Informe Práctica II

Andres Bedoya Cano

Alejandro Garzón

Descripción de requisitos funcionales

Jonathan Calle

Universidad de Medellín

Código limpio

2024

# Informe del Proyecto: Sistema de Gestión de Tareas Personales

## Introducción

En el presente informe se desarrollará la primera parte del proyecto denominado **"Sistema de Gestión de Tareas Personales"**. Este proyecto tiene como objetivo aplicar principios avanzados de Programación Orientada a Objetos (POO) y buenas prácticas de desarrollo de software, como el uso de principios SOLID y patrones de diseño. Además, se integrarán pruebas automatizadas, manejo de bases de datos, y una arquitectura escalable que permita el crecimiento y mejora continua del código con el fin de cumplir la primera entrega

Este documento describe los requisitos funcionales y no funcionales que debe cumplir el sistema, explicando cómo la aplicación satisface las necesidades del usuario final y aborda los problemas relacionados con la organización eficiente de tareas personales.

## 1. “Qué problema aborda la aplicación”

Las personas suelen gestionar varias tareas y pendientes de manera manual, lo que lleva a falta de organización, causando que se olviden tareas importantes, se pierda el control de prioridades o se distribuya mal el tiempo. Esta aplicación tiene como objetivo resolver estos problemas ofreciendo una plataforma donde los usuarios puedan gestionar sus tareas de forma clara, organizada y estructurada.

El "Sistema de Gestión de Tareas Personales" permitirá a los usuarios agregar, modificar, eliminar y organizar tareas, clasificándolas por prioridad, fecha límite, o etiquetas, facilitando así su administración y mejorando la productividad.

## 2. Requisitos Funcionales

Aquí detallamos las acciones y capacidades que el sistema debe ofrecer para cumplir con las expectativas del usuario.

### **2.1. Gestión de Tareas**

El sistema permitirá a los usuarios crear nuevas tareas con la siguiente información: título, descripción, fecha de creación, fecha límite (opcional), prioridad (alta, media, baja) y estado (pendiente o completada).

Los usuarios podrán editar tareas ya existentes, modificando el título, descripción, fecha límite y prioridad.

Los usuarios podrán marcar una tarea como completada.

Los usuarios podrán eliminar tareas que ya no sean necesarias.

### **2.2. Organización y Clasificación de Tareas**

El sistema permitirá clasificar las tareas según los siguientes criterios:

* + Prioridad: Alta, Media, Baja.
  + Fecha límite: Mostrar tareas ordenadas por la proximidad de su vencimiento.
  + Etiquetas: Los usuarios podrán asignar etiquetas personalizadas para organizar tareas en categorías (ej. "Trabajo", "Hogar", "Estudios").

### **2.3. Filtros de Búsqueda**

Los usuarios podrán buscar tareas utilizando diferentes filtros, tales como:

* + Palabras clave: Buscar en el título o descripción de las tareas.
  + Fecha límite: Filtrar tareas por una fecha límite específica o por un rango de fechas.
  + Estado: Buscar tareas pendientes o completadas.
  + Etiqueta: Filtrar tareas por una o varias etiquetas asignadas.

### **2.4. Persistencia de Datos**

El sistema almacenará las tareas en una base de datos SQLite, garantizando la persistencia de los datos entre sesiones.

Las tareas se cargarán automáticamente al iniciar la aplicación.

Todas las acciones realizadas (crear, editar, eliminar tareas) deberán reflejarse inmediatamente en la base de datos.

### **2.5. Interacción con el Usuario**

El sistema tendrá una interfaz de usuario (de manera gráfica) que sea clara y fácil de usar.

## 

## 

## 

## 3. Definición de Requisitos No Funcionales

Los requisitos no funcionales simplemente aseguran que la aplicación funcione de manera eficiente, sea mantenible y escalable, y garantice una buena experiencia de usuario.

### **3.1. Rendimiento**

La aplicación debe ser capaz de manejar grandes volúmenes de tareas (más de 100 tareas) sin una caída notable en el rendimiento.

Las búsquedas y filtrados de tareas deben realizarse en menos de 1 segundo.

### **3.2. Escalabilidad y Mantenibilidad**

El sistema debe estar diseñado siguiendo los principios SOLID, facilitando la extensibilidad y la fácil modificación del código.

Cada clase del sistema debe tener una única responsabilidad, favoreciendo el desacoplamiento y la modularidad.

El código debe estar preparado para escalabilidad futura, permitiendo agregar nuevas características sin necesidad de grandes cambios estructurales.

### **3.3. Usabilidad**

La interfaz de usuario debe ser intuitiva y accesible, permitiendo que los usuarios interactúen con la aplicación sin mucho esfuerzo

Las opciones de menú deben estar bien organizadas, y las instrucciones deben ser claras y concisas.

### **3.4. Pruebas y Automatización**

El sistema debe contar con pruebas unitarias automatizadas para verificar el correcto funcionamiento de las funcionalidades principales.