Universidad Autónoma de Zacatecas
Unidad Académica de Ingeniería Eléctrica

Ingeniería de Software

Análisis y Diseño Orientado a Objetos
DEFINICIÓN DEL PROYECTO

Alumno:
Marcos de Jesús Arredondo Rebollo

Profesor: Cristian Boyain

Fecha: 11 de abril de 2025

Definición del proyecto

Propósito del Proyecto:

El objetivo de este proyecto es desarrollar un Sistema de Control de Metas Personales que permita a los usuarios planificar, visualizar y analizar el progreso de sus objetivos personales, académicos y profesionales. Este sistema busca resolver el problema de la desorganización y falta de seguimiento que ocurre cuando las metas se registran de manera informal, proporcionando una herramienta digital para mejorar la productividad personal y aumentar las probabilidades de alcanzar los objetivos propuestos.

Alcance:

El sistema incluirá las siguientes funcionalidades:

- Registro de nuevas metas personales con título, descripción, fechas y estado.
- Consulta y visualización de metas registradas en un panel principal.
- Modificación y eliminación de metas existentes.
- Visualización de estadísticas de avance (metas completadas, en proceso o vencidas).
- Panel principal con mensaje de bienvenida y resumen del estado de las metas.
- Navegación intuitiva entre las diferentes secciones del sistema.

Análisis y Diseño Orientado a Objetos

Principios Fundamentales y su Aplicación

Abstracción

Implementación: El sistema modelará conceptos como Meta, GestorDeMetas y Estadísticas, capturando sus características esenciales.

Justificación: La abstracción nos permite reducir la complejidad del mundo real a modelos computacionales manejables, enfocándonos solo en los aspectos relevantes para la gestión de metas personales. Esto facilita tanto el diseño como la implementación del sistema, permitiéndonos crear representaciones claras de conceptos complejos.

Encapsulamiento

Implementación: Las entidades como Meta encapsularán sus datos (id, título, descripción, fechas, estado) y proporcionarán métodos controlados para su manipulación (EstaVencida(), MarcarComoCompletada()).

Justificación: El encapsulamiento es crucial para garantizar la integridad de los datos de las metas del usuario. Al ocultar los detalles de implementación y exponer solo interfaces bien definidas, evitamos modificaciones inadecuadas a los datos y facilitamos cambios futuros en la implementación sin afectar al resto del sistema.

Herencia

Implementación: Aunque inicialmente no se contempla una jerarquía compleja, se diseñará teniendo en cuenta la posibilidad de extender la clase Meta en futuras versiones (por ejemplo, MetaRecurrente, MetaGrupal).

Justificación: La herencia nos permitirá en el futuro ampliar las funcionalidades del sistema manteniendo un diseño coherente. Por ejemplo, si decidimos implementar diferentes tipos de metas con comportamientos específicos, podremos reutilizar el código de la clase base Meta y solo implementar las diferencias en las clases derivadas.

Polimorfismo

Implementación: Los métodos como EstaVencida() podrían comportarse de manera diferente según el tipo de meta, adaptándose a los requisitos específicos de cada subclase.

Justificación: El polimorfismo permitirá que el sistema maneje diferentes tipos de metas de manera uniforme, facilitando la extensibilidad. Por ejemplo, una MetaRecurrente podría tener una implementación diferente del método EstaVencida() comparado con una meta regular, pero el sistema podría tratarlas a través de la misma interfaz.

Patrones de Diseño Seleccionados

Patrón MVVM (Model-View-ViewModel)

Implementación: La arquitectura del sistema seguirá el patrón MVVM con:

Modelo: Clases de dominio (Meta, Estadísticas)

Vista: Interfaces de usuario en WPF para el panel principal, formularios y estadísticas

ViewModel: Intermediarios que conectan el modelo con la vista, gestionando la lógica de presentación

Justificación: El patrón MVVM es particularmente adecuado para este proyecto porque:

- Facilita la separación de la lógica de negocio de la interfaz de usuario, lo que mejora la mantenibilidad del código.
- Permite un enlace de datos bidireccional, simplificando la actualización de la interfaz cuando cambian los datos y viceversa.
- Mejora la capacidad de prueba al aislar los componentes.
- Es especialmente efectivo para aplicaciones WPF, que es la tecnología elegida para este proyecto.
- Permite que varios desarrolladores trabajen simultáneamente en diferentes componentes del sistema.

Patrón Singleton

Implementación: Se aplicará en el GestorDeMetas para garantizar que solo exista una instancia que coordine todas las operaciones relacionadas con las metas.

Justificación: El patrón Singleton es apropiado para este componente porque:

- Necesitamos un punto centralizado para gestionar todas las metas.
- Evita conflictos que podrían surgir si existieran múltiples gestores operando simultáneamente.
- Proporciona un acceso global y controlado a la funcionalidad de gestión de metas.
- Asegura la consistencia de los datos al tener un único punto de acceso y modificación.

Patrón Observer

Implementación: Se utilizará para notificar automáticamente a la interfaz de usuario cuando cambien los datos de las metas o las estadísticas.

Justificación: El patrón Observer permite:

- Mantener sincronizadas diferentes partes de la aplicación cuando cambian los datos.
- Actualizar automáticamente las estadísticas cuando se modifican, completan o vencen las metas.
- Reflejar inmediatamente los cambios en todas las vistas relevantes.
- Reducir el acoplamiento entre componentes al establecer relaciones de tipo publicador-suscriptor.

Patrón Repository

Implementación: Se implementará para abstraer y centralizar la lógica de acceso a datos, separando la lógica de negocio de la persistencia.

Justificación: El patrón Repository:

- Aísla la lógica de acceso a datos del resto de la aplicación.
- Facilita el cambio del mecanismo de almacenamiento en el futuro.
- Simplifica las pruebas al permitir la sustitución por repositorios de prueba.
- Proporciona una interfaz consistente para operaciones CRUD sobre las metas.

UML's Planeados

Diagrama de Casos de Uso:

Para visualizar las interacciones entre el usuario y el sistema, incluyendo casos como "Registrar Meta", "Consultar Metas", "Modificar Meta", "Eliminar Meta" y "Ver Estadísticas".

Diagrama de Clases:

Para modelar la estructura estática del sistema, representando las clases (Meta, GestorDeMetas, Estadísticas) con sus atributos, métodos y relaciones.

Diagrama de Secuencia:

Para mostrar las interacciones dinámicas entre objetos en escenarios clave como el registro de una nueva meta o la actualización de estadísticas.

Diagrama de Estados:

Para representar los diferentes estados por los que puede pasar una meta (Activa, Completada, Vencida) y las transiciones entre estos estados.

Diagrama de Componentes:

Para visualizar la organización y dependencias entre los componentes principales del sistema según el patrón MVVM (Modelos, Vistas, ViewModels).

Calendario General

Periodo del proyecto: 12 de abril al 13 de junio de 2025

Duración de cada sprint: 2 semanas

Total de sprints: 4 sprints completos + fase de cierre

Sprint 1: Preparación y Análisis
Fecha: 12 de abril - 25 de abril

Actividades:

Elaboración del documento de inicio del proyecto

Definición detallada de requisitos funcionales y no funcionales

Análisis de dominio del problema

Desarrollo de diagramas UML iniciales (casos de uso y clases)

Diseño de la arquitectura siguiendo el patrón MVVM

Configuración del entorno de desarrollo y repositorio de código

Entregables:

Documento de inicio del proyecto completo

Backlog del producto priorizado

Diagrama de casos de uso

Diagrama de clases inicial

Documento de arquitectura del sistema

Repositorio de código configurado

Revisión y retrospectiva del sprint

Sprint 2: Diseño y Base de Implementación
Fecha: 26 de abril - 9 de mayo

Actividades:

Diseño detallado de las clases del modelo (Meta, GestorDeMetas, Estadísticas)

Diseño de la base de datos para almacenamiento de metas

Creación de prototipos de interfaces de usuario

Implementación inicial de las clases del modelo

Desarrollo de la base de datos y componentes básicos de acceso a datos

Implementación del patrón Singleton para GestorDeMetas

Entregables:

Diseño detallado de clases y base de datos

Prototipos de interfaz de usuario

Modelo de dominio implementado a nivel básico

Estructura de base de datos funcionando

Revisión y retrospectiva del sprint

Sprint 3: Implementación Core
Fecha: 10 de mayo - 23 de mayo

Actividades:

Implementación completa de las clases del modelo

Desarrollo completo de componentes de acceso a datos

Implementación de ViewModels básicos

Creación de interfaces de usuario principales

Implementación del CRUD básico de metas

Testing unitario de componentes core

Entregables:

Modelo de dominio completamente implementado

Componentes de acceso a datos funcionando

Interfaces de usuario básicas operativas

Primera versión funcional del sistema (MVP - Producto Mínimo Viable)

Conjunto inicial de pruebas unitarias

Revisión y retrospectiva del sprint

Sprint 4: Funcionalidades Principales

Fecha: 24 de mayo - 6 de junio

Actividades:

Implementación completa del CRUD de metas

Desarrollo de la funcionalidad de estadísticas

Implementación del patrón Observer para actualizaciones de UI

Mejoras en la interfaz de usuario según feedback

Implementación de características adicionales prioritarias:

Categorización de metas

Filtros básicos de búsqueda

Validaciones y manejo de errores

Testing de integración

Entregables:

Sistema con funcionalidad completa de gestión de metas

Módulo de estadísticas implementado

UI mejorada con feedback visual

Funcionalidades adicionales prioritarias implementadas

Informe de pruebas de integración

Documentación técnica actualizada

Revisión y retrospectiva del sprint

Fase de Cierre: Refinamiento y Presentación

Fecha: 7 de junio - 13 de junio

Actividades:

Optimización de rendimiento

Corrección de bugs identificados

Implementación de características adicionales secundarias (si el tiempo lo permite):

Visualización de metas en formato de calendario

Recordatorios para metas próximas a vencer

Pulido final de la interfaz de usuario

Preparación de la documentación final

Creación de manual de usuario detallado

Elaboración de la presentación del proyecto

Testing de aceptación final

Preparación del entregable final

Entregables:

Sistema completo y refinado

Documentación técnica finalizada

Manual de usuario

Presentación del proyecto

Informe final de pruebas

Entrega del código fuente documentado

Retrospectiva final del proyecto