**Descubrimiento y Análisis de proyecto de clase**

Andrés David Elizalde Peralta

01220371068

Emily Julieth Nieves Badillo

01220371022

Santiago Alexander Ospina Pabon

01220371058

Docente: Nury Farelo Velasquez

Diseño de Software 1



Ingeniería de Software

Universidad de Santander (UDES)

Bucaramanga

2024

Tabla de contenido

[SISTEMA DE GESTIÓN DE BIBLIOTECA 4](#_Toc179983685)

[Análisis del contexto del problema 4](#_Toc179983686)

[Descripción: 4](#_Toc179983687)

[Necesidades: 4](#_Toc179983688)

[Roles de usuario del problema 5](#_Toc179983689)

[1. Administradores 5](#_Toc179983690)

[2. Bibliotecarios 6](#_Toc179983691)

[3. Usuarios Finales 6](#_Toc179983692)

[3.Todos los usuarios 7](#_Toc179983693)

[Diagrama de casos de uso 7](#_Toc179983694)

[Definición de Clases 8](#_Toc179983695)

[Clases identificadas 8](#_Toc179983696)

[Diagrama UML 9](#_Toc179983697)

[Clases y métodos explicados 9](#_Toc179983698)

[Patrones de diseño implementados 15](#_Toc179983699)

[Patrón Singleton 15](#_Toc179983700)

[Patrón Builder 16](#_Toc179983701)

[Requerimientos funcionales 17](#_Toc179983702)

[Gestión de usuarios: 17](#_Toc179983703)

[Gestión de libros 18](#_Toc179983704)

[Gestión de Préstamos y Devoluciones 18](#_Toc179983705)

[Gestión de Reservas 19](#_Toc179983706)

[Búsqueda y Filtrado Avanzado 19](#_Toc179983707)

[Informes y Estadísticas 20](#_Toc179983708)

[Seguridad y Privacidad 20](#_Toc179983709)

[Interfaz de Usuario 20](#_Toc179983710)

[Requisitos no funcionales 20](#_Toc179983711)

[Rendimiento 21](#_Toc179983712)

[Usabilidad 21](#_Toc179983713)

[Mantenimiento 21](#_Toc179983714)

[Documentación 22](#_Toc179983715)

[Sistemas o herramientas existentes como referencia 22](#_Toc179983716)

[Koha 22](#_Toc179983717)

[Evergreen 23](#_Toc179983718)

[LibraryWorld 23](#_Toc179983719)

[Diagrama base de datos propuesto 24](#_Toc179983720)

[Bibliografía 24](#_Toc179983721)

# SISTEMA DE GESTIÓN DE BIBLIOTECA

## Análisis del contexto del problema

### Descripción:

Las bibliotecas, como centros de información y conocimiento, enfrentan el desafío de gestionar grandes volúmenes de libros y recursos mientras ofrecen a los usuarios un acceso rápido y eficiente a la información que buscan. En una biblioteca común, la administración de inventarios, el seguimiento de préstamos, y la facilitación de la búsqueda de libros son procesos cruciales que demandan un sistema de gestión robusto. La evolución tecnológica ha permitido a muchas bibliotecas modernizar sus sistemas, pero todavía existen desafíos significativos en la integración de nuevas tecnologías que maximicen la eficiencia operativa y mejoren la experiencia del usuario.

En muchas bibliotecas, la gestión de inventarios y la búsqueda de libros se realiza manualmente o mediante sistemas obsoletos que no aprovechan las capacidades de búsqueda avanzada, lo que genera ineficiencias. Estos sistemas no solo ralentizan las operaciones diarias, sino que también pueden provocar errores en el registro de préstamos y devoluciones, dificultando el seguimiento del inventario y la disponibilidad de libros. Los usuarios finales a menudo encuentran difícil localizar los libros que necesitan, lo que reduce la efectividad de la biblioteca como recurso educativo y cultural.

A partir de lo cual surgen las siguientes necesidades:

### Necesidades:

#### Usuarios y roles:

El sistema debe servir a una variedad de usuarios, incluidos administradores, bibliotecarios y usuarios finales (estudiantes, investigadores, lectores en general). Cada grupo de usuarios tiene necesidades y permisos específicos:

* **Administradores:** Usuario con acceso a todas las funciones de la aplicación, posibilidad de gestión de roles, y modificación de información asociada a los libros.
* Bibliotecarios: Requieren acceso a funciones de préstamo y devolución, así como la capacidad de buscar y gestionar el catálogo.
* Usuarios Finales: Deben poder buscar libros fácilmente, verificar su disponibilidad, y reservarlos o pedir préstamos.

#### Interfaz de Usuario

El sistema debe ofrecer una interfaz de usuario intuitiva y accesible. La experiencia del usuario es fundamental, ya que afecta directamente la adopción del sistema. Una interfaz bien diseñada debería permitir a los usuarios realizar tareas comunes como la búsqueda de libros y la gestión de reservas de manera eficiente y sin complicaciones.

#### Funciones de búsqueda y reserva

La búsqueda avanzada es un componente crítico del sistema, permitiendo a los usuarios filtrar resultados por diversos criterios como autor, título, año de publicación, y género. Además, el sistema debe permitir a los usuarios reservar libros y recibir notificaciones sobre la disponibilidad de los mismos.

#### Inventario claro y preciso

Es necesario implementar técnicas de inventariado que permitan seguir con claridad y facilidad la disponibilidad de los libros a demás de implementar un sistema de software que mantenga actualizado en tiempo real dicho inventario.

## Roles de usuario del problema

### 1. Administradores

#### Gestionar Roles de Usuario

* **Descripción**: El administrador puede crear, editar, y eliminar roles de usuario. También puede asignar permisos específicos a cada rol.
* **Actores**: Administrador
* **Precondiciones**: El administrador debe estar autenticado.

#### Gestionar Información de Libros

* **Descripción**: Permite al administrador agregar, editar o eliminar información de los libros en el sistema.
* **Actores**: Administrador, Bibliotecario
* **Precondiciones**: El administrador debe estar autenticado.

#### Gestionar Usuarios

* **Descripción**: El administrador puede agregar, editar o eliminar usuarios del sistema.
* **Actores**: Administrador
* **Precondiciones**: El administrador debe estar autenticado.

#### Generación de informes

* **Descripción**: El administrador puede generar o consultar informes generados por el sistema
* **Actores**: Administrador
* **Precondiciones**: El administrador debe estar autenticado.

### 2. Bibliotecarios

#### Gestionar Préstamos y Devoluciones

* **Descripción**: El bibliotecario puede registrar préstamos y devoluciones de libros para los usuarios.
* **Actores**: Bibliotecario
* **Precondiciones**: El bibliotecario debe estar autenticado.

#### Gestionar Catálogo de Libros

* + **Descripción**: Permite al bibliotecario buscar libros en el catálogo y actualizar su disponibilidad.
  + **Actores**: Bibliotecario
  + **Precondiciones**: El bibliotecario debe estar autenticado.

### 3. Usuarios Finales

#### Buscar Libros

* + **Descripción**: Los usuarios finales pueden buscar libros en el catálogo utilizando diferentes filtros (título, autor, género, etc.).
  + **Actores**: Usuario Final, Bibliotecario
  + **Precondiciones**: El usuario debe estar autenticado o puede acceder como invitado (según configuración del sistema).

#### Reservar Libros

* + **Descripción**: Los usuarios pueden reservar libros que estén disponibles o en préstamo.
  + **Actores**: Usuario Final
  + **Precondiciones**: El usuario debe estar autenticado.

#### Consultar Disponibilidad de Libros

* + **Descripción**: Permite a los usuarios verificar la disponibilidad de un libro específico.
  + **Actores**: Usuario Final
  + **Precondiciones**: El usuario debe estar autenticado o acceder como invitado.

### 3.Todos los usuarios

#### Autentificación

* + **Descripción**: Todos los usuarios deben registrarse antes de realizar cualquiera acción que intervenga en la base de datos del sistema (préstamo de libros, actualización de información, etc)
  + **Actores**: Todos
  + **Precondiciones**: Ninguna.

### Diagrama de casos de uso

Diagrama

Descripción generada automáticamente

Ilustración 1. Diagrama de casos de uso

## Definición de Clases

### Clases identificadas

Para el desarrollo del sistema, se han identificado las siguientes clases:

* **Usuario**: Clase base que representa a cualquier tipo de usuario en el sistema.
* **Administrador**: Hereda de Usuario y maneja las tareas administrativas como la gestión de roles, usuarios, y generación de informes.
* **Bibliotecario**: Hereda de Usuario y se encarga de las tareas relacionadas con el préstamo y la devolución de libros.
* **Usuario Final**: Hereda de Usuario y representa a los usuarios que buscan libros y realizan reservas.
* **Libro**: Representa los libros dentro de la biblioteca.
* **Reserva**: Gestiona las reservas de los libros.
* **Prestamo**: Maneja los préstamos y devoluciones de los libros.
* **Rol**: Define los diferentes roles de los usuarios y sus permisos asociados.
* **Informe**: Permite la generación de informes por parte del administrador.

### Diagrama UML

Diagrama

Descripción generada automáticamente

Ilustración 2. Diagrama UML de las clases con sus características y métodos

### Clases y métodos explicados

En esta sección, se detalla cómo se implementan las clases en Python y la lógica de cada uno de sus métodos, asegurando que cumplan con las necesidades del sistema de gestión de bibliotecas.

**Clase Usuario**

La clase Usuario es la base de todos los tipos de usuarios en el sistema y gestiona la autenticación y el manejo de credenciales. Además, incluye atributos adicionales como dirección, teléfono y fecha de nacimiento.

**Atributos:**

* nomUsuario: Nombre de usuario.
* contraseña: Contraseña del usuario.
* nombres: Nombre(s) del usuario.
* apellidos: Apellidos del usuario.
* direccion: Dirección del usuario.
* telefono: Número de teléfono.
* rol: Rol del usuario (Administrador, Bibliotecario, Usuario Final).
* correoElectronico: Correo del usuario.
* fechaNacimiento: Fecha de nacimiento del usuario.

**Métodos:**

* autenticarse(): Permite a los usuarios autenticarse en el sistema.
* cambiarContraseña(): Permite a los usuarios cambiar su contraseña.

Texto

Descripción generada automáticamente

Código 1. Definición de la clase usuario con sus métodos

**Clase Administrador**

El Administrador hereda de Usuario y tiene las siguientes responsabilidades: gestionar roles, gestionar usuarios, generar informes y gestionar la información de los libros.

**Métodos:**

* gestionarRoles(): Asignar roles a los usuarios.
* gestionarUsuarios(): Crear, editar y eliminar usuarios.
* generarInformes(): Generar informes de préstamos, devoluciones y reservas.
* gestionarInformacionLibros(): Gestionar los detalles del catálogo de libros.

Texto

Descripción generada automáticamente

Código 2. Código para declarar la clase administrador, que hereda de usuario

**Clase Bibliotecario**

El Bibliotecario hereda de Usuario y tiene múltiples responsabilidades: registrar préstamos, registrar devoluciones, gestionar el catálogo de libros, buscar libros y gestionar la información de los libros.

**Atributos:**

* puntoTrabajo: Lugar físico donde trabaja el bibliotecario.

**Métodos:**

* registrarPrestamo(): Registrar el préstamo de un libro.
* registrarDevolucion(): Registrar la devolución de un libro.
* gestionarCatalogo(): Modificar, agregar o eliminar libros del catálogo.
* gestionarInformacionLibros(): Gestionar los detalles del catálogo de libros.
* buscarLibros(): Buscar libros en el sistema.

Texto

Descripción generada automáticamente

Código 3. Código para declarar la clase bibliotecario que hereda de usuario

**Clase UsuarioFinal**

El UsuarioFinal hereda de Usuario y se enfoca en la búsqueda, reserva y consulta de disponibilidad de los libros.

**Métodos:**

* buscarLibros(): Buscar libros en el catálogo.
* reservarLibro(): Reservar un libro disponible.
* consultarDisponibilidad(): Consultar la disponibilidad de un libro.

Texto

Descripción generada automáticamente

Código 4. Código para declarar la clase usuarioFinal que hereda de Usuario

**Clase Libro**

La clase Libro es clave para representar los libros en el sistema y maneja su disponibilidad y la información básica del catálogo.

**Atributos:**

* ISBN: Código único del libro.
* titulo: Título del libro.
* autor: Autor del libro.
* genero: Género o categoría del libro.
* ubicacion: Ubicación del libro en la biblioteca.
* disponibilidad: Estado de disponibilidad del libro (Disponible/Prestado/Reservado).
* idEditorial: Identificador de la editorial.
* idioma: Idioma del libro.
* año: Año de publicación.
* edicion: Edición del libro.
* idCategoria: Identificador de la categoría.

**Métodos:**

* actualizarDisponibilidad(): Cambiar el estado de disponibilidad del libro.
* obtenerInformacion(): Obtener información completa del libro.

Texto

Descripción generada automáticamente

Código 5. Código para declarar la clase libro

**Clase Prestamo**

La clase Prestamo gestiona el registro de préstamos y devoluciones de libros.

**Atributos:**

* idLibroLocal: Identificador del libro prestado.
* idUsuario: Identificador del usuario que realiza el préstamo.
* fechaInicio: Fecha de inicio del préstamo.
* fechaFin: Fecha de vencimiento del préstamo.
* fechaDev: Fecha de devolución del libro.

**Métodos:**

* registrarPrestamo(): Registrar un nuevo préstamo de libro.
* registrarDevolucion(): Registrar la devolución de un libro.

Texto

Descripción generada automáticamente

Código 6. Código para declarar la clase préstamo

**Clase Reserva**

La clase Reserva gestiona la reserva de libros por parte de los usuarios.

**Atributos:**

* libro: El libro reservado.
* usuario: El usuario que realiza la reserva.
* fechaReserva: La fecha en la que se realiza la reserva.

**Métodos:**

* reservar(): Realiza la reserva de un libro.
* cancelarReserva(): Cancela la reserva de un libro.

Texto

Descripción generada automáticamente

Código 7 Código para declarar la clase reserva

## Patrones de diseño implementados

### Patrón Singleton

#### Descripción del Patrón

#### En el contexto de una base de datos, el patrón Singleton puede garantizar que solo haya una conexión activa a la base de datos, evitando la creación de múltiples conexiones innecesarias y mejorando el rendimiento.

#### Problema que Resuelve

* + **Una sola conexión a la base de datos:** Crear múltiples conexiones a la base de datos puede ser costoso y poco eficiente. El Singleton asegura que solo exista una conexión activa en todo momento.
* **Globalmente accesible:** En una aplicación grande, puede haber múltiples módulos o clases que necesiten acceder a la base de datos. Con el Singleton, cada uno de estos módulos puede obtener la misma instancia de la conexión de manera segura.
* **Control centralizado:** Al centralizar la gestión de la conexión de base de datos en un solo lugar, puedes asegurarte de que la conexión se maneja de forma consistente, evitando errores como conexiones no cerradas o conexiones múltiples innecesarias.

#### Implementación del Singleton

import sqlite3

from sqlite3 import Error

class DatabaseConnection:

    \_instance = None

    def \_\_new\_\_(cls):

        if cls.\_instance is None:

            cls.\_instance = super(DatabaseConnection, cls).\_\_new\_\_(cls)

            cls.\_instance.\_connection = None

        return cls.\_instance

    def connect(self, db\_file):

        """Establecer una conexión con la base de datos SQLite"""

        if self.\_connection is None:

            try:

                self.\_connection = sqlite3.connect(db\_file)

                print("Conexión a la base de datos establecida.")

            except Error as e:

                print(f"Error al conectar a la base de datos: {e}")

        return self.\_connection

    def close(self):

        """Cerrar la conexión con la base de datos"""

        if self.\_connection:

            self.\_connection.close()

            print("Conexión a la base de datos cerrada.")

            self.\_connection = None

# Uso del Singleton para manejar la conexión a la base de datos

if \_\_name\_\_ == "\_\_main\_\_":

    # Crear la primera instancia del singleton

    db1 = DatabaseConnection()

    connection1 = db1.connect("example.db")

Código 8. Implementación patrón singleton en conexión a la base de datos

### Patrón Builder

**Clase Implementada**: Libro

#### Descripción del Patrón

El patrón Builder se utiliza para construir objetos complejos paso a paso. En este sistema, se aplica a la clase Libro, que puede tener múltiples atributos como título, autor, ISBN, género, y ubicación. El Builder facilita la creación de instancias de Libro al permitir un proceso de construcción más flexible y controlado.

#### Problema que Resuelve

El patrón Builder resuelve el problema de la creación rígida de objetos. En lugar de tener que pasar todos los parámetros del libro al constructor en un solo paso, el Builder permite agregar atributos uno por uno, lo cual es especialmente útil cuando se manejan objetos con muchos atributos o cuando ciertos atributos son opcionales.

#### Implementación del Builder

Texto

Descripción generada automáticamente

#### Justificación

El patrón Builder es ideal para la clase Libro, ya que los libros en el sistema de gestión de bibliotecas pueden tener una amplia variedad de atributos que se deben definir. El uso del Builder permite la creación de objetos Libro de forma controlada y con flexibilidad, lo cual es útil en situaciones donde no se tienen todos los datos del libro en un solo momento o algunos atributos son opcionales.

## Requerimientos funcionales

### Gestión de usuarios:

#### Registro de Usuarios:

El sistema debe permitir a los usuarios registrarse creando una cuenta con información personal básica (nombre, correo electrónico, número de identificación, etc.).

#### Autenticación y Autorización:

El sistema debe permitir a los usuarios autenticarse usando un nombre de usuario y contraseña.

#### Gestión de Roles:

Los administradores deben poder crear, modificar y eliminar roles de usuario.

Los administradores deben poder asignar permisos específicos a cada rol.

### Gestión de libros

#### Agregar Libros:

El sistema debe permitir a los administradores y bibliotecarios agregar nuevos libros al catálogo, ingresando detalles como título, autor, ISBN, género, año de publicación, y ubicación en la biblioteca.

#### Editar Información de Libros:

Los administradores y bibliotecarios deben poder editar la información de los libros existentes.

#### Eliminar Libros:

Los administradores y bibliotecarios deben poder eliminar libros del catálogo.

#### Buscar Libros:

El sistema debe permitir a todos los usuarios buscar libros utilizando múltiples criterios, como título, autor, ISBN, género, o palabras clave.

#### Ver Detalles de Libros:

Los usuarios deben poder ver los detalles completos de un libro, incluyendo su disponibilidad y ubicación en la biblioteca.

#### Consultar Disponibilidad de Libros:

El sistema debe mostrar si un libro está disponible, prestado, o reservado.

### Gestión de Préstamos y Devoluciones

#### Registrar Préstamos:

Los bibliotecarios deben poder registrar préstamos de libros a usuarios, vinculando el libro y el usuario en la base de datos.

#### Registrar Devoluciones:

Los bibliotecarios deben poder registrar la devolución de libros prestados, actualizando el estado del libro a "disponible".

#### Historial de Préstamos:

Los usuarios deben poder consultar su historial de préstamos.

Los administradores y bibliotecarios deben poder consultar el historial de préstamos de cualquier usuario.

#### Notificaciones:

El sistema debe enviar notificaciones a los usuarios cuando se acerque la fecha de devolución de un libro.

El sistema debe notificar a los usuarios cuando un libro reservado esté disponible para su recogida.

### Gestión de Reservas

#### Reservar Libros:

Los usuarios deben poder reservar libros que estén disponibles o actualmente en préstamo.

#### Cancelar Reservas:

Los usuarios deben poder cancelar una reserva antes de que el libro sea recogido.

#### Gestión de la Cola de Reservas:

El sistema debe manejar una cola de reservas para los libros solicitados por varios usuarios, asignando el libro al siguiente usuario en la cola cuando sea devuelto.

### Búsqueda y Filtrado Avanzado

#### Búsqueda de Libros:

El sistema debe permitir a los usuarios buscar libros utilizando filtros avanzados como autor, título, año de publicación, género, y disponibilidad.

#### Filtrado de Resultados:

Los usuarios deben poder filtrar los resultados de búsqueda por diferentes criterios, como la relevancia, la fecha de publicación, o la disponibilidad.

#### Sugerencias de Búsqueda:

El sistema debe ofrecer sugerencias de búsqueda a medida que los usuarios ingresan términos de búsqueda.

### Informes y Estadísticas

#### Generación de Informes:

El sistema debe permitir a los administradores generar informes sobre préstamos, devoluciones, reservas, y el uso del catálogo de libros.

#### Estadísticas de Uso:

El sistema debe proporcionar estadísticas de uso, como el libro más prestado, las categorías más populares, y la cantidad de usuarios activos.

### Seguridad y Privacidad

#### Control de Acceso:

El sistema debe implementar un control de acceso basado en roles para proteger las funcionalidades críticas y los datos sensibles.

#### Encriptación de Datos:

El sistema debe encriptar datos sensibles, como contraseñas y datos personales.

#### Copia de Seguridad:

El sistema debe realizar copias de seguridad automáticas de la base de datos en intervalos regulares.

### Interfaz de Usuario

#### Interfaz Intuitiva:

El sistema debe ofrecer una interfaz de usuario intuitiva y fácil de usar para todos los roles.

#### Diseño Responsive:

El sistema debe ser accesible desde dispositivos móviles y de escritorio, adaptando el diseño a diferentes tamaños de pantalla.

#### Accesibilidad:

El sistema debe cumplir con los estándares de accesibilidad, permitiendo el uso por personas con discapacidades.

## Requisitos no funcionales

Estos requisitos no funcionales son fundamentales para garantizar que el sistema de gestión de la biblioteca no solo cumpla con las funcionalidades requeridas, sino que también lo haga de manera eficiente, segura y sostenible a lo largo del tiempo.

### Rendimiento

#### Tiempo de Respuesta:

El sistema debe responder a las consultas de búsqueda de libros en menos de 2 segundos para una base de datos de hasta 10,000 libros.

#### Capacidad de Carga:

El sistema debe poder manejar simultáneamente alta cantidad usuarios sin degradar significativamente el rendimiento.

#### Escalabilidad:

El sistema debe ser escalable para manejar un crecimiento en el número de usuarios y la cantidad de datos, permitiendo agregar más recursos cuando sea necesario.

### Usabilidad

#### Interfaz de Usuario:

La interfaz debe ser intuitiva, con un diseño que facilite la navegación tanto a usuarios novatos como experimentados.

#### Accesibilidad:

El sistema debe cumplir con los estándares de accesibilidad (WCAG 2.1 AA), permitiendo su uso por personas con discapacidades visuales, auditivas, y motrices.

#### Compatibilidad Multidispositivo:

El sistema debe ser compatible con diferentes dispositivos y navegadores, incluyendo navegadores móviles, tablets y computadoras de escritorio.

#### Internacionalización:

El sistema debe soportar múltiples idiomas y adaptarse a las configuraciones regionales (por ejemplo, formato de fecha, moneda, etc.).

### Mantenimiento

#### Mantenibilidad:

El sistema debe estar diseñado de manera que sea fácil de mantener y actualizar. El código debe estar bien documentado, y las actualizaciones no deben interrumpir las operaciones del sistema.

#### Soporte para Actualizaciones:

El sistema debe permitir la implementación de actualizaciones sin necesidad de un tiempo de inactividad significativo, preferiblemente mediante actualizaciones en caliente (hot fixes).

#### Modularidad:

El sistema debe ser modular para que los componentes individuales puedan ser actualizados o reemplazados sin afectar al sistema en su totalidad.

### Documentación

#### Documentación del Usuario:

El sistema debe incluir una documentación completa y accesible para los usuarios finales, bibliotecarios y administradores.

#### Documentación Técnica:

El sistema debe incluir documentación técnica detallada para los desarrolladores, que describa la arquitectura, los flujos de trabajo, y las API.

## Sistemas o herramientas existentes como referencia

### Koha



Ilustración 3. Logo Koha

* **Descripción**: Koha es un sistema integrado de gestión de bibliotecas (ILS, por sus siglas en inglés) de código abierto, altamente configurable y escalable. Es utilizado por bibliotecas de todos los tamaños en todo el mundo (Koha, 2024).
* **Características Principales**:
  + Gestión de adquisiciones, préstamos, devoluciones y catalogación.
  + Búsqueda avanzada y personalizable.
  + Administración de usuarios y permisos basada en roles.
  + Soporte multilingüe.
  + Informes y estadísticas detalladas.
* **Tecnologías**: Escrito en Perl, utiliza MySQL como base de datos.

### ****Evergreen****

Imagen que contiene nombre de la empresa

Descripción generada automáticamente

Ilustración 4. Logo EverGreen

* **Descripción**: Evergreen es otro sistema ILS de código abierto que es ampliamente utilizado por consorcios de bibliotecas públicas. Está diseñado para ser altamente escalable y eficiente (Frasur, 2024).
* **Características Principales**:
  + Gestión de colecciones, circulación y catálogo en línea (OPAC).
  + Soporte para bibliotecas multiusuario y multibiblioteca.
  + Capacidad de integración con otros sistemas mediante API.
  + Informes personalizados y herramientas de análisis.
* **Tecnologías**: Desarrollado en C y Perl, con PostgreSQL como base de datos.

### ****LibraryWorld****

Imagen que contiene Texto

Descripción generada automáticamente

Ilustración 5. Logo Library World

* **Descripción**: LibraryWorld es un sistema ILS basado en la nube, diseñado principalmente para pequeñas bibliotecas y organizaciones (LibraryWorld, 2024).
* **Características Principales**:
  + Fácil de usar con una configuración mínima.
  + Gestión de catalogación, circulación y usuarios.
  + Búsqueda y filtrado sencillo de recursos.
  + Herramientas básicas de informes y estadísticas.
* **Tecnologías**: Servicio basado en la nube, accesible desde cualquier navegador.

## Diagrama base de datos propuesto

El presente es un diagrama entidad relación propuesto para una base de datos relacional que cumpla con las condiciones dadas para el proyecto

Imagen que contiene Escala de tiempo

Descripción generada automáticamente

Ilustración 6. Diagrama entidad relacion

# Bibliografía

Frasur, R. (19 de Julio de 2024). *Evergreeen ILS*. Obtenido de https://evergreen-ils.org/

Koha. (2024). *Official websote of koha libary software*. Obtenido de https://koha-community.org/

LibraryWorld. (19 de Julio de 2024). Obtenido de https://www.libraryworld.com/