# **Universidad Libre**

## **Desparches**

### **Documento de Arquitectura del Sistema (SAD)**

**Nombre del Equipo de Trabajo:** UpStream

**Nombre de los Integrantes:**

* **Laura Montaña**  
   **Rol:** Analista de Requisitos  
   **e-mail:** [laurag-montanag@unilibre.edu.co](mailto:laurag-montanag@unilibre.edu.co)
* **Alison Tinoco**  
   **Rol:** Diseñadora UX/UI  
   **e-mail:** [alisond-tinocom@unilibre.edu.co](mailto:alisond-tinocom@unilibre.edu.co)
* **Daniela Perez**  
   **Rol:** Desarrolladora Frontend  
   **e-mail:** [laurad-perezj@unilibre.edu.co](mailto:laurad-perezj@unilibre.edu.co)
* **Carlos Daza**  
   **Rol:** Desarrollador Backend / Arquitecto  
   **e-mail:** [carlose-dazac@unilibre.edu.co](mailto:carlose-dazac@unilibre.edu.co)
* **Sara Castelblanco**  
   **Rol:** Líder de Calidad / Tester  
   **e-mail:** [sarav-castelblancog@unilibre.edu.co](mailto:sarav-castelblancog@unilibre.edu.co)

**Tabla de Contenido**

[Universidad Libre 1](#_Toc368127452)

[Desparches 1](#_Toc747443712)

[Documento de Arquitectura del Sistema (SAD) 1](#_Toc2038183443)

[1. Propósito y Audiencia 2](#_Toc1431483380)

[2. Organización del Documento 2](#_Toc244502055)

[3. Propósito y Audiencia 5](#_Toc1574038904)

[4. Organización del Documento 5](#_Toc301825804)

[5. Convenciones 5](#_Toc1948773104)

[6. Terminología y Definiciones 6](#_Toc848794762)

[7. Documentos Relevantes 6](#_Toc1636194226)

[1. Problema a Resolver 7](#_Toc1966428383)

[2.2 Objetivos 7](#_Toc734758331)

[3. Stakeholders 8](#_Toc1023996613)

[1. Motivadores de Negocio 11](#_Toc1410511231)

[2. Restricciones de Tecnología 12](#_Toc1795880026)

[3. Atributos de Calidad 12](#_Toc1719314511)

[1. Escenarios Operacionales 17](#_Toc1172055285)

[1. Punto de Vista Funcional 21](#_Toc1121042057)

[2. Punto de Vista de Despliegue 23](#_Toc1557986032)

[3. Punto de Vista de Información 27](#_Toc1895221879)

[4. Punto de Vista de Concurrencia 30](#_Toc678453906)

[5. Punto de Vista de Desarrollo 31](#_Toc2093117551)

[6.1 Punto de Vista Funcional 33](#_Toc1156327365)

[6.2 Punto de Vista de Despliegue 33](#_Toc1434971444)

[6.3 Punto de Vista de Red 33](#_Toc1075087775)

[6.4 Punto de Vista de Dependencia Tecnológica 33](#_Toc1376648594)

[6.5 Punto de Vista de Información 33](#_Toc354839348)

[6.6 Punto de Vista de Concurrencia 33](#_Toc1031444634)

[6.7 Punto de Vista de Desarrollo 34](#_Toc364200619)

[6.8 Conclusión 34](#_Toc1878935563)

[7.1 Criterios de Evaluación Utilizados 35](#_Toc788074358)

[7.2 Métodos de Evaluación Aplicados 35](#_Toc2136527717)

[7.3 Resultados de la Evaluación 35](#_Toc1594782560)

[Fortalezas detectadas: 35](#_Toc379846026)

[Aspectos por mejorar: 36](#_Toc1426840618)

[7.4 Conclusión 36](#_Toc695264391)

[Responsabilidades Generales del Equipo 37](#_Toc151712523)

Sección 1. Descripción del Documento

## Propósito y Audiencia

Desparches es una plataforma digital innovadora que conecta a las personas con eventos y promociones exclusivas en su área cercana, integrando funciones como búsquedas geolocalizadas, recomendaciones personalizadas, reservas y compras en línea, así como interacción social. Está dirigida tanto a usuarios que buscan actividades en su ciudad como a organizadores y comercios que desean promocionar sus eventos u ofertas.

## Organización del Documento

1. Descripción General

* Descripción del Documento
* Propósito y Audiencia
* Organización del Documento
* Convenciones y Terminología

2. Generalidades del Proyecto

* Problema a Resolver
* Objetivos del Sistema
* Stakeholders Clave

3. Restricciones Arquitecturales

* Motivadores de Negocio
* Limitaciones Tecnológicas

4. Atributos de Calidad

5. Contexto del Sistema

* Escenarios Operacionales
* Casos de Uso Principales

6. Puntos de Vista Arquitectónicos

* Vista Funcional
* Vista de Despliegue
* Vista de Información
* Vista de Concurrencia

7. Evaluación de la Arquitectura

* Criterios de Evaluación
* Resultados y Conclusiones

8. Directorio

* Equipo de Trabajo y Responsabilidades

## Convenciones

Títulos principales:

* Negrita
* Tamaño 16
* Alineación a la izquierda
* Ejemplo: 1. Propósito y Audiencia

Subtítulos:

* Negrita
* Tamaño 14
* Ejemplo: 1.1 Propósito del Documento

Términos técnicos:

* Cursiva
* Ejemplo: Microservicios

Código o tecnologías:

* Código en línea (monoespaciado)
* Ejemplo: Node.js

Notas importantes:

* Bloque de cita con “>”
* Ejemplo:
* Nota: Esto es importante.

## Terminología y Definiciones

* **Stakeholder**: Persona o entidad con interés en el sistema (ejemplo: usuarios, desarrolladores).
* **Microservicios**: Arquitectura donde el sistema se divide en servicios independientes.
* **RGPD**: Reglamento General de Protección de Datos (normativa de privacidad).
* **API REST**: Interfaz de programación basada en HTTP para comunicación entre sistemas.
* **Escalabilidad**: Capacidad del sistema para manejar crecimiento en usuarios o carga.

## Documentos Relevantes

* **Requisitos del Sistema (SRS)**: Describe funcionalidades y requisitos no funcionales.
* **Plan de Proyecto**: Cronograma, hitos y asignación de recursos.
* **Manual de Usuario**: Guía para usuarios finales.
* **Políticas de Seguridad**: Protocolos para protección de datos y cumplimiento RGPD.
* **Estándar IEEE 1471-2000**: Marco de referencia para documentación arquitectónica.

Sección 2. Generalidades del Proyecto

## Problema a Resolver

Los principales problemas del proyecto Desparches se centran en la integración compleja de APIs externas (como Google Maps y pse) con datos precisos de geolocalización, especialmente en ciudades lejanas, lo que puede generar retraso y inconsistencias. También enfrenta retos de escalabilidad y rendimiento ante picos de uso, donde se requiere mantener búsquedas rápidas (<2 s) y soporte para 10,000 usuarios simultáneos. Además, compite con plataformas como Eventbrite y enfrenta una posible baja aceptación por parte de comercios locales. Otros desafíos clave incluyen garantizar seguridad (2FA, cifrado, RGPD), mantener una experiencia móvil fluida con React Native, y manejar la dependencia de recursos externos como AWS y Firebase.

* 1. **Descripción General del Sistema a Desarrollar**

Desparches es una plataforma digital que conecta a personas con eventos y promociones locales en tiempo real. Su arquitectura se organiza en subsistemas ofreciendo una experiencia personalizada, segura y eficiente.

* **Subsistema de Gestión de Usuarios:** Maneja el registro, autenticación y perfiles personalizados define roles.
* **Subsistema de Búsqueda y Geolocalización:** Permite buscar eventos cercanos con filtros y muestra resultados en un mapa.
* **Subsistema de Reservas y Pagos:** Facilita la compra directa de entradas o promociones. Incluye pagos seguros, cupones y confirmaciones.
* **Subsistema de Recomendaciones:** Sugiere eventos según gustos del usuario, comportamientos similares y tendencias locales.
* **Subsistema de Contenido:** Organiza la publicación de eventos con descripciones, fotos y horarios. Hay control de calidad.
* **Subsistema de Notificaciones:** Informa al usuario sobre eventos próximos, ofertas cercanas y cambios de último minuto.
* **Subsistema de Valoraciones y Comunidad:** Permite calificar eventos, dejar reseñas y ver amigos asistentes. Fomenta la confianza y la interacción.
* **Subsistema de Administración y Soporte:** Herramientas para moderadores y atención al usuario para mantener la calidad y resolver problemas.

**Propósito General:**  
 Desparches busca resolver la dispersión de información sobre eventos y promociones, mejorando la visibilidad de los comercios y facilitando a los usuarios el acceso a experiencias locales relevantes, todo en una sola app.

## Objetivos

La arquitectura del sistema Desparches orienta el desarrollo del proyecto a lo largo de su ciclo de vida, asegurando calidad, escalabilidad y funcionalidad.

**1. Fase de Análisis**  
 Entender y validar los requisitos clave, identificar componentes esenciales como geolocalización e integración con APIs de pago, priorizar funcionalidades críticas como la búsqueda en tiempo real y la seguridad en reservas, y detectar riesgos técnicos tempranos.

**2. Fase de Diseño**  
 Definir una estructura escalable y segura, mirando las capas del sistema (frontend, backend, base de datos), seleccionando tecnologías y garantizando atributos de calidad como rendimiento, escalabilidad y seguridad.

**3. Fase de Desarrollo**  
 Implementar funcionalidades asegurando una interfaz intuitiva con el diseño arquitectónico, facilitando la conexión entre componentes y optimizando recursos como el uso de caché. Se busca obtener un producto estable con versiones incrementales probadas y documentación actualizada.

**4. Fase de Implementación**  
 Garantizar un despliegue fluido con mínima interrupción, preparando la infraestructura para producción y capacitando a usuarios y administradores.

**5. Fase de Mantenimiento**  
 Evolucionar el sistema sin afectar su estabilidad, corrigiendo errores rápidamente, agregando nuevas funcionalidades y ajustando la escalabilidad según la demanda real.

## 3. Stakeholders

Matriz Stakeholder-Concerns (IEEE 1471-2000) para Desparches

(Grado de relevancia: Alto [A], Medio [M], Bajo [B])

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Stakeholder (IEEE 1471)** | **Funcionalidad** | **Usabilidad** | **Rendimiento** | **Seguridad** | **Integración** | **Mantenibilidad** | **Cumplimiento** |
| **Customer** (Cliente/Patrocinador) | A (ROI) | M | M | M | B | B | A (Negocio) |
| **End Users** (Usuarios finales) | A (Búsqueda/reservas) | A | A | M | B | B | B |
| **Application Developers** (Equipo de desarrollo) | A | M | A | A | A | A | M |
| **System Engineers** (Arquitectos) | A | B | A | A | A | A | A |
| **Project Manager** (Gerente) | M | B | M | M | M | M | A (Plazos) |
| **Maintainers** (Mantenimiento) | B | B | B | M | M | A | B |
| **Security/Certifiers** (Auditores) | B | B | B | A | B | B | A (RGPD) |

*Tabla 1: Listado de los Stakeholders*

|  |  |
| --- | --- |
| **Stakeholder** | **Descripción** |
| **Usuario Final** | Personas que usan la app para descubrir eventos/promociones y hacer reservas. |
| **Organizador/Comercio** | Empresas o individuos que publican eventos/promociones en la plataforma. |
| **Equipo de Desarrollo** | Desarrolladores frontend/backend, DevOps, y QA que implementan el sistema. |
| **Gerente de Proyecto** | Responsable de planificación, presupuesto y coordinación del equipo. |
| **Administrador de BD** | Gestiona la base de datos (PostgreSQL geoespacial, MongoDB). |
| **Soporte Técnico** | Atiende consultas de usuarios y organizadores (FAQ, chat). |
| **Auditor de Seguridad** | Verifica cumplimiento de normativas (ej: RGPD). |

*Tabla 2: Stakeholders y Espectativas*

|  |  |
| --- | --- |
| **Stakeholder** | **Expectativas** |
| **Usuario Final** | - Búsqueda rápida de eventos por ubicación/filtros. - Reservas/pagos sin errores. - Interfaz intuitiva. |
| **Organizador** | - Panel fácil para publicar eventos. - Estadísticas de participación. - Integración con APIs de pago. |
| **Equipo de Desarrollo** | - Arquitectura escalable (microservicios). - APIs estables para geolocalización. - Pruebas automatizadas. |
| **Gerente de Proyecto** | - Cumplir plazos y presupuesto (150K USD). - Mitigar riesgos (ej: competencia con Eventbrite). |
| **Administrador de BD** | - Alto rendimiento en consultas geoespaciales. - Backup de datos sensibles. |
| **Soporte Técnico** | - FAQ actualizada. - Herramientas para resolver incidencias rápido. |
| **Auditor de Seguridad** | - Cumplimiento de RGPD. - Cifrado de datos y autenticación segura. |

Sección 3. Restricciones Arquitecturales

## Motivadores de Negocio

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre del Motivador de Negocio** | **Descripción del Motivador de Negocio** | | |
| *Aumentar la base de usuarios activos en zonas urbanas mediante estrategias de engagement* | "Incrementar un 40% los usuarios activos mensuales (de 10,000 a 14,000) mediante notificaciones push personalizadas y programas de fidelización (ej: recompensas por check-in). | | |
| **Medida del Impacto** | | | |
| **Rangos** | **Cota Mínima** | | **Cota Máxima** |
| Ninguno | 0 | | 9,999 |
| Bajo | 10,000, | | 11,999, |
| Moderado | 12,000 | | 13,999 |
| Fuerte | 14,000 | | 19,999 |
| Muy Fuerte | 20,000 | | — |
| Asociación del Motivador con el Negocio | Definido Por: | **CEO** | |
| Ejecutador Por: | **Equipo de Marketing** | |
| Ubicación en el Portafolio del negocio | **Servicios digitales B2C** | |

|  |  |
| --- | --- |
| **Nombre del Motivador de Negocio** | **Descripción del Motivador de Negocio** |
| Mejorar la tasa de conversión de búsquedas a reservas mediante UX simplificado | Aumentar la tasa de conversión del 15% al 25% mediante rediseño del flujo de reservas (1-click) e integración de pagos sin fricciones. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Medida del Impacto** | | | |
| **Rangos** | **Cota Mínima** | | **Cota Máxima** |
| Ninguno | 0%, | | 14%, |
| Bajo | 15%, | | 19% |
| Moderado | 20% | | 24% |
| Fuerte | 25% | | 30% |
| Muy Fuerte | 31%+ | | — |
| Asociación del Motivador con el Negocio | Definido Por: | **Gerente de Producto** | |
| Ejecutador Por: | **Equipo de UX/Development** | |
| Ubicación en el Portafolio del negocio | **Plataforma de eventos** | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre del Motivador**  **de Negocio** | **Descripción del Motivador de Negocio** | | |
| Captar un 30% más de organizadores de eventos locales mediante partnerships | Aumentar la red de organizadores activos (de 500 a 650) mediante acuerdos con cámaras de comercio y beneficios por volumen de ventas. | | |
| **Medida del Impacto** | | | |
| **Rangos** | **Cota Mínima** | | **Cota Máxima** |
| Ninguno | 0 | | 499 |
| Bajo | 500 | | 549 |
| Moderado | 550 | | 649 |
| Fuerte | 650 | | 800 |
| Muy Fuerte | 801+ | | ----- |
| Asociación del Motivador con el Negocio | Definido Por: | Director Comercial | |
| Ejecutador Por: | Equipo de Sales | |
| Ubicación en el Portafolio del negocio | Alianzas estratégicas | |

## Restricciones de Tecnología

El proyecto **Desparches** enfrenta una serie de restricciones tecnológicas que afectan tanto el desarrollo como la operación del sistema. Uno de los principales desafíos es la integración con APIs externas, como Google Maps para geolocalización y pasarelas de pago (por ejemplo, PSE). Estas dependencias pueden generar inconsistencias en zonas con cobertura limitada, afectando la precisión de los datos y provocando retrasos en la carga de información.

A nivel de rendimiento, el sistema debe mantener tiempos de respuesta inferiores a 2 segundos en las búsquedas y escalar adecuadamente para soportar hasta 10,000 usuarios simultáneos. Esto implica diseñar una arquitectura robusta, eficiente y con un uso optimizado de los recursos.

Por otro lado, el sistema se enfrenta a factores externos como la competencia y la poca flexibilida de pequeños comercios a adoptar nuevas tecnologías, lo cual podría limitar la oferta inicial de contenido en la aplicación.

Desde el punto de vista técnico, se destacan tres áreas críticas:

* **Seguridad:** Es indispensable implementar mecanismos como la autenticación de dos factores (2FA), cifrado de datos sensibles y el cumplimiento de normativas de protección de datos, como el Reglamento General de Protección de Datos (RGPD).
* **Experiencia móvil:** La aplicación, desarrollada en React Native, debe estar optimizada para ofrecer un desempeño fluido incluso en dispositivos de gama media, que representan una porción significativa del mercado objetivo.
* **Dependencia de servicios externos:** La plataforma debe gestionar eficientemente recursos en la nube (AWS, Firebase) para garantizar alta disponibilidad, evitar caídas del sistema y controlar los costos asociados.

Para mitigar estas restricciones, se propone la implementación de soluciones técnicas como el uso de caché para acelerar el acceso a datos geoespaciales, balanceadores de carga para distribuir el tráfico en momentos de alta demanda, y pruebas exhaustivas de integración con servicios externos para garantizar la estabilidad operativa. La correcta priorización y ejecución de estas medidas será determinante para el éxito del proyecto.

## Atributos de Calidad

* + 1. **Árbol de Utilidad**

|  |  |  |  |
| --- | --- | --- | --- |
| Atributo de Calidad: | | Eficiencia | |
| Tiempo | ID | Descripción | Prioridad |
| Tiempo | EF-01 | Las búsquedas deben responder en menos de 2 segundos. | Alta |
| Recursos |  |  |  |
| Recursos | EF-02 | Uso de caché para reducir consultas a la base de datos. | Media |
| Recursos | EF-03 | Optimización de código para reducir consumo de CPU. | Media |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Atributo de Calidad: | | Fiabilidad | |
| Tolerancia a Fallas | ID | Descripción | Prioridad |
| Tolerancia a Fallas | FI-01 | El sistema debe seguir funcionando ante la caída de un servicio externo. | Alta |
| Tolerancia a Fallas | FI-02 | Uso de redundancia en servidores y bases de datos. | Media |
| Recuperabilidad |  |  |  |
| Recuperabilidad | FI-03 | Restauración completa del sistema en menos de 1 hora tras un fallo grave. | Alta |
| Recuperabilidad | FI-04 | Backups automáticos diarios de la base de datos y logs críticos. | Alta |
| Recuperabilidad | FI-05 | Sistema de alertas ante fallos para intervención inmediata. | Media |
| Disponibilidad |  |  |  |
| Disponibilidad | FI-06 | Disponibilidad del 99.9% mensual garantizada mediante arquitectura en la nube. | Alta |
| Disponibilidad | FI-07 | Monitoreo 24/7 y balanceo de carga para evitar cuellos de botella. | Alta |

|  |  |  |  |
| --- | --- | --- | --- |
| Atributo de Calidad: | | Mantenimiento | |
| Modificación | ID | Descripción | Prioridad |
| **Modificación** | MA-01 | Cambios en funcionalidades sin afectar el funcionamiento existente. | Alta |
| **Modificación** | MA-02 | Implementación de parches y actualizaciones rápidas. | Alta |
| Facilidad de Pruebas |  |  |  |
| **Facilidad de Pruebas** | MA-03 | Pruebas automatizadas para regresiones en cada despliegue. | Alta |
| **Facilidad de Pruebas** | MA-04 | Entorno de pruebas idéntico al de producción para validar cambios. | Media |
| Estabilidad |  |  |  |
| **Estabilidad** | MA-05 | Minimizar errores durante actualizaciones y evitar caídas. | Alta |

|  |  |  |  |
| --- | --- | --- | --- |
| **Estabilidad** | MA-06 | Control de versiones claro y documentado. | Media |
| Flexibilidad |  |  |  |
| **Flexibilidad** | MA-07 | Código modular para facilitar ampliaciones y modificaciones. | Alta |
| **Flexibilidad** | MA-08 | Uso de patrones de diseño que permiten cambios sin afectar otras partes del sistema. | Media |
| Escalabilidad |  |  |  |
| **Escalabilidad** | MA-09 | Arquitectura que permite agregar módulos o aumentar capacidad sin reescribir el sistema. | Alta |
| **Escalabilidad** | MA-10 | Adaptación rápida a nuevos requerimientos del negocio sin grandes costos. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Atributo de Calidad: | | Seguridad | |
| Integridad | ID | Descripción | Prioridad |
| Integridad | S01 | Validación y control de acceso para evitar modificaciones no autorizadas. | Alta |
| Integridad | S02 | Registro y auditoría de cambios para detectar alteraciones indebidas. | Media |
| Confidencialidad |  |  |  |
| Confidencialidad | S03 | Cifrado de datos sensibles en tránsito y en reposo (por ejemplo, datos de pago y ubicación). | Alta |
| Confidencialidad | S04 | Autenticación de dos factores para acceso de usuarios y comercios. | Alta |
| Confidencialidad | S05 | Políticas estrictas de privacidad conforme al RGPD. | Alta |
| Disponibilidad |  |  |  |
| Disponibilidad | S06 | Sistemas redundantes y respaldo para garantizar el acceso continuo. | Alta |
| Disponibilidad | S07 | Monitorización y alertas para detectar y mitigar ataques o fallos. |  |

* + 1. **Escenarios de Calidad**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Escenario de Calidad 1** | |  | **Stakeholder:** |  |
| **Atributo de Calidad** | Seguridad | | | |
| **Justificación** | Proteger datos sensibles y evitar accesos no autorizados. | | | |
| **Fuente** | Intentos de acceso externo malicioso. | | | |
| **Estímulo** | Intento de ingreso con credenciales inválidas. | | | |
| **Artefacto** | Sistema de autenticación y base de datos de usuarios. | | | |
| **Ambiente** | Sistema en producción, con usuarios activos. | | | |
| **Respuesta** | Bloqueo del acceso y registro del intento fallido. | | | |
| **Medida de la Respuesta** | Tiempo de respuesta < 1 segundo; registro inmediato. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Escenario de Calidad 2** | |  | **Stakeholder:** |  |
| **Atributo de Calidad** | Mantenimiento | | | |
| **Justificación** | Facilitar la implementación de actualizaciones y correcciones. | | | |
| **Fuente** | Requerimiento de actualización de software. | | | |
| **Estímulo** | Disponibilidad de un parche de seguridad. | | | |
| **Artefacto** | Código fuente y módulos del sistema. | | | |
| **Ambiente** | Entorno de desarrollo y pruebas. | | | |
| **Respuesta** | Aplicación del parche sin afectar la estabilidad. | | | |
| **Medida de la Respuesta** | Tiempo de aplicación < 30 minutos; sin errores post-parche. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Escenario de Calidad #** | |  | **Stakeholder:** |  |
| **Atributo de Calidad** | Eficiencia | | | |
| **Justificación** | Mejorar la experiencia del usuario con tiempos de respuesta rápidos. | | | |
| **Fuente** | Peticiones de acceso al sistema. | | | |
| **Estímulo** | Solicitud de carga de página o consulta de datos. | | | |
| **Artefacto** | Servidor y base de datos. | | | |
| **Ambiente** | Sistema en producción con tráfico normal. | | | |
| **Respuesta** | Respuesta rápida y sin bloqueos. | | | |
| **Medida de la Respuesta** | Tiempo de respuesta < 2 segundos. | | | |

Sección 4. Contexto

## Escenarios Operacionales

La plataforma *Desparches* está diseñada para operar en un entorno web y móvil, conectando a usuarios con eventos y promociones disponibles en su entorno geográfico cercano. El sistema debe estar disponible las 24 horas del día y ser capaz de atender múltiples usuarios simultáneamente. Un escenario común es un usuario que, al abrir la aplicación desde su teléfono móvil, recibe recomendaciones personalizadas de eventos y promociones según su ubicación actual. Otro escenario clave involucra a comercios locales que ingresan al sistema mediante un panel web para publicar promociones o eventos. La plataforma también contempla tareas en segundo plano como el envío automático de notificaciones y la actualización periódica del contenido ofrecido.

* + 1. **Escenario Usuario busca eventos cercanos desde su móvil**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Usuario busca eventos cercanos desde su móvil** | | | | |
|  | | | | |
| Stakeholder Asociado | Usuario final (asistente a eventos) | | ID | EO-01 |
| **Consideración Operacional** | | El sistema debe ofrecer una respuesta rápida y relevante basada en la ubicación del usuario. | | |
| Descripción general de la funcionalidad | | El usuario abre la aplicación, autoriza el acceso a su ubicación y recibe una lista de eventos próximos y relevantes en su zona. | | |
| Describa lo que el Stakeholder hace ahora o le gustaría poder hacer | | Actualmente, el usuario depende de redes sociales o páginas dispersas para enterarse de eventos. Le gustaría tener una plataforma unificada con recomendaciones personalizadas. | | |
| Describa cualquier entrada provista o disponible al  momento del inicio | | El usuario proporciona permiso para acceder a su ubicación, y el sistema accede a la base de datos de eventos activos. | | |
| Describa el contexto de la operación | | El usuario está en una zona urbana durante el fin de semana y quiere asistir a un evento cercano en las próximas horas. Usa su móvil y conexión a internet. | | |
| Describa cómo el sistema debe responder | | El sistema debe filtrar y mostrar eventos cercanos y recientes ordenados por relevancia o distancia, permitiendo al usuario ver detalles o marcar favoritos. | | |
| Describa las salidas que el sistema produce como resultado de la acción | | Lista de eventos con nombre, ubicación, imagen, horario, categoría y botón para ver más detalles o guardar. | | |
| Describa quién o qué usa la salida y para que es utilizada | | El usuario final usa la información para decidir a qué evento asistir. El sistema registra el interés del usuario para mejorar futuras recomendaciones.[[1]](#endnote-9245) | | |

**[[2]](#footnote-19788)**

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre de la Entidad | Usuario | ID | ENT-01 |
| **Descripción:** Representa a la persona que usa la aplicación para descubrir eventos o promociones cercanas. | | | |
| **Provee:** Preferencias, ubicación, historial de navegación. | | | |
| **Requiere:** Acceso a eventos, promociones, interfaz personalizada. | | | |
| **Casos de Uso:** Registrarse, iniciar sesión, buscar eventos, ver detalles, guardar favoritos, recibir notificaciones. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre de la Entidad | Comercio | ID | ENT-02 |
| **Descripción:** Representa a los establecimientos o promotores que publican eventos o promociones en la plataforma. | | | |
| **Provee:** Información de eventos o promociones, imágenes, horarios, ubicaciones. | | | |
| **Requiere:** Acceso al panel web para gestionar sus publicaciones y ver métricas. | | | |
| **Casos de Uso:** Publicar evento, editar evento, consultar métricas, gestionar promociones. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Título del Caso de Uso | | Buscar eventos cercanos | ID del Caso de Uso | CU-01 |
| Descripción General del Caso de Uso | | | | |
| El usuario accede a la aplicación, permite el uso de su ubicación, y obtiene una lista de eventos próximos según su zona. | | | | |
| Entidades Involucradas | | | | |
| Usuario, Evento | | | | |
| Precondiciones | | | | |
| El usuario debe haber iniciado sesión y autorizado el uso de su ubicación. | | | | |
| Flujo normal de Eventos | | | | |
| 1. | El usuario accede a la aplicación. | | | |
| 2. | El sistema solicita permiso para usar la ubicación. | | | |
| 3. | El sistema consulta la base de datos de eventos. | | | |
| 4. | Se filtran los eventos por distancia y relevancia. | | | |
| 5. | Se presenta la lista al usuario. | | | |
| Postcondiciones principales del caso de uso | | | | |
| Se muestran eventos relevantes en tiempo real al usuario, quien puede interactuar con ellos. | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Título del Caso de Uso | | Publicar promoción o evento | ID del Caso de Uso | CU-02 |
| Descripción General del Caso de Uso | | | | |
| Un comercio ingresa al sistema para crear o modificar una promoción o evento que será visible para los usuarios. | | | | |
| Entidades Involucradas | | | | |
| Comercio, Evento/Promoción | | | | |
| Precondiciones | | | | |
| El comercio debe estar registrado e iniciar sesión en el panel de gestión. | | | | |
| Flujo normal de Eventos | | | | |
| 1. | El comercio inicia sesión. | | | |
| 2. | Selecciona la opción de crear nuevo evento/promoción. | | | |
| 3. | Ingresa los datos (nombre, fecha, imagen, descripción, ubicación). | | | |
| 4. | El sistema valida y guarda la información. | | | |
| 5. | El evento queda disponible para los usuarios. | | | |
| Postcondiciones principales del caso de uso | | | | |
| El evento o promoción se publica exitosamente y aparece en el feed de usuarios relevantes. | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Título del Caso de Uso | | Guardar evento como favorito | ID del Caso de Uso | CU-03 |
| Descripción General del Caso de Uso | | | | |
| El usuario puede marcar un evento como favorito para acceder a él fácilmente en el futuro. | | | | |
| Entidades Involucradas | | | | |
| Usuario, Evento | | | | |
| Precondiciones | | | | |
| El usuario debe haber iniciado sesión. | | | | |
| Flujo normal de Eventos | | | | |
| 1. | El usuario visualiza un evento desde la lista. | | | |
| 2. | Pulsa el botón de “Guardar” o “Favorito”. | | | |
| 3. | El sistema almacena la preferencia en el perfil del usuario. | | | |

|  |
| --- |
| Postcondiciones principales del caso de uso |
| El evento queda guardado en el perfil del usuario y se puede acceder desde la sección de favoritos. |

Sección 5.Puntos de Vista y Modelos Arquitecturales

## 1. Punto de Vista Funcional

* + 1. **Descripción**

El punto de vista funcional de la arquitectura del sistema **Desparches** describe los componentes principales del sistema y sus responsabilidades funcionales. Este modelo define cómo se estructuran las funcionalidades clave (como la búsqueda de eventos, la visualización de promociones y la interacción con usuarios) y cómo se comunican entre sí. El propósito es asegurar que todos los requerimientos del sistema sean cubiertos mediante una arquitectura clara, modular y orientada a servicios.

El sistema se divide en módulos lógicos con responsabilidades específicas: autenticación, búsqueda geolocalizada, reservas y pagos, notificaciones, y administración. Esto permite que cada componente pueda evolucionar o mantenerse de forma independiente, respetando principios como la **separación de responsabilidades** y la **cohesión funcional**.

* + 1. **Modelo de Componentes**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | **Familia**   |  | | --- | |  |  |  | | --- | | • Module (✓) • C&C (✓) • Allocation ( ) |  |  | | --- | |  |   **Estilo Arquitectural**  Cliente-servidor con separación lógica en frontend (React Native) y backend (APIs REST). Se sigue un modelo orientado a servicios. |
| **Convención**  Cada componente tiene responsabilidades específicas. La comunicación se realiza vía HTTP/REST, siguiendo estándares modernos para sistemas distribuidos. |
| Versión: | 1.0 | Título:   |  | | --- | |  |  |  | | --- | | Modelo de Componentes – Plataforma Desparches | | Nomenclatura:   |  | | --- | |  |  |  | | --- | | ComponenteFuncional\_<nombre> | |
| Arquitecto: | Carlos Daza (Arquitecto de Software) | | Fecha: 13 de mayo de 2025 |

## 2. Punto de Vista de Despliegue

* + 1. **Descripción**

El punto de vista de despliegue describe cómo se distribuyen los componentes del sistema Desparches sobre la infraestructura física y virtual utilizada. La arquitectura sigue un modelo cliente-servidor apoyado en servicios en la nube, lo que permite escalabilidad, disponibilidad continua y mantenimiento simplificado. Los clientes (aplicaciones móviles y web) interactúan con el sistema mediante internet, consumiendo servicios expuestos por el backend alojado en una plataforma cloud (como Firebase o Heroku).

El backend se conecta con servicios especializados como Firebase Firestore (base de datos), Firebase Storage (almacenamiento de archivos multimedia) y Firebase Cloud Messaging (notificaciones push). El despliegue está diseñado para admitir múltiples usuarios concurrentes, con servicios desacoplados que pueden escalarse de forma independiente, optimizando el rendimiento del sistema bajo alta demanda.

* + 1. **Modelos de Plataforma de Ejecución**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | **Familia**   * Module * C&C ( ) * Allocation ( )   **Estilo Arquitectural**:  Convención: Arquitectura basada en la nube – cliente/servidor |
| Separación entre frontend (móvil/web) y backend (API + base de datos) |
| Versión: | 1.0 | Título: Despliegue de la Plataforma Desparches | Nomenclatura:   |  | | --- | |  |  |  | | --- | | Desparches\_Despliegue\_v1.0 | |
| Arquitecto: | Equipo de Desparches | | Fecha:   |  | | --- | |  |  |  | | --- | | 13 de mayo de 2025 | |

* + 1. **Modelos de Red**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | **Familia**   * Module( ) * C&C ( ) * Allocation ( )   **Estilo Arquitectural:**   |  | | --- | |  |  |  | | --- | | Cliente-servidor con red IP y servicios REST | |
| **Convención:** Acceso por HTTPS, separación de red pública (cliente) y red interna (nube) |
| Versión: | 1.0 | Título: Modelo de Red – Plataforma Desparches | Nomenclatura: Red pública cliente ↔ nube backend y servicios internos |
| Arquitecto: | Equipo de Desparches | | Fecha:   |  | | --- | |  |  |  | | --- | | 13/05/2025 | |

* + 1. **Modelos de Dependencia Tecnológica**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | **Familia**   * Module( ) * C&C ( ) * Allocation ( )   **Estilo Arquitectural:**   |  | | --- | |  |  |  | | --- | | Cliente-servidor con dependencias distribuidas en la nube | |
| **Convención:**   |  | | --- | |  |  |  | | --- | | Las dependencias están organizadas por capa (presentación, lógica, datos) | |
| Versión: | 1.0 | Título: Modelo de Dependencias Tecnológicas – Desparches | Nomenclatura: Cliente/Servidor/Servicios nube |
| Arquitecto: | Equipo de Desparches | | Fecha:   |  | | --- | |  |  |  | | --- | | 13/05/2025 | |

# 3. Punto de Vista de Información

* + 1. **Descripción**

El punto de vista de información representa la estructura estática de los datos que maneja el sistema **Desparches**, reflejando las entidades clave y sus relaciones. Este modelo es esencial para comprender cómo se organizan y almacenan los datos en la base de datos, así como para definir las dependencias entre los módulos de lógica del sistema.

En **Desparches**, las entidades centrales son **Usuario**, **Evento**, **Promoción**, **Reserva**, y **Notificación**. Estas entidades permiten soportar la funcionalidad del sistema de recomendación, búsquedas personalizadas, interacción social, reservas y pagos.

* + 1. **Modelos de Estructuras Estáticas de Datos**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Familia**   * Module( ) * C&C ( ) * Allocation ( )   **Estilo Arquitectural**   |  | | --- | |  |  |  | | --- | | Modelo entidad-relación extendido orientado a objetos para representar datos y relaciones dentro del sistema Desparches. Basado en un enfoque relacional lógico. | |
| **Convención**  Las entidades están nombradas con sustantivos en singular. Las relaciones son bidireccionales cuando aplica y llevan identificadores únicos. |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Versión: | 1.0 | Título:   |  | | --- | |  |  |  | | --- | | Modelo de Estructuras de Datos – Plataforma Desparches | | Nomenclatura:   |  | | --- | |  |  |  | | --- | | Entidad\_<nombre> y Rel\_<Entidad1>\_<Entidad2> | |
| Arquitecto: | Carlos Daza | | Fecha: 13 de mayo de 2025 |

* + 1. **Modelos de Flujo de Información**

|  |  |
| --- | --- |
|  | **Familia**   * Module( ) * C&C ( ) * Allocation ( )   **Estilo Arquitectural**  Modelo basado en flujo de datos asíncrono entre servicios REST, con componentes desacoplados para escalabilidad y resiliencia. |
| **Convención**  Los flujos representan transferencia de datos entre módulos, servicios externos y base de datos. |

|  |  |  |  |
| --- | --- | --- | --- |
| Versión: | 1.0 | Título: Modelo de Flujo de Información – Plataforma Desparches | Nomenclatura:  FlujoInfo\_<NombreProceso> |
| Arquitecto: | Carlos Daza | | Fecha:   |  | | --- | |  |  |  | | --- | | 15 de mayo de 2025 | |

* + 1. **Modelos de Ciclo de Vida de Información**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | **Familia**   * Module( ) * C&C ( ) * Allocation ( )   **Estilo Arquitectural**  Modelo basado en etapas claramente definidas de creación, validación, almacenamiento, actualización y eliminación de datos. |
| **Convención**  Cada entidad sigue un ciclo completo con estados definidos: Creado, Validado, Activo, Archivado, Eliminado. |
| Versión: | 1.0 | Título: Modelo basado en etapas claramente definidas de creación, validación, almacenamiento, actualización y eliminación de datos. | Nomenclatura:   |  | | --- | |  |  |  | | --- | | CicloVidaInfo\_<Entidad> | |
| Arquitecto: | Carlos Daza | | Fecha: 15 de mayo de 2025 |

## 4. Punto de Vista de Concurrencia

* + 1. **Descripción**

La plataforma Desparches está diseñada para operar en un entorno altamente concurrente, permitiendo la interacción simultánea de múltiples usuarios y comercios. Utiliza una arquitectura cliente-servidor con comunicación asíncrona basada en peticiones HTTP/HTTPS. Cada usuario realiza solicitudes independientes al servidor backend, el cual maneja múltiples hilos o funciones serverless para atender las peticiones sin bloquear otros procesos. Además, se ejecutan tareas en segundo plano como el envío de notificaciones o el refresco de contenido, lo que demanda una arquitectura concurrente eficiente. El sistema debe garantizar consistencia, disponibilidad y capacidad de respuesta bajo carga.

* + 1. **Modelos**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | **Familia**   * Module( ) * C&C ( ) * Allocation ( )   **Estilo Arquitectural:** Modelo basado en eventos asíncronos y servicios desacoplados |
| **Convención:** Cada cliente ejecuta procesos independientes; backend maneja concurrencia con hilos o funciones sin estado (stateless functions). |
| Versión: | 1.0 | Título: Modelo de Concurrencia – Desparches | Nomenclatura: Solicitudes concurrentes por usuarios + servicios en segundo plano |
| Arquitecto: | Equipo de Desparches | | Fecha:   |  | | --- | |  |  |  | | --- | | 13/05/2025 | |

## 5. Punto de Vista de Desarrollo

* + 1. **Descripción**

El punto de vista de desarrollo de **Desparches** muestra cómo se organiza el sistema desde la perspectiva del desarrollo de software. Describe la **estructura de código fuente**, las **asignaciones de responsabilidades entre los equipos**, y la manera en que se gestionan los **módulos del sistema** para facilitar el trabajo colaborativo, la reutilización, y el mantenimiento.

La aplicación se estructura siguiendo una arquitectura cliente-servidor, con una separación clara entre frontend (React Native para dispositivos móviles), backend (APIs REST en Node.js o Firebase Functions), y base de datos (NoSQL - Firebase Firestore). El desarrollo está dividido en **módulos funcionales**, cada uno con su propio repositorio o carpeta dentro del control de versiones, facilitando la colaboración entre los diferentes miembros del equipo.

* + 1. **Modelos**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | **Familia**   * Module( ) * C&C ( ) * Allocation ( )   **Estilo Arquitectural:** Arquitectura en capas, orientada a módulos y basada en controladores-servicios-modelos (MVC simplificado) |
| **Convención:** Estructura modular por funcionalidades. Uso de patrones MVC. Separación de responsabilidades en frontend y backend. |
| Versión: | 1.0 | Título:   |  | | --- | |  |  |  | | --- | | Punto de Vista de Desarrollo – Plataforma Desparches | | Nomenclatura: Modulo\_<nombre> o Feature\_<nombre> |
| Arquitecto: | Carlos Daza | | Fecha:  13/05/2025 |

Sección 6. Relaciones entre los Puntos de Vista

La arquitectura del sistema **Desparches** se basa en un enfoque de múltiples vistas, alineado con el estándar IEEE 1471-2000 (actualmente ISO/IEC/IEEE 42010), el cual permite modelar diferentes aspectos del sistema para responder de forma adecuada a las preocupaciones de los diversos stakeholders involucrados (IEEE, 2000). A continuación, se detallan las relaciones y consistencia entre cada punto de vista utilizado en este documento arquitectónico.

### **6.1 Punto de Vista Funcional**

El punto de vista funcional establece los módulos principales del sistema, incluyendo la autenticación, búsqueda de eventos y promociones, reservas, notificaciones y administración. Estos módulos definen los comportamientos esenciales del sistema y son la base sobre la cual se desarrollan las demás vistas. Este punto de vista guía tanto la organización lógica del sistema como su comportamiento esperado (Rozanski & Woods, 2012).

### **6.2 Punto de Vista de Despliegue**

Este punto de vista representa cómo se asignan los componentes funcionales a la infraestructura física o virtual. En Desparches, se utiliza una arquitectura cliente-servidor con servicios desplegados en la nube. Existe una relación directa entre los módulos definidos en la vista funcional y su distribución en servidores o servicios cloud, lo que asegura consistencia entre el diseño lógico y su ejecución física (Bass, Clements & Kazman, 2012).

### **6.3 Punto de Vista de Red**

El punto de vista de red complementa al de despliegue al describir cómo se comunican los nodos a través de protocolos como HTTPS. Las interacciones entre aplicaciones móviles, el frontend web y el backend se establecen mediante una red pública que garantiza la conectividad segura. Este modelo mantiene coherencia con la vista de información, ya que se intercambian datos definidos en las estructuras del dominio mediante estas rutas de comunicación.

### **6.4 Punto de Vista de Dependencia Tecnológica**

Esta vista detalla las herramientas y plataformas utilizadas (Firebase, React, Node.js, entre otras). Las dependencias tecnológicas se asignan a cada módulo funcional de forma coherente, lo que asegura compatibilidad entre diseño y desarrollo. Además, se relaciona directamente con el punto de vista de desarrollo, ya que define el entorno técnico sobre el cual se estructura el código y se gestiona la implementación.

### **6.5 Punto de Vista de Información**

El modelo de información describe las entidades clave del sistema (Usuario, Evento, Promoción, etc.), sus atributos, relaciones y ciclos de vida. Este punto de vista está alineado con la vista funcional, ya que cada módulo interactúa con estas entidades. Asimismo, mantiene consistencia con la vista de concurrencia y despliegue, garantizando que los datos estén disponibles de forma eficiente en múltiples instancias simultáneas del sistema.

### **6.6 Punto de Vista de Concurrencia**

Este punto de vista permite modelar el comportamiento del sistema bajo condiciones de uso simultáneo. Se define cómo el sistema gestiona múltiples usuarios y procesos concurrentes mediante mecanismos asincrónicos. Está estrechamente ligado a la vista funcional (donde se originan las acciones) y al despliegue (donde se ejecutan los servicios concurrentes), asegurando que el sistema funcione correctamente bajo carga.

### **6.7 Punto de Vista de Desarrollo**

Finalmente, el punto de vista de desarrollo organiza la estructura del sistema desde la perspectiva del equipo de ingeniería. Detalla cómo se distribuye el código fuente, los módulos y los equipos de trabajo. Este punto integra las decisiones tomadas en las demás vistas, asegurando que la implementación sea consistente con los modelos funcional, tecnológico y de información.

### **6.8 Conclusión**

La interrelación entre las vistas arquitectónicas permite garantizar la **consistencia, coherencia y trazabilidad** entre los distintos niveles de abstracción del sistema. Cada vista aporta una perspectiva complementaria, y juntas forman una representación integral y bien fundamentada de la solución. No se han identificado contradicciones entre vistas, lo cual refuerza la viabilidad técnica de la arquitectura planteada para Desparches.

Sección 7. Evaluación de la Arquitectura

## 7.1 Criterios de Evaluación Utilizados

Para garantizar la viabilidad, sostenibilidad y efectividad del sistema **Desparches**, se aplicaron los siguientes criterios de evaluación:

* **Eficiencia**: Tiempos de carga y respuesta < 2 segundos en búsquedas y navegación.
* **Escalabilidad**: Soporte esperado para al menos 10,000 usuarios concurrentes.
* **Usabilidad y Accesibilidad**: Interfaz intuitiva, adaptable a diferentes dispositivos y niveles de experiencia.
* **Seguridad**: Cumplimiento con RGPD, implementación futura de 2FA y cifrado de datos.
* **Mantenibilidad**: Modularidad del sistema que facilita actualizaciones y correcciones sin afectar otras funciones.
* **Disponibilidad**: Operación 24/7, tanto en entornos móviles como web.
* **Compatibilidad tecnológica**: Acceso desde distintos sistemas operativos (Windows, macOS, Android, iOS).

## 7.2 Métodos de Evaluación Aplicados

* **Análisis de escenarios de calidad** (eficiencia, seguridad, recuperación ante fallos, escalabilidad).
* **Evaluación comparativa** con plataformas similares como Eventbrite.
* **Revisión estructural basada en vistas arquitectónicas (IEEE 1471-2000)**.
* **Revisión de riesgos priorizados mediante matriz Impacto vs. Esfuerzo**.
* **Validación de prototipo funcional en Figma** frente a criterios de navegación, usabilidad y diseño visual.

## 7.3 Resultados de la Evaluación

#### **Fortalezas detectadas:**

* Arquitectura **orientada a servicios** con separación clara entre frontend y backend (cliente-servidor REST).
* Diseño modular que permite **incorporar nuevas funciones sin rediseñar la plataforma**.
* Interfaz accesible y adaptable, pensada para **usuarios con diferentes niveles de alfabetización digital**.
* Filtros avanzados y mapa interactivo que mejoran la personalización y experiencia de usuario.
* Entorno multiplataforma: compatible con PC, tablets y móviles.

#### **Aspectos por mejorar:**

* El prototipo actual **carece de backend funcional real**, lo que impide medir rendimiento y tiempos de respuesta reales.
* Alta **dependencia de terceros** (APIs como Google Maps, servicios de hosting como Firebase o AWS).
* Falta de mecanismos definidos para contingencias ante **fallas en servicios externos** o caídas del sistema.
* Se requiere reforzar el plan de pruebas de carga y estrés, así como definir escenarios de **recuperación ante fallos y backups**.

## 7.4 Conclusión

La arquitectura del sistema Desparches presenta una base robusta, moderna y alineada con las necesidades del negocio y los usuarios. Su diseño escalable, modular y centrado en la experiencia del usuario, la posiciona como una solución innovadora frente a la falta de plataformas unificadas para descubrir y gestionar eventos en tiempo real.

Se recomienda que, en la siguiente fase de desarrollo, se prioricen:

* La **implementación de los componentes backend**.
* La **integración efectiva con servicios externos** mediante pruebas controladas.
* La definición de **mecanismos de recuperación, auditoría de seguridad** y monitoreo en tiempo real.
* Una **evaluación real de rendimiento**, especialmente en entornos móviles con baja conectividad.

**Sección 8. Directorio**

**Nombre del Sistema:** Desparches  
 **Versión:** 1.0  
 **Fecha:** 13 de mayo de 2025  
 **Nombre del Equipo de Trabajo:** UpStream

**Integrantes del Equipo**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Rol** | **Correo electrónico** |
| **Laura Montaña** | Analista de Requisitos | laurag-montanag@unilibre.edu.co |
| **Alison Tinoco** | Diseñadora UX/UI | alisond-tinocom@unilibre.edu.co |
| **Daniela Perez** | Desarrolladora Frontend | laurad-perezj@unilibre.edu.co |
| **Carlos Daza** | Desarrollador Backend / Arquitecto | carlose-dazac@unilibre.edu.co |
| **Sara Castelblanco** | Líder de Calidad / Tester | sarav-castelblancog@unilibre.edu.co |

### **Responsabilidades Generales del Equipo**

* **Líder de Proyecto / Arquitectura (Carlos Daza):** define la estructura técnica del sistema, elige tecnologías y gestiona decisiones clave de arquitectura.
* **Diseño de Experiencia de Usuario (Alison Tinoco):** responsable del diseño visual, flujos de interacción y accesibilidad.
* **Implementación Frontend (Daniela Perez):** desarrollo de la interfaz del usuario en web y móvil.
* **Análisis y Recolección de Requisitos (Laura Montaña):** se encarga de entender las necesidades de los usuarios y traducirlas a requerimientos técnicos.
* **Pruebas y Validación (Sara Castelblanco):** ejecuta pruebas funcionales y no funcionales, asegura la calidad del sistema.

1. [↑](#endnote-ref-9245)
2. [↑](#footnote-ref-19788)