BASE DE DATOS					
Tecnología	**Para qué sirve**	**Ventajas**	**Desventajas**	**Cuándo usar / Evitar**	**Impacto en atributos de calidad**
PostgreSQL	Relacional avanzada	Muy confiable, soporta JSON, escalable	Configuración más compleja	Usar en sistemas críticos, evitar si se busca algo muy simple	Confiabilidad , Escalabilidad , Complejidad
MySQL/MariaDB	Relacional estándar	Fácil de usar, rápido, popular	Menos avanzado que PostgreSQL	Usar en apps web comunes, evitar en sistemas con consultas complejas	Rapidez , Simplicidad , Robustez
SQLite	Relacional ligera	Muy simple, portable	No escala para grandes volúmenes	Usar en apps móviles/prototipos, evitar en producción con muchos usuarios	Simplicidad , Escalabilidad , Portabilidad
MongoDB	NoSQL orientada a documentos	Flexible, JSON nativo, fácil de escalar	Integridad débil, requiere buen modelado	Usar en datos no estructurados, evitar en sistemas financieros	Flexibilidad , Escalabilidad , Integridad
Redis	Clave-valor en memoria	Ultra rápido, ideal para cache	Persistencia limitada	Usar como cache/sesiones, evitar como DB principal	Rendimiento , Persistencia , Escalabilidad
Cassandra	NoSQL distribuida	Alta disponibilidad, escalabilidad horizontal	Curva de aprendizaje alta	Usar en sistemas globales, evitar si no se necesita big data	Disponibilidad , Escalabilidad , Simplicidad
Elasticsearch	Búsqueda y análisis	Potente para búsquedas, escalable	No es DB generalista	Usar para logs/búsquedas, evitar como DB principal	Búsqueda , Rendimiento , Persistencia
Oracle DB	Relacional empresarial	Confiable, soporte robusto	Costosa, compleja	Usar en grandes corporativos, evitar en startups	Seguridad , Escalabilidad , Costo
SQL Server	Relacional de Microsoft	Integración MS, buen soporte BI	Costosa, menos portable	Usar en entornos Microsoft, evitar fuera de ese stack	Seguridad , Integración , Portabilidad
CouchDB	NoSQL orientada a documentos	Fácil replicación, tolerancia a fallos	Comunidad menor, menor rendimiento que Mongo	Usar en apps distribuidas, evitar si se busca alto rendimiento	Disponibilidad , Flexibilidad , Rendimiento
BACKEND					
Tecnología	**Para qué sirve**	**Ventajas**	**Desventajas**	**Cuándo usar / Evitar**	**Impacto en atributos de calidad**
Spring Boot (Java)	Framework para microservicios y APIs robustas	Madurez, seguridad, comunidad grande, integración con bases de datos	Curva de aprendizaje alta, configuración inicial más compleja	Usar en sistemas críticos, evitar si se requiere desarrollo muy rápido con poco equipo	Escalabilidad , Seguridad , Productividad
Node.js + Express	Servidor ligero con JS, APIs REST	Alto rendimiento en I/O, gran ecosistema npm	Seguridad requiere atención, rendimiento CPU-bound limitado	Usar en apps en tiempo real, evitar en cálculos pesados	Concurrencia , Escalabilidad , Seguridad
Django (Python)	Framework fullstack orientado a rapidez y seguridad	Productivo, incluye ORM, seguridad integrada	Más pesado, menos flexible en microservicios	Usar para MVPs seguros, evitar en servicios muy especializados	Productividad , Seguridad , Rendimiento
Flask (Python)	Microframework minimalista	Ligero, flexible, rápido de prototipar	Menos estructura, seguridad depende del dev	Usar para microservicios pequeños, evitar en proyectos grandes	Agilidad , Escalabilidad , Mantenibilidad
Go (Golang)	Lenguaje para servicios de alto rendimiento	Concurrencia nativa, binarios rápidos, eficiente	Ecosistema menor que Java/Node	Usar en sistemas críticos de alto rendimiento, evitar si se necesita rapidez de prototipado	Rendimiento , Escalabilidad , Productividad
.NET Core (C#)	Framework cross-platform para APIs y servicios	Rendimiento alto, integración empresarial, soporte Microsoft	Menos popular fuera de entornos MS, curva para devs no .NET	Usar en entornos corporativos, evitar si no hay experiencia en .NET	Seguridad , Rendimiento , Portabilidad
Ruby on Rails	Framework web fullstack	Extremadamente productivo, comunidad madura	Menor rendimiento, menos popular que antes	Usar en startups/MVPs, evitar en sistemas de alto tráfico	Productividad , Rendimiento , Escalabilidad
Quarkus (Java)	Framework Java optimizado para microservicios cloud	Arranque rápido, integración con Kubernetes	Menos maduro que Spring Boot	Usar en microservicios cloud-native, evitar si se busca comunidad amplia	Escalabilidad , Rendimiento , Comunidad
Micronaut (Java/Groovy)	Framework ligero para microservicios	Bajo consumo, nativo para serverless	Ecosistema más pequeño	Usar en arquitecturas serverless, evitar si se requiere ecosistema extenso	Rendimiento , Escalabilidad , Soporte
FastAPI (Python)	Framework moderno para APIs rápidas	Alto rendimiento con Python, documentación automática	Comunidad aún pequeña comparado con Django/Flask	Usar en microservicios rápidos y APIs modernas, evitar en fullstack	Productividad , Rendimiento , Mantenibilidad
FRONTEND					
Tecnología	**Para qué sirve**	**Ventajas**	**Desventajas**	**Cuándo usar / Evitar**	**Impacto en atributos de calidad**
React	Librería JS para interfaces	Popularidad, ecosistema enorme, flexible	Curva de aprendizaje, no tiene todo incluido	Usar en apps web grandes, evitar si se busca simplicidad	Escalabilidad , Productividad , Complejidad
Angular	Framework completo de Google	Robusto, modular, gran soporte empresarial	Verboso, curva de aprendizaje alta	Usar en proyectos corporativos, evitar en MVPs rápidos	Mantenibilidad , Robustez , Productividad
Vue.js	Framework progresivo JS	Simple, productivo, curva baja	Ecosistema menor que React/Angular	Usar en proyectos medianos, evitar en entornos muy corporativos	Simplicidad , Productividad , Escalabilidad
Svelte	Framework basado en compilación	Rápido, sintaxis clara	Comunidad más pequeña	Usar en apps ligeras, evitar si se requiere soporte enterprise	Rendimiento , Simplicidad , Ecosistema
Next.js (sobre React)	Framework SSR/SSG para React	SEO fuerte, optimización de rendimiento	Más complejo que React puro	Usar en sitios con SEO crítico, evitar si solo es app interna	SEO , Rendimiento , Complejidad
Nuxt.js (sobre Vue)	SSR/SSG sobre Vue	SEO, estructura organizada	Menos comunidad que Next.js	Usar en proyectos Vue con SEO, evitar en grandes corporativos	SEO , Productividad , Comunidad
Flutter Web	Framework de Google con Dart	UI consistente en web y móvil	Ecosistema web menos maduro	Usar si se busca compartir código web/móvil, evitar solo para web	Portabilidad , Rendimiento , Comunidad
Blazor (Microsoft)	Frontend con C# en navegador	Reutilización de código .NET	Poco usado fuera del mundo MS	Usar en entornos Microsoft, evitar si no hay .NET	Reutilización , Productividad , Popularidad
Bootstrap	Framework CSS/JS	Rápido de implementar, responsive fácil	Poco flexible visualmente	Usar en prototipos rápidos, evitar en proyectos personalizados	Rapidez , Flexibilidad , Productividad
Tailwind CSS	Framework utilitario CSS	Muy flexible, moderno	Curva inicial para clases utilitarias	Usar en proyectos modernos, evitar si el equipo no conoce utilitarios	Flexibilidad , Escalabilidad , Curva