inicializa el panel y actualiza la tabla con las devoluciones del día actual.

Métodos:

1. `public void actualizarTablaEntregasHoy()`
    - Descripción: Actualiza la tabla de devoluciones del día en curso con los datos de los préstamos registrados para el día actual.
2. `private void ajustarColumnaTexto()`
    - Descripción: Ajusta el ancho de las columnas de la tabla para que se ajusten correctamente al contenido.

Variables de instancia:

1. `javax.swing.JTable tablaReportes`
    - Descripción: Tabla utilizada para mostrar los datos de las devoluciones del día en curso.
2. `javax.swing.JScrollPane jScrollPane1`
    - Descripción: Panel de desplazamiento utilizado para contener la tabla.

Autor: Michael.

Clase: DevolucionesConcluidas

Descripción: Esta clase representa un panel en la interfaz de usuario que muestra las devoluciones ya concluidas. Se actualiza dinámicamente con los préstamos que ya han sido devueltos.

Paquete: frontend.reportes

Constructores:

1. `public DevolucionesConcluidas()`
    - Descripción: Constructor de la clase DevolucionesConcluidas. Inicializa el panel y actualiza la tabla con las devoluciones concluidas.

Métodos:

1. `public void actualizarTablaDevolucionesHechas()`
    - Descripción: Actualiza la tabla de devoluciones concluidas con los datos de los préstamos ya devueltos.
2. `private void ajustarColumnaTexto()`
    - Descripción: Ajusta el ancho de las columnas de la tabla para que se ajusten correctamente al contenido.

Variables de instancia:

1. `javax.swing.JTable tablaReportes`
    - Descripción: Tabla utilizada para mostrar los datos de las devoluciones concluidas.
2. `javax.swing.JScrollPane jScrollPane2`
    - Descripción: Panel de desplazamiento utilizado para contener la tabla.
3. `javax.swing.JScrollPane jScrollPane1`
    - Descripción: Panel de desplazamiento utilizado para contener el árbol de visualización (que no se utiliza en esta clase).
4. `javax.swing.JTree jTree1`
    - Descripción: Árbol de visualización (que no se utiliza en esta clase).

Autor: Michael

Clase: DineroRecaudado

Descripción: Esta clase representa un panel en la interfaz de usuario que muestra el dinero recaudado por préstamos en un intervalo de tiempo específico. Actualiza dinámicamente la tabla con los datos de los préstamos realizados en ese intervalo de tiempo.

Paquete: frontend.reportes

Constructores:

1. `public DineroRecaudado()`
    - Descripción: Constructor de la clase DineroRecaudado. Inicializa el panel y actualiza la tabla con los préstamos realizados en un intervalo de tiempo específico.

Métodos:

1. `public void actualizarTablaPrestamosPorTiempo()`
    - Descripción: Actualiza la tabla con los datos de los préstamos realizados en un intervalo de tiempo específico.
2. `private void ajustarColumnaTexto()`
    - Descripción: Ajusta el ancho de las columnas de la tabla para que se ajusten correctamente al contenido.

Variables de instancia:

1. `javax.swing.JTable tablaReportes`
    - Descripción: Tabla utilizada para mostrar los datos de los préstamos por intervalo de tiempo.
2. `javax.swing.JScrollPane jScrollPane1`
    - Descripción: Panel de desplazamiento utilizado para contener la tabla.

Autor: Michael

Clase: PrestamosConMora

Descripción: Esta clase representa un panel en la interfaz de usuario que muestra los préstamos que tienen mora. Actualiza dinámicamente la tabla con los datos de los préstamos que tienen mora, incluyendo información sobre los días de atraso y la mora correspondiente.

Paquete: frontend.reportes

Constructores:

1. `public PrestamosConMora()`
    - Descripción: Constructor de la clase PrestamosConMora. Inicializa el panel y actualiza la tabla con los préstamos que tienen mora.

Métodos:

1. `public void actualizarTablaEntregasConMora()`
    - Descripción: Actualiza la tabla con los datos de los préstamos que tienen mora.
2. `private void ajustarColumnaTexto()`
    - Descripción: Ajusta el ancho de las columnas de la tabla para que se ajusten correctamente al contenido.

Variables de instancia:

1. `javax.swing.JTable tablaReportes`
    - Descripción: Tabla utilizada para mostrar los datos de los préstamos con mora.
2. `javax.swing.JScrollPane jScrollPane1`

- Descripción: Panel de desplazamiento utilizado para contener la tabla.

Autor: Michael

Clase: PrestamosPorCarrera

Descripción: Esta clase representa un panel en la interfaz de usuario que muestra los préstamos organizados por carrera. Permite filtrar los préstamos por una carrera específica utilizando un ComboBox. Actualiza dinámicamente la tabla con los datos de los préstamos según la carrera seleccionada.

Paquete: frontend.reportes

Constructores:

1. `public PrestamosPorCarrera()`
    - Descripción: Constructor de la clase PrestamosPorCarrera. Inicializa el panel y actualiza la tabla con los préstamos organizados por carrera.

Métodos:

1. `public void actualizarTablaPrestamosPorCarrera()`
    - Descripción: Actualiza la tabla con los datos de los préstamos organizados por carrera.
2. `private String nombreCarrera(int codigoCarrera)`
    - Descripción: Devuelve el nombre de la carrera según su código.
3. `private void ajustarColumnaTexto()`
    - Descripción: Ajusta el ancho de las columnas de la tabla para que se ajusten correctamente al contenido.
4. `private void filtroCarreraBoxActionPerformed(java.awt.event.ActionEvent evt)`
    - Descripción: Maneja el evento de selección de carrera en el ComboBox de filtrado. Aplica un filtro a la tabla según la carrera seleccionada.

Variables de instancia:

1. `javax.swing.JTable tablaReportes`
    - Descripción: Tabla utilizada para mostrar los datos de los préstamos por carrera.
2. `javax.swing.JComboBox<String> filtroCarreraBox`
    - Descripción: ComboBox utilizado para seleccionar la carrera por la cual filtrar los préstamos.
3. `javax.swing.JScrollPane jScrollPane1`
    - Descripción: Panel de desplazamiento utilizado para contener la tabla.

Autor: Michael

Paquete frontend/reporte/PrestamosPorCarreraEnIntervalo

Clase: PrestamosPorCarreraEnIntervalo:

Descripción: la clase PrestamosPorCarreraEnIntervalo es un panel de la interfaz de usuario utilizado para mostrar un reporte de préstamos por carrera dentro de un intervalo de fechas. Permite filtrar los préstamos por fecha de inicio, fecha final y carrera.

Atributos:

1. textFieldFechaInicio, textFieldFechaFin: Campos de texto para ingresar la fecha de inicio y la fecha final del intervalo.
2. rowSorter: Clasificador de filas utilizado para filtrar la tabla de reportes.
3. filtroCarreraBox: ComboBox para seleccionar la carrera por la cual filtrar los préstamos.
4. jScrollPane1: Panel de desplazamiento que contiene la tabla de reportes.
5. tablaReportes: Tabla que muestra los reportes de préstamos por carrera.

Métodos:

1. initComponents(): Método privado que inicializa los componentes de la interfaz de usuario y establece sus escuchadores de eventos.
2. filtroCarreraBoxActionPerformed(java.awt.event.ActionEvent evt): Método privado que se activa cuando se selecciona una carrera en el ComboBox de filtrado, y actualiza el filtro de la tabla.
3. actualizarTablaPrestamosPorCarrera(): Método público que actualiza la tabla de reportes con los préstamos por carrera.
4. nombreCarrera(int codigoCarrera): Método privado que devuelve el nombre de una carrera dado su código.
5. ajustarColumnaTexto(): Método privado que ajusta el ancho de las columnas de la tabla según el contenido.
6. filtrarTabla(): Método privado que filtra los préstamos en la tabla según la fecha y la carrera seleccionadas.

Variables de declaración:

1. labelFechaInicio, labelFechaFin: Etiquetas para mostrar texto informativo en la interfaz de usuario.
2. panelBusqueda, panelComboBox: Paneles para organizar los campos de búsqueda y el ComboBox de filtrado en la interfaz de usuario.
3. modelo: Modelo de tabla utilizado para mostrar los préstamos por carrera en la tabla de reportes.

Autor: Alex

Paquete frontend/reporte/PrestamosPorUnEstudiante

Clase: PrestamosPorUnEstudiante

Descripción: la clase PrestamosPorUnEstudiante es un panel de la interfaz de usuario utilizado para mostrar un reporte de préstamos realizados por un estudiante en particular. Permite filtrar los préstamos por el nombre o el carnet del estudiante.

Atributos:

1. textFieldBusqueda: Campo de texto para ingresar el nombre o carnet del estudiante y filtrar los préstamos.
2. rowSorter: Clasificador de filas utilizado para filtrar la tabla de reportes.

Métodos:

1. initComponents(): Método privado que inicializa los componentes de la interfaz de usuario, como la tabla de reportes.
2. actualizarTablaIngresosIntervaloTiempo(): Método público que actualiza la tabla de reportes con los préstamos realizados por el estudiante.
3. ajustarColumnaTexto(): Método privado que ajusta el ancho de las columnas de la tabla según el contenido.
4. agregarCampoBusqueda(): Método privado que agrega el campo de búsqueda en la parte superior del panel.
5. filtrarTabla(String texto): Método privado que filtra los préstamos en la tabla según el texto ingresado en el campo de búsqueda.

Variables de declaración:

1. jScrollPane1: Panel de desplazamiento que contiene la tabla de reportes.
2. tablaReportes: Tabla que muestra los reportes de préstamos por estudiante.

Autor: Alex

Paquete frontend/reporte/PrestamosVigentesPorEstudiante

Clase: PrestamosVigentesPorEstudiante

Descripción: la clase PrestamosVigentesPorEstudiante es un panel de la interfaz de usuario que muestra los préstamos vigentes para un estudiante en particular. Los préstamos vigentes son aquellos cuya fecha de préstamo está entre la fecha actual y tres días después de la fecha actual.

Atributos:

1. textFieldBusqueda: Campo de texto utilizado para ingresar el nombre o carnet del estudiante y filtrar los préstamos.
2. rowSorter: Clasificador de filas utilizado para filtrar la tabla de reportes.

Métodos:

1. initComponents(): Método privado que inicializa los componentes de la interfaz de usuario, como la tabla de reportes.

2. actualizarTablaPrestamosIntervaloTiempo(): Método público que actualiza la tabla de reportes con los préstamos vigentes para el estudiante.
3. ajustarColumnaTexto(): Método privado que ajusta el ancho de las columnas de la tabla según el contenido.
4. agregarCampoBusqueda(): Método privado que agrega el campo de búsqueda en la parte superior del panel.
5. filtrarTabla(String texto): Método privado que filtra los préstamos en la tabla según el texto ingresado en el campo de búsqueda.

Variables de declaración:

1. jScrollPane1: Panel de desplazamiento que contiene la tabla de reportes.
2. tablaReportes: Tabla que muestra los reportes de préstamos vigentes por estudiante.

Autor: Alex

Paquete frontend/reporte/RecaudadoEnIntervaloTiempo

Clase RecaudadoEnIntervaloTiempo:

Descripción: la clase RecaudadoEnIntervaloTiempo es un panel de la interfaz de usuario que muestra los préstamos realizados en un intervalo de tiempo especificado por el usuario.

Atributos:

1. textFieldFechaInicio: Campo de texto para ingresar la fecha de inicio del intervalo.
2. textFieldFechaFin: Campo de texto para ingresar la fecha de fin del intervalo.
3. rowSorter: Clasificador de filas utilizado para filtrar la tabla de reportes.

Métodos:

1. initComponents(): Método privado que inicializa los componentes de la interfaz de usuario, como la tabla de reportes.
2. actualizarTablaIngresosIntervaloTiempo(): Método público que actualiza la tabla de reportes con los préstamos realizados dentro del intervalo de tiempo especificado.
3. ajustarColumnaTexto(): Método privado que ajusta el ancho de las columnas de la tabla según el contenido.
4. agregarCampoBusqueda(): Método privado que agrega los campos de búsqueda de fechas en la parte superior del panel.
5. filtrarTabla(): Método privado que filtra los préstamos en la tabla según las fechas ingresadas en los campos de búsqueda.

Variables de declaración:

1. jScrollPane1: Panel de desplazamiento que contiene la tabla de reportes.
2. tablaReportes: Tabla que muestra los reportes de préstamos realizados en el intervalo de tiempo especificado.

Autor: Alex

Paquete src/main/resources

1. images: Este directorio contiene todas las imágenes utilizadas en la aplicación, como logos, iconos, imágenes de fondo, etc.

2. configuration.properties: Este archivo contiene configuraciones específicas de la aplicación, como propiedades de conexión a una base de datos, configuraciones de interfaz de usuario, etc. Puede estar en formato de archivo de propiedades o en otro formato según las necesidades de tu aplicación.

3. other_resources: Este directorio contiene otros recursos que no son imágenes ni configuraciones, como archivos de texto, archivos XML, archivos JSON, etc. Estos recursos pueden ser necesarios para diversas funciones de la aplicación, como la carga de datos estáticos o la configuración de parámetros de la aplicación.