

CHECKLIST

TRABAJO INTEGRADOR PROGRAMACION WEB



12 de noviembre de 2024

ALUMNO: LUCIANO, SELENE

DOCENTE: ING. HAINCH, JESICA

Índice

[Descripción general 2](#_Toc182314954)

[Objetivos 2](#_Toc182314955)

[Usuario final 2](#_Toc182314956)

[Características principales 2](#_Toc182314957)

[Requisitos funcionales 3](#_Toc182314958)

[Requerimientos de usuario 3](#_Toc182314959)

[Requerimientos administrativos 4](#_Toc182314960)

[Arquitectura 5](#_Toc182314961)

# Descripción general

## Objetivos

El propósito principal del sitio es proporcionar una plataforma de gestión de tareas en la que los usuarios puedan crear, organizar y gestionar listas de tareas de manera eficiente ayudando a los usuarios a organizar mejor su tiempo y aumentar su productividad.



## Usuario final

El sitio está diseñado para **usuarios** que buscan una manera práctica de gestionar tanto sus tareas personales como profesionales y para los **administradores** quesupervisaran la plataforma, gestionaran los usuarios y analizaran el rendimiento general del sitio.

## Características principales

* Gestión de listas y tareas: Los usuarios pueden crear listas, añadir tareas, editarlas y organizarlas según dificultad y fecha de vencimiento.
* Control de estado de tareas: Los usuarios pueden marcar tareas como "pendientes" o "completadas" y actualizar sus detalles.
* Visualización de progreso mediante gráficos:
  + Gráfico de barras: Muestra la cantidad de tareas en los estados "pendientes", “vencidas” y "completadas".
  + Gráfico circular: Representa el número de tareas en cada lista para una visión rápida del volumen de trabajo.
* Perfil de usuario: Permite al usuario personalizar su información de perfil.

# Requisitos funcionales

## Requerimientos de usuario

* El sistema debe permitir a los usuarios crear una cuenta para poder acceder a la plataforma.
* El sistema debe permitir a los usuarios ingresar a sus cuentas y cerrar sesión.
* El sistema debe permitir a los usuarios crear nuevas listas para organizar sus tareas.
* El sistema debe permitir a los usuarios modificar el nombre y los detalles de las listas creadas.
* El sistema debe permitir a los usuarios eliminar listas que ya no necesiten.
* El sistema debe mostrar a los usuarios una vista general de todas las listas creadas.
* El sistema debe permitir a los usuarios agregar nuevas tareas a una lista específica.
* El sistema debe permitir a los usuarios modificar los detalles de las tareas, incluyendo el título, la descripción, la fecha de vencimiento y la prioridad.
* El sistema debe permitir a los usuarios cambiar el estado de las tareas entre "completadas" y "no completadas".
* El sistema debe permitir a los usuarios eliminar tareas que ya no necesiten.
* El sistema debe permitir a los usuarios filtrar tareas por estado (pendientes o completadas) y por prioridad (baja, media, alta).
* El sistema debe mostrar a los usuarios un gráfico que indique la cantidad de tareas en los estados "pendientes", "completadas" y "no completadas".
* El sistema debe mostrar a los usuarios un gráfico circular o de pastel que visualice cada lista y la proporción de tareas que contiene.

## Requerimientos administrativos

* El sistema debe permitir al administrativo acceder a un panel administrativo donde pueda ver, crear, modificar y eliminar cuentas de todos los usuarios registrados en el sistema.
* El sistema debe permitir al administrativo revisar y supervisar todas las listas y tareas creadas en la plataforma, independientemente del usuario que las haya generado.
* El sistema debe proporcionar al administrativo herramientas para auditar las actividades de los usuarios (por ejemplo, creación y eliminación de listas y tareas) con fines de seguimiento y análisis.
* El sistema debe permitir al administrativo ver y gestionar los registros de errores y logs de actividad para solucionar problemas y mejorar la estabilidad de la plataforma.

# Arquitectura

Se utilizo el modelo **MVC (Model-View-Controller)** que es un patrón de diseño arquitectónico utilizado para dividir la aplicación en tres componentes interconectados, lo que separa la lógica de negocio, la interfaz de usuario y la gestión de las aplicaciones del usuario. Esta separación ayuda a organizar mejor el código, facilita la colaboración entre desarrolladores y simplifica el mantenimiento y escalabilidad de las aplicaciones. A continuación, se explican sus partes implementadas en el proyecto:

* **Modelo**: El modelo es la capa encargada de definir la estructura de la base de datos y las relaciones entre los datos; en Django, cada modelo se define como una clase de Python.
* **Vista**: Las vistas son funciones o clases en Python que gestionan las solicitudes HTTP y devuelven una respuesta. Aquí es donde se procesa la lógica de negocio. Las vistas pueden recuperar datos del modelo, procesarlos y pasar los resultados al template para que se rendericen.
* **Controlador**: Los templates son archivos de presentación que combinan HTML con las etiquetas de plantilla de Django. Estas etiquetas permiten incrustar contenido dinámico y ejecutar lógica simple, como bucles y condicionales.

