**UNIVERSIDAD PRIVADA CUMBRE**

**FACULTAD DE CIENCIAS Y TECNOLOGÍA**

**INGENIERIA DE SISTEMAS**

****

**SISTEMA DE INFORMACIÓN PARA LA GESTIÓN DE VENTA DE SERVICIOS DEL CENTRO DE BIENESTAR “FABIOLA SPA” EN LA CIUDAD DE SANTA CRUZ DE LA SIERRA**

**JOSE ARIEL FLORES MAMANI**

**PROYECTO DE GRADO**

**PRESENTADO COMO REQUISITO PARCIAL**

**PARA OBTENER EL GRADO DE LICENCIATURA EN**

**INGENIERIA DE SISTEMAS**

**Santa Cruz de la Sierra – Bolivia**

**2023**

**Dedicatoria.** Dedicada a Dios Padre por brindarme las fuerzas necesarias y las oportunidades en mi vida .

*Con todo el amor, respeto y admiración dedico el presente proyecto a mis padres: Ovidio Flores Canaviri, Beatriz Mamani Huayta y a mis hermanos Mariela Flores Mamani y Rene Eduardo Flores Mamani   mi esposa e hija Carla Adahi Elena Ruiz, Ariely Jazheel Flores Elena Quienes me dieron grandes y valiosas enseñanzas, muchas gracias por la confianza, apoyo, sacrificio y amor.*

*Por su cariño, consejos, palabras de aliento y apoyo incondicional, durante todo este proceso, por estar conmigo en todo momento gracias hicieron de mí una mejor persona y de una u otra forma me acompañan en todos mis sueños y metas*

**Agradecimiento.**Dirigida a mi Padre Celestial gracias a su, fortaleza, sabiduría… y guiarme por el buen camino.

*A mis padres por la vida que me dieron con grandes enseñanzas y consejos con todo el cariño, esfuerzo, sacrificio, sencillez, compresión, disciplina y amor, gracias mamá y papá.*

*A mi familia, por haberme dado la oportunidad de formarme en esta prestigiosa universidad y haber sido mi apoyo durante todo este tiempo.*

*A la Universidad Privada Cumbre por forjar profesionales con principios, valores y ética profesional.*

*Especialmente al director de Carrera Ing. Amílcar Cruz Herbas, quien nos brindó el apoyo académico con responsabilidad y compromiso, dirigiéndonos a una superación constante.*

*A los docentes por su dedicación y tiempo en cada clase.*

*A la Empresa Fabiola SPA por permitirme el desarrollo de mi trabajo de investigación y brindarme las facilidades en cuanto a documentación requerida.*

**RESUMEN**

|  |  |
| --- | --- |
| **TITULO** | Sistema de Información para la Gestión de Ventas de Servicios del Centro de Bienestar "Fabiola Spa" en la Ciudad de Santa Cruz de la Sierra. |
| **AREA DE INVESTIGACION** | Desarrollo de software, análisis de arquitectura de software. |
| **TIPO DE ESTUDIO** | Descriptivo Explicativo. |
| **ELEMENTOS DE CONTENIDO** | La investigación propuesta busca, mediante la aplicación de la teoría y los conceptos básicos del Proceso Unificado de Desarrollo de Software implementar las nuevas tecnologías el proceso de gestión de datos para obtener la información requerida por la empresa, encontrar explicaciones a situaciones internas (información de clientes, información de personal, etc.) y del entorno (información de ventas de servicios, gestión de gastos, etc.) que afectan a la empresa. Esto permitirá al personal de administración de la empresa contrastar diferentes conceptos de la administración en una realidad concreta. |
| **CONTENIDO** | |
| **ANTECEDENTES** | La prestigiosa empresa que brinda servicios para la salud y bienestar de la población “Fabiola Spa" nace a través de un emprendimiento unipersonal que brindaba servicios en la calle en el año 2008 a fueras de una Universidad. Con el pasar de los años decide alquilar un espacio para implementar más servicios y también de personal, lamentablemente quebró por una falta de administración y vuelve nuevamente con el servicio en la calle. |
| **PLANTEAMIENTO DEL**  **PROBLEMA** | Los problemas dentro de la empresa se deben a que el registro de cada servicio prestado a los clientes, así como toda información en la empresa se realiza de forma manual. Por lo tanto, se debe automatizar los procesos de administración de información. |
| **OBJETIVO GENERAL** | Desarrollar un sistema de gestión y administración utilizando la metodología de Proceso Unificado de Desarrollo de Software, específicamente diseñado para cubrir las necesidades y operaciones de "Fabiola Spa", ubicado en la ciudad de Santa Cruz de la Sierra. |
| **JUSTIFICACIÓN** | Con la herramienta de gestión de ventas y gestión de gastos permiten encontrar soluciones concretas a problemas de gestión de automatización del procesamiento de datos, obtención de información precisa en el menor tiempo posible, control preciso del personal y clientes de la empresa, que inciden en los resultados de la empresa. |
| **CONTENIDO** | Resumen del proyecto. |

INDICE GENERAL

[1. INTRODUCCIÓN 1](#_Toc174279468)

[1.1. ANTECEDENTES 1](#_Toc174279469)

[1.1.1. Antecedentes de la Empresa 1](#_Toc174279470)

[1.1.2. Antecedentes del objeto de estudio 2](#_Toc174279471)

[1.2. DEFINICION DE PROBLEMA 2](#_Toc174279472)

[1.2.1. Formulación del problema 2](#_Toc174279473)

[1.2.2. Descripción del problema 3](#_Toc174279474)

[1.3. PROPUESTA 3](#_Toc174279475)

[1.4. OBJETIVOS 3](#_Toc174279476)

[1.4.1. Objetivo General 3](#_Toc174279477)

[1.4.2. Objetivo Especifico 3](#_Toc174279478)

[1.5. DEMILITACIÓN 4](#_Toc174279479)

[1.5.1. Alcance espacial 4](#_Toc174279480)

[1.5.2. Alcance de contenido 4](#_Toc174279481)

[1.5.3. Delimitación temporal 5](#_Toc174279482)

[1.6. JUSTIFICACIÓN 5](#_Toc174279483)

[1.6.1. Justificación practica 5](#_Toc174279484)

[1.6.2. Justificación académica 5](#_Toc174279485)

[1.7. DISEÑO METODOLÓGICO 6](#_Toc174279486)

[1.7.1. Tipos de investigación 6](#_Toc174279487)

[1.7.2. Metodología 6](#_Toc174279488)

[1.7.3. Fuentes de información 6](#_Toc174279489)

[1.8. PLANIFICACIÓN 7](#_Toc174279490)

[1.8.1. Plan de desarrollo de software 7](#_Toc174279491)

[1.8.2. Recursos 12](#_Toc174279492)

[2. MARCO TEÓRICO 15](#_Toc174279493)

[2.1. LENGUAJE UNIFICADO DE MODELADO (UML) 15](#_Toc174279494)

[2.1.1. Diagrama de Casos de Uso 15](#_Toc174279495)

[2.1.2. Diagrama de Actividad 17](#_Toc174279496)

[2.1.3. Diagrama de Clase 18](#_Toc174279497)

[2.1.4. Diagrama de Paquete 19](#_Toc174279498)

[2.1.5. Diagrama de Secuencia 20](#_Toc174279499)

[1.1.1. Diagrama de Colaboración 21](#_Toc174279500)

[1.1.2. Diagrama de Despliegue 22](#_Toc174279501)

[2.2. METODOLOGIA PROCESO UNIFICADO DESARROLLO DE SOFTWARE 23](#_Toc174279502)

[2.2.1. Iterativo e Incremental 23](#_Toc174279503)

[2.2.2. Dirigido por los casos de uso 23](#_Toc174279504)

[2.2.3. Centrado en la arquitectura 23](#_Toc174279505)

[2.2.4. Enfocado en los riesgos 23](#_Toc174279506)

[2.2.5. Fases 24](#_Toc174279507)

[2.3. ENTERPRISE ARCHITECT (Software) 24](#_Toc174279508)

[2.4. STAR UML (Software) 25](#_Toc174279509)

[2.5. TECNOLOGÍA Y ARQUITECTURA 26](#_Toc174279510)

[2.5.1. Internet 26](#_Toc174279511)

[2.5.2. Servicio Web 27](#_Toc174279512)

[2.5.3. Relación Cliente Servidor 28](#_Toc174279513)

[2.5.4. Proceso de Creación y Publicación de Páginas Web 29](#_Toc174279514)

[2.5.5. Arquitectura MVC (Modelo Vista Controlador) 30](#_Toc174279515)

[2.5.6. Tecnología Open Source 32](#_Toc174279516)

[2.6. TIPOS DE LENGUAJE DE PROGRAMACIÓN 33](#_Toc174279517)

[2.7.1. HTML5 34](#_Toc174279526)

[2.7.2. Html 35](#_Toc174279527)

[2.7.3. PHP (Hypertext Pre-processor) 36](#_Toc174279528)

[2.7.4. CSS3 (Hojas de Estilos en Cascada) 38](#_Toc174279529)

[2.7.5. JavaScript  40](#_Toc174279530)

[2.7.6. Tailwind 43](#_Toc174279531)

[2.7.7. Font Awesome 44](#_Toc174279532)

[2.8. GESTOR DE BASE DE DATOS 44](#_Toc174279533)

[2.8.1. MySQL 44](#_Toc174279534)

[2.8.2. Maria DB 46](#_Toc174279535)

[2.8.3. Lenguaje de Definición de Datos (DDL) 46](#_Toc174279536)

[2.8.4. Lenguaje de Manipulación de Datos (DML) 47](#_Toc174279537)

[2.9. XAMPP 47](#_Toc174279538)

[2.10. SERVIDOR HTTP APACHE 49](#_Toc174279539)

[2.11. VISUAL STUDIO CODE 50](#_Toc174279540)

[2.12. COMPOSER (SOFTWARE) 50](#_Toc174279541)

[2.13. LARAVEL 52](#_Toc174279542)

[2.13.1. Características 52](#_Toc174279543)

[2.13.2. Historia 53](#_Toc174279544)

[2.13.3. Patrón MVC 53](#_Toc174279545)

[2.13.4. Artisan CLI 54](#_Toc174279546)

[3. MATERIALES Y METODOS 55](#_Toc174279547)

[3.1. Modelo de negocio 55](#_Toc174279548)

[3.1.1. Identificación de los usuarios que realizan el proceso 55](#_Toc174279549)

[3.2. Diagrama de actividad 56](#_Toc174279550)

[3.2.1. Diagrama de actividad: Gestión de servicio 56](#_Toc174279551)

[3.2.2. Diagrama de actividad: Gestión de producto 57](#_Toc174279552)

[3.3. Requerimiento de Software 58](#_Toc174279553)

[3.3.1. Identificación de requisitos 58](#_Toc174279554)

[3.3.2. Listado de requisitos funcionales 59](#_Toc174279555)

[3.3.3. Listado de requisitos no funcionales 61](#_Toc174279556)

[3.4. Listado de Caso de Uso 61](#_Toc174279557)

[3.5. Listado de actores del sistema 62](#_Toc174279558)

[3.6. Diagrama de caso de uso general 63](#_Toc174279559)

[3.7. Análisis de arquitectura 64](#_Toc174279560)

[3.7.1. Modelo de paquete administrador de sistema 64](#_Toc174279561)

[3.7.2. Modelo de paquete recursos humanos 65](#_Toc174279562)

[3.7.3. Modelo de paquete ventas 65](#_Toc174279563)

[3.7.4. Modelo de paquete compras 66](#_Toc174279564)

[3.7.5. Modelo de paquete comercial 66](#_Toc174279565)

[3.8. Diagrama de clase 67](#_Toc174279566)

[3.9. Diagrama de secuencia gestión persona 54](#_Toc174279567)

[3.9.1. Descripción de caso de uso gestionar persona 55](#_Toc174279568)

[3.9.2. Diagrama de secuencia gestión usuario 56](#_Toc174279569)

[3.9.3. Descripción de caso de uso gestionar usuario 57](#_Toc174279570)

[3.9.4. Diagrama de secuencia gestión rol 58](#_Toc174279571)

[3.9.5. Descripción de caso de uso gestionar rol 59](#_Toc174279572)

[3.9.6. Diagrama de secuencia gestión personal 60](#_Toc174279573)

[3.9.7. Descripción de caso de uso gestionar personal 61](#_Toc174279574)

[3.9.8. Diagrama de secuencia gestión venta 62](#_Toc174279575)

[3.9.9. Descripción del caso de uso gestionar venta 63](#_Toc174279576)

[3.9.10. Diagrama de secuencia gestión compra 64](#_Toc174279577)

[3.9.12. Descripción del caso de uso gestionar compra 65](#_Toc174279578)

[3.10. Modelo físico de la base de datos 66](#_Toc174279579)

[3.10.1. Diagrama de despliegue 71](#_Toc174279580)

[4. MODELO DE PRUEBA 72](#_Toc174279581)

[4.1. Concepto de prueba de software 72](#_Toc174279582)

[4.2. Diseño de Casos de Prueba 73](#_Toc174279583)

[4.2.1. Prueba de caso de uso del sistema 73](#_Toc174279584)

[4.2.2. Descripción del caso de uso Prueba del sistema 74](#_Toc174279585)

[4.2.3. Caso de Uso Gestionar Persona 74](#_Toc174279586)

[4.2.4. Caso de Uso Gestionar Persona 75](#_Toc174279587)

[4.2.5 Descripción del Caso de Uso Gestionar Venta 83](#_Toc174279588)

[4.2.6 Caso de Uso Gestionar Venta 83](#_Toc174279589)

[4.2.7 Caso de Uso Gestionar Venta 84](#_Toc174279590)

[4.2.8 Flujos de Eventos 85](#_Toc174279591)

**TABLA DE ILUSTRACIONES**

[Ilustración 1 Simbología Diagrama de Caso de Uso 15](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279421)

[Ilustración 2 Simbología Diagrama de Actividad 17](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279422)

[Ilustración 3 Simbología Diagrama de Clase 18](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279423)

[Ilustración 4 Simbología Diagrama de Paquete 19](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279424)

[Ilustración 5 Simbología Diagrama de Secuencia 20](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279425)

[Ilustración 6 Simbología Diagrama de Colaboración 21](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279426)

[Ilustración 7 Simbología Diagrama de Despliegue 22](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279427)

[Ilustración 8 Logo Enterprise Architect 24](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279428)

[Ilustración 9 Logo Star UML 25](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279429)

[Ilustración 10 Relación Cliente/Servidor 28](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279430)

[Ilustración 11 Arquitectura MVC 30](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279431)

[Ilustración 12 Logo HTML5 34](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279432)

[Ilustración 13 Logo PHP 36](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279433)

[Ilustración 14 Logo CSS3 38](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279434)

[Ilustración 15 Logo JavaScript 40](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279435)

[Ilustración 16 Logo Ajax 42](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279436)

[Ilustración 17 Logo Tailwind 43](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279437)

[Ilustración 18 Logo MySQL 45](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279438)

[Ilustración 19 Logo XAMPP 47](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279439)

[Ilustración 20 Logo Servidor Apache 49](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279440)

[Ilustración 21 Logo Visual Studio Code 50](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279441)

[Ilustración 22 Logo Composer 51](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279442)

[Ilustración 23 Logo Laravel 52](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279443)

[Ilustración 24 Diagrama de actividad gestión servicio 56](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279444)

[Ilustración 25 Diagrama de actividad gestión de producto 57](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279445)

[Ilustración 26 Diagrama de caso de uso general 63](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279446)

[Ilustración 27 Análisis de la arquitectura - Diagrama de paquete 64](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279447)

[Ilustración 28 Modelo de paquete administrador de sistema 64](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279448)

[Ilustración 29 Modelo de paquete recursos humanos 65](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279449)

[Ilustración 30 Modelo de paquete ventas 65](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279450)

[Ilustración 31 Modelo de paquete compras 66](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279451)

[Ilustración 32 Modelo de paquete comercial 66](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279452)

[Ilustración 33 Diagrama de clases general 67](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279453)

[Ilustración 34 Diagrama de secuencia gestión persona 54](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279454)

[Ilustración 35 Diagrama de secuencia gestión usuario 56](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279455)

[Ilustración 36 Diagrama de secuencia gestión rol 58](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279456)

[Ilustración 37 Diagrama de secuencia gestión personal 60](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279457)

[Ilustración 38 Diagrama de secuencia gestión venta 62](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279458)

[Ilustración 39 Diagrama de secuencia gestión compra 64](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279459)

[Ilustración 40 Diagrama de despliegue 71](file:///C:\Users\jose.chavarria\Desktop\TFG.docx#_Toc174279460)

[Ilustración 41 Login de acceso al sistema Elaboración: Propia 73](#_Toc174279461)

[Ilustración 42 Menú Principal del Sistema Elaboración: Propia 73](#_Toc174279462)

[Ilustración 43 Gestionar Personas Elaboración: Propia 74](#_Toc174279463)

[Ilustración 44 Gestionar Persona Elaboración: Propia 75](#_Toc174279464)

[Ilustración 45 Gestionar Venta 83](#_Toc174279465)

[Ilustración 46 Registrar Venta 83](#_Toc174279466)

[Ilustración 47 Agregar Venta 84](#_Toc174279467)

**INDICE DE TABLAS**

[Tabla 1 Distribución de tiempo e iteraciones 8](#_Toc174279405)

[Tabla 2 Hitos de las fases 9](#_Toc174279406)

[Tabla 3 Cronograma general del proyecto 12](#_Toc174279407)

[Tabla 4 Descripción de negocio 55](#_Toc174279408)

[Tabla 5 Descripción de Actores 55](#_Toc174279409)

[Tabla 6 Identificación de requisitos 59](#_Toc174279410)

[Tabla 7 Listado de requisitos funcionales 60](#_Toc174279411)

[Tabla 8 Listado de requisitos no funcionales 61](#_Toc174279412)

[Tabla 9 Listado de Caso de Uso 62](#_Toc174279413)

[Tabla 10 Listado de actores del sistema 62](#_Toc174279414)

[Tabla 11 Descripción de caso de uso gestionar persona 55](#_Toc174279415)

[Tabla 12 Descripción de caso de uso gestionar usuario 57](#_Toc174279416)

[Tabla 13 Descripción de caso de uso gestionar rol 59](#_Toc174279417)

[Tabla 14 Descripción de caso de uso gestionar personal 61](#_Toc174279418)

[Tabla 15 Descripción del caso de uso gestionar venta 63](#_Toc174279419)

[Tabla 16 Descripción del caso de uso gestionar compra 65](#_Toc174279420)

****

**Capítulo I**

**INTRODUCCION**

1. INTRODUCCIÓN
   1. ANTECEDENTES

### Antecedentes de la Empresa

La prestigiosa empresa que brinda servicios para la salud y bienestar de la población “Fabiola Spa" nace a través de un emprendimiento unipersonal que brindaba servicios en la calle en el año 2008 a fueras de una Universidad.

Con el pasar de los años decide alquilar un espacio para implementar más servicios y también de personal, lamentablemente quebró por una falta de administración y vuelve nuevamente con el servicio en la calle.

Años después con una mente más explícita y con experiencia nuevamente alquila un espacio ahí es donde queda establecida y reconocida en el ámbito de la belleza.

Hoy cuenta con un edifico completo para brindar sus servicios a toda la población de Santa Cruz de la Sierra ubicado en la avenida Cañoto sobre el primer anillo junto a supermercados Fidalga entre calle Cuellar y Rafael Peña con una infraestructura moderna, con equipos de última generación. Con valores como la innovación, tendencia, vanguardia, moda, excelencia en calidad de servicio, humanidad, amabilidad, carisma, presencia y sobre todo respeto.

**Propósito de la empresa**: Fabiola Spa surge por la necesidad de suplir la creciente demanda de los servicios de belleza en la ciudad de Santa Cruz. Especializando nos constantemente con lo más nuevo en cortes, peinados, maquillaje etc. Innovando en cuanto a nuevas técnicas de belleza, productos de alta calidad por estilistas especializados y reconocidos de varios países de América.

**Misión****:** Prestar un servicio para la salud y bienestar con personal altamente calificado

Nuestra misión es brindar experiencias de relajación y rejuvenecimiento excepcionales a través de tratamientos holísticos, terapias naturales y técnicas innovadoras. Nos comprometemos a ofrecer un oasis de tranquilidad donde nuestros clientes puedan desconectar del estrés diario y revitalizarse.

**Visión****:** Ser la empresa líder en la satisfacción de necesidades de belleza y salud a nivel nacional e incursionar en el mercado internacional.

Ser el refugio de bienestar preferido por nuestros clientes, donde puedan revitalizar mente, cuerpo y espíritu, y experimentar la serenidad y el rejuvenecimiento en un ambiente acogedor y relajante.

**Objetivos:**

* Ofrecer una amplia gama de tratamientos terapéuticos que aborden las necesidades únicas de cada cliente.
* Proporcionar un ambiente relajante y seguro que promueva la calma y la tranquilidad.
* Mantener altos estándares de calidad en nuestros servicios y productos.
* Fomentar la formación continua de nuestros terapeutas para estar al tanto de las últimas tendencias y técnicas.
* Establecer relaciones duraderas con nuestros clientes, basadas en la confianza y la satisfacción.

### Antecedentes del objeto de estudio

Se llevará acabo en la misma sucursal donde se encuentra la empresa de la ciudad de Santa Cruz de la Sierra y actualmente manejan el registró de ingresos y egresos de forma manual en muchos casos ocasionando desbalance.

Toda la información se registra en cuaderno muchas veces se llega a perder y compran nuevo cuaderno siendo así es muy complicado tener un reporte de ingresos y egresos a los propietarios.

También observando la ampliación de la empresa y la exigencia de los clientes al acceder a los servicios del spa y la falta de control en los encargados de los productos ocasionando molestia y espera en los clientes frecuentes.

* 1. DEFINICION DE PROBLEMA

### Formulación del problema

¿Cómo se planea abordar los desafíos actuales relacionados con el registró manual de notas de venta y dificultad para cuadrar ingresos y egresos además mejorar el control de productos para evitar molestias en los clientes?.

### Descripción del problema

La empresa enfrenta problemas debido a una gestión inadecuada en el registro de ventas, lo que resulta en la falta de registro o incluso la pérdida de datos relacionados con las ventas.

A continuación, se detallan los problemas de recepción:

* Los ingresos y egresos del día son escritos en cuaderno manualmente, lo cual dificulta cuadrar.
* En algunos casos la perdida de nota de venta hace variar y no registrar los servicios del personal.
* Molestia en los clientes frecuentes por la falta de administración.
* El mal control de los productos ocasionando espera y en ocasiones molestia de parte de los clientes.
  1. PROPUESTA

Se propone la implementación de un sistema de gestión y administración simple, moderno y eficiente que proporcionará una solución integral para abordar los desafíos actuales de la empresa. Al automatizar procesos, mejorar la precisión de los registros y brindar herramientas de análisis, la empresa estará en una posición sólida para ofrecer un servicio de alta calidad y aumentar la satisfacción del cliente.

* 1. OBJETIVOS

### Objetivo General

Desarrollar un sistema de información para la gestión de ventas de servicios del Centro de Bienestar "Fabiola Spa" en la ciudad de Santa Cruz de la Sierra, utilizando la metodología del Proceso Unificado de Desarrollo de Software.

### Objetivo Especifico

1. Capturar los requisitos funcionales y no funcionales del sistema en base a las necesidades específicas del spa, incluyendo el registro de ventas, el control de ingresos y egresos.
2. Definir el alcance del desarrollo, identificando las características principales del sistema.
3. Diseñar una arquitectura MVC sólida para el sistema de gestión y administración que permita una integración fluida de los módulos.
4. Documentar los modelos de análisis, diseño e implementación del proceso unificado de desarrollo de software, abarcando la arquitectura, el diseño de la base de datos, las tecnologías empleadas y las decisiones de diseño adoptadas.
5. Establecer procedimientos de soporte y mantenimiento para atender posibles problemas y garantizar la funcionalidad continua del sistema.
6. Realizar pruebas exhaustivas para verificar la funcionalidad, seguridad y usabilidad del sistema antes de su implementación completa.
7. Evaluar la aceptación y eficacia del sistema por parte del personal y los clientes, recopilando retroalimentación y realizando ajustes según sea necesario.
8. Proporcionar capacitación al personal del spa para garantizar una transición sin problemas al nuevo sistema, asegurando su correcto uso y aprovechamiento.
   1. DEMILITACIÓN

### Alcance espacial

El sistema se implementará en la empresa "FABIOLA SPA" ubicado en la Av. Cañoto 202 junto al supermercado Fidalga entre calle Cuellar y Rafael Peña en la ciudad de Santa Cruz de la Sierra.

### Alcance de contenido

* Gestión de ventas y compras: se realizará una buena gestión a todos los datos almacenados sobre los egresos que se registra diariamente.
* Gestión de personal: se permitirá la funcionalidad de seguimiento del personal, incluyendo la asignación de sus respectivas áreas.
* Gestión de usuario y clientes: el sistema permitirá gestionar la información básica de los usuario y clientes.
* Gestión de compras: se desarrollará el registro automatizado de ventas y almacenamiento digital de notas de venta para la integridad de los datos.
* Gestión de productos: el sistema gestionara el registro de productos como nombre del producto y el stock disponible.

### Delimitación temporal

La duración estimada para la culminación del proyecto es de 5 meses teniendo como fecha de inicio el 01 de noviembre de 2023c y fecha de finalización el 14 de abril de 2024.

Los cuáles serán descritos detalladamente en la planificación temporal del proyecto.

* 1. JUSTIFICACIÓN

### Justificación practica

La implementación del sistema propuesto será de gran utilidad tanto para los propietarios como para los administradores del spa. Permitirá un seguimiento en tiempo real de los productos utilizados en los tratamientos y servicios, además de proporcionar un registro preciso de las ventas realizadas en diferentes periodos, ya sea mensual, diario o anual. La funcionalidad de manejo de reportes brindará una visión clara del desempeño del negocio y permitirá tomar decisiones informadas basadas en datos concretos.

Además, el sistema facilitará la gestión de clientes al almacenar información importante, preferencias y tratamientos anteriores, lo que mejorará la personalización y la experiencia del cliente.

### Justificación académica

La implementación del sistema propuesto para el spa se alinea perfectamente con los objetivos académicos al aplicar los conocimientos adquiridos en diversas asignaturas. A través del desarrollo de este software, poniendo en práctica conceptos teóricos y habilidades técnicas en un entorno real que permita facilitar el manejo de información en relación a los datos almacenados por el spa mediante una buena base de datos, desarrollo de software, utilización de lenguajes de programación modernos entre otros.

Además, la implementación del sistema permitirá explorar la integración de tecnología de la información en el ámbito de la administración y la gestión del spa.

La atención rápida y dinámica que el sistema ofrecerá es un ejemplo práctico de cómo la tecnología puede mejorar la eficiencia de las operaciones y la satisfacción del cliente.

* 1. DISEÑO METODOLÓGICO

La investigación se llevó a cabo siguiendo una metodología basada en enfoques explorativa y descriptiva de aplicación práctica, con el propósito de comprender la realidad operativa del spa y proponer soluciones pertinentes a los desafíos planteados.

### Tipos de investigación

El tipo de investigación que se empleó en este sistema de información es explorativa y descriptiva:

**Explorativa:** Se optó por un enfoque explorativa para analizar en profundidad las experiencias y necesidades de los clientes del spa en relación con los servicios ofrecidos.

**Descriptiva:** La investigación se centró en ofrecer soluciones específicas a los problemas detectados en la gestión y administración del spa.

### Metodología

Se aplicará la metodología PUDS, esta metodología destaca por su enfoque en iteraciones, interacción constante con los usuarios y adaptabilidad a cambios en los requisitos.

El proceso unificado de desarrollo de software está dirigido por caso de uso, centrado de arquitectura y es interactivo e incremental. Se enfocará en casos de uso, arquitectura y procesos de desarrollo de software para asegurar la eficacia y el alineamiento con las necesidades del spa. Como herramienta de modelado para artefactos del sistema se utilizará el lenguaje unificado de modelado UML ya que prescribe una notación estándar y semántica esenciales para modelado de sistema orientado a objeto.

### Fuentes de información

* **Principal:** La información fundamental se recopilará mediante entrevistas con los propietarios y el personal del spa para entender los procesos internos y desafíos actuales.
  + **Encargado de la empresa:** Gestionar toda la información de todos los registros del personal.
  + **Encargado de caja:** Gestionar todos los ingresos y egresos diarios del spa.
  + **Supervisores:** Es el encargado de la revisión de los productos del spa para realizar una atención al cliente eficaz
* **Secundaria:** La investigación se complementará con datos obtenidos de fuentes secundarias, como artículos, blogs y estudios relacionados con la industria de los spas y la gestión empresarial.
  1. PLANIFICACIÓN

### Plan de desarrollo de software

El proyecto de desarrollo de software del spa se llevará a cabo mediante la metodología proceso unificado de desarrollo de software (PUDS). Por considerarse una metodología de desarrollo evolutiva, que se caracteriza por su naturaleza iterativa e incremental con sus cuatro etapas: inicio, elaboración, construcción y transición. Realizando las iteraciones dentro de cada fase, que serán detalladas más adelante.

* El proyecto comienza el 14 de agosto con el desarrollo del Modelo de Negocio.
* En el modelo de requerimientos para comprender las necesidades, desafíos y definir objetivos claros de sistema se desarrollará requerimientos funcionales y no funcionales, casos de uso y modelo de dominio del sistema.
* En el modelo de análisis se desarrollará un análisis de la arquitectura y diagramas de colaboración para ilustrar las interacciones entre objeto y escenario clave.
* El modelo de diseño del sistema se desarrollarán los diagramas de despliegue para asegurarse de que esté disponible y funcional para su uso.
* En el modelo de pruebas se realizarán pruebas de rendimiento para asegurar que el sistema maneje cargas de trabajo esperadas e identificar y documentar cualquier error o problema identificado.
* En el modelo de implementación se desarrollará los casos de usos priorizados asegurándose en cumplir los requisitos funcionales.

**Plan de las fases:**

El desarrollo se llevará a cabo en base a fases con una o más iteraciones en cada una de ellas. La siguiente tabla muestra la distribución de tiempos y el número de iteraciones de cada fase (para las fases de construcción y transición es solo una aproximación muy preliminar).

|  |  |  |
| --- | --- | --- |
| **FASE** | **Nro. ITERACIONES** | **DURACION** |
| FASE DE INICIO | 1 | 4 SEMANAS |
| FASE DE ELABORACIÓN | 2 | 8 SEMANAS |
| FASE DE CONSTRUCCIÓN | 2 | 5 SEMANAS |
| FASE TRANSICIÓN | 2 | 3 SEMANAS |

Tabla 1 Distribución de tiempo e iteraciones

|  |  |  |  |
| --- | --- | --- | --- |
| **DESCRIPCION** | | **HITO** | |
| **FASES** | **NRO. ITERACIONES** | **ACTIVIDADES** | |
| FASE DE INICIO | 1 | Esta fase inicial se obtendrá todos los requisitos para luego analizar los requisitos del software desde una perspectiva del usuario las cuales serán tomados por medio de entrevistas. Los principales casos de uso serán identificados y se harán un refinamiento del plan de desarrollo del proyecto. La aceptación del cliente/usuario del plan de desarrollo marca el final de esta fase. | |
| FASE ELABORACION | 2 | Durante esta fase se analiza los requisitos obtenidos y se desarrolla un prototipo de arquitectura. Al final de esta fase todos los casos de uso que correspondan a requisitos serán implementados en la siguiente fase (Análisis/Diseño). La revisión y aceptación del prototipo de la arquitectura del sistema dan paso al final de esta fase. | |
| FASE CONSTRUCCION | 2 | En esta fase de construcción se termina de analizar y diseñar todos los casos de uso, refinando el Modelo de Análisis/Diseño. El producto se construye en base a 2 iteraciones cada una con pruebas y se valida con el cliente/usuario. Se empieza a elaborar el material de apoyo para el usuario. El hito finaliza con la versión beta, con toda la capacidad operacional y de funcionamiento del producto, se encuentra preparada a ser entregada a los usuarios para las pruebas correspondientes. |
| FASE TRANSICION | 2 | En esta fase se preparará el sistema para distribución, asegurando una fiabilidad en el sistema y la capacitación para el manejo del mismo a los usuarios. El hito que marca el fin de esta fase incluye, la entrega de toda la documentación del proyecto con los manuales de instalación y todo el material de apoyo al usuario, la finalización de la capacitación de los usuarios, el empaquetamiento soporte temporal del producto. |

Tabla 2 Hitos de las fases

**Cronograma del proyecto**

A continuación, se tiene un cronograma general del proyecto en el cual a un principio no se tenía el desglose completo de las tareas a efectuarse, el cronograma fue actualizado y reajustado a los tiempos previstos a medida que las tareas se fueron identificando.

|  |  |  |
| --- | --- | --- |
| **ID** | **NOMBRE TAREA** | **DURACION** |
| 1 | **FASE DE INICIO** | 24 días |
| 2 | **Iteración I1: Dominio de Información plan Proyecto** | 24 |
| 3 | Modelo de Negocio | 10 |
| 4 | Modelo de Requisitos | 10 |
| 5 | Gestión de Proyecto | 4 |
| 6 | **FASE DE ELABORACION** | 48 días |
| 7 | **Iteración E1:** | 20 |
| 8 | Modelo de Negocio | 4 |
| 9 | Modelo de Requisitos | 5 |
| 10 | Modelo de Diseño | 6 |
| 11 | Gestión de Proyecto | 5 |
| 12 | **Iteración E2:** | 28 |
| 13 | Modelo de Negocio | 4 |
| 14 | Modelo de Requisitos | 6 |
| 15 | Modelo de Diseño | 5 |
| 16 | Modelo de Implementación | 4 |
| 17 | Modelo Pruebas | 5 |
| 18 | Gestión de Proyecto | 4 |
| 19 | **FASE DE CONSTRUCCION** | 30 días |
| 20 | **Iteración C1:** | 15 |
| 21 | Modelo de Diseño | 6 |
| 22 | Modelo de Implementación | 5 |
| 23 | Modelo Pruebas | 4 |
| 24 | **Iteración C2:** | 15 |
| 25 | Modelo de Diseño | 6 |
| 26 | Modelo de Implementación | 5 |
| 27 | Modelo Pruebas | 4 |
| 28 | **FASE DE TRANSICION** | 18 días |
| 29 | **Iteración T1:** | 10 |
| 30 | Modelo de Diseño | 3 |
| 31 | Modelo de Implementación | 2 |
| 32 | Modelo Pruebas | 3 |
| 33 | Gestión de Proyecto | 2 |
| 34 | **Iteración T2:** | 12 |
| 35 | Modelo de Diseño | 4 |
| 36 | Modelo de Implementación | 6 |
| 37 | Gestión de Proyecto | 2 |
| TOTAL | | 141 días |

Tabla 3 Cronograma general del proyecto

### Recursos

Los elementos utilizados para el desarrollo del proyecto son los siguientes:

**COMPUTADOR PORTÁTIL**

**Hardware:**

* Core i7
* 8 GB Ram
* 500 GB HDD

**Software:**

* Sistema operativo Windows 10 y Ubuntu 18.04.3.
* Xampp
* Gestor de base de datos MariaDB (PhpMyAdmin)
* Architect Enterprise, StartUML.
* Visual Studio Code

Elementos utilizados para la implementación del sistema:

**COMPUTADOR DE ESCRITORIO CLIENTE**

**Hardware:**

* Core i5
* 8 GB Ram
* 1TB GB SSD

**Software:** -

* Sistema operativo Windows 10
* Navegador web



**Capítulo II**

**MARCO TEÓRICO**

1. MARCO TEÓRICO
   1. LENGUAJE UNIFICADO DE MODELADO (UML)[[1]](#footnote-2)

Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

### Diagrama de Casos de Uso[[2]](#footnote-3)

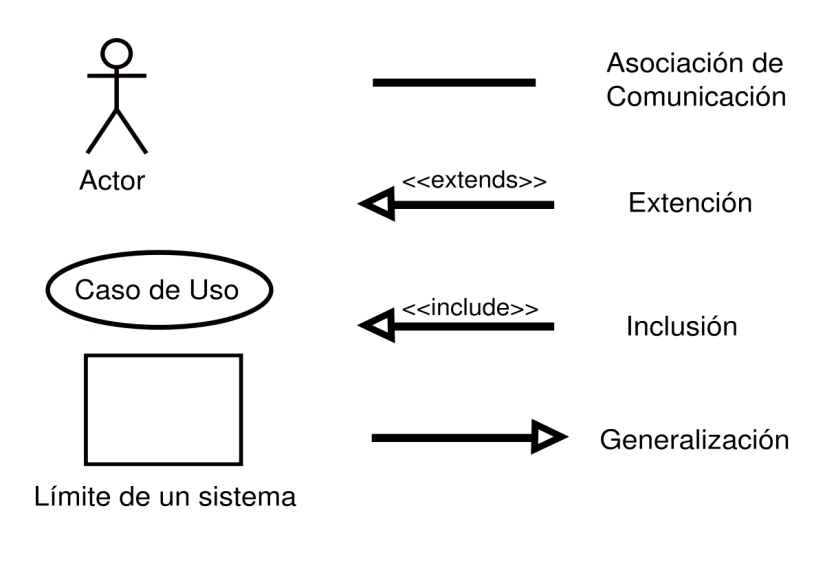


Ilustración 1 Simbología Diagrama de Caso de Uso

El diagrama de casos de uso representa la forma en como un Cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan (operaciones o casos de uso). Un diagrama de casos de uso consta de los siguientes elementos:

* **Actor:** Es un rol que un usuario juega con respecto al sistema. Es importante destacar el uso de la palabra rol, pues con esto se especifica que un Actor no necesariamente representa a una persona en particular, sino más bien la labor que realiza frente al sistema.
* **Casos de Uso:** Es una operación o tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso.

**Relaciones de Uso, Herencia y Comunicación:**

* **Asociación:** Es el tipo de relación más básica que indica la invocación desde un actor o caso de uso a otra operación (caso de uso). Dicha relación se denota con una flecha simple.
* **Dependencia o Instanciación:** Es una forma muy particular de relación entre clases, en la cual una clase depende de otra, es decir, se instancia (se crea). Dicha relación se denota con una flecha punteada.
* **Generalización:** Este tipo de relación es uno de los más utilizados, cumple una doble función dependiendo de su estereotipo, que puede ser de Uso (<<uses>>) o de Herencia (<<extends>>). Este tipo de relación está orientado exclusivamente para casos de uso (y no para actores).
* **Extends:** Se recomienda utilizar cuando un caso de uso es similar a otro (características).
* **Uses:** Se recomienda utilizar cuando se tiene un conjunto de características que son similares en más de un caso de uso y no se desea mantener copiada la descripción de la característica.

### Diagrama de Actividad[[3]](#footnote-4)

El Diagrama de Actividad es un diagrama de flujo del proceso multipropósito que se usa para modelar el comportamiento del sistema, se usan para modelar Casos de Usos, Clases, un método complicado o una actividad.

**Contenido[[4]](#footnote-5)**

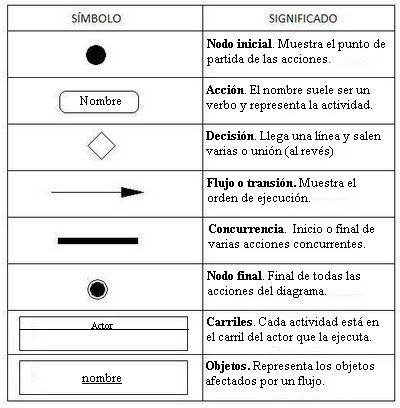


Ilustración 2 Simbología Diagrama de Actividad

* **Estados de actividad y estados de acción:** La representación de ambos es un rectángulo con las puntas redondeadas, en cuyo interior se representa bien una actividad o bien una acción.
* **Transiciones:** Las transiciones reflejan el paso de un estado a otro, bien sea de actividad o de acción. Esta transición se produce como resultado de la finalización del estado del que parte el arco dirigido que marca la transición.
* **Bifurcaciones:** Un flujo de control no tiene porqué ser siempre secuencial, puede presentar caminos alternativos. Para poder representar dichos caminos alternativos o bifurcación se utilizará como símbolo el rombo.
* **División y unión:** No sólo existe el flujo secuencial y la bifurcación, también hay algunos casos en los que se requieren tareas concurrentes. UML representa gráficamente el proceso de división, que representa la concurrencia, y el momento de la unión de nuevo al flujo de control secuencial, por una línea horizontal ancha.
* **Calles:** Cuando se modelan flujos de trabajo de organizaciones, es especialmente útil dividir los estados de actividades en grupos, cada grupo tiene un nombre concreto y se denominan calles. Cada calle representa a la parte de la organización responsable de las actividades que aparecen en esa calle.

### Diagrama de Clase[[5]](#footnote-6)

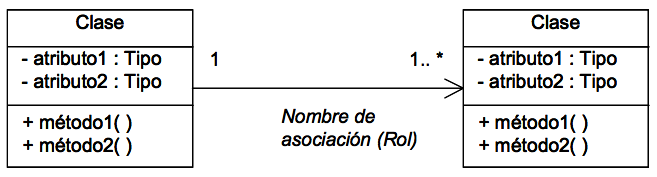


Ilustración 3 Simbología Diagrama de Clase

Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de agregación, ya que una clase es una descripción de conjunto de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y semántica; mostrando un conjunto de elementos que son estáticos, como las clases y tipos junto con sus contenidos y relaciones.

Un diagrama de clases está compuesto por los siguientes elementos:

* Clase: atributos, métodos y visibilidad.
* Relaciones: Herencia, Composición, Agregación, Asociación y Uso.

### Diagrama de Paquete[[6]](#footnote-7)

El objetivo de estos diagramas es obtener una visión más clara del sistema de información orientado a objetos, organizándolo en subsistemas, agrupando los elementos del análisis, diseño o construcción y detallando las relaciones de dependencia entre ellos. El mecanismo de agrupación se denomina Paquete.

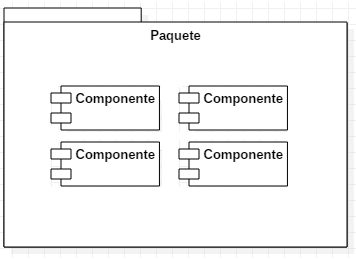


Ilustración 4 Simbología Diagrama de Paquete

* **Paquetes:** Un paquete es una agrupación de elementos, bien sea casos de uso, clases o componentes. Los paquetes pueden contener a su vez otros paquetes anidados que en última instancia contendrán alguno de los elementos anteriores.
* **Dependencias entre paquetes:** Existe una dependencia cuando un elemento de un paquete requiere de otro que pertenece a un paquete distinto. Es importante resaltar que las dependencias no son transitivas.

### Diagrama de Secuencia[[7]](#footnote-8)

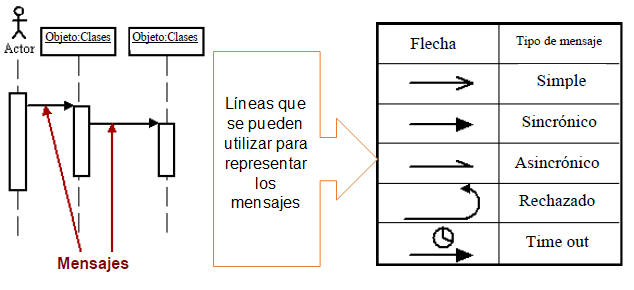


Ilustración 5 Simbología Diagrama de Secuencia

Un diagrama de secuencia es un tipo de diagrama de interacción porque describe cómo y en qué orden un grupo de objetos funcionan en conjunto.

**Componentes**

* Símbolo de objeto: Representa una clase u objeto en UML. El símbolo objeto demuestra cómo se comportará un objeto en el contexto del sistema. Los atributos de las clases no deben aparecer en esta figura.
* Casilla de activación: Representa el tiempo necesario para que un objeto finalice una tarea.
* Símbolo de actor: Muestra entidades que interactúan con el sistema o que son externas al sistema.
* Símbolo de línea de vida: Representa el paso del tiempo a medida que se extiende hacia abajo. Esta línea vertical discontinua representa eventos secuenciales que le ocurren a un objeto durante el proceso graficado.

**Mensajes**

* Símbolo de mensaje sincrónico: Representados por una línea continua y una punta de flecha sólida. Este símbolo se utiliza cuando un remitente debe esperar una respuesta a un mensaje antes de proseguir.
* Símbolo de mensaje asincrónico: Representados por una línea continua y una punta de flecha simple.
* Símbolo de mensaje de respuesta asincrónico: Representados por una línea discontinua y una punta de flecha simple.
* Símbolo de mensaje de respuesta: Están representados con una línea discontinua y una punta de flecha simple. Estos mensajes son las respuestas a las llamadas.
  + 1. **Diagrama de Colaboración[[8]](#footnote-9)**

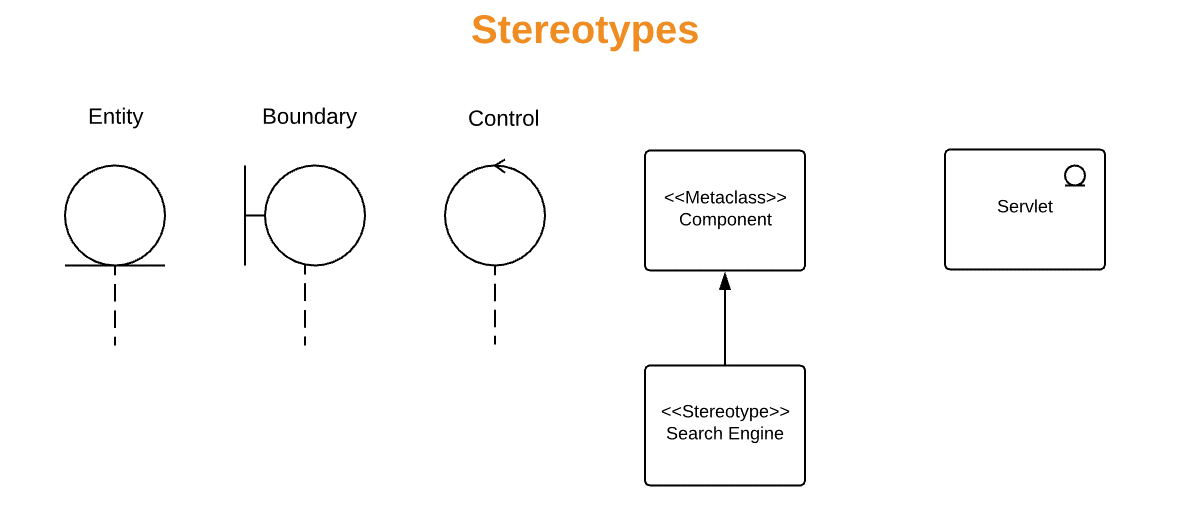


Ilustración 6 Simbología Diagrama de Colaboración

El Diagrama de Colaboración presenta una alternativa al diagrama de secuencia para modelar interacciones entre objetos en el sistema. Mientras que el diagrama de secuencia se centra en la secuencia cronológica del escenario que estamos modelando, el diagrama de colaboración se centra en estudiar todos los efectos de un objeto dado durante un escenario.

Es mostrar la implementación de una operación. La comunicación muestra los parámetros y las variables locales de la operación, así como asociaciones más permanentes.

Cuando se implementa el comportamiento, la secuencia de los mensajes corresponde a la estructura de llamadas anidadas y el paso de señales del programa.

* + 1. **Diagrama de Despliegue[[9]](#footnote-10)**

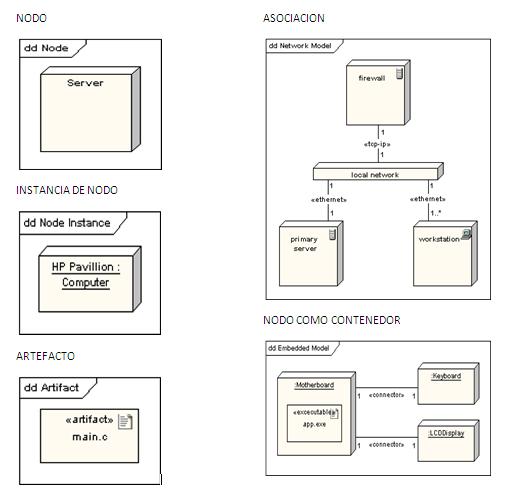


Ilustración 7 Simbología Diagrama de Despliegue

Los diagramas de despliegue son los complementos de los diagramas de componentes que, unidos, proveen la vista de implementación del sistema. Describen la topología del sistema la estructura de los elementos de hardware y el software que ejecuta cada uno de ellos. Los diagramas de despliegue representan a los nodos y sus relaciones.

Los nodos son conectados por asociaciones de comunicación tales como enlaces de red, conexiones TCP/IP. Muestran la configuración en funcionamiento del sistema incluyendo su software y su hardware.

* 1. METODOLOGIA PROCESO UNIFICADO DESARROLLO DE SOFTWARE[[10]](#footnote-11)

El Proceso Unificado de Desarrollo Software o simplemente Proceso Unificado es un marco de desarrollo de software que se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y por ser iterativo e incremental.

### Iterativo e Incremental

El Proceso Unificado es un marco de desarrollo iterativo e incremental compuesto de cuatro fases denominadas Inicio, Elaboración, Construcción y Transición. Cada una de estas fases es a su vez dividida en una serie de iteraciones (la de inicio puede incluir varias iteraciones en proyectos grandes). Estas iteraciones ofrecen como resultado un incremento del producto desarrollado que añade o mejora las funcionalidades del sistema en desarrollo.

### Dirigido por los casos de uso

Se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones. La idea es que cada iteración tome un conjunto de casos de uso o escenarios y desarrolle todo el camino a través de las distintas disciplinas: diseño, implementación, prueba, etc.

### Centrado en la arquitectura

El Proceso Unificado asume que no existe un modelo único que cubra todos los aspectos del sistema. Por dicho motivo existen múltiples modelos y vistas que definen la arquitectura de software de un sistema.

### Enfocado en los riesgos

Requiere que el equipo del proyecto se centre en identificar los riesgos críticos en una etapa temprana del ciclo de vida. Los resultados de cada iteración, en especial los de la fase de Elaboración deben ser seleccionados en un orden que asegure que los riesgos principales son considerados primero.

### Fases

El Proceso Unificado de desarrollo puede ser dividido en cuatro fases para su mejor desarrollo. Estas fases ayudando tanto a la elaboración como a la resolución de problemas.

**Inicio:** En la fase de inicio se define el negocio: facilidad de realizar el proyecto, se presenta un modelo, visión, metas, deseos del usuario, plazos, costos y viabilidad.

**Elaboración:** En esta fase se obtiene la visión refinada del proyecto a realizar, la implementación iterativa del núcleo de la aplicación, la resolución de riesgos altos, nuevos requisitos y se ajustan las estimaciones.

**Construcción:** Esta abarca la evolución hasta convertirse en producto listo incluyendo requisitos mínimos. Aquí se afinan los detalles menores como los diferentes tipos de casos o los riesgos menores.

**Transición:** En esta fase final, el programa debe estar listo para ser probado, instalado y utilizado por el cliente sin ningún problema. Una vez finalizada esta fase, se debe comenzar a pensar en futuras novedades para la misma.

Desde el punto de vista Técnico: el proyecto está formado por los flujos de trabajo fundamentales: captura de requerimientos, análisis, diseño, implementación y pruebas.

* 1. ENTERPRISE ARCHITECT (Software)[[11]](#footnote-12)



Ilustración 8 Logo Enterprise Architect

Es una herramienta escalable de modelado, diseño y administración, colaborativa, compatible con muchos estándares basada en UML y estándares relacionados. Ágil, intuitiva y extensible, con poderosas características para dominios específicos totalmente integrados, que permite visualizar, analizar, modelar, probar y mantener un amplio rango de sistemas, software, procesos y arquitecturas.

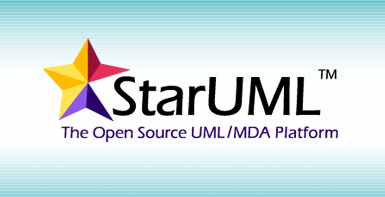
* 1. STAR UML (Software)[[12]](#footnote-13)

Ilustración 9 Logo Star UML

Es una herramienta UML de licencia gratuita (inicialmente comercial), desarrollada en 1996 y posteriormente en el 2005 modificada por la GLP para el modelamiento de software, basándose en estándares UML y DMA. Muy fácil de usar, debido a la simplicidad y rápida percepción de sus objetos, funciones y características, otra característica fundamental es que su código es compatible con lenguajes como C++ y Java.

El software heredó todas las características de la versión comercial y poco a poco ha ido mejorando sus características, entre las cuales se encuentran:

* Diagrama de casos de uso.
* Diagrama de clase.
* Diagrama de secuencia.
* Diagrama de colaboración.
* Diagrama de actividad.
* Diagrama de despliegue.
* Diagrama de composición estructural (UML 2.0).
* La capacidad de generar código a partir de los diagramas y viceversa, actualmente funcionando para los lenguajes c++, c# y java.
* Capacidad para generar documentación en formatos Word, Excel y PowerPoint sobre los diagramas.
* Compatibilidad con SO Windows.
  1. **TECNOLOGÍA Y ARQUITECTURA[[13]](#footnote-14)**
     1. **Internet**

El objeto de esta sección no es tanto definir lo que es Internet como el poner de manifiesto algunas de las características más destacadas de la misma que influyen decisivamente en las tecnologías de desarrollo vinculadas al servicio web, como es el caso del lenguaje PHP. Se asume que cualquier lector interesado en este libro es usuario de Internet y conoce los diferentes servicios que la red ofrece, posiblemente también su interés en el desarrollo de aplicaciones web.

Internet (INTERconected NETwork) es una red de redes de ordenadores de todo tipo que se comunican mediante un lenguaje común: el conocido como protocolo TCP/IP.

Esa primera característica de la heterogeneidad de los equipos conectados es clave para entender el funcionamiento de todos los servicios de la red y para comprender la necesidad de la portabilidad en cualquier desarrollo que se quiera hacer en el ámbito de Internet. A nivel de programadores de aplicaciones web, que no es necesario disponer de conocimientos técnicos sobre los protocolos de comunicación en los que se basa Internet.

Los desarrollos de aplicaciones realizados en Internet tienen también un campo de aplicación en aquellas redes privadas que usan los mismos sistemas y protocolos que Internet: las denominadas Intranets.

* + 1. **Servicio Web**

El servicio WWW, o simplemente Web, se podría definir como un amplio sistema multimedia de acceso a información heterogénea distribuida por toda la red en forma de documentos hipertextuales (hipertextos).

Como ya fue comentado en la introducción de este capítulo, este servicio surgió en 1990 en el CERN (Centre Européen de Recherche Nucléaire) con el objetivo de facilitar la distribución de información entre equipos investigadores geográficamente dispersos.

Se buscaba que los recursos disponibles en formato electrónico fuesen accesibles para cada investigador desde su propia terminal de forma clara y simple, posibilitando el salto entre elementos de información conexos.

El término hipertexto que empezó a hacerse popular a partir de la aparición de este servicio tiene, sin embargo, su definición en un trabajo de Ted Nelson en 1965, la definición original del término es: "Un cuerpo de material escrito o gráfico interconectado de un modo complejo que no se puede representar convenientemente sobre el papel; puede contener anotaciones, adiciones y notas de los estudiosos que lo examinan".

Algunas de las características destacadas de los hipertextos son:

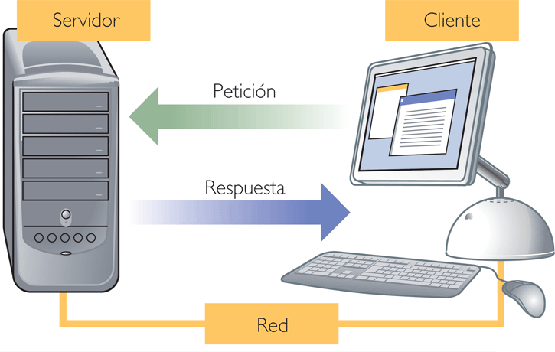
* Almacenamiento de un gran volumen de información.
* Facilidad de acceso y consulta.
* Presentación de una forma más agradable.
* Uso de todas las tecnologías de la información.
* Permiten una “navegación” individualizada.
* Estructuración multidimensional.
* Multiplataforma.
* Dinamismo e interactividad.
  + 1. **Relación Cliente Servidor**

Ilustración 10 Relación Cliente/Servidor

Todos los servicios que ofrece Internet, y por supuesto entre ellos el servicio web, se basan en la denominada relación cliente/servidor. En Internet se pueden encontrar dos tipos de equipos conectados:

**Servidores:** Ordenadores que ofrecen sus servicios al resto de equipos conectados. Suelen tener una presencia estable en la red, lo que se concreta en tener asignadas direcciones IP permanentes. En ellos es donde están alojadas, por ejemplo, las páginas web.

**Clientes:** Equipos que los usuarios individuales utilizan para conectarse a la red y solicitar servicios a los servidores. Durante el tiempo de conexión tienen presencia física en la red. Normalmente los proveedores de acceso a Internet asignan a estos equipos una dirección IP durante su conexión, pero esa dirección es variable, es decir, cambia de unas conexiones a otras (IP dinámica).

Los conceptos de cliente y servidor se suelen utilizar con dos significados diferentes, en referencia al hardware el sentido es el indicado anteriormente, el servidor hace referencia al equipo remoto al que se realiza la conexión y el cliente sería el equipo local utilizado para efectuar dicha conexión. Pero también se utilizan esos conceptos en referencia al software:

Programa servidor es el programa que debe estar ejecutándose en el equipo servidor para que este pueda ofrecer su servicio. Un documento HTML sin más almacenado en el equipo remoto no basta para que sea accesible como página web por el resto de usuarios de Internet, en ese equipo debe estar ejecutándose una aplicación servidor web.

Uno de los programas servidores web más conocido y utilizado es Apache, programa que también pertenece a la corriente open source. Existen otros servidores web como el Personal Web Server (PWS) o el IIS disponibles en los equipos Windows. En el caso de otros servicios como el correo electrónico o la transferencia de ficheros se necesitarían igualmente los correspondientes programas en el servidor.

Programa cliente es en este caso el software necesario en el equipo cliente para tener acceso al correspondiente servicio. Así, por ejemplo, los navegadores como el Internet Explorer o Mozilla son ejemplos de clientes web; un programa como Outlook es un ejemplo de cliente de correo electrónico y programas como WS\_FTP o CuteFTP son ejemplos de clientes FTP.

* + 1. **Proceso de Creación y Publicación de Páginas Web**

El proceso de creación y publicación de una página web debe pasar por una serie de fases:

* **Definición de la página:** Toda página web tiene detrás un código fuente que la define. Cuando se accede a una página web, aunque en el monitor se puedan visualizar documentos con imágenes y texto de diferentes tamaños, colores y formatos, debe tenerse presente que detrás de eso hay un documento de texto sin ningún tipo de formato y que incorpora una serie de instrucciones o comandos que son los que realmente generan la página que se visualiza. Ese documento fuente está definido en un lenguaje especial: el conocido HTML (HyperText Markup Language). Se trata de un lenguaje puramente descriptivo que incorpora una serie de comandos o etiquetas (tags) que permiten definir la estructura lógica del documento, dar formato al texto, añadir elementos no textuales.
* **Publicación del documento:** Una vez creado el documento HTML que define la página web, el siguiente paso es evidentemente publicarla para que esté disponible para el resto de usuarios de Internet. La publicación implica la transferencia del documento a un equipo servidor que disponga de un programa de servidor web. Puede optarse por utilizar algún servidor de alojamiento gratuito de páginas o utilizar los espacios que los proveedores de acceso a Internet suelen ofrecer a sus clientes. Otra posibilidad sería configurar un equipo propio para que actúe de servidor, para ello sería preciso contratar con algún proveedor una dirección IP fija y registrar el dominio que se quiera utilizar.
* **Acceso a las páginas web:** En el momento en que una página es publicada en el servidor, cualquier usuario de Internet podría acceder a ella. Para ello es preciso, por un lado, que el usuario utilice un programa adecuado (el navegador, explorador o cliente web), y que el código que define la página sea transferido por la red utilizando el protocolo http (hypertext transfer protocol). Sobre este último aspecto, en principio la mayoría de usuarios no deberían preocuparse. El navegador web es el que se encarga de interpretar los comandos HTML que se reciben y producir a partir de ellos la página web. Existen diferentes navegadores web que se pueden utilizar, por ejemplo, Internet Explorer, Netscape Navigator, HotJava, Mozilla, este último sigue las corrientes open source.
  + 1. **Arquitectura MVC (Modelo Vista Controlador)[[14]](#footnote-15)**

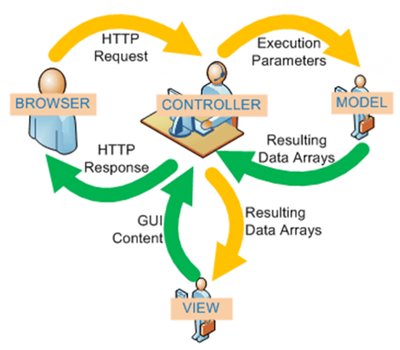


Ilustración 11 Arquitectura MVC

Es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

Se trata de un modelo muy maduro y que ha demostrado su validez a lo largo de los años en todo tipo de aplicaciones, y sobre multitud de lenguajes y plataformas de desarrollo.

**Modelo:** Que contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.

* Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
* Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor".
* Lleva un registro de las vistas y controladores del sistema.
* Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo (por ejemplo, un fichero por lotes que actualiza los datos, un temporizador que desencadena una inserción, etc.).

**Vista:** Interfaz de usuario, que compone la información que se envía al cliente y los mecanismos interacción con éste.

* Recibir datos del modelo y los muestra al usuario.
* Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
* Pueden dar el servicio de "Actualización ()", para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

**Controlador:** Que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

* Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
* Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada al método "Actualizar ()".
  + 1. **Tecnología Open Source**

Es el software cuyo código fuente y otros derechos que normalmente son exclusivos para quienes poseen los derechos de autor, son publicados bajo una licencia de código abierto o forman parte del dominio público. En las licencias compatibles con la Open Source Definición el propietario de los derechos de autor permite a los usuarios utilizar, cambiar y redistribuir el software, a cualquiera, para cualquier propósito, ya sea en su forma modificada o en su forma original.

Frecuentemente el software de código abierto se desarrolla de manera colaborativa y los resultados se publican en internet. ​ El software es el mejor ejemplo del desarrollo del código abierto y se compara con el llamado "contenido generado por los usuarios”. ​ La expresión software de código abierto surgió a partir de una campaña de mercadotecnia para el software libre.

**Historia**

El movimiento del software libre surgió en 1983. En 1998, un grupo de individuos defendieron la idea de cambiar la expresión free software (software libre) por open source software (software de código abierto), debido a la ambigüedad del primero (en inglés, free significa tanto gratis como libre) y al atractivo del segundo respecto a las empresas. Los desarrolladores de software pueden optar por publicar su trabajo bajo una licencia de código abierto, de manera que cualquiera pueda beneficiarse de él. Por lo general, este software permite a las personas crear modificaciones del software, compatibilizarlo con otros sistemas operativos o arquitecturas de hardware, compartirlo con otras personas y comerciarlo.

La Open Source Definition introduce una filosofía en cuanto al código abierto, y además define los términos de uso, modificación y redistribución del software de código abierto. Las licencias de software otorgan a los usuarios derechos que de otro modo estarían reservados, por la ley de derechos de autor, a los propietarios de los derechos de autor del software. Existen varias licencias de software de código abierto compatibles con la Open Source Definition. La más popular y destacada es la GNU General Public License (GPL).

Si bien la distribución de código abierto permite que el código fuente de un producto sea accesible al público, las licencias de código abierto posibilitan a los autores acomodar ese acceso a gusto.

* 1. TIPOS DE LENGUAJE DE PROGRAMACIÓN



Los lenguajes de programación pueden ser clasificados de acuerdo a varios criterios. Una de las primeras clasificaciones que se suele efectuar es la distinción entre lenguajes de bajo nivel y de alto nivel. La programación en los primeros resulta más dificultosa puesto que las instrucciones están muy próximas al hardware del equipo y resultan difíciles de entender por un programador no especialista. El ejemplo clásico de lenguaje de bajo nivel es el lenguaje ensamblador.

La mayor parte de los programadores optan por utilizar lenguajes cuyo código resulta más fácil de entender, por cuanto sus reglas sintácticas se asemejan más a la forma de comunicarse las personas; son lenguajes que están “más cerca” del programador, pero más lejos de la máquina a la que van dirigidos. Estos lenguajes son los denominados "lenguajes de alto nivel" y a ellos pertenecen los lenguajes de programación más conocidos.

Cuando se está desarrollando un programa usando un lenguaje de programación se genera un código (código fuente) que es comprensible para todo aquel usuario que tenga los conocimientos suficientes sobre el correspondiente lenguaje, pero que en ningún caso es comprensible directamente para la máquina. Parece claro por tanto que se necesita un proceso de “traducción” del código fuente que nosotros entendemos a instrucciones entendibles por la máquina. Ese proceso de traducción se puede realizar de dos maneras, y eso da pie a establecer una nueva clasificación de los lenguajes de programación:

* **Lenguajes compilados:** En un lenguaje compilado el código fuente pasa por un proceso denominado "compilación" en el que se genera un código denominado "objeto", que una vez enlazado con otros posibles módulos de código objeto necesarios, genera el fichero ejecutable con el programa. Ese fichero ejecutable es lo único necesario para poder utilizar el programa y contiene todas las instrucciones del mismo, pero en el formato entendible por la máquina.
* El aspecto más importante a destacar es que el proceso de compilación se realiza con anterioridad a cualquier ejecución o uso del programa; en ese proceso se comprueba la validez sintáctica del programa y si todo es correcto se genera el ejecutable. Si se produce un error en la compilación el programa no podrá ser utilizado.
* **Lenguajes interpretados:** En los lenguajes interpretados, la traducción de las instrucciones se va realizando de forma secuencial por una aplicación, denominada "intérprete", al mismo tiempo que se ejecuta el programa. De esta forma, si llegado un punto del programa el intérprete se encuentra con una instrucción errónea, el programa no continúa, pero sí que habrá podido ejecutar todas las sentencias previas. A diferencia de los lenguajes compilados, la verificación de la corrección sintáctica del programa no se realiza antes de la ejecución sino al mismo tiempo. En un lenguaje interpretado, todos los errores son detectados en tiempo de ejecución, tanto los debidos a errores sintácticos como los debidos a condiciones singulares producidas a la hora de ejecutar el programa.

### HTML5[[15]](#footnote-16)



Ilustración 12 Logo HTML5

Es la última versión de HTML. El término representa dos conceptos diferentes:

* Se trata de una nueva versión de HTML, con nuevos elementos, atributos y comportamientos.
* Contiene un conjunto más amplio de tecnologías que permite a los sitios Web y a las aplicaciones ser más diversas y de gran alcance. A este conjunto se le llama HTML5 y amigos, a menudo reducido a HTML5.

**Funciones**

* Semántica: Permite describir con mayor precisión cuál es su contenido.
* Conectividad: Permite comunicarse con el servidor de formas nuevas e innovadoras.
* Sin conexión y almacenamiento: Permite a las páginas web almacenar datos localmente en el lado del cliente y operar sin conexión de manera más eficiente.
* Multimedia: Nos otorga un excelente soporte para utilizar contenido multimedia como lo son audio y video nativamente.
* Gráficos y efectos 2D/3D: Proporciona una amplia gama de nuevas características que se ocupan de los gráficos en la web como lo son canvas 2D, WebGL, SVG, etc.
* Rendimiento e Integración: Proporciona una mayor optimización de la velocidad y un mejor uso del hardware.
* Acceso al dispositivo: Proporciona APIs para el uso de varios componentes internos de entrada y salida de nuestro dispositivo.
* CSS3: Nos ofrece una nueva gran variedad de opciones para hacer diseños más sofisticados.
  + 1. **Html[[16]](#footnote-17)**

HTML no es un lenguaje de programación, esto debes tenerlo muy en claro desde el principio, HTML es un lenguaje de marcado de hipertexto o “HyperText Markup Language” por el desarrollo de sus iniciales en inglés, básicamente este lenguaje se escribe en su totalidad con elementos, estos elementos están constituidos por etiquetas, contenido y atributos.

[[17]](#footnote-18)El origen de HTML se remonta a 1980, cuando el físico Tim Berners-Lee, trabajador del CERN (Organización Europea para la Investigación Nuclear) propuso un nuevo sistema de "hipertexto" para compartir documentos.

Los sistemas de "hipertexto" habían sido desarrollados años antes. En el ámbito de la informática, el "hipertexto" permitía que los usuarios accedieran a la información relacionada con los documentos electrónicos que estaban visualizando.

Tras finalizar el desarrollo de su sistema de "hipertexto", Tim Berners-Lee lo presentó a una convocatoria organizada para desarrollar un sistema de "hipertexto" para Internet. Después de unir sus fuerzas con el ingeniero de sistemas Robert Cailliau, presentaron la propuesta ganadora llamada WorldWideWeb (W3).

### PHP (Hypertext Pre-processor) [[18]](#footnote-19)

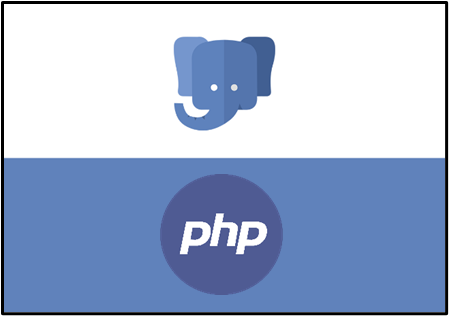


Ilustración 13 Logo PHP

PHP es un Acrónimo recursivo que significa PHP Hypertext Pre-processor (inicialmente PHP Tools, o, Personal Home Page Tools). Fue creado originalmente por Rasmus Lerdorf en 1994; sin embargo, la implementación principal de PHP es producida ahora por The PHP Group y sirve como el estándar de facto para PHP al no haber una especificación formal. Publicado bajo la PHP License, la Free Software Foundation considera esta licencia como software libre.

PHP es un lenguaje interpretado de propósito general ampliamente usado, diseñado especialmente para desarrollo web y que puede ser incrustado dentro de código HTML.

Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno.

Aunque todo en su diseño está orientado a facilitar la creación de página web, es posible crear aplicaciones con una interfaz gráfica para el usuario, utilizando la extensión PHP-Qt o PHP-GTK. También puede ser usado desde la Línea de órdenes, de la misma manera como Perl o Python pueden hacerlo, a esta versión de PHP se la llama PHP CLI (Command Line Interface).

Permite la conexión a diferentes tipos de servidores de bases de datos tales como MySQL, PostgreSQL, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite.

PHP es una alternativa a las tecnologías de Microsoft ASP y ASP.NET (que utiliza C# VB.NET como lenguajes), a ColdFusion de la compañía Adobe (antes Macromedia), a JSP/Java de Sun Microsystems, y a CGI/Perl. Aunque su creación y desarrollo se da en el ámbito de los sistemas libres, bajo la licencia GNU, existe además un IDE (Entorno de Desarrollo Integrado) comercial llamado Zend Studio.

Recientemente, CodeGear (la división de lenguajes de programación de Borland) ha sacado al mercado un IDE para PHP, denominado Delphi for PHP. Existe un módulo para Eclipse, uno de los IDE más populares.

Fue originalmente diseñado en Perl, con base en la escritura de un grupo de CGI binarios escritos en el lenguaje C por el Programador Danés-Canadiense Rasmus Lerdorf en el año 1994 para mostrar su Currículum vitae y guardar ciertos datos, como la cantidad de tráfico que su página web recibía. El 8 de junio de 1995 fue publicado "Personal Home Page Tools" después de que Lerdorf lo combinara con su propio Form Interpreter para crear PHP/FI.

* + 1. **CSS3 (Hojas de Estilos en Cascada)[[19]](#footnote-20)**



Ilustración 14 Logo CSS3

El nombre hojas de estilo en cascada viene del inglés Cascading Style Sheets, del que toma sus siglas. CSS es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C (World Wide Web Consortium) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores.

La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación.

La información de estilo puede ser adjuntada como un documento separado o en el mismo documento HTML. En este último caso podrían definirse estilos generales en la cabecera del documento o en cada etiqueta particular mediante el atributo "<style>".

CSS es un lenguaje utilizado para dar estética a un documento HTML (colores, tamaños de las fuentes, tamaños de elemento, con css podemos establecer diferentes reglas que indicarán como debe presentarse un documento. Podemos indicar propiedades como el color, el tamaño de la letra, el tipo de letra, si es negrita, si es itálica, también se puede dar forma a otras cosas que no sean letras, como colores de fondo de una página, tamaños de un elemento.

**Características**

* Atributo gradiente de colores en borde con CSS y Firefox: Posibilidad de definir el un gradiente de color en el borde de los elementos con CSS, en un atributo no estándar de Firefox.
* Bordes redondeados en CSS 3: Las características de CSS 3 incluyen bordes redondeados, a través del atributo border-radius, que define la curvatura que debe tener el borde del elemento.
* Múltiples imágenes de fondo con CSS: Cómo conseguir que un elemento de la página tenga varias imágenes de fondo a la vez, con CSS básico y con características de CSS.
* Colores RGBA en CSS 3: Veremos qué son los colores RGBA y su notación, que se incluyen en la especificación de Hojas de Estilo en Cascada CSS 3.
* Word-wrap en CSS 3: Una propiedad de CSS 3 que sirve para romper las palabras que son demasiado largas y no caben enteras por la anchura de una caja.
* Textos multi-columna con CSS 3: incorpora nuevos atributos para que el navegador se encargue de producir texto multicolumna, es decir, que maquete directamente el texto en varias columnas sin tener que hacer nosotros nada.
* Bordes con imágenes en CSS 3: El atributo border-image y varios otros de CSS 3 harán posible la utilización de imágenes como bordes de los elementos de la página, sin código HTML especial, simplemente con hojas de estilo.
* Sombras en CSS 3 con box-shadow: Crear sombras en CSS3 con el atributo box-shadow. Por fin podremos aplicar sombras a los elementos de la página, sin usar imágenes, Javascript ni nada extra, simplemente con un atributo de CSS 3.
* Resplandor exterior con CSS3: Cómo realizar un elemento que tenga un resplandor exterior con CSS3 y la propiedad box-shadow.
* Propiedad background-origin de CSS 3: La propiedad de CSS 3 background-origin permite decidir la posición de la imagen de fondo con respecto al borde, padding o el contenido del elemento.
* Atributos CSS3 overflow-x y overflow-y: Descripción de los atributos de CSS3 overflow-x y overflow-y, que sirven para definir cómo renderizar un contenido cuando sobrepasa los límites de un contenedor en la horizontal o vertical.
* Introducción a @font-face de CSS: Fuentes en CSS 3. Sintaxis y principales características de la regla CSS @font-face, que nos permite utilizar cualquier tipografía en una página web.
* Sombras en el texto con text-shadow de CSS: Cómo aplicar sombras y otros efectos en los textos con CSS y el atributo text-shadow.
  + 1. **JavaScript [[20]](#footnote-21)**

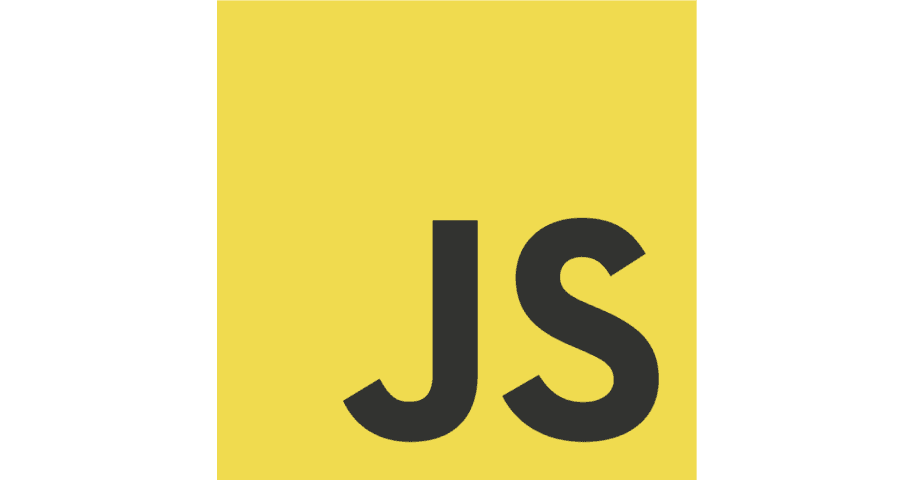


Ilustración 15 Logo JavaScript

Javascript es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos mucho más complejos.

Con Javascript podemos crear diferentes efectos e interactuar con nuestros usuarios.

Este lenguaje posee varias características, entre ellas podemos mencionar que es un lenguaje basado en acciones que posee menos restricciones.

Además, es un lenguaje que utiliza Windows y sistemas X-Windows, gran parte de la programación en este lenguaje está centrada en describir objetos, escribir funciones que respondan a movimientos del mouse, aperturas, utilización de teclas, cargas de páginas entre otros.

**Nacimiento JS**

Javascript nació con la necesidad de permitir a los autores de sitio web crear páginas que permitan intercambiar con los usuarios, ya que se necesitaba crear webs de mayor complejidad. El HTML solo permitía crear páginas estáticas donde se podía mostrar textos con estilos, pero se necesitaba interactuar con los usuarios.

En los años de 1990, Netscape creo Livescript; las primeras versiones de este lenguaje fueron principalmente dedicadas a pequeños grupos de diseñadores Web que no necesitaban utilizar un compilador, o sin ninguna experiencia en la programación orientada a objetos.

En diciembre de 1995, Netscape y Sun Microsystems (el creador del lenguaje Java) luego de unirse objetivo de desarrollar el proyecto en conjunto, reintroducen este lenguaje con el nombre de Javascript. En respuesta a la popularidad de Javascript, Microsoft lanzo su propio lenguaje de programación a base de script, VBScript (una pequeña versión de Visual Basic). En el año de 1996 Microsoft se interesa por competir con Javascript por lo que lanza su lenguaje llamado Jscript, introducido en los navegadores de Internet Explorer. A pesar de las diferentes críticas que se le hacen al lenguaje Javascript, este es uno de los lenguajes de programación más populares para la web. La estandarización de Javascript comenzó en conjunto con ECMA en noviembre de 1996. Es adoptado este estándar en junio de 1997 y luego también por la “Internacional Organization for Standardization” (ISO). El DOM por sus siglas en inglés “Modelo de Objetos del Documento” fue diseñado para evitar incompatibilidades.

**Funcionamiento**

Entre los diferentes servicios que se encuentran realizados con Javascript en Internet se encuentran:

* Correo.
* Chat.
* Buscadores de Información.

También podemos encontrar o crear códigos para insertarlos en las páginas como:

* Reloj.
* Contadores de visitas.
* Fechas.
* Calculadoras.
* Validadores de formularios.
* Detectores de navegadores e idiomas.

**Compatibilidad**

Javascript es soportado por la mayoría de los navegadores como Internet Explorer, Netscape, Opera, Mozilla Firefox, entre otros.

Con el surgimiento de lenguajes como PHP del lado del servidor y Javascript del lado del cliente, surgió Ajax en acrónimo de (Asynchronous JavaScript And XML). El mismo es una técnica para crear aplicaciones web interactivas. Este lenguaje combina varias tecnologías:

HTML y Hojas de Estilos CSS para generar estilos.

Implementaciones ECMAScript, uno de ellos es el lenguaje Javascript.

XMLHttpRequest es una de las funciones más importantes que incluye, que permite intercambiar datos asincrónicamente con el servidor web, puede ser mediante PHP, ASP, entre otros

* + - 1.  AJAX[[21]](#footnote-22)

Ilustración 16 Logo Ajax

Ajax parece ser la palabra de moda en el mundo del desarrollo de aplicaciones Web, AJAX no es una tecnología, sino la unión de varias tecnologías que juntas pueden lograr cosas realmente impresionantes. En resumen, es el acrónimo para Asynchronous JavaScript + XML y el concepto es: Cargar y renderizar una página, luego mantenerse en esa página mientras scripts y rutinas van al servidor buscando, en background, los datos que son usados para actualizar la página solo renderizando la página y mostrando u ocultando porciones de la misma incorporando:

* Presentación basada en estándares usando XHTML y CSS.
* Exhibición e interacción dinámicas usando el Document Object Model;
* Intercambio y manipulación de datos usando XML and XSLT;
* Recuperación de datos asincrónica usando XMLHttpRequest;

Una aplicación AJAX elimina la naturaleza “arrancar-frenar- arrancar-frenar” de la interacción en la Web introduciendo un intermediario -un motor AJAX- entre el usuario y el servidor. Parecería que sumar una capa a la aplicación la haría menos reactiva, pero la verdad es lo contrario. En vez de cargar una página Web, al inicio de la sesión, el navegador carga al motor AJAX (escrito en JavaScript y usualmente “sacado” en un frame oculto). Este motor es el responsable por renderizar la interfaz que el usuario ve y por comunicarse con el servidor en nombre del usuario.

El motor AJAX permite que la interacción del usuario con la aplicación suceda asincrónicamente (independientemente de la comunicación con el servidor). Así el usuario nunca estará mirando una ventana en blanco del navegador esperando a que el servidor haga algo. Google ha introducido en el último año (Orkut, Gmail, la última versión de Google Groups, Google Suggest, y Google Maps) son aplicaciones Ajax. (Para datos más técnicos de estas implementaciones Ajax, lean estos excelentes análisis de Gmail, Google Suggest, y Google Maps.) Otros están siguiendo la tendencia: muchas de las funciones que la gente ama en Flickr dependen de Ajax, y el motor de búsqueda de Amazon A9.com aplica tecnologías similares. Estos proyectos demuestran que Ajax no es solo técnicamente importante, sino también prácticos para aplicaciones en el mundo real.

### Tailwind [[22]](#footnote-23)

Ilustración 17 Logo Tailwind

Tailwind CSS es una potente herramienta para el desarrollo frontend. Está dentro de la clasificación de los frameworks CSS o también llamados frameworks de diseño. Permite a los desarrolladores y diseñadores aplicar estilos a los sitios web de una manera ágil y optimizada. Permite escribir los estilos por medio de clases que se incluyen dentro del código HTML y que afectan a un aspecto muy concreto y específico de las CSS, por ejemplo, el fondo de un elemento, el color del texto o simplemente el margen por la parte de arriba. Este enfoque se conoce como "Atomic CSS", por aplicarse mediante estilos muy determinados y simples. En Tailwind CSS a estas clases se les llama "utility classes" o clases de utilidad en español.

Además, Tailwind es una herramienta que se apoya en PostCSS para todo lo que es la generación del código CSS. Gracias a PostCSS se alcanza un flujo de desarrollo muy avanzado, personalizable, ágil y, sobre todo, extremadamente optimizado. Con ello, el código CSS resultante de un proyecto es realmente ajustado, porque con PostCSS se consige hacer que las clases que finalmente estén en el código de producción sean solamente las que el proyecto está usando realmente, ni una más.

### Font Awesome[[23]](#footnote-24)

Es un conjunto de herramientas de fuentes e iconos basado en CSS y MENOS. Fue hecho por Dave Gandy para su uso con Twitter Bootstrap, y luego se incorporó a BootstrapCDN.

Font Awesome tiene una participación de mercado del 20% entre los sitios web que usan Font Scripts de terceros en su plataforma, ubicándose en el segundo lugar después de Google Fonts.

* 1. **GESTOR DE BASE DE DATOS**
     1. **MySQL[[24]](#footnote-25)**

MySQL es un sistema de gestión de bases de datos (SGBD) multiusuario, multiplataforma y de código abierto bajo licencia GNU como también bajo una variedad de acuerdos propietarios, que pertenecía a la compañía sueca MySQL AB, que tenía casi todos los derechos del código fuente y se encargaba de desarrollar y mantener el sistema, vender soporte y servicios, como también las licencias para usar MySQL. Actualmente pertenece a Oracle Corporation tras comprarla en 2008.

MySQL es una base de datos polivalente. Fue creada usando una mezcla entre los lenguajes C y C++.



Ilustración 18 Logo MySQL

[[25]](#footnote-26)Algunas de las ventajas de MySQL son las siguientes:

* Multiplataforma: Linux, Windows, AIX, Solaris, y un largo de etcétera de sistemas la soportan.
* Múltiples motores de almacenamiento que se adaptan a las distintas necesidades de cada entorno: MyISAM, InnoDB, Memory, etc... Permite usar para cada tabla un motor de almacenamiento distinto.
* Gran velocidad a la hora de realizar operaciones.
* Soporta un amplio número de tipos de datos.
* Tiene una gran comunidad de desarrolladores, y una extensa documentación.
* Uso de transacciones e integridad relacional (dependiendo del tipo de motor de almacenamiento).
* Buena capacidad de indexación y búsqueda y uso de fulltext, que la dotan de una serie de herramientas para hacer búsquedas complejas usando patrones.

### Maria DB[[26]](#footnote-27)

MariaDB se trata de una “sustituto” de MySQL que corre bajo una licencia GPL. Por lo tanto, incorpora todas las mejoras de la versión de MySQL en la que se encuentre y además incluye diferentes mejoras:

* Incorpora nuevos motores de almacenamiento mucho más eficientes que son Aria y XtraDB, los cuales han sido desarrollados para ser los sustitutos de MyISAM e InnoDB respectivamente. Estos permiten ejecutar consultas más complejas y almacenarlas en caché y no en disco duro.
* Aparte de incluir los sustitutos de MyISAM e InnoDB también incorpora nuevos motores de almacenamiento:
* FederatedX: Reemplaza a Federated que incorpora MySQL.
* OQGRAPH: Nos permite que nuestro sistema de base de datos soporte el uso de jerarquías de estructuras y graphs complejos.
* SphinxSE: Nos permite usar searchd bajo Sphinx para permitir búsquedas de texto.
* Cassandra Storage Engine: Nos permite acceder a un clúster de datos. Cabe destacar que por defecto no viene instalado y deberá activarse por separado.
* También incorpora mejoras de rendimiento y versiones de seguridad más rápidas y más transparentes.
* Es de código libre y cuenta con el soporte de la comunidad de desarrolladores, pero también cuenta con el soporte de Oracle.

### Lenguaje de Definición de Datos (DDL)[[27]](#footnote-28)

Las sentencias DDL se utilizan para crear y modificar la estructura de las tablas, así como otros objetos de la base de datos.

* CREATE - para crear objetos en la base de datos.
* ALTER - modifica la estructura de la base de datos.
* DROP - borra objetos de la base de datos.
* TRUNCATE - elimina todos los registros de la tabla, incluyendo todos los espacios asignados a los registros.

### Lenguaje de Manipulación de Datos (DML)

Las sentencias de lenguaje de manipulación de datos (DML) son utilizadas para gestionar datos dentro de los schemas.

* SELECT - para obtener datos de una base de datos.
* INSERT - para insertar datos a una tabla.
* UPDATE - para modificar datos existentes dentro de una tabla.
* DELETE - elimina todos los registros de la tabla; no borra los espacios asignados a los registros.
  1. XAMPP[[28]](#footnote-29)



Ilustración 19 Logo XAMPP

XAMPP es un servidor independiente de plataforma de código libre. Te permite instalar de forma sencilla Apache en tu propio ordenador, sin importar tu sistema operativo (Linux, Windows, MAC o Solaris). Y lo mejor de todo es que su uso es gratuito. Incluye además servidores de bases de datos como MySQL ySQLite con sus respectivos gestores phpMyAdmin y phpSQLiteAdmin. Incorpora también el intérprete de PHP, el intérprete de Perl, servidores de FTP como ProFTPD o FileZilla FTP Serve, etc. entre muchas cosas más.

Paquetes básicos y gráficos:

* Apache, el servidor Web más famoso.
* MySQL, una excelente base de datos de código libre.
* PHP y Perl: lenguajes de programación.
* ProFTPD: un servidor FTP.
* GD (Graphics Draw): la librería de dibujo de gráficos.
* libpng: la librería oficial de referencia de PNG.
* libpeg: la librería oficial de referencia de JPEG.

Paquete de bases de datos:

* gdbm: la implementación GNU de la librería standard dbm de UNIX.
* SQLite: un motor de base de datos SQL muy pequeño y cero configuraciones.
* FreeTDS: una librería de base de datos que da a los programas de Linux y UNIX la habilidad de comunicarse con Microsoft SQL y Sybase.

Paquetes XML:

* expat: una librería parser de XML.
* Salbotron: una toolkit de XML.
* libxml: un parser C de XML y un toolkit para GNOME.

Paquetes PHP:

* PEAR: la librería de PHP.
* Una clase pdf que genera documentos PDF dinámicos con PHP.
* TURCK MMCache: un potenciador de la performance de PHP.

Otros paquetes:

* zlib: una librería de compresión.
* mod\_perl: empotra un intérprete de Perl en Apache.
* gettext: un conjunto de herramientas que asiste a los paquetes GNU para producir mensajes multilinguales.
* mcrypt: un programa de encriptación.
* IMAC C-Client: un API de correos

**Características y Requisitos**

Solamente requiere descargar y ejecutar un archivo .zip, .tar, o .exe, con unas pequeñas configuraciones en alguno de sus componentes que el servidor Web necesitará. XAMPP se actualiza regularmente para incorporar las últimas versiones de Apache/MySQL/PHP y Perl. También incluye otros módulos como OpenSSL y phpMyAdmin. Para instalar XAMPP se requiere solamente una pequeña fracción del tiempo necesario para descargar y configurar los programas por separado.

* 1. SERVIDOR HTTP APACHE[[29]](#footnote-30)



Ilustración 20 Logo Servidor Apache

El Servidor HTTP Apache es un servidor web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. Su nombre se debe a que Behelendorf quería que tuviese la connotación de algo que es firme y enérgico, pero no agresivo, y la tribu Apache fue la última en rendirse al que pronto se convertiría en gobierno de EEUU, y en esos momentos la preocupación de su grupo era que llegasen las empresas y "civilizasen" el paisaje que habían creado los primeros ingenieros de internet. Además, Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA. Era, en inglés, a patchy server (un servidor "parcheado").

* 1. VISUAL STUDIO CODE[[30]](#footnote-31)



Ilustración 21 Logo Visual Studio Code

Es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Incluye soporte para depuración, control de Git integrado, resaltado de sintaxis, finalización de código inteligente, fragmentos de código y refactorización de código. También es personalizable, de modo que los usuarios pueden cambiar el tema del editor, los métodos abreviados de teclado y las preferencias. Es gratuito y de código abierto.

**Características**

El código combina la interfaz de usuario optimizada de un editor moderno con asistencia y navegación de código enriquecido y una experiencia de depuración integrada, sin la necesidad de un IDE completo.

Visual Studio Code, cuenta con herramientas de Debug hasta opciones para actualización en tiempo real de nuestro código en la vista del navegador y compilación en vivo de los lenguajes que lo requieran (por ejemplo, en el caso de SASS a CSS).

Además de las extensiones, tendremos la posibilidad de optar por otros themes o bien configurarlo a nuestro gusto. Para modificar el esquema de colores y los iconos.

* 1. COMPOSER (SOFTWARE)[[31]](#footnote-32)

Es un administrador de dependencias para PHP que nos permite descargar paquetes desde un repositorio para agregarlo a nuestro proyecto.

Por defecto, se agregan a una carpeta llamada /vendor. De esta manera evitamos hacer las búsquedas manualmente y el mismo Composer se puede encargar de actualizar las dependencias que hayamos descargado por una nueva versión



Ilustración 22 Logo Composer

**Framework Soportados[[32]](#footnote-33)**

* Symfony versión 2 and later
* Laravel versión 4 and later
* CodeIgniter versión 3.0 and later
* CakePHP versión 3.0 and later
* FuelPHP versión 2.0 and later
* Drupal versión 8 and later
* TYPO3 versión 6.2 and later
* SilverStripe versión 3.0 later
* Magento versión 2.0 later
* Yii versión 1.1 and later
* Zend Framework versión 1 and later
* Silex (web framework)
  1. LARAVEL[[33]](#footnote-34)



Ilustración 23 Logo Laravel

Es un framework de código abierto para desarrollar aplicaciones y servicios web con PHP 5. Su objetivo es desarrollar aplicaciones con código PHP de forma elegante y simple. Fue creado en 2011 y tiene una gran influencia de frameworks como Ruby on Rails, Sinatra y ASP.NET MVC.

### Características

* Posee un poderoso conjunto de librerías.
* ORM increíble.
* Fácil enrutamiento y Sistema de ruteo, también RESTful.
* Autenticación simple.
* Blade, Motor de plantillas.
* Eloquent ORM.
* Basado en Composer.
* Soporte para el caché.
* Soporte para MVC.
* Usa componentes de Symfony.
* Adopta las especificaciones PSR-2 y PSR-4.

### Historia

En el 2011 su creador, Taylor Otwell, mostró Laravel por primera vez al mundo. Por lo que, se trata de un framework con un enfoque fresco y moderno; al ser bastante joven. Está hecho con la arquitectura MVC y resuelve necesidades actuales como manejo de eventos y autenticación de usuarios. Además, cuenta con un código modular y extensible por medio de un administrador de paquetes y un soporte robusto para manejo de bases de datos.

### Patrón MVC

Laravel propone en el desarrollo usar Routes with Closures, en lugar de un MVC tradicional con el objetivo de hacer el código más claro. Aun así, permite el uso de MVC tradicional.

**Modelo**

Laravel incluye un sistema de mapeo de datos relacional llamado Eloquent ORM que facilita la creación de modelos. Este ORM se funda en patrón active record y su funcionamiento es muy sencillo. Es opcional el uso de Eloquent, pues también dispone de otros recursos que nos facilitan interactuar con los datos, o específicamente la creación de modelos.

**Vista**

Laravel incluye de paquete un sistema de procesamiento de plantillas llamado Blade.

Este sistema de plantillas favorece un código mucho más limpio en las Vistas, además de incluir un sistema de Caché que lo hace mucho más rápido. El sistema Blade de Laravel, permite una sintaxis mucho más reducida en su escritura.

**Controlador**

Los controladores contienen la lógica de la aplicación y permiten organizar el código en clases sin tener que escribirlo todo en las rutas. Todos los controladores deben extenderse de la clase BaseController.

### Artisan CLI[[34]](#footnote-35)

Interfaz de línea de comandos (CLI por sus siglas en inglés de Command-line interface), la cual es un medio para la interacción con la aplicación donde los usuarios (en este caso los desarrolladores) dan instrucciones en forma de línea de texto simple o línea de comando. Artisan está basado en el componente Console de Symfony y nos ofrece un conjunto de comandos que nos pueden ayudar a realizar diferentes tareas durante el desarrollo e incluso cuando la aplicación se encuentra en producción.

.



**Capítulo III**

**MATERIALES Y METODOS**

1. MATERIALES Y METODOS
   1. Modelo de negocio

En el siguiente modelo de negocio se procederá a describir los procesos y actividades que se realiza dentro de la empresa.

|  |  |  |
| --- | --- | --- |
| **ID** | **PROCESO** | **DESCRIPCIÓN** |
| **P1** | Gestión venta de servicio | Se realiza la reserva con el servicio que desea del cliente.  Se asigna el personal correspondiente al servicio solicitado. |
| **P2** | Gestión de producto | Es el encargado del control del producto y también el que asigna productos al personal |

Tabla 4 Descripción de negocio

### Identificación de los usuarios que realizan el proceso

|  |  |
| --- | --- |
| **NOMBRE** | **DESCRIPCIÓN** |
| Recepcionista | Es aquella persona que verifica agenda y asigna personal al cliente. |
| Personal / Estilista | Es aquella persona que realiza el servicio al cliente. |
| Supervisor | Es el encargado del seguimiento y control de los productos además de brindar producto al personal. |
| Caja | Es la persona encargada del cobro de los servicios realizados |

Tabla 5 Descripción de Actores

* 1. Diagrama de actividad

### Diagrama de actividad: Gestión de servicio



Ilustración 24 Diagrama de actividad gestión servicio

### Diagrama de actividad: Gestión de producto

Ilustración 25 Diagrama de actividad gestión de producto

* 1. Requerimiento de Software

### Identificación de requisitos

|  |  |  |  |
| --- | --- | --- | --- |
| N° | REQUISITO | DESCRIPCIÓN | TIPO DE REQUISITO |
| 01 | Registro de servicios | Registra todos los servicios que brandara el spa. | RF |
| 02 | Registro del cliente | Registra los datos del cliente al que se le brindará el servicio. | RF |
| 03 | Inicio de sesión | El sistema debe solicitar inicio de sesión. | RF |
| 04 | Administración de usuario | Gestiona la creación, actualización y eliminación de usuarios en el sistema. | RF |
| 05 | Sistema web | El sistema debe ser accesible a través de un navegador web. | RNF |
| 06 | Sistema sin licencia | El sistema no debe requerir licencias pagadas para su uso. | RNF |
| 07 | Restricción de usuario | Restringe el acceso a ciertas funcionalidades según el rol del usuario. | RF |
| 08 | Gestión rol | Permite asignar y gestionar roles a los usuarios dentro del sistema. | RF |
| 09 | Buscar usuario por código | Permite buscar usuarios por su código único en el sistema. | RF |
| 10 | Registro de ventas | Registra las ventas realizadas en el spa. | RF |
| 12 | Control de Productos | Gestiona y controla el inventario de productos del spa. | RF |
| 13 | Sistema de fácil manejo | El sistema debe ser intuitivo y fácil de usar para los usuarios finales. | RNF |
| 14 | Gestión de personal | Permite administrar la información del personal que trabaja en el spa. | RF |
| 15 | Visualización de estadísticas | Permite visualizar estadísticas relevantes sobre el uso del sistema y el negocio. | RF |
| 17 | Control de servicios realizados | Realiza un seguimiento de los servicios que han sido completados. | RF |
| 18 | Reporte de servicio del mes | Genera un reporte de los servicios brindados en el mes. | RF |
| 19 | Control de clientes frecuentes | Identifica y gestiona a los clientes frecuentes del spa. | RF |
| 20 | Gestionar compra | Administra las compras de productos y servicios del spa. | RF |

Tabla 6 Identificación de requisitos

### Listado de requisitos funcionales

|  |  |  |
| --- | --- | --- |
| ID | NOMBRE | PRIORIDAD |
| RF01 | Registro de servicios | ALTA |
| RF02 | Registro de áreas | ALTA |
| RF03 | Registro de cliente | ALTA |
| RF04 | Inicio de sesión | ALTA |
| RF05 | Administración de usuario | ALTA |
| RF06 | Restricción de usuario | ALTA |
| RF07 | Gestión rol | ALTA |
| RF08 | Buscar usuario por código | MEDIA |
| RF09 | Registro de ventas | ALTA |
| RF10 | Registro de egresos | ALTA |
| RF11 | Control de Productos | MEDIA |
| RF12 | Gestión de personal | ALTA |
| RF13 | Visualización de estadísticas | ALTA |
| RF14 | Reporte de ventas | ALTA |
| RF15 | Control de servicios realizados | ALTA |
| RF16 | Gestión del servicio del mes | ALTA |
| RF17 | Control de clientes frecuentes | ALTA |
| RF18 | Gestionar compra | ALTA |

Tabla 7 Listado de requisitos funcionales

### Listado de requisitos no funcionales

|  |  |  |
| --- | --- | --- |
| ID | NOMBRE | PRIORIDAD |
| RNF01 | Sistema web | ALTA |
| RNF02 | Sistema sin licencia | ALTA |
| RNF03 | Sistema de fácil manejo | MEDIA |
| RNF04 | Sistema con logo de la empresa | MEDIA |

Tabla 8 Listado de requisitos no funcionales

* 1. Listado de Caso de Uso

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NOMBRE | REFERENCIA | PRIORIDAD |
| CU01 | Gestionar persona | RF03, RF05 | ALTA |
| CU02 | Gestionar privilegio | RF06 | ALTA |
| CU03 | Gestionar rol | RF07, RF08 | ALTA |
| CU04 | Gestionar usuario | RF04, RF05, RF06, RF08 | ALTA |
| CU05 | Gestionar área | RF02 | ALTA |
| CU06 | Gestionar servicio | RF01 | ALTA |
| CU07 | Gestionar personal | RF12 | ALTA |
| CU08 | Gestionar producto | RF11 | ALTA |
| CU09 | Gestionar ventas | RF09 | ALTA |
| CU10 | Gestionar proveedor | RF18 | ALTA |
| CU11 | Gestionar tipo persona | RF03 | ALTA |
| CU12 | Gestionar cliente | RF12 | ALTA |
| CU13 | Gestionar compra | RF01, RF10 | ALTA |

Tabla 9 Listado de Caso de Uso

* 1. Listado de actores del sistema

|  |  |
| --- | --- |
| ACTOR | DESCRIPCIÓN |
| Administrador del Sistema | Actor encargado de la gestión de usuario, roles y privilegios con acceso total del sistema. |
| Administrador general | Actor encargado de la gestión de proveedores y servicios en general. |
| Cajero | Actor encargado de la gestión de registro de productos, compras y ventas. |
| Supervisor | Actor encargado del control y registro de clientes y del personal |

Tabla 10 Listado de actores del sistema

* 1. Diagrama de caso de uso general

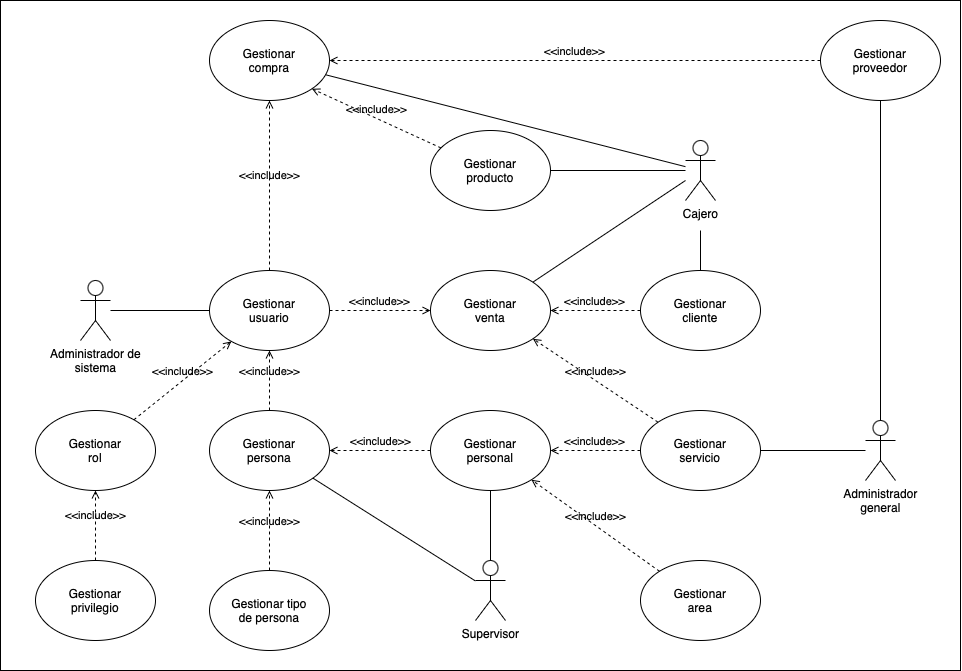


Ilustración 26 Diagrama de caso de uso general

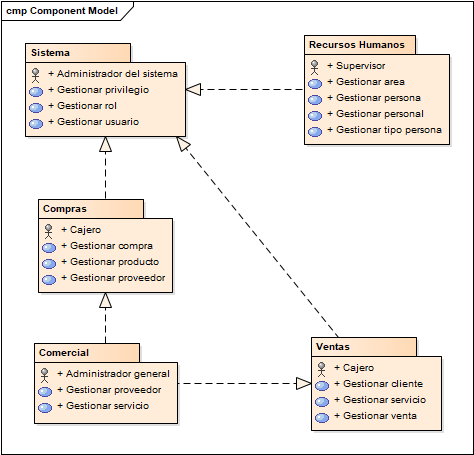
* 1. Análisis de arquitectura

Ilustración 27 Análisis de la arquitectura - Diagrama de paquete

### Modelo de paquete administrador de sistema

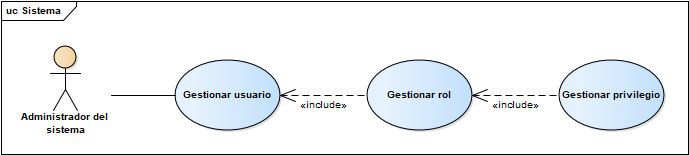


Ilustración 28 Modelo de paquete administrador de sistema

### Modelo de paquete recursos humanos

Ilustración 29 Modelo de paquete recursos humanos

### Modelo de paquete ventas

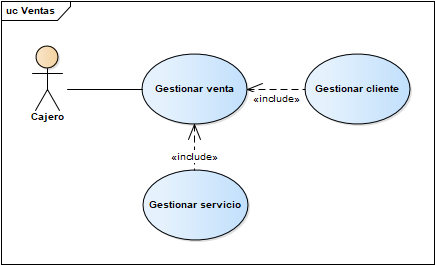


Ilustración 30 Modelo de paquete ventas

### Modelo de paquete compras

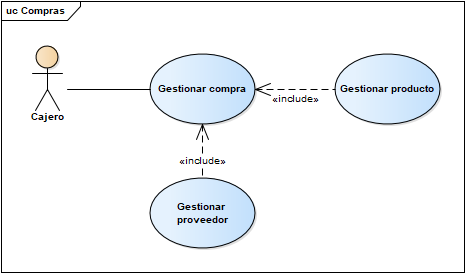


Ilustración 31 Modelo de paquete compras

### Modelo de paquete comercial

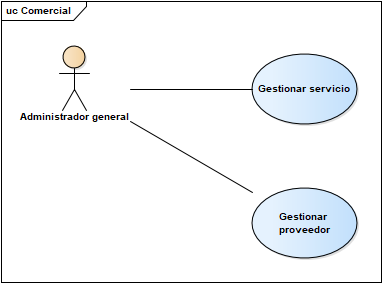


Ilustración 32 Modelo de paquete comercial

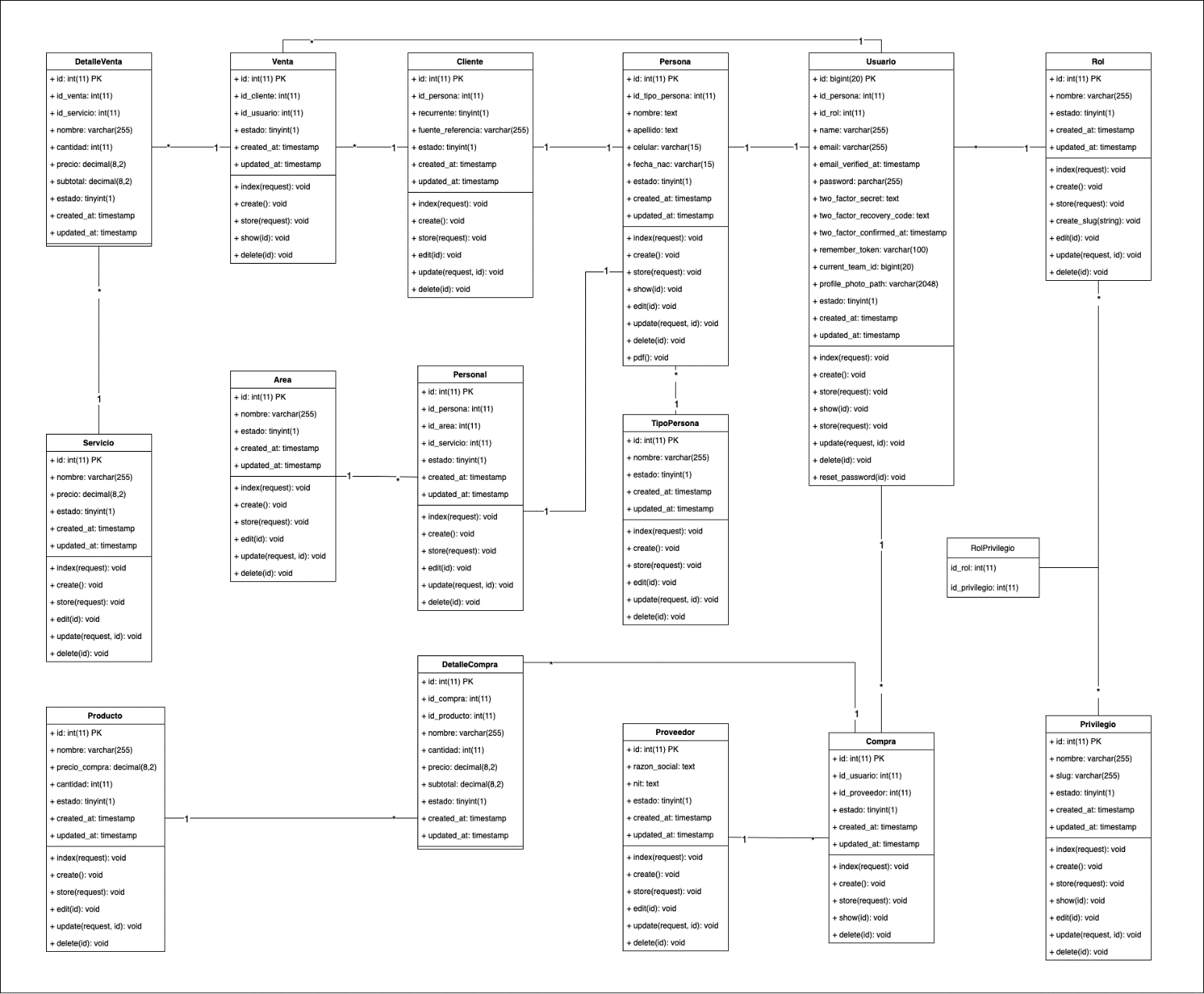
* 1.  Diagrama de clase

Ilustración 33 Diagrama de clases general

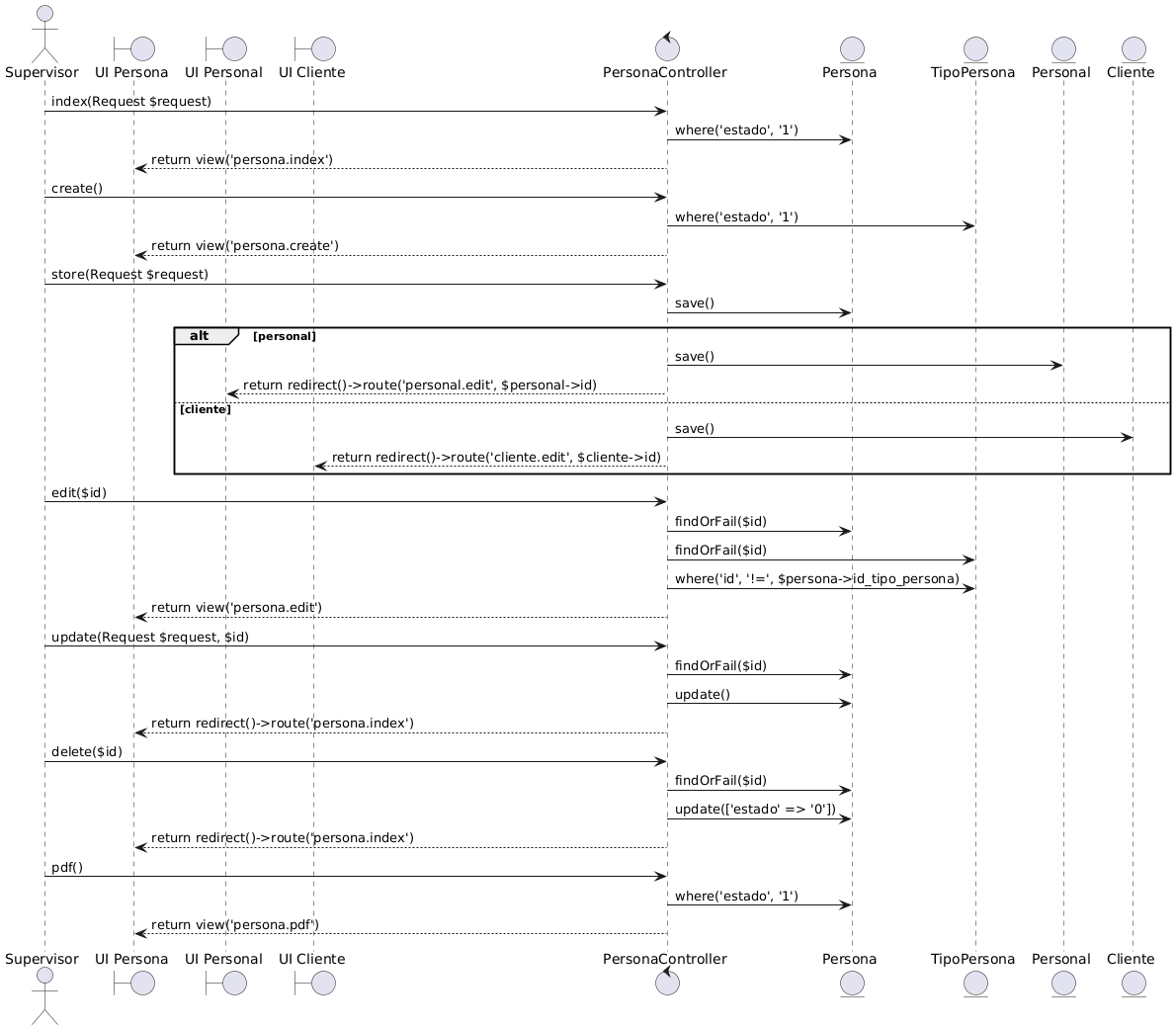
* 1.  Diagrama de secuencia gestión persona

Ilustración 34 Diagrama de secuencia gestión persona

### Descripción de caso de uso gestionar persona

|  |  |
| --- | --- |
| **Caso de uso** | **Gestionar persona** |
| **Actor** | Supervisor |
| **Activación** | Es activado por el supervisor, el cual se encargará de controlar los registros de las personas la cual se convertirán en cliente o personal. |
| **Propósito** | Tener registros de todas las personas en el  sistema. |
| **Precondiciones** | Guardar, modificar: Que el registro no exista. Eliminación: Que el registro no tenga ninguna relación al sistema. |
| **Flujo principal de eventos del actor** | El supervisor es el responsable de registrar todos los datos de la persona como también modificar y eliminar datos. También válida si los datos ingresados pertenecen a un cliente o personal. |
| **Flujo principal de eventos del sistema** | El sistema guarda, modifica y elimina los datos de las personas registradas, así también muestra los  datos. |
| **Variaciones y extensiones** | Tiene relación <<include>> con gestionar usuario. |

Tabla 11 Descripción de caso de uso gestionar persona

### Diagrama de secuencia gestión usuario

Ilustración 35 Diagrama de secuencia gestión usuario

### Descripción de caso de uso gestionar usuario

|  |  |
| --- | --- |
| **Caso de uso** | **Gestionar usuario** |
| **Actor** | Administrador de sistema |
| **Activación** | Es activado por el administrador de sistema, el cual se encargará de controlar los registros de los usuarios que accederán al sistema. |
| **Propósito** | Tener registros de todos los usuarios del sistema ya sea usuario cliente de la empresa. |
| **Precondiciones** | Guardar, Modificar: Que el registro no exista. Eliminación: Que el registro no tenga ninguna  relación. |
| **Flujo principal de eventos del actor** | El administrador registra a los usuarios asignados para la manipulación del sistema. |
| **Flujo principal de eventos del sistema** | El sistema guarda, modifica y elimina los datos de los usuarios, así también muestra los datos. |
| **Variaciones y extensiones** | Tiene relación <<include>> con gestionar roles y gestionar persona. |

Tabla 12 Descripción de caso de uso gestionar usuario

### Diagrama de secuencia gestión rol

Ilustración 36 Diagrama de secuencia gestión rol

### Descripción de caso de uso gestionar rol

|  |  |
| --- | --- |
| **Caso de uso** | **Gestionar rol** |
| **Actor** | Administrador de sistema |
| **Activación** | Es activado por el administrador de sistema el cual se registrará los diferentes roles para los distintos tipos de usuarios. |
| **Propósito** | Tener un registro de todos los roles que dispondrán los usuarios. |
| **Precondiciones** | Guardar, modificar: Que el registro no exista. Eliminación: Que el registro no tenga ninguna  relación. |
| **Flujo principal de eventos del actor** | El administrador general gestionara los roles asignados para los diferentes usuarios. |
| **Flujo principal de eventos del sistema** | El sistema guarda, modifica y elimina los datos de los roles, así también muestra los datos. |
| **Variaciones y extensiones** | Tiene relación <<include>> con gestionar usuarios |

Tabla 13 Descripción de caso de uso gestionar rol

### Diagrama de secuencia gestión personal

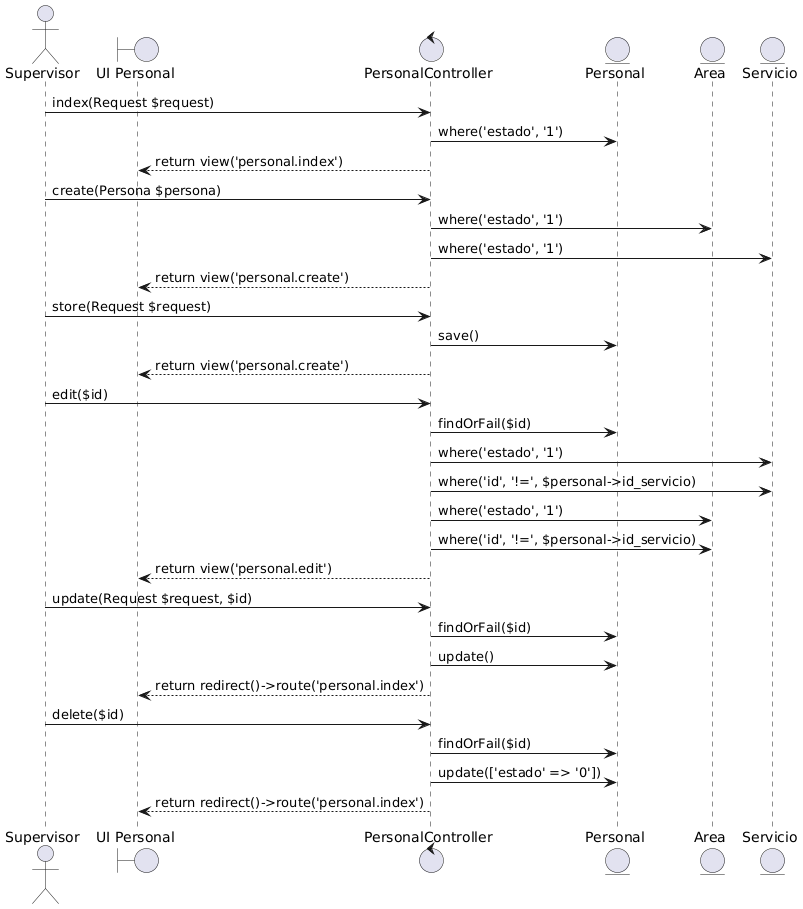


Ilustración 37 Diagrama de secuencia gestión personal

|  |  |
| --- | --- |
| **Caso de uso** | **Gestionar personal** |
| **Actor** | Supervisor |
| **Activación** | Es activado por el supervisor el cual registrara todos los datos del personal. |
| **Propósito** | Tener un registro de todo el personal. |
| **Precondiciones** | Guardar, modificar: Que el registro no exista. Eliminación: Que el registro no tenga ninguna relación. |
| **Flujo principal de eventos del actor** | El administrador registra todos sus datos al sistema para poder acceder a los distintos roles del usuario-personal. |
| **Flujo principal de eventos del sistema** | El sistema guarda, modifica y elimina los datos de los usuarios, así también muestra los datos. |
| **Variaciones y**  **extensiones** | Tiene relación <<include>> con gestionar persona. |

### Descripción de caso de uso gestionar personal

Tabla 14 Descripción de caso de uso gestionar personal

### Diagrama de secuencia gestión venta

Ilustración 38 Diagrama de secuencia gestión venta

### Descripción del caso de uso gestionar venta

|  |  |
| --- | --- |
| **Caso de uso** | **Gestionar venta** |
| **Actor** | Cajero |
| **Activación** | Es activado por el cajero de la empresa  registrando sus datos del cliente al sistema. |
| **Propósito** | Tener un registro de todas las ventas realizadas en el sistema para tener un mejor control. |
| **Precondiciones** | Guardar, modificar: Que el registro no exista. Eliminación: Que el registro no tenga ninguna relación. |
| **Flujo principal de eventos del actor** | El sistema permitirá registrar las diferentes ventas realizadas. |
| **Flujo principal de eventos del sistema** | El sistema guarda, modifica y elimina los datos. así también muestra los datos que se han registrado |
| **Variaciones y extensiones** | Tiene relación con cliente, usuario y servicio. |

Tabla 15 Descripción del caso de uso gestionar venta

### Diagrama de secuencia gestión compra

Ilustración 39 Diagrama de secuencia gestión compra

|  |  |
| --- | --- |
| **Caso de uso** | **Gestionar compra** |
| **Actor** | Cajero |
| **Activación** | Es activado por el cajero de la empresa  registrando sus datos del cliente al sistema. |
| **Propósito** | Tener un registro de todas las compras realizadas en el sistema para tener un mejor control. |
| **Precondiciones** | Guardar, modificar: Que el registro no exista. Eliminación: Que el registro no tenga ninguna relación. |
| **Flujo principal de eventos del actor** | El sistema permitirá registrar las diferentes compras realizadas. |
| **Flujo principal de eventos del sistema** | El sistema guarda, modifica y elimina los datos. así también muestra los datos que se han registrado. |
| **Variaciones y extensiones** | Tiene relación con cliente, usuario y servicio. |

### Descripción del caso de uso gestionar compra

Tabla 16 Descripción del caso de uso gestionar compra

* 1. Modelo físico de la base de datos

**Tabla Áreas**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Columna | Tipo | Nulo | Predeterminado | Comentarios |
| id (Primaria) | int(11) | No |  |  |
| nombre | varchar(255) | No |  |  |
| estado | tinyint(1) | No | 1 |  |
| created\_at | timestamp | No | current\_timestamp() |  |
| updated\_at | timestamp | No | current\_timestamp() |  |

**Tabla Clientes**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Columna | Tipo | Nulo | Predeterminado | Comentarios |
| id *(Primaria)* | int(11) | No |  |  |
| idPersona | int(11) | No |  |  |
| recurrente | tinyint(1) | No | 1 |  |
| fuente\_referencia | text | No | 'Ninguna' |  |
| estado | tinyint(1) | No | 1 |  |
| created\_at | timestamp | No | current\_timestamp() |  |
| updated\_at | timestamp | No | current\_timestamp() |  |

**Tabla Compras**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Columna | Tipo | Nulo | Predeterminado | Comentarios |
| id *(Primaria)* | int(11) | No |  |  |
| idUsuario | int(11) | No |  |  |
| idProveedor | int(11) | No |  |  |
| estado | tinyint(1) | No | 1 |  |
| created\_at | timestamp | No | current\_timestamp() |  |
| updated\_at | timestamp | No | current\_timestamp() |  |

**Tabla Detalle Compra**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Columna | Tipo | Nulo | Predeterminado | Comentarios |
| id *(Primaria)* | int(11) | No |  |  |
| idCompra | int(11) | No |  |  |
| nombre | varchar(255) | No |  |  |
| cantidad | int(11) | No |  |  |
| precio | decimal(8,2) | No |  |  |
| subtotal | decimal(8,2) | No |  |  |
| estado | tinyint(1) | No | 1 |  |
| created\_at | timestamp | No | current\_timestamp() |  |
| updated\_at | timestamp | No | current\_timestamp() |  |

**Tabla Detalle Venta**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Columna | Tipo | Nulo | Predeterminado | Comentarios |
| id *(Primaria)* | int(11) | No |  |  |
| idVenta | int(11) | No |  |  |
| nombre | varchar(255) | No |  |  |
| personal | varchar(255) | No |  |  |
| cantidad | int(11) | No |  |  |
| precio | decimal(8,2) | No |  |  |
| subtotal | decimal(8,2) | No |  |  |
| estado | tinyint(1) | No | 1 |  |
| created\_at | timestamp | No | current\_timestamp() |  |
| updated\_at | timestamp | No | current\_timestamp() |  |

**Tabla Personales**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Columna | Tipo | Nulo | Predeterminado | Comentarios |
| id (Primaria) | int(11) | No |  |  |
| idPersona | int(11) | No |  |  |
| idArea | int(11) | No |  |  |
| idServicio | int(11) | No |  |  |
| estado | tinyint(1) | No | 1 |  |
| created\_at | timestamp | No | current\_timestamp() |  |
| updated\_at | timestamp | No | current\_timestamp() |  |

**Tabla Personas**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Columna | Tipo | Nulo | Predeterminado | Comentarios |
| id (Primaria) | int(11) | No |  |  |
| idTipoPersona | int(1) | No |  |  |
| nombre | text | No |  |  |
| apellido | text | No |  |  |
| celular | varchar(15) | No |  |  |
| fecha\_nac | varchar(15) | No |  |  |
| estado | tinyint(1) | No | 1 |  |
| created\_at | timestamp | No | current\_timestamp() |  |
| updated\_at | timestamp | No | current\_timestamp() |  |

**Tabla Productos**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Columna | Tipo | Nulo | Predeterminado | Comentarios |
| id (Primaria) | int(11) | No |  |  |
| nombre | varchar(255) | No |  |  |
| estado | tinyint(1) | No | 1 |  |
| precio\_compra | decimal(8,2) | No |  |  |
| cantidad | int(11) | No |  |  |
| created\_at | timestamp | No | current\_timestamp() |  |
| updated\_at | timestamp | No | current\_timestamp() |  |

**Tabla Proveedores**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Columna | Tipo | Nulo | Predeterminado | Comentarios |
| id (Primaria) | int(11) | No |  |  |
| razon\_social | text | No |  |  |
| nit | text | No |  |  |
| estado | tinyint(1) | No | 1 |  |
| created\_at | timestamp | No | current\_timestamp() |  |
| updated\_at | timestamp | No | current\_timestamp() |  |

**Tablas Roles**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Columna | Tipo | Nulo | Predeterminado | Comentarios |
| id (Primaria) | int(11) | No |  |  |
| nombre | varchar(255) | No |  |  |
| estado | tinyint(1) | No | 1 |  |
| created\_at | timestamp | No | current\_timestamp() |  |
| updated\_at | timestamp | No | current\_timestamp() |  |

**Tabla Servicios**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Columna | Tipo | Nulo | Predeterminado | Comentarios |
| id (Primaria) | int(11) | No |  |  |
| nombre | varchar(255) | No |  |  |
| precio | decimal(8,2) | No |  |  |
| estado | tinyint(1) | No | 1 |  |
| created\_at | timestamp | No | current\_timestamp() |  |
| updated\_at | timestamp | No | current\_timestamp() |  |

**Tabla Privilegios**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Columna | Tipo | Nulo | Predeterminado | Comentarios |
| id (Primaria) | int(11) | No |  |  |
| nombre | varchar(255) | No |  |  |
| slug | varchar(225) | No |  |  |
| estado | tinyint(1) | No | 1 |  |
| created\_at | timestamp | No | current\_timestamp() |  |
| updated\_at | timestamp | No | current\_timestamp() |  |

**Tabla Tipo Personas**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Columna | Tipo | Nulo | Predeterminado | Comentarios |
| id (Primaria) | int(11) | No |  |  |
| nombre | varchar(255) | No |  |  |
| estado | tinyint(1) | No | 1 |  |
| created\_at | timestamp | No | current\_timestamp() |  |
| update\_at | timestamp | No | current\_timestamp() |  |

**Tabla Usuarios**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Columna | Tipo | Nulo | Predeterminado | Comentarios |
| id (Primaria) | bigint(20) | No |  |  |
| idPersona | int(11) | No |  |  |
| idRol | int(11) | No |  |  |
| name | varchar(255) | No |  |  |
| email | varchar(255) | No |  |  |
| email\_verified\_at | timestamp | Sí | NULL |  |
| password | varchar(255) | No |  |  |
| two\_factor\_secret | text | Sí | NULL |  |
| two\_factor\_recovery\_codes | text | Sí | NULL |  |
| two\_factor\_confirmed\_at | timestamp | Sí | NULL |  |
| remember\_token | varchar(100) | Sí | NULL |  |
| current\_team\_id | bigint(20) | Sí | NULL |  |
| profile\_photo\_path | varchar(2048) | Sí | NULL |  |
| estado | tinyint(1) | No | 1 |  |
| created\_at | timestamp | Sí | NULL |  |
| updated\_at | timestamp | Sí | NULL |  |

**Tabla Ventas**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Columna | Tipo | Nulo | Predeterminado | Comentarios |
| id (Primaria) | int(11) | No |  |  |
| idUsuario | int(11) | No |  |  |
| idCliente | int(11) | No |  |  |
| estado | tinyint(1) | No | 1 |  |
| created\_at | timestamp | No | current\_timestamp() |  |
| updated\_at | timestamp | No | current\_timestamp() |  |

### Diagrama de despliegue

Ilustración 40 Diagrama de despliegue



**Capítulo IV**

**PRUEBAS DEL SISTEMA**

1. MODELO DE PRUEBA
   1. Concepto de prueba de software

Las pruebas son básicamente un conjunto de actividades dentro del desarrollo de software. Dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento de dicho proceso de desarrollo.

La prueba es una técnica de ejecución de un programa con la intención de mostrar un error no descubierto hasta ese momento.

* Pruebas estáticas

Son el tipo de pruebas que se realizan sin ejecutar el código de la aplicación.

Puede referirse a la revisión de documentos, ya que no se hace una ejecución de código. Esto se debe a que se pueden realizar "pruebas de escritorio" con el objetivo de seguir los flujos de la aplicación.

* Pruebas dinámicas

Todas aquellas pruebas que para su ejecución requieren la ejecución de la aplicación.

Las pruebas dinámicas permiten el uso de técnicas de caja negra y caja blanca con mayor amplitud. Debido a la naturaleza dinámica de la ejecución de pruebas es posible medir con mayor precisión el comportamiento de la aplicación desarrollada.

a) Pruebas de caja Blanca o pruebas estructurales

Este tipo de prueba se centra en los detalles procedimentales del software, por lo que su diseño está fuertemente ligado al código fuente. El ingeniero de pruebas escoge distintos valores de entrada para examinar cada uno de los posibles flujos de ejecución del programa y cerciorarse de que se devuelven los valores de salida adecuados.

b) Pruebas de caja Negra

Se denomina caja negra a aquel elemento que es estudiado desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno.

En otras palabras, de una caja negra nos interesará su forma de interactuar con el medio que le rodea, entendiendo qué es lo que hace, pero sin dar importancia a cómo lo hace.

* 1. Diseño de Casos de Prueba

### Prueba de caso de uso del sistema

A continuación de muestra capturas del sistema, donde se aplicará la técnica de prueba de la caja negra donde se aprecia las interfaces para su prueba.

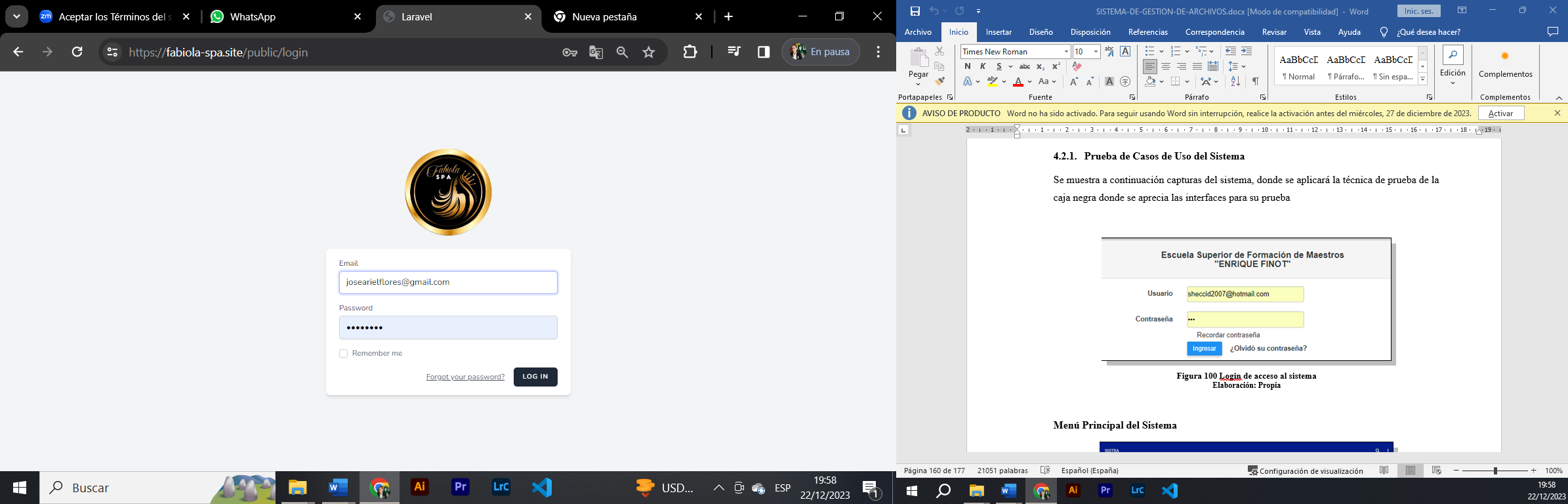


Ilustración 41 Login de acceso al sistema Elaboración: Propia

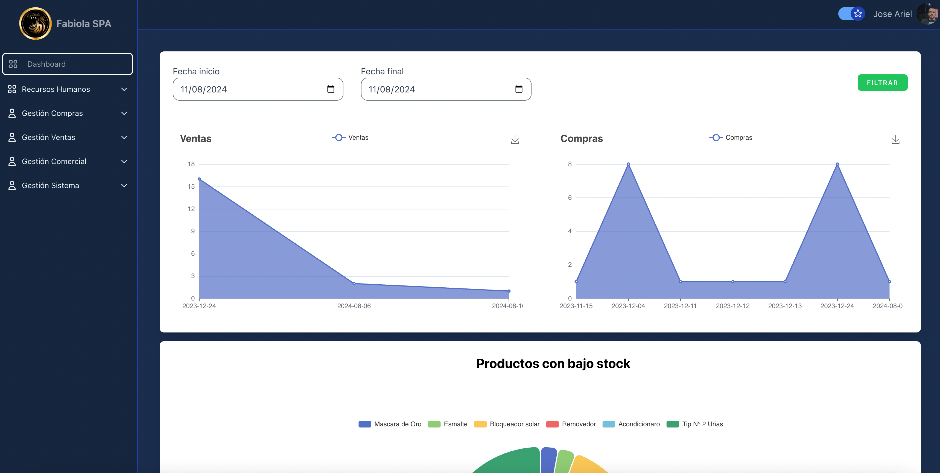
**Menú Principal del Sistema**

Ilustración 42 Menú Principal del Sistema Elaboración: Propia

### Descripción del caso de uso Prueba del sistema

### Caso de Uso Gestionar Persona

**Caso de Prueba :** Gestionar Personas

**Actores :** Supervisor

**Tipo :** Primario

**Resumen :** El supervisor gestiona datos de las personas que forman parte de la empresa Fabiola Spa que se encuentra en la ciudad de Santa Cruz de la Sierra.

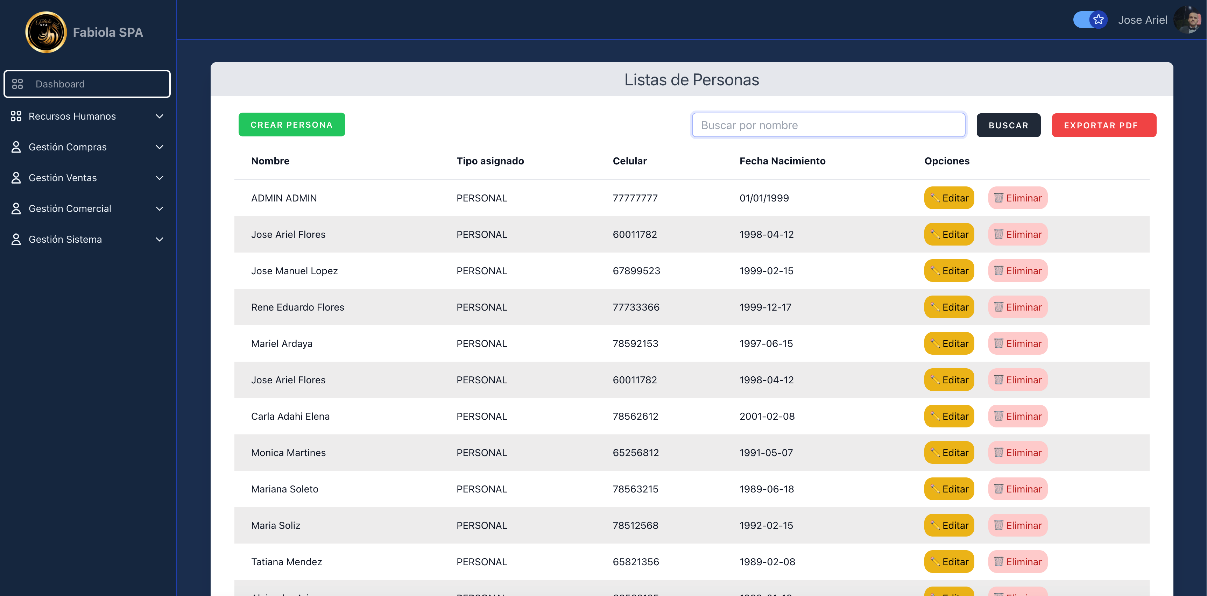
**Interfaz 1.-** Caso de Uso Gestionar Persona

Ilustración 43 Gestionar Personas Elaboración: Propia

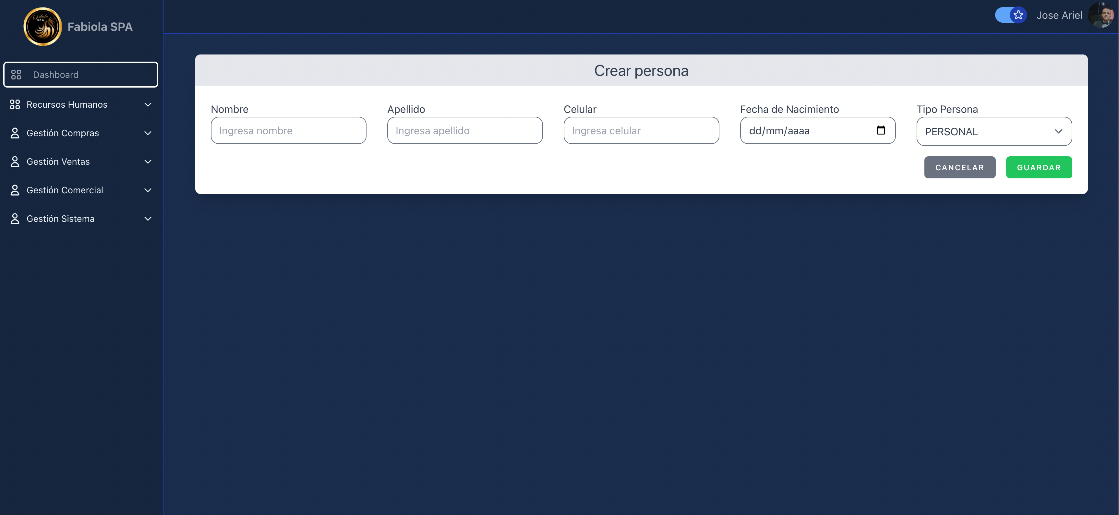
**Interfaz 2.-** Caso de Uso Registrar Persona

Ilustración 44 Gestionar Persona Elaboración: Propia

### Caso de Uso Gestionar Persona

Para evaluar y probar la funcionalidad del caso de uso “Gestionar Persona” se describe un escenario del caso de prueba, a continuación, se establece el siguiente escenario:

|  |  |
| --- | --- |
| **Caso de Prueba** | **Registro de Persona** |
| Entrada | nombre = “Ana Carla”  apellido = “Medina”  celular = “69115678”;  fecha\_nac = “16’-07-1998”;  id\_tipo\_persona = 2 |
| Resultado | * El id es autoincrementado. * Se realiza la inserción correspondiente del registro en la tabla tbl\_cliente o tbl\_personal de acuerdo al tipo de persona. |

**Flujo de Eventos**

|  |  |
| --- | --- |
| **FLUJO DE LOS EVENTOS GESTIONAR PERSONAS** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El supervisor luego de haber ingresado con su usuario y contraseña, despliega el menú principal del sistema, y en el menú selecciona personal. | 1. El sistema muestra el listado de los perdonas registrados, donde nos muestra:    1. Opciones    2. Buscar    3. Crear Persona |
| **OPCIONES**  **3.** Muestra la interfaz 1, con las opciones (Editar y eliminar)  **BUSCAR**  **5.** Muestra la interfaz 1, con opciones de búsqueda por apellido y nombre de la persona  **NUEVO REGISTRO**  Muestra la interfaz 2, para crear un nuevo registro de Persona.  **9.** Se presiona en el botón Guardar.  **11.** Si no se desean guardar los datos, se vuelve hacia atrás con el ícono que se muestra en la parte superior de la interfaz 1. | **4.** El sistema permite realizar las opciones de:  Botón Editar, que nos permite modificar los datos de la persona.  Botón Eliminar, que nos permite dar de baja a la persona.  **6.** El sistema permite realizar búsquedas con las opciones de ingresar Apellido y Nombre de la Persona a través del botón Buscar y muestra los resultados.  **8.** El sistema activa todos los controles necesarios para realizar un nuevo registro.   * Ingresa el nombre de la persona * Ingresa el apellido de la persona * Ingresa el número de celular de la persona. * Selecciona la fecha de nacimiento de la persona. * Selecciona el tipo de persona que que pertenece la persona.   **10.** Se guardan los datos del registro, y se muestra en la interfaz 1, los nuevos datos guardados.  **12.** El sistema retorna a la interfaz 1 deshabilitando los controles y sin mostrar los datos no guardados. |

### Descripción del Caso de Uso Gestionar Venta

### Caso de Uso Gestionar Venta

**Caso de Prueba :** Gestionar Ventas

**Actores :** Caja

**Tipo :** Primario

**Resumen :** El encargado de Caja gestiona datos las ventas de servicios a los clientes de la empresa Fabiola Spa que se encuentra en la ciudad de Santa Cruz de la Sierra.

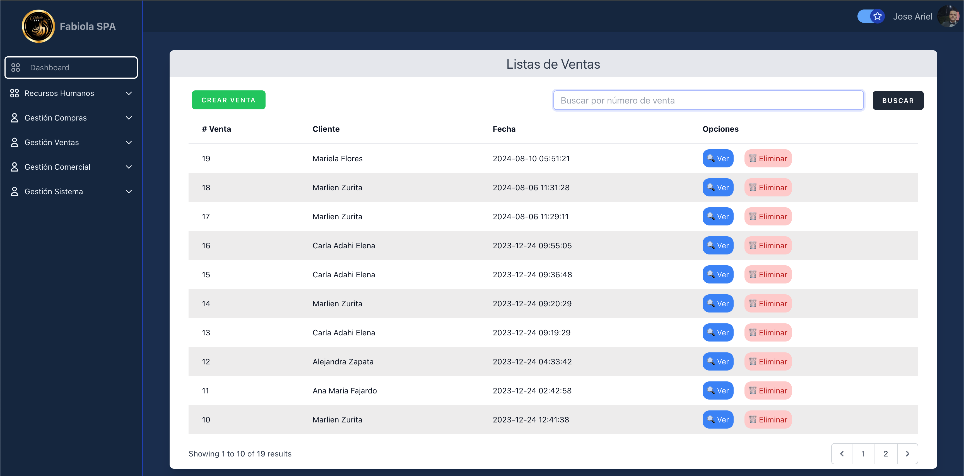
**Interfaz 1.-** Caso de Uso Gestionar venta

Ilustración 45 Gestionar Venta

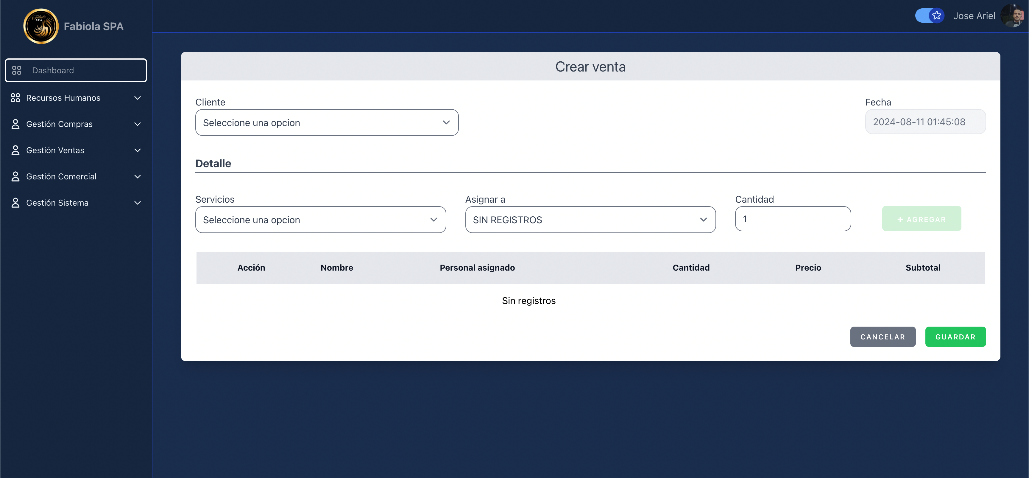
**Interfaz 2.-** Caso de Uso Registrar Venta

Ilustración 46 Registrar Venta

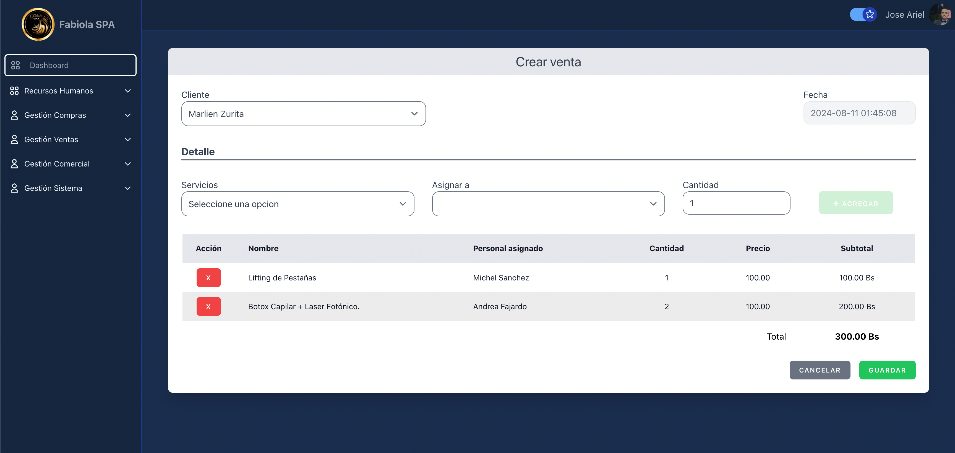
**Interfaz 2.-** Caso de Uso Agregar Venta

Ilustración 47 Agregar Venta

### Caso de Uso Gestionar Venta

Para comprobar y evaluar la funcionalidad del caso de uso “Gestionar Venta”, se describe un escenario particular del caso de prueba, a continuación, se detalla.

|  |  |
| --- | --- |
| **Caso de Prueba** | **Registro de Trámite** |
| Entrada | id\_usuario = 1  id\_cliente = 2  created\_at = “11-02-2024” |
| Resultado | * El id es autoincrementado. * El id\_cliente es llave foránea de la tabla tbl\_cliente donde se hereda los datos del cliente a la Venta. * El id\_usuario es llave foránea de la tabla tbl\_usuario donde se hereda el usuario que realiza la venta. * La id\_venta, nombre, id\_personal, cantidad, precio, subtotal se agrega al detalle venta. |

### Flujos de Eventos

|  |  |
| --- | --- |
| **FLUJO DE LOS EVENTOS GESTIONAR TRÁMITE** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El Cajero luego de haber ingresado con su usuario y contraseña, despliega el menú principal del sistema, y en el menú selecciona **Gestionar ventas**.   **OPCIONES**   1. Muestra la interfaz 1, con la opción VER   **BUSCAR**   1. Muestra la interfaz 1, con opciones de búsqueda por número de venta. | 1. El sistema muestra el listado de los trámites registrados, donde nos muestra:    1. Opciones    2. Buscar    3. Nuevo Registro 2. El sistema permite realizar opción de:   Botón **Ver**, que nos muestra el detalle de las ventas realizadas.   1. El sistema permite realizar búsquedas con las opciones de ingresar número de ventas a través del botón **Buscar** y muestra los resultados. |

|  |  |
| --- | --- |
| **NUEVO REGISTRO**  **7.** Muestra la interfaz 1, para crear un nuevo registro de venta.  **9.** Ingresa el número de venta para su búsqueda, muestra la interfaz 2.  **11**. Selecciona la venta, interfaz 2.  **13**. Selecciona el cliente, servicios y la asignación, interfaz 2.  **15.** Muestra la interfaz 2, se presiona en el botón Guardar.  **17.** Si no se desean guardar los datos, se presiona el botón CANCELAR Interfaz 2. | **8**. El sistema activa todos los controles necesarios para realizar un nuevo registro.  **10**. Busca y tiene la opción **botón**  seleccionar, Interfaz 2.  **12**. Muestra el formulario para registrar una nueva venta con sus opciones disponible, interfaz 2.  **14**. Muestra lo agregado por el usuario según el tipo de servicio o y la cantidad, donde se calcula el servicio con la cantidad y la sumatoria del total, interfaz 2.  **16.** Se guardan los datos del registro, y se muestra en la interfaz 1, los nuevos datos guardados.  **18.** El sistema retorna a la interfaz 1 deshabilitando los controles y sin mostrar los datos no guardados. |

**CONCLUSIONES**

* Se obtuvieron los requerimientos funcionales y no funcionales relacionados con la información de la empresa.
* Se diseñó un modelo que se adapta a los requerimientos de información de la empresa.
* Se desarrolló el modelo de requerimientos del sistema.
* Se documentaron los modelos de análisis, diseño e implementación de acuerdo con PUDS.
* Se realizaron pruebas exhaustivas para verificar la funcionalidad, seguridad y usabilidad del sistema antes de su implementación completa.
* Se evaluó la aceptación y eficacia del sistema por parte del personal y los clientes, recopilando retroalimentación y realizando los ajustes necesarios.
* El sistema desarrollado optimiza los procesos internos de la empresa, lo que contribuye a mejorar la eficiencia operativa y la satisfacción del cliente.
* La flexibilidad del sistema permite su adaptación a futuros cambios en los requisitos de la empresa, garantizando su sostenibilidad a largo plazo.
* Basado en el estudio de los requerimientos de software y operaciones de la empresa, el software desarrollado facilita la implementación de mejoras o la adición de nuevos módulos según las necesidades de la empresa.
* Se recomienda realizar respaldos diarios de la base de datos, incluyendo copias en la nube, para asegurar la integridad de la información ante cualquier eventualidad ajena a las operaciones diarias.
* Se sugiere utilizar un UPS en el equipo donde se instalará el sistema, para disponer de tiempo suficiente para guardar la información y evitar la pérdida de datos en caso de un corte de energía.

**BIBLIOGRAFÍA**

* (s.f.). Obtenido de UML: https://www.ecured.cu/UML
* *Ajax*. (s.f.). Obtenido de https://www.maestrosdelweb.com/ajax/
* *Apache*. (s.f.). Obtenido de https://www.ecured.cu/Servidor\_HTTP\_Apache
* *Architec Enterprise*. (s.f.). Obtenido de http://www.sparxsystems.com.ar/products/ea/editions.php
* *Arquiectura MVC*. (s.f.). Obtenido de https://www.troopsf.com/mvc/#:~:text=Modelo%20Vista%20Controlador%20(MVC)%20es,control%20en%20tres%20componentes%20distintos
* *Artisan CLI*. (s.f.). Obtenido de https://styde.net/artisan-interfaz-linea-comandos-de-laravel
* *Casos de usos*. (s.f.). Obtenido de https://unadzsurlab.com/UML/U1/diagramas\_de\_casos\_de\_uso.html
* *Composer*. (s.f.). Obtenido de http://anexsoft.com/que-es-composer-y-como-se-usa-en-php
* *CSS3*. (s.f.). Obtenido de https://www.ecured.cu/CSS3
* *DDL y DML*. (s.f.). Obtenido de http://www.edu4java.com/es/sql/sql4.html
* *Diagrama de actividad*. (s.f.). Obtenido de https://www.ibiblio.org/pub/linux/docs/LuCaS/Tutoriales/doc-modelado-sistemas-UML/multiple-html/x291.html
* *Diagrama de actividad UML*. (s.f.). Obtenido de https://unadzsurlab.com/UML/U1/diagramas\_de\_actividades.html
* *Diagrama de clase*. (s.f.). Obtenido de https://www.ecured.cu/Diagrama\_de\_Clase
* *Diagrama de colaboracion*. (s.f.). Obtenido de https://manuel.cillero.es/doc/metodologia/metrica-3/tecnicas/diagrama-de-interaccion/diagrama-de-colaboracion/
* *Diagrama de despliegue*. (s.f.). Obtenido de https://www.ecured.cu/Diagrama\_de\_despliegue
* *Diagrama de paquetes*. (s.f.). Obtenido de https://manuel.cillero.es/doc/metodologia/metrica-3/tecnicas/diagrama-de-paquetes/
* *Diagrama de secuencia*. (s.f.). Obtenido de https://www.lucidchart.com/pages/es/diagrama-de-secuencia
* *FontAwesome*. (s.f.). Obtenido de https://www.ionos.es/digitalguide/paginas-web/diseno-web/como-se-pueden-integrar-icon-fonts/
* *Historia HTML*. (s.f.). Obtenido de https://uniwebsidad.com/libros/xhtml/capitulo-1/breve-historia-de-html
* *Html*. (s.f.). Obtenido de https://developer.mozilla.org/es/docs/Learn/Getting\_started\_with\_the\_web/HTML\_basics
* HTML5. (s.f.). Obtenido de https://developer.mozilla.org/es/docs/Web/HTML
* *JavaScript*. (s.f.). Obtenido de https://www.maestrosdelweb.com/que-es-javascript/
* *Laravel*. (s.f.). Obtenido de https://www.ecured.cu/Laravel#Caracter.C3.ADsticas
* *MariaDB*. (s.f.). Obtenido de https://www.drauta.com/que-es-mariadb
* *MySQL*. (s.f.). Obtenido de https://www.alegsa.com.ar/Dic/mysql.php#gsc.tab=0
* *PHP*. (s.f.). Obtenido de https://www.nextu.com/blog/que-es-html-rc22/
* *Proceso unificado de desarrollo de software*. (s.f.). Obtenido de https://software-b.com/sb25/soporte/proceso-unificado.html
* *Soporte composer*. (s.f.). Obtenido de https://en.wikipedia.org/wiki/Composer\_(software)
* *StarUml*. (s.f.). Obtenido de http://codigoprogramacion.com/articulos/staruml
* *TailwindCSS*. (s.f.). Obtenido de https://tailwindcss.com/
* *Tecnologia y arquitectura.* (s.f.). Obtenido de https://www.editdiazdesantos.com/wwwdat/pdf/9788479787066.pdf
* *Ventajas SQL*. (s.f.). Obtenido de http://www.tuprogramacion.com/glosario/que-es-mysql/
* *VS Code*. (s.f.). Obtenido de https://www.ecured.cu/Visual\_Studio\_Code
* *Xampp*. (s.f.). Obtenido de https://www.ionos.es/digitalguide/servidores/herramientas/instala-tu-servidor-local-xampp-en-unos-pocos-pasos/

1. https://www.ecured.cu/UML [↑](#footnote-ref-2)
2. https://unadzsurlab.com/UML/U1/diagramas\_de\_casos\_de\_uso.html [↑](#footnote-ref-3)
3. https://www.ibiblio.org/pub/linux/docs/LuCaS/Tutoriales/doc-modelado-sistemas-UML/multiple-html/x291.html [↑](#footnote-ref-4)
4. <https://unadzsurlab.com/UML/U1/diagramas_de_actividades.html> [↑](#footnote-ref-5)
5. <https://www.ecured.cu/Diagrama_de_Clase> [↑](#footnote-ref-6)
6. <https://manuel.cillero.es/doc/metodologia/metrica-3/tecnicas/diagrama-de-paquetes/> [↑](#footnote-ref-7)
7. <https://www.lucidchart.com/pages/es/diagrama-de-secuencia> [↑](#footnote-ref-8)
8. <https://manuel.cillero.es/doc/metodologia/metrica-3/tecnicas/diagrama-de-interaccion/diagrama-de-colaboracion/> [↑](#footnote-ref-9)
9. <https://www.ecured.cu/Diagrama_de_despliegue> [↑](#footnote-ref-10)
10. <https://software-b.com/sb25/soporte/proceso-unificado.html> [↑](#footnote-ref-11)
11. <http://www.sparxsystems.com.ar/products/ea/editions.php> [↑](#footnote-ref-12)
12. <http://codigoprogramacion.com/articulos/staruml> [↑](#footnote-ref-13)
13. <https://www.editdiazdesantos.com/wwwdat/pdf/9788479787066.pdf> [↑](#footnote-ref-14)
14. <https://www.troopsf.com/mvc/#:~:text=Modelo%20Vista%20Controlador%20(MVC)%20es,control%20en%20tres%20componentes%20distintos> [↑](#footnote-ref-15)
15. <https://developer.mozilla.org/es/docs/Web/HTML> [↑](#footnote-ref-16)
16. <https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/HTML_basics> [↑](#footnote-ref-17)
17. <https://uniwebsidad.com/libros/xhtml/capitulo-1/breve-historia-de-html> [↑](#footnote-ref-18)
18. <https://www.nextu.com/blog/que-es-html-rc22/> [↑](#footnote-ref-19)
19. <https://www.ecured.cu/CSS3> [↑](#footnote-ref-20)
20. <https://www.maestrosdelweb.com/que-es-javascript/> [↑](#footnote-ref-21)
21. <https://www.maestrosdelweb.com/ajax/> [↑](#footnote-ref-22)
22. <https://tailwindcss.com/> [↑](#footnote-ref-23)
23. <https://www.ionos.es/digitalguide/paginas-web/diseno-web/como-se-pueden-integrar-icon-fonts/> [↑](#footnote-ref-24)
24. <https://www.alegsa.com.ar/Dic/mysql.php#gsc.tab=0> [↑](#footnote-ref-25)
25. <http://www.tuprogramacion.com/glosario/que-es-mysql/> [↑](#footnote-ref-26)
26. <https://www.drauta.com/que-es-mariadb> [↑](#footnote-ref-27)
27. <http://www.edu4java.com/es/sql/sql4.html> [↑](#footnote-ref-28)
28. <https://www.ionos.es/digitalguide/servidores/herramientas/instala-tu-servidor-local-xampp-en-unos-pocos-pasos/> [↑](#footnote-ref-29)
29. <https://www.ecured.cu/Servidor_HTTP_Apache> [↑](#footnote-ref-30)
30. <https://www.ecured.cu/Visual_Studio_Code> [↑](#footnote-ref-31)
31. <http://anexsoft.com/que-es-composer-y-como-se-usa-en-php> [↑](#footnote-ref-32)
32. <https://en.wikipedia.org/wiki/Composer_(software)> [↑](#footnote-ref-33)
33. <https://www.ecured.cu/Laravel#Caracter.C3.ADsticas> [↑](#footnote-ref-34)
34. <https://styde.net/artisan-interfaz-linea-comandos-de-laravel> [↑](#footnote-ref-35)