Memoria descriptiva de la aplicación de escritorio SisBib

1. Introducción

La presente memoria descriptiva documenta el funcionamiento, instalación y uso de la aplicación de escritorio SisBib desarrollada bajo el framework Qt y utilizando una base de datos SQLite.

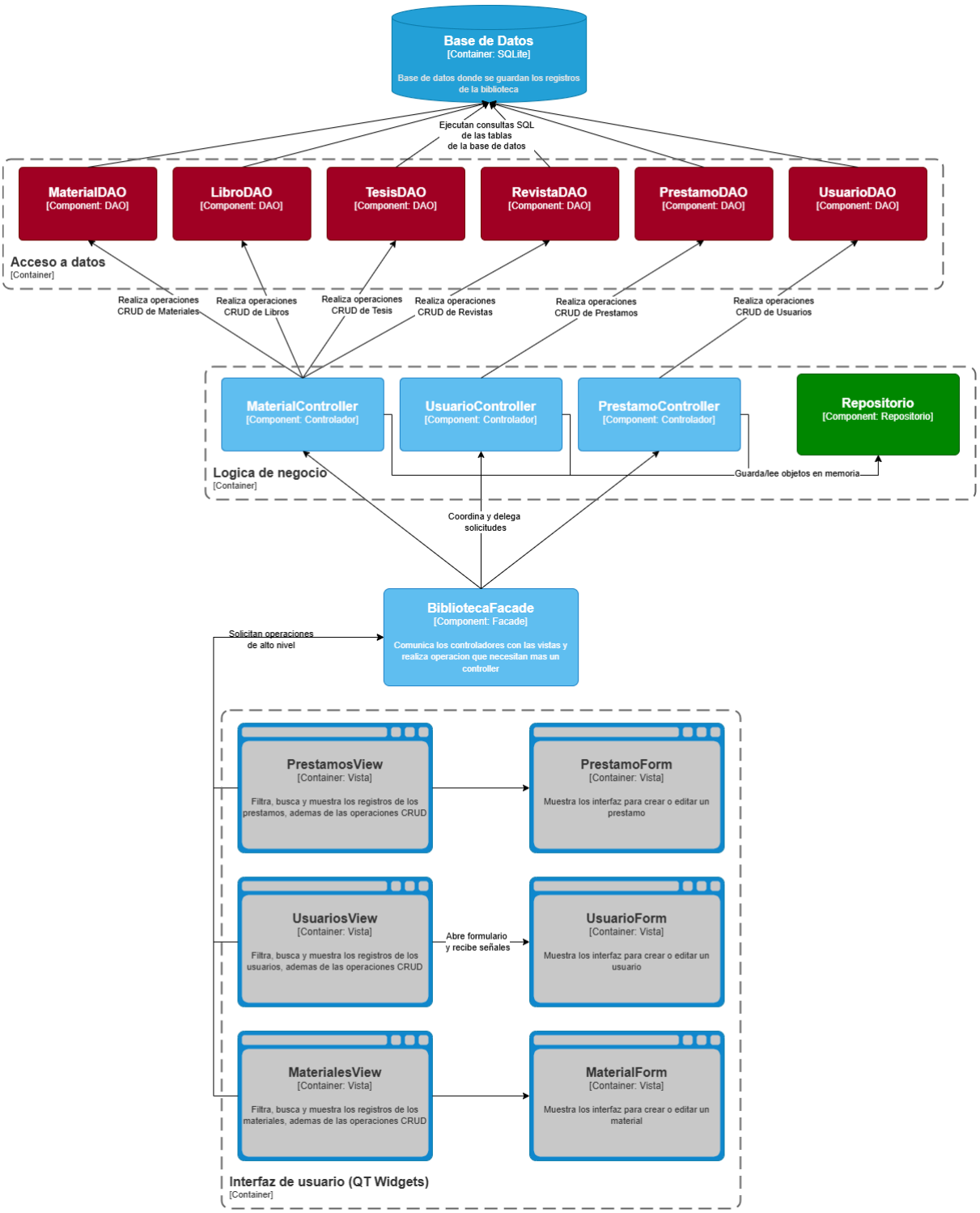
SisBib es un sistema de gestión bibliotecaria orientado a la administración de préstamos de materiales, permitiendo registrar nuevos préstamos verificando la disponibilidad de materiales y los límites correspondientes, así como gestionar su devolución. Además, la aplicación incluye módulos para la gestión de usuarios y la gestión de materiales, ofreciendo funcionalidades de creación, modificación y eliminación de registros. El objetivo principal del sistema es facilitar el control interno de una biblioteca, optimizando los procesos y asegurando la consistencia de la información almacenada.

2. Arquitectura y plataforma tecnológica

SisBib es una aplicación de escritorio desarrollada en C++ utilizando el framework Qt 6, siguiendo la arquitectura Modelo–Vista–Controlador (MVC) para organizar la lógica del sistema. El sistema utiliza una base de datos SQLite, integrada localmente en la aplicación, lo que permite operar sin necesidad de un servidor externo.

La aplicación está diseñada para ejecutarse en equipos con sistema operativo Windows 10 o superior (64 bits) y requiere las bibliotecas de ejecución de Qt correspondientes a su versión (QtCore, QtGui, QtWidgets, QtSql, entre otras).

El entorno de ejecución incluye el motor de base de datos SQLite 3, manejado a través del módulo Qt SQL, que permite la conexión y operaciones CRUD sobre los datos de usuarios, materiales y préstamos.



El sistema implementa una arquitectura basada en el patrón Modelo–Vista–Controlador (MVC), complementada con componentes de acceso a datos (DAO), un repositorio en memoria, una capa de fachada para coordinación de operaciones entre los controllers y las vistas y una base de datos relacional.

2.1 Base de Datos

Es un sistema relacional que almacena la información persistente del sistema:
Tablas: Material, Libro, Revista, Tesis, Usuario, Prestamo.

Las relaciones siguen el modelo ERD del proyecto.

Interacciones:

- Solo los DAOs acceden a la base de datos.
- No se comunica directamente con controllers ni vistas.

2.2 Modelos

Los Modelos representan las entidades principales del dominio del sistema, son 6: Material, Libro, Revista, Tesis, Usuario y Prestamo. Son clases simples que encapsulan datos y reglas mínimas relacionadas con la estructura de cada objeto.

Responsabilidades de los Modelos

- Definir la estructura de datos (atributos).
- Ser transportados entre las capas (View → Facade → Controller → DAO).
- Ser convertidos desde y hacia registros de la base de datos.

2.3 Controladores

Cada entidad principal del sistema tiene un Controller: UsuarioController, MaterialController, PrestamoController.

Los Controllers encapsulan la lógica de negocio, gestionan la memoria interna del sistema y se encargan de manipular los objetos de dominio (Modelos).

Responsabilidades:

- Validación básica de operaciones.
- Coordinación con DAOs para consultar o modificar la base de datos.
- Coordinación con el repositorio en memoria para búsquedas y filtrados.
- Mantener un Repositorio en Memoria para respuestas rápidas.
- Entregar objetos o listas a la Vista a través del Facade.

Interacciones:

- Se comunican con los DAO para el acceso a datos
- Se comunican con el repositorio en memoria para ejecutar operaciones CRUD sobre ella y obtener objetos filtrados.

2.4 Repositorio en Memoria

Es una estructura auxiliar usada únicamente por los Controllers. No forma parte de la BD, no es accedido por otras capas, y existe para optimizar el acceso a los datos ya cargados.

Responsabilidades:

- Almacenar listas de Usuarios, Materiales y Préstamos en RAM.
- Proveer métodos para obtener, eliminar, agregar, filtrar o actualizar objetos en memoria.

Interacciones:

- Solo se comunica con los Controllers (unidireccional).
- No es accedido por DAOs ni por las Vistas.

2.5 Fachada

La arquitectura incluye un Facade que actúa como un punto unificado de comunicación entre las vistas y los controladores.

Su propósito es simplificar el acceso a las funcionalidades del sistema cuando una operación requiere más de un Controller.

Responsabilidades:

- Coordinar acciones entre múltiples controladores.
- Simplificar la interfaz que la Vista necesita conocer.
- Reducir el acoplamiento directo de las vistas con la lógica interna.
- Interacciones:
- La vista solicita operaciones a los controladores a través del Facade o del propio facade.
- El facade ejecuta operaciones de los controllers que necesiten más de uno de ellos.

2.5 Vistas

La capa de presentación está compuesta por vistas principales y vistas de formulario (Form Views) desarrolladas en Qt. Su función es interactuar directamente con el usuario final, mostrando listas, formularios y resultados.

Responsabilidades:

- Capturar entradas del usuario.
- Mostrar información (usuarios, materiales, préstamos).
- Abrir formularios específicos (por ejemplo, “Registrar material”, “Editar usuario”).

Interacciones:

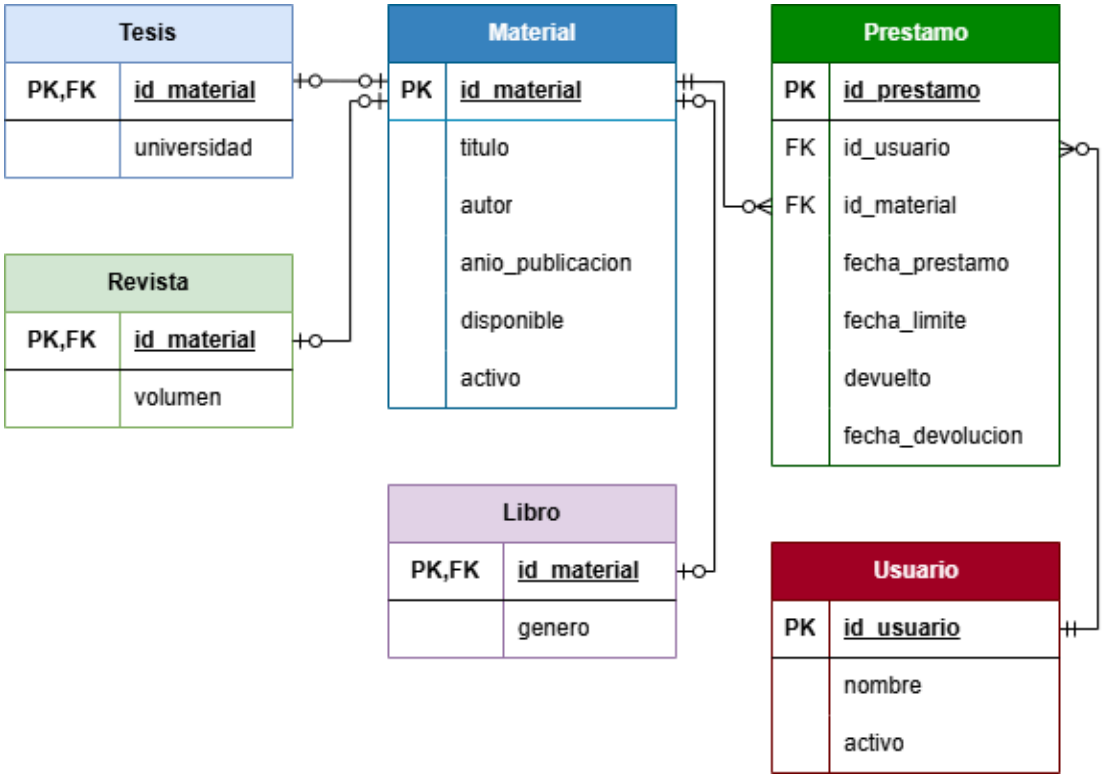
- Se comunican únicamente con la Fachada para solicitar operaciones de negocio.
- Entre vistas y vistas-form existe una relación de "abre" o "invoca", representada como una dependencia simple, las vistas también reciben señales de los form para actualizar sus registros

3. Diseño de Base de Datos

Para la persistencia de la información, el sistema utiliza una base de datos relacional SQLite. El diseño del esquema sigue las reglas de normalización para garantizar la integridad de los datos y minimizar la redundancia.

3.1 Modelo Entidad-Relación

El siguiente diagrama ilustra las entidades principales del sistema, sus atributos y las relaciones existentes entre ellas.



3.2 Estrategia de Generalización

Se ha implementado un patrón de Generalización/Especialización para el manejo del inventario bibliográfico.

- Entidad Padre (Material): Centraliza los atributos comunes a todas las publicaciones, como el titulo, autor, anio_publicacion y el estado de disponibilidad (disponible, activo).
- Entidades Hijas (Libro, Revista, Tesis): Contienen únicamente los atributos específicos de cada tipo de documento.

Esta relación se mantiene mediante la clave primaria id_material, que actúa simultáneamente como clave foránea en las tablas hijas, vinculando un registro específico (ej: una Tesis) con su información general en la tabla Material.

3.3 Diccionario de Datos

A continuación, se describen las tablas principales identificadas en el esquema:

A. Tabla Material: Representa el inventario físico de la biblioteca.

- disponible: Booleano que indica si el libro está físicamente en la biblioteca o prestado actualmente.
- activo: Permite el "borrado lógico" para no perder el historial de préstamos de libros dados de baja.

B. Tablas de Especialización

- **Libro:** Almacena el género literario o técnico de la obra.
- **Revista:** Almacena el número de volumen de la publicación.
- **Tesis:** Registra la universidad de origen del documento.

C. Tabla Usuario: Almacena la información de los usuarios de la biblioteca, tiene los campos nombre y activo (para el borrado lógico).

D. Tabla Prestamo (Transaccional): Es la tabla que registra los préstamos realizados mediante la relación entre un usuario y un material.

- id_usuario: ID del material prestado
- id_material: ID del usuario al que se prestó el material.
- fecha_prestamo: Fecha de registro del préstamo.
- fecha_limite: Fecha máxima de devolución.
- fecha_devolucion: Fecha en la que el usuario entregó el libro.
- devuelto: Indica si el material fue devuelto o no.

4. Funcionalidades del sistema

SisBib permite la gestión de préstamos y devoluciones así como los usuarios de la biblioteca y los materiales bibliográficos. Sus principales funcionalidades incluyen:

- Registrar nuevos materiales bibliográficos (libros, revistas y tesis).
- Editar información existente (título, autor, año, tipo, etc.).
- Eliminar materiales del registro.
- Listar todos los materiales disponibles.
- Guardar y cargar los datos desde una base de datos SQLite.
- Registrar usuarios que solicitan préstamos.
- Consultar y gestionar información de cada usuario.
- Realizar préstamos de materiales disponibles a usuarios registrados.
- Registrar la devolución de materiales prestados.
- Controlar el estado de cada material (disponible / prestado).
- Buscar materiales por título.
- Filtrar materiales según disponibilidad.
- Buscar usuarios por nombre.
- Buscar préstamos por nombre de usuario
- Filtrar préstamos según estado (pendientes, vencidos, devueltos).

5. Proceso de ejecución del proyecto

Clonar el repositorio

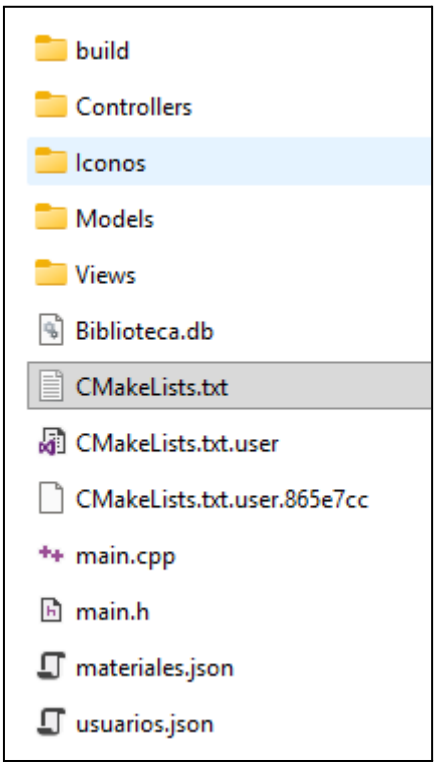
El código fuente puede obtenerse utilizando Git mediante el siguiente comando:

```shell
https://github.com/rescobedoq/sbd.git
```

También puede descargarse en formato .zip desde la interfaz web de GitHub.

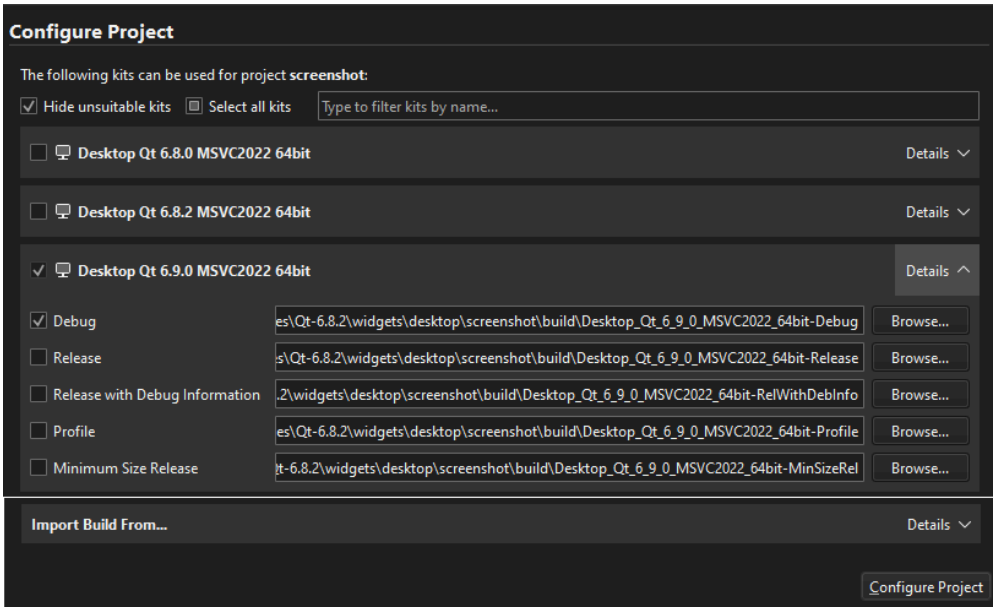
Abrir el proyecto en Qt Creator

Una vez descargado, debe abrir el archivo de proyecto dirigiéndose hacia File > Open File or Proyect en Qt y seleccionando el archivo CMakeLists.txt.



Qt Creator detectará automáticamente el tipo de proyecto y solicitará su configuración inicial. En la ventana de configuración, se debe seleccionar un kit de desarrollo compatible, por ejemplo:

- Desktop Qt 6.x.x MinGW 64-bit
- Desktop Qt 6.x.x MSVC



Qt Creator generará las carpetas de build necesarias.

Compilar el proyecto

Con la configuración completa, basta con presionar el botón Build o la tecla Ctrl + B. Qt Creator compilará automáticamente todos los módulos de la aplicación.

6. Conclusiones

La aplicación de escritorio SisBib constituye una solución integral para la gestión de usuarios, materiales y préstamos. Su diseño basado en la arquitectura Modelo–Vista–Controlador (MVC), complementado con componentes especializados como DAOs, repositorios en memoria y una fachada de coordinación, permite una estructura organizada, escalable y de fácil mantenimiento.