TRABAJO PRÁCTICO - GRUPO 4

Arquitectura y Planificación.

Integrantes:

- Maximiliano Sebastian Barrionuevo
- Ciro Martín López
- Juan Manuel Galante

Profesores:

- Daniel Alvarez.
- Carlos Nelson Jimenez.

Introducción

En este documento se detalla la arquitectura y planificación del sistema “Portal de Noticias”. Se presentan las decisiones de diseño adoptadas, la estructura modular de la aplicación y la planificación del desarrollo mediante un modelo de ciclo de vida apropiado.

En primer lugar, se describe la arquitectura del sistema a partir de sus módulos principales. Estos módulos cumplen propósitos específicos e interactúan entre sí según comportamientos bien definidos. Definir esta arquitectura permite entender la correspondencia entre los requerimientos funcionales y los elementos arquitectónicos que los implementan.

La solución adoptada se basa en una arquitectura cliente-servidor, complementada con el patrón de diseño Modelo-Vista-Controlador (MVC). Esta elección facilita la separación de responsabilidades, mejora la mantenibilidad del código y promueve la escalabilidad del sistema.

Por último, se incluye una planificación estructurada con tareas organizadas cronológicamente, sus dependencias y un diagrama de Gantt, que permitió coordinar las diferentes etapas del desarrollo de manera eficiente.

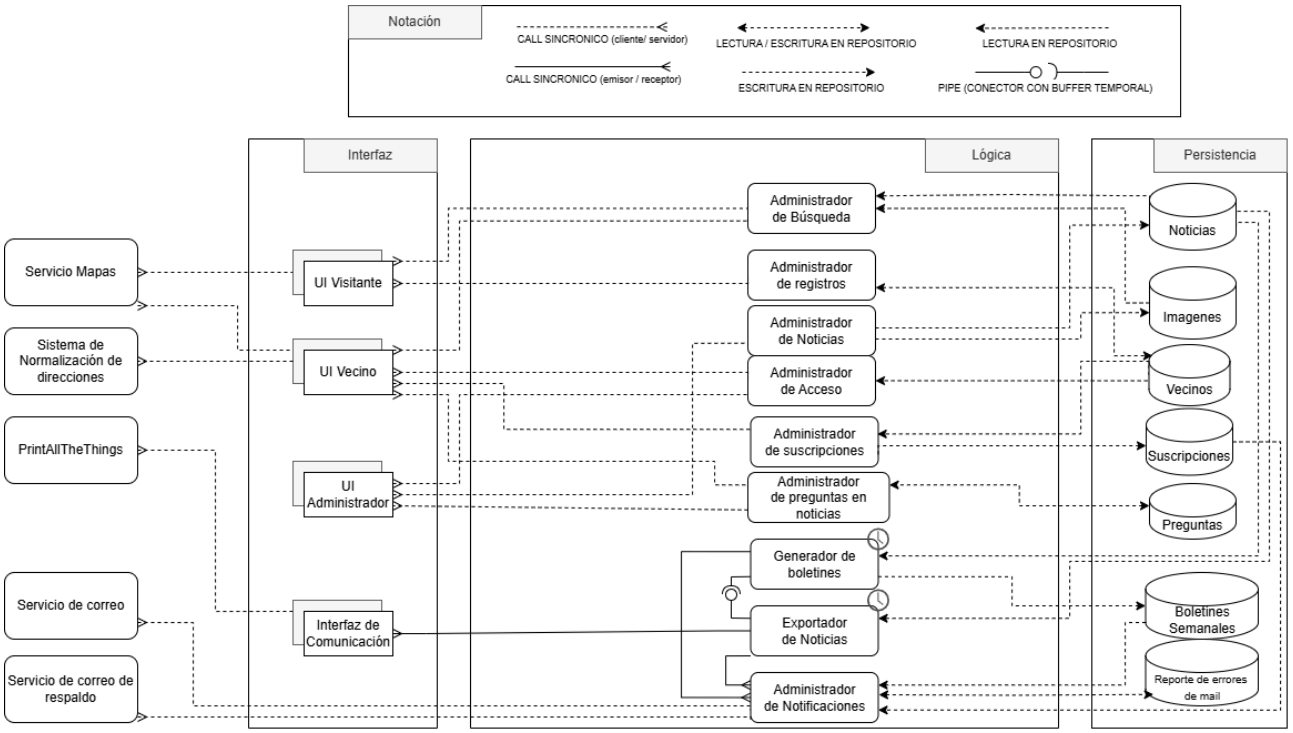
Arquitectura

Para el desarrollo del sistema “Portal de Noticias” optamos por la arquitectura basada en el modelo cliente - servidor, en la que el cliente realiza solicitudes al servidor y se comunica con servicios externos, como la normalización de dirección y la visualización del mapa. Esta estructura es conveniente para el desarrollo de aplicaciones webs, ya que permite la separación de las responsabilidades, la distribución de la lógica y la escalabilidad del sistema en el futuro
.
Además, la aplicación sigue el patrón de diseño **Modelo-Vista-Controlador (MVC)**. Este patrón organiza el código en tres componentes principales:

- Modelo (Persistencia): Encargado de la gestión de datos, como las noticias, preguntas y boletines.
- Vista (Interfaz): Compuesta por las interfaces gráficas utilizadas por visitantes, vecinos y administradores.
- Controlador (Lógica): Gestiona la lógica de negocio, el flujo de datos entre modelo y vista, y la interacción con APIs externas.

Esta organización facilita el mantenimiento del código, la reutilización de componentes y la separación clara de responsabilidades dentro de la interfaz web.

Diagrama



Decisiones de Diseño

Debido a la limitación de tiempo decidimos no usar una arquitectura en capas (Layered), y optamos por el estilo arquitectónico “cliente-servidor” con patrón MVC, ideal para aplicaciones webs como esta. Esta arquitectura nos ofrece simplicidad sin perder modularidad.

Justificación de la decisión de diseño

La decisiones de diseño se basa en los siguientes beneficios:

- Modularidad: Este diseño nos brinda la posibilidad de separar responsabilidades entre cliente y servicios.
- Escalabilidad: En un futuro, se podría conectar a un backend real sin cambiar la interfaz.
- Simplicidad: Permite implementar funcionalidades en el frontend con claridad.
- Adaptabilidad: Permite integrar fácilmente librerías como Leaflet y Tailwind

Atributos de calidad que nos permite abordar esta estructura:

- Usabilidad: Interfaz moderna, clara y responsiva
- Estabilidad: manejo de errores ante fallos en funcionalidades como en la normalización de direcciones.
- Seguridad: Autenticación de cuentas como la de administradores.
- **Escalabilidad**: Estructura preparada para un posible crecimiento de la plataforma.

Descripción de Módulos

A continuación se detallan los módulos principales del sistema:

Módulos de interfaz:

UI Visitantes: Interfaz para usuarios no registrados. Permite visualizar noticias en el listado, en el mapa y buscar por filtros. Además permite registrarse e iniciar sesión para acceder a la interfaz de vecino o administrador.

UI Vecino: Interfaz para usuarios registrados. Además de lo del visitante, puede suscribirse al boletín y realizar preguntas.

UI Administrador: Interfaz exclusiva para administradores. Permite publicar noticias, responder y moderar preguntas, gestionar usuarios y exportar boletines.

Interfaz de Comunicación: Encargada de comunicar las salidas del sistema al exterior, como el envío de boletines o exportaciones a servicios externos.

Módulos de lógica:

Administrador de Búsqueda: Filtra las noticias por título, fecha o contenido. Se conecta con el repositorio de noticias.

Administrador de Registros: Maneja el alta de nuevos vecinos y gestiona sus datos.

Administrador de Noticias: Permite crear, editar y eliminar noticias. También se encarga de gestionar sus imágenes y las direcciones normalizadas.

Administrador de Acceso: Controla el inicio de sesión de los administradores y vecinos, validando credenciales.

Administrador de Suscripciones: Administra a los vecinos suscriptos al boletín, permitiendo dar altas, bajas y consultas.

Administrador de Preguntas en noticias: Permite visualizar las preguntas de los vecinos, marcarlas como respondidas o inapropiadas, y generar respuestas.

Generador de Boletines: Agrupa noticias nuevas para formar los boletines semanales y pasarlas al exportador.

Exportador de Noticias: Exporta las noticias a servicios externos como PrintAllTheThings. Registra errores en caso de fallos.

Administrador de Notificaciones: Se comunica con los servicios de correo (principal y de respaldo) para el envío de boletines.

Componentes externos:

Servicio Mapas: Recibe coordenadas normalizadas para ubicar noticias en un mapa interactivo.

Sistema de Normalización: Recibe direcciones ingresadas por el usuario y devuelve una dirección con coordenadas listas para ubicarlas en el mapa.

PrintAllTheThings: Servicio externo que se encarga de imprimir y exportar las noticias seleccionadas.

Servicio de correo: Recibe los boletines generados y los distribuye a los vecinos suscriptos.

Servicio de correo de respaldo: Recibe los boletines generados y los distribuye a los vecinos suscriptos. Solo se usa este servicio si es que el primer servicio de correo falla.

Persistencia:

Noticias: Almacena todas las noticias publicadas.

Imágenes: Contiene las imágenes adjuntas de cada noticia.

Vecinos: Contiene información de los usuarios registrados.

Suscripciones: Guarda a los vecinos que se suscribieron al boletín.

Preguntas: Almacena preguntas realizadas por los vecinos y sus posibles respuestas.

Boletines Semanales: Guarda los boletines ya generados para evitar duplicados.

Reporte de errores de mail: Registro de fallas de envío por parte de los servicios de correo.

Planificación.

Ciclo de vida del proyecto

El proyecto fue organizado en varias fases sucesivas, con tareas claramente definidas, fechas tentativas y dependencias entre etapas. Para su gestión, se adoptó el modelo de ciclo de vida Sashimi, una variante del modelo en cascada que permite solapamientos parciales entre etapas.

El modelo Sashimi es una variante del modelo en cascada tradicional que permite cierta superposición entre fases como análisis, diseño, implementación y pruebas. Estas características lo vuelven útil para proyectos que constan con plazos ajustados. Al permitirse revisar y mejorar entregas anteriores mientras se avanza con las posteriores, se reduce el riesgo de errores acumulativos y mejora la adaptabilidad del equipo.

El desarrollo se estructuró en tres grandes etapas: Análisis, Diseño e Implementación, con una última fase de despliegue.

Modelo de Ciclo de Vida Adoptado: Sashimi

Este modelo permitió:

- Realizar entregas parciales de funcionalidades.

- Ajustar el análisis y el diseño sin retrasar el desarrollo.
- Coordinar tareas entre los distintos integrantes del equipo con mayor fluidez.
- Aprovechar el solapamiento de etapas para acelerar los tiempos sin comprometer la calidad.

Tareas del proyecto

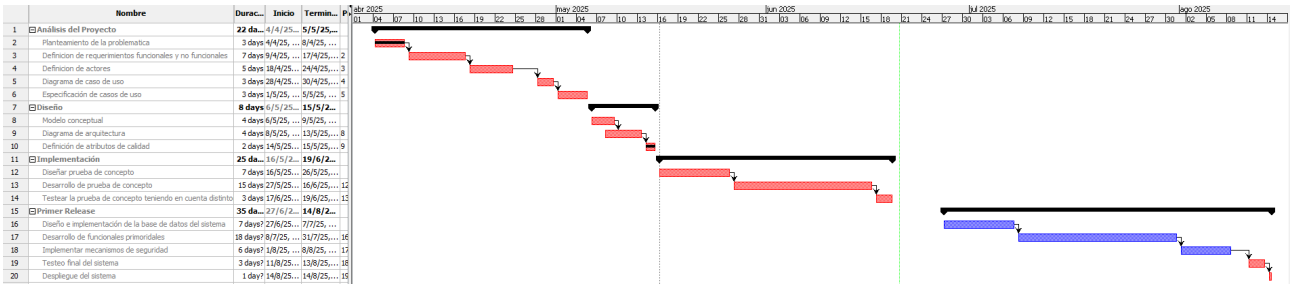
A continuación, se detallan las principales tareas del proyecto, incluyendo una división por etapas, duración estimada de tareas y dependencias generales. Estas actividades fueron reflejadas en el Diagrama de Gantt.

N° de Tarea	Tarea	Tiempo de duración	Dependencias
1	**Análisis del Proyecto**	**22 días**	
2	Planteamiento de la problemática	3 días	
3	Definición de requerimientos funcionales y no funcionales	7 días	2
4	Definicion de actores	5 días	3
5	Diagrama de casos de uso	3 días	4
6	Especificación de casos de uso	3 días	5
7	**Diseño**	**8 días**	
8	Modelo conceptual	4 días	
9	Diagrama de arquitectura	4 días	8
10	Definición de atributos de calidad	2 días	9
11	**Implementación 25 días**	**25 días**	
12	Diseñar prueba de concepto	7 días	

13	Desarrollo de prueba de concepto	15 días	12
14	Testear la prueba de concepto teniendo en cuenta distintos actores	3 días	13
15	**Primer Release**	**35 días**	
16	Diseño e implementación de la base de datos del sistema	7 días	
17	Desarrollo de funcionales primordiales	18 días	16
18	Implementar mecanismos de seguridad	6 días	17
19	Testeo final del sistema	3 días	18
20	Despliegue del sistema	1 días	19

Diagrama de Gantt

El **Diagrama de Gantt** fue elaborado para organizar las tareas del equipo y garantizar el cumplimiento de los plazos establecidos.



Ver el diagrama completo: link

Descargar diagrama: link