

**UNIVERSIDAD MAYOR DE SAN ANDRÉS**  
**FACULTAD DE CIENCIAS PURAS Y NATURALES**  
**CARRERA DE INFORMÁTICA**



**PROYECTO DE GRADO**

**SISTEMA WEB PARA EL CONTROL Y SEGUIMIENTO DE  
VENTAS DE PRODUCTOS ARTESANALES  
CASO: BOLIVIA TECH HUB**

**Proyecto de Grado para obtener el Título de Licenciatura en Informática  
Mención Ingeniería de Sistemas Informáticos**

**POSTULANTE: KARLA BELEN LIMACHI MACHACA**

**TUTOR METODOLÓGICO: LIC. FREDDY MIGUEL TOLEDO PAZ**

**ASESOR: PH. D. YOHONI CUENCA SARZURI**

**LA PAZ – BOLIVIA**

**2018**



**UNIVERSIDAD MAYOR DE SAN ANDRÉS  
FACULTAD DE CIENCIAS PURAS Y NATURALES  
CARRERA DE INFORMÁTICA**



**LA CARRERA DE INFORMÁTICA DE LA FACULTAD DE CIENCIAS PURAS Y NATURALES PERTENECIENTE A LA UNIVERSIDAD MAYOR DE SAN ANDRÉS AUTORIZA EL USO DE LA INFORMACIÓN CONTENIDA EN ESTE DOCUMENTO SI LOS PROPÓSITOS SON ESTRICTAMENTE ACADÉMICOS.**

**LICENCIA DE USO**

El usuario está autorizado a:

- a) visualizar el documento mediante el uso de un ordenador o dispositivo móvil.
- b) copiar, almacenar o imprimir si ha de ser de uso exclusivamente personal y privado.
- c) copiar textualmente parte(s) de su contenido mencionando la fuente y/o haciendo la referencia correspondiente respetando normas de redacción e investigación.

El usuario no puede publicar, distribuir o realizar emisión o exhibición alguna de este material, sin la autorización correspondiente.

**TODOS LOS DERECHOS RESERVADOS. EL USO NO AUTORIZADO DE LOS CONTENIDOS PUBLICADOS EN ESTE SITIO DERIVARA EN EL INICIO DE ACCIONES LEGALES CONTEMPLADOS EN LA LEY DE DERECHOS DE AUTOR.**

## **DEDICATORIA**

*El presente proyecto de grado va dedicado en primer lugar a Dios, por haberme permitido llegar hasta este punto, haberme dado salud y por iluminar mi mente en cada paso que doy, A mis padres Alicia y Ancelmo por ser el pilar fundamental en todo lo que soy, en mi educación, tanto académica, como de la vida, por su incondicional apoyo, a mis hermanos Leonardo, Rocio y Arturo por sus valiosas sugerencias; si no fuera por ellos, no hubiera llegado a ser la persona que ahora soy, ni todo lo alcanzado en este tiempo, agradezco por su ayuda y apoyo desprendido.*

## **AGRADECIMIENTOS**

Primero y antes que nada quisiera agradecer a Dios por la oportunidad que me brindó para realizar este proyecto y aprender de él.

A mis padres ya que me brindaron apoyo incondicional en mis estudios y a lo largo de mi vida.

A mi tutor Lic. Freddy Miguel Toledo Paz, por toda la sabiduría y experiencia profesional, por sus consejos, colaboración y paciencia a lo largo del desarrollo de éste Proyecto, que es un gran pasó en la vida de un estudiante Universitario.

A mi asesor Ph. D. Yohoni Cuenca Sarzuri por el conocimiento que me otorgó en las materias que pase con él, por el apoyo, asesoramiento y guía para la realización de este trabajo.

## RESUMEN

El presente Proyecto de Grado consiste en implementar un SISTEMA WEB DE CONTROL Y SEGUIMIENTO DE VENTAS DE PRODUCTOS ARTESANALES CASO: BOLIVIA TECH HUB, surge del gran inconveniente que sufren los artesanos, que no disponen de oportunidades para publicar y promover su producción al mercado.

El proyecto se centró en la venta de productos artesanales, desde el registro de productos, hasta reportes de ventas, administración de ventas, creación de carrito de compras, utilización de la pasarela de pago de Paypal y autenticación para controlar acceso al sistema.

En la parte introductoria se muestra los antecedentes y actividades que realiza la institución, también se muestra el análisis de los problemas y los objetivos propuestos.

Para el desarrollo del proyecto se aplicó la metodología de desarrollo ágil SCRUM apoyándose en la metodología KANBAN y la metodología de desarrollo UWE para el modelado del diseño.

El Sistema es un producto de calidad de acuerdo a la métrica de calidad **Web-Site QEM**.

Para el funcionamiento pleno del proyecto también se toma como tarea final implementar la seguridad, además de poner en funcionamiento en el servidor designado.

Finalmente se puede concluir que los objetivos planteados fueron alcanzados y que el Sistema cumple con los requerimientos establecidos por el cliente.

**Palabras clave:** Metodología ágil Kanban, UWE, **Web-Site QEM**, Sistema Web, COCOMO II.

## ABSTRACT

The present Project of Degree consists of implementing a WEB SYSTEM OF CONTROL AND MONITORING OF SALES OF ARTISAN PRODUCTS CASE: BOLIVIA TECH HUB, arises from the great inconvenience suffered by the artisans, which does not offer opportunities to publish and promote their production to the market. The project focused on the sale of artisan products, from the registration of products, to sales reports, sales management, the creation of shopping carts, the use of the Paypal payment account and authentication to control access to the system. In the introductory part, the antecedents and the activities carried out by the institution are shown, as well as the analysis of the proposed problems. For the development of the project, the SCRUM development methodology was applied based on the KANBAN methodology and the UWE development methodology for design modeling.

The System is a quality product according to the quality metric QEM Website. For the complete functioning of the project, it is also taken as a final task to implement the security, in addition to operating on the designated server. Finally, it can be concluded that the proposed objectives were achieved and that the system meets the requirements established by the client.

**Keywords:** Agile Kanban Methodology, UWE, Web-Site QEM, Web System, COCOMO II.

## ÍNDICE

<b>CAPÍTULO I .....</b>	<b>1</b>
<b>MARCO INTRODUCTORIO.....</b>	<b>1</b>
1.1. INTRODUCCIÓN .....	2
1.2. ANTECEDENTES .....	3
1.2.1. ANTECEDENTES INSTITUCIONALES.....	3
1.2.2. PROYECTOS SIMILARES.....	3
1.3. PLANTEAMIENTO DEL PROBLEMA .....	6
1.3.1. PROBLEMA PRINCIPAL.....	6
1.3.2. PROBLEMAS SECUNDARIOS .....	6
1.4. DEFINICIÓN DE OBJETIVOS .....	7
1.4.1. OBJETIVO GENERAL .....	7
1.4.2. OBJETIVOS ESPECÍFICOS .....	7
1.5. JUSTIFICACIÓN .....	7
1.5.1. JUSTIFICACIÓN SOCIAL .....	7
1.5.2. JUSTIFICACIÓN TÉCNICA.....	7
1.5.3. JUSTIFICACIÓN ECONÓMICA.....	8
1.6. ALCANCES Y LÍMITES .....	8
1.6.1. DELIMITACIÓN TEMÁTICA .....	8
1.6.2. DELIMITACIÓN ESPACIAL.....	9
1.6.3. DELIMITACIÓN TEMPORAL .....	9
1.7. APORTES.....	9
1.7.1. PRÁCTICO .....	9
1.7.2. TEÓRICO.....	9
1.8. METODOLOGÍA .....	10
1.8.1. METODOLOGÍA DE INVESTIGACIÓN .....	10
1.8.2. METODOLOGÍA DE DESARROLLO DE SOFTWARE.....	10

<b>CAPÍTULO II.....</b>	<b>11</b>
<b>MARCO TEÓRICO.....</b>	<b>11</b>
2.1. INTRODUCCIÓN.....	12
2.2. INGENIERÍA SOFTWARE.....	12
2.3. METODOLOGÍAS DE DESARROLLO .....	13
2.3.1. METODOLOGÍAS TRADICIONALES .....	14
2.3.2. METODOLOGÍAS AGILES .....	14
2.3.3. COMPARACIÓN DE METODOLOGÍAS .....	15
2.3.4. MANIFIESTO POR EL DESARROLLO ÁGIL .....	16
2.4. SCRUM .....	17
2.4.1. CARACTERÍSTICAS.....	17
2.4.2. PRINCIPIOS BÁSICOS .....	18
2.4.3. PRÁCTICAS DE SCRUM.....	18
2.4.4. ROLES DE SCRUM .....	20
2.4.5. EVENTOS DE SCRUM.....	21
2.4.5.1. EL SPRINT .....	21
2.4.5.2. PLANIFICACIÓN DEL SPRINT .....	22
2.4.5.3. SCRUM DIARIO .....	23
2.4.5.4. REVISIÓN DEL SPRINT.....	24
2.4.5.5. RETROSPECTIVA DEL SPRINT .....	26
2.4.6. ARTEFACTOS DE LA METODOLOGÍA SCRUM .....	28
2.4.6.1. PRODUCT BACKLOG .....	28
2.4.6.2 SPRINT BACKLOG .....	29
2.4.6.3. INCREMENTO.....	30
2.4.7. FASES DE SCRUM.....	30
2.4.7.1. PREGAME .....	31
2.4.7.2. DEVELOPMENT.....	33
2.4.7.3. POSTGAME.....	33



2.4.8. REUNIONES DE TRABAJO EN UN CONTEXTO SCRUM .....	33
2.5. KANBAN .....	35
2.5.1. CARACTERÍSTICAS.....	36
2.5.2. ROLES KANBAN .....	36
2.5.3. TABLERO KANBAN.....	37
2.5.4. VISUALIZAR EL WORKFLOW (FLUJO DE TRABAJO).....	39
2.5.5. FASES DEL KANBAN .....	39
2.5.6. REUNIONES DE EQUIPO .....	41
2.6. COMBINACIÓN DE SCRUM Y KANBAN SCRUMBAN.....	41
2.6.1. DIFERENCIAS BÁSICAS ENTRE SCRUM Y KANBAN .....	41
2.6.2. BENEFICIOS DE LA INTEGRACIÓN .....	42
2.7. INGENIERÍA WEB.....	42
2.8. LENGUAJE UNIFICADO DE MODELADO (U.M.L.).....	43
2.8.1. VISTAS DEL UML.....	44
2.8.2. DEFINICIÓN DE MODELO.....	45
2.8.3. DIAGRAMAS UML .....	46
2.8.3.1. DIAGRAMAS ESTRUCTURALES.....	48
2.8.3.2. DIAGRAMAS DE COMPORTAMIENTO.....	49
2.9. SISTEMA DE INFORMACIÓN .....	50
2.10. SEGURIDAD .....	50
2.10.1. INYECCIONES SQL.....	51
2.10.2. .HTACCESS.....	51
2.10.3. CIFRADOS POR BLOQUE – BLOWFISH.....	51
<b>CAPÍTULO III .....</b>	<b>53</b>
<b>MARCO APLICATIVO .....</b>	<b>53</b>
3.1 INTRODUCCIÓN .....	54
3.2. FASE PREGAME.....	54
3.2.1. CONCEPCIÓN Y EXPLORACIÓN .....	55

3.2.2. ROLES SCRUMBAN .....	55
3.2.3. ARQUITECTURA DE SOFTWARE .....	56
3.2.4. OBTENCIÓN DE REQUERIMIENTOS.....	57
3.2.5. PRODUCT-BACKLOG.....	58
3.2.6. INICIALIZACIÓN.....	60
3.3. FASE DEVELOPMENT .....	69
3.3.1. DESARROLLO DE LOS SPRINTS .....	69
3.4. PRIMER SPRING: MÓDULO DE REGISTRO Y LOGIN DE USUARIOS.....	70
3.4.1. ETAPA DE ANÁLISIS.....	70
3.4.1.1. DIAGRAMA DE CASO DE USO.....	71
3.4.2. ETAPA DE DISEÑO .....	72
3.4.2.1. DIAGRAMA DE CLASES .....	72
3.4.2.2. DIAGRAMA DE ESTADO .....	73
3.4.3. ETAPA DE IMPLEMENTACIÓN .....	74
3.4.3.1. DIAGRAMA DE SECUENCIA .....	75
3.4.3.2. DIAGRAMA DE COMPONENTES .....	76
3.4.4. RESULTADOS .....	76
3.5. SEGUNDO SPRING: MÓDULO DE ADMINISTRACIÓN DE CUENTAS DE USUARIOS .....	78
3.5.1. ETAPA DE ANÁLISIS.....	78
3.5.1.1. DIAGRAMA DE CASOS DE USO.....	79
3.5.2. ETAPA DE DISEÑO .....	81
3.5.2.1. DIAGRAMA DE CLASES .....	81
3.5.2.2. DIAGRAMA DE ESTADOS .....	82
3.5.3. ETAPA DE IMPLEMENTACIÓN .....	82
3.5.3.1. DIAGRAMA DE SECUENCIA .....	83
3.5.3.2. DIAGRAMA DE COMPONENTES .....	84
3.5.4. RESULTADOS .....	84
3.6. TERCER SPRING: MÓDULO DE CARRITO DE COMPRAS .....	85

3.6.1. ETAPA DE ANÁLISIS.....	85
3.6.1.1. DIAGRAMA DE CASOS DE USO.....	86
3.6.2. ETAPA DE DISEÑO .....	87
3.6.2.1. DIAGRAMA DE CLASES .....	87
3.6.2.2. DIAGRAMA DE ESTADO .....	88
3.6.3. ETAPA DE IMPLEMENTACIÓN .....	89
3.6.3.1. DIAGRAMA DE SECUENCIA .....	89
3.6.3.2. DIAGRAMA DE COMPONENTES .....	91
3.6.4. RESULTADOS .....	91
3.7. CUARTO SPRING: MÓDULO DE ADMINISTRACIÓN DE CATÁLOGO Y PRODUCTOS .....	92
3.7.1. ETAPA DE ANÁLISIS.....	92
3.7.1.1. DIAGRAMA DE CASO DE USO .....	93
3.7.2. ETAPA DE DISEÑO .....	94
3.7.2.1. DIAGRAMA DE CLASES .....	94
3.7.2.2. DIAGRAMA DE ESTADO .....	95
3.7.3. ETAPA DE IMPLEMENTACIÓN .....	96
3.7.3.2. DIAGRAMA DE COMPONENTES .....	97
3.7.4. RESULTADOS .....	97
3.8. POSTGAME Y FASE DE ESTABILIZACIÓN, PRUEBAS Y REPARACIONES .....	99
3.4.1. PRUEBAS Y REPARACIONES .....	99
3.4.2. PRUEBAS DE CAJA NEGRA .....	100
<b>CAPITULO IV.....</b>	<b>101</b>
<b>CALIDAD Y SEGURIDAD.....</b>	<b>101</b>
4.1. INTRODUCCIÓN.....	102
4.2. CALIDAD DE SOFTWARE.....	102
4.2.1. DEFINIENDO METAS DE EVALUACIÓN .....	102
4.2.2. ESPECIFICANDO REQUERIMIENTOS DE CALIDAD.....	102

4.2.3 ESPECIFICACIÓN DE CARACTERÍSTICAS DE CALIDAD.....	103
4.2.3.1. USABILIDAD.....	103
4.2.3.2. FUNCIONALIDAD .....	109
4.2.3.3. CONFIABILIDAD .....	111
4.2.3.4. EFICIENCIA.....	112
4.3. SEGURIDAD .....	113
4.3.1. TIPOS DE SEGURIDAD.....	114
4.3.1.1. SEGURIDAD EN EL CLIENTE .....	114
4.3.1.2. SEGURIDAD EN EL SERVIDOR.....	114
4.3.1.3. SEGURIDAD EN LA COMUNICACIÓN.....	114
<b>CAPITULO V .....</b>	<b>116</b>
<b>ANÁLISIS DE COSTO BENEFICIO .....</b>	<b>116</b>
5.1. INTRODUCCIÓN .....	117
5.2. COCOMO II .....	117
5.2.1. MÉTRICAS DE SOFTWARE.....	117
5.2.2. ESTIMACIÓN DEL ESFUERZO .....	121
5.3. COSTO DE SOFTWARE.....	122
5.3.1. COSTO DE ELABORACIÓN DEL PROYECTO .....	123
5.3.2. COSTE DE IMPLEMENTACIÓN DEL PROYECTO .....	123
5.3.3. COSTE TOTAL .....	123
5.4. CÁLCULO BENEFICIO VAN Y TIR .....	124
5.4.1. VALOR ACTUAL NETO (VAN) .....	124
5.4.2. TASA INTERNA DE RETORNO (TIR).....	125
5.5. COSTO BENEFICIO.....	126
<b>CAPÍTULO VI.....</b>	<b>127</b>
<b>CONCLUSIONES Y RECOMENDACIONES .....</b>	<b>127</b>
6.1. CONCLUSIONES .....	128

6.2. RECOMENDACIONES.....	128
<b>BIBLIOGRAFÍA .....</b>	<b>130</b>
<b>ANEXOS .....</b>	<b>136</b>
ANEXO A – ÁRBOL DE PROBLEMAS .....	137
ANEXO B – ÁRBOL DE OBJETIVOS .....	138
ANEXO C- MARCO LÓGICO .....	139

## ÍNDICE DE TABLAS

Tabla 2. 1 Comparación de metodologías .....	15
Tabla 2. 2 Practicas de la gestión ágil .....	19
Tabla 2. 3 Roles Scrum .....	20
Tabla 2. 4 Elementos de revisión de un sprint.....	25
Tabla 2. 5 Subfases de Pregame .....	31
Tabla 2. 6 Características de un reunión de trabajo en un contexto SCRUM .....	34
Tabla 2. 7 Características de Kanban .....	36
Tabla 2. 8 Elementos de un tablero Kanban.....	38
Tabla 2. 9 Representación de vistas mediante diagramas .....	44
Tabla 2. 10 Diagramas Estructurales.....	48
Tabla 2. 11 Diagramas de Comportamiento.....	49
Tabla 3. 1. Tabla Scrumban.....	54
Tabla 3. 2 Tareas realizadas para la obtención de requisitos.....	55
Tabla 3. 3 Fases de trabajo, limite WIP y responsables .....	55
Tabla 3. 4 Lista de requerimientos .....	57
Tabla 3. 5 Product-Backlog.....	58
Tabla 3. 6 Diccionario de datos de la tabla de administrador.....	62
Tabla 3. 7 Diccionario de datos de la tabla de categorías.....	62
Tabla 3. 8 Diccionario de datos tabla de comercio.....	63
Tabla 3. 9 Diccionario de datos de la tabla de compra.....	64
Tabla 3. 10 Diccionario de datos de la tabla de pedido .....	65
Tabla 3. 11 Diccionario de datos de la tabla de productos .....	66
Tabla 3. 12 Diccionario de datos de la tabla de publicidad.....	67
Tabla 3. 13 Diccionario de datos de la tabla de usuarios .....	68
Tabla 3. 14 Configuración del sistema .....	69
Tabla 3. 15 Primer Sprint: módulo de registro y login de usuarios.....	70

Tabla 3. 16 Especificación del caso de uso de inicio de sesión y registro .....	71
Tabla 3. 17 Segundo sprint módulo de administración de usuario.....	78
Tabla 3. 18 Especificación del caso de uso de administración de usuarios.....	80
Tabla 3. 19 Tercer sprint módulo de carrito de compras.....	85
Tabla 3. 20 Cuarto spring módulo de administración de catálogo y productos .....	92
Tabla 3. 21 Errores encontrados en la plataforma .....	99
Tabla 4. 1 Comprensibilidad global del sitio.....	104
Tabla 4. 2 Característica: Mecanismo de Ayuda y Retroalimentación en Línea .....	104
Tabla 4. 3 Aspectos de Interfaces y Estéticos .....	106
Tabla 4. 4 Website QEM Evaluación de misceláneas .....	107
Tabla 4. 5 Total de Usabilidad.....	108
Tabla 4. 6 Aspectos de Búsqueda y Recuperación.....	109
Tabla 4. 7 Aspectos de Dominio Orientados al Usuario .....	110
Tabla 4. 8 Total Funcionalidad.....	111
Tabla 4. 9 Evaluación de confiabilidad .....	112
Tabla 4. 10 Evaluación de desempeño .....	113
Tabla 5. 1 Calculo del punto función no ajustado .....	118
Tabla 5. 2 Escala de niveles de influencia.....	119
Tabla 5. 3 Suma de Factor de complejidad .....	119
Tabla 5. 4 Factor LDC/PF .....	120
Tabla 5. 5 Modelo básico para tipos de proyecto.....	122
Tabla 5. 6 Costo de elaboración del software.....	123
Tabla 5. 7 Costo total del software .....	123
Tabla 5. 8 Calculo del VAN .....	124

## ÍNDICE DE FIGURAS

Figura 2. 1. Framework tradicional de Scrum .....	27
Figura 2.2 Proceso scrum .....	31
Figura 2. 3 Tablero Kanban .....	37
Figura 2. 4 Panel Kanban .....	40
Figura 2. 5 Historia de UML .....	44
Figura 2. 6 Diagramas empleados por UML .....	47
Figura 2. 7 Vistas de un software y sus respectivos diagramas UML.....	47
Figura 3. 1 Esquema de arquitectura .....	56
Figura 3. 2 Base de datos relacional .....	62
Figura 3. 3 Diagrama de caso de uso de inicio de sesión y registro .....	71
Figura 3. 4. Diagrama de clases de inicio de sesión y registro.....	73
Figura 3. 5 Diagrama de estados de inicio de sesión y registro .....	74
Figura 3. 6 Diagrama de secuencia de inicio de sesión y registro.....	75
Figura 3. 7. Diagrama de componentes de inicio de sesión y registro .....	76
Figura 3. 8. Entorno gráfico de inicio de sesión.....	77
Figura 3. 9 Entorno gráfico de solicitud de nueva contraseña.....	77
Figura 3. 10 Entorno gráfico de registro.....	78
Figura 3. 11. Diagrama de casos de uso del módulo de administración de usuarios .....	80
Figura 3. 12 diagrama de clases del módulo de administración de usuarios.....	81
Figura 3. 13 Diagrama de estados del módulo de administración de usuarios.....	82
Figura 3. 14 Diagrama de secuencia del módulo de administración de usuarios .....	83
Figura 3. 15. Diagrama de componentes del módulo de administración de usuarios .....	84
Figura 3. 16 Plataforma de administración.....	85
Figura 3. 17 Diagrama de casos de uso del módulo de carrito de compras.....	87
Figura 3. 18 Diagrama de clases del módulo de carrito de compras .....	88
Figura 3. 19 Diagrama de estado del módulo de carrito de compras .....	89
Figura 3. 20 Diagrama de secuencia de módulo carrito de compras .....	90



Figura 3. 21 Diagrama de componentes .....	91
Figura 3. 22 Captura de pantalla del carrito de compra.....	92
Figura 3. 23 Diagrama de caso de uso del módulo de catálogo .....	94
Figura 3. 24 Diagrama de clases del módulo de catálogo .....	95
Figura 3. 25 Diagrama de estados de módulo de catálogo .....	96
Figura 3. 26 Diagrama de componentes de módulo de catálogo.....	97
Figura 3. 27 Captura de pantalla de cabecera del catálogo .....	98
Figura 3. 28 Captura de pantalla de la lista catálogo.....	98
Figura 3. 29 Captura de pantalla de la lista productos.....	99

# CAPÍTULO I

## MARCO INTRODUCTORIO



## **1.1. INTRODUCCIÓN**

En estos tiempos se puede notar el rápido crecimiento de las Tecnologías de Información y Comunicación TIC, uso de dispositivos móviles y sobre todo el uso de redes sociales, que están transformando la comunicación y el acceso a la información.

La información es el arma principal que ayudará a los productos y servicios, a la productividad, a penetrar en el ambiente competitivo del mundo moderno empresarial. Debe quedar claro que las computadoras, la tecnología informática y la información de calidad no son los fines, sino simplemente las armas competitivas que apoyan a las organizaciones para alcanzar las metas para una mayor productividad.

Además de lo mencionado, existe una presión constante por parte de la sociedad actual, que por distintos motivos necesita información cada vez más rápida y a la vez confiable. Algunas instituciones o empresas desconocen los beneficios que pueden traer las Tecnologías de la Información y Comunicación en especial en el área del comercio, las empresas que se dedican al comercio tienen la necesidad o están obligadas a informar en relación con los diferentes tipos de transacciones que realiza sea oportuna y constantemente actualizada, así como también responder los requerimientos de los usuarios, facilitándoles el trabajo al momento gestionar la información.

Las artesanías en cualquier parte del mundo son una expresión cultural, que día a día gana más adeptos, Bolivia es un país multicultural, por lo tanto, tiene muchas artesanías típicas. Cuando el turista extranjero llega a Bolivia, puede encontrar una variedad de artículos artesanales, artículos de decoración, accesorios de vestir, ropa, entre otros, confeccionados por sociedades de artesanos, compuestas generalmente por familias enteras, quienes encontraron en el desarrollo de este arte una forma de ingreso, es así que muchas familias bolivianas se sustentan día a día con el trabajo artesanal, por otra parte se contempló un gran inconveniente que sufren los artesanos, no disponen de oportunidades para publicar y promover su producción al mercado.

Bolivia Tech Hub es un espacio de desarrollo colaborativo de proyectos tecnológicos relacionados a TIC's. Actualmente uno de los proyectos de emprendimiento es brindar oportunidades al sector de artesanos de difundir y lanzar sus productos al mercado.

En tal sentido, se presenta el desarrollo de una **Sistema web para el control y seguimiento de ventas caso: Bolivia Tech Hub**.

## **1.2. ANTECEDENTES**

### **1.2.1. ANTECEDENTES INSTITUCIONALES**

Bolivia Tech Hub es una entidad dedicada al desarrollo colaborativo de proyectos tecnológicos relacionados a TIC. Está ubicada en la Zona Sopacachi Pasaje Fabiani No. 2687 esq. Sanchez Lima.

Inició actividades a mediados de noviembre del 2014, brinda oportunidades de aprendizaje, desarrollo, ofertas laborales en nuevas áreas de estudio en base a retos, competencias nacionales e internacionales denominados Hackathones<sup>1</sup> (Bolvia Tech Hub, 2015).

Inicialmente contaba con un administrador del lugar, un manejador de proyectos y tres secretarias. Actualmente el equipo de trabajo aumentó a un *product manager*, cinco programadores, un diseñador, tres pasantes, una publicista y una secretaria.

### **1.2.2. PROYECTOS SIMILARES**

A continuación se mencionan algunos proyectos que se relacionan de alguna manera con el trabajo actual de investigación, extraídas de la biblioteca de la Carrera de Informática de la Facultad de ciencias Puras y Naturales de la Universidad Mayo de San Andrés (FCPN-UMSA) y proyectos similares nacionales e internacionales.

---

<sup>1</sup> Hackathon es un evento organizado por hackers, para hackers, con el fin de programar o construir una solución de forma colaborativa, durante un plazo determinado de horas, de preferencia en el mismo espacio físico.

**Título:** Sistema de información para el control y seguimiento de ventas on-line de productos farmacéuticos distribuidora PHARMICA

**Autor:** Michael Flores Gonzales

**Año:** 2016

**Resumen:** Ofrece información sobre pedidos, ventas y el seguimiento de estos y además que la información acerca del comportamiento de la empresa PHARMICA esté almacenada en una base de datos. Para la implementación del sistema se utilizaron los lenguajes de Programación: Php, Html, JavaScript, Css, los frameworks: CodeIgniter, JQuery, una base de datos creada en Mysql. La metodología utilizada es Proceso Unificado de Racional RUP para el análisis y diseño en sus diferentes fases, el modelado del sistema con el Lenguaje de Modelado Unificado UML (Flores Gonzales, 2016).

**Título:** Sistema web de control de compras, ventas e inventarios para Comercial Ariana

**Autor:** Eymi Escarlet Carrillo Cruz

**Año:** 2017

**Resumen:** Se llegó a desarrollar el sistema web de compras ventas e inventarios, como herramienta de desarrollo de aplicaciones web se utilizó el framework Laravel en su versión 5.4 complementado con el gestor de base de datos MySQL. Para el diseño responsivo (adaptable a dispositivos móviles) se utilizó el framework Bootstrap, acompañado de Javascript, JQuery y PHP (Carrillo Cruz, 2017).

**Título:** Sistema web de control de compras, ventas e inventarios y verificación de temperatura de medicamentos usando RFID y alarmas tempranas caso: “farmacias la casa de salud”

**Autor:** Vladimir Quelca Quispe

**Año:** 2016

**Resumen:** El desarrollo del proyecto se basó en las fases propuestas por la Metodología de Desarrollo Agil XP (Extreme Programming – Programación Extrema) y se complementó la fase de diseño con la ayuda de IFML (Lenguaje de Modelado de Flujos de Interacción) el

cual está basado en WebML (Lenguaje de Modelado Web), los cuales fueron muy útiles al momento de diseñar las funciones y la interfaz del usuario (Quelca Quispe, 2016).

**Título:** Sistema web para el proceso de ventas en la empresa RYSOFT

**Autor:** Robinson Manuel Yañez Romero

**Año:** 2017

**Resumen:** Abarca el análisis, diseño e implementación de un sistema web para el proceso de ventas en la empresa Rysoft. El tipo de investigación es Aplicada – experimental. Se utilizó la metodología RUP; se utilizó el lenguaje de programación PHP, para la maquetación se utilizó el Framework Bootstrap y para la base de datos se empleó MYSQL (Yañez Romero, 2017).

**Título:** Desarrollo e implementación de un sistema de gestión de ventas de repuestos automotrices en el almacén de auto repuestos eléctricos marcos en la parroquia POSORJA cantón GUAYAQUIL, provincia GUAYAS

**Autor:** Arana Quijije Julia Valeria

**Año:** 2014

**Resumen:** La metodología utilizada en el diseño del sistema se desarrolló en la estructura de red de cliente servidor, utilizando formularios HTML y lenguaje de programación PHP y Apache. Para el desarrollo ágil del sistema se empleó la metodología de prototipado, con los métodos inductivo-deductivo. Concluida la implementación del sistema, las pruebas de tiempo evidenciaron la optimización de los procesos de ventas y servicios técnicos en un 76% (Arana Quijije, 2014).

**Título:** Desarrollo de un sistema de gestión para la venta de pasajes de la empresa Flor Móvil SAC

**Autor:** Jhubel Favio Vásquez Rudas

**Año:** 2014

**Resumen:** Para diseñar y crear este sistema se está utilizando lenguaje de etiquetas HTML, el lenguaje de programación Java y un sistema gestor de datos MySQL para generar contenidos dinámicos. Además, se utilizarán diferentes herramientas que ayuden a cumplir con los requerimientos especificados en el diseño (Vasquez Rudas, 2014).

### **1.3. PLANTEAMIENTO DEL PROBLEMA**

La artesanía pasa por momentos de crisis que amenazan con cerrar talleres que funcionan desde hace muchos años. El artesano, sobre todo aquel que explota en solitario o con su familia en un pequeño taller, se encuentran con dificultades a la hora de comercializar su producto fuera de su región, e incluso fuera de su propio lugar de residencia.

El primer problema que aflige a la artesanía es de índole comercial. Los artesanos no disponen de oportunidades para difundir y lanzar su producción al mercado local, departamental, y mucho menos al internacional.

#### **1.3.1. PROBLEMA PRINCIPAL**

¿Cómo incrementar ventas de productos artesanales para Bolivia Tech Hub?

#### **1.3.2. PROBLEMAS SECUNDARIOS**

- Los productos artesanales no disponen de algún medio de difusión, lo que ocasiona disminución de ventas.
- No existe información visual de precios y descripción de productos artesanales, lo que ocasiona desconfianza al adquirir un producto.
- Turistas tanto del exterior como interior del país desconocen tiendas de productos artesanales, lo que ocasiona la no adquisición de dicho producto.
- Los productos solo se comercializan en el lugar que se está visitando, lo que ocasiona ventas limitadas.
- Adquisición de productos de calidad dudosa, lo que ocasiona disgusto por parte de clientes.
- Inexistencia de reportes de ventas con descripción detallada, lo que ocasiona un control inadecuado del stock de cada producto.

## **1.4. DEFINICIÓN DE OBJETIVOS**

### **1.4.1. OBJETIVO GENERAL**

Desarrollar un sistema web para el control y seguimiento de ventas de productos artesanales para Bolivia Tech Hub.

### **1.4.2. OBJETIVOS ESPECÍFICOS**

- Implementar un medio de difusión de productos artesanales, para la incrementar de ventas.
- Brindar información visual de precios y descripción de productos artesanales.
- Ofrecer información precisa de tiendas de productos artesanales.
- Expandir el mercado de productos artesanales.
- Proporcionar información de los talleres artesanales.
- Generar reportes de ventas con descripción detallada.

## **1.5. JUSTIFICACIÓN**

### **1.5.1. JUSTIFICACIÓN SOCIAL**

A través del presente proyecto se busca mejorar el alcance de la diversidad cultural de productos artesanales en nuestro país.

El contar con un sistema web para control y seguimiento de ventas permitirá que las personas sean informadas de la diversidad cultural de productos artesanales, además tengan la oportunidad de acceder a tiendas y productos en lugares remotos con un coste menor al que supone abrir tiendas físicas en cada ciudad.

### **1.5.2. JUSTIFICACIÓN TÉCNICA**

Para la implementación de este prototipo se tiene a disposición tecnologías y herramientas de diseño, desarrollo e implementación como frameworks e ingeniería web que nos permite la aplicación de metodologías sistemáticas, disciplinadas y cuantificables para un desarrollo eficiente.



### 1.5.3. JUSTIFICACIÓN ECONÓMICA

La institución cuenta con: servidor PHP y gestor de base de datos MySQL por el cual paga 140 Bs al mes, un servidor en la nube Heroku<sup>2</sup> que es de forma gratuita, internet por el cual paga 600 Bs al mes.

Por lo cual se tendría un gasto de 740 Bs al mes, pero como ya se pagaban esos gastos antes del análisis e implementación del sistema, se emplearán herramientas de software libre y código abierto, por ende, el proyecto tendrá 0 Bs de inversión.

### 1.6. ALCANCES Y LÍMITES

#### 1.6.1. DELIMITACIÓN TEMÁTICA

Los alcances del sistema serán los siguientes:

- **Módulo de Registro y login de usuarios**, se desarrollará una interface de registro y login.
- **Módulo de administración de cuentas de usuario**, se desarrollará una interface de administrador donde se podrá modificar, asignar un rol a los usuarios.
- **Módulo de Carrito de compra** el cual sirve para tener un detalle de los productos, tendrá la información básica de los productos, la cantidad de productos, el impuesto y el total, que posteriormente será vendido vía internet por PayPal y otros mediante tarjeta de crédito/debito, brindando la seguridad en las transacciones, eliminando la posibilidad de repudio de las operaciones por parte del comprador, además se mandara un recibo al mail para comprar que se ha cumplido con el pago.
- **Módulo de administración de catálogo, productos y proveedores**, se desarrollara una interface donde se muestre un catálogo, el detalle de cada producto: sus precios, características y demás, la información de los proveedores. Por otro lado se

---

<sup>2</sup> Heroku es una plataforma como servicio de computación en la Nube que soporta distintos lenguajes de programación

desarrollara también una interfaz de administrador donde se registre y edite los catálogos, productos y proveedores.

### **1.6.2. DELIMITACIÓN ESPACIAL**

El sistema se implementara para Bolivia Tech Hub, está ubicada en la Zona Sopacachi Pasaje Fabiani No. 2687 esq. Sanchez Lima. Alto de la Alianza, se espera que tenga un alcance Internacional.

### **1.6.3. DELIMITACIÓN TEMPORAL**

El periodo de tiempo a ser tomado será a partir de la puesta en marcha del sistema, debido a que no existen antecedentes oficiales sobre el presente trabajo de grado.

## **1.7. APORTES**

### **1.7.1. PRÁCTICO**

El proyecto ayudará a difundir y lanzar productos al mercado nacional, realizar mayor control de ventas de los insumos, de igual manera los registros de los productos donde nos servirá para optimizar el tiempo y eficiencia con respecto al manejo de la información, reducción de costes logísticos, adecuación del stock a la demanda y generación de comunidad fidelizada.

### **1.7.2. TEÓRICO**

El aporte teórico será la conjunción de las metodologías, técnicas y herramientas y además del desarrollo de acuerdo con las etapas determinadas.

Como metodología ágil se elijo Kanban que está centrada en mejorar el control de procesos que mantiene un flujo continuo y mejora el proceso de mejoras ya que el flujo es visible (Bjorkholm & Bjorkholm, 2015). La metodología Kanban será combinada con Scrum ya que ambas ofrecen una solución a medida para este proyecto.

Para el modelado del sistema se utilizará *Unified Modeling Language* UML que es un lenguaje estándar en el análisis y diseño de sistemas.

## **1.8. METODOLOGÍA**

### **1.8.1. METODOLOGÍA DE INVESTIGACIÓN**

La metodología de investigación que se utilizara para el presente proyecto de grado es el método científico, con un enfoque cuantitativo, de tipo descriptivo.

El método científico está conformado por etapas, las cuales son:

- Observación sistemática obteniendo conocimiento valido
- Hipótesis la cual es una suposición que forma parte de una posible solución
- Experimentación y emisión de conclusiones.

Un enfoque cuantitativo, ya que se seguirá un conjunto de procesos secuenciales y probatorios, además el proyecto es delimitado y concreto. Se considera lo que se ha investigado anteriormente.

### **1.8.2. METODOLOGÍA DE DESARROLLO DE SOFTWARE**

La metodología ágil que se empleara para el desarrollo es Kanban combinada con Scrum. Kanban trabaja a base de tarjetas, estas tarjetas de tareas son pegadas en la pared que muestran el estado actual de la tarea y Scrum ofrece Gestión regular de las expectativas del cliente, basada en resultados tangibles, resultados anticipados, flexibilidad y adaptación respecto a las necesidades del cliente, cambios en el mercado, mitigación sistemática de los riesgos del proyecto (Kniberg & Skarin, 2010).

Además de utilizar UML para el modelado de datos, ya que mediante UML es posible establecer la serie de requerimientos y estructuras necesarias para plasmar un sistema de software previo al proceso intensivo de escribir código. Provee especificaciones gráficas formales para un proceso de diseño completo que puede ser asistido por herramientas de diseño visuales (Teniente, Olivé, Mayol, & Gómez, 2005).

## CAPÍTULO II

### MARCO TEÓRICO



## **2.1. INTRODUCCIÓN**

En el desarrollo de este capítulo se describe la teoría sobre las metodologías, técnicas y herramientas que se utiliza para el desarrollo del Sistema web para el control y seguimiento de ventas caso: Bolivia Tech Hub, sin embargo no contempla la teoría completa, pero contiene los conceptos más importantes para que pueda aplicarse y despejar cualquier duda.

## **2.2. INGENIERÍA SOFTWARE**

Una de las primeras definiciones de ingeniería de software fue establecida por Frits Bauer (1985) quien afirma. “Ingeniería de Software es el establecimiento y uso de principios robustos de ingeniería, orientados a obtener software que sea fiable y funcione de manera eficiente sobre máquinas reales” (pág. 3).

Pressman (2010) refiere que la ingeniería de software es una disciplina que integra métodos, herramientas y procedimientos para el desarrollo de software de computadora.

El software es un componente crucial en productos basados en computadoras y la evolución de sistemas, y una de las tecnologías más importantes en todo el mundo. En los últimos años, el software pasó de ser la solución de problemas a una industria en sí misma. Ciertamente, aún hay problemas para desarrollar software de alta calidad a tiempo y dentro del presupuesto asignado.

Los sistemas y aplicación basados en web llegaron a ser sistemas sofisticados de simples conjuntos de contenido de información, estos sistemas sofisticados presentan una funcionalidad compleja y contenido en multimedios. Sobre todo, dichas webapps tienen características y requerimientos únicos, son software (Pressman, 2010).

El proceso de software es un proceso de transformación, que reúne cinco actividades estructurales: comunicación, planeación, modelado, construcción y despliegue que son aplicables a todos los proyectos de software.

Con la comunicación se busca entender los objetivos de los participantes respecto del proyecto, y reunir los requerimientos que ayuden a definir las características y funciones del software. La planeación define el trabajo de ingeniería de software al describir las tareas técnicas por realizar, los riesgos probables, los recursos que se requieren, los productos del trabajo que se obtendrán y una programación de las actividades. El modelado crea un bosquejo del objeto por hacer a fin de entender el panorama general, cómo ajustan entre sí las partes constituyentes. La construcción, esta actividad combina la generación de código y las pruebas que se requieren para descubrir errores en éste. Despliegue, el software se entrega al consumidor que lo evalúa y que le da retroalimentación, misma que se basa en dicha evaluación.

### **2.3. METODOLOGÍAS DE DESARROLLO**

Desarrollar un software es una tarea compleja, y si no se utiliza por parte del equipo de trabajo una metodología para su construcción desde la fase inicial, aumenta la probabilidad de obtener un resultado que no satisfaga las expectativas de los clientes y usuarios finales (Boaventura, Peña, Verdecia, & Fustiel, 2016).

La selección de la metodología adecuada para desarrollar determinado proyecto es clave para el éxito del mismo, ya que se llevarían a cabo actividades requeridas de acuerdo las características específicas del proyecto y su entorno.

En la actualidad existen dos grupos de metodologías para el desarrollo de software. Inicialmente se crearon las metodologías tradicionales, luego surgieron las metodologías ágiles. Las metodologías tradicionales proporcionan importancia al seguimiento y la planificación predictiva, que establecen un rigor en el proceso de desarrollo de software. Las metodologías ágiles surgen como reacción de la filosofía utilizada en las metodologías tradicionales. Los métodos ágiles están basados en entregas frecuentes de versiones de software funcionales, con mayor relevancia a la planificación adaptativa, colaboración con el cliente y respuestas ante los cambios inherentes al desarrollo de software (Boaventura, Peña, Verdecia, & Fustiel, 2016).

Las metodologías imponen un proceso disciplinado sobre el desarrollo de software con el objetivo de hacerlo más predecible y eficiente, donde predecir no significa perder la capacidad adaptativa, no significa evitar la introducción de cambios en los requisitos, ni evitar que nuevos requisitos surjan sino definir un camino reproducible para obtener resultados confiables. Definen, además, una representación que permite facilitar la manipulación de modelos, la comunicación e intercambio de información entre todas las partes involucradas en la construcción de un sistema (Gacitúa, 2003).

### **2.3.1. METODOLOGÍAS TRADICIONALES**

Las metodologías tradicionales de desarrollo de software son orientadas por planeación. Inician el desarrollo de un proyecto con un riguroso proceso de licitación de requerimientos, previo a etapas de análisis y diseño. Con esto tratan de asegurar resultados con alta calidad circunscritos a un calendario.

En las metodologías tradicionales se concibe un solo proyecto, de grandes dimensiones y estructura definida; se sigue un proceso secuencial en una sola dirección y sin marcha atrás; el proceso es rígido y no cambia; los requerimientos son acordados de una vez y para todo el proyecto, demandando grandes plazos de planeación previa y poca comunicación con el cliente una vez ha terminado ésta (Khurana & Sohal, 2011).

### **2.3.2. METODOLOGÍAS AGILES**

Las metodologías ágiles son flexibles, pueden ser modificadas para que se ajusten a la realidad de cada equipo y proyecto.

Los proyectos ágiles se subdividen en proyectos más pequeños mediante una lista ordenada de características. Cada proyecto es tratado de manera independiente y desarrolla un subconjunto de características durante un periodo de tiempo corto, de entre dos y seis semanas (Navarro Cadavid, Fernández Martínez, & Morales Vélez, 2013). La comunicación con el cliente es constante al punto de requerir un representante de él durante el desarrollo. Los proyectos son altamente colaborativos y se adaptan mejor a los cambios; de hecho, el cambio en los requerimientos es una característica esperada y deseada, al igual

que las entregas constantes al cliente y la retroalimentación por parte de él. Tanto el producto como el proceso son mejorados frecuentemente (Ghosh, 2013).

### 2.3.3. COMPARACIÓN DE METODOLOGÍAS

La tabla 2.1 muestra aspectos relevantes de las metodologías de desarrollo tradicional contrastándolas con los aspectos relevantes de las metodologías de desarrollo ágil.

**Tabla 2. 1 Comparación de metodologías**

<b>Metodologías ágiles</b>	<b>Metodologías tradicionales</b>
Se basan en heurísticas provenientes de prácticas de producción de código	Se basan en normas provenientes de estándares seguidos por el entorno de desarrollo
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 muy controlado, numerosas normas
Contrato flexible e incluso inexistente	Contrato prefijado
El cliente es parte del desarrollo	Cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10)	Grupos grandes
Pocos artefactos	Más artefactos
Menor énfasis en la arquitectura del software	La arquitectura del software es esencial



**Fuente:** (Canós, 2005).

#### **2.3.4. MANIFIESTO POR EL DESARROLLO ÁGIL**

En 2001 se crea el Manifiesto por el desarrollo ágil de software, documento en el que se acuerdan cuatro principios básicos para el desarrollo de software, que establece prioridades y marca diferencias de fondo frente a los sistemas tradicionales: individuos e interacciones, por encima de procesos y herramientas; software funcionando, por encima de documentación extensiva; colaboración con el cliente, por encima de negociación contractual; y respuesta ante el cambio, por encima de seguir un plan (Beck, y otros, 2001).

Los principios que dan origen al manifiesto implican la satisfacción del cliente mediante entregas tempranas y continuas de software que funcione; requerimientos cambiantes en cualquier etapa del proyecto; participación activa del cliente; simplicidad; equipos de desarrollo motivados y auto-organizados; comunicación efectiva; auto inspecciones; y adaptación (Navarro Cadavid, Fernández Martínez, & Morales Vélez, 2013).

El manifiesto por el desarrollo ágil de software es el resultado del trabajo colaborativo de un grupo formado por diecisiete personas, entre desarrolladores de software, escritores y consultores, quienes lo construyeron y suscribieron en 2001. La firma y publicación del Manifiesto en ese año no implica que esa sea la fecha de origen de las metodologías ágiles o que antes de ese año no existieran, sino el reconocimiento de la necesidad de un lineamiento común capaz de hacer posible algún tipo de agrupación entre ellas (Sommerville, 2010).

En una primera selección de metodologías ágiles surge del manifiesto: Scrum, Extreme Programming [XP], Dynamic System Development Method [DSDM], Crystal, Adaptive Software Development [ASD] y Feature-Driven Development [FDD], están representadas en él, a través de al menos una de las personas que lo suscribieron (Beck, y otros, 2001).

## **2.4. SCRUM**

Su nombre no corresponde a una sigla, sino a un concepto deportivo, propio del rugby, relacionado con la formación requerida para la recuperación rápida del juego ante una ‘infracción menor (International Rugby Board, 2012). Su primera referencia en el contexto de desarrollo data de 1986, cuando Takeuchi y Nonaka utilizan el Rugby Approach para definir un nuevo enfoque en el desarrollo de productos, dirigido a incrementar su flexibilidad y rapidez, a partir de la integración de un equipo interdisciplinario y múltiples fases que se traslapan entre sí (Takeuchi & Nonaka, 1986).

La metodología Scrum para el desarrollo ágil de software es un marco de trabajo diseñado para lograr la colaboración eficaz de equipos en proyectos, que emplea un conjunto de reglas y artefactos y define roles que generan la estructura necesaria para su correcto funcionamiento.

### **2.4.1. CARACTERÍSTICAS**

- Scrum divide una organización en equipos pequeños, interdisciplinarios y auto-organizados.
- Divide el trabajo en una lista de entregables pequeños y concretos. Ordena la lista por orden de prioridad y estima el esfuerzo relativo de cada elemento.
- Divide el tiempo en iteraciones cortas de longitud fija (generalmente de 1 a 4 semanas), con código potencialmente entregable y demostrado después de cada iteración.
- Optimiza el plan de entregas y actualiza las prioridades en colaboración con el cliente, basada en los conocimientos adquiridos mediante la inspección del entregable después de cada iteración.
- Optimiza el proceso teniendo una retrospectiva después de cada iteración

### **2.4.2. PRINCIPIOS BÁSICOS**

Scrum propone trabajar en ciclos sobre entregas parciales de un producto final más amplio. Esta metodología de trabajo permite poder distribuir mejor el tiempo y evita que ante un proyecto muy extenso se quede estancado.

Con Scrum es posible desglosar estos proyectos amplios en una lista de tareas y, de esa manera, el trabajo se vuelve más ágil. Al comenzar cada ciclo se definen qué tareas se van a realizar a lo largo del mismo y al finalizar se entregan resultados concretos. Además, plantea una división de roles entre el equipo de trabajo lo que fomenta la cooperación con nuestros compañeros (Wingu, 2016).

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 (Scrum Alliance, 2012).

Los llamados Equipos Scrum son auto-gestionados, multifuncionales y trabajan en iteraciones. La autogestión les permite elegir la mejor forma de hacer el trabajo, en vez de tener que seguir lineamientos de personas que no pertenecen al equipo y carecen de contexto (Li, Moe, & Dybå, 2010).

Los integrantes del equipo tienen todos los conocimientos necesarios (por ser multifuncionales) para llevar a cabo el trabajo. La entrega del producto se hace en iteraciones; cada iteración crea nuevas funcionalidades o modifica las que el dueño del producto requiera (Navarro Cadavid, Fernández Martínez, & Morales Vélez, 2013).

### **2.4.3. PRÁCTICAS DE SCRUM**

Palacio (2014) refiere que SCRUM controla de forma empírica y adaptable la evolución del proyecto, empleando prácticas de la gestión ágil, ver tabla 2.2.

**Tabla 2. 2 Practicas de la gestión ágil**

<b>Prácticas de la gestión ágil</b>	<b>Descripción</b>
<b>Revisión de las iteraciones</b>	Al finalizar cada iteración (normalmente 30 días) se lleva a cabo una revisión con todas las personas implicadas en el proyecto. Este es el periodo máximo que se tarda en reconducir una desviación en el proyecto o en las circunstancias del producto.
<b>Desarrollo Incremental</b>	Durante el proyecto, las personas implicadas no trabajan con diseños o abstracciones. El desarrollo incremental implica que al final de cada iteración se dispone de una parte del producto operativa que se puede inspeccionar y evaluar.
<b>Desarrollo evolutivo</b>	Los modelos de gestión ágil se emplean para trabajar en entornos de incertidumbre e inestabilidad de requisitos. Intentar predecir en las fases iniciales cómo será el producto final, y sobre dicha predicción desarrollar el diseño y la arquitectura del producto no es realista, porque las circunstancias obligarán a remodelarlo muchas veces.
<b>Auto organización</b>	Durante el desarrollo de un proyecto son muchos los factores impredecibles que surgen en todas las áreas y niveles. La gestión predictiva confía la responsabilidad de su resolución al gestor de proyectos.
<b>Colaboración</b>	Las prácticas y el entorno de trabajo ágiles

	<p>facilitan la colaboración del equipo. Ésta es necesaria, porque para que funcione la auto-organización como un control eficaz cada miembro del equipo debe colaborar de forma abierta con los demás, según sus capacidades y no según su rol o su puesto.</p>
--	--

**Fuente:** (Palacio, 2014).

#### 2.4.4. ROLES DE SCRUM

Schwaber & Sutherland (2011) describe tres roles en la tabla 2.3:

**Tabla 2. 3 Roles Scrum**

Nombre de rol	Descripción
<b>Scrum master</b>	El Scrum Master es el responsable en asegurar que se entienda y se adopte Scrum. Los Scrum Masters hacen esto asegurándose de que el Equipo Scrum trabaja ajustándose a la teoría, prácticas y reglas de Scrum.
<b>Dueño del producto</b>	Es una sola persona y representa a los interesados, es el responsable de maximizar el valor del producto y el trabajo del equipo de desarrollo; tiene entre sus funciones gestionar la lista ordenada de funcionalidades requeridas o Product Backlog.
<b>Equipo de desarrollo</b>	Tiene como responsabilidad convertir lo que el cliente quiere, el Product Backlog, en

	<p>iteraciones funcionales del producto; el equipo de desarrollo no tiene jerarquías, todos sus miembros tienen el mismo nivel y cargo: desarrollador. El tamaño óptimo del equipo está entre tres y nueve personas.</p>
--	--

**Fuentes:** (Schwaber & Sutherland, 2011)

### 2.4.5. EVENTOS DE SCRUM

En Scrum existen diferentes eventos predefinidos con el fin de crear regularidad y minimizar la necesidad de reuniones no definidas en Scrum. Todos los eventos son compartimientos o periodos de tiempo limitado *time-boxes*, de tal modo que todos tienen una duración máxima (Schwaber & Beedle, 2002).

Una vez que comienza un Sprint, su duración es fija y no puede acortarse o alargarse. Los otros eventos pueden terminar siempre que se alcance el objetivo del evento, asegurando que se emplee una cantidad apropiada de tiempo sin permitir desperdicio en el proceso.

Además del propio Sprint, que es un contenedor del resto de eventos, cada uno de los eventos de Scrum constituye una oportunidad formal para la inspección y adaptación en algún aspecto. Estos eventos se diseñaron específicamente para habilitar los pilares vitales de transparencia e inspección. La falta de alguno de estos eventos da como resultado una reducción de la transparencia y constituye una oportunidad perdida de inspección y adaptación (Schwaber & Sutherland, Scrum Guid, 2016).

#### 2.4.5.1. EL SPRINT

El corazón de Scrum es el Sprint, es un compartimiento o periodo de tiempo (*time-box*) de un mes o menos durante el cual se crea un incremento de producto “Terminado” utilizable y potencialmente desplegable. Es más conveniente si la duración de los Sprints es consistente a lo largo de todo el esfuerzo de desarrollo. Cada nuevo Sprint comienza inmediatamente después de la finalización del Sprint anterior (Rubin, 2012).

Los Sprints contienen y consisten en la Planificación del Sprint (*Sprint Planning*), los Scrums Diarios (*Daily Scrums*), el trabajo de desarrollo, 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 objetivos de calidad no disminuyen.
- El alcance puede clarificarse y renegociarse entre el Propietario del Producto (*Product Owner*) y el Equipo de desarrollo a medida que se va aprendiendo más.

Cada Sprint puede considerarse un proyecto con un horizonte no mayor de un mes. Al igual que los proyectos, los Sprints se usan para alcanzar algo. Cada Sprint tiene una definición de lo que se construirá, un diseño y un plan flexible que guiará su construcción, el trabajo del equipo y el producto resultante (Rubin, 2012).

Los Sprints están limitados a un mes calendario. Cuando el horizonte de un Sprint es demasiado grande la definición de lo que se está construyendo podría cambiar, la complejidad podría incrementarse y el riesgo podría aumentar. Los Sprints habilitan la predictibilidad al asegurar la inspección y adaptación del progreso al menos en cada mes calendario. Los Sprints también limitan el riesgo del coste a un mes calendario.

#### **2.4.5.2. PLANIFICACIÓN DEL SPRINT**

El trabajo a realizar durante el Sprint se planifica en la reunión de Planificación del Sprint (*Sprint Planning*). Este plan se crea mediante el trabajo colaborativo de todo el Equipo Scrum (Schwaber & Sutherland, Scrum Guid, 2016).

La Planificación del Sprint (*Sprint Planning*) tiene una duración máxima de ocho horas para un Sprint de un mes. Para Sprints más cortos el evento es usualmente más corto. El Scrum Master se asegura de que el evento se lleve a cabo y que los asistentes entiendan su

propósito. El Scrum Master enseña al Equipo Scrum a mantenerse dentro del periodo de tiempo.

La Planificación de Sprint responde a las siguientes preguntas:

- ¿Qué puede entregarse en el Incremento resultante del Sprint que comienza?
- ¿Cómo se conseguirá hacer el trabajo necesario para entregar el Incremento?

#### **2.4.5.3. SCRUM DIARIO**

El Scrum Diario es una reunión con una duración máxima de tiempo de 15 minutos para que el Equipo de Desarrollo (*Development Team*) sincronice sus actividades y cree un plan para las siguientes 24 horas. Esto se lleva a cabo inspeccionando el trabajo avanzado desde el último Scrum Diario (*Daily Scrum*) y haciendo una proyección acerca del trabajo que podría completarse antes del siguiente (Schwaber & Sutherland, Scrum Guid, 2016).

El Scrum Diario (*Daily Scrum*) se realiza a la misma hora y en el mismo lugar todos los días para reducir la complejidad. Durante la reunión, cada miembro del Equipo de Desarrollo (*Development Team*) explica:

- ¿Qué hice ayer para ayudar al Equipo de Desarrollo (*Development Team*) a lograr el Objetivo del Sprint?
- ¿Qué haré hoy para ayudar al Equipo de Desarrollo (*Development Team*) a lograr el Objetivo del Sprint?
- ¿Detecto algún impedimento que evite que el Equipo de Desarrollo (*Development Team*) o yo terminen el Objetivo del Sprint?

El Equipo de Desarrollo (*Development Team*) utiliza el Scrum Diario (*Daily Scrum*) para evaluar el progreso hacia el objetivo del Sprint y para evaluar qué tendencia sigue este progreso hacia la finalización del trabajo contenido en la Pila del Sprint (*Sprint Backlog*). El Scrum Diario (*Daily Scrum*) optimiza las posibilidades de que el Equipo de Desarrollo (*Development Team*) cumpla el objetivo del Sprint. Cada día, el Equipo de Desarrollo



(*Development Team*) debería entender cómo intenta trabajar en conjunto como un equipo auto-organizado para lograr el Objetivo del Sprint y crear el Incremento esperado hacia el final del Sprint. El Equipo de Desarrollo a menudo se vuelve a reunir inmediatamente después del Scrum Diario (*Daily Scrum*), para tener discusiones detalladas, o para adaptar o replanificar el resto del trabajo del Sprint.

#### **2.4.5.4. REVISIÓN DEL SPRINT**

Al final del Sprint se lleva a cabo una Revisión de Sprint (*Sprint Review*) para inspeccionar el Incremento y adaptar la Pila del Producto (*Product Backlog*) si fuese necesario (Truex, Baskerville, & Travis, 2000).

Durante la Revisión de Sprint (*Sprint Review*), el Equipo Scrum y los interesados colaboran acerca de lo que se hizo durante el Sprint. Basándose en esto y en cualquier cambio a la Pila del Producto.

(*Product Backlog*) durante el Sprint, los asistentes colaboran para determinar las siguientes cosas que podrían hacerse para optimizar el valor.

Se trata de una reunión informal, no una reunión de *seguimiento*, y la presentación del Incremento tiene como objetivo facilitar la retroalimentación de información y fomentar la colaboración.

Se trata de una reunión restringida a un compartimiento máximo de tiempo de cuatro horas para Sprints de un mes. Para Sprints más cortos, se reserva un tiempo usualmente más corto.

El Scrum Master se asegura de que el evento se lleve a cabo y que los asistentes entiendan su propósito. El Scrum Master enseña a todos a mantener el evento dentro del bloque de tiempo fijado.

La Revisión de Sprint (*Sprint Review*) incluye los elementos que se muestran en la tabla 2.4:

**Tabla 2. 4 Elementos de revisión de un sprint**

Elementos	Descripción
Asistentes	Son el Equipo Scrum y los interesados clave invitados por El Propietario del Producto ( <i>Product Owner</i> ).
Propietario del Producto ( <i>Product Owner</i> )	Explica qué elementos de la Pila del Producto ( <i>Product Backlog</i> ) se han “Terminado” y cuales no se han “Terminado”.
Equipo de Desarrollo ( <i>Development Team</i> )	<ul style="list-style-type: none"> <li>• Habla acerca de qué estuvo bien durante el Sprint, qué problemas aparecieron y cómo fueron resueltos esos problemas.</li> <li>• Hace una demostración del trabajo que ha “Terminado” y responde preguntas acerca del Incremento.</li> </ul>
Propietario del Producto (Product Owner)	Habla acerca de la Pila del Producto ( <i>Product Backlog</i> ) en su estado actual. Proyecta fechas de finalización probables en el tiempo basándose en el progreso obtenido hasta la fecha (si es necesario).
Grupo completo	Colabora acerca de qué hacer a continuación, de modo que la Revisión del Sprint proporcione información de entrada valiosa para Reuniones de Planificación de Sprints subsiguientes.
Revisión	<ul style="list-style-type: none"> <li>• Revisión de cómo el mercado o el uso potencial del producto podría</li> </ul>

	<p>haber cambiado lo que es de más valor para hacer a continuación.</p> <ul style="list-style-type: none"> <li>• Revisión de la línea de tiempo, presupuesto, capacidades potenciales y mercado para la próxima entrega prevista del producto.</li> </ul>
--	---

**Fuente:** (Truex, Baskerville, & Travis, 2000).

#### **2.4.5.5. RETROSPECTIVA DEL SPRINT**

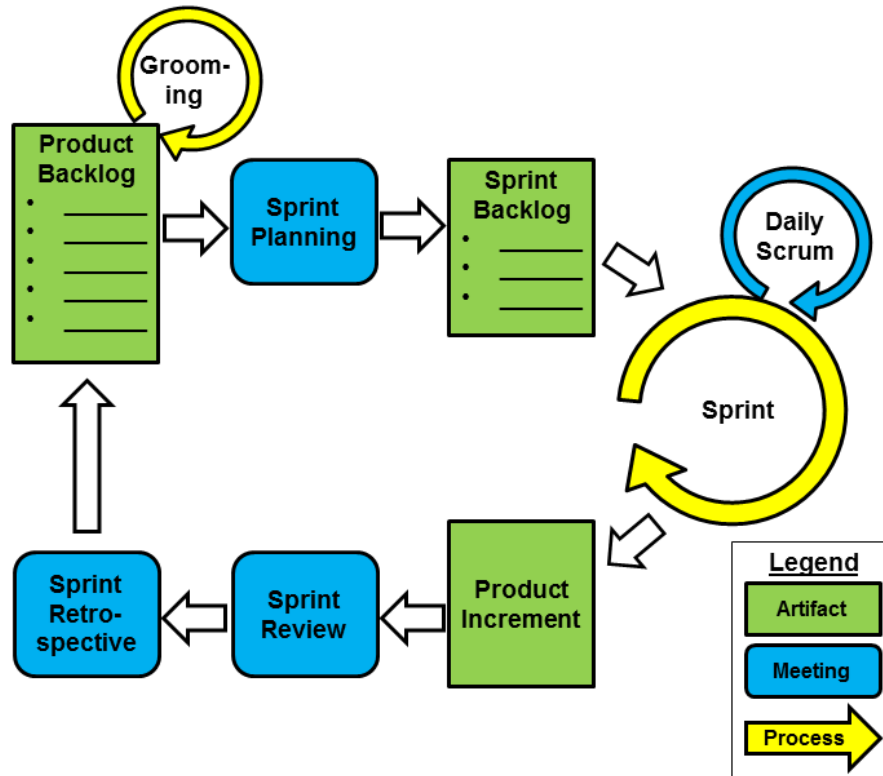
La Retrospectiva del Sprint (*Sprint Retrospective*) es una oportunidad para el Equipo Scrum de inspeccionarse a sí mismo y de crear un plan de mejoras que sean abordadas durante el siguiente Sprint.

La Retrospectiva del Sprint (*Sprint Retrospective*) tiene lugar después de la Revisión de Sprint (*Sprint Review*) y antes de la siguiente Planificación de Sprint. Se trata de una reunión restringida a un compartimiento máximo de tiempo de tres horas para Sprints de un mes. Para Sprints más cortos se reserva un tiempo usualmente más corto. El Scrum Master se asegura de que el evento se lleve a cabo y que los asistentes entiendan su propósito. El Scrum Master enseña a todos a mantener el evento dentro del bloque de tiempo fijado. El Scrum Master participa en la reunión como un miembro del equipo ya que la responsabilidad del proceso Scrum recae sobre él. El propósito de la Retrospectiva del Sprint (*Sprint Retrospective*) es:

- Inspeccionar cómo fue el último Sprint en cuanto a personas, relaciones, procesos y herramientas;
- Identificar y ordenar los elementos más importantes que salieron bien y las posibles mejoras;

- Crear un plan para implementar las mejoras a la forma en la que el Equipo Scrum desempeña su trabajo.

Este framework tradicional de Scrum se puede ver como un todo en la Figura 2.1.



**Figura 2. 1.** Framework tradicional de Scrum

Fuente: (Rubin, 2012).

El Scrum Master motiva al equipo para que mejore, dentro del marco de proceso Scrum, su proceso de desarrollo y sus prácticas para hacerlos más efectivos y amenos para el siguiente Sprint. Durante cada Retrospectiva del Sprint (*Sprint Retrospective*), el Equipo Scrum planifica formas de aumentar la calidad del producto mediante la adaptación de la Definición de “Terminado” (*Definition of “Done”*) según sea conveniente.

Al final de la Retrospectiva del Sprint (*Sprint Retrospective*) el Equipo Scrum debería haber identificado mejoras que implementará en el próximo Sprint. El hecho de implementar estas mejoras en el siguiente Sprint constituye la adaptación subsecuente a la inspección del Equipo de Desarrollo (*Development Team*) mismo. Aunque las mejoras pueden implementarse en cualquier momento, la Retrospectiva del Sprint (*Sprint Retrospective*) ofrece un evento dedicado para este fin, enfocado en la inspección y la adaptación.

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

Los artefactos de Scrum son subproductos de las actividades del marco de trabajo que le brindan dirección y transparencia al equipo (Blankenship, Bussa, & Millett, 2011). Los artefactos de Scrum son:

- Product Backlog
- Sprint Backlog
- Incremento

##### **2.4.6.1. PRODUCT BACKLOG**

El Product Backlog es una lista ordenada de todo lo que podría necesitarse en el producto y es la única fuente de requerimientos para los cambios que se realizarán en el producto. El Product Owner es responsable del Product Backlog incluyendo su contenido, disponibilidad y jerarquización (Rising & Janoff, 2000).

El Product Backlog nunca se termina. Inicialmente, contiene los requerimientos de lo que se conoce y entiende al momento de iniciar el desarrollo. El Product Backlog evoluciona conforme el producto y su entorno evolucionan. El Product Backlog es dinámico; cambia constantemente de acuerdo a lo que el producto requiere para mantenerse adecuado, competitivo y útil. Mientras exista un producto, su Product Backlog también existe (Rising & Janoff, 2000).

El Product Backlog incluye todas las características, funciones, requerimientos, mejoras y correcciones que constituyen los cambios que deben introducirse en el producto en futuras versiones. Los elementos del Product Backlog deben contener los siguientes atributos: **descripción, orden, estimación y valor.**

Conforme el producto se utiliza y su valor útil aumenta y recibe retroalimentación del mercado, el Product Backlog crece y se convierte en una lista más exhaustiva. Cambios en las necesidades del negocio, las condiciones del mercado o la tecnología tienen un impacto directo en el Product Backlog, por lo que es un Artefacto dinámico (Fried, 2000).

Es común que varios Equipos Scrum trabajen en un mismo producto y, por lo tanto, utilicen un mismo Product Backlog para identificar el trabajo a realizar. En estos casos, se podrá incluir un atributo que agrupe los elementos registrados.

El refinamiento del Product Backlog es la tarea de añadir detalles, estimaciones, y orden a los elementos del Product Backlog. Es un proceso continuo en el que el Product Owner y el Equipo de Desarrollo colaboran en aportar detalles a los elementos en El Product Backlog. Durante el refinamiento del Product Backlog, los artículos son revisados y examinados. El Equipo Scrum decide cómo y cuándo se realiza el refinamiento.

#### **2.4.6.2 SPRINT BACKLOG**

El Sprint Backlog es el conjunto de elementos seleccionados del Product Backlog para el Sprint, además de un plan para la entrega del Incremento y el cumplimiento del objetivo del Sprint. El Sprint Backlog es una proyección realizada por el Equipo de Desarrollo sobre la funcionalidad que estará en el próximo Incremento y el trabajo necesario para convertir esa funcionalidad en un incremento "Terminado" (Clifton & Dunlap, 2003).

El Sprint Backlog hace visible todo el trabajo que el equipo de desarrollo ha identificado como necesario para cumplir con el objetivo del Sprint.

El Sprint Backlog es un plan con suficiente detalle como para que los cambios en curso puedan ser entendidos durante el Scrum Diario. El Equipo de Desarrollo modifica el Sprint

Backlog durante todo el ciclo y, conforme avanza en el plan y entiende mejor el trabajo necesario para lograr el objetivo, el Sprint Backlog emerge y se concreta (Sutherland, 2007).

Conforme se identifican nuevas tareas, el Equipo de Desarrollo las añade al Sprint Backlog. Conforme se realizan o completan, el tiempo para completar el trabajo se actualiza. Cuando se identifican tareas que ya no son necesarias, se eliminan.

Sólo el Equipo de Desarrollo puede realizar cambios en el Sprint Backlog, el cual representa una imagen en tiempo real de lo que planea realizar durante el Sprint. Por lo tanto, este artefacto pertenece únicamente al Equipo de Desarrollo.

En cualquier momento, el trabajo restante para completar el Sprint puede ser cuantificado. El Equipo de Desarrollo monitorea este progreso para asegurarse que, por lo menos, **cada Scrum Diario refleje la probabilidad de alcanzar el objetivo del Sprint**. Mediante el seguimiento de las tareas pendientes en el Sprint, el equipo de desarrollo puede administrar su progreso.

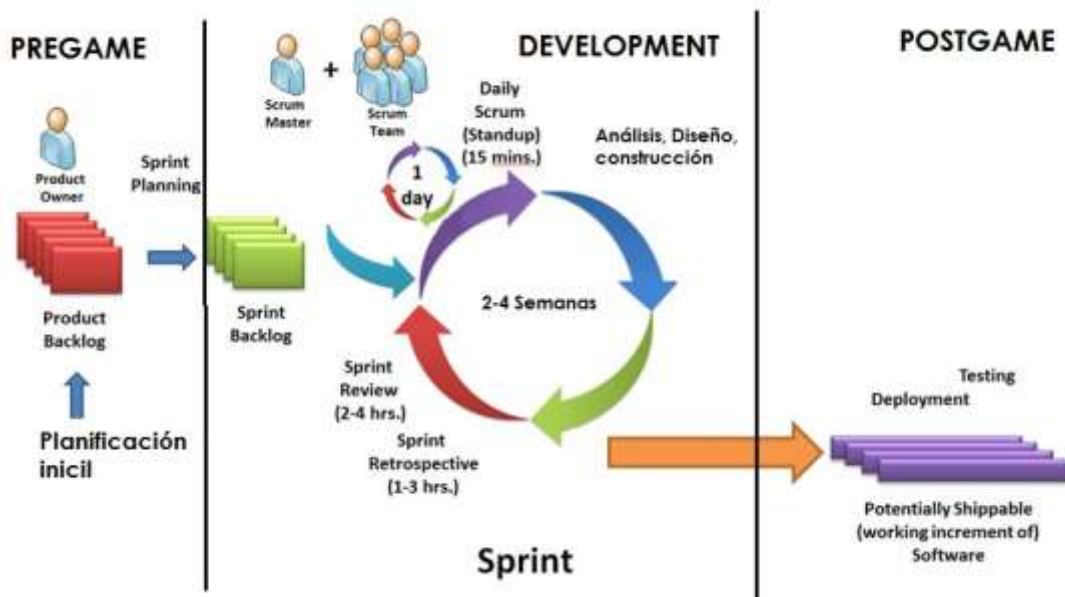
#### **2.4.6.3. INCREMENTO**

El Incremento es la suma de todos los elementos de la Pila del Producto (*Product Backlog*) completados durante un Sprint y el valor de los incrementos de todos los Sprints anteriores. Al final de un Sprint el nuevo Incremento debe estar “Terminado”, lo cual significa que está en condiciones de ser utilizado y que cumple la Definición de “Terminado” del Equipo Scrum.

El incremento debe estar en condiciones de utilizarse sin importar si el Propietario del Producto (*Product Owner*) decide liberarlo o no (Schwaber & Sutherland, Scrum Guid, 2016).

#### **2.4.7. FASES DE SCRUM**

Scrum consta de tres fases: *Pregame*, *Development* y *Postgame* (Schwaber & Beedle, 2002). En la figura 2.2 se muestra los elementos de scrum en su respectiva fase.



**Figura 2.2** Proceso scrum

Fuente: (Schwaber & Beedle, 2002)

#### 2.4.7.1. PREGAME

La fase de *pregame* incluye dos subfases: *planning* y *architecture*, que se muestra en la tabla 2.5.

**Tabla 2. 5** Subfases de Pregame

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 conocido. 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



	<p>necesario para su implementación es estimado. La lista <i>product backlog</i> se actualiza constantemente 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>
<p><b>Arquitectura</b></p>	<p>Se planifica en función de los elementos actuales en la lista <i>product backlog</i>. En el caso de una mejora de un sistema existente, se identifican los cambios necesarios para implementar los ítems de <i>backlog</i> junto con los problemas que pueden causar. 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>

**Fuente:** (Schwaber & Beedle, 2002)

#### **2.4.7.2. DEVELOPMENT**

La fase de Development también llamada Game Phase es la parte ágil del enfoque de Scrum. Esta fase se trata como una caja negra donde se espera lo impredecible. Las diferentes variables ambientales y técnicas (como marco de tiempo, calidad, requisitos, recursos, tecnologías y herramientas de implementación e incluso métodos de desarrollo) identificadas en Scrum, que pueden cambiar durante el proceso, se observan y controlan mediante diversas prácticas de Scrum durante el proceso. En lugar de tomar estos asuntos en consideración solo al comienzo del proyecto de desarrollo de software, Scrum apunta a controlarlos constantemente para poder adaptarse de manera flexible a los cambios.

En la fase de desarrollo, el sistema se desarrolla en Sprints (Figura 2.2). Los sprints son ciclos iterativos donde la funcionalidad se desarrolla o mejora para producir nuevos incrementos. Cada Sprint incluye las fases tradicionales de desarrollo de software: fases de requisitos, análisis, diseño, evolución y entrega (Figura 2.2). La arquitectura y el diseño del sistema evolucionan durante el desarrollo de Sprint. Se espera que un Sprint dure de una semana a un mes. Puede haber, por ejemplo, de tres a ocho Sprints en un proceso de desarrollo de sistemas antes de que el sistema esté listo para su distribución. También puede haber más de un equipo construyendo el incremento.

#### **2.4.7.3. POSTGAME**

La fase postgame contiene el cierre del lanzamiento. Esta fase es ingresada cuando se ha llegado a un acuerdo que las variables ambientales como los requisitos se completan. En este caso, no se pueden encontrar más elementos y problemas ni se pueden inventar nuevos. El sistema ahora está listo para el lanzamiento y la preparación para esto se hace durante la fase postgame, incluidas las tareas como la integración, las pruebas del sistema y la documentación (Figura 2.2).

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

El objetivo de las reuniones de trabajo es facilitar la transferencia de información y la colaboración entre los miembros del equipo para aumentar su productividad, al poner de

manifiesto puntos en que se pueden ayudar unos a otros, éstas reuniones se muestran en la tabla 2.6:

**Tabla 2. 6 Características de una reunión de trabajo en un contexto SCRUM**

Características	Descripción
<b>Planificación de sprint</b>	Se realiza al principio de cada ciclo de sprint, y está encaminada a seleccionar el conjunto de requerimientos del <i>Product backlog</i> que serán abordados, el equipo de trabajo que será necesario y el tiempo que se estima (entre 1 y 4 semanas) para su desarrollo.
<b>Reunión diaria</b>	Conocida como <i>Daily Scrum</i> , se realiza al comienzo de cada día en que ese esté ejecutando un sprint. Es una reunión corta (no más de 30 minutos) en la que los integrantes del equipo responden las siguientes preguntas: <ul style="list-style-type: none"> <li>• ¿Qué has hecho desde la última reunión?</li> <li>• ¿Qué problemas has encontrado para realizar el trabajo previsto?</li> <li>• ¿Qué planeas hacer antes de la próxima reunión?</li> </ul>
<b>Revisión de sprint</b>	Una vez concluido el ciclo de sprint se mantiene una reunión en la que se define qué parte del trabajo previsto se ha completado y qué parte permanece pendiente. En cuanto al trabajo completado se realiza una revisión (demo) del mismo al <i>Product owner</i> y otros usuarios que pudiesen estar involucrados.
<b>Retrospectiva de sprint</b>	Es una reunión en la que todos los miembros del equipo realizan una valoración del trabajo realizado en el último sprint, identificando puntos de mejora de cara a

	los siguientes a realizar. El objetivo principal es introducir un componente de mejora continua en el proceso.
<b>Incremento</b>	Es la suma de todos los elementos del <i>Product Backlog</i> completos durante un Sprint, más el valor de todos los Incrementos anteriores. Al final de un Sprint, el nuevo incremento debe poder marcarse como "Terminado", lo que significa que debe estar en condición de ser utilizable y cumplir con la definición del Equipo Scrum de "Terminado". El incremento debe estar en condición de ser utilizado, independientemente de que el <i>Product Owner</i> decida o no liberarlo.

**Fuente:** (Blankenship, Bussa, & Millett, 2011).

## 2.5. KANBAN

El término viene del japonés: Kan, visual, y ban, tarjeta. Kanban permite visualizar el flujo de trabajo en una barra de tareas a través de tarjetas. Propone distribuir las mismas en una serie de columnas. Kanban trabaja con tableros que pueden ser tanto físicos como digitales y permite una visualización clara de todas las tareas a realizar, en qué etapa está cada una y quién es el encargado de las mismas (Brown & Rocher, 2013).

El Kanban es un sistema de gestión del trabajo en curso (WIP3), que sirve principalmente para asegurar una producción continua y sin sobrecargas en el equipo de producción multimedia. El Kanban es un sistema de gestión donde se produce exactamente aquella cantidad de trabajo que el sistema es capaz de asumir (Avison & Fitzgerald, 1999).

El Kanban es un sistema de trabajo *just in time*, lo que significa que evita sobrantes innecesarios de stock, que en la gestión de proyectos multimedia equivale a la inversión innecesaria de tiempo y esfuerzo en lo que no se necesitara y evita sobrecargar al equipo (Bermejo, 2010).

### 2.5.1. CARACTERÍSTICAS

Según Ibarra Guzmán & Castañeda Islas (2014) las principales características de Kanban se lo puede observar en la tabla 2.7:

**Tabla 2. 7 Características de Kanban**

<b>Características</b>	<b>Descripción</b>
<b>Visualiza el flujo de trabajo</b>	Divide el trabajo en bloques, escribe cada elemento en una tarjeta y se coloca en el tablero. Utiliza columnas con nombre para ilustrar dónde está cada elemento en el flujo de trabajo.
<b>Limita el trabajo en curso (WIP<sup>3</sup>)</b>	Asigna límites concretos al número de elementos que pueden estar en progreso en cada estado del flujo de trabajo.
<b>Mide el tiempo de ciclo (lead time) medio para completar un elemento</b>	Optimiza el proceso para que el lead time sea tan pequeño y predecible como sea posible.

**Fuente:** (Ibarra Guzmán & Castañeda Islas, 2014).

### 2.5.2. ROLES KANBAN

Kanban, a diferencia de otras metodologías ágiles como Scrum, no prescribe roles o reuniones. Pero existen una serie de roles que pueden ser adoptados en Kanban, estos han sido creados a partir de la observación en organizaciones que han utilizado el método y lo han evolucionado de forma colaborativa (Palacios, 2016).

- **Service Request Manager:** Se encarga de gestionar la demanda y los requisitos dentro del sistema Kanban, manejando las relaciones con los *stakeholders* y

---

<sup>3</sup> Del inglés, work in progress.



**Tabla 2. 8 Elementos de un tablero Kanban**

<b>Elementos</b>	<b>Descripción</b>
<b>Tareas o actividades a realizar</b>	Se representan en el tablero mediante tarjetas haciendo alusión al origen etimológico de la palabra Kanban. Las tarjetas que representan cada tarea o actividad se mueven a través de las columnas describiendo el estado en el que se encuentra.
<b>Columnas</b>	Estas representan las fases o etapas por las que debe pasar una actividad antes de considerarse terminada. Definir el número de columnas que debe tener un tablero Kanban siempre dependerá de las características del proyecto al que se quiera aplicar y del manual de procesos que se tenga en cada ente organizacional, ya que, como se mencionó anteriormente, la intención del Kanban no es contaminar ni cambiar el flujo y las formas de trabajo sino aportar una representación visual al mismo.
<b>WIP</b>	No es propiamente un elemento tangible dentro del tablero pero sin el cual no se podría tener un control sobre las actividades máximas soportadas por el equipo de trabajo. Esto es tal vez lo que más dificultad presenta al momento de implantar un

	tablero <i>Kanban</i> o la metodología por si misma debido a que nunca se tiene definido el número finito de tareas o actividades que se deben realizar por cada fase que compone el proceso de desarrollo del proyecto
--	---

**Fuente:** (Sundén & Hammarberg, 2014).

#### **2.5.4. VISUALIZAR EL WORKFLOW (FLUJO DE TRABAJO)**

Consiste en la representación visual de todo el flujo de trabajo mediante un tablero físico, asequible, que refleje la realidad del equipo en cada momento (Kniberg & Skarin, 2010).

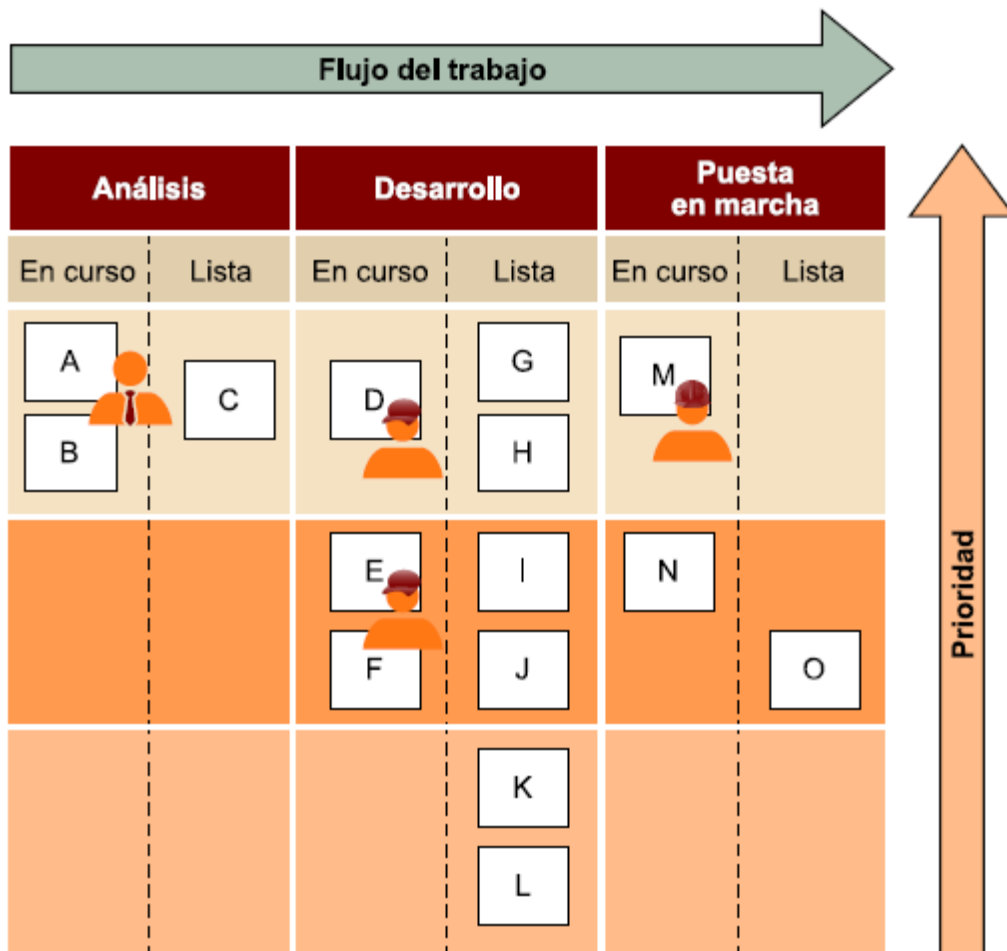
El objetivo de mostrar todo el proceso consiste en:

- Entender mejor el proceso actual.
- Conocer los problemas que puedan surgir y tomar decisiones.
- Mejorar la comunicación entre todo el equipo del proyecto.
- Hacer futuros procesos más predecibles.
- Conocer la información visual de qué miembro del equipo está ejecutando cada tarea.

#### **2.5.5. FASES DEL KANBAN**

La figura 2.2 muestra un tablero Kanban constituido por 3 columnas, que representan las diferentes fases por las que una tarea tiene que pasar para ser terminada.





**Figura 2. 4** Panel Kanban

**Fuente:** (Bermejo, 2010)

En esta imagen se puede observar un panel constituido por tres columnas, que representan las diferentes fases por las que una tarea tiene que fluir para ser desarrollada (análisis, desarrollo y puesta en marcha).

Cada fase está subdividida en dos estados, que son en curso y lista, para pasar a la siguiente fase; esta división está representada por la línea discontinua de cada fase. El estado en curso significa que el equipo está actualmente trabajando en esta tarea, en esta fase y el estado lista significa que el equipo ya ha acabado el trabajo que tenía que ejecutar en esta fase y la

tarea está esperando a que el sistema pueda asumirla para la siguiente fase. Esta división nos ayuda a localizar atascos en el proceso de producción (Morales, 2015).

Las filas podrían ser diferentes proyectos en los que la empresa está trabajando, pero lo más habitual es que las filas indiquen prioridad, donde las tareas más superiores son las más prioritarias.

### **2.5.6. REUNIONES DE EQUIPO**

Kanban establece reuniones rápidas de todo el equipo alrededor del tablero, para que cada miembro cuente en qué está trabajando y en qué etapa se encuentra. En este breve encuentro se reenfochan las prioridades (Stellman & Greene, 2014).

Estas reuniones no deben tener una duración mayor a 15 minutos y pueden realizarse diariamente o de manera más espaciada (dependiendo de las necesidades y posibilidades de cada organización).

## **2.6. COMBINACIÓN DE SCRUM Y KANBAN SCRUMBAN**

Scrumban combina las mejores características de ambos métodos. Reúne la naturaleza preceptiva de Scrum y la capacidad de mejora del proceso de Kanban, permitiendo a los equipos moverse hacia el desarrollo Agile y a mejorar constantemente sus procesos. Scrumban se está haciendo especialmente popular en industrias en las que el desarrollo del proyecto y el mantenimiento van juntos (Palacios, 2016).

### **2.6.1. DIFERENCIAS BÁSICAS ENTRE SCRUM Y KANBAN**

El Kanban no prescribe roles ni iteraciones fijas. Las entregas se pueden definir en función de las necesidades de la empresa: de manera regular (todos los viernes) o cada vez que se acabe un servicio y se ponga en producción. Scrum no prescribe límite WIP o, como mínimo, no directamente. Scrum limita el trabajo en curso al trabajo comprometido en la iteración actual, es decir, si en el *sprint planning meeting* se decide llevar a cabo diez funcionalidades en dos semanas, el primer día del *sprint*, el equipo podría tener en desarrollo las diez a la vez (Bermejo, 2010).

Scrum no permite la entrada de trabajo nuevo cuando la iteración ha empezado. Si en el *sprint* se han comprometido diez tareas, Scrum no permite cambiarlas ni añadir otras nuevas a media iteración, por lo tanto en equipos donde las tareas estén definidas a corto plazo, iteraciones muy cortas son casi obligatorias. Si en la empresa pueden entrar tareas de hoy para mañana, estas no pueden ser asumidas por Scrum puesto que sirve desarrollos en iteraciones fijas de como mínimo una semana (Stellman & Greene, 2014).

### **2.6.2. BENEFICIOS DE LA INTEGRACIÓN**

- Permite conocer en estado real el proceso de ejecución del proyecto.
- Introduce soluciones oportunas ante eventuales errores.
- Permite un mayor análisis de tareas realizadas.
- Mejora la interacción entre los miembros de un grupo en las reuniones periódicas.
- Aumenta la productividad de proyectos complejos o multiproyectos.
- Favorece una mayor adaptabilidad de las herramientas a las exigencias del proyecto.

### **2.7. INGENIERÍA WEB**

El área de Ingeniería Web es relativamente una nueva dirección de la Ingeniería de Software para el desarrollo de Aplicaciones Web (Kappel, Pröll, Reich, & Retschizgge, 2006). La Ingeniería Web trata varios aspectos, metodologías, herramientas y técnicas que hacen único del desarrollo y construcción de aplicaciones que se ejecutan en la World Wide Web (Murugesan, 2008).

Pressman (2010) refiere que el diseño de webapps<sup>4</sup> incluye actividades técnicas y no técnicas, que incluyen lo siguiente: establecer la vista y sensación de la webapp, creando la distribución estética de la interfaz de usuario, definiendo la estructura arquitectónica general, desarrollando el contenido y la funcionalidad que residen en la arquitectura y planeando la navegación que ocurre dentro de la webapp.

---

<sup>4</sup> Aplicaciones web

Para el desarrollo de modelos conceptuales de aplicaciones Web existen varios métodos de diseño en Ingeniería Web, por ejemplo: OOHDM (*Object-Oriented Hypermedia Design Model*), WebML (Web Modeling Language), OO-H (Object Oriented approach), UWE (UML Web Engineering), entre otros. UWE fue uno de los primeros proyectos usado especialmente para aplicaciones Web (Ceke, Durek, & Kasapovic, 2013).

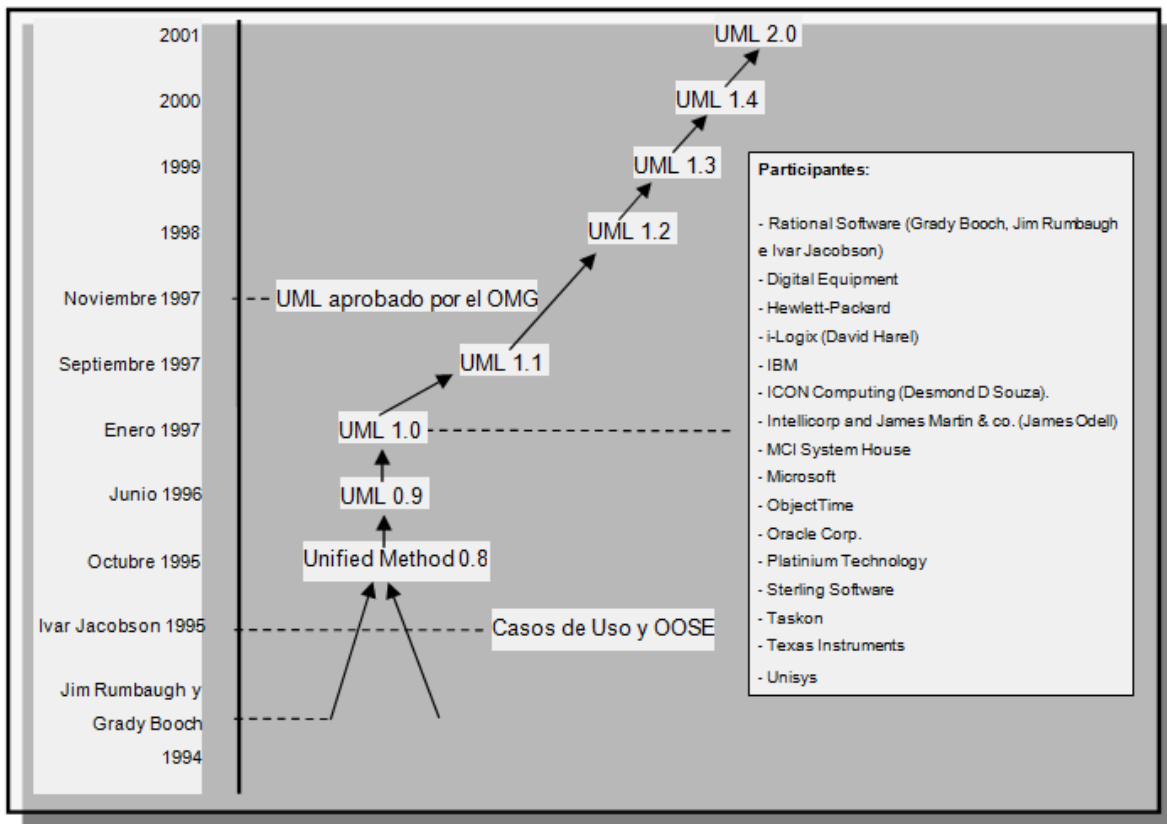
La Ingeniería Web basada en UML, es un proceso del desarrollo para aplicaciones Web enfocado sobre el diseño sistemático, la personalización y la generación semiautomática de escenarios que guíen el proceso de desarrollo de una aplicación Web. UWE describe una metodología de diseño sistemática, basada en las técnicas de UML, la notación de UML y los mecanismos de extensión de UML (Koch & Kraus, 2002).

## **2.8. LENGUAJE UNIFICADO DE MODELADO (U.M.L.)**

UML se define como un "lenguaje que permite especificar, visualizar y construir los artefactos de los sistemas de software" (Booch, Rumbaugh, & Jacobson, 1998, p. 141). Es un lenguaje notacional (que, entre otras cosas, incluye el significado de sus notaciones) destinado a los sistemas de modelado que utilizan conceptos orientados a objetos.

El UML es un estándar para construir modelos orientados a objetos. Nació en 1994 por iniciativa de Grady Booch y Jim Rumbaugh para combinar sus dos famosos métodos: el de Booch y el OMT (*Object Modeling Technique*, Técnica de Modelado de Objetos). Más tarde se les unió Ivar Jacobson, creador del método OOSE (*Object-Oriented Software Engineering*, Ingeniería de Software Orientada a Objetos).

En respuesta a una petición del OMG (*Object Management Group*, Grupo de Administración de Objetos) para definir un lenguaje y una notación estándar del lenguaje de construcción de modelos, en 1997 propusieron el UML como candidato (Larman C. , 1999). Ver Figura. 2.5.



**Figura 2. 5** Historia de UML

**Fuente:** (Letelier Torres, 2002)

### 2.8.1. VISTAS DEL UML

En la construcción de software usando UML, existen cinco vistas para visualizar, especificar, construir y documentar la arquitectura del software (Booch, Rambaugh, & Jacobson, 1998). UML permite representar cada vista mediante un conjunto de diagramas, las vistas se muestran en la tabla 2.9:

**Tabla 2. 9** Representación de vistas mediante diagramas

Vistas	Descripción
Vista de casos de uso	Muestra la funcionalidad del sistema desde el punto de vista de un actor externo que interactúa con él. Esta vista es útil a

	clientes, diseñadores y desarrolladores.
<b>Vista de diseño</b>	<p>Muestra la funcionalidad del diseño dentro del sistema en términos de la estructura estática y comportamiento dinámico del sistema.</p> <p>Esta vista es útil a diseñadores y desarrolladores. Se definen propiedades tales como: persistencia, concurrencia, interfaces y estructuras internas a las clases.</p>
<b>Vista de procesos</b>	<p>Muestra la concurrencia del sistema, comunicación y sincronización.</p> <p>Útil a desarrolladores e integradores.</p>
<b>Vista de implementación</b>	<p>Muestra la organización de los componentes de código.</p> <p>Útil a desarrolladores.</p>
<b>Vista de implantación (también conocida como vista de despliegue)</b>	<p>Muestra la implantación del sistema en la arquitectura física.</p> <p>Útil a desarrolladores, integradores y verificadores.</p>

**Fuente:** (Booch, Rumbaugh, & Jacobson, 1998)

### 2.8.2. DEFINICIÓN DE MODELO

Un sistema (tanto en el mundo real como en el mundo del software) suele ser extremadamente intrincado, por ello es necesario dividir el sistema en partes o fragmentos si queremos entender y administrar su complejidad. Estas partes podemos representarlas como modelos que describan y abstraigan sus aspectos esenciales (Rea Cortés, 2004).

Un modelo captura una vista de un sistema del mundo real. Es una abstracción de dicho

sistema considerando un cierto propósito. Así, el modelo describe completamente aquellos aspectos del sistema que son relevantes al propósito del modelo y a un apropiado nivel de detalle.

Los modelos se componen de otros modelos, de diagramas y documentos que describen detalles del sistema. El UML especifica varios diagramas.

Si queremos caracterizar los modelos, podemos poner de manifiesto la información estática o dinámica de un sistema. Un modelo estático describe las propiedades estructurales del sistema; en cambio, un modelo dinámico describe las propiedades de comportamiento de un sistema.

Es importante mencionar que el UML es un lenguaje para construir modelos; no guía al desarrollador en la forma de realizar el análisis y diseño orientado a objetos ni indica cuál proceso de desarrollo adoptar (Larman C. , 1999, p. 37).

Para modelar un sistema es suficiente utilizar una parte de UML, "el 80 por ciento de la mayoría de los problemas pueden modelarse usando alrededor del 20 por ciento de UML" (Rea Cortés, 2004, p. 93).

### **2.8.3. DIAGRAMAS UML**

UML es un lenguaje notacional. Parte importante de esta notación son los diagramas que nos permiten modelar un sistema.

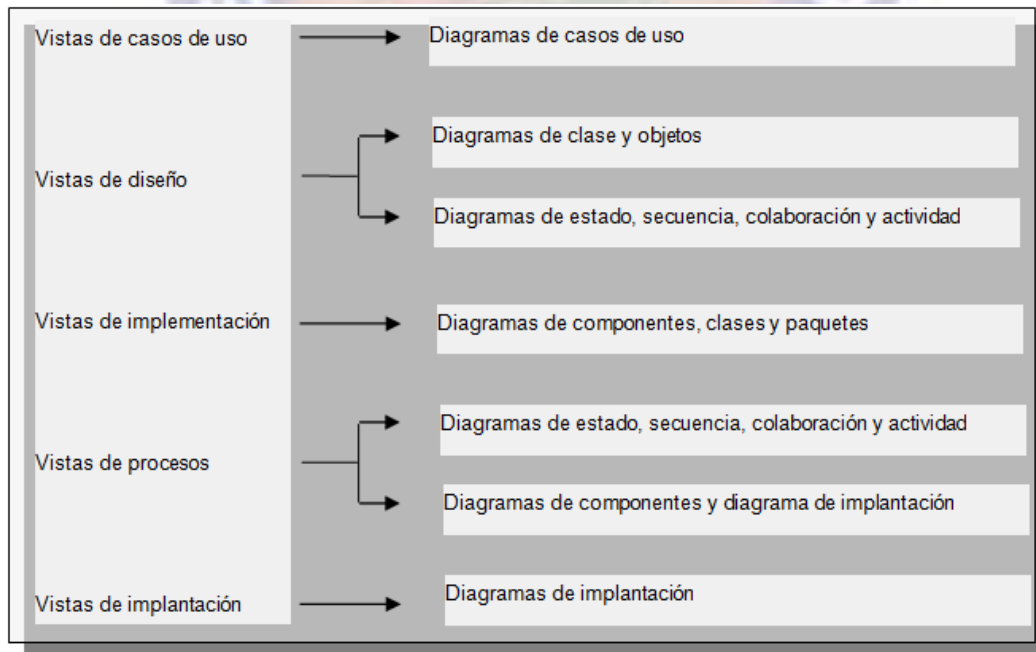
Un diagrama es una representación gráfica de una colección de elementos de modelado, la mayoría de las veces mostrados como grafo conexo de vértices (cosas) y arcos (relaciones). Los buenos diagramas hacen el sistema que se está desarrollando más comprensible y cercano a los objetivos (Booch, Rumbaugh, & Jacobson, 1997).

En UML se definen nueve diagramas, los cuales se pueden mezclar en cada vista, ver figura 2.6 y 2.7.

1. Diagramas de casos de uso
  2. Diagramas de objetos.
  3. Diagramas de clase
  4. Diagramas de actividades
- Diagramas de interacción:
5. Diagrama de secuencia
  6. Diagrama de colaboración
  7. Diagramas de estado
- Diagramas de implantación:
8. Diagrama de componentes
  9. Diagrama de instalación
- Nota: El diagrama de objetos en realidad no se provee como un tipo de diagrama separado. En diagramas de secuencia, diagramas de colaboración y en diagramas de actividad se modelan objetos.

**Figura 2. 6** Diagramas empleados por UML

**Fuente:** (Booch, Rambaugh, & Jacobson, 1997)



**Figura 2. 7** Vistas de un software y sus respectivos diagramas UML

**Fuente:** (Booch, Rambaugh, & Jacobson, 1997)



### 2.8.3.1. DIAGRAMAS ESTRUCTURALES

Los cuatro diagramas estructurales de UML, tabla 2.10, existen para visualizar, especificar, construir y documentar los aspectos estáticos del sistema. Están organizados sobre grupos de objetos que se encontrarán cuando se esté modelando un sistema (Booch, Rumbaugh, & Jacobson, 1997).

**Tabla 2. 10 Diagramas Estructurales**

<b>Nombre del diagrama</b>	<b>Descripción</b>
<b>Diagrama de clases</b>	Un diagrama de este tipo muestra un conjunto de clases, interfaces, y sus relaciones.
<b>Diagrama de objetos</b>	Muestra un conjunto de objetos y sus relaciones. A diferencia de los diagramas anteriores, estos diagramas se enfocan en la perspectiva de casos de uso, y prototipos
<b>Diagrama de componentes</b>	Muestra el conjunto de componentes y sus relaciones y se utilizan para ilustrar la vista de la implementación estática de un sistema.
<b>Diagrama de implantación</b>	Muestra un conjunto de nodos y sus

	relaciones, se usan para ilustrar la vista de implantación estática de un sistema.
--	--

**Fuente:** (Booch, Rumbaugh, & Jacobson, 1997)

### 2.8.3.2. DIAGRAMAS DE COMPORTAMIENTO

Los cinco diagramas comportamiento de UML, ver tabla 2.11, son usados para visualizar, especificar, construir y documentar los aspectos dinámicos de un sistema. Se puede pensar en los aspectos dinámicos como las representaciones de las partes cambiantes del sistema (Booch, Rumbaugh, & Jacobson, 1997).

**Tabla 2. 11 Diagramas de Comportamiento**

<b>Nombre del diagrama</b>	<b>Descripción</b>
<b>Diagrama de casos de uso</b>	Muestra el conjunto de casos de uso y actores (incluyendo sus relaciones). Estos diagramas se utilizan para ilustrar la vista del caso de uso del sistema.
<b>Diagrama de secuencia</b>	Es un diagrama de interacción que enfatiza el orden en tiempo de los mensajes.
<b>Diagrama de colaboración</b>	Es un diagrama de interacción que enfatiza la organización estructural de los objetos que envían y reciben mensajes. El diagrama de colaboración muestra un

	conjunto de objetos, las ligas entre ellos y los mensajes enviados y recibidos por dichos objetos.
<b>Diagrama de estado</b>	Muestra una máquina de estado, consistente en estados, transiciones, eventos y actividades. Estos diagramas enfatizan el comportamiento.

**Fuente:** (Booch, Rumbaugh, & Jacobson, 1997)

## 2.9. SISTEMA DE INFORMACIÓN

En lo que respecta a los sistemas propiamente dichos hay un amplio consenso en cuanto a las características que deben tener y maneras de obrar, sin embargo, no ocurre lo mismo con el concepto de sistema de información, del cual existen muchas definiciones, matices y escuelas.

De todas formas, hablando en términos generales, podemos decir que un sistema de información es un conjunto de componentes que interaccionan entre sí para alcanzar un fin determinado, el cual es satisfacer las necesidades de información de dicha organización. Estos componentes pueden ser personas, datos, actividades o recursos materiales en general, los cuales procesan la información y la distribuyen de manera adecuada, buscando satisfacer las necesidades de la organización (Morales, 2015).

## 2.10. SEGURIDAD

La seguridad informática es una disciplina que se encarga de proteger la integridad y la privacidad de la información almacenada en un sistema informático. De todas formas, no existe ninguna técnica que permita asegurar la inviolabilidad de un sistema.

Un sistema informático puede ser protegido desde un punto de vista lógico (con el desarrollo de software) o físico (vinculado al mantenimiento eléctrico, por ejemplo). Por

otra parte, las amenazas pueden proceder desde programas dañinos que se instalan en la computadora del usuario (como un virus) o llegar por vía remota (los delincuentes que se conectan a Internet e ingresan a distintos sistemas) (Carrodegua, 2015).

### **2.10.1. INYECCIONES SQL**

Es un método de infiltración de código intruso que se vale de una vulnerabilidad informática presente en una aplicación en el nivel de validación de las entradas para realizar operaciones sobre una base de datos.

El origen de la vulnerabilidad radica en el incorrecto chequeo o filtrado de las variables utilizadas en un programa que contiene, o bien genera, código SQL. Es, de hecho, un error de una clase más general de vulnerabilidades que puede ocurrir en cualquier lenguaje de programación o script que esté embebido dentro de otro.

Se conoce como Inyección SQL, indistintamente, al tipo de vulnerabilidad, al método de infiltración, al hecho de incrustar código SQL intruso y a la porción de código incrustado (Carrodegua, 2015).

### **2.10.2. .HTACCESS**

La configuración de un servidor web que contiene comandos que le indican al servidor cómo actuar en determinadas circunstancias. El uso más común del htaccess es la restricción de determinados archivos y directorios de un servidor a través de protección con clave.

También es utilizado para re direccionar visitantes, banear o permitir determinadas direcciones IP para acceder al servidor, para llamar a una página personalizada para un error como el error 404 (página no encontrada) y para técnicas SEO. Por lo general la configuración se guarda en un archivo de texto llamado ".htaccess" y puede manipularse manualmente (Carrodegua, 2015).

### **2.10.3. CIFRADOS POR BLOQUE – BLOWFISH**

Blowfish es un algoritmo de cifrado por bloques simétrico libre de patente creado por Bruce Schneier en 1993 como una alternativa para reemplazar a DES como estándar de

cifrado. Este algoritmo está compuesto por 18 semiclaves (K) y 4 cajas (substitution boxes S). Es un proceso relativamente simple y altamente seguro ya que a la fecha no se conoce ningún tipo de criptoanálisis efectivo contra este algoritmo de cifrado (Naciek, 2013).



# CAPÍTULO III

## MARCO APLICATIVO

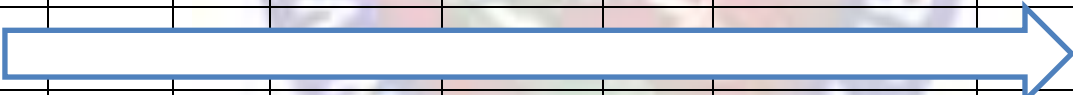


### 3.1 INTRODUCCIÓN

En este capítulo se detalla la forma de organización y métodos de trabajo del sistema, se hará uso de las metodologías y herramientas descritas anteriormente, las mismas que nos servirán para el desarrollo del sistema y todos sus módulos.

En la Tabla 3.1 se puede apreciar el modelo de proceso de desarrollo que se utilizó en el presente proyecto, en esta tabla se modela la combinación de la metodología SCRUMBAN, además de pasos que son importantes para la implementación del producto final.

**Tabla 3. 1. Tabla Scrumban**

HACER			SIGUIENTE	HACIENDO			POR APROBAR	SPRINT HECHO
Stories	Backlog	Sprint		Análisis(4)	Diseño (3)	Implementación(3)		
								

**Fuente:** Elaboración propia

### 3.2. FASE PREGAME

La fase inicial es la más importante, en Scrum el objetivo de la fase inicial denominada Pregame, es construir el Product-Backlog.

En el caso de la metodología Kanban, en la fase de exploración, se identifica a los usuarios involucrados en el desarrollo del producto, como también los requerimientos que presentan, las tareas que se realizarán y la planeación del desarrollo del producto.

### 3.2.1. CONCEPCIÓN Y EXPLORACIÓN

En esta fase, se realizaron 3 tareas de manera paulatina para la obtención de requisitos, los mismos se observan de manera resumida en la Tabla 3.2.

**Tabla 3. 2 Tareas realizadas para la obtención de requisitos**

Tarea	Descripción de la tarea
Entrevistas Personales	Se realizaron entrevistas con turistas que realizaban compras en tiendas, se mantuvieron entrevistas frecuentes con clientes de Bolivia Tech Hub.
Observación	Se observaron las dificultades con las que cuenta las personas que realizan compras de productos artesanales, así mismo la falta de un mecanismo en la cual se puedan apoyar para gestionar espacios públicos que puedan ser destinados a tiendas de productos artesanales.
Documentación	Se documentaron todas las observaciones y se generó un flujo de procesos que permita la compra y venta de productos artesanales.

**Fuente:** Elaboración propia

### 3.2.2. ROLES SCRUMBAN

La tabla 3.3 muestra las fases de trabajo detectados en el tablero Kanban, el límite WIP y los responsables que componen del equipo Kanban.

**Tabla 3. 3 Fases de trabajo, limite WIP y responsables**

FASE	WIP	RESPONSABLE
------	-----	-------------



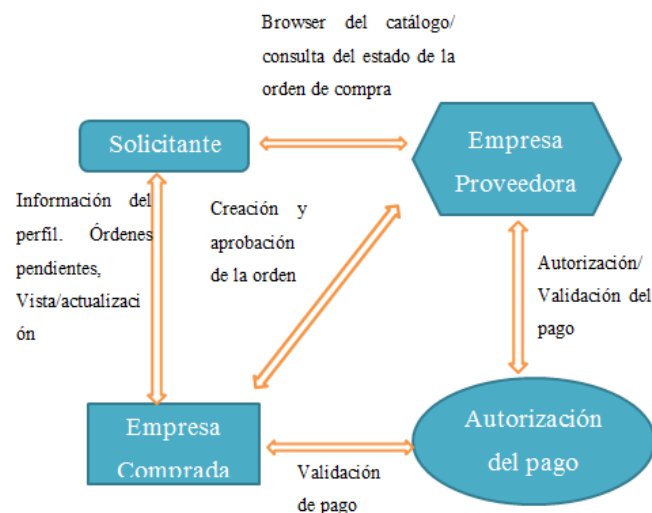
Pedido	-	<b>Flow Manager:</b> Karla Belen Limachi Machaca
Análisis	4	<b>Analista:</b> Karla Belen Limachi Machaca
Diseño	3	<b>Diseñador:</b> Karla Belen Limachi Machaca
Implementar	3	<b>Desarrollador:</b> Karla Belen Limachi Machaca
Producción	-	<b>Flow Manager:</b> Karla Belen Limachi Machaca

**Fuente:** Elaboración propia

### 3.2.3. ARQUITECTURA DE SOFTWARE

La idea central de la arquitectura de software es dividir la funcionalidad del sistema de comercio entre las actividades de venta y las de compra de tal forma que cada organización maneje aquellas funciones lógicamente conectadas a ella.

El servidor de Web proporciona tanto el catálogo de productos como la orden de pedido. Es decir, que el servidor del vendedor y el servidor de transacción están en un solo sistema, y no hay algún medio de pago explícito.



**Figura 3. 1 Esquema de arquitectura**

**Fuente:** Elaboración propia

En la figura 3.1 el usuario va seleccionando los artículos a través de un check box o agregando al carrito de compras. Cuando el usuario está listo hace click en el botón denominado “checkout”, el cual inicia el proceso de pago dentro de la transacción.

### 3.2.4. OBTENCIÓN DE REQUERIMIENTOS

A continuación, la tabla 3.4 muestra cada uno de los requerimientos que fueron descritos por el cliente y los futuros usuarios del sistema:

**Tabla 3. 4 Lista de requerimientos**

<b>Requerimientos</b>	<b>Descripción</b>
<b>Registrar información de artículos nuevos para la empresa</b>	El administrador debe registrar el ingreso de artículos nuevos para la empresa.
<b>Contacto directo con el vendedor</b>	El sistema debe proporcionar un contacto directo con el vendedor de la empresa, mediante la introducción de usuario y contraseña que se le fue asignado por el administrador luego ingresará al sistema.
<b>Administrar usuarios</b>	Todo sistema debe contar con este módulo, donde el usuario pueda administrar los ajustes básicos del sistema así como permitir la asignación de un rol a un usuario.
<b>Administración de clientes</b>	Crear, listar, modificar y eliminar registros de clientes.
<b>Clasificar los productos en categorías</b>	Crear, listar, modificar y eliminar registros de las categorías, y a la vez permitir adicionar uno o más categorías a cada producto.
<b>Registro de ventas</b>	Las ventas se deben registrar cada vez que

	se realiza una venta de productos.
<b>Registrar datos de proveedores</b>	Se debe realizar el registro de los datos personales de todos los proveedores que trabajan con la empresa.
<b>Registro de compras o abastecimiento relacionando a los proveedores</b>	<ul style="list-style-type: none"> <li>• Hacer el registro correspondiente de cada compra relacionando al proveedor.</li> <li>• Hacer el registro de la cantidad de cada producto, el cual debe ser adicionado al stock ya existente en el caso de que el producto sea nuevo, se deberá ir al registro de productos.</li> </ul>
<b>Entorno grafico amigable al usuario</b>	El usuario deberá adaptarse fácilmente al uso del sistema.

**Fuente:** Elaboración propia

### 3.2.5. PRODUCT-BACKLOG

A partir de la lista básica de requerimientos de la plataforma descrita en la Tabla 3.2, se realizó el análisis correspondiente y en coordinación con el personal de Bolivia Tech Hub se construyó el Product-Backlog como se observa en la Tabla 3.5.

**Tabla 3. 5 Product-Backlog**

<b>ID</b>	<b>Nombre</b>	<b>Prioridad</b>	<b>Modulo</b>
R1	Diseño de Registro, login y logout	Media	Módulo de registro y login de usuario
R2	Creación de controlador y desarrollo de CRUD	Alta	Módulo de registro y login de usuario
R3	Autenticación local	Alta	Módulo de registro y login de usuario

R4	Asignación de roles	Alta	Módulo de administración de cuentas de usuario
R5	Recuperación de contraseña	Media	Módulo de administración de cuentas de usuario
R6	Validar rol	Media	Módulo de administración de cuentas de usuario
R7	Validar credenciales	Media	Módulo de administración de cuentas de usuario
R8	autenticación con red social	Baja	Módulo de registro y login de usuario
R9	Creación de controlador y desarrollo de CRUD	Alta	*Módulo de administración de cuentas de usuario
R10	Creación de servicios	Alta	Módulo de administración de cuentas de usuario
R11	Diseño del carrito de compra	Alta	Módulo de carrito de compras
R12	Adicionar productos al carrito	Alta	Módulo de carrito de compras
R13	Listar línea de compra del carrito	Media	Módulo de carrito de compras
R14	Eliminar productos del carrito	Alta	Módulo de carrito de compras
R15	Comprar producto	Alta	Módulo de carrito de compras
R16	Generar recibo	Alta	Módulo de carrito de compras
R17	Añadir catálogo	Media	Módulo de administración de catálogo y productos

R18	Eliminar catálogo	Media	Módulo de administración de catálogo y productos
R19	Buscar catálogo	Media	Módulo de administración de catálogo y productos
R20	Ver artículo en detalle	Media	Módulo de administración de catálogo y productos
R21	Ver novedades y ofertas	Media	Módulo de administración de catálogo y productos
R22	CRUD de productos	Alta	Módulo de administración de catálogo y productos

**Fuente:** Elaboración propia

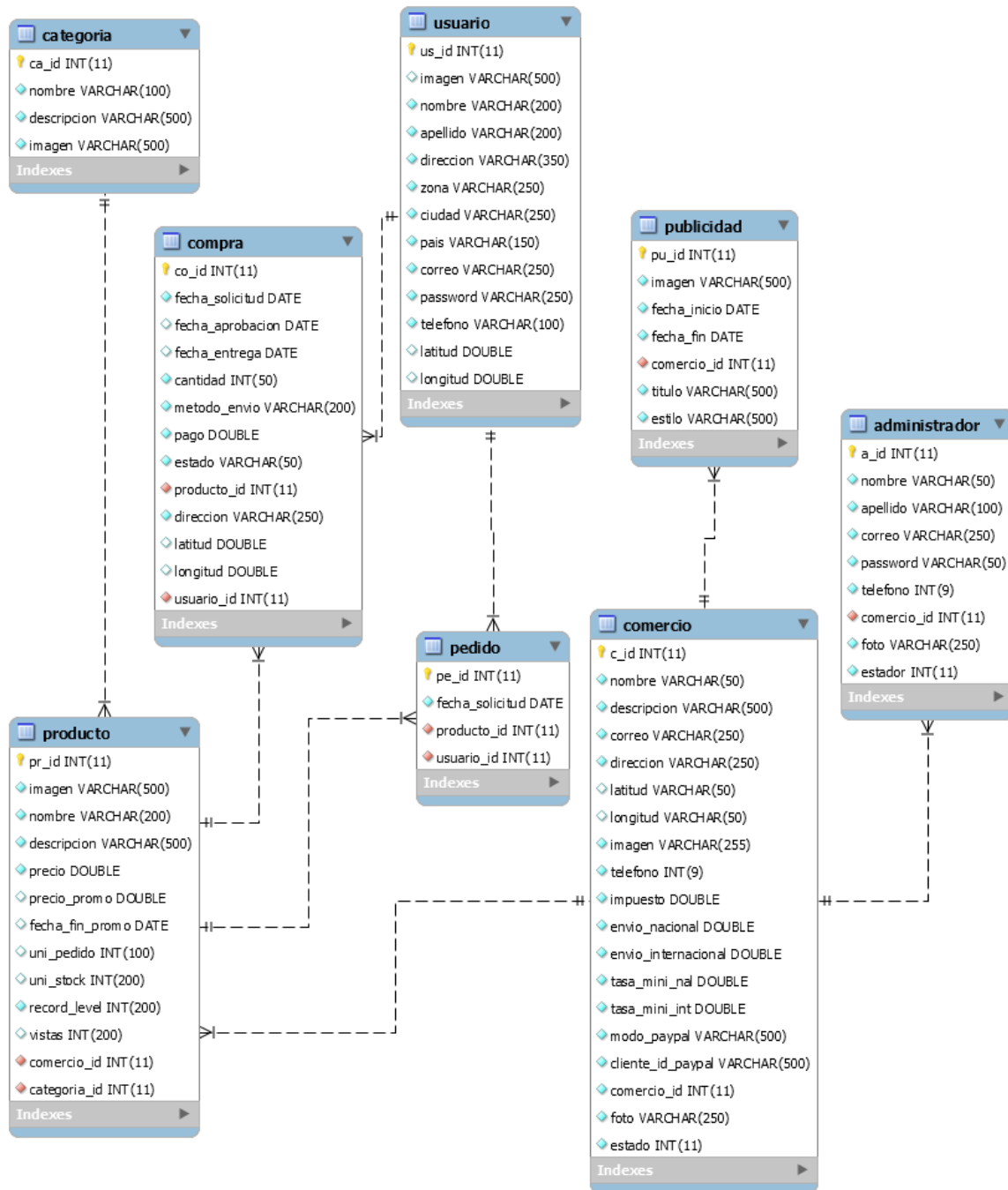
### 3.2.6. INICIALIZACIÓN

Esta fase será adoptada dentro de la fase Pre-Game para el desarrollo del sistema. Se considera dentro de la metodología Kanban, y es en la que se preparan y verifican los asuntos críticos que involucran el desarrollo del sistema como también la configuración para el proyecto.

Ya definido el Product-Backlog como también la arquitectura de la plataforma, se procede al modelado de la base de datos, figura 3.2.

Después de diseñar el Modelo Relacional se realizó su diccionario de datos: la tabla 3.6 muestra el diccionario de datos de la tabla administración, la tabla 3.7 muestra el diccionario de datos de la tabla categorías, la tabla 3.8 corresponde a la tabla de comercio,

la tabla 3.9 muestra el diccionario de datos de la tabla compra, la tabla 3.10 contiene el diccionario de datos de la tabla pedido, las tablas 3.11, 3.12 y 3.13 corresponden a las tablas productos, publicidad y usuarios.



### Figura 3. 2 Base de datos relacional

Fuente: Elaboración propia

**Tabla 3. 6 Diccionario de datos de la tabla de administrador**

Columna	Tipo	Nulo	Descripción
a_id (Primaria)	int(11)	No	clave primaria del administrador
nombre	varchar(50)	No	nombre del administrador
apellido	varchar(100)	No	apellido del administrador
correo	varchar(250)	No	correo del administrador
password	varchar(50)	No	password del administrador
teléfono	int(9)	No	teléfono móvil del administrador
comercio_id	int(11)	No	clave foránea de la tabla comercio
foto	varchar(250)	No	foto del administrador
estado	int(11)	No	estado del administrador

Fuente: Elaboración propia

**Tabla 3. 7 Diccionario de datos de la tabla de categorías**

Columna	Tipo	Nulo	Comentarios
ca_id (Primaria)	int(11)	No	clave primaria de categoría

nombre	varchar(100)	No	nombre de categoría
descripción	varchar(500)	No	descripción de categoría
imagen	varchar(500)	No	imagen de la categoría

**Fuente:** Elaboración propia

**Tabla 3. 8 Diccionario de datos tabla de comercio**

Columna	Tipo	Nulo	Comentarios
c_id ( <i>Primaria</i> )	int(11)	No	clave primaria del comercio
nombre	varchar(50)	No	nombre del comercio
descripción	varchar(500)	No	descripción del comercio
correo	varchar(250)	No	correo del comercio
Dirección	varchar(250)	No	dirección del comercio
Latitud	varchar(50)	Sí	latitud del comercio
Longitud	varchar(50)	Sí	longitud del comercio
Imagen	varchar(255)	No	imagen del comercio
Teléfono	int(9)	No	teléfono o móvil del comercio
Impuesto	double	No	impuesto del comercio



envió_nacional	double	No	precio envió comercio
envió_internacional	double	No	envió internacional comercio
tasa_mini_nal	double	No	tasa mínima nacional
tasa_mini_int	double	No	tasa mínima internacional
modo_paypal	varchar(500)	No	modo de pago paypal
cliente_id_paypal	varchar(500)	No	cliente id paypal
comercio_id	int(11)	No	clave foránea de la tabla comercio
Foto	varchar(250)	No	foto del comercio
Estado	int(11)	No	estado del comercio

**Fuente:** Elaboración propia

**Tabla 3. 9 Diccionario de datos de la tabla de compra**

<b>Columna</b>	<b>Tipo</b>	<b>Nulo</b>	<b>Comentarios</b>
co_id ( <i>Primaria</i> )	int(11)	No	clave primaria de la compra
fecha_solicitud	date	No	fecha solicitud de la compra
fecha_aprobación	date	Sí	fecha aprobación de la compra

fecha_entrega	Date	Sí	fecha entrega de la compra
cantidad	int(50)	No	cantidad por unidad de la compra
método_envío	varchar(200)	No	método de envío de la compra
Pago	double	No	pago de compra
Estado	varchar(50)	No	estado de la compra
producto_id	int(11)	No	clave foránea referencia la tabla producto
Dirección	varchar(250)	No	dirección de envío
Latitud	double	Sí	latitud de envío de la compra
Longitud	double	Sí	longitud de envío de la compra
usuario_id	int(11)	No	clave foránea referencia la tabla usuario

**Fuente:** Elaboración propia

**Tabla 3. 10 Diccionario de datos de la tabla de pedido**

Columna	Tipo	Nulo	Comentarios
pe_id ( <i>Primaria</i> )	int(11)	No	clave primaria del pedido
fecha_solicitud	date	No	fecha solicitud del pedido

producto_id	int(11)	No	clave foránea referencia la tabla producto
usuario_id	int(11)	No	clave foránea referencia la tabla usuario

Fuente: Elaboración propia

**Tabla 3. 11 Diccionario de datos de la tabla de productos**

Columna	Tipo	Nulo	Comentarios
pr_id ( <i>Primaria</i> )	int(11)	No	clave primaria del producto
imagen	varchar(500)	No	imagen del producto
nombre	varchar(200)	No	nombre del producto
descripción	varchar(500)	No	descripción del producto
Precio	Double	No	precio del producto
precio_promo	Double	Sí	precio promoción del producto
fecha_fin_promo	Date	Sí	fecha fin de la promoción del producto
uni_pedido	int(100)	Sí	unidades en pedido del producto
uni_stock	int(200)	Sí	unidades en stock del

			producto
record_level	int(200)	No	record level de ventas del producto
Vistas	int(200)	Sí	vistas que tiene el producto
comercio_id	int(11)	No	clave foránea referencia la tabla comercio
categoria_id	int(11)	No	clave foránea referencia la tabla categoría

**Fuente:** Elaboración propia

**Tabla 3. 12 Diccionario de datos de la tabla de publicidad**

Columna	Tipo	Nulo	Comentarios
pu_id (PK)	int(11)	No	clave primaria de la publicidad
imagen	varchar(500)	No	imagen de la publicidad
fecha_inicio	Date	No	fecha inicio de la publicidad
fecha_fin	Date	No	fecha fin de la publicidad
comercio_id	int(11)	No	clave foránea referencia la tabla comercio
Titulo	varchar(500)	No	título de la publicidad
estilo	varchar(500)	No	estilo de la publicidad

Fuente: Elaboración propia

**Tabla 3. 13 Diccionario de datos de la tabla de usuarios**

Columna	Tipo	Nulo	Comentarios
us_id ( <i>Primaria</i> )	int(11)	No	clave primaria del usuario
imagen	varchar(500)	Sí	imagen del usuario
nombre	varchar(200)	No	nombre del usuario
apellido	varchar(200)	No	descripcion del usuario
direccion	varchar(350)	No	direccion del usuario
zona	varchar(250)	No	zona del usuario
ciudad	varchar(250)	No	ciudad del usuario
pais	varchar(150)	No	pais del usuario
correo	varchar(250)	No	correo del usuario
password	varchar(250)	No	contraseña del usuario
telefono	varchar(100)	No	telefono o movil del usuario
latitud	Double	Sí	latitud del usuario
longitud	Double	Sí	longitud del usuario

Fuente: Elaboración propia

Antes de iniciar el desarrollo, se configuraron los ambientes de trabajo de los aplicativos a desarrollar, en la Tabla 3.7, se puede observar la base de configuración del aplicativo web, como también las versiones que fueron utilizadas para el desarrollo.

**Tabla 3. 14 Configuración del sistema**

<b>Nro.</b>	<b>Configuración</b>	<b>Versión</b>
1	PHP	7.2.5
2	Laravel Framework	5.6
3	Laragon	2.2
3	JSON Web Token	0.5
4	Bootstrap	4.0.0
5	JQuery	3.2.1

**Fuente:** Elaboración propia

### **3.3. FASE DEVELOPMENT**

En esta fase se describe el desarrollo de la aplicación web, donde se describe el desarrollo de cada uno de los Sprints.

#### **3.3.1. DESARROLLO DE LOS SPRINTS**

El proceso de desarrollo se sujetara a la presentación de 4 Sprints, que son los siguientes:

- Primer Spring: Módulo de registro y login de usuarios
- Segundo Spring: Módulo de administración de cuentas de usuario
- Tercer Spring: Módulo de carrito de compras
- Cuarto Spring: Módulo de administración de catálogo, productos y proveedores

### 3.4. PRIMER SPRING: MÓDULO DE REGISTRO Y LOGIN DE USUARIOS

#### 3.4.1. ETAPA DE ANÁLISIS

La tabla 3.15 muestra el sprint asignado para este módulo, que contiene tareas.

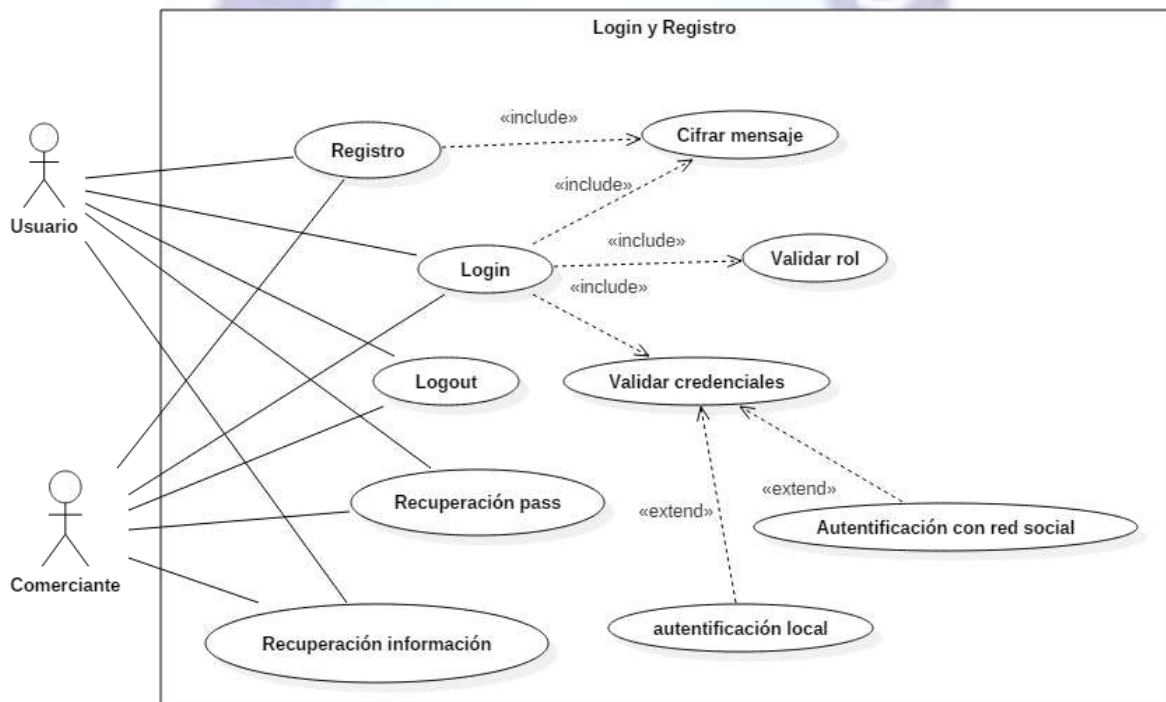
**Tabla 3. 15 Primer Sprint: módulo de registro y login de usuarios**

Sprint o Iteración: 1		Inicio	Fin	Duración
		07/03/2018	12/03/2018	5 días
Nro	Tarea	Duración tarea		Estado
1	Interfaz y lógica de registro de usuario	3 horas		Completado
2	Interfaz y lógica de iniciar de sesión	4 horas		Completado
3	Interfaz y lógica de cerrar sesión	2 horas		Completado
4	Implementación de recuperación de contraseña	5 horas		Completado
5	Implementación y desarrollo de recuperación de información	3 horas		Completado
6	autenticación con red social	3 días		Completado
7	Autenticación local	1 día		Completado

**Fuente:** Elaboración propia

### 3.4.1.1. DIAGRAMA DE CASO DE USO

Se diseñó el diagrama de casos de uso para los requisitos: interfaz y lógica de registro de usuario, interfaz y lógica de iniciar de sesión, interfaz y lógica de cerrar sesión, implementación de recuperación de contraseña, implementación y desarrollo de recuperación de información, autenticación con red social, autenticación local, el cual se lo visualiza en la figura 3.3, y en la tabla 3.16 se muestra las especificaciones del caso de uso.



**Figura 3. 3 Diagrama de caso de uso de inicio de sesión y registro**

Fuente: Elaboración propia

**Tabla 3. 16 Especificación del caso de uso de inicio de sesión y registro**

<b>Caso de uso</b>	caso de uso de inicio de sesión y registro
--------------------	--



<b>Actores</b>	Usuario y comerciante
<b>Descripción</b>	Permite al usuario y comerciante crear una cuenta en el sistema, ya sea por una red social o una cuenta local.
<b>Precondiciones</b>	Los actores deben contar con conexión a internet.
<b>Flujo Normal</b>	<ol style="list-style-type: none"> <li>1. Los actores ingresan a la página principal.</li> <li>2. El sistema despliega un modal de inicio de sesión.</li> <li>3. Se tiene opción de registrarse si no se tiene una cuenta, las opciones de registro son redes sociales y cuenta local.</li> <li>4. El actor escoge una de las opciones, si no contaba con una cuenta.</li> </ol>
<b>Postcondiciones</b>	El sistema ejecuta la opción elegida por el actor y muestra los cambios realizados.

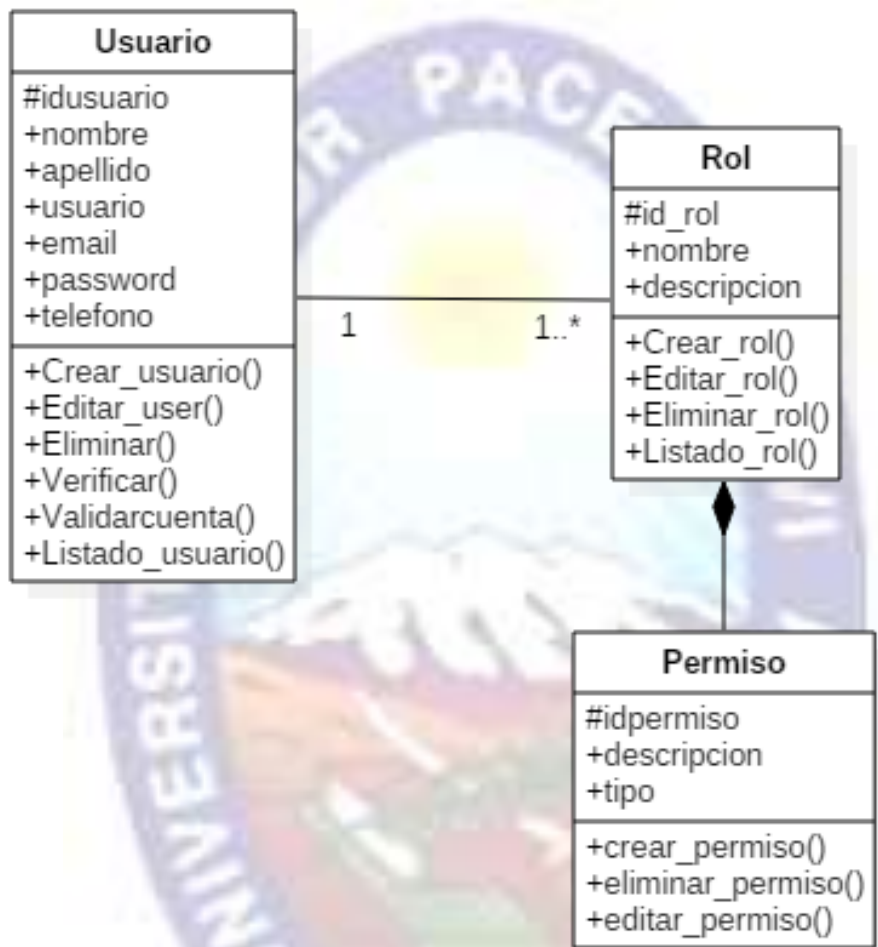
**Fuente:** Elaboración propia

### 3.4.2. ETAPA DE DISEÑO

En la fase de diseño se utilizó los diagramas de clases y de estados.

#### 3.4.2.1. DIAGRAMA DE CLASES

Se diseñó el diagrama de clases para los requisitos: interfaz y lógica de registro de usuario, interfaz y lógica de iniciar de sesión, interfaz y lógica de cerrar sesión, implementación de recuperación de contraseña, implementación y desarrollo de recuperación de información, autenticación con red social, autenticación local, el cual se lo visualiza en la figura 3.4.

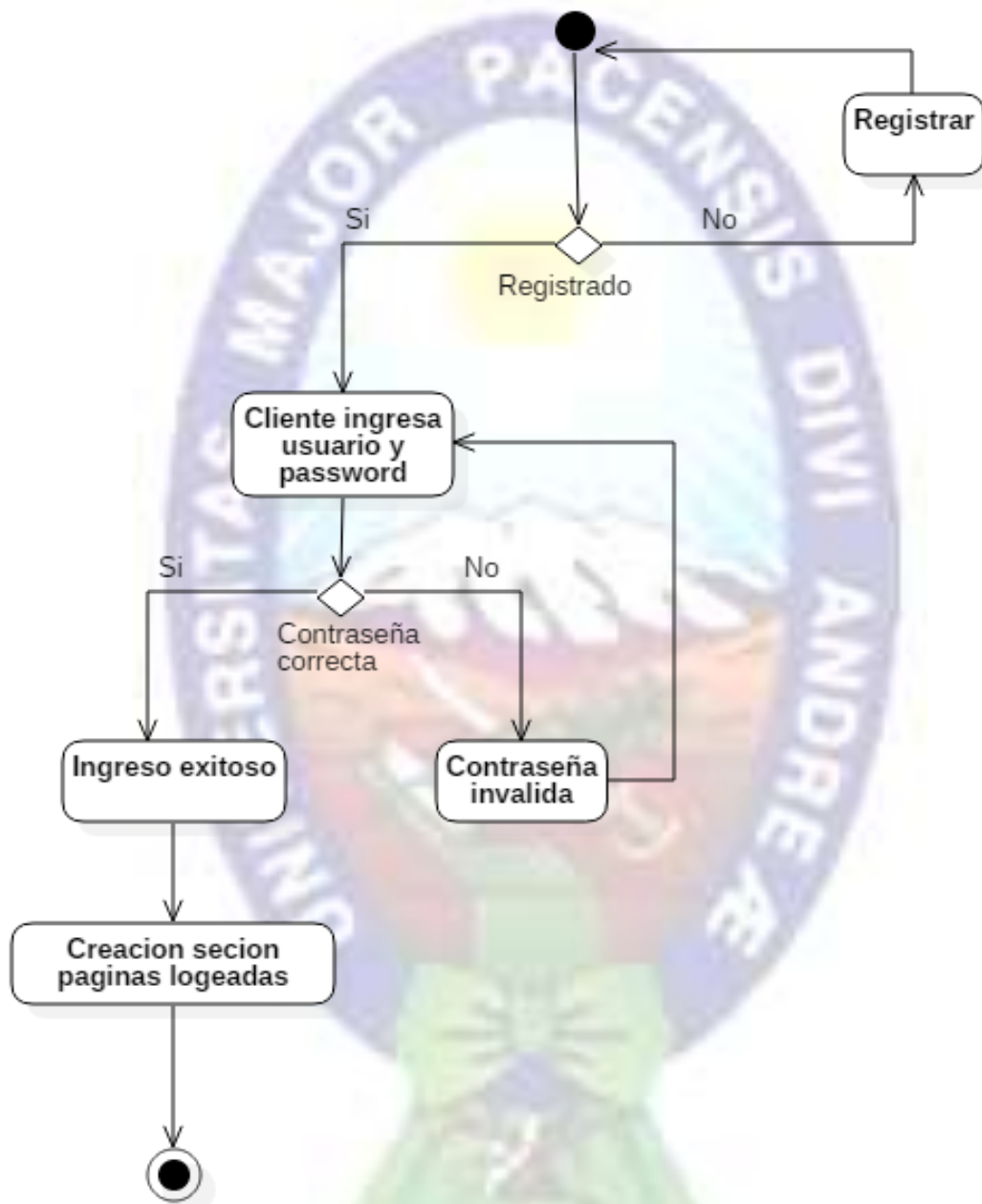


**Figura 3. 4. Diagrama de clases de inicio de sesión y registro**

**Fuente:** Elaboración propia

### 3.4.2.2. DIAGRAMA DE ESTADO

Se diseñó el diagrama de Estado para los requisitos: interfaz y lógica de registro de usuario, interfaz y lógica de iniciar de sesión, interfaz y lógica de cerrar sesión, implementación de recuperación de contraseña, implementación y desarrollo de recuperación de información, autenticación con red social, autenticación local, el cual se lo visualiza en la figura 3.5.



**Figura 3. 5 Diagrama de estados de inicio de sesión y registro**

**Fuente:** Elaboración propia

### 3.4.3. ETAPA DE IMPLEMENTACIÓN

En la fase de diseño se utilizó los diagramas de secuencia y de componentes.

### 3.4.3.1. DIAGRAMA DE SECUENCIA

Se diseñó el diagrama de Secuencia para los requisitos login de usuario, registro de usuarios tanto proveedores como clientes y Asignación de roles, el cual se lo visualiza en la figura 3.6.

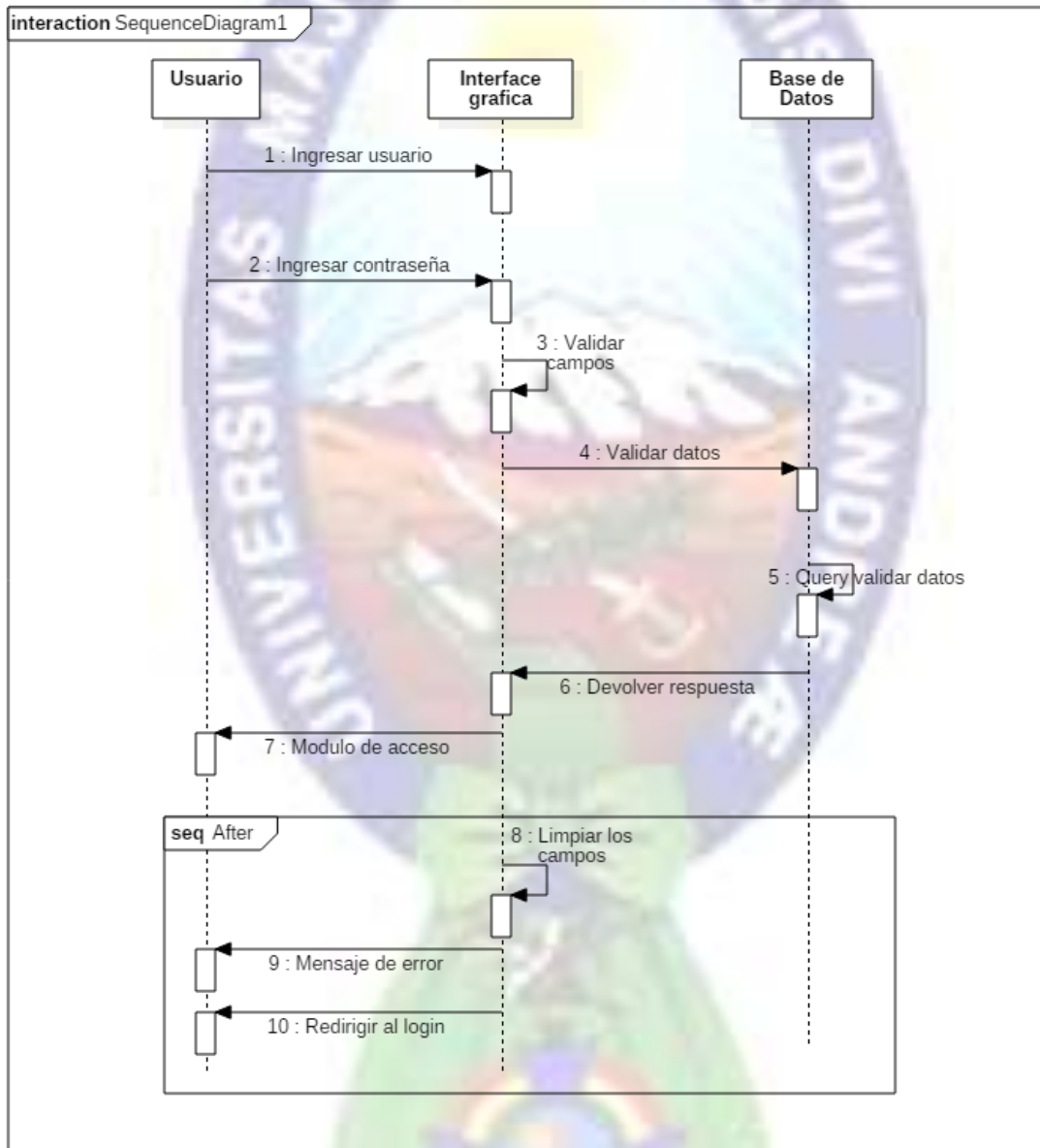


Figura 3. 6 Diagrama de secuencia de inicio de sesión y registro

Fuente: Elaboración propia

### 3.4.3.2. DIAGRAMA DE COMPONENTES

Se diseñó el diagrama de Componentes para los requisitos login de usuario, registro de usuarios tanto proveedores como clientes y Asignación de roles, el cual se lo visualiza en la figura 3.7.



**Figura 3. 7. Diagrama de componentes de inicio de sesión y registro**

**Fuente:** Elaboración propia

### 3.4.4. RESULTADOS

Para esta parte se desarrolló el inicio de sesión y registró como se puede observar la captura de:

- Modal de inicio de sesión en la figura 3.8
- El modal de solicitud de nueva contraseña figura 3.9
- El modal de registro con redes sociales o registro de cuenta local en la figura 3.9 que muestra una captura de pantalla.

The screenshot shows a mobile application interface for logging in. At the top, there is a teal header with the word "INGRESAR" in white. Below the header, there are two buttons: "Ingreso con Facebook" (blue) and "Ingreso con Google" (red). Underneath these are two input fields: "Correo Electrónico" (Email) and "Contraseña" (Password). A CAPTCHA section follows, with a checkbox labeled "No soy un robot" and a "reCAPTCHA" logo. At the bottom of the form is a large teal button labeled "ENVIAR" (SEND). Below the button, there are two links: "¿Ovidaste tu contraseña?" (Forgot your password?) and "¿No tienes una cuenta registrada? | Registrarse" (Don't you have an account? | Register).

**Figura 3. 8. Entorno gráfico de inicio de sesión**

**Fuente:** Elaboración propia

The screenshot shows a mobile application interface for requesting a new password. At the top, there is a teal header with the text "SOLICITUD DE NUEVA CONTRASEÑA" in white. Below the header is a single input field labeled "Correo Electrónico" (Email). Underneath the input field is a large teal button labeled "ENVIAR" (SEND). At the bottom of the form, there is a link: "¿Ya tienes una cuenta Cuenta? | Ingresar" (Do you already have an account? | Log in).

**Figura 3. 9 Entorno gráfico de solicitud de nueva contraseña**

**Fuente:** Elaboración propia

**REGISTRARSE** ✕

Registro con Facebook      Registro con Google

NOMBRE COMPLETO

Correo Electrónico

Contraseña

Al registrarse, usted acepta nuestras condiciones de uso y políticas de privacidad  
 Leer más

**ENVIAR**

[¿Ya tienes una cuenta registrada? | Ingresar](#)

**Figura 3. 10 Entorno gráfico de registro**

Fuente: Elaboración propia

### 3.5. SEGUNDO SPRING: MÓDULO DE ADMINISTRACIÓN DE CUENTAS DE USUARIOS

#### 3.5.1. ETAPA DE ANÁLISIS

La tabla 3.17 muestra el sprint asignado para este módulo, que contiene tareas.

**Tabla 3. 17 Segundo sprint módulo de administración de usuario**

Sprint o Iteración: 2		Inicio	Fin	Duración
		15/03/2018	30/03/2018	16 días
Nro	Tarea	Duración tarea		Estado

1	Validar el registro de usuario directo – lado del cliente	2 días	Completado
2	Validar el registro de usuario directo – lado del servidor	2 horas	Completado
3	Solicitar confirmación de email con PHPMailer	3 horas	Completado
4	Configuración del servidor local para envío de correos electrónicos	2 días	Completado
5	Validar email existente con AJAX	4 días	Completado
6	CRUD de usuarios	3 días	Completado

**Fuente:** Elaboración propia

### 3.5.1.1. DIAGRAMA DE CASOS DE USO

Se diseñó el diagrama de casos de uso para los requisitos: validar el registro de usuario directo – lado del cliente, validar el registro de usuario directo – lado del servidor, solicitar confirmación de email con PHPMailer, configuración del servidor local para envío de correos electrónicos, validar email existente con AJAX, el cual se lo visualiza en la figura 3.11.





**Figura 3. 11. Diagrama de casos de uso del módulo de administración de usuarios**

**Fuente:** Elaboración propia

**Tabla 3. 18 Especificación del caso de uso de administración de usuarios**

<b>Caso de uso</b>	caso de uso del módulo de administración de usuarios
<b>Actores</b>	Administración de usuario y usuario
<b>Descripción</b>	Administra, modificación, eliminación y el listado de los registros de los usuarios.
<b>Precondiciones</b>	El administrativo debe introducir los datos del estudiante, previamente verificados según los requisitos solicitados por la institución.

<b>Postcondiciones</b>	El sistema ejecuta la opción elegida por el administrador y se muestran los cambios realizados.
------------------------	---

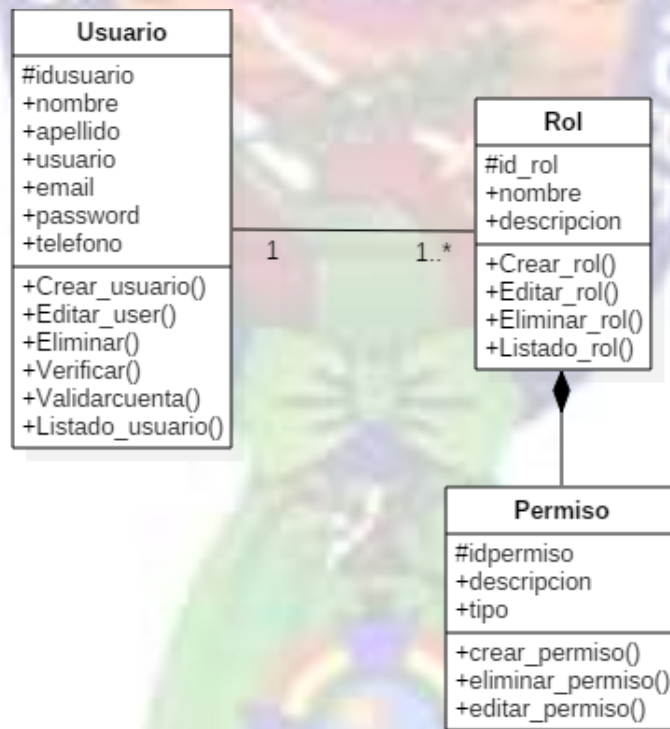
**Fuente:** Elaboración propia

### 3.5.2. ETAPA DE DISEÑO

En la fase de diseño se utilizó los diagramas de clases y de estados.

#### 3.5.2.1. DIAGRAMA DE CLASES

Se diseñó el diagrama de clases para los requisitos: validar el registro de usuario directo – lado del cliente y lado del servidor, solicitar confirmación de email con PHPMailer, configuración del servidor local para envío de correos electrónicos, validar email existente con AJAX, el cual se lo visualiza en la figura 3.12.



**Figura 3. 12 diagrama de clases del módulo de administración de usuarios**

**Fuente:** Elaboración propia

### 3.5.2.2. DIAGRAMA DE ESTADOS

Se diseñó el diagrama de estados para los requisitos: validar el registro de usuario directo – lado del cliente, validar el registro de usuario directo – lado del servidor, solicitar confirmación de email con PHPMailer, configuración del servidor local para envío de correos electrónicos, validar email existente con AJAX, el cual se lo visualiza en la figura 3.13.

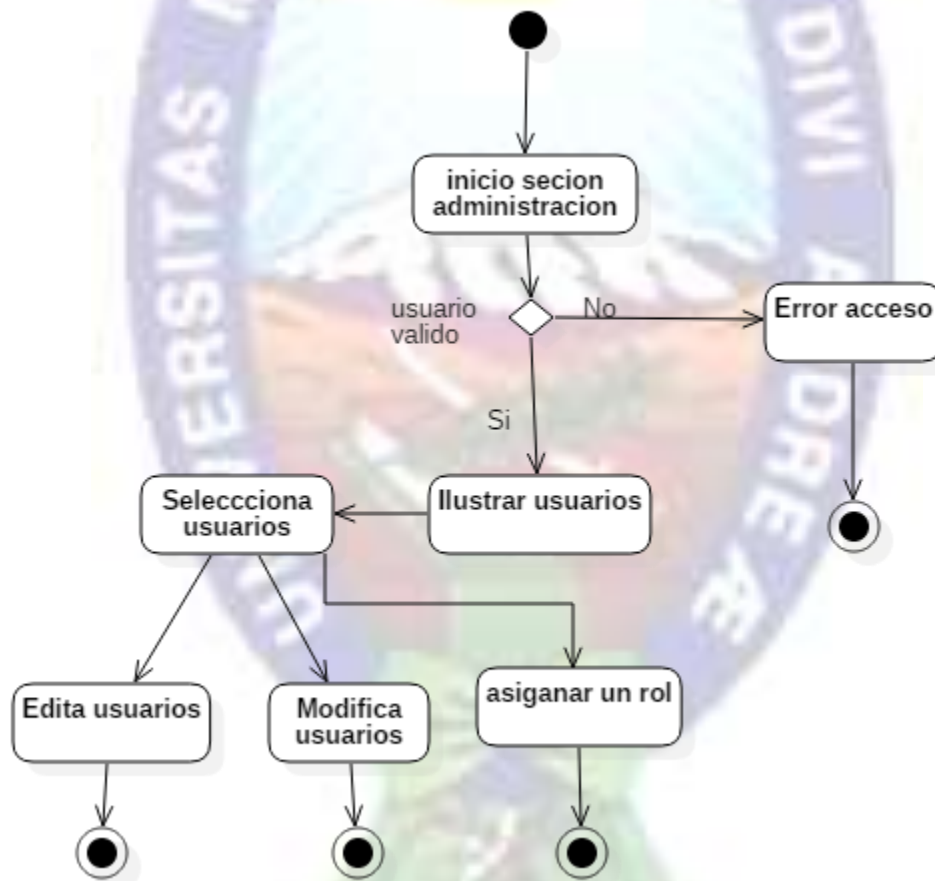


Figura 3. 13 Diagrama de estados del módulo de administración de usuarios

Fuente: Elaboración propia

### 3.5.3. ETAPA DE IMPLEMENTACIÓN

En la fase de diseño se utilizó los diagramas de secuencia y de componentes.

### 3.5.3.1. DIAGRAMA DE SECUENCIA

Se diseñó el diagrama de secuencia para los requisitos: validar el registro de usuario directo – lado del cliente, validar el registro de usuario directo – lado del servidor, solicitar confirmación de email con PHPMailer, configuración del servidor local para envío de correos electrónicos, validar email existente con AJAX, el cual se lo visualiza en la figura 3.14.

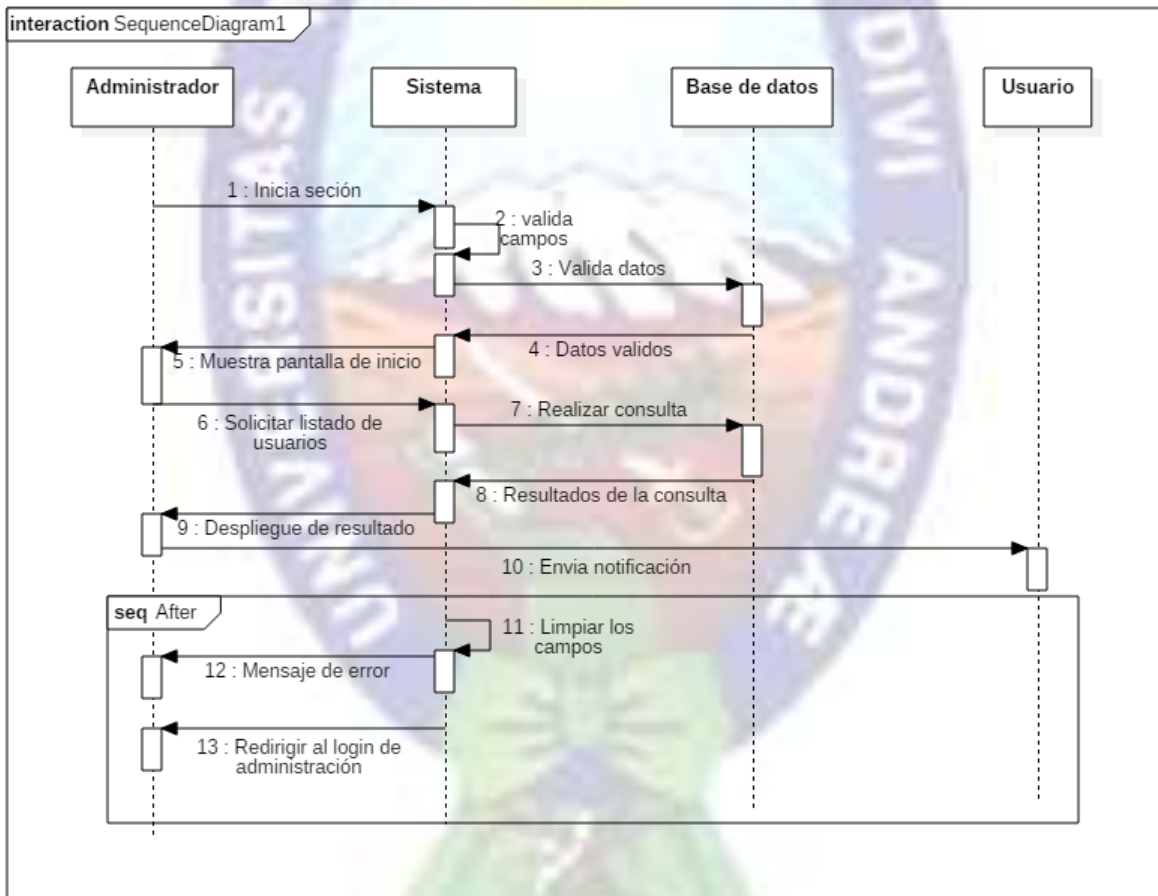
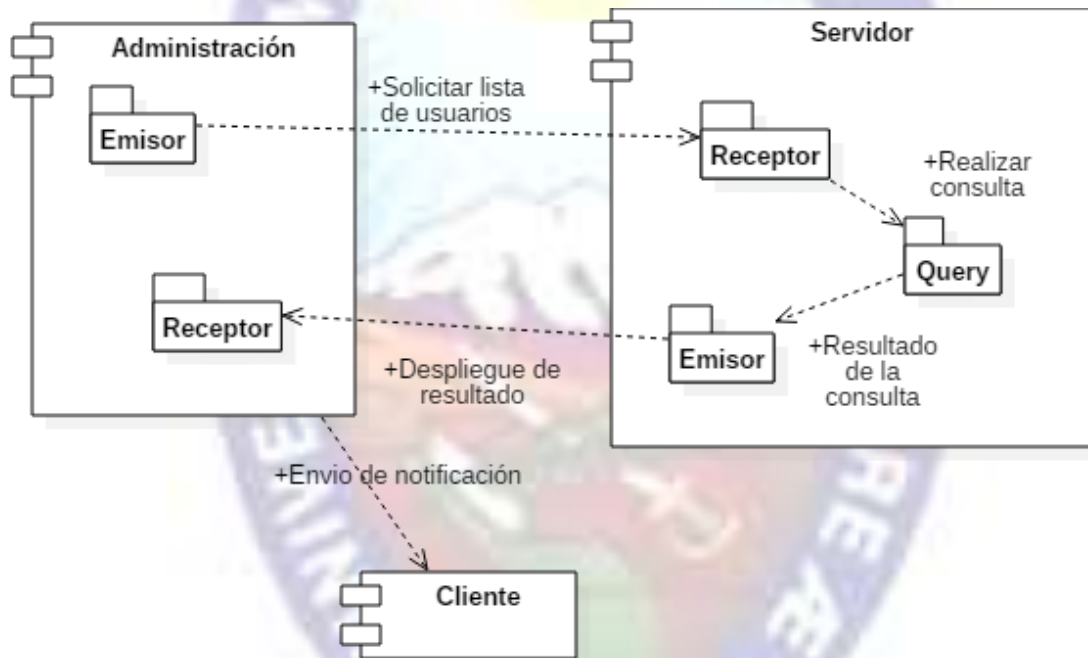


Figura 3. 14 Diagrama de secuencia del módulo de administración de usuarios

Fuente: Elaboración propia

### 3.5.3.2. DIAGRAMA DE COMPONENTES

Se diseñó el diagrama de componentes para los requisitos: validar el registro de usuario directo – lado del cliente, validar el registro de usuario directo – lado del servidor, solicitar confirmación de email con PHPMailer, configuración del servidor local para envío de correos electrónicos, validar email existente con AJAX,, el cual se lo visualiza en la figura 3.15.

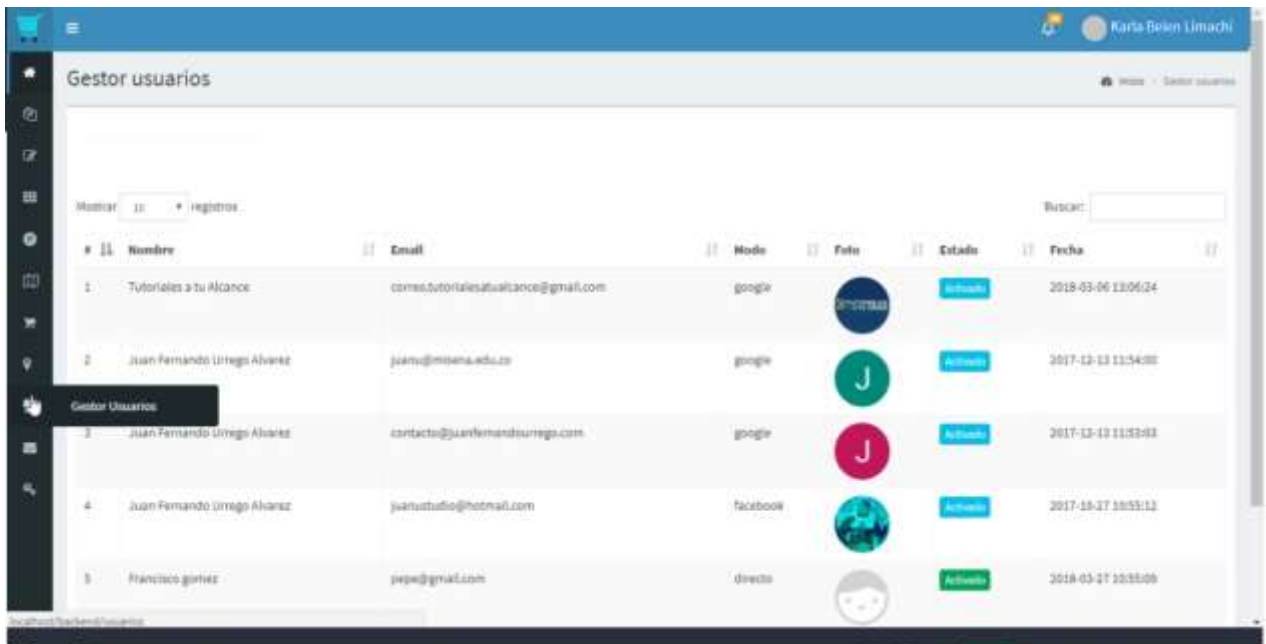


**Figura 3. 15. Diagrama de componentes del módulo de administración de usuarios**

**Fuente:** Elaboración propia

### 3.5.4. RESULTADOS

Para esta parte se desarrolló el módulo de administración de usuarios, se puede observar la figura 3.16 que muestra una captura de pantalla la plataforma administrativa.



**Figura 3. 16** Plataforma de administración

Fuente: Elaboración propia

### 3.6. TERCER SPRING: MÓDULO DE CARRITO DE COMPRAS

#### 3.6.1. ETAPA DE ANÁLISIS

La tabla 3.19 muestra el sprint asignado para este módulo, que contiene tareas.

**Tabla 3. 19** Tercer sprint módulo de carrito de compras

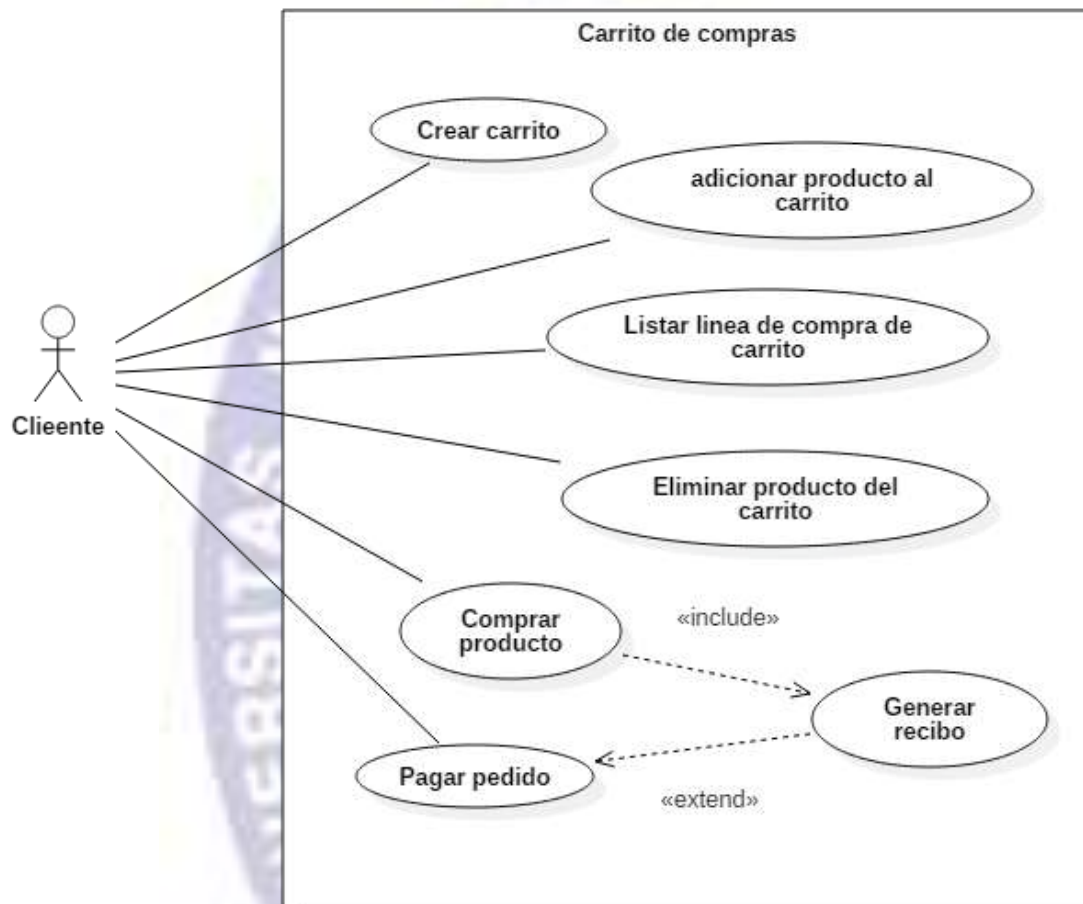
Sprint o Iteración: 3		Inicio	Fin	Duración
		26/04/2018	29/05/2018	32 días
Nro	Tarea	Duración tarea		Estado
1	Maquetación del carrito de compra	3 días		Completado
2	Agregando los productos físicos al locaStorage	1 día		Completado

3	Hacer dinámico el carrito de compras	2 días	Completado
4	Borrar productos del carrito de compras	2 días	Completado
5	Actualizar precio subtotal	1 día	Completado
6	Actualizar la cesta cuando cambia la cantidad	3 días	Completado
7	Agregar estilos al checkout	3 días	Completado
8	Evaluar envío y tasa de impuesto	4 días	Completado
9	Evaluar el total de la compra	1 día	Completado
10	Pasarela de pagos paypal	6 días	Completado
11	Aprobación de compra	2 días	Completado
12	Creación de ítem comentarios después de compra	3 días	Completado

**Fuente:** Elaboración propia

### 3.6.1.1. DIAGRAMA DE CASOS DE USO

Se diseñó el diagrama de casos de uso para los requisitos: maquetación del carrito de compra, agregando los productos físicos al localStorage, Hacer dinámico el carrito de compras, comprar producto, generar recibo y adicionar productos al carrito, el cual se lo visualiza en la figura 3.17.



**Figura 3. 17 Diagrama de casos de uso del módulo de carrito de compras**

**Fuente:** Elaboración propia

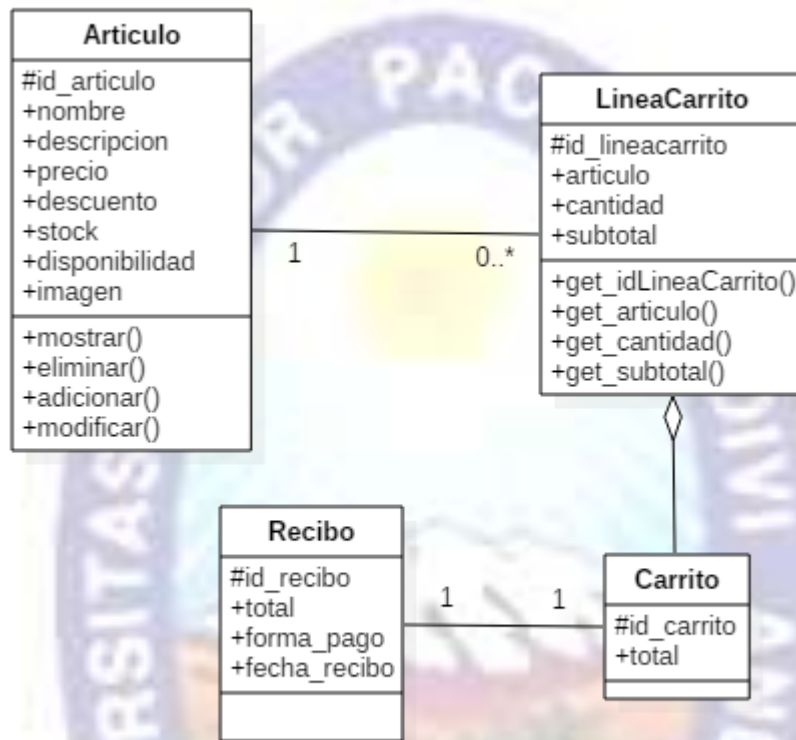
### **3.6.2. ETAPA DE DISEÑO**

En la fase de diseño se utilizó los diagramas de clases y de estados.

#### **3.6.2.1. DIAGRAMA DE CLASES**

Se diseñó el diagrama de clases para los requisitos: maquetación del carrito de compra, agregando los productos físicos al localStorage, Hacer dinámico el carrito de compras, comprar producto, generar recibo y adicionar productos al carrito, el cual se lo visualiza en la figura 3.18.





**Figura 3. 18 Diagrama de clases del módulo de carrito de compras**

**Fuente:** Elaboración propia

### 3.6.2.2. DIAGRAMA DE ESTADO

Se diseñó el diagrama de estado para los requisitos: maquetación del carrito de compra, agregando los productos físicos al localStorage, Hacer dinámico el carrito de compras, comprar producto, generar recibo y adicionar productos al carrito, el cual se lo visualiza en la figura 3.19.



**Figura 3. 19 Diagrama de estado del módulo de carrito de compras**

Fuente: Elaboración propia

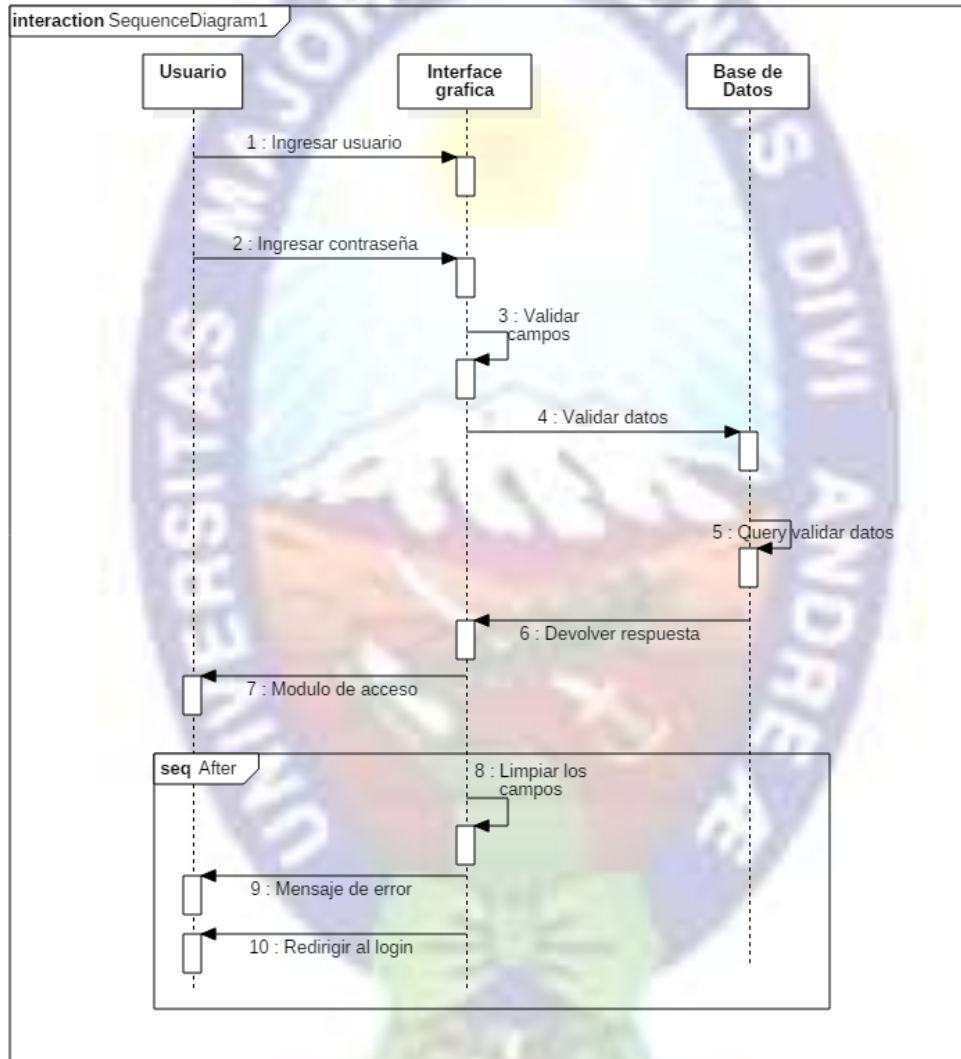
### 3.6.3. ETAPA DE IMPLEMENTACIÓN

En la fase de diseño se utilizó los diagramas de secuencia y de componentes.

#### 3.6.3.1. DIAGRAMA DE SECUENCIA

Se diseñó el diagrama de secuencia para los requisitos: maquetación del carrito de compra, agregando los productos físicos al locaStorage, Hacer dinámico el carrito de compras,

comprar producto, generar recibo y adicionar productos al carrito, el cual se lo visualiza en la figura 3.20.

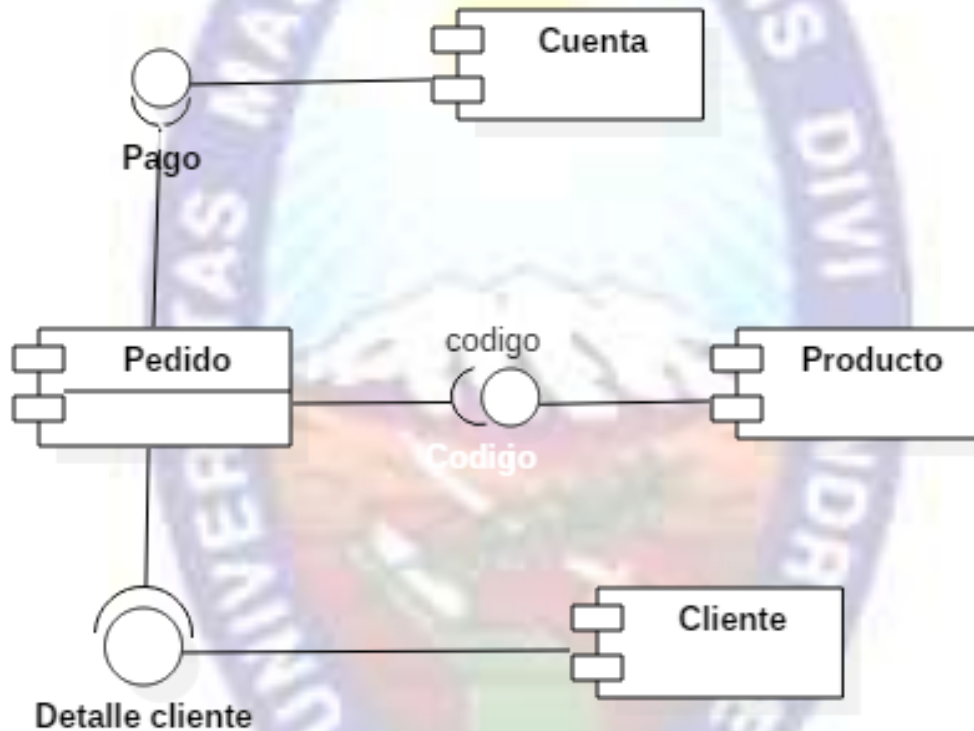


**Figura 3. 20 Diagrama de secuencia de módulo carrito de compras**

**Fuente:** Elaboración propia

### 3.6.3.2. DIAGRAMA DE COMPONENTES

Se diseñó el diagrama de componentes para los requisitos: crear carrito de compra, listar línea de compra del carrito, eliminar producto del carrito, comprar producto, generar recibo y adicionar productos al carrito el cual se lo visualiza en la figura 3.21.

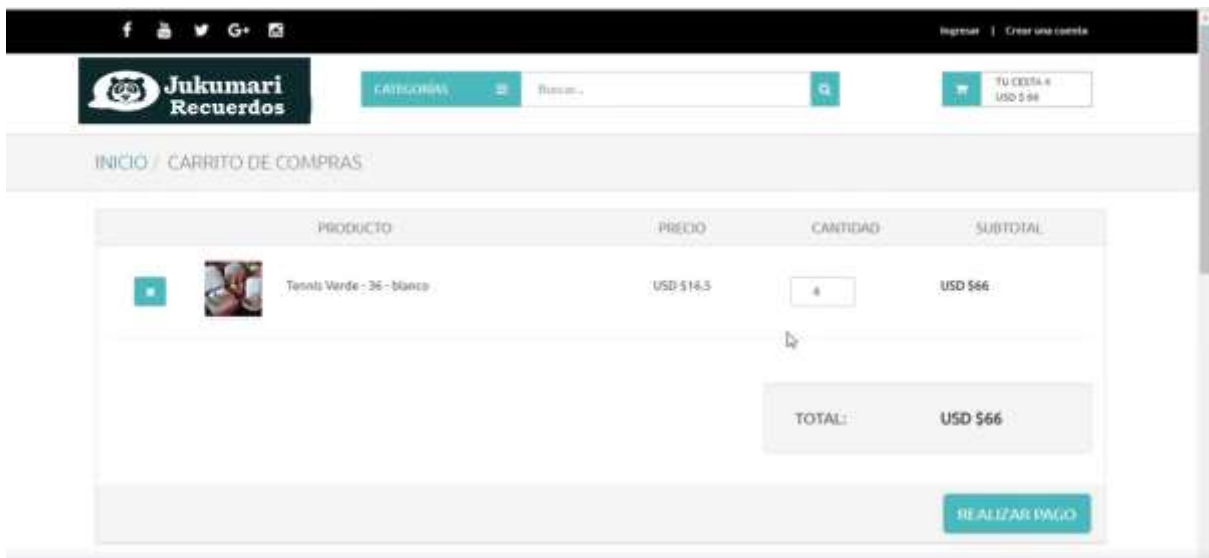


**Figura 3. 21 Diagrama de componentes**

**Fuente:** Elaboración propia

### 3.6.4. RESULTADOS

Para esta parte se desarrolló el módulo de carrito de compras, se puede observar la figura 22 que muestra una captura de pantalla.



**Figura 3. 22 Captura de pantalla del carrito de compra**

**Fuente:** Elaboración propia

### **3.7. CUARTO SPRING: MÓDULO DE ADMINISTRACIÓN DE CATÁLOGO Y PRODUCTOS**

#### **3.7.1. ETAPA DE ANÁLISIS**

La tabla 3.20 muestra el sprint asignado para este módulo, que contiene tareas.

**Tabla 3. 20 Cuarto spring módulo de administración de catálogo y productos**

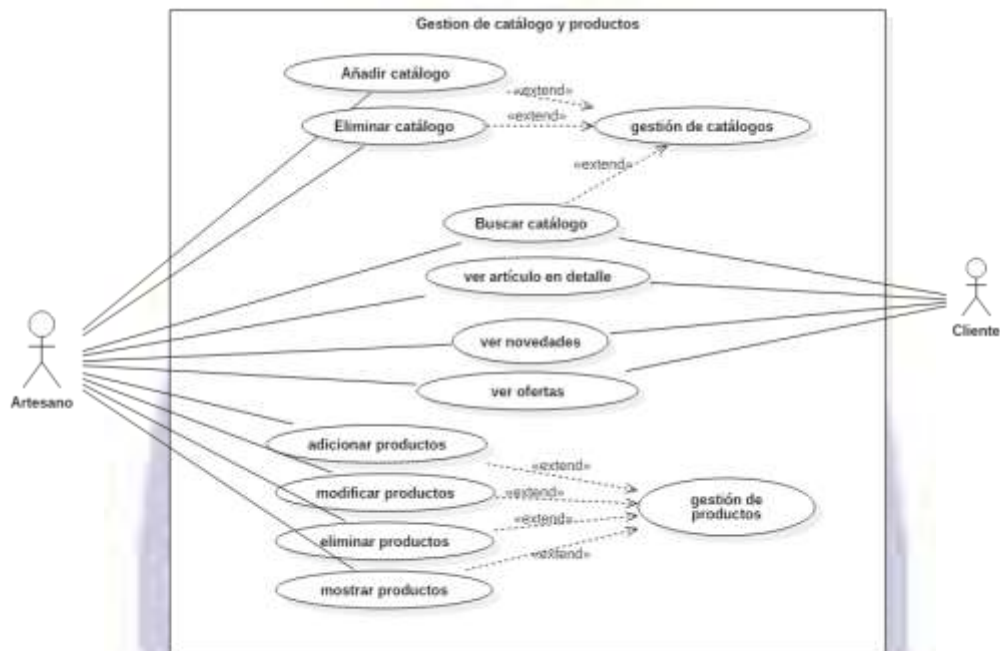
<b>Sprint o Iteración: 4</b>		<b>Inicio</b>	<b>Fin</b>	<b>Duración</b>
		03/04/2018	25/04/2018	22 días
<b>Nro</b>	<b>Tarea</b>	<b>Duración tarea</b>		<b>Estado</b>
1	Maquetación de la vitrina de productos y catalogo	2 días		Completado
2	Maquetación de productos en formato lista	2día		Completado

3	Datos dinámicos de los productos	3 días	Completado
4	Maquetación de la página de productos con datos dinámicos	2 días	Completado
5	Diseño e implementación del buscador	3 días	Completado
6	Creación el visor de imágenes del producto	3 días	Completado
7	Maquetar sección de comentarios	3 días	Completado
8	Diseño e implementación de contador de vistas por productos	4 días	Completado

**Fuente:** Elaboración propia

### 3.7.1.1. DIAGRAMA DE CASO DE USO

Se diseñó el diagrama de casos de uso para los requisitos: gestión de catálogo y gestión de productos, maquetación de la vitrina de productos y catálogo, maquetación de productos en formato lista, maquetación de la página de productos con datos dinámicos, diseño e implementación del buscador, maquetar sección de comentarios, creación el visor de imágenes del producto, maquetar sección de comentarios y diseño e implementación de contador de vistas por productos, el cual se lo visualiza en la figura 3.23.



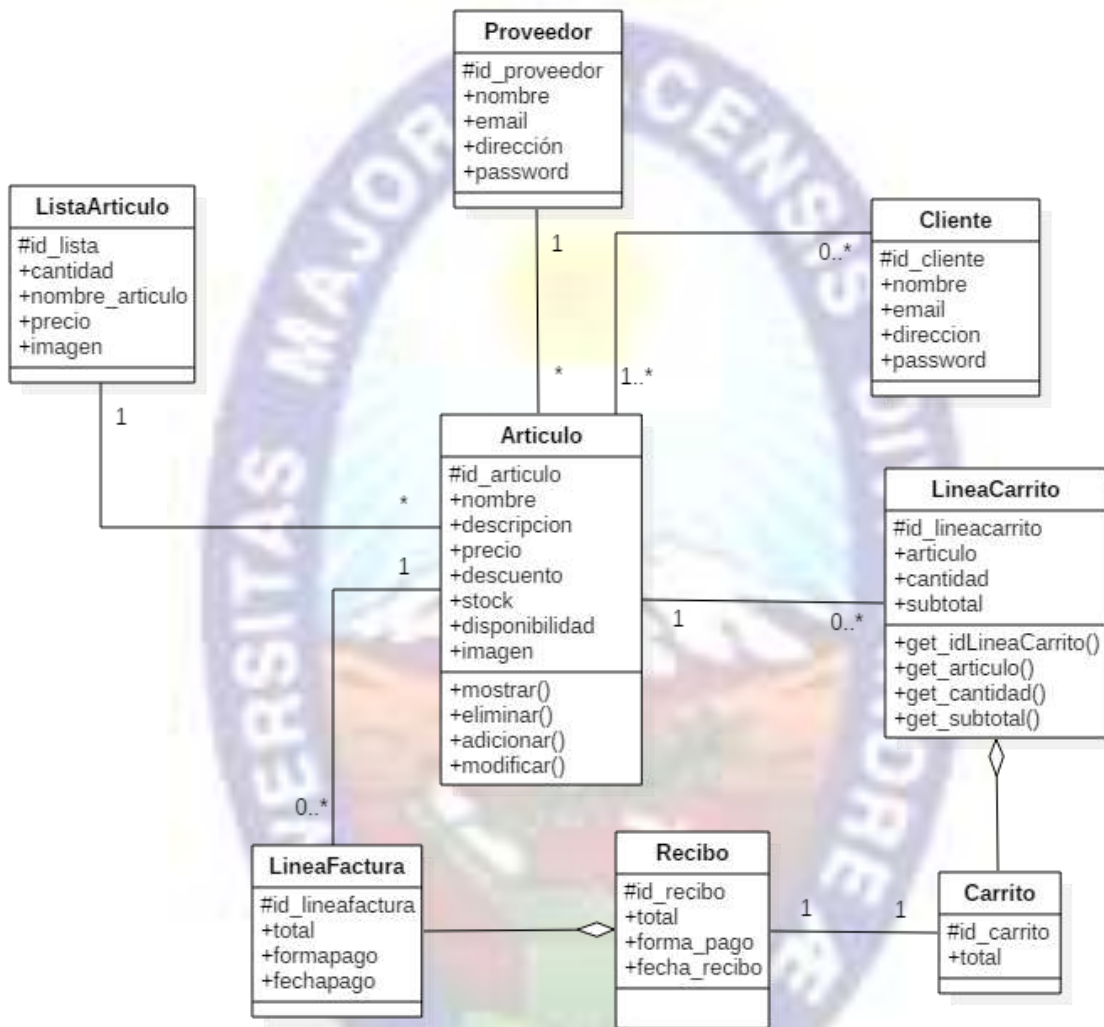
**Figura 3. 23 Diagrama de caso de uso del módulo de catálogo**  
 Fuente: Elaboración propia

### 3.7.2. ETAPA DE DISEÑO

En la fase de diseño se utilizó los diagramas de clases y de estados.

#### 3.7.2.1. DIAGRAMA DE CLASES

Se diseñó el diagrama de casos de uso para los requisitos: gestión de catálogo y gestión de productos, maquetación de la vitrina de productos y catálogo, maquetación de productos en formato lista, maquetación de la página de productos con datos dinámicos, diseño e implementación del buscador, maquetar sección de comentarios, creación el visor de imágenes del producto, maquetar sección de comentarios y diseño e implementación de contador de vistas por productos, el cual se lo visualiza en la figura 3.24.



**Figura 3. 24 Diagrama de clases del módulo de catálogo**

**Fuente:** Elaboración propia

### 3.7.2.2. DIAGRAMA DE ESTADO

Se diseñó el diagrama de casos de uso para los requisitos: gestión de catálogo y gestión de productos, maquetación de la vitrina de productos y catálogo, maquetación de productos en formato lista, maquetación de la página de productos con datos dinámicos, diseño e implementación del buscador, maquetar sección de comentarios, creación el visor de



imágenes del producto, maquetar sección de comentarios y diseño e implementación de contador de vistas por productos, el cual se lo visualiza en la figura 3.25.



**Figura 3. 25 Diagrama de estados de módulo de catálogo**

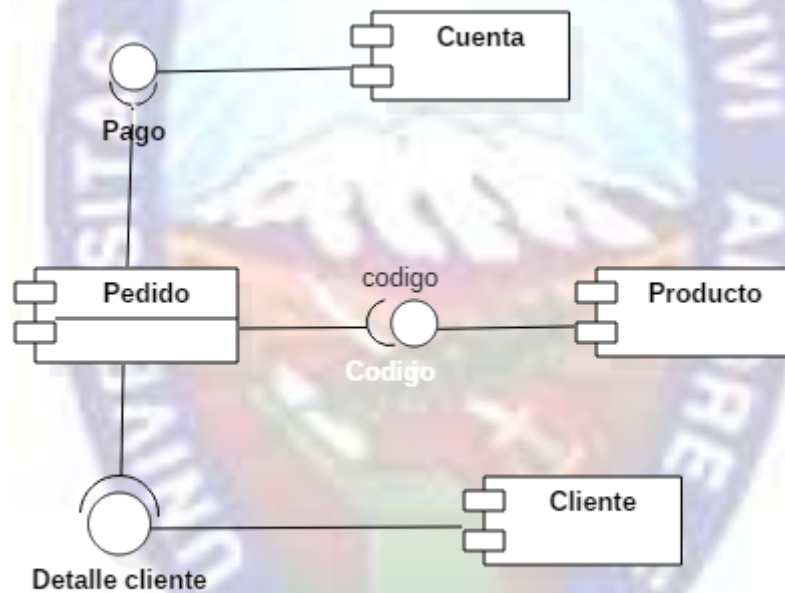
**Fuente:** Elaboración propia

### 3.7.3. ETAPA DE IMPLEMENTACIÓN

En la fase de diseño se utilizó los diagramas de secuencia y de componentes.

### 3.7.3.2. DIAGRAMA DE COMPONENTES

Se diseñó el diagrama de casos de uso para los requisitos: gestión de catálogo y gestión de productos, maquetación de la vitrina de productos y catálogo, maquetación de productos en formato lista, maquetación de la página de productos con datos dinámicos, diseño e implementación del buscador, maquetar sección de comentarios, creación el visor de imágenes del producto, maquetar sección de comentarios y diseño e implementación de contador de vistas por productos, el cual se lo visualiza en la figura 3.26.



**Figura 3. 26 Diagrama de componentes de módulo de catálogo**

**Fuente:** Elaboración propia

### 3.7.4. RESULTADOS

Para esta parte se desarrolló el módulo de administración de catálogo, se puede observar la figura 3.27, la figura 3.28 y la figura 3.29 que muestra una captura de pantalla.



Figura 3. 27 Captura de pantalla de cabecera del catálogo

Fuente: Elaboración propia

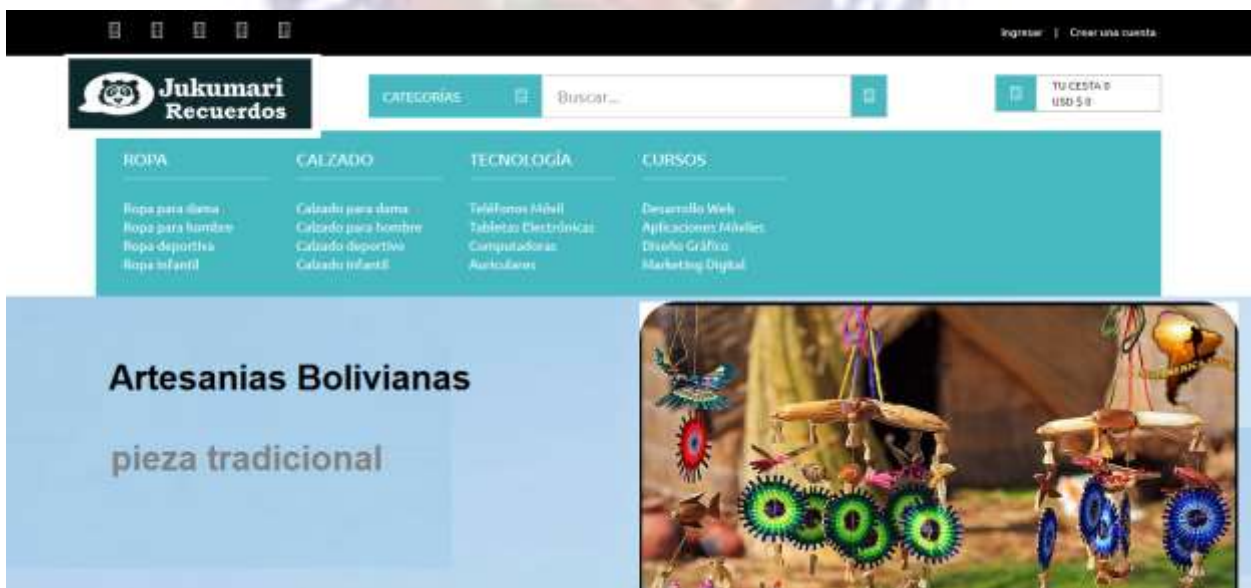


Figura 3. 28 Captura de pantalla de la lista catálogo

Fuente: Elaboración propia



Figura 3. 29 Captura de pantalla de la lista productos

Fuente: Elaboración propia

### 3.8. POSTGAME Y FASE DE ESTABILIZACIÓN, PRUEBAS Y REPARACIONES

En esta fase se finaliza el desarrollo de la plataforma, para ello se realizaron las pruebas necesarias y se dará solución a las mismas, además se asegurara la calidad del producto desarrollado utilizando pruebas de aceptación del producto.

#### 3.4.1. PRUEBAS Y REPARACIONES

Al realizar las pruebas a los sistemas, se encontraron un total de 7 errores y observaciones, los cuales se describen en la Tabla 3.21.

Tabla 3. 21 Errores encontrados en la plataforma

Nro	Descripción
1	Para actualizar los datos del catálogo se tiene que actualizar toda la página necesariamente

2	La llamada a los servicios en ocasiones demora más de 30 segundos.
3	El envío de imágenes al momento de cargar un producto demora hasta 5 minutos.
4	Errores no controlados cuando no se tiene acceso a intent
5	La lista de productos no cuenta con un scroll.
6	El servicio para el cargado de imágenes del carrusel demora más de 30 segundos

**Fuente:** Elaboración propia

Luego de analizar los errores encontrados se procedió a realizar las reparaciones respectivas a cada uno de los errores.

### **3.4.2. PRUEBAS DE CAJA NEGRA**

Las pruebas de caja negra se centran principalmente en lo que se quiere de un módulo, específica de un software, es decir, es una manera de encontrar casos específicos en ese modulo que atiendan a su especificación

Las pruebas de caja negra son, ni más ni menos que, pruebas funcionales dedicadas a “mirar” en el exterior de lo que se prueba y pruebas unitarias. Estas pruebas se denominan de varias formas, pruebas de caja “opaca”, pruebas de entrada/salida, pruebas inducidas por datos...los sinónimos son muchos y muy variados. En el presente proyecto las pruebas de caja negra se la realizan bajo los siguientes niveles:

- **Pruebas unitarias**, se realiza al momento del desarrollo de cada uno de los módulos del sistema verificando su funcionalidad.
- **Pruebas de integración**, esta prueba se realiza cuando todos los módulos están diseñados y desarrollados, para su posterior prueba en general.

# CAPITULO IV

## CALIDAD Y SEGURIDAD



## **4.1. INTRODUCCIÓN**

En este capítulo se hará un análisis posterior al desarrollo e implementación del SISTEMA WEB PARA EL CONTROL Y SEGUIMIENTO DE VENTAS DE PRODUCTOS ARTESANALES CASO: BOLIVIA TECH HUB, en este análisis se comprobará la calidad del Software, seguridad y costos mediante un análisis y haciendo uso de uno de los estándares.

## **4.2. CALIDAD DE SOFTWARE**

Con el desarrollo de aplicaciones cada vez más complejas orientadas a la web se ha hecho necesario adoptar metodologías de desarrollo de software especialmente enfocadas a este medio, siempre teniendo como objetivo esencial la calidad, **Web-Site QEM**, define un enfoque integral, sistemático y cuantitativo para evaluar y comparar productos Web, tanto en la fase operativa como en la fase de desarrollo del ciclo de vida Web.

Según Olcina 1999, **Web-Site QEM**, es esencialmente integral, flexible y robusto, y cubre la mayor parte de las actividades en el proceso de evaluación, comparación, y selección de artefactos Web.

Una de las metas principales de la evaluación y comparación de calidad de artefactos Web, radica en comprender el grado de cumplimiento de un conjunto de características y subcaracterísticas con respecto a los requerimientos de calidad establecidos (Olcina, 1999).

### **4.2.1. DEFINIENDO METAS DE EVALUACIÓN**

Como detalla Olsina 1999, en esta fase deben definirse y refinar las metas y el alcance del proceso de evaluación. Para la evaluación del presente sistema web se ha escogido la fase operativa, la meta principal consiste en “comprender la calidad global de un sitio web desde el punto de vista del visitante”.

### **4.2.2. ESPECIFICANDO REQUERIMIENTOS DE CALIDAD**

Como especifica Olsina 1999, en esta fase los evaluadores deben acordar y especificar las características, sub-características y atributos de calidad agrupándolas en un árbol de

requerimientos. Respecto de las características de calidad de más alto nivel, se sigue la misma clasificación conceptual que la prescrita en el estándar ISO [ISO/IEC 9126]. Estas características de alto nivel son: **usabilidad, funcionalidad, confiabilidad, eficiencia, portabilidad, y mantenibilidad**, donde se deben seleccionar el subconjunto de características de primer nivel, conforme a las metas y el perfil de usuario seleccionado, en presente caso se dejaron fuera las características de *Mantenibilidad* y *Portabilidad* por la poca relevancia que tienen para el perfil seleccionado.

En la siguiente tabla puede apreciarse el árbol de requerimientos de calidad. Mencionar que para la construcción de este árbol se toma como base las características, sub-características y atributos citados por Olsina 1999, que es un modelo básicamente estándar para utilizarlo en distintos dominios.

#### **4.2.3 ESPECIFICACIÓN DE CARACTERÍSTICAS DE CALIDAD**

Para la medida de calidad se especificarán a continuación las características de usabilidad, funcionalidad, confiabilidad, eficiencia y mantenibilidad.

##### **4.2.3.1. USABILIDAD**

Es una característica de calidad de producto de alto nivel, que se la puede medir mediante cálculo a partir de métricas directas e indirectas, representa la capacidad o potencialidad del producto para ser utilizado, comprendido y operado por los usuarios, además de ser atractivo para cualquiera.

El criterio de evaluación es un criterio binario, discreto y absoluto. Solo se pregunta si está disponible representado por 1 y si no está disponible con un 0.

Según Olsina de 1999 para evaluar la usabilidad se debe considerar las siguientes características:

- **Comprensibilidad global del sitio:** Es una característica que representa a todas aquellas facilidades que permiten la audiencia, tener una rápida comprensión tanto de la estructura organizativa, como el contenido del sitio web como un todo, facilitando el rápido acceso y recorrido del mismo con sus componentes. Por tal



razón, los atributos y sub-características se hallan principalmente en la página principal o en los primeros niveles del sitio, tabla 4.1.

**Tabla 4. 1 Comprensibilidad global del sitio**

<b>Característica: Comprensibilidad Global del Sitio</b>		
<b>Nro.</b>	<b>Sub-característica/s</b>	<b>Resultado</b>
1.	Esquema de Organización Global	0,67
1.1	Mapa del sitio	0,95
1.2	Tabla de contenidos	1,00
1.3	Índice	0,00
2.	Calidad del sistema de etiquetado	1,00
3.	Visita guiada orientada al usuario	1,00
4.	Mapa de imagen	1,00
<b>TOTAL</b>		<b>0,83</b>

**Fuente:** Elaboración propia

- Mecanismo de ayuda y retroalimentación en línea: Este atributo representa a un conjunto de preguntas (agrupadas y enlazadas) que se realizan con mayor frecuencia, y que están ya publicadas en el sitio con sus respectivas respuestas. A su vez, las respuestas pueden estar enlazadas a otros contenidos. Esto favorece al mecanismo de aprendizaje y/o ayuda evitando potencialmente la demora cognitiva de los visitantes, tabla 4.2.

**Tabla 4. 2 Característica: Mecanismo de Ayuda y Retroalimentación en Línea**

<b>Característica: Mecanismo de Ayuda y Retroalimentación en Línea</b>		
<b>Nro.</b>	<b>Sub-característica/s</b>	<b>Resultado</b>
1.	Calidad de la ayuda	0,90
1.1	Ayuda explicada orientada al usuario	0,80
1.2	Ayuda de la búsqueda	1,00
2.	Indicador de última actualización	1,00
2.1	Global	1,00
2.2	Restringido	1,00
3.	Directorio de direcciones	1,00
3.1	Directorio e-mail	1,00
3.2	Directorio te-fax	1,00
3.3	Directorio Correo Postal	1,00
4.	Facilidad FAQ	0,00
5.	Retroalimentación	0.67

5.1	Cuestionario	0,00
5.2	Libro de invitado	1,00
5.3	Comentarios/sugerencias	1,00
<b>TOTAL</b>		<b>0.71</b>

**Fuente:** Elaboración propia

- Aspectos de interfaces y estéticos: Son factores y elementos relativos a la interacción del usuario, enfocados a un entorno o dispositivos concretos cuyo resultado es la generación de una percepción positiva o negativa de dicho servicio, producto o dispositivo. El diseño de los elementos de la interfaz debe facilitar la interacción del usuario con la funcionalidad, debe generar y formalizar documentos hipertextuales comprensibles, interactivos, navegables y facilitar su visualización, tabla 4.3.

**Tabla 4. 3 Aspectos de Interfaces y Estéticos**

<b>Característica: Aspectos de Interfaces y Estéticos</b>		
<b>Nro.</b>	<b>Sub-característica/s</b>	<b>Resultado</b>
1	Cohesión al agrupar los objetos de control principales	1,00
2	Permanencia y estabilidad en la presentación de los controles principales	1,00

2.1	Permanencia de los controles directos	1,00
2.2	Permanencia de los controles indirectos	1,00
2.3	Estabilidad	1,00
3	Aspectos de Estilo	1,00
3.1	Uniformidad en el color de los enlaces	1,00
3.2	Uniformidad en el estilo global	1,00
3.3	Guía del estilo global	1,00
4	Preferencia Estética	1,00
<b>TOTAL</b>		<b>1,00</b>

**Fuente:** Elaboración propia

- Misceláneas: Este atributo modela el número de lenguajes extranjeros soportados por un sitio (sitios de dominios de aplicación de índole académica, museos, comercio electrónico y otros). Además, especifica el nivel de soporte de cada lenguaje: Total (todas las páginas del sitio), parcial (algunos sub-sitios del sitio), o mínimo (algunas páginas o documentos de algunos sub-sitios). No se computa obviamente el lenguaje nativo del sitio web, tabla 4.4.

**Tabla 4. 4 Website QEM Evaluación de misceláneas**

<b>Característica: Misceláneas</b>		
<b>Nro.</b>	<b>Sub-característica/s</b>	<b>Resultado</b>
1.	Soporte de lenguaje extranjero	0,00
2.	Atributo “Qué es lo nuevo”	1,00
3.	Indicador de resolución de pantalla	1,00
<b>TOTAL</b>		<b>0,67</b>

**Fuente:** Elaboración propia

La usabilidad de la aplicación se determinará por el promedio de las anteriores características, como se muestra en la siguiente tabla 4.5:

**Tabla 4. 5 Total de Usabilidad**

<b>Nro.</b>	<b>Criterio</b>	<b>Resultado</b>
1.	Comprensibilidad global del sitio	0,92
2.	Mecanismos de ayuda y retroalimentación en línea	0,71
3.	Aspectos de interfaces y estéticos	1,00

4.	Misceláneas	0,67
<b>TOTAL</b>		<b>0,83</b>

**Fuente:** Elaboración propia

#### 4.2.3.2. FUNCIONALIDAD

Para determinar la calidad de la funcionalidad de la aplicación se debe analizar la búsqueda y exploración de contenidos. El criterio de evaluación es un criterio binario, discreto y absoluto. Solo se pregunta si está disponible representado por 1 y si no está disponible con un 0.

Según Olsina de 1999 para evaluar la funcionalidad se debe considerar las siguientes características:

- Aspectos de búsqueda y recuperación: Es una característica que modela el mecanismo que permite tener un modo directo de encontrar información, tabla 4.6.

**Tabla 4. 6 Aspectos de Búsqueda y Recuperación**

<b>Característica: Aspectos de Búsqueda y Recuperación</b>		
<b>Nro.</b>	<b>Sub-característica/s</b>	<b>Resultado</b>
1.	Mecanismo de búsqueda en el sitio web	0,75
1.1	Búsqueda restringida	1,00
1.1.1	De destinos	1,00
1.1.2	De viajes	1,00

1.1.3	De asientos	1,00
1.2	Búsqueda global	0,50
2.	Mecanismos de recuperación	1,00
2.1	Nivel de Personalización	1,00
2.1	Nivel de retroalimentación en la recuperación	1,00
<b>TOTAL</b>		<b>0,88</b>

**Fuente:** Elaboración propia

- Aspectos de dominio orientados al usuario: Es la información que el usuario puede tener y la información de aprobación de operaciones realizadas por este, tabla 4.7.

**Tabla 4. 7 Aspectos de Dominio Orientados al Usuario**

<b>Característica: Aspectos de Dominio Orientados al Usuario</b>		
<b>Nro.</b>	<b>Sub-característica/s</b>	<b>Resultado</b>
1.	Relevancia del contenido	1,00
1.1	Información sobre el viaje	1,00
1.2	Información sobre	1,00

	destinos	
2.	Servicios on-line	1,00
<b>TOTAL</b>		<b>1,00</b>

**Fuente:** Elaboración propia

La funcionalidad de la aplicación se determinará por el promedio de las anteriores características, como se muestra en la siguiente tabla 4.8:

**Tabla 4. 8 Total Funcionalidad**

<b>Nro.</b>	<b>Criterio</b>	<b>Resultado</b>
1.	Aspectos de búsqueda y recuperación	0,88
2.	Aspectos de navegación y exploración	0,83
3.	Aspectos del dominio orientados al usuario	1,00
<b>TOTAL</b>		<b>0,90</b>

**Fuente:** Elaboración propia

#### **4.2.3.3. CONFIABILIDAD**

La medición de esta característica está definida por el complemento de los casos de deficiencia encontrados en la aplicación.

El criterio elemental es uno de variable normalizada, continuo y absoluto; en donde si  $BL =$



Número de enlaces rotos encontrados. TL = Número total de enlaces del sitio. La fórmula para computar la variable es:

$$X = 100 - (BL * 100/TL) * 10;$$

Donde, si  $X < 0$  entonces  $X = 0$ .

- No deficiencia: Este atributo representa básicamente a los enlaces encontrados que conducen a nodos destinos ausentes (también llamados enlaces ausentes o pendientes), tabla 4.9.

**Tabla 4. 9 Evaluación de confiabilidad**

<b>Característica: Confiabilidad</b>		
<b>Nro.</b>	<b>Sub-característica/s</b>	<b>Resultado</b>
1.	No deficiencia	1,00
1.1	Errores de enlace	0,00
1.1.1	Enlaces rotos	0,00
1.1.2	Enlaces inválidos	0,00
1.1.3	Enlaces no implementados	0,00
1.2	Errores o deficiencias varias	0,00
1.2.1	Permanecía de los controles contextuales	0,00
1.2.2	Deficiencias o resultados inesperados	0,00
1.2.3	Nodo destino en construcción	0,00
<b>CONFIABILIDAD TOTAL</b>		<b>1,00</b>

**Fuente:** Elaboración propia

#### **4.2.3.4. EFICIENCIA**

Es una característica de calidad de producto de alto nivel que se la puede medir mediante cálculo a partir de métricas directas e indirectas y principalmente representa a la relación

entre el grado de performance del artefacto y la cantidad de recursos (tiempo, espacio, etc.) usados bajo ciertas condiciones.

El criterio de evaluación es un criterio binario, discreto y absoluto. Solo se pregunta si está disponible representado por 1 y si no está disponible con un 0.

Según Olsina de 1999 para evaluar la eficiencia se debe considerar las siguientes características:

- Desempeño: Se mide el tamaño de todas las páginas (estáticas) del sitio web considerando todos sus componentes gráficos, tabulares y contextuales. El tamaño de cada página se especifica como una función del tiempo de espera y de la velocidad mínima establecida para una línea de comunicación dada, tabla 4.10.

**Tabla 4. 10 Evaluación de desempeño**

<b>Característica: Desempeño</b>		
<b>Nro.</b>	<b>Sub-característica/s</b>	<b>Resultado</b>
1.	Páginas de acceso rápido	1,00
<b>TOTAL</b>		<b>1,00</b>

**Fuente:** Elaboración propia

### **4.3. SEGURIDAD**

Los problemas más frecuentes de seguridad en un sistema web, suelen venir de la configuración de las herramientas que se utilizan para su desarrollo o también pueden venir de un posible fallo en el diseño lógico.

- Entre las amenazas más comunes se encuentran.
- Ingreso de usuarios no valido.
- Control de Acceso roto.
- Administración de sesión y autenticación rota.
- Inyecciones de código.
- Manejo de errores inadecuados.
- Administración de seguridad insegura.

### **4.3.1. TIPOS DE SEGURIDAD**

Existen 4 tipos de seguridad en los sistemas web:

- Seguridad en el cliente.
- Seguridad en el servidor
- Seguridad en las comunicaciones.
- Seguridad en la aplicación

#### **4.3.1.1. SEGURIDAD EN EL CLIENTE**

Uno de los mecanismos de seguridad es la validación por el lado del cliente. Mecanismos que se encargan de validar la información antes que llegue al servidor.

#### **4.3.1.2. SEGURIDAD EN EL SERVIDOR**

También es necesario realizar controles por lado del servidor tanto en el servidor de la aplicación y el servidor de la base de datos.

Normalmente un servidor de aplicaciones proporciona muchos servicios que no son necesarios para el funcionamiento de la aplicación y que pueden producir problemas de seguridad, para el presente caso se tomó en cuenta la desactivación de las opciones innecesarias para el funcionamiento del servidor.

En cuanto a la seguridad en el servidor de base de datos, el mayor problema son las inyecciones SQL (Lenguaje de consultas estructuradas), ataques realizados a la base de datos. Para el presente caso se procedió a diseñar y codificar una serie procedimientos almacenados para ocultar los meta-caracteres inyectados, la base de datos para el sistema web contiene una serie de procedimientos almacenados encargados de realizar altas, bajas, listados, modificaciones y búsquedas.

#### **4.3.1.3. SEGURIDAD EN LA COMUNICACIÓN**

Como se hizo la utilización del framework laravel que tiene las siguientes características en cuanto a seguridad.

- **HTTP Middleware:** Middleware se encarga de **analizar y filtrar las llamadas HTTP en tu servidor**. Puedes instalarlo para que se encargue de verificar que

se trate de un usuario registrado, de evitar problemas de tipo Cross-Site-Scripting (XSS) y otras medidas de seguridad.

- Autenticación: Laravel viene listo para **implementar autenticación de usuarios de forma nativa** e incluye la opción de “recordar” al usuario. Además, permite incluir parámetros adicionales, lo que nos asegurará, por ejemplo, si se trata de un usuario activo.
- Integración con Stripe: **Laravel Cashier** incluye todo lo necesario para integrar tu desarrollo con este servicio de cobro. Además, este se puede sincronizar e integrar con el sistema de autenticación de usuarios. Así que ya no te tienes que preocupar por cómo **integrar un sistema de cobros** a tu desarrollo.
- Encriptación: Una aplicación segura necesita ser capaz de encriptar sus datos. Con Laravel se tiene todo lo necesario para empezar a usar seguridad OpenSSL y cifrado AES-256-CBC. Adicionalmente, todos los valores encriptados están firmados por un **código de autenticación de mensaje** que detecta si el mensaje encriptado fue alterado.

## **CAPITULO V**

# **ANÁLISIS DE COSTO BENEFICIO**



## 5.1. INTRODUCCIÓN

La técnica de análisis de costo y beneficio, tiene como objetivo fundamental proporcionar una medida sobre la rentabilidad de un proyecto, haciendo una comparación de los costos previstos con los beneficios esperados en la realización del mismo. Esta técnica se debe utilizar al comparar proyectos y así poder tener una buena toma de decisiones.

## 5.2. COCOMO II

COCOMO II es un modelo que permite estimar el coste, esfuerzo y tiempo cuando se planifica una nueva actividad de desarrollo software. Está asociado a los ciclos de vida modernos. El modelo original COCOMO ha tenido mucho éxito pero no puede emplearse con las prácticas de desarrollo software más recientes tan bien como con las prácticas tradicionales. COCOMO II sigue los principios de apertura usados en el COCOMO original de esta manera todos sus algoritmos y relaciones están disponibles públicamente.

### 5.2.1. MÉTRICAS DE SOFTWARE

Para estimación del tamaño de software COCOMO II utiliza tres técnicas: puntos objeto, puntos función no ajustados y líneas de código fuente. Además se emplean otros parámetros relacionados al tamaño que contempla aspectos como reusó, reingeniería, conversión y mantenimiento. En el presente proyecto se hace uso del punto función no ajustado como base para medir tamaño en los modelos de estimación.

Los Puntos Función procuran cuantificar la funcionalidad de un sistema de software. La meta es obtener un número que caracterice completamente al sistema. COCOMO II considera solamente UFP (Puntos Función no ajustados) (Gómez & Migani, 2007).

La fórmula para calcular los puntos función, es la siguiente:

$$FP = UFP \times TCF$$

Dónde:

- UFP: Puntos Función no Ajustados

- TCF: Factor de Complejidad Técnica

Estimación de punto función no ajustado

Una vez identificados los ítems se clasifican de acuerdo al grado de complejidad en: bajo, promedio o alto. Se asigna un peso a cada ítem según el tipo y el grado de complejidad correspondiente. Finalmente los UFP son calculados mediante la sumatoria de los pesos de todos los ítems identificados, tabla 5.1.

$$UFP = \sum_{i=1}^5 (cantidad\_items_i \times peso_i)$$

**Tabla 5. 1** Calculo del punto función no ajustado

Parámetros de medición	Cuenta	Factor de ponderación	Total
Nº Entradas Externas (Inputs)	11	4	44
Nº Salidas Externas (Outputs)	9	5	45
Nº Archivo Lógicos Internos	12	6	72
Nº Archivos Externos de Interface	9	7	63
Solicitudes Externas (Queries)	12	7	84
UFP			308

**Fuente:** Elaboración propia

Los factores de ponderación asociadas a cada tipo de ítem han sido derivadas de la observaciones realizadas por (Boehm, 1995). Para el cálculo del Factor de Complejidad Técnica, TCF, se considera la siguiente fórmula:

$$TFC = 0.65 + 0.01 \times \sum_{i=1}^{14} F_i$$

Donde los  $F_i$  corresponden a los pesos asignados a los factores, los pesos se consideran dentro de una escala de 0 a 5, tabla 5.3.

**Tabla 5. 2 Escala de niveles de influencia**

0	1	2	3	4	5
Sin influencia	Moderado	Incidental	Medio	Significativo	Esencial

**Fuente:** Elaboración propia

Los factores de ajuste son representados en 14 puntos que se muestran en la tabla 5.3 y se debe asignar un puntaje del 0 al 5 a cada factor.

**Tabla 5. 3 Suma de Factor de complejidad**

<b>F</b>	<b>Factor de complejidad</b>	<b>Valor</b>
1	Mecanismos de recuperación y back-up confiables	1
2	Comunicación de Datos	3
3	Funciones de Procesamiento Distribuido	2
4	Performance	2
5	Configuración usada rigurosamente	1
6	Entrada de datos on-line	3
7	Factibilidad Operativa	3
8	Actualización de archivos on-line	2
9	Interfaces Complejas	1



10	Procesamiento Interno Complejo	3
11	Reusabilidad	3
12	Fácil Instalación	3
13	Soporte de múltiples instalaciones	4
14	Facilidad de cambios y amigabilidad	4
15	Total	35

**Fuente:** Elaboración propia

De la fórmula de factor de complejidad técnica se tiene:

$$TFC = 0.65 + 0.01 \times \sum_{i=1}^{14} F_i$$

$$TFC = 0.65 + 0.01 * 35$$

$$TFC = 1$$

Aplicando la formula FP se tiene:

$$FP = UFP \times TCF$$

$$FP = 308 \times 1 = 308$$

Por tanto, el punto función (PF) es de 308 para la utilización de este proyecto.

Para determinar el esfuerzo nominal en el modelo COCOMO II los puntos función no ajustados tienen que ser convertidos a líneas de código fuente considerando el lenguaje de implementación, para esto se toma en cuenta la siguiente tabla 5.4.

**Tabla 5. 4 Factor LDC/PF**

Lenguaje	Nivel	Factor LDC/PF
----------	-------	---------------

C	2.5	128
Ansi Basix	5	64
Java	6	53
ASP	9	36
PHP	11	29
Visual c++	9.5	35

**Fuente:** (Gómez & Migani, 2007)

La fórmula para convertir los puntos función no ajustados a líneas de código fuente, es la siguiente:

$$LDC = FP \times Factor\ LDC/PF$$

$$LDC = 308 \times 29 = 8932$$

### 5.2.2. ESTIMACIÓN DEL ESFUERZO

El esfuerzo necesario para concretar un proyecto de desarrollo de software, cualquiera sea el modelo empleado, se expresa en meses/persona (PM) y representa los meses de trabajo de una persona full-time, requeridos para desarrollar el proyecto.

El modelo usado para calcular el esfuerzo es el propuesto por COCOMO.

$$E = a_b(KLD)^{b_b}$$

Dónde:

- E: Esfuerzo aplicado por persona
- (KLDC: Número líneas de código)

$$D = C_b(E)^{d_a}$$

Dónde:

- D: Tiempo de desarrollo en meses

Se utiliza para obtener una primera aproximación rápida del esfuerzo y hace uso de la tabla 5.5 de constantes para calcular distintos aspectos de costes:

**Tabla 5. 5 Modelo básico para tipos de proyecto**

Proyecto de software	$a_a$	$b_b$	$c_c$	$d_d$
Orgánico	2.4	1.05	2.5	0.38
Semi-acoplado	3	1.12	2.5	0.35
Empotrado	3.6	1.2	2.5	0.32

**Fuente:** (Gómez & Migani, 2007)

Aplicando las formulas anteriormente descritas y tomando en cuenta un tipo de proyecto semi-acoplado se tiene lo siguiente:

$$E = a_b(KLD)^{b_b}$$

$$E = 3 \times (8.9)^{1.12} = 34.20$$

Para hallar el tiempo de desarrollo en meses, utilizamos la fórmula:

$$D = C_b(E)^{d_d}$$

$$D = 2.5 \times (34.20)^{0.35} = 7.4$$

El personal requerido será igual a: Numero de Programadores=E/D

Por tanto:

$$\text{Número de Programadores} = 34.20 / 7.4 = 4.62 \Rightarrow 5$$

### 5.3. COSTO DE SOFTWARE

Tomando como base el salario aproximado de un programador junior 350 \$, esta cifra se toma en cuenta para estimar el costo del software:

Costo de desarrollo= Nro Programadores\*salario Programador\*duración

$$\text{Costo de desarrollo} = 5 * 350\$ * 7 = 12250\$$$

### 5.3.1. COSTO DE ELABORACIÓN DEL PROYECTO

Los costos de elaboración del proyecto se detallan en la tabla 5.6:

**Tabla 5. 6 Costo de elaboración del software**

Descripción	Costo Total (\$us)
Análisis y Diseño del proyecto	400
Pasajes	40
Material de escritorio	15
Otros	12
Total	467

**Fuente:** Elaboración propia

### 5.3.2. COSTE DE IMPLEMENTACIÓN DEL PROYECTO

En la institución se cuenta con la tecnología para implementar el sistema web, Servidores de aplicación y base de datos, por esta razón el costo es 0.

### 5.3.3. COSTE TOTAL

El coste total del sistema es la suma de los costos estos se presentan en la tabla 5.7:

**Tabla 5. 7 Costo total del software**

Descripción	Costo total (\$us)
Costo de desarrollo	12250
Coste de elaboración	467
Costo de implementación	0
Total	12717

**Fuente:** Elaboración propia

## 5.4. CÁLCULO BENEFICIO VAN Y TIR

El VAN y el TIR son herramientas financieras que permite evaluar la rentabilidad de un proyecto de inversión, con estos dos métodos se calculara el beneficio del proyecto.

### 5.4.1. VALOR ACTUAL NETO (VAN)

El VAN es un procedimiento que permite calcular el valor presente de un determinado número de flujos de futuros ingresos y egresos que tendrá el proyecto. Este método consiste en descontar el momento actual, es decir actualizar por medio de una tasa de flujos de caja futuros del proyecto, a ese valor se le resta la inversión inicial, así el valor obtenido es el valor actual neto.

$$VAN = \sum \frac{Ganancias}{(1 + k)^n} - l_0$$

Dónde:

- VAN: Valor Actual Neto.
- Ganancias: ingreso del flujo anual.
- $l_0$ : Es el valor del desembolso inicial de la inversión.
- $K$ : Tasa de descuento o tasa de interés al préstamo.
- $n$ : Es el número de periodos considerados.

Los valores de ganancia esperados para el presente proyecto se calculan para 4 años, en este caso se utilizará una tasa de 11% que es un índice de préstamo, para calcular el VAN se tiene lo siguiente:

$$Inversion = 12717\$us \text{ y } k = 10\%$$

Los valores de ganancia esperados se detallan en la siguiente tabla:

**Tabla 5. 8 Calculo del VAN**

Año	Flujo de caja neto	$(1 - k)^n$	$\frac{Ganancias}{(1 + k)^n}$
-----	--------------------	-------------	-------------------------------

1	0	1.1	0
2	4000	1.21	3305.7851
3	6500	1.331	4883.5462
4	9000	1.4641	6147.1210
Total	19500	5.10	14336.4523

**Fuente:** Elaboración propia

De la fórmula:

$$VAN = \sum \frac{Ganancias}{(1+k)^n} - l_0$$

$$VAN = 14336.4523 - 12717 = 1619.4523$$

Se tiene que el valor obtenido es mayor a cero por lo que la inversión es factible en un principio y el proyecto es rentable.

#### 5.4.2. TASA INTERNA DE RETORNO (TIR)

El TIR es una tasa de descuento de un proyecto de inversión para que sea rentable. Cuando el VAN toma un valor igual a cero,  $k$  pasa a llamarse TIR. En términos generales, las mejores inversiones son las que las que proporcionan mayor TIR.

- Si el TIR es inferior a la tasa de descuento de la empresa, la inversión debería ser desestimada.
- Si El TIR es mayor a la tasa de descuento de la empresa la inversión es factible.

$$0 = -l_0 + \sum_{i=1}^n \frac{Q_i}{(1+TIR)^i}$$

Dónde:

- TIR: Tasa interna de retorno
- $l_0$ : Valor del desembolso inicial de la inversión
- $k$ : Tasa de interés de ahorro.
- $n$ : Es el número de periodos considerados.

Entonces para hallar el TIR, se necesita la inversión de 12717 \$us. Aplicando la formula se obtiene 14.217%, como es superior a la tasa de descuento; la inversión es factible.

### 5.5. COSTO BENEFICIO

Para hallar el costo beneficio de un proyecto se aplica la siguiente ecuación:

$$\frac{C: \text{costo}}{B: \text{Beneficio}}$$

Reemplazando los valores previamente calculados en la ecuación, tenemos:

$$\frac{C: \text{costo}}{B: \text{Beneficio}} = \frac{12717 \text{ $us.}}{4875 \text{ $us}}$$

Por tanto, por cada dólar invertido, la empresa tiene una ganancia de 2.61\$.



# CAPÍTULO VI

## CONCLUSIONES Y RECOMENDACIONES





## **6.1. CONCLUSIONES**

Luego de plantear el problema, diseñar, desarrollar y probar el sistema web para el BOLIVIA TECH HUB, aplicando todas las metodologías de análisis y diseño de software, se logró cumplir el objetivo general planteado en un inicio, y el desarrollo del sistema fue exitoso.

Tomando en cuenta los objetivos previamente planteados se llegó a las siguientes conclusiones:

- Se logró implementar un medio de difusión de productos artesanales, para la incrementar de ventas.
- Se logró brindar información visual de precios y descripción de productos artesanales.
- Se logró ofrecer información precisa de tiendas de productos artesanales.
- Se logró expandir el mercado de productos artesanales.
- Se logró proporcionar información de los talleres artesanales.
- Se logró centralizar la información de productos artesanales.
- Se logró centralizar la información de talleres artesanales.
- Se mejoró la administración de ventas.
- Se mejoró la gestión de las ventas que se desarrollan.

De esta manera se alcanzó el objetivo general y los objetivos específicos, planteado para mejorar el servicio de venta de productos artesanales, de manera que las personas sean informadas de la diversidad cultural de productos artesanales, además tengan la oportunidad de acceder a tiendas y productos en lugares remotos con un coste menor al que supone abrir tiendas físicas en cada ciudad.

## **6.2. RECOMENDACIONES**

A la culminación del presente proyecto se efectúan las siguientes recomendaciones:

- El sistema puede ser complementado con la opción de poder generar reportes en Excel.
- Desarrollar una aplicación móvil para mayor comodidad de los clientes, que navegan desde dispositivos móviles.
- Para el uso del sistema es recomendable cambiar las contraseñas de administradores mensualmente, por razones de seguridad del sistema.
- Aumentar al módulo de autenticación un CAPTCHA que aparezca cuando el usuario falle muchas veces en su autenticación, para evitar robo de cuentas.
- Realizar copias de seguridad de la base de datos si es posible diariamente o en su defecto semanalmente, ya que la información que contiene es muy importante para el buen funcionamiento del sistema.



## **BIBLIOGRAFÍA**

- Arana Quijije, J. V. (2014). *Desarrollo e implementación de un sistema de gestión de ventas de repuestos automotrices en el almacén de auto repuestos eléctricos marcos en la parroquia POSORJA cantón GUAYAQUIL, provincia GUAYAS*. Universidad Estatal Península De Santa Elena Facultad De Sistemas Y Telecomunicaciones, Ecuador. La Libertad: Universidad Estatal Península De Santa Elena.
- Avison, D., & Fitzgerald, G. (1999). *Information Systems Development: Methodologies, Techniques, and Tools*. McGraw-Hill.
- Bauer, F. (1985). *Software Engineering: A report on a Conference sponsored by the NATO Science Comittee*. Englewood Cliffs, New York, United States: Prentice-Hall.
- Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., y otros. (2001). *Manifesto for Software Agile Development*. Recuperado el Marzo de 2018, de Agile Manifesto: <http://agilemanifesto.org/>
- Bermejo, M. (2010). *El Kanban*. España: España de Creative Commons.
- Bjorkholm, T., & Bjorkholm, J. (2015). *Kanban in 30 Days*. Birmingham: Impactt Publishing Ltd.
- Blankenship, J., Bussa, M., & Millett, S. (2011). *Pro Agile .NET Development with Scrum*. New York: Apress.
- Boaventura, J., Peña, E., Verdecia, P., & Fustiel, Y. (2016). Elección entre una metodología ágil y tradicional basado en técnicas de soft computing. *Revista Cubana de Ciencias Informáticas, 10*, 145-158.
- Bolvia Tech Hub. (2015). *PROMOVIENDO LA TECNOLOGIA E INNOVACION*. La Paz.
- Booch, G., Rambaugh, J., & Jacobson, I. (1997). *The UML specification documents*. USA: Rational Software Corp.

- Booch, G., Rumbaugh, J., & Jacobson, I. (1998). *The UML specification documents*. NY: Rational Software Corp.
- Brown, J., & Rocher, G. (2013). *The Definitive Guide to Grails 2*. New York: Apress.
- Canós, J. (2005). Metodologías Ágiles en. *Universidad Politécnica*.
- Carrillo Cruz, E. (2017). *Sistema web de control de compras, ventas e inventarios para Comercial Ariana*. UMSA, La Paz. La Paz: Universidad Mayor de San Andres.
- Carrodegua, N. (2015). *Proteccion de formularios*. Obtenido de <http://norfipc.com/inf/como-proteger-formularios-web-evitar-inyeccion-codigosql>.
- Ceke, D., Durek, M., & Kasapovic, S. (2013). Web application functional size estimation based on COSMIC method and UWE approach. *Information & Communication Technology Electronics & Microelectronics*, 396-403.
- Clifton, M., & Dunlap, J. (2003). What is SCRUM? *Development Lifecycle*.
- Cornejo Velázquez, E. (2015). Uso de tableros virtuales Kanban como herramienta para mejorar productividad en equipos de trabajo. *Administración para el desarrollo*.(9), 79-95.
- Flores Gonzales, M. (2016). *Sistema de información para el control y seguimiento de ventas on-line de productos farmacéuticos distribuidora PHARMICA (Proyecto de Grado)*. Universidad Mayor de San Andres, La Paz. La Paz: Universidad Mayor de San Andres.
- Fried, L. (2000). When Bigger Is Not Better: Productivity and Team Size in Software Development. *Software Engineering Tools, Techniques, Practice*, 2(1), 15–25.
- Gacitúa, R. (2003). Métodos de desarrollo de software: El desafío pendiente de la estandarización. *Theoria*, 12(1), 23-42.

- Ghosh, S. (2013). *Systemic comparison of the application*. Recuperado el 21 de Febrero de 2018, de Project Management Center for Excellence: <https://pm.umd.edu/files/public/>
- Gómez, M., & Migani, S. (2007). *Cocoma -Un Modelo De Estimacion De Proyectos De Software*.
- Ibarra Guzmán, D., & Castañeda Islas, U. (2014). Metodología ágil scrumban en el proceso de desarrollo y Metodología ágil scrumban en el proceso de desarrollo y. *Research in Computing Science*, 79, 97-107.
- Ibarra, D., Castañeda, U., Perez, C., & Pedroza, B. (2014). Metodología ágil scrumban en el proceso de desarrollo y mantenimiento de software de la norma moprosoft. *Computing Science*.
- International Rugby Board. (Mayo de 2012). *Ley 20, Scrum. En Leyes del juego de Rugby*. Recuperado el Marzo de 2018, de <http://www.irblaws.com>
- Kappel, G., Pröll, B., Reich, S., & Retschizgge, W. (2006). *Web engineering: The discipline of systematic development of web applications*. John Wiley & Sons.
- Khurana, H., & Sohal, J. (2011). Agile: The necessitate of contemporary. *International Journal of*, 3(2), 1031-1039.
- Kniberg , H., & Skarin, M. (2010). *Kanban and Scrum making the most of both*. Estados Unidos: publisher InfoQ.
- Koch, N., & Kraus, A. (2002). *The expressive Power of UML-based Web Engineering*. Munich: Universidad de Munich Alemania.
- Larman, C. (1999). *UML y patrones Introducción al análisis y diseño orientado a objetos*. México: Pearson Educación.
- Larman, C. (2003). *Agile & iterative development: a manager's*. Boston: Addison-Wesley.

- Letelier Torres, P. (2002). *Desarrollo de Software Orientado a Objetos usando UML*.  
Obtenido de <http://www.dsic.upv.es/~uml/>
- Li, J., Moe, N., & Dybå, T. (2010). *Transition from a plan-driven process to scrum: a longitudinal case study on software quality*. New York: McGraw-Hill.
- Morales, R. (2015). *Gestión de tareas con Kanban. Una introducción a la gestión visual del trabajo*. Madrid, España: Rainer.
- Murugesan, S. (2008). *Web application development: Challenges and the role of web engineering*. En *Web Engineering: Modelling and Implementing Web Applications*. London: Springer London.
- Naciek, A. (2013). *Cifrados por Bloque – Blowfish*. Obtenido de Blowfish.:  
<http://lumbreras-criptografia.blogspot.com/2013/07/cifrados-por-bloqueblowfish>
- Navarro Cadavid, A., Fernández Martínez, J., & Morales Vélez, J. (2013). Revisión de metodologías ágiles para el desarrollo de software. *PROSPECTIVA*, 11(2), 30-39.
- Nolivos, G., Coronel, F., & Campaña, M. (2010). *Implementación de un Sistema Web*. Ecuador: Escuela Politécnica del Ejército.
- Palacio, J. (2014). Recuperado el Marzo de 2018, de Gestión de Proyectos Scrum Manager:  
[http://www.scrummanager.net/files/sm\\_proyecto.pdf](http://www.scrummanager.net/files/sm_proyecto.pdf)
- Palacios, J. (2016). *KANBAN (III): Reuniones y roles Kanban*. Obtenido de jeronimopalacios: <https://jeronimopalacios.com/2016/09/kanban-iii-reuniones-roles-kanban/>
- Pérez, H. (2010). *Propuesta de Análisis y Diseño Basada en UML y UWE para la*. Guatemala: Universidad de San Carlos de Guatemala. Facultad de Ingeniería. Escuela de.

- Pressman, R. S. (2010). *Ingeniería del software* (Septima ed.). (P. Roig Vázquez, Ed.) México, México: Mc Gran Hill.
- Quelca Quispe, V. (2016). *Sistema web de control de compras, ventas e inventarios y verificación de temperatura de medicamentos usando RFID y alarmas tempranas caso: "farmacias la casa de salud"*. La paz: Universidad Mayor de San Andres.
- Rea Cortés, J. (2004). *UML y los procesos de desarrollo*. Puebla: Universidad de las Américas Puebla.
- Rising, L., & Janoff, N. (2000). The Scrum Software Development Process for Small Teams. *IEEE Software*, 17(4), 26 - 32.
- Rubin, K. (2012). *Essential scrum: A practical guide to the most popular agile process*. New York: Addison-Wesley.
- Schwaber, K., & Beedle, M. (2002). *Agile software development with scrum*. New Jersey: Prentice Hall.
- Schwaber, K., & Sutherland, J. (2011). *The Scrum guide*. Recuperado el Marzo de 2018, de <http://www.scrumguides>
- Schwaber, K., & Sutherland, J. (2012). *Software in 30 days: How agile managers beat the odds*. New York: John Wiley & Sons.
- Schwaber, K., & Sutherland, J. (2016). *Scrum Guid*. Londres: Scrum.Org and ScrumInc.
- Scrum Alliance. (2012). *Scrum: the basics*. Recuperado el Marzo de 2018, de Scrum Alliance: <http://www.scrumalliance>.
- Sepulveda Castaño, J. (2016). *Propuesta de aplicación de scrumban para gestionar el proceso de generacion de proyectos de I+ D+ i con el modelo canvas: Estudio Preliminar*. Medellin: UNIVERSIDAD EAFIT.

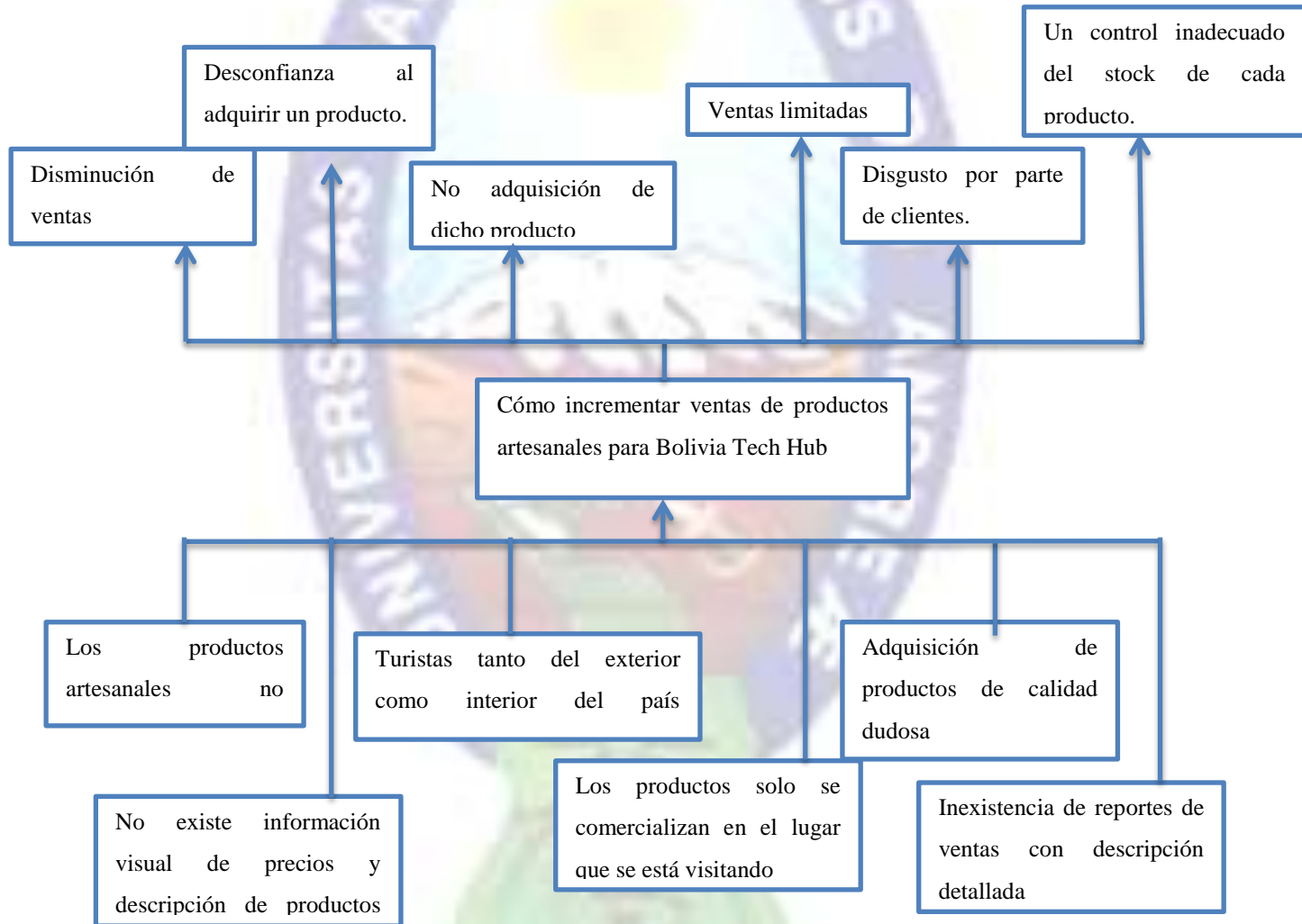
- Sommerville, I. (2010). *Software engineering* (Novena ed.). Boston: Addison Wesley.
- Stellman, A., & Greene, J. (2014). *Learning Agile*. United States of America: O'REILLY.
- Sundén, J., & Hammarberg, M. (2014). *Kanban in Action*. New York: Manning Publications Co.
- Sutherland, J. (2007). *Scrum Tuning: Lessons learned from Scrum implementation at Google*. London.
- Takeuchi, H., & Nonaka, I. (1986). *The new product development game*. *Harvard Business Review*. Cambridge.
- Teniente, E., Olivé, A., Mayol, E., & Gómez, C. (2005). *Diseño de sistemas software en UML*. Barcelona: Universidad Politécnica de Cataluña.
- Truex, D., Baskerville, R., & Travis, J. (2000). *Methodical systems development: The deferred meaning of system development methods*. *Accounting* (Vol. 10). Management and Information.
- Vasquez Rudas, J. F. (2014). *Desarrollo de un sistema de gestión para la venta de pasajes de la empresa Flor Móvil SAC*. Lima: Universidad Inca Garcilaso de la Vega .
- Wingu. (2016). *Manual de Metodologías Ágiles*. Agosto, Argentina: Tecnologías sin fines de lucro.
- Yañez Romero, R. (2017). *Sistema web para el proceso de ventas en la empresa RYSOFT*. Universidad Cesar Vallejo Facultad de Ingeniería. Lima: Universidad Cesar Vallejo.



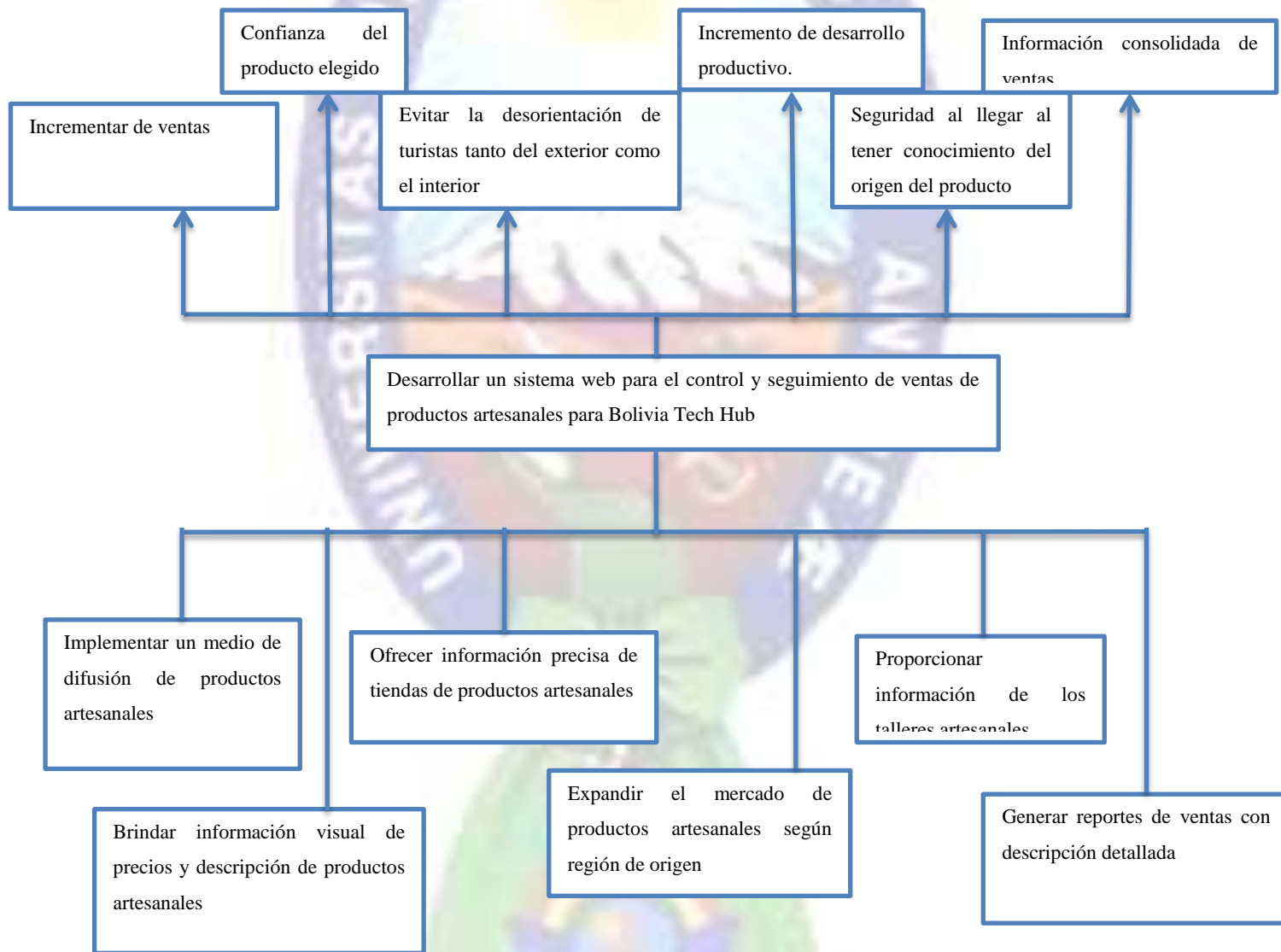


## **ANEXOS**

## ANEXO A – ÁRBOL DE PROBLEMAS



## ANEXO B – ÁRBOL DE OBJETIVOS



### ANEXO C- MARCO LÓGICO

RESUMEN NARRATIVO DE LOS OBJETIVOS	INDICADORES	MEDIOS DE VERIFICACIÓN	SUPUESTOS
<p><b>FIN</b></p> <p>Desarrollar una sistema web, empleando herramientas basadas en laravel 5, Bootstrap, MySQL, para realizar la venta y promoción de productos artesanales.</p>	<ul style="list-style-type: none"> <li>• Productos disponibles en almacén</li> <li>• Consultas sobre accesorios.</li> <li>• Reducción de tiempo de búsqueda de productos</li> </ul>	<ul style="list-style-type: none"> <li>• Sitio indexado en motores de búsqueda</li> <li>• Sitio publicado en internet</li> </ul>	<p>Disponibilidad de un servidor con soporte para PHP y MySQL.</p>
<p><b>PROPÓSITO</b></p> <p>Publicar en internet, información relacionada a productos artesanos. Además del catálogos de productos disponibles para venta</p>	<ul style="list-style-type: none"> <li>• Con el sistema web, se pretende llegar al público en general, y personas aficionadas a las artesanías.</li> <li>• Se promocionan</li> </ul>	<p>Documentos de diseño, manuales de usuario e informes de implementación.</p>	<p>La implementación del sitio web se realizará en un plazo no mayor a 4 meses de iniciado el proyecto.</p>

	nuevos productos lanzados al mercado		
<b>RESULTADOS</b>			
<ul style="list-style-type: none"> <li>• Facilitar la promoción de productos.</li> <li>• Permitir la administración de contenidos vía web.</li> <li>• Emplear nuevas herramientas para el desarrollo de aplicaciones con contenido enriquecido.</li> </ul>	<ul style="list-style-type: none"> <li>• Aplicación web publicada en internet.</li> <li>• Información disponible sobre productos</li> <li>• Los encargados de la venta pueden modificar los contenidos.</li> </ul>	<ul style="list-style-type: none"> <li>• Documentos de diseño</li> <li>• Ayuda en línea, páginas con indicaciones para facilitar la navegación, y la administración del sitio.</li> <li>• Registro visual sobre productos artesanales.</li> </ul>	<ul style="list-style-type: none"> <li>• Se emplean metodologías de diseño web actuales.</li> <li>• Se implementan nuevas herramientas mejorándose el rendimiento y el almacenamiento de los artículos e información publicada en el sitio web.</li> </ul>
<b>ACTIVIDADES</b>		Documentos que registren información relativa a	
<ul style="list-style-type: none"> <li>• Entrevistas con el cliente</li> <li>• Planificación y análisis</li> </ul>	<ul style="list-style-type: none"> <li>• Entrevista con el cliente, 1 día.</li> <li>• Planificación y análisis, 1 semana</li> </ul>	<ul style="list-style-type: none"> <li>• Definición del plan-borrador</li> <li>• Crear el informe de</li> </ul>	<ul style="list-style-type: none"> <li>• Se cuenta con recursos para la elaboración y desarrollo del proyecto.</li> <li>• Existe disposición de</li> </ul>

<ul style="list-style-type: none"> <li>• Especificación de requerimientos</li> <li>• Modelo estructural</li> <li>• Modelo de Hipertexto</li> <li>• Modelo de presentación</li> <li>• Modelo de personalización</li> <li>• Revisar fases de ingeniería.</li> <li>• Implementación</li> <li>• Evaluación del cliente</li> </ul>	<ul style="list-style-type: none"> <li>• Modelo estructural, 1 semana y 4 días</li> <li>• Modelo de hipertexto, 2 semanas y 2 días</li> <li>• Revisión de las fases de ingeniería, 1 semana.</li> </ul>	<p>investigación preliminar</p> <ul style="list-style-type: none"> <li>• Definir los requisitos</li> <li>• Registrar términos en el glosario</li> <li>• Implementar un prototipo</li> <li>• Definir casos de uso</li> <li>• Definir el modelo conceptual-borrador</li> <li>• Definir la arquitectura del sistema-borrador</li> <li>• Refinar el plan</li> <li>• Creación de prototipo antes de la implementación</li> </ul>	<p>las entidades para proporcionar información verídica sobre productos artesanales.</p> <ul style="list-style-type: none"> <li>• Se solicita, un tiempo adicional para revisar la evaluación del cliente, y realizar posibles modificaciones al sistema.</li> </ul>
---	---	---	--



# DOCUMENTACIÓN