Ficha Técnica de ***“Akuma no Pet”***

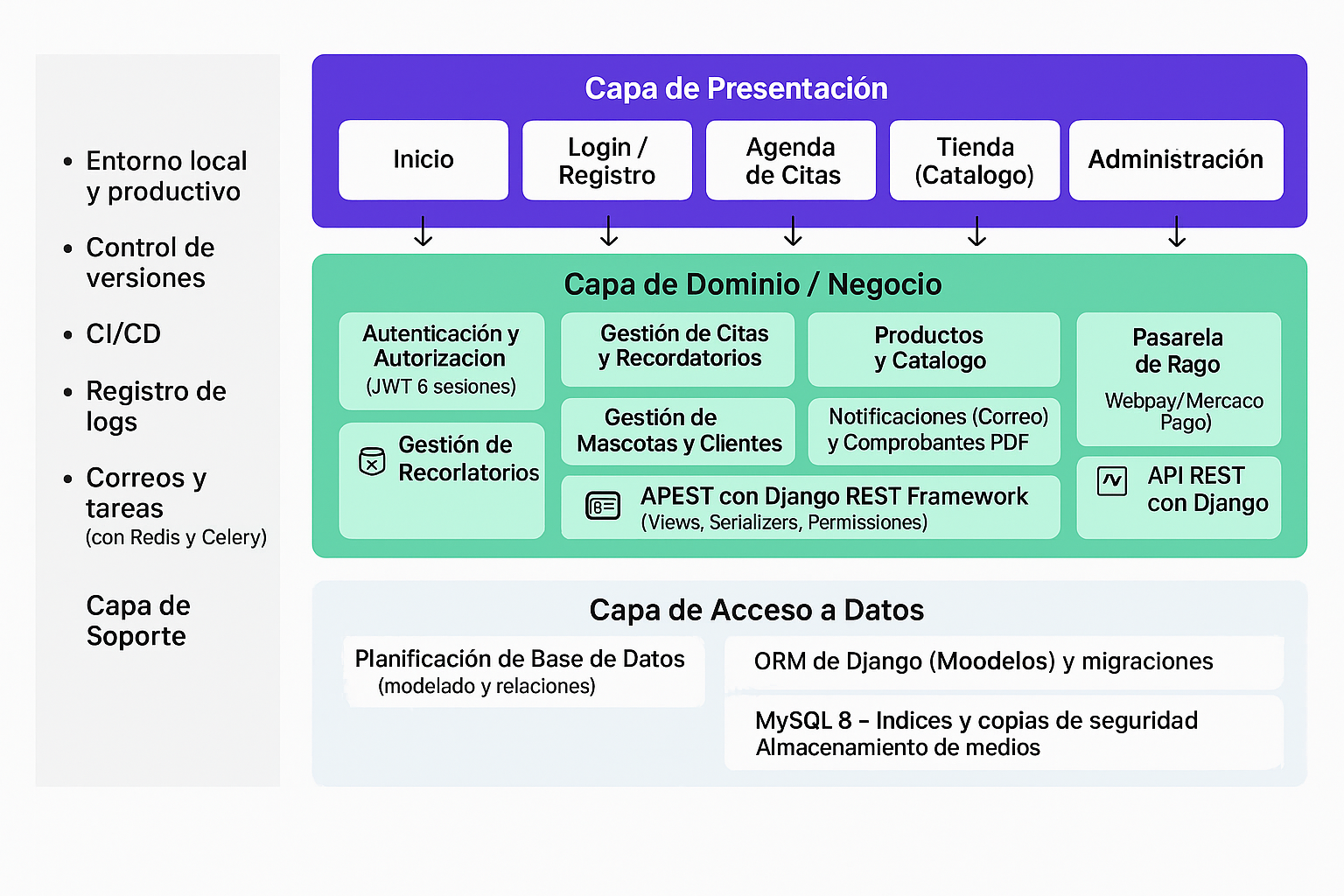
| Fecha | 01 de septiembre de 2025 |
| --- | --- |
| Autores | Roberto Olguin, Mario Galarce Alvaro Quiroz |
| Versión | 1 |

Alcance de la Ficha Técnica (Detalle)

| **1. Información general del sistema** | |
| --- | --- |
| Nombre Sistema | Akuma no Pet |
| Objetivo Principal | Ofrecer un agendamiento 100% en línea, simple e intuitivo, que reduzca llamados y errores, mejore la experiencia del dueño de mascota y aumente la eficiencia del centro veterinario. |
| Responsables del proyecto | **● Scrum Master:**  ○Roberto Olguin  **● Equipo de desarrollo:**  *○ Team RPZ:*  ■ Mario Galarce  ■ Alvaro Quiroz  **● Líder Técnico:**  ○ Mario Galarce  **● Product owner:**  ○ Ricardo Aravena  **● Stakeholder:**  ○ Gabriela Silva |
| Usuarios destinatarios | * **Externos:** clientes/dueños de mascotas. * **Internos:** veterinarios, recepcionistas, administradores. |

| **2. Descripción funcional de alto nivel** | |
| --- | --- |
| **Lista de funcionalidades principales** | |
| * Autenticación y registro (cliente/admin). * Gestión de clientes, mascotas y fichas clínicas. * **Agenda de citas** (crear, reagendar, cancelar, recordatorios). * **Catálogo** (productos y servicios), **carrito** y **pedidos**. * **Pagos en línea** y comprobantes. * Panel administrativo: usuarios, inventario, citas, reportes. | |
| **Caso de Uso** | |
| Usuario no Registrado | Usuario Registrado |
|  | |
| **Relación con procesos de negocio existentes** | |
| **Relación con procesos de negocio:** digitaliza agenda, reduce llamados/manualidad, ordena inventario/ventas y mejora trazabilidad clínica. | |

| **3. Arquitectura propuesta** | |
| --- | --- |
| **Tipo de arquitectura** | Arquitectura por ***capas***. |
| **Diagrama arquitectura** | |
|  | |
| **Patrón de diseño** | Monolito modular |
| **APIs** | **APIs:** REST/JSON sobre HTTPS para app web y posibles clientes futuros (móvil/desktop). |



| **4. Lenguajes de programación, frameworks y librerías** | |
| --- | --- |
| Lenguajes | * Python. * HTML. * CSS. * JavaScript. |
| Frameworks | * Django 5.x, Django REST Framework 3.15+. |
| Dependencias clave | * **djangorestframework**, **django-cors-headers**, **mysqlclient**/**django-mysql**, **Pillow** (imágenes). * **djangorestframework-simplejwt** (JWT) o sesiones Django (si sólo SSR). * **FullCalendar.js** (agenda visual), **Axios** (HTTP). |
| Versiones | **Django/DRF acelera CRUD/API con seguridad integrada; MySQL ofrece transaccionalidad y performance; stack estándar, mantenible y escalable.**  **Python 3.12.**  **Frameworks: Django 5.x, Django REST Framework 3.15+.**  **Frontend: HTML5, CSS3, JavaScript .** |

| **5. Requerimientos funcionales y no funcionales** | | |
| --- | --- | --- |
| **FUNCIONALES** | | |
| **N°** | **Nombre** | **Descripción** |
| RF1 | Registro | *Page* que permite a un usuario no registrado crear una cuenta en la página. |
| RF2 | Login | *Page* que permite a un usuario registrado iniciar sesión en la página. |
| RF3 | Autenticación Login | Sistema que se encarga de validar los datos ingresados en la pantalla de inicio de sesión. |
| RF4 | CRUD clientes/mascotas | El sistema debe permitir **crear, leer, actualizar y eliminar** información de clientes y sus mascotas (especie, raza, edad, historial básico). |
| RF5 | Agenda (crear/editar/cancelar) | Los clientes deben poder **agendar citas** para sus mascotas, mientras que el administrador/veterinario podrá editarlas, confirmarlas o cancelarlas. |

| RF6 | Recordatorios por correo | Enviar notificaciones automáticas por email a los clientes recordando sus citas. |
| --- | --- | --- |
| RF7 | Catálogo de productos/servicios | Mostrar en la tienda una lista de productos (alimentos, accesorios, medicamentos) y servicios veterinarios, con precios e imágenes. |
| RF8 | Carrito de compras | El cliente podrá añadir productos al carrito, modificar cantidades y eliminar ítems antes de confirmar el pedido. |
| RF9 | Creación de pedidos | Integrar una pasarela de pago (Webpay/Mercado Pago) que permita pagar de forma segura, validando con webhooks. |
| RF10 | Pago online | Función para cargar los datos del archivo .xml en el formulario. |
| RF11 | Comprobante/boleta (PDF simple) | Una vez confirmado el pago, generar un comprobante o boleta en PDF que el cliente pueda descargar. |
| RF12 | Panel administrativo | Los administradores podrán gestionar usuarios, citas, inventario de productos y pedidos desde un panel de control. |
| RF13 | Reportes básicos | El sistema debe mostrar reportes simples:   * Cantidad de citas por día/semana/mes. * Ventas totales en un período determinado. |
| RF14 | Recuperación de contraseña | El usuario podrá restablecer su contraseña mediante un correo de recuperación. |
| RF15 | Roles y permisos | El sistema debe diferenciar roles (cliente, veterinario, administrador) y limitar |
| RF16 | Historial de citas | Cada cliente podrá visualizar el historial de citas de sus mascotas, con fecha, motivo y estado. |
| RF17 | Buscador de productos y servicios | Los usuarios podrán filtrar productos por nombre, categoría o precio. |
| RF18 | Cierre de sesión | Función para limpiar la sesión del usuario y desconectarse. |
| RF19 | Recuperación de contraseña | *Page* accesible desde el inicio de sesión para poder solicitar el cambio de contraseña del usuario en caso de olvidarla. |
| RF20 | Gestión de inventario | El administrador podrá actualizar stock, precios y disponibilidad de productos. |

| **NO FUNCIONALES** | | |
| --- | --- | --- |
| N° | Nombre | Descripción |
| RNF 1 | Soportar 50 usuarios | El sistema debe ser capaz de manejar simultáneamente hasta 50 usuarios conectados sin degradación significativa en el rendimiento. |

| RNF 2 | Protección CSRF y JWT | Formularios web deben usar tokens CSRF y la API debe autenticar con JWT |
| --- | --- | --- |
| RNF 3 | Autenticación de Dos Factores | Verificación adicional para el *Login,* enviando un código al correo al usuario para aumentar la seguridad del sitio. |
| RNF4 | Diseño Responsivo | Interfaz agradable a la vista y simple de utilizar para los usuarios, adaptable para distintos dispositivos, evitando así confusiones innecesarias y garantizando una experiencia óptima. |
| RNF 5 | Cifrado de datos de Almacenamiento | Los datos sensibles de la plataforma serán cifrados utilizando servicios de encriptación vigentes. |
| RNF 6 | Mensaje de error de Login/Registro | Se mostrará un mensaje de error en caso de presentarse un problema a la hora de confirmar datos del login y registro. |
| RNF 7 | Indicación visual de sesión activa | Al utilizar el servicio, se resaltan elementos como el nombre del usuario para señalar al usuario que su sesión está actualmente activa, en el caso de que se deje en segundo plano. En caso contrario, se le mostrará al usuario un mensaje para avisarle. |
| RNF 8 | | Búsquedas paginadas | | --- | | | Listados grandes (productos, citas) deben estar paginados para no afectar el rendimiento. | | --- |  |  | | --- | |

| **6. Integraciones con otros sistemas o servicios externos** | |
| --- | --- |
| APIs externas | * ***Pagos: Mercado Pago*** * ***Correo/Notificaciones:*** *SMTP corporativo o* ***SendGrid*** *(confirmaciones y recordatorios).* * ***Mapas/Ubicación:*** *Google Maps (módulo informativo).* |
| Protocolo de integración | * ***Protocolos:******REST/JSON****,* ***HTTPS****.* |
| Estrategia para manejo de fallos en integraciones | * Reintentos idempotentes en confirmación de pago, colas para emails, logs de errores y alertas. |

| **7. Aspectos de seguridad y cumplimiento normativo** | |
| --- | --- |
| Autenticación y | **AuthZ/AuthN:** JWT por API o sesiones Django + permisos DRF por rol. |

| autorización |  |
| --- | --- |
| Encriptación | * **Encriptación:** HTTPS/TLS; contraseñas con **Argon2** o **PBKDF2**. |
| Manejo de Logs y auditorías | * logging de accesos y eventos críticos (citas, pagos). |
| Cumplimiento Normativo | * **Cumplimiento normativo:** **Ley 19.628** de protección de datos; buenas prácticas ISO 27001; **PCI-DSS** aplicable a manejo de pagos (tokenización vía pasarela). |

| **8. Infraestructura de despliegue** | |
| --- | --- |
| Entorno de ejecución | **Entorno de ejecución recomendado (normal y simple):**   * **SO:** Ubuntu Server (opción Windows Server si ya disponible). * **Reverse proxy:** Nginx. * **App server:** Gunicorn/Uvicorn (Django + DRF). * **BD:** MySQL. * **Asíncrono:** Redis + Celery (recordatorios, correos, PDFs). * **Certificados:** Let’s Encrypt. |
| Despliegue | La aplicación se desplegará VPS o PaaS |
| Entornos definidos | * Desarrollo * Testing * Staging * Producción. |
| Estrategias de despliegue | ***BLUE/GREEN***  **Blue:** Release estable en producción.  **Green:** Ambiente de pruebas pre-lanzamiento a producción. |

| **9. Lineamientos de escalabilidad y mantenimiento** | |
| --- | --- |
| Estrategias para escalar horizontal o verticalmente | * **Vertical:** aumentar vCPU/RAM. * **Horizontal:** múltiples workers Gunicorn, balanceo con Nginx; cache Redis para sesiones y throttling. |
| Políticas de backup y recuperación ante desastres | * *Backups* ***diarios*** *de MySQL (retención 14–30 días).* * *RPO ≤ 24 h, RTO ≤ 4 h.* |
| Mantenimiento correctivo, evolutivo y preventivo | * **Correctivo:** hotfix con rollback. * **Evolutivo:** módulos futuros (historial clínico avanzado, recetas, stock por sucursal). * **Preventivo:** parches de seguridad mensuales, actualización de dependencias. |
| Documentación y manuales para operación futura | * API (OpenAPI/Swagger), manual de usuario, manual operativo (despliegue, backups, recuperación). |

Se solicita que la ficha técnica sea entregada en un plazo no mayor a días hábiles desde la recepción de la presente solicitud.



Firma del solicitante