UNIVERSIDAD TÉCNICA DEL NORTE

Facultad de Ingeniería en Ciencias Aplicadas

Carrera de Ingeniería en Sistemas Computacionales

DESARROLLO DEL MÓDULO DE PLANTILLAS PARA EL SISTEMA INTEGRADO DE ACTIVIDAD DOCENTE (SIAD) DE LA CARRERA DE SOFTWARE DE LA UNIVERSIDAD TÉCNICA DEL NORTE, APLICANDO EL ESTÁNDAR ISO/IEC 25010 PARA MANTENIBILIDAD

Trabajo de grado previo a la obtención del título de Ingeniera en Sistemas Computacionales

Autor:

Patricia Stefania Morillo Montenegro

Director:

MSc. Xavier Mauricio Rea Peñafiel

Ibarra – Ecuador

2021

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN
A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información.

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	0401722459		
APELLIDOS Y NOMBRES:	MORILLO MONTENEGRO PATRICIA STEFANIA		
DIRECCIÓN:	TULCÁN (FLORESTA Y CARABOBO)		
EMAIL:	psmorillom@utn.edu.ec		
TELÉFONO FIJO:	(062) 240-076	**TELÉFONO MÓVIL:**	0988090889

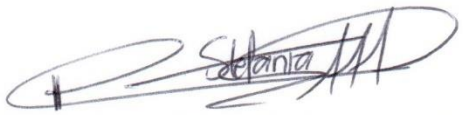
DATOS DE LA OBRA	
TÍTULO:	Desarrollo del módulo de plantillas para el sistema integrado de actividad docente (SIAD) de la carrera de software de la universidad técnica del norte, aplicando el estándar ISO/IEC 25010 para mantenibilidad.
AUTOR:	MORILLO MONTENEGRO PATRICIA STEFANIA
FECHA:	20 de agosto del 2021
PROGRAMA:	PREGRADO
TÍTULO POR EL QUE OPTA:	INGENIERA EN SISTEMAS COMPUTACIONALES
DIRECTOR:	MSC. REA PEÑAFIEL XAVIER MAURICIO
ASESOR:	MSC. QUIÑA MERA JOSÉ ANTONIO
ASESOR:	MSC. GRANDA GUDIÑO PEDRO DAVID

2. CONSTANCIAS

La autora manifiesta que la obra objeto de la presente autorización es original y fue desarrollada sin violar derechos de autor de terceros, por lo tanto, es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 11 días del mes de noviembre del 2021.

LA AUTORA:

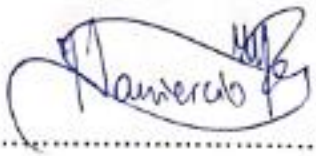


.....

Patricia Stefania Morillo Montenegro

CERTIFICACIÓN DEL DIRECTOR DE TESIS

En mi calidad de tutor del trabajo de grado presentado por la egresada **PATRICIA STEFANIA MORILLO MONTENEGRO,** titulado "DESARROLLO DEL MÓDULO DE PLANTILLAS PARA EL SISTEMA INTEGRADO DE ACTIVIDAD DOCENTE (SIAD) DE LA CARRERA DE SOFTWARE DE LA UNIVERSIDAD TÉCNICA DEL NORTE, APLICANDO EL ESTÁNDAR ISO/IEC 25010 PARA MANTENIBILIDAD", certifico que reúne los requisitos y méritos para ser presentado en defensa pública frente al tribunal asignado.



.....

ING. MAURICIO REA, MSC.

Dedicatoria

A Clarita, quien además de una grandiosa mujer y madre, ha sido mi amiga, mi refugio, mi paz y la razón por la que creo que el amor incondicional en las personas, existe.

A Carlos, mi padre, ya que muchos tenemos la fortuna de una buena madre, pero no tantos el privilegio de un buen papá y el mío es uno admirable, que ha estado a mi lado como mi guía, defensor y consejero.

Papitos, ustedes son mi fuerza y mi mayor ejemplo, todo lo que soy y he alcanzado hasta ahora, ha requerido no solo mi esfuerzo, sino también el de ustedes y en mayor proporción. Nada de esto habría sido posible sin su apoyo, su dedicación y cada sacrificio hecho en favor de ayudarme a construir mi futuro. Este es un sueño compartido y el logro es tan mío como suyo, por eso se lo dedico principalmente a los dos.

Quiero dedicar también este trabajo:

A Johana, porque gran parte de mi motivación nació de querer ser un buen referente para ella y aunque es mi hermanita menor, ahora es de cierta forma quien me da ejemplo. Tú fuiste quien vivió más de cerca este proceso junto a mí y lo que implicó, te dedico esto como una muestra de que todo lo que pasamos juntas, por difícil que fuera, valió la pena y el esfuerzo.

A Fernando, quien siempre ha estado pendiente de mí y de cómo me puede ayudar. Tu aporte ha sido muy significativo a lo largo de este camino, con tu apoyo me has demostrado que de verdad puedo contar contigo, mi buen hermano mayor.

Y a mi tía Inesita, porque del resto de mi familia es probablemente quien más comparte conmigo la ilusión de este logro.

Con infinito amor…

Stefy.

Agradecimiento

A mis padres, hermanos y familiares, por todo lo que representan y su valiosa contribución para el alcance de esta meta.

A mis amigos, especialmente a aquellos que han demostrado seguir ahí a pesar del tiempo y la distancia. Ustedes son la familia que descubrí fuera de casa y por quienes considero a la carrera universitaria como mi mejor etapa, gracias por su amistad y cada momento compartido.

A Andrés, porque en él encontré más que un novio, un amigo incondicional que ha estado presente casi desde el inicio de este proceso. Gracias por tu amor, tu tiempo y apoyo, pero sobre todo, por la forma en que crees y me motivas a creer en mí.

Al ingeniero Mauricio Rea, por su amistad y por la paciencia, tiempo y conocimientos dedicados en la dirección de este proyecto.

Finalmente, a la Universidad Técnica del Norte y la Carrera de Ingeniería en Sistemas Computacionales, por haberme dado la oportunidad de formarme en sus aulas y con excelentes maestros que recordaré a lo largo de mi vida profesional.

A todos, ¡Gracias por tanto!

Stefy.

Tabla de Contenido

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Tomando en cuenta al tiempo como uno de los recursos más valiosos hoy en día, se puede entender por qué continuamente se busca más formas de optimizarlo mediante la automatización de tareas, desde las más complejas hasta las más simples, ocasionando una evolución tecnológica en continuo ascenso y de alguna manera, una exigencia de adaptación en la sociedad que rápidamente se familiariza más con la tecnología y las facilidades que esta ofrece.

Es así que la Carrera de Software de la Universidad Técnica del Norte optó por desarrollar el Sistema Integrado de Actividad Docente, con la finalidad de llevar un registro más ordenado y óptimo de las funciones que los maestros llevan a cabo dentro de la carrera, sin embargo, aunque dicho sistema cumple con el objetivo para el que fue creado, aún es posible reducir el tiempo que toma el registro de actividades, ya que son repetitivas.

En este proyecto de titulación se propone el desarrollo de un módulo adicional, donde los usuarios puedan generar plantillas a partir de sus trabajos previos o viceversa, cambiando únicamente aspectos específicos para cada caso, evitando crear todo desde cero al contar con un punto de partida.

La propuesta se efectuó bajo la metodología Scrum y en base al estándar ISO-IEC 25010, específicamente a la característica de mantenibilidad.

Como resultado final se obtuvo un módulo de plantillas que cuenta con 3 de las 5 subcaracterísticas de mantenibilidad y cumple con los requisitos establecidos al inicio de la planificación, mismos que están detallados en el Capítulo II de este documento.

Palabras clave: Template, mantenibilidad, Scrum.

ABSTRACT

Taking into account time as one of the most valuable resources today, it can be understood why more ways to optimize it are continually being sought by automating tasks, from the most complex to the simplest, causing a technological evolution in continuous ascent and in some way, a demand for adaptation in society that quickly becomes more familiar with technology and the facilities it offers.

Thus, the Software Career of the Technical University of the North chose to develop the Integrated System of Teaching Activity, in order to keep a more orderly and optimal record of the functions that teachers carry out within the career, without However, although this system fulfills the objective for which it was created, it is still possible to reduce the time it takes to record activities since they are repetitive.

In this degree project, the development of an additional module is proposed, where users can generate templates from their previous work or vice versa, changing only specific aspects for each case, avoiding creating everything from scratch by having a starting point.

The proposal was made under the Scrum methodology and based on the ISO-IEC 25010 standard, specifically the maintainability characteristic.

As a final result, a template module was obtained that has 3 of the 5 maintainability sub-characteristics and meets the requirements established at the beginning of planning, these requirements are detailed in Chapter II of this document, detailed in this document.

Keywords: Template, maintainability, Scrum.

INTRODUCCIÓN

PROBLEMA

ANTECEDENTES

El avance de la tecnología y su influencia cada vez mayor, ha hecho del mundo un lugar tecnológico, donde la creación de herramientas que faciliten el trabajo de las personas se ha vuelto una necesidad. Es así que la mayoría de las organizaciones buscan automatizar sus procesos de modo que les permita ser más productivos, ya que minimizan costes y posibilidad de errores, aumentan la velocidad en ejecución de tareas, evitan la acumulación de papel y reducen el riesgo de pérdida de información (ISOTools, 2018).

En el caso particular de la Carrera de Software, los procesos de vinculación, investigación y titulación son realizados de forma individual por los docentes, lo que demanda mucho tiempo que puede ser empleado en otras actividades, por lo tanto, se ha desarrollado el Sistema Integrado de Actividad Docente (SIAD) con la finalidad de optimizar la forma en que se da seguimiento a las funciones de la docencia durante los periodos académicos.

Sin embargo, aunque la creación del SIAD ha sido de gran ayuda, existen procedimientos repetitivos debido a que, en procesos de la misma naturaleza, los parámetros de creación son similares, por lo tanto, sería idóneo que el sistema permita la creación de plantillas.

“Una plantilla es un medio que posibilita construir un diseño predefinido, facilitan la reproducción de copias idénticas o actúan como un punto de partida” (Pérez Porto & Merino, 2014).

SITUACIÓN ACTUAL

Actualmente, el Sistema Integrado de Actividad Docente no cuenta con un módulo de plantillas para ninguno de los procesos que maneja, lo que hace que los docentes tengan que repetir el mismo procedimiento cada vez que crean los informes de sus actividades.

Tomando en cuenta que la carrera está orientada principalmente al desarrollo de software, cada producto entregado debe ser lo más óptimo posible y en este caso, es posible reducir aún más el tiempo de creación de un nuevo ítem, partiendo ya no de cero, sino desde una copia de trabajos previos con parámetros similares, variando únicamente aspectos específicos para cada caso.

PROSPECTIVA

El módulo resultante contribuirá a que el Sistema Integrado de Actividad Docente cuente con una opción en la que, de acuerdo a su criterio o necesidad, el docente cree plantillas de sus proyectos que le permitan disponer de un template para trabajos futuros.

Las plantillas guardadas serán mantenibles, es decir que podrán ser modificadas de forma eficaz y eficiente para mejorarlas, corregirlas o adaptarlas a cambios (ISO-25000, 2019) sin afectar el curso normal del sistema.

PLANTEAMIENTO DEL PROBLEMA

La mayoría de las actividades que los docentes desarrollan o dirigen, tienen características que no varían demasiado de unos a otros, tales como los requerimientos, objetivos, criterios de evaluación, actividades a ejecutarse durante el desarrollo, entre otras, sin embargo, estos datos deben ser ingresados al sistema cada que se cree un proyecto nuevo, lo que no es muy eficiente.

En el sitio web de la carrera se expone que su misión es formar ingenieros capaces de generar, fomentar y ejecutar procesos tecnológicos, de conocimientos científicos y de innovación en el desarrollo de distintas soluciones informática (CISIC-UTN, 2019).

Partiendo de lo antes mencionado, se puede decir que además de mejorar la eficiencia, la posibilidad de crear plantillas dentro del sistema sería una solución tecnológica que mejoraría en gran medida la satisfacción de los docentes, ya que el proceso de creación de una nueva actividad no sería tan repetitivo, por lo tanto, serán los principales beneficiados.

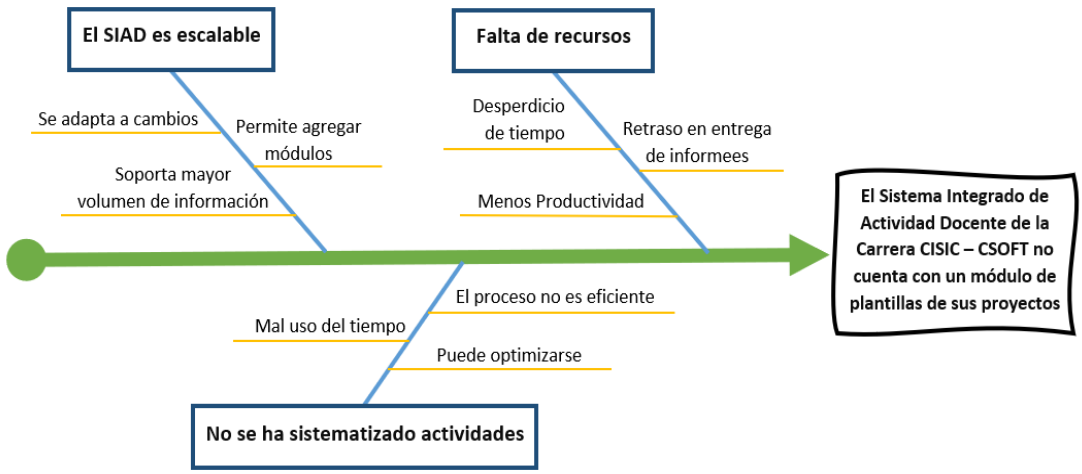


Figura 1: Espina de pescado
Fuente propia

OBJETIVOS

OBJETIVO GENERAL

Desarrollar el módulo de plantillas para el Sistema Integrado de Actividad Docente (SIAD) de la Carrera de Software de la Universidad Técnica del Norte, verificando la característica de mantenibilidad según el estándar ISO/IEC 25010.

OBJETIVOS ESPECÍFICOS

- Elaborar un marco teórico que fundamente el desarrollo de la aplicación.
- Desarrollar el módulo de plantillas en base la metodología Scrum.
- Verificar el funcionamiento con la característica de mantenibilidad del estándar ISO 25010.

ALCANCE

La investigación y análisis de la característica de mantenibilidad del software será documentada dentro del marco teórico del proyecto.

El desarrollo del módulo se llevará en base a la metodología Scrum, en lenguaje de programación Java Enterprise, Wildlfy junto a Java EE como servidor de aplicaciones y el IDE Eclipse. El usuario podrá guardar templates de cualquiera de sus actividades creadas, además podrá visualizar las plantillas existentes y editarlas si considera necesario.

La verificación del funcionamiento se realizará en base al nivel de satisfacción de los usuarios.

Figura 2: Esquema del módulo de plantillas
Fuente propia

JUSTIFICACIÓN

En el campo del desarrollo de software, cada sistema es diseñado en base a los requerimientos de un cliente, no obstante, es posible que sus necesidades puedan sufrir alguna variante, causando que las funcionalidades del programa sean insuficientes o deban sujetarse a ciertas adaptaciones, por lo que la mantenibilidad cobra una gran importancia en el nivel de satisfacción del contratista (Rodríguez et al., 2015).

Según el artículo “Analizando la Mantenibilidad de Software Desarrollado Durante la Formación Universitaria”, la mantenibilidad es un factor altamente significativo en el éxito del producto de software, además es un atributo importante de calidad (Pérez González et al., 2015).

Por lo tanto y teniendo en cuenta que el sistema es escalable, agregar un módulo mantenible al Sistema Integrado de Actividad Docente, será un plus que aumente su calidad y teniendo como funcionalidad la generación de plantillas, permitirá a sus usuarios reducir el tiempo de ejecución de ciertas tareas, haciéndolos más productivos y proactivos, lo que finalmente desencadena en un beneficio para todos quienes conforman la carrera.

Al crear un escenario de trabajo más óptimo para los usuarios, el impacto de este proyecto será sobre todo social, sin embargo, también es un aporte orientado a los objetivos 4 y 9 del desarrollo sostenible, el primero hace referencia a la educación de calidad, misma que empieza por los maestros; y el segundo que hace énfasis en que “sin tecnología e innovación, la industrialización no ocurrirá, y sin industrialización, no habrá desarrollo” (ONU, 2015).

CONTEXTO

Con ayuda de portales bibliográficos se busca encontrar trabajos de investigación relacionados al tema a desarrollar, empezando por los sitios locales y llegando hasta los de mayor alcance.

Como resultado de la búsqueda, se seleccionan tres proyectos de tesis que, si bien no están relacionados con el diseño de plantillas, aportan datos importantes acerca de la característica de mantenibilidad del estándar ISO 2510 con la que se pretende trabajar y serán de gran utilidad al momento de aplicarla.

En la tabla 1 se muestran los trabajos antes mencionados, señalando el nombre, el portal al que pertenecen, una breve descripción de cómo aportan al proyecto actual y cuál es la principal diferencia.

TABLA 1: Contexto

Portal	Proyecto	Diferencia
Universidad Técnica del Norte Ibarra – Ecuador	Benchmarking de Sistemas ERP (Planificación de Recursos Empresariales) Open Source Aplicado A La Empresa Pública Yachay (Narváez Flores, 2019).	Evalúa y compara distintos tipos de software de gestión empresarial, en base a mantenibilidad, es decir que emplea software existente, no desarrolla un sistema en el que se aplique dicha característica.
Universidad de Cuenca Cuenca – Ecuador	Evaluación de un Modelo de Calidad para Objetos de Aprendizaje basado en un Modelo I* (Solís Cabrera, 2018).	Analiza las características que se consideran al construir modelos de calidad, entre las que se encuentra la mantenibilidad, sin embargo, se aplican a objetos de aprendizaje, no a software.
Universidad Nacional de Colombia Medellín - Colombia	Factores de Mantenibilidad en el Desarrollo de Aplicaciones Web (Meza González, 2017).	Contempla factores de mantenibilidad que se deben tomar en cuenta al desarrollar aplicaciones web, sin hacer uso del estándar ISO 25010.

Fuente propia

CAPÍTULO 1

Marco Teórico

1.1 Programación modular

En el mundo del desarrollo de software existen varias opciones al momento de programar, entre las más relevantes se encuentran la programación estructurada y la modular. El origen de esta última es un tema de debate entre varios autores en torno a si estaba presente antes de la programación estructurada o después y se considera una técnica de refinamiento, puesto que es el resultado de descomponer mediante análisis descendente un problema mayor en subproblemas denominados módulos (Beltrán & Aguirre, 2011), mismos que se encargarán de resolver tareas específicas de forma independiente y pueden ser agregados al sistema principal de forma paulatina.

La complejidad de un módulo o subsistema siempre será inferior a la del conjunto y a pesar de que cada uno tiene independencia de funcionamiento y una función específica como se muestra en el ejemplo de la figura 3, todos se enlazan a la estructura superior por lo que pueden comunicarse entre ellos y poner a disposición del resto el resultado de sus funciones, haciendo de esta una de las técnicas más prácticas al crear soluciones informáticas, ya que permite a los desarrolladores trabajar de forma ordenada, agregar funcionalidades y obtener un código más eficiente.

Figura 3: Ejemplo de estructura modular
Fuente propia

1.1.1 Ventajas de la programación modular

- División del problema en partes menos complejas.
- Es ideal para trabajar en equipo.
- Reutilización de código.
- Detección y corrección oportuna de errores.
- Mayor facilidad al realizar pruebas, cambios y mantenimiento.

1.1.2 Desventajas de la programación modular

- Dividir el problema requiere un buen análisis para evitar la redundancia entre módulos.
- Si no se cuenta con un equipo, el desarrollo toma más tiempo.
- La integración de las partes puede ser compleja.
- Si se fracciona el desarrollo en demasiadas partes se reduce la efectividad del sistema.

Una vez que se conocen los pro y contras de este tipo de programación, es oportuno recalcar la importancia de tener claro los criterios de descomposición antes del desarrollo, tales como el tamaño o la independencia funcional. Esta fragmentación puede obtenerse como resultado de un análisis Top-down (de arriba hacia abajo) o uno Bottom-up (de abajo hacia arriba), sin embargo, el sitio dCodinGames recomienda emplear la primera de las estrategias, ya que, partiendo de lo general a lo particular, los subproblemas se irán haciendo visibles de a poco, al igual que la necesidad de adicionar más módulos (Nieva, 2018).

Según Luís Rodríguez , maestro adjunto de la UNIR en España “un módulo se debe dividir hasta que se consiga un nivel aceptable de independencia funcional, es decir que debe tener mucha cohesión y poco acoplamiento” (Rodríguez Baena, 2011).

1.2 Plantilla

De forma general, una plantilla es un diseño con formato preestablecido que sirve de modelo o patrón para la creación más ágil de nuevos ejemplares con las mismas características. Pueden ser utilizadas por personas o por sistemas automatizados (Vázquez, 2013), por lo que son de gran ayuda para la mayoría de industrias, tales como la textil, arquitectónica, mecánica, de diseño gráfico, tecnológica, entre otras, donde el tiempo es un factor determinante y muchos de los procedimientos para crear productos o generar servicios, son repetitivos y de rutina.

1.2.1 Las plantillas y la tecnología.

En el ámbito de las TIC's, se puede hacer uso de plantillas cuando se trabaja varias veces sobre documentos o presentaciones con la misma estructura, en cuyo caso solo se requiere abrir una de ellas y editar el contenido, no obstante, el mayor impacto en esta rama se da en la creación de páginas web, en vista de que la mayoría de negocios y empresas requieren de un sitio en la web que les permita tener un alcance mucho mayor, crecer y contar con una carta de presentación.

En la actualidad existen muchas plataformas que ofrecen diseños predefinidos para la creación de páginas web, tanto así que según W3Techs (w3techs.com), empresa dedicada a proporcionar información sobre el uso de la tecnología web, más del 60% de los sitios existentes en la red han sido creados con alguna de ellas y aunque existe una amplia variedad sigue siendo WordPress quien lidera el mercado, acaparando un 63,9% de ese porcentaje y un 39,3% a nivel general (Soltano, 2020), tal como se muestra en la tabla 2 que lista los diez ejemplares con mayor popularidad al momento.

TABLA 2: Plataformas de administración de contenidos más utilizadas

Plataforma	% Global	% Específico
WordPress	39,3	63,9
Shopify	3,2	5,1
Joomla	2,2	3,6
Drupal	1,5	2,5
Wix	1,5	2,5
SquareSpace	1,4	2,3
Bitrix	1,0	1,7
Magento	0,7	1,2
OpenCart	0,6	1,0
PrestaShop	0,5	0,8

Fuente: (Soltano, 2020)

1.2.2 Ventajas del uso de plantillas.

- Ahorro de tiempo.
- Reducción de esfuerzos.
- Creación de copias idénticas.
- Disponer de un punto de partida para crear algo nuevo.
- Separar la estructura del contenido.
- Hacer uso cuantas veces se requiera conservando el diseño original.
- Permiten de mejoras.

1.2.3 Desventajas del uso de plantillas.

- Pueden quedar obsoletas.
- No admiten cambios radicales.
- Requieren actualizaciones.
- Se pierde originalidad.

1.3 Aplicaciones Web

Una aplicación web es un programa que da paso una mejor comunicación entre las empresas y sus clientes mediante la web, por lo que se han vuelto muy relevantes en el diario vivir de la mayoría de las personas.

Poseen arquitectura cliente-servidor, es decir un proveedor de servicios y un dispositivo que consuma dicho servicio. Como ilustra la figura 4, el funcionamiento de una aplicación web se da de la siguiente manera:

- El usuario hace una petición al servidor web a través de Internet y la interfaz de la aplicación.
- El servidor envía la solicitud al servidor de aplicaciones web.
- El servidor de aplicaciones ejecuta la tarea solicitada y una vez generados los resultados los devuelve información solicitada o datos procesados al servidor web.
- El servidor web lleva la información solicitada al dispositivo cliente.
- La información solicitada aparece en la pantalla del usuario.

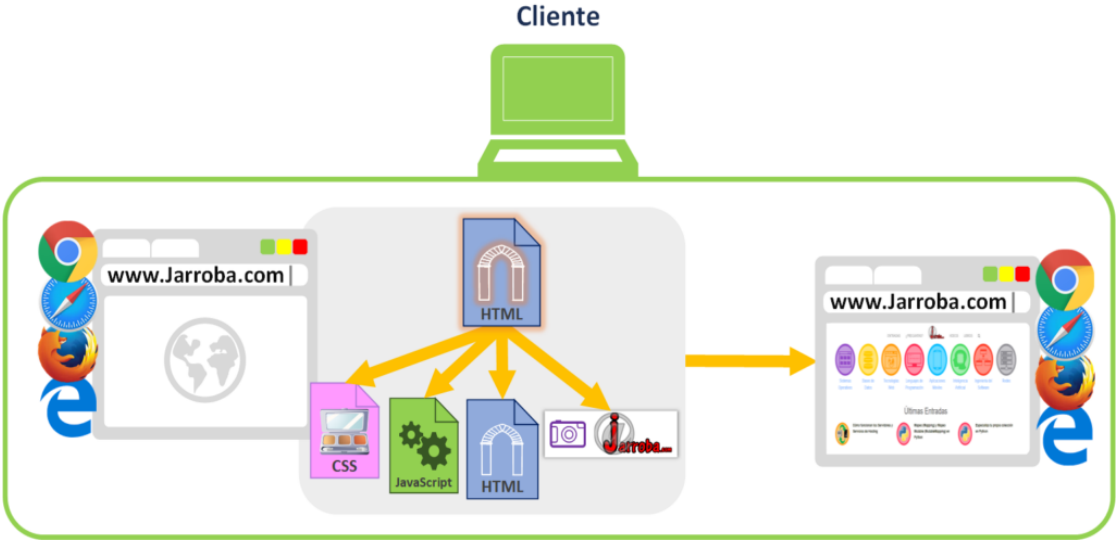


Figura 4: Arquitectura cliente-servidor
Fuente: (Invarato, 2019)

1.3.1 Beneficios.

Este tipo de aplicaciones son preferidas por muchas empresas porque permiten a los miembros de su institución realizar tareas conjuntas como trabajar simultáneamente en documentos compartidos vía web, crear informes, archivos e intercambiar información sin importar donde se encuentren siempre y cuando tengan a la mano un dispositivo inteligente y servicio de internet.

Además de los mencionados, existen muchos más beneficios, tales como:

- No es necesaria la instalación en disco duro, así que no causa limitaciones de espacio.
- Requieren menos soporte, mantenimiento y requisitos técnicos.
- Reducción de costos.
- Se mantienen actualizadas.
- No existen inconvenientes de compatibilidad entre usuarios.
- Multiplataformas.
- Menos responsabilidad para el desarrollador.
- Reducen la piratería de software. Referencia bibliográfica

1.3.2 Ejemplos

Algunos ejemplos de aplicaciones web incluyen correo web, procesadores de texto y hojas de cálculo. Dentro de tanta variedad, el sitio Indeed propone los siguientes:

Edición de videos y fotos, la conversión y el escaneo de archivos.

Los programas de correo electrónico populares como Yahoo y Gmail, y los servicios de mensajería instantánea.

Las aplicaciones como Google Docs, Google Slides, Google Sheets que permiten el trabajo conjunto.

Aplicaciones evolucionadas como Facebook o Dropbox que admiten descarga y uso en un teléfono o tablet.

Carritos de compras, subastas, wikis y banca en línea. (Indeed, 2020)

1.4 Lenguajes de programación

Un lenguaje de programación es considerado un idioma que permite la comunicación entre el programador y un ordenador, esto es posible gracias a que cada uno tiene expresiones propias que con cierta sintaxis y semántica tienen un significado que la máquina puede entender y procesar. Cuando un lenguaje ha logrado un dominio suficiente en la resolución de problemas, los programas que se desarrollen suelen ser soluciones adecuadas en ese campo (Mcguire, 2009).

Existe una gran variedad en el mercado, pero actualmente, los de mayor demanda son los que presenta la figura 5.

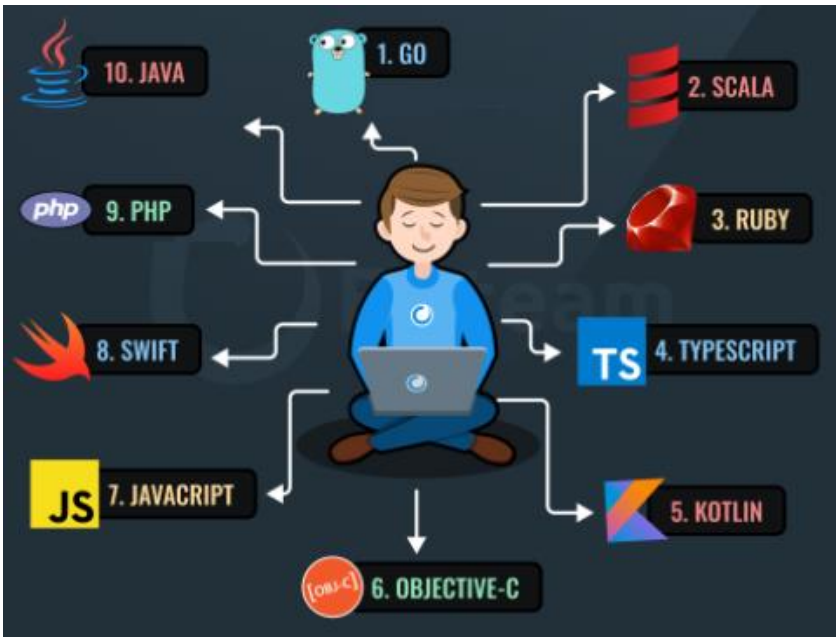


Figura 5: Lenguajes de programación con mayor demanda
Fuente: (EdTeams, 2020)

"Cada lenguaje de programación es útil con su propio traductor, es decir, intérprete o compilador, según sea el caso. instrucciones dadas" (Usman et al., 2016).

1.4.1 Lenguajes compilados.

Son aquellos que requieren del uso de un compilador que se encargue de tomar el código fuente y traducirlo en lenguaje de máquina o código binario para que el computador pueda entender las instrucciones del programa y ejecutarlo (Anónimo, 2008).

Características.

- Son eficientes.
- Traducen cada instrucción una sola vez y la utiliza cuantas veces sea necesario.
- No devuelven resultados el código tiene errores.
- Pueden no ser compatibles con otros lenguajes.

1.4.2 Lenguajes interpretados.

Este tipo de lenguajes hacen uso de un intérprete que se encarga de convertir las instrucciones del programa a código máquina, pero lo hacen de forma progresiva conforme se vaya leyendo el programa, por lo que siempre serán necesarios los dos componentes para la ejecución (Anónimo, 2008).

Características.

- Pueden adaptarse a cualquier intérprete sin necesidad de cambios.
- Requieren del intérprete cada vez que se ejecuta una instrucción.
- Ejecutan programas aunque tengan errores, siempre que no lea las líneas que los contienen.
- Tienen compatibilidad con otros lenguajes.

1.5 Java

Lanzado en 1995 como componente central de Sun Microsystems, Java es un lenguaje de programación de alto nivel compatible con varios sistemas operativos y que ha sabido adaptarse a varios modelos de negocio.

El sitio Tutorials Point como parte de su curso Java, destaca acerca de este lenguaje la siguiente lista de características:

Java es:

- Orientado a objetos: Todo es un objeto y se basa en el modelo Object.
- Simple: Está diseñado para que sea fácil de aprender
- Seguro: Permite desarrollar archivos libres de virus y sistemas. Usa cifrado de clave pública.

- Arquitectura neutral: El código compilado es ejecutable en muchos procesadores.
- Portátil: Ser de arquitectura neutral y no tener una implementación dependiente aspectos de la especificación hacen que sea portátil.
- Robusto: Elimina situaciones propensas a errores, enfatizando principalmente en la verificación de estos en tiempo de compilación y ejecución.
- Multiproceso: Es posible escribir programas que puede realizar muchas tareas simultáneamente, permitiendo construir aplicaciones interactivas que puedan funcionar sin problemas.
- Interpretado: Su código de bytes se traduce a la máquina durante la ejecución y no se almacena en ningún lugar. El proceso de desarrollo es incremental y ligero.
- Alto rendimiento: Gracias al uso de compiladores Just-In-Time.
- Distribuido: Está diseñado para el entorno distribuido de Internet.
- Dinámico: Los programas Java pueden llevar una amplia cantidad de información que se puede utilizar para verificar y resolver accesos a los objetos en tiempo de ejecución. (tutorialspoint, n.d.)

Ediciones Java.

Como muestra la figura 6, Java cuenta con tres ediciones:

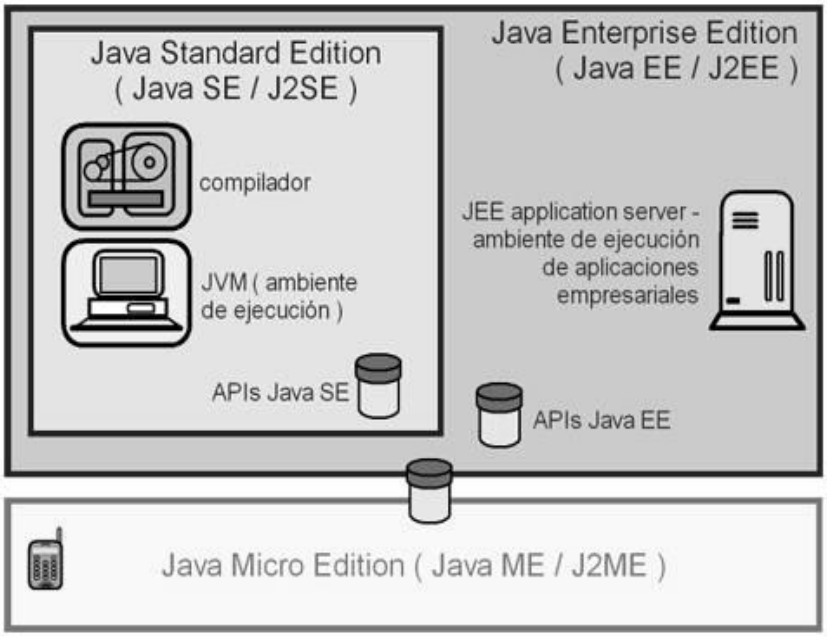


Figura 6: Ediciones Java
Fuente: (Arak, 2012)

a) Estándar (SE)

Permite el desarrollo e implementación de aplicaciones Java en escritorios y servidores. Brinda además de otras características, una interfaz de usuario ideal para las exigencias en la actualidad y permite administrar aplicaciones.

b) Incrustado (ME)

Ofrece un entorno de ejecución optimizado y seguro, ideal para dispositivos basados en red. Su diseño es ideal y seguro para dispositivos con recursos limitados en cuanto a. memoria y capacidad de procesamiento.

c) Empresarial (EE)

Esta edición tiene un alcance mucho mayor, ya que está pensada para aplicaciones empresariales de una dimensión mucho mayor, donde se ofrece una API y un entorno de ejecución de aplicaciones que admite la integración de aplicaciones y datos heredados. (Oracle, 2020)

1.6 Java Server Faces (JSF).

JSF es la tecnología web estándar de Java usada sobre todo para el desarrollo empresarial a gran escala. Permite crear interfaces web basadas en componentes y orientadas a eventos que permite el acceso a datos y lógica del lado del servidor y encapsula

tecnologías como HTML, CSS y JavaScript del lado del cliente, lo que permite la creación de interfaces web sin mucha interacción con estas tecnologías (Tyson, 2018).

Se trata de un documento XML que representa componentes formales en un árbol lógico, que están respaldados por objetos Java, son independientes del HTML y tienen la gama completa de capacidades, incluido el acceso a bases de datos y API remotas.

Otra de sus características es que los datos se introducen y se muestran en forma de texto, pero se almacenan en un formato dependiente de la aplicación, así que necesitará conversores y validadores asociados para hacer la conversión.

1.6.1 Componentes.

Cuenta con diversos componentes y etiquetas. Cristian Henao expone una detallada lista de algunos de ellos, no obstante, para ejemplo se ha tomado los siguientes:

- <h:form /> Formulario de entrada
- <h:panelGrid /> Tabla con componentes
- <h:inputText /> Entrada de texto
- <h:inputTextArea /> Entrada de texto multilínea.
- <h:outputText /> Muestra un texto estático
- <h:outputLabel /> Se puede asociar a otro componente
- <h:outputLink /> Enlace o hipertexto
- <h:message /> Mensajes de página
- <h:commandButton /> Botón que realiza acciones
- <h:commandLink /> Botón asociado a un enlace
- <h:selectOneMenu /> Menú para selección de una sola opción
- <h:selectManyMenu /> Menú para selección de una sola varias opciones
- <h:dataTable /> Tabla para lista de datos
- <h:column /> Columna dentro de DataTable
- <h:graphicImage /> Permite vincular imágenes (Henao, 2017)

1.6.2 Arquitectura

Este framework maneja arquitectura MVC (modelo, vista, controlador), por ende, está orientado a recoger datos del usuario, pasarlos a la capa del modelo de la aplicación, realizar las acciones correspondientes en la aplicación y retornar el resultado (Alicante, 2014) como se aprecia en la figura 7.

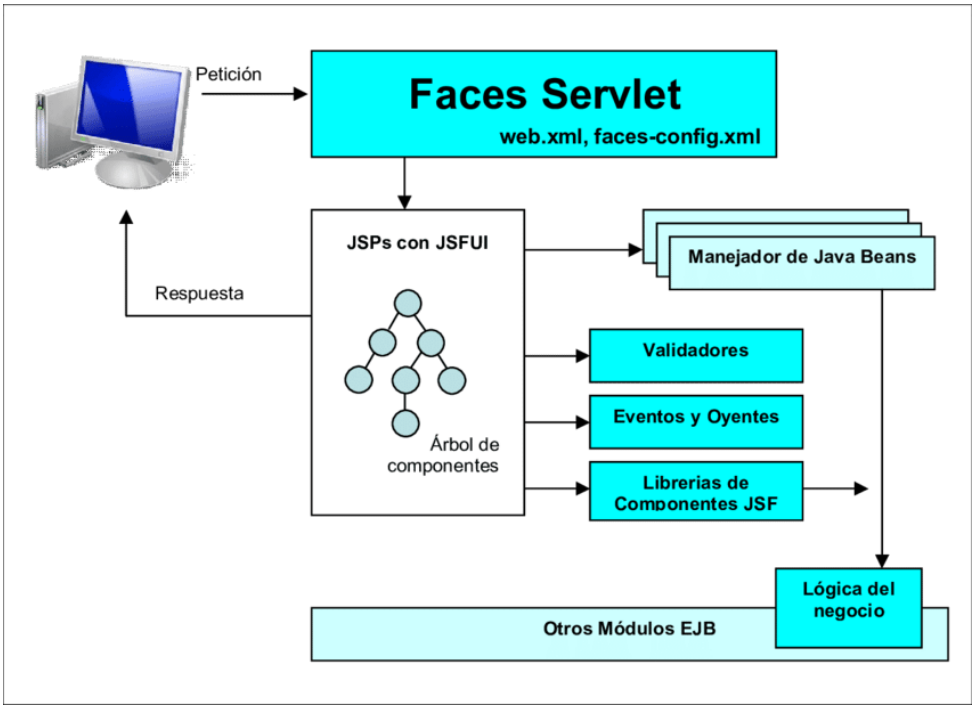


Figura 7: Arquitectura de componentes de una aplicación JSF
Fuente: (Sanoja, 2020)

JSF también ha generado una gran cantidad de marcos y bibliotecas, que se han mantenido al día con las mejoras recientes del lado del cliente, entre los más destacados se puede mencionar a PrimeFaces como uno de ellos.

1.6.3 Beneficios

TABLA 3: Beneficios de JSF

Beneficio	Detalle
Similar a HTML	El código JSF con que se crean las vistas es muy fácil de usar por desarrolladores y diseñadores web porque es muy parecido al HTML estándar.
Se integra dentro de la página JSP	Se encarga de la recogida y generación de los valores de los elementos de la página
Permite introducir JavaScript	Acelera la respuesta de la interfaz en el cliente
Es extensible	Se pueden desarrollar nuevos componentes a medida
Modificable	Se puede hacer uso de API's para cambiar el comportamiento del framework ya que controlan su funcionamiento

Fuente: (Muñoz, 2012)

1.6.4 Ciclo de vida

El ciclo de vida de JSF es similar al de una página JSP, sin embargo, debido a sus características, el ciclo incluye las seis fases presentes en la figura 8.

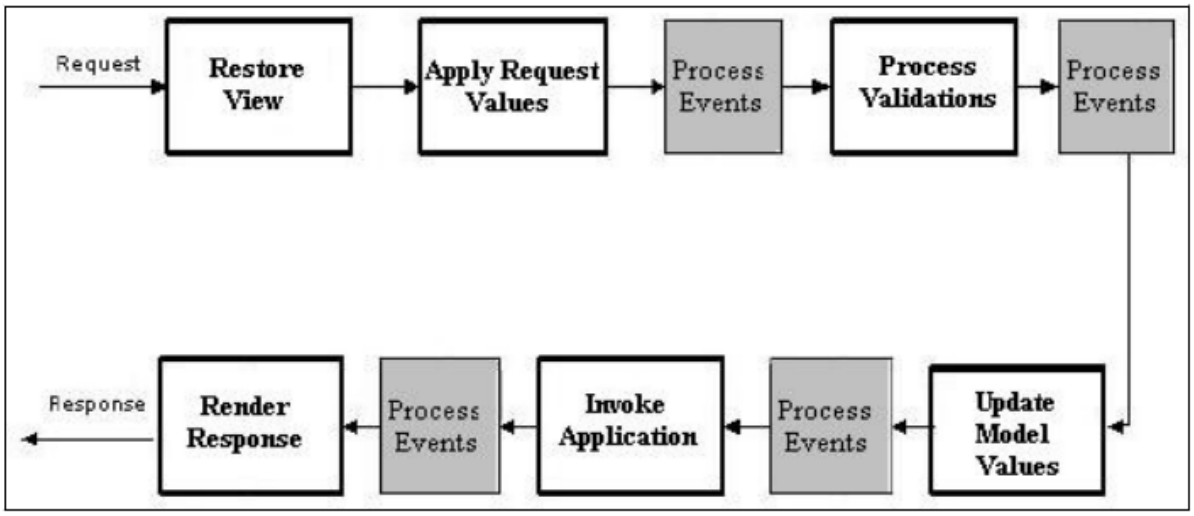


Figura 8: Ciclo de vida de JSF
Fuente: (Viveros, 2010)

(Viveros, 2010) describe brevemente cada etapa refiriendo lo siguiente:

Restore View: inicia cuando se hace una petición, por lo tanto, es la primera en ejecutarse y su función está dada en favor de estructurar la página con todos los componentes que la conformen.

Apply Request Values: almacena los componentes creados en la fase pasada luego de obtener el valor devuelto por la petición.

Process Validations: valida los componentes almacenados en base a las reglas declaradas.

Update Model Values: la condición para llegar a esta fase es haber superado con éxito las anteriores. Ya estando dentro los valores locales de los componentes son utilizados para actualizar los beans que están ligados a dichos componentes [Geary & Geary, 2004].

Invoke Application: se hace el llamado a la acción u operación que corresponde al evento inició el proceso.

Render Response: como su nombre lo insinúa, renderiza la respuesta antes de devolverla al cliente, quien finalmente recibe el resultado de un flujo normal o no dependiendo del resultado de las fases anteriores.

1.7 Primefaces.

Es un popular marco de código abierto que reúne un conjunto de más de 100 componentes para el desarrollo de aplicaciones web enriquecidas y posee una amplia documentación que permite a su gran número de usuarios acoplarse mejor a su uso.

PrimeFaces tiene que ver con el front-end y es totalmente compatible con los siguientes marcos de referencia:

- Spring Core (Integración JSF Centric JSF-Spring)
- Spring WebFlow (Integración Spring Centric JSF-Spring)
- Spring Roo (complemento PrimeFaces)
- EJB
- CDI

1.7.1 Características.

Según el sitio EcuRed, PrimeFaces se caracteriza principalmente por lo siguiente:

- Librería con componentes Ajax de fácil uso.
- Ligero, un Jar, cero configuraciones y no requiere dependencias.
- No requiere complicadas configuraciones.
- Documentación abundante.
- Alrededor de 30 temas preconfigurados.
- Soporte para interfaces de usuario sobre dispositivos móviles, provee de un kit para este tipo de desarrollo.
- Cada vez más usuarios la usan, quienes aportan con ideas que le permiten mantenerse en mejora continua.
- Los u
- No ofrece ningún tipo de resistencia a la integración de JSF con Spring Framework. (EcuRed, 2015)

1.7.2 Componentes

En la figura 9 se muestran algunos de los más de 100 componentes que esta librería posee.

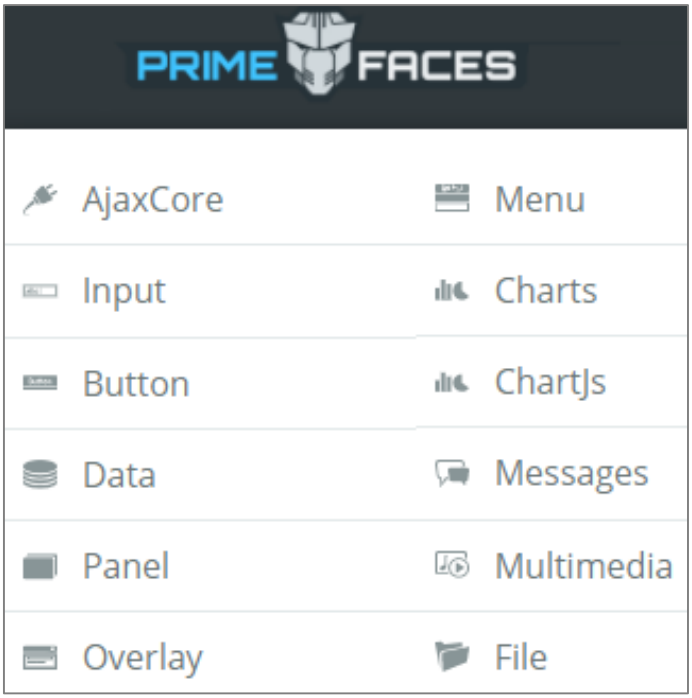


Figura 9: Componentes Primefaces
Fuente:(Primefaces, 2020)

1.7.3 Beneficios

TABLA 4: Beneficios de Primefaces

Beneficio	Detalle
Código abierto	Alojado en GitHub, todos los componentes son de código abierto y de uso gratuito bajo licencia MIT.
Temas	No se limita a una sola apariencia. Posee variedad de opciones que incluyen material y diseño plano.
Plantillas	Plantillas de aplicación basadas en facetas altamente personalizables diseñadas profesionalmente para comenzar en poco tiempo.
Accesibilidad	Totalmente accesible y de conformidad con los estándares de la Sección 508.
Soporte Pro	Con los servicios exclusivos de la cuenta Pro, no se necesita publicar preguntas en el foro de la comunidad y problemas en el rastreador.

Productividad	Permite dedicar tiempo a la lógica empresarial en lugar de ocuparse de los complejos requisitos de la interfaz de usuario.
Comunidad	Admite unirse a su comunidad y ser parte de la fundación de código abierto activa, vibrante y en crecimiento que es.
Móvil	Experiencia de usuario móvil mejorada con elementos de diseño sensibles al tacto optimizados.

Fuente: (Primefaces, 2020)

1.8 SIAD – UTN

El Sistema Integrado de Actividad Docente (SIAD) es una aplicación web desarrollada con la finalidad de dar un mejor manejo a las actividades que realizan los maestros de la carrera de software, de modo que puede haber constancia del avance en el cumplimiento de las mismas y un estándar en la entrega de informes.

El SIAD está desarrollado en java y tiene estructura modular, donde cada módulo (descrito brevemente en la tabla 5), cumple con funciones específicas que al ser integradas, arrojan en conjunto como resultado un sistema multifuncional en el que los docentes pueden llevar un registro ordenado de horarios, clases, reportes, seguimiento de vinculación, prácticas, titulación, tutorías y demás.

TABLA 5: Módulos que conforman el SIAD

Nombre	Función
Módulo de personalización de frontales.	Provee un conjunto de plantillas Facelets que son usadas en la interfaz de todo el sistema.
Módulo de seguimiento de trabajos de titulación.	Permite llevar un registro y control del avance en la ejecución de actividades programadas para el desarrollo de tesis.
Dashboard de alertas.	Notifica y alerta mediante colores el avance o retraso en el cumplimiento de pendientes, con el fin de tomar acciones a tiempo.
Módulo de firmas digitales.	Aporta con la posibilidad de firmar digital mente documentos de forma segura.
Módulo de auditoría informática	Contribuye a la seguridad de los usuarios del sistema mediante métodos como el cifrado de contraseñas.

Fuente propia

1.9 Metodologías de trabajo

Cuando se ha decidido empezar un proyecto, la toma de decisiones es un aspecto crucial, ya que es de ahí que depende el éxito o el fracaso en su ejecución.

Una de las decisiones más importantes es la elección del marco de trabajo, actualmente muchas industrias las emplean sabiendo que una buena o mala elección, influirá significativamente en el alcance del propósito y aunque en los últimos años ha crecido considerablemente la oferta, muy pocas opciones garantizan una gestión eficaz, que sea equitativa con los miembros del equipo y que además permita llevar un control en el transcurso de la ejecución del proyecto.

1.9.1 Scrum

Scrum es una metodología ágil muy popular en lo que respecta a gestión de proyectos, esto gracias a que prioriza la responsabilidad, el trabajo en equipo y el progreso iterativo para lograr un objetivo claro.

Este marco de trabajo se basa en tres pilares: adaptación, inspección y transparencia, y aunque podría parecer lo contrario, tiene una ideología simple que se basa en comenzar con lo que resulte menos complejo y hacer un seguimiento del progreso para realizar los cambios que se requieran.

Recibe su nombre debido a que en rugby, existe una formación donde cada miembro de un equipo tiene un papel específico que desempeñar, al igual que scrum conformando el equipo con roles específicos como el scrum master, el propietario del producto y el equipo de desarrollo, así que es de ahí de donde nace su nombre (Ronche, 2020).

Roles.

a) Scrum master

Es quien facilita el equipo y lleva el control de avances, además es responsable de que se utilicen las mejores herramientas y prácticas de scrum.

b) Propietario del producto

Es un miembro del equipo que actúa como enlace entre los clientes y el equipo de desarrollo. Responsable de garantizar que todas las expectativas para el resultado final hayan sido claramente comunicadas y acordadas por las partes interesadas.

c) Equipo de desarrollo

Este grupo trabaja en conjunto para lograr un objetivo común. Trabajan para crear y probar las versiones incrementales del producto final.

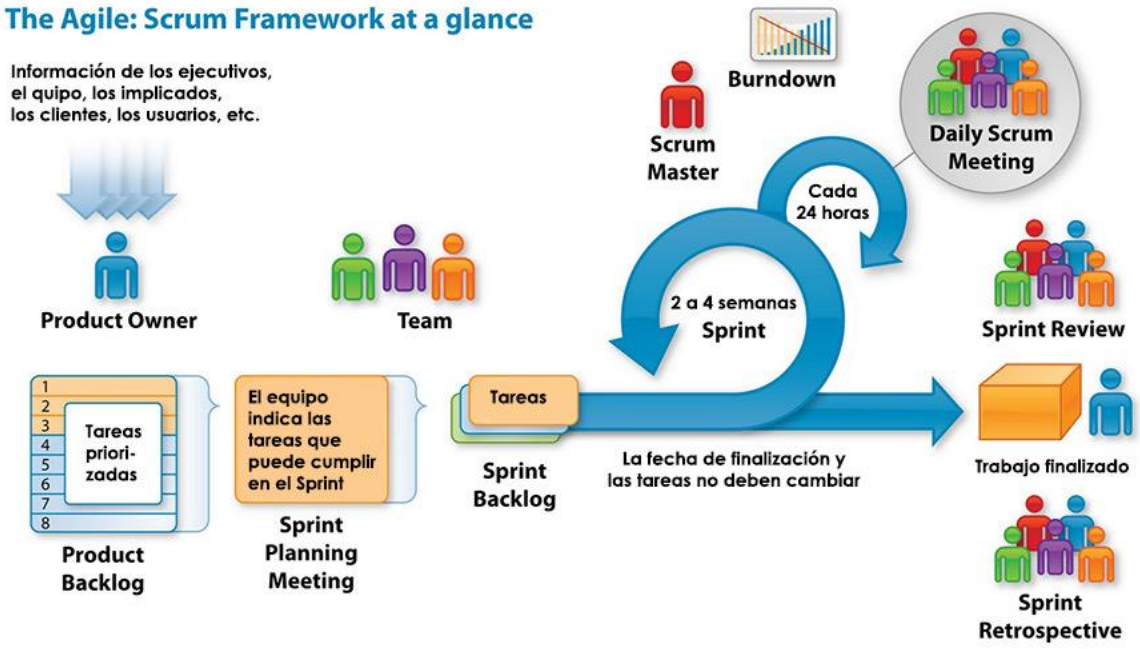


Figura 10: Estructura Scrum
Fuente: (Jeferson, 2018)

Características.

Un proceso de scrum se diferencia de otras metodologías ágiles debido a sus prácticas y conceptos específicos. Scrum tiene tres categorías: cajas de tiempo, artefactos y roles.

Generalmente, los principios de scrum se utilizan para lidiar con el desarrollo de software y productos complicados. Debido a que es parte del enfoque ágil, también utiliza prácticas incrementales e iterativas durante el proceso de desarrollo.

De forma similar a los marcos de cascada clásicos, se sabe que scrum aumenta la productividad y reduce el tiempo. También permite a las organizaciones de cualquier tamaño adaptarse sin problemas a los requisitos que cambian rápidamente y crear un producto escalable para cumplir con los objetivos cambiantes de una empresa.

Un método scrum puede beneficiar a cualquier organización ayudándola a:

- Tener un mayor control sobre el proyecto y el cronograma
- Espere cambios y manejarlos mejor
- Ofrecer mejores estimaciones, dedicando menos tiempo a ello
- Aumentar sustancialmente la calidad de la entrega

Scrum centra su enfoque en la responsabilidad, ya que es un proceso que involucra iteraciones hacia un objetivo o meta bien definido. Todos están trabajando para adoptar estrategias rápidamente.

El propietario de un producto colabora estrechamente con el equipo para determinar y centrarse en la funcionalidad del sistema. Lo hacen estableciendo una acumulación de productos, que incluye todo lo que se debe lograr para entregar un software que funcione correctamente. El trabajo pendiente debe tener correcciones de errores, requisitos no funcionales y características, entre otras cosas.

Figura 11: Valores Scrum
Fuente: (Salazar, 2020)

Cuando se establecen las prioridades, los equipos multifuncionales necesitan estimar y registrarse para proporcionar incrementos de software dentro de Sprints continuos que generalmente duran 30 días. Después de realizar el trabajo pendiente de un Sprint, solo el equipo puede agregar funcionalidad al Sprint. Una vez que se realiza la entrega del Sprint, se evalúa la acumulación de productos y se vuelve a priorizar si es necesario, y el siguiente conjunto de entregables se selecciona para formar parte del próximo Sprint.

En un estudio realizado por Forbes, el 49 por ciento de los altos directivos encuestados por la empresa creían que un scrum es un enfoque eficaz. Afirmaron que su popularidad se debe al hecho de que se centra en los clientes. Además, es una metodología probada y probada para una colaboración óptima.

Debido a que también reduce los errores de manera efectiva y ayuda a finalizar las entregas de proyectos de manera oportuna, la escoria está ganando más atención en el mundo ágil. Incluso se puede destacar que en uno de los informes que State of Agile saca cada año se da a conocer que con el 70% de preferencia, Scrum es la metodología ágil más popular dentro de los equipos.

Los beneficios de scrum ya se han extendido a otras funciones comerciales, incluidas marketing y TI. En estas áreas, hay proyectos que necesitan avanzar en presencia de ambigüedad y complejidad, algo en lo que Scrum puede ayudar mucho.

La mayoría de los equipos de liderazgo también están comenzando a basar sus prácticas de gestión ágiles en scrum. A menudo lo usan junto con Kanban y prácticas lean, que son subconjuntos de ágiles.

Artefactos.

- **Lista de Producto** (Product Backlog)**:** es una lista ordenada de todos los requisitos que debe cumplir el producto final, los elementos de esta lista pueden ser actualizados.
- **Lista de Tareas** (Sprint Backlog)**:** es el conjunto de tareas de la Lista de Producto que los miembros del equipo de desarrollo lo van a realizar durante el Sprint.
- **Incremento:** es el resultado de todos los elementos de la Lista de Producto que se han completado durante el Sprint.

1.10 Calidad de Software

"La calidad de software se refiere al grado de desempeño de las características que debe cumplir un sistema computacional durante su ciclo de vida, de modo que se garantice al cliente contar con un sistema confiable, que satisfaga sus necesidades en cuanto a funcionalidad y eficiencia” (Callejas Cuervo et al., 2017).

Con el afán de mejorar los procesos que permitan llegar a un producto de software con calidad, a lo largo del tiempo se han diseñado diversos estándares tomando en cuenta las diversas etapas del desarrollo, es así que se cuenta con modelos de calidad a nivel de proceso, de producto y de uso, mismos que han ido evolucionando, o que han sido útiles como puntos de referencia para la creación de las normas más actuales.

La figura 12, muestra una línea de tiempo de cómo se han ido creando distintos modelos de calidad hasta llegar a la familia de las ISO 25000 que se empleará en este trabajo y cuya elaboración se basó principalmente en una de sus antecesoras, la ISO 9126.

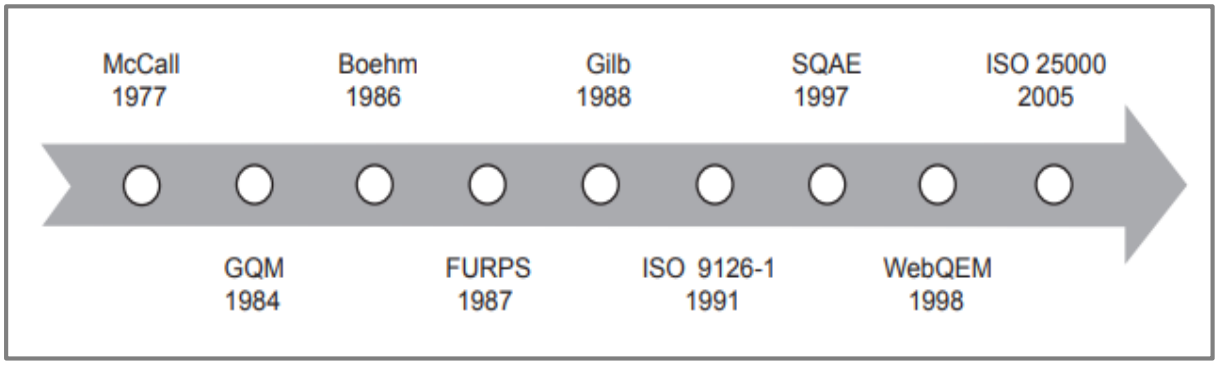


Figura 12: Modelos de calidad a nivel de producto
Fuente: (Callejas Cuervo et al., 2017)

Javier López (2015) define a la calidad como un concepto complejo y multidimensional que puede ser descrita desde las siguientes perspectivas:

a) Trascendental

   La calidad es reconocida, pero no descrita. Las definiciones no son objetivas ni cuantificables.

b) Usuario

   La calidad está directamente ligada a la satisfacción de las necesidades del usuario, tomando en cuenta características como la fiabilidad, el rendimiento o la eficiencia.

c) Fabricación o proceso

   Se encauza en el proceso de desarrollo que maneje la empresa para cumplir con el nivel de conformidad de acuerdo a sus políticas internas y a la capacidad de producción, para lo que se tiene en cuenta características como la tasa de defectos o costes de re-trabajo.

d) Producto

   Las características de calidad del producto se definen a partir de líneas de código, complejidad, diseño, etc.

e) Aspectos económicos

Miden la calidad con base en los costes, precios, productividad. (Valenciano López, 2015)

1.11 ISO/IEC 25010

Este modelo de calidad es la base sobre la cual se instaura todo el sistema para la evaluación de la calidad del producto. En este modelo se determinan las características de calidad que se van a tener en cuenta a la hora de evaluar las propiedades de un producto software determinado.

La industria desarrolladora de software ha evolucionado notablemente, pero a pesar de ello aún existen deficiencias en la calidad del producto final entregado, por lo que es importante contar con ciertos parámetros de calidad, teniendo en cuenta su importancia debido al nivel de impacto.

La calidad del software se puede interpretar como el grado en que dicho producto satisface los requisitos de sus usuarios aportando de esta manera un valor. Son precisamente requisitos como funcionalidad, rendimiento, seguridad, mantenibilidad, etc., los que se encuentran representados en el modelo ISO/IEC 25010. Es uno de los más utilizados debido a que para realizar la evaluación, categoriza la calidad del producto en características y subcaracterísticas (listadas en la figura 13) y las analiza directamente en el software.

Figura 13: ISO 25010, características y subcaracterísticas
Fuente (ISO-25000, 2019)

1.12 Característica de mantenibilidad

Las métricas de software han sido muy criticadas en los últimos años, porque los críticos malinterpretan la intención detrás de la tecnología, así que, en esencia, son buenas

herramientas de modelado, pero para que puedan aprovecharse en medición, depende de la coherencia y la forma con que se apliquen.

(Malhotra & Chug, 2016) afirman que:

> El mantenimiento de software es una actividad costosa que consume una parte importante del costo del proyecto total. Varias actividades que se llevan a cabo durante el mantenimiento incluyen la adición de nuevas funciones, eliminación de código obsoleto, corrección de errores, etc. La mantenibilidad del software significa la facilidad con la que estas operaciones se pueden realizar. Si la capacidad de mantenimiento se puede medir en las primeras fases del desarrollo del software, ayuda a una mejor planificación y una utilización óptima de los recursos.
>
> La medición de propiedades de diseño como el acoplamiento, la cohesión, etc. en las primeras fases de desarrollo a menudo nos lleva a derivar la mantenibilidad correspondiente con la ayuda de modelos de predicción.

"Muchos factores problemáticos en la fase de desarrollo de software afectan la capacidad de mantenimiento de los entregables, por lo que comprenderlos puede ayudar no solo a reducir la incidencia de fallas en el proyecto, sino también a garantizar la mantenibilidad del mismo" (Chen & Huang, 2009).

Figura 14: Subcaracterísticas de la mantenibilidad
Fuente: Propia

Como muestra a figura 14 el estándar ISO 25010 divide a la mantenibilidad en cinco subcaracterísticas, descritas con mayor detalle más adelante.

1.12.1 Reusabilidad

"La reusabilidad es un factor de calidad importante que permite volver a aplicar un diseño a un nuevo problema sin mucho esfuerzo adicional" (Huda et al., 2015).

El constante ascenso en la demanda de productos de software ha provocado un crecimiento cada vez mayor en la cantidad y calidad de herramientas tecnológicas que salen al mercado, lo que según Cristian Pacífico (2004) supone un grave problema en la industria, que la reusabilidad podría resolver "debido a que el software está típicamente compuesto por partes similares y la mayoría desarrollos nuevos pueden ser ensamblados a partir de componentes preexistentes" (Pacífico, 2004).

A continuación, en la tabla 6 se detallan otros aspectos característicos de esta subcaracterística:

TABLA 6: Consideraciones acerca de la reusabilidad

Ventajas	Reduce tiempos de ejecución, costes de desarrollo y esfuerzo intelectual.
	Incrementa la productividad.
	Software nuevo a partir de componentes existentes.
	Mayor fiabilidad al usar software que ya ha sido probado.
Recursos Reutilizables	Especificaciones de requisitos y diseños previamente definidos.
	Código probado y depurado con anterioridad.
	Experiencia del equipo de desarrollo.
	Paquetes de software de propósito general.
Tipos	Oportunista. - Cuando se reutiliza piezas que se ajustan a un problema.
	Sistemática. - Se planifica de modo que todo componente reutilizado sea ideado, a priori. Implica almacenamiento en repositorios, inversiones iniciales y diseño componentes genéricos o específicos para una determinada aplicación, que puedan reutilizarse con facilidad.
	Top-Down. - Se determinan las piezas necesarias que encajan unas con otras, se van desarrollando poco a poco y requiere alta inversión a comienzo.

Fuente: (Valenciano López, 2015)

1.12.2 Modularidad

La modularidad “se conoce como el grado, en el que un sistema se fracciona en módulos de menor tamaño de modo las consecuencias generadas por una modificación en uno de ellos sea mínimas para el resto” (Rodríguez & Piattini, 2007).

Como se había mencionado anteriormente, esta propiedad permite que un cambio en los componentes de un sistema tenga un impacto mínimo en los demás.

1.12.3 Analizabilidad

La analizabilidad está enfocada en el proceso de detección de las deficiencias del software y la estimación del esfuerzo para mitigar o resolver el problema.

Es así que Suhel y Raees Ahmad (2012), definen a la analizabilidad como un atributo clave en la capacidad de mantenimiento para productos de alta calidad, ya que permite diagnosticar carencias o causas de fallas en el software e identificar las partes a modificar para la solución (Ahmad Khan & Ahmad Khan, 2012).

1.12.4 Capacidad de ser probado

Se refiere a lo factible que resulte establecer criterios de prueba en un software o alguno de sus componentes y aplicar dichos criterios para su evaluación.

El sitio Programación y más (https://programacionymas.com/) señala como más importantes los siguientes tipos de pruebas:

- Unitarias
- De integración
- Funcionales
- De punta a punta
- De regresión
- De humo
- De aceptación
- De rendimiento

Existes otros tipos de pruebas y se emplean teniendo en cuenta distintos factores como la necesidad y presupuesto del cliente o el tamaño, complejidad y estructura del sistema.

1.12.5 Capacidad de ser modificado

“Capacidad del producto que permite que sea modificado de forma efectiva y eficiente sin introducir defectos o degradar el desempeño” (ISO-25000, 2019).

Los cambios serán menos significativos mientras más temprana sea la etapa en que se realicen y su costo aumenta a medida que se avance en el desarrollo.

1.13 Modelo MANTuS

1.13.1 Introducción

Para la verificación la característica de mantenibilidad, se toma como marco de referencia el modelo MANTuS propuesto por Erazo, Florez y Pino, mismo que está basado en la mantenibilidad del software y que según sus autores permitió potenciar dicha característica en un producto tomado como caso de estudio, evidenciando la efectividad de llevar a cabo las prácticas propuestas (Erazo Martínez et al., 2016).

Aunque el modelo se enfoca en las 5 subcaracterísticas de mantenibilidad especificadas en el estándar ISO/IEC 25010, también se hizo un análisis de otras normas con el fin de aportar mayor precisión al diseño, teniendo como resultando una lista de atributos a considerar al momento de medir el grado de mantenibilidad de un sistema.

La figura 22 especifica la lista antes mencionada, detallando la subcaracterística a la que aporta cada atributo y en que parte del proceso de desarrollo de software puede aplicarse.

Subcaracterísticas / Atributos	CA	M	CM	CP	R	Procesos
Acoplamiento	X	X	X	X	X	Diseño arquitectural, Diseño detallado
Anidación	X		X	X		Construcción de software
Capacidad de expansión			X			Operación de software
Cohesión	X	X	X	X	X	Diseño detallado
Comentarios	X		X		X	Construcción de software
Complejidad	X	X	X	X	X	Diseño arquitectural, Diseño detallado, Construcción de software
Consistencia	X		X			Análisis de requisitos, Diseño arquitectural, Diseño detallado, Construcción de software, Pruebas de evaluación de software, Operación de software
Documentación	X		X	X	X	Análisis de requisitos, Diseño arquitectural, Diseño detallado, Construcción de software, Pruebas de evaluación de software, Operación de software
Duplicación	X		X	X		Construcción de software
Encapsulamiento	X	X	X	X		Diseño detallado
Estandarización	X		X			Construcción de software
Facilidad de lectura	X		X	X	X	Construcción de software
Facilidad de entendimiento	X		X	X	X	Diseño arquitectural, Diseño detallado, Construcción de software
Herencia	X		X	X	X	Diseño detallado
Polimorfismo	X		X		X	Diseño detallado
Simplicidad	X	X	X	X	X	Análisis de requisitos, Diseño arquitectural, Diseño detallado, Construcción de software
Tamaño	X		X	X	X	Construcción
Trazabilidad	X		X			Análisis de requisitos, Diseño arquitectural, Diseño detallado, Construcción de software, Pruebas de evaluación de software

CA = Capacidad para ser analizado, M = Modularidad, CM = Capacidad para ser modificado, CP = Capacidad para ser probado, y R = Reusabilidad.

Figura 15: Atributos de mantenibilidad según MANTuS
Fuente: (Erazo Martínez et al., 2016)

Para que un producto de software sea mantenible debe cumplir con ciertas cualidades desde su construcción, es así que el modelo de referencia detalla una serie de prácticas base como se puede observar en la figura 23, junto a un código de identificación, una descripción y a qué ítem de la figura 24 contribuyen al ser implementadas para el cumplimiento de las subcaracterísticas.

Prácticas base
DEV-MANT.4.PB1: Utilizar comentarios. Hacer uso de comentarios adecuados en el código fuente. [Resultados: a, c, d] **DEV-MANT.4.PB2: Describir propósitos.** Explicar el propósito de las funciones, subrutinas, variables y constantes. [Resultados: a, c, d] **DEV-MANT.4.PB3: Verificar los flujos de control.** Incluir solamente los flujos de control necesarios. [Resultados: b, c, e, f] **DEV-MANT 4.PB4: Organizar código.** El código fuente debe ser indentado. [Resultado: c, d] **DEV-MANT.4.PB5: Asignar nombres claros.** Dar nombres descriptivos a las variables. [Resultado: c, d] **DEV-MANT.4.PB6: Controlar abreviaciones.** Hacer poco uso de abreviaciones. [Resultado: c, d] **DEV-MANT.4.PB7: Controlar sentencias.** Evitar sentencias largas en el código fuente, es mejor mantener las líneas de código cortas, aunque esto implica separar las sentencias en múltiples líneas. [Resultado: c, d] **DEV-MANT.4.PB8: Controlar paréntesis.** Hacer correcto uso de los paréntesis para mejorar la facilidad de lectura de las expresiones aritméticas y lógicas. [Resultado: c, d] **DEV-MANT.4.PB9: Determinar funcionalidades.** Implementar solamente las funcionalidades necesarias. [Resultados: b, c, e, f] **DEV-MANT.4.PB10: Dividir métodos.** Mantener métodos simples, es decir, dividir los métodos que tengan muchas condiciones. [Resultados: b, c, e, f] **DEV-MANT.4.PB11: Controlar tamaño.** Evitar que el tamaño del código crezca innecesariamente. [Resultados: c, f] **DEV-MANT.4.PB12: Controlar nivel de anidación.** Evitar las estructuras de control anidadas innecesarias, ya que estas son más complejas que las estructuras de control secuenciales. [Resultados: b, c, e, f, g] **DEV-MANT.4.PB13: Evitar duplicación.** Detectar código duplicado, extrayéndolo en un nuevo procedimiento y reemplazando todas las instancias del código duplicado por llamadas al nuevo procedimiento. [Resultados: b, e, f, i] **DEV-MANT.4.PB14: Definir estándares.** Tener un conjunto de estándares de programación en la escritura de código para evitar la individualidad entre los programadores. [Resultados: d, h] **DEV-MANT.4.PB15: Utilizar convenciones.** Hacer uso de convenciones estándar para el nombrado de paquetes, clases, métodos y variables. [Resultados: d, h] **DEV-MANT.4.PB16: Controlar tamaño de unidad.** Las piezas de funcionalidad de más bajo nivel deben mantenerse pequeñas para que sean centradas y fáciles de entender. [Resultados: b, c, e, f] **DEV-MANT.4.PB17: Verificar trazabilidad.** Verificar periódicamente que los artefactos de una etapa sean consistentes con artefactos de la etapa anterior. [Resultados: c, j, k] **DEV-MANT.4.PB18: Actualizar artefactos.** Actualizar los artefactos cuando se presenten cambios. [Resultados: c, j, k]
Producto de trabajo de salida
PTS-06 Unidad de software que considera la mantenibilidad [Resultados: a, b, c, d, e, f, g, h, i, j, k]
PTS-04 Registro de trazabilidad [Resultados: j, k]

Figura 16: Prácticas base durante la construcción
Fuente: (Erazo Martínez et al., 2016)

		CA	M	CM	CP	R
a)	Las unidades de software cuentan con buenos comentarios.					
b)	Las unidades de software tienen baja complejidad.					
c)	Las unidades de software son fáciles de entender.					
d)	Las unidades de software cuentan con alta facilidad de lectura.					
e)	Las unidades de software son simples.					
f)	Las unidades de software tienen el tamaño adecuado.					
g)	Las unidades de software cuentan con bajo nivel de anidación.					
h)	Las unidades de software tienen un estándar definido.					
i)	Las unidades de software evitan la duplicación.					
j)	Las unidades de software son consistentes.					
k)	Las unidades de software cuentan con trazabilidad.					

Figura 17: Check list para cumplimiento de subcaracterísticas de mantenibilidad
Fuente: (Erazo Martínez et al., 2016)

1.13.2 Proceso de validación

Para el proceso de verificación de la mantenibilidad según el marco de referencia en cuestión, deberá verificar lo siguiente en su construcción:

Paso 1: Definición de las prácticas base evidenciadas en el código del proyecto.

Paso 2: Elaboración del check list del resultado de cumplimiento de las subcaracterísticas de mantenibilidad a partir del del grupo de atributos propuesto.

Paso 3: Elaborar la guía donde se especifiquen modificaciones a realizar en el código con el aporte de participantes, esto con el fin de comprobar lo representado en el paso anterior.

Paso 4: Recolectar los valores obtenidos por los participantes en cada actividad de la guía.

Paso 5: Tabular resultados.

Paso 6: Aplicar las fórmulas para la obtención de los porcentajes de cumplimiento.

CAPÍTULO 2

Desarrollo

Considerando la eficiencia de las metodologías ágiles, se selecciona Scrum como marco de trabajo y la característica de mantenibilidad de la ISO 25010 como complemento para lograr un resultado de calidad y adaptable a cambios.

1.1 Definición de roles

Se listan los roles y responsabilidades asignadas a cada usuario dentro del proyecto.

a) MSc. Mauricio Rea
Propietario del producto: Encargado de proporcionar los requisitos.

b) Stefania Morillo
Equipo de desarrollo: Responsable de la codificación del proyecto y el cumplimiento de los requisitos proporcionados por el propietario.

c) MSc. Antonio Quiña
Líder de proyecto: Delegado a funciones de seguimiento en cada Sprint y brindar asesoría oportuna cuando se requiera.

d) MSc. Pedro Granda
Evaluador de calidad: Encargado de la valoración y aprobación del sistema.

1.2 Matriz de planificación

Se distribuye el tiempo de desarrollo conforme a la metodología Scrum, en este caso, se planifica llevar a término el proyecto en 4 Sprints, cada uno durará no más de tres semanas y emplearán 14 horas semanales como se observa en la siguiente tabla:

TABLA 7: Planificación de Sprints

Sprint	Fecha Inicio	Fecha Fin	Tiempo (horas)
1	03/08/2020	14/08/2020	28
2	17/08/2020	28/08/2020	28

3	31/08/2020	18/09/2020	42
4	21/09/2020	09/10/2020	42

Fuente: Propia

1.3 Detalle de Sprints

1.3.1 Sprint 1

Establece la fase inicial del proyecto, donde se socializa las expectativas acerca del mismo y se define el rol que tendrá cada miembro del team.

Los puntos tratados se documentan para el posterior análisis de donde se podrá obtener los requerimientos, historias de usuario y casos de uso.

La planificación de este primer Sprint abarca dos semanas (28 horas), teniendo 03/08/2020 como fecha inicial y 14/08/2020 como fecha final. El detalle de las actividades se describe en la tabla a continuación:

TABLA 8: Actividades Sprint 1

Actividad	Fase	Tarea	Estado	Duración (horas)
Planificación	Planificación	Elaborar la matriz de planificación del Sprint 1.	Realizado	2
Reunión Scrum Team	Desarrollo	El product owner brinda la información del proyecto y se establece el propósito del mismo.	Realizado	2
		Documentación de requisitos en alto nivel.	Realizado	3
		Identificación del equipo y asignación de roles.	Realizado	1
		Definición de presupuesto.	Realizado	2
Toma de requisitos	Desarrollo	Análisis, especificación y documentación de requisitos funcionales.	Realizado	3
		Análisis, especificación y documentación de requisitos no funcionales.	Realizado	2
Elaboración de historias de usuario	Desarrollo	Elaboración de historias de usuario generales.	Realizado	3
		Elaboración de historias de usuario específicas.	Realizado	3

Product backlog	Desarrollo	Se define el product backlog para el cumplimiento de las historias de usuario.	Realizado	3
Casos de uso	Desarrollo	Definir los casos de uso	Realizado	3
Reunión de revisión	Desarrollo	Revisar el avance del desarrollo en base a lo planificado.	Realizado	1
			Total	**28**

Fuente: Propia

1.3.2 Sprint 2

Se acuerda aspectos específicos para la construcción el del proyecto tales como la arquitectura y herramientas a utilizar, todo se documenta y una vez listo el entorno de desarrollo se da paso al inicio de la codificación.

Este Sprint tiene una duración de dos semanas (28 horas), empezando el 17/08/2020 y concluyendo el 28/08/2020. El detalle de las actividades se describe en la tabla 9.

TABLA 9: Actividades Sprint 2

Actividad	Fase	Tarea	Estado	Duración (horas)
Planificación	Planificación	Elaborar la matriz de planificación del Sprint 2.	Realizado	2
Diagrama de flujo	Desarrollo	Análisis del diagrama de flujo	Realizado	1
		Elaboración del diagrama de flujo	Realizado	3
Arquitectura de Software	Desarrollo	Definición de la arquitectura del módulo	Realizado	1
		Definición de herramientas para el desarrollo	Realizado	1
		Análisis y documentación de herramientas seleccionadas	Realizado	3
Herramientas de desarrollo	Desarrollo	Instalación del sistema operativo	Realizado	3
		Instalación del IDE, servidor de aplicaciones y motor de base de datos	Realizado	2
Entorno de desarrollo	Desarrollo	Configuración de herramientas	Realizado	3
		Creación de base de datos y subida del backup	Realizado	1

		Creación del proyecto y clonación en el equipo local desde GitHub	Realizado	1
Inicio de la codificación	Desarrollo	Reconocimiento y familiarización con el sistema	Realizado	3
		Creación de las entidades de base de datos en el proyecto	Realizado	1
		Análisis de prototipos e integración de pantallas	Realizado	2
Reunión de revisión	Desarrollo	Revisar el avance del desarrollo en base a lo planificado.	Realizado	1
			Total	**28**

Fuente: Propia

1.3.3 Sprint 3

Continúa el desarrollo de la funcionalidad del módulo durante tres semanas (42 horas), desde el 31/08/2020 hasta el 18/09/2020.

TABLA 10: Actividades Sprint 3

Actividad	Fase	Tareas	Estado	Duración (horas)
Planificación	Planificación	Elaborar la matriz de planificación del Sprint 3.	Realizado	2
Pantalla Inicio para gestión de plantillas	Desarrollo	Modelar y agregar la página principal para gestión de plantillas	Realizado	3
		Modelar la interfaz de la página para creación de plantillas desde cero	Realizado	2
		Escribir el algoritmo para creación de plantillas desde cero	Realizado	4
		Listar las plantillas existentes	Realizado	2
		Codificar las funciones restantes del CRUD	Realizado	5
		Probar la funcionalidad de cada componente en la pantalla inicio	Realizado	2
		Verificar en base de datos los cambios esperados al ejecutar alguna opción del CRUD	Realizado	2
		Corregir errores	Realizado	4

		Agregar en la lista de proyectos la opción "crear plantilla" para cada uno de ellos	Realizado	1
		Modelar el formulario que se mostrará al llamar la acción del botón "crear plantilla"	Realizado	1
Generación de plantilla a partir de un proyecto		Codificar el método de creación de plantilla a partir de proyecto	Realizado	6
		Probar la funcionalidad	Realizado	2
		Verificar que los registros se inserten correctamente en base de datos	Realizado	1
		Corregir errores	Realizado	4
Reunión de revisión	Desarrollo	Revisar el avance del desarrollo en base a lo planificado.	Realizado	1
			Total	**42**

Fuente: Propia

1.3.4 Sprint 4

Comprende la etapa final del proyecto, donde se cumple con los últimos requerimientos, se realizan las pruebas y se valora el producto final.

Tiene una duración de tres semanas (42 horas), iniciando el 21/09/2020 y finalizando el 09/10/2020.

TABLA 11: Actividades Sprint 4

Actividad	Fase	Tareas	Estado	Duración (horas)
Planificación	Planificación	Elaborar la matriz de planificación del Sprint 4.	Realizado	2
Generación de proyecto a partir de una plantilla	Desarrollo	Modelar el formulario para creación de proyectos a partir de plantillas	Realizado	1
		Analizar las posibilidades para reajuste de fechas en proyectos con tiempo de duración distintos	Realizado	2
		Construir el algoritmo para el reajuste	Realizado	6
		Codificar el método de creación de proyecto a partir de plantilla	Realizado	6
		Probar la funcionalidad	Realizado	2

		Verificar que los registros se inserten correctamente en base de datos	Realizado	2
		Corregir errores	Realizado	6
Pruebas finales de funcionamiento y resultados	Desarrollo	Elegir un método de pruebas aplicable	Realizado	1
		Ejecutar las pruebas	Realizado	4
		Analizar y evaluar de resultados	Realizado	4
		Ultimar detalles	Realizado	2
		Elaborar conclusiones	Realizado	2
		Elaborar recomendaciones	Realizado	2
			Total	**42**

Fuente: Propia

1.4 Requisitos

A continuación, en las tablas 12, 13 y 14 se describen los requisitos funcionales, no funcionales y se definen los indicadores de mantenibilidad alcanzables.

1.4.1 Requisitos Funcionales

TABLA 12: Requisitos Funcionales

Código	Nombre	Tipo	Prioridad	Detalle
RF1	Creación de proyectos	Requisito	Alta	La generación de plantillas puede realizarse bajo los siguientes escenarios: - Plantillas desde cero - Plantillas a partir de proyectos - Proyectos a partir de plantillas – Plantillas generales.
RF2	Gestión de plantillas	Requisito	Alta	El usuario podrá visualizar, editar o eliminar sus plantillas
RF3	Plantillas desde cero	Requisito	Alta	La creación de una plantilla partiendo de cero no requiere información muy específica
RF4	Plantillas a partir de proyectos	Requisito	Alta	La plantilla creada debe ser una copia exacta del proyecto exceptuando los integrantes

RF5	Proyectos a partir de plantillas	Requisito	Alta	Para la creación del proyecto se despliega un formulario donde el usuario proporciona aspectos específicos, tales como el nombre, objetivo general, observaciones, fecha de inicio y fecha fin.
		Requisito	Alta	Si las nuevas fechas difieren a las del proyecto original, se hace un reajuste en los tiempos de ejecución de actividades y subactividades, de modo que el proyecto calce en el nuevo periodo de duración
RF6	Plantillas Generales	Requisito	Alta	El administrador podrá crear plantillas generales que serán visibles para todos los usuarios
RF7	Listado de plantillas	Requisito	Alta	Podrán visualizarse dos tipos de plantillas: - Plantillas generales - Plantillas propias del usuario

Fuente: Propia

1.4.2 Requisitos No Funcionales

TABLA 13: Requisitos No Funcionales

Código	Nombre	Tipo	Prioridad	Detalle
RNF1	Acceso al módulo	Requisito	Alta	El módulo de plantillas será accesible mediante el ingreso al SIAD
RNF2	Mantenibilidad ISO25010	Requisito	Alta	El módulo deberá cumplir con la subcaracterística de modularidad
				El módulo tendrá capacidad de ser modificado
				El módulo ser capaz de ser probado

Fuente: Propia

1.4.3 Definición de indicadores de mantenibilidad ISO/IEC 25010

Se muestra en la tabla 14 las 5 subcaracterísticas de la mantenibilidad y el requisito para el cumplimiento de cada una:

TABLA 14: Indicadores de mantenibilidad

Subcaracterística	Requisito
Capacidad de ser analizado	Debe ser fácil de leer y entender, tener niveles bajos de anidación y otros aspectos de esta naturaleza.
Modularidad	Contar con la capacidad de que, al modificar un elemento, otros no se vean afectados.
Capacidad de ser modificado	En caso de necesitar cambios, la ejecución de estos no debe afectar la funcionalidad del resto del sistema.
Reusabilidad	El código debe poder utilizarse para la elaboración de nuevos proyectos, sin mayor dificultad de adaptación.
Capacidad de ser probado	Debe ser adaptable a criterios de evaluación que permitan medir la el cumplimiento de los requerimientos.

Fuente: Propia

1.5 Cartillas de historias de usuario

1.5.1 Historia de Usuario N° 1

En la tabla 15 se muestra la primera historia de usuario donde se detallan todas las pantallas que se crearán para que el docente pueda gestionar de forma ordenada sus plantillas.

TABLA 15: HU1 | Diseño de vistas

Usuario: Docente	**Nombre de la historia:** Diseño de las vistas
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 6	**Sprint asignado:** 2
Programador responsable: Stefania Morillo	

Descripción

Creación de las diferentes vistas del módulo para que el usuario pueda gestionar sus plantillas.

- Las plantillas existentes se muestran a manera de listas en la página principal, una general para todos los docentes y otra exclusiva del usuario.
- En la misma página cada plantilla tiene opciones para visualizar, editar, eliminar o crear proyecto partiendo de ella.
- La creación y edición de plantillas usa 4 vistas diferentes: datos generales, objetivos específicos, actividades y subactividades.
- Para la creación de proyecto a partir de plantilla se despliega un cuadro de diálogo que permite proporcionar la información específica del caso.

Observaciones

- El docente no puede modificar o borrar una plantilla general, únicamente visualizarla o crear un proyecto a partir de la misma.
- Se usa la pantalla principal de proyectos para agregar una opción adicional que permita la creación de plantillas a partir de cualquiera de ellos.

Fuente: Propia

Entregable:

En la figura 15 se observa los listados de plantillas generales y por usuario en la página de inicio, con las respectivas opciones para cada elemento de la lista.

Figura 18: Página inicio de plantillas
Fuente: Propia

1.5.2 Historia de Usuario N° 2

Se definen los pormenores en cuanto a la creación de plantillas, mismos que están descritos en la tabla 16.

TABLA 16: HU2 | Creación de plantillas

Usuario: Docente	**Nombre de la historia:** Creación de plantillas
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 5	**Sprint asignado:** 3
Programador responsable: Stefania Morillo	
Descripción Si se parte de cero, la creación se va realizando a modo de maestro detalle para que el usuario pueda proveer los datos de la plantilla de forma ordenada, así: • Se muestra inicialmente la vista donde se proporciona los datos generales del proyecto, una vez llenos todos los campos requeridos pasa a la siguiente pantalla. • Se agregan los objetivos específicos que se requieran y se pasa a la siguiente pantalla. • Se muestra una lista de los objetivos agregados anteriormente, se selecciona uno, se agrega las actividades que se llevarán a cabo para cumplirlo y se pasa a la siguiente pantalla. • Se muestra una lista de las actividades agregadas anteriormente, se selecciona una, se agrega las subactividades en caso de ser necesarias y finalmente concluye la creación. Si se parte desde un proyecto, se crea internamente una copia del mismo y se muestra al usuario un mensaje para que pueda confirmar que fue creada con éxito, en cuyo caso estará disponible en la página inicio de plantillas.	
Observaciones: Sin observaciones	

Fuente: Propia

Entregable:

La imagen 16 muestra las pantallas para creación de una plantilla si se parte de cero, en cuyo caso, al ser un maestro detalle se deberá proporcionar inicialmente los datos generales del proyecto (cabecera) para luego continuar con los datos específicos (detalle).

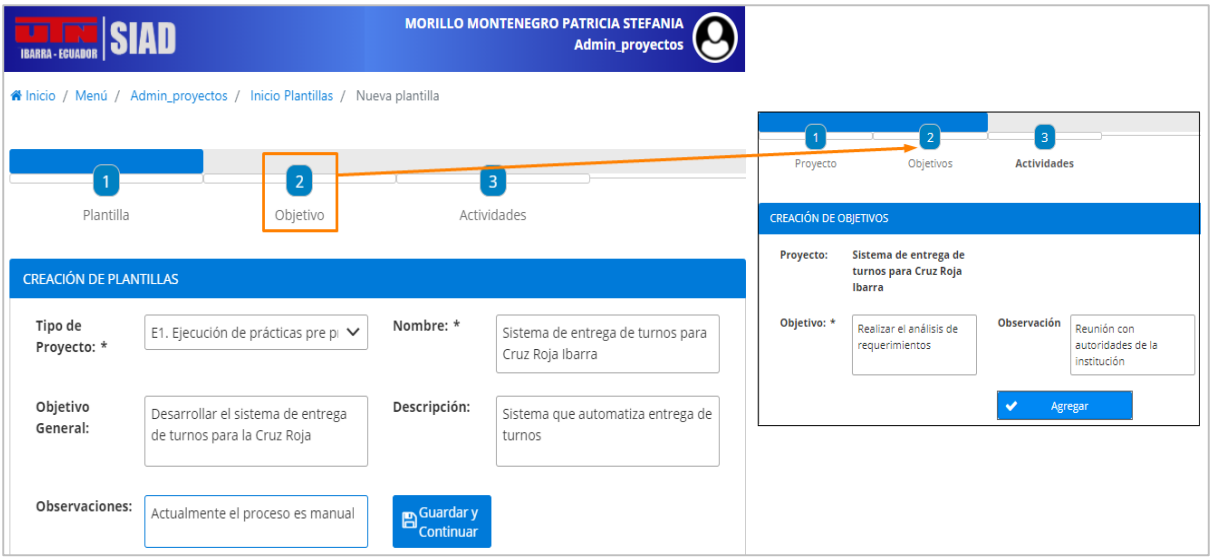


Figura 19: Creación de plantilla desde cero
Fuente: Propia

1.5.3 Historia de Usuario N° 3

En la tabla 17 se describe el mecanismo para la edición de una plantilla.

TABLA 17: HU3 | Edición de plantillas

Usuario: Docente	**Nombre de la historia:** Edición de plantillas
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 1	**Sprint asignado:** 3
Programador responsable: Stefania Morillo	
Descripción Al igual que en la creación, para editar el usuario va de pantalla en pantalla, con la diferencia que tendrá cargado todos los datos existentes en la plantilla para que modifique únicamente lo que crea pertinente.	
Observaciones: Sin observaciones	

Fuente: Propia

Entregable:

Para la edición de plantillas se cargan los datos de la misma en un cuadro de diálogo como en la figura 17.

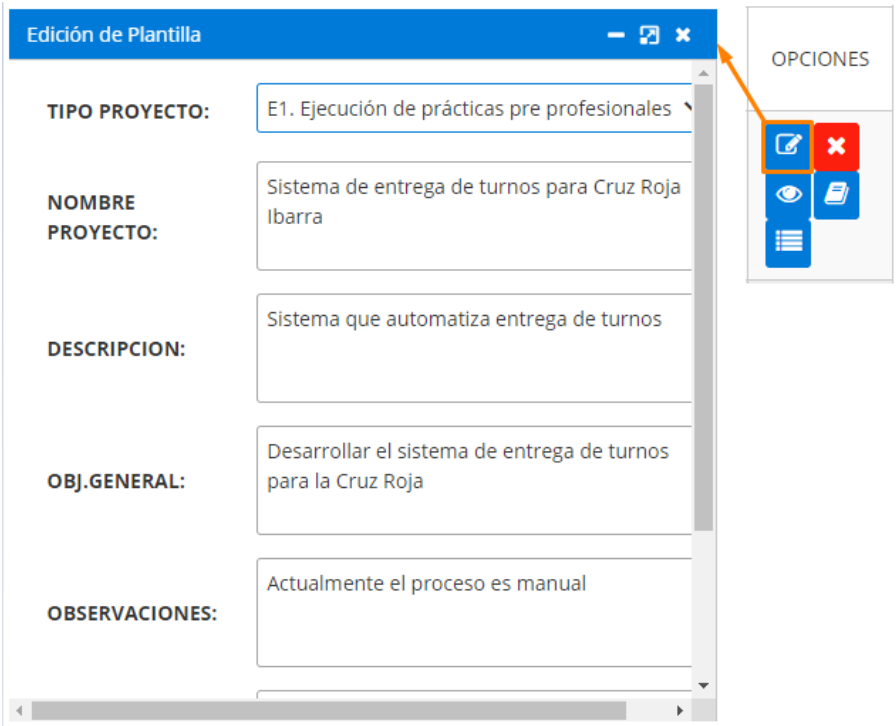


Figura 20: Edición de plantillas
Fuente: Propia

1.5.4 Historia de Usuario N° 4

En la tabla 18 se especifica cómo serán eliminadas las plantillas.

TABLA 18: HU4 | Eliminación de plantillas

Usuario: Docente	**Nombre de la historia:** Eliminación de plantillas
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 1	**Sprint asignado:** 3
Programador responsable: Stefania Morillo	
Descripción La eliminación se realiza en cascada para que el usuario no tenga que perder tiempo eliminando cada registro derivado de la plantilla, sino que baste con borrarla desde la cabecera.	
Observaciones Se muestra un mensaje de confirmación antes de eliminar definitivamente la plantilla	

Fuente: Propia

Entregable:

Previo a eliminar la plantilla, se despliega el siguiente cuadro de diálogo para seguridad del usuario.

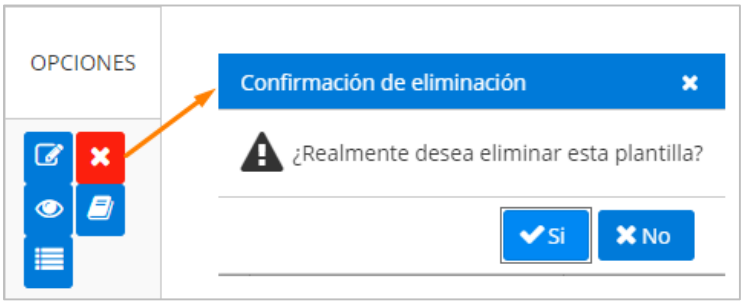


Figura 21: Eliminación de plantilla
Fuente: Propia

1.5.5 Historia de Usuario N° 5

En la tabla 19 se muestran las especificaciones al momento de crear un nuevo proyecto usando como punto de partida una de las plantillas disponibles.

TABLA 19: HU5 | Creación de proyecto desde plantilla

Usuario: Docente	**Nombre de la historia:** Creación de proyecto a partir de plantilla
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 2	**Sprint asignado:** 4
Programador responsable: Stefania Morillo	
Descripción Al partir la creación desde una plantilla, el usuario proporciona los datos específicos del proyecto, tales como: nombre, objetivo general, observaciones y las fechas de inicio y finalización que tenga previstas. Si la duración del nuevo proyecto no coincide con la de la plantilla, se hace un reajuste de fechas, de modo que todos los objetivos, actividades y subactividades se realicen conforme al nuevo lapso establecido, es decir que hay un aumento o disminución en el tiempo de ejecución de cada elemento según corresponda.	
Observaciones: En el reajuste de fechas, los fines de semana no se toman en cuenta.	

Fuente: Propia

Entregable:

El usuario proporciona únicamente datos específicos del proyecto como se muestra en la imagen.

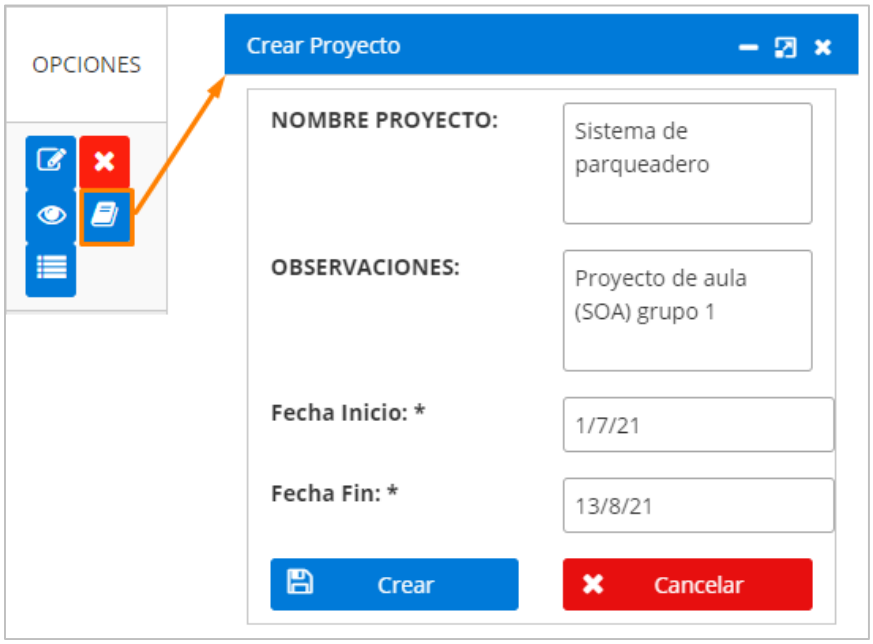


Figura 22:Creación de proyecto a partir de plantilla
Fuente: Propia

1.5.6 Historia de Usuario N° 6

Las funciones del usuario administrador se describen en la tabla 20:

TABLA 20: HU6 | Plantillas generales

Usuario: Administrador	**Nombre de la historia:** Plantillas generales
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 2	**Sprint asignado:** 4
Programador responsable: Stefania Morillo	
Descripción Desde la administración, se crean, editan o eliminan plantillas generales, es decir aquellas que se consideren básicas para cualquier usuario o se utilicen con mayor frecuencia.	
Observaciones:	

Fuente: Propia

1.6 Casos de Uso

1.6.1 Caso 1 - Plantillas generales

En la figura 20 se muestran los casos de usos que pueden darse en plantillas generales, tanto para el usuario administrador como para el docente.

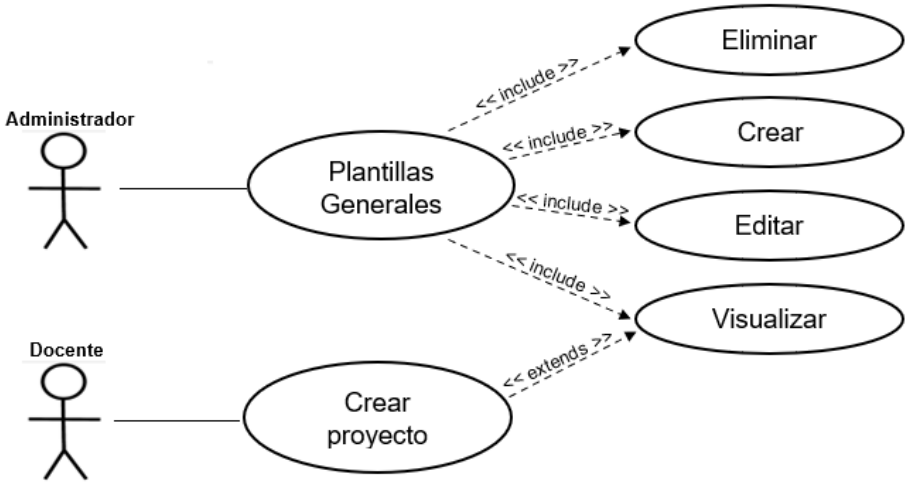


Figura 23: Caso de uso de plantillas generales
Fuente: Propia

1.6.2 Caso 2 - Plantillas propias del usuario

En la figura siguiente se evidencia las posibilidades que dispone el usuario docente en cuanto a la gestión de sus plantillas.

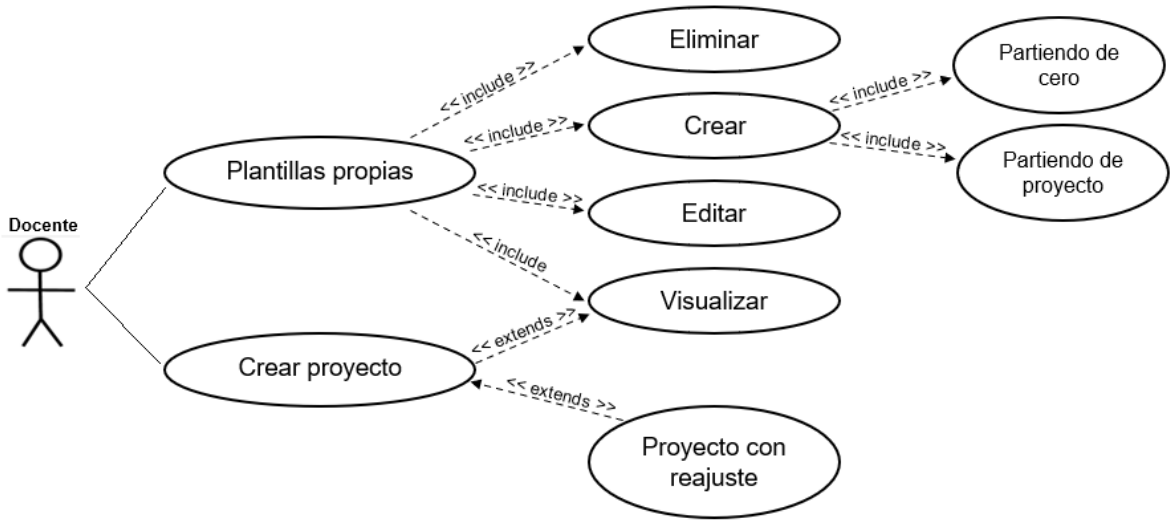


Figura 24:Caso de uso de plantillas propias del usuario
Fuente: Propia

1.7 Arquitectura de Software

El proyecto se desarrolla bajo arquitectura MVC (Modelo, vista, controlador), empleando diagramas UML para la representación de casos de uso y las siguientes herramientas para el desarrollo:

- IDE Eclipse JEE v.2021-03 como entorno de desarrollo.
- Servidor de aplicaciones Wildfly v22.0.1 Final
- Sistema Operativo Windows
- JSF y Primefaces.

1.8 Pruebas de Funcionamiento

1.8.1 Pruebas Funcionales

TABLA 21: Pruebas funcionales

Código Requisito	Código Prueba	Objetivo de la Prueba	Apreciaciones
RF1	PF1	Corroborar que el módulo permita crear plantillas desde cero, a partir de proyectos y a partir de plantillas, además de las generales.	Pueden generarse plantillas bajo los 4 escenarios.
RF2	PF2	Probar que los usuarios puedan gestionar correctamente sus plantillas.	Las plantillas se visualizan, editan y eliminan de forma adecuada.
RF3	PF3	Verificar que la creación de plantillas desde cero no requiera información muy específica.	La información que requiere la creación de una plantilla desde cero es la básica para la posterior creación de cualquier proyecto.
RF4	PF4	Comprobar que las plantillas creadas a partir de proyectos sean la copia exacta del proyecto.	Se genera una copia del proyecto en los campos que coinciden en las dos tablas.
RF5	PF5	Verificar que, ingresando la información básica del nuevo proyecto, el módulo cree una plantilla con reajuste en las fechas de su ejecución.	El nuevo proyecto presenta reajuste en el tiempo de duración de actividades y subactividades en base a las fechas de inicio y fin del proyecto proporcionadas.

RF6	PF6	Comprobar que el administrador pueda crear plantillas generales que estén disponibles para todos los usuarios	El usuario tiene acceso a las plantillas creadas por el administrador.
RF7	PF7	Confirmar que las plantillas se listen por: generales y propias del usuario.	El usuario puede visualizar los dos tipos de listas en su pantalla de inicio de plantillas.

Fuente: Propia

1.8.2 Informe de Pruebas Funcionales

TABLA 22: Informe de pruebas funcionales

Código Prueba	# Casos de Prueba	Nivel de aceptación	Observaciones
PF1	8	100%	Se efectuaron dos casos de prueba por cada escenario de creación de plantillas.
PF2	4	100%	Se hicieron dos casos de prueba por tipo de usuario.
PF3	2	100%	Sin observaciones
PF4	2	100%	Sin observaciones
PF5	5	100%	Se presentó error al violar la restricción de campos requeridos para la creación de proyecto, debido a campos nulos en ciertas plantillas, mismo que fue corregido. Se ingresaron distintos rangos de fechas para verificar el ajuste.
PF6	2	100%	Sin observaciones
PF7	2	100%	Sin observaciones

Fuente: Propia

1.9 Verificación de Mantenibilidad con MANTuS

1.9.1 Paso 1.

A continuación, se detalla la lista de prácticas base presentes, cada una con una breve descripción de cómo se ve evidenciada en el código:

- **DEV-MANT.4.PB1. -** Se hace uso de comentarios a lo largo de la codificación del módulo.

- **DEV-MANT.4.PB2. -** Los comentarios describen la función de los métodos empleados y de las sentencias o variables que pudieran no ser tan fáciles de interpretar por personas ajenas al equipo responsable del desarrollo.

En la figura 25 se muestran comentarios en el código como evidencia de las dos prácticas anteriores.

```
// lectura de las actividades en objetivos de plantilla
List<PltActividad> actividadesObjetivo = findAllPltActividadesByObjetivo(objetivo.getId
for (PltActividad actividad : actividadesObjetivo) {
    // días que dura originalmente la actividad
    int duracionActividad = obtenerDiasLaborablesEntreFechas(actividad.getFechaInicio()
            actividad.getFechaFin());
    // nueva duración en base al periodo del proyecto
    int nuevaDuracionAct = obtenerAjuste(diasTotales, duracionActividad);

    // número de días hasta la fecha inicio de la actividad
    int diasInicioAct = obtenerDiasLaborablesEntreFechas(proyecto.getFechaInicio(),
            actividad.getFechaInicio());
```

Figura 25: Comentarios en código fuente
Fuente: Propia

- **DEV-MANT 4.PB4. -** El código fuente está indentado, de modo que se puede identificar de mejor manera dónde inician o finalizan las clases, métodos y etiquetas o su contenido, tal como se muestra en la figura 26.

```
<div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav">
        <p:menubar style="background-color:#E4E5E5 ; border: #CAE1E8;">
            <p:menuitem value="Nueva Plantilla" title="Nueva Plantilla"
                style="font-size: 16px;" icon="fa fa-plus-square"
                action="#{plnBeanPlanificacionDocente.NewPlantilla()}" />
            <p:menuitem value="Actualizar Plantillas"
                style="font-size: 16px;" title="Actualizar Plantillas"
                icon="fa fa-refresh" update="form2"
                actionListener="#{plnBeanPlanificacionDocente.actRecargar
        </p:menubar>
    </ul>
</div>
```

Figura 26: Código Indentado
Fuente: Propia

- **DEV-MANT.4.PB5. -** Los nombres de las variables describen claramente a qué hacen referencia (figura 27).

```
private List<PltProyecto> listaPlantillas;
private int idPltProyectoSeleccionado;
private String nombreProyecto;
private Date fechaInicio;
private Date fechaFin;
```

Figura 27: Nombres de variables

Fuente: Propia

- **DEV-MANT.4.PB6. -** Se hace uso de abreviaciones únicamente cuando se trata de prefijos que se han estandarizado a lo largo del proyecto**.** En el ejemplo de la figura 28 se puede observar “plt” como prefijo o sufijo, haciendo referencia a que son las utilizadas por el módulo de plantillas.

```
//PLANTILLAS
private PltProyecto edPlantilla;
private int id_plt_proyecto;
private String nombrePlt;
private String descripcionPlt;
private String observacionesPlt;
private Date fechaInicioPlt;
private Date fechaFinPlt;
```

Figura 28: Uso de abreviaciones
Fuente: Propia

- **DEV-MANT.4.PB7. –** Se dividen las sentencias largas para mayor de facilidad de lectura (figura 29).

```
try {
    nuevaPltSubAct = modelPlanificacion.agregarPltSubAct(actSeleccionada,
            nombrePltSubAct, descripcionPltSubAct, observacionesPltSubAct,
            validado);
```

Figura 29: División de líneas de código
Fuente: Propia

- **DEV-MANT.4.PB8. –** Como se ve en la figura 30, se hace uso de los paréntesis para mayor claridad en la operación.

```
int diasAjuste = 0;
diasAjuste = Math.round((diasTotalesProy * cantDias) / diasTotalesPlt);
if (diasAjuste == 0)
    diasAjuste++;
```

Figura 30: Uso correcto de paréntesis
Fuente: Propia

- **DEV-MANT.4.PB9. –** No se ha implementado funcionalidades innecesarias.
- **DEV-MANT.4.PB10. –** Existen métodos que llaman a otros dentro de la misma clase, como en el caso de la figura 31, donde se señalan los métodos pequeños que están siendo llamados desde uno de mayor tamaño.

```
// nueva fecha inicio en base al periodo del proyecto
int nuevoInicioAct = obtenerAjuste(diasTotalesPlt, diasTotalesProy, diasInicioAct);

Date fechaInicioActividad = calcularFecha(proyecto.getFechaInicio(), nuevoInicioAct);
Date fechaFinActividad = calcularFecha(actividad.getFechaInicio(), nuevaDuracionAct);
```

Figura 31: División de métodos
Fuente: Propia

- **DEV-MANT.4.PB11. –** El tamaño del código es controlado.
- **DEV-MANT.4.PB12. –** No se usan más de 3 ciclos anidados.
- **DEV-MANT.4.PB13. –** No se ha detectado código duplicado.
- **DEV-MANT.4.PB15. –** Se maneja un estándar al momento de nombrar paquetes y más componentes, como muestra la figura 32.

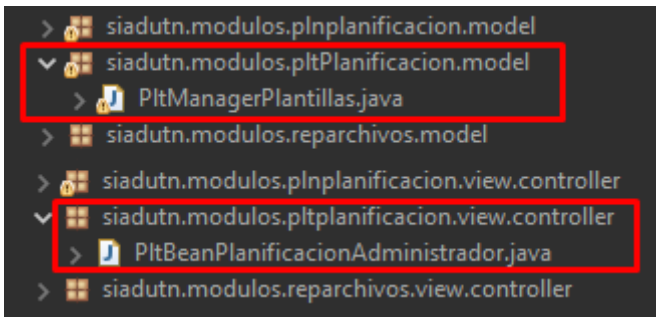


Figura 32: Nombrado de paquetes con estándar
Fuente: Propia

- **DEV-MANT.4.PB17. –** Tiene consistencia. con artefactos de la etapa anterior
- **DEV-MANT.4.PB18. –** En caso de presentarse cambios, los artefactos del proyecto se actualizan.

1.9.2 Paso 2.

De acuerdo a las pruebas base empleadas, se obtiene como resultado la tabla 23, donde CA representa la capacidad de ser analizado, M la modularidad, CM la capacidad de ser modificado, CP la capacidad de ser probado y R la reusabilidad.

TABLA 23: Cumplimiento de subcaracterísticas de mantenibilidad

Las unidades de software:	CA	M	CM	CP	R
Cuentan con buenos comentarios.	X		X		X
Tienen baja complejidad.	X	X	X	X	X
Son fáciles de entender.	X		X	X	X

Son fáciles de leer.	X		X	X	X
Son simples.	X	X	X	X	X
Tienen el tamaño adecuado.	X	X	X	X	X
Cuentan con bajo nivel de anidación.	X		X	X	
Tienen un estándar definido.	X		X		
Evitan la duplicación.	X		X	X	
Son consistentes.	X		X		
Cuentan con trazabilidad.	X		X		

Fuente: Propia

1.9.3 Paso 3.

Con el objetivo de corroborar la efectividad del uso buenas prácticas en cuanto a mantenibilidad, se pide la colaboración de voluntarios ajenos al equipo de desarrollo para hacer cambios en el módulo, medir sus tiempos de ejecución y si logran realizar lo que se les pide.

Para dicho fin primero se familiariza a los participantes con la funcionalidad del módulo. Una vez que tengan conocimiento se les entrega una guía previamente diseñada donde están descritos los requerimientos, que para este caso fueron los siguientes:

TABLA 24: Requerimientos para pruebas

Identificador	Requerimiento
RP1	La creación de plantillas desde creo permite ingresar una fecha final menor a la inicial, corrija la contradicción.
RP2	Muestre la lista de plantillas en la respectiva página de inicio.
RP3	Controle que no se puedan dejar campos vacíos en la edición de plantillas.
RP4	Cree un mensaje de información cuando un proyecto desde plantilla se cree exitosamente.

RP5	Agregue una opción de gestión de plantillas en su página inicio.

Fuente: Propia

1.9.4 Paso 4.

La guía permite al participante registrar si logró o no localizar la parte del código que necesita y qué tiempo le tomó hacerlo. Del mismo modo si logró cumplir con los requerimientos y cuánto tardó en su ejecución.

1.9.5 Paso 5.

Los valores recolectados en el paso anterior son tabulados y totalizados.

1.9.6 Paso 6.

El total de aciertos y los tiempos de ejecución, permitirán emplear las fórmulas de cuyo resultado se obtendrá la conclusión final.

Para encontrar la tasa de éxito se aplica la siguiente fórmula:

$$TEF = 1 - \frac{A}{B}$$

Donde A representa al número de fallos cuyas causas no se han encontrado y B al número de fallos registrados. El resultado varía entre 0 y 1. Se considera exitoso mientras más cercano a 1 sea (Erazo Martínez et al., 2016).

CAPÍTULO 3

Resultados

Tomando en cuenta que la mantenibilidad es esencial durante el ciclo de vida de un software, ya que ayuda a reducir costos y esfuerzo en cualquiera de sus etapas, se toma el proceso de construcción de software propuesto por el modelo de referencia MANTuS, obteniendo:

3.1 Resultados por participante

TABLA 25: Resultados participante 1

Identificador de requerimiento	Modificación		Análisis	
	¿Correcto?	Duración	¿Correcto?	Duración
RP1	Sí	00:15:09	Sí	00:03:51
RP2	Sí	00:08:40	Sí	00:04:22
RP3	Sí	00:01:28	Sí	00:01:34
RP4	Sí	00:02:54	Sí	00:05:03
RP5	Sí	00:02:23	Sí	00:00:51
Total	5	00:30:34	5	00:15:41
Observaciones:				

Fuente: Propia

TABLA 26: Resultados participante 2

Identificador de requerimiento	Modificación		Análisis	
	¿Correcto?	Duración	¿Correcto?	Duración
RP1	Sí	00:12:33	Sí	00:05:03
RP2	Sí	00:07:58	Sí	00:05:24
RP3	Sí	00:01:35	Sí	00:02:09
RP4	Sí	00:02:27	Sí	00:05:43
RP5	Sí	00:03:08	Sí	00:01:11
Total	5	00:27:41	5	00:19:30
Observaciones:				

Fuente: Propia

TABLA 27: Resultados participante 3

Identificador de requerimiento	Modificación		Análisis	
	¿Correcto?	Duración	¿Correcto?	Duración
RP1	Sí	00:15:35	Sí	00:03:12
RP2	Sí	00:09:13	Sí	00:05:41
RP3	Sí	00:02:24	Sí	00:02:01
RP4	Sí	00:02:28	Sí	00:04:33
RP5	Sí	00:02:41	Sí	00:00:54
Total	5	00:34:21	5	00:16:21
Observaciones:				

Fuente: Propia

TABLA 28: Resultados participante 4

Identificador de requerimiento	Modificación		Análisis	
	¿Correcto?	Duración	¿Correcto?	Duración
RP1	Sí	00:17:21	Sí	00:02:18
RP2	Sí	00:08:33	Sí	00:06:11
RP3	Sí	00:01:57	Sí	00:02:05
RP4	Sí	00:02:12	Sí	00:05:16
RP5	Sí	00:03:54	Sí	00:00:56
Total	5	00:33:57	5	00:16:46
Observaciones:				

Fuente: Propia

3.2 Tasa de éxito en el encuentro de fallos (TEF)

$$TEF = 1 - \frac{A}{B}$$

$$TEF = 1 - \frac{0}{5}$$

$$TEF = 1 - 0$$

$$TEF = 1$$

CONCLUSIONES

- Al término de la investigación que fundamenta el desarrollo del proyecto se adquirió una base conceptual que influyó positivamente en la adquisición de nuevos conocimientos que fueron de gran utilidad durante la elaboración de este trabajo.

- El uso de metodologías para el desarrollo de productos de software es de gran ayuda para su ejecución ya que proporcionan una guía que permite realizar un trabajo ordenado y apegado a una planificación.

- Scrum como metodología de desarrollo provee un panorama claro del estado de un proyecto, ya que controla su progreso conforme se avanza en la planificación, por ende, beneficia a la detección y corrección temprana de errores que surgen durante el proceso, reduciendo así el riesgo al fracaso.

- Una estructura modular tiene muchas ventajas, sin embargo, es importante cuidar el nivel de división de los componentes de modo que tengan suficiente independencia, pero puedan comunicarse con los demás elementos en caso de ser necesario.

- En la actualidad existen un sinfín de estándares para todo tipo de industria, una buena elección puede impactar efectivamente el resultado de cualquier investigación y un claro ejemplo de esto es la familia de estándares ISO/IEC 25000 que están enfocadas en la búsqueda de productos de software de calidad.

- El modelo proporcionado por ISO/IEC 25010 es muy completo, ya que describe características y subcaracterísticas con que se puede mejorar la calidad del software y desde distintos enfoques.

- La característica de mantenibilidad de la ISO/IEC 25010 es un gran aporte para empresas dedicadas al mundo del desarrollo de software, esto gracias a que busca la calidad desde la etapa construcción, evitando malas prácticas, facilitando el trabajo en equipos de desarrollo y generando sobre todo un impacto económico, ya que su buena implementación puede reducir notoriamente los extensos periodos de prueba, cuidando así uno de los recursos más valiosos como es el tiempo y aumentando la productividad en el desarrollo de nuevos proyectos.

- Para que un producto sea considerado mantenible debe cumplir con ciertas subcaracterísticas que pueden evaluarse desde el código, al medir la capacidad que tenga de ser analizado, probado, modificado y reutilizado por personas que no hayan estado inmersas en el desarrollo, tal como se pudo evidenciar en el módulo de plantillas del Sistema Integrado de Actividad Docente.

RECOMENDACIONES

- Realizar una buena investigación que permita adquirir las bases de conocimiento necesarias para su uso en la elaboración de este tipo de proyectos.

- Usar una metodología ágil, ya que puede resultar mucho más conveniente al dar la posibilidad de hacer pruebas incluso desde el inicio del desarrollo.

- Se debe elegir un estándar para un proyecto, es importante tomarse el tiempo de investigar para escoger el que más se apegue a las necesidades.

- Escribir y documentar el código fuente de forma adecuada de modo que pueda facilitar el desarrollo de trabajos futuros.

- Analizar el tamaño de los módulos, si se va a trabajar con esta estructura, no deben ser muy grandes para que cumplan con el objetivo de facilitar su desarrollo, ni muy pequeño para que no pierda relevancia dentro de la estructura.

- Para la división de módulos resulta muy útil analizar que todas sus funciones vayan enfocadas hacia un mismo objetivo, de no ser así, lo más probable es que sea posible dividir más.

- Existen herramientas para medir la mantenibilidad de un producto de software, no obstante, se debe considerar el tipo de arquitectura que se maneja, ya que podría no ser compatible.

REFERENCIAS

Ahmad Khan, S., & Ahmad Khan, R. (2012). Analyzability Quantification Model of Object Oriented Design. *Procedia Technology*, *4*, 536–542. https://doi.org/10.1016/j.protcy.2012.05.085

Alicante, U. (2014). *El ciclo de vida de JSF*. 1–36. http://www.jtech.ua.es/j2ee/publico/jsf-2012-13/sesion03-apuntes.pdf

Anónimo. (2008). *Lenguajes de programación*. *1563*, 1–16.

Arak. (2012). *Desarrollo en Java EE*. http://metagdes.blogspot.com/2012/05/desarrollo-en-java-ee.html

Beltrán, F., & Aguirre, C. (2011). *Programación modular*. http://virtual.usalesiana.edu.bo/web/conte/archivos/162.pdf

Callejas Cuervo, M., Alarcón Aldan, A., & Álvarez Carreño, A. M. (2017). Software quality models, a state of the art. *Entramado*, *13*(1), 236–250. http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S1900-38032017000100236

Chen, J. C., & Huang, S. J. (2009). An empirical analysis of the impact of software development problem factors on software maintainability. *Journal of Systems and Software*, *82*(6), 981–992. https://doi.org/10.1016/j.jss.2008.12.036

CISIC-UTN. (2019). *Misión y Visión*. https://www.utn.edu.ec/fica/carreras/sistemas/?page_id=10

EcuRed. (2015). *Primefaces*. https://www.ecured.cu/Primefaces

EdTeams. (2020). *Lenguajes de prgramación con mayor demanda*. https://twitter.com/edteamlat/status/1229787424579969024

Erazo Martínez, J., Florez Gómez, A., & Pino, F. J. (2016). Generando productos software mantenibles desde el proceso de desarrollo: El modelo de referencia MANTuS Creating maintainable software products from the development process: The reference model MANTuS. *Revista Chilena de Ingeniería*, *24*(3), 420–434.

Henao, C. (2017). *Componentes básicos Java Server Faces - JSF*. http://codejavu.blogspot.com/2017/12/componentes-basicos-java-server-pages.html

Huda, M., Sharma Arya, Y. D., & Hasan Khan, M. (2015). Quantifying Reusability of Object Oriented Design: A Testability Perspective. *Journal of Software Engineering and*

Applications, *08*(04), 175–183. https://doi.org/10.4236/jsea.2015.84018

Indeed. (2020). *Aplicación web*. https://www.indeed.com/career-advice/career-development/what-is-web-application

Invarato, R. (2019). *Cliente vs Servidor*. https://jarroba.com/cliente-vs-servidor/

ISO-25000. (2019). *ISO/IEC 25010*. https://iso25000.com/index.php/en/iso-25000-standards/iso-25010?limit=3&start=6

ISOTools. (2018). *¿Por qué automatizar los procesos en tu organización?* https://www.isotools.org/2018/03/28/por-que-automatizar-los-procesos-en-tu-organizacion/

Jeferson. (2018). *Proceso o Método de desarrollo SCRUM*. https://www.codehoven.com/metodo-de-desarrollo-scrum/

Malhotra, R., & Chug, A. (2016). Software Maintainability: Systematic Literature Review and Current Trends. *International Journal of Software Engineering and Knowledge Engineering*, *26*(8), 1221–1253. https://doi.org/10.1142/S0218194016500431

Mcguire, M. (2009). Programming Language Notes. *Mathematica*.

Meza González, J. D. (2017). *Factores de mantenibilidad en el desarrollo de aplicaciones web* [Universidad Nacional de Colombia]. http://bdigital.unal.edu.co/57890/7/1026144734.2017.pdf

Muñoz, J. M. (2012). *JSF, características principales, ventajas y puntos a destacar*. https://josemmsimo.wordpress.com/2012/07/30/jsf-caracteristicas-principales-ventajas-y-puntos-a-destacar/

Narváez Flores, L. A. (2019). *Benchmarking de Sistemas ERP (Planificación de Recursos Empresariales) open source aplicado a la empresa pública Yachay*. Universidad Técnica del Norte.

Nieva, G. (2018). *Programación Modular*. https://dcodingames.com/programacion-modular-p1/

ONU. (2015). *Objetivos de desarrollo sostenible*. https://www.un.org/sustainabledevelopment/es/

Oracle. (2020). *Java Documentation*. https://docs.oracle.com/en/java/

Pacífico, C. (2004). *Conceptos de Reusabilidad de Software*. https://slideplayer.es/slide/3263874/

Pérez González, H. G., Martínez Pérez, F. E., Nava Muñoz, S. E., Núñez Varela, A. S., Vázquez Escalante, M., & Flores Saucedo, J. A. (2015). Analizando la Mantenibilidad de Software Desarrollado Durante la Formación Universitaria. *Revista Latinoamericana de Ingenieria de Software*, *3*(6), 231. https://doi.org/10.18294/relais.2015.231-236

Pérez Porto, J., & Merino, M. (2014). *Definición de plantilla*. https://definicion.de/plantilla/

Primefaces. (2020). *Primefaces showcase*. https://www.primefaces.org/showcase/index.xhtml

Rodríguez Baena, L. (2011). *Fundamentos de Programación* (p. 6). http://www.colimbo.net/documentos/documentacion/106/FPI04_Programacion_Modular_(11-12).pdf

Rodríguez, M., Pedreira, Ó., & Fernández, C. M. (2015). Certificación de la mantenibilidad del producto software. Un caso práctico. *XI Jornadas Iberoamericanas de Ingenieria de Software e Ingenieria Del Conocimiento, JIISIC 2015*, *3*(3), 93–105. https://doi.org/10.18294/relais.2015.127-134

Rodríguez, M., & Piattini, M. (2007). KEMIS: Entorno para la Medición de la Calidad del Producto Software. *Procesos y Métricas*, 163–176.

Ronche, J. (2020). *Scrum y su conexión con el Rugby*. https://www2.deloitte.com/es/es/pages/technology/articles/scrum-y-su-conexion-con-el-rugby.html

Salazar, L. (2020). *Diez comportamientos atípicos en Ágil y Scrum*. http://www.gazafatonarioit.com/2020/04/diez-comportamientos-atipicos-en-agil-y.html

Solís Cabrera, V. A. (2018). *Evaluación de un modelo de calidad para objetos de aprendizaje basado en un modelo i** [Universidad de Cuenca]. https://dspace.ucuenca.edu.ec/bitstream/123456789/29527/1/Trabajo de Titulación.pdf

Soltano, S. (2020). *Estadísticas de uso de los sistemas de gestión de contenido*. https://w3techs.com/technologies/overview/content_management

tutorialspoint. (n.d.). *Java*. https://www.tutorialspoint.com/java/index.htm

Tyson, M. (2018). *Presentación de JSF*. https://www.infoworld.com/article/3322533/what-is-jsf-introducing-javaserver-faces.html

Usman, O., Owoade, A., Abimbola, B., & Ogunsanwo, G. (2016). Introduction to computer programming. *Brit Chem Eng*, *16*(2–3), 206–210. https://doi.org/10.2307/2003721

Valenciano López, J. (2015). *Auditoría Mantenibilidad Aplicaciones Según ISO/IEC 25000*. 93. http://eprints.ucm.es/37485/1/AUDITORÍA MANTENIBILIDAD APLICACIONES SEGÚN LA ISO_IEC 25000.pdf

Vázquez, J. (2013). *Plantillas*. https://es.calameo.com/read/002851209bd60ee28bd84

Viveros, C. A. (2010). *Capítulo 3. JavaServer Faces*. 23. http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/viveros_s_ca/capitulo3.pdf