Universidad Rafael Landívar
Facultad de Ingeniería.
Ingeniería en Informática y Sistemas.
Lenguajes Formales y Autómatas - Sección: 02
Catedrático: Ing. Vivian Damaris Campos González.

Biblioteca Virtual

Manual Técnico

Estudiante: Tony Alexander Balán Mendoza.
Carné: 1202124

Gabriel Emilio Toyom Jimenez.
Carné: 1051524

Guatemala, 05 de septiembre de 2025.

ÍNDICE

I. RESUMEN

I. Descripción del sistema

El siguiente programa simula un sistema de gestión de préstamos de una biblioteca digital para la URL. El sistema es capaz de procesar y leer los registros de préstamos de libros desde un archivo de texto que contiene la información necesaria para generar reportes en consola y exportar los resultados en formato HTML. Así mismo, el sistema implementa un análisis léxico básico para la lectura de archivos de texto de información estructurada implementando estructura de datos que faciliten la gestión y acceso a los registros de usuarios, listado de libros prestados, historial de préstamos, etc. Además, también permite llevar las estadísticas clave del sistema como el libro y el usuario más frecuente.

II. Problema de resolución

La URL requiere de un programa que funcione desde consola para simular el sistema de gestión de préstamos de una biblioteca digital. El sistema debería ser capaz de procesar un archivo de texto que contiene múltiples registros de préstamos de libros, analizar la información y generar reportes claros.

III. Aspectos generales del sistema

- Entorno de Desarrollo: El código fuente del sistema se desarrolló en el IDE de Apache Netbeans 24. También, se utilizó Git para llevar un control de versiones para gestionar el desarrollo colaborativo y mantener el historial de cambios en el código fuente.

- Lenguaje de programación: Se implementó como lenguaje principal del sistema a Java y HTML para la generación de reportes.

- Estructuras de datos personalizadas:

    - Tablas hash: Utilizadas para acceder a los listados de libros disponibles y usuarios únicos registrados por la biblioteca.
    - Listas (ArrayList): Utilizadas para manejar los listados de préstamos realizados y préstamos vencidos.

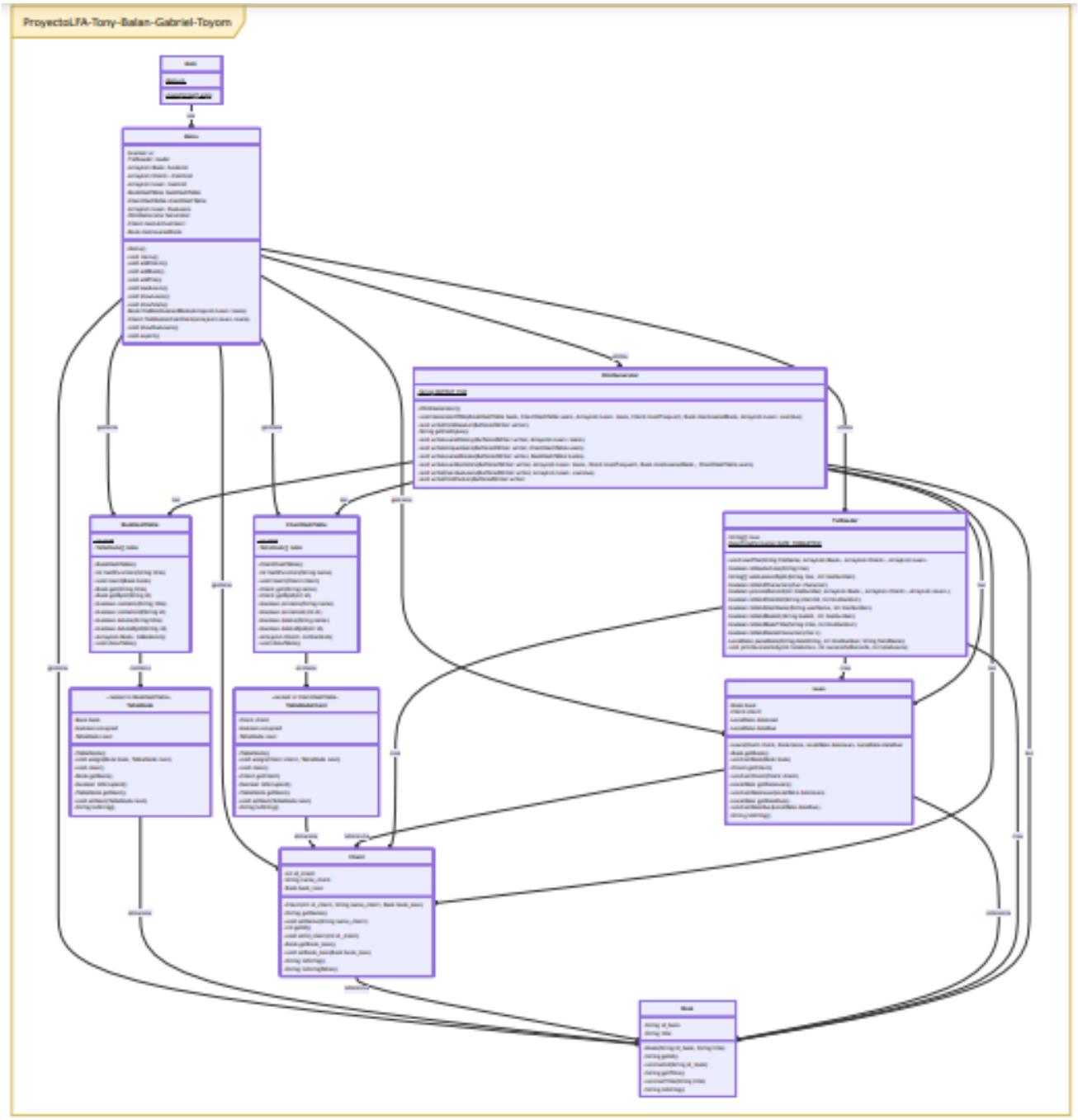
- Archivos de configuración: Se utilizó el archivo “prueba.txt” para comprobar el funcionamiento de la aplicación y realizar los cambios pertinentes para cumplir con los requerimientos del sistema.

II. ARQUITECTURA DEL SISTEMA

I. Diagrama de clases

Figura 01

Diagrama de clases del proyecto

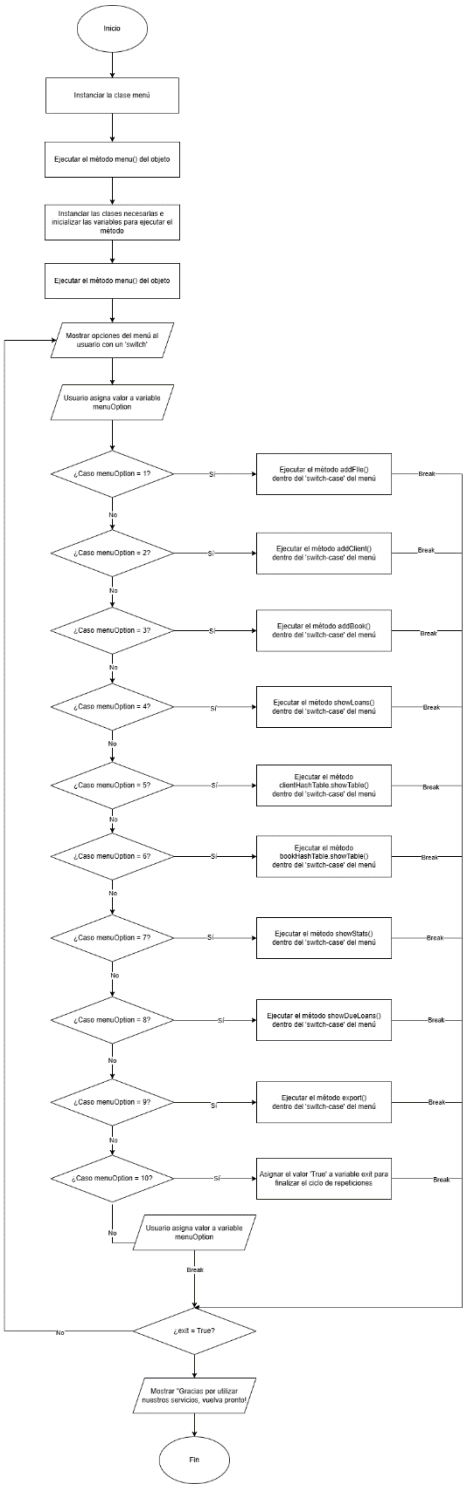


Nota. Fuente: Elaboración propia en Draw.io. https://drive.google.com/file/d/1lR-5mZOyNcgPXNO6e_gsgOSk24f_TKjN/view?usp=sharing

II. Diagrama de flujo

Figura 02

Diagrama del flujo principal del proyecto



Nota. Fuente: Elaboración propia en Draw.io. https://drive.google.com/file/d/1o-wluHOLeEc8CXHPmmd3fvg4Y6MPDRZ1/view?usp=sharing

III. Descripción de clases importantes

1. Clase Menú:

Propósito

Gestionar la interacción principal del sistema de préstamos de libros mediante un menú de opciones. Permite cargar datos, mostrar estadísticas, exportar información y manejar préstamos vencidos.

Atributos

- Scanner sc: Para capturar la entrada del usuario desde consola.
- TxtReader reader: Encargado de leer archivos .txt con registros de libros, clientes y préstamos.
- ArrayList<Book> bookList: Lista de libros cargados desde archivo.
- ArrayList<Client> clientList: Lista de clientes cargados desde archivo.
- ArrayList<Loan> loanList: Lista de préstamos cargados desde archivo.
- BookHashTable bookHashTable: Tabla hash para almacenar libros de forma eficiente.
- ClientHashTable clientHashTable: Tabla hash para almacenar clientes de forma eficiente.
- ArrayList<Loan> DueLoans: Lista de préstamos vencidos.
- HtmlGenerator Generator: Generador de archivos HTML con estadísticas y registros.
- Client mostActiveClient: Cliente con más préstamos realizados.
- Book mostLoanedBook: Libro más prestado.

Métodos

- menu(): Muestra el menú principal y gestiona la navegación entre opciones.
- addClient(): Inserta los clientes en la tabla hash si la lista no está vacía.
- addBook(): Inserta los libros en la tabla hash si la lista no está vacía.
- addFile(): Solicita el nombre del archivo y carga los datos desde el archivo .txt.
- loadLoans(): Calcula fechas de vencimiento y detecta préstamos vencidos.
- showLoans(): Muestra todos los préstamos registrados.
- showStats(): Muestra estadísticas como total de préstamos, libro más prestado y cliente más activo.
- findMostLoanedBook(ArrayList<Loan> loans): Determina el libro más prestado.
- findMostActiveClient(ArrayList<Loan> loans): Determina el cliente más activo.
- showDueLoans(): Muestra los préstamos vencidos.
- export(): Exporta los datos a un archivo HTML usando HtmlGenerator.

Notas sobre lógica interna

Primero, el método addFile() utiliza TxtReader para leer un archivo .txt y llenar las listas de libros, clientes y préstamos. Así mismo, se calcula la fecha de vencimiento como 15 días

después del préstamo. Si la fecha actual supera la fecha de vencimiento y se agrega a DueLoans evitando duplicados. Por su parte, para las estadísticas del libro más prestado y cliente más frecuente se recorren las listas para contar los préstamos por libro y por cliente. Por último, se genera un archivo HTML con todos los datos relevantes, incluyendo estadísticas y préstamos vencidos.

2. Clase TxtReader

Propósito

Leer archivos de texto que contienen registros de libros, clientes y préstamos, y cargar esa información en listas correspondientes para su posterior procesamiento en el sistema.

Atributos

Esta clase no tiene atributos definidos. Su funcionalidad se basa en el método público readFile().

Métodos

void readFile(String fileName, ArrayList<Book> bookList, ArrayList<Client> clientList, ArrayList<Loan> loanList): Lee el archivo .txt especificado y carga los datos en las listas proporcionadas. El archivo debe estar estructurado en secciones delimitadas por encabezados como: libros, clientes, préstamos.

Notas sobre lógica interna

Primero, el método readFile utiliza un BufferedReader para leer línea por línea el archivo. De este modo, se lee cada línea que se divide por comas y se convierte en un objeto correspondiente (Book, Client, Loan). Por su parte, para los préstamos que tengan una fecha de devolución como vacía se asigna ese campo como null.

3. Clase HtmlGenerator

Propósito

Generar un archivo HTML que resume la información del sistema de préstamos, incluyendo libros, clientes, préstamos, estadísticas y préstamos vencidos.

Atributos

Esta clase no tiene atributos definidos. Su funcionalidad se basa en el método público GenerateHTML().

Métodos

void GenerateHTML(BookHashTable bookHashTable, ClientHashTable clientHashTable, ArrayList<Loan> loanList, Client mostActiveClient, Book mostLoanedBook, ArrayList<Loan> DueLoans)
Genera un archivo HTML con la información proporcionada. El archivo incluye:

- Tabla de libros.
- Tabla de clientes.
- Historial de préstamos.
- Estadísticas (libro más prestado, cliente más activo).
- Préstamos vencidos.

Notas sobre lógica interna

Primero, el método define una constante de clase que almacena el nombre de salida del archivo de texto. Así mismo, utiliza etiquetas HTML básicas (<html>, <head>, <body>, <table>, etc.) para estructurar la información. Además, recorre las listas de libros, clientes y préstamos para generar filas en tablas HTML y el archivo se guarda como reporte.html en el directorio raíz del proyecto.

4. Clases BookHashTable y ClientHashTable

Propósito

Implementar una tabla hash para almacenar y acceder eficientemente a objetos de tipo Book y Client, respectivamente. Permiten insertar, mostrar y convertir los datos almacenados en listas.

Atributos

- private final int SIZE: Tamaño fijo de la tabla hash (valor: 100).

- private final ArrayList<T>[] table: Arreglo de listas que representa la tabla hash, donde T es Book o Client.

Métodos

- void insert(T item): Inserta un objeto en la tabla hash usando su id como clave. Si ya existe una lista en la posición calculada, se agrega el objeto a esa lista.

- void showTable(): Muestra todos los elementos almacenados en la tabla hash, agrupados por índice.

- ArrayList<T> to(Book/Client)List(): Convierte todos los elementos de la tabla hash en una sola lista.

Notas sobre lógica interna

Primero, El índice en la tabla se calcula como id % SIZE, donde id es el identificador del libro o cliente. Esto distribuye los elementos en la tabla de forma uniforme. Así mismo, si varios elementos tienen el mismo índice, se almacenan en una lista (ArrayList) dentro de esa posición. Esto permite manejar colisiones de forma sencilla. Por último, el método toList() recorre toda la tabla y concatena los elementos en una sola lista.