Elección de servidor web y de BBDD

Servidor web:

Apache HTTP Server

Arquitectura: process-based, cada conexión crea un proceso/hilo → mayor consumo de memoria.

Módulos: gran cantidad (mod_ssl, mod_auth, mod_rewrite…), fáciles de activar/desactivar.

Configuración: permite .htaccess (config local por directorio).

Seguridad: muy auditado, pero el uso de .htaccess puede abrir vulnerabilidades si no se controla.

PROS	CONTRAS
Altamente compatible con aplicaciones *legacy* (PHP clásico, WordPress, etc.).	Escalabilidad limitada por su arquitectura.
Comunidad y soporte muy amplios.	Menor rendimiento en escenarios de alta concurrencia.
Configuración rápida en entornos sencillos.	

Nginx

Arquitectura: event-driven, asynchronous → maneja decenas de miles de conexiones simultáneas con bajo consumo de recursos.

Rol doble: servidor web y reverse proxy/balanceador de carga.

Configuración: centralizada (nginx.conf y sites-available/); no existe .htaccess.

Seguridad: diseño minimalista → menor superficie de ataque.

PROS	CONTRAS
Excelente rendimiento en aplicaciones modernas y sitios estáticos.	Configuración más rígida (no .htaccess).
Perfecto como reverse proxy delante de backends (Node.js, Django, etc.).	Requiere más conocimientos iniciales que Apache.
Bajo consumo de CPU y memoria.	

Nuestra elección: Apache HTTP Server

Motivo: Hemos elegido Apache porque es más sencillo de configurar y administrar en comparación con Nginx, y dado que este no es un proyecto muy grande, la diferencia de rendimiento no será crítica. Además, la compatibilidad con `.htaccess` nos facilita realizar configuraciones rápidas y locales sin necesidad de modificar archivos globales.

Servidor BBDD:

MySQL

Arquitectura: motor principal InnoDB (ACID), orientado a simplicidad y buen rendimiento en CRUD básicos.

Características: integración fácil con entornos LAMP, soporte gráfico (phpMyAdmin, Workbench).

PROS	CONTRAS
Muy popular y documentado.	Bajo control de Oracle.
Fácil de usar y configurar.	Menos potente en consultas complejas.
Amplia compatibilidad con frameworks web.	Optimizador de consultas limitado frente a PostgreSQL.

MariaDB

Arquitectura: fork libre y 100% compatible con MySQL, añade motores extra (Aria, ColumnStore).

Características: rendimiento mejorado en algunas consultas, totalmente open source

PROS	CONTRAS
Libre, comunitario y sin dependencia de Oracle.	Comunidad más pequeña.
Compatible con MySQL.	Menos adoptado en grandes proyectos que PostgreSQL.
Mejoras de rendimiento y nuevos motores.	Algunas funciones aún menos maduras.

PostgreSQL

Arquitectura: MVCC, ACID completo, diseñado para cargas pesadas y consultas avanzadas.

Características: soporte para JSONB, arrays, funciones personalizadas, índices avanzados (GIN, GiST, BRIN).

PROS	CONTRAS
Muy robusto y escalable.	Administración más compleja.
Excelente para consultas complejas.	Curva de aprendizaje más alta.
Gran control de integridad y concurrencia.	Requiere más recursos que MySQL/MariaDB en entornos pequeños.

Nuestra elección: PostgreSQL

Motivo: Hemos decidido utilizar PostgreSQL porque ofrece mayor robustez, escalabilidad y soporte para consultas complejas que MySQL o MariaDB. Además, permite trabajar con datos estructurados y semiestructurados (JSONB, arrays, funciones personalizadas), lo cual es ideal para este proyecto que debe gestionar usuarios, monitorización y logs de incidencias. Aunque su administración es más compleja, garantiza integridad de datos y mejor rendimiento en entornos con alta concurrencia.