(Pets Heaven)

**Julian Velez Quitian Kevin Palacios Arce Cristian Arrieta Romero Juana Violeta Espejo**

**CENTRO INDUSTRIAL Y DESARROLLO EMPRESARIAL – CIDE ANÁLISIS Y DESARROLLO DE SOFTWARE - ADSO**

**SOACHA, CUNDINAMARCA 2025**

****

1. **Introducción**
2. **Objetivos del Proyecto**
   1. **Objetivo general**
   2. **Objetivos específicos**
3. **Alcance del proyecto**
4. **Personal Involucrado**
   1. **Diagramas de Casos de Uso**
5. **Características de los usuarios**
6. **Requerimientos**
   1. **Requerimientos funcionales**
   2. **Requerimientos no funcionales**
7. **Arquitectura del proyecto**
   1. **Definición metodología**
   2. **Ciclo de vida**

**7.3 Diagrama de despliegue**

1. **Fundamentos y herramientas utilizadas**
2. **Requisitos del Sistema**
3. **Procesos del Software**
4. **Instalación de Aplicaciones**
5. **Diagrama de Clases**
6. **Modelo entidad Relación**
7. **Diagrama entidad relación**
8. **Diccionario de Datos**
9. **Glosario**
10. **Referencias**

****

**Control de versiones**

|  |  |  |  |
| --- | --- | --- | --- |
| **FECHA** | **VERSIÓN** | **DESCRIPCION** | **AUTOR** |
| 20/06/2025 | 1.0 | Primera versión del manual técnico del sistema de gestión veterinaria Pets Heaven. | Julian Velez Quitian Estudiante de Análisis y desarrollo de software, Servicio nacional de aprendizaje.  **Tel**: 3134853945  **Correo:**  [julivelez337@gmail.com](mailto:julivelez337@gmail.com) |

**Tabla 1. Control de versiones del manual técnico 1. introducción**

El presente documento servirá para identificar las técnicas usadas para el desarrollo de este aplicativo web, que se dará con el fin de explicar y facilitar el entendimiento de los procesos tales como, registrar usuarios, administradores, veterinarios y mascotas, así mismo el sistema de roles, agendamiento e historiales médicos con los que el sistema cuenta, dando explicaciones claras para que el personal que tenga contacto con él pueda darle el uso correspondiente y eficiente.

Así mismo tener en cuenta que esto no debe ser tomado como una guía de aprendizaje sino como una guía para documentar las diferentes técnicas usadas para la creación de este proyecto, adicional leer detenidamente los pasos descritos para la debida instalación del sistema el cual debe tener en cuenta que el usuario contara con conocimientos básicos en el uso de equipos tales como computadores y sus periféricos.

Además, se debe tener en cuenta la secuencia de usos o implementaciones de herramientas tecnológicas (React, Emailjs, Supabase, Express), Dependencias (librerías de react, boxicons, supabase, emailjs, sweetalert, framer-motion, fullcalendar, axios, compresorjs en el front end y cors, express, mysql, express-rate-limit, jsonwebtoken, bcrypt, dotenv,

cookie-parser en el back end.), Apis (BackEnd) consumibles para poder generar los llamados específicos.

1. **Objetivos del proyecto**
   1. **Objetivo general**

Desarrollar un sistema web de gestión veterinaria que automatice los procesos administrativos y clínicos, con el fin de incrementar la productividad de la clínica y la satisfacción del cliente.

* 1. **Objetivos específicos**
     + **Fase de análisis**
       - Realizar el levantamiento de los requerimientos del software.
       - Establecer requisitos funcionales y no funcionales.
       - Identificación de riesgos.
       - Definición de arquitectura.
     + **Fase de diseño**
       - Diseño de la arquitectura.
       - Diagramas UML.
       - Definición de tecnologías.

**-Fase de implementación y despliegue**

* + - * Codificación según el diseño.
      * Uso de control de versiones.
      * Configuración de servidores.

1. **Alcance del proyecto**

El proyecto está destinado para servir a veterinarias de todo Soacha con más de 500 clientes mensuales, el sistema ofrecerá servicios en tiempo real para el agendamiento de

citas, usuarios, etc. además de incorporar roles para identificar a cada usuario dentro del sistema.

1. **Personal involucrado**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nombre** | **Rol** | **Categoría profesional**  **l** | **Responsabilidad** | **Contacto** | **Apro v** |
| Julian velez | Encargado de manuales y documenta ción. | Tecnólogo en análisis y desarrollo de  software | Redacción de documentación pertinente | **Tel**: 3134853945  **Correo:**  [julivelez337@gmail.com](mailto:julivelez337@gmail.com) | si |
| Kevin Palacios | Arquitecto de software. | Tecnólogo en análisis y desarrollo de software | Responsable de la accesibilidad del programa | **Tel**: 3114675099  **Correo:**  [kevin\_spalacios@gmail.com](mailto:kevin_spalacios@gmail.com) | si |
| Cristian Arrieta | Analista de datos, Desarrollad or full stack. | Tecnólogo en análisis y desarrollo de software | Responsable de la seguridad, y la infraestructura del software. | **Tel**: 3205554483  **Correo:**  cristianarrietar04@gmail  .com | si |
| Juana Violeta | Desarrollad or FrontEnd. | Tecnólogo en análisis y desarrollo de software | Responsable del diseño web. | **Tel**: 3224523961  **Correo:**  juanavioleta1325@gmail  .com | si |

**Tabla 2. Encargado principal del desarrollo del aplicativo**

**5. Características de los usuarios**

**Tabla 3. Características de Cliente**

|  |  |
| --- | --- |
| **TIPO DE USUARIO** | Cliente |
| **FORMACIÓN** | Persona natural con conocimientos básicos en el uso de  aplicación web, móviles o de escritorio. |
| **HABILDADES** | Conocimiento en el manejo de  aplicaciones web |
| **ACTIVIDADES** | -Navegar en el sitio web  -Registrarse/logearse  -Reestablecer contraseña  -Ver historial de su mascota  -Agendar, reprogramar, cancelar citas médicas en agenda personal  -Actualizar foto de perfil de su mascota  -Actualizar información de perfil propio |

**Tabla 3. Características de Administrador**

|  |  |
| --- | --- |
| **TIPO DE USUARIO** | Administrador |
| **FORMACIÓN** | Formación técnica para desarrollar las funciones de  administración del sistema de información. |
| **HABILDADES** | -Conocimiento extenso en el manejo de aplicaciones  web, móviles o de escritorio.   * Capacidad de resolución rápida de problemas. * Capacidad de comunicación oral y escrita. * Trabajo en equipo. |
| **ACTIVIDADES** | -Navegar en el sitio web  -Logearse  -Reestablecer contraseña  -Ver historial de mascotas  -Agendar, reprogramar, cancelar citas médicas en agenda general y personal  -CRUD de usuarios |

****

|  |  |
| --- | --- |
|  | -CRUD de perfil propio  -Asignar roles |

**Tabla 3. Características de usuario Veterinario**

|  |  |
| --- | --- |
| **TIPO DE USUARIO** | Personal |
| **FORMACIÓN** | Estudios centrados en ciencias y matemáticas, obtener una licenciatura, en ciencias biológicas, y completar una  escuela de veterinaria. |
| **HABILDADES** | -Aplicar conocimientos  -Afecto por los animales  -Conocimiento en el manejo de aplicaciones  web, móviles o de escritorio.   * Capacidad de resolución rápida de problemas. * Capacidad de comunicación oral y escrita. * Trabajo en equipo.   -Capacidad de adaptación |
| **ACTIVIDADES** | -Navegar en el sitio web  -Logearse  -Reestablecer contraseña  -Ver historial de mascotas  -Agendar, reprogramar, cancelar citas médicas en agenda personal  -CRUD de perfil propio |

1. **Requerimientos**
   1. **Requerimientos funcionales**
      * **Gestión de Usuarios:** El sistema debe permitir el registro y login de usuarios (clientes, veterinarios y administradores)
      * **Gestión de Mascotas:** El sistema debe permitir registrar nuevas mascotas con datos como nombre, especie, raza, edad, propietario, etc.

**-Agendar citas:** El usuario debe poder agendar citas seleccionando fecha, hora, tipo de consulta y veterinario disponible.

El usuario debe poder agendar citas seleccionando fecha, hora, tipo de consulta y veterinario disponible.

* + - **Historial Médico de Mascotas:** El veterinario debe poder registrar diagnósticos, tratamientos y vacunas aplicadas.
    - **Búsqueda de Mascotas o Citas:** El sistema debe permitir buscar mascotas por nombre o filtrar citas por fecha o veterinario.
    - **Gestión de Servicios:** El sistema debe mostrar servicios para el cuidado de mascotas.
    - **Panel de Administración:** El administrador debe poder gestionar usuarios, veterinarios, servicios y revisar reportes.
  1. **Requerimientos no funcionales**

**-Usabilidad:** La interfaz debe ser intuitiva y fácil de usar para personas sin conocimientos técnicos.

**-Rendimiento:** El sistema debe responder a las acciones del usuario en menos de 2 segundos en condiciones normales.

**-Seguridad:** Las contraseñas deben almacenarse cifradas. Los usuarios deben tener roles para limitar accesos, Generar copias de seguridad cada semana.

**-Escalabilidad:** El sistema debe poder adaptarse para manejar un número creciente de mascotas, usuarios y citas.

**-Disponibilidad:** El sistema debe estar disponible el 99% del tiempo, salvo en mantenimiento programado.

**-Compatibilidad:** El sistema debe ser accesible desde navegadores y dispositivos móviles modernos.

**-Mantenibilidad:** El código debe estar documentado para facilitar futuras actualizaciones o correcciones.

**-Accesibilidad:** La página debe ser accesible a personas con discapacidades visuales, dando descripciones a objetos y facilitando la movilidad dentro de la página.

1. **Arquitectura del proyecto**
   1. **Definición metodología**

La metodología que hemos adaptado en este proyecto es SCRUM ya que desde el inicio hemos usado historias épicas, y la competitividad entre desarrolladores así mejorando la efectividad y velocidad del desarrollo.

* 1. **Ciclo de vida**

El ciclo de vida que manejamos es:

**-Análisis**: En esta fase se identifican las necesidades del cliente o del proyecto. Se recopila y documenta la información necesaria para entender qué Problema se debe resolver. Aquí se definen los requisitos funcionales y no funcionales, y se establecen las bases para el desarrollo del sistema.

**-Diseño:** Con base en los requisitos definidos en el análisis, se planifica cómo se va a construir el sistema. Esto incluye la arquitectura del software, estructura de bases de datos, diseño de interfaces y flujo de la aplicación.

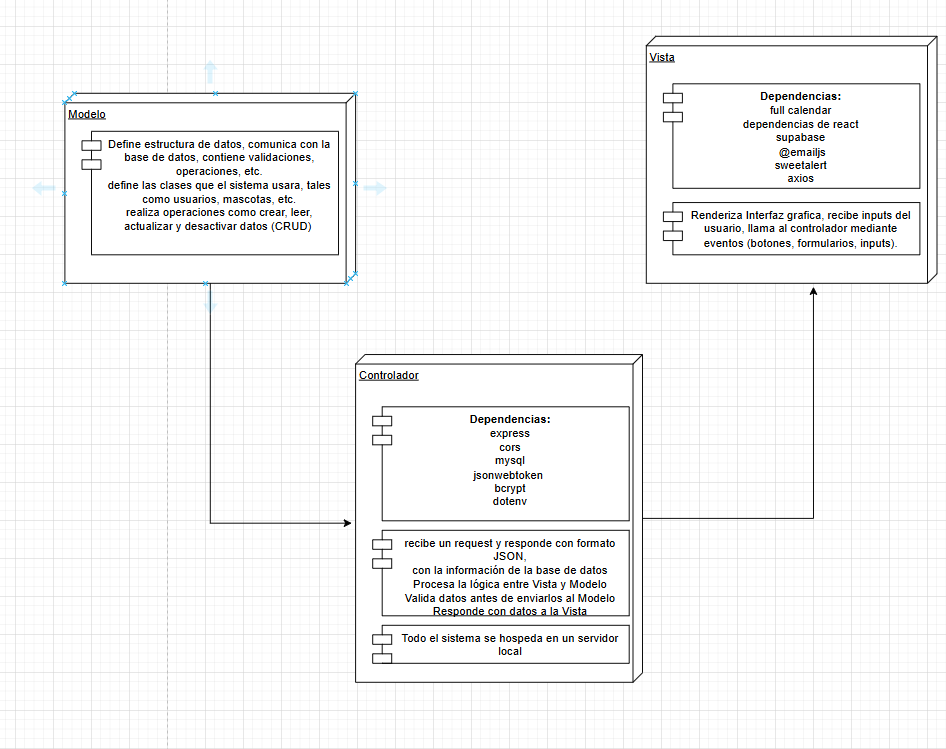
**-Codificación:** Es la fase donde se lleva a cabo la implementación del diseño usando en este caso Node, Express, React y mysql, css 3, HTML, Javasript. Se crean los módulos, componentes y funcionalidades del sistema según lo planificado.

**-Despliegue:** Una vez el sistema ha sido desarrollado y probado, se pone en marcha en el entorno real (producción). Aquí se aseguran aspectos como la configuración del servidor, base de datos y accesos del usuario.

**-Mantenimiento:** Después del despliegue, el sistema puede requerir mejoras, corrección de errores o adaptaciones a nuevos requerimientos



**7.3 Diagrama de despliegue**

****

Este sistema está basado en la arquitectura MODELO-VISTA-CONTROLADOR (MVC), separando responsabilidades para facilitar el mantenimiento y escalabilidad del proyecto.

**-MODELO (base de datos):** Esta sección es la encarada de definir la estructura de datos. Se comunica

con la base de datos así mismo también define las clases que el sistema usara, y representa la lógica del negocio y la forma en la que debe comportarse.

**-VISTA (front-end):** Esta sección se encarga de mostrar la interfaz grafica al usuario, recibiendo sus inputs

atreves de botones y formularios usando dependencias como full calendar, supabase, emailjs, etc con

el fin de que la interfaz sea lo más interactiva y entendible posible.

**-CONTROLADOR (back-end):** Esta sección se encarga de recibir request y responderla por medio de un

archivo de formato JSON con la información de la base de datos, procesa información entre el modelo

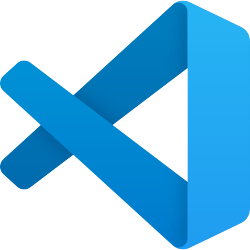
y la vista validando Dichos datos y dando respuesta en la vista.

**8.Fundamentos y herramientas utilizadas**

A continuación, se mostrará una lista de todos las herramientas usadas y necesarias para el despliegue

y el debido funcionamiento del sistema.

**-VISUAL STUDIO CODE:**



Visual studio code es un editor de código fuente ligero y a la vez eficaz, el cual fue usado para realizar

enteramente el código del sistema, con ayuda de extensiones que el mismo programa da acceso.

**-XAMPP:**

Xampp es una distribución de Apache, el cual se usó como servidor local para la base de datos.

**-SUPABASE:**

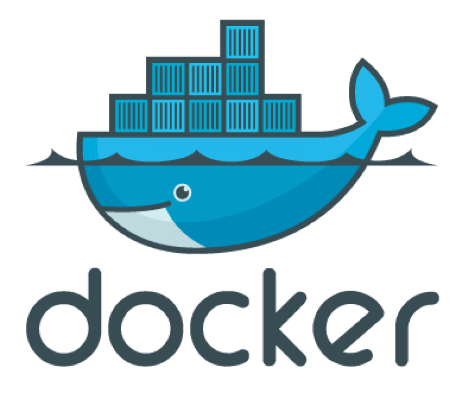


Supabase es una plataforma de desarrollo de postgres, proporcionando alojamiento de datos, usamos esta

plataforma para almacenar las imágenes que se suben por medio de los formularios y solo retornando la url

que se almacena en la base de datos principal, usando para este fin el sistema de buckets (contenedor para

archivos) implementado en supabase.

**-DOCKER:**

Docker es una plataforma de software que permite a desarrolladores crear, probar e implementar aplicaciones

de forma rápida, lo usamos con el fin de que nuestro sistema sea portable y mulltiplataforma.

**-EXPRESS:**



Express es el framework web más popular de Node, y es a librería subyacente para un gran número de otros

frameworks proporcionando mecanismo para la escritura de manejadores de peticiones con diferentes

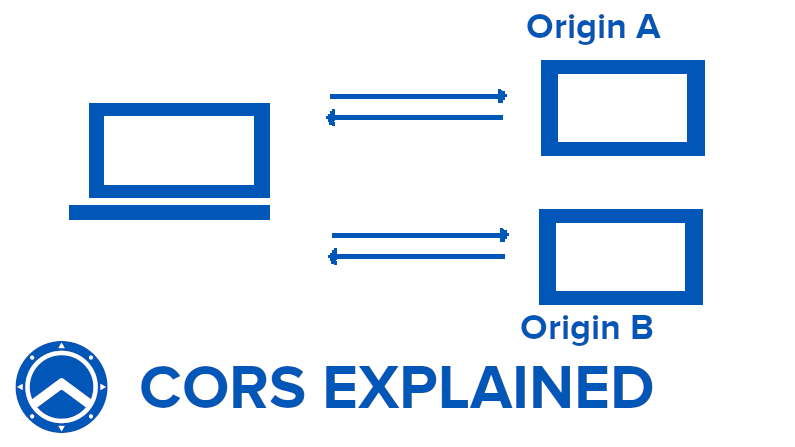
verbos (GET, POST, PUT, DELETE) HTTP.

**-NODE.JS:**



Node.js es un entorno de ejecución JavaScript de código abierto y multiplataforma que se usa para desarrollar aplicaciones escalables de lado del servidor y la red.

**-CORS:**



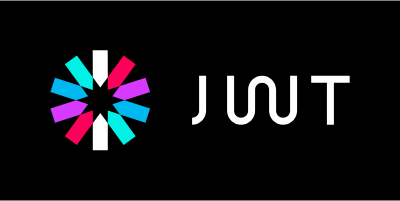
Cors (Intercambio de recursos de origen cruzado) es un mecanismo basado en cabeceras HTTP que permite

a un servidor indicar cualquier dominio, esquema o puerto con un origen distinto al suyo.

**-MYSQL:**

MYSQL es un sistema de administración relacional de base de datos de código abierto que se usa para almacenar

y gestionar datos usado por su fiabilidad, rendimiento, escalabilidad y facilidad de uso.

**-JSON WEB TOKEN:**

JWT (JSON Web Token) es un mecanismo para poder propagar entre dos partes y de forma segura la identidad

de un determinado usuario.

**-BCRYPT:**

Bcrypt es un algoritmo de hashing de contraseñas diseñado para protegerlas frente a ataques maliciosos, siendo

diseñado intencionalmente lento y seguro.

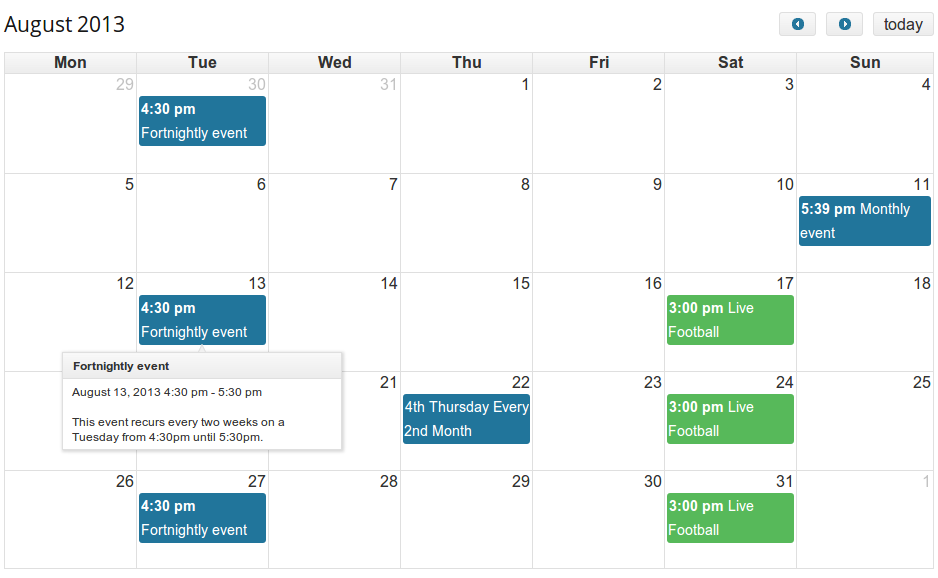
**-DOTENV:**



Dotevn es un módulo sin dependencias, disponible atreves de paquetes npm, que carga variables de entorno

desde un archivo .env en nuestro caso.

**-FULL CALENDAR:**



Full calendar es una librería que trae por defecto una vista de un calendario con sus funcionalidades permitiendo

representar datos procedentes de la base de datos para que sea en tiempo real y muy fácil y útil para todo tipo

de usuario.

**-DEPENDECIAS DE REACT:**

Las dependencias de React son elementos fundamentales que permiten que una aplicación se ejecute de

manera fluida y sin problemas, entre las usadas tenemos:

**react-dom**  
Librería que permite renderizar componentes de React en el DOM del navegador. Es el puente entre

React y el HTML real.

**react-hook-form**  
Librería para manejar formularios en React de forma simple y eficiente. Permite validaciones, manejo

de errores e integración con inputs sin escribir tanto código.

**react-router**  
Es una librería de enrutamiento para React que permite crear navegación entre diferentes páginas o componentes sin recargar la página. Define rutas (<Route>) y enlaces (<Link>) dentro de una aplicación

SPA (Single Page Application).

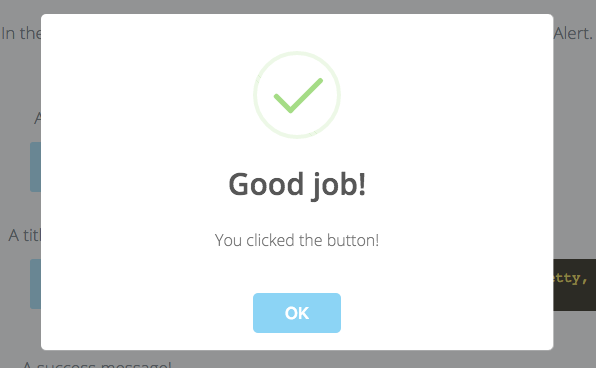
**-EMAILJS:**

Emailjs es una biblioteca ligera que ayuda a las aplicaciones a enviar correos electrónicos atreves de un servidor

SMTP, en este caso usado para enviar códigos de verificación de usuarios para el cambio de contraseña, y el

en la página.

**-SWEETALERT:**



Sweetalert es un complemento atractivo y responsivo para reemplazar las ventanas emergentes de JavaScript

usado para generar alertas más llamativas y personalizadas.

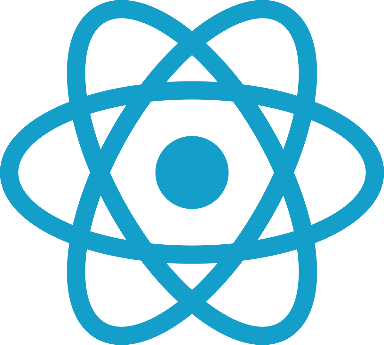
**- AXIOS:**



Axios es un cliente HTTP basado en promesas para node. js y el navegador. Es isomorfo

(puede ejecutarse en el navegador y en Node.

**-REACT JS:**



React es una biblioteca de JavaScript desarrollada para crear interfaces de usuario interactivas y modernas

Especialmente para aplicaciones web basada en componentes.

**-CSS:**



CSS es un tipo de lenguaje que permite definir y crear la presentación visual de los

documentos que ya se encuentran estructurados y escritos dentro del sistema de

información en lenguaje HTML. En conclusión, CSS permite generar el diseño visual de las

páginas web e interfaces de usuario.

**9.Requisitos del Sistema**

**9.1 Lado del servidor:** Software necesario (node.js, npm o yarm, base de datos MySql) servidor web (puede ser

local), sistema operativo compatible (Windows 10/11), Linux, macOS.

**9.2 Lado del cliente:** Navegador moderno (Chrome, FireFox, Edge, Safari, Google, Opera), JavaScript habilitado

(navegadores lo tienen activado de por sí), conexión a internet estable, resolución de pantalla adecuada y un

mínimo de 4 de RAM

**NOTA: No necesita instalar nada ya que es un servicio web.**

**10.Procesos del Software**

**10.1 Procesos de entrada:**

-Ingresa a la web (usuarios).

-Ingresa datos de registro o logeo (Administrador, personal y cliente).

-Ingresar datos para el registro de mascotas (Registros básicos en el sistema.).

-Ingresar datos para el registro de clientes (Registros básicos en el sistema.

-Ingresar datos para el registro de personal (Registros básicos en el sistema.).

-Ingresar datos para el agendamiento de una cita médica (Registros básicos en el sistema.).

-Ingresar datos para el ingreso de vacunas (Registros básicos en el sistema.).

-Ingresar datos para el ingreso de servicios (Registros básicos en el sistema.).

-Generar resultados de consulta (Factura, historial médico).

**10.2 Procesos de salida:**

- Consulta de usuario (Usuarios).

- Consulta de personal (Personal).

- Consulta de categorías, servicios y categorías (Servicios, facturas).

- Consulta de citas (Citas).

- Consulta de facturas (Facturas).

- Consulta de vacunas (Vacunas).

- Descarga de facturas (Facturas).

- Descarga de historial médico (Historial médico.).

1. **Instalación de Aplicaciones**