

SEP

SECRETARÍA DE
EDUCACIÓN PÚBLICA



TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE APIZACO

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

“DESARROLLO E IMPLEMENTACIÓN DEL CONTROLADOR PARA EL SISTEMA
COTIZADOR DE UNA EMPRESA FABRICADORA DE VIGAS DE MADERA”

TESIS

PARA OBTENER EL GRADO DE:

MAESTRO EN SISTEMAS COMPUTACIONALES

PRESENTA:

LIC. ÁLVARO CUATETA GARCÍA

DIRECTORES:

M. EN C. MARÍA GUADALUPE MEDINA BARRERA

M. EN C. JOSÉ JUAN HERNÁNDEZ MORA

APIZACO, TLAXCALA

AGOSTO 2017

SEP

SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Apizaco

División de Estudios de Posgrado e Investigación

Año del Centenario de la Promulgación de la Constitución Política de los Estados Unidos Mexicanos.

Apizaco, Tlax., 28 de Agosto de 2017

No. de Oficio: DEPI/368/17

ASUNTO: Se Autoriza Impresión de Tesis de Grado.

LIC. ALVARO CUATETA GARCÍA
CANDIDATO AL GRADO DE MAESTRO
EN SISTEMAS COMPUTACIONALES
No. de Control: M06370497
P R E S E N T E.

Por este medio me permito informar a usted, que por aprobación de la Comisión Revisora asignada para valorar el trabajo, mediante la Opción: I **Tesis de Grado por Proyecto de Investigación**, de la **Maestría en Sistemas Computacionales**, que presenta con el tema: "DESARROLLO E IMPLEMENTACIÓN DEL CONTROLADOR PARA EL SISTEMA COTIZADOR DE UNA EMPRESA FABRICADORA DE VIGAS DE MADERA" y conforme a lo establecido en el Procedimiento para la Obtención del Grado de Maestría en el Instituto Tecnológico, la División de Estudios de Posgrado e Investigación a mi cargo le emite la:

AUTORIZACIÓN DE IMPRESIÓN


Debiendo entregar un ejemplar del mismo debidamente encuadernado y seis copias en CD en formato PDF, para presentar su Acto de Recepción Profesional a la brevedad.

Sin otro particular por el momento, le envío un cordial saludo.

ATENTAMENTE

PENSAR PARA SERVIR, SERVIR PARA TRIUNFAR®




DR. JOSÉ FEDERICO CASCO VÁSQUEZ
JEFE DE LA DIVISIÓN DE ESTUDIOS DE
POSGRADO E INVESTIGACIÓN.

SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL
DE MÉXICO
INSTITUTO TECNOLÓGICO DE APIZACO
DIVISIÓN DE ESTUDIO
DE POSGRADO E INVESTIGACIÓN

JFCV/MJSH*mebr.

C.p. Expediente.

"Año del Centenario de la Promulgación de la Constitución Política de los Estados Unidos Mexicanos"

Apizaco, Tlax., 25 de Agosto de 2017

ASUNTO: Aprobación del trabajo de Tesis de Maestría.

DR. JOSÉ FEDERICO CASCO VÁSQUEZ
JEFE DE LA DIVISIÓN DE ESTUDIOS DE
POSGRADO E INVESTIGACIÓN.
P R E S E N T E.

Por este medio se le informa a usted, que los integrantes de la **Comisión Revisora** para el trabajo de tesis de maestría que presenta el **LIC. ALVARO CUATETA GARCÍA**, con número de control **M06370497**, candidato al grado de **Maestro en Sistemas Computacionales** y egresado del **Instituto Tecnológico de Apizaco**, cuyo tema es "**DESARROLLO E IMPLEMENTACIÓN DEL CONTROLADOR PARA EL SISTEMA COTIZADOR DE UNA EMPRESA FABRICADORA DE VIGAS DE MADERA**", fue:

A P R O B A D O

Lo anterior, al valorar el trabajo profesional presentado por el candidato y constatar que las observaciones que con anterioridad se le marcaron así como correcciones sugeridas para su mejora ya han sido realizadas.

Por lo que se avala se continúe con los trámites pertinentes para su titulación.

Sin otro particular por el momento, le envió un cordial saludo.

LA COMISIÓN REVISORA

M.C. MARÍA GUADALUPE MEDINA BARRERA

M.C.C. MARÍA JANAI SÁNCHEZ HERNÁNDEZ

M.C. JOSÉ JUAN HERNÁNDEZ MORA

M.C. JUAN RAMOS RAMOS

C. p.- Interesado.

Dedicatoria

El presente trabajo lo dedico a mis dos hijos Estela Yamilet y Axel, que son la razón de mi vida el tesoro más grande que Dios me regaló y el motivo de mí existir.

A mi amada esposa, por su apoyo y ánimo que me brinda día con día para alcanzar nuevas metas, tanto profesionales como personales.

A mis padres Alvaro Reynaldo y Eva por ser el pilar fundamental en todo lo que soy, en toda mi educación, tanto académica, como de la vida, por su incondicional apoyo perfectamente mantenido a través del tiempo.

Agradecimientos

Agradezco al Consejo Nacional de Ciencia y Tecnología por todo el apoyo que me brindo para mi preparación.

Agradezco al Instituto Tecnológico de Apizaco que me permitió superarme tanto en el ámbito profesional y personal brindándome las herramientas y conocimientos que hoy son de suma importancia para el desarrollo de mis actividades.

A mis directos de tesis la M. en C María Guadalupe Medina Barrera y mi codirector el M. en C. José Juan Hernández Mora, quienes me guiaron, aconsejaron y brindaron todo el apoyo necesario para llevar a cabo este trabajo. Les agradezco su paciencia, buenos consejos y la confianza que depositaron en mí.

A todos mis distinguidos maestros de la Maestría en Sistemas Computacionales del Instituto Tecnológico de Apizaco que mediante su entusiasmo y enseñanzas han contribuido en mi formación académica.

A mis compañeros María de los Ángeles, Luis, Edgar, Diego y Oscar por confiar en mí y haber hecho de mi etapa de maestría un trayecto de vivencias que nunca olvidare.

A mi esposa por entenderme en todo, gracias a ella porque en todo momento fue un apoyo incondicional en mi vida, fue la felicidad encajada en una sola persona, fue mi todo reflejado en otra persona a la cual amo demasiado, y por la cual estoy dispuesto a enfrentar todo y en todo momento.

Resumen

Hoy en día es muy común contar con un software para la optimización de los procesos en las empresas, ya que aumenta la calidad y competitividad en el trabajo. En esta tesis se presenta la solución a un problema de una empresa fabricante de vigas de madera por lo que se tiene la necesidad de un proyecto tecnológico.

Lo anterior implica el desarrollo de un software cotizador creado a la medida para automatizar la realización de cotizaciones para sus clientes, siendo este el primer cotizador en su clase por los cálculos personalizados para sus productos, además de la optimización de las cotizaciones.

El beneficio de este sistema es agilizar el proceso para una cotización y así poder obtener proyectos de forma oportuna.

Aplicando diferentes metodologías como ágiles SCRUM y espiral, en conjunto con el patrón de diseño MVC se desarrolló el módulo controlador y la integración de los demás componentes, dando como resultado un software funcional para la empresa. Al sistema se le aplican una serie de pruebas para encontrar posibles errores en el cotizador, y a su vez mejorar la calidad del mismo, el sistema está desarrollado en lenguaje java y Frameworks como JSF, Hibernate y Primefaces.

Abstract

Nowadays it is very common to have a software for the optimization of the processes in the companies, since it increases the quality and competitiveness in the work. In this thesis presents the solution to a problem of a company of manufactures wooden beams so that it has the need of a technological project.

This implies the development of a software quote created to automate the execution of quotes for your customers, this being the first in its class by custom calculations for their products, in addition to the optimization of the contributions. The benefit of this system is to streamline the process for a quote so that projects can be obtained in a timely manner.

Applying different methodologies such as SCRUM and spiral, in conjunction with the MVC design pattern the controller module and the integration of the other components were developed in this work, resulting in functional software for the company. The system is applied a series of tests to find possible errors in the quote, and in turn improve the quality of it, the system is developed in Java language and Frameworks such as JSF, Hibernate and Primefaces

Índice

| | |
|---|------------|
| RESUMEN | I |
| ABSTRACT | II |
| ÍNDICE | III |
| ÍNDICE DE FIGURAS | VII |
| ÍNDICE DE TABLAS | IX |
| 1 INTRODUCCIÓN | 12 |
| 1.1 DESCRIPCIÓN DEL PROBLEMA | 13 |
| 1.2 OBJETIVO GENERAL..... | 14 |
| 1.3 OBJETIVOS ESPECÍFICOS..... | 14 |
| 1.4 ALCANCES | 14 |
| 1.5 LIMITACIONES | 15 |
| 1.6 JUSTIFICACIÓN..... | 15 |
| 1.7 PREGUNTA DE INVESTIGACIÓN | 16 |
| 1.8 ORGANIZACIÓN DEL CONTENIDO | 16 |
| 1.9 ESTADO DEL ARTE | 17 |
| 1.10 ANÁLISIS COMPARATIVO | 32 |
| 1.11 COMPARACIÓN DE SISTEMAS DE COTIZACIÓN EN EL MERCADO..... | 33 |
| 1.12 ANÁLISIS DEL ESTADO DEL ARTE | 36 |
| 1.13 COMENTARIOS | 39 |
| 2 FUNDAMENTOS | 40 |
| 2.1 EL PATRÓN DE DISEÑO MODELO VISTA CONTROLADOR (MVC) | 40 |
| 2.2 EL COMPONENTE CONTROLADOR | 41 |
| 2.3 INGENIERÍA DE SOFTWARE | 43 |
| 2.4 MODELO DEL PROCESO DE SOFTWARE..... | 44 |

| | | |
|-------|---|----|
| 2.4.1 | <i>El modelo en cascada</i> | 44 |
| 2.4.2 | <i>El modelo incremental</i> | 46 |
| 2.4.3 | <i>El modelo construcción de prototipos</i> | 47 |
| 2.4.4 | <i>El modelo DRA</i> | 48 |
| 2.4.5 | <i>El modelo lineal secuencial</i> | 50 |
| 2.4.6 | <i>El modelo espiral</i> | 50 |
| 2.5 | METODOLOGÍAS ÁGILES | 52 |
| 2.5.1 | <i>Metodología ágil DAS</i> | 54 |
| 2.5.2 | <i>Metodología ágil MDSD</i> | 55 |
| 2.5.3 | <i>Metodología ágil Cristal</i> | 55 |
| 2.5.4 | <i>Metodología ágil Scrum</i> | 56 |
| 2.5.5 | <i>Roles</i> | 58 |
| 2.5.6 | <i>Los artefactos</i> | 59 |
| 2.5.7 | <i>El Incremento</i> | 61 |
| 2.5.8 | <i>Los Eventos</i> | 61 |
| 2.5.9 | <i>Las herramientas</i> | 63 |
| 2.6 | INGENIERÍA DE REQUERIMIENTOS | 65 |
| 2.6.1 | <i>Identificación de los casos de uso</i> | 67 |
| 2.6.2 | <i>Descripción de los actores</i> | 69 |
| 2.6.3 | <i>Reglas del negocio</i> | 70 |
| 2.6.4 | <i>Requerimientos funcionales</i> | 71 |
| 2.6.5 | <i>Requerimientos no funcionales</i> | 73 |
| 2.7 | MODELADO | 76 |
| 2.8 | CONCEPTOS DE DISEÑO | 77 |
| 2.9 | EVALUACIÓN DE SOFTWARE | 78 |
| 2.9.1 | <i>Calidad</i> | 79 |
| 2.9.2 | <i>Modelo en V</i> | 79 |
| 2.9.3 | <i>Modelo en espiral</i> | 82 |

| | | |
|----------|--|------------|
| 2.10 | COTIZADOR WEB | 83 |
| 2.11 | TECNOLOGÍAS DE DESARROLLO JAVA | 85 |
| 2.12 | COMENTARIOS | 86 |
| 3 | METODOLOGÍA | 88 |
| 3.1 | METODOLOGÍA GENERAL DEL SISTEMA | 88 |
| 3.1.1 | <i>Modelo Vista Controlador</i> | <i>88</i> |
| 3.2 | SCRUM APLICADO AL SISTEMA | 90 |
| 3.2.1 | <i>Definición de los roles</i> | <i>90</i> |
| 3.2.2 | <i>Seguimiento del Sprint.....</i> | <i>92</i> |
| 3.2.3 | <i>Revisión del Sprint.....</i> | <i>92</i> |
| 3.2.4 | <i>Planeación del Sprint</i> | <i>94</i> |
| 3.2.5 | <i>Gráfica de producto.....</i> | <i>98</i> |
| 3.3 | METODOLOGÍA IMPLEMENTADA AL CONTROLADOR..... | 99 |
| 3.4 | METODOLOGÍA DE DESARROLLO SCRUM | 100 |
| 3.5 | COMENTARIOS | 103 |
| 4 | DESARROLLO DEL CONTROLADOR | 104 |
| 4.1 | DESARROLLO DEL SPRINT | 104 |
| 4.1.1 | <i>Sprint 1 Recolección de la información.....</i> | <i>104</i> |
| 4.1.2 | <i>Sprint 2 Administración y validación de usuarios.....</i> | <i>107</i> |
| 4.1.3 | <i>Sprint 3 Administración de clientes y productos</i> | <i>109</i> |
| 4.1.4 | <i>Sprint 4 Administración de madera y materiales</i> | <i>114</i> |
| 4.1.5 | <i>Sprint 5 Administrar tratamiento y acabado.....</i> | <i>118</i> |
| 4.1.6 | <i>Sprint 6 Reporte de cotizaciones</i> | <i>122</i> |
| 4.1.7 | <i>Sprint 7 Cotización</i> | <i>125</i> |
| 4.1.8 | <i>Sprint 8 Eliminar Cotización.....</i> | <i>127</i> |
| 4.2 | COMENTARIOS | 130 |
| 5 | PRUEBAS Y RESULTADOS..... | 131 |

| | | |
|----------|--|------------|
| 5.1 | DESCRIPCIÓN DEL PLAN DE PRUEBAS | 131 |
| 5.2 | MÓDULO VALIDACIÓN DE USUARIOS | 132 |
| 5.3 | MÓDULO ADMINISTRACIÓN DE USUARIOS | 134 |
| 5.4 | MÓDULO ADMINISTRACIÓN DE CLIENTES | 135 |
| 5.5 | MÓDULO ADMINISTRACIÓN DE PRODUCTOS | 136 |
| 5.6 | MÓDULO ADMINISTRACIÓN DE MATERIALES | 137 |
| 5.7 | MÓDULO ADMINISTRACIÓN DE MADERA..... | 138 |
| 5.8 | MÓDULO ADMINISTRACIÓN DE TRATAMIENTOS..... | 139 |
| 5.9 | MÓDULO ADMINISTRACIÓN DE ACABADOS | 140 |
| 5.10 | MÓDULO ADMINISTRACIÓN DE PERSONAL..... | 141 |
| 5.11 | MÓDULO DE REPORTES | 142 |
| 5.12 | MÓDULO COTIZACIÓN | 143 |
| 5.13 | MÓDULO NUEVA COTIZACIÓN | 144 |
| 5.14 | COMENTARIOS | 147 |
| 6 | CONCLUSIONES Y RECOMENDACIONES | 148 |
| 6.1 | CONCLUSIONES | 148 |
| 6.2 | TRABAJOS FUTUROS | 150 |
| | BIBLIOGRAFÍA | 151 |
| | ANEXOS | 156 |
| A. | PLAN DE PRUEBAS | 157 |
| B. | CARTA DE LIBERACIÓN DE LA ESTANCIA Y CARTA DE SATISFACCIÓN DEL CLIENTE DONDE AVALA EL TRABAJO REALIZADO..... | 184 |
| C. | PUBLICACIONES DE ARTÍCULOS..... | 187 |

Índice de figuras

| | |
|---|-----|
| Figura 2.1 Proceso del modelo vista controlador (Cervantes, 2013)..... | 43 |
| Figura 2.2 El modelo en cascada (Sommerville, 2011). | 44 |
| Figura 2.3 El modelo incremental (Pressman, 2010). | 46 |
| Figura 2.4 Paradigma de construcción de prototipos (Pressman, 2010)..... | 48 |
| Figura 2.5 El modelo DRA (Pressman, 2010). | 49 |
| Figura 2.6 Modelo lineal secuencia (Pressman, 2010)..... | 50 |
| Figura 2.7 Un modelo espiral típico (Pressman, 2010). | 51 |
| Figura 2.8 Desarrollo de las principales metodologías (Switzerland, 2015) | 53 |
| Figura 2.9 Diagrama del ciclo iterativo Scrum (Manager, 2015)..... | 58 |
| Figura 2.10 Ejemplo de grafica del producto (Palacio, 2015)..... | 63 |
| Figura 2.11 Grafica de avance (Palacio, 2015) | 64 |
| Figura 2.12 Proceso en espiral para la ingeniería de requerimientos (Sommerville, 2011) | 67 |
| Figura 2.13 Tipos de requerimientos no funcionales (Sommerville, 2011)..... | 74 |
| Figura 2.14 Modelo en V (TutorialsPoint, 2017)..... | 80 |
| Figura 2.15 Estrategia de pruebas (Pressman, 2010)..... | 82 |
| Figura 3.1 Patrón de diseño Modelo Vista Controlador | 89 |
| Figura 3.2 Metodología general para el desarrollo del sistema basado en Scrum.... | 91 |
| Figura 3.3 Gráfica del producto | 99 |
| Figura 3.4 Modelo vista controlador del cotizador | 99 |
| Figura 3.5 Metodología de desarrollo centrado en el controlador | 101 |
| Figura 4.1 Caso de uso del cotizador..... | 106 |
| Figura 4.2 Diagrama de caso de uso de alta de clientes..... | 111 |
| Figura 4.3 Diagrama de caso de uso de alta de productos | 113 |
| Figura 4.4 Diagrama de caso de uso de alta de madera..... | 115 |
| Figura 4.5 Caso de uso para alta de materiales..... | 117 |
| Figura 4.6 Diagrama de caso de uso de alta de un tratamiento..... | 120 |
| Figura 4.7 Caso de uso para alta de acabados..... | 122 |

| | |
|---|-----|
| Figura 4.8 Diagrama de caso de uso del reporte de cotización..... | 123 |
| Figura 4.9 Diagrama de caso de uso del pie tablón | 124 |
| Figura 4.10 Diagrama de caso de uso de cotización..... | 127 |
| Figura 4.11 Diagrama de caso de uso de eliminación de cotizador | 129 |
| Figura 5.1 Interfaz: Inicio de sesión..... | 132 |
| Figura 5.2 Bienvenida del sistema | 133 |
| Figura 5.3 Administración de usuarios | 134 |
| Figura 5.4 Interfaz: Administración de clientes | 135 |
| Figura 5.5 Interfaz: Administración de productos | 136 |
| Figura 5.6 Interfaz: Administración de materiales | 137 |
| Figura 5.7 Interfaz: Administración de madera..... | 138 |
| Figura 5.8 Interfaz: Administración de tratamientos | 139 |
| Figura 5.9 Interfaz: Administración de acabados | 140 |
| Figura 5.10 Interfaz: Administración de personal | 141 |
| Figura 5.11 Interfaz: reportes de cotizaciones..... | 142 |
| Figura 5.12 Interfaz: Administración de cotizaciones | 143 |
| Figura 5.13 Interfaz: Nueva cotización | 145 |
| Figura 5.14 Interfaz: agregar productos | 145 |
| Figura 5.15 Interfaz: resumen de cotización | 146 |

Índice de tablas

| | |
|--|-----|
| Tabla 1.1 Comparación de los trabajos relacionados..... | 32 |
| Tabla 1.2 Comparación de los sistemas en el mercado..... | 33 |
| Tabla 2.1 Requerimientos funcionales | 68 |
| Tabla 2.2 Descripción de los autores | 69 |
| Tabla 2.3 Reglas del negocio | 71 |
| Tabla 2.4 Descripción detallada de los casos de uso..... | 72 |
| Tabla 2.5 Requerimientos no funcionales del producto..... | 75 |
| Tabla 2.6 Requerimientos no funcionales de organización | 75 |
| Tabla 2.7 Requerimientos no funcionales externos..... | 75 |
| Tabla 3.1 Pila del Sprint | 92 |
| Tabla 3.2 Pila del producto..... | 94 |
| Tabla 3.3 Reglas del negocio | 95 |
| Tabla 4.1 Sprint Obtención de los requisitos..... | 104 |
| Tabla 4.2 Sprint validación de usuarios..... | 107 |
| Tabla 4.3 Descripción de validación de usuarios | 108 |
| Tabla 4.4 Sprint Administración de clientes..... | 109 |
| Tabla 4.5 Descripción del controlador para la administración de clientes | 110 |
| Tabla 4.6 Sprint Administración de productos | 112 |
| Tabla 4.7 Descripción del controlador para la administración de Productos | 112 |
| Tabla 4.8 Administración de madera..... | 114 |
| Tabla 4.9 Descripción del controlador para la administración de madera | 114 |
| Tabla 4.10 Sprint Administración de materiales | 116 |
| Tabla 4.11 Descripción del controlador para la administración de materiales..... | 116 |
| Tabla 4.12 Administración de tratamientos | 118 |
| Tabla 4.13 Descripción del controlador para la administración de tratamiento ... | 118 |
| Tabla 4.14 Sprint Administración de acabados | 120 |
| Tabla 4.15 Descripción del controlador para la administración de acabados..... | 121 |
| Tabla 4.16 Sprint Generar reportes de cotización | 122 |

| | |
|---|-----|
| Tabla 4.17 Descripción del controlador para los reportes de cotización | 123 |
| Tabla 4.18 Sprint Generar reportes del pie tablón..... | 124 |
| Tabla 4.19 Descripción del controlador para los reportes del pie tablón | 125 |
| Tabla 4.20 Sprint Generar una cotización. | 125 |
| Tabla 4.21 Descripción del controlador para la administración de las cotizaciones | 126 |
| Tabla 4.22 Sprint Eliminación de cotizador | 128 |
| Tabla 4.23 Descripción del controlador para la eliminación de una cotización ... | 128 |
| Tabla 5.1 Prueba de integración: Inicio de sesión..... | 133 |
| Tabla 5.2 Administración de usuarios | 134 |
| Tabla 5.3 Administración de clientes | 135 |
| Tabla 5.4 Administración de productos | 136 |
| Tabla 5.5 Administración de materiales | 137 |
| Tabla 5.6 Administración de madera..... | 138 |
| Tabla 5.7 Administración de tratamientos | 139 |
| Tabla 5.8 Administración de acabados | 140 |
| Tabla 5.9 Administración de personal | 141 |
| Tabla 5.10 Reportes | 142 |
| Tabla 5.11 Administración de cotizaciones | 144 |
| Tabla 5.12 Nueva cotización | 146 |

Acrónimos

GUI Interfaz Gráfica de Usuario.

MVC Modelo Vista Controlador.

SCRUM Framework de desarrollo ágil

XP Programación extrema

XML Lenguaje de marcas extensible (extensible markup language).

Servlet Pequeño programa que se ejecuta en el navegador.

HTTP Protocolo de transferencia de hipertexto.

JSF JavaServer Faces

1 Introducción

La empresa fabricante de vigas de madera laminada del estado de Tlaxcala, tiene entre sus actividades el proceso de construcción y manufactura de piezas de madera, de igual manera se realizan proyectos de construcción para diferentes empresas en distintos estados del país y otros países tales como Alemania e Italia, entre otros.

Dicha empresa requiere automatizar la realización de cotizaciones para sus clientes, por lo que se propone la creación de un sistema web, que permita emitir cotizaciones de forma rápida y efectiva a sus clientes.

En la actualidad se utiliza una hoja de cálculo, sin embargo, se requiere mejorar el proceso, ya que los encargados de convencer al cliente, requieren un sistema en donde se apoyen para llevar a cabo dicha acción en línea.

El sistema a crear pretende establecer en un servidor el acceso en los diferentes puntos del país, sin importar donde se encuentre el operador de dicho sistema a través del servicio de Internet.

Se contempla un sistema con niveles de seguridad de usuarios y contraseñas para el control de las ventas. Los usuarios que se contemplan:

- El operador de ventas.
- El usuario administrado.
- El usuario para reportes.

El operador se encargará de llevar a cabo las cotizaciones para algún cliente interesado, este sin permisos para hacer alguna modificación a los materiales o insumos.

Por otro lado, el usuario administrador puede llevar a cabo cotizaciones además de tener acceso de alta, edición y eliminación de un producto o servicio y el usuario para reporte solo tiene la capacidad de generar reportes.

El sistema contará con un módulo que proporciona los reportes necesarios para cada usuario. Ambos usuarios podrán generar reportes necesarios y de acuerdo a requerimientos específicos.

1.1 Descripción del problema

Bajo el diseño de un proyecto tecnológico, se identifica la necesidad de desarrollar la parte controlador del patrón de diseño MVC, para la aplicación que será capaz de realizar las cotizaciones a la medida de proyectos que fluyen dentro del negocio de Metal y Madera.

El controlador se sincronizará con la base de datos, donde se guardarán los registros necesarios para realizar la cotización correspondiente.

El controlador interactuará con el módulo de la interfaz web, para realizar las operaciones desde cualquier sitio con acceso a internet, se podrán realizar reportes de producción, de venta, así como también acciones de insertar, modificar y eliminar materiales y servicios, además podrán hacer búsquedas etc.

La aplicación tendrá el acceso restringido para los distintos usuarios, se contemplan cuatro distintos tipos usuario: el administrador, jefe de área, el operador y usuario.

- El administrador, este usuario posee todos los permisos del sistema, con la finalidad de realizar mantenimiento al mismo.
- Jefe de área, este usuario tiene la capacidad de aplicar descuentos y modificaciones en los productos e insumos, además de poder generar reportes específicos.

- Operador, es el responsable de realizar las cotizaciones y reportes necesarios para las ventas o adquisiciones de proyectos.
- Usuario, este sólo es responsable de generar reportes.

La aplicación a desarrollar podrá ser programada para ejecutar estas tareas cuando sea solicitado en cualquier lugar.

1.2 Objetivo General

Desarrollar bajo el modelo MVC el módulo controlador de una aplicación web COTIZADOR que permita la integración del modelo de datos y la vista, para cotizar sus productos y servicios de la empresa.

1.3 Objetivos específicos

- ✓ Configurar los Frameworks de JavaServer Faces e Hibérnate para simplificar el desarrollo en NetBeans.
- ✓ Realizar la conexión para el mapeo de los datos de Mysql con Hibérnate.
- ✓ Desarrollar las distintas clases en java para los eventos y las peticiones en el entorno NetBeans.
- ✓ Integrar los diferentes módulos (modelo y vista) para implementar el sistema bajo el MVC.
- ✓ Realizar un conjunto de pruebas de funcionamiento de un sistema integral, bajo una metodología en espiral.

1.4 Alcances

- ✓ Diseño y desarrollo del controlador para la herramienta de cotización.

- ✓ Diseño del sistema web (Cotizador).
- ✓ Obtener la información actualizada que se requiera de una forma rápida y oportuna.

1.5 Limitaciones

En este proyecto se comprende el desarrollo del módulo controlador en el modelo MVC y su integración con la vista y el modelo. La base de datos y la interfaz no forman parte de este trabajo.

1.6 Justificación

La empresa en la actualidad trabaja con una hoja de cálculo, que sirve de apoyo para generar una cotización, sin embargo, tiene como desventaja que este proceso se realiza de forma manual y presenta resultados con datos incompletos. Se realiza un análisis detallado a la hoja de cálculo, poniendo énfasis en las operaciones que realiza para obtener los datos necesarios y generar una cotización. La hoja de cálculo presenta cierta variación en los datos y estos afectan directamente la oferta, lo que conlleva a que se realice un trabajo extra para llegar a una oferta exacta y competitiva.

Por lo anterior se requiere un sistema que sea capaz de realizar las cotizaciones con el mínimo trabajo posible por parte del operador, además el sistema tendrá la capacidad de generar las cotizaciones en el menor tiempo posible y así poder trabajar con distintos proyectos al mismo tiempo.

Aplicando un patrón de diseño MVC (Modelo-Vista-Controlador) hacia servicios web, para dividir y facilitar las tareas y así poder desarrollar la herramienta correspondiente utilizando Frameworks.

Trabajando conjuntamente con una metodología ágil la cual permitirá el desarrollo de la parte controlador, que cubrirá las necesidades para garantizar la fiabilidad entre los módulos, agilizará la cotización a la medida de la empresa y proporcionará información correcta y exacta.

La implementación de los distintos Framework tiene como objetivo una aplicación que cumpla con las tareas a realizar, utilizando tecnologías a la vanguardia y buscando una simplicidad y escalabilidad de las distintas partes del modelo MVC.

1.7 Pregunta de investigación

¿Es posible implementar un módulo controlador, bajo el modelo MVC que integre efectivamente los componentes de un sistema cotizador web?

1.8 Organización del contenido

A continuación, se presenta de manera general la estructura de este trabajo:

En el Capítulo I Introducción, se describe en forma general cómo es el planteamiento del problema, su justificación, objetivos, alcances, limitaciones y la pregunta del proyecto.

Así mismo se presenta el estado del arte y su análisis.

En el Capítulo II Fundamentos, se presenta una descripción de los conceptos básicos del patrón de diseño Modelo Vista Controlador, así como las herramientas, Hibernate y JSF.

Además, metodologías ágiles, tales como Scrum.

En el Capítulo III Metodología, se muestran las consideraciones que se tomaron en cuenta para desarrollar el controlador para el sistema cotizador.

En el Capítulo IV Desarrollo del controlador, se describe el desarrollo del controlador para el sistema cotizador, siguiendo la metodología elegida para la aplicación del Modelo-Vista-Controlador.

En el Capítulo V Pruebas y resultados, en este capítulo se muestran los resultados obtenidos de la implementación e integración del sistema cotizador.

Y por último en el **Capítulo VI** Conclusiones y recomendaciones, se presentan las conclusiones alcanzadas y las recomendaciones para futuros trabajos. Así como las pruebas realizadas para su funcionamiento.

Anexo A: Plan de pruebas.

Anexo B: Carta de liberación de la estancia y carta de satisfacción del cliente donde avala el trabajo realizado.

Anexo C: Publicaciones de artículos.

1.9 Estado del arte

En este apartado se describen los trabajos relacionados con este tipo de aplicación,

Self-Organizing Roles on Agile Software Development Teams (Hoda, Noble, & Marshall, 2013)

A pesar de la larga y rica historia de los equipos de auto-organización y de su reciente popularidad con los métodos ágiles, se tiene poca investigación sobre el tema dentro Ingeniería de software, la comprensión de estos papeles ayudará a los equipos de desarrollo de software y sus gerentes comprender y ejecutar sus funciones y responsabilidades como un equipo de auto-organización.

A través de una investigación de la teoría fundamentada participaron 58 profesionales ágiles de 23 organizaciones de software en Nueva Zelanda y la India durante un período de cuatro años, se identificaron de manera informal, implícita, transitoria, las funciones espontáneas que hacen los equipos Ágiles auto-organización.

Roles de equipo Agile auto-organización aseguran que un solo equipo es capaz de lograr y mantener la auto-organización por atender a las diferentes necesidades del equipo, tales como la necesidad de formación, el apoyo de la alta dirección, la implicación del cliente, etc.

Equipos auto-organizados están en el centro de software Ágil de desarrollo. La falta de investigación de auto-organización equipos de ingeniería de software significa que hay poca evidencia para explicar cómo los equipos ágiles se organizan en la práctica.

Efectos de la experiencia de desarrollo en el Aprendizaje y la Aplicación del desarrollo guiado por pruebas de software. (Latorre, 2014)

En los últimos años, los investigadores han llevado a cabo varios estudios dentro de la academia y la industria sobre la eficacia de esta práctica de desarrollo de software. Se han investigado su utilidad en comparación con otras técnicas de desarrollo, centrándose principalmente en la calidad del código y la productividad.

Desde el punto de vista de la investigación, el objetivo es evaluar qué difícil es aprender UTDD por profesionales sin ninguna experiencia previa en esta técnica. Desde el punto de vista industrial, el objetivo es evaluar la posibilidad de utilizar esta práctica de desarrollo de software como una solución efectiva para tener en cuenta en proyectos reales.

Los desarrolladores calificados con los conocimientos adecuados y sin ningún tipo de experiencia previa en el primer test de desarrollo pueden aprender rápidamente las

reglas UTDD y ellos, después de practicar durante un corto espacio de tiempo, aplicar correctamente en pequeñas tareas de programación.

El estudio indica que sólo dos temas lograron una conformidad UTDD superior al 90 por ciento durante el desarrollo de los requisitos finales, a diferencia de lo que sucedió durante los requisitos iniciales, donde incluso algunos de los participantes mostraron un 100 por ciento de cumplimiento del proceso.

Un enfoque integrado de MAS-CommonKADS, Modelo–Vista–Controlador web y optimización de la aplicación de estrategias para la web basado en sistema experto de desarrollo. (Hasan & R.K., 2011)

Se presenta un marco para el desarrollo de sistemas expertos basados en la web con enfoque integrado de MAS-Common KADS agente metodología orientada, modelo, vista y Controlador (MVC), arquitectura de aplicaciones web y de optimizar las estrategias para lidiar con la complejidad del desarrollo y para lograr una alta usabilidad.

El estudio servirá de guía en la ingeniería de software de desarrollo de dicho sistema en relación con las tecnologías de Internet.

Sin embargo, las características disponibles en las aplicaciones web especialmente accesibilidad remota, actualización centralizada, minimizado en dependencia del sistema de cliente final, el costo del servicio y multi-plataforma disponibilidad no puede ser ignorada. Eficiencia de aplicación web dependen de muchos factores como; tecnología de servidor web, servidor de carga y eficiencia de procesamiento, velocidad en el canal de comunicación entre el equipo cliente y el servidor web; eficiencia de cliente propio ordenador; y también la complejidad de la aplicación web.

Hay muchos factores limitantes que influyen en la complejidad del software bajo desarrollo como negocio/requisitos de proceso, la metodología adoptada, ingeniero de software, herramientas de aplicación utilizan conocimientos y también los requisitos del usuario final.

Diseño de un Modelo MVC para el Desarrollo Rápido de Aplicaciones Web, (Dragos & Altar, 2014)

El desarrollo de aplicaciones Web ha recorrido un largo camino desde el comienzo de la World Wide Web. Una miríada de tecnologías y lenguajes de programación se utilizan ahora para construir aplicaciones web. El entorno web actual utiliza HTML y CSS para presentar los datos a los usuarios y la interacción se realiza a través de JavaScript. Estas tecnologías son llamadas "front-end" o "client-side".

Front end y back end convergen las tecnologías para construir aplicaciones web, sino porque la World Wide Web ha evolucionado a un ritmo rápido y porque los desarrolladores deben utilizar un número bastante amplio de tecnologías para crear una sola aplicación web, el resultado de su trabajo es a menudo difícil de mantener y reparar.

Su plataforma propuesta unifica todos los aspectos mencionados anteriormente en un solo paquete que es robusto y fácil de utilizar y mantener.

La estructura superior puede describirse como sigue. La fuente de datos puede ser cualquier tipo de base de datos. El sistema ORM es un conjunto de clases construido específicamente para más conocidos sistemas de administración de bases de datos. Una clase principal proporciona la cartografía básica, las relaciones y la gestión de un tipo general de tabla en el DBMS deseado. Las relaciones deben ser asignadas como para permitir la obtención de datos relacionados en una forma orientada a objetos, sin la necesidad de realizar complejas consultas escritas por el desarrollador. Ambos impacientes y la carga diferida de datos deben ser considerados.

Los modelos están contruidos como los niños de la clase que hereda la funcionalidad de ORM principales de esa clase principal. Además, los modelos están enriquecidos con capacidades únicas que describen el objeto de la vida real que debe ser modelada.

El controlador es una clase base que maneja el modelo de seguridad de acceso a la consulta, la ruta y la solución. Controladores están incorporados como hijos del controlador básico y heredar la funcionalidad y están enriquecidos con la lógica de la aplicación correspondiente a las acciones deseadas

La dependencia de los componentes MVC arquitectura distribuida. (Hamid Mcheick, 2011)

Este artículo se indica el papel tan importante que tienen los conectores en la arquitectura distribuida. Ellos son el módulo de software para describir la dependencia de componentes y proporcionar un canal para vincular componentes juntos.

En particular, los conectores desempeñar un papel significativo en el sistema distribuido. En este estudio se enfocó en especial en el diseño del conector de Model-View-Controller (MVC) arquitectura distribuida.

Modelo, Vista y Controlador (MVC), la arquitectura de un sistema se divide en tres tipos de módulos: controlador, modelo y Vista.

Es muy importante que la vista está separada de la estructura de datos del sistema, el controlador se implementa como un componente independiente o se combina con la vista, el modelo se ocupa de la lógica y los datos de la aplicación, y también es responsable de la actualización de la información de vista y recibir los comandos del controlador; la vista es responsable de la presentación; el controlador está a cargo de las aportaciones de los usuarios, incluidos los eventos de teclado o ratón, y notifica al modelo mediante eventos.

En arquitectura distribuida, el diseño de los conectores de MVC hace que los diseños de la arquitectura de software se vuelven más y más complicado, porque los componentes (M-V-C) pueden ser distribuidos en distintos equipos a través de redes y algunos de los planteamientos tradicionales, tales como el método basado en un mecanismo, no están directamente disponibles para interactuar con los componentes distribuidos.

MVC clásico puede ser aplicado con frecuencia a las aplicaciones de escritorio que se ejecutan en un equipo local. Los conectores entre los componentes se denominan como llamada de método (o llamada a función). Por lo tanto, este tipo de MVC es también llamado método MVC.

Una evaluación sobre los componentes del modelo de fábrica en función de la experiencia adquirida en el proceso de Ingeniería de Requisitos: UN ESTUDIO PRELIMINAR. (Mastura & Rusli, 2014)

Este artículo se hace mención de la importancia del desarrollo del software.

En la ingeniería de software, es rápidamente conocimientos acumulados y cambiado paralelamente con las necesidades del usuario.

Con el creciente volumen de conocimientos y experiencias, la mayoría de las organizaciones tienen a menudo problemas identificando el contenido, ubicación y utilización del conocimiento. Las organizaciones de software se esfuerzan constantemente para evitar errores, reducir el trabajo, repetir el éxito de los procesos, aumentar la productividad y, sin embargo, producir productos de alta calidad.

Esta investigación se refiere a estas cuestiones en particular en: (i) la insuficiente gestión de requisitos, y (ii) la falta de exigencia de proceso. La primera cuestión es sobre los requisitos que no se almacenan correctamente en una ubicación media, no

está centralizada y causa problemas en mantener y manipular el requisito de conocimientos.

Mientras que el segundo se refiere a los requisitos de los procesos que no están debidamente documentados o que falta o el re-métodos utilizados son demasiado genéricos y no están correctamente adaptadas a proyectos concretos, por lo que los resultados no son óptimos.

En la ingeniería de software, es rápidamente conocimientos acumulados y cambiado paralelamente con las necesidades del usuario.

La ingeniería de requisitos (RE) es un problema complejo y de actividades intensivas en conocimiento e implica a muchos interesados de diferentes antecedentes trabajar en diferentes fases y actividades. Se pone especial énfasis en la utilización de técnicas de repetible y sistemático para asegurar la exhaustividad, coherencia y pertinencia de los requisitos del sistema.

Los principales procesos de ingeniería de requisitos, es decir, estudio de factibilidad, la obtención, análisis y especificación de requisitos, y el requisito de validación.

- 1) Estudio de Factibilidad es el proceso de identificar si el desarrollo de una idea de proyecto es práctico, beneficioso o viable para una organización.
- 2) Requisitos de la obtención y análisis de información es el proceso de descubrimiento, revisar, documentar y comprender las necesidades del usuario y las limitaciones del sistema. Esto puede lograrse a través de la observación de los sistemas existentes, la discusión.
- 3) La especificación de requisitos es el proceso de traducción y documentación de las necesidades del usuario y las restricciones de forma clara y precisa en un documento que define un conjunto de requisitos.
- 4) El requisito de la validación es el proceso de garantizar que la especificación es coherente con la definición de necesidades.

5) Administración de requisitos regularmente, se perdió en el nuevo ciclo de vida principal. Una vez que el sistema esté instalado y usado regularmente, surgirán nuevas necesidades. Por lo tanto, es importante comprender y controlar los cambios en los requisitos del sistema

Ingeniería inversa para requisitos de software (Syed & Ho-Jin, 2007)

En el artículo se presentada una versión revisada del modelo tradicional de re-ingeniería de proceso, también se ha discutido brevemente sobre ingeniería inversa de software, ingeniería de requisitos y de sus prácticas y actividades básicas.

Un problema fundamental en el mantenimiento y evolución de sistemas heredados es entender el tema. La razón es que la mayoría de estos sistemas heredados no tienen la documentación adecuada. La ingeniería inversa puede ayudar de alguna manera con el mantenimiento adecuado de estos sistemas heredados.

La ingeniería inversa es el proceso de analizar un sistema para identificar los componentes y sus interrelaciones y crear representaciones del sistema en otro formulario o en un nivel de abstracción superior, en otras palabras, está pensado principalmente para recuperar y registrar información de alto nivel sobre el sistema.

Los requisitos de software se ocupan de la transformación de requisitos de software en una descripción del software necesario, parámetros de rendimiento y configuración a través de la definición de proceso iterativo, intercambio de estudios, análisis y creación de prototipos. Algunas otras terminologías comúnmente utilizadas para la ingeniería de requisitos de software son el análisis de las necesidades, la definición de los requisitos funcionales y no funcionales de especificaciones, y requisitos.

Los requisitos de software representan necesidades, deseos, expectativas, limitaciones, las prioridades, la trazabilidad, la cuantificación del riesgo y especificaciones

La obtención de requisitos de software es la tarea de adquirir los requisitos del software. Se considera la etapa donde los ingenieros intentan averiguar lo que las partes interesadas desean realmente.

También se refiere a las actividades de verificación de la exhaustividad, coherencia y viabilidad de los requisitos del software. Puede ayudar a descubrir las anomalías entre los límites del software, a la comprensión de las interacciones con su entorno y elaborar los requisitos del sistema para obtener los requisitos de software

Los documentos de requisitos deben ir a través de los procedimientos de verificación y validación para asegurarse de que todos los problemas son reconocidos relacionadas con los requisitos.

Todavía existen los sistemas heredados a causa de su importante papel en la industria. Pero para hacer frente a los cambios de requisitos y reglas de negocio, el mantenimiento de este software de ser necesario. El problema principal que llega a la actividad de mantenimiento de estos sistemas heredados, es el problema de la comprensión del sistema como en la mayoría de los casos; los sistemas no están bien documentados.

Para resolver estos tipos de problemas, los investigadores están tratando de encontrar maneras de hacer que la información poco clara para una imagen clara. Pero la mayoría de los investigadores sólo se concentran en el diseño de recuperación y utilizar la información en la arquitectura y en la fase de diseño del ciclo de vida.

Diseño y realización del Centro de Servicio Universitario sistema basado en MVC (Jin & Yao, 2014)

El sistema de centro de servicio universitario es una plataforma completa para la universidad para integrar la gestión de TI, que también pueden ofrecer una variedad de servicios para la mayoría de maestros y alumnos.

El MVC es el patrón Model-View-Controller que divide una aplicación, el procesamiento de entrada y salida en tres capas: Capa de modelo-vista-controlador

La capa de vista representa la interfaz de usuario y de aplicación web puede ser HTML, JSP, XML y el Applet y así sucesivamente. La transformación de la vista basada en MVC está limitado a la adquisición y procesamiento de datos y las peticiones del usuario en lugar de los procesos de negocio de la vista.

La función de la capa del modelo de proceso es el negocio y hacer las reglas de negocio. Acepta los datos solicitados de la capa de la vista y devuelve los resultados finales que se están procesando. Esta capa es el núcleo del método de diseño MVC que no ofrecer un modelo, no sólo organiza y administra estos modelos para facilitar la reconstrucción y mejorar la reutilización de modelos.

Un modelo de negocio es importante ya que almacenar objetos entidad y datos.

La capa del controlador acepta las peticiones del usuario y coincide con la vista y el modelo capa juntos para completar las solicitudes. Sólo es un distribuidor y no realiza ningún procesamiento de datos.

Característica de patrón de diseño MVC. Desde el patrón MVC se facilitará el suministro de varias vistas y visualizar simultáneamente múltiples conjuntos de formatos de datos, perfectamente lograr la separación de los datos de rendimiento y control.

Además, el modelo MVC es ampliamente utilizado en J2EE (Java 2 Plataforma Arquitectura Enterprise Edition). La plataforma J2EE, con su riqueza de funciones del sistema, enlazan perfectamente con casi todas las bases de datos relacionales, servidores de procesamiento de transacción y los servidores de mensajería a través de JDBC, XML y otras API.

El trabajo presenta un sistema de centro de servicio universitario basado en el patrón de diseño MVC, utilizando la tecnología JSP básico + Js para lograr una página

dinámica. El patrón de diseño MVC se encarga de separar la capa de lógica de negocios y capa de datos.

Ya que el diseño hace una separación clara de vista, lógica de negocio y capa de datos, permitiendo que el sistema no necesita modificar todo el uno al ampliar y cambiar las funciones del sistema.

Lo único que necesita hacer es reestructurar la lógica empresarial. Las características del diseño son que pueden servir no sólo a los profesores y alumnos en la escuela, y ser capaz de gestionar el departamento de la escuela. Los módulos del sistema son flexibles y pueden ser aumentadas rápidamente o eliminado.

Un marco de presentación para simplificar el desarrollo de aplicaciones Java EE clientes ligeros (Caballé, Ortega, & Camps, 2014)

El objetivo principal de este documento es informar sobre la construcción de una infraestructura de software que simplifica enormemente el desarrollo de la capa de presentación de aplicaciones Java EE con clientes ligeros

Las ventajas del patrón MVC son una clara separación de las preocupaciones, lo que se traduce en aplicaciones más flexibles, que finalmente son más fáciles de gestionar y actualizar. Sin embargo, el uso del patrón MVC tiene muchas tareas repetitivas que compatible con todas las aplicaciones deben realizar, haciendo el trabajo de desarrollo tedioso y complejo.

A fin de superar estas limitaciones, proponen un marco web Java EE que tome ventaja de los patrones de diseño que proporcionan a los desarrolladores muchas ventajas:

- Simplificar y estandarizar la validación de los parámetros de entrada.
- Centralizar el control de la gestión de flujo de trabajo.
- La reutilización de software de alto nivel
- Simplifican muchas tareas de desarrollo repetitivo y tedioso.
- Se revisan los principales marcos de software y patrones de diseño de Java EE.

Apache Struts. Es un framework Java EE en apoyo para el desarrollo de aplicaciones web, haciendo que su ejecución sea más fácil y más simple. Con este fin, automatizando ciertas tareas alivia este marco común y tedioso trabajo de desarrollo en el dominio de aplicación. Struts 2 también hace que las aplicaciones Java EE más robusta y flexible, proporcionando una solución arquitectónica dentro del dominio del flujo de trabajo.

Spring MVC 3. Es un marco acordado para simplificar el desarrollo de aplicaciones empresariales en Java. Spring utiliza simple JavaBeans con resultados similares al complejo de Enterprise JavaBeans (EJB) de versiones anteriores de EJB3, que requiere utilizar métodos complejos como `ejbActivate()` y `ejbpassivate()`, etc.

Java Server Faces 2. Java Server Faces (JSF) es una especificación publicada en 2001 por el Java Community Process (JCP) como Java Specification Request (JSR-127). En 2006 se incorporó el JSF 1.2 en Java EE 5 como JSR 252. En esta versión se utilizó JSP pero con problemas de integración con JSF debido a los diferentes ciclos de vida y como resultado Facelets se presenta como una alternativa a JSP con más funcionalidades, como el soporte para páginas XHTML. Por último, JSF 2 fue desarrollado como JSR 314 convirtiéndose en la opción preferida en Java EE 6 para el desarrollo web. JSF se inspiró en muchos marcos de aplicaciones web de código abierto, con lo que muchas de sus características.

El propósito de JSF es desarrollar aplicaciones web de forma similar a las aplicaciones de escritorio con Java Swing, AWT, SWT y APIs relacionadas. De esta manera, JSF simplificar la construcción de aplicaciones web, proporcionando una web que gestiona las acciones realizadas por el usuario y traducirlos en eventos que se envían al servidor para actualizar la página web. De esta manera, JSF desarrolla aplicaciones web en la ventana del cliente es una página HTML en vez de JFrame o similar.

El resultado del análisis de los patrones de diseño establece que se simplifica y hace más fácil el desarrollo de la aplicación de pruebas siguiendo las acciones de Struts 2,

que proporciona un modelo claro y la aplicación de muchos patrones de diseño de Java EE, por ejemplo, MVC-2 y ApplicationController.

Modelo de Datos de la aplicación MVC en Hadoop para Entrega de la inteligencia de negocios (Romsaiyud, 2014)

El modelo-vista-controlador (MVC) es un patrón de arquitectura de software para implementar interfaces de usuario. Divide una determinada aplicación de software en tres partes interrelacionadas, así como a representaciones internas separadas de la información de la forma en que se presenta la información o aceptado por el usuario.

La biblioteca de software Apache Hadoop es un marco que permite el procesamiento distribuido de grandes conjuntos de datos en clústeres de ordenadores utilizando modelos de programación simple. Está diseñado para escalar desde un único servidor a miles de máquinas, cada una de ellas ofreciendo locales de almacenamiento y computación.

El patrón del repositorio es un patrón de acceso a datos que promueve un enfoque más flexible de acceso a datos. En lugar de tener un controlador o el modelo de negocio que contiene la lógica de acceso a datos, de una clase o conjunto de clases llamado repositorio asume la responsabilidad de conservar el modelo de negocio de la aplicación. El patrón de repositorio bien complementa el principio clave del diseño del patrón MVC separación de preocupaciones.

El sistema consta de cinco pasos; 1.) Los orígenes de datos que se genera a partir de 5 categorías de fuentes de datos como el sistema de registro web, correo web, blogs, noticias y la Wikipedia en inglés, que están incluidos en el sistema de archivos de datos en el sistema local, 2.) Mapa divide la fase en rangos de entrada por el InputFormat y crea un mapa para cada rango de tareas en la entrada. El JobTracker distribuye esas tareas a los nodos del trabajador. La salida de cada tarea mapa está dividido en un grupo de pares clave-valor para cada reducir el valor predeterminado es 64 MB,

3.) Reducir la fase, recoge los diversos resultados y los combina para contestar el mayor problema del nodo maestro estaba tratando de resolver. Reducir, cada uno tira de la partición correspondiente de las máquinas donde los mapas ejecutados, entonces escribe su salida hacia HDFS. Así, la reducción es capaz de recopilar los datos de todos los mapas de las teclas es responsable y combinarlos para solucionar el problema.

4) Modelo de datos crea una Entity Data Model (EDM) para datos recopilados como un archivo de clase y 5.) La interfaz de usuario, para generar la aplicación que se ejecuta en todos los navegadores y dispositivos como dispositivo móvil en todas las plataformas.

El objetivo principal de este trabajo es extraer, transformar y almacenar los datos recopilados a partir de los datos de sistema de archivos o de DFS en Hadoop. El Hadoop almacena el conjunto de datos como el procesamiento orientado por lotes que tiene 5 categorías de tipos de archivos. La perspectiva clave conducir en este estudio es usar el Model-View-Controller (MVC) patrón para utilizar y aplicar los datos obtenidos dependen de cada requisito del usuario. Basado en el experimento, los patrones MVC para apoyar la separación de los componentes de los programas (y reusó de la lógica de negocio a través de aplicaciones)

El estudio y la práctica de desarrollo de software ágil Scrum de importación (Guang-yong, 2011)

Scrum se ha convertido en un importante método de proceso de software de la industria del software.

Es difícil cumplir el requisito de software cuando la entrega final, por lo cual el autor presidió la participación o proyecto en el pasado, aunque el método de ciclo de vida tradicionales utilizadas para el proyecto.

El desarrollo, en la práctica tradicional de desarrollo presentó la parte práctica de prácticas ágiles de desarrollo, y la primera prueba tiene éxito presenta el equipo mediante la práctica y comparación de ingeniería de software tradicional, ágil de prácticas sobre mejorar la calidad de los productos y la eficiencia de la producción jugar con más grandes de papel, tiene la ventaja absoluta tratar con rápidos cambios comerciales.

Scrum es un marco ágil, es un proceso de desarrollo iterativo e incremental, más populares en pocos años. Scrum se basa en la experiencia de la adaptación, basándose en una comprensión de la naturaleza de las actividades de desarrollo de software, proporciona un marco suelto por el equipo de auto-organización, utilizando el método del proceso exploratorio para organizar un proceso de desarrollo. Scrum énfasis en el mejoramiento de iteraciones para encontrar la mejor manera de solucionar el problema.

Scrum proporciona un conjunto de métodos ágiles para ayudar a mejorar la fuerza general de los equipos de software. La idea básica de Scrum es un software unificado los objetivos en materia de prestación de servicios, dependen de la sabiduría del equipo que realiza el proyecto de evaluación, planificación y diseño, desarrollo y pruebas. Scrum es un marco de controles, adaptativa.

Define tres roles: propietario del producto, ScrumMaster, equipo, producto propietario (PM) reunir los requisitos de producto formado.

La práctica ha demostrado que el scrum es un proceso de la mejora continua, con la mejora de la capacidad de auto-organización y establecer un grupo de aprendizaje, y suma constantemente en la práctica y mejora, Scrum va a mejorar continuamente la productividad, mejora de procesos sin fin.

En este artículo el proyecto importado correctamente Scrum software de desarrollo ágil, integra la tecnología en el marco de Scrum, ampliado y mejorado en el proceso de aplicación de Scrum. La productividad está mejorando notablemente de un equipo de proyecto, garantizando la calidad de los proyectos, en los plazos de entrega.

1.10 Análisis Comparativo

En la tabla 1.1 se muestran las principales características encontradas en los trabajos relacionados y los trabajos de esta tesis.

Tabla 1.1 Comparación de los trabajos relacionados

| Referencia | Proyecto | Arquitectura utilizada | Características | Lenguaje |
|----------------------------------|---|------------------------|--|---|
| (Hasan & R.K., 2011) | Desarrollo de sistemas expertos basados en la web | MVC | Separación de los componentes. Reutilización de código y pruebas | Microsoft ASP |
| (Dragos & Altar, 2014) | Diseño de un MVC para un desarrollo rápido de aplicaciones web | MVC | Separación de los componentes. Prototipos rápidos de las vistas | Laravel, Fuel PHP, Ruby on Rails and ASP.NET MVC. |
| (Hamid Mcheick, 2011) | El papel tan importante que tiene los conectores en el MVC | MVC | Separación de los componentes Capa de presentación Capa de dependencias Capa de transporte | Java |
| (Jin & Yao, 2014) | Sistema de centro de servicio universitario basado en el patrón de diseño MVC | MVC | Separación de componentes en capas. | J2EE |
| (Caballé, Ortega, & Camps, 2014) | construcción de una infraestructura de software que simplifica el desarrollo de la capa de presentación | MVC | Presenta los marcos para el desarrollo de aplicaciones bajo MVC. Apache Struts Spring MVC 3 Java Server Faces 2 | Java EE |
| (Romsaiyud, 2014) | Extraer, transformar y almacenar los datos recopilados a partir de los datos de sistema de archivos o de DFS en Hadoop. | MVC | Separación de componentes. | Java |

Analizando las características presentadas en la tabla 1.1 no se cuenta con algún sistema cotizador de madera a la medida. En los trabajos presentados se trabaja

igualmente con la separación de los componentes, además de estar desarrollados bajo la misma plataforma Java. Para el desarrollo al igual que los proyectos se utilizan frameworks para un desarrollo más rápido y estructurado entre los componentes.

1.11 Comparación de sistemas de cotización en el mercado

Se realiza un estudio de los diferentes sistemas para cotización que existen en el mercado, tanto privado como comercial, para analizar sus características y comparar las que tienen en común con el sistema cotizador. En la tabla 1.2 se muestran los sistemas incluyendo la plataforma en la que corren, sus principales características, así como la empresa que lo fabrica.

Se analizan los distintos softwares de tipo cotizador, para identificar características específicas y así tener una base para la comprender el funcionamiento de los sistemas en línea.

Tabla 1.2 Comparación de los sistemas en el mercado

| Nombre y descripción | Características | Empresa |
|--|--|---|
| <p>cotizadorweb.com</p> <p>Es un sistema de cotizaciones en línea, el cual le permite a usted y a su personal de ventas, administrar y enviar cotizaciones en minutos de sus productos y/o servicios, reduciendo el tiempo de respuesta y brindando un mejor servicio a sus clientes.</p> | <ul style="list-style-type: none"> • No se necesita descargar e instalar • Podrá conocer que cotización envió cada agente, cuando, a quién, el contenido • Recibe un reporte diario por correo sobre las cotizaciones realizadas • Muestra en qué estado se encuentra cada una: solicitada, enviada, aceptada, rechazada, en seguimiento, etc. | <p>Banbu Code S.A de C.V (Bambu Code S.A de C.V., 2016)</p> |

Tabla 1.3 Comparación de los sistemas en el mercado (continuación)

| Nombre y descripción | Características | Empresa |
|--|---|--|
| <p>ExportandOnline</p> <p>El Cotizador Online es un sistema versátil que le permitirá a usted o a su personal de ventas realizar cotizaciones de los productos o servicios que ofrezca de una manera rápida, logrando optimizar la atención al cliente.</p> | <ul style="list-style-type: none"> • Algoritmos o lógicas matemáticas (La Inversión varía por el número de operaciones) • Manejo de las cotizaciones a través de un listado general. • Seguimiento del estado de sus cotizaciones. • Base de datos actualizada de las cotizaciones generadas. • Generación de cotización en formato PDF el cual puede ser descargado o visualizado desde la web. • Módulos de configuración con respecto al tipo de moneda que maneje y el tipo de cambio. • Módulo extra para la administración de sus clientes, puede agregar, editar, dar de baja y visualizar los datos del mismo. | <p>ExportandOnline (ExportandOnline, 2016)</p> |
| <p>NewWeb</p> <p>Sitio web que les permita a los visitantes estimar costos en base a sus necesidades. Generar más visitas al sitio web optimizado, tanto en forma natural como en anuncios de pago por clic.</p> | <ul style="list-style-type: none"> • Cotizar productos en línea • Permite a los visitantes fácilmente crear su cotización y enviar pedido | <p>Inteligencia en Tecnología y Servicios S.C. (Inteligencia en Tecnología y Servicios, S.C, 2016)</p> |
| <p>COTIZADOR WEB</p> <p>Cotizador para calcular el costo para el desarrollo de un sitio web</p> | <ul style="list-style-type: none"> • Permite elegir el número de páginas • Plataforma en la que se desea realizar • Si incluye el diseño • Si el sitio contendrá imágenes, redes sociales, formularios etc. | <p>Todowebhosting (Todowebhosting.com, 2016)</p> |

Tabla 1.4 Comparación de los sistemas en el mercado (continuación)

| Nombre y descripción | Características | Empresa |
|--|---|------------------------------------|
| <p>COTIZADOR WEB Calcule el precio aproximado de su sitio web</p> | <ul style="list-style-type: none"> • Permite elegir el número de cuentas • Si contendrá banners • Si almacenara catálogo de productos • Si desea un posicionamiento en la web | Web Hostin (Miweb.com.mx, 2016) |
| <p>Axa Cotiza tu seguro de autos y encuentra el paquete que se adapte a tus necesidades.</p> | <ul style="list-style-type: none"> • Tiene los campos para elegir la marca, modelos, año, descripción. • Captura los datos del propietario. | Axa Seguros (Axa, 2016) |

Los sistemas que se muestran en la tabla 1.2 son para plataformas web, y solo se debe contar con un navegador. Ninguno de los sistemas contiene su costo en línea, se debe mandar la información para un presupuesto.

En base a estos datos se puede verificar que estos sistemas de cotización son distintos al sistema cotizador que se pretende realizar, ya que esta echo a la medida que la empresa lo necesita con las operaciones personalizadas para el cálculo de cada producto.

1.12 Análisis del estado del Arte

Hoy en día la reutilización de código, facilidad de mantenimiento y el rendimiento en las aplicaciones web toma un papel muy importante a la hora de programar, el identificar comportamientos autónomos para un ahorro considerable en términos de número de ejecuciones.

Utilizando el entorno de desarrollo IDE como eclipse, al cual se realizaron pruebas en distintas organizaciones para validar la funcionabilidad, y los estudios demuestran que es más amigable con los usuarios, ya que se adapta a las necesidades de los desarrolladores.

La usabilidad en los sistemas es un punto muy importante, en la actualidad si un sistema no tiene una buena usabilidad, el desarrollo del sistema se conduce a una frustración y a la insatisfacción. Por lo que se proponen directrices para el desarrollo del software y así mejorar la calidad del diseño y reducir la complejidad.

La utilización de framework proporciona una arquitectura de comunicación para la construcción de sistemas, además de agilizar y hacer más simple la estructura del código. Son herramientas que en la actualidad ayudan en el desarrollo, ya que disminuyen el tiempo de codificación y la reutilización de código.

Complementado con una metodología para el desarrollo, los equipos de desarrollo de software sostienen un ritmo de diseño simple, que provee pruebas para una mejor estructura de los sistemas de arquitectura de aplicaciones web y de optimizar las estrategias para lidiar con la complejidad del desarrollo y para lograr una alta usabilidad.

EL modelo MVC clásico puede ser aplicado con frecuencia a las aplicaciones de escritorio que se ejecutan en un equipo local.

Con el Modelo-Vista-Controlador (MVC), la arquitectura de un sistema se divide en tres tipos de módulos: controlador, modelo y vista.

Desde el patrón MVC se logra efectivamente la separación de la expresión y el contenido, ofreciendo un mínimo de código, tasa de repetición para los programadores que necesitan una variedad de manipulación de datos. Esto facilitará el suministro de varias vistas y visualizar simultáneamente múltiples conjuntos de formatos de datos, y perfectamente lograr la separación de los datos de rendimiento y control.

Las ventajas del patrón MVC son una clara separación de los componentes, lo que se traduce en aplicaciones más flexibles, que finalmente son más fáciles de gestionar y actualizar. Sin embargo, el uso del patrón MVC tiene muchas tareas repetitivas que son compatibles con todas las aplicaciones, haciendo el trabajo de desarrollo tedioso y complejo.

El controlador es una clase base que maneja el modelo de seguridad de acceso a la consulta, la ruta y la solución. Los controladores están incorporados como hijos del controlador básico y heredan la funcionalidad y están enriquecidos con la lógica de la aplicación correspondiente a las acciones deseadas.

En el Front end y back end convergen las tecnologías para construir aplicaciones web, ya que ha evolucionado a un ritmo rápido y los desarrolladores deben utilizar un número bastante amplio de tecnologías para crear una sola aplicación web.

Se estudió el propósito de JSF, y es desarrollar aplicaciones web de forma similar a las aplicaciones de escritorio con Java Swing, AWT, SWT y APIs relacionadas. De esta manera, JSF simplifica la construcción de aplicaciones web, proporcionando una web que gestiona las acciones realizadas por el usuario y traducirlos en eventos que se envían al servidor para actualizar la página web.

La biblioteca de software Apache Hadoop es un marco que permite el procesamiento distribuido de grandes conjuntos de datos en clústeres de ordenadores utilizando modelos de programación simple. Está diseñado para escalar desde un único servidor a miles de máquinas, cada una de ellas ofreciendo almacenamiento.

Una de las principales limitaciones de la ingeniería de software es predecir el proceso y las actividades necesarias de forma secuencial con el fin de producir el producto y el resultado aceptable.

En la ingeniería de software, es rápidamente conocimientos acumulados y cambiado paralelamente con las necesidades del usuario. Las organizaciones de software se esfuerzan constantemente para evitar errores, reducir el trabajo, repetir el éxito de los procesos, aumentar la productividad y producir productos de alta calidad.

Los principales procesos de ingeniería de requisitos, es decir, estudio de factibilidad, la obtención, análisis y especificación de requisitos, y el requisito de validación.

Un problema fundamental en el mantenimiento y evolución de sistemas heredados es entender el tema. La razón es que la mayoría de estos sistemas heredados no tienen la documentación adecuada. La ingeniería inversa puede ayudar de alguna manera con el mantenimiento adecuado de estos sistemas heredados.

Los requisitos de software representan necesidades, deseos, expectativas, limitaciones, las prioridades, la trazabilidad, la cuantificación del riesgo y especificaciones

Los documentos de requisitos deben ir a través de los procedimientos de verificación y validación para asegurarse de que todos los problemas conocidos sean relacionados con los requisitos.

Se propone trabajar con un grupo de 3 integrantes aplicando una metodología de desarrollo ágil en donde se validará el trabajo en grupo y si se cumple con los procesos que demanda Scrum, además de aplicar el proceso de ingeniería de requerimientos cumpliendo con los procesos que conlleva todo el desarrollo de sistema, desde su etapa de análisis hasta su implementación y pruebas.

1.13 Comentarios

En este Capítulo se muestra un antecedente de la empresa y el problema que presenta, al realizar un análisis sobre la problemática que tiene la empresa se propone una solución, lo que conlleva a la realización de un software para sus cotizaciones.

La propuesta de solución propone los objetivos para el sistema cotizador, se plantean los alcances y las posibles limitaciones que puede tener.

Se plantean metodologías para el desarrollo del sistema, ya que se deberá presentar un software competitivo que este a la vanguardia y que cumpla con las características del cliente, diseñando un software que será de suma importancia para las operaciones que se realizan a diario en la empresa.

Se realiza búsqueda para identificar en los distintos softwares de tipo cotizador y así tener una base para la elaboración del sistema, no se tiene sistemas cotizadores en el ámbito de la investigación, solo se cuenta con sistemas cotizadores para el ámbito comercial y privado de diferentes características.

Se hace una investigación acerca de temas relacionados con las metodologías que se implementaran o utilizaran con el fin de tener información actualizada de la aplicación de estas metodologías.

2 Fundamentos

En este Capítulo se presenta la parte teórica requerida para determinar el desarrollo del proyecto, componiéndose de una recopilación de procedimientos y herramientas que ayudan al desarrollador de software para el desarrollo del sistema informático. Las metodologías planteadas indican los pasos a seguir para el desarrollo del proyecto y son tomadas como reglas.

2.1 El Patrón de diseño Modelo Vista Controlador (MVC)

Uno de los patrones más antiguos es el Modelo - Vista - Controlador (MVC). Originalmente desarrollado en el lenguaje Smalltalk, este patrón es sumamente prevalente en actualidad, especialmente con la proliferación de aplicaciones web y móviles. Antes de un diagrama se muestra una definición de cada componente.

- Modelo: Representa los datos y, opcionalmente, la lógica empresarial del sistema.
- Vista: Acepta entradas al sistema y muestra los resultados al cliente.
- Controlador: Acepta la entrada de la vista y solicita los datos del modelo; determina las acciones tanto para la vista y el modelo basado en lógica programada.

La principal ventaja de este modelo es que la vista se separa del modelo y, por lo tanto, la vista puede cambiarse sin afectar el modelo. Además, el modelo y el controlador pueden ser probados y verificados sin vistas a través de las pruebas unitarias y controladores simulados, (Crookshanks, 2015).

Definición: Trygve Reenskaug es el creador del patrón Modelo-Vista-Controlador. Durante su estancia en Xerox Parc entre 1978 y 1979 desarrolló las ideas en las que

se basa el patrón conocido hoy en día, y sus ideas se usaron para implementar el Modelo-Vista-Controlador en Smalltalk-80. El objetivo fundamental que perseguía Reenskaug era reducir la distancia entre el modelo mental del usuario y el modelo computacional. Para hacer creer al usuario que ve y manipula la información del dominio directamente, (Bergin, 2016).

2.2 El componente controlador

Un controlador es el vínculo entre el usuario y el sistema, proporciona al usuario la entrada disponiendo puntos de vista relevantes para presentarse en los sitios apropiados de la pantalla. Proporciona medios de salida para el usuario, presentando al usuario menús y otros medios para dar instrucciones y datos. El controlador proporciona una salida al usuario, lo traduce a los mensajes apropiados y pasa estos mensajes para uno o más puntos de las vistas.

Un controlador nunca debe complementar las vistas, por ejemplo, nunca se debería conectar las vistas de nodos dibujando flechas entre ellos.

Por el contrario, una vista nunca debe saber acerca de la entrada del usuario, tales como las operaciones del ratón y las pulsaciones de teclas. Siempre debe ser posible escribir un método en un controlador que envía mensajes con opiniones que reproducir exactamente cualquier secuencia de comandos de usuario, (Reenskaug, 1979).

Controlador: Intermediario entre la vista y el modelo. Es quien controla las interacciones del usuario solicitando los datos al modelo (Base de Datos) y entregándolos a la vista para que ésta lo presente al usuario.

Aunque se pueden encontrar diferentes implementaciones del MVC, el flujo que sigue el control se puede observar en la figura 2.1, generalmente es como se describe a continuación:

El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, da clic con el ratón).

El controlador recibe (por parte de los objetos de la interfaz de usuario) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, a través de un gestor de eventos.

El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario. Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se reflejan los cambios en el modelo. El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, se podría utilizar el patrón Observador para proveer un monitoreo indirecto entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. Este uso del patrón Observador no es posible en las aplicaciones Web puesto que las clases de la vista están desconectadas del modelo y del controlador.

En general el controlador no pasa objetos de dominio (el modelo) a la vista, aunque puede dar la orden a la vista para que se actualice. Nota: En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.

La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente, (Cervantes, 2013) .

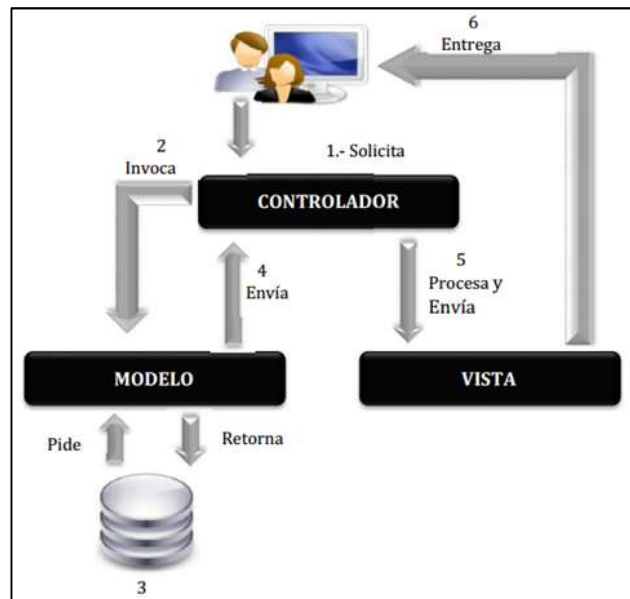


Figura 2.1 Proceso del modelo vista controlador (Cervantes, 2013)

Características

- El controlador es el cerebro de la aplicación.
- Determina las acciones a realizar para cada una de las peticiones del modelo y la vista.
- Se encarga de coordinar todos los procesos.
- Es el encargado de recibir los eventos de entrada.
- Contiene las reglas de la gestión de los eventos.

2.3 Ingeniería de Software

La ingeniería de software es una disciplina de ingeniería que se interesa por todos los aspectos de la producción de software, desde las primeras etapas de la especificación del sistema hasta el mantenimiento del sistema después de que se pone en operación, (Sommerville, 2011).

2.4 Modelo del proceso de software

Para la construcción de un producto es importante ejecutar una serie de actividades imprescindibles, el cual sirve como un mapa para el proceso de software. Los ingenieros de software tienen un papel importante para el desarrollo de este documento ya que contiene la estabilidad control y organización de las actividades de forma detalladas para llegar a un producto final.

2.4.1 El modelo en cascada

El primer modelo publicado sobre el proceso de desarrollo de software se derivó a partir de procesos más generales de ingeniería de sistemas. Este modelo se ilustra en la figura 2.2. Debido al paso de una fase en cascada a otra, este modelo se conoce como “modelo en cascada” o ciclo de vida del software. El modelo en cascada es un ejemplo de un proceso dirigido por un plan; en principio, usted debe planear y programar todas las actividades del proceso, antes de comenzar a trabajar con ellas.

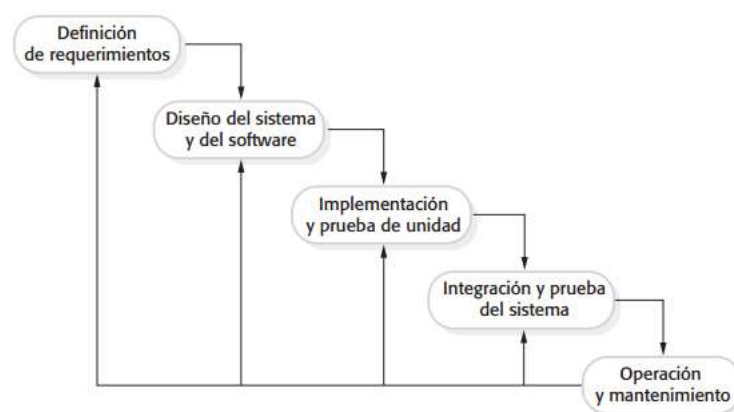


Figura 2.2 El modelo en cascada (Sommerville, 2011).

Las principales etapas del modelo en cascada reflejan directamente las actividades fundamentales del desarrollo:

Análisis y definición de requerimientos. Los servicios, las restricciones y las metas del sistema se establecen mediante consulta a los usuarios del sistema. Luego, se definen con detalle y sirven como una especificación del sistema.

Diseño del sistema y del software. El proceso de diseño de sistemas asigna los requerimientos, para sistemas de hardware o de software, al establecer una arquitectura de sistema global. El diseño del software implica identificar y describir las abstracciones fundamentales del sistema de software y sus relaciones.

Implementación y prueba de unidad. Durante esta etapa, el diseño de software se realiza como un conjunto de programas o unidades del programa. La prueba de unidad consiste en verificar que cada unidad cumpla con su especificación.

Integración y prueba de sistema. Las unidades del programa o los programas individuales se integran y prueban como un sistema completo para asegurarse de que se cumplan los requerimientos de software. Después de probarlo, se libera el sistema de software al cliente.

Operación y mantenimiento. Por lo general (aunque no necesariamente), ésta es la fase más larga del ciclo de vida, donde el sistema se instala y se pone en práctica. El mantenimiento incluye corregir los errores que no se detectaron en etapas anteriores del ciclo de vida, mejorar la implementación de las unidades del sistema e incrementar los servicios del sistema conforme se descubren nuevos requerimientos.

En principio, el resultado de cada fase consiste en uno o más documentos que se autorizaron (“firmaron”). La siguiente fase no debe comenzar sino hasta que termine la fase previa. En la práctica, dichas etapas se traslapan y se nutren mutuamente de información.

Debido a los costos de producción y aprobación de documentos, las iteraciones suelen ser onerosas e implicar un rediseño significativo. Por lo tanto, después de un pequeño

número de iteraciones, es normal detener partes del desarrollo, como la especificación, y continuar con etapas de desarrollo posteriores.

Durante la fase final del ciclo de vida (operación y mantenimiento), el software se pone en servicio. Se descubren los errores y las omisiones en los requerimientos originales del software. Surgen los errores de programa y diseño, y se detecta la necesidad de nueva funcionalidad. Por lo tanto, el sistema debe evolucionar para mantenerse útil.

Hacer tales cambios (mantenimiento de software) puede implicar la repetición de etapas anteriores del proceso, (Sommerville, 2011).

2.4.2 El modelo incremental

El modelo incremental combina elementos del modelo lineal secuencial (aplicados repetidamente) con la filosofía interactiva de construcción de prototipos. Como muestra la Figura 2.3, el modelo incremental aplica secuencias lineales de forma escalonada mientras progresa el tiempo en el calendario.

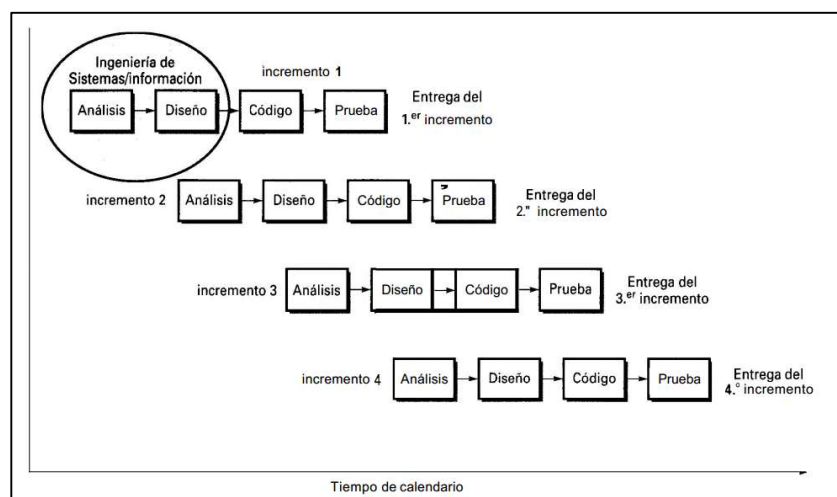


Figura 2.3 El modelo incremental (Pressman, 2010).

El modelo incremental entrega el software en partes pequeñas, pero utilizables, llamadas (incrementos). En general, cada incremento se construye sobre aquél que ya ha sido entregado.

El desarrollo incremental es particularmente útil cuando la dotación de personal no está disponible para una implementación completa en la fecha límite que se haya establecido para el proyecto. Los primeros incrementos se pueden implementar con menos personas, (Pressman, 2010).

2.4.3 El modelo construcción de prototipos

El paradigma de construcción de prototipos (Figura. 2.4) comienza con la recolección de requisitos. El desarrollador y el cliente encuentran y definen los objetivos globales para el software, identifican los requisitos conocidos y las áreas del esquema en donde es obligatoria más definición. Entonces aparece un (diseño rápido). El diseño rápido se centra en una representación de esos aspectos del software que serán visibles para el usuario/cliente (por ejemplo: enfoques de entrada y formatos de salida). El diseño rápido lleva a la construcción de un prototipo.

El prototipo lo evalúa el cliente/usuario y se utiliza para refinar los requisitos del software a desarrollar. La iteración ocurre cuando el prototipo se pone a punto para satisfacer las necesidades del cliente, permitiendo al mismo tiempo que el desarrollador comprenda mejor lo que se necesita hacer.

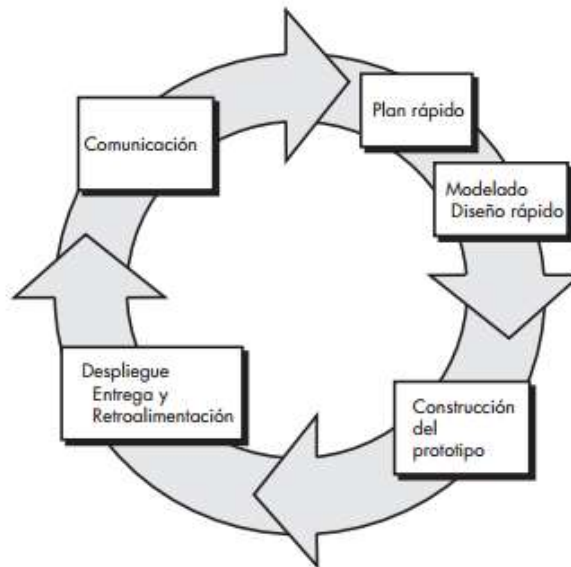


Figura 2.4 Paradigma de construcción de prototipos (Pressman, 2010)

El prototipo puede servir como (primer sistema). El que Brooks recomienda tirar. Aunque esta puede ser una visión idealizada. Es verdad que a los clientes y a los que desarrollan les gusta el paradigma de construcción de prototipos. A los usuarios les gusta el sistema real y a los que desarrollan les gusta construir algo inmediatamente, (Pressman, 2010).

2.4.4 El modelo DRA

El Desarrollo Rápido de Aplicaciones (DRA) es un modelo de proceso del desarrollo del software lineal secuencial que enfatiza un ciclo de desarrollo extremadamente corto. El modelo DRA es una adaptación a (alta velocidad) del modelo lineal secuencial en el que se logra el desarrollo rápido utilizando una construcción basada en componentes. Si se comprenden bien los requisitos y se limita el ámbito del proyecto,

el proceso DRA permite al equipo de desarrollo crear un «sistema completamente funcional» dentro de períodos cortos de tiempo (por ejemplo: de 60 a 90 días).

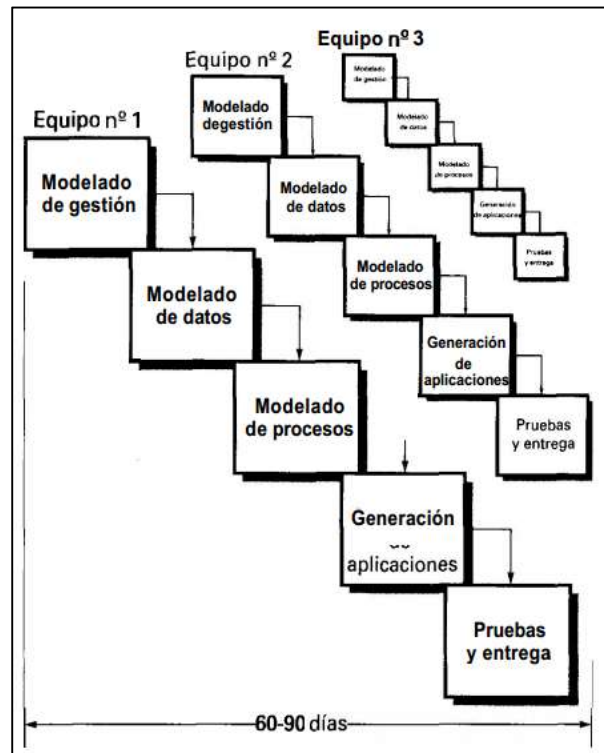


Figura 2.5 El modelo DRA (Pressman, 2010).

El modelo de proceso DRA se ilustra en la Figura 2.5. Obviamente, las limitaciones de tiempo impuestas en un proyecto DRA demandan (ámbito en escalas). Si una aplicación de gestión puede modularse de forma que permita completarse cada una de las funciones principales en menos de tres meses (utilizando el enfoque descrito anteriormente), es un candidato del DRA. Cada una de las funciones pueden ser afrontadas por un equipo DRA separado y ser integradas en un solo conjunto, (Pressman, 2010).

2.4.5 El modelo lineal secuencial

Llamado algunas veces (ciclo de vida básico o modelo en cascada), el modelo lineal secuencial sugiere un enfoque sistemático, secuencial, para el desarrollo del software que comienza en un nivel de sistemas y progresa con el análisis, diseño, codificación, pruebas y mantenimiento. La Figura 2.6 muestra el modelo lineal secuencial para la ingeniería del software

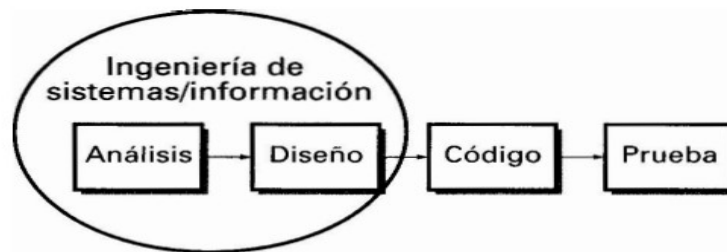


Figura 2.6 Modelo lineal secuencia (Pressman, 2010).

El modelo lineal secuencial es el paradigma más antiguo y más extensamente utilizado en la ingeniería del software, (Pressman, 2010).

2.4.6 El modelo espiral

El modelo en espiral, propuesto originalmente por Boehm, es un modelo de proceso de software evolutivo que conjuga la naturaleza iterativa de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial. Proporciona el potencial para el desarrollo rápido de versiones incrementales del software. En el modelo espiral, el software se desarrolla en una serie de versiones incrementales.

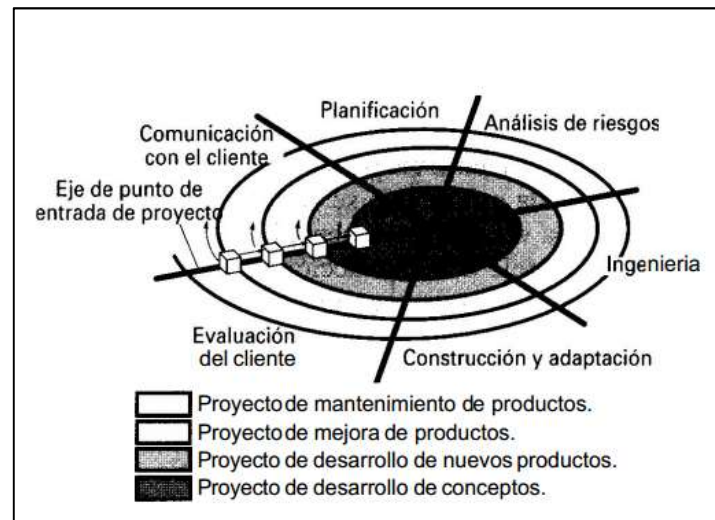


Figura 2.7 Un modelo espiral típico (Pressman, 2010).

La Figura 2.7 representa un modelo en espiral que contiene seis regiones de tareas

- Comunicación con el cliente-las tareas requeridas para establecer comunicación entre el desarrollador y el cliente.
- Planificación - las tareas requeridas para definir recursos, el tiempo y otra información relacionadas con el proyecto.
- Análisis de riesgos-las tareas requeridas para evaluar riesgos técnicos y de gestión.
- Ingeniería - las tareas requeridas para construir una o más representaciones de la aplicación.
- Construcción y acción - las tareas requeridas para construir, probar, instalar y proporcionar soporte al usuario (por ejemplo: documentación y práctica)
- Evaluación del cliente-las tareas requeridas para obtener la reacción del cliente según la evaluación de las representaciones del software creadas durante la etapa de ingeniería e implementada durante la etapa de instalación, (Pressman, 2010).

2.5 Metodologías Ágiles

Definición

Ágil, como concepto, se dieron a conocer a finales de la década de 1990 a través de los esfuerzos para hacer frente a dificultades percibidas con los procesos de desarrollo de soluciones existentes que están arraigados, y debían su rigidez, impulsada por el plan de prácticas.

Por lo tanto, proyectos ágiles exhiben características de comunicación abierta entre actores heterogéneos, comportamiento emergente dentro de la auto-organización de los equipos y una cultura de apertura y aprendizaje. (Switzerland, 2015)

Historia

Los orígenes de Ágil están arraigados en parte en los sectores industrial y manufacturero japonés que muchos conceptos ágiles deben su patrimonio. Estos incluyen el concepto de control visual encontrados en el sistema de producción Toyota que posteriormente ágil de información anticipada de los radiadores, los gráficos utilizados en ágil Kanban de asignación y seguimiento de tareas, y la continua influencia de pensamiento lean sobre Agile hoy. La síntesis del pensamiento occidental y oriental de manera persuasiva establecidos por los autores que introdujo el término "Scrum", refleja el espíritu en el que el manifiesto fue concebido y apunta a una génesis fundada en el aprendizaje organizacional y la potenciación del equipo.

La figura 2.8 ilustra el desarrollo histórico de las principales metodologías ágiles desde finales de los años ochenta y principios de los noventa (la influencia de las formas tradicionales de pensamiento es indicado por cuadros discontinua).

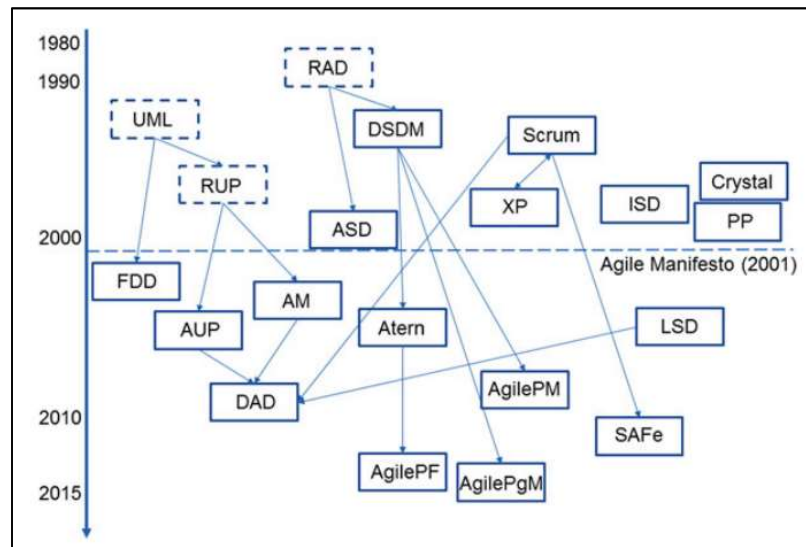


Figura 2.8 Desarrollo de las principales metodologías (Switzerland, 2015)

En 1994 el método de desarrollo de sistemas dinámicos (DSDM), el primer método de desarrollo verdaderamente ágiles, surgió de la RAD como un marco independiente que ha evolucionado con el tiempo hasta abarcar un ámbito más amplio de lo que uno podría tradicionalmente asociar con proyectos Agile (p. ej., la inclusión explícita de la gobernabilidad, calidad y riesgo) que culminaron en la gestión ágil de proyectos tal como se describe en el marco de proyectos ágiles DSDM (AgilePF DSDM), la gestión ágil de proyectos (AgilePM) y el Programa DSDM de marcos de gestión (AgilePgM). RAD emplea prácticas iterativas que se basó en el modelo espiral y otras ideas que se habían practicado por IBM en la década de 1990.

En paralelo Scrum, cuyas raíces se remontan a su patrimonio industrial en 1986 donde fue concebida como una metodología de desarrollo industrial que trajo consigo la "innovación continua, gradualmente y en forma de espiral", tomó forma a partir de las ideas derivadas de aprendizaje organizacional y pasó a ejercer una influencia considerable en el mundo ágil impactando en casi todas las otras metodologías.

Los equipos ágiles tienden a ser pequeños heterogéneos compuesto de especialistas "generalizar", capaz de participar en diferentes tipos de trabajos (por ejemplo, análisis, desarrollo, pruebas, etc.).

En términos generales, un equipo debe equilibrar la necesidad de adaptación (p. ej., innovación) en contra de la potencial presión para homogeneizar.

Hoy existen varias metodologías bien establecidas metodologías ágiles en uso, que van desde los ligeros enfoques que tienen un fuerte enfoque en el desarrollo de productos tales como programación extrema (XP) y Scrum con mayores metodologías como la gestión de proyectos de desarrollo de software centrado dinámico método o la arquitectura orientada a escala marco ágil (SAFe). Cada metodología tiene su propia cultura, idioma y prácticas. Por ejemplo, XP y Scrum expresar sus principios en términos humanista (p. ej., el respeto, la valentía de DSDM), mientras que puede apelar a los más cómodos en un entorno maduro con un fuerte enfoque estructural como se pone de manifiesto en su formulación de valores (por ejemplo, centrarse en las necesidades de la empresa, entregar a tiempo, colaborar, demostrar el control), (Switzerland, 2015).

2.5.1 Metodología ágil DAS

El desarrollo adaptativo de software (DAS) fue propuesto por Jim Highsmith como una técnica para elaborar software y sistemas complejos. Los fundamentos filosóficos del DAS se centran en la colaboración humana y en la organización propia del equipo. Highsmith argumenta que un enfoque de desarrollo adaptativo basado en la colaboración es “tanto una fuente de orden en nuestras complejas interacciones, como de disciplina e ingeniería”.

En realidad, Highsmith afirma que los desarrolladores de software sobreestiman con frecuencia su propia comprensión (de la tecnología, del proceso y del proyecto) y que

el aprendizaje los ayudará a mejorar su nivel de entendimiento real. Los equipos DAS aprenden de tres maneras: grupos de enfoque, revisiones técnicas y análisis post mórtem del proyecto, (Sommerville, 2011).

2.5.2 Metodología ágil MDSD

El método de desarrollo de sistemas dinámicos (MDSD) es un enfoque de desarrollo ágil de software que “proporciona una estructura para construir y dar mantenimiento a sistemas que cumplan restricciones apretadas de tiempo mediante la realización de prototipos incrementales en un ambiente controlado de proyectos”. La filosofía MDSD está tomada de una versión modificada de la regla de Pareto: 80 por ciento de una aplicación puede entregarse en 20 por ciento del tiempo que tomaría entregarla completa (100 por ciento).

El MDSD es un proceso iterativo de software en el que cada iteración sigue la regla de 80 por ciento. Es decir, se requiere sólo suficiente trabajo para cada incremento con objeto de facilitar el paso al siguiente. Los detalles restantes se terminan más tarde, cuando se conocen los requerimientos del negocio y se han pedido y efectuado cambios, (Sommerville, 2011).

2.5.3 Metodología ágil Cristal

Alistair Cockburn creó la familia Cristal de métodos ágiles a fin de obtener un enfoque de desarrollo de software que premia la “maniobrabilidad” durante lo que Cockburn caracteriza como “un juego cooperativo con recursos limitados, de invención y comunicación, con el objetivo primario de entregar software útil que funcione y con la meta secundaria de plantear el siguiente juego”.

Para lograr la maniobrabilidad, Cockburn y Highsmith definieron un conjunto de metodologías, cada una con elementos fundamentales comunes a todos, y roles, patrones de proceso, producto del trabajo y prácticas que son únicas para cada uno. La familia Cristal en realidad es un conjunto de ejemplos de procesos ágiles que han demostrado ser efectivos para diferentes tipos de proyectos. El objetivo es permitir que equipos ágiles seleccionen al miembro de la familia Cristal más apropiado para su proyecto y ambiente, (Sommerville, 2011).

2.5.4 Metodología ágil Scrum

Este modelo fue identificado y definido por Ikujiro Nonaka e Hirotaka Takeuchi a principios de los 80, al analizar cómo desarrollaban los nuevos productos las principales empresas de manufactura tecnológica: Fuji-Xerox, Canon, Honda, Nec, Epson, Brother, 3M y Hewlett-Packard. (Manager, 2015)

Scrum puede describirse mejor como una metodología de desarrollo de producto con leves aspiraciones de gestión de proyectos (por ejemplo, seguimiento y generación de informes de ligero) como su enfoque radica en la gestión de requisitos de software y desarrollo. El ámbito de aplicación de Scrum no alcanza a otras actividades como la gestión del cambio empresarial, desarrollo de sistemas o la migración de datos y se remite a las prácticas existentes dentro de una organización para cubrir la iniciación del proyecto, gestión de riesgos, la liberación y la implementación y procesos de administración del cambio. De hecho, las metodologías ágiles que tienen un alcance más amplio han argumentado que Scrum puede ser incorporado exitosamente dentro de sus marcos. Scrum comparte una herencia común con XP y ambos emplean la terminología y las prácticas similares. Las diferencias son, sin embargo, la aparente en su estructura y filosofía. (Switzerland, 2015)

A partir del concepto o visión de la necesidad del cliente, se construye el producto de forma incremental a través de iteraciones breves que comprenden fases de

exploración y revisión. Estas iteraciones (en Scrum llamadas *sprints*) se repiten de forma continua hasta que el cliente da por terminado el producto. Se inicia el desarrollo con la visión general del producto, especificando y dando detalle a las funcionalidades o partes que tienen mayor prioridad de negocio, y que pueden llevarse a cabo en un periodo de tiempo breve (según los casos pueden tener duraciones desde una semana hasta no más de dos meses). Cada uno de estos periodos de desarrollo es una iteración que finaliza con la entrega de una parte (incremento) operativa del producto. Estas iteraciones son la base del desarrollo ágil, y Scrum gestiona su evolución en reuniones breves diarias donde todo el equipo revisa el trabajo realizado el día anterior y el previsto para el siguiente. En la figura 2.9 se observa como Scrum controla de forma empírica y adaptable la evolución del proyecto, a través de las siguientes prácticas de la gestión ágil:

Revisión de las Iteraciones: al finalizar cada iteración (*sprint*) se lleva a cabo una revisión con todas las personas implicadas en el proyecto, es por tanto la duración del *sprint*, el periodo máximo que se tarda en reconducir una desviación en el proyecto o en las circunstancias del producto.

Desarrollo incremental: las personas implicadas no trabajan con diseños o abstracciones. El desarrollo incremental implica que al final de cada iteración se dispone de una parte de producto operativa, que se puede inspeccionar y evaluar.

Desarrollo evolutivo: 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 resultado final, y sobre dicha predicción desarrollar el diseño y la arquitectura del producto no es realista, porque las circunstancias obligarán a remodelarlo muchas veces, (Sánchez Flores, 2013).

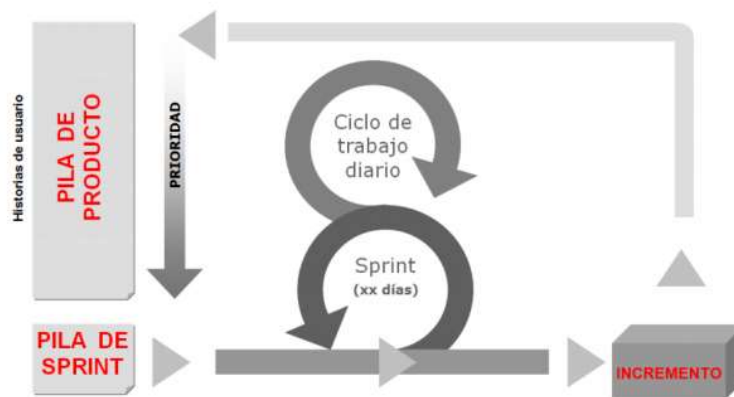


Figura 2.9 Diagrama del ciclo iterativo Scrum (Manager, 2015)

Scrum es un modelo de desarrollo ágil caracterizado por:

- Adoptar una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto.
- Basar la calidad del resultado más en el conocimiento tácito de las personas en equipos auto organizados, que en la calidad de los procesos empleados.
- Solapamiento de las diferentes fases del desarrollo, en lugar de realizarlas una tras otra en un ciclo secuencial o de cascada. (Manager, 2015)

2.5.5 Roles

Son aquellas personas que intervienen, o tienen relación directa o indirecta con el proyecto, se clasifican en dos grupos.

El responsable. Es responsable de asegurar que el proceso es comprendido y seguido. Es responsable de asegurar que el equipo Scrum se adhiere a los valores, prácticas y normas Scrum, ayuda a que el Equipo Scrum y la organización adopten Scrum, enseña al Equipo Scrum mediante entrenamiento y liderándolo para que sea más productivo y construya productos de mayor calidad. El *responsable* ayuda a que el Equipo Scrum comprenda y utilice la auto-gestión y a ser multidisciplinario.

El Propietario del Producto. Es la única persona responsable de gestionar la *pila del producto* y asegurar el valor del trabajo que el equipo lleva a cabo. El Propietario del Producto es una persona, no un comité. Pueden existir comités que le aconsejen o le influyeran, pero aquellos que quieran cambiar la prioridad de un elemento tendrán que convencerle. También es el responsable de la financiación necesaria para el proyecto, de tomar las decisiones el resultado final, fechas de lanzamiento y el retorno de inversión. Por regla general y si no se trata de proyectos internos, el propietario del producto suele ser el responsable del proceso de adquisición del cliente.

El Equipo de desarrollo. Es el equipo del proyecto y tiene la autoridad para decidir las acciones necesarias y para auto-organizarse con la finalidad de alcanzar los objetivos del sprint. El equipo está formado por desarrolladores con todos los conocimientos necesarios para convertir los requerimientos del Propietario del Producto en un incremento potencialmente utilizable del producto al final del Sprint. Los Equipos de desarrolladores convierten la *pila del producto* en incrementos de funcionalidad potencialmente entregables en cada Sprint. Los Equipos también se auto-organizan. Nadie - ni siquiera el *responsable* - dice al Equipo cómo convertir el *Pila del producto* en incrementos de funcionalidad entregable. El Equipo busca por su cuenta la mejor forma de hacerlo (Cervantes, 2013) .

2.5.6 Los artefactos

Pila del producto

La ingeniería del software clásica diferencia dos ámbitos de requisitos:

- Requisitos del sistema
- Requisitos del software

Los requisitos del sistema forman parte del proceso de adquisición, y por tanto es responsabilidad del cliente la definición del problema y de las funcionalidades que debe aportar la solución.

La pila del producto es el inventario de funcionalidades, mejoras, tecnología y corrección de errores que deben incorporarse al producto a través de los sucesivos *sprints*.

Representa todo aquello que esperan el cliente, los usuarios, y en general los interesados. Todo lo que suponga un trabajo que debe realizar el equipo debe estar reflejado en esta pila.

Estos son algunos ejemplos de posibles entradas a una pila de producto:

- Permitir a los usuarios la consulta de las obras publicadas por un determinado autor.
- Reducir el tiempo de instalación del programa.
- Mejorar la escalabilidad del sistema.
- Permitir la consulta de una obra a través de un API web

Pila del Sprint

La pila del sprint es la lista que descompone las funcionalidades de la pila del producto (historias de usuario) en las tareas necesarias para construir un incremento: una parte completa y operativa del producto.

La realiza el equipo durante la reunión de planificación del sprint, auto asignando cada tarea a un miembro del equipo, e indicando en la misma lista cuánto tiempo o esfuerzo se prevé que falta para terminarla.

La pila del sprint descompone el trabajo en unidades de tamaño adecuado para monitorizar el avance a diario, e identificar riesgos y problemas sin necesidad de procesos de gestión complejos.

Es también una herramienta para la comunicación visual directa del equipo.

2.5.7 El Incremento

El incremento es la parte de producto producida en un *sprint*, y tiene como característica el estar completamente terminada y operativa, en condiciones de ser entregada al cliente.

No se deben considerar como Incremento a prototipos, módulos o sub-módulos, ni partes pendientes de pruebas o integración.

Idealmente en Scrum:

- Cada elemento de la pila del producto se refiere a funcionalidades entregables, no a trabajos internos del tipo “diseño de la base de datos”.
- Se produce un “incremento” en cada iteración

2.5.8 Los Eventos

Los eventos contienen las tareas necesarias para el desarrollo del controlador, una vez teniendo esto, la siguiente fase es el seguimiento del sprint, la cual puede regresar a la etapa de la planeación del sprint en caso de que sea necesario o continuar con el modelo de desarrollo para el controlador. Una vez terminada la etapa de seguimiento del sprint, se inicia la fase de la revisión del sprint, en la que se pueden requerir cambios o mejoras constantes.

Planificación del sprint

La planificación del sprint está compuesta por 2 partes:

- Pila del producto, comienza con la definición de los elementos que vamos a desarrollar, la codificación e integración de la base de datos y las interfaces, realizando el código correspondiente para el correcto funcionamiento de los distintos módulos que integran el sistema.
- Pila del sprint, donde se realiza el análisis de los datos obtenidos en los requerimientos funcionales para dar paso a la generación del código, ya terminada la codificación se realiza una pequeña prueba del código para su correcto funcionamiento y así dar paso a su implementación en el sistema

Una vez definidos las tareas del sprint sigue la ejecución de tareas o bitácora. Se van a tomar cada una de las tareas definidas y comienza la implementación de la lista de la bitácora.

Seguimiento del Sprint

A lo largo de esta fase se lleva a cabo breves reuniones diarias, donde se verifica el avance de las tareas y el trabajo que está previsto para la jornada de acuerdo al cronograma de actividades.

El cliente del producto revisa que esté lista la tarea y que se cumpla con las especificaciones requeridas, y si es necesario propone algún cambio, hasta que la tarea quede correctamente da paso al siguiente *Sprint*.

Revisión del sprint

Al finalizar un sprint y cuando tengamos listo el proceso de desarrollo es hora de presentar el proyecto, todo esto se presenta a los miembros del equipo, donde se revisa que todas las peticiones del cliente se cumplan. La reunión incluye una demostración de lo que el equipo ha construido durante este periodo.

La retrospectiva, los miembros del equipo analizan sobre las fallas o complicaciones que surgieron y se proponen nuevas formas para la realización de las tareas.

El incremento se realiza cuando se cumplen las etapas de la pila del sprint, dando paso a ejecutar otra actividad de la bitácora, (Palacio, 2015).

2.5.9 Las herramientas

Gráfico de producto.

El gráfico de producto o gráfico “burn up” es una herramienta de planificación del propietario del producto, que muestra visualmente la evolución previsible del producto.

Proyecta en el tiempo su construcción, en base a la velocidad del equipo.

La proyección se realiza sobre un diagrama cartesiano que representa en el eje de ordenadas el esfuerzo estimado para construir las diferentes historias de la pila del producto, y en el de las abscisas el tiempo, medido en *sprints* o en tiempo real. En la figura 2.10 se puede visualizar un ejemplo de lo que contiene (Palacio, 2015).

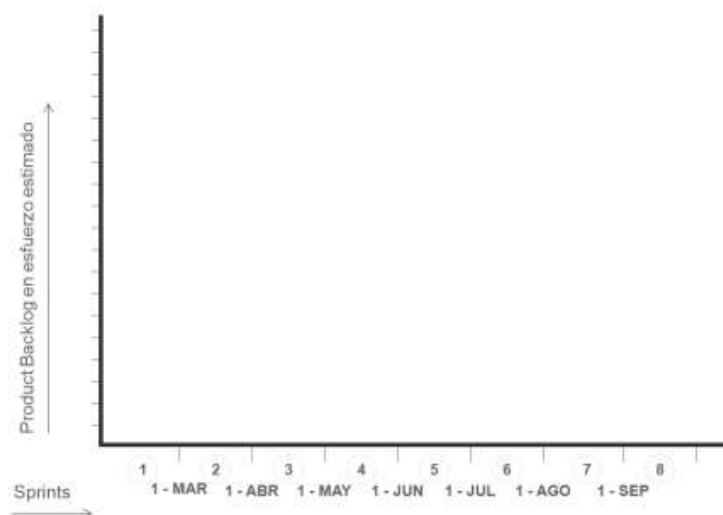


Figura 2.10 Ejemplo de grafica del producto (Palacio, 2015)

Gráfico de avance

También se suele llamar a este gráfico con su nombre inglés: burn-down”.

Lo actualiza el equipo en el Scrum diario, para comprobar el ritmo de avance, y detectar desde el primer momento si es el previsto, o por el contrario se puede ver comprometida o adelantada la entrega prevista al final de *sprint*. En la figura 2.11 se puede visualizar lo que contiene.

La estrategia ágil para el seguimiento del proyecto se basa en:

- Medir el trabajo que falta, no el realizado.
- Seguimiento cercano del avance (diario de ser posible).

Y este gráfico trabaja con ambos principios:

- Registra en el eje Y el trabajo pendiente.
- Se actualiza a diario.



Figura 2.11 Gráfica de avance (Palacio, 2015)

2.6 Ingeniería de requerimientos

Los requerimientos para un sistema son descripciones de lo que el sistema debe hacer: el servicio que ofrece y las restricciones en su operación. Tales requerimientos reflejan las necesidades de los clientes por un sistema que atienda cierto propósito, como sería controlar un dispositivo, colocar un pedido o buscar información. Al proceso de descubrir, analizar, documentar y verificar estos servicios y restricciones se le llama ingeniería de requerimientos (IR).

El término “requerimiento” no se usa de manera continua en la industria del software. En algunos casos, un requerimiento es simplemente un enunciado abstracto de alto nivel en un servicio que debe proporcionar un sistema, o bien, una restricción sobre un sistema. En el otro extremo, consiste en una definición detallada y formal de una función del sistema.

Algunos de los problemas que surgen durante el proceso de ingeniería de requerimientos son resultado del fracaso de hacer una separación clara entre esos diferentes niveles de descripción. En este texto se distinguen con el uso del término “requerimientos del usuario” para representar los requerimientos abstractos de alto nivel; y “requerimientos del sistema” para caracterizar la descripción detallada de lo que el sistema debe hacer. Los requerimientos del usuario y los requerimientos del sistema se definen del siguiente modo:

1. Los requerimientos del usuario son enunciados, en un lenguaje natural junto con diagramas, acerca de qué servicios esperan los usuarios del sistema, y de las restricciones con las cuales éste debe operar.
2. Los requerimientos del sistema son descripciones más detalladas de las funciones, los servicios y las restricciones operacionales del sistema de software. El documento de requerimientos del sistema (llamado en ocasiones especificación funcional) tiene que definir con exactitud lo que se implementará. Puede formar parte del contrato entre el comprador del sistema y los desarrolladores del software, (Sommerville, 2011).

Las fuentes más comunes de requisitos de software son clientes, compradores, agencias de adquisición, usuarios, operadores, dominio, expertos, analistas de sistemas, las especificaciones del sistema, etc.

Las principales actividades de ingeniería de requisitos de software incluyen requisitos de la obtención, análisis, especificación y validación.

Obtención de requisitos de software- es la tarea de adquirir los requisitos del software. Se considera la etapa donde los requisitos ingenieros intentan averiguar lo que las partes interesadas desean realmente.

Análisis de requerimientos de software "las tareas de organizar, interpretar, comprender y clasificar los requisitos de software". También se refiere a las actividades de verificación de la exhaustividad, coherencia y viabilidad de los requisitos del software. Puede ayudar a descubrir las anomalías entre los límites del software, a la comprensión de las interacciones con su entorno y elaborar los requisitos del sistema para obtener los requisitos de software. En esta etapa, realmente los requisitos técnicos se centran en los problemas y tratar de comprender a aquellos sin concentrarse en la solución. Otro tipo de modelos como los modelos de flujo de datos, modelos de estado y los modelos de objetos, etc. están contruidos y en algún momento, la arquitectura de la solución debe ser derivada. Este es el punto en donde los requisitos y el proceso de diseño se superponen con los otros.

Las especificaciones de requisitos de software- se refieren a las actividades que describen los requisitos de software de una manera formal de especificación de requisitos de software generalmente o SRS. SRS actúa como la base para un acuerdo entre el lado del cliente y los contratistas o proveedores al lado de "lo que el producto de software es hacerlo tan bien como lo que no se espera que lo haga".

Validación de requisitos de software es la actividad de obtención de la aprobación de la especificación del software. Los documentos de requisitos deben ir a través de los procedimientos de verificación y validación para asegurarse de que todos los problemas son reconocidos relacionados con los requisitos. También ayuda a

asegurarse de que son los requisitos de derecho o en otra forma, ayudará a construir el software adecuado. (Syed & Ho-Jin, 2007)

Para la adquisición y análisis de requerimientos se utilizó un proceso iterativo con retroalimentación continua, se utilizó el modelo en espiral para pasar de una actividad a otras actividades. Las actividades están entrelazadas en un proceso iterativo alrededor de un espiral como se muestra en la figura 2.12.

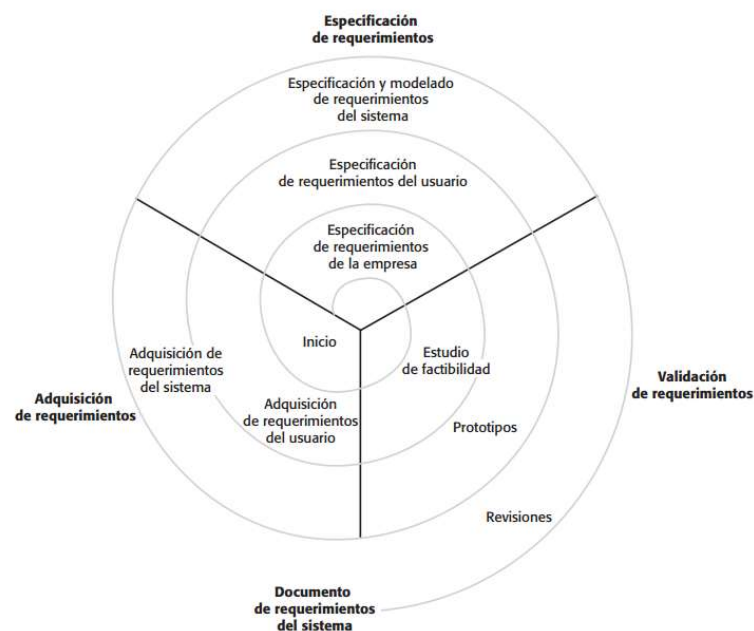


Figura 2.12 Proceso en espiral para la ingeniería de requerimientos (Sommerville, 2011)

2.6.1 Identificación de los casos de uso

Este apartado permite identificar en términos generales cual es la funcionalidad que tendrá el producto, expresando cada requerimiento funcional como un caso de uso. A cada uno se le asignara un número único, nombre, prioridad y complejidad como se muestra en la tabla 2.1

Tabla 2.1 Requerimientos funcionales

| Número | Nombre | Prioridad | Complejidad |
|--------|--------|-----------|-------------|
| | | | |
| | | | |

Referencias de la tabla

Numero de caso de uso: es un número correlativo que se asigna conforma se van identificando los casos de uso y así facilita su identificación.

Nombre del caso de uso: es una frase representativa y clara de la función que va a realizar conforme al caso de uso y este no debe repetirse.

Prioridad: se utiliza para clasificar al caso de uso por su importancia para el negocio comparada con los otros casos de uso del sistema, los valores de clasificación posibles son útil, esencial y deseable.

- Esencial si la existencia del caso de uso es imprescindible para el cumplimiento del propósito del producto, debería clasificarse el caso de uso como imprescindible
- Útil si el caso de uso optimiza el funcionamiento del producto, pero su no existencia no compromete el cumplimiento del propósito del producto, debería clasificarse el caso de uso como útil.
- Deseable si el caso de uso aporta funcionalidad interesante para los usuarios finales, pero la misma es prescindible, y no impacta en absoluto en el cumplimiento del objetivo del producto, debería clasificarse el caso de uso como deseable

Complejidad: la complejidad clasifica a los casos de uso en función del esfuerzo que demanda su especificación y construcción, los valores posibles son:

- Simple son funciones de administración de los datos (tipo ingresos, eliminaciones, modificaciones y consultas) con acceso a una única entidad de datos con hasta 10 campos de entrada, con validaciones simples.
- Mediano son funciones de administración de los datos (tipo ingresos, eliminaciones, modificaciones y consultas) con acceso a varias entidades de datos (para consultas o referencias) con hasta 15 campos de entrada, con validaciones simples.
- Complejo son funciones tipo cabecera detalle o consultas con criterios, con acceso a varias entidades de datos y validaciones de consistencia.
- Muy Complejo son funciones esenciales, que corresponden con la resolución de la lógica de negocio en la aplicación, conllevan algoritmos computacionales complicados.
- Extremadamente Complejo en este nivel de complejidad se reserva para aquella funcionalidad que realmente trae aparejado un nivel de dificultad tal, que, por sus algoritmos computacionales o su lógica asociada, va a requerir un tiempo de resolución importante. Puede que existan proyectos que no tengan este tipo de casos de uso, y de existir, por lo general son pocos en cantidad.

2.6.2 Descripción de los actores

Esta sección muestra una identificación de sectores del ambiente que tengan una vinculación con el producto de software a construir. Se debe especificar para cada actor el nombre, categoría y tipo e incorpora una breve descripción del rol que cumple el actor en relación al sistema como se puede observar en la tabla 2.2.

Tabla 2.2 Descripción de los actores

| Nombre del actor | Descripción | Tipo | Categoría |
|------------------|-------------|------|-----------|
| | | | |

Referencias de la tabla:

- Nombre del Actor: es un nombre representativo del rol que cumple el actor con respecto al producto.
- Descripción: se proporciona una breve explicación que permita determinar las características esperables para ese actor.
- Categoría: los valores posibles para la categorización de los actores según su categoría son:
- Persona Esta categoría representa roles desempeñados por personas que se van a relacionar directamente con uno o más casos de uso del producto.
- Hardware Esta categoría representa roles de computadoras o algún otro dispositivo con el que uno o más casos de uso del sistema deberán interactuar.
- Software Esta categoría representa la relación de uno o más casos de uso con software que es externo y fuera del control del producto que se va a construir.

Tipo: los actores se pueden clasificar para clarificar su rol con relación al producto en dos tipos que son:

- Concreto un actor concreto es aquel que tiene un rol específico en relación a uno o más casos de uso del sistema.
- Abstracto cuando actores diferentes juegan roles comunes pueden abstraer ese comportamiento en un actor común, denominado actor abstracto.

2.6.3 Reglas del negocio

Se incluyen todas las operaciones de forma detallada para que el sistema pueda ejecutar las tareas correspondientes como se muestra en la tabla 2.3. Estas operaciones estarán situadas en el módulo del controlador, siendo ejecutadas las veces que sean necesarias para que el sistema tenga una operación correcta.

Las operaciones estarán encargadas de calcular y proporcionar la información requerida por los usuarios.

Tabla 2.3 Reglas del negocio

| Operación | Descripción |
|-----------|-------------|
| | |

Referencias de la tabla

- Operación se compone del nombre de la operación.
- Descripción: contiene las operaciones de forma detallada.

2.6.4 Requerimientos funcionales

Requerimientos funcionales. Son enunciados acerca de servicios que el sistema debe proveer, de cómo debería reaccionar el sistema a entradas particulares y de cómo debería comportarse el sistema en situaciones específicas. En algunos casos, los requerimientos funcionales también explican lo que no debe hacer el sistema.

Tales requerimientos dependen del tipo de software que se esté desarrollando, de los usuarios esperados del software y del enfoque general que adopta la organización cuando se escriben los requerimientos. Al expresarse como requerimientos del usuario, los requerimientos funcionales se describen por lo general de forma abstracta que entiendan los usuarios del sistema. Sin embargo, requerimientos funcionales más específicos del sistema detallan las funciones del sistema, sus entradas y salidas, sus excepciones, etcétera, (Sommerville, 2011).

Se describen cada uno de los casos de uso identificados y listados con anterioridad para especificarse en su forma detallada como se muestra en la tabla 2.4.

Tabla 2.4 Descripción detallada de los casos de uso

| | | |
|---|------------------------|--|
| Nombre del Caso de Uso: | | ID: |
| Prioridad: <input type="checkbox"/> Esencial <input type="checkbox"/> Útil <input type="checkbox"/> Deseable | | Significativo para la Arquitectura: <input type="checkbox"/> Si <input type="checkbox"/> No |
| Complejidad: <input type="checkbox"/> Simple <input type="checkbox"/> Mediano <input type="checkbox"/> Complejo <input type="checkbox"/> Muy Complejo <input type="checkbox"/> Extremadamente Complejo | | |
| Actor Principal: | | Actor Secundario: |
| Tipo de Caso de Uso: <input type="checkbox"/> Concreto <input type="checkbox"/> Abstracto | | |
| Objetivo: | | |
| Precondiciones: Ninguna | | |
| Post- Condiciones | Éxito: | |
| | Fracasos: 1) | |
| Curso Normal | | Alternativas |
| 1. | | |
| Observaciones: | | |

Descripción de la plantilla

- Nombre del caso de uso: Cada requerimiento debe contener un nombre que debe ser igual al de la lista de requerimientos.
- Id: Cada requerimiento cuenta con un id que se le asigna desde la lista de requerimientos, siendo único y no se puede repetir.
- Complejidad: se clasifica a los casos de uso en función del esfuerzo que demanda su especificación y construcción, los valores posibles que se les asigno en la lista de requerimientos.
- Actores: representar el tipo de usuario del sistema, no tiene que ser siempre humano, puede ser también un sistema informático.

- Tipo de caso de uso, se selecciona conforma al rol con relación al producto.
- Objetivo del caso de uso, se describe la función principal que realizara el caso de uso.
- Precondiciones: son los requisitos que deben cumplirse antes para poder realizar la tarea actual.
- Post-condiciones: son las actividades que se tienes después ejecutar el caso de uso, las cueles se dividen en:
 - Éxito si se cumple con el objetivo.
 - Fracaso: es el comportamiento del sistema en caso de que se produzca una situación de falla o que no permite concluir como son: errores, mensajes o fallas del sistema.
- Curso normal: estos campos contienen la secuencia normal de interacciones del caso de uso, para realizar completar el objetivo.
- Alternativas: secuencias que se desprenden de las interacciones normales del curso normal.
- Observaciones: son notas que se deben tener en cuenta o consideraciones a tomar.

2.6.5 Requerimientos no funcionales

Requerimientos no funcionales Son limitaciones sobre servicios o funciones que ofrece el sistema. Incluyen restricciones tanto de temporización y del proceso de desarrollo, como impuestas por los estándares. Los requerimientos no funcionales se suelen aplicar al sistema como un todo, más que a características o a servicios individuales del sistema

Los requerimientos no funcionales, como indica su nombre, son requerimientos que no se relacionan directamente con los servicios específicos que el sistema entrega a sus usuarios. Pueden relacionarse con propiedades emergentes del sistema, como

fiabilidad, tiempo de respuesta y uso de almacenamiento. De forma alternativa, pueden definir restricciones sobre la implementación del sistema, como las capacidades de los dispositivos I/O u las representaciones de datos usados en las interfaces con otros sistemas. (Sommerville, 2011)

Los requerimientos no funcionales son asignados comúnmente en lenguaje natural en esta sección de especificación. Los requerimientos son aplicables al producto en general. Para el caso de los requerimientos no funcionales aplicables a un caso de uso en particular, además de referenciar la descripción desde la especificación de ese caso o casos de uso.

Los requerimientos no funcionales surgen a través de necesidades del usuario, debido a restricciones presupuestales, políticas de la organización, necesidad de interoperabilidad con otro software o sistemas de hardware, o factores externos como regulaciones de seguridad o legislación sobre privacidad. La figura 2.10 es una clasificación de requerimientos

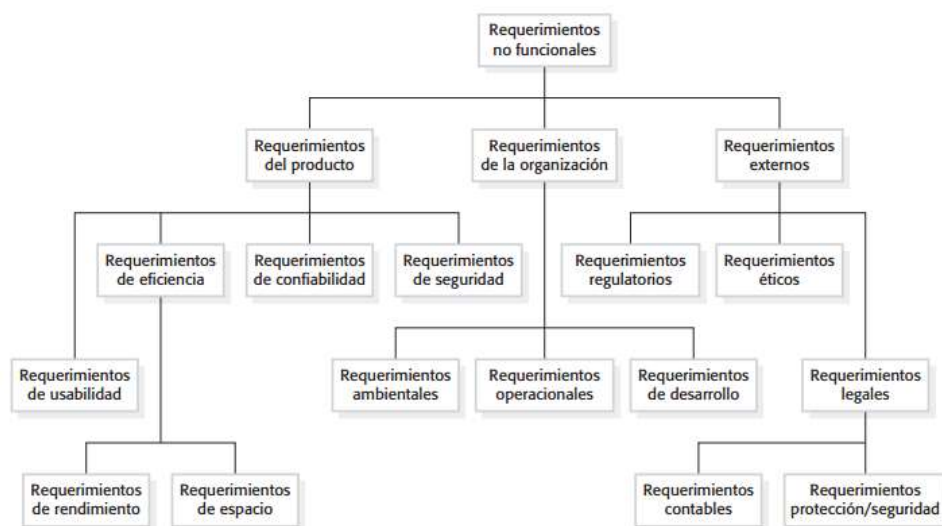


Figura 2.13 Tipos de requerimientos no funcionales (Sommerville, 2011)

En las siguientes tablas se muestra la plantilla utilizada para listar los requerimientos no funcionales, estos son separados en requerimientos no funcionales del producto en

la tabla 2.5, los requerimientos no funcionales de la organización como se observa en la tabla 2.6 y en los requerimientos no funcionales externos en la tabla 2.7.

Tabla 2.5 Requerimientos no funcionales del producto

| Requerimientos no funcionales del producto | | |
|--|-----------|-------------|
| Nro. | Categoría | Descripción |
| | | |

Tabla 2.6 Requerimientos no funcionales de organización

| Requerimientos no funcionales de organización | | |
|---|-----------|-------------|
| Nro. | Categoría | Descripción |
| | | |

Tabla 2.7 Requerimientos no funcionales externos

| Requerimientos no funcionales externos | | |
|--|-----------|-------------|
| Nro. | Categoría | Descripción |
| | | |

Referencias de las tablas

- **Nro:** contiene el un valor descriptivo para identificar el requerimiento no funcional.
- **Categoría:** se identifica a que categoría pertenece de los requerimientos no funcionales de la figura 2.10.
- **Descripción:** se describe el significado del requerimiento.

2.7 Modelado

Se crean modelos para entender mejor la entidad real que se va a construir. Cuando ésta es física (por ejemplo, un edificio, un avión, una máquina, etc.), se construye un modelo de forma idéntica, pero a escala. Sin embargo, cuando la entidad que se va a construir es software, el modelo debe adoptar una forma distinta. Debe ser capaz de representar la información que el software transforma, la arquitectura y las funciones que permiten que esto ocurra, las características que desean los usuarios y el comportamiento del sistema mientras la transformación tiene lugar. Los modelos deben cumplir estos objetivos en diferentes niveles de abstracción, en primer lugar, con la ilustración del software desde el punto de vista del cliente y después con su representación en un nivel más técnico.

En el trabajo de ingeniería de software se crean dos clases de modelos: de requerimientos y de diseño. Los modelos de requerimientos (también conocidos como modelos de análisis) representan los requerimientos del cliente mediante la ilustración del software en tres dominios diferentes: el de la información, el funcional y el de comportamiento. Los modelos de diseño representan características del software que ayudan a los profesionales a elaborarlo con eficacia: arquitectura, interfaz de usuario y detalle en el nivel de componente.

Principios del modelado del diseño. El modelo del diseño del software es análogo a los planos arquitectónicos de una casa. Se comienza por representar la totalidad de lo que se va a construir (por ejemplo, un croquis tridimensional de la casa) que se refina poco a poco para que guíe la construcción de cada detalle (por ejemplo, la distribución de la plomería). De manera similar, el modelo del diseño que se crea para el software da varios puntos de vista distintos del sistema.

No escasean los métodos para obtener los distintos elementos de un diseño de software. Algunos son activados por datos, lo que hace que sea la estructura de éstos la que determine la arquitectura del programa y los componentes de procesamiento

resultantes. Otros están motivados por el patrón, y usan información sobre el dominio del problema (el modelo de requerimientos) para desarrollar estilos de arquitectura y patrones de procesamiento. Otros más están orientados a objetos, y utilizan objetos del dominio del problema como impulsores de la creación de estructuras de datos y métodos que los manipulan. No obstante, la variedad, todos ellos se apegan a principios de diseño que se aplican sin importar el método empleado. (Pressman, 2010)

2.8 Conceptos de diseño

El diseño de software agrupa el conjunto de principios, conceptos y prácticas que llevan al desarrollo de un sistema o producto de alta calidad. Los principios de diseño establecen una filosofía general que guía el trabajo de diseño que debe ejecutarse. Deben entenderse los conceptos de diseño antes de aplicar la mecánica de éste, y la práctica del diseño en sí lleva a la creación de distintas representaciones del software que sirve como guía para la actividad de construcción que siga.

El objetivo del diseño es producir un modelo o representación que tenga resistencia, funcionalidad y belleza. Para lograrlo, debe practicarse la diversificación y luego la convergencia. Belady afirma que “la diversificación es la adquisición de un repertorio de alternativas, materia prima del diseño: componentes, soluciones con los componentes y conocimiento, todo lo cual está contenido en catálogos, libros de texto y en la mente”. Una vez que se reúne este conjunto diversificado de información, deben escogerse aquellos elementos del repertorio que cumplan los requerimientos definidos por la ingeniería y por el modelo de análisis.

A medida que esto ocurre, se evalúan las alternativas, algunas se rechazan, se converge en “una configuración particular de componentes y, con ello, en la creación del producto final”. La diversificación y la convergencia combinan la intuición y el

criterio con base en la experiencia en la construcción de entidades similares, un conjunto de principios heurísticos que

guían la forma en la que evoluciona el modelo, un conjunto de criterios que permiten evaluar la calidad y un proceso iterativo que finalmente conduce a una representación del diseño definitivo.

El diseño del software cambia continuamente conforme evolucionan los nuevos métodos, surgen mejores análisis y se obtiene una comprensión más amplia. Incluso hoy, la mayor parte de las metodologías de diseño de software carece de profundidad, flexibilidad y naturaleza cuantitativa, que normalmente se asocian con las disciplinas de diseño de ingeniería más clásicas.

No obstante, sí existen métodos para diseñar software, se dispone de criterios para el diseño con calidad y se aplica la notación del diseño. En este capítulo, se estudian los conceptos y principios fundamentales aplicables a todo el diseño de software, los elementos del modelo del diseño y el efecto que tienen los patrones en el proceso de diseño. En los capítulos 9 a 13 se presentarán varias metodologías de diseño de software, según se aplican en la obtención de arquitecturas e interfaces en el nivel de componente, así como a enfoques de diseño basados en patrones y orientados a web. (Pressman, 2010)

2.9 Evaluación de software

La calidad en la ingeniería del software, que depende en gran medida de la pericia del equipo que lo desarrolla, puede definirse como un conjunto de características o cualidades, tales como: eficiencia, fiabilidad, usabilidad, funcionalidad, mantenibilidad, portabilidad, etc., variando la importancia de cada una de ellas de un producto a otro. Dicho de otra manera, es el cumplimiento de los requisitos contractuales por parte del producto software desarrollado, así como durante el proceso de desarrollo. (Inteco, 2008)

2.9.1 Calidad

Según David Garvin, de Harvard Business School, sugiere que “la calidad es un concepto complejo y de facetas múltiples” que puede describirse desde cinco diferentes puntos de vista.

1. *El punto de vista trascendental* que la calidad es algo que se reconoce de inmediato, pero que no es posible definir explícitamente.
2. *El punto de vista del usuario* concibe la calidad en términos de las metas específicas del usuario final. Si un producto las satisface, tiene calidad.
3. *El punto de vista del fabricante* la define en términos de las especificaciones originales del producto. Si éste las cumple, tiene calidad.
4. *El punto de vista del producto* sugiere que la calidad tiene que ver con las características inherentes (funciones y características) de un producto.
1. *El punto de vista basado en el valor* la mide de acuerdo con lo que un cliente está dispuesto a pagar por un producto, (Pressman, 2010).

2.9.2 Modelo en V

El modelo en V se suele entender como una metodología de testing, sin embargo, se trata en realidad de una adaptación del ciclo de vida clásico o en cascada realizado por el gobierno federal alemán.

La V del modelo representa a dos secuencias de fases, la primera se corresponde con la secuencia de fases de desarrollo del proyecto y la segunda con la secuencia de fases de testing del proyecto.

Ventajas:

- La relación entre las etapas de desarrollo y los distintos tipos de pruebas facilitan la localización de fallos.

- Es un modelo sencillo y de fácil aprendizaje
- Hace explícito parte de la iteración y trabajo que hay que revisar
- Especifica bien los roles de los distintos tipos de pruebas a realizar
- Involucra al usuario en las pruebas

Desventajas:

- Es difícil que el cliente exponga explícitamente todos los requisitos
 - El cliente debe tener paciencia pues obtendrá el producto al final del ciclo de vida
 - Las pruebas pueden ser caras y, a veces, no lo suficientemente efectivas
 - El producto final obtenido puede que no refleje todos los requisitos del usuario.
- (TutorialsPoint, 2017)

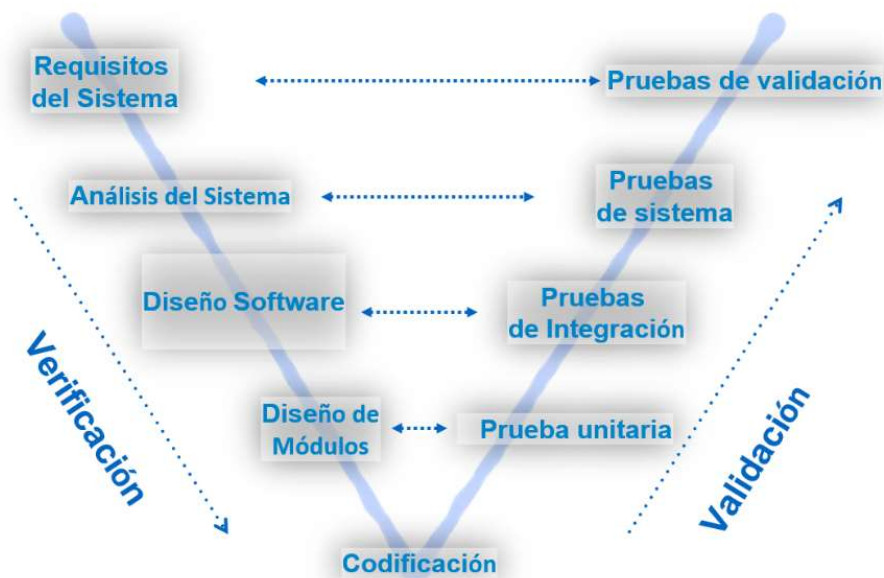


Figura 2.14 Modelo en V (TutorialsPoint, 2017)

En la figura 2.14 se muestran las fases del modelo, del cual tomaremos de base para poder crear nuestro propio modelo, dependiendo de las necesidades y especificaciones del software, a continuación, explicaremos el modelo:

- Lado izquierdo: especificaciones de los requerimientos del servicio hasta el detalle del diseño del servicio.
- Lado derecho: se focaliza en las actividades de validación que se llevan a cabo en contra de las especificaciones definidas a la izquierda.
- A cada paso de la izquierda, hay implicación directa con la parte equivalente en el lado derecho.
- Dentro de cada ciclo de desarrollo repetitivo se pueden aplicar los conceptos del modelo V sobre los requerimientos de aprobación de estabilidad contra el diseño, con cada diseño repetitivo considerado para el grado de integridad y competencia que justificara el lanzamiento al cliente para juicio y valoración.

Dentro del modelo en V están las siguientes estrategias de pruebas, que son el tipo de problema a descubrir. Las pruebas más típicas son:

- **Pruebas de Unidad:** el cual tiene como objetivo localizar defectos y probar el funcionamiento del software, por ejemplo, datos válidos y erróneos al ingresar al sistema, y como responde el sistema ante dichas pruebas.
- **Pruebas de Integración:** En cual se debe checar detalladamente el flujo de la información entre los módulos y descubrir errores asociados con la interfaz.
- **Pruebas de Sistema:** en el cual se debe de revisar el funcionamiento del software, donde tiene que constar que cumpla con los requisitos funcionales y no funcionales.
- **Pruebas de Aceptación:** En esta prueba es donde, el cliente nos confirma si el software cumple con todo lo que pidió y nos confirma de igual manera si es exitoso o no.

2.9.3 Modelo en espiral

Una estrategia para probar el software también puede verse en el contexto de la espiral (figura 2.15). La prueba de unidad comienza en el vértice de la espiral y se concentra en cada unidad (por ejemplo, componente, clase o un objeto de contenido de una webapp) del software como se implementó en el código fuente. La prueba avanza al moverse hacia afuera a lo largo de la espiral, hacia la prueba de integración, donde el enfoque se centra en el diseño y la construcción de la arquitectura del software. Al dar otra vuelta hacia afuera de la espiral, se encuentra la prueba de validación, donde los requerimientos establecidos como parte de su modelado se validan confrontándose con el software que se construyó. Finalmente, se llega a la prueba del sistema, donde el software y otros elementos del sistema se prueban como un todo, (Pressman, 2010).

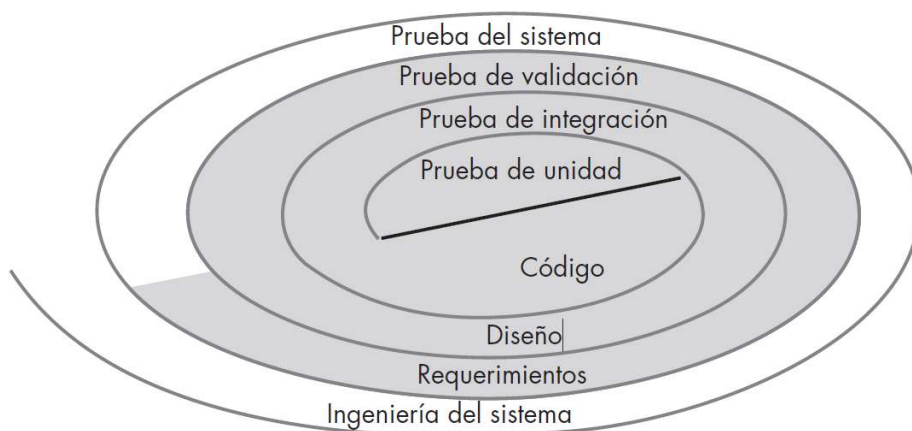


Figura 2.15 Estrategia de pruebas (Pressman, 2010)

Inicialmente, las pruebas se enfocan en cada componente de manera individual, lo que garantiza que funcionan adecuadamente como unidad. De ahí el nombre de prueba de unidad. Esta prueba utiliza mucho de las técnicas de prueba que ejercitan rutas específicas en una estructura de control de componentes para asegurar una cobertura

completa y la máxima detección de errores. A continuación, los componentes deben ensamblarse o integrarse para formar el paquete de software completo. La prueba de integración aborda los conflictos asociados con los problemas duales de verificación y construcción de programas. Durante la integración, se usan más las técnicas de diseño de casos de prueba que se enfocan en entradas y salidas, aunque también pueden usarse técnicas que ejercitan rutas de programa específicas para asegurar la cobertura de las principales rutas de control.

Después de integrar (construir) el software, se realiza una serie de pruebas de orden superior. Deben evaluarse criterios de validación (establecidos durante el análisis de requerimientos). La prueba de validación proporciona la garantía final de que el software cumple con todos los requerimientos informativos, funcionales, de comportamiento y de rendimiento.

Ventajas:

- Resolución temprana de riesgos.
- Definición de arquitectura en sus fases iniciales.
- Basado en un proceso continuo de verificación de la calidad.
- Ideal para productos con un nivel alto de inestabilidad de los requerimientos.

Desventajas:

- No aplicable a proyectos bajo contrato.
- No recomendable en proyectos simples. (TutorialsPoint, 2017)

2.10 Cotizador Web

Es aquel documento o información que el departamento de compras usa en una negociación. Es un documento informativo que no genera registro contable. Cotización son la acción y efecto de cotizar (poner precio a algo, estimar a alguien o algo en relación con un fin, pagar una cuota).

Es un sistema de cotizaciones en línea, donde permite al personal administrar y enviar cotizaciones en minutos de sus productos o servicios, reduciendo el tiempo de respuesta y brindando una mejor solución al cliente.

Son sistemas versátiles que le permite a usted la comparación de productos o servicios para tomar o elegir lo que más convenga. Cuentan con módulos configurables para que cubran las necesidades del negocio.

Algunas características

- Contiene algoritmos matemáticos que automatizan las operaciones, agilizando los cálculos.
- Manejo de las cotizaciones en línea, al estar en una maquina se puede tener acceso en cualquier momento, para consultar cotizaciones anteriores.
- Seguimiento del estatus de las cotizaciones, se puede dar seguimiento a las cotizaciones en las distintas etapas que se encuentren, además de realizar alguna modificación haciendo que la cotización cambie de etapa.
- Bases de datos con información actualizada, al contar una base de datos se puede acceder a la información actualizada en cualquier momento.
- Generación de cotizaciones en distintos formatos (PDF, TXT, XML), al contar con la información en una base de datos se puede acceder en cualquier momento y obtener la información en distintos formatos.
- Módulos para la administración de los clientes, se puede aplicar distintas restricciones o configuración para el acceso a los sistemas de cotizaciones.
- Apartados de configuración con respecto a los tipos de moneda, algunos sistemas cuentan con un apartado que permite la modificación del tipo de moneda para los cálculos de precios.

2.11 Tecnologías de desarrollo Java

JavaBean

Un buen número de desarrolladores Java web piensan que los JavaBeans son simplemente clases con algunas propiedades expuestas a través de métodos getter y setter (descriptores de acceso get y set).

Por ejemplo, una clase Java con los métodos getName y setName expone una propiedad llamada escritura y lectura de nombre. Sin embargo, las propiedades son sólo la punta del iceberg; JavaBeans es una arquitectura de componentes de pleno derecho diseñado con soporte de herramientas en mente.

Esto es significativo, porque significa que hay mucho más a él que sólo propiedades. JavaBeans se ajustan a un conjunto de patrones que permiten a otras clases de Java para descubrir dinámicamente eventos y otros metadatos, además de propiedades. Como cuestión de hecho, JavaBeans es la tecnología que permite swing y hace posible que los entornos de desarrollo para proporcionar constructores GUI para aplicaciones de escritorio y applets.

El uso de JavaBeans, se puede desarrollar un componente que no sólo coopera bien con un constructor de interfaz gráfica de usuario visual, sino también proporciona un asistente especializado (o personalizador) para caminar al usuario a través del proceso de configuración. JavaBeans también incluye un potente modelo de eventos (el mismo utilizado con Swing y componentes JSF), servicios de persistencia, y otras interesantes características.

Entender el poder de JavaBeans le ayudará a comprender el poder de JSF. Al igual que los componentes Swing, todos los componentes JSF es un JavaBean de pleno derecho. Además, se enfrenta a los componentes están diseñados para trabajar con el respaldo de los granos - objetos que se implementan como JavaBeans y también controlar los eventos, (MANN, 2005).

Peticiones get y post

Una petición get obtiene (o recupera) la información de un servidor. Dichas peticiones comúnmente recuperan un documento HTML o una imagen.

Una petición post envía datos a un servidor, como la información de autenticación o los datos de un formulario que recopilan la entrada del usuario. Por lo general, las peticiones post se utilizan para enviar un mensaje a un grupo de noticias o a un foro de discusión, pasar la entrada del usuario a un proceso manejador de datos en el servidor, y almacenar o actualizar los datos en un servidor que éste ocupa, (Deitel & Deitel, 2008).

2.12 Comentarios

En este Capítulo se mencionó los conceptos del patrón de diseño modelo vista controlador y las características de cada componente. Una vez que se comprendió el funcionamiento del patrón de diseño, se analiza la parte controlador que es el vínculo entre el usuario y el sistema.

Se mencionó el funcionamiento del controlador con los demás componentes, su interacción y la forma en como la información fluye para que un sistema funcione correctamente, ya que se tienen distintas formas para la implementación en el flujo de información.

Se describieron los diferentes modelos para el proceso del software, como son el modelo en cascada, el incremental, construcción de prototipos, el modelo de desarrollo rápido de aplicaciones, lineal secuencial y el modelo en espiral para tener una perspectiva de los diferentes modelos y su funcionamiento.

Se realizó un análisis a las diferentes metodologías ágiles, donde se presentó su historia y su evolución al paso del tiempo, una descripción de cada una y de cómo es

la metodología ágil de desarrollo adaptativo de software (DAS), el método de desarrollo de sistema dinámico (MDSD), metodología ágil cristal y la metodología ágil Scrum.

También se pudo contemplar detalladamente la metodología ágil Scrum ya que esta es la que se está utilizando, por lo que se mencionaron todos los componentes que lo conforman.

Se abordó el tema de la ingeniería de requisitos ya que es fundamental para la recolección, análisis y documentación de los requisitos, donde se analiza y se utiliza un proceso en espiral para la recolección de los requisitos, ya que las actividades se entrelazan en un proceso iterativo o incremental.

Se identificaron las diferentes tablas para los casos de uso, descripción de los autores, las reglas del negocio y los requerimientos funcionales y no funcionales, así como las plantillas a utilizar.

Se mencionan conceptos de modelado y diseño ya que son una parte fundamental para el desarrollo del sistema. El modelado de la aplicación es una parte fundamental para representar la información de todo el sistema en diagramas. Así mismo el diseño web ya que abarca diferentes aspectos que son de suma importancia para la experiencia del usuario hacia el sistema

3 Metodología

En este Capítulo se describe la metodología utilizada para el desarrollo de un sistema cuyo objetivo es el de realizar las cotizaciones con el mínimo trabajo posible por parte del operador, bajo la combinación del MVC (Modelo Vista Controlador) y la metodología ágil Scrum para el desarrollo de las partes que integran el sistema.

3.1 Metodología general del sistema

El objetivo de este trabajo comprende el desarrollo e implementación de un sistema de información, cuyo propósito es poder generar cotizaciones con el mínimo de trabajo posible, tratando de brindar confiabilidad, eficiencia y exactitud de la información. Asimismo, se pretende reducir el tiempo para generar una cotización por parte del usuario.

Con la implementación de la arquitectura de diseño modelo-vista-controlador, y la metodología ágil Scrum para el desarrollo de las demás partes que integran el sistema.

3.1.1 Modelo Vista Controlador

El MVC (Modelo - Vista - Controlador) aplicado al sistema cotizador, divide el trabajo en 3 componentes, separando y dando una pauta para el desarrollo de cada uno de los componentes, los cuales se pueden apreciar en la figura 3.1.



Figura 3.1 Patrón de diseño Modelo Vista Controlador

- **El modelo** proporciona la base de datos, la cual contiene la información que el controlador necesita para realizar operaciones de inserción, eliminación, consulta o actualización de datos.

Este módulo contiene información referente a: clientes, usuarios, materiales, productos, tratamientos, acabados, maderas, servicios y cotizaciones.

- **La vista** representa la forma más legible y amigable, donde se representan las acciones que los usuarios están realizando y así puedan interactuar fácilmente, proporciona información a la lógica de negocios mediante Java y recibe la información para ser visualizada.

- **El controlador** es el elemento más importante ya que gestiona y administra los eventos enviados por las interfaces y el modelo.

Las interfaces hacen una petición a los JavaBean en Java, estos se conectan y realizan las tareas necesarias mediante Hibernate a la base de datos, la base de datos proporciona la información obtenida a Hibernate, el cual responde a

los JavaBean y estos a la vistas para las funciones de agregar, modificar, eliminar o buscar materiales, clientes, instalación y servicios, asimismo realiza el cálculo de las distintas operaciones correspondientes para la realización de una cotización, gestiona el permiso y acceso de cada usuario, y genera las búsquedas correspondientes así como los reportes.

3.2 Scrum aplicado al sistema

3.2.1 Definición de los roles

Para el desarrollo del sistema cotizador se contemplan lo roles mencionados en la sección 2.7.5.

Propietario del producto: El director general de la empresa fabricante de vigas de madera laminada encargado de desempeñar este rol, y entre sus principales actividades:

- La reunión inicial con el equipo de trabajo, donde se obtienen los requisitos del sistema.
- La reunión mensual con el equipo de trabajo, para la revisión de cada avance de los *Sprint* establecidos.
- Es responsable de verificar que cada incremento del sistema funcione correctamente y satisfaga las necesidades de la empresa.
- Es responsable de realizar las observaciones necesarias y pertinentes, así como solicitar cambios al equipo de trabajo.

Equipo de desarrollo: Está integrado por un equipo de 3 personas para el desarrollo del sistema, quienes tienen que realizar diferentes actividades, pero todas enfocadas al proyecto, para después integrar cada parte en el *sprint* y así obtener un incremento del sistema.

Scrum Manager: Esta persona está encargada de dirigir al equipo, teniendo como principales actividades:

Asegurarse que existan una lista de requisitos, planificaciones, reuniones, sincronización del equipo y quitar los impedimentos.

En la Figura 3.2, se muestra la metodología general para el desarrollo basada en Scrum, la cual está conformada por el planificador del sprint que se desprende de las entrevistas con el cliente, recopilación y análisis de requisitos.

A partir de las entrevistas contenidas en la Pila del sprint, se realizará el análisis, diseño, codificación, entrega parcial de prototipos y pruebas.

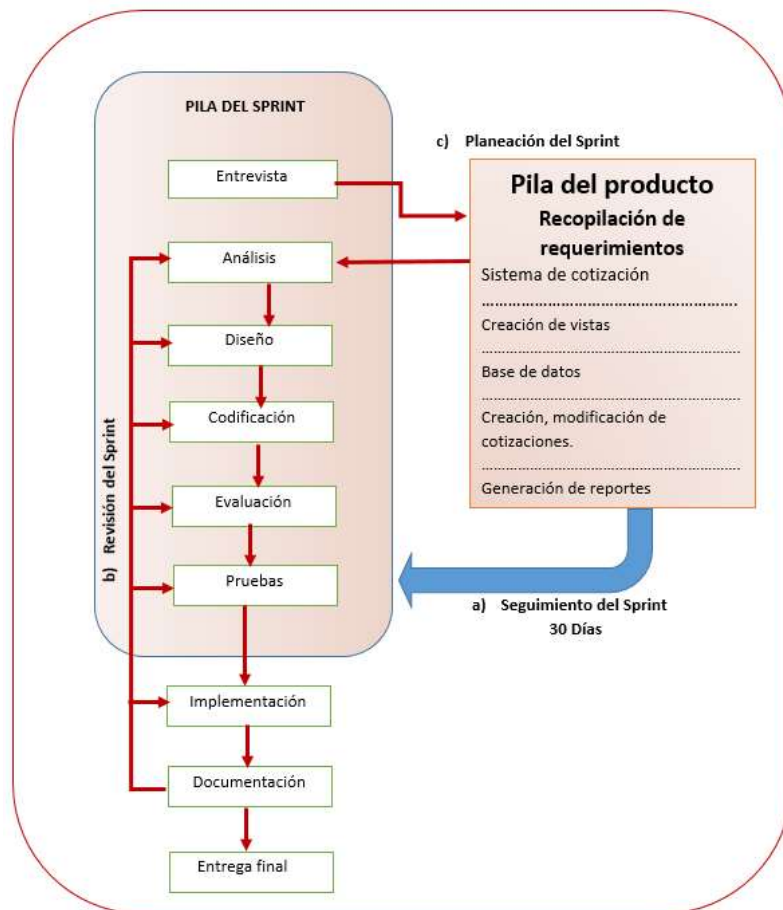


Figura 3.2 Metodología general para el desarrollo del sistema basado en Scrum

3.2.2 Seguimiento del Sprint

Cada iteración constará de 30 días y en la etapa de implementación se verificará el funcionamiento del producto por parte del cliente, una vez cubierta la etapa de implementación se crearán los manuales, dando como resultado la entrega final del producto.

Después de cada tarea se revisa que todos los puntos de la bitácora se cumplan, además si existiera un problema se puede replantear y asignar tareas a otro miembro del equipo, también se puede regresar a cualquier etapa anterior.

3.2.3 Revisión del Sprint.

Se enlistan los puntos que se deberán seguir para el desarrollo del proyecto.

Pila del sprint, Esta parte se conforma por la lista que divide las tareas de la pila del producto, requerimientos funcionales, pruebas y las tareas necesarias para construir un incremento al sistema cotizador que se pueden visualizar en la tabla 3.1.

Tabla 3.1 Pila del Sprint

| Número | Actor | Nombre | Prioridad | Complejidad | Sprint |
|--------|------------|-----------------------|-----------|-------------|--------|
| 001 | AD, JA, OP | Autenticar usuarios | Esencial | Complejo | 2 |
| 002 | AD, JA | Alta de usuarios | Esencial | Complejo | 2 |
| 003 | AD, JA | Listar usuarios | Útil | Mediano | 2 |
| 004 | AD, JA | Modificar usuarios | Esencial | Complejo | 2 |
| 005 | AD, JA | Eliminar usuarios | Esencial | Complejo | 2 |
| 006 | AD, JA | Alta de clientes | Esencial | Complejo | 3 |
| 007 | AD, JA | Listado de clientes | Útil | Mediano | 3 |
| 008 | AD, JA | Modificar de clientes | Esencial | Complejo | 3 |
| 009 | AD, JA | Eliminar de clientes | Esencial | Complejo | 3 |
| 010 | AD, JA | Alta de productos | Esencial | Complejo | 3 |

Tabla 3.2 Pila del Sprint (continuación)

| 011 | AD, JA | Listar productos | Útil | Mediano | 3 |
|-----|------------|--------------------------------|----------|--------------|---|
| 012 | AD, JA | Modificar productos | Esencial | Complejo | 3 |
| 013 | AD, JA | Eliminar productos | Esencial | Complejo | 3 |
| 014 | AD, JA | Alta de materiales | Esencial | Complejo | 4 |
| 015 | AD, JA | Listar materiales | Útil | Mediano | 4 |
| 016 | AD, JA | Modificar materiales | Esencial | Complejo | 4 |
| 017 | AD, JA | Eliminar materiales | Esencial | Complejo | 4 |
| 018 | AD, JA | Alta de madera | Esencial | Complejo | 4 |
| 019 | AD, JA | Listar madera | Útil | Mediano | 4 |
| 020 | AD, JA | Modificar madera | Esencial | Complejo | 4 |
| 021 | AD, JA | Eliminar de madera | Esencial | Complejo | 4 |
| 022 | AD, JA | Alta de tratamiento | Esencial | Complejo | 5 |
| 023 | AD, JA | Listar tratamiento | Útil | Mediano | 5 |
| 024 | AD, JA | Modificar tratamiento | Esencial | Complejo | 5 |
| 025 | AD, JA | Eliminar tratamiento | Esencial | Complejo | 5 |
| 026 | AD, JA | Alta de acabado | Esencial | Complejo | 5 |
| 027 | AD, JA | Listar de acabado | útil | mediano | 5 |
| 028 | AD, JA | Modificar acabado | Esencial | Complejo | 5 |
| 029 | AD, JA | Eliminar acabado | Esencial | Complejo | 5 |
| 030 | AD, JA, US | Reportes de cotizaciones | Esencial | Complejo | 6 |
| 031 | AD, JA, US | Reporte de pie tablón vendidos | Esencial | Complejo | 6 |
| 032 | AD, JA, US | Reporte de orden de producción | Esencial | Complejo | 6 |
| 033 | AD, JA, OP | Generar cotización | Esencial | Muy complejo | 7 |
| 034 | AD, JA, OP | Listar cotización | Esencial | Complejo | 7 |
| 035 | AD, JA, OP | Modificar cotización | Esencial | Complejo | 8 |
| 036 | AD, JA, OP | Eliminar cotización | Esencial | Complejo | 8 |

Los módulos se han asignado conforme a los actores que utilizan el sistema y las abreviaturas quedan de la siguiente forma:

- AD: Administrador
- JA: Jefe de área

- OP: Operador
- US: Usuario

3.2.4 Planeación del Sprint

En este apartado se contemplan a grandes rasgos los requerimientos que se deberán cumplir para el desarrollo del proyecto

Pila del producto, comienza con la definición de los elementos que vamos a desarrollar, la codificación e integración de la base de datos y las interfaces, en la tabla 3.2 se presenta los actores del sistema con los requerimientos funcionales y no funcionales. Todos estos obtenidos del análisis de requerimientos, y descritos de forma simple.

Tabla 3.3 Pila del producto

| Requisito | Descripción |
|-------------|---|
| Funcionales | Administrador <ul style="list-style-type: none"> • Administración de usuarios (alta modificación, eliminación y consultas) • Administración de clientes (alta, modificación, eliminación y consulta) • Administración de materiales (alta, modificación, eliminación y consulta) • Administración de servicio (alta, modificación, eliminación y consulta) • Administración de instalación (alta, modificación eliminación y consulta) • Cotizar (alta, modificación y consulta) |
| | Jefe de área <ul style="list-style-type: none"> • Administración de usuarios (alta modificación, eliminación y consultas) • Administración de clientes (alta, modificación, eliminación y consulta) • Administración de materiales (alta, modificación, eliminación y consulta) • Administración de servicio (alta, modificación, eliminación y consulta) • Administración de instalación (alta, modificación eliminación y consulta) • Cotizar (alta, modificación y consulta) • Reportes (generación, impresión) |
| | Operador <ul style="list-style-type: none"> • Cotizar (alta, modificación y consulta) • Reportes (generación, impresión) |
| | Usuarios <ul style="list-style-type: none"> • Reportes (generación, impresión) |

Tabla 3.4 Pila del producto (continuación)

| Requisito | Descripción |
|----------------|---|
| No funcionales | <ul style="list-style-type: none"> • Usabilidad: El sistema permite el uso de usuarios al mismo tiempo. • Confiabilidad: El sistema deberá evitar el ingreso de información incorrecta por medio de mensajes que el sistema genere. • Eficiencia: El sistema deberá ser utilizado toda la jornada de trabajo. • Portabilidad: El sistema se desarrollará en su totalidad en Java Enterprise Edition y MySQL. • Mantenibilidad: El sistema deberá ser creado pensando en la implementación de módulos futuros. • Implementación: La empresa proporcionará el servidor donde correrá el sistema. • Seguridad: El sistema utiliza usuario y contraseña para el acceso |

Reglas del negocio

En la tabla 3.3 residen las operaciones que ejecutara el programa para el correcto funcionamiento, las cuales son peticiones del usuario para realiza él envió del proceso correspondiente, además de todas las reglas que deben cumplirse y que están establecidas por la empresa para el funcionamiento del cotizador.

Tabla 3.5 Reglas del negocio

| Operación | Descripción | Nombre corto |
|----------------------------|---|-----------------------|
| Factor del salario | 52.72 el cual puede ser modificado por el usuario. | A= Factor del salario |
| Precio del pie tablón | Suma de costo del proceso más la madera | B= Pie tablón |
| Precio de compra de madera | Precio de acuerdo a la selección de la madera seleccionada de la base de datos, multiplicado por un porcentaje del desperdicio. | (B+ madera) *%des=PT |

Tabla 3.6 Reglas del negocio (continuación)

| Operación | Descripción | Nombre corto |
|--|--|----------------------------------|
| Calculo del desperdicio | Costo de la suma del pie tablón en bruto menos la suma del pie tablón en neto entre la suma del pie tablón en neto, todo esto multiplicado por un porcentaje que depende de la selección del ancho de las pulgadas de la tabla | $(Ptb-B)*\%=des$ |
| Calculo del tipo de acabado | Valor obtenido de la BD dependiendo que tipo de acabado que seleccione el cliente | Ac= tipo de acabado |
| recepciones descarga contabilizaciones y acomodo de madera | Total de horas obtenidas de la suma de domo (301) y alfarje (352) entre 61906.95 que es un cálculo que ya se tiene de años anteriores. | $Des=(301+352)*61906.95$ |
| Costos del proceso canteo | El cálculo de estos costos ya está calculado en su porcentaje y se obtienen de la base de datos. | $Cos=x$ $x=valor\ de\ la\ Bd$ |
| total, costos directo proceso de vigas y pisos | Es la suma del precio de compra de madera más el cálculo del tipo de acabado más recepciones de descarga y los costos del proceso. | $Ct=PT+Ac+Cos$ |
| Precio total de la madera | Es la suma del costo de cada madera que fue seleccionada por el usuario. | $Ptm=sumatoria\ del\ Ct$ |
| Calculo del costo por km. | El valor de los km. Introducidos por el usuario multiplicado por 21 que es el precio unitario por km. | $Km=valor *21$ |
| Calculo de casetas | Es el valor introducido por el usuario en kilómetros y multiplicado por 4 que es el precio unitario por km. | $Cas=valor *4$ |
| Descarga de material | Costo del pie tablón neto de cada producto por el 0.32 que es el precio unitario de cada tablón. | $Dcr=Pt*0.32$ |
| Precio del transporte | Comprende la suma de casetas, descarga de material y costo por km, indicados anteriormente. | $Trans=cas +Dcr$ |

Tabla 3.7 Reglas del negocio (continuación)

| Operación | Descripción | Nombre corto |
|---|--|-------------------------|
| Número de vigas | Suma de los números de unidades de los productos (viga laminada, viga curva, viga i, tablero) que el usuario selecciono. | $Vg=vl+vi+ta$ |
| IVA | Calculo del 16% sobre el valor total. | Iva=16% |
| Total de vigas procesadas | Se obtiene la suma de unidades de las vigas de madera utilizadas para cada producto que el usuario selecciono. | TI=suma de las Vg |
| Costo con acabado | Suma del valor del acabado, seleccionado por el usuario más el costo del pie tablón establecido en el sistema. | $CosA = TI + acabado$ |
| Costo con tratamiento | Se aumenta el costo del tratamiento seleccionado por el usuario al precio del pie tablón. | $CosT=TI + tratamiento$ |
| Costo sin tratamiento | Costo del pie tablón sin aumento de costo. | TI |
| Costo de madera lijada | Se aumenta un costo a cada pie tablón. | $cosM=TI+Pt$ |
| Costo de madera no lijada | Costo del pie tablón sin aumento de costo. | Pt |
| Costo con transporte de tratamiento | Suma de un valor de transporte introducido por el usuario al precio del tratamiento. | $Trans= Pt+ valor x$ |
| Costo sin transporte de tratamiento | No se realiza ningún aumento de transporte por el tratamiento. | Valor |
| Calculo del porcentaje del grado de dificultad | Aumento de un porcentaje al precio del pie tablón. | $Pt + 30\%$ |
| Precio de venta por pieza de la viga laminada | Multiplicación del ancho por el peralte y largo entre 30.48 por número de unidades. | PT + los demás costos |
| Precio de venta total de la viga laminada | Suma de las vigas laminadas para calcular el precio total de las vigas. | Suma de los PT |
| Precio de venta por pieza de la viga laminada curva | Multiplicación del ancho, peralte y largo introducidos por el usuario entre 12 por el número de unidades del producto. | PT + los demás costos |

Tabla 3.8 Reglas del negocio (continuación)

| Operación | Descripción | Nombre corto |
|---|--|-----------------------|
| Precio de venta total de la viga laminada curva | Suma del total de las vigas curvas | Suma de los PT |
| Precio de venta por pieza de la viga I | Multiplicación del ancho por el peralte y largo entre 30.84 por número de unidades | Pt + los demás costos |
| Precio de venta total de la viga I | Suma de los precios de cada viga para obtener el total | Suma de los PT |
| Transporte | Realiza el aumento de un valor al costo total | Trans + Pt |
| Levantamiento y planos | Aumentar 2% al precio total de la madera | Pt +2% |
| Calculo estructural | Aumentar 6% a precio I total de la madera | Pt +6% |
| Habilitaciones | Aumentar 8% precio al total de la madera | Pt +8% |
| Instalación primer nivel | Aumentar 30% precio al total de la madera | Pt +30% |
| Instalación segundo nivel | Mas el 5% precio sobre el primer nivel | Pt +35% |
| Instalación tercer nivel | Mas el 5% precio sobre el primer nivel | Pt +40% |

3.2.5 Gráfica de producto

El seguimiento del proyecto se fue midiendo a través de la gráfica de producto, en la figura 3.3 se muestra la gráfica utilizada para el seguimiento total de los meses, en donde se presentan en el eje de la y las horas estimadas del proyecto y en el eje de la x el número de sprint.

Las horas estimadas restantes, son las que se estimaron para la conclusión del proyecto, y se van restando conforma se va liberando las tareas de cada sprint.

Las horas estimadas son las que se van trabajando, y se van liberando realmente, y en la que se puede ver realmente el tiempo que falta para conclusión del proyecto.

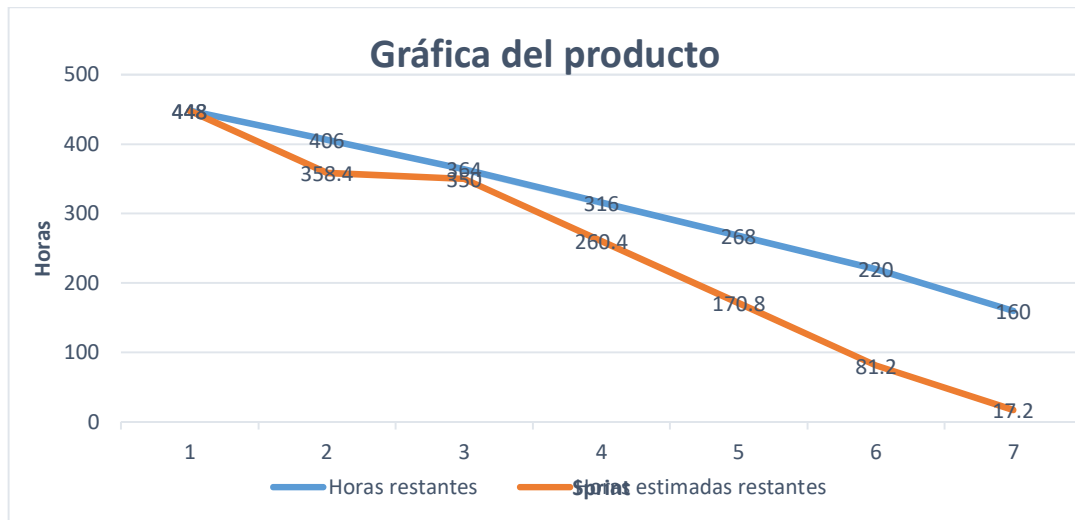


Figura 3.3 Gráfica del producto

3.3 Metodología implementada al controlador

En base a la arquitectura de diseño MVC, en este trabajo nos enfocamos especialmente en la capa del controlador, la cual se encarga de gestionar los eventos de entrada y salida, así como la coordinación de los procesos proveniente de la capa de diseño y del modelo.

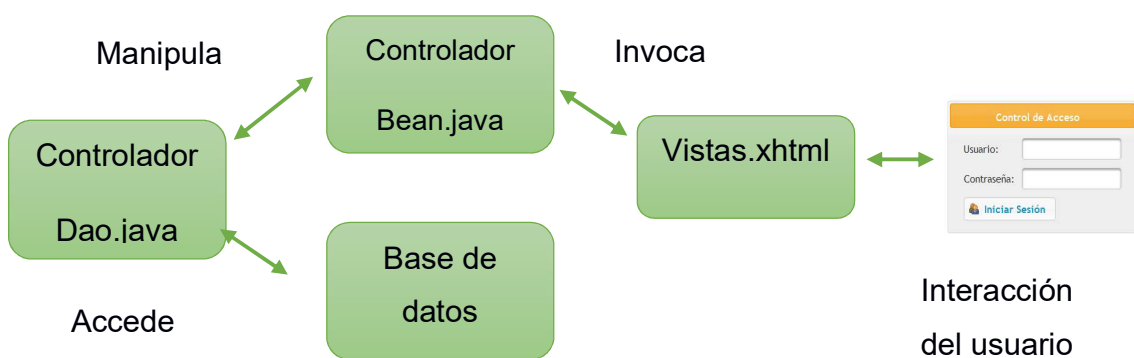


Figura 3.4 Modelo vista controlador del cotizador

En la figura 3.4 se muestra la implementación del controlador con Modelo-Vista-Controlador, debido a que el controlador tiene comunicación con la vista y modelo se tiene archivos java que realizan la comunicación con la base de datos y con la vista, haciendo que el controlador gestione y administre las operaciones, por ejemplo:

Para acceder al sistema el usuario introduce su datos y realizar la validación, los cuales son ingresados por una interfaz que es un archivo con extensión .xhtml, el cual se comunica con el controlador para que procese la información y se encargue de validar el usuario en la base de datos, al realizar la comunicación con la base de datos el controlador administra que proceso sigue o que interfaz debería mostrar, si el acceso es correcto puede acceder al sistema y si no mostrar un mensaje de error.

3.4 Metodología de desarrollo Scrum

Se elige una metodología Scrum para el diseño del controlador, porque propone un trabajo en paralelo y hace que todos los miembros permanezcan en comunicación constante. Con Scrum se obtiene lo mejor de los miembros del equipo, además de que los roles son intercambiables y todos tiene el mismo rango, las mejoras son constantes y es adaptable a cambios imprevistos.

A partir de la metodología general basada en Scrum se desglosa la metodología para el desarrollo del controlador mostrada en la figura 3.5.

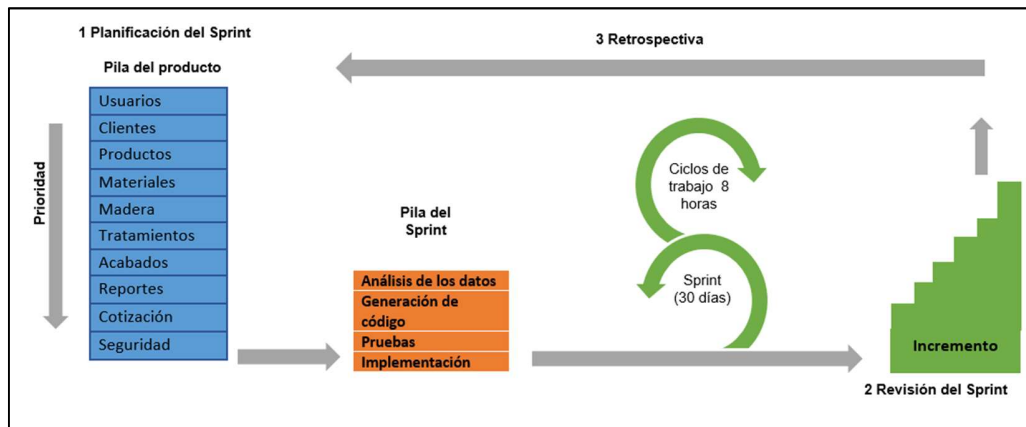


Figura 3.5 Metodología de desarrollo centrado en el controlador

Planificación del sprint

En la planificación del *Sprint* se contemplan 2 componentes, la pila del producto y la pila del *sprint*

En la pila del producto se abordan los distintos módulos que estará compuesto el sistema cotizador, se deberá programar y entregar cada módulo para dar paso al siguiente.

En la pila del *Sprint* se contempla el proceso para el desarrollo de cada módulo, en la cual se tiene el análisis de los datos para tener conocimiento de cómo debe ser el programado el módulo, posteriormente se realiza la codificación el componente y al mismo tiempo realizar pruebas del funcionamiento del mismo, una vez concluido este proceso se realiza la implementación con todo el sistema.

Todo esto en un tiempo de 8 horas de trabajo y entregas de 30 días para cada módulo, cabe mencionar que en componentes se puede requerir más tiempo por el grado de dificultad.

El Incremento

El incremento es un módulo producido en un *sprint*, y debe estar completamente terminada y operativa, en condiciones de ser entregada al cliente para su evaluación, si se cumple con lo establecido y requerido por el cliente se da paso al siguiente componente de la bitácora.

Revisión del *sprint*

Al finalizar un *sprint* y se realiza una presentación con el equipo de trabajo para validar que cumpla con lo solicitado por el cliente, una vez esté listo se procede a la presentación al cliente para así poder realizar un incremento o no, todo esto dependiendo a lo que el cliente solicite.

Si se presenta alguna falla los miembros del equipo analizar sobre las complicaciones que surgieron y se proponen nuevas formas para la realización de las tareas.

3.5 Comentarios

En este Capítulo se presentó la metodología implementada para el desarrollo del componente controlador del sistema cotizador.

Al emplear una tecnología ágil, en conjunto con el patrón de diseño Modelo-Vista-Controlador, se obtiene los tres módulos para el desarrollo de cada uno, por lo que permite un desarrollo guiado del parte controlador, con la finalidad de desarrollar el controlador en el menor tiempo posible.

Con la metodología ágil Scrum se involucra siempre al cliente para las especificaciones de requisitos, además de obtener las operaciones esenciales para el funcionamiento del controlador.

Se realizó una descripción de cada componente del patrón de diseño MVC, se especificó el funcionamiento del modelo, así como el de la vista y el controlador enfocado ya al sistema cotizador, además de la interacción de los tres componentes entre sí.

Se presentó el diagrama Scrum del sistema en general, donde se puede ver claramente el proceso de la pila del Sprint y los componentes que la conforman, así también los de la pila del producto y el seguimiento del *sprint*. Se muestra claramente el proceso general que se sigue para el desarrollo del sistema.

Enseguida se presenta los componentes, en donde tenemos la planeación que incluye los requerimientos funcionales y no funcionales.

Se presentó una metodología para el controlador, donde se indica el proceso que realiza el controlador del sistema cotizador, todo esto guiado por la metodóloga Scrum aplicada al controlador, donde se tiene una planificación, prioridades, revisiones del sprint y retrospectiva. Además de presentaron una descripción de los componentes de la metodología Scrum que se está utilizando para el desarrollo del sistema cotizador.

4 Desarrollo del controlador

En este Capítulo se presenta como se aplicó la metodología Scrum presentada anteriormente para el desarrollo del componente controlador, además de la integración de los componentes modelo y la vista del sistema para tener como resultado la aplicación cotizadora.

4.1 Desarrollo del Sprint

En este apartado se describirá como fue la elaboración del controlador en cada uno de los *Sprint* que se realizaron para el desarrollo del controlador. También se presentan los diagramas para cada iteración.

4.1.1 *Sprint* 1 Recolección de la información

En el primer *Sprint* se generó la recopilación y obtención de los requisitos funcionales, no funcionales y los actores del sistema, así como las entrevistas necesarias con el cliente para tener una mejor comprensión de los requisitos, de este *sprint* se genera la pila del producto como se puede apreciar en la tabla 4.1.

Tabla 4.1 *Sprint* Obtención de los requisitos

| Obtención de los requisitos | |
|-----------------------------|---|
| Actores | <ul style="list-style-type: none">• Administrador• Jefe de área• Operador• Usuario |

Tabla 4.2 Sprint Obtención de los requisitos (continuación)

| | |
|-------------------------------|---|
| Requerimientos funcionales | <ul style="list-style-type: none"> • Administración de usuarios (alta modificación, eliminación y consultas) • Administración de clientes (alta, modificación, eliminación y consulta) • Administración de materiales (alta, modificación, eliminación y consulta) • Administración de servicio (alta, modificación, eliminación y consulta) • Administración de instalación (alta, modificación eliminación y consulta) • Cotizar (alta, modificación y consulta) |
| Requerimientos no funcionales | <ul style="list-style-type: none"> • El sistema permite el uso de usuarios al mismo tiempo. • El sistema deberá evitar el ingreso de información incorrecta por medio de mensajes que el sistema genere. • El sistema deberá ser utilizado toda la jornada de trabajo. • El sistema se desarrollará en su totalidad en Java Enterprise Edition y MySQL • El sistema deberá ser creado pensando en la implementación de módulos futuros. • La empresa proporcionará el servidor donde correrá el sistema • El sistema utiliza usuario y contraseña para el acceso |

En la figura 4.1 se presenta el diagrama general del sistema, elaborado a partir de los requerimientos funciones y los actores obtenidos en el primer *Sprint*.

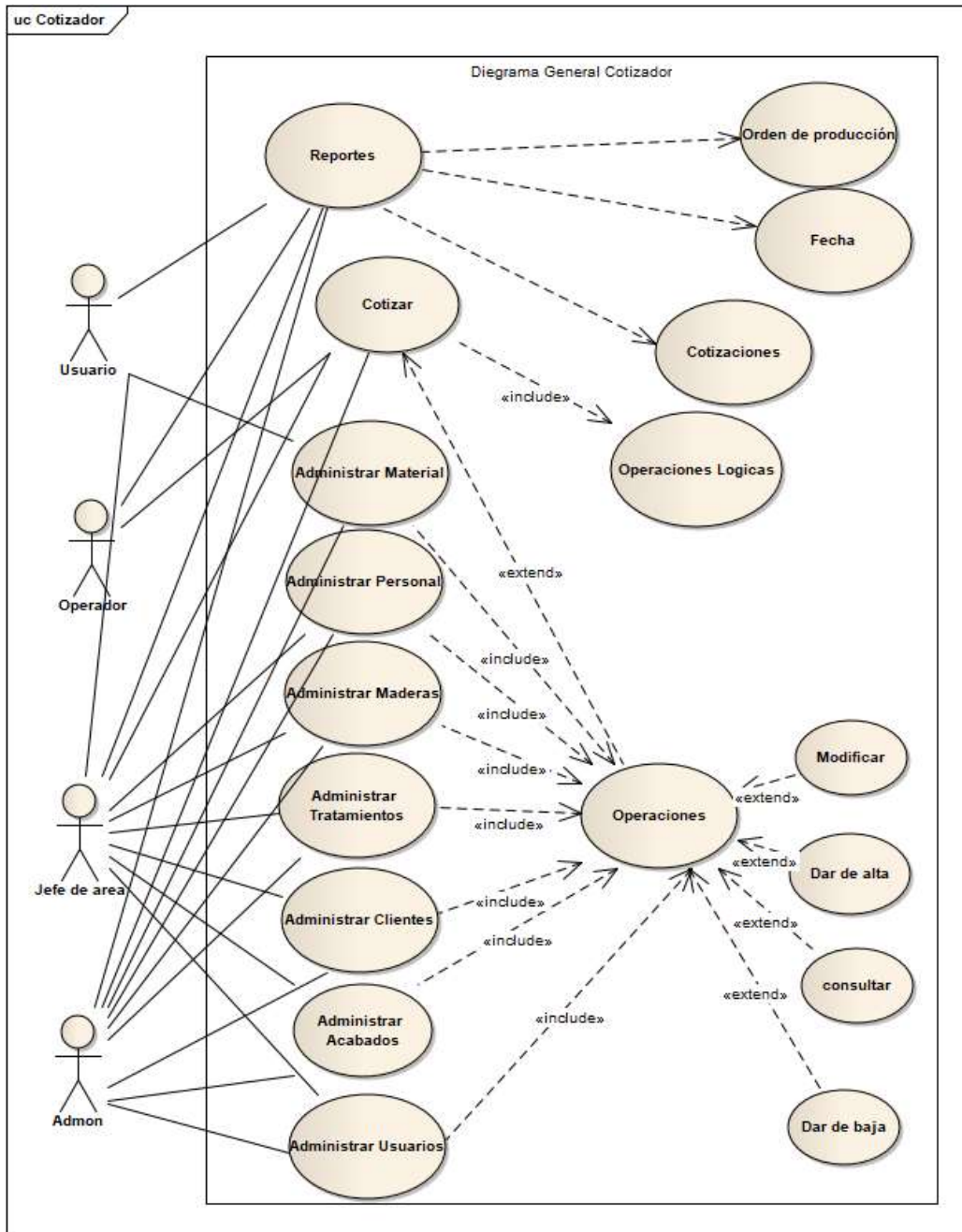


Figura 4.1 Caso de uso del cotizador

4.1.2 *Sprint 2* Administración y validación de usuarios

En el segundo sprint tabla 4.2, se realiza el proceso de acceso al sistema por los usuarios, ya que se tiene privilegios a los distintos módulos del sistema. Además de no permitir la navegación en los módulos sin estar autenticados, forzando el acceso solo por el uso de usuario y contraseña.

Tabla 4.3 *Sprint validación de usuarios*

| Validación de usuarios | |
|-----------------------------------|--|
| Duración: | 26 días hábiles |
| Objetivo: | Es realizar el funcionamiento para validar el acceso al sistema de los diferentes tipos de usuario. |
| Descripción: | En este sprint se programaron las clases usuarioBean, UsuarioDAO y PlantillaController para poder controlar el acceso a los distintos usuarios, además de la seguridad y acceso a los diferentes módulos del sistema. |
| Desarrollo: | Se contemplan 180 líneas de código |
| Administración de Usuarios | |
| Objetivo: | Es realizar el funcionamiento para agregar, modificar, y eliminar un usuario. |
| Descripción: | En este <i>sprint</i> se programaron las clases UsuariosBean y UsuariosDAO para poder crear, modificar y eliminar usuarios. Este módulo contempla almacenar clientes en la base de datos para alimentar de información al cotizador, se han utilizado diferentes clases para las distintas operaciones que realiza dado los privilegios que tiene cada usuario |
| Desarrollo: | Se contemplan 120 líneas de código |

En la tabla 4.3 se describe el *Sprint* para la administración de usuarios, y como se realizó esta acción en el controlador.

Tabla 4.4 Descripción de validación de usuarios

| Interfaz | Acciones del controlador |
|--|--|
| Pantalla principal para acceso al sistema. | Una vez que el usuario haya colocado su usuario y contraseña se mandan a usuarioBean. UsuarioBean procesa la información y la manda a usuarioDAO. usuarioDAO realiza la consulta con la base de datos y retorna la información a usuarioBean para su validación, si es correcto permite el acceso y de lo contrario manda un mensaje de error y pide el usuario nuevamente. |
| Alta de usuarios. | El usuario inicia sesión para poder realizar el alta de un usuario. Los datos obtenidos de la interfaz, que son los que el usuario proporcione se mandan a usuariosBean. En usuariosBeans se procesa la información y se mandan a usuariosDAO. Por medio de usuariosDAO se realiza la inserción a la tabla en la base de datos. |
| Consulta los usuarios almacenados en la base de datos. | El usuario inicia sesión para poder realizar el listado de los usuarios. En usuariosBean procesa la información y se mandan a usuariosDAO. Por medio de usuariosDAO se realiza la consulta a la base de datos y almacena la información. La información obtenida de la base de datos se manda a usuariosBean. usuariosBean manda la información para ser presentada en la pantalla y sea visualizada por el usuario. |
| Modificación de un usuario. | Al tener la lista de los usuarios se acciona modificar, en el cual usuariosBean manda información del usuario a la vista para que el usuario modifique los campos. Después de modificar se mandan los datos a usuariosBean y este realiza la comunicación con usuariosDAO. usuariosDAO procesa la información y actualiza la información en la tabla. usuariosDAO obtiene la información actualidad y la manda a usuariosBean para que la presenta al usuario en la interfaz. |

En la figura 4.1 se muestra el caso de uso para la administración de un usuario.

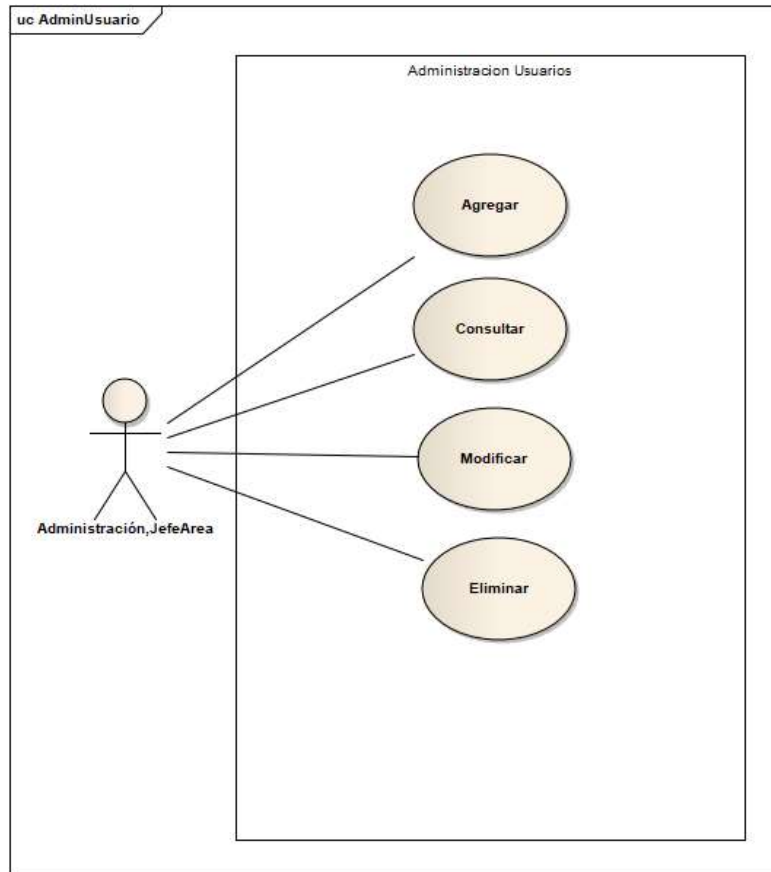


Figura 4. 1 Caso de uso administración usuarios

4.1.3 Sprint 3 Administración de clientes y productos

En la tabla 4.4 se describe el *Sprint* para la administración de clientes, y como se realizó esta acción en el controlador.

Tabla 4.5 Sprint Administración de clientes

| Administración de clientes | |
|----------------------------|---|
| Duración: | 24 días hábiles |
| Objetivo: | Es realizar el funcionamiento para agregar, modificar, y eliminar un cliente. |

Tabla 4.6 Sprint Administración de clientes (continuación)

| | |
|--------------|---|
| Descripción: | En este sprint se programaron las clases ClientesBean y ClientesDAO para poder crear, modificar y eliminar clientes. Este módulo contempla almacenar clientes en la base de datos para alimentar de información al cotizador, se han utilizado diferentes clases para las distintas operaciones que realiza dado los privilegios que tiene cada usuario. Teniendo Administrador y jefe de área el poder crear, modificar y eliminar. El operador solo crear y modificar, pero no eliminar |
| Desarrollo: | Se contemplan 120 líneas de código |

En la tabla 4.5 se puede observar el controlador para esta parte de los clientes.

Tabla 4.7 Descripción del controlador para la administración de clientes

| Interfaz | Acciones del controlador |
|--|---|
| Alta de clientes. Muestra todos los que se deben llenar para agregar un cliente. | <p>El inicia sesión para poder realizar el alta de un cliente.</p> <p>Los datos obtenidos de la interfaz, que son los que el usuario proporcione se mandan a ClientesBean.</p> <p>En ClientesBean se procesa la información y se mandan a ClientesDAO.</p> <p>Por medio de ClientesDAO se realiza la inserción a la tabla en la base de datos.</p> |
| Consulta los clientes almacenados en la base de datos. | <p>El usuario inicia sesión para poder realizar el listado de los clientes.</p> <p>En clientesBean procesa la información y se mandan a clientesDAO.</p> <p>Por medio de clientesDAO se realiza la consulta a la base de datos y almacena la información.</p> <p>La información obtenida de la base de datos se manda a clientesBean.</p> <p>clientesBean manda la información para ser presentada en la pantalla y sea visualizada por el usuario.</p> |

Tabla 4.8 Descripción del controlador para la administración de clientes(continuación)

| Interfaz | Acciones del controlador |
|-----------------------------|---|
| Modificación de un cliente. | <p>Al tener la lista de los clientes se acciona modificar, en el cual clientesBean manda información del cliente a la vista para que el usuario modifique los campos.</p> <p>Después de modificar se mandan los datos a clientesBean y este realiza la comunicación con clientesDAO.</p> <p>clientesDAO procesa la información y actualiza la información en la tabla.</p> <p>clientesDAO obtiene la información actualidad y la manda a cleintesBean para que la presenta al usuario en la interfaz.</p> |

En la figura 4.2 se muestra el caso de uso para el alta de un cliente.

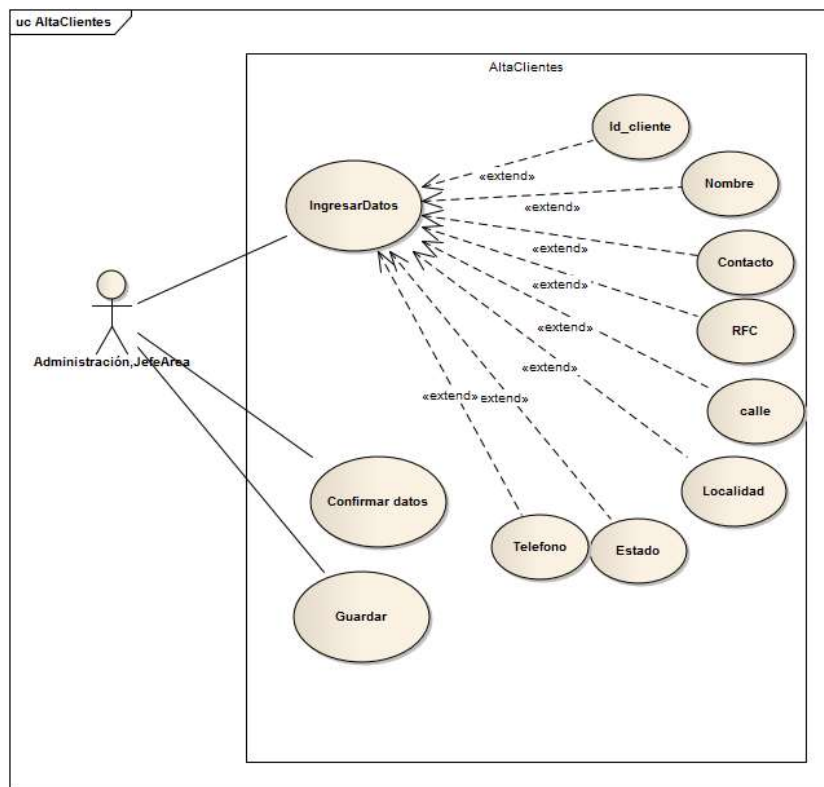


Figura 4.2 Diagrama de caso de uso de alta de clientes

En la tabla 4.6 se describe el *Sprint* para la administración de productos, y como se realizó esta acción en el controlador.

Tabla 4.9 Sprint Administración de productos

| Administración de Productos | |
|------------------------------------|--|
| Duración: | 14 días hábiles |
| Objetivo: | Es realizar el funcionamiento para agregar, modificar, y eliminar un producto. |
| Descripción: | En este sprint se programaron las clases ProductosBean y ProductosDAO para poder crear, modificar y eliminar productos. Este módulo contempla almacenar los productos en la base de datos para alimentar de información al cotizador, se han utilizado diferentes clases para las distintas operaciones que realiza dando los privilegios que tiene cada usuario. Teniendo solo el Administrador y jefe de área el poder crear, modificar y eliminar. El operador solo crear y modificar, pero no eliminar |
| Desarrollo: | Se contemplan 120 líneas de código |

En la tabla 4.7 se puede observar el controlador para esta parte de los productos.

Tabla 4.10 Descripción del controlador para la administración de Productos

| Interfaz | Acciones del controlador |
|---|--|
| Alta de producto. Muestra todos los que se deben llenar para agregar un producto. | El usuario inicia sesión para poder realizar el alta de un producto. Los datos obtenidos de la interfaz, que son los que el usuario proporcione se mandan a ProductosBean. En ProductosBean se procesa la información y se mandan a ProductosDAO. Por medio de ProductosDAO se realiza la inserción a la tabla en la base de datos. |
| Consulta los productos almacenados en la base de datos. | El usuario inicia sesión para poder realizar el listado de los productos. En ProductosBean procesa la información y se mandan a ProductosDAO. Por medio de ProductosDAO se realiza la consulta a la base de datos y almacena la información. La información obtenida de la base de datos se manda a ProductosBean. ProductosBean manda la información para ser presentada en la pantalla y sea visualizada por el usuario. |

Tabla 4.11 Descripción del controlador para la administración de Productos (continuación)

| Interfaz | Acciones del controlador |
|------------------------------|--|
| Modificación de un producto. | <p>Al tener la lista de los productos se acciona modificar, en el cual ProductosBean manda información del producto a la vista para que el usuario modifique los campos.</p> <p>Después de modificar se mandan los datos a ProductosBean y este realiza la comunicación con ProductosDAO.</p> <p>ProductosDAO procesa la información y actualiza la información en la tabla.</p> <p>ProductosDAO obtiene la información actualidad y la manda a ProductosBeans para que la presenta al usuario en la interfaz.</p> |

En la figura 4.3 se muestra el caso de uso para el alta de un producto.

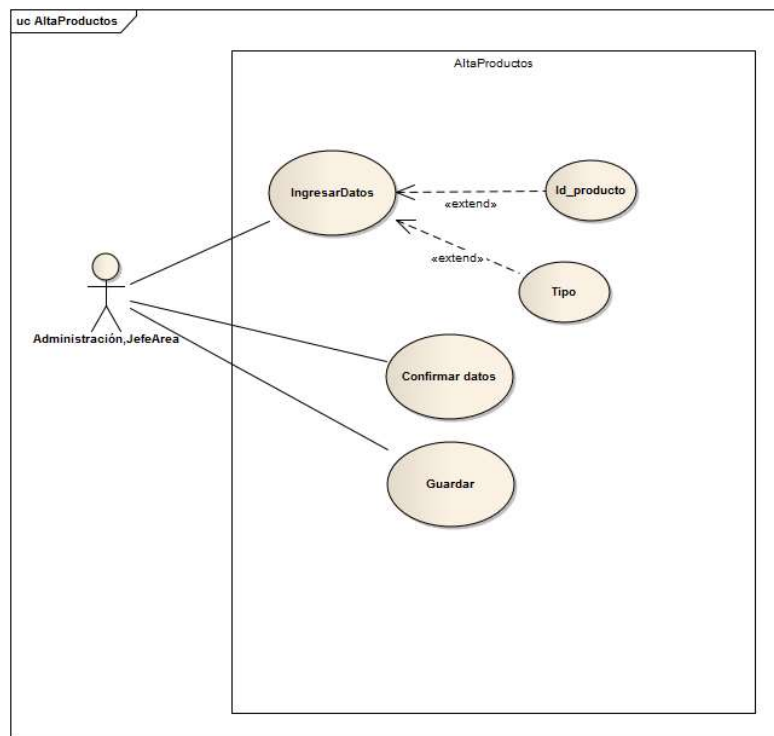


Figura 4.3 Diagrama de caso de uso de alta de productos

4.1.4 *Sprint* 4 Administración de madera y materiales

En la tabla 4.8 se describe el *Sprint* para la administración de madera, y como se realizó esta acción en el controlador.

Tabla 4.12 Administración de madera

| Administración de Madera | |
|---------------------------------|---|
| Duración: | 14 días hábiles |
| Objetivo: | Es realizar el funcionamiento para agregar, modificar, y eliminar una madera. |
| Descripción: | En este <i>sprint</i> se programaron las clases MaderaBean y MaderaDAO para poder crear, modificar y eliminar la madera. Este módulo contempla almacenar la madera en la base de datos para alimentar de información al cotizador, se han utilizado diferentes clases para las distintas operaciones que realiza dando los privilegios que tiene cada usuario. Teniendo solo el Administrador y jefe de área el poder crear, modificar y eliminar. El operador solo crear y modificar, pero no eliminar |
| Desarrollo: | Se contemplan 120 líneas de código |

En la tabla 4.9 se puede observar el controlador para esta parte de la madera.

Tabla 4.13 Descripción del controlador para la administración de madera

| Interfaz | Acciones del controlador |
|--|---|
| Alta de madera. Muestra todos los que se deben llenar para agregar una madera. | El usuario inicia sesión para poder realizar el alta de una madera. Los datos obtenidos de la interfaz, que son los que el usuario proporciono se mandan a MaderaBean. En MaderaBean se procesa la información y se mandan a MaderaDAO. Por medio de MaderaDAO se realiza la inserción a la tabla en la base de datos. |

Tabla 4.14 Descripción del controlador para la administración de madera (continuación)

| Interfaz | Acciones del controlador |
|--|---|
| Consulta la madera almacenada en la base de datos. | <p>El usuario inicia sesión para poder realizar el listado de la madera.</p> <p>En MaderaBean procesa la información y se mandan a MaderaDAO. Por medio de MaderaDAO se realiza la consulta a la base de datos y almacena la información.</p> <p>La información obtenida de la base de datos se manda a MaderaBean.</p> <p>MaderaBean manda la información para ser presentada en la pantalla y sea visualizada por el usuario.</p> |
| Modificación de una madera. | <p>Al tener la lista de la madera se acciona modificar, en el cual MaderaBean manda información de la madera a la vista para que el usuario modifique los campos.</p> <p>Después de modificar se mandan los datos a MaderaBean y este realiza la comunicación con MaderaDAO.</p> <p>MaderaDAO procesa la información y actualiza la información en la tabla.</p> <p>MaderaDAO obtiene la información actualidad y la manda a MaderaBean para que la presenta al usuario en la interfaz.</p> |

En la figura 4.3 se muestra el caso de uso para el alta de una madera.

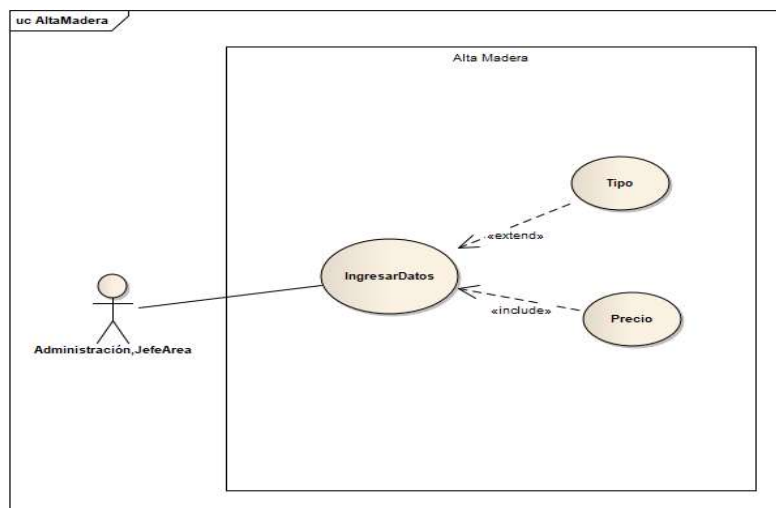


Figura 4.4 Diagrama de caso de uso de alta de madera

En la tabla 4.10 se describe el *Sprint* para la administración de los materiales, y como se realizó esta acción en el controlador.

Tabla 4.15 *Sprint Administración de materiales*

| Administración de Materiales | |
|-------------------------------------|--|
| Duración: | 14 días hábiles |
| Objetivo: | Es realizar el funcionamiento para agregar, modificar, y eliminar materiales. |
| Descripción: | En este <i>sprint</i> se programaron las clases MaterialesBean y MaterialesDAO para poder crear, modificar y eliminar materiales. Este módulo contempla almacenar los productos en la base de datos para alimentar de información al cotizador, se han utilizado diferentes clases para las distintas operaciones que realiza dando los privilegios que tiene cada usuario. Teniendo solo el Administrador y jefe de área el poder crear, modificar y eliminar. El operador solo crear y modificar, pero no eliminar |
| Desarrollo: | Se contemplan 120 líneas de código |

En la tabla 4.11 se puede observar el controlador para esta parte de los materiales.

Tabla 4.16 *Descripción del controlador para la administración de materiales*

| Interfaz | Acciones del controlador |
|--|--|
| Alta de materiales. Muestra todos los que se deben llenar para agregar un material. | El usuario inicia sesión para poder realizar el alta de un material. Los datos obtenidos de la interfaz, que son los que el usuario proporciona se mandan a MaterialesBean. En MaterialesBean se procesa la información y se mandan a clientesDAO. Por medio de MaterialesDAO se realiza la inserción a la tabla en la base de datos. |

Tabla 4.17 *Descripción del controlador para la administración de materiales (continuación)*

| Interfaz | Acciones del controlador |
|--|---|
| Consulta los materiales almacenados en la base de datos. | El usuario inicia sesión para poder realizar el listado de los materiales. En MaterialesBean procesa la información y se mandan a MaterialesDAO. |

| | |
|-------------------------------------|---|
| | <p>Por medio de MaterialesDAO se realiza la consulta a la base de datos y almacena la información.</p> <p>La información obtenida de la base de datos se manda a MaterialesBean.</p> <p>MaterialesBean manda la información para ser presentada en la pantalla y sea visualizada por el usuario.</p> |
| <p>Modificación de un material.</p> | <p>Al tener la lista de los materiales se acciona modificar, en el cual MaterialesBean manda información del usuario a la vista para que el usuario modifique los campos.</p> <p>Después de modificar se mandan los datos a MaterialesBean y este realiza la comunicación con MaterialesDAO.</p> <p>MaterialesDAO procesa la información y actualiza la información en la tabla.</p> <p>MaterialesDAO obtiene la información actualidad y la manda a MaterialesBean para que la presenta al usuario en la interfaz.</p> |

En la figura 4.5 se muestra el caso de uso para el alta de un material.

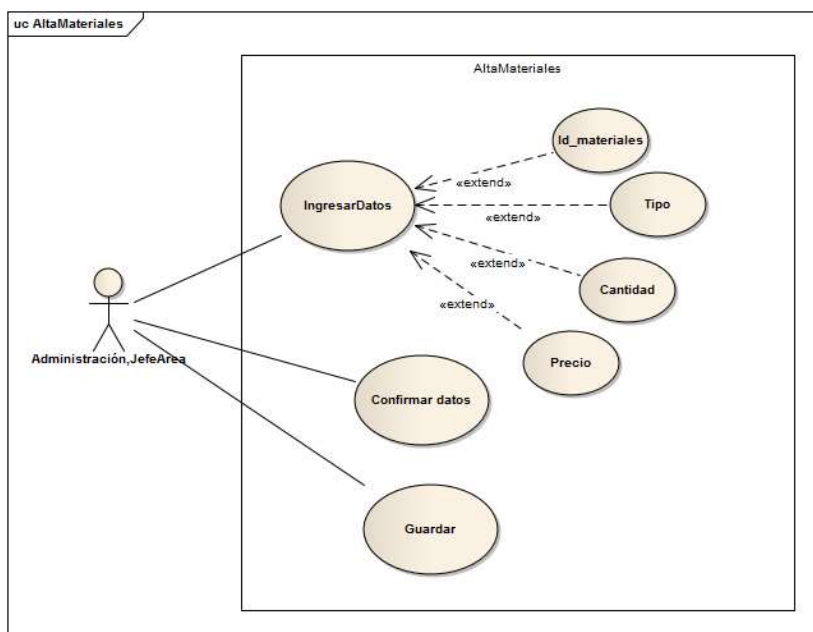


Figura 4.5 Caso de uso para alta de materiales

4.1.5 *Sprint* 5 Administrar tratamiento y acabado

En la tabla 4.12 se describe el *Sprint* para la administración de tratamientos, y como se realizó esta acción en el controlador.

Tabla 4.18 Administración de tratamientos

| Administración de Tratamientos | |
|---------------------------------------|---|
| Duración: | 14 días hábiles |
| Objetivo: | Es realizar el funcionamiento para agregar, modificar, y eliminar un tratamiento. |
| Descripción: | En este <i>sprint</i> se programaron las clases TratamientoBean y TratamientoDAO para poder crear, modificar y eliminar el tratamiento. Este módulo contempla almacenar los tratamientos en la base de datos para alimentar de información al cotizador, se han utilizado diferentes clases para las distintas operaciones que realiza dando los privilegios que tiene cada usuario. Teniendo solo el Administrador y jefe de área el poder crear, modificar y eliminar. El operador solo crear y modificar, pero no eliminar |
| Desarrollo: | Se contemplan 120 líneas de código |

En la tabla 4.13 se puede observar el controlador para esta parte de los tratamientos.

Tabla 4.19 Descripción del controlador para la administración de tratamiento

| Interfaz | Acciones del controlador |
|--|---|
| Alta de tratamiento. Muestra todos los que se deben llenar para agregar un tratamiento. | El inicia sesión para poder realizar el alta de una madera. Los datos obtenidos de la interfaz, que son los que el usuario proporciono se mandan a TratamientoBean. En TratamientoBean se procesa la información y se mandan a TratamientoDAO. Por medio de TratamientoDAO se realiza la inserción a la tabla en la base de datos. |

Tabla 4.20 Descripción del controlador para la administración de tratamiento (continuación)

| Interfaz | Acciones del controlador |
|--|--|
| <p>Consulta de los tratamientos almacenados en la base de datos.</p> | <p>El usuario inicia sesión para poder realizar el listado de los tratamientos.</p> <p>En TratamientoBean procesa la información y se mandan a TratamientoDAO.</p> <p>Por medio de TratamientoDAO se realiza la consulta a la base de datos y almacena la información.</p> <p>La información obtenida de la base de datos se manda a TratamientoBean.</p> <p>TratamientoBean manda la información para ser presentada en la pantalla y sea visualizada por el usuario.</p> |
| <p>Modificación de un tratamiento.</p> | <p>Al tener la lista de los tratamientos se acciona modificar, en el cual TratamientoBean manda información del usuario a la vista para que el usuario modifique los campos.</p> <p>Después de modificar se mandan los datos a TratamientoBean y este realiza la comunicación con TratamientoDAO.</p> <p>TratamientoDAO procesa la información y actualiza la información en la tabla.</p> <p>TratamientoDAO obtiene la información actualidad y la manda a MaderaBean para que la presenta al usuario en la interfaz.</p> |

En la figura 4.6 se muestra el caso de uso para el alta de un tratamiento.

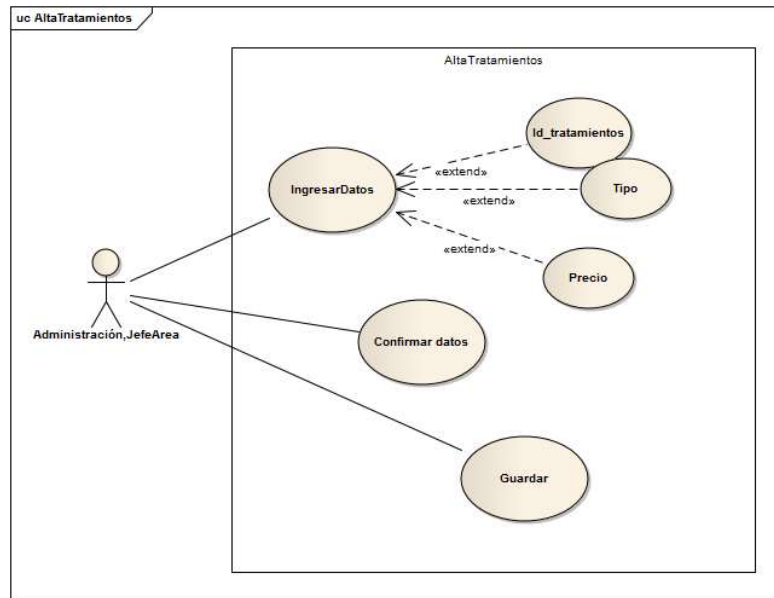


Figura 4.6 Diagrama de caso de uso de alta de un tratamiento

En la tabla 4.14 se describe el *Sprint* para la administración de los acabados, y como se realizó esta acción en el controlador.

Tabla 4.21 *Sprint* Administración de acabados

| Administración de Acabados | |
|----------------------------|---|
| Duración: | 14 días hábiles |
| Objetivo: | Es realizar el funcionamiento para agregar, modificar, y eliminar acabados. |
| Descripción: | En este <i>sprint</i> se programaron las clases AcabadosBean y AcabadosDAO para poder crear, modificar y eliminar acabados. Este módulo contempla almacenar los acabados en la base de datos para alimentar de información al cotizador, se han utilizado diferentes clases para las distintas operaciones que realiza dando los privilegios que tiene cada usuario. Teniendo solo el Administrador y jefe de área el poder crear, modificar y eliminar. El operador solo crear y modificar, pero no eliminar |
| Desarrollo: | Se contemplan 120 líneas de código |

En la tabla 4.15 se puede observar el controlador para esta parte de los acabados.

Tabla 4.22 Descripción del controlador para la administración de acabados

| Interfaz | Acciones del controlador |
|--|---|
| Alta de acabados. Muestra todos los que se deben llenar para agregar un acabado. | <p>El usuario inicia sesión para poder realizar el alta de un acabado.</p> <p>Los datos obtenidos de la interfaz, que son los que el usuario proporciono se mandan a AcabadosBean.</p> <p>En AcabadosBean se procesa la información y se mandan a AcabadosDAO.</p> <p>Por medio de AcabadosDAO se realiza la inserción a la tabla en la base de datos.</p> |
| Consulta los acabados almacenados en la base de datos. | <p>El inicia sesión para poder realizar el listado de los acabados.</p> <p>En AcabadosBean procesa la información y se mandan a AcabadosDAO.</p> <p>Por medio de AcabadosDAO se realiza la consulta a la base de datos y almacena la información.</p> <p>La información obtenida de la base de datos se manda a AcabadosBean.</p> <p>AcabadosBean manda la información para ser presentada en la pantalla y sea visualizada por el usuario.</p> |
| Modificación de un acabado. | <p>Al tener la lista de los acabados se acciona modificar, en el cual AcabadosBean manda información del usuario a la vista para que el usuario modifique los campos.</p> <p>Después de modificar se mandan los datos a AcabadosBean y este realiza la comunicación con AcabadosDAO.</p> <p>AcabadosDAO procesa la información y actualiza la información en la tabla.</p> <p>AcabadosDAO obtiene la información actualidad y la manda a AcabadosBean para que la presenta al usuario en la interfaz.</p> |

En la figura 4.7 se muestra el caso de uso para el alta de un acabado.

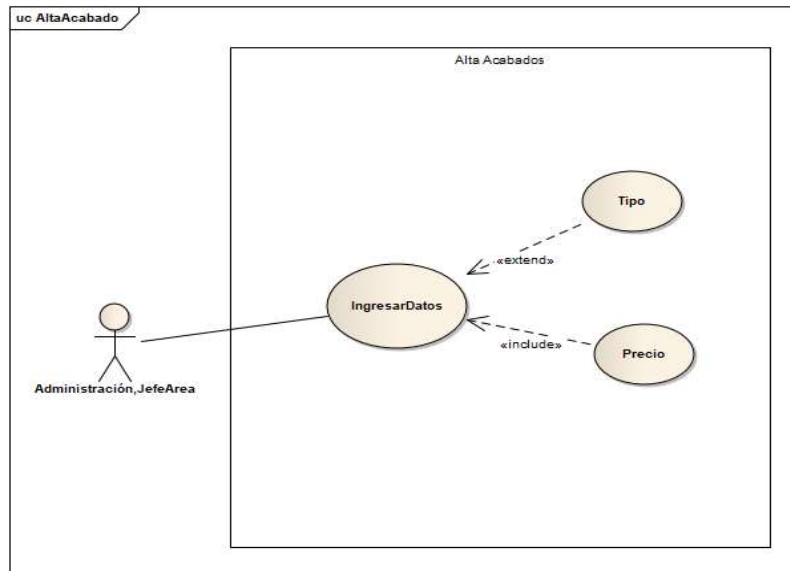


Figura 4.7 Caso de uso para alta de acabados

4.1.6 Sprint 6 Reporte de cotizaciones

En la tabla 4.16 se describe el *Sprint* para la generación de reporte, y como se realiza esta acción en el controlador.

Tabla 4.23 *Sprint* Generar reportes de cotización

| Generación de reportes | |
|------------------------|--|
| Duración: | 10 días hábiles |
| Objetivo: | Es realizar el funcionamiento para generar reporte de cotizaciones |
| Descripción: | En este <i>sprint</i> se programaron las clases <code>DetalleCotizaBean</code> y <code>CotizacionDAO</code> para poder generar reportes de las cotizaciones. Este módulo contempla generar de la base de datos los reportes de las cotizaciones. Administrador, jefe de área, usuario y el operador pueden generar los reportes. |
| Desarrollo: | Se contemplan 150 líneas de código |

En la tabla 4.17 se puede observar el funcionamiento del controlador para la parte de la generación de reportes de las cotizaciones generadas.

Tabla 4.24 Descripción del controlador para los reportes de cotización

| Interfaz | Acciones del controlador |
|-------------------------------------|--|
| Generar reporte de las cotizaciones | <p>Se inicia sesión para poder realizar el reporte de las cotizaciones.</p> <p>Se seleccionan los campos para poder generar el reporte.</p> <p>Los datos obtenidos de la interfaz, que son los que el usuario proporciona se mandan a DetallecotizaBean.</p> <p>En DetallecotizaBean se procesa la información y se mandan a CotizacionDAO.</p> <p>Por medio de CotizacionDAO se realiza la consulta a la tabla en la base de datos.</p> <p>Se envían los datos de la consulta</p> |

En la figura 4.8 se muestra el caso de uso para generar reportes.

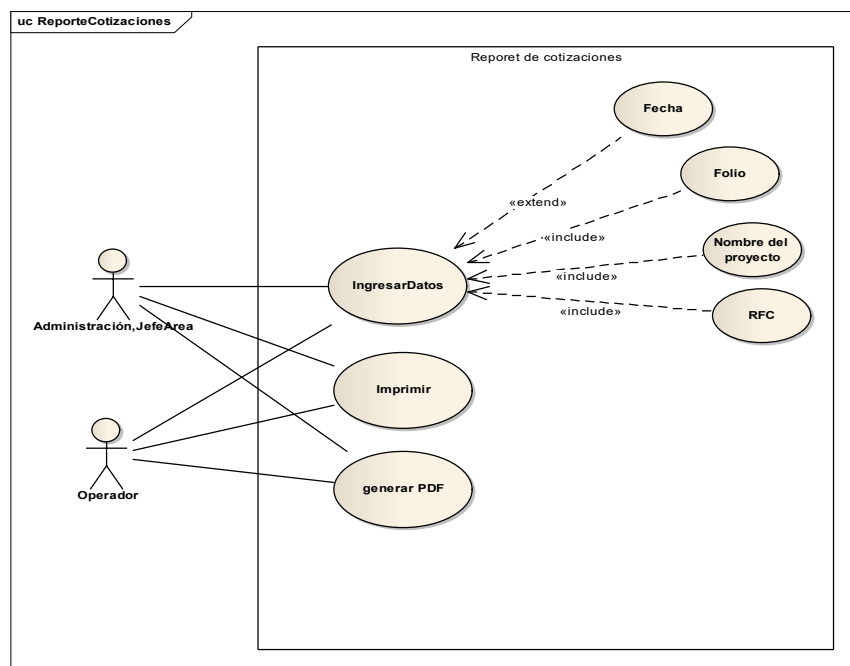


Figura 4.8 Diagrama de caso de uso del reporte de cotización.

En la tabla 4.18 se describe el Sprint para la generación de reporte, y como se realiza esta acción en el controlador.

Tabla 4.25 Sprint Generar reportes del pie tablón

| Reportes de pie tablón | |
|------------------------|---|
| Duración: | 14 días hábiles |
| Objetivo: | Es realizar el funcionamiento para generar reporte del pie tablón |
| Descripción: | En este <i>sprint</i> se programaron las clases CotizadorBean y DetallecotizaDAO para poder generar reportes del pie tablón. Este módulo contempla generar de la base de datos los reportes de los pies tablón de las cotizaciones. Administrador, jefe de área, usuario y el operador pueden generar los reportes. |
| Desarrollo: | Se contemplan 180 líneas de código |

En la figura 4.9 se muestra el caso de uso para generar reportes de pie tablón.

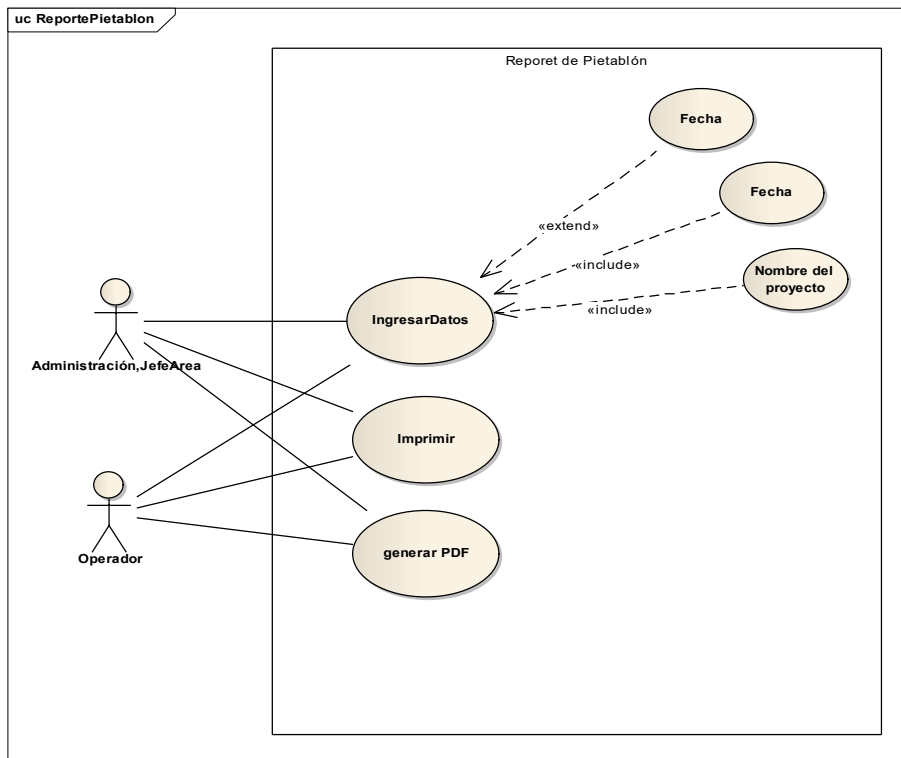


Figura 4.9 Diagrama de caso de uso del pie tablón

En la tabla 4.13 se puede observar el funcionamiento del controlador para la parte de la generación de reportes del pie tablón procesados en cada cotización.

Tabla 4.26 Descripción del controlador para los reportes del pie tablón

| Interfaz | Acciones del controlador |
|---|--|
| Generar reporte del pie tablón procesados en cada cotización. | <p>Se inicia sesión para poder realizar el reporte de las cotizaciones.</p> <p>Se seleccionan los campos para poder generar el reporte.</p> <p>Los datos obtenidos de la interfaz, que son los que el usuario proporciono se mandan a CotizadorBean.</p> <p>En CotizadorBean se procesa la información y se mandan a DetallecotizaDAO.</p> <p>Por medio de DetallecotizaDAO se realiza la consulta a la tabla en la base de datos.</p> <p>Se envían los datos de la consulta</p> |

4.1.7 Sprint 7 Cotización

En la tabla 4.20 se describe el Sprint para generar una cotización y como se realizó el proceso en el controlador.

Tabla 4.27 Sprint Generar una cotización.

| Cotización | |
|--------------|--|
| Duración: | 30 días hábiles |
| Objetivo: | Es realizar el funcionamiento para generar y modificar una cotización. |
| Descripción: | En este sprint se programaron las clases OperacionesBean, y CotizadorDAO para poder crear una cotización. Este módulo contempla realizar todos los cálculos de la lógica de negocios y almacenar la información en una cotización, se han utilizado diferentes clases para las distintas operaciones que realiza dando los privilegios que tiene cada usuario. Teniendo solo el Administrador, jefe y el operador el poder crear, modificar la cotización. |
| Desarrollo: | Se contemplan 1020 líneas de código |

En la tabla 4.21 se puede observar el controlador para el proceso de cotizar en el controlador.

Tabla 4.28 Descripción del controlador para la administración de las cotizaciones

| Interfaz | Acciones del controlador |
|--|--|
| Generación de una cotización de productos. | <p>El inicia sesión para poder realizar la cotización de un cliente.</p> <p>Los datos obtenidos de la interfaz, que son los que el usuario proporciono se mandan a OperacionesBean.</p> <p>En OperacionesBean se procesa la información y se mandan a CotizadorDAO.</p> <p>Por medio de CotizadorDAO se realizan las operaciones y cálculos necesario obtenidos de la base de datos, y así poder realizar la inserción de una cotización en la base de datos.</p> |
| Consulta las cotizaciones almacenadas en la base de datos. | <p>El usuario inicia sesión para poder realizar el listado de las cotizaciones.</p> <p>En OperacionesBean procesa la información y se mandan a CotizadorDAO.</p> <p>Por medio de CotizadorDAO se realiza la consulta a la base de datos y alacena la información.</p> <p>La información obtenida de la base de datos se manda a OperacionesBean.</p> <p>OperacionesBean manda la información para ser presentada en la pantalla y sea visualizada por el usuario.</p> |
| Modificación de una cotización. | <p>Al tener la lista de las cotizaciones se acciona modificar, en el cual OperacionesBean manda información del usuario a la vista para que el usuario modifique los campos.</p> <p>Después de modificar se mandan los datos a OperacionesBean y este realiza la comunicación con CotizadorDAO.</p> <p>CotizadorDAO procesa la información y actualiza la información en la tabla.</p> <p>CotizaDAO obtiene la información actualidad y la manda a OperacionesBean para que la presente al usuario en la interfaz.</p> |

En la figura 4.10 se muestra el caso de uso para la generación de una cotización de productos.

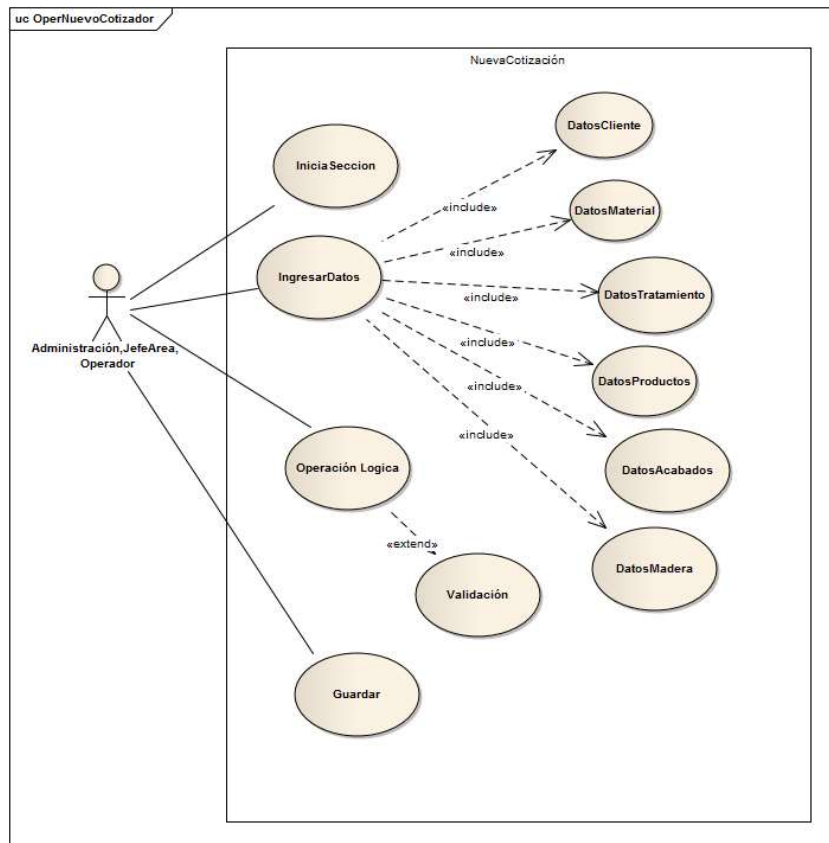


Figura 4.10 Diagrama de caso de uso de cotización

4.1.8 Sprint 8 Eliminar Cotización

En la tabla 4.22 se describe el Sprint para la eliminación de una cotización, y como se realizó esta acción en el controlador.

Tabla 4.29 *Sprint Eliminación de cotizador*

| Eliminar cotización | |
|----------------------------|---|
| Duración: | 30 días hábiles |
| Objetivo: | Es realizar el funcionamiento eliminar una cotización. |
| Descripción: | En este <i>sprint</i> se programaron las clases OperacionesBean y CotizadorDAO para poder modificar y eliminar cotizaciones. Este módulo contempla realizar alguna modificación a las cotizaciones en la base de datos para o eliminarla, se han utilizado diferentes clases para las funciones de modificar y eliminar, dando los privilegios que tiene cada usuario. El Administrador, jefe de área y el operador tienen el poder para modificar y eliminar |
| Desarrollo: | Se contemplan 120 líneas de código |

En la tabla 4.23 se puede observar el controlador para esta parte de los productos.

Tabla 4.30 *Descripción del controlador para la eliminación de una cotización*

| Interfaz | Acciones del controlador |
|--------------------------|---|
| Eliminar una cotización. | <p>Al tener la lista la cotización se acciona eliminar, en el cual OperacionesBean manda información de la cotización a la vista para que el usuario confirme la eliminación.</p> <p>Después de eliminar se mandan los datos a OperacionesBean y este realiza la comunicación con CotizadorDAO.</p> <p>CotizadorDAO procesa la información y actualiza la información en la tabla.</p> <p>CotizadorDAO obtiene la información actualidad y la manda a OperacionesBean para que la presenta al usuario en la interfaz.</p> |

En la figura 4.11 se muestra el caso de uso para eliminar una cotización.

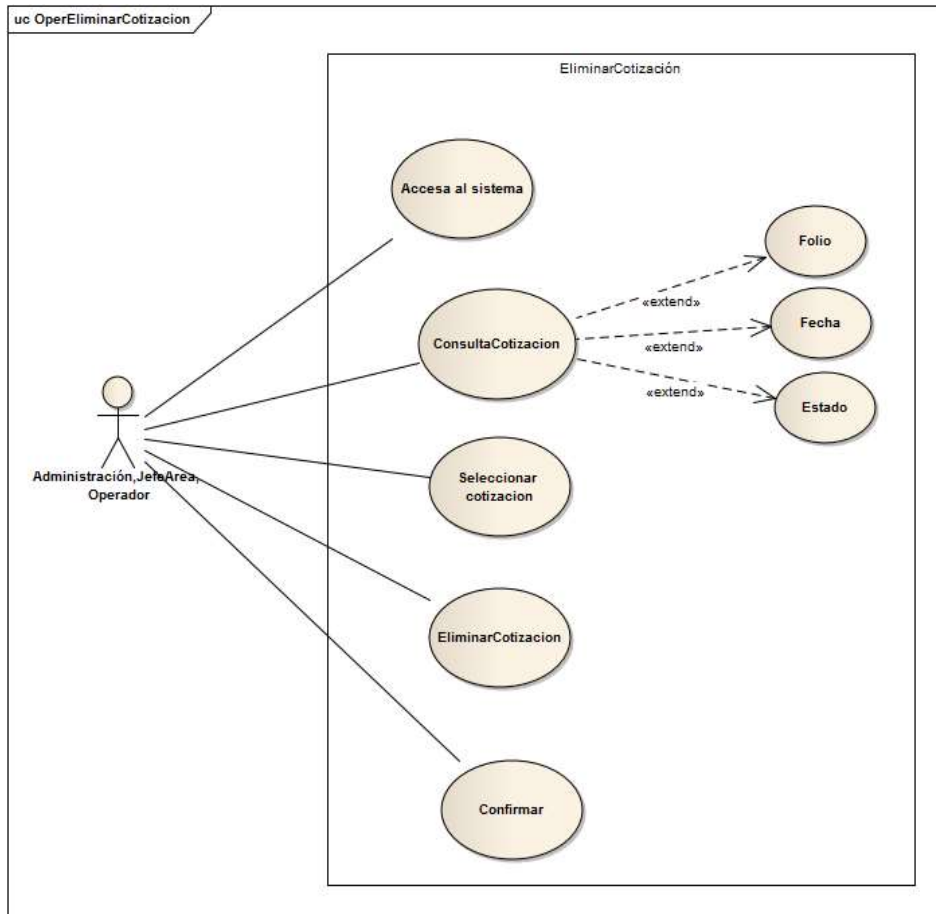


Figura 4.11 Diagrama de caso de uso de eliminación de cotizador

4.2 Comentarios

Aplicando una metodología para el desarrollo del componente controlador, se logra el desarrollo de cada módulo que integra el sistema cotizador.

En este Capítulo se mostró el desarrollo de cada sprint donde se describe el comportamiento de cada componente, y posteriormente se describe el funcionamiento que realiza el controlador para poder realizar las tareas solicitadas, también se presentó el diagrama de caso de uso de cada componente para así tener un panorama de las actividades que realiza cada actor.

El desarrollo bajo Scrum dio la oportunidad de realizar cada componente en conjunto con el cliente ya que se tenían retroalimentaciones para tener un módulo funcional que cumplía con las características solicitadas por el cliente.

Por último, se realiza la integración del componente vista y modelo para así tener un módulo entregable y al final tener el sistema cotizador.

5 Pruebas y Resultados

En este Capítulo se muestran los resultados obtenidos de la implementación e integración del sistema cotizador. Además de la validación de cada componente que integra el sistema, para tener un correcto funcionamiento del cotizador, así como los apartados que lo integran.

5.1 Descripción del plan de pruebas

Para validar que el sistema cumpla con los requisitos que el cliente desea, se realizó un plan de pruebas (ver anexo A) para comprobar si el sistema funciona de manera correcta.

Se utilizó el modelo en espiral para la elaboración del plan pruebas mencionado en la sección 2.8 en la que se contemplan 4 etapas para el recorrido del modelo, y en cada una realizando lo siguiente:

- **Pruebas de Unidad:** En estas pruebas se realizarán los casos de prueba de uso, en donde se tiene como objetivo localizar defectos y probar el funcionamiento del software.
- **Pruebas de Integración:** En estas pruebas se realizarán reportes de la integración de los distintos componentes, para validar el flujo de la información entre los módulos y descubrir si tiene errores en la asociación con la interfaz.
- **Pruebas de Validación:** En esta se verifica que todos los elementos cumplen con todos los requerimientos funcionales y no funcionales.
- **Pruebas de Sistema:** En esta prueban se verifica que los elementos interactúen entre de forma correcta en todos los componentes del sistema, en el cual se presenta un reporte validando cada módulo.

En este Capítulo se presentan solo las pruebas de integración que se aplicaron a cada módulo del sistema cotizador.

En el anexo A: Plan de pruebas, se muestran la planeación completa de todas las pruebas aplicadas al sistema. Los casos de prueba, las pruebas de unidad y las pruebas de sistema. Una vez ejecutadas las pruebas, y al detectarse errores en el sistema se realizará un seguimiento para corregir los incidentes presentados.

A continuación, se presentan las pruebas de integración de todos los módulos del sistema cotizador.

5.2 Módulo validación de usuarios

En la ilustración 5.1 se muestra la pantalla de inicio de sesión, la cual solicita introducir el usuario y contraseña para que el sistema realice la validación del usuario y permita el acceso al sistema cotizador según correspondan sus respectivos privilegios.

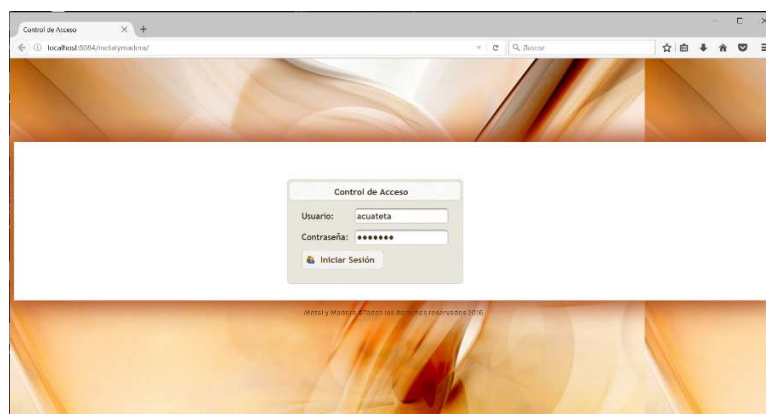


Figura 5.1 Interfaz: Inicio de sesión

En la figura 5.2 se muestra la ventana del resultado satisfactorio del acceso al sistema con el usuario y contraseña, dando la bienvenida en la parte superior derecha al usuario que se autentico.

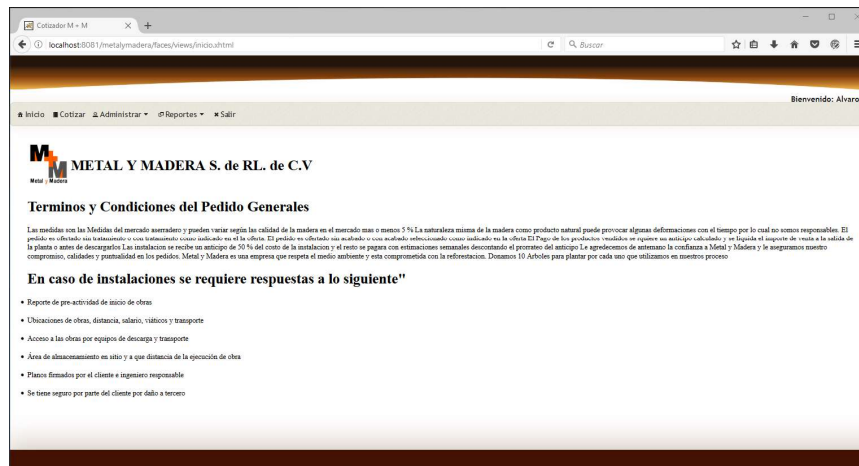


Figura 5.2 Bienvenida del sistema

En la tabla 5.1 se muestran las pruebas de integración para el inicio de sesión, en la que se valida el acceso al sistema con un usuario y contraseña. La tabla muestra el componente o subsistema a evaluar. En la descripción de la prueba se muestran las acciones que se deben realizar y en la columna de elementos a integrar se contemplan los elementos que serán evaluados. En la descripción se indican los datos de entrada para validar los componentes y en el resultado se muestra si la prueba fue exitosa. Finalmente, en la descripción de defectos se enlistan las fallas que se presentaron al realizar la prueba.

Tabla 5.1 Prueba de integración: Inicio de sesión

| Componente o subsistema | Descripción de la prueba | Elementos a integrar | Descripción | Resultado | Descripción de defecto |
|-------------------------|---|----------------------|---|--|------------------------|
| Inicio de sesión | Iniciar el sistema cotizador y solicitar usuario y contraseña para acceder. | Usuario | Se accede con un usuario administrador, jefe de área, operador y usuario. | Se accede de forma satisfactoria al sistema con los distintos usuarios | Sin defectos. |
| | | Contraseña | | | |

5.3 Módulo administración de usuarios

En la figura 5.3 se muestra la interfaz para el módulo de administración de los usuarios, la cual se encuentra dentro del menú administrar y usuarios.

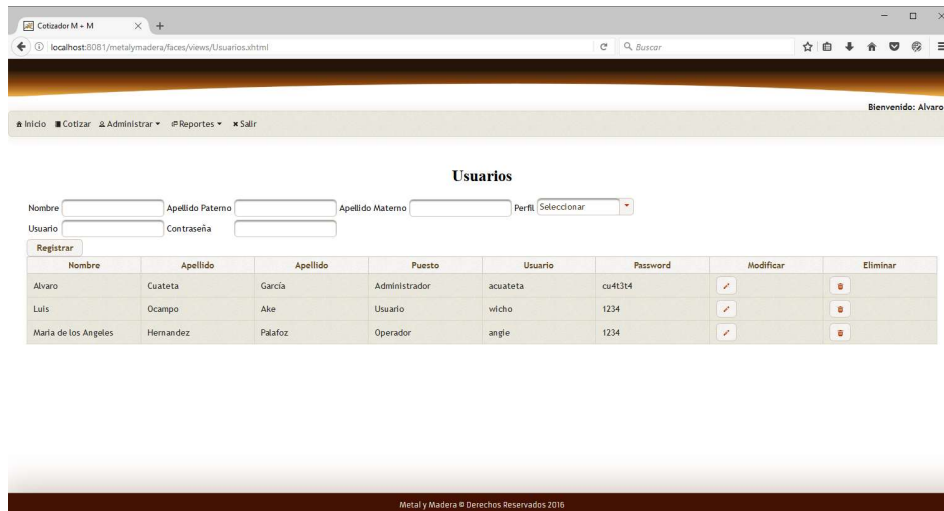


Figura 5.3 Administración de usuarios

En la tabla 5.2 se muestra la prueba de integración para el módulo administración de los usuarios, teniendo resultados satisfactorios en la integración de los componentes.

Tabla 5.2 Administración de usuarios

| Componente o subsistema | Descripción de la prueba | Elementos a integrar | Descripción | Resultado | Descripción de defecto |
|-------------------------|--|----------------------|---|--|------------------------|
| Usuarios | La utilización del botón de registrar, modificar y eliminar. Así como consultar. | Registrar | Almacena la información en la BD | Se almacenan los datos en la BD correctamente | Sin defectos. |
| | | Listar | Lista los datos almacenados en la BD. | Muestra los datos de la BD correctamente. | Sin defectos |
| | | Modificar | Modifica los datos almacenados en la BD | Se realiza la modificación de los campos en la BD correctamente. | Sin defectos |
| | | Eliminar | Elimina los datos de la BD. | Se elimina el registro de la BD correctamente. | Sin defectos |

5.4 Módulo administración de clientes

En la figura 5.4 se presenta la interfaz para el módulo administración de los clientes, se accede desde el menú administrar y posteriormente clientes.

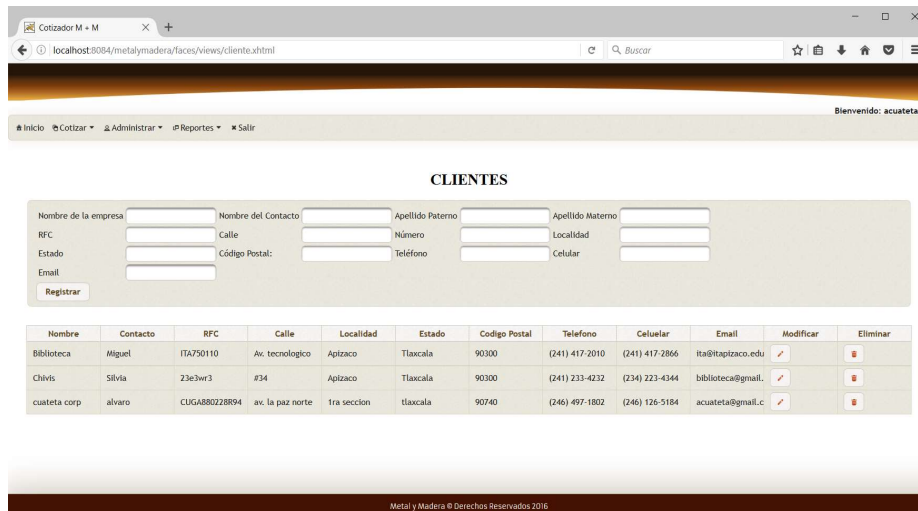


Figura 5.4 Interfaz: Administración de clientes

En la tabla 5.3 se muestra la prueba de integración para la administración del módulo de clientes, en la cual se tienen resultados satisfactorios en la integración de los componentes.

Tabla 5.3 Administración de clientes

| Componente o subsistema | Descripción de la prueba | Elementos a integrar | Descripción | Resultado | Descripción de defecto |
|-------------------------|--|----------------------|---|--|------------------------|
| Clientes | La utilización del botón de registrar, modificar y eliminar. Así como consultar. | Registrar | Almacena la información en la BD | Se almacenan los datos en la BD correctamente | Sin defectos. |
| | | Listar | Lista los datos almacenados en la BD. | Muestra los datos de la BD correctamente. | Sin defectos |
| | | Modificar | Modifica los datos almacenados en la BD | Se realiza la modificación de los campos en la BD correctamente. | Sin defectos |
| | | Eliminar | Elimina los datos de la BD. | Se elimina el registro de la BD correctamente. | Sin defectos |

5.5 Módulo administración de productos

En la figura 5.5 se presenta la interfaz para el módulo administración de los productos que maneja el cotizador, su ubicación es en el menú administrar y posteriormente productos.

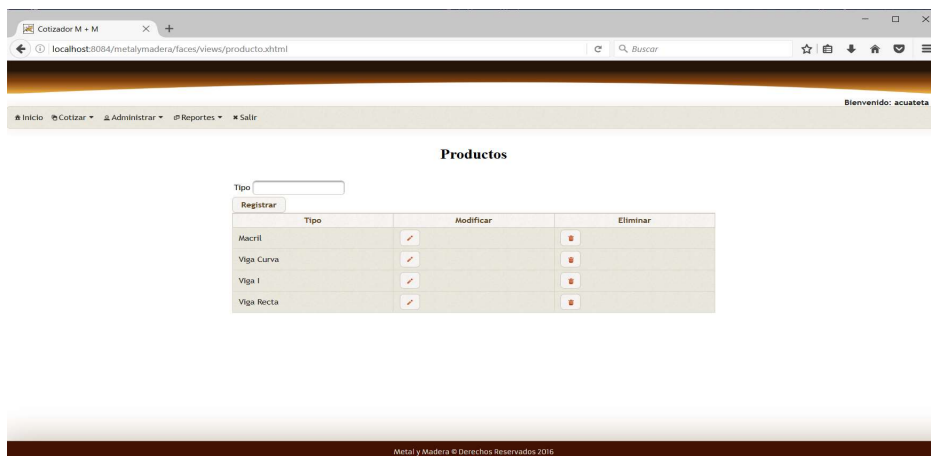


Figura 5.5 Interfaz: Administración de productos

En la tabla 5.4 se muestra la prueba de integración para la administración del módulo de los productos, en la cual se tienen resultados satisfactorios en la integración de los componentes.

Tabla 5.4 Administración de productos

| Componente o subsistema | Descripción de la prueba | Elementos a integrar | Descripción | Resultado | Descripción de defecto |
|-------------------------|--|----------------------|---|--|------------------------|
| Productos | La utilización del botón de registrar, modificar y eliminar. Así como consultar. | Registrar | Almacena la información en la BD | Se almacenan los datos en la BD correctamente | Sin defectos. |
| | | Listar | Lista los datos almacenados en la BD. | Muestra los datos de la BD correctamente. | Sin defectos |
| | | Modificar | Modifica los datos almacenados en la BD | Se realiza la modificación de los campos en la BD correctamente. | Sin defectos |
| | | Eliminar | Elimina los datos de la BD. | Se elimina el registro de la BD correctamente. | Sin defectos |

5.6 Módulo administración de materiales

En la figura 5.6 se presenta la interfaz para el módulo administración de los materiales, su ubicación es en el menú administrar y posteriormente materiales.

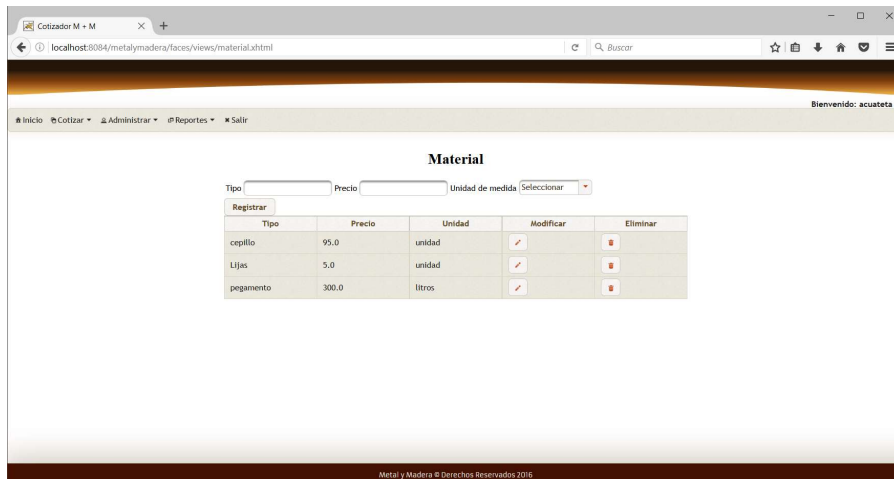


Figura 5.6 Interfaz: Administración de materiales

En la tabla 5.5 se muestra la prueba de integración para la administración del módulo de los materiales, en la cual se tienen resultados satisfactorios en la integración de los componentes.

Tabla 5.5 Administración de materiales

| Componente o subsistema | Descripción de la prueba | Elementos a integrar | Descripción | Resultado | Descripción de defecto |
|-------------------------|--|----------------------|---|--|------------------------|
| Materiales | La utilización del botón de registrar, modificar y eliminar. Así como consultar. | Registrar | Almacena la información en la BD | Se almacenan los datos en la BD correctamente | Sin defectos. |
| | | Listar | Lista los datos almacenados en la BD. | Muestra los datos de la BD correctamente. | Sin defectos |
| | | Modificar | Modifica los datos almacenados en la BD | Se realiza la modificación de los campos en la BD correctamente. | Sin defectos |
| | | Eliminar | Elimina los datos de la BD. | Se elimina el registro de la BD correctamente. | Sin defectos |

5.7 Módulo administración de madera

En la figura 5.7 se presenta la interfaz para la administración de la madera, su ubicación es en el menú administrar y posteriormente madera.

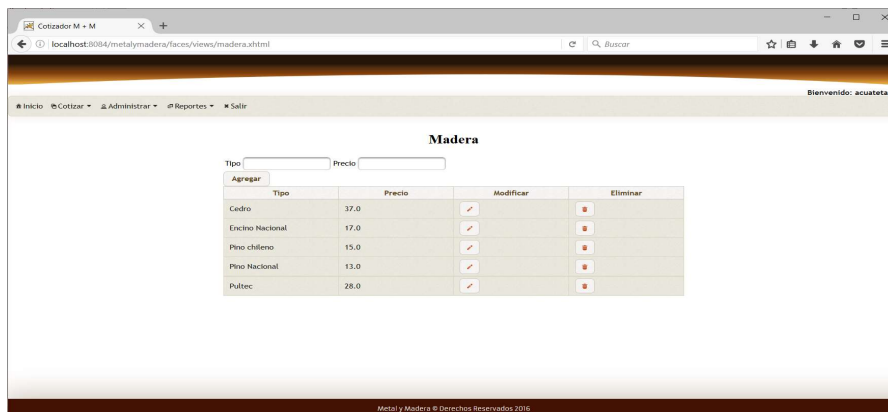


Figura 5.7 Interfaz: Administración de madera

En la tabla 5.6 se muestra la prueba de integración para la administración del módulo de madera, en la cual se tienen resultados satisfactorios en la integración de los componentes.

Tabla 5.6 Administración de madera

| Componente o subsistema | Descripción de la prueba | Elementos a integrar | Descripción | Resultado | Descripción de defecto |
|-------------------------|--|----------------------|---|--|------------------------|
| Madera | La utilización del botón de registrar, modificar y eliminar. Así como consultar. | Registrar | Almacena la información en la BD | Se almacenan los datos en la BD correctamente | Sin defectos. |
| | | Listar | Lista los datos almacenados en la BD. | Muestra los datos de la BD correctamente. | Sin defectos |
| | | Modificar | Modifica los datos almacenados en la BD | Se realiza la modificación de los campos en la BD correctamente. | Sin defectos |
| | | Eliminar | Elimina los datos de la BD. | Se elimina el registro de la BD correctamente. | Sin defectos |

5.8 Módulo administración de tratamientos

En la figura 5.8 se presenta la interfaz para el módulo administración de los tratamientos, su ubicación es en el menú administrar y posteriormente tratamientos.

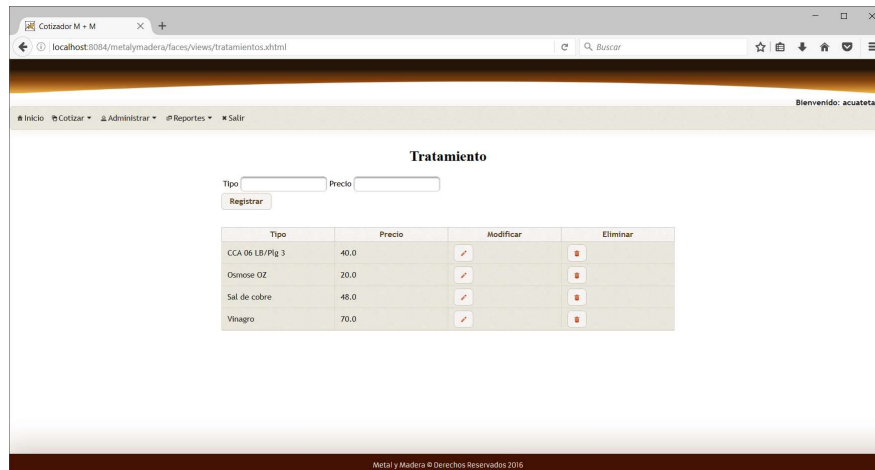


Figura 5.8 Interfaz: Administración de tratamientos

En la tabla 5.7 se muestra la prueba de integración para la administración del módulo de los tratamientos, en la cual se tienen resultados satisfactorios en la integración de los componentes.

Tabla 5.7 Administración de tratamientos

| Componente o subsistema | Descripción de la prueba | Elementos a integrar | Descripción | Resultado | Descripción de defecto |
|-------------------------|--|----------------------|---|--|------------------------|
| Tratamiento | La utilización del botón de registrar, modificar y eliminar. Así como consultar. | Registrar | Almacena la información en la BD | Se almacenan los datos en la BD correctamente | Sin defectos. |
| | | Listar | Lista los datos almacenados en la BD. | Muestra los datos de la BD correctamente. | Sin defectos |
| | | Modificar | Modifica los datos almacenados en la BD | Se realiza la modificación de los campos en la BD correctamente. | Sin defectos |
| | | Eliminar | Elimina los datos de la BD. | Se elimina el registro de la BD correctamente. | Sin defectos |

5.9 Módulo administración de acabados

En la figura 5.9 se presenta la interfaz para el módulo administración de los tratamientos, su ubicación es en el menú administrar y posteriormente tratamientos.

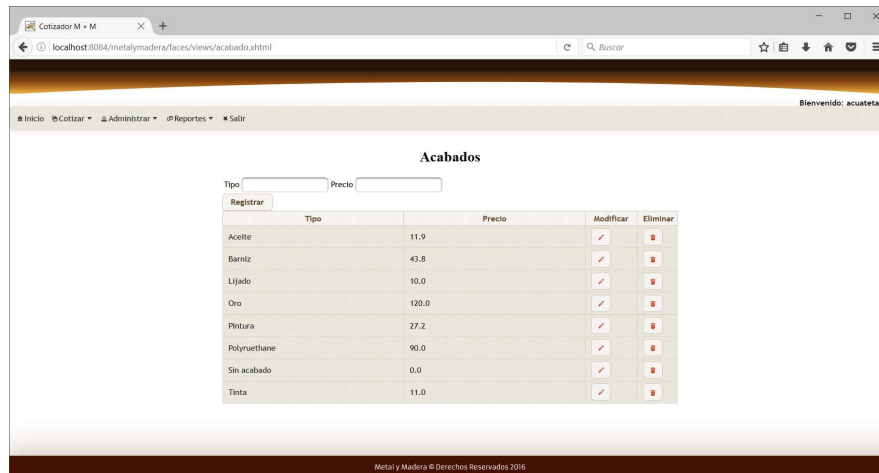


Figura 5.9 Interfaz: Administración de acabados

En la tabla 5.8 se muestra la prueba de integración para la administración del módulo de los acabados, en la cual se tienen resultados satisfactorios en la integración de los componentes.

Tabla 5.8 Administración de acabados

| Componente o subsistema | Descripción de la prueba | Elementos a integrar | Descripción | Resultado | Descripción de defecto |
|-------------------------|--|----------------------|---|--|------------------------|
| Acabados | La utilización del botón de registrar, modificar y eliminar. Así como consultar. | Registrar | Almacena la información en la BD | Se almacenan los datos en la BD correctamente | Sin defectos. |
| | | Listar | Lista los datos almacenados en la BD. | Muestra los datos de la BD correctamente. | Sin defectos |
| | | Modificar | Modifica los datos almacenados en la BD | Se realiza la modificación de los campos en la BD correctamente. | Sin defectos |
| | | Eliminar | Elimina los datos de la BD. | Se elimina el registro de la BD correctamente. | Sin defectos |

5.10 Módulo administración de personal

En la figura 5.10 se presenta la interfaz para el módulo administración del personal, su ubicación es en el menú administrar y posteriormente personal.

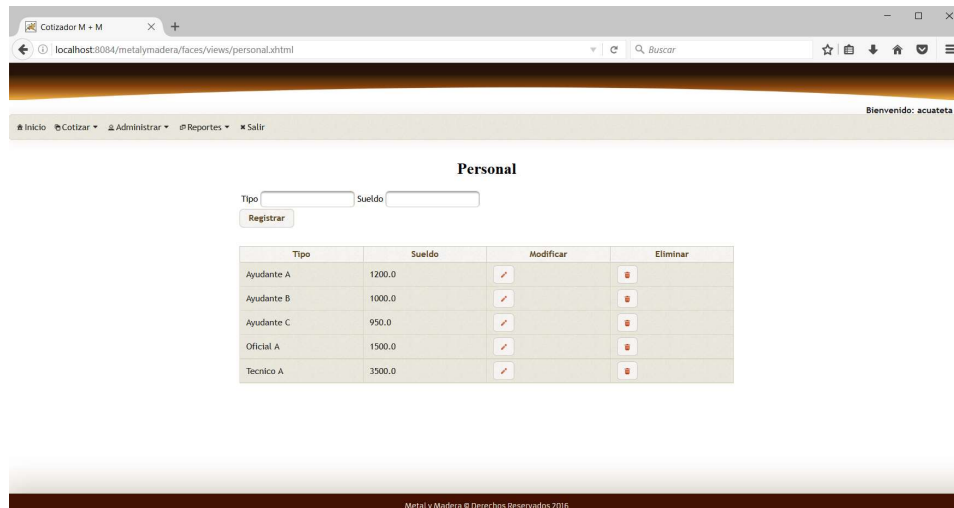


Figura 5.10 Interfaz: Administración de personal

En la tabla 5.9 se muestra la prueba de integración para la administración del módulo del personal, en la cual se tienen resultados satisfactorios en la integración de los componentes.

Tabla 5.9 Administración de personal

| Componente o subsistema | Descripción de la prueba | Elementos a integrar | Descripción | Resultado | Descripción de defecto |
|-------------------------|--|----------------------|---|--|------------------------|
| Personal | La utilización del botón de registrar, modificar y eliminar. Así como consultar. | Registrar | Almacena la información en la BD | Se almacenan los datos en la BD correctamente | Sin defectos. |
| | | Listar | Lista los datos almacenados en la BD. | Muestra los datos de la BD correctamente. | Sin defectos |
| | | Modificar | Modifica los datos almacenados en la BD | Se realiza la modificación de los campos en la BD correctamente. | Sin defectos |
| | | Eliminar | Elimina los datos de la BD. | Se elimina el registro de la BD correctamente. | Sin defectos |

5.11 Módulo de reportes

En la figura 5.11 se presenta la interfaz para el módulo de generación de reportes de las cotizaciones existentes en el sistema, su ubicación es en al menú reportes.

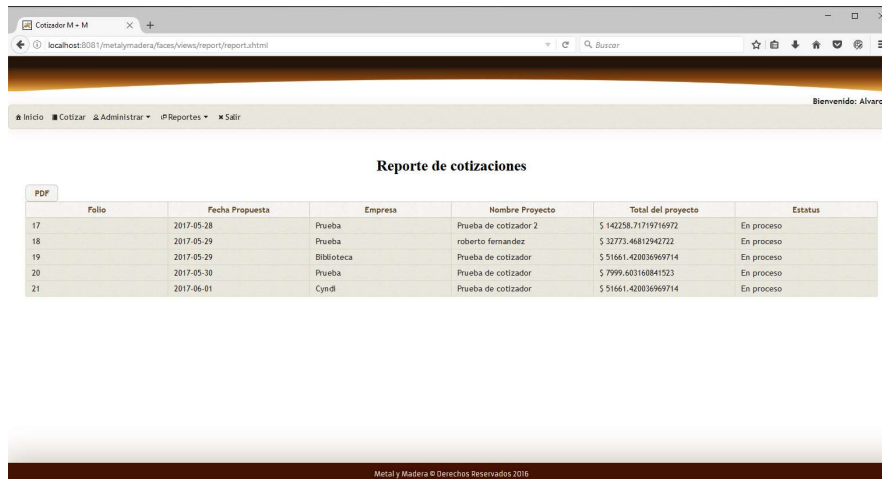


Figura 5.11 Interfaz: reportes de cotizaciones

En la tabla 10 se muestra la prueba de integración para el módulo generación de los reportes, en la cual se tienen resultados correctos en la integración de los componentes.

Tabla 5.10 Reportes

| Componente o subsistema | Descripción de la prueba | Elementos a integrar | Descripción | Resultado | Descripción de defecto |
|--------------------------|---|----------------------|--|---|------------------------|
| Reportes de cotizaciones | Utilización de botón “PDF” y listado de las cotizaciones. | PDF | Permite generar un reporte en PDF de las cotizaciones. | El reporte se genera correctamente. | Sin defectos. |
| | | Listar | Lista las cotizaciones que existen en la BD. | Se listan las cotizaciones correctamente. | Sin defectos |
| Orden de producción | Búsqueda por folio y generación de archivos PDF | Buscar | Permite buscar el folio en la BD. | Realiza la búsqueda de forma correcta | Sin defectos. |
| | | PDF | Genera un documento PDF de la cotización. | Se genera el documento correctamente. | Sin defectos |

Tabla 5.11 Reportes (continuación)

| Componente o subsistema | Descripción de la prueba | Elementos a integrar | Descripción | Resultado | Descripción de defecto |
|-------------------------|--|----------------------|--|---|------------------------|
| Cotización por fecha | Realizar una búsqueda por fecha y estado | Campos de fechas. | Permite buscar cotizaciones en un rango de fechas. | Realiza la búsqueda en el rango de fechas establecidas correctamente. | Sin defectos. |
| | | Opciones de estado | Permite seleccionar el estado de la cotización. | Se muestran los resultados conforme la selección del estado. | Sin defectos. |
| | | PDF | Genera un documento PDF de la cotización. | Se genera el documento correctamente. | Sin defectos |

5.12 Módulo cotización

En la figura 5.12 se muestra la interfaz para el módulo administración de las cotizaciones y el botón para generar una nueva cotización, este apartado lo podemos encontrar en el menú cotizar.

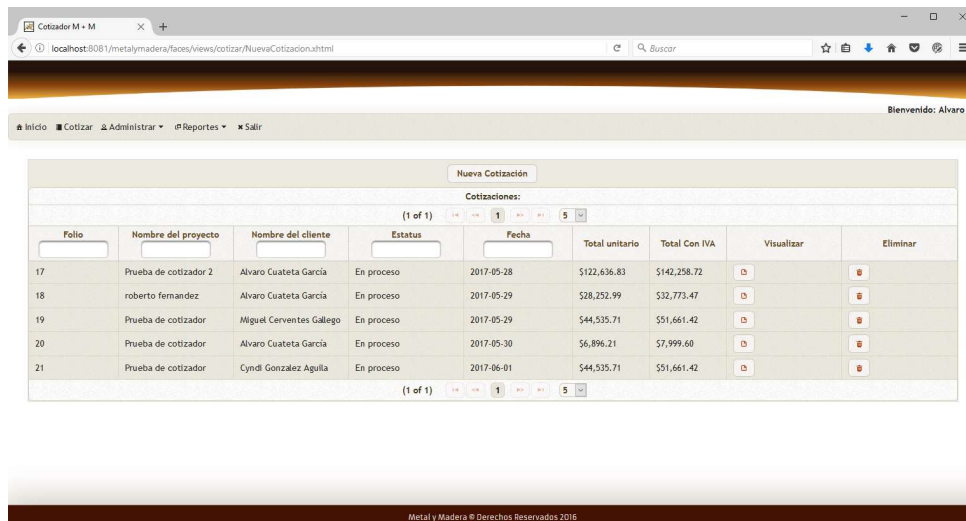


Figura 5.12 Interfaz: Administración de cotizaciones

En la tabla 11 se muestra la prueba de integración para el módulo administración de las cotizaciones, en la cual se tienen resultados correctos con la integración de los componentes.

Tabla 5.12 Administración de cotizaciones

| Componente o subsistema | Descripción de la prueba | Elementos a integrar | Descripción | Resultado | Descripción de defecto |
|-------------------------|--|----------------------|--|--|------------------------|
| Cotización | La utilización del botón Nueva cotización, modificar y eliminar. | Nueva Cotización | Muestra la ventana para generar una cotización. | Realiza el direccionamiento correcto | Sin defectos. |
| | | Listar | Lista las cotizaciones existentes en la BD | Muestra los datos de la BD correctamente. | Sin defectos |
| | | Modificar | Permite la modificación de las cotizaciones almacenadas en la BD | Se realiza la modificación de los campos en la BD correctamente. | Sin defectos |
| | | Eliminar | Permite la eliminación de la cotización. | Se elimina la cotización correctamente. | Sin defectos |

5.13 Módulo nueva cotización

En la figura 5.13 se muestra la interfaz para la generación de una nueva cotización, en donde se agregan los productos a cotizar y se realizan los cálculos correspondientes, para acceder a este apartado debemos entrar al menú cotizar y seleccionar nueva cotización.

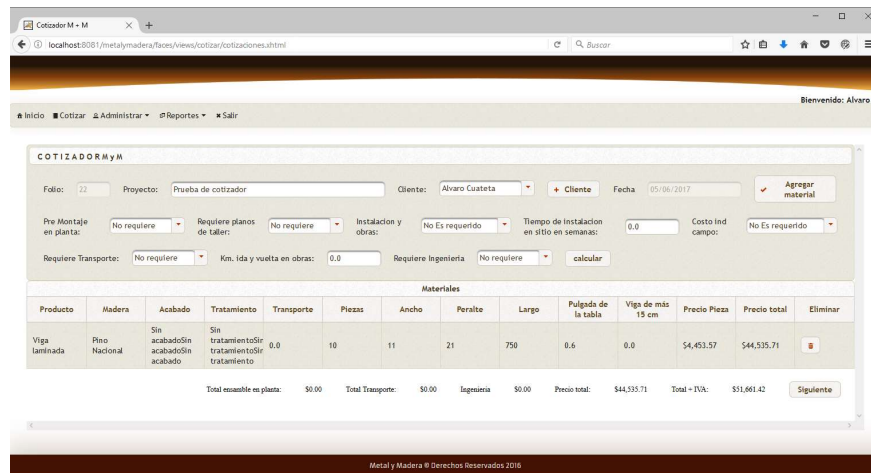


Figura 5.13 Interfaz: Nueva cotización

En la figura 5.14 se muestra la interfaz para agregar un producto con las características deseadas, al dar clic en agregar el producto se agrega en la ventana de nueva cotización.



Figura 5.14 Interfaz: agregar productos

En la figura 5.15 se muestra la ventana del resumen de la cotización, donde se pueden visualizar toda la información de los productos y el costo de los mismos.

Figura 5.15 Interfaz: resumen de cotización

En la tabla 5.12 se muestra la prueba de integración para la generar una nueva cotización, en la cual se tienen resultados correctos con la integración de los distintos componentes.

Tabla 5.13 Nueva cotización

| Componente o subsistema | Descripción de la prueba | Elementos a integrar | Descripción | Resultado | Descripción de defecto |
|-------------------------|--|----------------------|--|---|------------------------|
| Nueva cotización | La utilización de los botones, clientes calcular, agregar material y guardar | Cliente | Permite agregar clientes nuevos a la BD | Agrega un cliente nuevo correctamente. | Sin defectos. |
| | | Agregar material | Permite agregar materiales a la cotización | Muestra la ventana para agregar y agrega los productos. | Sin defectos |
| | | Calcular | Permite realizar el cálculo de la cotización. | Realiza el cálculo de los costos correctamente. | Sin defectos |
| | | Siguiente | Muestra el resumen de la cotización | Se muestra correctamente el detalle de la cotización. | Sin defectos |
| | | Imprimir | Imprime el detalle de la cotización. | Se genera un PDF correctamente. | Sin defectos |
| | | Guardar | Guarda las cotizaciones y el detalle de cotización en la BD. | Se guardan los registros correctamente. | Sin defectos |

5.14 Comentarios

Para validar el trabajo realizado se aplicó un plan de pruebas, todo esto para garantizar la fiabilidad y funcionamiento del sistema, encontrando y corrigiendo los errores de forma oportuna.

Se aplicaron pruebas unitarias que fueron validadas con los casos de prueba, posteriormente se aplicaron pruebas de integración para el funcionamiento de los componentes, el reporte de pruebas de integración esta presentado en este Capítulo, el pan de pruebas también incluye las pruebas de sistema y de validación.

El resultado obtenido después de múltiples pruebas fue un sistema sin fallas para garantizar el funcionamiento. Se probó y validó cada módulo del sistema para al final tener un software funcional y sin errores.

6 Conclusiones y recomendaciones

En este Capítulo se presentan las conclusiones obtenidas en el desarrollo del sistema cotizador, además de los trabajos que se pueden seguir realizando tomando como referencia lo presentado anteriormente.

6.1 Conclusiones

En la primera etapa se realiza el análisis de los requisitos como se presentó en la sección 3.2.4, donde se obtiene los requerimientos funcionales, no funcionales y la lista de actores que conforman el sistema. Posteriormente se describe toda la parte teórica utilizada para el desarrollo en el capítulo 2, dando un panorama de las herramientas que existen y su funcionamiento.

Así mismo, se realizó un análisis de las diferentes metodologías presentadas en la sección 2.5, y finalmente para la obtención de los requisitos y pruebas se utilizó una metodología en espiral, la cual se puede consultar en la sección 4.1.1 y para el demás desarrollo se utilizó SCRUM.

El objetivo de este trabajo fue dividir el desarrollo del módulo controlador del sistema cotizador, para después realizar la integración con la vista y el modelo.

Al combinar la metodología en espiral en el primer *Sprint* (sección 4.1.1) para la recolección y validación de los requisitos, donde se obtienen buenos resultados ya que paso por un proceso de:

- Adquisición de los requerimientos en este se recopiló la información sobre el sistema cotizador y el sistema existente, de estos también se pudieron identificar los requerimientos del usuario y los del sistema.

- Especificación de requerimientos, en este apartado se identificó en términos generales cual es la funcionalidad que tendrá el producto, expresando cada requerimiento funcional como un caso de uso.
- Validación de los requerimientos, se procedió a la validación de los requisitos directamente con el cliente donde se realizaron prototipos aplicando los cambios y mejoras correspondientes, haciendo de estos un requerimiento que cumple con las características que el cliente deseaba.

Después del proceso de validación por el cliente se realizó el segundo *sprint*, en el que se describió el desarrollo del controlador con la metodología Scrum en la sección 3.1, así como la descripción de cada sprint para cada componente del sistema en la sección 4.1, y posteriormente se realizan las pruebas de cada módulo que comprende el sistema.

Por último, se realizó una serie de pruebas presentadas en el Capítulo 5, para encontrar los errores y así poder garantizar el desarrollo y funcionamiento del sistema.

Al aplicar la metodología en espiral para la validación de las pruebas (sección 2.9), se puede decir que después de corregir los errores encontrados en las pruebas aplicadas, se logra garantizar un sistema confiable al 100 por ciento como se muestra en el plan de pruebas (anexo A). El sistema presentó múltiples fallas en las primeras etapas las cuales se corrigieron aplicando y repitiendo las distintas pruebas que se pueden apreciar en el reporte de incidencia (anexo A), todo esto para garantizar que el módulo no presentara errores y así poder entregar un sistema confiable y sin fallas.

Se recomienda la aplicación de la metodología Scrum en combinación con el patrón de diseño Modelo-Vista-Controlador para el desarrollo de un software, ya que se tiene un resultado satisfactorio tanto para el cliente como para el proyecto por la constante comunicación con el cliente.

Al trabajar con la arquitectura Modelo-Vista-Controlador en combinación con herramientas de desarrollo como Hibernate y JavaServer se Faces, se facilita la

implementación de sistemas que proporcionan grandes beneficios en la industria, ya que mejoran sus procesos, tiempo y competitividad.

Por lo tanto, teniendo como base la pregunta de investigación ¿Es posible implementar el módulo controlador, bajo el modelo MVC, para un sistema cotizador? se llega a la conclusión de que con una buena comunicación en equipo ha sido posible el desarrollo aplicando SCRUM en combinación con el patrón de diseño Modelo-Vista-Controlador y diferentes herramientas que facilitan el desarrollo e implementación del proyecto y así poder entregar un sistema eficiente que garantiza la integridad de la información.

6.2 Trabajos futuros

Como trabajo futuro se propone continuar con la segunda fase del sistema para que pueda trabajar en la nube y proporcionar mayores beneficios como:

- Los usuarios puedan realizar una cotización de los productos, ya que este sistema está limitado a los usuarios de la empresa.
- Aumentar la seguridad del sistema ya que al estar al público en general puede ser víctima de ataques.

Bibliografía

- Álvarez, J. M. (2004). *Aplicación de un Sistema Experto para el desarrollo de Sistema Evaluador del modelo Capability Maturity Model (CMM) niveles dos y tres. Licenciatura en Ingeniería en Sistemas Computacionales*. Cholula, Puebla, México: Universidad de las Américas Puebla.
- Askarinejadamiri, Z. (2016). Personality requirements in requirement engineering. *Second International Conference on Web Research (ICWR)*.
- Axa, M. (2016, Septiembre 2016). *Axa reinventando los seguros*. Retrieved from Axa reinventando los seguros: <https://axa.mx/home>
- Bambu Code S.A de C.V. (2016, Septiembre 13). *cotizador web*. Retrieved from cotizador web: <http://cotizadorweb.com/>
- Bergin, J. (2016, Mayo 30). *Building Graphical User Interfaces*. Retrieved from <http://csis.pace.edu/~bergin/mvc/mvcgui.html>
- Buitrago, J. A. (2013). Propuesta de un Framework Multinivel para el diseño de laboratorio de acceso remoto. *UIS.Ingeniería*, 35-45.
- Caballé, S., Ortega, J.-A., & Camps, J.-M. (2014). A Presentation Framework to Simplify the Development of Java EE Application Thin Clients. *Eighth International Conference on Complex, Intelligent and Software Intensive Systems*.
- Carvajal, L., Moreno, A. M., & Sanch, M. I. (2013, Noviembre). Usability through Software Design. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*.
- Cervantes, E. (2013). *DISEÑO E IMPLEMENTACIÓN DE LA BASE DE DATOS DE UN SISTEMA DE INFORMACIÓN PARA LA ADMINISTRACION DE DOCUMENTOS DE CALIDAD DE UNA EMPRESA PRODUCTORA DE FILTROS DE PAPEL* Tesis MSC. Apizaco: Instituto Tecnológico de Apizaco.

- Crookshanks, E. (2015). *Practical Software Development Techniques: Tools and Techniques for Building*. New York: Springer Science+Business Media New York.
- Deitel, P. J., & Deitel, H. M. (2008). *CÓMO PROGRAMAR EN JAVA*. México: PEARS ON EDUCACIÓN.
- Dragos, P. P., & Altar, A. (2014). Designing an MVC Model for Rapid Web Application Development. *ScienceDirect*, 1172 – 1179.
- Draxler, S., Gunnar, S., & Boden, A. (2014, Noviembre 1). Keeping the Development Environment Up to Date A Study of the Situated Practices of Appropriating the Eclipse IDE. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 40.
- ExportandoOnline. (2016, Septiembre 13). *ExportandoOnline*. Retrieved from ExportandoOnline: <http://www.exportando-online.com/portales-web-diseno-web.php>
- Guang-yong, H. (2011). study and practice of Import Scrum agile software development. *IEEE*, 217-220.
- Gutiérrez, G. I. (2013). *“Implementación de la lógica de negocio de un sistema informático para la administración de documentos del sistema de calidad de una empresa fabricante de filtros de papel, usando metodologías ágiles” Tesis MSC*. Apizaco, Tlaxcala: INSTITUTO TECNOLÓGICO DE APIZACO.
- Hamid Mcheick, Y. Q. (2011). DEPENDENCY OF COMPONENTS IN MVC DISTRIBUTED ARCHITECTURE. *IEEE CCECE*.
- Hasan, S., & R.K., I. (2011). An integrated approach of MAS-CommonKADS, Model-View-Controller and web application optimization strategies for web-based expert system development. *Expert Systems with Applications*, 417–428.

- Hoda, R., Noble, J., & Marshall, S. (2013, Marzo). Self-Organizing Roles on Agile Software Development Teams. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*.
- Inteco. (2008). *GUÍA DE MEJORES PRÁCTICAS DE CALIDAD DE PRODUCTO*. Inteco.
- Inteligencia en Tecnología y Servicios, S.C. (2016, Septiembre 13). *newWweb*. Retrieved from newWweb: <http://www.newwwweb.com.mx/node/129>
- Jin, P., & Yao, J. (2014). Design and Realization of College Service Center System Based on MVC. *IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA)*.
- Latorre, R. (2014). Effects of Developer Experience on Learning and Applying Unit Test-Driven Developmen . *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*.
- Manager, S. (2015). *Gestión de proyectos Scrum Manager*. Scrum Manager®. Retrieved from <http://www.scrummanager.net/bok>
- MANN, K. D. (2005). *JavaServer Faces in Action*. Greenwich: MANNING.
- Maras, M. O., Stula, J. C., & Ivica, C. (2013). Identifying Code of Individual Features in Client-Side Web Applications. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERIN*, 39.
- Martinez, R., & Barrio, P. (2016, Noviembre 10). *Sobre Pruebas de software y Servicios de TI*. Retrieved from <http://excelza.blogspot.mx/2009/12/sobre-pruebas-de-software-y-servicios.html>
- Mastura, H., & Rusli, A. (2014). An Evaluation on Components of Experience Based Factory Model in Requirement Engineering Process: A Preliminary Study. *International Conference on Information Technology and Multimedia (ICIMU)*, 18 – 20.

- Miwep.com.mx. (2016, Septiembre 13). *MiWeb Host*. Retrieved from MiWeb Host: <https://miwep.com.mx/cotizador-web/>
- Palacio, J. (2015). *Gestión de proyectos Scrum Manager*. Scrum Manager.
- Pressman, R. S. (2010). *INGENIERÍA DEL SOFTWARE. Un enfoque práctico*. Madrid (España): Mc Graw Hill.
- Reenskaug, T. (1979, Diciembre 10). MODELS - VIEWS - CONTROLLERS. *Xerox PARC technical note*.
- Romsaiyud, W. (2014). Applying MVC Data Model on Hadoop for Delivering the Business Intelligence. *International Conference on ICT and Knowledge Engineering*.
- Sánchez Flores, V. (2013). *Diseño e implementación de la interfaz de usuario de un sistema informático para la administración de documentos de sistemas de calidad de una empresa fabricante de filtros de papel MSC Tesis*. Apizaco: Instituto Tecnológico de Apizaco.
- Sommerville, I. (2011). *Ingeniería de Software*. México: PEARSON EDUCACIÓN.
- Switzerland, A. M. (2015). *Managing Agile Strategy, Implementation, Organisation*. New York: Springer Cham Heidelberg. doi:10.1007/978-3-319-16262-1
- Syed, A. F., & Ho-Jin, C. (2007). Software Reverse Engineering to Requirements. *International Conference on Convergence Information Technology*.
- Todowebhosting.com. (2016, Septiembre 13). *Todowebhosting.com*. Retrieved from Todowebhosting.com: <http://todowebhosting.com/cotizador-de-paginas-web/>
- TutorialsPoint. (2017, Enero 23). *TutorialsPoint*. Retrieved from TutorialsPoint: http://www.tutorialspoint.com/es/software_engineering/software_development_life_cycle.htm
- Umi, S., Jauari, A. N., & Masfu, H. (2015). Implementing Singleton method in Design of MVCBased PHP Framework. *International Electronics Symposium*.

Wikipedia. (2016). *Cotización*. <https://es.wikipedia.org/wiki/Cotizaci%C3%B3n>. Retrieved Mayo 23, 2016, from <https://es.wikipedia.org/wiki/Cotizaci%C3%B3n>

Wood, S., Michaelides, G., & Thomson, C. (2012). Successful extreme programming: Fidelity to the methodology or good teamworking? *SciVerse ScienceDirect*.

Anexos

A. Plan de pruebas

Introducción

El siguiente documento pretende guiar el proceso de pruebas para el sistema de cotizador.

Dentro de esta fase pueden desarrollarse varios tipos de pruebas. Se comprenden pruebas funcionales, pruebas de usabilidad, pruebas de compatibilidad, pruebas de seguridad, etc. son pruebas que nos ayudaran a reducir el mayor número de errores.

Una estrategia de prueba integra técnicas de casos de prueba en una serie de pasos planeados y bien definidos que llevan a la construcción exitosa del software. Se describen los pasos que hay que llevar a cabo como parte de la prueba, cuándo se deben planificar y realizar estos pasos, y cuántos recursos se requieren.

Objetivo

Comprobar si el sistema cumple sus requisitos específicos que el cliente espera.

Alcance

Se pretende garantizar y controlar la calidad durante el proceso del desarrollo del sistema.

Pruebas propuestas para el sistema cotizador

Para las pruebas utilizaremos y llevaremos a cabo el modelo en espiral el cual avanza a lo largo del modelo de adentro hacia afuera pasando por las distintas unidades hasta completar su proceso.

- Pruebas de Unidad: En estas pruebas se realizarán los casos de prueba de uso, en donde se tiene como objetivo localizar defectos y probar el funcionamiento del software.
- Pruebas de Integración: En estas pruebas se realizarán reportes de la integración de los distintos componentes, para validar el flujo de la información entre los módulos y descubrir si tiene errores en la asociación con la interfaz.
- Pruebas de Validación: En esta se verifica que todos los elementos cumplen con todos los requerimientos funcionales y no funcionales.
- Pruebas de Sistema: En esta prueban se verifica que los elementos interactúen entre de forma correcta en todos los componentes del sistema, en el cual se presenta un reporte validando cada módulo.

Estructura para casos de prueba

- Nombre del caso de prueba. Nombre significativo del caso de prueba que permita identificar el propósito de la prueba.
- Identificador de caso de prueba (Id). Identificador único del caso de prueba.
- Propósito. Define el propósito del caso de prueba.
- Prerrequisito. Define las necesidades para la ejecución del componente.
- Entrada, Definir los datos necesarios para la ejecución del componente.
- Pasos. Acciones del usuario en el sistema para realizar la tarea.
- Criterio de éxito. Permiten determinar que el caso de éxito ha sido exitoso.
- Criterio de falla. Identifica los criterios por el cual el caso de prueba es fallido.

- Módulos asociados. Componente que será ejecutado una vez al ejecutar el componente.
- Autor. Nombre de quien elaboro el caso de prueba.
- Fecha. Fecha en la que se diseña el caso de prueba.

Reporte de incidencia de los casos de prueba y pruebas de integración

Reporte de incidencias

| Caso de prueba | Falla | Solución | Estatus |
|--|--|--|-------------|
| Administración de usuarios CP02 | Se tiene un problema al realizar la inserción en un registro, ya no los guarda en la base de datos. | Se realiza seguimiento al comportamiento de código línea por línea, se encuentra una sesión abierta que se ocupa para listar los registros. Se cierra la sesión, se realiza nuevamente la inserción y registra los datos correctamente. | Solucionado |
| Administración de madera: CP06 | El botón para agregar un registro nuevo no dice la palabra homologada "Registrar". | Se le indica a la persona encargada de las vistas para que realice el cambio. Demora un poco este proceso ya que se tienen problemas de comunicación con el encargado de las vistas y demora en entregar los cambios. | Solucionado |
| Nueva Cotización CP11 | El sistema presenta problemas en la interfaz, ya que no los componentes no están colocados de la forma correcta. | Se le pide a la persona encargada de la vista soluciones el error. Demora un poco este proceso ya que se tienen problemas de comunicación con el encargado de las vistas y demora en entregar los cambios. | Solucionado |
| Nueva Cotización CP11 | El sistema presenta problemas al agregar los productos. | Se revisa el flujo de datos para identificar donde está el error. Se pide a la persona de la vista que revise el comportamiento de las ventanas. Se analiza el código, línea por línea para identificar que es lo que no permite agregar lo elementos. Se cambia elementos de la vista ya que estos son los que causan problemas. | Solucionado |


| | | | |
|---|--|---|--------------------|
| <p>Nueva Cotización CP11</p> | <p>Al agregar un producto el sistema está realizando los cálculos incorrectos.</p> | <p>Se valida el flujo de datos, se analizan las operaciones en Excel para confirmar el resultado, se recorre línea por línea el código y se identifica el error, por lo que se corrige actualizando la formula.</p> | <p>Solucionado</p> |
| <p>Nueva Cotización CP11</p> | <p>Al seleccionar el grado de complejidad no actualiza el costo total.</p> | <p>Se analiza el estado de flujo de datos para validar que el proceso es el correcto, por lo que se procede a analizar el código donde encontramos problemas en las peticiones set y get, no se estaba mandando a llamar correctamente el valor para que actualizara el campo y lo mostrara.</p> | <p>Solucionado</p> |
| <p>Nueva Cotización CP11</p> | <p>El sistema no guarda de forma correcta en la base de datos.</p> | <p>Se presenta un problema a la hora que pasa los datos a una lista, ya que el dato es un tipo objeto y requiere un valor entero, se realizan los cambios correspondientes, pero no realiza la inserción.</p> <p>Se verifica y se tiene problemas con la inserción de un campo relacionado, se cambia el tipo que contiene la variable para dar solución, pero sin éxito. Se realizan diferentes cambios, pero el error persiste. Se investiga y no el error los POJOS que creo Hibernate ya que no están generador correctamente y por las anotaciones no permite la inserción. Se eliminan y se generar nuevamente pero ahora seleccionando las características del lenguaje JDK 5.</p> <p>Se procede a realiza la inserción y se insertan de forma correcta en la base de datos, ya no manda error al insertar datos en tablas relacionadas.</p> | <p>Solucionado</p> |
| <p>Administrar Cotizaciones CP12</p> | <p>No muestra la lista de las cotizaciones.</p> | <p>Se manda a revisar la vista, se valida y esta correcta. Se realizan pruebas y se logran visualizar quitando el campo relacionado, se verifica el código para</p> | <p>Solucionado</p> |


| | | | |
|--------------------------------|--|--|-------------|
| | | descartar algún error, todo está correcto, se detecta que para que el sistema no muestra los registro porque Hibernate cerro la conexión, se modifica el tipo LAZY a todas las relaciones muchos a uno. | |
| Reporte de cotizaciones | Se genera un reporte y no se muestra, se genera en el servidor | Se analiza el código, se le realizan los cambios correspondientes pero el sistema lo sigue generando en el servidor y no en el cliente. Se cambia de librería para utilizar la de primeFaces la cual genera el reporte correctamente en la computadora que realiza la petición | Solucionado |
| Orden de producción | Los reportes se generan en el cliente | Se realiza el cambio de librería como se hizo con los reportes de cotizaciones | Solucionado |
| Cotización por fecha | Los reportes se generan en el cliente | Se realiza el cambio de librería como se hizo con los reportes de cotizaciones | Solucionado |

Casos de prueba



| | | | |
|--------------------------|--|------------------|------|
| Nombre | Acceso al sistema | Id prueba | CP01 |
| Propósito | Verificar si el sistema permite el acceso a los diferentes tipos de usuarios almacenados en la base de datos. | | |
| Prerrequisitos | El sistema deberá tener almacenado un usuario de cada tipo. | | |
| Entrada | El sistema validara al usuario de tipo: <ul style="list-style-type: none"> • Administrador • Jefe de área • Operador • Usuario | | |
| Pasos | 1.- Visitar la página del cotizador 2.- Introducir un usuario y contraseña 3.- Clic en "Iniciar Sesión" 4.- Repetir el proceso con diferente usuario | | |
| Criterio de éxito | El sistema permite el acceso exitoso a los distintos usuarios | | |
| Criterio de falla | | | |


| | |
|--------------------------|--|
| Módulos asociados | Ventana de bienvenida donde muestra el nombre del usuario. |
| Autor | Alvaro Cuateta |
| Fecha | 23/01/107 |


| | | | |
|-----------------------|--|------------------|------|
| Nombre | Administración de Usuarios | Id prueba | CP02 |
| Propósito | Verificar que el sistema pueda agregar un tipo de usuarios, modificar los usuarios existentes y eliminar usuarios. | | |
| Prerrequisitos | Tener acceso al sistema como usuario administrador o jefe de área. | | |
| Entrada | <p>El usuario deberá introducir:</p> <ul style="list-style-type: none"> • Nombre del usuario • Apellido paterno • Apellido materno • Puesto • Usuario • Contraseña • Privilegio | | |
| Pasos | <p>Agregar</p> <ol style="list-style-type: none"> 1. Acceder al sistema como administrador o jefe de área. 2. Llenar los campos solicitados 3. Clic en el botón “Registrar” 4. Visualizar el registro en la página. <p>Modificar</p> <ol style="list-style-type: none"> 1. Acceder al sistema como administrador o jefe de área. 2. Clic en el botón  3. Modificar los campos requeridos 4. Clic en el botón “Actualizar” 5. Los registros se muestran con la información actualizada <p>Eliminar</p> | | |



| | |
|--------------------------|--|
| | <ol style="list-style-type: none"> 1. Acceder al sistema como administrador o jefe de área. 2. Clic en el botón  3. Confirmar la eliminación del registro 4. Se actualiza la tabla sin el registro. |
| Criterio de éxito | El sistema permite agregar, modificar y eliminar de forma establecida sin ningún problema. |
| Criterio de falla | No registra en la base de datos. |
| Módulos asociados | Sin módulos asociados |
| Autor | Alvaro Cuateta |
| Fecha | 23/01/107 |

| | | | |
|-----------------------|---|------------------|------|
| Nombre | Administración de Clientes | Id prueba | CP03 |
| Propósito | Verificar que el sistema pueda agregar un tipo de cliente, modificar los clientes existentes y eliminar clientes. | | |
| Prerrequisitos | Tener acceso al sistema como usuario administrador o jefe de área. | | |
| Entrada | <p>El usuario deberá introducir:</p> <ul style="list-style-type: none"> • Nombre de la empresa • Nombre del contacto • Apellido paterno • Apellido materno • RFC • Calle • Número • Localidad • Estado • C.P. • Teléfono • Celular • Email | | |
| Pasos | <p>Agregar</p> <ol style="list-style-type: none"> 1. Acceder al sistema como administrador o jefe de área. 2. Llenar los campos solicitados 3. Clic en el botón "Registrar" | | |



| | |
|--------------------------|---|
| | <p>4. Visualizar el registro en la página.</p> <p>Modificar</p> <ol style="list-style-type: none"> 1. Acceder al sistema como administrador o jefe de área. 2. Clic en el botón  3. Modificar los campos requeridos 4. Clic en el botón “Actualizar” 5. Los registros se muestran con la información actualizada <p>Eliminar</p> <ol style="list-style-type: none"> 1. Acceder al sistema como administrador o jefe de área. 2. Clic en el botón  3. Confirmar la eliminación del registro 4. Se actualiza la tabla sin el registro. |
| Criterio de éxito | El sistema permite agregar, modificar y eliminar de forma establecida sin ningún problema. |
| Criterio de falla | |
| Módulos asociados | Sin módulos asociados |
| Autor | Alvaro Cuateta |
| Fecha | 23/01/107 |

| | | | |
|-----------------------|--|------------------|------|
| Nombre | Administración de Productos | Id prueba | CP04 |
| Propósito | Verificar que el sistema pueda agregar un tipo de producto, modificar productos existentes y eliminar los productos. | | |
| Prerrequisitos | Tener acceso al sistema como usuario administrador o jefe de área. | | |
| Entrada | El usuario deberá introducir: <ul style="list-style-type: none"> • Nombre del producto | | |
| Pasos | <p>Agregar</p> <ol style="list-style-type: none"> 1. Acceder al sistema como administrador o jefe de área. 2. Llenar los campos solicitados 3. Clic en el botón “Registrar” 4. Visualizar el registro en la página. <p>Modificar</p> <ol style="list-style-type: none"> 1. Acceder al sistema como administrador o jefe de área. 2. Clic en el botón  3. Modificar los campos requeridos 4. Clic en el botón “Actualizar” 5. Los registros se muestran con la información actualizada <p>Eliminar</p> | | |



| | |
|--------------------------|--|
| | <ol style="list-style-type: none"> 1. Acceder al sistema como administrador o jefe de área. 2. Clic en el botón  3. Confirmar la eliminación del registro 4. Se actualiza la tabla sin el registro. |
| Criterio de éxito | El sistema permite agregar, modificar y eliminar de forma establecida sin ningún problema. |
| Criterio de falla | |
| Módulos asociados | Sin módulos asociados |
| Autor | Alvaro Cuateta |
| Fecha | 23/01/107 |

| | | | |
|-----------------------|---|------------------|------|
| Nombre | Administración de Materiales | Id prueba | CP05 |
| Propósito | Verificar que el sistema pueda agregar un material, modificar materiales existentes y eliminar los materiales. | | |
| Prerrequisitos | Tener acceso al sistema como usuario administrador o jefe de área. | | |
| Entrada | <p>El usuario deberá introducir:</p> <ul style="list-style-type: none"> • Tipo de material • Precio • Unidad de medida. | | |
| Pasos | <p>Agregar</p> <ol style="list-style-type: none"> 1. Acceder al sistema como administrador o jefe de área. 2. Llenar los campos solicitados 3. Clic en el botón “Registrar” 4. Visualizar el registro en la página. <p>Modificar</p> <ol style="list-style-type: none"> 1. Acceder al sistema como administrador o jefe de área. 2. Clic en el botón  3. Modificar los campos requeridos 4. Clic en el botón “Actualizar” 5. Los registros se muestran con la información actualizada <p>Eliminar</p> <ol style="list-style-type: none"> 1. Acceder al sistema como administrador o jefe de área. 2. Clic en el botón  3. Confirmar la eliminación del registro 4. Se actualiza la tabla sin el registro. | | |



| | |
|--------------------------|--|
| Criterio de éxito | El sistema permite agregar, modificar y eliminar de forma establecida sin ningún problema. |
| Criterio de falla | |
| Módulos asociados | Sin módulos asociados |
| Autor | Alvaro Cuateta |
| Fecha | 23/01/107 |

| | | | |
|--------------------------|---|------------------|------|
| Nombre | Administración de Maderas | Id prueba | CP06 |
| Propósito | Verificar que el sistema pueda agregar un tipo de madera, modificar maderas existentes y eliminar las maderas. | | |
| Prerrequisitos | Tener acceso al sistema como usuario administrador o jefe de área. | | |
| Entrada | El usuario deberá introducir: <ul style="list-style-type: none"> • Tipo de madera • Precio | | |
| Pasos | <p>Agregar</p> <ol style="list-style-type: none"> 1. Acceder al sistema como administrador o jefe de área. 2. Llenar los campos solicitados 3. Clic en el botón “Registrar” 4. Visualizar el registro en la página. <p>Modificar</p> <ol style="list-style-type: none"> 1. Acceder al sistema como administrador o jefe de área. 2. Clic en el botón  3. Modificar los campos requeridos 4. Clic en el botón “Actualizar” 5. Los registros se muestran con la información actualizada <p>Eliminar</p> <ol style="list-style-type: none"> 1. Acceder al sistema como administrador o jefe de área. 2. Clic en el botón  3. Confirmar la eliminación del registro 4. Se actualiza la tabla sin el registro. | | |
| Criterio de éxito | El sistema permite agregar, modificar y eliminar de forma establecida sin ningún problema. | | |
| Criterio de falla | Botón para agregar no dice Registrar | | |
| Módulos asociados | Sin módulos asociados | | |



| | |
|--------------|----------------|
| Autor | Alvaro Cuateta |
| Fecha | 23/01/107 |

| | | | |
|--------------------------|---|------------------|------|
| Nombre | Administración de Tratamientos | Id prueba | CP07 |
| Propósito | Verificar que el sistema pueda agregar un tipo de tratamiento, modificar tratamientos existentes y eliminar los tratamientos. | | |
| Prerrequisitos | Tener acceso al sistema como usuario administrador o jefe de área. | | |
| Entrada | El usuario deberá introducir: <ul style="list-style-type: none"> • Tipo de tratamiento • Precio | | |
| Pasos | <p>Agregar</p> <ol style="list-style-type: none"> 1. Acceder al sistema como administrador o jefe de área. 2. Llenar los campos solicitados 3. Clic en el botón “Registrar” 4. Visualizar el registro en la página. <p>Modificar</p> <ol style="list-style-type: none"> 1. Acceder al sistema como administrador o jefe de área. 2. Clic en el botón  3. Modificar los campos requeridos 4. Clic en el botón “Actualizar” 5. Los registros se muestran con la información actualizada <p>Eliminar</p> <ol style="list-style-type: none"> 1. Acceder al sistema como administrador o jefe de área. 2. Clic en el botón  3. Confirmar la eliminación del registro 4. Se actualiza la tabla sin el registro. | | |
| Criterio de éxito | El sistema permite agregar, modificar y eliminar de forma establecida sin ningún problema. | | |
| Criterio de falla | | | |
| Módulos asociados | Sin módulos asociados | | |
| Autor | Alvaro Cuateta | | |
| Fecha | 23/01/107 | | |

| | | | |
|---------------|----------------------------|------------------|------|
| Nombre | Administración de Acabados | Id prueba | CP08 |
|---------------|----------------------------|------------------|------|

| | |
|--------------------------|---|
| Propósito | Verificar que el sistema pueda agregar un tipo de acabado, modificar acabados existentes y eliminar los acabados. |
| Prerrequisitos | Tener acceso al sistema como usuario administrador o jefe de área. |
| Entrada | El usuario deberá introducir: <ul style="list-style-type: none"> • Tipo de acabado • Precio |
| Pasos | <p>Agregar</p> <ol style="list-style-type: none"> 1. Acceder al sistema como administrador o jefe de área. 2. Llenar los campos solicitados 3. Clic en el botón “Registrar” 4. Visualizar el registro en la página. <p>Modificar</p> <ol style="list-style-type: none"> 1. Acceder al sistema como administrador o jefe de área. 2. Clic en el botón  3. Modificar los campos requeridos 4. Clic en el botón “Actualizar” 5. Los registros se muestran con la información actualizada <p>Eliminar</p> <ol style="list-style-type: none"> 1. Acceder al sistema como administrador o jefe de área. 2. Clic en el botón  3. Confirmar la eliminación del registro 4. Se actualiza la tabla sin el registro. |
| Criterio de éxito | El sistema permite agregar, modificar y eliminar de forma establecida sin ningún problema. |
| Criterio de falla | |
| Módulos asociados | Sin módulos asociados |
| Autor | Alvaro Cuateta |
| Fecha | 23/01/107 |

| | | | |
|-----------------------|--|------------------|------|
| Nombre | Administración de Personal | Id prueba | CP09 |
| Propósito | Verificar que el sistema pueda agregar un tipo de personal, modificar el personal existente y eliminar personal. | | |
| Prerrequisitos | Tener acceso al sistema como usuario administrador o jefe de área. | | |


| | |
|--------------------------|--|
| Entrada | El usuario deberá introducir: <ul style="list-style-type: none"> • Tipo de personal • Sueldo |
| Pasos | <p>Agregar</p> <ol style="list-style-type: none"> 1. Acceder al sistema como administrador o jefe de área. 2. Llenar los campos solicitados 3. Clic en el botón “Registrar” 4. Visualizar el registro en la página. <p>Modificar</p> <ol style="list-style-type: none"> 1. Acceder al sistema como administrador o jefe de área. 2. Clic en el botón  3. Modificar los campos requeridos 4. Clic en el botón “Actualizar” 5. Los registros se muestran con la información actualizada <p>Eliminar</p> <ol style="list-style-type: none"> 1. Acceder al sistema como administrador o jefe de área. 2. Clic en el botón  3. Confirmar la eliminación del registro 4. Se actualiza la tabla sin el registro. |
| Criterio de éxito | El sistema permite agregar, modificar y eliminar de forma establecida sin ningún problema. |
| Criterio de falla | |
| Módulos asociados | Sin módulos asociados |
| Autor | Alvaro Cuateta |
| Fecha | 23/01/107 |


| Nombre | Generación de Reportes | Id prueba | CP10 |
|-----------------------|--|-----------|------|
| Propósito | Verificar que la generación de reportes se realice con la información correspondiente a lo requerido por el usuario. | | |
| Prerrequisitos | Tener acceso al sistema. Visitar la ventana de reportes | | |
| Entrada | El usuario deberá introducir: <ul style="list-style-type: none"> • Seleccionar un reporte | | |
| Pasos | <p>Reporte de cotizaciones</p> <ol style="list-style-type: none"> 1. Acceder al sistema y seleccionar en el menú Reportes y cotizaciones. 2. Clic en el botón “PDF” de la cotización seleccionada | | |

| | |
|--------------------------|--|
| | <p>3. Imprimir o guardar el reporte en formato "pdf".</p> <p>Orden de producción</p> <p>4. Acceder al sistema y seleccionar en el menú Reportes y orden de producción</p> <p>5. Introducir un número de folio y clic en el botón buscar.</p> <p>6. Clic en el icono de "PDF" o Excel para descargar en formato pdf o xlsx.</p> <p>7. Imprimir o guardar el reporte.</p> <p>Cotización por fecha</p> <p>8. Acceder al sistema y seleccionar en el menú Reportes y cotización por fecha.</p> <p>9. Introducir el rango de fechas para realizar la búsqueda y seleccionar el estado.</p> <p>10. Clic en el botón buscar para desplegar las cotizaciones.</p> <p>11. Clic en el icono de "PDF" o Excel para descargar en formato pdf o xlsx.</p> <p>12. Imprimir o guardar el reporte.</p> |
| Criterio de éxito | El sistema genero de manera correcta los reportes seleccionados |
| Criterio de falla | |
| Módulos asociados | Cotizaciones |
| Autor | Alvaro Cuateta |
| Fecha | 23/01/107 |

| | | | |
|-----------------------|---|------------------|------|
| Nombre | Nueva Cotización | Id prueba | CP11 |
| Propósito | Verificar que el sistema realice una cotización y la almacene en la BD. | | |
| Prerrequisitos | <p>Tener acceso al sistema.</p> <p>Tener datos de un cliente, madera, acabado, tratamiento</p> | | |
| Entrada | <p>El usuario deberá introducir:</p> <ul style="list-style-type: none"> • Nombre del proyecto • Seleccionar un cliente • Clic en agregar material <ul style="list-style-type: none"> ○ Tipo de producto ○ Madera ○ Acabado ○ Preparación para acabado ○ Aplicación de acabado ○ Tratamiento ○ Transporte | | |

| | |
|--------------------------|---|
| | <ul style="list-style-type: none"> ○ Transporte Fuera ○ Cantidad ○ Ancho ○ Peralte ○ Largo ○ Tabla de 1 o 2 pulgadas ○ Vigas de más de 15 cm |
| Pasos | Nueva Cotización <ol style="list-style-type: none"> 1. Acceder al sistema y seleccionar en el menú Cotizar y después en el botón "Nueva Cotización". 2. Llenar y seleccionar los campos 3. Agregar los materiales deseados 4. Clic en el botón "Guardar" 5. Verificar la cotización se muestre en la lista de cotizaciones. |
| Criterio de éxito | El sistema genero de manera correcta la cotización y la visualizo en la ventana de cotizaciones |
| Criterio de falla | |
| Módulos asociados | Cotizaciones |
| Autor | Alvaro Cuateta |
| Fecha | 23/01/107 |

| | | | |
|-----------------------|---|------------------|------|
| Nombre | Administrar Cotizaciones | Id prueba | CP12 |
| Propósito | Verificar que el sistema modifique y elimine una cotización. | | |
| Prerrequisitos | Tener acceso al sistema. | | |
| Entrada | El usuario deberá introducir: <ul style="list-style-type: none"> • Nombre del proyecto • Seleccionar un cliente • Clic en agregar material • Vigas de más de 15 cm | | |
| Pasos | Modificar <ol style="list-style-type: none"> 1. Acceder al sistema. 2. Clic en el botón  3. Modificar los campos requeridos 4. Clic en el botón "Actualizar" 5. Los registros se muestran con la información actualizada Eliminar | | |

| | |
|--------------------------|---|
| | <ol style="list-style-type: none">1. Acceder al sistema.2. Clic en el botón 3. Confirmar la eliminación de la cotización4. Se actualiza la tabla sin el registro. |
| Criterio de éxito | El sistema genero de manera correcta la cotización y la visualizo en la ventana de cotizaciones |
| Criterio de falla | |
| Módulos asociados | Sin módulos asociados |
| Autor | Alvaro Cuateta |
| Fecha | 23/01/07 |

Pruebas de Integración “Cotizador”

| PRUEBAS DE INTEGRACIÓN | | | | | | |
|-------------------------|--|----------------------|---|--|---------------|---------------------|
| Fecha: | 20/01/2017 | Lugar: | Instituto tecnológico de Apizaco | | | |
| Participantes: | Alvaro Cuateta García | | | | | |
| Componente o subsistema | Descripción de la prueba | Elementos a integrar | Descripción | Resultado | Resultado | Fecha de aprobación |
| Inicio de sesión | Iniciar el sistema cotizador y solicitar usuario y contraseña para acceder | Usuario | Se accede con un usuario administrador, jefe de área, operador y usuario. | Se accede de forma satisfactoria al sistema con los distintos usuarios | Sin defectos. | 12/01/2017 |
| | | Contraseña | | Sin defectos. | | |
| Usuarios | La utilización del botón de registrar, modificar y eliminar. Así como consultar. | Registrar | Almacena la información en la BD | Se almacenan los datos en la BD correctamente | Sin defectos. | 12/01/2017 |
| | | Listar | Lista los datos almacenados en la BD. | Muestra los datos de la BD correctamente. | Sin defectos. | |
| | | Modificar | Modifica los datos almacenados en la BD | Se realiza la modificación de los campos en la BD correctamente. | Sin defectos. | |
| | | Eliminar | Elimina los datos de la BD. | Se elimina el registro de la BD correctamente. | Sin defectos. | |
| Cientes | La utilización del botón de registrar, modificar y eliminar. Así como consultar. | Registrar | Almacena la información en la BD | Se almacenan los datos en la BD correctamente | Sin defectos. | 12/01/2017 |
| | | Listar | Lista los datos almacenados en la BD. | Muestra los datos de la BD correctamente. | Sin defectos. | |
| | | Modificar | Modifica los datos almacenados en la BD | Se realiza la modificación de los | Sin defectos. | |
| | | Eliminar | Elimina los datos de la BD. | campos en la BD correctamente. | Sin defectos. | |

| | | | | | | |
|-------------------|--|-----------|---|--|---------------|------------|
| Productos | La utilización del botón de registrar, modificar y eliminar. Así como consultar. | Registrar | Almacena la información en la BD | Se almacenan los datos en la BD correctamente | Sin defectos. | 12/01/2017 |
| | | Listar | Lista los datos almacenados en la BD. | Muestra los datos de la BD correctamente. | Sin defectos. | |
| | | Modificar | Modifica los datos almacenados en la BD | Se realiza la modificación de los campos en la BD correctamente. | Sin defectos. | |
| | | Eliminar | Elimina los datos de la BD. | Se elimina el registro de la BD correctamente. | Sin defectos. | 12/01/2017 |
| Materiales | La utilización del botón de registrar, modificar y eliminar. Así como consultar. | Registrar | Almacena la información en la BD | Se almacenan los datos en la BD correctamente | Sin defectos. | 12/01/2017 |
| | | Listar | Lista los datos almacenados en la BD. | Muestra los datos de la BD correctamente. | Sin defectos. | |
| | | Modificar | Modifica los datos almacenados en la BD | Se realiza la modificación de los campos en la BD correctamente. | Sin defectos. | |
| | | Eliminar | Elimina los datos de la BD. | Se elimina el registro de la BD correctamente. | Sin defectos. | 12/01/2017 |
| Madera | La utilización del botón de registrar, modificar y eliminar. Así como consultar. | Registrar | Almacena la información en la BD | Se almacenan los datos en la BD correctamente | Sin defectos. | 12/01/2017 |
| | | Listar | Lista los datos almacenados en la BD. | Muestra los datos de la BD correctamente. | Sin defectos. | |
| | | Modificar | Modifica los datos almacenados en la BD | Se realiza la modificación de los campos en la BD correctamente. | Sin defectos. | 12/01/2017 |
| | | Eliminar | Elimina los datos de la BD. | Se elimina el registro de la BD correctamente. | Sin defectos. | 12/01/2017 |

| | | | | | | |
|---------------------|---|-----------|---|--|---------------|------------|
| Tratamientos | La utilización del botón de registrar, modificar y eliminar. Así como consultar. | Registrar | Almacena la información en la BD | Se almacenan los datos en la BD correctamente | Sin defectos. | 12/01/2017 |
| | | Listar | Lista los datos almacenados en la BD. | Muestra los datos de la BD correctamente. | Sin defectos. | |
| | | Modificar | Modifica los datos almacenados en la BD | Se realiza la modificación de los campos en la BD correctamente. | Sin defectos. | |
| | | Eliminar | Elimina los datos de la BD. | Se elimina el registro de la BD correctamente. | Sin defectos. | 12/01/2017 |
| Acabados | La utilización del botón de registrar, modificar y eliminar. Así como consultar. | Registrar | Almacena la información en la BD | Se almacenan los datos en la BD correctamente | Sin defectos. | 12/01/2017 |
| | | Listar | Lista los datos almacenados en la BD. | Muestra los datos de la BD correctamente. | Sin defectos. | |
| | | Modificar | Modifica los datos almacenados en la BD | Se realiza la modificación de los campos en la BD correctamente. | Sin defectos. | |
| | | Eliminar | Elimina los datos de la BD. | Se elimina el registro de la BD correctamente. | Sin defectos. | 12/01/2017 |
| Personal | La utilización del botón de registrar, modificar y eliminar. Así como consultar los registros | Registrar | Almacena la información en la BD | Se almacenan los datos en la BD correctamente | Sin defectos | 12/01/2017 |
| | | Listar | Lista los datos almacenados en la BD. | Muestra los datos de la BD correctamente. | Sin defectos | |
| | | Modificar | Modifica los datos almacenados en la BD | Se realiza la modificación de los campos en la BD correctamente. | Sin defectos | |
| | | Eliminar | Elimina los datos de la BD. | Se elimina el registro de la BD correctamente. | Sin defectos | 12/01/2017 |

| | | | | | | |
|---------------------------------|--|--------------------|--|---|--------------|------------|
| Reportes de cotizaciones | Utilización de botón "PDF" | PDF | Permite generar un reporte en PDF de las cotizaciones. | El reporte se genera correctamente. | Sin defectos | 12/01/2017 |
| | | Listar | Lista las cotizaciones que existen en la BD. | Se listan las cotizaciones correctamente. | Sin defectos | |
| Orden de producción | Búsqueda por folio y generación de archivos PDF | Buscar | Permite buscar el folio en la BD. | Realiza la búsqueda de forma correcta | Sin defectos | |
| | | PDF | Genera un documento PDF de la cotización. | Se genera el documento correctamente. | Sin defectos | |
| Cotización por fecha | Realizar una búsqueda por fecha y estado | Campos de fechas. | Permite buscar cotizaciones en un rango de fechas. | Realiza la búsqueda en el rango de fechas establecidas correctamente. | Sin defectos | |
| | | Opciones de estado | Permite seleccionar el estado de la cotización. | Se muestran los resultados conforme la selección del estado. | Sin defectos | |
| | | PDF | Genera un documento PDF de la cotización. | Se genera el documento correctamente. | Sin defectos | |
| Cotización | La utilización del botón Nueva cotización, modificar y eliminar. | Nueva Cotización | Permite le generación de una cotización | Sin defectos encontrados. | | 12/01/2017 |
| | | Listar | Lista las cotizaciones existentes en la BD | Sin defectos encontrados. | | |
| | | Modificar | Permite la modificación de las cotizaciones almacenadas en la BD | Sin defectos encontrados. | | |
| | | Eliminar | Permite la eliminación de la cotización. | Sin defectos encontrados. | | |
| Nueva cotización | La utilización de los botones, clientes | Cliente | Permite agregar clientes nuevos a la BD | Sin defectos encontrados. | | 12/01/2017 |

| | | | | |
|--------------------------------------|------------------|---|---------------------------|--|
| calcular, agregar material y guardar | Agregar material | Permite agregar materiales a la cotización | Sin defectos encontrados. | |
| | Calcular | Permite realizar el cálculo de la cotización. | Sin defectos encontrados. | |
| | Guardar | Permite guardar los datos en la BD. | Sin defectos encontrados. | |

Pruebas de Sistema “Cotizador”

| N° | Especificación de Requerimientos | Tipo de Prueba | Descripción de Prueba | Resultado Esperado | Resultado |
|----|----------------------------------|------------------|--|--|-----------|
| P1 | Autenticación de usuarios | Prueba de Estrés | Se debe ingresar el nombre de usuario y contraseña, para tener acceso al sistema. | Acceder al sistema como administrador | Correcto |
| P2 | Alta de usuarios | Link | Dar la navegacion correcta para el llenado del formulario. | Link bien direccionado | Correcto |
| P3 | Alta de usuarios | Contenido | Validar que los campos del formulario reciban los tipos de datos correctos correspondientes a la BD. | Los campos no cumplen con validación de contenido. | Correcto |

| | | | | | |
|------------|--------------------------|-----------------|--|---|----------|
| P4 | Alta de usuarios | Funcionabilidad | La función de guardar, realiza la validacion de los datos y los almacena en la BD. | Datos almacenados en la base de datos correctamente. | Correcto |
| P5 | Modificación de usuarios | Link | Dar la navegacion correcta para la mofificacion de informacion del usuario. | Link bien direccionado. | Correcto |
| P6 | Modificación de usuarios | Funcionalidad | La función de modifica, valida y actuliza registros del usuario en la BD. | Modificación de los registros en la BD correctamente. | Correcto |
| P7 | Eliminación de usuarios | Link | Dar la navegacion correcta para eliminar un registro de la tabla. | Link direccionado correctamente. | Correcto |
| P8 | Eliminación de usuarios | Funcionabilidad | La funcion de eliminar, confirma y elimina el registro seleccionado. | Se realiza la validación y realiza la eliminación. | Correcto |
| P9 | Alta clientes | Link | Dar el direccionamiento correcto al formulario agregar cliente. | Link direccionado correctamente. | Correcto |
| P10 | Alta clientes | Funcionabilidad | La funcion de guardar, realiza la vadicacion de los datos y los almacena en la BD. | Datos almacenados correctamente. | Correcto |
| P11 | Alta clientes | Contenido | Validar que los campos del formulario reciban los tipos de datos correctos correspondientes a la BD. | No todos los campos cumplen con tipos de datos. | Correcto |
| P12 | Modificación clientes | Link | Dar el direccionamiento correcto al formulario de modificar clientes. | Link direccionado correctamente. | Correcto |
| P13 | Modificación clientes | Funcionalidad | La funcion actualiza los datos del formulario en la BD. | Registros modificados correctamente. | Correcto |
| P14 | Eliminación clientes | Link | Dar el direccionamiento correcto a la ventana de confirmacion. | No se envía el link a la ventana de confirmación. | Correcto |
| P15 | Eliminación clientes | Funcionalidad | La funcion de eliminar registros de la tabla, | No valida y solo elimina el registro directamente. | Correcto |

| | | | | | |
|------------|------------------------|-----------------|--|--|----------|
| | | | confirma y elimina el registro de la BD. | | |
| P16 | Alta productos | Link | Dar el direccionamiento correcto al formulario de productos. | Link direccionado correctamente. | Correcto |
| P17 | Alta productos | Funcionabilidad | La funcion de guardar productos, realiza la vadicacion de los datos y los almacena en la BD. | Datos almacenados correctamente. | Correcto |
| P18 | Alta productos | Contenido | Validar que los campos del formulario reciban los tipos de datos correctos correspondientes a la BD. | No todos los campos cumplen con tipos de datos. | Correcto |
| P19 | Modificación productos | Link | Dar el direccionamiento correcto al formulario modificar productos. | Link direccionado correctamente. | Correcto |
| P20 | Modificación productos | Funcionalidad | La funcion actualiza los datos del formulario en la BD. | Registros modificados correctamente. | Correcto |
| P21 | Eliminación productos | Link | Dar el direccionamiento correcto a la ventana de confirmacion. | No se envía el link a la ventana de confirmación. | Correcto |
| P22 | Eliminación productos | Funcionalidad | La funcion de eliminar registros de la tabla, confirma y elimina el registro de la BD. | No valida y solo elimina el registro directamente. | Correcto |
| P23 | Alta materiales | Link | Dar el direccionamiento correcto al formulario de materiales. | Link direccionado correctamente. | Correcto |
| P24 | Alta materiales | Funcionabilidad | La funcion de guardar materiales, realiza la vadicacion de los datos y los almacena en la BD. | Datos almacenados correctamente. | Correcto |
| P25 | Alta materiales | Contenido | Validar que los campos del formulario reciban los tipos de datos correctos correspondientes a la BD. | No todos los campos cumplen con tipos de datos. | Correcto |

| | | | | | |
|------------|-------------------------|-----------------|--|--|----------|
| P26 | Modificación materiales | Link | Dar el direccionamiento correcto al formulario modificar materiales. | Link direccionado correctamente. | Correcto |
| P27 | Modificación materiales | Funcionalidad | La funcion actualiza los datos del formulario en la BD. | Registros modificados correctamente. | Correcto |
| P28 | Eliminación materiales | Link | Dar el direccionamiento correcto a la ventana de confirmacion. | No se envía el link a la ventana de confirmación. | Correcto |
| P29 | Eliminación materiales | Funcionalidad | La funcion de eliminar registros de la tabla, confirma y elimina el registro de la BD. | No valida y solo elimina el registro directamente. | Correcto |
| P30 | Alta madera | Link | Dar el direccionamiento correcto al formulario de madera. | Link direccionado correctamente. | Correcto |
| P31 | Alta madera | Funcionabilidad | La funcion de guardar madera, realiza la vadicacion de los datos y los almacena en la BD. | Datos almacenados correctamente. | Correcto |
| P32 | Alta madera | Contenido | Validar que los campos del formulario reciban los tipos de datos correctos correspondientes a la BD. | No todos los campos cumplen con tipos de datos. | Correcto |
| P33 | Modificación madera | Link | Dar el direccionamiento correcto al formulario modificar madera. | Link direccionado correctamente. | Correcto |
| P34 | Modificación madera | Funcionalidad | La funcion actualiza los datos del formulario en la BD. | Registros modificados correctamente. | Correcto |
| P35 | Eliminación madera | Link | Dar el direccionamiento correcto a la ventana de confirmacion. | No se envía el link a la ventana de confirmación. | Correcto |
| P36 | Eliminación madera | Funcionalidad | La funcion de eliminar registros de la tabla, confirma y elimina el registro de la BD. | Se realiza la validación y realiza la eliminación. | Correcto |

| | | | | | |
|------------|--------------------------|-----------------|--|--|----------|
| P37 | Alta tratamiento | Link | Dar el direccionamiento correcto al formulario de tratamiento. | Link direccionado correctamente. | Correcto |
| P38 | Alta tratamiento | Funcionabilidad | La funcion de guardar tratamiento, realiza la vadicacion de los datos y los almacena en la BD. | Datos almacenados correctamente. | Correcto |
| P39 | Alta tratamiento | Contenido | Validar que los campos del formulario reciban los tipos de datos correctos correspondientes a la BD. | No todos los campos cumplen con tipos de datos. | Correcto |
| P40 | Modificación tratamiento | Link | Dar el direccionamiento correcto al formulario modificar tratamiento. | Link direccionado correctamente. | Correcto |
| P41 | Modificación tratamiento | Funcionalidad | La funcion actualiza los datos del formulario en la BD. | Registros modificados correctamente. | Correcto |
| P42 | Eliminación tratamiento | Link | Dar el direccionamiento correcto a la ventana de confirmacion. | No se envía el link a la ventana de confirmación. | Correcto |
| P43 | Eliminación tratamiento | Funcionalidad | La funcion de eliminar registros de la tabla, confirma y elimina el registro de la BD. | Se realiza la validación y realiza la eliminación. | Correcto |
| P44 | Alta acabado | Link | Dar el direccionamiento correcto al formulario de acabado. | Link direccionado correctamente. | Correcto |
| P45 | Alta acabado | Funcionabilidad | La funcion de guardar acabado, realiza la vadicacion de los datos y los almacena en la BD. | Datos almacenados correctamente. | Correcto |
| P46 | Alta acabado | Contenido | Validar que los campos del formulario reciban los tipos de datos correctos correspondientes a la BD. | No todos los campos cumplen con tipos de datos. | Correcto |
| P47 | Modificación acabado | Link | Dar el direccionamiento correcto al formulario modificar acabado. | Link direccionado correctamente. | Correcto |

| | | | | | |
|------------|-----------------------|-----------------|--|--|----------|
| P48 | Modificación acabado | Funcionalidad | La funcion actualiza los datos del formulario en la BD. | Registros modificados correctamente. | Correcto |
| P49 | Eliminación acabado | Link | Dar el direccionamiento correcto a la ventana de confirmacion. | No se envía el link a la ventana de confirmación. | Correcto |
| P50 | Eliminación acabado | Funcionalidad | La funcion de eliminar registros de la tabla, confirma y elimina el registro de la BD. | Se realiza la validación y realiza la eliminación. | Correcto |
| P51 | Alta personal | Link | Dar el direccionamiento correcto al formulario de personal. | Link direccionado correctamente. | Correcto |
| P52 | Alta personal | Funcionabilidad | La funcion de guardar personal, realiza la vadicacion de los datos y los almacena en la BD. | Datos almacenados correctamente. | Correcto |
| P53 | Alta personal | Contenido | Validar que los campos del formulario reciban los tipos de datos correctos correspondientes a la BD. | No todos los campos cumplen con tipos de datos. | Correcto |
| P54 | Modificación personal | Link | Dar el direccionamiento correcto al formulario modificar personal. | Link direccionado correctamente. | Correcto |
| P55 | Modificación personal | Funcionalidad | La funcion actualiza los datos del formulario en la BD. | Registros modificados correctamente. | Correcto |
| P56 | Eliminación personal | Link | Dar el direccionamiento correcto a la ventana de confirmacion. | No se envía el link a la ventana de confirmación. | Correcto |
| P57 | Eliminación personal | Funcionalidad | La funcion de eliminar registros de la tabla, confirma y elimina el registro de la BD. | Se realiza la validación y realiza la eliminación. | Correcto |
| P58 | Alta cotización | Link | Dar el direccionamiento correcto al formulario de cotizar. | Link direccionado correctamente. | Correcto |
| P59 | Alta cotización | Contenido | Validar que los campos del formulario reciban los tipos | Datos ingresados correcto. | Correcto |

| | | | | | |
|------------|---------------------------------|---------------|---|---|----------|
| | | | de datos correctos correspondientes a la BD. | | |
| P60 | Alta cotización | Funcionalidad | La función de guardar madera, realiza la validación de los datos y los almacena en la BD. | No se realiza el guardado de registros en la base de datos. | Correcto |
| P61 | Modificación cotización | Link | Dar el direccionamiento correcto al formulario de modificación | No se encuentra interfaz de modificación | Correcto |
| P62 | Modificación cotización | Contenido | Validar que los campos modificados sean actualizados correctamente. | No se encuentra formulario. | Correcto |
| P63 | Modificación cotización | Funcionalidad | La función de modificar la cotización, realiza el cambio de los datos en la BD | No se encuentra la función. | Correcto |
| P64 | Eliminar cotización | Link | Dar el direccionamiento correcto a la ventana de confirmación. | No hay interfaz de eliminación | Correcto |
| P65 | Eliminar cotización | Funcionalidad | La función de eliminar registros de la tabla, confirma y elimina el registro de la BD. | No se encuentra la interfaz. | Correcto |
| P66 | Generación de PDF de cotización | Link | Direccionar a la generación del pdf con la cotización realizada. | Se direcciona correctamente. | Correcto |
| P67 | Generación de PDF de cotización | Funcionalidad | La función realiza la generación de pdf de la cotización realizada. | Se genera el pdf de la cotización adecuadamente. | Correcto |
| P68 | Reportes de cotizaciones | Link | Dar el direccionamiento correcto a la interfaz de reportes. | Da el direccionamiento correcto | Correcto |
| P69 | Reportes de cotizaciones | Funcionalidad | La función de generar reportes de las cotizaciones existentes. | La función cumple con el reporte de cotización. | Correcto |
| P70 | Respaldo de BD | Link | Dar el direccionamiento correcto a la interfaz de respaldo. | Da el direccionamiento correcto. | Correcto |
| P71 | Respaldo de BD | Funcionalidad | La función de respaldo realiza la conexión a la BD y | Respaldo correcto. | Correcto |

| | | | | | |
|--|--|--|----------------------------------|--|--|
| | | | realiza el respaldo de la misma. | | |
|--|--|--|----------------------------------|--|--|

B. Carta de liberación de la estancia y carta de satisfacción del cliente donde avala el trabajo realizado



METAL Y MADERA CONSTRUCTIVOS J&M, S. DE R.L. DE C.V.
R.F.C. MMC 0312186J0

Tlaxcala, Tlaxcala a 09 de Enero del 2017

Asunto: Liberación de estancias

MTRO. FELIPE PASCUAL ROSARIO AGUIRRE
DIRECTOR
INSTITUTO TECNOLÓGICO DE APIZACO
PRESENTE

A través de este medio informo a usted que el C Lic. Álvaro Cuateta García alumno de la Maestría en Sistemas Computacionales con No de control M06370497 realizo sus estancias técnicas con el objetivo de "Desarrollar bajo el modelo MVC el modulo controlador de una aplicación cotizador que permita la integración del modelo y la vista, para cotizar sus productos y servicios" durante el periodo comprendido del 02 Febrero del 2016 a 06 de Enero del 2017.

Sin más por el momento reciba un cordial saludo

ATENTAMENTE


NOE ESPINO TELLEZ
ADMINISTRACION Y FINANZAS


Metal y Madera

LARDIZABAL No. 17 DPTO 4
COL. CENTRO
TLAXCALA, TLAXCALA, MEXICO C.P.90000
TEL: 52 (246) 46 20214
Email: espino@metalymadera.mx
www.metalymadera.mx



METAL Y MADERA CONSTRUCTIVOS J&M, S. DE R.L. DE C.V.
R.F.C. MMC 0312186J0

Tlaxcala, Tlaxcala a 09 de Enero del 2017

Asunto: Liberación de estancias

MTRO. FELIPE PASCUAL ROSARIO AGUIRRE
DIRECTOR
INSTITUTO TECNOLÓGICO DE APIZACO
PRESENTE

A través de este medio informo a usted que el C Lic. Álvaro Cuateta García alumno de la Maestría en Sistemas Computacionales con No de control M06370497 realizo sus estancias técnicas con el objetivo de "Desarrollar bajo el modelo MVC el modulo controlador de una aplicación cotizador que permita la integración del modelo y la vista, para cotizar sus productos y servicios" durante el periodo comprendido del 02 Febrero del 2016 a 06 de Enero del 2017.

Sin más por el momento reciba un cordial saludo

ATENTAMENTE


NOE ESPINO TELLEZ
ADMINISTRACION Y FINANZAS

Metal y Madera

LARDIZABAL No. 17 DPTO 4
COL. CENTRO
TLAXCALA, TLAXCALA, MEXICO C.P.90000
TEL: 52 (246) 46 20214
Email: espino@metalmadera.mx
www.metalmadera.mx

C. Publicaciones de artículos



DESARROLLO DE UNA APLICACIÓN UTILIZANDO METODOLOGÍAS ÁGILES, PARA LA COTIZACIÓN DE SERVICIOS EN UNA EMPRESA FABRICADORAS DE VIGAS DE MADERA LAMINADA

Lic. Alvaro Cuateta García¹, M. en C. María Guadalupe Medina Barrera²,
M. en C. José Juan Hernández Mora³ y M. en C. María Janai Sánchez Hernández⁴

Resumen—En este artículo se presenta la metodología para la implementación de un sistema que será capaz de realizar las cotizaciones a la medida de proyectos que fluyen dentro de una empresa fabricadora de vigas de madera laminada. Siguiendo la metodología de desarrollo ágil Scrum y bajo el patrón de diseño modelo vista controlador (MVC) se identifica la necesidad de desarrollar la parte controlador del patrón de diseño MVC, además de guiar el diseño y desarrollo del controlador bajo la metodología propuesta por Scrum para obtener el mejor resultado del proyecto. El controlador se sincroniza con la base de datos, donde se guardaran los registros necesarios para realizar la cotización correspondiente, además de que interactuara con el módulo de la interfaz web, para realizar las operaciones desde cualquier sitio con acceso a internet.

Palabras clave—Metodologías Ágiles SCRUM, Modelo Vista Controlador (MVC).

Introducción

La empresa fabricadora de vigas de madera laminada tiene entre sus actividades el proceso de construcción y manufactura de piezas de madera, al igual se realizan proyectos de construcción para diferentes empresas en distintos estados del país tales como Alemania, Italia entre otros.

Dicha empresa requiere automatizar la realización de cotizaciones para sus clientes, por lo que se propone la creación de un sistema informático, que le permita emitir cotizaciones de la forma rápida y efectiva de sus clientes.

En la actualidad utiliza una hoja de cálculo, sin embargo se requiere mejorar el proceso, ya que los encargados de convencer al cliente, requieren un sistema en donde se apoyan para llevar a cabo la dicha acción.

Bajo el diseño de un proyecto tecnológico, se identifica la necesidad de desarrollar la parte controlador del patrón de diseño MVC, para la aplicación que será capaz de realizar las cotizaciones a la medida de proyectos que fluyen dentro del negocio de vigas de madera laminada.

Complementando con la competencia que existe en el mercado, se tiene la necesidad de desarrollar un sistema que sea rápido eficiente y eficaz y así la empresa cuente con una ventaja para aumentar su productividad y satisfacer las necesidades de sus clientes en el menor tiempo posible. Para el desarrollo se requieren metodologías que garanticen la fiabilidad del software y las metodologías ágiles han ganado bastante popularidad desde hace algunos años, en la ingeniería del software juega un papel muy importante en el desarrollo, portabilidad, mantenibilidad, funcionalidad, fiabilidad y productividad del software. (Urquiza Yllescas J. F., 2010)

Descripción del Método

Definición del problema

El desarrollo de la parte controlador que cubrirá las necesidades para garantizar la fiabilidad entre los módulos, agilizará la cotización a la medida de la empresa, mediante el modelo MVC (Modelo Vista Controlador) y Scrum.

Metodología propuesta

Con el patrón de diseño MVC se propone la construcción de tres componentes y dividir el desarrollo del sistema y establecen las etapas para el desarrollo de cada módulo con Scrum.

Modelo-Vista-Controlador

El modelo vista controlador separa la presentación e interacción de los datos del sistema. El sistema se estructura en tres componentes lógicos que interactúan entre sí.

¹ Lic. Alvaro Cuateta García Egresado del Instituto Tecnológico de Apizaco, estudiante de la Maestría en Sistemas Computacionales del mismo. acuateta@gmail.com (autor correspondiente)

² M. en C. María Guadalupe Medina Barrera es profesora de la Maestría en Sistemas Computacionales del Instituto Tecnológico de Apizaco.

³ M. en C. José Juan Hernández Mora es profesor de la Maestría en Sistemas Computacionales del Instituto Tecnológico de Apizaco

⁴ M. en C. María Janai Sánchez Hernández es profesora de la Maestría en Sistemas Computacionales del Instituto Tecnológico de Apizaco.

El componente "Modelo" maneja los datos del sistema y las operaciones asociadas a esos datos.
El componente "vista" define y gestiona cómo se presentan los datos al usuario.
El componente "controlador" dirige la interacción del usuario (por ejemplo, teclas oprimidas, clics del mouse, etcétera) y pasa estas interacciones a los componentes "Vista" y "Modelo". (Somerville, 2011)

La figura 1 representa el proceso de trabajo de este modelo: (Gomez, 2014)

- El Administrador solicita una acción al servidor apache
- El servidor atiende la petición y manda llamar al JSF correspondiente
- El JSF llama a Hibernate necesario
- El modelo Hibernate atiende la petición y realiza las operaciones con Mysql correspondientes
- El modelo devuelve el resultado
- El JSF llama a las vistas primefaces, enviándole los datos procesados de los JSF
- La vista presenta los datos
- El controlador devuelve la vista al servidor
- El servidor presenta el resultado al cliente



Figura 1. Modelo Vista Controlador.

Métodos ágiles

Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida para desarrollar software de manera rápida. (Beck K, 2013)

La metodología Scrum

Se ha elegido esta metodología por que el controlador requiere cambios constantes, se basa en sprints orientados a la entrega del producto que se van completando mediante iteraciones cortas, ayudando a planificar, controlar y evaluar el desarrollo permitiendo que los JSP sean adaptable a los cambios.

El equipo scrum propone el marco técnico como son los roles y consta de 5 personas (2 profesores y 3 testistas de Maestría) para el desarrollo e implementación del sistema.

El dueño del producto que representa a los clientes como son el gerente y la persona de ventas.

El scrum Master equivalente al director de proyecto, el cual mantiene los procesos y trabaja junto con el jefe del proyecto.

Sprint: se denomina sprint a cada ciclo o iteración de trabajo que produce una parte del producto terminada y funcionalmente operativa (incremento). Son de dos a cuatro semanas y durante cada reunión, se revisará el avance del sprint determinando las actividades en el siguiente sprint. (Manager, Abril 2015)

Pila del producto. Se especificaron los requisitos funcionales y no funcionales, mejoras y correcciones que constituyen los cambios que harán al proyecto.

Pila del sprint: lista de los trabajos que debe realizar el equipo durante el sprint para generar el incremento previsto. La pila del sprint está conformado por las tareas de mayor prioridad para obtener el producto requerido.

Incremento: resultado de cada sprint.

En la Figura 2 se muestra el diagrama del ciclo iterativo scrum del proyecto

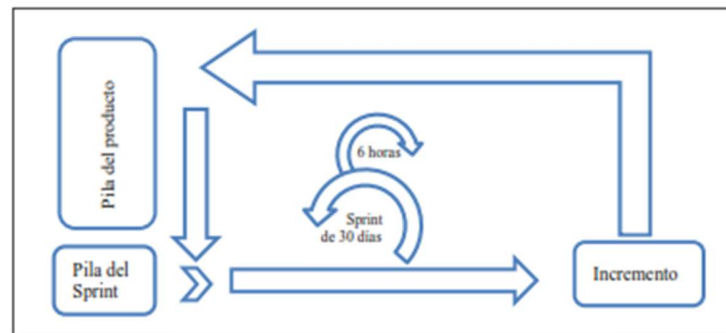


Figura 2. Diagrama del ciclo iterativo Scrum

Aplicación del Modelo

Definición de roles en el desarrollo del sistema

En la figura 3 se muestra el modelado del sistema en general, incluyendo los actores y privilegios que permiten el acceso correspondiente.

Adicionalmente se presentan del uso de las operaciones que tiene el sistema.

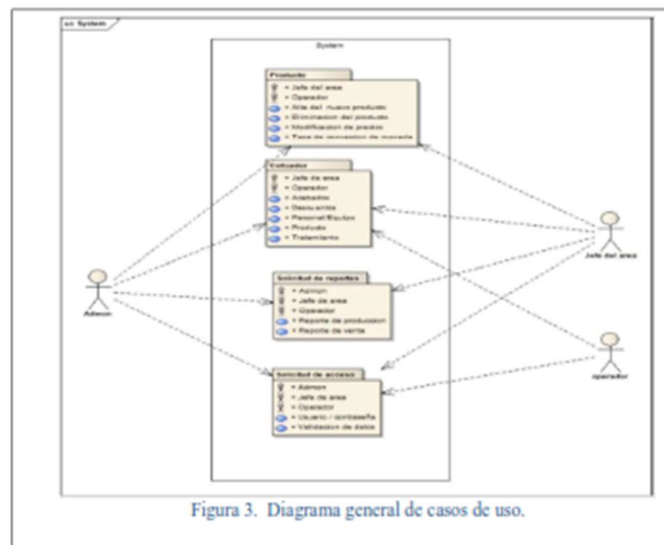


Figura 3. Diagrama general de casos de uso.

En la figura 4 se presenta el modelo entidad relación de la base de datos siendo el resultado del segundo sprint, correspondiente a la continuación del primer sprint y siguiendo el ciclo iterativo Scrum.



EL TECNOLÓGICO NACIONAL DE MÉXICO
Y EL INSTITUTO TECNOLÓGICO DE APIZACO

OTORGAN LA PRESENTE

CONSTANCIA

A

ÁLVARO CUATETA GARCÍA

POR OBTENER EL PRIMER LUGAR
CON EL PROYECTO
"SISTEMA COTIZADOR DE MADERA INDUSTRIAL"
EN LA CATEGORÍA III. SISTEMAS DE INFORMACIÓN
EN EL MARCO DEL SEMINARIO DE TECNOLOGÍAS DE LA
INFORMACIÓN Y COMUNICACIONES 2017
DE ESTA INSTITUCIÓN

APIZACO, TLAX., MAYO 24 DE 2017



SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL
DE MÉXICO
INSTITUTO TECNOLÓGICO DE APIZACO

DIRECCIÓN

MTRO. FELIPE PASCUAL ROSARIO AGUIRRE
DIRECTOR DEL IT DE APIZACO

