| **Tecnología** | **Tipo** | **Ventajas** | **Desventajas** | **Casos de uso** |
| --- | --- | --- | --- | --- |
| **HTML** | Lenguaje de marcado | - Es el estándar para la web.  - Fácil de aprender y usar.  - Compatible con todos los navegadores. | - No es un lenguaje de programación.  - Necesita CSS y JavaScript para funcionalidades avanzadas. | Creación de páginas web estáticas, estructura de sitios web y aplicaciones web. |
| **CSS** | Lenguaje de estilos | - Separa contenido y diseño.  -Permite personalizar el aspecto de las páginas.  -Compatible con animaciones y transiciones.  -CSS frameworks como Bootstrap facilitan el desarrollo. | -Puede volverse complejo en proyectos grandes sin preprocesadores como SASS o LESS.  - Inconsistencias en la renderización entre navegadores. | Diseño y estilización de sitios web, desarrollo de interfaces responsivas. |
| **JavaScript** | Lenguaje de programación | - Lenguaje universal en la web (frontend y backend con Node.js).  - Gran comunidad y ecosistema de librerías (React, Angular, Vue).  -Soporta programación asíncrona con async/await. | - Código ejecutado en el navegador es vulnerable a ataques si no se maneja correctamente.  - Puede ser difícil de depurar en proyectos grandes.  - Manejo inconsistente de tipos de datos. | Desarrollo frontend interactivo, aplicaciones web dinámicas, backend con Node.js. |
| **React** | Biblioteca frontend | - Reutilización de componentes.  - Virtual DOM mejora el rendimiento.  - Gran comunidad y soporte de Facebook.  - Compatible con React Native para apps móviles. | - Curva de aprendizaje moderada.  - Necesita herramientas adicionales como Webpack y Babel.  - Puede ser excesivo para proyectos pequeños. | Aplicaciones de una sola página (SPA), interfaces dinámicas, dashboards. |
| **Angular** | Framework frontend | - Arquitectura escalable con TypeScript.  - Soporte a largo plazo de Google.  - Seguridad integrada y herramientas avanzadas.  - Compatible con PWA y SSR. | - Curva de aprendizaje alta.  - Mayor peso en comparación con otras tecnologías frontend.  - Menos flexible para proyectos pequeños. | Aplicaciones empresariales, aplicaciones de gran escala con alta seguridad. |
| **Vue.js** | Framework frontend | - Sintaxis sencilla y fácil de aprender.  - Ligero y rápido en comparación con React y Angular.  - Buena documentación.  - Se puede integrar fácilmente en proyectos existentes. | - Menos soporte empresarial en comparación con React y Angular.  - Comunidad más pequeña.  - No es tan eficiente en proyectos muy grandes como Angular. | Aplicaciones interactivas, dashboards, proyectos de tamaño medio. |
| **Node.js** | Entorno backend | - Basado en JavaScript (mismo lenguaje frontend/backend).  - No bloqueante, permite alta concurrencia.  - Compatible con Express.js para construir APIs rápidamente.  - Gran ecosistema de paquetes con NPM. | - No es la mejor opción para tareas que requieren alto uso de CPU.  - Callbacks anidados pueden dificultar la lectura del código.  - La falta de un estándar definido puede hacer que los proyectos sean inconsistentes. | APIs REST, aplicaciones en tiempo real (chats, websockets). |
| **Django** | Framework backend | - Basado en Python, fácil de aprender y escalable.  - Seguridad integrada (protección contra XSS, CSRF).  - Modelo MVC bien estructurado.  - Admin panel automático para gestionar datos. | - Puede ser demasiado pesado para proyectos pequeños.  - Menos flexible para desarrollos microservicio comparado con Flask o FastAPI.  - Curva de aprendizaje para principiantes en frameworks. | Aplicaciones seguras y escalables, comercio electrónico, paneles administrativos. |
| **Flask** | Microframework backend | - Ligero y flexible, permite personalización total.  - Fácil de aprender y usar.  - Requiere menos configuración que Django.  - Ideal para APIs RESTful. | - No tiene tantas herramientas preintegradas como Django.  - Necesita más configuraciones manuales para seguridad y autenticación. | Aplicaciones pequeñas y medianas, APIs y microservicios. |
| **Laravel** | Framework backend | - Basado en PHP, sintaxis clara y organizada.  - Incluye autenticación y seguridad por defecto.  - ORM (Eloquent) facilita la interacción con bases de datos.  - Permite desarrollo rápido con Artisan CLI. | - Puede ser más pesado que otros frameworks.  - Menos eficiente para microservicios en comparación con Node.js o Flask. | Aplicaciones web con PHP, comercio electrónico, sistemas de gestión de contenido (CMS). |
| **MySQL** | Base de datos relacional | - Muy usado en la industria.  - Seguridad y soporte transaccional ACID.  - Compatible con múltiples lenguajes de programación.  - Soporta grandes volúmenes de datos. | - Menos flexible con datos no estructurados.  - Puede requerir optimización para consultas complejas. | Aplicaciones con estructuras de datos definidas, sistemas bancarios, ecommerce. |
| **MongoDB** | Base de datos NoSQL | - Escalable y flexible con JSON/BSON.  - No requiere una estructura de datos fija.  - Excelente para aplicaciones en tiempo real y big data.  - Fácil integración con Node.js y Express. | - No es ideal para aplicaciones que requieren alta consistencia de datos.  - Menos eficiente en operaciones que requieren transacciones complejas. | Aplicaciones con grandes volúmenes de datos, sistemas de recomendación, aplicaciones IoT. |