**Arquitectura del sistema – Sistema de Gestión Inmobiliaria**

Para el desarrollo del sitio web de la inmobiliaria se implementará una **arquitectura cliente-servidor**, la cual permitirá separar claramente las responsabilidades entre el cliente (interfaz de usuario) y el servidor (lógica del negocio y acceso a datos).

El proyecto se basará en el **patrón de diseño Modelo-Vista-Controlador (MVC)**, lo que facilitará la organización del código y la mantenibilidad del sistema.

* **Modelo:** encargado de gestionar los datos y la lógica del negocio.
* **Vista:** responsable de la presentación de la información al usuario.
* **Controlador:** actúa como intermediario entre el modelo y la vista, procesando las solicitudes del usuario y actualizando la información mostrada.

El desarrollo se realizará **de manera nativa, sin el uso de frameworks externos (como Laravel)**, con el objetivo de comprender y controlar directamente la **lógica interna del sistema**, asegurando una implementación personalizada, flexible y ajustada a las necesidades específicas del proyecto.

**Cliente (Frontend)**

El **cliente** corresponde al **navegador web del usuario final**, encargado de **renderizar la interfaz gráfica del sistema** y ejecutar la **lógica del lado del cliente**, como las validaciones de formularios y la interacción dinámica con los elementos visuales del sitio.

En la arquitectura **Modelo-Vista-Controlador (MVC)**, esta capa implementa la **Vista**, cuya principal función es **presentar la información al usuario** de manera clara, accesible y estéticamente coherente.

Para su desarrollo se emplearán las siguientes tecnologías:

* **HTML (HyperText Markup Language):** se utilizará para **estructurar el contenido** del sitio web, definiendo los elementos básicos de las páginas como textos, encabezados, formularios, botones, enlaces e imágenes. Representa la base sobre la cual se construye la interfaz de usuario.
* **CSS (Cascading Style Sheets):** permitirá **dar formato y estilo visual** al contenido definido en HTML, controlando aspectos como los colores, tipografías, márgenes, alineaciones y disposición de los elementos en pantalla. Gracias a CSS, se logra una interfaz coherente y atractiva para el usuario.
* **SASS (Syntactically Awesome Style Sheets):** es un **preprocesador de CSS** que facilita la escritura y mantenimiento del código de estilos mediante el uso de **variables, anidamientos, mixins y funciones**, lo que mejora la organización y reutilización del diseño visual del sitio.
* **JavaScript:** se empleará para **incorporar interactividad y dinamismo** al sitio web, permitiendo responder a las acciones del usuario en tiempo real, validar datos en los formularios antes de ser enviados al servidor y modificar elementos del DOM sin necesidad de recargar la página.
* **AJAX (Asynchronous JavaScript and XML):** se utilizará para **realizar peticiones asíncronas al servidor**, permitiendo que ciertas partes del sitio se actualicen dinámicamente sin recargar completamente la página. Esto mejora la **experiencia del usuario** al hacer el sitio más ágil y fluido.

En conjunto, estas tecnologías conforman la **capa de presentación del sistema**, garantizando una interfaz moderna, funcional y adaptable a diferentes dispositivos y navegadores.

**Servidor (Backend)**

El **servidor** será el encargado de **procesar la lógica interna del sitio web** y **gestionar la conexión con la base de datos**, asegurando el correcto funcionamiento del sistema y la comunicación entre la capa de presentación (Frontend) y los datos almacenados.

En esta etapa se implementarán el **Modelo** y el **Controlador** del patrón **Modelo-Vista-Controlador (MVC)**, que permiten estructurar el código de manera modular, separando la lógica del negocio del manejo de datos y de la presentación al usuario.

Para el desarrollo del backend se emplearán las siguientes herramientas y tecnologías:

* **PHP (Hypertext Preprocessor):** se utilizará en su forma **pura (sin frameworks externos)** para **codificar la lógica del sistema**, procesar las solicitudes del usuario y gestionar la interacción entre el cliente, el controlador y la base de datos. PHP permitirá implementar las reglas de negocio y generar dinámicamente las páginas que se mostrarán al usuario.
* **Patrón Active Record:** se aplicará para **gestionar la persistencia de datos** de manera orientada a objetos, asociando cada clase con una tabla de la base de datos. Este patrón simplifica las operaciones de consulta, inserción, modificación y eliminación de registros, reduciendo el código repetitivo y mejorando la organización del modelo.
* **MySQL:** será el **sistema gestor de bases de datos relacional (SGBD)** utilizado para **almacenar y administrar la información** del sistema, incluyendo los datos de propiedades, usuarios y consultas realizadas. Su integración con PHP permite ejecutar operaciones de manera eficiente y segura mediante consultas SQL.
* **Apache HTTP Server:** se empleará como **servidor web**, responsable de **recibir las solicitudes del navegador y entregar las respuestas generadas por PHP**. Apache gestionará la comunicación entre el cliente y el servidor, garantizando la correcta interpretación de los scripts PHP y el acceso a los recursos del sitio.
* **XAMPP:** se utilizará para **configurar el entorno de desarrollo local**, ya que incluye Apache, PHP y MySQL integrados en un único paquete. Esto facilita la instalación, configuración y pruebas del sistema antes de su implementación en un servidor de producción.
* **Gulp.js:** se implementará como **herramienta de automatización de tareas**, permitiendo optimizar el flujo de desarrollo mediante la **compilación automática de archivos SASS y JavaScript**, la minificación de código y la actualización en tiempo real del sitio durante las pruebas.
* **Node.js:** será el **entorno de ejecución de JavaScript del lado del servidor** necesario para el funcionamiento de Gulp. Node.js permitirá ejecutar scripts que automaticen procesos y gestionen dependencias durante el desarrollo del proyecto.
* **NPM (Node Package Manager):** se utilizará para **instalar y administrar las dependencias** necesarias, tales como los paquetes requeridos por SASS, Gulp y otras herramientas auxiliares que contribuyan a mejorar la eficiencia del entorno de desarrollo.

En conjunto, estas tecnologías conforman la **capa de lógica y gestión del sistema**, garantizando un funcionamiento estable, eficiente y completamente controlado por el equipo de desarrollo, sin depender de frameworks externos.