

**UNIVERSIDAD ADVENTISTA DE BOLIVIA**  
**FACULTAD DE INGENIERÍA**  
**INGENIERÍA DE SISTEMAS**

**SISTEMA WEB DE GESTIÓN DE PASANTÍAS PARA  
ESTUDIANTES DEL INSTITUTO TECNOLÓGICO "ESCUELA  
INDUSTRIAL SUPERIOR PEDRO DOMINGO MURILLO"**

**CASO DE ESTUDIO**

Presentado como requisito para obtener el grado académico de

Licenciado en Ingeniería de Sistemas

Por:

**Juan Carlos Mendoza Gutiérrez**

Vinto, 2024

**DEDICATORIA:**

A todos Los Que Yo Quiero Dedicarles

## **AGRADECIMIENTOS:**

A todas las instituciones y personas que me apoyaron

## **RESUMEN**

La Escuela Industrial Superior Pedro Domingo Murillo, proporciona pasantías a sus estudiantes en diferentes empresas, las cuales requieren de un estricto seguimiento del proceso de inserción y desvinculación, además de la retroalimentación brindada por la misma al instituto. Este proceso, realizado de forma manual y con planillas excel, presenta muchas omisiones y errores, por lo que se ve por conveniente la realización de un sistema web para la gestión de las prácticas.

Se tomó la metodología Scrum para el desarrollo del sistema, sobre un patrón arquitectónico MVC, utilizando el lenguaje de programación PHP, manejo del framework BOOTSTRAP, CSS, JAVASCRIPT y como base de datos MySQL, se hicieron pruebas unitarias y de aceptación.

Como resultado de este trabajo, se presentó la el primer sprint completo, que consiste en el acceso al sistema, además se tiene el modelado de la base de datos completo y la planificación del resto de sprints, así como un prototipo no funcional del sistema completo

**Palabras clave:** Pasantía, Scrum, Sistema Web



## ÍNDICE

<b>INTRODUCCIÓN</b>	<b>1</b>
<b>CAPÍTULO 1</b>	<b>2</b>
<b>EL PROBLEMA</b>	<b>2</b>
<b>    1.1. ANTECEDENTES</b>	<b>2</b>
<b>    1.2. PROBLEMA</b>	<b>3</b>
1.2.1. Situación Problemática	3
1.2.1. Formulación del problema	3
<b>    1.3. OBJETIVOS</b>	<b>4</b>
1.3.1. Objetivo general	4
1.3.2. Objetivos específicos	4
<b>    1.4. ALCANCES</b>	<b>4</b>
1.4.1. Alcance del proyecto	4
1.4.2. Alcance del producto	5
<b>    1.5. LÍMITES</b>	<b>5</b>
1.5.1. Alcance Geográfico:	5
1.5.2. Usuarios Internos y Externos:	5
1.5.3. Integración con Sistemas Externos:	5
1.5.4. Escalabilidad:	5
1.5.5. Idioma y Localización:	6
1.5.6. Costos y Recursos:	6
1.5.7. Seguridad de Datos:	6

1.5.8. Plazos de Implementación:	6
1.5.9. Aprobación y Validación:	6
<b>1.6. JUSTIFICACIÓN</b>	<b>7</b>
1.6.1. Justificación Social:	7
1.6.2. Justificación Técnica	7
<b>1.7. ESTUDIO DE FACTIBILIDAD</b>	<b>8</b>
1.7.1. Factibilidad Técnica:	8
1.7.2. Factibilidad económica:	9
1.7.3. Factibilidad operacional:	9
<b>CAPÍTULO 2</b>	<b>11</b>
<b>MARCO TEÓRICO</b>	<b>11</b>
<b>2.1 INTRODUCCIÓN</b>	<b>11</b>
<b>2.2 CONCEPTO DE PASANTÍAS</b>	<b>11</b>
2.2.1 OBJETIVOS DE LAS PASANTÍAS	11
<b>2.3. METODOLOGÍA.</b>	<b>12</b>
2.3.1. MÉTODO Y TÉCNICAS DE INVESTIGACIÓN.	12
2.3.1.1. METODO ANALITICO	12
2.3.1.2. Encuesta.	12
2.3.1.3. Entrevista.	12
2.3.1.4. Referencia o Bibliográfica.	13
<b>2.4 INGENIERÍA DE SOFTWARE.</b>	<b>13</b>
<b>2.5. MODELOS DE PROCESO DE SOFTWARE</b>	<b>15</b>
<b>2.6. PROCESO UNIFICADO.</b>	<b>15</b>

<b>2.7 METODOLOGÍAS</b>	<b>18</b>
<b>2.7.1. METODOLOGÍA UWE</b>	<b>18</b>
<b>2.7.2. METODOLOGÍA DE DESARROLLO ÁGIL</b>	<b>19</b>
2.7.2.1. METODOLOGÍA DE DESARROLLO SCRUM	20
2.7.2.2. CARACTERÍSTICAS	21
2.7.2.3. PRINCIPIOS BÁSICOS	21
2.7.2.4. PRÁCTICAS DE SCRUM	22
2.7.2.5. ROLES DE SCRUM	24
2.7.2.6. EVENTOS DE SCRUM	24
2.7.2.6.1. EL SPRINT	25
2.7.2.6.2. PLANIFICACIÓN DE SPRINT	25
2.7.2.6.3. SCRUM DIARIO	26
2.7.2.6.3. REVISIÓN DE SPRINT	27
2.7.2.6.4. RETROSPECTIVA DEL SPRINT	27
2.7.2.6.5. ARTEFACTOS DE LA METODOLOGÍA SCRUM	27
2.7.2.6.6. PRODUCT BACKLOG	27
2.7.2.6.7. SPRINT BACKLOG	27
2.7.2.6.8. INCREMENTO	27
2.7.2.7. FASES DE SCRUM	28
2.7.2.7.1. PREGAME	28
2.7.2.7.2. DEVELOPMENT	30
2.7.2.7.3. POSTGAME	30
2.7.2.7.4. REUNIONES DE TRABAJO EN UN CONTEXTO SCRUM	31

<b>2.8 MODELADO DE NEGOCIO</b>	<b>31</b>
2.8.1. DIAGRAMAS UML	31
2.8.2. DIAGRAMA DE CASOS DE USO	31
2.8.3 DIAGRAMA DE SECUENCIA	32
2.8.4. DIAGRAMA DE CLASES	33
2.8.5. DIAGRAMA DE COLABORACIÓN	33
2.8.6. DISEÑO DE BASE DE DATOS	34
2.8.7. MODELO RELACIONAL	34
<b>2.9. HERRAMIENTAS DEL DESARROLLO DEL SOFTWARE</b>	<b>34</b>
2.9.1 FRAMEWORK	34
2.9.1.1. Tipos de Frameworks:	35
2.9.1.2. Ventajas de Utilizar Frameworks:	35
2.9.1.3. Desventajas de Utilizar Frameworks:	36
<b>2.10. INGENIERÍA DE REQUERIMIENTOS.</b>	<b>36</b>
<b>2.11. BASE DE DATOS</b>	<b>37</b>
2.11.1. BASES DE DATOS RELACIONALES:	39
2.11.2. MySQL	40
2.11.3. Ediciones de MySQL:	41
2.11.4. Casos de Uso de MySQL:	42
2.11.5. Empresas y Organizaciones que Utilizan MySQL:	42
2.11.6. Versiones de MySQL:	42
<b>2.12. ARQUITECTURA DEL SISTEMA WEB</b>	<b>42</b>
2.12.1. Lógica de negocio.	44

2.12.2. Administración de los datos.	44
2.12.3. Interfaz	44
2.12.4. Modelo de dos capas:	44
2.12.5. Modelo de n-capas:	45
<b>2.13. INTRODUCCIÓN A LA APLICACIONES WEB</b>	<b>45</b>
2.13.1. DEFINICIÓN DE APLICACIONES WEB	45
2.13.2. TCP / IP	45
2.13.3. INTERFACES	46
2.13.4. SERVICIOS WEB	46
<b>2.14. PRINCIPIOS DE FUNCIONAMIENTO DE LOS SERVICIOS WEB</b>	<b>47</b>
2.14.1 HTML 5	47
2.14.2. CSS	48
2.14.3. Formatos CSS	48
<b>2.15. BOOTSTRAP 5</b>	<b>48</b>
<b>2.16. LARAVEL</b>	<b>49</b>
2.16.1 CARACTERÍSTICAS	50
<b>2.17. SWEET ALERT</b>	<b>51</b>
<b>2.18. AJAX</b>	<b>51</b>
<b>2.19. JSON</b>	<b>52</b>
<b>2.20. MOCKUPS</b>	<b>52</b>
<b>2.21. PHP</b>	<b>53</b>
2.21.1. Sintaxis	53
2.21.2. Características de PHP	53

<b>2.22. Modelo MVC</b>	<b>54</b>
2.22.2. El controlador es responsable de:	55
2.22.3 Las vistas son responsables de:	55
<b>2.23. JAVASCRIPT</b>	<b>57</b>
<b>2.24. WEB SERVICES</b>	<b>57</b>
2.24.1. Tecnología Web Services	58
2.24.2. XML	58
2.24.3. SOAP	58
2.24.4. WSDL	59
2.24.5. UDDI	59
<b>2.25. SESIONES</b>	<b>59</b>
<b>2.26. TECNOLOGÍAS PARA IMPLEMENTAR SERVIDOR WEB</b>	<b>60</b>
2.26.1. LINUX	60
2.26.2 SISTEMA DE ADMINISTRACIÓN DE BASE DE DATOS	60
2.26.3. MYSQL	61
2.26.4. MARÍA DB	61
<b>2.27. PRUEBAS</b>	<b>62</b>
<b>2.28. SEGURIDAD</b>	<b>63</b>
<b>CAPÍTULO 3</b>	<b>66</b>
<b>MARCO PRÁCTICO</b>	<b>66</b>
<b>3.1. DISEÑO DEL MODELADO DE NEGOCIO</b>	<b>66</b>
3.1.1 MODELO DE NEGOCIO ACTUAL	66
3.1.2 MODELO DE NEGOCIO ALTERNATIVO	68

<b>3.2. ARQUITECTURA</b>	<b>72</b>
<b>3.2.1. MODELO VISTA CONTROLADOR</b>	<b>72</b>
3.2.1.1. Modelo (Model):	72
3.2.1.2. Vista (View):	72
3.2.1.3. Controlador (Controller):	72
<b>3.3. DIAGRAMA DE ENTIDAD – RELACIÓN</b>	<b>74</b>
<b>3.4. ACTORES</b>	<b>74</b>
<b>3.5. CASOS DE USO</b>	<b>75</b>
3.5.1. IDENTIFICAR CASOS DE USO	75
<b>3.6. DIAGRAMAS DE CASOS DE USO</b>	<b>78</b>
<b>3.7. DIAGRAMA DE CASO DE USO EXTENDIDO</b>	<b>81</b>
<b>3.8. DIAGRAMA DE ACTIVIDADES</b>	<b>93</b>
3.8.1. Inicio de sesión	93
3.8.2. Subir al sistema los convenios	94
3.8.3. Modificar los convenios	94
3.8.4. Eliminar Convenios	95
3.8.5. Alerta de expiración de convenios	95
3.8.6. Información de pasantías	96
<b>3.8.7. Empresa o Institución Publicar pasantías</b>	<b>96</b>
<b>3.8.8. Enviar solicitud de pasantía</b>	<b>97</b>
<b>3.8.9. Información de Pasantías</b>	<b>98</b>
<b>3.9. MAQUETACIÓN WEB</b>	<b>99</b>
<b>3.10. FASE PRE GAME</b>	<b>100</b>

<b>3.11. CONCEPCIÓN Y EXPLORACIÓN</b>	<b>100</b>
<b>3.12. ROLES SCRUM</b>	<b>102</b>
3.12.1. Roles Scrum Asignados en el Proyecto	102
<b>3.13. OBTENCIÓN DE REQUERIMIENTOS</b>	<b>102</b>
<b>3.14. PRODUCT-BACKLOG</b>	<b>103</b>
3.14.1. SPRINT 1	105
3.14.2. SPRINT 2	105
3.14.3. SPRINT 3	105
<b>CONCLUSIONES</b>	<b>110</b>
<b>RECOMENDACIONES</b>	<b>110</b>
<b>BIBLIOGRAFÍA</b>	<b>111</b>

## **INTRODUCCIÓN**

Con la constante evolución tecnológica, numerosas empresas e instituciones han emprendido la digitalización de procesos documentales, buscando otorgar mayor accesibilidad a los usuarios y agilizar trámites.

En este contexto globalizado, el desafío de mantenerse actualizado se vuelve imperativo, especialmente dado el creciente acceso de las personas a los servicios de internet. Es esencial forjar estrategias y desarrollar planes que permitan que la institución se mantenga a la vanguardia, brindando a los estudiantes facilidades para mantenerse al día con la información relevante.

La implementación de una plataforma digital no solo busca optimizar la gestión administrativa de las prácticas en la industria, sino que también aborda una problemática existente: en la actualidad, la mayoría de los institutos de educación superior carece de un sistema centralizado de publicaciones y acceso a los convenios establecidos por la institución, los estudiantes deben acudir de manera presencial a la institución hasta en más de tres ocasiones a fin de solicitar una carta de prácticas en la industria. Esta ineficiencia no solo afecta a su comodidad, sino también a la productividad.

La visión que se busca en la cual los estudiantes sean capaces de optimizar sus trámites administrativos, fomentando una participación más amplia, ahorrando tiempo y promoviendo una interacción más fluida entre estudiante e institución.

La implementación de plataformas como esta, no solo abordará estas limitaciones, sino que también simplificará procesos, ofreciendo un entorno accesible y flexible para los trámites estudiantiles, buscando crear un entorno digital que facilite la experiencia estudiantil y promueva la excelencia en todos los aspectos de la vida académica.

## **CAPÍTULO 1**

### **EL PROBLEMA**

#### **1.1. ANTECEDENTES**

El instituto tecnológico “Escuela Industrial Superior Pedro Domingo Murillo” fue fundado en el gobierno del Gral. Enrique Peñaranda con R.S. Nº 1019 del 10 de febrero de 1942 como “Escuela Nacional de Artes y Oficios». El 4 de agosto de 1942, comenzaron oficialmente las actividades académicas. En fecha 27 de agosto de 1969 con R.S. 150876 se asimila a instituto de enseñanza superior ubicado en la ciudad de La Paz, fue galardonado con el Cóndor de los Andes. «Grado Oficial» bajo R.S. 211445 del 7 de octubre de 1992. En la actualidad cuenta con carreras orientadas al ámbito industrial, las cuales son Mecánica Industrial, Química Industrial, Electricidad Industrial, Metalurgia fundición y siderurgia, Mecánica Automotriz, Electrónica, Industria Textil y Confección e Informática Industrial. Además tiene subsedes en Achica Arriba, Tajani y Corocoro.

La escuela brinda a los estudiantes la posibilidad de realizar prácticas en la industria en diferentes empresas e instituciones con las que cuenta con convenio.

En la actualidad, el proceso para obtener la carta de solicitud de prácticas en la industria por parte de los estudiantes requiere una serie de pasos. Primero, el estudiante solicita la carta a la jefatura de carrera. Luego, el equipo de jefatura de carrera verifica si se cumplen los requisitos necesarios y genera una carta dirigida a Dirección Académica, esta segunda carta debe ser recogida por el estudiante. Finalmente, Dirección Académica proporciona la carta definitiva de prácticas en la industria, la cual debe ser entregada a la empresa o institución correspondiente a la que el estudiante está postulando. Este proceso implica que el estudiante debe visitar la institución hasta en tres ocasiones distintas. Además, es importante destacar que actualmente el registro de estudiantes a los cuales se les hace seguimiento de prácticas se realiza de manera manual en cada unidad administrativa, lo que añade una limitación en la gestión de este proceso y susceptibilidad sobre la veracidad de la documentación entregada por el estudiante.

## **1.2. PROBLEMA**

### **1.2.1. Situación Problemática**

Considerando los antecedentes mencionados, se identifican los siguientes problemas:

- El limitado seguimiento del progreso y el desempeño de los estudiantes durante sus pasantías en la industria, provoca una falta de retroalimentación oportuna y a la incapacidad de identificar a los estudiantes que pueden necesitar apoyo adicional o que están teniendo dificultades en sus prácticas.
- Dado que el proceso de obtención de cartas de solicitud de prácticas involucra a distintas unidades administrativas, existe el riesgo de que la comunicación sea deficiente o que la información se pierda en el proceso. Esto puede generar confusiones, retrasos en la emisión de cartas y una experiencia generalmente insatisfactoria para los estudiantes.
- Los procesos inefficientes para la centralización de la información y recopilación de datos sobre las pasantías de los estudiantes provocan dificultades en la institución para evaluar la efectividad de los convenios.
- Métodos no efectivos y confiables para verificar la autenticidad de los documentos presentados por los estudiantes generan preocupaciones sobre la integridad del proceso de selección y asignación de pasantías.

### **1.2.1. Formulación del problema**

El empleo de inefficientes procesos manuales y funcionalidades reducidas aplicadas en el proceso de autorización para realizar las prácticas en la Industria provoca falta de seguimiento y evaluación efectiva de los estudiantes en prácticas, comunicación inefficiente entre los departamentos involucrados, falta de transparencia en el proceso para los estudiantes y dificultades en la recopilación y análisis de datos sobre pasantías.

### **1.3. OBJETIVOS**

#### **1.3.1. Objetivo general**

Desarrollar e implementar un sistema web para el proceso de gestión, autorización y seguimiento de las prácticas en la industria de los estudiantes del I.T. “E.I.S.P.D.M”.

#### **1.3.2. Objetivos específicos**

- Diseñar e implementar un módulo de seguimiento detallado del progreso y el desempeño de los estudiantes durante sus pasantías en la industria.
- Diseñar e implementar componentes del sitio web que mejoren la comunicación y coordinación entre las distintas unidades administrativas involucradas en el proceso de obtención de cartas de solicitud de prácticas.
- Implementar una funcionalidad que permita la centralización y recopilación de datos de pasantías para evaluar la efectividad de convenios institucionales.
- Implementar un sistema web de verificación de documentos que sea efectivo y confiable en el proceso de selección y asignación de pasantías.

### **1.4. ALCANCES**

#### **1.4.1. Alcance del proyecto**

- Seguimiento del Progreso y Desempeño Estudiantil.
- Mejora de la Comunicación y Coordinación Administrativa.
- Aumento de la Transparencia en el Proceso de Obtención de Cartas de Prácticas.
- Sistema Centralizado de Gestión de Datos de Pasantías.
- Sistema de Verificación de Documentos para Selección y Asignación de Pasantías.

#### **1.4.2. Alcance del producto**

- Integrar Nuevos Componentes para Comunicación y Coordinación Administrativa.
- Desarrollar Seguimiento de Pasantías Estudiantiles.

- Desarrollar el Módulo de Transparencia en el Proceso de Obtención de Cartas de Prácticas.
- Analizar y Centralizar GESTIÓN de Datos de Pasantías.
- Desarrollar Módulo de AUTENTICACIÓN de Documentos para Selección de Pasantías.

## **1.5. LÍMITES**

### **1.5.1. Alcance Geográfico:**

El sistema se implementará exclusivamente para el Instituto Tecnológico "Escuela Industrial Superior Pedro Domingo Murillo" y sus sedes en Achica Arriba, Tajani y Corocoro. No se considera la expansión a otras instituciones o ubicaciones.

### **1.5.2. Usuarios Internos y Externos:**

Los usuarios internos incluirán estudiantes, personal administrativo y académico del instituto, así como empresas o instituciones colaboradoras. Los usuarios externos, como estudiantes de otras instituciones o empresas externas, no tendrán acceso al sistema, se considerará, previo registro y verificación , a estudiantes antiguos para el registro al sistema.

### **1.5.3. Integración con Sistemas Externos:**

Si bien se evaluará la integración con sistemas existentes en la institución, los detalles específicos de esta integración estarán sujetos a la viabilidad técnica y los recursos disponibles. No se garantiza la integración total con sistemas externos.

### **1.5.4. Escalabilidad:**

El sistema se diseñará para manejar la cantidad de estudiantes y pasantías típicas en el Instituto Tecnológico "Escuela Industrial Superior Pedro Domingo Murillo". Sin embargo, no se considerará la escalabilidad para una expansión masiva de estudiantes o pasantías en el futuro.

#### **1.5.5. Idioma y Localización:**

El sistema se desarrollará en un idioma específico (por ejemplo, español) y se adaptará a las regulaciones y requisitos locales vigentes en la ubicación del instituto. No se proporcionará soporte multilingüe ni adaptación a regulaciones de otros países.

#### **1.5.6. Costos y Recursos:**

Los costos y recursos asociados con el desarrollo, implementación y mantenimiento del sistema estarán limitados al presupuesto asignado y los recursos disponibles en el Instituto Tecnológico "Escuela Industrial Superior Pedro Domingo Murillo". No se considera una asignación ilimitada de recursos.

#### **1.5.7. Seguridad de Datos:**

Si bien se implementarán medidas de seguridad para proteger los datos de los estudiantes y las empresas colaboradoras, la seguridad total está sujeta a las mejores prácticas y recursos disponibles en el proyecto.

#### **1.5.8. Plazos de Implementación:**

El proyecto tendrá plazos definidos para cada fase, y la implementación se llevará a cabo de acuerdo con un calendario predefinido. No se considerarán extensiones significativas de plazos sin justificación.

#### **1.5.9. Aprobación y Validación:**

La aceptación y validación del sistema por parte de los usuarios se realizará de acuerdo con los procedimientos y criterios preestablecidos. No se considerará la aprobación sin cumplir con los requisitos acordados.

### **1.6. JUSTIFICACIÓN**

#### **1.6.1. Justificación Social:**

El sistema propuesto mejora la experiencia estudiantil al simplificar y agilizar el proceso de solicitud y seguimiento de pasantías. Esto se traduce en una reducción significativa de la carga administrativa que los estudiantes enfrentan actualmente, además de brindarles acceso claro y transparente a la información sobre el estado de sus trámites. Como resultado, se promueve una experiencia estudiantil más satisfactoria y menos estresante,

permitiendo a los estudiantes enfocarse en su formación profesional con mayor confianza y comodidad.

Se mejorará el apoyo a la formación profesional al posibilitar un seguimiento exhaustivo del progreso y el desempeño de los estudiantes durante sus pasantías en la industria. Esta capacidad permitirá identificar de manera eficiente a aquellos estudiantes que puedan requerir asistencia adicional, brindando la oportunidad de ofrecerles el apoyo necesario para maximizar su desarrollo profesional. Además, al facilitar la recopilación y evaluación de datos sobre las pasantías, se promoverá la mejora continua de la calidad de la formación profesional proporcionada por la institución.

Al simplificar y mejorar el proceso de pasantías, se está contribuyendo a la formación de profesionales más competentes y altamente calificados. Estos profesionales, una vez que completen sus estudios y pasantías con éxito, estarán mejor preparados para hacer contribuciones valiosas al desarrollo de la sociedad y la industria, impulsando el progreso y la innovación en sus respectivos campos.

#### **1.6.2. Justificación Técnica**

Esta automatización no solo mejorará la eficiencia operativa, sino que también reducirá de manera significativa los errores humanos que pueden surgir en el proceso. Al minimizar la intervención manual en tareas administrativas, se garantizará un flujo de trabajo más fluido y preciso, permitiendo que los recursos humanos se enfoquen en actividades de mayor valor agregado y toma de decisiones estratégicas.

Un aspecto crítico en la implementación de este sistema es la mejora de la seguridad de los datos. Se pondrán en marcha sólidas medidas de seguridad destinadas a salvaguardar la información de los estudiantes y las empresas colaboradoras. Esto incluirá la protección de la confidencialidad y la integridad de los datos, asegurando que la información esté resguardada de accesos no autorizados y cualquier intento de manipulación indebida. La implementación de estas medidas brindará un nivel de seguridad que evitará casos de falsificación de documentos.

## **1.7. ESTUDIO DE FACTIBILIDAD**

### **1.7.1. Factibilidad Técnica:**

El Instituto Tecnológico "Escuela Industrial Superior Pedro Domingo Murillo" tiene una infraestructura de red sólida y equipos de cómputo adecuados para ejecutar el sistema de gestión de pasantías propuesto.

La selección de tecnologías ha sido clave en la viabilidad técnica del proyecto. A continuación, se presentan los aspectos relevantes a tener en cuenta:

- La utilización de Laravel 9 como marco de desarrollo en el lado del servidor ofrece una cimentación sólida y versátil para la ejecución de un proyecto. Gracias a su conjunto de características y una sintaxis altamente intuitiva, Laravel posibilita la eficiente gestión de las peticiones de los clientes y la implementación de la lógica empresarial de manera efectiva.
- Sistema de gestión de bases de datos MySQL: La incorporación de MySQL como sistema de gestión de bases de datos asegura la confiabilidad y eficiencia en el almacenamiento y administración de los datos del proyecto. Dada su amplia adopción en la industria, MySQL respalda la integridad y la disponibilidad de la información de manera sólida.
- Características del servidor, donde se aloja el sistema Web.
- Características mínimas del ordenador que será utilizado para el desarrollo del Sistema.

Procesador: Intel Core i7-11700 GHz

RAM instalada: 16,0 GB (15,9 GB usable)

- Tipo de plataforma: Sistema operativo de 64 bits y procesador compatible con la arquitectura x64.

Además, disponemos de hardware con las especificaciones técnicas necesarias para satisfacer los requisitos de rendimiento de cada una de las herramientas a utilizar en la fase de implementación del proyecto.

### **1.7.2. Factibilidad económica:**

En términos de viabilidad económica del proyecto, se ha realizado un análisis del costo del servidor. Se ha evaluado la adquisición de un Servidor Virtual Privado (VPS) con un valor de 14,99 \$us/mes Esto suma un total de 179.88 dólares anuales, el proveedor del dominio es Namesheap el cual cobra 13,98 dólares anuales. El instituto tecnológico “Escuela Industrial Superior Pedro Domingo Murillo” cuenta con la capacidad financiera necesaria para la adquisición de estos elementos requeridos para llevar a cabo la implementación del proyecto, lo que lo hace económicamente factible.”

El precio de desarrollo e implementación del proyecto tendría un costo de 12000 Bs incluyendo impuestos de ley, el cual cuenta con el tiempo de prueba y mantenimiento, durante un año teniendo ajustes y adaptaciones POST-EVALUACIÓN pero al ser un proyecto de grado no se realizará cobro alguno.

### **1.7.3. Factibilidad operacional:**

La institución cuenta con personal capacitado en el manejo de tecnologías de la información los cuales ya trabajaron con distintos sistemas web de carácter académico.

## **CAPÍTULO 2**

### **MARCO TEÓRICO**

#### **2.1 INTRODUCCIÓN**

Para el desarrollo del proyecto se usan los conceptos descritos en el presente capítulo, relacionados con la metodología a usar durante el desarrollo del sistema, así como las distintas herramientas y métodos que servirán para realizar el presente proyecto y abordarlo.

#### **2.2 CONCEPTO DE PASANTÍAS**

Las pasantías, o prácticas profesionales, son una parte esencial de la formación de los estudiantes de instituciones de educación superior. Proporcionan la oportunidad de aplicar los conocimientos adquiridos en un entorno real de trabajo y adquirir experiencia práctica. Además, las pasantías permiten a los estudiantes explorar y comprender las dinámicas laborales en su campo de estudio y facilitan la transición de la educación a la vida laboral. Es vital que las pasantías se gestionen de manera eficiente y transparente, no solo para los estudiantes sino también para la institución y las empresas colaboradoras.

##### **2.2.1 OBJETIVOS DE LAS PASANTÍAS**

1. Ofrecer al estudiante la oportunidad de aplicar los conocimientos teóricos y prácticos adquiridos, en Instituciones Públicas y/o Privadas de manera práctica.
2. Intercambiar información tecnológica, científica y humanística entre la carrera y las instituciones públicas, privadas y otros.
3. Contribuir al mejoramiento del conocimiento científico, tecnológico y humanístico del estudiante, a través de la participación directa en la solución de problemas específicos en el ámbito de las instituciones públicas, privadas y otros.
4. Complementar la formación académica del estudiante mediante el contacto directo con el campo laboral.

5. Implementar relaciones permanentes entre la Carrera y las instituciones públicas, privadas y otras a fin de contribuir a mejorar la calidad académica de la carrera.

## 2.3. METODOLOGÍA.

### 2.3.1. MÉTODO Y TÉCNICAS DE INVESTIGACIÓN.

#### 2.3.1.1. METODO ANALITICO

El método analítico es un enfoque de resolución de problemas que se basa en la descomposición de un problema o sistema complejo en sus componentes individuales para comprenderlo mejor. Implica el análisis detallado de los elementos que componen un todo, identificando sus características, relaciones y funciones.

En el contexto de la investigación y el análisis, el método analítico se utiliza para examinar y desglosar un tema, problema o conjunto de datos en sus partes constituyentes. Los pasos típicos en la aplicación del método analítico incluyen:

- **Identificación del problema:** Definir claramente el problema o el objeto de estudio.
- **Descomposición:** Dividir el problema en partes más pequeñas y manejables.
- **Análisis de componentes:** Estudiar cada parte en detalle, considerando sus propiedades, características y relaciones con otras partes.
- **Síntesis:** Reconstruir el problema o el sistema a partir del conocimiento adquirido de sus componentes.
- **Conclusiones:** Llegar a conclusiones o soluciones basadas en el análisis de las partes individuales.

#### 2.3.1.2. Encuesta.

Una encuesta es la aplicación de un cuestionario a un grupo representativo del universo que estamos estudiando. Un estudio de caso comprende una entrevista extensa con una guía de preguntas o de indicadores para detectar sobre la persona o la comunidad todos los elementos que nos permitan conocer de ella desde sus orígenes hasta el momento actual (Paz, 2017, pág. 82).

### **2.3.1.3. Entrevista.**

Un sondeo, a diferencia de una entrevista, es un interrogatorio sin un rigor científico, que nos permite obtener una información general pero muy útil sobre el tema que estamos investigando, cómo se ha recibido cierto suceso o cómo se comporta la gente ante algún hecho (Paz, 2017, pág. 79).

### **2.3.1.4. Referencia o Bibliográfica.**

“Son las fuentes primarias consultadas por el investigador a lo largo del informe. Recordemos que se incluyen al final siguiendo un estilo de publicaciones APA, Vancouver, etc.” (Hernández, 2014, pág. 523).

Utilizando este método se realizará el análisis de las funciones de: planificación, seguimiento y evaluación de pasantías. Específicamente de cada una de las etapas del proceso de coordinación de práctica profesional del instituto tecnológico “Escuela Industrial Superior Pedro Domingo Murillo”, desde la obtención de la información de los estudiantes que no cuentan con el conocimiento del proceso, realización y seguimiento de trámites reglamentarios, seguimiento del desarrollo de pasantías en las empresas, control y organización de las evidencias del proceso, hasta la aprobación del informe final de prácticas y verificación de las competencias adquiridas durante el proceso de desarrollo de prácticas. Antes de llevar a cabo el análisis, se efectuó la recopilación de la documentación que respalda el procedimiento reglamentario de ejecución de pasantías del instituto tecnológico “Escuela Industrial Superior Pedro Domingo Murillo”.

## **2.4 INGENIERÍA DE SOFTWARE.**

La Ingeniería de Software es un conjunto de métodos, herramientas y procesos de software que tiene como fin elaborar o producir un software de alta calidad.

Según (Pressman, 2010, pág. 11), “la ingeniería de software es una tecnología con varias capas. Como se aprecia en la Figura 2.1, cualquier enfoque de ingeniería (incluso la de software), debe basarse en un compromiso organizacional con la calidad. La administración total de la calidad, Six Sigma y otras filosofías similares alimentan la cultura de mejora continua, y es esta cultura la que lleva en última instancia al desarrollo de enfoques cada vez más eficaces de la ingeniería de software.

El fundamento en el que se apoya la ingeniería de software es el compromiso con la calidad”.



**Figura 2.1 Capas de Ingeniería de Software**

Fuente: (Pressman, 2010, Pág. 12)

“El fundamento para la ingeniería de software es la capa de proceso. El proceso de ingeniería de software es el aglutinante que une las capas de la tecnología y permite el desarrollo racional y oportuno del software de cómputo. El proceso define una estructura que debe establecerse para la obtención eficaz de tecnología de ingeniería de software. El proceso de software forma la base para el control de la administración de proyectos de software, y establece el contexto en el que se aplican métodos técnicos, se generan productos del trabajo (modelos, documentos, datos, reportes, formatos, etc.), se establecen puntos de referencia, se asegura la calidad y se administra el cambio de manera apropiada”.

Los métodos de ingeniería de software ofrecen la experiencia técnica necesaria para la creación de software. Estos métodos abarcan una amplia gama de actividades, como comunicación, análisis de requisitos, diseño de modelos, programación, pruebas y soporte. Se fundamentan en un conjunto de principios fundamentales que rigen cada área de la tecnología, incluyendo actividades de modelado y otras técnicas descriptivas.

Las herramientas de ingeniería de software proporcionan soporte automatizado o semiautomatizado para el proceso y los métodos. Cuando estas herramientas se integran de manera que la información generada por una pueda ser utilizada por otra, se establece

un sistema conocido como Ingeniería de Software Asistida por Computadora, que respalda el desarrollo de software.

La Ingeniería de Software SE del inglés Software Engineering, es un enfoque sistemático del desarrollo, operación, mantenimiento y retiro del software", que en palabras más llanas, se considera que "la Ingeniería de Software es la rama de la ingeniería que aplica los principios de la ciencia de la computación y las matemáticas para lograr soluciones costo-efectivas (eficaces en costo o económicas) a los problemas de desarrollo de software", es decir, "permite el elaborar consistentemente productos correctos, utilizables y costo-efectivos" (Angelfire, 2020) .

La ingeniería del software es el proceso formal de desarrollo de software en el que las necesidades del usuario se traducen en requerimientos, estos se transforman en diseño que se implementa en código que se prueba, documenta y se certifica para su uso operativo.

## 2.5. MODELOS DE PROCESO DE SOFTWARE

Los modelos de proceso de software se pueden conceptualizar como una representación simplificada del desarrollo de software, proporcionando una visión general del proceso. Estos modelos abarcan las diversas actividades de la ingeniería de software, los productos de software resultantes y el papel desempeñado por las personas en dicho proceso. La mayoría de estos modelos se fundamenta en uno de los tres paradigmas generales de desarrollo de software:

- **Enfoque en cascada:** Involucra las etapas anteriores y las divide en fases de proceso individuales, como la definición de requisitos, el diseño de software, la implementación, las pruebas, entre otras.
- **Desarrollo iterativo:** Esta metodología combina las fases de especificación, desarrollo y validación, generando de manera ágil un sistema inicial a partir de especificaciones inicialmente abstractas.
- **Modelo Prototipado:** Un modelo actúa como prototipo para la construcción del sistema definitivo.
- **Transformación Formal:** La implementación del sistema implica una conversión formal del modelo matemático del sistema.

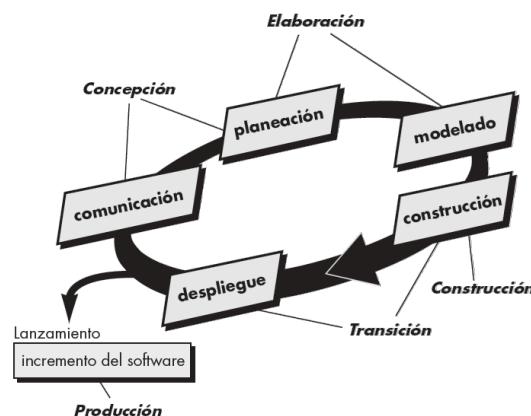
- **Desarrollo basado en Reutilización:** El sistema es ensamblado a partir de componentes existentes.

## 2.6. PROCESO UNIFICADO.

De acuerdo con (Pressman, 2010, pág. 46), el Proceso Unificado es en esencia un intento de amalgamar las mejores características de los modelos tradicionales del proceso de desarrollo de software con los principios más destacados de las metodologías ágiles. Este enfoque reconoce la importancia de la comunicación con el cliente y utiliza métodos directos, como el empleo de casos de uso, para expresar la perspectiva del cliente sobre un sistema. Pone énfasis en la relevancia de la arquitectura del software y promueve que el arquitecto se centre en objetivos clave, tales como la comprensibilidad, la capacidad de realizar cambios futuros y la reutilización. Propone un flujo de proceso iterativo e incremental, creando una sensación evolutiva fundamental en el desarrollo contemporáneo de software.

Según (Métodos, 2020), la metodología RUP, acrónimo de Rational Unified Process (Proceso Unificado Racional), es un enfoque propietario en la ingeniería de software desarrollado por Rational Software, posteriormente adquirido por IBM. El proceso, ahora conocido como Irup (abreviatura de Rational Unified Process), se ha convertido en una marca destacada en el ámbito del software, proporcionando técnicas que deben seguir los miembros del equipo de desarrollo de software para mejorar su productividad durante el proceso de desarrollo. El Proceso Unificado destaca por su enfoque en casos de uso, su atención a la arquitectura y su naturaleza interactiva e incremental.

Fases del proceso unificado. Las fases del proceso unificado son las siguientes:



## **Figura 2.2 Fases del Procesos Unificado**

**Fuente y Elaboración: (Pressman 2010)**

Según Pressman (2010, pág. 47), la fase de elaboración abarca las actividades de comunicación y modelado del modelo general del proceso. Durante esta fase, se mejoran y amplían los casos de uso preliminares desarrollados en la fase de concepción, y se aumenta la representación de la arquitectura, incorporando cinco puntos de vista diferentes del software: modelos del caso de uso, de requerimientos, de diseño, de implementación y de despliegue. En algunos casos, la elaboración crea una "línea de base de la arquitectura ejecutable", que representa un sistema ejecutable de "primer corte".

La fase de construcción, según Pressman (2010, pág. 47), es idéntica a la actividad de construcción definida para el proceso general del software. Utilizando el modelo de arquitectura como entrada, esta fase desarrolla o adquiere los componentes del software que harán operativos los casos de uso para los usuarios finales. Se completan los modelos de requerimientos y diseño iniciados durante la fase de elaboración para reflejar la versión final del incremento de software. Luego, todas las características y funciones necesarias para el incremento de software, como el lanzamiento, se implementan en código fuente. Durante este proceso, se diseñan y realizan pruebas unitarias para cada componente, junto con actividades de integración, como el ensamblaje de componentes y pruebas de integración.

Según Pressman (2010, pág. 48), la fase de transición del proceso unificado abarca las últimas etapas de la actividad general de construcción y la primera parte de la actividad de despliegue general, que incluye la entrega y retroalimentación. Durante esta fase, el software se proporciona a los usuarios finales para pruebas beta, quienes informan tanto defectos como cambios necesarios. Además, el equipo de software genera la información de apoyo necesaria, como manuales de usuario, guías de solución de problemas y procedimientos de instalación, requerida para el lanzamiento. Al concluir la fase de transición, el software incrementado se convierte en un producto utilizable que se lanza.

Según Pressman (2010, pág. 48), la fase de producción del Proceso Unificado coincide con la actividad de despliegue del proceso general. Durante esta fase, se monitorea el

uso del software, se brinda apoyo para el ambiente de operación e infraestructura, y se informan defectos y solicitudes de cambio para su evaluación. Es posible que, al mismo tiempo que se llevan a cabo las fases de construcción, transición y producción, comience el trabajo en el siguiente incremento del software. Esto implica que las cinco fases del Proceso Unificado no ocurren secuencialmente, sino que se desarrollan de manera escalonada.

El flujo de trabajo de la ingeniería de software se distribuye a lo largo de todas las fases del Proceso Unificado. En este contexto, un flujo de trabajo se asemeja al conjunto de tareas necesario para completar una acción importante de la ingeniería de software y los productos de trabajo que resultan de la finalización exitosa de esas tareas. Es importante señalar que no todas las tareas identificadas para el flujo de trabajo del Proceso Unificado se llevan a cabo en todos los proyectos de software. El equipo adapta el proceso, acciones, tareas, subtareas y productos del trabajo para satisfacer sus necesidades específicas.

## **2.7 METODOLOGÍAS**

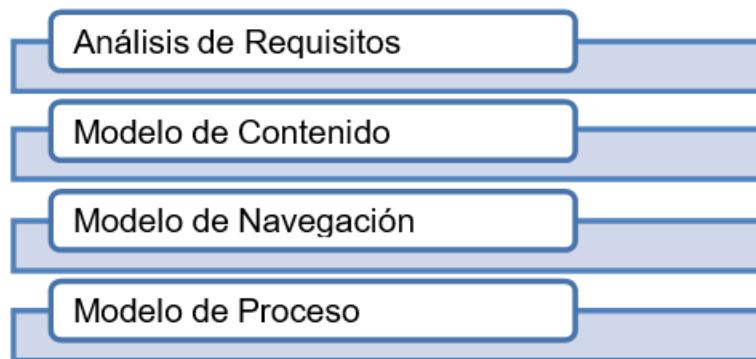
### **2.7.1. METODOLOGÍA UWE**

UWE es una metodología que mejora la especificación de aplicaciones web durante su proceso de desarrollo, utilizando una notación estandarizada basada en UML (Unified Modeling Language) para sus modelos y métodos. Esto simplifica la transición y facilita la creación de aplicaciones web de manera más efectiva. La metodología define claramente la construcción de cada uno de los elementos del modelo.

En su implementación se deben contemplar las siguientes etapas y modelos.

- Análisis de requisitos. Plasma los requisitos funcionales de la aplicación Web mediante un modelo de casos de uso.
- Modelo de contenido. Define, mediante un diagrama de clases, los conceptos a detalle involucrados en la aplicación.
- Modelo de navegación. Implica la navegación de los objetos dentro de la aplicación y un conjunto de elementos tales como índices, menús y consultas.

- Modelo de proceso. Representa el aspecto que tienen las actividades que se conectan con cada clase de proceso (Menéndez, Guerrero, Domínguez, 2014).



**Figura 2.3 Representación Gráfica de Metodología UWE**

**Fuente:** (Menéndez, Guerrero, & Domínguez, 2014)

UML-Based Web Engineering (UWE) es una propuesta metodológica basada en el Proceso Unificado (Jacobson, Booch & Rumbaugh, 1999) y UML para el desarrollo de aplicaciones web (Hennicker & Koch, 2000, Koch, 2001). UWE abarca la totalidad del ciclo de vida de aplicaciones de este tipo, con un enfoque especial en aplicaciones personalizadas que se ajustan a necesidades específicas.

En este estudio, nuestro enfoque principal se centra en analizar la metodología de captura de requisitos propuesta por UWE. Esta metodología distingue claramente entre las fases de solicitud, definición y validación de los requisitos. El producto final de la captura de requisitos en UWE consiste en un modelo de casos de uso, complementado por documentación que detalla los usuarios del sistema, las reglas de adaptación, los casos de uso y la interfaz. Además, UWE sugiere varias técnicas adecuadas para la captura de requisitos en sistemas web, que incluyen entrevistas, cuestionarios, casos de uso, escenarios y un glosario para definir los requisitos. Para la validación, se proponen walk-throughs, auditorías y prototipos. (Escalona & Koch, 2002, págs. 14, 15).

### **2.7.2. METODOLOGÍA DE DESARROLLO ÁGIL**

La metodología de desarrollo ágil es un enfoque de gestión de proyectos que se basa en la iteración y la colaboración entre equipos multidisciplinarios. En contraste con las

metodologías tradicionales de desarrollo, las metodologías ágiles promueven la flexibilidad, adaptabilidad y la entrega continua de incrementos funcionales del producto. Estas metodologías se centran en la satisfacción del cliente a través de la entrega temprana y constante de valor (Cockburn, y otros, 2001).

A continuación, se presentará una tabla de comparación entre metodologías ágiles y enfoques tradicionales:

<b>Metodologías Ágiles</b>	<b>Metodologías Tradicionales</b>
Elaboradas según heurísticas derivadas de prácticas en la producción de código	Elaboradas conforme a las normativas establecidas por los estándares seguidos en el entorno de desarrollo.
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Equipos reducidos, compuestos por menos de diez miembros, que colaboran en el mismo lugar de trabajo.	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura de software	La arquitectura del software es esencial y se expresa mediante modelos

**Tabla 2.1**

#### **2.7.2.1. METODOLOGÍA DE DESARROLLO SCRUM**

El término "Scrum", que se traduce como "Melé" y refleja un aspecto clave del método, destaca la importancia de la cohesión del equipo. En inglés, expresa el hecho de que el método se fundamenta en gran medida en la colaboración del equipo en su conjunto, conformado por el Product Owner, el Scrum Master y el equipo de desarrollo. Similar a la unión necesaria entre los jugadores de rugby durante una melé para lograr avanzar con el balón, el equipo en Scrum trabaja conjuntamente para alcanzar un objetivo común. Es importante destacar que el logro del objetivo no se da de inmediato, sino a través de varias iteraciones que permiten al equipo evaluar su progreso y adaptarse a cambios en las necesidades, superando obstáculos en el proceso (Subra & Vannieuwenhuyse, 2018).

En el ámbito del desarrollo ágil de software, la metodología Scrum se presenta como un conjunto de directrices diseñadas para fomentar la colaboración efectiva entre equipos en proyectos. Esta metodología se respalda en un conjunto de normas y herramientas, estableciendo roles específicos que configuran la estructura esencial para su adecuado desempeño.

#### **2.7.2.2. CARACTERÍSTICAS**

- SCRUM es un enfoque ágil de gestión de proyectos cuyo principal propósito es maximizar la productividad de un equipo.
- El enfoque busca minimizar la burocracia y las actividades que no contribuyen directamente a la creación de software funcional y resultados en cortos períodos de tiempo (cada 30 días), a través de iteraciones o Sprints.
- Ideal para proyectos con un rápido cambio de requerimientos.
- Colaboración estrecha con el cliente.
- Predisposición y respuesta al cambio.
- Desarrollo incremental con entregas frecuentes.
- Comunicación verbal directa.
- Motivación, compromiso y responsabilidad del equipo por la autogestión.
- Simplicidad de procesos (sólo artefactos necesarios).

- Evitar la burocracia innecesaria.

#### **2.7.2.3. PRINCIPIOS BÁSICOS**

Los principios fundamentales de Scrum residen en su enfoque iterativo y colaborativo para el desarrollo de proyectos. Se basa en la flexibilidad y la adaptación a medida que los equipos trabajan en incrementos de funcionalidades. Scrum abraza la retroalimentación constante y la mejora continua, permitiendo a los equipos responder de manera ágil a los cambios y necesidades del proyecto (Schwaber & Sutherland, 2017).

Scrum utiliza un enfoque incremental que tiene como fundamento la teoría de control empírico de procesos. Esta teoría se fundamenta en transparencia, inspección y adaptación; la transparencia, que garantiza la visibilidad en el proceso de las cosas que pueden afectar el resultado; la inspección, que ayuda a detectar variaciones indeseables en el proceso; y la adaptación, que realiza los ajustes pertinentes para minimizar el impacto de las mismas.

#### **2.7.2.4. PRÁCTICAS DE SCRUM**

Palacio (2014) menciona que SCRUM gestiona la evolución del proyecto de manera empírica y adaptable, mediante la aplicación de prácticas propias de la gestión ágil, lo cual se puede consultar en la tabla 2.2.

Prácticas de la gestión ágil	Descripción
Revisión de las iteraciones	Al concluir cada iteración, que generalmente tiene una duración de 30 días, se realiza una revisión con todas las partes involucradas en el proyecto. Este periodo máximo de tiempo permite identificar y abordar cualquier desviación en el proyecto o en las condiciones del producto de manera oportuna.

Desarrollo Incremental	A lo largo del proyecto, las personas involucradas no se basan en diseños o abstracciones. El enfoque de desarrollo incremental implica que al concluir cada iteración, se cuenta con una parte del producto funcional que se puede revisar y evaluar.
Desarrollo evolutivo	Los modelos de gestión ágil se utilizan para operar en contextos caracterizados por la incertidumbre y la inestabilidad de los requisitos. Tratar de prever en las etapas iniciales cómo será el producto final y basar en esa predicción el desarrollo del diseño y la arquitectura del producto no es realista, ya que las circunstancias requerirán numerosas modificaciones a lo largo del proceso.
Auto organización	En el transcurso del desarrollo de un proyecto, surgen numerosos factores impredecibles en todas las áreas y niveles. En la gestión predictiva, la responsabilidad de resolver estos problemas recae en el director de proyecto.

Colaboración	Las prácticas y el entorno de trabajo ágiles fomentan la colaboración del equipo, lo cual es fundamental, ya que para que la autoorganización funcione como un método eficaz, cada miembro del equipo debe colaborar abiertamente con los demás, aprovechando sus habilidades y no limitándose por su rol o posición.
--------------	---

**Tabla 2.2**

*Prácticas de la gestión ágil*

**Fuente:** (Palacio, 2014)

#### 2.7.2.5. ROLES DE SCRUM

(Scrum Alliance, 2012) describe tres roles en la tabla 2.3:

Nombre de rol	Descripción
<b>ScrumMaster</b>	El rol del Scrum Master se define como un 'líder servicial' cuya función es asistir al equipo Scrum en la adhesión a su proceso. Es esencial que cuente con un sólido entendimiento de Scrum y la capacidad de instruir a otros en sus matices.

<b>Product Owner</b>	El Product Owner tiene la única responsabilidad de definir el producto más valioso posible dentro del plazo establecido. Este objetivo se logra gestionando el flujo de trabajo hacia el equipo, que a su vez se materializa mediante la selección y el perfeccionamiento de los elementos del Product Backlog.
<b>Equipo de desarrollo</b>	El Equipo de Desarrollo está conformado por los especialistas encargados de llevar a cabo las tareas necesarias para entregar el Incremento de Producto. Tienen la capacidad de autoorganizarse para cumplir con sus responsabilidades.

**Tabla 2.3 Roles Scrum**

**Fuente:** (Scrum Alliance, 2012)

#### 2.7.2.6. EVENTOS DE SCRUM

Los eventos en Scrum son reuniones y actividades que estructuran el proceso. Incluyen el **Sprint**, la **Planificación de Sprint**, el **Scrum Diario**, la **Revisión de Sprint** y la **Retrospectiva del Sprint**. Cada evento tiene un propósito específico para garantizar la colaboración y la entrega eficiente de trabajo (Schwaber & Sutherland, 2017).

##### A. EL SPRINT

El Sprint constituye el núcleo central de Scrum y se refiere a un periodo de tiempo fijo, generalmente de 1 a 4 semanas, en el cual el Equipo de Desarrollo lleva a cabo la creación de un incremento de funcionalidades. Al concluir el Sprint, el trabajo debe ser potencialmente entregable y estar listo para ser revisado.

Los Sprints comprenden y abarcan actividades como la Planificación del Sprint (Sprint Planning), los Scrums Diarios (Daily Scrums), el desarrollo del trabajo, la Revisión del Sprint (Sprint Review) y la Retrospectiva del Sprint (Sprint Retrospective).

Durante el Sprint:

- No se realizan cambios que puedan afectar al objetivo del Sprint (Sprint Goal).
- Los estándares de calidad no deben decaer.
- A lo largo del Sprint, el alcance puede aclararse y renegociarse entre el Propietario del Producto (Product Owner) y el Equipo de Desarrollo, a medida que se obtiene un mayor entendimiento de los detalles y requisitos.

Cada periodo de Sprint puede ser visto como un proyecto con una duración máxima de un mes. Al igual que en los proyectos convencionales, los Sprints tienen un objetivo específico. Cada Sprint cuenta con una definición clara de lo que se construirá, un diseño y un plan flexible que orienta la construcción, el trabajo del equipo y el producto resultante (Rubin, 2012).

La duración de los Sprints está limitada a un mes calendario. Esta limitación temporal asegura que el enfoque del Sprint no sea excesivamente amplio, evitando posibles cambios en la definición del proyecto, incrementos en la complejidad y riesgos adicionales. Los Sprints brindan previsibilidad al permitir la inspección y adaptación del progreso al menos una vez al mes calendario. Además, al tener una duración acotada, los Sprints limitan el riesgo financiero a un mes calendario.

## B. PLANIFICACIÓN DE SPRINT

La Sesión de Planificación de Sprint es un encuentro que define las tareas a abordar durante el Sprint. Durante esta reunión, el Equipo de Desarrollo colabora con el Product Owner para identificar los elementos que formarán parte del Sprint Backlog y establecer los objetivos a alcanzar.

La duración máxima de la Reunión de Planificación de Sprint es de ocho horas cuando el Sprint abarca un mes. En casos de Sprints más cortos, la reunión tiende a ser más breve. El Scrum Master se encarga de garantizar la realización exitosa del evento y de asegurar que los participantes comprendan claramente su propósito. Además, el Scrum Master orienta al Equipo Scrum para que se adhiera al marco de tiempo establecido.

La Reunión de Planificación de Sprint aborda las siguientes interrogantes:

- ¿Qué es posible entregar en el Incremento que surgirá en el próximo Sprint?
- ¿De qué manera se llevará a cabo el trabajo necesario para lograr la entrega del Incremento?.

## C. SCRUM DIARIO

El Scrum Diario es una reunión diaria de 15 minutos donde el Equipo de Desarrollo sincroniza su trabajo y planifica el día. Cada miembro responde a tres preguntas:

- ¿Cuáles acciones realicé ayer para contribuir al logro del Objetivo del Sprint por parte del Equipo de Desarrollo?.
- ¿Qué acciones tomaré hoy para contribuir al cumplimiento del Objetivo del Sprint por parte del Equipo de Desarrollo?.
- ¿Existen obstáculos que están impidiendo que el Equipo de Desarrollo o yo alcancemos el objetivo del Sprint?.

El Equipo de Desarrollo hace uso de la Reunión Diaria de Scrum (Daily Scrum) para evaluar el avance en dirección al propósito del Sprint, y para analizar la tendencia que sigue este progreso en términos de completar las tareas incluidas en el Backlog del Sprint. La Reunión Diaria de Scrum optimiza las oportunidades de que el Equipo de Desarrollo alcance exitosamente el objetivo del Sprint.

A diario, los miembros del Equipo de Desarrollo deben comprender cómo colaborar de manera efectiva como un grupo autoorganizado para alcanzar el Objetivo del Sprint y generar el Incremento previsto al final del Sprint. Frecuentemente, después de la Reunión Diaria de Scrum, el Equipo de Desarrollo se reúne nuevamente para llevar a cabo discusiones detalladas o para ajustar y replanificar el trabajo restante del Sprint.

## D. REVISIÓN DE SPRINT

La Revisión de Sprint es una reunión al final del Sprint en la que el Equipo de Desarrollo presenta el trabajo completado y el Product Owner revisa el incremento de funcionalidades. Esta reunión permite obtener retroalimentación y ajustar las prioridades.

## **E. RETROSPECTIVA DEL SPRINT**

La Retrospectiva del Sprint es una reunión que ocurre después de la Revisión de Sprint. El Equipo de Desarrollo analiza su proceso, identifica oportunidades de mejora y define acciones para implementar en futuros Sprints.

## **F. ARTEFACTOS DE LA METODOLOGÍA SCRUM**

Los artefactos en Scrum son elementos tangibles que facilitan la transparencia y la toma de decisiones. Estos artefactos son esenciales para asegurar una comunicación efectiva y un entendimiento común entre los miembros del equipo y las partes interesadas.

## **G. PRODUCT BACKLOG**

El **Product Backlog** es una lista ordenada de elementos que representan el trabajo a realizar en el proyecto. Cada elemento, llamado ítem del backlog, puede ser una característica, una funcionalidad o cualquier otra unidad de trabajo. El **Product Owner** es responsable de mantener y priorizar el Product Backlog en función de las necesidades del cliente y el valor que aporta (Schwaber & Sutherland, 2017).

## **H. SPRINT BACKLOG**

El **Sprint Backlog** consiste en una lista de elementos elegidos del Product Backlog para el Sprint actual. Es el Equipo de Desarrollo quien decide qué elementos se comprometen a completar durante dicho Sprint. Esta lista se actualiza diariamente durante el Scrum Diario con el fin de reflejar el progreso y las tareas que aún están pendientes.

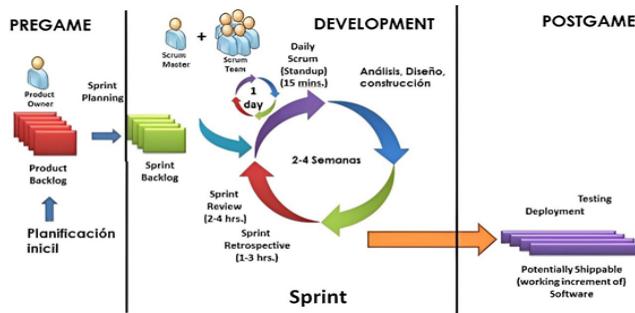
## **I. INCREMENTO**

La acumulación consiste en la totalidad de los elementos que se han completado durante un Sprint, así como en el trabajo realizado en Sprints anteriores. Representa el valor entregable y potencialmente utilizable al final de cada Sprint. El Incremento debe ser evaluado al final de cada Sprint en la **Revisión de Sprint**.

### **2.7.2.7. FASES DE SCRUM**

La **Figura 2.1** ilustra de manera detallada las tres fases cruciales del proceso Scrum. En la fase 'Pregame', la planificación inicial toma forma, abarcando elementos como el product backlog y el sprint planning. La fase 'Development' se enfoca en el sprint, donde

se lleva a cabo el desarrollo del producto. Finalmente, la fase 'Postgame' comprende las etapas finales del proceso, tales como el despliegue y las pruebas, asegurando que el producto esté listo para su lanzamiento.



**Figura 2.4**  
**Proceso Scrum**

Nota. Proceso Scrum: Pregame, Development, Postgame.

#### H. PREGAME

La etapa inicial, conocida como pregame, comprende dos subetapas: planning y architecture, las cuales se detallan en la tabla 2.4.

Subfase	Descripción
<b>Planificación</b>	Incluye la definición del sistema que se está desarrollando. Se crea una lista <i>product backlog</i> que contiene todos los requisitos que son actualmente conocidos. Los requisitos pueden originarse del cliente, ventas y división de marketing, atención al cliente o desarrolladores de software. Los requisitos son prioritarios y el esfuerzo 32 necesario para su implementación es estimado.

	<p>La lista <i>product backlog</i> se actualiza con nuevos y más artículos detallados, así como con estimaciones más precisas y nueva prioridad pedidos. La planificación también incluye la definición del equipo del proyecto, herramientas y otros recursos, evaluación de riesgos y cuestiones de control, necesidades de capacitación y aprobación de la gestión de verificación. En cada iteración, el <i>product backlog</i> es actualizado y la acumulación es revisada por el <i>scrum team</i> para obtener su compromiso para la próxima iteración</p>
<b>Arquitectura</b>	<p>Se planifica en función de los elementos actuales en la lista <i>product backlog</i>. Cuando se trata de mejorar un sistema existente, se detectan las modificaciones requeridas para llevar a cabo los elementos del backlog, al mismo tiempo que se identifican los problemas potenciales que podrían surgir. Se realiza una reunión de revisión de diseño para revisar las propuestas para la implementación y las decisiones se toman sobre la base de esta revisión. Además, se preparan planes preliminares para los contenidos de las publicaciones.</p>

**Tabla 2.4**

*Subfases de Pregame*

Nota. Subfases de Pregame en Scrum: Planificación y Arquitectura

## I. DEVELOPMENT

La etapa de Desarrollo, también conocida como Fase de Juego, representa la parte ágil del enfoque Scrum. En esta fase, se maneja como una caja negra donde se espera lo

impredecible. Las diversas variables ambientales y técnicas, como el marco de tiempo, la calidad, los requisitos, los recursos, las tecnologías y las herramientas de implementación, así como los métodos de desarrollo, que son identificados en Scrum, y que pueden cambiar durante el proceso, se vigilan y controlan a través de varias prácticas de Scrum durante todo el proceso. En lugar de abordar estos asuntos únicamente al inicio del proyecto de desarrollo de software, Scrum se enfoca en supervisarlos constantemente para poder adaptarse de manera flexible a los cambios.

En la etapa de desarrollo, el sistema avanza mediante ciclos iterativos denominados Sprints (conforme a la Figura 2.1.). Los Sprints representan períodos de tiempo en los cuales se construye o mejora la funcionalidad para generar nuevos incrementos. Cada Sprint abarca las etapas tradicionales del desarrollo de software: requisitos, análisis, diseño, evolución y entrega (conforme a la Figura 2.1.). Durante el desarrollo de un Sprint, la arquitectura y el diseño del sistema evolucionan. Se establece una duración estimada para un Sprint, que puede variar desde una semana hasta un mes. Por ejemplo, en un proceso de desarrollo de sistemas, podría haber de tres a ocho Sprints antes de que el sistema esté preparado para su distribución. Además, es posible que más de un equipo participe en la construcción del incremento.

## **J. POSTGAME**

La etapa postgame abarca el cierre del ciclo. Esta fase se inicia cuando se ha alcanzado un consenso de que las variables ambientales, como los requisitos, han sido cumplidas. En este punto, ya no se buscan más elementos ni problemas, ni se introducen nuevas cuestiones. El sistema está ahora listo para el lanzamiento, y la preparación para este paso se lleva a cabo durante la fase postgame, que incluye tareas como la integración, las pruebas del sistema y la documentación (conforme a la Figura 2.1.).

## **K. REUNIONES DE TRABAJO EN UN CONTEXTO SCRUM**

Las reuniones en Scrum son eventos clave que permiten la colaboración y la comunicación efectiva entre los miembros del equipo y las partes interesadas. Estas reuniones están diseñadas para asegurar una planificación precisa, un seguimiento constante y una adaptación ágil a lo largo del proceso de desarrollo (Schwaber & Sutherland, 2017).

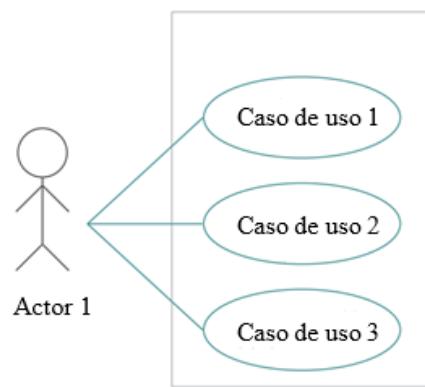
## 2.8 MODELADO DE NEGOCIO

### 2.8.1. DIAGRAMAS UML

El UML se basa en la orientación a objetos, un sistema que surgió antes que el UML en el ámbito de los lenguajes de programación. El primer lenguaje orientado a objetos fue Simula, creado en los años 1960, y le siguieron otros como Smalltalk, C++, Java o C#, más reciente. Los objetos se describen de forma formal en un lenguaje de programación, usando una sintaxis estricta. Esta sintaxis es difícil de entender para los que no son programadores y también para los que lo son. A diferencia de las máquinas, los humanos prefieren usar lenguajes gráficos para representar abstracciones, porque les resultan más fáciles de dominar y les permiten tener una visión global de los sistemas en menos tiempo (Debrauwer & Van der Heyde, 2016).

### 2.8.2. DIAGRAMA DE CASOS DE USO

Los requisitos funcionales que se desean o se tienen para un sistema se expresan mediante los casos de uso, que incluyen los actores (los roles que asumen los usuarios del sistema) y las relaciones entre actores y funcionalidades. Esto también define los límites del sistema, es decir, qué funcionalidades pertenecen al sistema y cuáles no. Los casos de uso son útiles para las fases de modelado, desarrollo y validación. Son una forma de comunicación entre los informáticos y sus clientes y, por lo tanto, son una base para elaborar los aspectos funcionales de la especificación de requisitos.



**Figura 2.5**

*Diagrama de Casos de Uso*

Nota. Ilustración de la estructura de los diagramas de casos de uso

### 2.8.3 DIAGRAMA DE SECUENCIA

Los diagramas de secuencia UML sirven para modelar escenarios de interacciones entre los objetos que pertenecen a las clases y los actores de un sistema de software. Los elementos que participan se comunican mediante mensajes. Además, los diagramas de secuencia UML permiten usar fragmentos combinados para modelar concurrencia, comportamiento alternativo, ciclos y excepciones que reflejan el comportamiento algorítmico de un conjunto de acciones importantes de un sistema.

En la Figura 2.6, se presenta un diagrama de secuencia UML que ilustra cómo interactúan dos roles: el "Vendedor" (representado por Ana) y el "Gestor de Inventario" (representado por Carlos).

En esta representación, observamos que Ana, en su rol de Vendedor, inicia la interacción enviando un mensaje síncrono para verificar la disponibilidad de un producto (VerificarDisponibilidad) a Carlos, el Gestor de Inventario. La respuesta a este mensaje es la información sobre la disponibilidad actual del producto (R1).

Luego, Ana, en su papel de Vendedor, procede a enviar una solicitud síncrona para registrar una venta (RegistroVenta) nuevamente a Carlos. Como respuesta, Ana recibe una confirmación de la transacción y detalles adicionales (R2).

Para cerrar este ciclo de interacción, Carlos, el Gestor de Inventario, envía una notificación asíncrona de actualización de inventario (ActualizaciónInventario) a Ana. Esta notificación informa a Ana sobre los cambios en el inventario después de la venta, proporcionando una visión actualizada de la disponibilidad de productos.

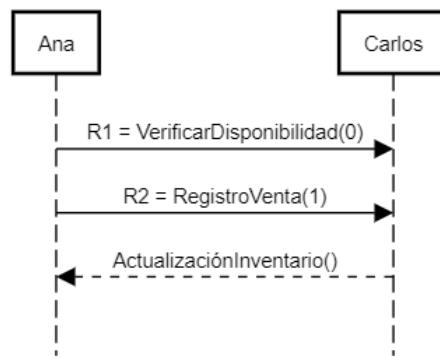


Figura 2.6

### ***Diagrama de Secuencia***

Nota. Ejemplo de la estructura del diagrama de secuencia.

#### **2.8.4. DIAGRAMA DE CLASES**

El Diagrama de Clases muestra la estructura estática de un sistema, representando las clases, sus atributos, métodos y las relaciones entre ellas. Es una herramienta esencial para modelar la organización y las jerarquías de las clases en un sistema.

Nombre de la clase
Atributo 1: Tipo
Atributo 2: Tipo
Atributo 3: Tipo
Operación 1(): void
Operación 2(): boolean

**Figura 2.7**  
***Diagrama de Clases***

Nota. Ejemplo de la estructura de un diagrama de clases

#### **2.8.5. DIAGRAMA DE COLABORACIÓN**

Un diagrama de objetos presenta los objetos y sus relaciones entre sí, mientras que un diagrama de colaboraciones extiende esta representación al mostrar tanto las relaciones como los mensajes enviados entre objetos. A diferencia de los diagramas de objetos, los diagramas de colaboraciones incluyen información sobre los mensajes intercambiados entre objetos. Generalmente, se evita la inclusión de multiplicidades en estos diagramas para prevenir confusiones.

Para representar un mensaje en este contexto, se traza una flecha cerca de la línea de asociación que conecta dos objetos, apuntando hacia el objeto receptor. La naturaleza del mensaje se indica mediante una etiqueta cercana a la flecha, típicamente utilizando la letra "l" para indicar al objeto receptor que ejecute una de sus operaciones. La

representación del mensaje se concluye con un par de paréntesis que pueden contener los parámetros necesarios, en caso de que existan (IngenieriaSystems, 2023).

#### **2.8.6. DISEÑO DE BASE DE DATOS**

El diseño de bases de datos es un proceso esencial en la ingeniería de sistemas, donde se define la estructura que utilizará una base de datos. Esto incluye definir cómo se almacenarán, organizarán y manipularán los datos.

#### **2.8.7. MODELO RELACIONAL**

El Modelo Relacional es un modelo de datos basado en la lógica de primer orden y la teoría de conjuntos. Fue propuesto por Edgar F. Codd en 1970. En este modelo, los datos se organizan en tablas, y las relaciones entre datos se representan mediante valores comunes en diferentes tablas.

- **Relaciones:** Una tabla en el modelo relacional.
- **Tuplas:** Las filas de una tabla, que representan una entidad.
- **Atributos:** Las columnas de una tabla, que representan las propiedades de una entidad.

### **2.9. HERRAMIENTAS DEL DESARROLLO DEL SOFTWARE**

Las herramientas utilizadas en el desarrollo de software son fundamentales para agilizar el proceso de creación, depuración y mantenimiento de sistemas. En este apartado, se explorarán algunas de las herramientas clave utilizadas en el desarrollo del sistema.

#### **2.9.1 FRAMEWORK**

Los frameworks (marcos de trabajo o estructuras de desarrollo) son conjuntos de herramientas, bibliotecas y pautas de programación que proporcionan una estructura predefinida para el desarrollo de aplicaciones o sistemas de software. Estas estructuras ayudan a los desarrolladores a agilizar el proceso de desarrollo, mejorar la calidad del código y mantener la coherencia en sus proyectos. Aquí tienes información relevante sobre los frameworks:

#### **2.9.1.1. Tipos de Frameworks:**

- **Frameworks de desarrollo web:** Estos son ampliamente utilizados en el desarrollo de aplicaciones web. Algunos ejemplos populares incluyen Angular, React y Vue.js para el desarrollo de interfaces de usuario, y Django y Ruby on Rails para el desarrollo de aplicaciones web completas.
- **Frameworks de desarrollo de aplicaciones móviles:** Para el desarrollo de aplicaciones móviles, se utilizan frameworks como React Native y Flutter, que permiten escribir una sola base de código para múltiples plataformas (iOS y Android).
- **Frameworks de desarrollo de escritorio:** En el desarrollo de aplicaciones de escritorio, se encuentran frameworks como Electrón y JavaFX, que permiten crear aplicaciones de escritorio multiplataforma.
- **Frameworks de backend:** Estos son esenciales para la construcción del lado del servidor de una aplicación. Ejemplos incluyen Express.js para Node.js, Spring para Java y Ruby on Rails.
- **Frameworks de desarrollo de juegos:** En la industria de los videojuegos, se utilizan frameworks como Unity y Unreal Engine para simplificar el proceso de creación de videojuegos.
- **Frameworks de aprendizaje automático:** Para el desarrollo de aplicaciones de aprendizaje automático e inteligencia artificial, se utilizan frameworks como TensorFlow y PyTorch.

#### **2.9.1.2. Ventajas de Utilizar Frameworks:**

- **Productividad:** Los frameworks proporcionan un conjunto de herramientas y patrones de diseño que permiten a los desarrolladores escribir código de manera más eficiente.
- **Mantenimiento:** Los proyectos basados en frameworks son más fáciles de mantener a lo largo del tiempo, ya que siguen convenciones y buenas prácticas de desarrollo.

- **Escalabilidad:** Los frameworks suelen ser escalables y permiten la adición de nuevas funcionalidades de manera más sencilla.
- **Seguridad:** Algunos frameworks incluyen características de seguridad por defecto, lo que ayuda a proteger las aplicaciones contra vulnerabilidades comunes.

#### **2.9.1.3. Desventajas de Utilizar Frameworks:**

- Curva de aprendizaje: Los desarrolladores pueden necesitar tiempo para aprender el framework, lo que puede retrasar el inicio de un proyecto.
- Limitaciones: En algunos casos, los frameworks pueden imponer limitaciones en la forma en que se desarrolla una aplicación, lo que podría no ser adecuado para ciertos proyectos.
- Tamaño del código final: Los frameworks a veces generan código adicional que puede afectar el tamaño y la velocidad de la aplicación.

### **2.10. INGENIERÍA DE REQUERIMIENTOS.**

La Ingeniería de Requerimientos es el procedimiento de reunir, analizar y confirmar las necesidades del usuario en el contexto de un sistema de software.

Según (Pressman, 2010, pág. 102). “La ingeniería de requerimientos proporciona el mecanismo apropiado para entender lo que desea el cliente, analizar las necesidades, evaluar la factibilidad, negociar una solución razonable, especificar la solución sin ambigüedades, validar la especificación y administrar los requerimientos a medida que se transforman en un sistema funcional. Incluye siete tareas diferentes: concepción, indagación, elaboración, negociación, especificación, validación y administración.” Es relevante destacar que algunas de estas actividades se llevan a cabo simultáneamente, y todas se ajustan de acuerdo a los requerimientos específicos del proyecto.

- a) **Concepción.** Esta etapa permite establecer la identificación de la necesidad del usuario, cliente o negocio.
- b) **Indagación.** Permitirá la obtención de los requerimientos de manera organizada.

- c) **Elaboración.** La información adquirida del cliente durante la etapa de concepción e investigación se amplía y perfecciona durante la fase de elaboración.
- d) **Negociación.** Esta etapa permitirá la modificación o eliminación de requerimientos de acuerdo a las prioridades del usuario, cliente o negocio.
- e) **Especificación.** Es la documentación de los requerimientos necesarios.
- f) **Validación.** La excelencia de los productos resultantes del proceso de ingeniería de requisitos.
- g) **Administración de los requerimientos.** Identificar, controlar y dar seguimiento a los requerimientos y a sus cambios en cualquier momento del desarrollo del proyecto.

“La ingeniería de requerimientos es una etapa particularmente crítica en el proceso del software ya que los errores en esta etapa originan inevitablemente problemas posteriores en el diseño e implementación del sistema” (Sommerville,2005, pág. 84).

## 2.11. BASE DE DATOS

Una base de datos es una colección organizada y estructurada de datos que se almacenan de manera eficiente y se pueden acceder, administrar y actualizar fácilmente. Las bases de datos son fundamentales en muchas aplicaciones de software y sistemas de información, ya que permiten almacenar y recuperar datos de manera rápida y precisa. Aquí hay algunos conceptos clave relacionados con las bases de datos:

1. **Sistemas de Gestión de Bases de Datos (DBMS):** Un sistema de administración de bases de datos (DBMS) es una aplicación informática que habilita a los usuarios para crear, mantener y gestionar bases de datos. Ejemplos de sistemas de gestión de bases de datos ampliamente conocidos abarcan Microsoft SQL Server, Oracle, MySQL, PostgreSQL y MongoDB.

### 2. Tipos de Bases de Datos:

- **Bases de Datos Relacionales:** Se emplean tablas con el fin de estructurar y presentar la información de manera organizada. Cada tabla consta de filas y columnas, y se pueden establecer relaciones entre tablas. Ejemplos incluyen SQL Server y MySQL.

- **Bases de Datos No Relacionales:** Almacenan datos de manera más flexible y escalable que las bases de datos relacionales. Pueden incluir bases de datos de documentos, bases de datos de grafos y bases de datos clave-valor. Ejemplos incluyen MongoDB y Neo4j.
  - **Bases de Datos en Memoria:** Almacenan datos en la memoria principal de la computadora en lugar de en discos duros, lo que las hace extremadamente rápidas para acceder a datos. Ejemplos incluyen Redis y Memcached.
3. **SQL (Structured Query Language):** SQL es un lenguaje de programación utilizado para administrar bases de datos relacionales. Facilita la ejecución de operaciones como consultas, inserciones, actualizaciones y eliminaciones de datos en la base de datos.
  4. **Modelo de Datos:** Representa cómo se organizarán y relacionarán los datos en la base de datos. Ejemplos de modelos de datos incluyen el modelo entidad-relación (ER) y el modelo de datos relacional.
  5. **Normalización:** Es el proceso de organizar los datos en una base de datos relacional de manera eficiente y sin redundancia. La normalización se utiliza para eliminar problemas de actualización y garantizar la integridad de los datos.
  6. **Transacciones:** Una transacción es una secuencia de una o más operaciones de base de datos que se ejecutan como una unidad. Deben ser atómicas (es decir, se realizan por completo o no se realizan en absoluto), consistentes (deben llevar la base de datos de un estado válido a otro estado válido), aisladas (las transacciones no deben interferir entre sí) y duraderas (una vez que se confirma una transacción, sus efectos deben ser permanentes).
  7. **Índices:** Los índices se utilizan para acelerar la búsqueda y recuperación de datos en una base de datos. Mejoran el rendimiento de las consultas al proporcionar un acceso más rápido a los registros.
  8. **Seguridad y Privacidad:** Las bases de datos deben estar protegidas para garantizar que solo las personas autorizadas puedan acceder a los datos. Esto se logra mediante la autenticación, la autorización y otras medidas de seguridad.

9. **Respaldos y Restauración:** Es esencial realizar copias de seguridad regulares de las bases de datos para evitar la pérdida de datos debido a fallas del sistema o errores humanos. Los procedimientos de restauración permiten recuperar datos a partir de las copias de seguridad.
10. **Escalabilidad:** Las bases de datos deben poder escalar para manejar un aumento en la cantidad de datos o usuarios. Esto se logra mediante técnicas como la partición de tablas y la replicación.

#### **2.11.1. BASES DE DATOS RELACIONALES:**

Una base de datos relacional (RDBMS) es un tipo de sistema de gestión de bases de datos que utiliza un modelo de datos relacional para organizar y administrar datos. En un sistema de base de datos relacional, los datos se almacenan en tablas compuestas por filas y columnas. Cada fila de la tabla representa una entidad única, y cada columna contiene un atributo o campo específico de esa entidad.

#### **Principales Componentes de una Base de Datos Relacional:**

1. **Tablas:** Son la estructura central de una base de datos relacional. Cada tabla está diseñada para contener datos relacionados con una entidad específica, como clientes, productos o pedidos. Cada fila en una tabla representa una instancia de esa entidad y se llama "registro". Cada columna representa un atributo o campo de la entidad.
2. **Claves Primarias:** Cada tabla debe tener una columna o conjunto de columnas que actúen como clave primaria. La clave primaria consiste en un valor único que distingue de manera exclusiva cada entrada o registro en la tabla.
3. **Relaciones:** Las relaciones se establecen entre tablas mediante el uso de claves primarias y claves ajenas (o foráneas). Las claves ajenas son columnas en una tabla que hacen referencia a la clave primaria de otra tabla. Esto permite relacionar datos entre tablas.
4. **SQL (Structured Query Language):** Se trata del lenguaje empleado para la interacción con bases de datos relacionales. Su utilidad radica en la capacidad de

realizar consultas, así como en la inserción, actualización y eliminación de registros, la definición de estructuras de tablas y la aplicación de restricciones de integridad.

#### **Características Clave:**

1. **Integridad de Datos:** Las bases de datos relacionales utilizan restricciones y reglas para garantizar la integridad de los datos. Esto incluye la validación de datos, la aplicación de claves primarias y foráneas, y la normalización para evitar redundancias.
2. **Flexibilidad y Escalabilidad:** Las bases de datos relacionales son flexibles y escalables. Puedes agregar nuevas tablas, campos o registros sin afectar otras partes de la base de datos.
3. **Recuperación de Datos:** Los RDBMS permiten realizar consultas complejas para recuperar datos específicos según múltiples criterios. Esto facilita la búsqueda y recuperación de información.

#### **Ventajas:**

- Estructura organizada y fácil de entender.
- Apoyo a la integridad de los datos y cumplimiento de reglas.
- Recuperación eficiente de datos a través de consultas SQL.
- Amplia disponibilidad de sistemas RDBMS comerciales y de código abierto.

#### **Desventajas:**

- Puede ser menos eficiente para ciertas aplicaciones con grandes cantidades de datos no estructurados.
- La modificación de la estructura de la base de datos puede ser complicada y costosa.
- No siempre es la mejor opción para aplicaciones web modernas que requieren escalabilidad masiva.

#### **Ejemplos de Sistemas RDBMS Populares:**

1. MySQL
2. PostgreSQL
3. Oracle Database
4. Microsoft SQL Server
5. SQLite

### **2.11.2. MySQL**

MySQL es un sistema de gestión de bases de datos relacional (RDBMS) de código abierto que ha ganado una gran popularidad en la comunidad de desarrollo de software. Aquí tienes una descripción detallada de MySQL:

#### **Características Clave de MySQL:**

1. **Código Abierto:** MySQL es un software de código abierto, lo que significa que puedes utilizarlo, modificarlo y distribuirlo de forma gratuita. Esto lo hace accesible y asequible para una amplia variedad de aplicaciones y organizaciones.
2. **Rendimiento:** MySQL es reconocido por su alto rendimiento y eficiencia en la administración de bases de datos. Puede gestionar de manera efectiva grandes conjuntos de datos y consultas complejas.
3. **Escalabilidad:** MySQL es escalable y se puede utilizar en aplicaciones de cualquier tamaño, desde proyectos personales hasta sistemas empresariales de alto tráfico. Es comúnmente utilizado en aplicaciones web y en servidores de bases de datos.
4. **Multiplataforma:** MySQL es compatible con una variedad de sistemas operativos, incluyendo Linux, Windows y macOS, lo que le permite funcionar en entornos heterogéneos.
5. **Almacenamiento de Datos Relacional:** MySQL almacena datos en tablas relacionales, lo que significa que los datos se organizan en filas y columnas. Esto es ideal para aplicaciones que requieren estructura y coherencia de datos.

6. **Lenguaje SQL:** MySQL utiliza el lenguaje SQL (Structured Query Language) para administrar y consultar datos. Los usuarios pueden realizar consultas complejas para recuperar datos de manera eficiente.
7. **Características de Seguridad:** MySQL ofrece una serie de características de seguridad, como la capacidad de definir permisos y contraseñas para garantizar la integridad de los datos y protegerlos contra accesos no autorizados.

#### **2.11.3. Ediciones de MySQL:**

MySQL se ofrece en varias ediciones, cada una dirigida a un público y caso de uso específicos:

1. **MySQL Community Edition:** Es la edición de código abierto y es gratuita. Es adecuada para proyectos pequeños, desarrollo y aplicaciones de pequeña escala.
2. **MySQL Standard Edition:** Es una edición comercial que ofrece un soporte técnico más sólido y características adicionales. Es adecuada para aplicaciones empresariales de tamaño mediano.
3. **MySQL Enterprise Edition:** Es la edición más avanzada y cara, que proporciona un alto nivel de soporte, seguridad y herramientas de administración avanzadas. Está destinada a aplicaciones empresariales críticas.

#### **2.11.4. Casos de Uso de MySQL:**

MySQL se utiliza en una amplia gama de aplicaciones, incluyendo:

- Sitios web y aplicaciones web dinámicas.
- Sistemas de gestión de contenido (CMS).
- Aplicaciones empresariales, como sistemas de gestión de recursos humanos y contabilidad.
- Aplicaciones móviles.
- Almacenamiento y recuperación de datos de registro.
- Sistemas de informes y análisis.

#### **2.11.5. Empresas y Organizaciones que Utilizan MySQL:**

Muchas empresas y organizaciones importantes utilizan MySQL en sus operaciones, como Facebook, Twitter, Netflix, YouTube, Adobe, y más.

#### **2.11.6. Versiones de MySQL:**

MySQL ha tenido varias versiones importantes a lo largo de los años. La versión más reciente es MySQL 8.0, que incluye mejoras en seguridad, rendimiento y características. Es importante mantenerse al día con las actualizaciones para beneficiarse de las últimas mejoras y correcciones de seguridad.

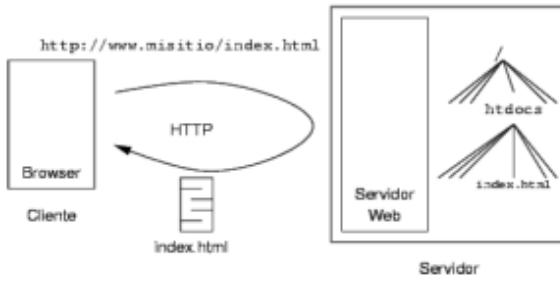
### **2.12. ARQUITECTURA DEL SISTEMA WEB**

Una aplicación Web es proporcionada por un servidor Web y utilizada por usuarios que se conectan desde cualquier punto vía clientes Web (browsers o navegadores).

La arquitectura de un sitio web se compone esencialmente de tres elementos clave:

- Servidor Web: Este componente es responsable de distribuir las páginas de información previamente formateadas a los clientes que las requieren.
- Conexión de Red: La conexión de red facilita la comunicación entre el servidor web y los clientes. Se emplea el protocolo HTTP para gestionar las solicitudes de los clientes.
- Clientes: Estos son uno o más navegadores web utilizados por los usuarios para acceder a las páginas web. Los clientes realizan solicitudes a través de la conexión de red y reciben las páginas web del servidor web.

Cuando un cliente efectúa una solicitud utilizando el protocolo HTTP, el servidor web la recibe y localiza la página web correspondiente en su sistema de archivos. Posteriormente, el servidor web envía esta página de vuelta al navegador del cliente que realizó la solicitud.



**Figura 2.8. Modelo Cliente Servidor**

**Fuente:** Programación Web, Instituto Tecnológico de Matehuala

Las aplicaciones web se sustentan en el modelo cliente/servidor, en el que los servidores web desempeñan un papel fundamental y emplean páginas web como su interfaz principal.

Las páginas web constituyen el componente central de cualquier aplicación o sitio web. Los navegadores web solicitan estas páginas al servidor web, ya sean páginas almacenadas previamente o generadas de manera dinámica con información actualizada. En ciertos entornos de desarrollo de aplicaciones web, las páginas web incorporan código HTML y scripts dinámicos que son ejecutados por el servidor antes de ser entregados al navegador del cliente.

Una vez que se entrega una página, la conexión entre el browser y el servidor Web se rompe, es decir que la lógica del negocio en el servidor solamente se activa por la ejecución de los scripts de las páginas solicitadas por el browser (en el servidor, no en el cliente). Cuando un navegador web ejecuta un script en el lado del cliente, es importante destacar que dicho script no dispone de acceso directo a los recursos del servidor. En este contexto, existen otros componentes que no son considerados scripts, como los applets (aplicaciones especiales que se ejecutan dentro de un navegador) o los componentes ActiveX. Los scripts en el lado del cliente suelen estar escritos en lenguajes como JavaScript o VBScript, y suelen combinarse con código HTML.

En gran medida, las colecciones de páginas web son dinámicas y se generan en tiempo real utilizando tecnologías como ASP.NET (ASP), PHP, entre otras. Estas páginas se organizan de manera lógica para ofrecer servicios al usuario.

El acceso a las páginas está agrupado también en el tiempo (sesión). Los componentes de una aplicación Web son:

#### **2.12.1. Lógica de negocio.**

- Parte más importante de la aplicación.
- Define los procesos que involucran a la aplicación.
- Conjunto de operaciones requeridas para proveer el servicio.

#### **2.12.2. Administración de los datos.**

Manipulación de BD y archivos.

#### **2.12.3. Interfaz**

Los usuarios acceden a las aplicaciones web a través de diversos dispositivos, como navegadores web, dispositivos móviles, PDAs, entre otros. La funcionalidad de estas aplicaciones es accesible mediante un navegador y está limitada y controlada por la propia aplicación.

Las aplicaciones web se estructuran siguiendo un modelo de capas, donde cada capa representa un componente que procesa o manipula la información. Los diferentes tipos de capas en este modelo incluyen:

#### **2.12.4. Modelo de dos capas:**

La información se desplaza a través de dos estratos entre la interfaz y la gestión de datos.

#### **2.12.5 Modelo de n-capas:**

La información pasa a través de distintos niveles, siendo el modelo de tres capas el más comúnmente utilizado.

### **2.13. INTRODUCCIÓN A LA APLICACIONES WEB**

Una aplicación web es un programa informático distribuido cuya interfaz de usuario se puede acceder desde un cliente web, generalmente a través de un navegador web.

### **2.13.1. DEFINICIÓN DE APLICACIONES WEB**

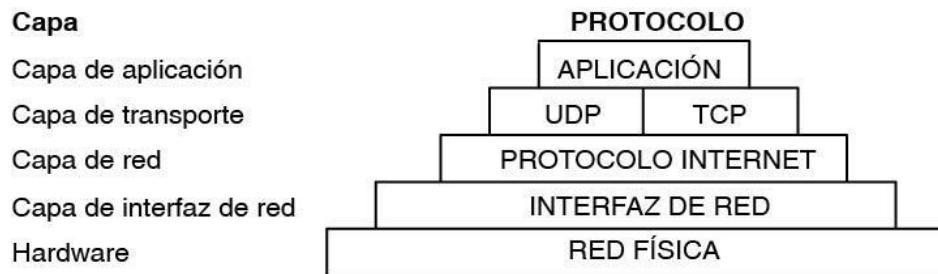
“La web tiene la finalidad de funcionar como sistema abierto de información, en donde el ingreso y actualización del conocimiento en los sistemas de representación permite la interacción y publicación documental de diversos individuos, tanto especialistas como aficionados a temas específicos, lo que implica una saturación informativa y sobrecarga del sistema digital” (Gonzales, 2011, pág. 1).

“Un sitio web debe verse como un conjunto complejo en el que van a estar conectados entre sí (Base de datos, servidores, redes, reglas de acceso, etc.), por tal motivo las aplicaciones deben estar diseñadas sobre las necesidades reales de los usuarios. Es importante tener un sistema web enfocado al usuario que a los gustos del programador” (Cabello, 2015).

### **2.13.2. TCP / IP**

La familia de protocolos de Internet es un conjunto de protocolos de red en los que se fundamenta Internet, y posibilita la transferencia de datos entre redes de computadoras. En algunos contextos, este conjunto puede ser denominado como el conjunto de protocolos TCP/IP, haciendo referencia a los dos protocolos de mayor relevancia que conforman dicha colección:

El Protocolo de Control de Transmisión (TCP) y el Protocolo de Internet (IP) fueron los dos primeros en ser definidos y siguen siendo los más ampliamente utilizados de la familia de protocolos de Internet.



**Figura 2.9.. Conjunto de protocolos TCP/IP**

**Fuente: IBM**

TCP/IP establece de manera precisa el mecanismo de transferencia de información desde el emisor hasta el receptor. En primer lugar, los programas de aplicación envían mensajes o corrientes de datos a uno de los protocolos de la capa de transporte de Internet, UDP (User Datagram Protocol) o TCP (Transmission Control Protocol). Estos protocolos, al recibir los datos provenientes de la aplicación, fragmentan dichos datos en unidades más pequeñas denominadas paquetes. Luego, agregan una dirección de destino y posteriormente transmiten estos paquetes a la capa de protocolo subsiguiente, conocida como la capa de red de Internet.

La capa de red de Internet se encarga de encapsular el paquete en un datagrama IP (Protocolo de Internet). Esta capa añade tanto la cabecera como el pie del datagrama, determina la ruta para enviar el datagrama (ya sea directamente a un destino o a través de una pasarela), y luego deriva el datagrama a la capa de interfaz de red.

La capa de interfaz de red, a su vez, recibe estos datagramas IP y los convierte en tramas de datos que son transmitidas a través de hardware de red específico.

Por ejemplo, redes Ethernet o de Red en anillo.

#### **2.13.3. INTERFACES**

La interfaz de usuario es un entorno visual que integra diversos controles y elementos que posibilitan la comunicación e interacción de un usuario con un dispositivo electrónico.

#### **2.13.4. SERVICIOS WEB**

Los servicios web son aplicaciones modulares, independientes y autónomas que tienen la capacidad de describir, publicar, ubicar e invocar funciones a través de una red.

En las aplicaciones web modernas, se utilizan diversas plataformas de software que permiten la creación de aplicaciones en línea. Estas aplicaciones pueden desarrollarse en Java, .Net, Angular JS, Node.js, entre otras. A partir de esta perspectiva, surge la pregunta sobre la naturaleza de los servicios web. En esencia, el entorno en el que operan no es visible para los usuarios comunes. Con frecuencia, estas aplicaciones individuales requieren establecer algún tipo de comunicación entre sí. Sin embargo, dado que se desarrollan utilizando distintos lenguajes de programación, resulta desafiante garantizar una comunicación precisa entre estas aplicaciones.

## **2.14. PRINCIPIOS DE FUNCIONAMIENTO DE LOS SERVICIOS WEB**

Cuando nos referimos a los servicios web, es fundamental considerar cómo operan, ya que esto facilita la comprensión del sistema en funcionamiento. En este contexto, el cliente inicia un conjunto de solicitudes hacia un servicio web, las cuales son dirigidas al servidor que aloja dicho servicio web. Estas solicitudes se ejecutan mediante llamadas a procedimientos remotos, es decir, procedimientos que se invocan en respuesta a una solicitud específica.

### **2.14.1 HTML 5**

Lo que mencionas acerca del funcionamiento de las páginas web y el lenguaje de programación es correcto. HTML (HyperText Markup Language) es el lenguaje de marcado utilizado para definir la estructura y el contenido de las páginas web. Es fundamental para la representación visual de los elementos en un navegador.

HTML ha evolucionado a lo largo de los años, y HTML5 es la última versión del estándar HTML. Ofrece muchas mejoras y características nuevas, incluyendo un mayor soporte para multimedia, lo que permite a los desarrolladores de sitios web incorporar contenido multimedia directamente en las páginas sin depender de tecnologías como Flash Player, que solían ser comunes en el pasado.

HTML5 ha permitido una experiencia de navegación más rica y mejor en la web, y se ha convertido en el estándar dominante para el desarrollo web moderno. Su adopción ha impulsado la creación de aplicaciones web avanzadas y sitios más interactivos y dinámicos.

### **2.14.2. CSS**

El diseño de documentos HTML solía ser complicado y engorroso antes de la introducción de CSS, principalmente debido a la necesidad de utilizar atributos de estilo directamente en el código HTML. En particular, las etiquetas de estilo demandaban explicaciones minuciosas y reiterativas de los siguientes componentes:

- Colores de las fuentes
- Estilos de fondo
- Elementos de alineación
- Fronteras

- Tamaños

La noción de "cascada" proviene de la posibilidad de aplicar varias hojas de estilo a una misma página web. CSS fue desarrollado por el W3C.

#### **2.14.3. Formatos CSS**

CSS establece el estilo para los siguientes tipos de documentos:

- Lenguaje de marcado de hipertexto (HTML)
- Lenguaje de marcado de hipertexto Extensible (XHTML)
- Lenguaje de marcado extensible (XML)
- Gráfico vectorial escalable (SVG)
- Lenguaje de interfaz de usuario XML (XUL)

#### **2.15. BOOTSTRAP 5**

Bootstrap es un marco de trabajo front-end utilizado para desarrollar aplicaciones web y sitios web con un enfoque "mobile first", lo que significa que se adaptan al tamaño de la pantalla del dispositivo del usuario. Fue creado por Twitter en 2010 y originalmente se llamó "Twitter Blueprint", pero en 2011 se convirtió en un proyecto de código abierto y cambió su nombre a Bootstrap. Desde entonces, ha experimentado varias actualizaciones y actualmente se encuentra en la versión 5.

Este marco combina CSS y JavaScript para estilizar elementos en una página HTML y ofrece mucho más que simplemente cambiar los estilos de botones y enlaces. Proporciona componentes interactivos que facilitan la interacción con el usuario, como menús de navegación, controles de página, barras de progreso y otros elementos.

La característica principal de Bootstrap es su capacidad para permitir la construcción de sitios web responsivos, lo que significa que las páginas se diseñan de manera que funcionen de manera óptima en dispositivos de escritorio, tabletas y teléfonos inteligentes, garantizando una experiencia de usuario consistente y organizada en diferentes plataformas.

#### **2.16. LARAVEL**

En el año 2011, Codeigniter se encontraba entre los frameworks PHP más populares, pero carecía de características esenciales para el desarrollo de aplicaciones web, como

la autenticación. Ante esta carencia, Taylor Otwell, un programador web, decidió crear un nuevo framework que abordara estas necesidades.

Inicialmente, Laravel no seguía el patrón de arquitectura MVC (Modelo-Vista-Controlador) y se centraba principalmente en resolver problemas relacionados con la autenticación. Sin embargo, la primera versión presentaba funcionalidades que fueron muy bien recibidas por la comunidad de desarrolladores. La segunda versión se lanzó en menos de seis meses y marcó la adopción definitiva del patrón MVC para la arquitectura de Laravel. Además, se incorporó el lema: "Liberándote del código espagueti, Laravel te ayuda a crear aplicaciones maravillosas mediante una sintaxis simple y expresiva. El desarrollo debe ser una experiencia creativa que disfrutes, no algo doloroso. Disfruta del aire fresco".

### 2.16.1 CARACTERÍSTICAS

1. **Elegante sintaxis:** Laravel se destaca por su sintaxis limpia y expresiva, lo que facilita la escritura de código limpio y legible. Utiliza el patrón de diseño MVC (Modelo-Vista-Controlador) para separar la lógica de la aplicación en componentes bien definidos.
2. **Rutas y controladores:** Laravel ofrece un sistema de enrutamiento simple y flexible que permite definir rutas de manera clara y asociarlas con controladores. Esto facilita la creación de aplicaciones web con múltiples páginas y acciones.
3. **ORM (Object-Relational Mapping):** Eloquent, el ORM de Laravel, simplifica la interacción con la base de datos al permitirte trabajar con tablas de la base de datos como si fueran objetos. Esto simplifica las consultas y la manipulación de datos.
4. **Plantillas Blade:** Laravel utiliza Blade, un motor de plantillas que facilita la creación y gestión de las vistas de la aplicación. Blade permite la inclusión de componentes, la herencia de plantillas y la creación de diseños reutilizables.
5. **Sistema de autenticación:** Laravel proporciona una funcionalidad de autenticación completa con solo un comando. Esto facilita la implementación de sistemas de registro, inicio de sesión y gestión de usuarios.

6. **Seguridad:** Laravel se preocupa por la seguridad y proporciona funciones integradas para proteger la aplicación contra ataques comunes, como la protección contra CSRF (Cross-Site Request Forgery) y la autenticación de usuarios.
7. **Migraciones y semillas:** Laravel ofrece herramientas para administrar la estructura de la base de datos mediante migraciones y llenar la base de datos con datos de prueba a través de semillas. Esto es especialmente útil en entornos de desarrollo y pruebas.
8. **Gestión de dependencias:** Laravel utiliza Composer, un administrador de dependencias de PHP, para administrar las bibliotecas y paquetes de terceros utilizados en el proyecto.
9. **TDD (Desarrollo Guiado por Pruebas):** Laravel fomenta las mejores prácticas de desarrollo, incluido el TDD. La integración con PHPUnit facilita las pruebas unitarias y funcionales.
10. **Soporte de API:** Laravel es adecuado para el desarrollo de aplicaciones web y API RESTful. La autenticación, la validación y la serialización de datos son fáciles de implementar.
11. **Comunidad activa:** Laravel cuenta con una comunidad activa y una amplia gama de recursos, incluyendo documentación detallada, tutoriales y paquetes adicionales creados por la comunidad.

## 2.17. SWEET ALERT

SweetAlert es una extensión de jQuery que permite dar un aspecto profesional a los mensajes que presentamos a los usuarios, siguiendo las tendencias actuales. Además, brinda la capacidad de configurar el plugin de múltiples maneras diferentes.

Algo muy bueno es que SweetAlert se centra automáticamente en la página web y se ve muy bien si estás usando una computadora de escritorio, un dispositivo móvil o una tableta.

## 2.18. AJAX

AJAX significa JavaScript asíncrono y XML (Asynchronous JavaScript and XML). AJAX es un conjunto de técnicas de desarrollo web que posibilitan que las aplicaciones web operen de manera asincrónica, procesando solicitudes al servidor en segundo plano. Para comprender mejor este término, examinemos cada componente por separado.

- JavaScript es un lenguaje de programación ampliamente reconocido que, entre sus diversas funciones, se encarga de gestionar el contenido dinámico de un sitio web y facilita la interacción dinámica del usuario.
- Por otro lado, XML es una variante de un lenguaje de marcado similar a HTML, como sugiere su nombre: eXtensible Markup Language. Mientras que HTML está diseñado principalmente para mostrar datos, XML está concebido para contener y transportar información.
- Tanto JavaScript como XML operan de manera asincrónica en el contexto de AJAX. Esta característica permite que cualquier aplicación web que utilice AJAX envíe y recupere datos del servidor sin necesidad de recargar la página completa.

## 2.19. JSON

JSON (JavaScript Object Notation) es un formato de intercambio de datos ligero, de fácil lectura y escritura para los usuarios, así como de sencillo análisis y generación por sistemas informáticos. Se basa en un subconjunto del lenguaje de programación JavaScript, específicamente en el Estándar ECMA-262 3a Edición de diciembre de 1999. Aunque es completamente agnóstico con respecto al lenguaje, utiliza convenciones familiares para programadores que trabajan con lenguajes de la familia C, como C, C++, C#, Java™, JavaScript, Perl, Python y otros similares. Estas características hacen que JSON sea un formato óptimo para la transferencia de datos entre diversas aplicaciones.

## 2.20. MOCKUPS

Un mockup, en el ámbito del diseño y la publicidad, representa un montaje visual de alta calidad que se utiliza extensamente para presentar diseños a clientes. Puede abarcar una amplia gama de elementos, como logotipos, páginas web o envases de productos.

Estas representaciones gráficas capturan todas las facetas del diseño, incluyendo paletas de colores, tipografía, tamaños de imágenes, iconografía y la apariencia general del producto o proyecto. Los mockups permiten exhibir de manera efectiva cómo se verá el diseño final en una presentación de alta calidad.

La versatilidad de los mockups es notable, ya que pueden utilizarse para representar desde logotipos hasta páginas web, pasando por aplicaciones móviles, tarjetas de visita, folletos y prácticamente cualquier diseño concebible por un diseñador gráfico.

## 2.21. PHP

PHP es un lenguaje de código abierto ampliamente utilizado, especialmente en el desarrollo web, y puede ser integrado en el código HTML. Su popularidad se debe a que muchas páginas web y portales están construidos con PHP. Ser de código abierto implica que está disponible de forma gratuita para todos los desarrolladores que deseen utilizarlo. La característica de ser incrustado en HTML significa que en un mismo archivo, se pueden combinar segmentos de código PHP y HTML, siguiendo reglas específicas.

### 2.21.1. Sintaxis

La estructura sintáctica de PHP se encuentra basada en los fundamentos de programación de C. El intérprete de PHP se encarga de ejecutar exclusivamente el código contenido entre sus marcadores de inicio y fin. La finalidad de estos marcadores radica en la separación del código PHP del resto del contenido, como, por ejemplo, el código HTML5. En los archivos que contienen solo código PHP, el delimitador "?>" se puede omitir.

De hecho, PHP-FIG, mediante sus Recomendaciones Estándar para PHP (PHP Standard Recommendation), aconseja la omisión del marcador "?>", ya que esta práctica ayuda a prevenir la inclusión accidental de contenido HTML. Por ejemplo, si se envía un carácter "no PHP" (que no es procesado por el intérprete de PHP), no se podrán ejecutar ciertas acciones como enviar encabezados HTTP a través de la función header(), ya que el proceso de respuesta ya ha comenzado.

Las variables se prefijan con el símbolo del dólar (\$) y no es necesario indicar su tipo. Las variables, a diferencia de las funciones, son sensibles a las distinciones entre letras mayúsculas y minúsculas. Las cadenas de caracteres pueden ser encapsuladas tanto en comillas dobles como en comillas simples. No obstante, en el caso de las comillas dobles, es posible insertar variables directamente en la cadena sin requerir la operación de concatenación.

### 2.21.2. Características de PHP

Orientado hacia la creación de aplicaciones web dinámicas con acceso a información almacenada en bases de datos, PHP se destaca como un lenguaje de programación de relativa facilidad en su aprendizaje. En su evolución, se enfocó en la simplificación de diversas especificaciones, como lo ejemplifica la definición de variables primitivas, especialmente notoria en el manejo de PHP arrays.

Un rasgo distintivo es que el código fuente escrito en PHP permanece invisible tanto para el navegador web como para el cliente, dado que es el servidor quien se encarga de ejecutar el código y transmitir el resultado en formato HTML al navegador.

La capacidad de PHP para establecer conexiones con la mayoría de los sistemas de gestión de bases de datos en uso en la actualidad es notable, destacando su compatibilidad con MySQL y PostgreSQL.

Aunque PHP no impone una metodología específica en el desarrollo de aplicaciones, los programadores pueden aplicar cualquier técnica de programación o enfoque de desarrollo que les permita crear código organizado, estructurado y manejable. Esto se ilustra, por ejemplo, en las implementaciones de PHP del patrón de diseño Modelo Vista Controlador (MVC), que posibilitan la separación de las responsabilidades relacionadas con el manejo y acceso a los datos, la lógica de control y la interfaz de usuario en tres componentes independientes.

## 2.22. Modelo MVC

El Modelo Vista Controlador (MVC) es un estilo de arquitectura de software que divide los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes separados. Este modelo, ampliamente probado y maduro, ha demostrado su eficacia a lo largo de los años en diversas aplicaciones y en múltiples lenguajes y plataformas de desarrollo.

- El Modelo incluye una representación de los datos manejados por el sistema, su lógica de negocio y los mecanismos de persistencia.
- La vista, o interfaz de usuario, compone la información enviada al cliente y los mecanismos de interacción con él.

- El Controlador actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ambos y realizando transformaciones para adaptar los datos a las necesidades de cada uno.

#### **2.22.1. El modelo es el responsable de:**

- 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.).

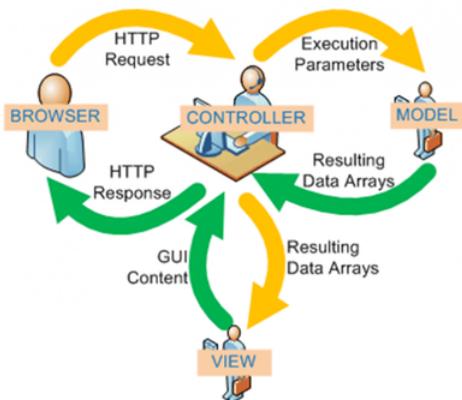
#### **2.22.2. El controlador es responsable de:**

- Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
- Incluye normativas para la administración de eventos, estructuradas como "SI ocurre el Evento Z, entonces se ejecuta la Acción W". Estas acciones pueden implicar solicitudes al modelo o a las vistas. Una solicitud a las vistas podría consistir en invocar el método "Actualizar()". Por otro lado, una solicitud al modelo podría ser "Obtener\_tiempo\_de\_entrega(nueva\_orden\_de\_venta)"

#### **2.22.3 Las vistas son responsables de:**

- 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).

El flujo que sigue el control generalmente es el siguiente:



**Figura 2.10. Flujo del Modelo Vista Controlador**

Fuente: Universidad de Alicante

- a) El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.)
- b) El controlador recibe notificaciones de acciones realizadas por el usuario, las cuales son emitidas por los objetos de la interfaz-vista. El controlador es el encargado de administrar estos eventos, lo que suele llevarse a cabo a través de un gestor de eventos (handler) o una función de devolución de llamada (callback).
- c) El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). En muchos casos, los controladores de gran complejidad se organizan mediante la aplicación de un patrón de comando, el cual encapsula las acciones y simplifica su ampliación.
- d) El controlador asigna a los objetos de la vista la responsabilidad de representar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo debe mantener su independencia con respecto a la vista, evitando el conocimiento directo de esta última. No obstante, es posible implementar el patrón Observador para establecer una comunicación indirecta entre el modelo y la vista, lo que permite que el modelo notifique a las partes interesadas acerca de cualquier modificación. Un objeto vista puede registrarse con el modelo y aguardar las

actualizaciones, pero el modelo en sí mismo sigue sin tener información detallada sobre la vista. El controlador no transfiere objetos de dominio (como el modelo) directamente a la vista, pero puede emitir órdenes a la vista para que esta se actualice. Cabe mencionar que en algunas implementaciones, la vista no dispone de acceso directo al modelo, y en su lugar, es el controlador quien se encarga de transmitir los datos del modelo a la vista.

- e) La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

### **2.23. JAVASCRIPT**

JavaScript es un lenguaje de programación o de secuencias de comandos que te permite implementar funciones complejas en páginas web, cada vez que una página web hace algo más que sentarse allí y mostrar información estática para que la veas, muestra oportunas actualizaciones de contenido, mapas interactivos, animación de Gráficos 2D/3D, desplazamiento de máquinas reproductoras de vídeo, etc., puedes apostar que probablemente JavaScript está involucrado.

### **2.24. WEB SERVICES**

El concepto de Servicios Web describe una metodología estandarizada para integrar aplicaciones web mediante el uso de XML, SOAP, WSDL y UDDI sobre los protocolos de Internet. XML se utiliza para describir datos, SOAP para la transferencia de datos, WSDL para definir servicios disponibles y UDDI para identificar servicios existentes. Facilita la comunicación entre empresas y entre éstas y sus clientes, posibilitando el intercambio de datos sin conocer detalles específicos de los Sistemas de Información respectivos.

A diferencia de los modelos Cliente/Servidor, como un servidor de páginas web, los Servicios Web no proporcionan una interfaz gráfica (GUI) al usuario. En cambio, comparten la lógica empresarial, datos y procesos a través de una interfaz de programas en la red, conectando programas sin interactuar directamente con los usuarios. Los desarrolladores pueden añadir una interfaz para usuarios a los Servicios Web, ya sea a través de una página web o un programa ejecutable, para ofrecer funcionalidades específicas a los usuarios proporcionadas por un Servicio Web en particular.

Los Servicios Web posibilitan la comunicación entre aplicaciones de distintos orígenes sin la necesidad de desarrollar programas costosos, ya que la comunicación se realiza mediante XML. No están limitados a un sistema operativo o lenguaje de programación específico; por ejemplo, un programa en Java puede comunicarse con otro en Pearl, y las aplicaciones Windows pueden interactuar con las aplicaciones Unix. Además, prescinden del uso de navegadores (Explorer) y del lenguaje de especificación HTML.

El modelo de computación distribuida de los Servicios Web simplifica la comunicación entre aplicaciones. Por ejemplo, la aplicación responsable de procesar órdenes de compra puede interactuar con el sistema de inventario, permitiendo que este último informe a la aplicación de compras sobre los artículos que necesitan ser adquiridos al encontrarse por debajo del nivel mínimo. Debido a la fuerte integración que proporcionan a las aplicaciones, los Servicios Web han ganado popularidad y contribuyen a la mejora de los procesos empresariales. Algunos incluso sugieren que están impulsando la próxima evolución de la web.

#### **2.24.1. Tecnología Web Services**

Los Servicios Web se componen de diversas tecnologías que colaboran con los estándares en desarrollo para garantizar la seguridad y el funcionamiento. Esto permite que la combinación de varios Servicios Web, sin importar la empresa proveedora, esté asegurada. A continuación, se presentan de manera concisa los estándares que están siendo adoptados por los Servicios Web

#### **2.24.2. XML**

Extensible Markup Language, abreviado como XML, es una especificación creada por el W3C. Facilita a los desarrolladores la creación de sus propias etiquetas, lo que les permite definir, transmitir, validar e interpretar datos entre aplicaciones y organizaciones.

#### **2.24.3. SOAP**

Conocido como SOAP, que es la abreviatura de Simple Object Access Protocol, este protocolo de mensajería está construido en XML y se emplea para codificar los datos de los requisitos de los Servicios Web, así como para responder a los mensajes antes de su envío a través de la red. Los mensajes SOAP son

independientes de los sistemas operativos y pueden ser transportados mediante protocolos que operan en Internet, tales como SMTP, MIME y HTTP.

#### **2.24.4. WSDL**

El término "WSDL", que proviene de las siglas en inglés de "Web Services Description Language", hace referencia a un lenguaje definido en XML. Su función es la de describir los Servicios Web como conjuntos de puntos de comunicación con la capacidad de intercambiar mensajes. El WSDL es parte integral de UDDI y parte del registro global de XML, en otras palabras, es un estándar de uso público (no se requiere pagar licencias ni royalties para usarlo).

#### **2.24.5. UDDI**

Es la abreviación de Universal Description, Discovery and Integration. Se trata de un directorio distribuido que funciona en la Web y brinda a las empresas la posibilidad de publicar sus servicios web. Esto permite que otras compañías conozcan y utilicen los servicios web que han sido publicados. Su funcionamiento es similar al de las páginas amarillas

### **2.25. SESIONES**

El término "sesión", en el ámbito de la informática, hace referencia al periodo de tiempo en el que tiene lugar un intercambio de información entre dos sistemas informáticos o entre un usuario y un sistema. Una sesión se inicia en un momento específico y concluye poco después. Durante una sesión de comunicación establecida, es posible que se produzcan varios mensajes en ambas direcciones. Por lo general, una sesión es con estado, lo que significa que al menos una de las partes comunicantes debe almacenar información sobre el historial de la sesión para poder mantener la comunicación, aunque también puede ser sin estado, donde cada comunicación consiste en solicitudes independientes con sus respectivas respuestas.

Establecer una sesión es un requisito fundamental para llevar a cabo una comunicación orientada a la conexión y constituye el paso básico para la transmisión en modos de comunicación sin conexión. Es importante destacar que una transmisión unidireccional por sí sola no define una sesión.

La implementación del transporte de comunicación puede realizarse como parte de los protocolos y servicios en la capa de aplicación, en la capa de sesión o en la capa de transporte dentro del modelo OSI.

## 2.26. TECNOLOGÍAS PARA IMPLEMENTAR SERVIDOR WEB

### 2.26.1. LINUX

Linux representa un sistema operativo de software libre de código abierto, conocido por su alta estabilidad, seguridad y capacidad para manejar múltiples usuarios y tareas simultáneamente. Su diseño se orienta hacia la libertad y la gratuidad, y ofrece una variedad de distribuciones notables que se adaptan tanto al entorno doméstico como al empresarial. Destaca por su ligereza y eficiencia en la gestión de recursos limitados, siendo una alternativa ideal a los programas de código cerrado o de pago

### 2.26.2 SISTEMA DE ADMINISTRACIÓN DE BASE DE DATOS

El DBMS (sistema de administración de base de datos) es el software que maneja todo acceso a la base de datos. De manera conceptual, lo que sucede es lo siguiente (figura 5):

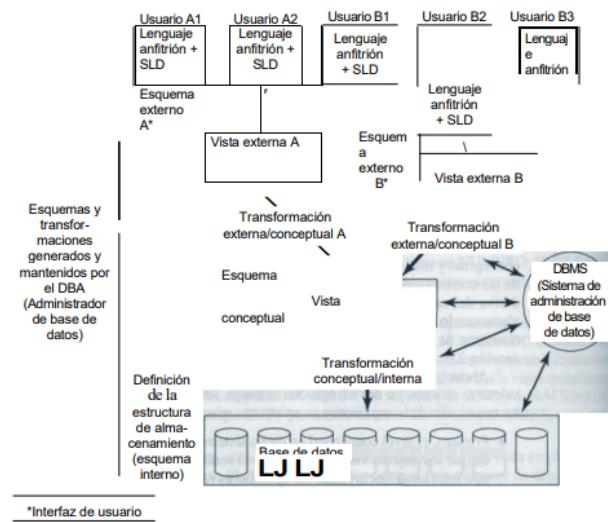


Figura 2.11. Arquitectura detallada del sistema.

Fuente: C.J. Date-Introducción a los sistemas de Bases de Datos Relacionales – Séptima Edición

1. Un usuario emite una petición de acceso, utilizando algún sublenguaje de datos específico (por lo regular SQL).
2. El DBMS intercepta esa petición y la analiza.
3. El DBMS inspecciona, en su momento, (las versiones objeto de) el esquema externo para ese usuario, la transformación externa/conceptual correspondiente, el esquema conceptual, la transformación conceptual/interna y la definición de la estructura de almacenamiento.
4. El sistema de gestión de bases de datos (DBMS) realiza las operaciones necesarias en la base de datos almacenada.

#### **2.26.3. MYSQL**

MySQL es el sistema de gestión de bases de datos relacional más ampliamente utilizado en la actualidad, y se fundamenta en código abierto. Inicialmente desarrollado por MySQL AB, la empresa fue adquirida por Sun Microsystems en 2008, y posteriormente, en 2010, fue adquirida por Oracle Corporation, que ya poseía el motor InnoDB para MySQL.

MySQL opera bajo una doble licencia; por un lado, cuenta con una licencia de código abierto, y por otro, dispone de una versión comercial gestionada por Oracle. Las versiones Enterprise, dirigidas a empresas que desean integrarlo en productos propietarios, ofrecen características adicionales, como herramientas de monitorización y soporte técnico oficial.

#### **2.26.4. MARÍA DB**

MariaDB es un sistema de gestión de bases de datos estrechamente vinculado a MySQL, ya que fue desarrollado por uno de los creadores originales, Michael "Monty" Widenius. La motivación detrás de su desarrollo era mantener el software de gestión de bases de datos dentro del ámbito del software libre.

Este sistema de gestión de bases de datos, MariaDB, conserva las funciones distintivas de MySQL y, a su vez, introduce mejoras significativas. Entre estas mejoras se incluyen la capacidad de ejecutar y almacenar en caché consultas complejas directamente, una nueva gestión de conexiones a bases de datos, la posibilidad de acceder a clústeres de

datos (lo cual resulta útil para entornos en la nube) y el soporte para la utilización de jerarquías de gráficos y estructuras más complejas.

En términos de seguridad y rendimiento, MariaDB presenta mejoras continuas y evoluciona constantemente gracias a la contribución activa de una amplia comunidad que respalda su desarrollo.

## 2.27. PRUEBAS

En el desarrollo de software, se realizan varios tipos de pruebas para garantizar que la aplicación funcione correctamente y cumpla con sus requisitos. Estos son algunos de los tipos de pruebas más comunes:

- Pruebas de Unidad (Unit Testing): Estas pruebas se centran en verificar el correcto funcionamiento de unidades individuales de código, como funciones, métodos o componentes. Los desarrolladores escriben pruebas para cada unidad y las ejecutan para garantizar que funcionen según lo previsto.
- Pruebas de Integración (Integration Testing): Estas pruebas se realizan para asegurarse de que las diversas unidades de código o módulos se integren adecuadamente entre sí. El objetivo es detectar posibles problemas de comunicación o incompatibilidades entre componentes.
- Pruebas de Regresión (Regression Testing): Se realizan pruebas de regresión después de realizar cambios en el código para asegurarse de que las nuevas modificaciones no hayan introducido errores en partes previamente funcionales de la aplicación.
- Pruebas de Funcionalidad (Functional Testing): Estas pruebas evalúan si la aplicación cumple con sus requisitos funcionales, es decir, si realiza las funciones previstas correctamente. Pueden incluir pruebas de casos de uso, pruebas de escenarios y pruebas de extremo a extremo.
- Pruebas de Aceptación (Acceptance Testing): Son pruebas realizadas por el cliente o el usuario final para determinar si la aplicación satisface sus necesidades y cumple con los requisitos acordados. Pueden ser pruebas de aceptación del usuario (UAT) o pruebas de aceptación del cliente.

- Pruebas de Usabilidad (Usability Testing): Se centran en la experiencia del usuario y evalúan la facilidad de uso de la aplicación. Los evaluadores comprueban la navegación, el diseño de la interfaz de usuario y la eficiencia general de la aplicación.
- Pruebas de Carga (Load Testing): Se realizan para evaluar cómo se comporta la aplicación bajo cargas pesadas o condiciones de tráfico intenso. El objetivo es identificar posibles cuellos de botella y problemas de rendimiento.
- Pruebas de Estrés (Stress Testing): Similar a las pruebas de carga, pero se somete a la aplicación a condiciones extremas para evaluar su comportamiento bajo estrés, como la capacidad de recuperación después de un fallo.
- Pruebas de Seguridad (Security Testing): Se realizan para identificar vulnerabilidades y asegurarse de que la aplicación sea segura contra posibles ataques, como inyección de SQL, cross-site scripting (XSS) y otros ataques cibernéticos.
- Pruebas de Rendimiento (Performance Testing):\*Evalúan el rendimiento de la aplicación en términos de velocidad, eficiencia y consumo de recursos. Pueden incluir pruebas de velocidad, pruebas de resistencia y pruebas de escalabilidad.
- Pruebas de Compatibilidad (Compatibility Testing): Aseguran que la aplicación funcione correctamente en diferentes navegadores, sistemas operativos y dispositivos.
- Pruebas de Localización e Internacionalización (L10n y i18n Testing): Se realizan para verificar la adaptabilidad de la aplicación a diferentes idiomas y regiones.
- Pruebas de Mantenibilidad (Maintainability Testing): Evalúan cuán fácil es mantener y actualizar el código de la aplicación a lo largo del tiempo.
- Pruebas de No Funcionales (Non-functional Testing): Estas pruebas se centran en características no funcionales, como la seguridad, el rendimiento, la escalabilidad y la usabilidad.

- Pruebas de Humo (Smoke Testing): Estas pruebas iniciales se realizan para asegurarse de que la aplicación funcione correctamente antes de realizar pruebas más exhaustivas.

## 2.28. SEGURIDAD

La importancia de las medidas de seguridad en el desarrollo de software varía según el contexto y el tipo de aplicación, pero algunos aspectos suelen considerarse especialmente críticos:

- Validación de Entradas: La validación de entradas es fundamental para prevenir ataques de inyección, como la inyección de SQL y scripts. Un error en la validación de entradas puede abrir la puerta a graves vulnerabilidades.
- Autenticación y Autorización: La gestión de usuarios y sus permisos es fundamental. La falta de autenticación y control de acceso puede llevar a violaciones de datos y acceso no autorizado.
- Cifrado: El cifrado es esencial para proteger datos confidenciales en tránsito y en reposo. La falta de cifrado puede exponer datos a riesgos de seguridad.
- Gestión de Sesiones: Un manejo inadecuado de sesiones puede permitir ataques como la suplantación de identidad. Mantener sesiones seguras es crucial.
- Protección contra XSS y CSRF: Los ataques de Cross-Site Scripting (XSS) y Cross-Site Request Forgery (CSRF) son comunes y pueden explotarse para robar datos o realizar acciones no deseadas en nombre de un usuario.
- Almacenamiento Seguro de Contraseñas: El almacenamiento seguro de contraseñas es fundamental, ya que las contraseñas son un objetivo principal para los atacantes. Un almacenamiento inseguro puede llevar a violaciones masivas.
- Actualizaciones y Parches: Mantener el software actualizado es esencial para mitigar vulnerabilidades conocidas. La falta de actualizaciones puede exponer aplicaciones a amenazas conocidas.
- Educación en Seguridad: La concienciación y la formación en seguridad son importantes para todo el equipo de desarrollo. Los errores humanos son una de las principales causas de vulnerabilidades.

- Pruebas de Seguridad: Realizar pruebas de seguridad, como pruebas de penetración y análisis de código, es clave para descubrir y corregir problemas antes de que sean explotados por atacantes.
- Gestión de Identidad y Acceso: La correcta gestión de identidad y acceso asegura que solo los usuarios autorizados tengan acceso a los sistemas y datos. La falta de una sólida IAM puede dar lugar a violaciones de seguridad.
- Desarrollo Seguro: La implementación de prácticas de desarrollo seguro, como el principio de "seguridad en todas partes", es esencial para garantizar que la seguridad sea una consideración constante en todas las etapas del ciclo de vida del desarrollo.
- Seguridad en la Nube: Si la aplicación se ejecuta en la nube, se deben seguir buenas prácticas de seguridad en la nube, como la configuración adecuada de permisos y la protección de datos en tránsito y en reposo.

## **CAPÍTULO 3**

### **MARCO PRÁCTICO**

En el presente capítulo, se ha procedido a efectuar un análisis exhaustivo y minucioso de los componentes físicos y lógicos inherentes al sistema de la aplicación. Se ha enfocado especialmente en la exposición detallada de sus características y capacidades funcionales. Esta tarea se ha llevado a cabo en consonancia con el modelo seleccionado como marco de referencia, persiguiendo la meta primordial de alcanzar los objetivos predefinidos para el desarrollo del proyecto de software.

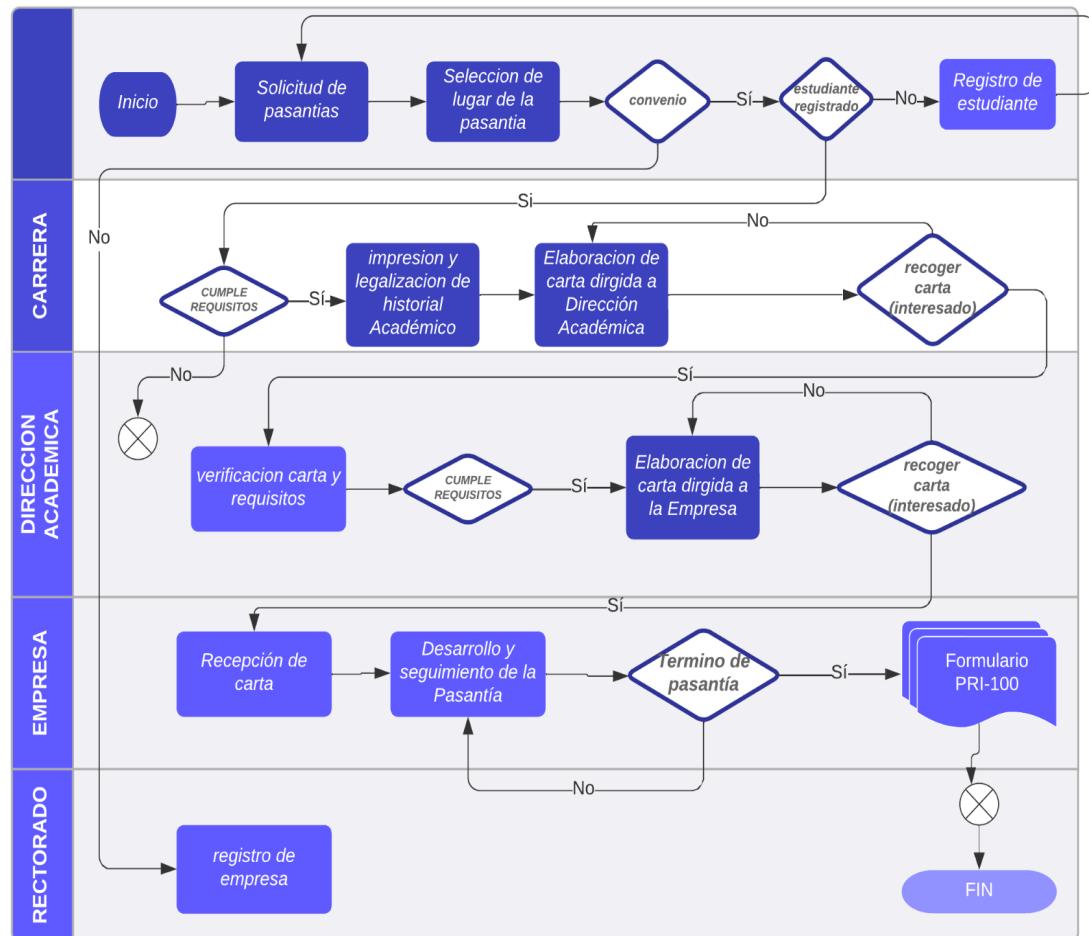
#### **3.1. DISEÑO DEL MODELADO DE NEGOCIO**

Para emprender la fase inicial del desarrollo del sistema, se procede con la elaboración del modelo de negocio vigente. A través de esta fase, se efectuará una descripción minuciosa del estado actual de los procedimientos y actividades relacionadas con la solicitud de pasantías. En consecuencia, con base en el modelo de negocio actual, se propondrá una alternativa de modelado de negocio. El propósito principal de esta alternativa es optimizar y perfeccionar los procedimientos en uso en la actualidad.

##### **3.1.1 MODELO DE NEGOCIO ACTUAL**

El siguiente modelo está basado en entrevistas realizadas a las diferentes unidades involucradas en el proceso de la solicitud de pasantías las cuales son realizadas por los estudiantes de El instituto tecnológico “Escuela Industrial Superior Pedro Domingo Murillo”.

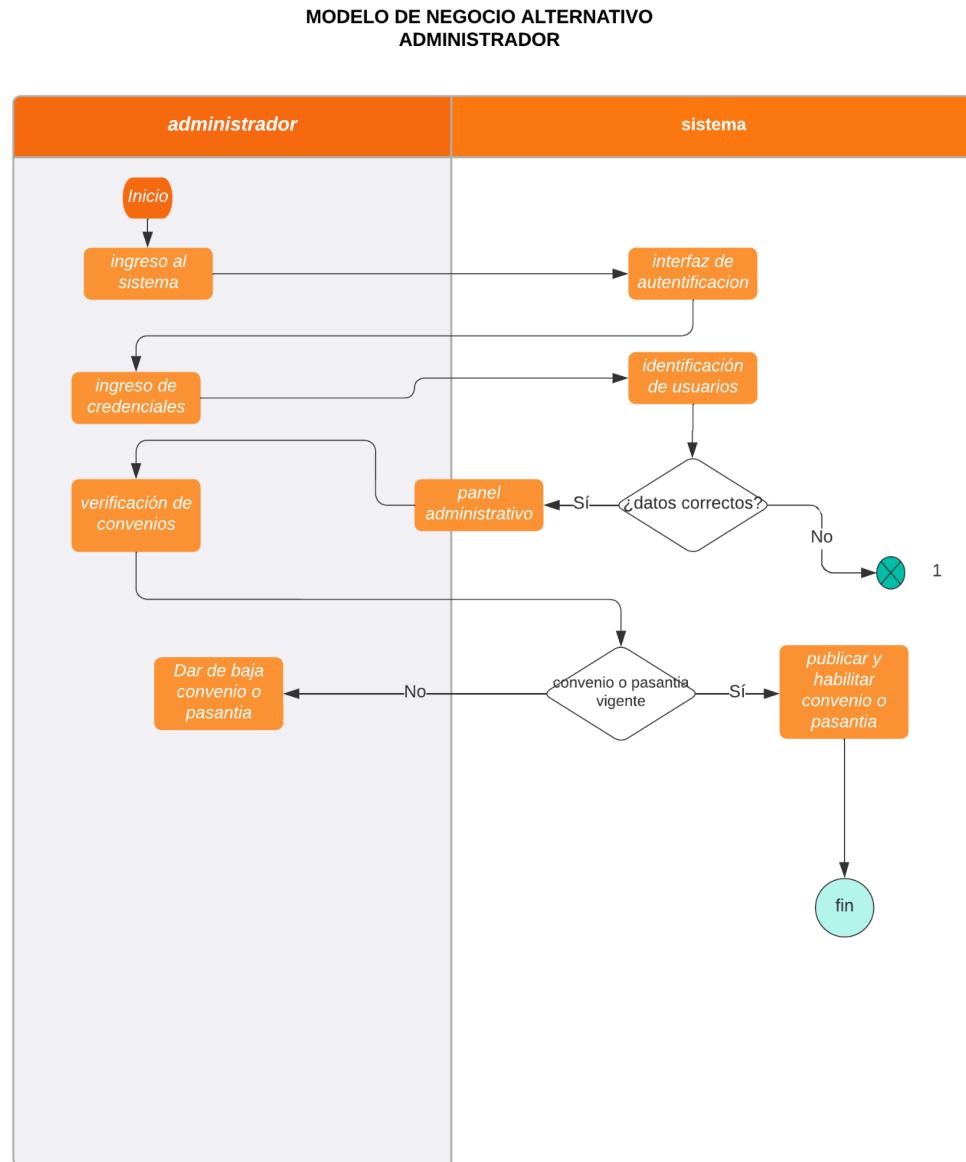
**MODELO DE NEGOCIO ACTUAL**



**Figura 3.1. Modelo de negocio actual**

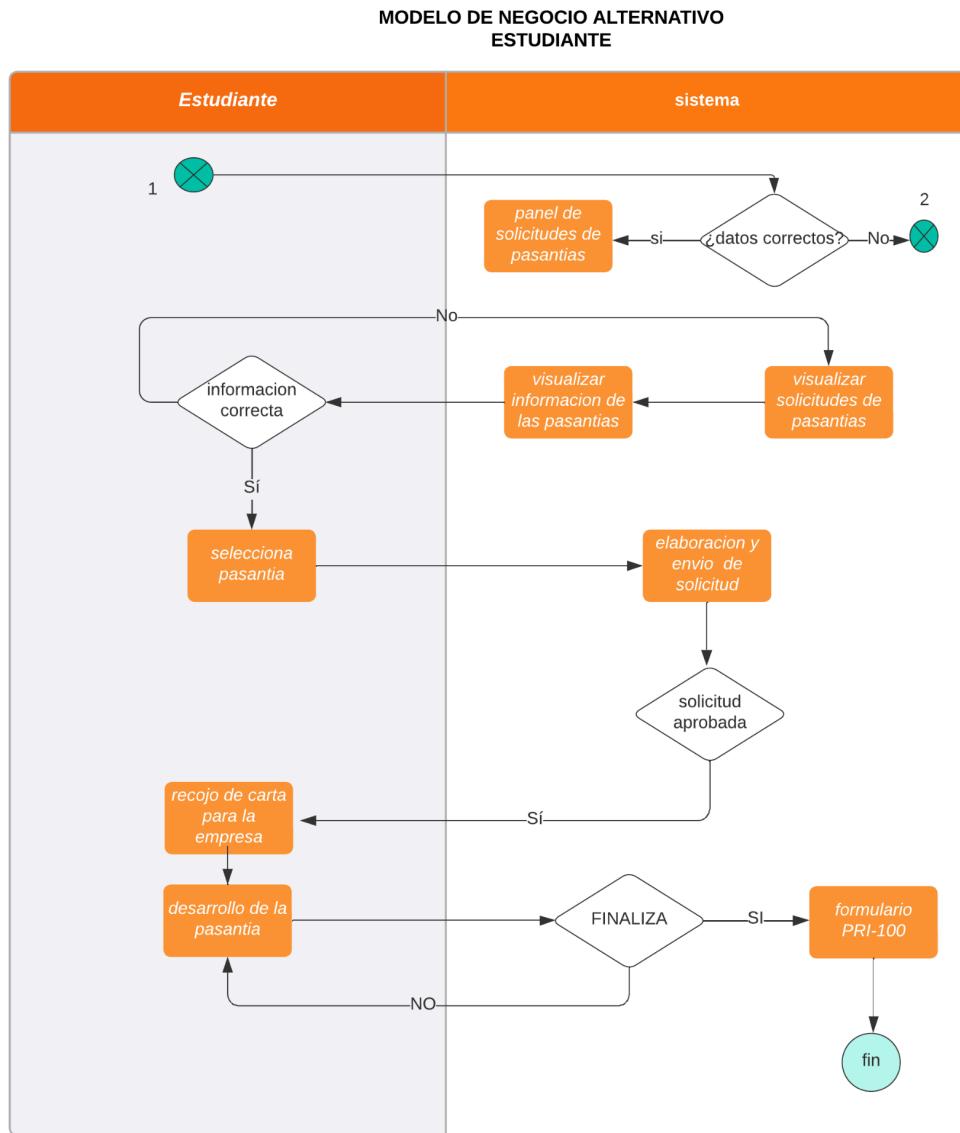
**Fuente:** elaboración propia

### 3.1.2 MODELO DE NEGOCIO ALTERNATIVO



**Figura 3.2 . Modelo de negocio alternativo “administrador”**

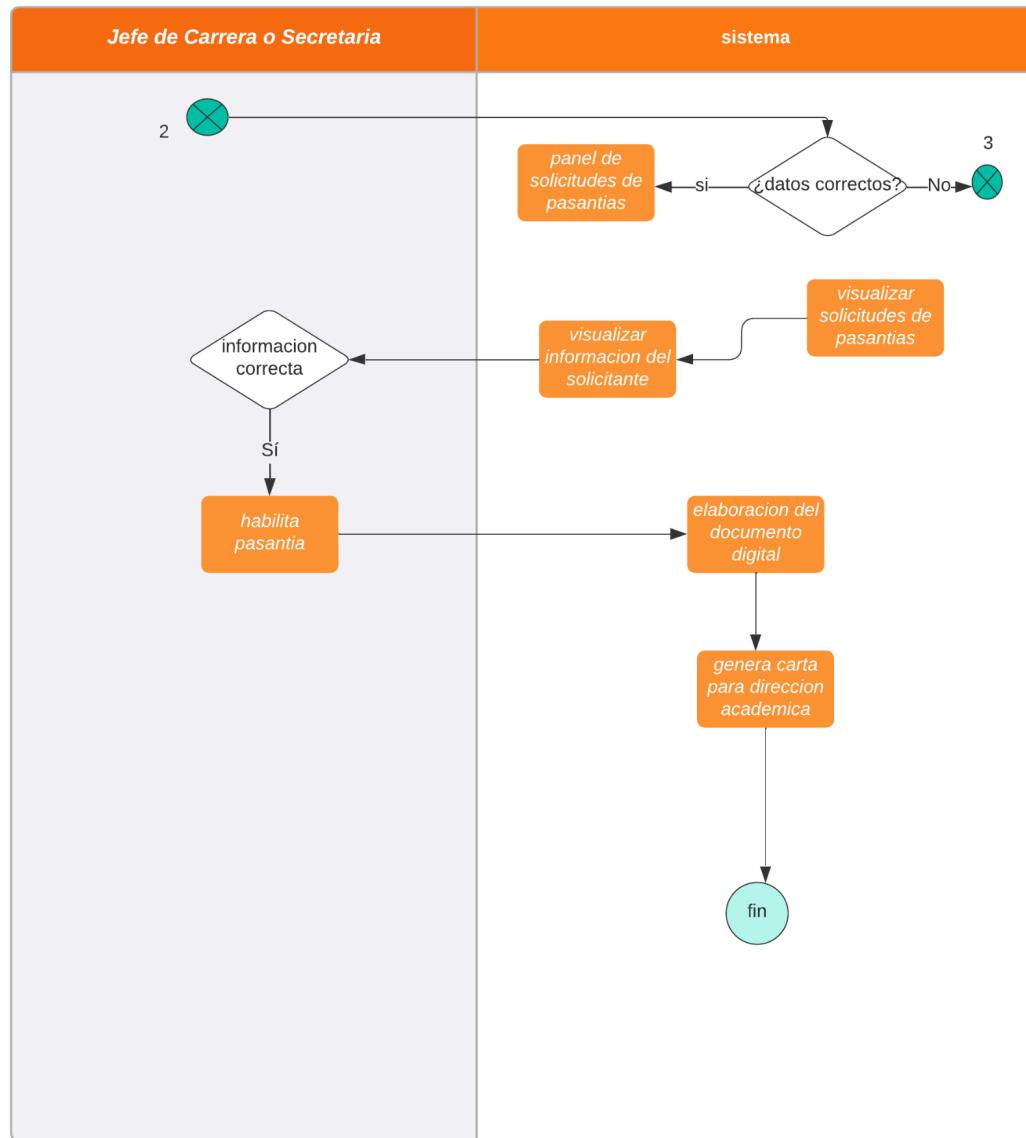
**Fuente:** elaboración propia



**Figura 3.3. Modelo de negocio alternativo “estudiante”**

**Fuente:** elaboración propia

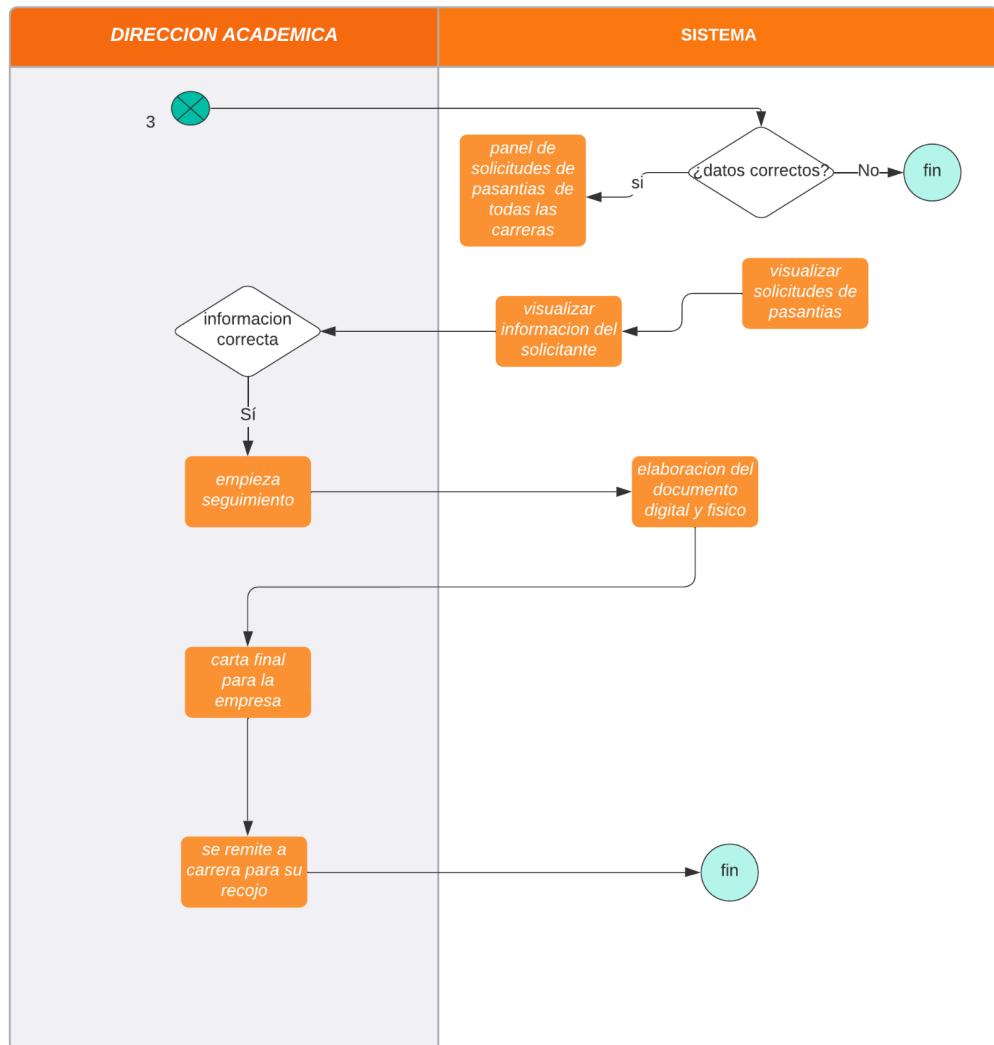
**MODELO DE NEGOCIO ALTERNATIVO  
JEFE DE CARRERA O SECRETARIA**



**Figura 3.4. Modelo de negocio alternativo “Jefe de carrera o Secretaria”**

**Fuente:** elaboración propia

**MODELO DE NEGOCIO ALTERNATIVO  
DIRECCIÓN ACADÉMICA**



**Figura 3.5. Modelo de negocio alternativo “Dirección Académica”**

**Fuente:** elaboración propia

## **3.2. ARQUITECTURA**

### **3.2.1. MODELO VISTA CONTROLADOR**

#### **3.2.1.1. Modelo (Model):**

La capa del modelo comprende la estructura de datos y la lógica empresarial subyacente del sistema de gestión de pasantías. En este contexto, el modelo almacena y administra la información relacionada con las pasantías, como los detalles de los estudiantes, el estado de sus pasantías, la documentación requerida y los registros de seguimiento. Además, se encarga de la lógica empresarial, incluyendo la verificación del cumplimiento de los requisitos de pasantía y el seguimiento del progreso y desempeño de los estudiantes en sus pasantías. La gestión eficiente de los datos y la lógica empresarial son cruciales para el correcto funcionamiento del sistema.

#### **3.2.1.2. Vista (View):**

La capa de vista se ocupa de la presentación de datos al usuario final y de la interacción con este. En el proyecto de gestión de pasantías, la vista representa la interfaz de usuario (UI) que los estudiantes utilizan para solicitar cartas de pasantías, obtener información sobre el estado de sus trámites y acceder al seguimiento de sus pasantías. La vista debe ser diseñada de manera que sea intuitiva y fácil de usar para proporcionar a los estudiantes una experiencia transparente y efectiva.

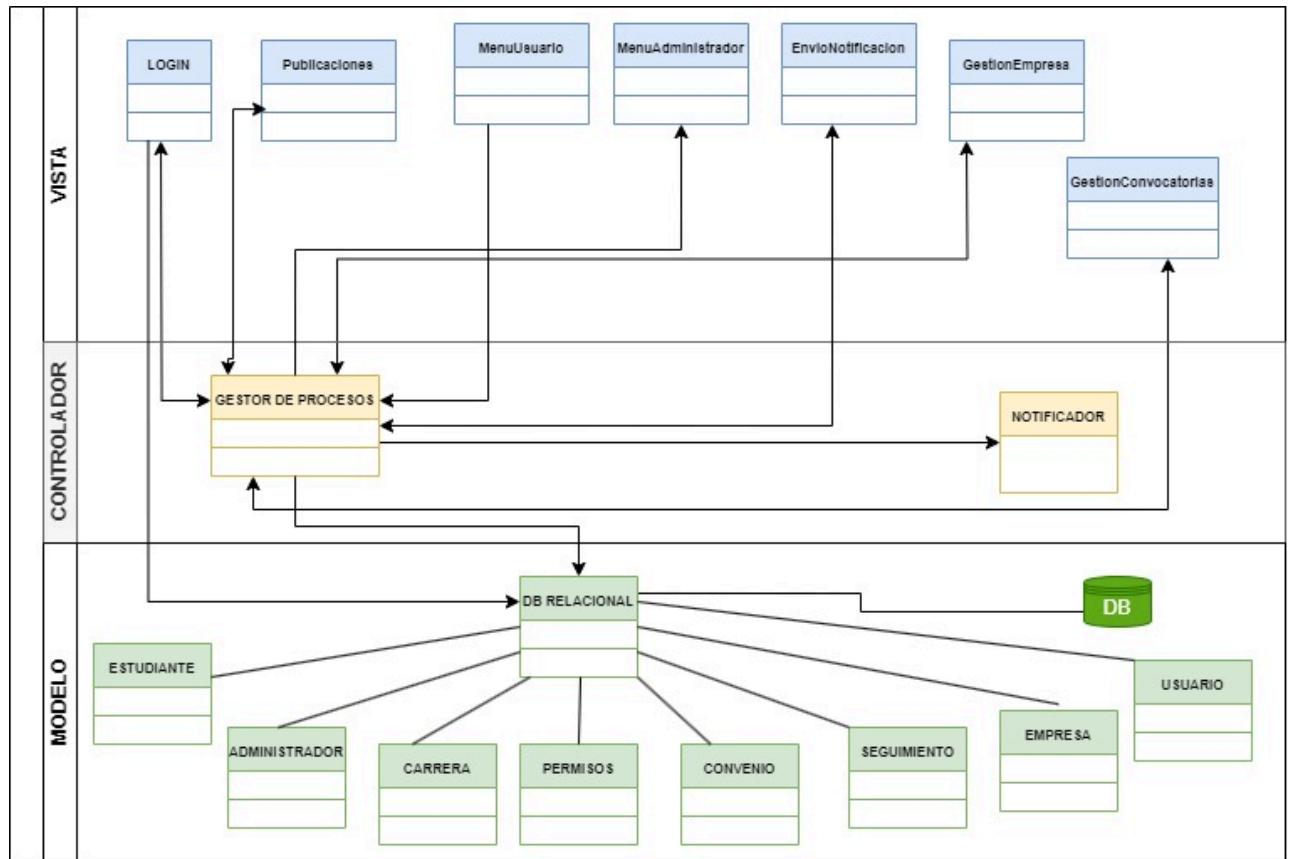
#### **3.2.1.3. Controlador (Controller):**

El controlador actúa como un intermediario que facilita la comunicación y la interacción entre el modelo y la vista. Se encarga de procesar las solicitudes de los estudiantes, verificar el cumplimiento de los requisitos, actualizar el modelo con la información de las pasantías y presentar los datos correspondientes en la vista. También asume la responsabilidad de gestionar la autenticación y autorización de usuarios, garantizando que solo los estudiantes autorizados tengan acceso a sus datos personales y de pasantía.

La asignación de tareas dentro de la arquitectura MVC se realiza de la siguiente manera:

- **Modelo (Model):** Responsable de la gestión de datos y lógica empresarial, incluyendo el seguimiento del progreso de pasantías, verificación de requisitos y seguridad de datos.
- **Vista (View):** Encargada de la presentación de datos de manera accesible y clara para los estudiantes, incluyendo la interfaz de solicitud de cartas de pasantías y el seguimiento de pasantías.
- **Controlador (Controller):** Actúa como intermediario que recibe y procesa las solicitudes de los estudiantes, verifica el cumplimiento de los requisitos, actualiza el modelo con la información pertinente y presenta datos en la vista. Además, gestiona la autenticación y autorización de usuarios para garantizar la seguridad de los datos y el acceso adecuado.

Esta arquitectura MVC permite una separación efectiva de las preocupaciones y facilita el mantenimiento y escalabilidad del sistema de gestión de pasantías en un entorno académico como es el de El instituto tecnológico “Escuela Industrial Superior Pedro Domingo Murillo” .



**Figura 3.6. Modelo Vista Controlador**

**Fuente:** elaboración propia

### 3.3. DIAGRAMA DE ENTIDAD – RELACIÓN

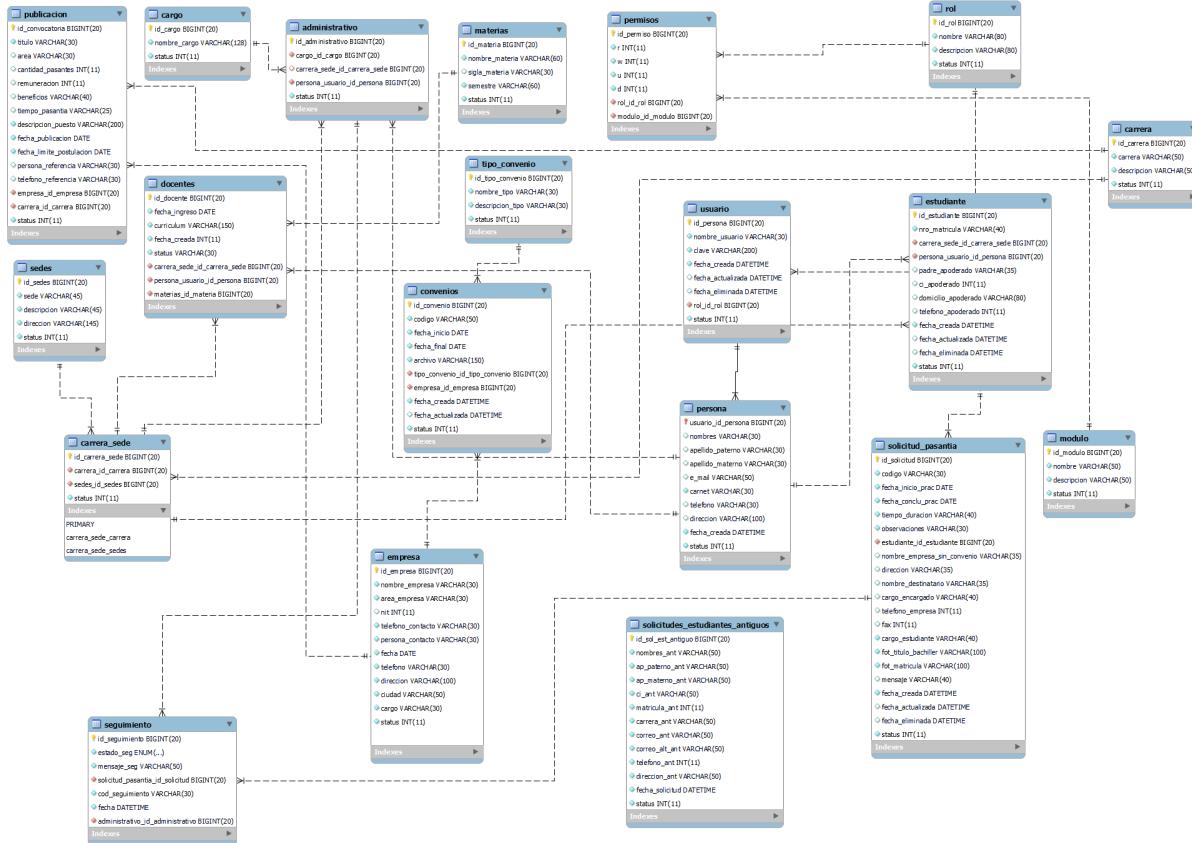


Figura 3.7. Modelo-R

Fuente: Elaboración propia

El diagrama de Entidad Relación muestra la relación de las tablas que interactúan para que se pueda cumplir ser los cimientos del sistema. El administrador podrá dar roles a los administrativos, estudiantes, para que puedan editar, agregar, eliminar, cada carrera tendrá sus propios datos de estudiantes, del administrador de tal carrera y sede.

### 3.4. ACTORES

- Rector
- Secretaría del rectorado

- Dirección académica
- Jefatura de carrera
- Empresas
- Estudiantes

### **3.5. CASOS DE USO**

#### **3.5.1. IDENTIFICAR CASOS DE USO**

**Caso de uso:** Inicio de sesión

**Actores:** Rector, secretaria de rectorado, dirección académica, jefatura de carrera, empresas y estudiantes

##### **Descripción**

- Rector, secretaria de rectorado, dirección académica, jefatura de carrera, deberán ingresar al sistema colocando su correo y contraseña designada.
- Las empresas ingresarán con los datos proporcionados por la secretaría de rectorado.
- El estudiante deberá ingresar al sistema colocando su correo institucional y contraseña (número de carnet).

**Caso de uso:** Validar Usuario

**Actores:** Rector, secretaria de rectorado, dirección académica, jefatura de carrera, empresas y estudiantes

##### **Descripción**

- Dependiendo del actor que sea el sistema mostrará limitaciones en cuanto a acciones y menús.

**Caso de uso:** Subir convenio

**Actores:** Secretaria de rectorado

**Descripción:**

- La Secretaría de Rectorado se encargará de subir al sistema todos los convenios vigentes que tenga la institución con las distintas empresas.
- Para esto se debe llenar el formulario: Con el nombre de la empresa, fecha expiración del convenio, área para el cual solicita pasantes, cuantos practicantes solicita, requerimientos que debe cumplir el postulante, el tiempo disponible para postularse, dirección, número de contacto y cualquier otro dato relevante que brinde la empresa.

**Caso de uso:** Modificar convenio

**Actores:** Secretaría de rectorado, rector

**Descripción:**

- Tanto el rector como la secretaría de rectorado podrán modificar los convenios en caso de existir errores en su información.

**Caso de uso:** Eliminar convenio

**Actores:** Rector, secretaria de rectorado

**Descripción:**

- El rector podrá eliminar convenios del sistema.
- La secretaría de rectorado podrá eliminar convenios del sistema cuando estos no se renueven.

**Caso de uso:** Alerta de expiración de convenio

**Actores:** Rector, secretaria de rectorado, jefatura de carrera

**Descripción:**

- Al incluir un convenio en el sistema también se deberá colocar la fecha de vigencia de este.
- El sistema se encargará de mostrar una alerta con el tiempo de vigencia que le queda a cada convenio.

**Caso de uso:** Subir información sobre pasantía

**Actores:** Secretaría de rectorado, empresas

**Descripción:**

- La empresa deberá acudir a secretaría donde le proporcionará datos para ingresar al sistema y poder ingresar información sobre la pasantía que ofrecen mediante un formulario.
- Las empresas al ingresar al sistema deberán ser dirigidas directamente al formulario sin poder ver el resto de la información.

**Caso de uso:** Acceso a información de prácticas en la industria

**Actores:** Jefatura de carrera, estudiantes

**Descripción:**

- El estudiante una vez dentro del sistema deberá ingresar a la carrera que pertenece y a la pestaña de prácticas en la industria, aparecerá un formulario donde el estudiante deberá colocar; nombre completo, número de matrícula y semestre al que pertenece.
- El encargado de carrera deberá comprobar la veracidad de esta información, una vez comprobada se le dará o negará el acceso al estudiante.

**Caso de uso:** Enviar solicitud de pasantía

**Actores:** Dirección académica, jefatura de carrera, estudiantes

**Descripción:**

- Cuando el estudiante solicite la aprobación para la pasantía el sistema le pedirá llenar el formulario PRI-100 y subir la documentación pertinente.
- Jefatura de carrera recibirán esta información donde será revisada, adjuntando la documentación necesaria se entregará a dirección académica.

- Dirección académica se encargará de verificar que toda la documentación está en orden.

**Caso de uso:** Respuestas a las prácticas en la industria

**Actores:** Dirección académica, jefatura de carrera, estudiantes

**Descripción:**

- Si los documentos enviados por el estudiante son correctos, la jefatura de carrera enviará de manera interna a dirección académica, de existir observaciones se las hará conocer al estudiante mediante una notificación.
- En dirección académica se preparará la carta dependiendo de si toda la documentación fue entregada, se enviará una notificación al estudiante en la cual se le indicará si todo se encuentra en orden y su solicitud ha sido aprobada o si tiene alguna observación en cualquiera de los dos casos se le indicará la fecha en la que debe pasar a la institución con los documentos físicos que fueron enviados al sistema.

### 3.6. DIAGRAMAS DE CASOS DE USO

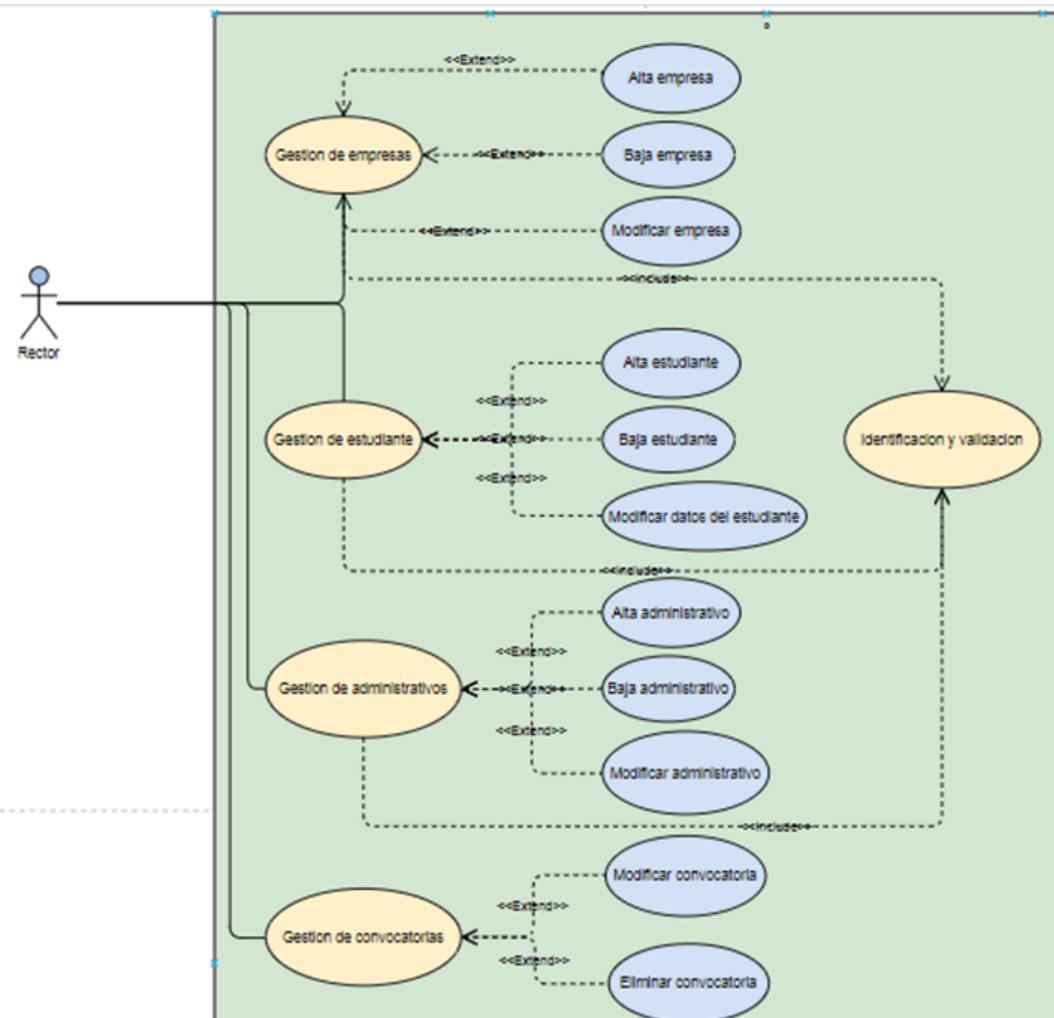


Figura 3.8. Diagrama general

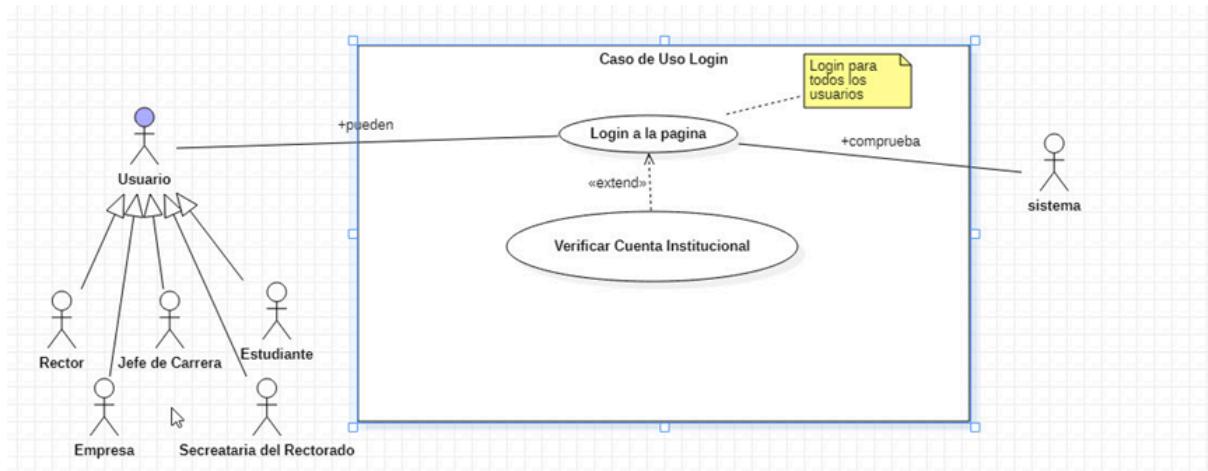


Figura 3.9. Caso de uso Login

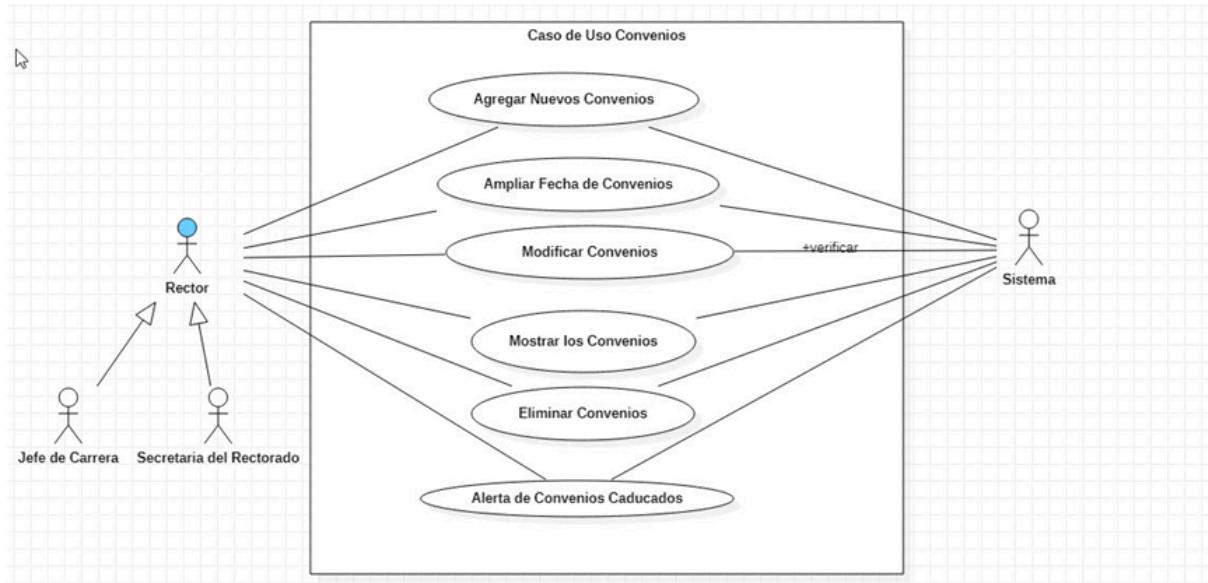


Figura Nro. 3.10 Convenios

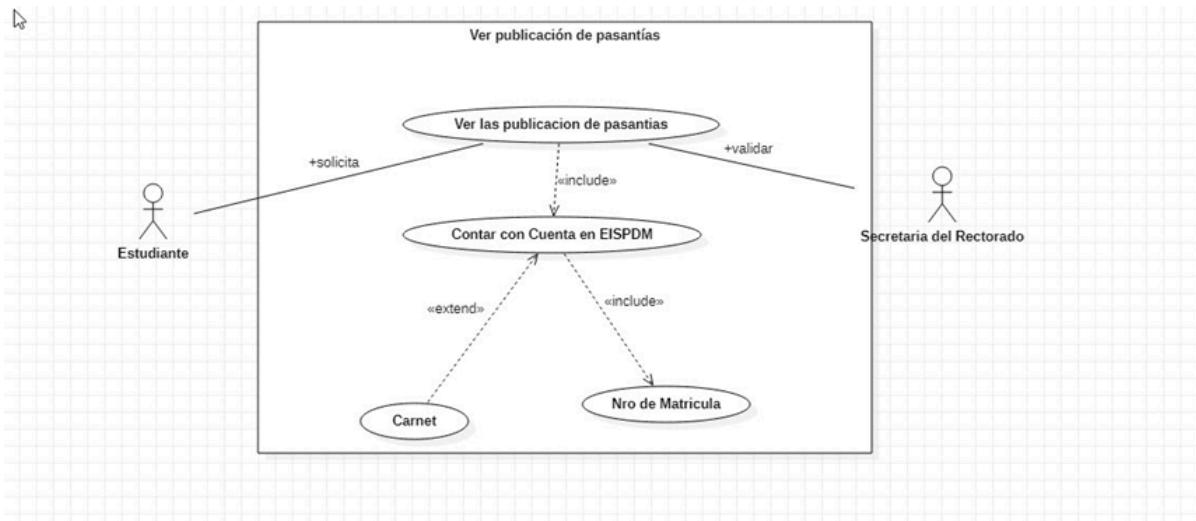


Figura Nro. 3.11. Publicaciones

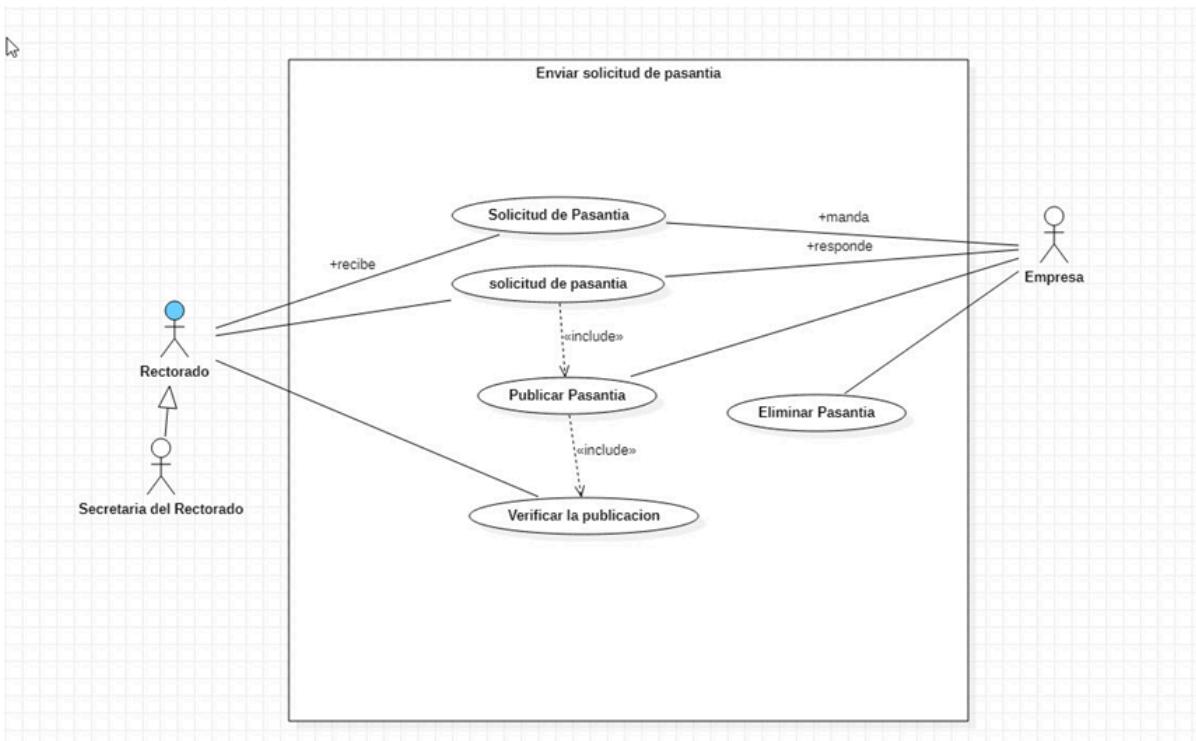


Figura Nro. 3.12 Solicitud de Prácticas en la industria

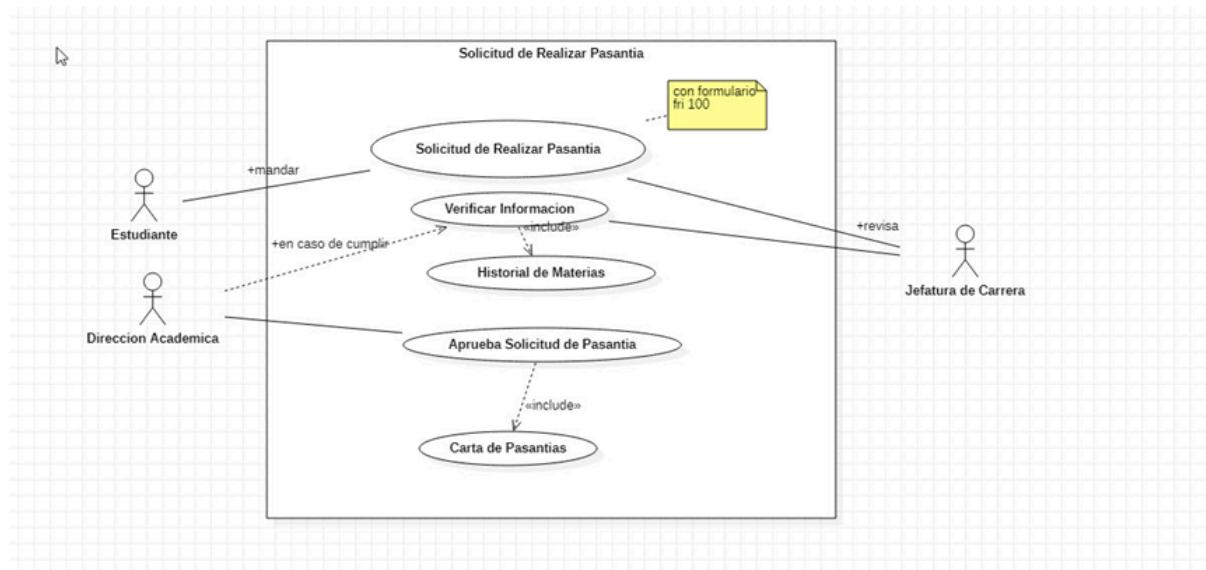


Figura Nro. 3.13 Realizar la solicitud

### 3.7. DIAGRAMA DE CASO DE USO EXTENDIDO

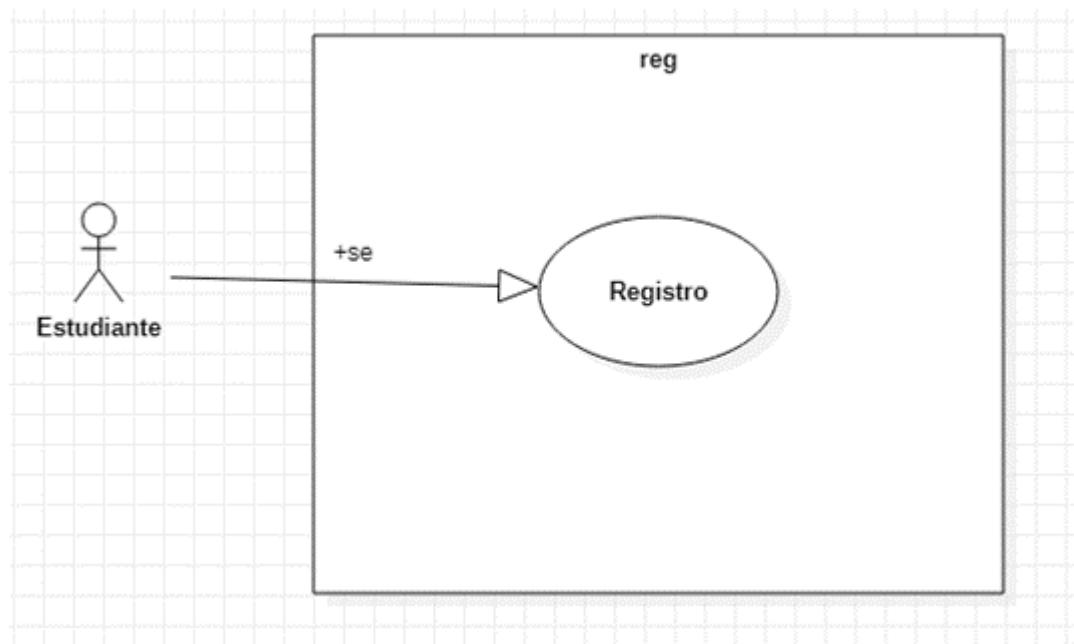


Figura Nro. 3.14 Registro de Estudiante

**Nombre:** Registro de Estudiante

**Descripción:** El usuario la primera vez que ingrese debe registrarse

**Actores:** Estudiante

**Precondiciones:** ninguno

**Requisitos no funcionales:** ninguno

**Flujo de eventos:**

1. El estudiante para obtener una cuenta debe proveer sus datos personales para registrarse en la plataforma de búsqueda de prácticas en la industria.
2. Una vez que el estudiante ya obtuvo su cuenta podrá ver todas las publicaciones de las empresas.

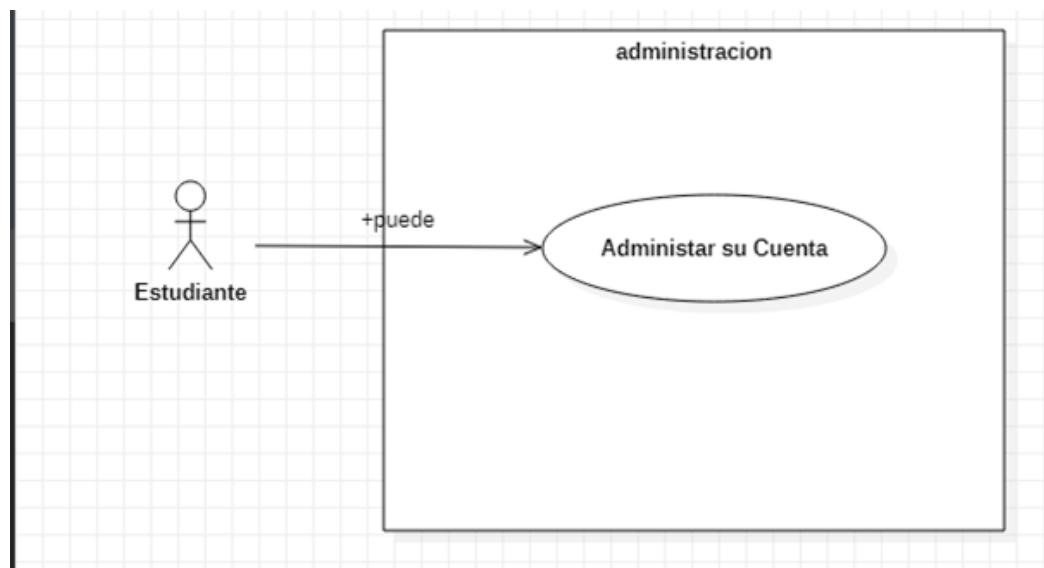


Figura Nro. 3.15 Administración de Cuentas

**Nombre:** Administración de cuentas

**Descripción:** El estudiante podrá actualizar sus datos personales, como cambiar el nombre, cambiar la contraseña, foto del perfil

**Actores:** Estudiante

**Precondiciones:** tener cuenta en la plataforma

**Requisitos no funcionales:** ninguna

**Flujo de eventos:**

1. Ingresar a su cuenta personal
2. Editar sus datos personales.

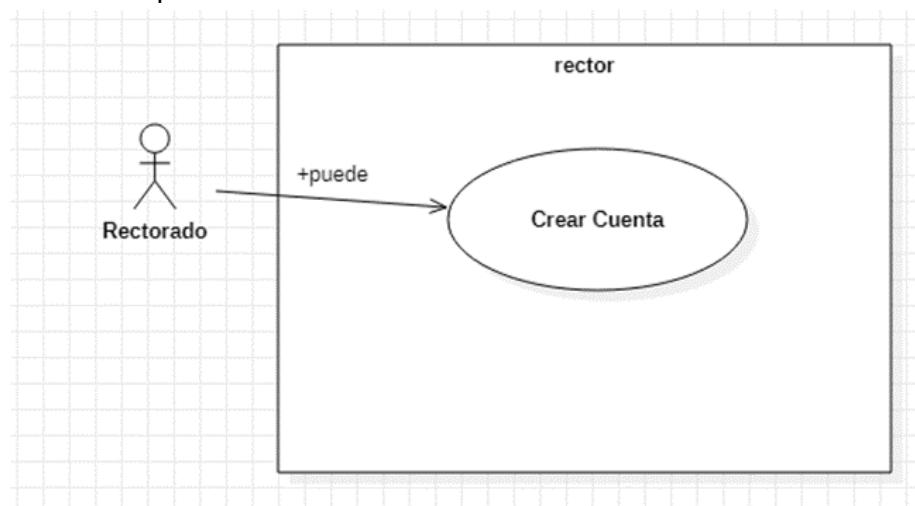


Figura Nro. 3.16 Crear Cuenta

**Nombre:** Crear Cuenta

**Descripción:** El usuario la primera vez que ingrese debe registrarse, para esto debe seguir los pasos correspondientes:

En caso de rectorado, tienen que brindar su nombre, carnet, cargo en la institución y contraseña

**Actores:** Rectorado

**Precondiciones:** ninguna

**Requisitos no funcionales:** ninguna

**Flujo de eventos:**

1. Debe crear una cuenta nueva.
2. Rellenar todos los datos en el formulario.

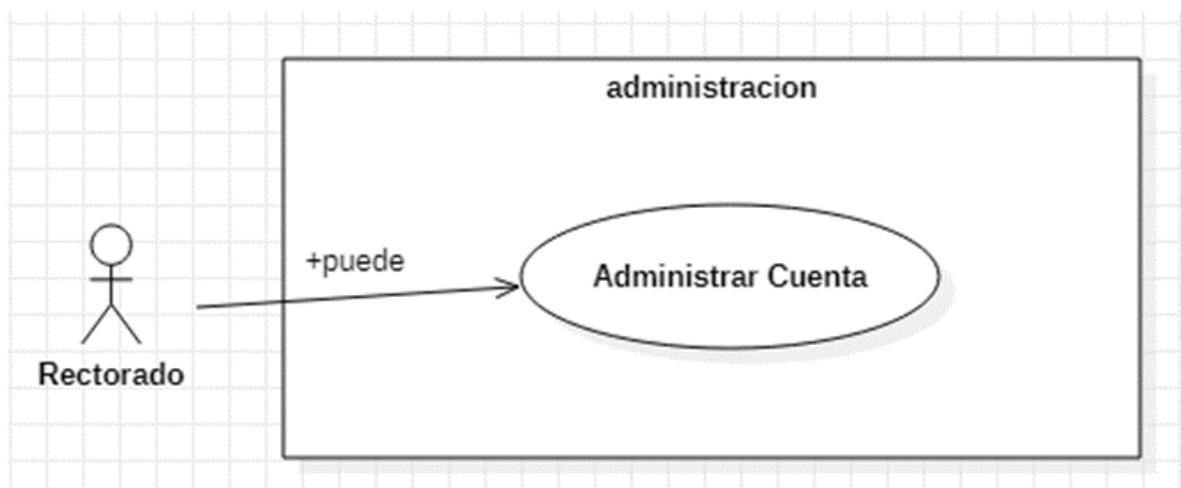


Figura Nro. 3.17 Administración de cuentas Rectorado

**Nombre:** Administración de cuentas Rectorado

**Descripción:** Rectorado podrá ver las empresas que están registrados en la plataforma tanto los de convenios o invitados, así mismo todos los estudiantes registrados en la plataforma.

**Actores:** Rectorado

**Precondiciones:** contar con una cuenta

**Requisitos no funcionales:** ninguna

### **Flujo de eventos:**

1. Control de empresas con convenio.
2. Control de estudiantes de la escuela, como ser eliminar datos del estudiante.
3. Renovar contratos de las empresas de convenio.
4. Podrá dar acceso a las empresas para que puedan tener una cuenta en la plataforma previa validación

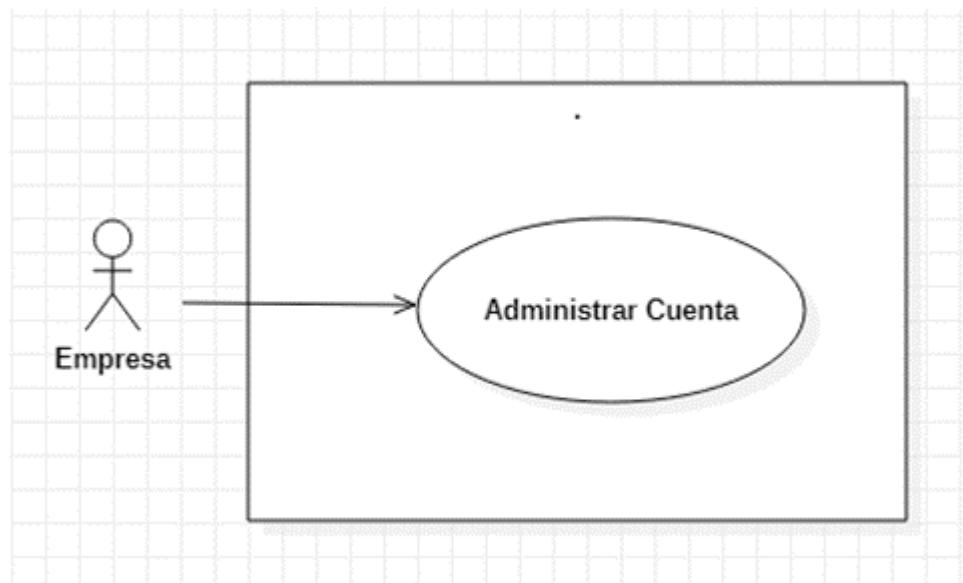


Figura Nro. 3.18 Crear cuenta de la empresa o agregar cuenta

**Nombre:** Crear cuenta de la empresa o agregar cuenta

**Descripción:** En caso de una empresa, debe registrar el nombre de dicha empresa, ubicación, numero de contacto, imagen de esta, correo, contraseña

**Actores:** Empresa

**Precondiciones:** ninguna

**Requisitos no funcionales:** ninguna

**Flujo de eventos:**

1. Debe crear una cuenta nueva.
2. Rellenar todos los datos en el formulario.

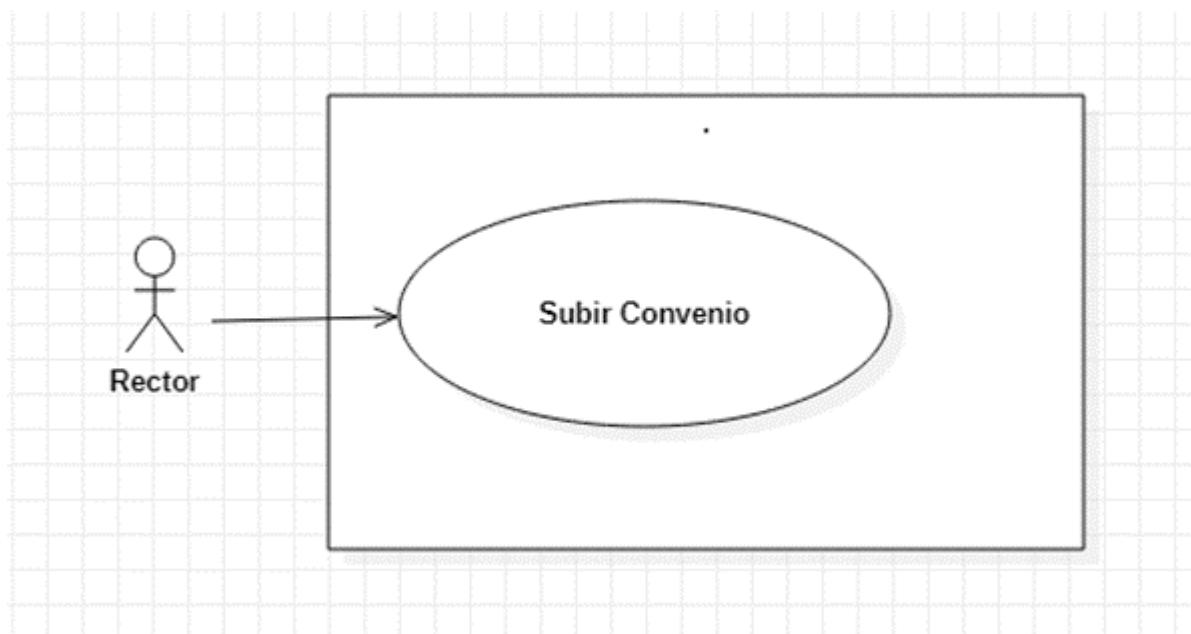


Figura Nro. 3.19 Subir Convenio

**Nombre:** Subir Convenio

**Descripción:** Rectorado se encargará de subir al sistema todos los convenios vigentes que tenga la institución con las distintas empresas.

Para esto se debe llenar el formulario: Con el nombre de la empresa, fecha expiración del convenio, área para el cual solicita pasantes, cuantos practicantes solicita, requerimientos que debe cumplir el postulante, el tiempo disponible para postularse, dirección, número de contacto y cualquier otro dato relevante que brinde la empresa

**Actores:** Rectorado

**Precondiciones:** tener una cuenta

**Requisitos no funcionales:** ninguna

Flujo de eventos:

1. Rectorado subirá al sistema todos los datos de las empresas de convenios.
2. Mandará la cuenta y contraseña a la empresa.
3. Registrará fecha límite de convenio

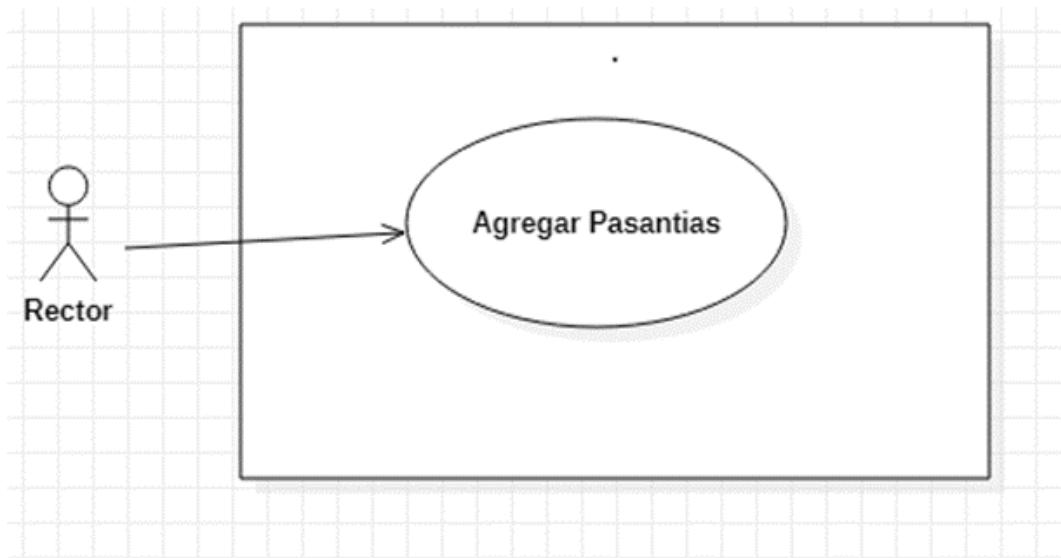


Figura Nro. 3.20 Agregar prácticas en la industria

**Nombre:** Agregar prácticas en la industria

**Descripción:** El sistema admitirá las prácticas en la industria de las empresas, pero antes de ser publicadas esta pasara por rectorado para que puedan verificar la veracidad de dichas prácticas en la industria, una vez verificada estas prácticas en la industria recién podrán ser vistas por los usuarios

**Actores:** Rectorado

**Precondiciones:** solicitud aceptada de la institución

**Requisitos no funcionales:** ninguna

**Flujo de eventos:**

1. Se recibirá las notificaciones de las prácticas en la industria
2. Se verificará las prácticas en la industria enviadas por las empresas para ser publicadas en la plataforma.
3. Se aceptará las prácticas en la industria o caso contrario serán devueltos para su verificación.
4. Se publicará las prácticas en la industria en la plataforma

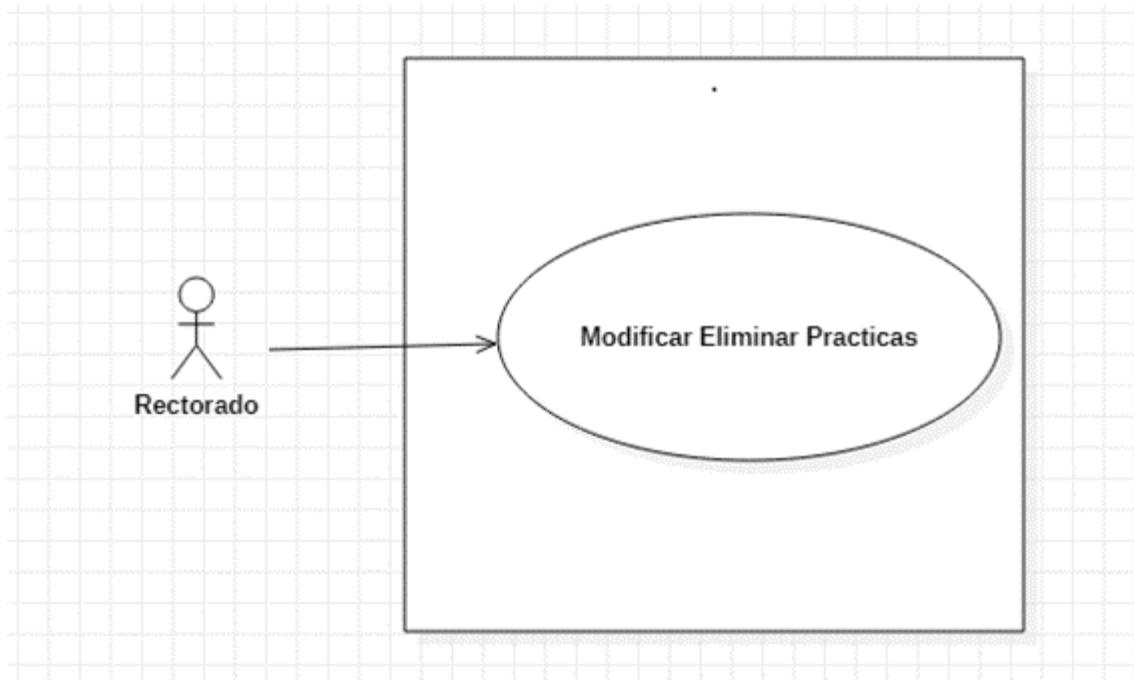


Figura Nro. 3.21 Modificar o eliminar información de prácticas en la industria

**Nombre:** Modificar o eliminar información de prácticas en la industria

**Descripción:** Los únicos que podrán alterar dichas prácticas en la industria será rectorado ya que cuando ingresen, ellos serán los que cuenten con más acciones en el sistema como ser: publicar, verificar, eliminar la información.

Al haber expirado el tiempo de un convenio el sistema notificará esto a rectorado para que este pueda ser eliminado, o en caso de ser renovado cambiar la fecha de término del convenio.

**Actores:** Rectorado

**Precondiciones:** contar con una cuenta

**Requisitos no funcionales:** ninguna

### **Flujo de eventos:**

1. Verificar contenido de prácticas en la industria.
2. Alterar las prácticas en la industria
3. Eliminar prácticas en la industria
4. Renovar convenios.
5. Eliminar convenios.

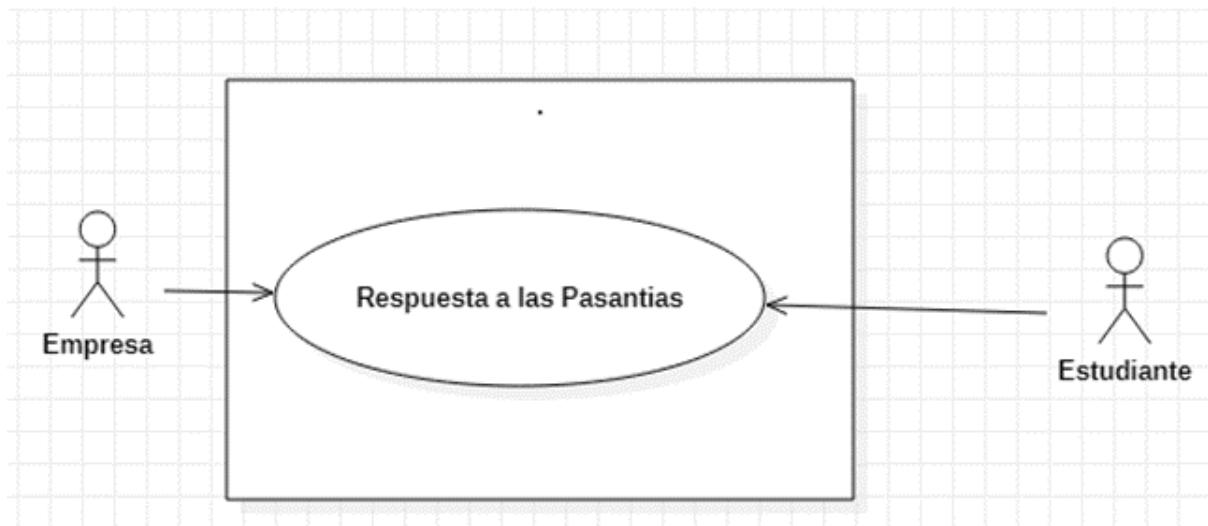


Figura Nro. 3.22 Respuestas a las prácticas en la industria

**Nombre:** Respuestas a las prácticas en la industria

**Descripción:** Los estudiantes cuando ingresen tendrán un panel donde estarán todas las prácticas en la industria, de las cuales verán la información al ingresar en ellas, solo podrán ver, pero no realizar ningún cambio en estas, además podrán postularse a dicha pasantía donde tendrán que enviar sus documentos al correo de la empresa para que puedan ser revisados.

**Actores:** Empresas, Estudiantes

**Precondiciones:** contar con una cuenta

**Requisitos no funcionales:** ninguna

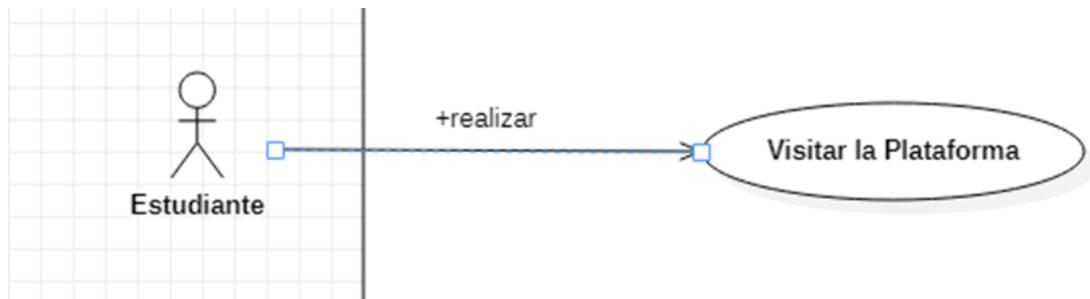


Figura Nro. 3.23 Visitar Plataforma

**Nombre:** Visitar Plataforma

**Descripción:** Permite al estudiante ver las publicaciones de las pasantías en la plataforma web.

**Actores:** Estudiante

**Precondiciones:** Ninguna

**Requisitos no funcionales:** ninguno

**Flujo de Eventos:**

1. El estudiante busca en el internet acerca de las pasantías
2. El estudiante visita la plataforma de búsqueda de pasantías, y busca las pasantías que están publicadas en la plataforma ya sean instituciones con convenio y sin convenio para poder realizar sus pasantías.

**Post Condiciones:** tener cuenta institucional.

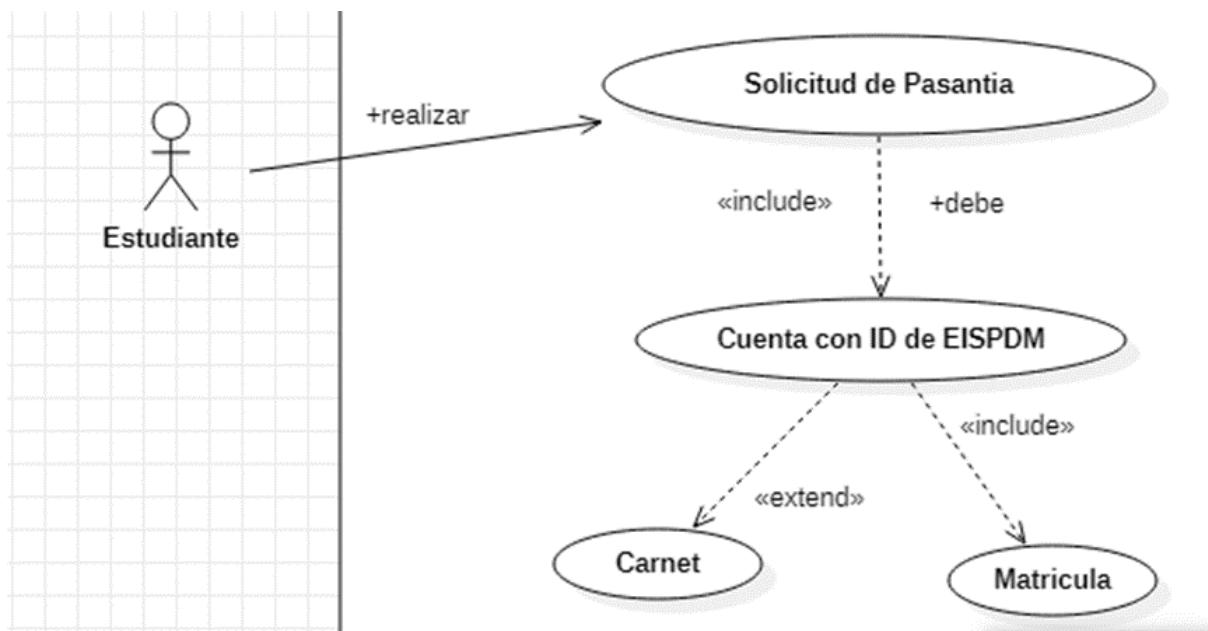


Figura Nro. 3.24 Solicitud de Pasantía

**Nombre:** Solicitud de Pasantía

**Descripción:** El estudiante puede realizar una solicitud de pasantía donde deberá contar con una cuenta institucional ya sea su carnet o matrícula.

**Actores :** Estudiante

**Precondiciones:** Ninguna

**Requisitos no funcionales:** ninguno

### Flujo de eventos:

1. El estudiante una vez estando en la plataforma busca las pasantías.
2. El estudiante puede mandar una solicitud a una empresa siempre y cuando tenga una cuenta en la plataforma con su id de matrícula.
3. También el estudiante podrá usar su carnet que está enlazada con una cuenta institucional.

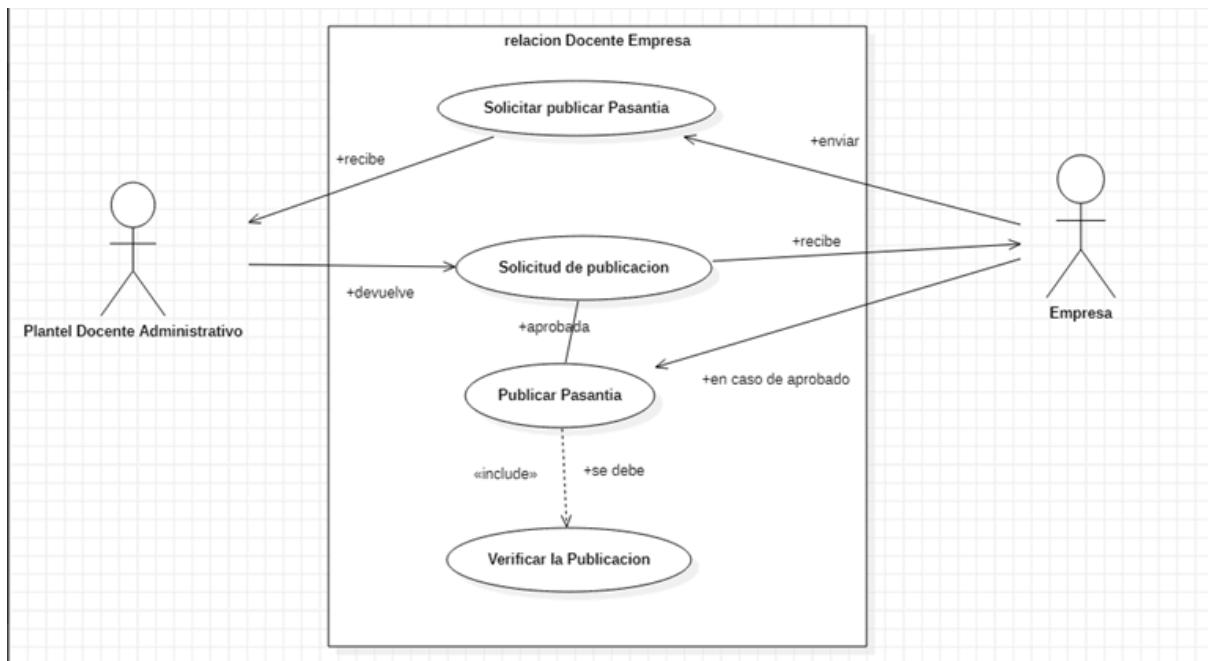


Figura Nro. 3.25 Administrativo - Empresa

**Nombre:** Administrativo - Empresa

**Descripción:** El Plantel Administrativo del I.T. "E.I.S.P.D.M." tiene el control de la plataforma de pasantías con ciertos privilegios de administrador, en este caso específicamente en recibir solicitudes de las empresas interesadas en realizar una publicación de pasantía en la institución.

Por otra parte, la empresa puede mandar solicitudes del I.T. "E.I.S.P.D.M." para poder publicar las pasantías, una vez aceptadas estas solicitudes pueden ser publicadas, siempre y cuando el contenido de la pasantía sea verificado por la institución.

**Actores:** Plantel Administrativo, Empresa

**Precondiciones:** Ninguno

**Requisitos no funcionales:** Ninguno

**Flujo de Evento:**

1. La empresa interesada envía una solicitud a la institución para contactarse.
2. El plantel Administrativo recibe las solicitudes, mediante la plataforma, de diferentes instituciones o empresas que desean realizar convenios o solo ser invitados para publicar las pasantías.
3. El plantel administrativo de acuerdo a tipo de solicitud puede devolver la información a la empresa solicitante. Las mismas empresas serán verificadas seriamente.
4. La empresa una vez recibido el visto bueno de parte de la institución podrá firmar el convenio o solo ser invitado.
5. La empresa publicará las pasantías en la plataforma del I.T. "E.I.S.P.D.M." para buscar estudiantes interesados que puedan hacer sus pasantías en la institución y así otorgar la certificación a los estudiantes.
6. El plantel administrativo aprobará todas las publicaciones que se van a realizar en la plataforma de pasantías de la institución.

**Post condiciones:** ninguna

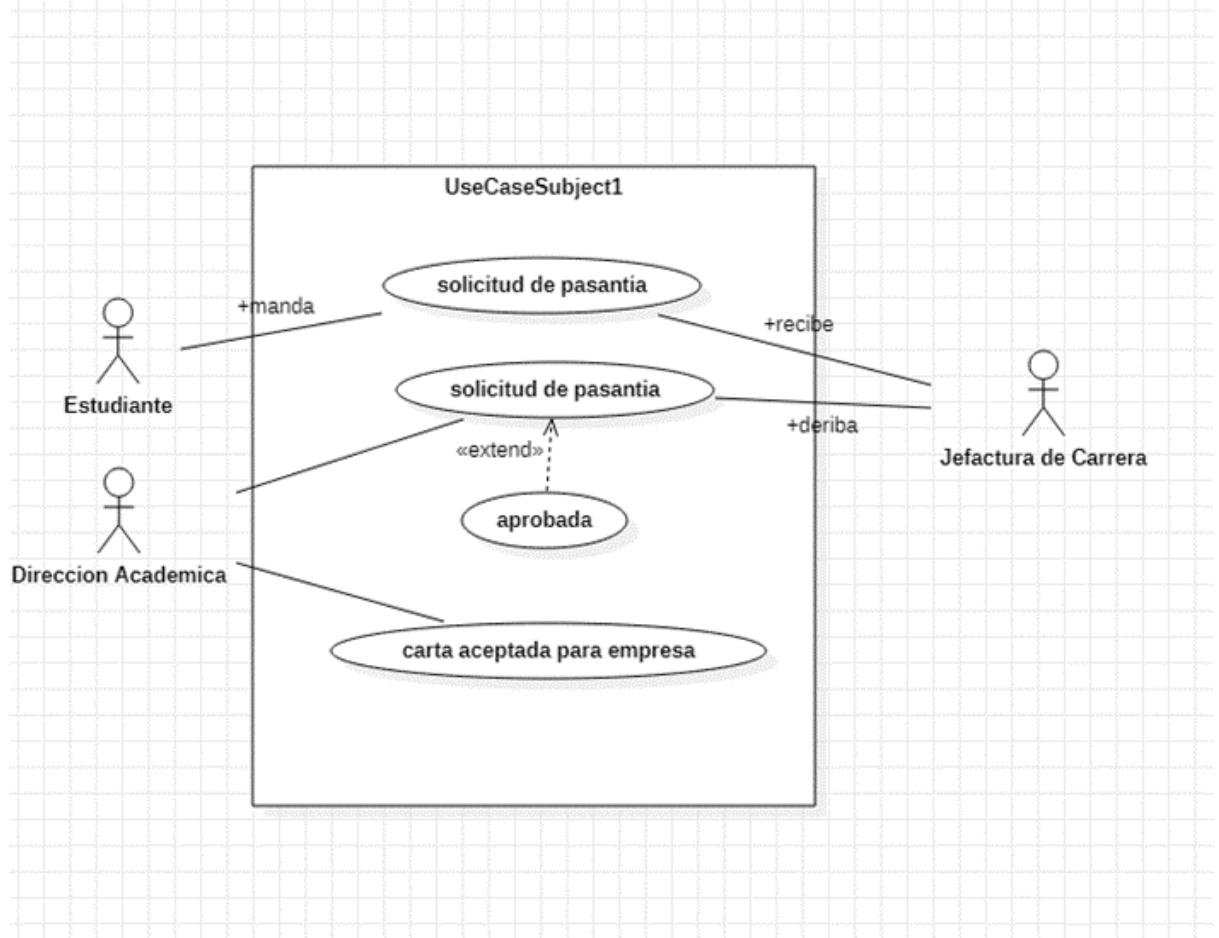


Figura Nro. 3.26 Enviar solicitud de Pasantía

**Nombre:** Enviar solicitud de pasantía

**Descripción:** Cuando el estudiante solicite la aprobación para la pasantía el sistema solicitará llenar el formulario PRI -100 y subir la documentación pertinente.

Encargados de carrera recibirán esta información donde será revisada y entregada a secretaría de académica.

Secretaría de académica se encargará de verificar que toda la documentación está en orden, depende de esto se me hará una notificación al estudiante en la cual se le indicará si todo se encuentra en orden y su solicitud ha sido aprobada o si tiene alguna

observación en cualquiera de los dos casos se le indicará la fecha en la que debe pasar a la institución.

**Actores:** Rector, Dirección Académica, Jefe de Carrera, estudiantes

**Precondiciones:** Ninguno

**Requisitos no funcionales:** Ninguno

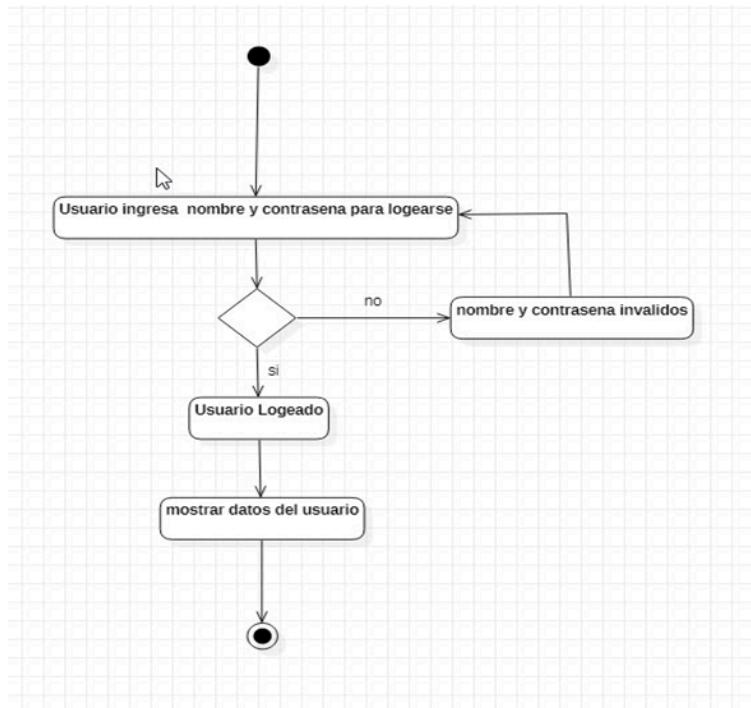
**Flujo de Evento:**

1. Mandar carta de solicitud al jefe de carrera.
2. Jefe de carrera recibe la solicitud
3. Acepta la solicitud del estudiante
4. Una vez aceptada la solicitud con sello de jefatura
5. Envío a Dirección Académica.
6. Dirección Académica acepta la carta de solicitud
7. Enviar la carta al estudiante para que pueda remitir la misma a la empresa.

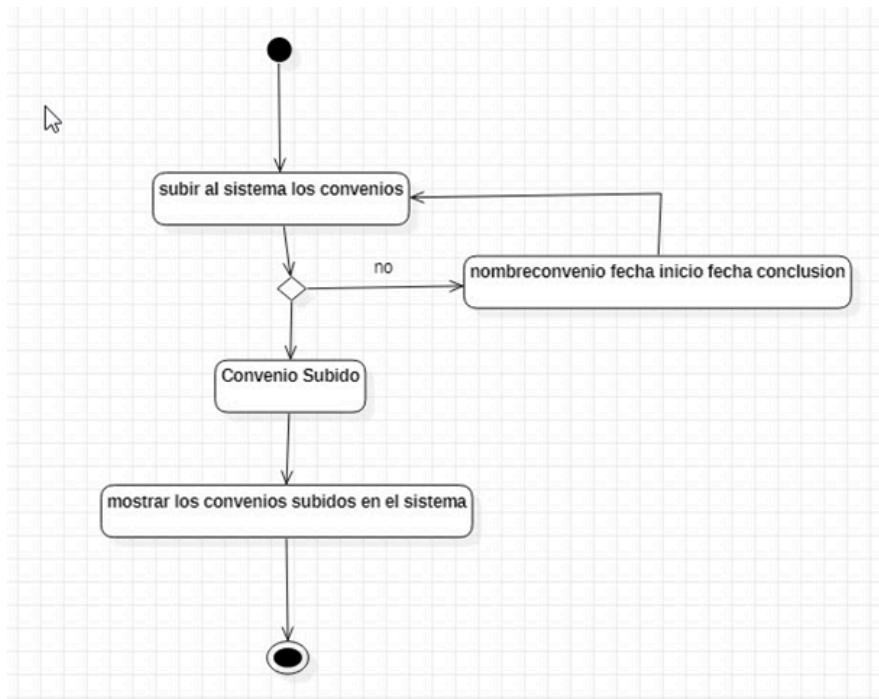
**Post condiciones:** ninguna

## 3.8. DIAGRAMA DE ACTIVIDADES

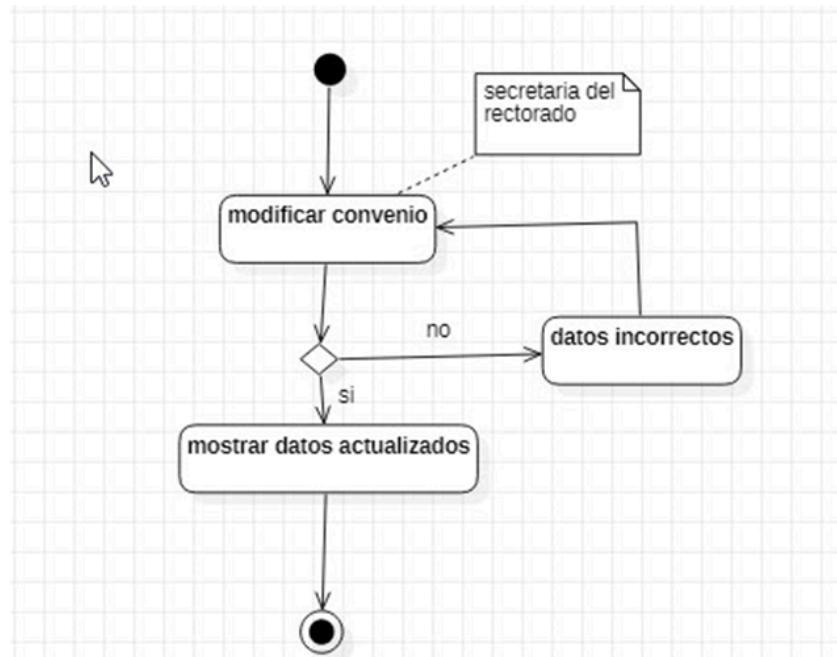
### 3.8.1. Inicio de sesión



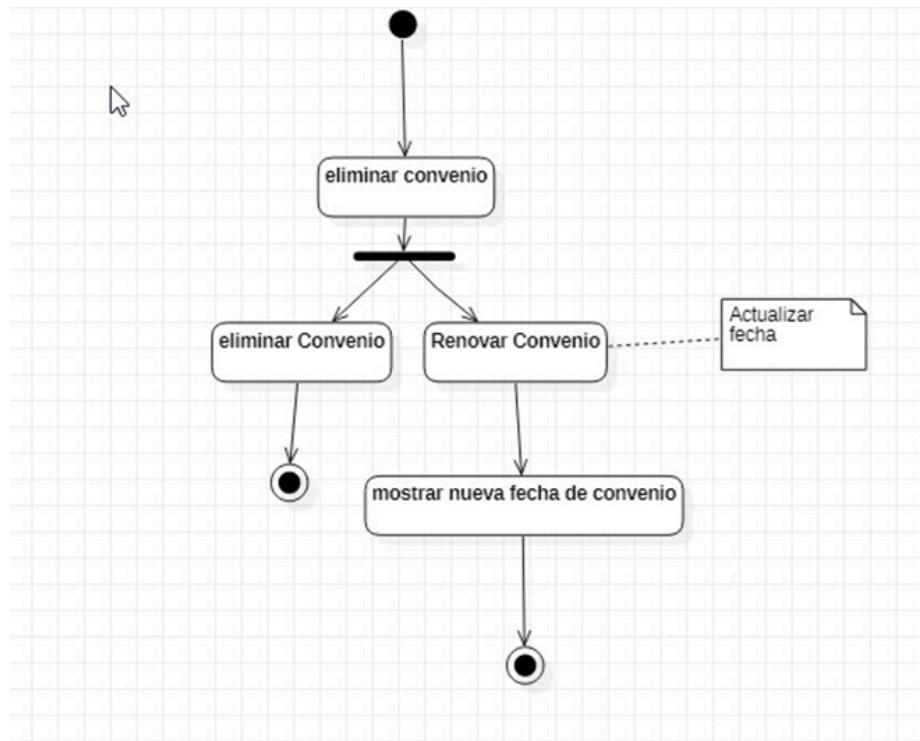
### 3.8.2. Subir al sistema los convenios



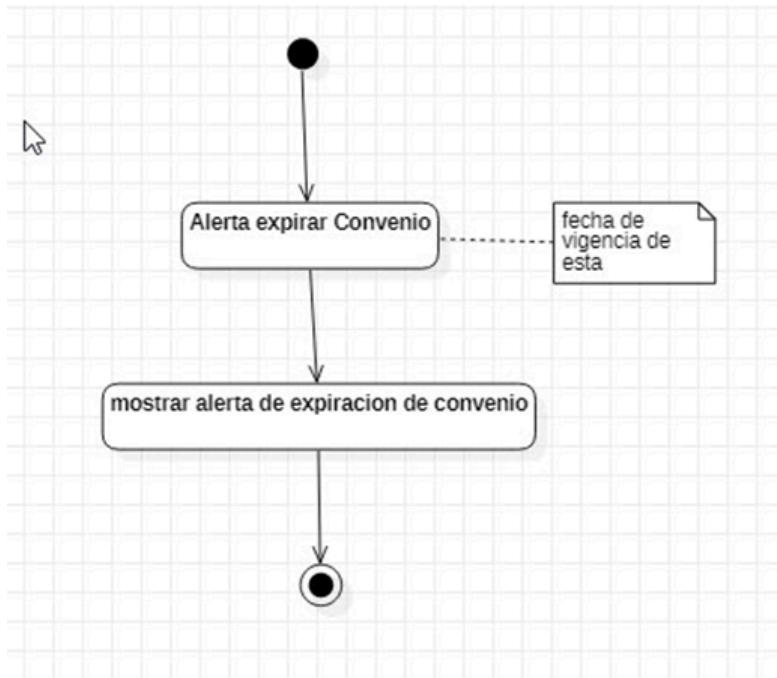
### 3.8.3. Modificar los convenios



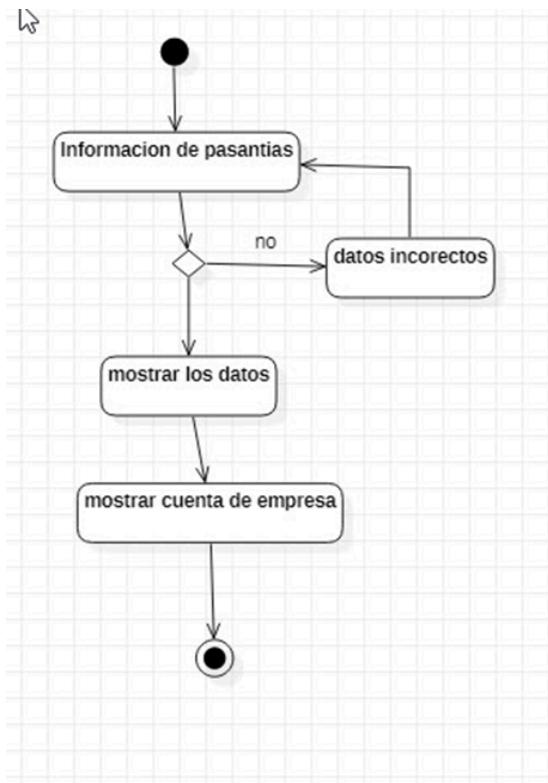
### 3.8.4. Eliminar Convenios



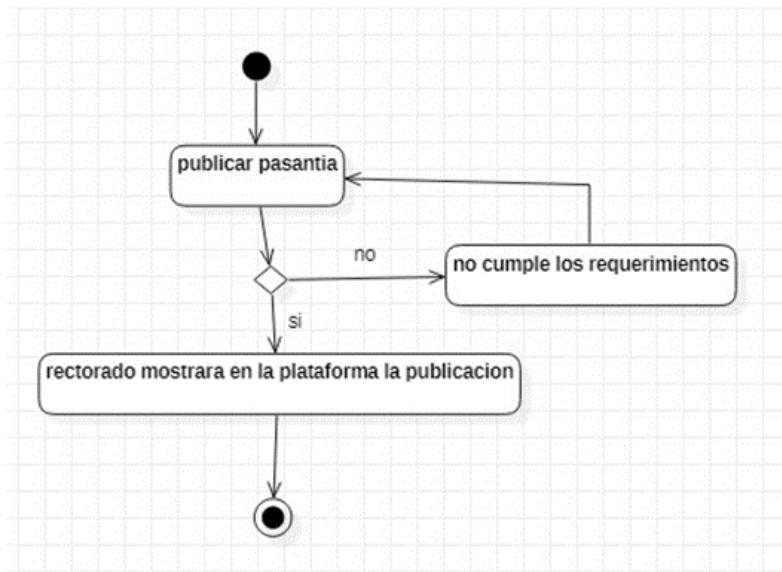
### 3.8.5. Alerta de expiración de convenios



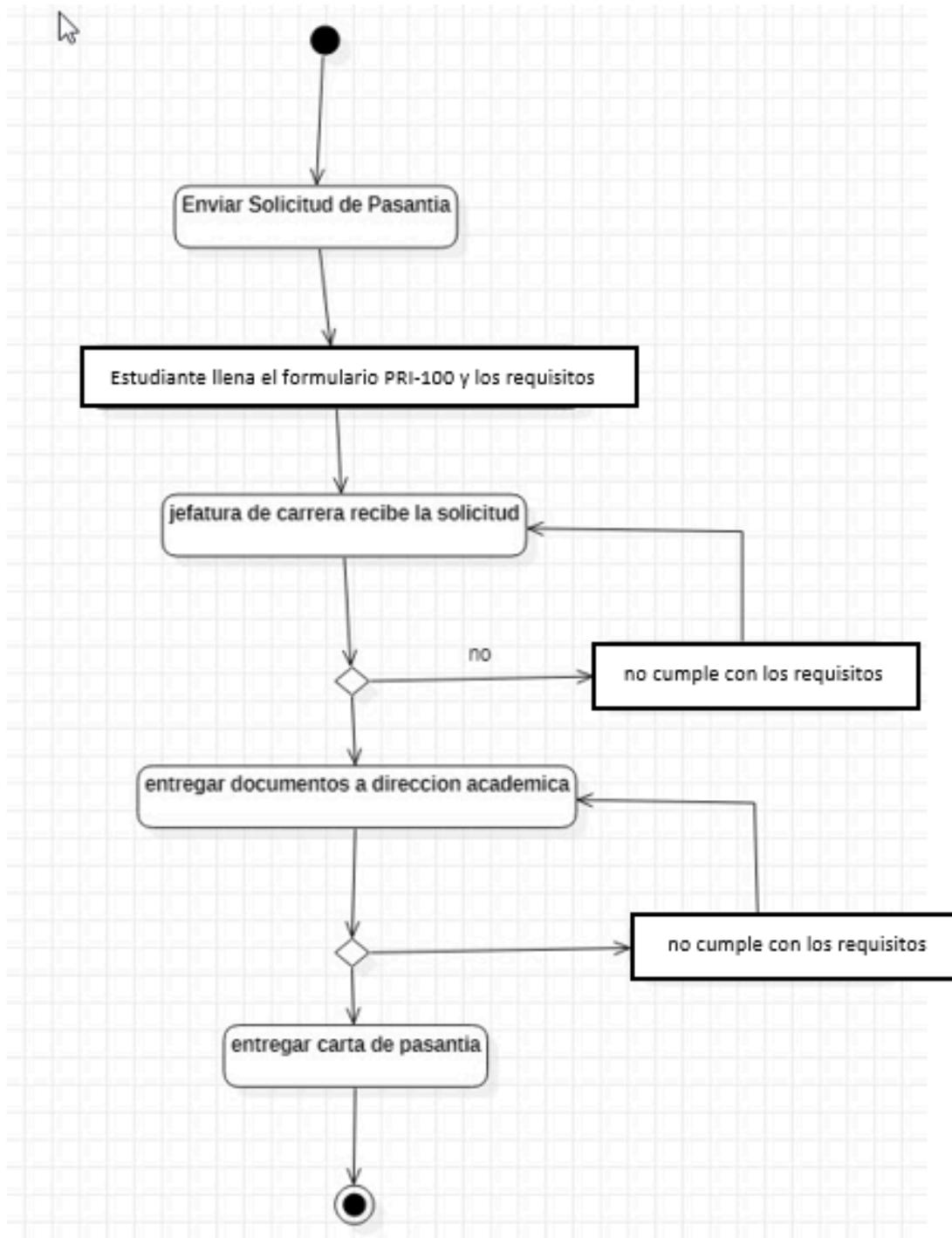
### 3.8.6. Información de pasantías



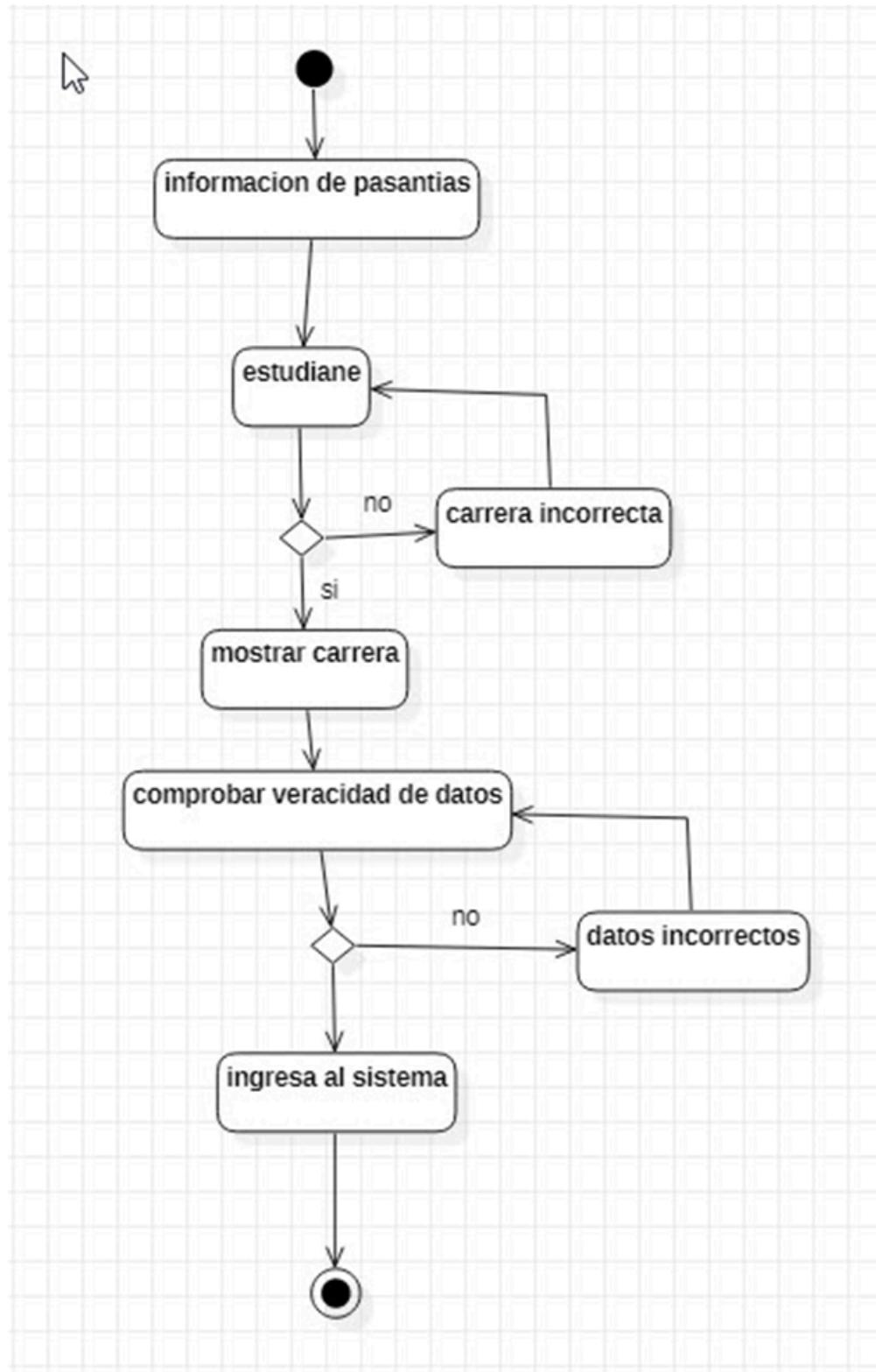
### 3.8.7. Empresa o Institución Publicar pasantías



### 3.8.8. Enviar solicitud de pasantía



### 3.8.9. Información de Pasantías



### 3.9. MAQUETACIÓN WEB



Figura Nro. 3.8 Login



Figura Nro. 3.9 Panel Admin

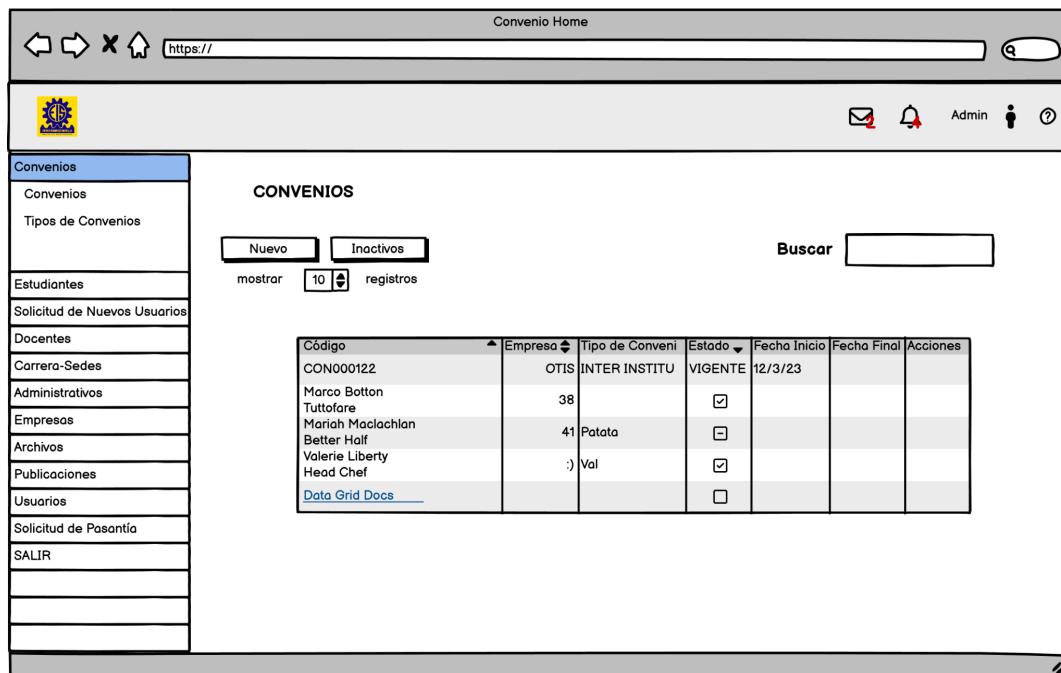


Figura Nro. 3.10 Panel Convenio

### **3.10. FASE PRE GAME**

La fase inicial es de gran relevancia en nuestro enfoque basado en Scrum. En esta etapa, conocida como "Pregame" en Scrum, nuestro objetivo principal radica en la construcción del Product-Backlog. Esto se alinea con las prácticas y roles definidos por la metodología Scrum que se está aplicando en este proyecto.

### **3.11. CONCEPCIÓN Y EXPLORACIÓN**

Durante esta etapa, se llevaron a cabo gradualmente tres actividades con el propósito de recopilar requisitos, los cuales se encuentran sintetizados en la Tabla 3.1 de manera resumida.

Tareas realizadas para la obtención de requisitos

TAREA	DESCRIPCIÓN DE LA TAREA
Entrevistas Personales	Durante esta actividad, se llevaron a cabo entrevistas personales con las autoridades y personal administrativo del I.T."E.I.S.P.D.M" El objetivo principal de estas entrevistas fue recopilar información clave sobre el procedimiento y requisitos específicos para el sistema en desarrollo.

Observación	<p>En esta actividad, se realizó un proceso de observación detallada de los procedimientos y requisitos solicitados en las distintas carreras del I.T."E.I.S.P.D.M" . Esto incluyó la observación de cómo se gestiona la documentación y cualquier otro aspecto relevante para el sistema en desarrollo.</p> <p>El propósito fue adquirir una comprensión más profunda de los procesos existentes en las carreras del I.T."E.I.S.P.D.M" y cómo podrían integrarse en el nuevo sistema.</p>
Documentación	<p>En esta actividad, se procedió a documentar minuciosamente todas las observaciones realizadas durante la etapa de observación en las carreras del I.T."E.I.S.P.D.M". Esta documentación será esencial para comprender y optimizar los procesos dentro del nuevo sistema que estamos desarrollando.</p>

Tabla 3.1

Fuente: Elaboración propia

### **3.12. ROLES SCRUM**

En esta sección, se presenta de manera clara y concisa los roles Scrum asignados en el proyecto. Los roles definidos para este proyecto se resumen en la Tabla 3.2 a continuación:

#### **3.12.1. Roles Scrum Asignados en el Proyecto**

ROLES	
PRODUCT OWNER:	Juan Carlos Mendoza Gutiérrez
SCRUM MASTER:	Juan Carlos Mendoza Gutiérrez
EQUIPO DE DESARROLLO:	Juan Carlos Mendoza Gutiérrez
USUARIOS:	Personal interno y partes interesadas del I.T.” E.I.S.P.D.M.” encargados de utilizar y gestionar el sistema web para la gestión de pasantías.

Tabla 3.2

Fuente: Elaboración propia

### **3.13. OBTENCIÓN DE REQUERIMIENTOS**

En esta sección, se detallan los requerimientos del sistema obtenidos a través de la interacción con el cliente y futuros usuarios. La Tabla 3.3 presenta una lista de requerimientos con sus respectivas descripciones:

#### **Lista de requerimientos**

Requerimiento	Descripción
Registrar convenios y solicitudes de pasantes	El administrador debe ser capaz de registrar la información de los productos nuevos que ingresan a la empresa.
Control de acceso para el usuario	El sistema debe permitir a los usuarios acceder al sistema mediante un usuario y contraseña asignados por el administrador
Administración de usuarios	El sistema debe contar con un módulo para administrar usuarios, incluyendo ajustes básicos y asignación de roles.
Gestión de solicitudes de pasantía	Los usuarios con privilegios administrativos deben poder crear, listar, modificar y eliminar registros de estudiantes, convenios y pasantías.
Seguimiento de solicitudes de pasantía	El usuario con rol de estudiante podrá realizar la verificación y seguimiento de su trámite
Clasificación convenios para cada carrera	Debe ser posible crear, listar, modificar y eliminar categorías, así como asignar una carrera determinada cada convenio y solicitudes de pasantías
Registro de convenios	Se debe registrar cada nuevo convenio realizado con la institución
Registro de datos de las empresas	Se debe registrar todos los datos de las empresas o instituciones tanto públicas como privadas

Tabla 3.3

Fuente: Elaboración propia

### 3.14. PRODUCT-BACKLOG

A partir de la lista de requerimientos definidos previamente (consultar Tabla 3.3), se procedió a realizar un análisis detallado de cada uno de ellos, en estrecha colaboración con el personal del I.T."E.I.S.P.D.M". Este proceso culminó en la construcción del Product-Backlog, el cual se presenta a continuación en la Tabla 3.5.

#### Product-Backlog

ID	Historia de Usuario	Prioridad	Módulo	Estimación	SPRING
1	Control de acceso para el administrador	Alta	Módulo de autenticación de usuarios	3	1
2	Registrar convenios o solicitudes de pasantes nuevos.	Alta	Módulo de registro de pasantías y convenios	5	1
3	Administración de usuarios	Alta	Módulo de administración de usuarios	10	1

4	Gestión de pasantías y convenios	Alta	Módulo de administración de pasantías y convenios	10	<b>2</b>
5	Clasificación de pasantías para carreras	Alta	Módulo de administración de categorías	5	<b>2</b>
6	Registro y seguimiento de solicitudes de pasantía de los estudiantes	Alta	Módulo de registro de solicitud de pasantía	5	<b>3</b>
7	Registro de datos de solicitudes de estudiantes	Alta	Módulo de registro de solicitud de estudiantes	10	<b>3</b>
8	Registro de solicitudes aprobadas y concluidas	Alta	Módulo de solicitudes aprobadas y concluidas	8	<b>3</b>

Tabla 3.5.

**Fuente:** Elaboración  
propia

### **3.14.1. SPRINT 1**

- Realizar el Análisis y Diseño del proyecto.
- Realizar el Modelamiento de la Base de datos.
- Diagramas
- Historia de Usuario.
- Acceso al Sistema.
- Registrar Usuario.
- Registrar pasantía.

### **3.14.2. SPRINT 2**

- Historia de Usuario.
- Registrar Carrera.
- Registrar Convenio.
- Registrar Docente.
- Registrar Administrativo.
- Clasificación de Carreras

### **3.14.3. SPRINT 3**

- Historia de Usuario:
- Registro de Estudiante
- Registro de solicitudes
- Seguimiento de solicitud de pasantía

HISTORIA DE USUARIO	
Número: 001	Nombre: Acceso al Sistema
Autor: Administrador y usuario	
Modificación de historia número:	Iteración asignada
Prioridad del negocio:	
<b>Descripción:</b>	
<ol style="list-style-type: none"> <li>1. El usuario ingresa las credenciales de acceso al sistema (Clave y Contraseña).</li> <li>2. El Sistema recibe los parámetros de accesos y valida las credenciales de acceso con los datos registrados en la base de datos y que el usuario se encuentre activo para acceder.</li> <li>3. Validados los datos, el sistema otorga el acceso mostrando el menú de opciones.</li> </ol>	
<b>Alternativo:</b>	
<ol style="list-style-type: none"> <li>1. En caso que el usuario no pueda ingresar a la aplicación, el sistema mostrará un mensaje al usuario indicando:           <ol style="list-style-type: none"> <li>A. Que el usuario está vencido o desactivado.</li> <li>B. Que las credenciales son incorrectas.</li> <li>C. Que no tiene acceso a la red.</li> </ol> </li> <li>2. El sistema mostrará al usuario el perfil del menú de opciones, de acuerdo al nivel de acceso registrado en el sistema.</li> </ol>	
<b>Observación:</b>	
En caso de no ingresar el usuario al sistema por cualquier opción no registrado en las opciones, deberá informar al usuario.	

## PROTOTIPO

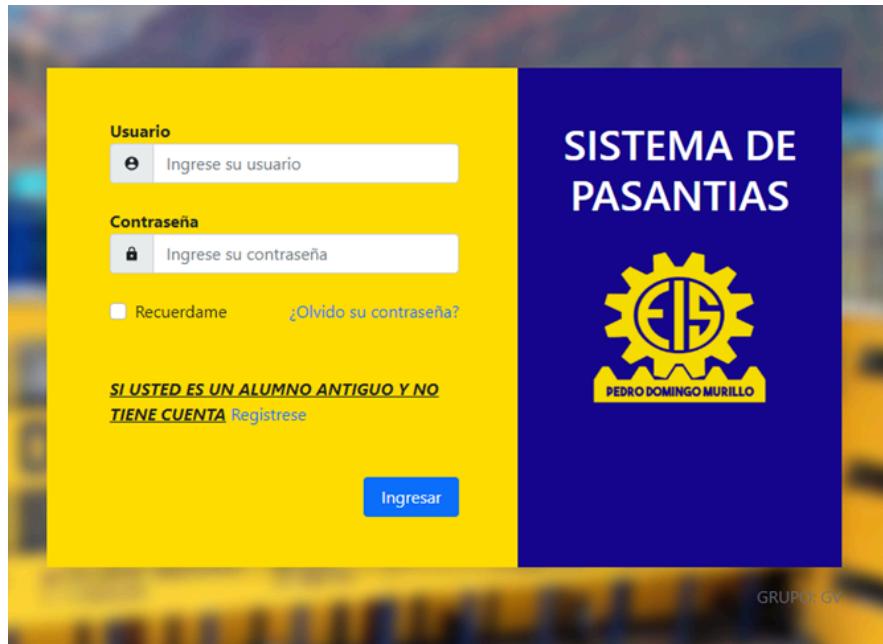


Tabla 3.6.

**Fuente:** Elaboración  
propia

HISTORIA DE USUARIO	
Número:012	Nombre: Registrar Usuario
Autor: Administrador y usuario	
Modificación de historia número:	Iteración asignada
Prioridad del negocio:	

Descripción:

1. El usuario del Sistema ingresa al módulo de "Lista de Opciones"
2. El contenedor de módulos registrados muestra en la parte derecha
3. Para eliminar un registro de usuarios registrados, el usuario del sistema deberá seleccionar una o varias casillas de eliminación ubicadas en el lado derecho; debiendo hacer clic en el ícono "Eliminar". El sistema cambiará el valor del Activo de 1 a 0, eliminando de forma lógica el registro, la aplicación mostrará un mensaje al usuario indicando la eliminación de forma correcta.
4. El sistema hará una nueva carga de datos con los registros de estado activo con valor 1.

Alternativo:

1. En caso que el usuario no pueda ingresar a la aplicación, el sistema mostrará un mensaje al usuario indicando:
  - a. Que el usuario está vencido o desactivado.
  - b. Que las credenciales son incorrectas.
  - c. Que no tiene acceso a la red.
2. El sistema mostrará al usuario el perfil del menú de opciones, de acuerdo al nivel de acceso registrado en el sistema.

Observación:

En caso de no ingresar el usuario al sistema por cualquier opción no registrado en las opciones, deberá informar al usuario.

## PROTOTIPO

## HISTORIA DE USUARIO

Número:002	Nombre: <input type="text"/> Registrar <input type="text"/> PASANTÍA -CONVENIO
Autor: Administrador	
Modificación de historia número:	<input type="text"/> Iteración asignada
Prioridad del negocio:	<input type="text"/>

**Descripción:**

1. El administrador del Sistema ingresa al módulo de CONVENIOS mediante las Opciones de Menú.
2. El sistema muestra la interfaz “CONVENIOS”, que contiene las funciones de Buscar, Nuevo, inactivos, Eliminar, ver y Editar.
3. El contenedor de CONVENIOS registrados muestra en la parte inferior la numeración de las páginas del contenedor (1, 2, 3,...).
4. Para eliminar un registro de CONVENIOS Registrados, el usuario del sistema deberá seleccionar una o varias casillas de eliminación ubicadas en el lado derecho; debiendo hacer clic en el ícono “Eliminar”. El sistema cambiará el valor del Activo de 1 a 0, eliminando de forma lógica el registro, la aplicación mostrará un mensaje al usuario indicando la eliminación de forma correcta.
5. El sistema hará una nueva carga de los Convenios Registrados con los registros de estado Activo con valor 1.
6. Para imprimir, el usuario hará un clic en el ícono del lado derecho donde dice ver

**A. REGISTRAR CONVENIO**

1. El administrador del Sistema al hacer clic en el Botón “Nuevo” de la interfaz “CONVENIOS”, la aplicación mostrará la interfaz “NUEVO CONVENIO”.
2. El administrador seleccionará a la empresa a la que pertenece , Tipo de convenio, fecha de inicio, fecha final, y también adjuntará imagen de la carta de solicitud en pdf no debe exceder de los 4 MB.
3. Para realizar el registro del CONVENIO, el Administrador hará clic en el botón “Grabar” de la interfaz “Nuevo convenio”. El sistema mostrará un mensaje de registro satisfactorio o en caso contrario registro no realizado.

**B. EDITAR CONVENIO**

1. El Administrador digita el nombre o parte del nombre del CONVENIO en el cuadro de Texto, y hará clic en el botón "Buscar", el contenedor mostrará en la parte inferior las coincidencias de la búsqueda.
2. El Administrador para modificar los datos, deberá hacer clic en el ícono de "Editar" que se encuentra en el lado derecho del contenedor de datos de la interfaz "CONVENIO"
3. El sistema mostrará la interfaz "Actualizar Convenio" con todos los datos del convenio seleccionado que se encuentra registrado en la base de datos.
4. El Administrador modificará la información, que le permita la aplicación.
5. Con los cambios realizados en la interfaz "Actualizar Convenio", el usuario hará clic en el botón "Actualizar", el sistema mostrará el mensaje de confirmación de cambios correctos o cambios incorrectos.

**Observación:**

1. Antes de grabar al nuevo convenio, el sistema validará que la información esté completa y correcta.
2. En caso que el contenedor de datos presente menos de 10 registros, el sistema automáticamente no mostrará la numeración de paginación.

## Prototipo

Código	Empresa	Tipo de Convenio	Estado	Fecha Inicio	Fecha Final	Acciones
CONV20220001	OTIS	INTERINSTITUCIONAL	Vigente	2022-10-12	2022-10-27	
CONV20220002	ENTEL	EMPRESARIAL	Vigente	2022-10-18	2022-11-04	
CONV20220004	OTIS	INTERINSTITUCIONAL	Vigente	2022-10-19	2022-10-21	
CONV20230002	ESCUELA SUPERIOR DE FORMACION	INTERINSTITUCIONAL	Vigente	2023-05-10	2023-12-08	

**Nuevo Convenio**

**Empresa**  
--SELECCIONE LA EMPRESA--

**Tipo de Convenio**  
--SELECCIONE EL TIPO DE CONVENIO--

**Fecha de Inicio**  
dd/mm/aaaa

**Fecha Final**  
dd/mm/aaaa

**Convenios**  
Seleccionar archivo Ninguno archivo selec.  
Peso máximo de la imagen 4MB

## **CONCLUSIONES**

Se logró implementar un sistema de seguimiento que permite un monitoreo detallado del desempeño de los estudiantes durante sus pasantías. Los estudiantes ahora pueden acceder a información actualizada sobre su progreso y recibir retroalimentación de manera más eficiente.

Se introdujeron componentes que facilitan la comunicación y coordinación entre las unidades administrativas. La colaboración entre los departamentos involucrados en la emisión de cartas de prácticas ha mejorado significativamente.

Se logró un aumento en la transparencia del proceso de obtención de cartas de prácticas para los estudiantes. Los estudiantes ahora tienen una visión más clara del estado de sus solicitudes y los trámites en curso.

Se implementó con éxito un sistema centralizado que recopila y almacena datos sobre las pasantías de los estudiantes permitiendo una gestión más eficaz de la información relacionada con las pasantías y una evaluación más precisa de los programas.

Se introdujo un sistema de verificación de documentos que aumenta la efectividad y confiabilidad en el proceso de selección y asignación de pasantías logrando reducir las preocupaciones sobre la autenticidad de los documentos presentados por los estudiantes.

## **RECOMENDACIONES**

Dado que la aprobación y validación del sistema están sujetas a procedimientos y criterios preestablecidos, se debe trabajar de cerca con los usuarios para asegurar que el sistema cumpla con sus expectativas y requisitos desde el principio. La retroalimentación temprana es clave.

Aunque no se considera una expansión masiva, es aconsejable diseñar componentes del sistema de manera que puedan escalarse para acomodar un aumento razonable en la cantidad de estudiantes y pasantías en el futuro, de modo que no sea necesario rediseñar el sistema por completo.

## BIBLIOGRAFÍA

- ARCE, A. (2018). Programación PHP (Primera Ed. ed.). Versión.
- ACENS. (2020). Framework para el desarrollo ágil de aplicaciones. Acens Technologies.
- ATEHORTÚA, F. A., BUSTAMANTE, R., & VALENCIA, J. (2008). Sistema de Gestión Integral. Colombia: Ed. Universidad de Antioquia.
- ATT. (2019). Resolución Administrativa Interna ATT-DJ-RAI LP 11/2019. Reglamento de Pasantías y Trabajo Dirigido RE-009, 31.
- BRAVO, L. (2015). Sistema informático de documentos y archivo en escalafón administrativo. La Paz: Ingeniería de sistemas informáticos, Universidad Mayor de San Andrés.
- B., G. (2022, 2 marzo). ¿Qué es AJAX y cómo funciona? Tutoriales Hostinger. Recuperado 10 de octubre de 2022, de <https://www.hostinger.es/tutoriales/que-es-ajax>
- Estrada Web Group. (s. f.). Mensajes de notificación profesionales al usuario con jQuery y SweetAlert. Recuperado 10 de octubre de 2022, de <https://estradawebgroup.com/Post/Mensajes-de-notificacion-profesionales-al-usuario-con-jQuery-y-Sweetalert/4252>
- Hernández, J. (2021, 19 noviembre). Linux. Concepto de - Definición de. Recuperado 10 de octubre de 2022, de <https://conceptodefinicion.de/linux/>
- HostingPlus. (2020, 14 diciembre). Qué es MariaDB y cuáles son sus características | Blog | Hosting Plus Perú. Hosting Plus. Recuperado 10 de octubre de 2022, de <https://www.hostingplus.pe/blog/que-es-mariadb-y-cuales-son-sus-caracteristicas/>
- Modelo vista controlador (MVC). Servicio de Informática ASP.NET MVC 3 Framework. (s. f.). Recuperado 10 de octubre de 2022, de <https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html>

- Ortega, C. (2021, 19 julio). Método analítico: Qué es, para qué sirve y cómo realizarlo. QuestionPro. Recuperado 10 de octubre de 2022, de <https://www.questionpro.com/blog/es/metodo-analitico/>
- ¿Qué son los Web Services? (2008, 18 agosto). Tecnologías de la Información y Procesos de Negocios. Recuperado 10 de octubre de 2022, de <https://msaffirio.com/2006/02/05/%C2%BFque-son-los-web-services/>
- Robledano, A. (2021, 3 diciembre). Qué es MySQL: Características y ventajas. OpenWebinars.net. Recuperado 10 de octubre de 2022, de <https://openwebinars.net/blog/que-es-mysql/>