Un dibujo de una persona

Descripción generada automáticamente con confianza bajaUn dibujo de un perro

Descripción generada automáticamente con confianza media

**Universidad Autónoma de Zacatecas**

**Unidad Académica de Ingeniería Eléctrica**

**Programa de Ingeniería en Software**

**ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS**

**System Design Specification  (SDS)**

**Alumno: Marcos de Jesús Arredondo Rebollo**

**Docente: Cristian Boyain**

**25/05/2025**

**Versión: 1.0**

**1. Introducción**

**1.1 Propósito**

Este documento tiene como objetivo definir el diseño técnico de la aplicación **Sistema de Control de Metas Personales**. El propósito del documento es proporcionar una guía clara y detallada sobre los componentes principales del sistema, sus interacciones y cómo estos están organizados bajo la arquitectura Model-View-ViewModel (MVVM). Además, este diseño pretende garantizar que los desarrolladores involucrados comprendan plenamente el funcionamiento del sistema, facilitando su construcción y mantenimiento futuro.

**1.2 Casos de uso**

Los casos de uso tomados para hacer el diseño son :

| **Primera Iteración** |
| --- |
| **CU-1 Crear Meta** |
| **CU-2 Consultar Metas** |
| **CU-3 Modificar Meta** |
| **CU-4 Eliminar Meta** |
| **CU-5 Marcar Meta como Completada** |
| **CU-6 Ver Estadísticas** |
| **CU-7 Visualizar Panel Principal** |

**1.3 Referencias**

* *SRS - Sistema de Control de Metas Personales*

**1.4 Definiciones, acrónimos, abreviaturas**

| **Abreviaturas** | **Definición** |
| --- | --- |
| **SCMP** | **Sistema de Control de Metas Personales** |
| **UI** | **Usuario Individual (persona que gestiona sus metas personales)** |
| **CU** | **Identificador de Casos de Uso** |
| **MVVM** | **Arquitectura Model-View-ViewModel** |
| **WPF** | **Windows Presentation Foundation** |

**2. Tecnologías y Herramientas**

El sistema será desarrollado utilizando:

* **Framework:** WPF (Windows Presentation Foundation) con .NET Framework, implementando el patrón **MVVM**
* **Lenguaje de programación:** C#
* **Base de datos:** SQLite
* **Arquitectura:** MVVM (Model-View-ViewModel)
* **Plataforma:** Windows Desktop Application

La elección de estas herramientas para desarrollar el sistema aporta beneficios significativos en cuanto a rendimiento, usabilidad, seguridad y facilidad de desarrollo:

**2.1 WPF con MVVM**

* **Separación de responsabilidades:** Clara división entre lógica de presentación, lógica de negocio y datos
* **Testabilidad:** Permite realizar pruebas unitarias de la lógica de negocio sin dependencias de UI
* **Mantenibilidad:** Código más organizado y fácil de mantener
* **Binding de datos:** Sincronización automática entre la interfaz y los datos

**2.2 SQLite**

* **Portabilidad:** Base de datos embebida que no requiere instalación de servidor
* **Rendimiento:** Excelente para aplicaciones de escritorio con datos locales
* **Simplicidad:** Fácil configuración y mantenimiento
* **Confiabilidad:** Base de datos robusta y ampliamente utilizada

**2.3 C#**

* **Productividad:** Lenguaje moderno con características avanzadas
* **Integración:** Perfecta integración con el ecosistema .NET y WPF

**3. Arquitectura del Sistema**

**3.1 Descripción y justificación de la arquitectura**

**3.1.1 Modelo**

Gestiona la lógica de negocio, las entidades de dominio y el acceso a datos. Su función principal es encapsular las reglas de negocio, validar datos y proporcionar servicios para la manipulación de metas personales.

**3.1.2 Vista**

Se encarga de mostrar la información al usuario y recibir sus interacciones. La Vista está diseñada para ser declarativa mediante XAML, siendo independiente de la lógica de negocio y comunicándose únicamente con el ViewModel a través de data binding.

**3.1.3 ViewModel**

Actúa como intermediario entre la Vista y el Modelo. Contiene la lógica de presentación, expone propiedades y comandos que la Vista puede enlazar, y coordina las operaciones entre la interfaz de usuario y la lógica de negocio.

**3.1.4 Justificación del uso de MVVM**

Se eligió MVVM para el Sistema de Control de Metas Personales debido a:

* **Separación clara de responsabilidades:** Cada capa tiene una función específica bien definida
* **Testabilidad mejorada:** La lógica de presentación puede ser probada sin dependencias de UI
* **Data Binding:** Sincronización automática entre la interfaz y los datos
* **Escalabilidad:** Facilita el crecimiento y mantenimiento del sistema a largo plazo
* **Estándar en WPF:** Patrón recomendado y ampliamente adoptado para aplicaciones WPF

**4. Componentes del Sistema**

Para el **Sistema de Control de Metas Personales**, los componentes se distribuyen en las capas **MVVM** de la siguiente manera:

**4.1 Modelo (Model)**

El Modelo contiene la lógica de negocio, las entidades de dominio y los servicios de acceso a datos.

**Componentes del Modelo:**

**Entidades:**

* **Meta:** ID de meta, título, descripción, fecha de inicio, fecha objetivo, fecha de finalización, estado
* **EstadoMeta:** Enumeración con valores (Pendiente, EnProceso, Completada, Vencida)

**Servicios:**

* **MetaService:** Lógica de negocio para gestión de metas
* **EstadisticaService:** Cálculo de métricas y estadísticas
* **DataService:** Acceso a la base de datos SQLite

**4.2 Vista (View)**

La Vista es responsable de la interfaz de usuario, mostrando los datos proporcionados por el ViewModel y capturando las interacciones del usuario.

**Pantallas/Views y sus componentes:**

* **MainWindow:** Ventana principal de la aplicación con navegación central
* **PanelPrincipalView:** Dashboard con resumen ejecutivo y navegación principal
* **CrearEditarMetaView:** Formulario para crear y modificar metas
* **ListaMetasView:** Vista tabular de todas las metas con opciones de filtrado
* **EstadisticasView:** Panel con gráficos y métricas de rendimiento
* **ConfirmacionView:** Diálogos de confirmación para acciones críticas

**4.3 ViewModel**

El ViewModel maneja la lógica de presentación y actúa como intermediario entre la Vista y el Modelo.

**Componentes del ViewModel:**

* **MainViewModel:** Coordinador principal de la aplicación y navegación
* **PanelPrincipalViewModel:** Gestión del dashboard y resumen de datos
* **CrearEditarMetaViewModel:** Lógica para formularios de metas
* **ListaMetasViewModel:** Gestión de la lista de metas y filtros
* **EstadisticasViewModel:** Preparación de datos estadísticos para visualización

**5. Diagrama de clases**

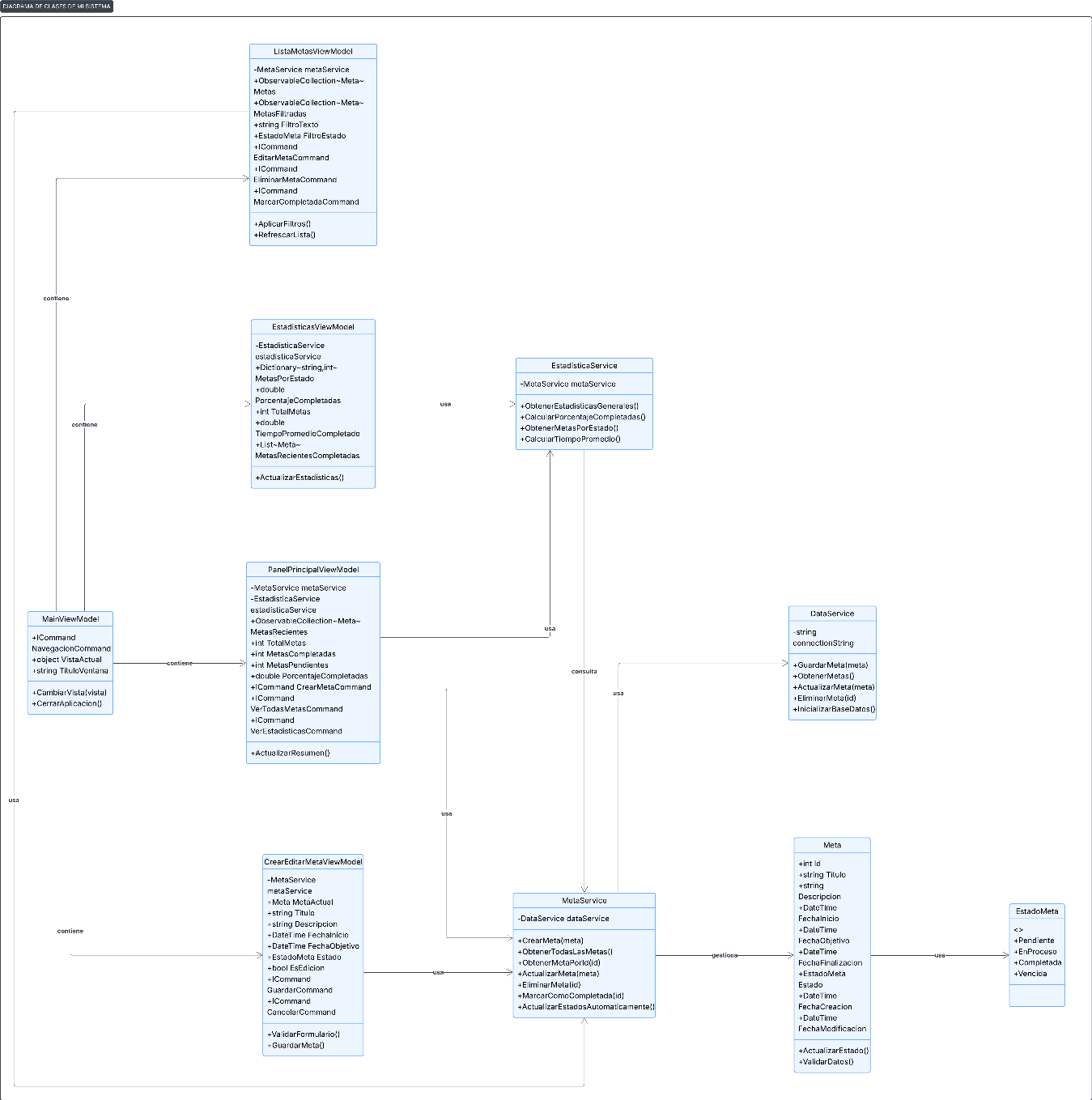


Figura 1 – Diagrama de clases.

**6. Diseño de base de datos**

**Estructura de la base de datos SQLite:**

**Tabla: Metas**

* Id (INTEGER PRIMARY KEY AUTOINCREMENT)
* Titulo (TEXT NOT NULL)
* Descripcion (TEXT)
* FechaInicio (DATETIME NOT NULL)
* FechaObjetivo (DATETIME)
* FechaFinalizacion (DATETIME)
* Estado (INTEGER NOT NULL) -- 0:Pendiente, 1:EnProceso, 2:Completada, 3:Vencida
* FechaCreacion (DATETIME NOT NULL)
* FechaModificacion (DATETIME NOT NULL)

**Tabla: Configuraciones**

* Clave (TEXT PRIMARY KEY)
* Valor (TEXT)

La base de datos mantiene un diseño simple y eficiente, optimizado para operaciones locales rápidas y almacenamiento mínimo de espacio.