**[VitaLink]**

**(DAS) Documento Arquitectura Sistema**

**Versión 1.0**

**Identificación de Documento**

| **Identificación** | DAS1 |
| --- | --- |
| **Proyecto** | VitaLink |
| **Versión** | 1.0 |

| **Documento mantenido por** | Equipo de Arquitectura |
| --- | --- |
| **Fecha de ultima revisión** | 2025-10-06 |
| **Fecha de próxima revisión** | 2025-12-01 |

| **Documento aprobado por** | Dirección de Proyecto |
| --- | --- |
| **Fecha de última aprobación** | 2025-10-06 |

**Historia de Revisiones**

| **Fecha** | **Versión** | **Descripción** | **Autor** |
| --- | --- | --- | --- |
| 2025-09-20 | 1.0 | Desarrollo de documentos |  |
| 2025-10-06 | 1.5 | Cierre documentación |  |
|  |  |  |  |
|  |  |  |  |

**Tabla de Contenidos**

[**1**](#_heading=h.p2v0d73bkfgm) **Introducción 3**

[1.1](#_heading=h.686mwuhf426) Contexto del Problema 3

[1.2](#_heading=h.qik036wuci8h) Propósito 3

[1.3](#_heading=h.dpucorn4mynq) Ámbito 3

[1.4](#_heading=h.qbsio6t1qbk1) Definiciones, acrónimos y abreviaciones 3

[1.5](#_heading=h.ptzbsokyx5up) Referencias 3

[1.6](#_heading=h.c3nk1uexxs58) Resumen ejecutivo 3

[1.7](#_heading=h.8wxwihx5hao2) Representación 4

[**2**](#_heading=h.eztn8kg1e3rm) **Metas y Restricciones de la Arquitectura 5**

[2.1](#_heading=h.77dyzx3rh7kd) Metas de la arquitectura 5

[2.2](#_heading=h.j1fp8usjydz5) Restricciones de la Arquitectura 5

[2.3](#_heading=h.t8fnvxy88gul) Otros antecedentes y consideraciones 5

[**3**](#_heading=h.ivaxk3uv47s3) **Vista de Escenarios 6**

[3.1](#_heading=h.mit1n3o6zrpv) Modelo de Casos de Uso 6

[3.2](#_heading=h.6tqzsnrv5c4s) Casos de Usos Extendidos 6

[3.3](#_heading=h.uzykbl4hcutz) Especificación de los Escenarios de Calidad Relevantes 7

[**4**](#_heading=h.ixek7big6a2n) **Vista de Procesos 8**

[**5**](#_heading=h.gwgq2nb3t4co) **Vista Lógica 9**

[5.1](#_heading=h.z54bri1imehl) Parte Estructural ( Diagrama de Clases y Diagrama Relacional) 9

[*5.1.1*](#_heading=h.vhareeefc2v) *Descripción de Clases 9*

[*5.1.2*](#_heading=h.4h47wuxrqb04) *Descripción de Tablas 10*

[5.2](#_heading=h.38ikyllrfkcv) Parte Dinámica (Diagrama de Secuencias) 11

[**6**](#_heading=h.mclahywk9uvt) **Vista de Desarrollo o Despliegue 12**

[**7**](#_heading=h.pnehpd7qt0yo) **Vista Fisica 13**

[**8**](#_heading=h.arsefm49yko0) **Decisiones de Diseño y Selección de Alternativas 14**

[**9**](#_heading=h.qys3cqoy1pof) **Análisis de Reutilización 15**

1. **Introducción**
   1. **Contexto del Problema**

El proyecto surge ante la necesidad de digitalizar y centralizar el registro de crisis de pánico dentro de entornos clínicos, optimizando la gestión de datos entre pacientes y profesionales de la salud. Actualmente, los registros se realizan de manera manual o con herramientas no integradas, lo que genera pérdida de información, baja trazabilidad y falta de métricas útiles para la evaluación clínica.

Desde una perspectiva técnica, se requiere desarrollar un sistema compuesto por una aplicación móvil y un servicio web interconectado mediante APIs RESTful, con autenticación segura, gestión de usuarios por rol y almacenamiento de datos en tiempo real. Esta infraestructura busca mejorar la continuidad terapéutica, garantizar la privacidad de la información sensible y ofrecer soporte analítico para la toma de decisiones médicas.

* 1. **Propósito**

El propósito del proyecto es desarrollar un sistema integral orientado a la gestión y seguimiento de crisis de pánico, compuesto por una aplicación móvil para pacientes y un servicio web para profesionales de la salud. El sistema permitirá registrar eventos, consultar métricas clínicas y mantener una comunicación estructurada entre los distintos perfiles de usuario. Además, busca garantizar la seguridad, trazabilidad y disponibilidad de los datos mediante una arquitectura moderna y escalable, sirviendo como base para futuras integraciones con otros sistemas médicos o plataformas institucionales.

* 1. **Ámbito**

Frontend web para doctores (admin).

App móvil para pacientes.

API y BaaS (Supabase) para autenticación, base de datos y políticas RLS.

ETL ligero para métricas y paneles.

* 1. **Definiciones, acrónimos y abreviaciones**

| **ACRONIMO** | **DESCRIPCION** |
| --- | --- |
| *RLS* | Row Level Security |
| *JWT* | JSON Web Token |
| *BaaS* | Backend as a Service |
| *DAS* | Documento de Arquitectura de Sistema |
| *CU* | Backend as a Service |
| *PostgreSQL* | Sistema de gestión en base de datos utilizada en este proyecto |
| *RPC* | Remote Procedure Call (Procedimiento Almacenado) |

* 1. **Referencias**

Estándares de autenticación y autorización basados en JWT.

Políticas RLS y esquemas de Supabase.

- [Informe ERS](https://docs.google.com/document/d/1HHJSG3mBfjG8w2mygU0FlCsJiRsQpksK/edit?usp=drive_link&ouid=112919173007538279422&rtpof=true&sd=true)

* 1. **Resumen ejecutivo**

La solución propuesta se basa en una arquitectura modular respaldada por servicios gestionados. Supabase se utiliza como plataforma principal para la autenticación, la gestión de la base de datos PostgreSQL y el control de acceso mediante Row Level Security (RLS). El sistema establece una relación jerárquica en la cual el profesional de la salud crea, habilita y administra las cuentas de sus pacientes, manteniendo la trazabilidad y responsabilidad clínica.

La aplicación móvil opera exclusivamente para pacientes validados y vinculados a un profesional activo. Se implementan vistas especializadas para el registro de eventos, seguimiento del tratamiento y visualización de métricas clínicas. En todo el desarrollo se priorizan principios de seguridad, auditabilidad, escalabilidad y mantenibilidad del sistema.

* 1. **Representación**

La arquitectura del sistema **VitaLink** está representada siguiendo el enfoque del framework 4+1 y las recomendaciones del proceso unificado. Las vistas incluidas en esta versión del documento son:

* **Vista de Escenarios**: Describe los casos de uso más significativos, presenta los actores y una descripción de sus casos de uso asociados. De igual forma describe los escenarios de calidad más relevantes para la arquitectura.

**Casos de uso significativos (resumen):**

* CU1 — Registrar crisis de pánico (Paciente).
* CU2 — Visualizar historial clínico (Doctor).
* CU3 — Gestionar pacientes (Doctor).
* CU4 — Consultar métricas y reportes (Doctor / Paciente).
* CU5 — Envío y recepción de notificaciones en tiempo real (Paciente / Doctor).
* CU6 — Autenticación y gestión de accesos (Todos los actores).
* CU7 — Auditoría y revisión de logs (Administrador).
* **Vista de Procesos**: Describe los procesos involucrados para darle sentido a la ejecución del sistema, así como sus relaciones de comunicación y sincronización.

**Flujo de autenticación:**

1. Doctor crea/valida cuenta.
2. Paciente recibe credenciales/habilitación.
3. App móvil solicita token a Supabase Auth.
4. API valida token y aplica RLS.

**Flujo de registro de crisis:**

1. Paciente.
2. App móvil.
3. Endpoint REST (POST /events).
4. Backend valida y persiste en PostgreSQL.
5. Trigger/worker actualiza métricas agregadas.
6. Notificación al Doctor si aplica.

**Flujo de métricas:**

1. Solicitud del Doctor.
2. Backend ejecuta consultas agregadas / vistas materializadas.
3. Respuesta con series temporales para render en UI.

**Eventos en tiempo real:**

1. Cambios críticos publicados vía Supabase Realtime / WebSocket.
2. Clientes suscritos reciben actualizaciones; alternativamente FCM para notificaciones push.

**Sincronización y coherencia:** escrituras síncronas en la tabla principal y procesos asíncronos (workers) para cómputos pesados; políticas de reconciliación ante fallos de red.

* **Vista Lógica**: Describe la arquitectura del sistema presentando varios niveles de refinamiento. Indica los módulos lógicos principales, sus responsabilidades y dependencias.
* **Módulo de Autenticación y Roles**
  + - Responsabilidad: registro, login, gestión de roles, tokens.
    - Interacción: todos los actores.
    - Tecnologías: Supabase Auth, JWT, RLS.
* **Módulo de Gestión de Pacientes**
  + - Responsabilidad: CRUD de pacientes, vinculación paciente–doctor, permisos.
    - Interacción: Doctor (gestiona), Paciente (consulta/edita su perfil).
* **Módulo de Registro de Eventos**
  + - Responsabilidad: captura y validación de eventos clínicos (fecha, hora, síntomas, duración, contexto).
    - Interacción: Paciente (crea), Doctor (visualiza/filtra).
* **Módulo de Seguimiento y Métricas**
  + - Responsabilidad: agregación, series temporales, gráficos, reportes clínicos.
    - Interacción: Doctor y Paciente (vistas personalizadas).
* **Módulo de Notificaciones**
  + - Responsabilidad: envío de alertas push/web y distribución de eventos en tiempo real.
    - Tecnologías: Firebase Cloud Messaging (FCM) o Supabase Realtime.
* **Módulo de Comunicación API**
  + - Responsabilidad: definir endpoints, validaciones, manejo de errores y versión de API.
    - Tecnologías: Django REST Framework.
* **Módulo de Auditoría / Administración**
  + - Responsabilidad: logs, reports de uso, revisión de auditorías, gestión de backups y políticas RTO/RPO.
    - Interacción: Administrador.
* **Vista de Desarrollo o Despliegue**: Describe los componentes de deployment construidos y sus dependencias.
  + **Cliente móvil:** dispositivos Android con acceso a internet.
  + **Servidor backend:** desplegado en **Supabase** (servicios gestionados) y Django hosteado en **Vercel**.
  + **Base de datos:** **PostgreSQL** en Supabase.
  + **Repositorio:** **GitHub** para control de versiones e integración continua (CI/CD).
  + **Infraestructura de seguridad:** autenticación basada en JWT, políticas RLS y cifrado de datos en tránsito (HTTPS).
* **Vista Física**: Describe restricciones tecnológicas, normativas, estándares, etc., los cuales influyen sobre las decisiones arquitectónicas, del producto y del proceso de desarrollo.

**Restricciones tecnológicas**

* Uso de **Supabase** como BaaS principal (PostgreSQL gestionado, Auth, Realtime).
* Frontend móvil en **React Native**; backend en **Django + DRF**.
* Comunicaciones **RESTful**; manejo de errores con códigos HTTP y payloads estandarizados.
* Notificaciones mediante **FCM** o canal Realtime.
* Control de versiones con **GitHub** y CI/CD automatizado.

**Restricciones normativas y de seguridad**

* Cumplimiento de la **Ley N° 19.628 (Chile)** sobre protección de datos personales y buenas prácticas de privacidad.
* **RLS** para aislamiento por fila y minimización del acceso a datos sensibles.
* Encriptación TLS para todas las comunicaciones (HTTPS) y cifrado recomendado para datos sensibles en reposo.
* Registro obligatorio de auditoría para operaciones críticas (creación/edición/eliminación de datos clínicos).

**Estándares y buenas prácticas**

* APIs RESTful versionadas y documentadas (OpenAPI/Swagger).
* Pruebas automatizadas (unitarias y de integración) y pruebas de seguridad (vulnerabilities scanning).
* Políticas de backup y recuperación (RTO/RPO definidos).
* Metodología de trabajo ágil (Scrum) con Sprints, definición de Done y revisión continua.

1. **Metas y Restricciones de la Arquitectura**

A continuación, se revisan las metas y restricciones de la arquitectura.

* 1. **Metas de la arquitectura**

Seguridad: RLS, mínimo privilegio, cifrado en tránsito, autenticación por correo/contraseña o

magic-link.

Privacidad: Aislamiento por doctor, acceso del paciente solo a sus datos.

Disponibilidad: ≥ 99,5% objetivo.

Escalabilidad: Horizontal en BaaS y CDN.

Trazabilidad: Auditoría de operaciones clínicas.

Modificabilidad/Reuso: Capas desacopladas, contratos API estables.

Operatividad: Observabilidad básica y alarmas.

* 1. **Restricciones de la Arquitectura**

Tiempo acotado de construcción (cronograma académico).

Uso de Supabase para auth y datos.

Sin licenciamiento adicional de pago.

Políticas de datos sensibles de salud.

* 1. **Otros antecedentes y consideraciones**

Repositorio único con monorepo ligero para web y móvil.

Librería UI consistente para web y móvil.

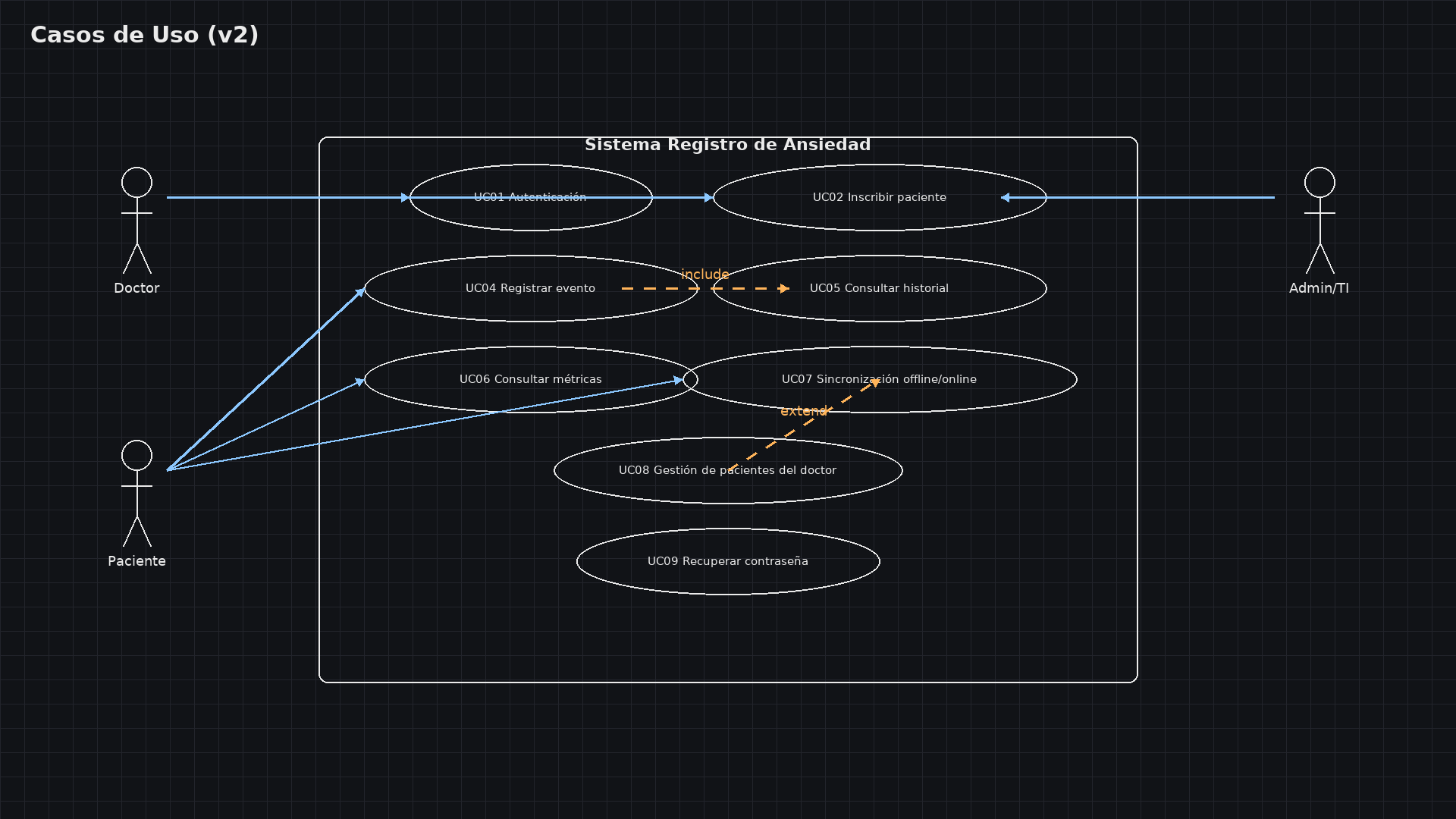
**Vista de Escenarios**

Esta sección describe en detalle el conjunto de escenarios funcionales y no funcionales que obtuvieron la mayor prioridad en el análisis. Para esto se presenta y describe el diagrama de casos de uso y los casos de uso prioritarios, así como los escenarios en que uno o más atributos de calidad se ven involucrados de manera significativa.

* 1. **Modelo de Casos de Uso**

Agregar el modelo de caso uso general del sistema

**Ilustración 1: Diagrama de Caso Uso General del Sistema**



* 1. **Casos de Usos Extendidos**

Los casos de uso considerados son los más relevantes para el desarrollo de la arquitectura. Se adjunta el documento o planilla caso uso.

**Adjuntar la planilla caso uso extendido**

A continuación, se listan los casos de uso relevantes, los cuales pueden ser encontrados con su especificación detallada en el documento “Casos de Uso Extendido”.

| **Código** | **Nombre** | **Actores** | **Prioridad** |
| --- | --- | --- | --- |
| CU-001 | Autentificar Usuario | Todos los usuarios | Media |
| CU-002 | Registrar Paciente | Doctor | Alta |
| CU-003 | Habilitar / Deshabilitar paciente | Doctor | Alta |
| CU-004 | Registrar evento de salud | Paciente | Alta |
| CU-005 | Consultar eventos y métricas | Doctor / Paciente | Media |
| CU-006 | Reasignar paciente a otro doctor | Doctor | Media |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* 1. **Especificación de los Escenarios de Calidad Relevantes**

Después de un análisis en conjunto con los stakeholders, los escenarios de calidad se expresan a continuación:

| Identificador: Q-SEC-01 | | |
| --- | --- | --- |
| Escenario(s): | | Acceso de paciente solo si está habilitado y vinculado. |
| Atributos de Calidad relevantes: | | Seguridad, Privacidad. |
| Componentes del Escenario | Estímulos: | Inicio de sesión de un paciente no habilitado. |
| Fuente del estimulo | Aplicación móvil del paciente. |
| Ambiente: | Operación normal del sistema. |
| Artefacto: | Módulo de autenticación / RLS. |
| Respuesta: | El sistema bloquea el acceso y muestra un mensaje de estado informando que la cuenta no está habilitada. |
| Medida de Respuesta | 100 % de intentos bloqueados según las políticas RLS definidas. |

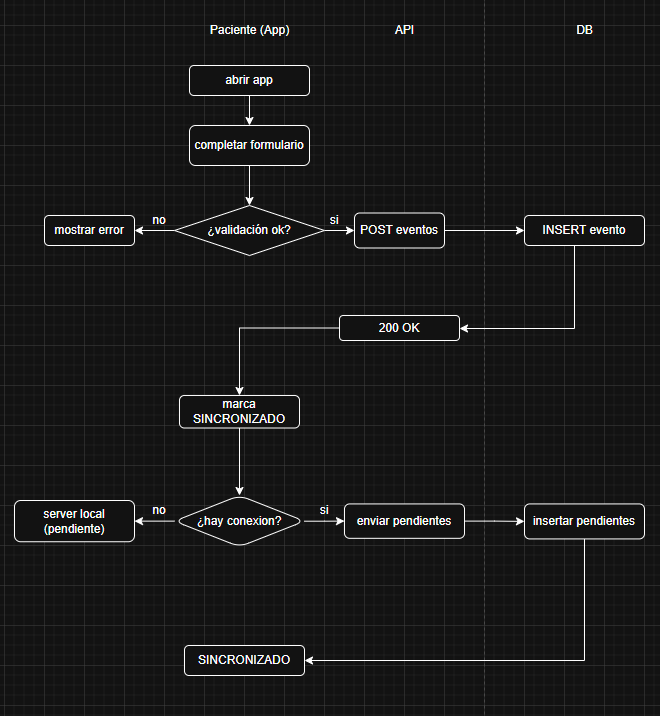
| Identificador: Q-AUD-01 | | |
| --- | --- | --- |
| Escenario(s): | | Registro de evento con trazabilidad. |
| Atributos de Calidad relevantes: | | Trazabilidad, Integridad. |
| Componentes del Escenario | Estímulos: | Paciente guarda un nuevo evento clínico. |
| Fuente del estimulo | Aplicación móvil del paciente. |
| Ambiente: | Uso normal. |
| Artefacto: | Servicio de eventos en la base de datos. |
| Respuesta: | El sistema persiste el evento incluyendo created\_by, timestamp y, opcionalmente, un hash de integridad. |
| Medida de Respuesta | 100 % de eventos almacenados con metadatos de auditoría completos. |

| Identificador: Q-AVA-01 | | |
| --- | --- | --- |
| Escenario(s): | | Continuidad del servicio ante caída del sistema o interrupción del servicio. |
| Atributos de Calidad relevantes: | | Disponibilidad, Recuperación, Confiabilidad. |
| Componentes del Escenario | Estímulos: | Ocurre una caída inesperada del servicio de base de datos o del backend. |
| Fuente del estimulo | Evento interno del sistema / proveedor de infraestructura. |
| Ambiente: | Producción. |
| Artefacto: | BaaS (Supabase), API de servicios. |
| Respuesta: | El sistema realiza el failover automático a la réplica, mantiene la operación esencial y reanuda las funciones críticas sin pérdida de datos. |
| Medida de Respuesta | RTO ≤ 15 minutos y RPO ≤ 5 minutos en el 100 % de los casos registrados. |

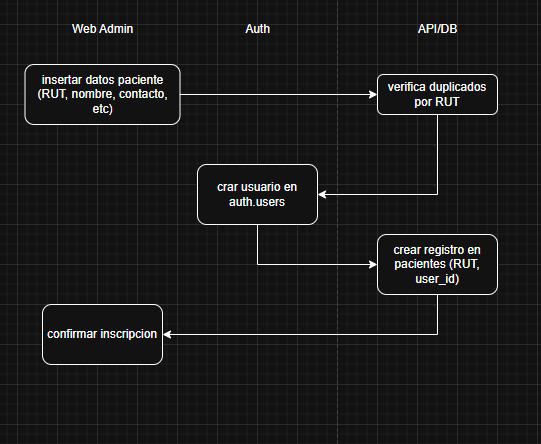
1. **Vista de Procesos**

**Ilustración 2: Diagramas de Actividades**

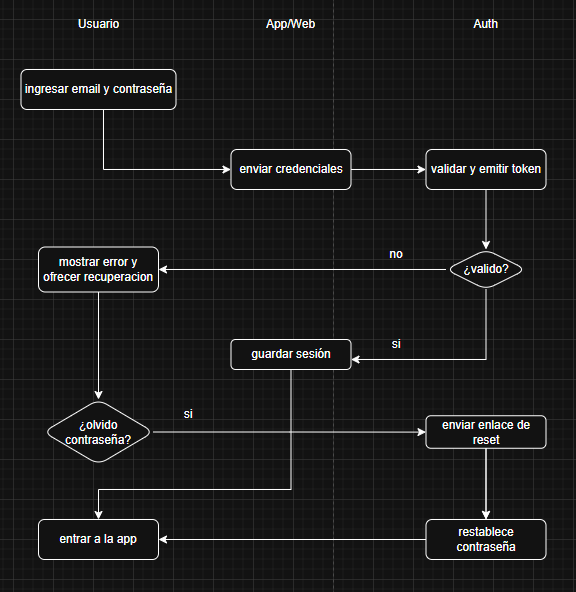
**1. Registrar evento + sincronización**



**2. Inscribir paciente (por el doctor)**



**3. Autenticación y recuperación**

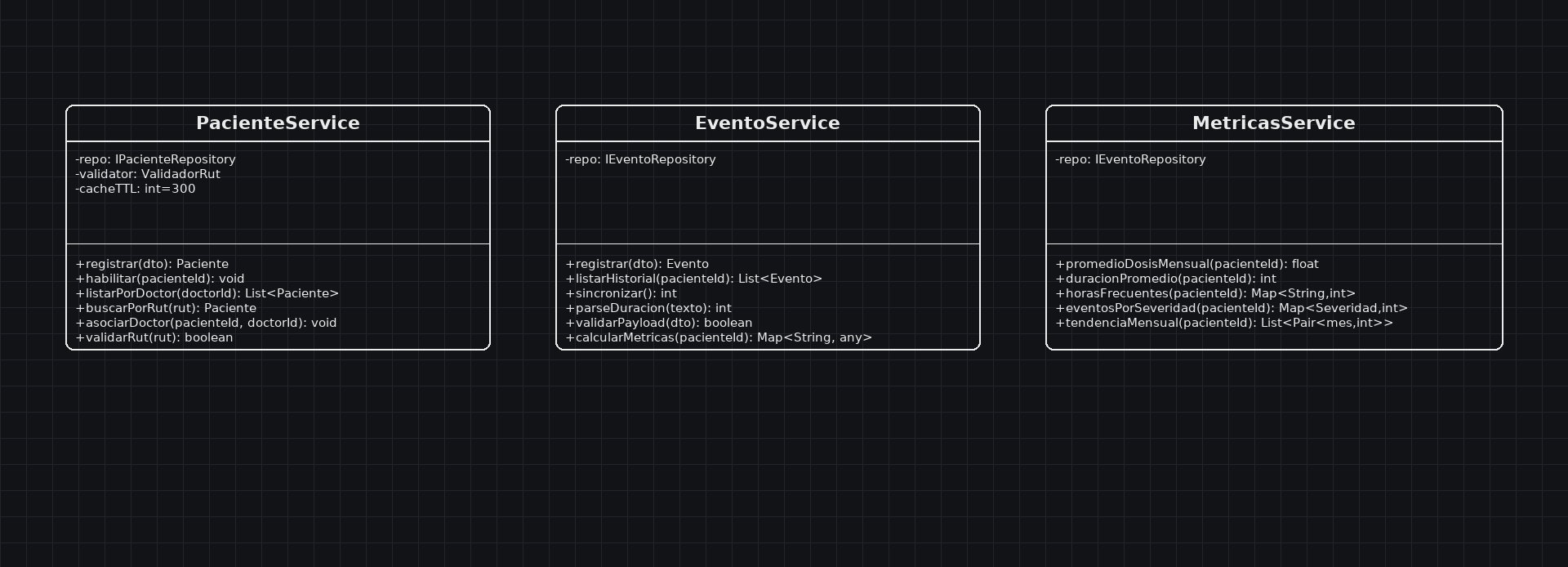


1. **Vista Lógica**

A continuación, se presenta una vista lógica de la aplicación expresado en tres diagramas, uno de ellos que muestra la parte estructural o estática de la aplicación (clases) y a la base de datos (modelo relacional).otra vista que representa la parte dinámica (secuencias).

* 1. **Parte Estructural ( Diagrama de Clases y Diagrama Relacional)**

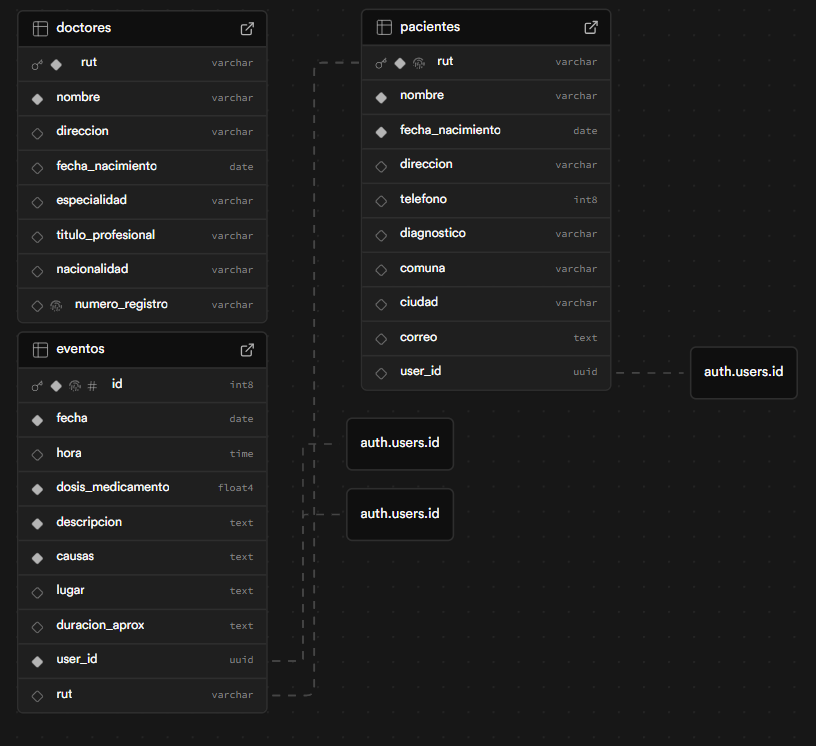
**Ilustración 3: Diagrama de Clases**

****

### Descripción de Clases

| **Código** | **Nombre** | **Descripción** |
| --- | --- | --- |
| CL-001 | Paciente | Orquesta casos de uso de pacientes. Valida RUT y datos, evita duplicados, crea/habilita/deshabilita pacientes, y lista por doctor. |
| CL-002 | Evento | Gestiona eventos clínicos del paciente. Registra eventos, obtiene historial por paciente y sincroniza pendientes offline→online |
| CL-003 | Métricas | Calcula indicadores derivados de eventos. Promedio de dosis, duración promedio, horas frecuentes, y agregados por severidad/tendencia. |
|  |  |  |

**Ilustración 4: Diagrama de Base Datos (Relacional)**



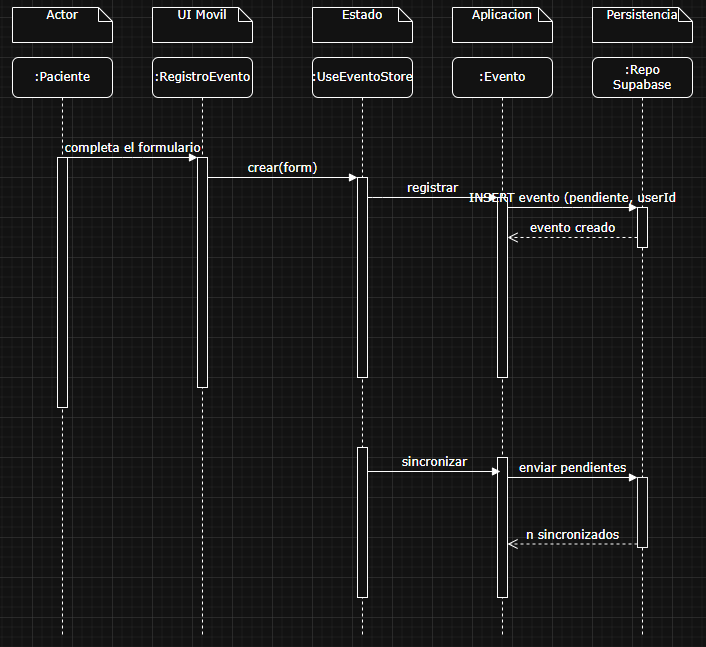
### Descripción de Tablas

| **Código** | **Nombre** | **Descripción** |
| --- | --- | --- |
| TB-001 | DOCTOR | doctores asociados a la clínica |
| TB-002 | PACIENTES | pacientes atendidos en la clínica que requieren seguimiento |
| TB-003 | EVENTOS | evento registrados por los pacientes durante el tratamiento |
|  |  |  |

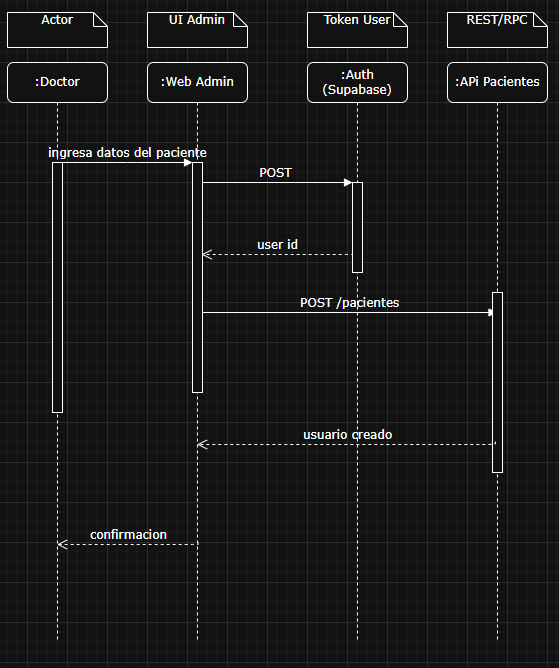
* 1. **Parte Dinámica (Diagrama de Secuencias)**

**Ilustración 4: Diagramas de Secuencias**

**Paciente registra eventos**



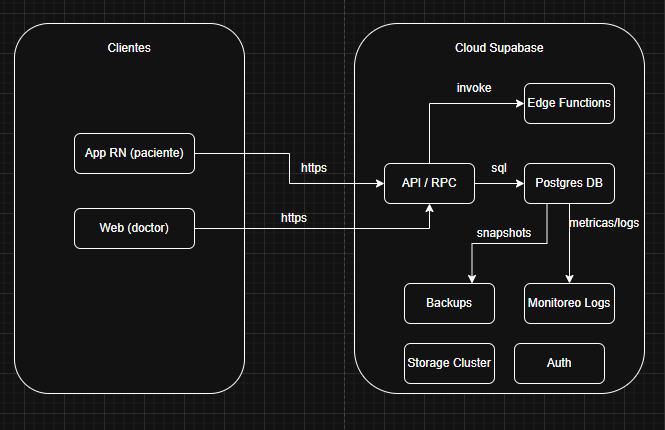
**Doctor registra pacientes**



1. **Vista de Desarrollo o Despliegue**

En esta vista se describen las componente o modulos en las cuales se dividirá o implementará el sistema

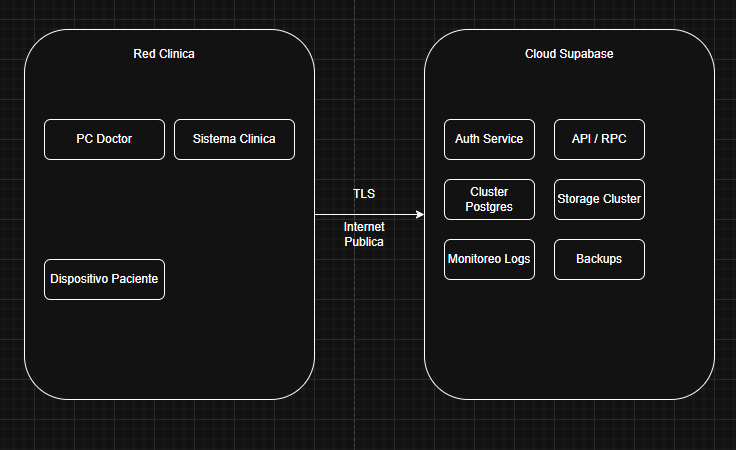
**Ilustración 5: Diagrama de Despliegue**



1. **Vista Física**

En esta vista se despliegan los nodos que participan con el sistema.

**Ilustración 6: Diagrama de componentes**



1. **Decisiones de Diseño y Selección de Alternativas**

Las principales decisiones arquitectónicas se tomaron considerando la restricción de *tiempo de construcción*. Dado que el proyecto debe implementarse en un período ajustado y sin holguras, se privilegió la adopción de una arquitectura conocida, de bajo riesgo y que facilite una implementación rápida y segura.

Asimismo, la arquitectura se **modularizó**, permitiendo que cada módulo sea **testeable de forma unitaria**, con el objetivo de asegurar que cada componente presente una baja tasa de fallos y sea fácilmente mantenible de manera independiente.

Un segundo propósito considerado en la **selección de alternativas** fue **garantizar la escalabilidad y mantenibilidad** del sistema. Por esta razón, se optó por una **arquitectura basada en servicios desacoplados**, integrados mediante **APIs RESTful**, lo que permite la independencia entre el **frontend móvil (React Native)**, el **backend web (Django)** y la **base de datos en la nube (Supabase - PostgreSQL)**.

Se eligió **Supabase** como plataforma *Backend as a Service (BaaS)* por su facilidad de integración, autenticación nativa, gestión de almacenamiento y soporte en tiempo real.  
 En cuanto al **frontend**, se seleccionó **React Native** por su capacidad **multiplataforma**, lo que permite una única implementación para **Android e iOS**.  
 Para el **desarrollo web**, se optó por **Django**, un framework maduro, seguro y basado en Python, ideal para la gestión del panel administrativo y los servicios médicos del sistema.

Finalmente, se adoptó la **metodología ágil Scrum** para garantizar **entregas iterativas**, **control de calidad continuo** y **flexibilidad ante cambios** durante el desarrollo del sistema.

1. **Análisis de Reutilización**

El desarrollo del sistema considera la **reutilización de componentes, módulos y patrones de diseño** previamente implementados en otras partes del proyecto o en soluciones existentes, con el objetivo de **optimizar los tiempos de desarrollo** y **mantener la coherencia funcional y visual** de la plataforma.

Los principales elementos reutilizables identificados son los siguientes:

* **Código base:** Se reutilizará la lógica del mantenedor de usuarios en los módulos de doctores y pacientes, adaptando las validaciones y roles según el tipo de usuario.
* **Diseño gráfico:** Se emplearán plantillas y estilos visuales uniformes (paleta de colores, tipografías y logotipos) en todos los mantenedores del sistema (clientes, doctores, pacientes y eventos), asegurando una experiencia visual consistente y profesional.
* **Componentes front-end:** Se reutilizarán componentes de interfaz desarrollados en **React Native** y **Django** (formularios, alertas, menús, botones), con el fin de mantener una estructura homogénea y reducir la duplicación de código.
* **Módulos de autenticación y permisos:** Se reutilizarán las funciones de autenticación provistas por **Supabase Auth** para la gestión de usuarios y roles, aplicando la misma lógica tanto en las interfaces web como móviles.
* **Conexión a base de datos:** Se reutilizarán las consultas y endpoints **RESTful** ya definidos para las operaciones **CRUD**, permitiendo una integración fluida entre el backend y los diferentes módulos del sistema.

En conjunto, estas reutilizaciones permitirán **reducir el esfuerzo de desarrollo**, **mejorar la mantenibilidad del sistema** y **asegurar una mayor estandarización** a nivel de arquitectura y experiencia de usuario.