**2.1.6 Entregables EDT / WBS (Estructura de Desglose del Trabajo)**

La **EDT (Estructura de Desglose del Trabajo)** o **WBS (Work Breakdown Structure)** es una representación jerárquica del alcance del proyecto, organizada en paquetes de trabajo que facilitan la gestión, asignación y seguimiento de cada componente del desarrollo. Los entregables se derivan directamente de esta estructura y permiten establecer hitos clave, dependencias y cronograma.

|  |  |  |  |
| --- | --- | --- | --- |
| Nivel | Código WBS | Entregable | Descripción |
| 1 | 1.0 | Inicio del proyecto | Documentación base para formalizar y arrancar el proyecto. |
| 2 | 1.1 | Acta constitutiva del proyecto | Documento formal que autoriza el inicio del proyecto, define objetivos y alcance. (“Plantilla del acta de constitución de proyecto (Project Charter)”) |
| 2 | 1.2 | Designación de roles y equipo | Asignación de miembros del equipo y responsabilidades. |
| 1 | 2.0 | Planeación | Actividades y documentos para guiar la ejecución técnica y administrativa. |
| 2 | 2.1 | Plan de trabajo general (Scrum/Gantt) | Calendario, sprints, dependencias y línea base. |
| 2 | 2.2 | Identificación de stakeholders | Tabla de interesados con influencia y participación. |
| 2 | 2.3 | Matriz RACI | Distribución clara de roles: Responsable, Aprobador, Consultado, Informado. |
| 1 | 3.0 | Ejecución técnica | Desarrollo del sistema y sus funcionalidades principales. |
| 2 | 3.1 | Arquitectura del sistema | Documento y diagrama técnico del diseño estructural. |
| 2 | 3.2 | Módulos funcionales (login, dashboard, reportes, filtros) | Codificación de cada módulo en back y front. |
| 2 | 3.3 | Pruebas técnicas y automatizadas | QA funcional, pruebas unitarias e integración continua. |
| 1 | 4.0 | Despliegue y documentación | Puesta en marcha del sistema y documentación de usuario. |
| 2 | 4.1 | Manual de instalación y configuración | Instrucciones técnicas para servidores físico y en la nube. |
| 2 | 4.2 | Manual de usuario | Guía operativa para usuarios finales y administrativos. |
| 2 | 4.3 | Presentación y cierre de fase | Evaluación de KPIs, demo funcional y cierre de desarrollo web. |

**2.1.6.1 Acta Constitutiva del Proyecto**

|  |  |
| --- | --- |
| Elemento | Contenido |
| Nombre del Proyecto | Plataforma Digital Integral (Web + Proyección Móvil) |
| Propósito del Proyecto | Desarrollar una solución tecnológica que automatice procesos clave, mejore la experiencia de usuario y permita futuras integraciones móviles. |
| Product Owner (PO) | Bicho |
| Scrum Máster / PM | Bicho |
| Equipo de desarrollo | Backend, Frontend, QA, UI/UX, Seguridad, Infraestructura |
| Alcance Inicial | Desarrollo completo de plataforma web operativa, con arquitectura preparada para versión móvil futura. |
| Presupuesto estimado | 5,000,000 |
| Duración estimada | 10 meses divididos en 18 sprints |
| Criterios de éxito | Cumplimiento funcional, usabilidad validada, tiempo de respuesta <2s, disponibilidad >99.9% |
| Fecha de aprobación | 1 junio del 2025 |
| Firmas de aprobación | Product Owner, Scrum Máster, Representantes legales |

**2.1.6.2 Matriz de Roles y Responsabilidades (RACI)**

La matriz RACI permite asignar claramente quién es **Responsable** (R), **Aprobador** (A), **Consultado** (C) e **Informado** (I) en cada actividad clave del proyecto:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Actividad / Entregable | Product Owner | Scrum Máster / PM | Desarrollador Backend | Frontend / UI-UX | QA / Testing | Infraestructura |
| Planificación de Sprints | C | A/R | I | I | I | I |
| Desarrollo de Módulos | I | C | R | R | C | I |
| Diseño UX/UI | I | C | I | R | C | I |
| Implementación de seguridad | I | C | R | C | C | R |
| Pruebas funcionales y automatizadas | I | C | C | C | R | I |
| Configuración de entorno y despliegue | I | C | C | I | C | R |
| Documentación técnica y manuales | I | R | C | C | C | C |
| Demo y validación de entregables | A/R | R | I | I | I | I |

**Leyenda:**

* **R**: Responsable (quien ejecuta)
* **A**: Aprobador (quien valida)
* **C**: Consultado (quien da opinión experta)
* **I**: Informado (quien debe conocer el avance)

**3. Requerimientos**

**3.1 Requerimientos funcionales**

Los requerimientos funcionales describen las funcionalidades que el sistema debe ofrecer para satisfacer las necesidades del usuario y cumplir los objetivos del proyecto. (“Requisitos funcionales - Requisitos para”)

**Principales requerimientos funcionales:**

1. **Gestión de usuarios**
   * Registro, autenticación y administración de perfiles.
   * Gestión de roles y permisos para control de acceso (administradores, usuarios finales, etc.).
   * Recuperación y restablecimiento seguro de contraseñas.
2. **Módulo de autenticación y seguridad**
   * Autenticación basada en tokens JWT.
   * Implementación de sesiones seguras con control de expiración.
   * Validación de entrada de datos para evitar inyección y ataques XSS.
3. **Gestión de contenido y datos**
   * CRUD (Crear, Leer, Actualizar, Eliminar) para entidades principales del sistema.
   * Filtros y búsquedas avanzadas con criterios múltiples (fecha, categoría, usuario).
   * Generación de reportes exportables en formatos PDF.
4. **Notificaciones y alertas**
   * Sistema de notificaciones internas con estados (vistas/no vistas).
   * Alertas visuales para eventos críticos, errores y expiraciones.
   * Configuración y personalización de notificaciones por usuario.
5. **Optimización y rendimiento**
   * Implementación de caching y lazy loading para mejorar tiempos de carga.
   * Minimización y compresión de recursos estáticos (JS, CSS).
   * Monitoreo de performance y análisis continuo para identificar cuellos de botella.
6. **Pruebas y calidad**
   * Pruebas unitarias y funcionales automatizadas integradas en pipeline CI/CD.
   * Pruebas piloto controladas con usuarios reales para validar funcionalidades.
   * Pruebas de penetración para detectar vulnerabilidades.
7. **Despliegue y mantenimiento**
   * Configuración de entorno preproducción para pruebas finales.
   * Despliegue en ambiente productivo con monitoreo activo.
   * Documentación técnica y manuales de usuario actualizados.
8. **Interfaz y experiencia de usuario**
   * Diseño responsivo y accesible, compatible con múltiples dispositivos y navegadores.
   * Microinteracciones para mejorar la navegación y feedback visual.
   * Ajustes de accesibilidad conforme a estándares WCAG 2.1. (Anexos, Seccion 2)
9. **Integración futura**
   * Preparación de infraestructura y arquitectura para la versión móvil.
   * Modularidad del código para facilitar integración y escalabilidad.

**3.2 Requerimientos no funcionales**

Los requerimientos no funcionales definen criterios y restricciones sobre el desempeño, usabilidad, confiabilidad y otros atributos cualitativos del sistema.

**Principales requerimientos no funcionales:**

1. **Rendimiento**
   * Tiempo máximo de respuesta para operaciones críticas no mayor a 2 segundos.
   * Capacidad para manejar al menos 500 usuarios concurrentes sin degradación significativa.
   * Disponibilidad del sistema mínima del 99.9% en producción.
2. **Seguridad**
   * Protección contra ataques comunes según OWASP Top 10. (Anexos, Sección 3)
   * Encriptación segura de datos sensibles tanto en tránsito (TLS) como en reposo. (Anexos, Sección 4)
   * Control de acceso granular basado en roles y permisos.
3. **Escalabilidad**
   * Arquitectura modular para facilitar la incorporación de nuevas funcionalidades.
   * Capacidad para escalar horizontalmente en servidores físicos y en la nube.
4. **Mantenibilidad**
   * Código documentado y estructurado según estándares de desarrollo.
   * Uso de herramientas de control de versiones y seguimiento de incidencias.
   * Documentación técnica clara y actualizada.
5. **Usabilidad**
   * Interfaz intuitiva y accesible para usuarios con diferentes niveles de conocimiento.
   * Soporte para navegación por teclado y compatibilidad con lectores de pantalla.
   * Feedback inmediato para acciones de usuario (errores, confirmaciones, cargas).
6. **Compatibilidad**
   * Soporte para los principales navegadores web actuales (Chrome, Firefox, Edge, Safari).
   * Adaptabilidad a dispositivos móviles y tabletas.
7. **Portabilidad**
   * Capacidad para desplegarse en diferentes entornos: desarrollo, pruebas, producción.
   * Compatibilidad con plataformas de nube (AWS, Azure, GCP) y servidores físicos. (CHECAR)
8. **Confiabilidad**
   * Mecanismos de respaldo y recuperación ante fallos.
   * Registro detallado de logs para auditoría y diagnóstico.

**3.3 Reglas de negocio**

Las reglas de negocio determinan las políticas, restricciones y procedimientos que deben cumplirse en el funcionamiento del sistema para garantizar coherencia y alineación con objetivos empresariales o institucionales.

**Ejemplos de reglas de negocio:**

1. **Gestión de usuarios**
   * Solo los usuarios con rol administrador pueden modificar roles y permisos.
   * El bloqueo de usuarios se activa tras 5 intentos fallidos consecutivos de inicio de sesión.
2. **Validaciones de datos**
   * Los campos de fecha deben respetar la relación lógica: fecha inicio ≤ fecha fin.
   * Los reportes no podrán incluir datos de usuarios inactivos por más de 6 meses.
3. **Notificaciones**
   * Las notificaciones críticas deben mostrarse inmediatamente y repetirse hasta que sean leídas.
   * Los usuarios pueden configurar qué tipos de alertas desean recibir por correo electrónico.
4. **Seguridad**
   * Todas las contraseñas deben cumplir con una longitud mínima de 10 caracteres y complejidad (mayúsculas, minúsculas, números y símbolos).
   * Las sesiones expiran tras 30 minutos de inactividad.
5. **Acceso a datos**
   * El acceso a información sensible está restringido según el nivel de permiso asignado.
   * Los datos de auditoría deben mantenerse almacenados al menos 2 años.
6. **Despliegue**
   * Los despliegues en producción solo se realizarán tras aprobación del Product Owner.
   * Cada despliegue debe ir acompañado de un plan de rollback para restaurar el sistema en caso de fallo.

**3.4 Requisitos de usuario**

Los requisitos de usuario reflejan las expectativas, necesidades y preferencias de los usuarios finales para la correcta interacción con el sistema.

**Principales requisitos de usuario:**

1. **Facilidad de uso**
   * La interfaz debe ser sencilla, con un diseño limpio y organizado.
   * Las operaciones básicas deben realizarse con un mínimo número de pasos.
2. **Accesibilidad**
   * El sistema debe ser usable por personas con discapacidades visuales, auditivas o motoras.
   * Soporte para navegación mediante teclado y lectores de pantalla.
3. **Disponibilidad**
   * Los usuarios deben poder acceder al sistema en cualquier momento, salvo en mantenimientos programados.
4. **Personalización**
   * Posibilidad de configurar notificaciones, temas visuales y preferencias personales.
5. **Ayuda y soporte**
   * Disponibilidad de manuales, tutoriales y FAQs integrados.
   * Canales de soporte accesibles (chat, correo, formularios).
6. **Feedback y transparencia**
   * El sistema debe proporcionar retroalimentación clara tras cada acción (éxito, error, proceso en curso).
   * Visualización clara de estados, errores y alertas.
7. **Seguridad y privacidad**
   * Confianza en que sus datos personales están protegidos y no se compartirán sin consentimiento.
   * Control sobre la visibilidad de su información y actividad dentro del sistema.

**Consideraciones para elegir la arquitectura:**

| **Criterio** | **Detalle relevante para el proyecto** |
| --- | --- |
| **Carga de procesamiento** | Tienes un servidor físico que realizará el procesamiento fuerte. |
| **Interacción de usuarios** | Servidor en la nube para mostrar datos, acceso simultáneo desde web. |
| **Escalabilidad** | Se debe poder crecer en usuarios y funcionalidades sin impacto fuerte. |
| **Seguridad** | Autenticación, roles, protección OWASP, encriptación. |
| **Mantenibilidad** | Código modular, con facilidad para futuras actualizaciones o versiones móviles. |
| **Disponibilidad** | Sistema con alta disponibilidad y monitoreo. |
| **Tecnologías actuales** | Uso de frameworks modernos, integración con CI/CD, pruebas automatizadas. |

**Opciones comunes y evaluación**

|  |  |  |  |
| --- | --- | --- | --- |
| Arquitectura | Descripción breve | Ventajas relevantes | Desventajas / Consideraciones |
| Monolítica tradicional | Todo en un solo servidor y aplicación integrada. | Fácil de desarrollar al inicio. | Poco escalable, difícil mantenimiento y despliegue. |
| Cliente-Servidor (2 capas) | Servidor para lógica y base de datos, cliente para UI. | Simple, divide responsabilidades básicas. | Escalabilidad limitada para muchos usuarios. |
| 3 capas (Presentación, Lógica, Datos) | Separación clara en UI, lógica y base de datos. | Mejor organización, fácil mantenimiento. | Requiere infraestructura y coordinación. |
| Microservicios | Módulos independientes desplegados y escalables por separado. | Escalabilidad, resiliencia, despliegue ágil. | Mayor complejidad, requiere orquestación y monitoreo. |
| Arquitectura basada en la nube híbrida | Procesamiento pesado en servidor físico local + interfaz en la nube. | Optimiza recursos y balancea cargas. | Necesita sincronización y gestión de latencia. |

**Arquitectura para el Proyecto**

**Arquitectura híbrida de 3 capas con servidor físico para procesamiento y nube para presentación:**

* **Capa de presentación:** Aplicación web alojada en la nube, accesible desde cualquier dispositivo, gestionando la interfaz de usuario, validaciones básicas y consultas livianas.
* **Capa de lógica / procesamiento:** Servidor físico local encargado de la lógica compleja, cálculos, procesamiento de datos, seguridad y acceso a la base de datos.
* **Capa de datos:** Base de datos centralizada en el servidor físico, con backups regulares y replicación si se requiere.
* **Monitoreo y logging:** Integrados con herramientas que corren en la nube para supervisar el estado del servidor físico y la aplicación web.
* **CI/CD y pruebas:** Integradas para permitir despliegues ágiles y control de calidad automatizado.

**Justificación**

* Aprovecha la potencia del servidor físico para procesos intensivos.
* Utiliza la nube para garantizar alta disponibilidad y acceso rápido a la UI.
* Facilita escalabilidad gradual, manteniendo el control sobre datos sensibles localmente.
* Mejora seguridad al segmentar funciones y limitar accesos.
* Es compatible con las mejores prácticas modernas (modularidad, pruebas, despliegue continuo).
* Permite evolución hacia microservicios o versión móvil en el futuro.

**3.5 Fases de la Metodología Design Thinking**

La **metodología Design Thinking** es un enfoque centrado en el usuario que permite resolver problemas complejos mediante la comprensión profunda de las necesidades reales, la experimentación iterativa y el diseño colaborativo de soluciones. Su estructura se basa en cinco fases secuenciales pero flexibles: **Empatizar, Definir, Idear, Prototipar y Evaluar**.

Esta metodología resulta especialmente valiosa en proyectos tecnológicos e innovadores, ya que facilita la **alineación entre los requerimientos técnicos y humanos**, asegurando que el producto o sistema desarrollado **responda a necesidades auténticas y verificadas** por los usuarios.

**3.5.1 Comprensión y aplicación del paso “Empatizar”**

**Descripción general:**

La fase de **Empatizar** tiene como objetivo principal **conocer al usuario en profundidad**, no solo desde el punto de vista funcional, sino también emocional, conductual y aspiracional. Implica una inmersión profunda en su realidad para identificar **deseos, frustraciones, motivaciones y contextos de uso**.

Para lograrlo, se utilizan técnicas cualitativas como entrevistas, observación directa, encuestas, sesiones de co-creación y pruebas preliminares.

**Herramientas aplicadas:**

A continuación, se detallan las herramientas utilizadas durante la aplicación de esta fase en el contexto del sistema en desarrollo:

**Mapa de necesidades**

|  |  |
| --- | --- |
| Elemento | Descripción |
| ¿Qué es? | Una herramienta que organiza las necesidades explícitas e implícitas de los usuarios detectadas en entrevistas, encuestas o pruebas de campo. |
| ¿Cómo se aplicó? | Se elaboró a partir de encuestas a usuarios finales, entrevistas con profesionales del sector y análisis de plataformas similares. |
| Resultado | Identificación de necesidades clave como: agilidad de acceso, centralización de datos, notificaciones personalizadas y facilidad de navegación. |

**Mapa de empatía**

|  |  |
| --- | --- |
| Dimensión | Aplicación específica |
| ¿Qué dice? | “Necesito un sistema que no sea complicado y que me diga cuándo actuar.” |
| ¿Qué piensa? | “Ojalá no tenga que depender de soporte técnico cada vez que ingreso.” |
| ¿Qué siente? | Frustración por sistemas anteriores lentos o con errores frecuentes. |
| ¿Qué hace? | Usa plataformas limitadas o poco intuitivas. |
| ¿Qué ve? | Comparaciones con otros sistemas, referencias por otros usuarios o empresas. |
| ¿Qué escucha? | Opiniones de colegas o experiencias previas negativas. |

**Buyer Persona**

|  |  |
| --- | --- |
| Perfil | Descripción |
| Nombre ficticio: | Laura Rodríguez |
| Edad: | 34 años |
| Ocupación: | Responsable de gestión administrativa en PyME |
| Tecnología: | Usuario medio, maneja Excel, correo, CRM básicos |
| Necesidades clave: | Facilidad de acceso, claridad en notificaciones, reportes exportables |
| Puntos de dolor: | Tiempos de carga largos, sistemas confusos, soporte lento |

**Test de usuario preliminar**

|  |  |  |
| --- | --- | --- |
| Fase del prototipo | Interacción observada | Feedback recibido |
| Primer wireframe | Dificultades para encontrar botones clave, flujo confuso | “No sé a dónde debo ir para generar un reporte.” |
| Prototipo funcional simple | Mejora en navegación, pero dificultad en filtros avanzados | “¿Y si quiero filtrar por fecha exacta y por categoría a la vez?” |
| Versión con mejoras | Fluidez de navegación, comprensión de jerarquías, claridad en resultados visuales | “Ahora se siente más intuitivo, parece que entiende cómo trabajo.” |

El desarrollo de esta fase permitió construir una **visión empática, validada y centrada en el usuario final**, lo que asegura una solución **funcionalmente relevante y emocionalmente aceptable**. Se definieron claramente las necesidades principales, los perfiles prioritarios y las condiciones del entorno en el que operará el sistema.

Esta comprensión profunda **fundamentó todo el ciclo de desarrollo posterior** y reforzó la aplicación de principios de accesibilidad, seguridad, claridad y eficiencia en el diseño y arquitectura general del sistema.

**3.5.3 Comprensión del paso de “Idear”**

La etapa de “**Idear**” dentro del proceso de diseño centrado en el usuario representa el momento en que, con base en los hallazgos obtenidos durante las fases de *Empatizar* y *Definir*, se generan soluciones creativas y viables a los problemas detectados. Esta fase requiere un enfoque divergente, donde se prioriza la cantidad y variedad de ideas, antes de converger hacia aquellas más factibles, innovadoras y alineadas con las necesidades reales de los usuarios.

**En el contexto del proyecto:**

Durante esta fase, el equipo interdisciplinario involucrado en el desarrollo de la plataforma web para el cuidado de mascotas participó en sesiones estructuradas de lluvia de ideas, análisis comparativo de plataformas similares (benchmarking) y ejercicios de co-creación con usuarios clave (dueños de mascotas, veterinarios, y personal de tiendas especializadas).

Se exploraron distintos enfoques para resolver los retos más críticos identificados en las fases anteriores, tales como:

* La dificultad de encontrar profesionales veterinarios certificados cercanos.
* El seguimiento desorganizado del historial clínico de las mascotas.
* La falta de integración entre servicios de comercio electrónico y atención médica veterinaria.
* El bajo nivel de accesibilidad en plataformas existentes.
* La fragmentación de servicios en aplicaciones distintas.

**Metodologías empleadas:**

* **Brainstorming guiado**, orientado a generar ideas sin restricciones.
* **Mapa de Afinidad**, para clasificar las ideas según funcionalidad, valor al usuario y viabilidad técnica.
* **Técnica SCAMPER**, que permitió reformular soluciones existentes y adaptarlas al contexto del sistema.
* **Matriz de Impacto/Esfuerzo**, para priorizar las funcionalidades y características a implementar.

**Resultados clave:**

La fase de ideación permitió delimitar un conjunto de funcionalidades innovadoras y técnicamente alcanzables para la primera versión del sistema, incluyendo:

* Un **buscador filtrado de clínicas y servicios veterinarios** según localización y especialidad.
* Un **módulo de historial clínico digital compartida** entre usuario y profesional.
* Un **carrito de compras inteligente** que recomienda productos según el perfil de la mascota.
* La futura **integración de geolocalización, escaneo QR** y **notificaciones personalizadas** en la aplicación móvil.