**Analisis de recursos (Solo nuestro modulo del proyecto).**

| **Aspecto** | **Laravel** | **FlutterFlow & supabase** |
| --- | --- | --- |
| **Tiempo desarrollo formulario** | 8 hrs  (sin contemplar tiempo de aprendizaje). | 4 hrs flutterflow  1 hr supabase  (Contemplando tiempo de aprendizaje) |
| **Tiempo de elaboración de los artefactos necesarios** | 3 hrs | 3 hrs |
| **Curva de aprendizaje** | **Moderada**  (Conocimientos de PHP, MVC, base de datos) | **Baja**  (plataforma interactiva no-code/low-code) |
| **Recurso humano** | 1 desarrollador | 1 desarrollador |
| **Infraestructura** | Hosting/vps. | Hosting/despliegue de supabase. |
| **Artefactos de diseño necesarios** | Diagrama de caso de uso  Diagrama de flujo  Diagrama entidad-relación  (Si fuera la arquitectura completa necesitamos también diagrama de clases y diagrama de arquitectura). | Diagrama de caso de uso  Diagrama entidad-relación  Mockups de UI (bocetos a mano).  (Para la arquitectura completa probablemente se necesiten los mismos artefactos). |
| **Control de versiones** | Permite integración con herramientas de control de versiones. | Permite integración con repositorios de github y otras herramientas de control de versiones (incluyendo la propia de flutter flow) |
| **Documentación y pruebas** | Documentación manual o con herramientas de terceros, con posibilidad de tests unitarios e integración continua. | Documentación basada en prototipos, con menos opciones para testing automatizado. |

**Atributos de calidad (Marco de referencia SQuaRE ISO/IEC 25010:2023)**

**International Organization for Standardization/International Electrotechnical Commission**

| **Atributo** | **Flutterflow & Supabase** | **Laravel** | **Notas** |
| --- | --- | --- | --- |
| **Rendimiento**  (Tiempos de respuesta, Uso de CPU/Memoria) | Ambas herramientas están optimizadas para tener tiempos de respuesta cortos y optimización de recursos que ya entregan por defecto al desarrollador. | Depende totalmente del manejo de recursos que haga el desarrollador en la parte del manejo del código pero es altamente optimizable. | En este apartado se está tomando lo que teóricamente sucedería en la arquitectura completa, en este módulo es casi imperceptible la diferencia entre ambas. |
| **Compatibilidad**  (Coexistencia) | Provee herramientas para exportar la aplicación a diferentes ambientes como el móvil, aplicación web y aplicaciones de escritorio. | Despliegue como aplicación web.  (También podría ser un API REST) | En este apartado se toma en cuenta sólo nuestro módulo del sistema. |
| **Usabilidad** | **Protección de errores del usuario:**  Permite validaciones estándar y personalizadas con expresiones regulares.  **Accesibilidad:**  Tiene integradas funciones como lector de pantalla para mejorar la inclusión y la experiencia del usuario. | **Protección de errores del usuario:**  Contiene validaciones estándar y con expresiones regulares.  **Accesibilidad:**  Con las funciones para mejorar la accesibilidad hay que profundizar | En este apartado solo se toma en cuenta la versión que se tiene hasta ahora, es decir, la aplicación web. |
| **Seguridad** | **Resistencia a ataques:** Para el envío del formulario se hace uso del token **CSRF** para mantener seguro el sistema de ataques.  **Integridad:**  También se aplican validaciones estrictas y reglas de negocio para asegurar la integridad de los datos. | La parte de la seguridad es gestionada por supabase y confiando en que flutter flow genera código seguro. |  |
| **Mantenibilidad** | **Modularidad:** Esta versión de la aplicación está hecha en una arquitectura de MVC por lo tanto mantiene un diseño modular. **Reusabilidad, modificabilidad:**  Se generan componentes que pueden ser reutilizados en distintas partes del sistema.  Es muy sencillo implementar cambios teniendo buen conocimiento del funcionamiento del sistema. | **Modularidad:** Genera un proyecto con código modular al utilizar los widgets.  **Reusabilidad, modificabilidad:** Es modificable hablando desde la parte del diseño, para cambios muy complejos o estructurales es mejor tratar el código por fuera de flutter flow. |  |
| **Flexibilidad** | **Adaptabilidad:**  Laravel es un framework muy flexible que para construir este módulo del sistema fue más que necesario.  **Capacidad de reemplazar:**  Migrar un proyecto Laravel a otro framework es complicado por el acoplamiento de la lógica de negocio al mismo framework. Específicamente en este módulo del sistema será sencillo probablemente pero por el tamaño del proyecto.  **Escalabilidad:** Sin duda si quisiéramos integrar la arquitectura completa llevaría más tiempo pero sería relativamente sencillo de implementar con la arquitectura correcta. | **Adaptabilidad:**  Flutter Flow por sí mismo es óptimo para el diseño de las UI y lógica sencilla pero los cambios de arquitectura se delegan a supabase.  **Capacidad de reemplazar:**  Es bastante sencillo de reemplazar exportando el código y supabase permite migrar datos a otros postgreSQL  **Escalabilidad:**  Sencillo de escalar al implementar el sistema completo o integrar todos los módulos ya hechos. |  |

**Idoneidad funcional:  
¿Hace todo lo que debería hacer?**

**¿Lo hace de manera precisa y correcta?**

**¿Lo que hace es realmente útil y relevante para los usuarios y el propósito del sistema?**

**Conclusiones:**

Laravel ofrece un control total y flexibilidad ilimitada en el desarrollo, lo que puede resultar en una mayor mantenibilidad y adecuación funcional para proyectos complejos, pero a costa de un mayor tiempo de desarrollo y una mayor curva de aprendizaje. Es ideal cuando necesitas un backend web muy robusto y personalizado.

Flutter Flow & Supabase se enfoca en la velocidad de desarrollo y la facilidad de uso, especialmente para aplicaciones móviles y web con interfaces probadas. Destaca en usabilidad y escalabilidad del backend gestionado, sacrificando algo de flexibilidad en la lógica de negocio muy personalizada. Es ideal para MVPs, startups, y proyectos donde la rapidez de despliegue y una buena experiencia de usuario móvil son clave.