DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

TRABAJO DE TITULACIÓN, PREVIO LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS E INFORMÁTICA

TEMA: MODELO METODOLÓGICO PARA EL DESARROLLO DE APLICACIONES WEB BASADO EN EL ANÁLISIS DE METODOLOGÍAS, Y ESTÁNDARES ISO/IEC/IEEE. CASO PRÁCTICO: SISTEMA GESTIÓN DE CONJUNTO HABITACIONAL PARA LA EMPRESA SLONCORP.

AUTOR: GRIJALVA ORQUERA, ESTEBAN ANDRÉS

DIRECTORA: ING. INFANTES MANTILLA, CATHERINE DE LOURDES

SANGOLQUÍ

2020

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA DE SISTEMAS E INFORMATICA

CERTIFICACIÓN

Certifico que el trabajo de titulación, "***MODELO METODOLÓGICO PARA EL DESARROLLO DE APLICACIONES WEB BASADO EN EL ANÁLISIS DE METODOLOGÍAS, Y ESTÁNDARES ISO/IEC/IEEE. CASO PRÁCTICO: SISTEMA GESTIÓN DE CONJUNTO HABITACIONAL PARA LA EMPRESA SLONCORP***" fue realizado por el señor ***Grijalva Orquera, Esteban Andrés,*** el mismo que ha sido revisado en su totalidad, analizado por la herramienta de verificación de similitud de contenido; por lo tanto, cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí, 14 de febrero de 2020

Firma:

Ing. Infantes Mantilla, Catherine de Lourdes

C.C. 0912252590

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA DE SISTEMAS E INFORMATICA

AUTORÍA DE RESPONSABILIDAD

Yo, ***Grijalva Orquera, Esteban Andrés,*** declaro que el contenido, ideas y criterios del trabajo de titulación: ***Modelo metodológico para el desarrollo de aplicaciones web basado en el análisis de metodologías, y estándares ISO/IEC/IEEE. Caso práctico: Sistema gestión de Conjunto Habitacional para la empresa Sloncorp*** es de mí autoría y responsabilidad, cumpliendo con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Consecuentemente el contenido de la investigación mencionada es veraz.

Sangolquí, 14 de febrero de 2020

Firma:

Grijalva Orquera, Esteban Andrés

CC. 1715957674

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA DE SISTEMAS E INFORMATICA

AUTORIZACIÓN

Yo, ***Grijalva Orquera, Esteban Andrés***, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: ***Modelo metodológico para el desarrollo de aplicaciones web basado en el análisis de metodologías, y estándares ISO/IEC/IEEE. Caso práctico: Sistema gestión de Conjunto Habitacional para la empresa Sloncorp*** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 14 de febrero de 2020

Firma:

Grijalva Orquera, Esteban Andrés

CC. 1715957674

DEDICATORIA

Para todos quienes estuvieron involucrados en este largo proceso.

AGRADECIMIENTO

A Dios, por quien todas las cosas existen y subsisten, a mi familia, por su apoyo incondicional, y a aquellas personas que invirtieron su tiempo y esfuerzo para guiarme en la elaboración de este trabajo.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El desarrollo de sistemas, con el pasar del tiempo y los avances tecnológicos, ha adquirido mayor importancia para la ejecución de diversas tareas en la vida cotidiana de las personas. La aplicación de la Ingeniería de Software, a la automatización de procesos o la resolución de problemas, ha posibilitado la implementación de múltiples sistemas que actualmente son utilizados en esta interacción. El objetivo del proceso de Ingeniería de Software es satisfacer las necesidades y requerimientos de los clientes, mediante un aplicativo, de manera efectiva y con niveles de calidad aceptables. Con este fin, se han creado metodologías con varios enfoques y objetivos, aplicables a problemáticas específicas, como también estándares que regulan estos procesos. El presente trabajo propone la elaboración de un modelo metodológico para el desarrollo de software basado en la integración de diferentes metodologías orientadas a la web, y procesos obtenidos de los estándares ISO /IEC / IEEE 12207 de Ingeniería de Requerimientos e ISO / IEC / IEEE 29148 de Ingeniería de Software que serán incluidos a las fases del ciclo de vida del software. El modelo es aplicado en la automatización de los procesos de Gestión de Conjuntos Habitacionales mediante la implementación de un aplicativo orientado a la web, con los propósitos de validar la metodología, y generar un producto que garantice la calidad del sistema con base en las métricas generales definidas para el desarrollo de software.

PALABRAS CLAVE

- **REQUERIMIENTOS**
- **METODOLOGÍAS**
- **ESTÁNDARES**
- **CALIDAD**

ABSTRACT

Through the pass of time and through the evolution of technology, software development has become more important in people's daily life for the execution of different tasks. The application of Software Engineering, to process automation and/or problem solving, has enabled the implementation of multiple systems that are currently used in this interaction. The objective of the Software Engineering process is to meet the needs and requirements of customers effectively and with acceptable quality levels, through an application. To this end, methodologies have been created with various approaches and objectives, applicable to specific problems, as well as standards that regulate these processes. The current document proposes the creation of a methodological model for software development based on the integration of different web-oriented methodologies, and processes obtained from the ISO / IEC / IEEE 12207 Requirements Engineering standard and the ISO / IEC / IEEE 29148 Software Engineering standard, that will be included in the software life cycle's phases. The model is applied in the automation of the Housing Complexes administration processes through the implementation of a web-oriented application, with the purposes of validating the methodology, and generating a product that guarantees the quality of the system based on the general metrics defined for software development.

KEY WORDS

- **REQUIREMENTS**
- **METHODOLOGIES**
- **STANDARDS**
- **QUALITY**

CAPITULO I - INTRODUCCIÓN

1.1 ANTECEDENTES

Eduardo de Jesús Cerecer (Ibarra, 2016), arquitecto de ITESO – Universidad Jesuita de Guadalajara, refiriéndose a la gestión de conjuntos habitacionales acota que "En la etapa de operación, propone un protocolo de buenas prácticas, como llevar a cabo sistemas y campañas para el ahorro energético, un sistema adecuado de residuos sólidos urbanos y reciclaje." Esto quiere decir que para una correcta administración se necesita proponer y ejecutar un protocolo de buenas prácticas en donde se puedan solventar todas las necesidades y servicios requeridos por la comunidad de personas dentro de un conjunto habitacional.

Por otro lado, Félix Bombarolo (CF+S, 1997), máster en hábitat y vivienda, ya se había referido a los problemas dentro de los conjuntos habitacionales diciendo "Existe cada vez más claridad respecto de la coordinación que debe existir en el tratamiento de los problemas habitacionales y urbanos". De esto se puede deducir que la resolución de los conflictos en organización, administración y convivencia dentro de estos espacios siempre ha sido una meta que debe ser alcanzada.

La gestión de conjuntos habitacionales ha sido y sigue siendo un tema de interés en el que se ha intervenido para mejorar la disponibilidad de los servicios ofrecidos dentro de cada comunidad. El objetivo de contar con una administración correcta es el mejorar la convivencia de las personas, poniendo disponible toda la información referente al control de servicios, áreas, unidades habitacionales, y personas, a través de un riguroso proceso de planificación.

El proceso utilizado en la gestión generalmente es realizado de forma manual, por un administrador delegado, quien fue elegido por votación de la comunidad. Esta persona es la encargada de realizar todas las actividades referentes a limpieza, organización, servicios básicos, entre otros. Esta información se basa en encuestas realizadas en diferentes urbanizaciones y edificios ubicados en diferentes partes de la ciudad (Conjunto San Martín – Sector Rumipamba, Torre San Francisco – Sector 6 de Diciembre, Edificio Bujase – Sector La Gasca, Urbanización La Granja – Sector Mariana de Jesús.)

1.2 PROBLEMÁTICA

La continua y vertiginosa evolución de la tecnología ha generado que actualmente el desarrollo y uso de aplicaciones orientadas a solucionar problemas reales, sean implementadas a través de software que permita al usuario consultar, procesar y analizar la información usando los navegadores de la Web. El incremento progresivo de las aplicaciones Web ha sido uno de los factores que ha influenciado en la efectividad y calidad de dichas aplicaciones, creando un mercado de ideas de desarrollo web muy creativas pero muy fugaces, ya que, no se fundamentan en un desarrollo organizado, estructurado, con objetivos claros y definidos, lo que conlleva a no ser sostenibles en el tiempo, y, se convierten en páginas de información “muerta”, información “caduca”, perdiendo el sentido mismo de la disponibilidad, inmediatez y accesibilidad de la información.

Muchos son los estudios, metodologías y estándares de desarrollo de software que se han implementado sobre software de aplicaciones “no web” o “de escritorio”, y, a pesar que el desarrollo web sigue siendo un “software”, éste tiene que ser analizado como tal, dado la particularidad de sus características y el impacto que generan a nivel del usuario final.

El principal problema identificado en el desarrollo de aplicaciones web, es que inicialmente se lo conceptualiza como una aplicación orientada al marketing, a la venta, preventa y postventa de algún producto o servicio, razón por la que se hacen a un lado criterios de estandarización, calidad y mejoramiento del software como tal. El problema se puede identificar al observar en el mercado tecnológico, en redes sociales y demás una serie de oportunidades de software que promueven la creación de páginas, sitios y espacios web, sin establecer condiciones, características estándares, que tengan una guía y procedimiento claramente definido bajo normas de estandarización.

Adicionalmente a lo expuesto es necesario considerar que si bien existen algunos sitios y páginas web que han sido desarrolladas bajo determinados esquemas de desarrollo y seguimiento, no se ha podido aún identificar un modelo de desarrollo de software web que promueva una estandarización de procedimientos, garantice un proceso de mejoramiento continuo y proponga una sostenibilidad en el tiempo. Los sitios o páginas web creados y expuestos en el mercado actualmente responden de manera directa a un requerimiento netamente comercial por lo que su calidad y efectividad es medida a través de la funcionalidad que ofrece, dejando a un lado todos los criterios de calidad en los procedimientos no funcionales.

1.3 JUSTIFICACIÓN

El desarrollo de un modelo metodológico es un proceso que conlleva una serie de recursos de investigación, análisis, contrastación de resultados e inferencia, con lo cual se proponen procedimientos claramente definidos que se fundamentan en frameworks, esquemas técnicos, estándares y mejores prácticas, razón por la cual, el presente proyecto no solo realiza una recopilación de datos relacionados con las metodologías de desarrollo web, sino que pretende

realizar una propuesta de modelo que integre conceptos técnicos de desarrollo en programación, diseño, esquematización, seguridad, portabilidad de data y exposición de resultados.

El modelo metodológico propuesto deberá establecer adicionalmente todos los lineamientos y parámetros de medición que permitan controlar y dar seguimiento tanto a los esquemas de desarrollo, codificación y programación del software como a los esquemas funcionales orientados a la implementación del mismo.

El modelo metodológico a desarrollar presentará en su contexto general no solo un establecimiento de procedimientos estandarizados, sino la especificación de métricas de desarrollo que permitan determinar la calidad y aseguramiento de la misma para el software. Bajo este concepto, se propone crear una aplicación real del modelo y las conceptualizaciones de las métricas definidas mediante el desarrollo de un sistema de administración de un conjunto habitacional, el cual se concibe para dos tipos de usuarios: al administrador, y a la comunidad. Primero al administrador, ya que automatizando las tareas se podrá llevar un control más eficiente de todo lo que sucede dentro del conjunto habitacional y se podrá tomar decisiones más acertadas acerca de las diferentes circunstancias que pueden transcurrir día a día. Por otro lado, el segundo beneficiario es la comunidad, ya que podrá obtener acceso a toda esta información respecto al estado del conjunto y sus necesidades y problemas de una manera más rápida. De esta forma se podrá trabajar en conjunto: notificando los problemas o resoluciones por parte del administrador, y acatando las decisiones o haciendo sugerencias por parte de la comunidad.

Es conocido que la mayoría de los conjuntos habitacionales en Quito tiene una administración manual, que muchas veces no es transparente, o que en otros casos no es eficiente y por lo tanto

genera inconformidad en las personas. La automatización de todo el proceso de gestión mejorará sin duda el nivel de satisfacción de la comunidad respecto a la convivencia dentro del conjunto.

La importancia de desarrollar este proyecto se fundamenta en poder determinar un modelo metodológico adaptado a las normas y estándares del desarrollo de aplicaciones web, no solo para definir la funcionalidad y servicio que ofrece la aplicación web, sino para determinar las métricas de desarrollo y establecer la forma en la que pueden ser controladas y monitoreadas para llegar a una calidad que incluya un mejoramiento continuo y una sostenibilidad tecnológica efectiva.

1.4 OBJETIVOS

a. Objetivo General

Desarrollar un modelo metodológico para el desarrollo de aplicaciones web basado en el análisis de metodologías, y estándares ISO/IEC/IEEE a fin de definir un esquema de métricas de calidad de dicho software.

b. Objetivos Específicos

- Identificar las metodologías y estándares de desarrollo de aplicaciones web que a la fecha de ejecución se encuentren vigentes en el mercado tecnológico.
- Realizar una revisión de literatura para determinar las funcionalidades, características, beneficios y dificultades de metodologías y estándares utilizados para el desarrollo de aplicaciones web.
- Elaborar un modelo metodológico para el desarrollo de aplicaciones web bajo estándares y normas ISO/IEC/IEEE.

- Identificar las principales métricas de calidad funcionales y no funcionales que deberán ser consideradas en el proceso de desarrollo de software a fin de asegurar la calidad de este.
- Aplicar el modelo metodológico en el desarrollo de un sistema de gestión de Conjunto Habitacional, a fin de validar los procedimientos y métricas propuestas.

1.5 ALCANCE

El proyecto a ser desarrollado iniciará con una investigación teórica sobre la Ingeniería de Software, con el fin de determinar qué aspectos son necesarios para que una metodología de desarrollo sea considerada exitosa.

A continuación se realizará una investigación de diferentes metodologías de desarrollo de software, al igual que diferentes estándares ISO/IEEE, para la creación de una propuesta metodológica que combinará varios aspectos de cada una.

Una vez definida la metodología propuesta, se procederá a su aplicación a un sistema de gestión de Conjunto Habitacional. Al haber finalizado la aplicación se realizará una evaluación de los resultados obtenidos para determinar si la metodología tuvo una implementación exitosa, al igual que sus defectos o falencias.

CAPITULO II - MARCO TEORICO

2.1 FUNDAMENTOS TEÓRICOS

2.1.1 Ingeniería de Software

El término Ingeniería de Software nace de una conferencia realizada en 1968 por un grupo de estudio de NATO (North Atlantic Treaty Organization) respecto a las Ciencias de la Computación. Durante esta conferencia los temas de mayor importancia fueron la falencia en las teorías y métodos de diseño y producción, la aparente inevitabilidad de fallos en sistemas grandes, la inexistencia de métodos para medir los avances de un proyecto, la inmensa inversión en sistemas definidos incorrectamente que consecuentemente no funcionan como es requerido, entre otros (Braude & Bernstein, 2016, pág. 2).

Más adelante se definió a la Ingeniería de Software como una disciplina que involucra todos los aspectos de desarrollo y mantenimiento de un producto de software. Actualmente, la IEEE la define como "la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software" (The Institute of Electrical and Electronics Engineers , 1990, pág. 30).

En general, la Ingeniería de Software busca la creación de un producto que cumpla con las siguientes características (Leach, 2016, págs. 10, 11):

- **Eficiencia:** El software es producido en el tiempo esperado y con los recursos disponibles.
- **Confiabilidad:** El comportamiento del software es el esperado.

- **Usabilidad:** El software puede ser usado adecuadamente por el usuario, el sistema operativo, y el entorno previamente configurado.
- **Modificabilidad:** El software puede ser modificado fácilmente si los requerimientos del sistema cambian.
- **Portabilidad:** El software puede ser migrado a otras computadoras o sistemas sin mayor esfuerzo en reescribir el código.
- **Capacidad de Prueba:** El software puede ser probado fácilmente, lo que significa que el código fue escrito modularmente.
- **Reusabilidad:** Algunas partes o todo el software puede ser utilizado nuevamente en otros proyectos.
- **Mantenibilidad:** El software puede ser fácilmente entendido y cambiado a través del tiempo en el caso de que se generen problemas.
- **Interoperabilidad:** El software puede interactuar adecuadamente con otros sistemas.
- **Exactitud:** El software produce el resultado correcto.

2.1.1.1 Ciclo de Vida

La Ingeniería de Software es un proceso que abarca un conjunto de actividades generales que se describen a continuación (Leach, 2016, pág. 11):

- Análisis del problema
- Definición de los requerimientos
- Diseño del software
- Codificación del software
- Pruebas e integración del código

- Instalación y entrega del software
- Documentación
- Mantenimiento
- Aseguramiento de la calidad
- Entrenamiento
- Estimación de recursos
- Gestión del Proyecto

Todas estas actividades se encuentran agrupadas en fases, y a este conjunto de fases se lo denomina Ciclo de Vida del Desarrollo de Software. Todo el ciclo representa las "etapas identificables a través de las cuales el software evoluciona durante su tiempo de vida" (Mall, 2014, pág. 34).

El ciclo de vida cuenta con las fases explicadas en la siguiente tabla:

Tabla 1
Descripción de Fases del Ciclo de Vida del Software

Fase	Descripción
Análisis	Es el proceso mediante el cual se entienden y se describen las necesidades y los requerimientos para una aplicación. Este proceso es estudiado por la rama de la Ingeniería de Software llamada Ingeniería de Requerimientos.
Diseño	Creación de una representación o modelo del sistema que va a ser construido. Es un conjunto de documentos que contienen texto y diagramas que sirven como la base sobre la cual una aplicación podrá ser programada. Estos documentos deberán satisfacer los requerimientos definidos en la fase previa.
Implementación	La traducción de los diseños a un código apropiado, libre de errores, y mantenible. Etapa en la que los modelos son traducidos a un sistema funcional (nuevamente cumpliendo con los requerimientos acordados).
Validación	Ejecución del código proveyendo entradas reales y evaluando si el resultado obtenido es el resultado requerido. El resultado incorrecto se traduce como una falla o error en la implementación, diseño, o análisis de requerimientos.

Continúa

Mantenimiento	El proceso mediante el cual se modifica el software o sus componentes para corregir fallas, mejorar el rendimiento u otros atributos, o adaptar dicho sistema a un ambiente distinto con nuevos requerimientos o necesidades.

Fuente: (Braude & Bernstein, 2016, págs. 32 - 36)

2.1.1.2 Modelos

Los Modelos de Desarrollo de Software describen el orden y ejecución de las actividades en métodos iterativos y no iterativos. La siguiente tabla muestra los diferentes modelos existentes y sus características:

Tabla 2
Descripción de los Modelos de Desarrollo de Software

Modelo	Enfoque	Características	Aplicabilidad
Modelo Cascada	Ordena las fases de desarrollo en forma de caída de agua (cascada) o secuencial, permitiendo la ejecución de cada fase siempre y cuando la anterior haya finalizado.	• No es iterativo • Requiere que cada etapa sea exacta y esté libre de errores para no afectar a las fases subsiguientes • Requisitos definidos únicamente en la fase inicial • Permite una fácil planificación del proyecto • Permite estimación de tiempos muy aproximados	Utilizado para proyectos en donde los desarrolladores poseen un conocimiento total de los requerimientos; los mismos que son concretos, precisos, e inmutables. Adicionalmente el producto final es específico para el cliente y no es apto para su venta genérica.
Modelo de Prototipos	Creación de diversos prototipos funcionales a lo largo de todo el ciclo de vida del software.	• Es iterativo • La totalidad de los requerimientos no es necesariamente conocida antes del diseño y desarrollo del software • Se generan prototipos con funcionalidades básicas que evolucionan hasta llegar al producto final • Permite actualizaciones en los requerimientos a lo largo del desarrollo • Existe una alta participación del	Se puede usar para la creación de productos genéricos donde puede o no haber un cliente específico. Recomendado para casos en los que el cliente requiera ver avances funcionales del sistema periódicamente.

Continúa

			usuario de acuerdo con la cantidad de prototipos presentados	
Modelo Espiral	Similar al de prototipos. Añade la evaluación de riesgo para cada prototipo durante su período de pruebas.	•	Mismas características que el modelo de prototipos • La evaluación del riesgo consiste por lo general en evitar que el proyecto fracase debido a recursos, tiempo, planificación, etc.	Proyectos con requerimientos escalables donde el cliente es específico.
Modelo Impulsado por el Mercado	Proveer un producto alta calidad y funcionalidad para mantener o incrementar la participación en el mercado.	•	Es iterativo • No existen requerimientos fijos • Funcionalidad añadida incluso días antes del lanzamiento del producto • Se realizan varios lanzamientos del producto	Aplicaciones web y móviles que requieren mantenimiento, mejoras, y actualización continua.
Modelo de desarrollo Código Abierto	Se basa en la creencia fundamental de que el contenido intelectual del software debe estar disponible para cualquier individuo de maner gratuita.	•	El código fuente puede ser públicamente modificado por cualquier individuo • Múltiples revisiones del producto antes de su lanzamiento por diferentes personas con diferentes niveles de conocimientos • Los cambios realizados por cualquier individuo deben estar disponibles sin ningún costo	Desarrollo de componentes, librerías, sistemas operativos, etc.
Modelo de desarrollo Ágil	Una forma de prototipado que depende de un extenso conjunto de componentes de alta calidad que trabajan en conjunto.	•	Es iterativo • Evolución de Requerimientos y soluciones en el tiempo de acuerdo con las necesidades del proyecto. • Los principios del desarrollo ágil pueden ser encontrados en "Agile Manifesto" (Beck et al., 2001)	Desarrollo de componentes y subsistemas. También puede ser aplicado al desarrollo de aplicaciones móviles y web.

Fuente: (Leach, 2016, págs. 13 – 28; Braude & Bernstein, 2016, págs. 37 - 51)

Si bien todos los modelos deben cumplir las mismas fases, la principal diferencia radica en el enfoque de cada una. Por ejemplo, en los modelos iterativos, las actividades podrán realizarse varias veces durante todo el ciclo de vida del software, mientras que en los modelos no iterativos las repeticiones de las actividades están restringidas. Los modelos iterativos se enfocan en

garantizar en cada fase del ciclo de vida que las actividades hayan sido completadas rigurosamente, afectando de esta manera la cantidad de tiempo invertido en el proceso de Ingeniería. Por otro lado, los modelos no iterativos se enfocan en presentar el producto final en el menor tiempo posible tratando de garantizar el cumplimiento de las actividades sin repeticiones.

2.1.2 Ingeniería de Requerimientos

La Ingeniería de Requerimientos es actividad interdisciplinaria que se ejecuta sobre de los dominios del cliente y del proveedor para definir y mantener los requerimientos que el software debe cumplir. Se enfoca en descubrir, obtener, desarrollar, analizar, determinar métodos de verificación, validar, comunicar, documentar y gestionar requerimientos (ISO/IEC/IEEE, 2011, pág. 6).

Los requerimientos son la base para cualquier proyecto de software, definiendo todo lo que los clientes necesitan y todo lo que el sistema debe realizar para satisfacer estas necesidades. Generalmente las especificaciones no se encuentran definidas muy claramente al inicio, y pueden ser afectadas por varios factores o metas que también podrán cambiar con el paso del tiempo (Dick, Hull, & Jackson, 2017, pág. 2).

Los requerimientos pueden ser divididos en tres categorías (Laplante, 2009, págs. 6 - 11):

- Requerimientos Funcionales (RF)
- Requerimientos No Funcionales (RNF)
- Requerimientos de Dominio

Los **Requerimientos Funcionales (RF)** describen a los servicios que el sistema debe proveer y cómo el sistema debe reaccionar según las entradas de información (inputs). Existen varias maneras de representar los RF como lenguaje natural, modelos visuales, y los diferentes métodos más formales y rigurosos.

Los **Requerimientos No Funcionales (RNF)** describen el comportamiento del sistema frente a algunos atributos observables como la confiabilidad, reusabilidad, mantenibilidad, entre otros. Los RNF, muy frecuentemente, pueden interactuar entre ellos; lo que significa que en el caso de mejora de un RNF se perjudica otro (ej. incremento en seguridad versus disminución en velocidad).

Los **Requerimientos de Dominio** se derivan del dominio de la aplicación. Estos tipos de requerimientos pueden consistir en nuevos RF o restricciones en RF, o pueden especificar como ciertos cálculos deben ser realizados. Existen estándares en la industria, restricciones en el hardware, acuerdos entre entidades reguladores, entre otros, que pueden afectar directamente el sistema con requerimientos de dominio.

2.2 CARACTERÍSTICAS DE SOFTWARE

La ISO/IEC/IEEE 24765:2017 define a las características del software como “inherentes, posiblemente accidentales, rasgos, calidad o propiedades del software.” Algunos ejemplos de características del software pueden ser: funcionalidad, rendimiento, atributos, restricciones de diseño, número de estados, entre otros (ISO/IEC/IEEE, 2017).

Dependiendo de las características el software puede ser de tipo: cliente – servidor, web, o móvil.

2.2.1 Software Cliente - Servidor

Es un tipo de software en el cuál intervienen dos objetos distribuidos: el cliente y el servidor. Ambos objetos establecen una relación en donde el cliente solicita al servidor que ejecute alguna operación o trabajo (ISO/IEC, 1999).

2.2.2 Software Web

Es un tipo de software distribuido que se implementa en múltiples lenguajes y estilos, que incorpora varios componentes, y que está construido y diseñado para ejecutarse en un navegador web. Algunos de los componentes con los que interactúan las aplicaciones web son lenguajes de scripting, archivos planos HTML, bases de datos, imágenes, y complejas interfaces de usuario. Este software generalmente se encuentra distribuido geográficamente en lugares distintos durante el desarrollo y el despliegue, y se comunica de diversas maneras con sus diferentes componentes (Offutt, 2002, pág. 25).

2.2.3 Software Móvil

Es un tipo de software diseñado para ejecutarse en los sistemas operativos específicos de los dispositivos móviles. Este software puede ser desarrollado en las plataformas ofrecidas por los fabricantes (enfoque nativo), como también mediante el uso de tecnología web. (Charland & LeRoux, 2011, pág. 50)

2.3 LINEAMIENTOS DE ASEGURAMIENTO DE LA CALIDAD DE SOFTWARE

2.3.1 Estándar

Un estándar se define como el documento, establecido por consenso y aprobado por un organismo reconocido, que proporciona, para uso común y repetido, reglas, lineamientos o

características para actividades o sus resultados, orientados a lograr el grado óptimo de orden en un contexto dado. Adicionalmente los estándares deben estar basados en los resultados consolidados de la de la ciencia, tecnología, y experiencia, y deben estar orientados al fomento de beneficios óptimos para la comunidad (ISO/IEC, 2013).

2.3.2 Normativa

La normativa se define como el conjunto de reglas promulgadas por un organismo regulador de acuerdo con los estatutos y directivas legales (generalmente de un país) (ISO, 2018). Es un documento que establece normas legislativas vinculantes, que es adoptado por una autoridad (ISO/IEC, 2013).

2.3.3 Metodología

La metodología es un proceso que describe los pasos necesarios a seguir para completar las actividades contempladas por cada fase del ciclo de vida del software; definiendo los lineamientos a seguir durante la ejecución de cada fase (Mall, 2014, pág. 35).

2.4 APLICABILIDAD

2.4.1 Calidad

La ejecución de todo el ciclo de vida cumpliendo todas sus fases y actividades garantiza un producto de software de alta calidad. Sin embargo, para poder definir dicho término debemos repasar los siguientes conceptos (Schulmeyer, 2008, págs. 2 - 6):

- **Objeto. -** es la entidad a la que se aplica la calidad. Para este estudio el objeto es un producto de software.

- **Proceso. -** conjunto de actividades realizadas para alcanzar un objetivo, por ejemplo, el desarrollo de un producto de software. La calidad del objeto dependerá de la calidad del proceso.

- **Requerimiento. -** es una funcionalidad, condición, o característica necesaria que el objeto debe poseer para satisfacer el contrato, estándar, o especificación documentada.

- **Usuario. -** el cliente o usuario final del sistema.

- **Evaluación. -** proceso por el cual se determina la completitud de los requerimientos. Incluye métodos como análisis, inspecciones, revisiones, y pruebas.

- **Métrica. -** es una medida cuantificable del nivel en que un sistema, componente o proceso posee un atributo específico (The Institute of Electrical and Electronics Engineers , 1990, pág. 47).

Tomando en cuenta estas definiciones se define la calidad como el grado en el que el **objeto** satisface un conjunto de atributos o requerimientos especificados para cumplir un objetivo o meta prefijada (Schulmeyer, 2008, pág. 6). También es definida como el mayor grado en el que el producto de software cumple con los requerimientos especificados (documentados), y al mismo tiempo estos requerimientos cumplen con los deseos y necesidades de los usuarios (Braude & Bernstein, 2016, pág. 23).

2.4.2 Gestión de la Calidad

La gestión de calidad es el conjunto de todas las actividades que son requeridas para la planificación de la calidad en una organización, y todas las actividades necesarias para satisfacer los objetivos de la calidad (Nanda, 2005, pág. 8). Está compuesta por cuatro elementos como se muestra en la siguiente figura.

Figura 1. Elementos de la Gestión de la Calidad

2.4.2.1 Planeación de la Calidad

(Nanda, 2005, págs. 9-12)

Agrupa todas las actividades a realizarse para establecer objetivos de calidad, identificar requerimientos de calidad, planificar el Sistema de Gestión de Calidad (SGC), y planificar la ejecución del proceso de acuerdo con el SGC.

Durante el establecimiento de los objetivos de calidad se deben establecer todas las métricas (cualitativas y cuantitativas) que serán evaluadas durante el proceso de desarrollo del software y determinarán la calidad final del mismo.

La identificación de los requerimientos de calidad se realizará mediante la recopilación de todos los requerimientos funcionales, no funcionales, y de dominio, los cuáles serán evaluados según las métricas definidas para determinar si el sistema cumple con la calidad deseada.

La planificación de un SGC implica la planificación de todos los elementos que son necesarios para cumplir con los requisitos de calidad. Esto incluye el establecimiento de modelos de desarrollo, establecimiento de puntos de control (hitos), definición de métodos, establecimiento de normas, identificación de los recursos necesarios, establecimiento de lineamientos para el proceso de desarrollo del sistema, entre otros.

La planificación de la ejecución del proceso de acuerdo con el SGC definido deberá realizarse de modo que se puedan cumplir los requisitos de calidad. Esta ejecución se realizará en contexto de la planificación para el desarrollo del software.

2.4.2.2 Control de la Calidad

(Nanda, 2005, págs. 12, 13)

El mejoramiento de la calidad se encuentra descrito por el siguiente gráfico:

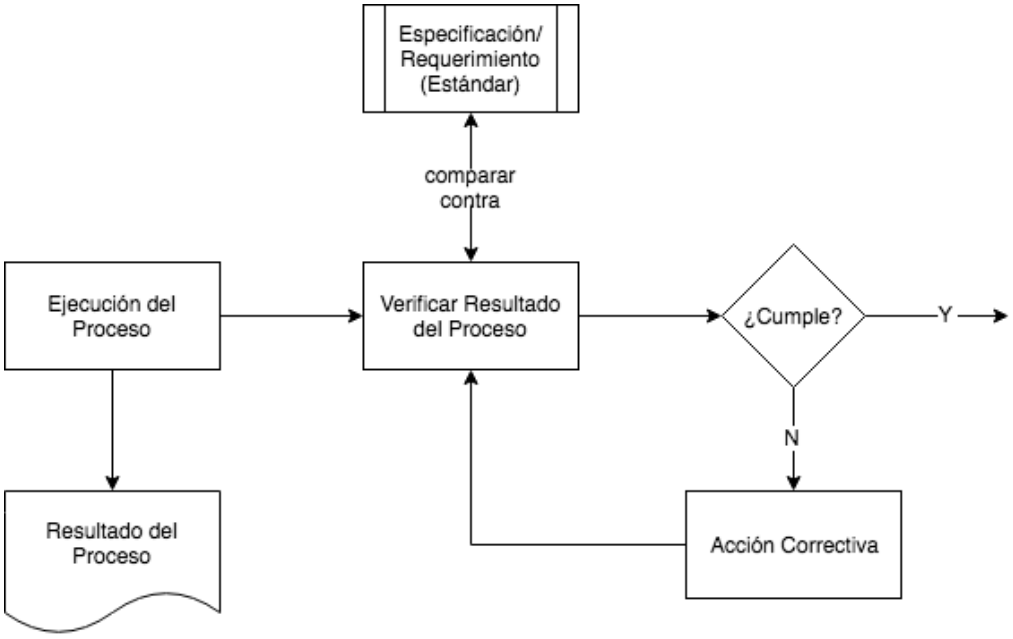


Figura 2. Proceso de Mejoramiento de la Calidad

Se ejecuta un proceso o actividad para evaluar si la calidad es la deseada. Dicho resultado se verifica comparándolo con las especificaciones, requerimientos, o estándares aplicables. Las discrepancias encontradas en el resultado son definidas como deficiencias que necesitan ser corregidas. Estas discrepancias son corregidas mediante la determinación e implementación de acciones correctivas adecuadas, y cuando sea necesario, acciones preventivas, para evitar que se presenten discrepancias similares en el futuro. El resultado del proceso o actividad corregida se vuelve a examinar para verificar si la deficiencia ya no está presente. En el caso que no se pueda

corregir la deficiencia deberán tomarse acciones para mitigar el riesgo de los requerimientos que no fueron satisfechos con la calidad requerida.

El control de la calidad no se aplica únicamente al resultado final, sino también puede ser aplicado durante la ejecución de la actividad o proceso; facilitando así, la determinación de acciones preventivas y correctivas. Generalmente las actividades de control de calidad son clasificadas como reactivas ya que su objetivo primario es detectar y eliminar defectos que ya se encuentra en el sistema, opuesto a las actividades de aseguramiento de la calidad, que generalmente son clasificadas como proactivas ya que su objetivo primario es prevenir los defectos de un sistema.

2.4.2.3 Aseguramiento de la Calidad

(Nanda, 2005, págs. 13 - 15)

El Aseguramiento de la Calidad (QA) es un conjunto de actividades que definen y evalúan la adecuada ejecución de los procesos de software para proporcionar evidencia que establezca la confianza de que los procesos son apropiados y producen software de una calidad adecuada para los fines previstos (IEEE Computer Society, 2014, pág. 8).

El objetivo de asegurar la calidad de un producto de software radica en garantizar que la ejecución de todos los procesos que giran en torno al desarrollo de este, produzca el resultado esperado de acuerdo a los requerimientos de calidad definidos.

2.4.2.4 Mejoramiento de la Calidad

(Nanda, 2005, págs. 15 - 18)

El mejoramiento de la calidad puede definirse como el conjunto de actividades realizadas para mejorar la calidad de los procesos de desarrollo del sistema y la calidad del sistema en sí en base a las métricas definidas.

El ciclo de Deming (también conocido como PDCA - Plan, Do, Check, Act) define un proceso efectivo de mejoramiento de la calidad:

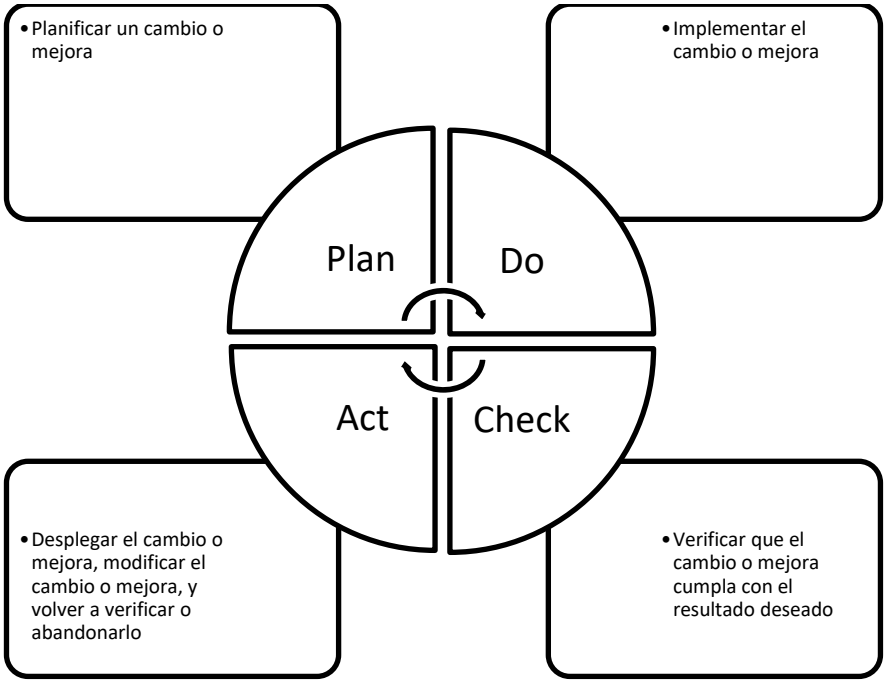


Figura 3. Ciclo de Deming (PDCA - Plan, Do, Check, Act)

2.4.2.5 Sistema de Gestión de Calidad (SGC)

El Sistema de Gestión de Calidad consiste en la estructura, procedimientos, procesos, y recursos organizacionales necesarios para implementar la gestión de la calidad (ISO, 1994, pág. 14, citado en Nanda, 2005, págs. 18, 19). El SGC tiene un enfoque definido, responsabilidades,

procesos definidos, y recursos necesarios para cumplir la planeación de la calidad, control de calidad, aseguramiento de la calidad, y mejoramiento continuo de la calidad.

Las fases para la implementación de un SGC son la Planificación, Definición, Refinamiento, Despliegue, y Mejora Continua (Nanda, 2005, págs. 18, 19).

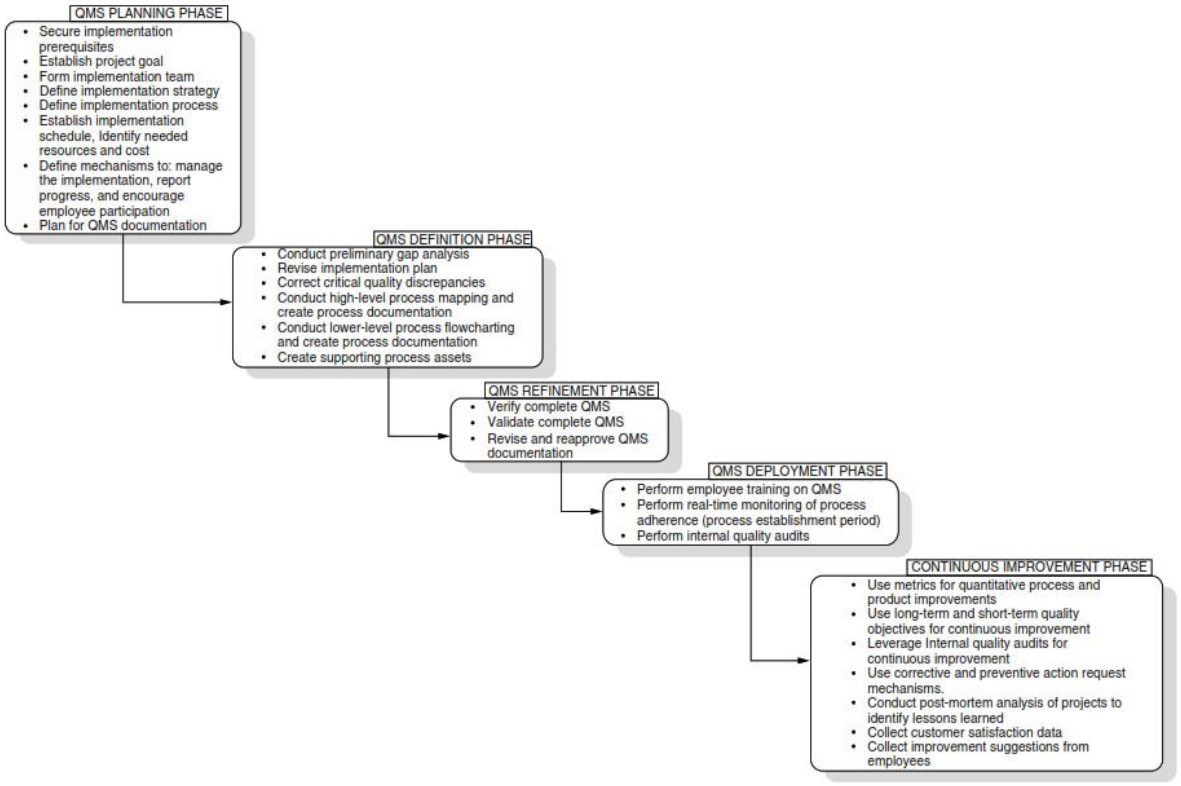


Figura 4. Fases de implementación de un Sistema de Gestión de Calidad

2.4.2.5.1 Fase de Planificación

Esta fase involucra la especificación del objetivo de la implementación del SGC, y establece la ruta mediante la cual la organización podrá cumplir el objetivo definido. Los planes diseñados en esta fase podrán ser revisados más adelante, pero en su mayor parte, las decisiones tomadas

tendrán un efecto profundo en el ritmo y el esfuerzo de la implementación del SGC. La planificación se realizará en base a los prerrequisitos, objetivo, equipo, estrategia, proceso, recursos, tiempos, costos, entre otros, de la implementación (Nanda, 2005, págs. 47, 48).

2.4.2.5.2 Fase de Definición

Esta fase involucra la definición y documentación del SGC de la organización. En el caso de que la organización haya seleccionado diversos estándares para sistemas de gestión de calidad, esta fase involucra la definición del SGC en concordancia con los requerimientos de dichos estándares. Algunas de las actividades realizadas en esta fase son análisis de requerimientos, revisión del plan de implementación, corrección de deficiencias críticas en el proceso, mapeo de procesos de alto y bajo nivel, creación de procesos de soporte, documentación adicional del SGC, entre otros (Nanda, 2005, págs. 48 - 50).

2.4.2.5.3 Fase de Refinamiento

Esta fase involucra la verificación de todo el SGC para asegurar que todos los procesos interactúen como originalmente se planificó, y adicionalmente, que todos los procesos sean mutuamente consistentes y estén correctamente definidos. Esta fase también involucra una validación final para asegurar que todos los elementos del SGC cumplan con los requerimientos de calidad de la organización. Las deficiencias identificadas durante esta fase son tratadas mediante la petición de una acción correctiva desde el propietario respectivo del proceso (Nanda, 2005, pág. 50).

2.4.2.5.4 Fase de Despliegue

Esta fase involucra la institucionalización del SGC a dentro de toda la organización. En esta fase el SGC es implementado de forma incremental para que sea adoptada gradualmente y se convierta en la nueva forma de trabajo. A medida que cada proceso es definido, documentado y aprobado por los responsables, entra a la fase de despliegue del SGC. La aplicación de un SGC provocará que la organización examine todos sus procesos existentes y por lo tanto la gran mayoría de estos serán afectados en cierto nivel. Generalmente la implementación del SGC resulta en cambios (grandes y pequeños) en los procesos existentes por lo que se debe definir un tiempo estimado en el que la organización y los responsables de cada proceso se adapten (Nanda, 2005, págs. 50 - 52).

2.4.2.5.5 Fase de Mejora Continua

Una vez completo el despliegue del SGC, el enfoque de la organización cambiará hacia el monitoreo de las necesidades del sistema para que sea continuamente mejorado y optimizado. Esta es la fase final y que nunca termina dentro de la implementación del SGC. Algunos de los mecanismos de mejora continua son programa de métricas para mejoras de procesos y del producto de software, establecer nuevos objetivos de calidad, auditorías internas de calidad, peticiones de acciones preventivas y correctivas, entre otros (Nanda, 2005, págs. 52, 53).

2.4.3 Calidad en el Ciclo de Vida del Software

2.4.3.1 Calidad en los Requerimientos

La base para un sistema son los requerimientos, por lo tanto, el no tenerlos o no definirlos adecuadamente tiene consecuencias diversas. El grupo Standish en su reporte CHAOS presenta

los factores de éxito de un proyecto, en donde se puede evidenciar que los primeros tres están directamente relacionados con la definición adecuada de los requerimientos.

Tabla 3
Standish Group - Reporte CHAOS

Factores de Éxito de los Proyectos		**% de Respuesta**
1	Involucramiento del usuario	15.9
2	Apoyo de la dirección ejecutiva	13.9
3	Definición clara de requerimientos	13
4	Planificación adecuada	9.6
5	Expectativas reales	8.2
6	Hitos con menor alcance o más pequeños	7.7
7	Personal competente	7.2
8	Posesión	5.3
9	Visión y Objetivos Claros	2.9
10	Personal trabajador y enfocado	2.4
Otros		13.9

Fuente: (The Standish Group, 2014, pág. 9)

Las métricas para evaluar la calidad de los requerimientos se muestran en la siguiente tabla:

Tabla 4
Métricas de Evaluación de Requerimientos

Métricas de Evaluación de Requerimientos

Métrica	Descripción
Accesibilidad	Fácil acceso la descripción de los requerimientos y a sus detalles. Para eso se requiere una correcta organización (numeración) y clasificación de su estado actual (implementado, pendiente, entre otros).
Exhaustividad	En qué medida todos los requerimientos y necesidades del cliente han sido plasmados en el documento.
Comprensibilidad	El lenguaje utilizado para la descripción del requerimiento debe ser sencillo para que cualquier actor lo entienda.
No Ambigüedad	La interpretación del requerimiento debe ser solamente una.
Consistencia	No debe existir contradicción entre requerimientos.
Priorización	La implementación de los requerimientos debe realizarse de manera ordenada para que no exista conflicto entre diversas funcionalidades.
Seguridad	Todo lo relacionado con la confidencialidad, privacidad, cifrado, entre otras.
Completitud	Los requerimientos deben contemplar las relaciones existentes entre ellos para las diversas funcionalidades a realizar. Se considerará completo a un requerimiento cuando su integración con las demás partes haya sido desarrollada.
Comprobabilidad	El requerimiento podrá ser probado para determinar su operatividad en el sistema final.
Rastreabilidad	Fácil localización del requerimiento en la interacción con el usuario, codificación, documentación, entre otras.

Fuente: (Braude & Bernstein, 2016, págs. 331 - 343)

La definición de los requerimientos se da en la etapa inicial del desarrollo del proyecto y por esta razón el impacto en la calidad del producto final es proporcionalmente mayor. Mientras el documento de requerimientos exprese de mejor manera las necesidades del cliente, el producto tendrá mayor calidad. Por esta razón se debe hacer un especial énfasis en dicha fase para generar

grandes beneficios en las etapas finales. Mejorar los requerimientos significará mejorar la calidad del producto (Dick, Hull, & Jackson, 2017, pág. 11).

2.4.3.2 Calidad en el Diseño

Las métricas descritas en la siguiente tabla son evaluadas para definir la calidad del diseño.

Tabla 5
Métricas de Evaluación del Diseño

Métricas de Evaluación del Diseño

Métrica	Descripción
Comprensibilidad	El diseño plasmado debe ser entendido por todos los responsables del proyecto de manera sencilla.
Modularidad	Divisiones del diseño en diferentes módulos de acuerdo a los requerimientos.
Cohesión	Cada parte del diseño deberá ser comprendida junto con las partes asociadas.
Acoplamiento	Las partes relacionadas del diseño deberán ser claras y mantener las relaciones estrictamente necesarias para evitar confusión.
Suficiencia	Que tan evidente es la representación de los requerimientos.
Robustez	Control de los resultados (output) respecto a las entradas erróneas (input).
Flexibilidad	Capacidad de adaptación a nuevos requerimientos.
Reutilización	Capacidad de uso del diseño en otras aplicaciones.
Eficiencia	La implementación basada en el diseño deberá tener las características deseadas (velocidad, almacenamiento, uso de recursos, etc.)
Confiabilidad	Cantidad aceptable de fallos.

Fuente: (Braude & Bernstein, 2016, págs. 508 - 523)

2.4.3.3 Calidad en la Implementación

Existen dos partes en la evaluación de la calidad de la implementación. La primera es la verificación, en donde la calidad se mide en base a la observación (inspección) del código fuente. La otra parte es la evaluación de la completitud del código (validación), que se refiere básicamente a las pruebas (Braude & Bernstein, 2016, pág. 585).

Las métricas utilizadas para la evaluación de la implementación se muestran en la siguiente tabla:

Tabla 6
Métricas de Evaluación de la Implementación

Métricas de Evaluación de la Implementación

Métrica	Descripción
Suficiencia	Mide el porcentaje de requerimientos y especificaciones de diseño que realmente fueron implementados.
Robustez	Hasta qué punto la aplicación controla el resultado de acuerdo a entradas de información anormales o erróneas.
Flexibilidad	Facilidad de acomodar nuevos requerimientos o realizar cambios a los preexistentes.
Reusabilidad	Facilidad de uso o integración del código en otras aplicaciones o sistemas.
Eficiencia	Optimización del código respecto al uso de recursos disponibles.
Confiabilidad	Nivel de confianza en que la aplicación realizará lo que fue codificada para realizar.
Escalabilidad	Facilidad de crecimiento y adaptación del sistema.
Seguridad	Estricto control sobre el manejo de la información, privacidad, cifrado, entre otras.

Fuente: (Braude & Bernstein, 2016, págs. 586 - 599)

El objetivo de la evaluación de la implementación es detectar y corregir defectos en el código antes que sean introducidos en el sistema.

2.4.3.4 Calidad en las Pruebas

Las pruebas son un proceso de validación cuyo propósito es detectar la mayor cantidad de defectos categorizándolos por el nivel de impacto que tengan. Los defectos son detectados cuando los datos ingresados en el sistema no producen los resultados esperados (Braude & Bernstein, 2016, págs. 628, 629).

La relación de esta fase con la calidad del producto final es evidente. Si el producto contiene fallas, su calidad se verá disminuida de acuerdo con la gravedad de estas.

2.4.3.5 Calidad en el Mantenimiento

(Braude & Bernstein, 2016, pág. 756)

Los sistemas de software continúan evolucionando aún después de su lanzamiento. El mantenimiento de software consiste en las actividades realizadas durante este período posterior al lanzamiento; algunas de las cuáles pueden ser la reparación de defectos no detectados, implementación de mejoras, y adaptación a cambios en el entorno.

Mientras los sistemas evolucionan, su complejidad aumenta y la capacidad de comprensión de este disminuye. Es por esta razón se deben realizar actividades periódicas para reducir la complejidad del sistema y aumentar su eficiencia. De esta manera la calidad inicial con la que el producto fue terminado (lanzado) se mantiene a lo largo del tiempo.

2.5 PARÁMETROS DE EVALUACIÓN, SEGUIMIENTO Y CONTROL

2.5.1 Indicadores

Un indicador es una medida que proporciona una estimación o evaluación de los atributos especificados derivados de un modelo con respecto a los objetivos, metas, riesgos y problemas de un proyecto (IEEE, 2017).

2.5.2 Factores Críticos

Los factores críticos son los atributos o características que un software debe poseer de manera obligatoria para satisfacer los estándares de calidad, requerimientos de usuario, funcionalidad requerida, entre otros, dentro de un proyecto para ser considerado como exitoso.

2.5.3 Métrica

La métrica es “una medida cuantificable del nivel en que un sistema, componente o proceso posee un atributo específico” (The Institute of Electrical and Electronics Engineers , 1990, pág. 47). El conjunto de métricas serán la base para determinar la calidad del producto de software respecto en función del cumplimiento de los requerimientos especificados para su implementación.

Las métricas principales definidas para el desarrollo de software son las siguientes (Leach, 2016, págs. 10, 11):

- **Eficiencia:** El software es producido en el tiempo esperado y con los recursos disponibles.
- **Confiabilidad:** El comportamiento del software es el esperado.

- **Usabilidad:** El software puede ser usado adecuadamente por el usuario, el sistema operativo, y el entorno previamente configurado.
- **Modificabilidad:** El software puede ser modificado fácilmente si los requerimientos del sistema cambian.
- **Portabilidad:** El software puede ser migrado a otras computadoras o sistemas sin mayor esfuerzo en reescribir el código.
- **Capacidad de Prueba:** El software puede ser probado fácilmente, lo que significa que el código fue escrito modularmente.
- **Reusabilidad:** Algunas partes o todo el software puede ser utilizado nuevamente en otros proyectos.
- **Mantenibilidad:** El software puede ser fácilmente entendido y cambiado a través del tiempo en el caso de que se generen problemas.
- **Interoperabilidad:** El software puede interactuar adecuadamente con otros sistemas.
- **Exactitud:** El software produce el resultado correcto.

2.6 METODOLOGÍAS DE DESARROLLO DE SOFTWARE WEB

2.6.1 UWE (UML-Based Web Engineering)

UWE es una metodología de desarrollo de software basada en el modelado de UML (Unified Modeling Language), que nació a finales de los años 90, con la idea de encontrar una manera estandarizada para construir modelos de análisis y diseño de Sistemas. UWE se enfoca en la separación de los diferentes conceptos que describen a un Sistema Web mediante la construcción de modelos en de las fases de análisis, diseño e implementación. Cada modelo representa un enfoque o visualización del mismo Sistema Web pero corresponde a un concepto específico. (UML-Based Web Engineering: An Approach Based on Standards, 2008, págs. 157 - 167)

2.6.1.1 Técnicas

Los conceptos manejados son Diagrama de Contenido para la definición de requerimientos de dominio de la aplicación y su interrelación, Diagrama de Estructura de Navegación para la definición de las rutas de navegación del sistema, y Diagrama de Presentación para la esquematización del sistema y las tareas de comunicación entre usuario y máquina.

UWE propone un esquema combinado de modelo cascada y modelo de prototipos durante el ciclo de vida del Sistema Web, mediante la creación y transformación de modelos (diagramas UML) durante las fases de análisis y diseño; y para la fase de implementación, la generación automática (sistematización) del código fuente a partir de los modelos previos. (UML-Based Web Engineering: An Approach Based on Standards, 2008, págs. 157 - 167)

2.6.1.2 Recursos Técnicos

La esencia de UWE se encuentra en el uso del lenguaje o notación UML para la esquematización de los conceptos de diagrama de contenido, diagrama de estructura de navegación, y diagrama de presentación. Para esto, se requiere la creación de diagramas de casos de uso, actividades, y navegación respectivamente.

Durante la sistematización, la metodología propone la creación de un perfil UML o notación específica para UWE que incluye lenguaje, relaciones, reglas, y sintaxis de modelos como complemento a UML.

La transformación de los diagramas se puede apreciar en el siguiente gráfico:

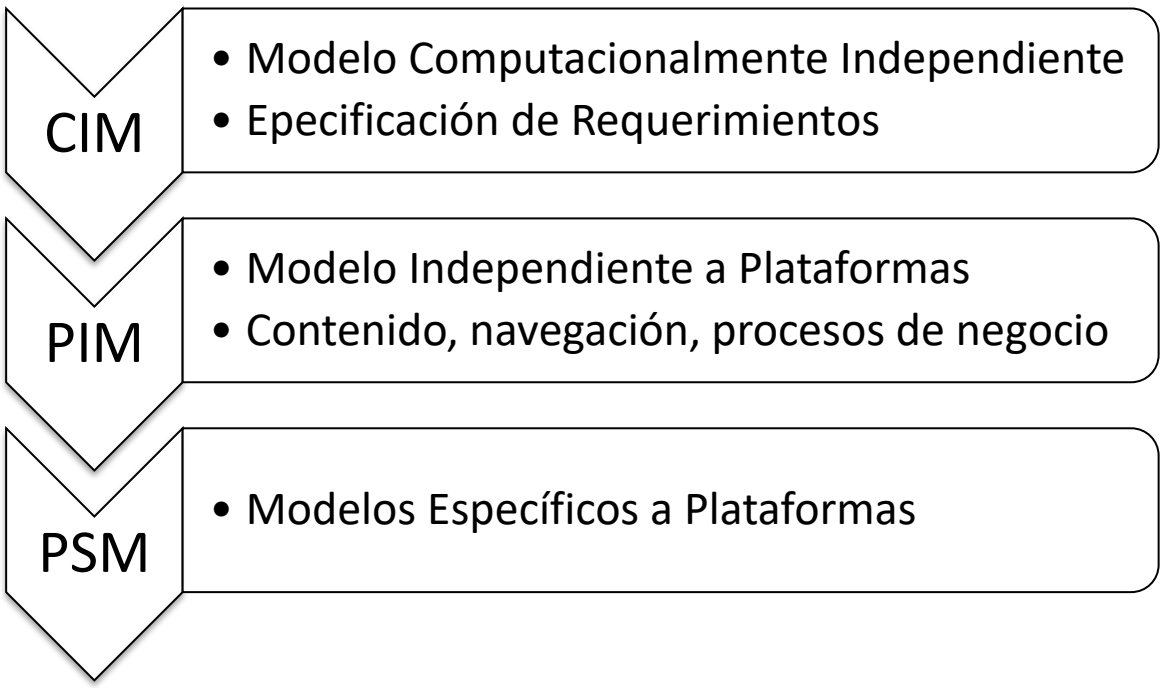


Figura 5. Transformación de Diagramas en la metodología UWE

El proceso empieza con el modelo de negocio también conocido como Modelo Computacionalmente Independiente (CIM), que es utilizado para especificar los requerimientos. Los Modelos Independientes a Plataformas (PIM) se derivan de este modelo de requerimientos. El conjunto de estos modelos representa los diferentes aspectos de las aplicaciones Web, abarcando el contenido, la navegación, los procesos de negocio, de presentación, y de adaptación del sistema. En el siguiente paso del proceso, todos los diagramas son integrados en uno solo que describe el modelo del sistema Web. Finalmente, usando la integración de los diagramas, se pueden generar los Modelos Específicos a Plataformas que serán utilizados como el punto de partida para la generación de código o sistematización.

Para la fase de generación de código fuente UWE propone el uso de herramientas CASE (Computed Aided Software Engineering), ArgoUWE de preferencia, u otras que sean compatibles con su perfil UML (Rossi, Pastor, Schwabe, & Olsina, 2008, págs. 180 - 183).

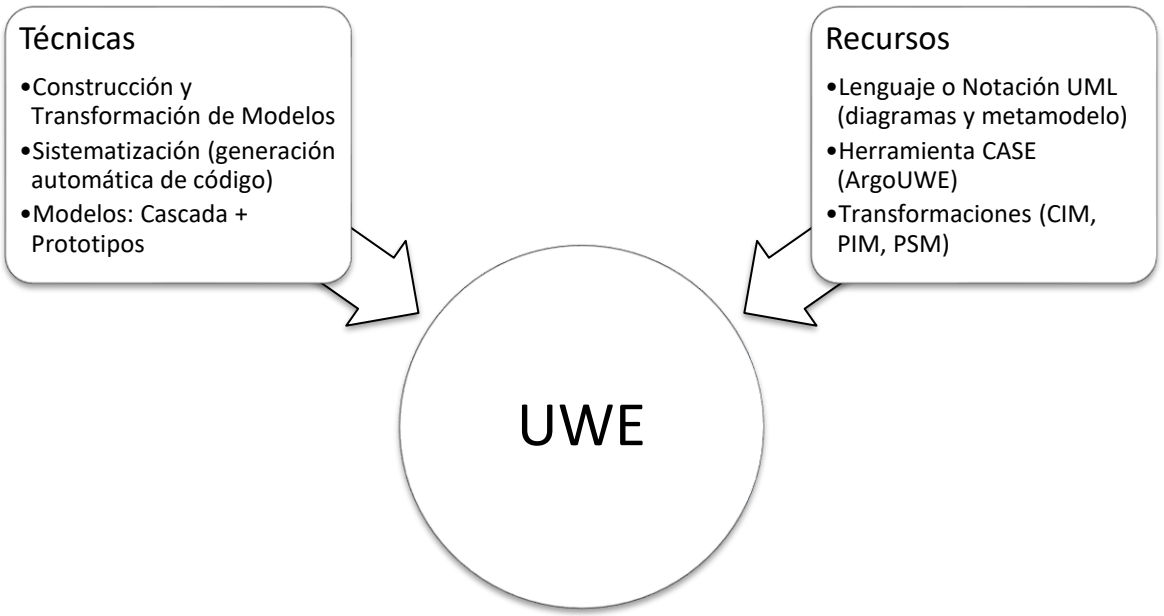


Figura 6. Técnicas y Recursos de la metodología UWE

2.6.1.3 Modelo

(Rossi, Pastor, Schwabe, & Olsina, 2008, págs. 163 - 189)

UWE plantea un modelo para el desarrollo de Sistemas Web que abarca principalmente las fases de análisis, diseño, e implementación.

La fase de análisis se enfoca en el dominio de la aplicación, y en la definición de sus requerimientos. UWE propone el desarrollo de un modelo de contenido que estará descrito por diferentes diagramas UML. Los diagramas iniciales son los de casos de uso, que definen la navegación y los procesos contemplados por el sistema. Por otra parte, los diagramas de actividades son utilizados para describir más detalladamente los diferentes requerimientos. El resultado de esta primera fase será el modelo de contenido plasmado en un diagrama de entidad -

relación, el modelo de usuario que incluye toda su información, relaciones e interacción con los procesos del negocio, y los diagramas de casos de uso y actividades como documentación de los requerimientos del sistema web.

Durante la fase de diseño se deberá crear el modelo de estructura de navegación. Para esto, se utilizarán como insumo los diseños obtenidos de la fase de análisis. El diagrama de entidad - relación será transformado a un diagrama de navegación, en donde las clases permanecen, y las relaciones se transforman a enlaces. Cada asociación deberá ser diagramada para las diferentes secuencias de navegación del sistema, incluyendo nuevos elementos UML como procesos, menús, y accesos a primitivos (link al inicio, tours guiados, y consultas a la base de datos). El enfoque de la fase de diseño es describir la navegabilidad del Sistema Web mediante este nuevo diagrama.

Finalmente, en la fase de implementación, UWE propone, en primer lugar, la generación de un modelo de presentación, y a continuación, la sistematización de los diagramas para la generación del código fuente. El enfoque del modelo de presentación es definir y describir la interfaz de usuario y su comunicación con el sistema. El diagrama de navegación será transformado a un diagrama de clases de presentación que incluirá nuevos elementos UML como campos de texto, botones, imágenes, etc. La información de los diferentes nodos generados se agrupará para formar una sola página web. Y a su vez las diferentes agrupaciones (páginas web) describirán la interacción del usuario para los diferentes requerimientos definidos de todo el Sistema Web en la fase inicial de análisis.

Al obtener todos los diagramas se inicia la sistematización o transformación de los modelos a código fuente. Las transformaciones se realizan de CMI a PIM y de PIM a PSM. Al obtener los modelos PSM, y mediante el uso de una herramienta CASE, se podrá generar un prototipo que podrá ser evaluado para su control y mejora de la calidad. Este prototipo contempla todos los requerimientos definidos en la fase inicial.

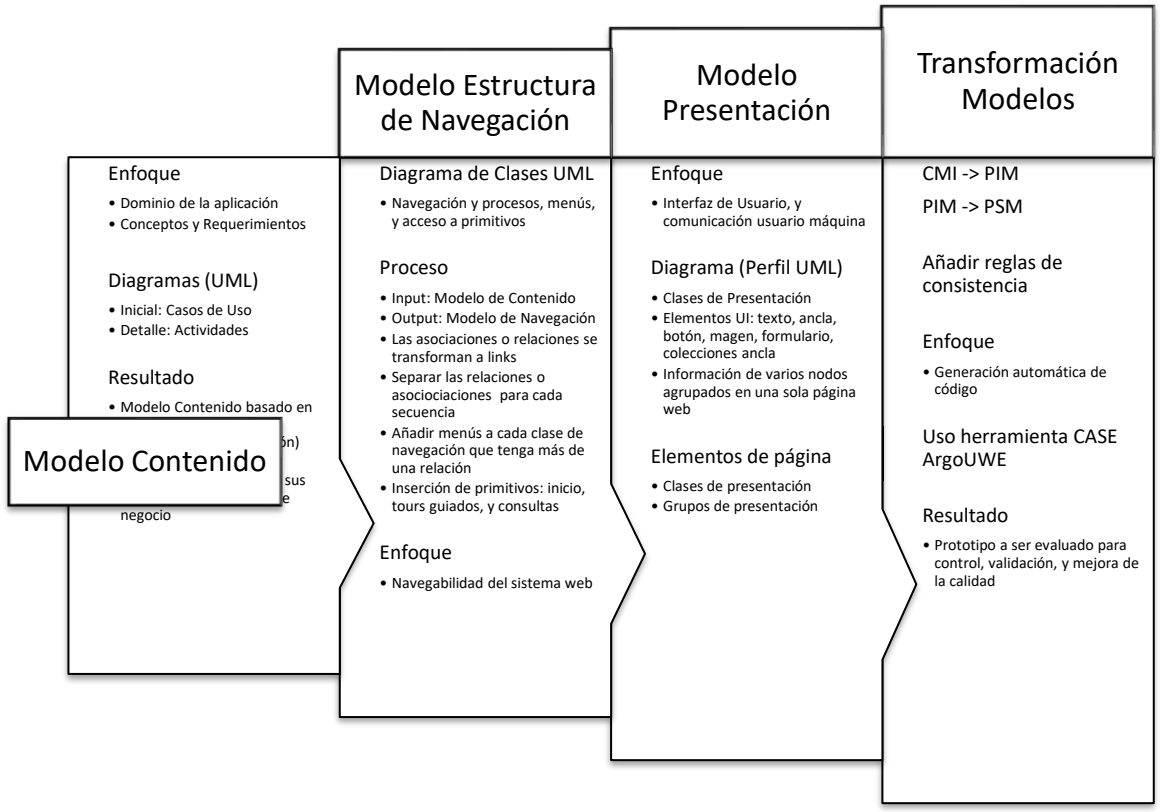


Figura 7. Modelo de la metodología UWE

2.6.2 SCRUM

Scrum es una metodología ágil creada en los 1990 para la gestión y desarrollo de productos. Su fundamento es el empirismo en donde el conocimiento es generado a través de las experiencias y la toma de decisiones en base a lo conocido. Scrum enfatiza el trabajo en equipo, la delegación de responsabilidades, y el progreso iterativo hacia un objetivo definido (Schwaber & Sutherland, 2017, págs. 1 - 5).

2.6.2.1 Objetivos

Scrum tiene tres objetivos principales (SeaLights, 2018):

- Medición de Entregables
- Medición de Efectividad
- Medición del Equipo

Cada uno de estos objetivos tiene métricas asociadas.

2.6.2.1.1 Medición de Entregables

Las métricas evaluadas en este objetivo son (SeaLights, 2018):

- **Éxito de la meta del Sprint**. – verificar la completitud del objetivo planteado en el sprint.
- **Defectos Escapados y Densidad de Defectos**. – cuantificar la cantidad de defectos encontrados por el usuario en producción, al mismo tiempo que el volumen de defectos por “módulo” del sistema.
- **Velocidad del Equipo**. – cuantificar la cantidad de historias finalizadas por el equipo para poder estimar el rendimiento de este en futuros Sprints.

- **Progreso del Sprint (velocidad de reducción)**. – verificar si el equipo se encuentra dentro del calendario planteado.

2.6.2.1.2 Medición de Efectividad

Las métricas evaluadas en este objetivo son (SeaLights, 2018):

- **Tiempo de Comercialización**. – tiempo que el sistema demora en empezar a entregar valor a los clientes.
- **Retorno de la Inversión (ROI)**. – cálculo del total de rédito del sistema versus el costo de desarrollo de los Sprints.
- **Redistribución de Capital**. – validar la viabilidad económica para continuar con el proyecto Scrum o no.
- **Satisfacción del Cliente**

2.6.2.1.3 Medición del Equipo

Las métricas evaluadas en este objetivo son (SeaLights, 2018):

- **Retrospectiva**. – medición cualitativa del progreso y la salud del equipo.
- **Satisfacción del Equipo**. – evaluar la satisfacción de los miembros con su trabajo para evitar conflictos o enfrentarlos.
- **Rotación de miembros del Equipo**. – si la rotación es alta significa que el ambiente no es saludable, y si la rotación es baja el ambiente es considerado como saludable.

2.6.2.2 Técnicas

(Beck et al., 2001)

Scrum se basa en la combinación de los modelos de desarrollo ágil, espiral, y de prototipos. El proceso de esta metodología gira en torno a un Sprint (espacio de tiempo definido para cumplir con objetivos específicos definidos por el equipo: dueño del producto, equipo de desarrollo, y maestro de Scrum).

Durante la implementación de la metodología se deben cumplir con varios eventos. El Daily Scrum es una reunión diaria durante el sprint para definir el trabajo que el equipo de desarrollo realizará durante la jornada. La Planificación del Sprint es una reunión en la que se define el objetivo del Sprint y todo el trabajo o tareas que se llevarán a cabo por los miembros del equipo. La Revisión del Sprint es la reunión de evaluación de tareas completadas y pendientes, y se planificación (parcial) del trabajo que se llevará a cabo en el siguiente sprint (incluyendo tareas incompletas). Por último, la Retrospectiva del Sprint es la reunión para la elaboración de un plan de mejora que se ejecutará en el siguiente sprint; adicionalmente, se realiza la revisión de personas, relaciones, procesos, y herramientas con el objetivo de mejorar la efectividad del equipo.

2.6.2.3 Recursos Técnicos

(Beck et al., 2001)

Scrum utiliza un backlog del producto para almacenar toda la información necesaria sobre el producto. Esta es la única fuente de requerimientos para cualquier cambio que se realice al producto. El responsable de este recurso es el dueño del producto.

Durante la ejecución de un Sprint se crea un plan de cumplimiento del objetivo; determinando las tareas que serán realizadas junto con sus responsables. Este listado de tareas es definido como backlog del producto. El cumplimiento, según la planificación establecida, de todas las tareas aumentará la funcionalidad del producto de manera incremental.

La suma de todos los elementos del backlog del producto que han sido completados exitosamente dentro de todos los Sprints ejecutados se denomina incremento. Este recurso generalmente es tangible y sujeto periódicamente a revisiones y/o validaciones.

2.6.2.4 Elementos

(Beck et al., 2001)

Para la implementación de la metodología Scrum se debe utilizar varios elementos. El **Tablero** es el elemento donde se presenta el estado de las tareas junto con sus responsables y tiempos de ejecución. El **Flujo de Tareas** es el proceso por el cual una tarea pasa del estado inicial al estado completado. **La Herramienta o Software de Control** es utilizada para almacenar el backlog del producto y del sprint (ej. Jira). Finalmente, el **Calendario de Planificación y Eventos** es utilizado para agendar y llevar un buen control del tiempo sobre las fechas límites de entrega, reuniones, presentaciones, etc.

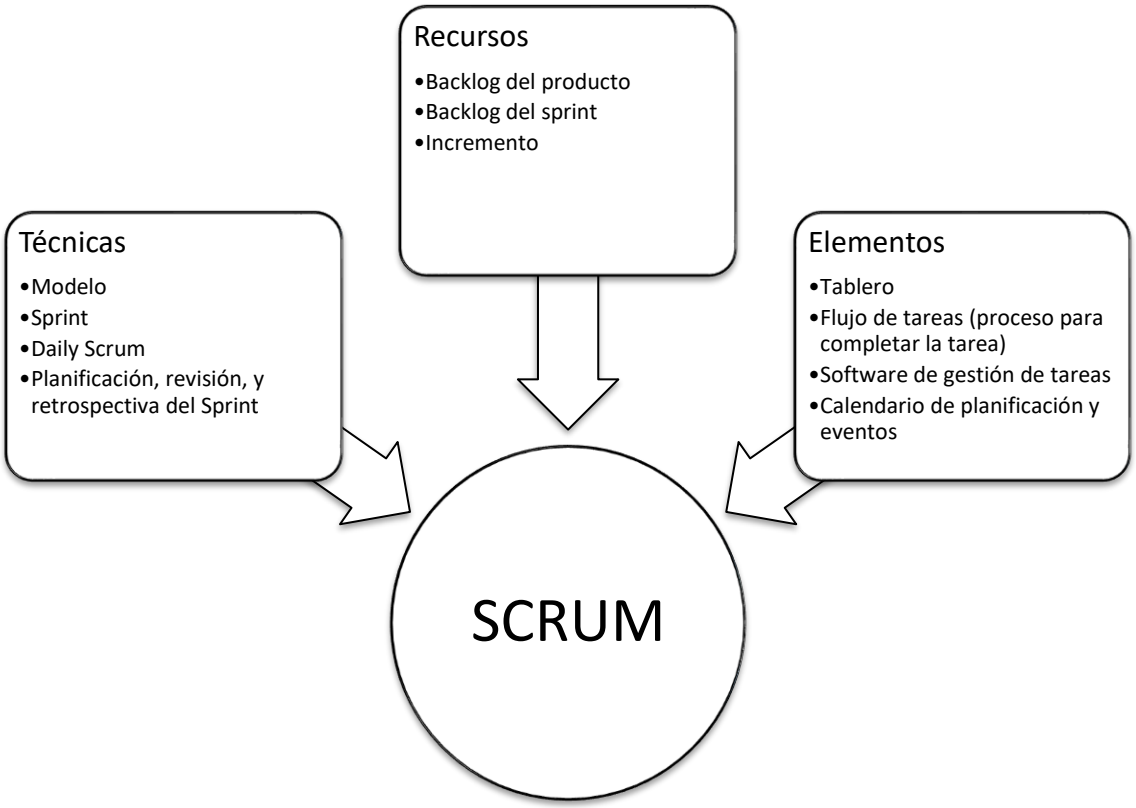


Figura 8. Técnicas, Recursos, y Elementos de la metodología SCRUM.

2.6.2.5 Modelo

(Beck et al., 2001) (SeaLights, 2018)

Scrum propone un modelo de implementación incremental iterativo. El proceso inicia con la planificación del sprint. En esta fase se define todo el trabajo que será realizado durante el período de tiempo acordado junto con el objetivo que se deberá cumplir al finalizar el sprint. Este objetivo es tangible y se lo denomina incremento (prototipo). Para obtener los resultados deseados, se desarrolla una plan que viabilizará la entrega del producto con los recursos

especificados. La duración máxima de esta fase es de máximo ocho horas para un sprint de un mes.

Una vez finalizada la fase de planificación, comienza la fase de ejecución del sprint. Durante esta fase se realizan todas las tareas definidas y planificadas que se encuentran en el backlog del sprint. Cada tarea tendrá asignada un responsable y un tiempo de desarrollo. Para evaluar el progreso de cada tarea y de sus responsables, se llevan a cabo reuniones diarias denominadas daily scrum, en las que se evalúa el progreso de cada tarea. Los responsables exponen el trabajo que han realizado, que van a realizar el día actual, y si es que existe algún impedimento que bloquea el avance. La duración máxima de esta reunión diaria es de 15 minutos.

Una vez terminado el sprint se procede a la fase de revisión. En esta fase se realiza una inspección del incremento, evaluando todo el trabajo realizado durante el sprint. Esta revisión incluye tareas que posiblemente no pudieron ser completadas por cualquier motivo al igual que las tareas que sí fueron cumplidas. Adicionalmente se exponen problemas y se buscan resultados o soluciones a los mismos. Esta fase también incluye la exposición de ideas para llevar a cabo en el siguiente sprint. El tiempo máximo de duración de esta fase es de 4 horas.

Para finalizar la iteración de Scrum, se ejecuta la fase de retrospectiva del sprint. El objetivo de esta fase es la inspección del equipo para la creación de un plan de mejoras que se llevará a cabo durante la ejecución del siguiente sprint. Se evalúan personas, relaciones, procesos y herramientas, elementos exitosos y mejoras potenciales.

Una vez finalizadas las cuatro fases de la metodología se regresa a la primera fase de planificación, donde se podrá definir el nuevo trabajo a realizar para añadir valor al incremento (prototipo) hasta completar los estándares de calidad y requerimientos definidos.

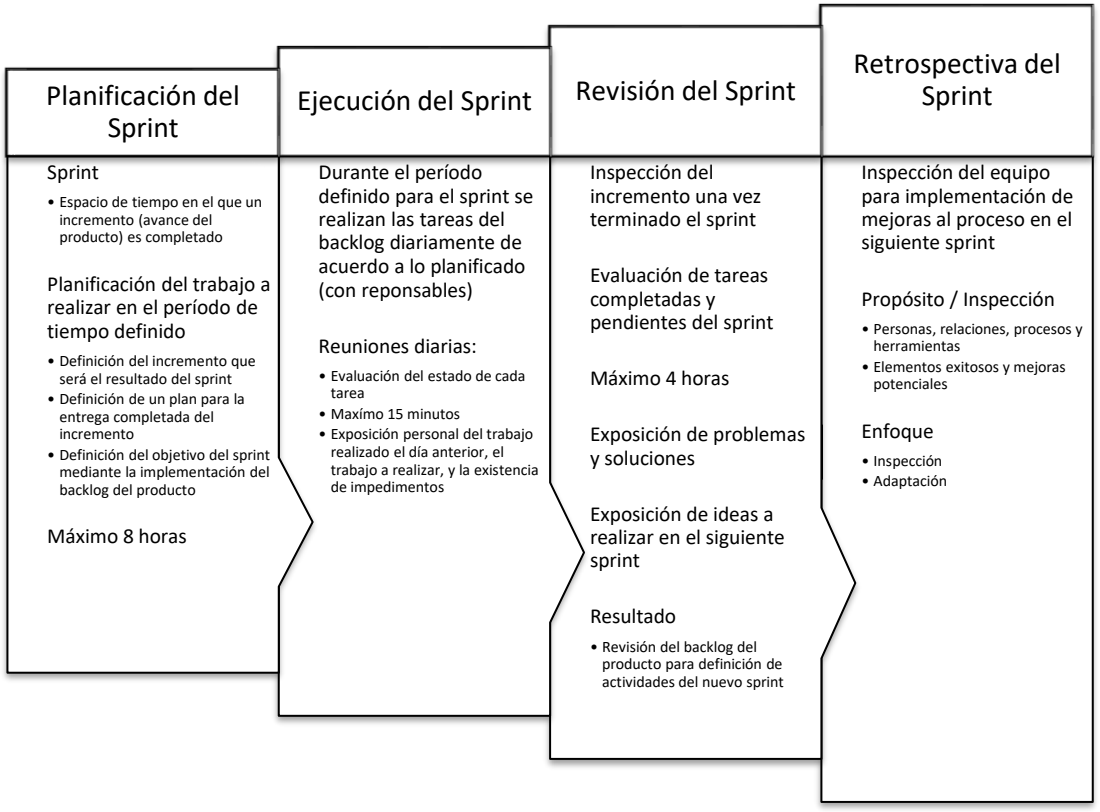


Figura 9. Modelo de la metodología SCRUM.

2.6.3 Factores Críticos

Tanto UWE como SCRUM son metodologías que proponen diferentes fases y procesos a aplicar. En cada fase es posible identificar factores críticos que permiten el desarrollo continuo y fluido de las metodologías. Se consideran factores críticos a los elementos indispensables, de mayor peso, o más influyentes dentro de cada fase de las metodologías.

En UWE podemos identificar varios factores críticos. Dentro de la fase de Modelo de Contenido son el diagrama entidad - relación y los requerimientos. La creación del diagrama de entidad - relación describe todos los requerimientos del sistema web. Una correcta definición de requerimientos permitirá la elaboración de un diagrama completo y descriptivo de la aplicación y esto a su vez servirá de insumo para la siguiente fase. En Modelo de Estructura de Navegación se define un solo factor crítico; el diagrama de navegación. Este diagrama deberá satisfacer todos los requerimientos de navegabilidad del sistema, y a su vez servirá de insumo para la fase siguiente. El modelo de presentación tiene como factor crítico el diagrama de presentación; el cual describe todas las pantallas y la interacción que el usuario tendrá con el sistema. En la fase final de Transformación de Modelos, se define al prototipo como factor crítico. Este elemento es el producto resultante de la ejecución de todas las fases anteriores, y contiene la información que ha sido recopilada en cada requerimiento y diagrama de todas las fases previas.

Tabla 7
Factores Críticos metodología UWE

Factores Críticos UWE			
Modelo Contenido	**Estructura de Navegación**	**Modelo de Presentación**	**Transformación de Modelos**
Diagrama entidad - relación	Diagrama de Navegación	Diagrama de Presentación	Prototipo
Requerimiento			

La metodología Scrum también define varios factores críticos dentro de cada fase. Durante la planificación, el objetivo del sprint y la estrategia, o plan de ejecución son los elementos más importantes. De esta forma se puede definir las diversas tareas que se podrán ejecutar para alcanzar el objetivo, junto con el plan para llevar a cabo el trabajo de la mejor manera. El fracaso en la definición del objetivo o en la elaboración de la estrategia llevará a una iteración

desperdiciada, consumiendo tiempo y recursos del proyecto. En la siguiente fase, la ejecución del sprint los roles más importantes los tienen el backlog del sprint junto con el incremento. La ejecución y cumplimiento de todas las tareas planificadas y almacenadas en el backlog del sprint permitirán que el incremento resultante cumpla con el objetivo planificado en la fase previa. En la fase de revisión del sprint, es muy importante identificar las tareas incompletas al igual que cualquier bloqueo que exista para las mismas o cualquier otro proceso. Se deberá distribuir las tareas incompletas de tal manera que se pueda recuperar el tiempo perdido, u optimizarlo de la mejor manera. Los bloqueos deberán ser tratados por los miembros del equipo para buscar la mejor alternativa de solución. En el caso de que existan tareas y bloqueos permanentes, el tiempo de ejecución del proyecto será mayor al planificado, resultando en mayor consumo de recursos, e inclusive el fracaso. Finalmente, en la fase de retrospectiva del sprint el factor crítico identificado es el plan de mejoras. En cada iteración de la metodología Scrum, se identifica potenciales mejoras que deben ser implementados, de la misma manera que problemas que deben ser solucionados. La elaboración e implementación de un plan de mejoras para la siguiente iteración, resultará en la mejora de la sinergia del equipo, aumentando de esta manera su afinidad, confianza, y productividad.

Tabla 8

Factores Críticos metodología SCRUM

Factores Críticos Scrum			
Planificación del Sprint	**Ejecución del Sprint**	**Revisión del Sprint**	**Retrospectiva del Sprint**
Objetivo del Sprint	Backlog del Sprint	Tareas incompletas del backlog del sprint	Plan de mejoras
Estrategia / Plan para completar el objetivo	Incremento	Bloqueos	

2.7 ESTÁNDARES

2.7.1 ISO/IEC/IEEE 29148:2011

Es un estándar para Ingeniería de Sistemas y Software, Procesos del Ciclo de Vida, e Ingeniería de Requerimientos. Su enfoque se encuentra en el tratamiento unificado de los procesos y productos involucrados en la ingeniería de requerimientos a lo largo del ciclo de vida de los sistemas y software. Para esto, especifica procesos que pueden ser implementados en la Ingeniería de Requerimientos, junto con los elementos informativos resultantes de la implementación de estos procesos (ISO/IEC/IEEE, 2011, págs. vii - 1).

2.7.1.1 Requerimiento

Se define al requerimiento como el enunciado que traduce o expresa una necesidad y sus condiciones o restricciones asociadas (ISO/IEC/IEEE, 2011, pág. 8).

2.7.1.1.1 Características

Un requerimiento posee características generales y específicas. Las características generales son que el requerimiento debe ser verificable, debe ser cumplido para solventar el problema de un stakeholder, debe ser calificado por condiciones medibles y limitado por restricciones, y debe definir el rendimiento del sistema cuando es utilizado por el usuario.

Las características específicas incluyen que el requerimiento debe ser necesario o indispensable, debe ser implementado independientemente defiendo el objetivo de su realización y no el proceso para cumplirlo, no debe ser ambiguo sino específico, debe ser consistente, completo, singular. factible, rastreable (seguimiento), y verificable.

Adicionalmente el requerimiento posee características grupales, es decir que el conjunto de requerimientos debe ser completo, consistente, asequible y congruentemente enlazado (ISO/IEC/IEEE, 2011, págs. 11, 12).

2.7.1.1.2 Condiciones del Lenguaje

Para la correcta definición de un requerimiento se deben cumplir algunas condiciones. Primero, el requerimiento deberá ser expresado en lenguaje natural; es decir con sujeto, verbo, y predicado; de manera que se especifique el sujeto del requerimiento (sistema, software, etc.) y lo que se debe realizar (operar con cierta capacidad, proveer campos para, etc.).

Dentro del uso de lenguaje natural intervienen condiciones más específicas como el uso de la palabra "deberá" para requerimientos mandatorios, evitar el uso de la palabra "deberá" en declaraciones no mandatorias, para las preferencias utilizar la palabra "debería", para las sugerencias utilizar la palabra "podría", y finalmente utilizar declaraciones positivas evitando en la medida de lo posible las negativas.

Para la definición de los requerimientos se puede utilizar tablas de acción - condición, o diagramas de casos de uso (ISO/IEC/IEEE, 2011, pág. 12).

2.7.1.1.3 Restricciones

Se debe considerar que en la definición de requerimientos para un sistema existen diversas restricciones como son las interfaces, limitaciones físicas (de espacio), leyes del país, tiempo de duración, presupuesto, plataformas tecnológicas preexistentes, y las capacidades y limitaciones del usuario (ISO/IEC/IEEE, 2011, pág. 57).

2.7.1.1.4 Atributos

Los requerimientos poseen diversos atributos como la identificación que se refiere al número, tag, o nombre único para cada uno, la prioridad que se puede definir en una escala de acuerdo a criterios específicos, las dependencias existentes entre requerimientos, el riesgo con sus consecuencias y nivel o grado, la fuente u origen del requerimiento, la razón fundamental que justifica la realización, y la dificultad de la implementación definida en una escala (ISO/IEC/IEEE, 2011, págs. 12 - 14).

2.7.1.1.5 Clasificación

El estándar clasifica a los requerimientos en funcionales, de rendimiento, de interface, de restricción de diseño, de proceso, no funcionales, y de factores humanos.

Los requerimientos funcionales son los elementos funcionales del sistema, o las tareas que el sistema ejecutará. Los requerimientos de rendimiento se refieren a la respuesta de una tarea específica del sistema bajo ciertas condiciones o circunstancias. Para esto se valida la usabilidad (calidad en el uso) y la satisfacción del usuario en cuanto al diseño y su evaluación del cumplimiento de las necesidades. Los requerimientos de interface incluyen a las externas e internas. Las interfaces externas hacen referencia a la interacción con otros sistemas, mientras que las interfaces internas a la interacción entre elementos del sistema. Los requerimientos de restricción de diseño se refieren a la arquitectura definida para el sistema. Los requerimientos de proceso son definidos en base a la conformidad con las leyes, requerimientos administrativos, entre otros. Y por último los requerimientos no funcionales se refieren a los requerimientos de calidad como transportabilidad, flexibilidad, usabilidad, etc. y a los requerimientos de factores humanos como la eficiencia, rendimiento satisfacción, etc. (ISO/IEC/IEEE, 2011, págs. 13 - 14).

2.7.1.2 Procesos

El estándar sugiere la aplicación de algunos procesos de manera iterativa o recursiva. Para la aplicación iterativa se requiere la repetición de un proceso o conjunto de procesos al mismo nivel del sistema. De esta manera las preguntas se realizarán en base a los requerimientos, análisis de riesgos, y oportunidades. El objetivo de esta aplicación es asegurar la calidad de la información previo al siguiente paso o proceso (ISO/IEC/IEEE, 2011, págs. 14 - 15).

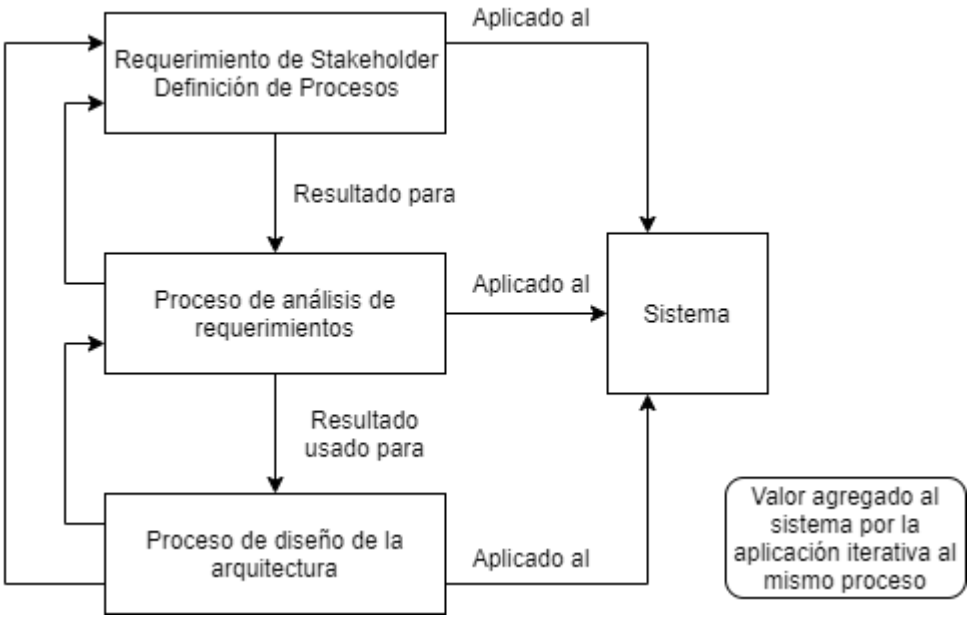


Figura 10. Aplicación iterativa de procesos ISO/IEEE 29148:2011

Por otra parte, la aplicación recursiva requiere que un proceso o conjunto de procesos se repita en los niveles sucesivos de los elementos del sistema dentro de la estructura del sistema. (ISO/IEC/IEEE, 2011, págs. 15 - 16)

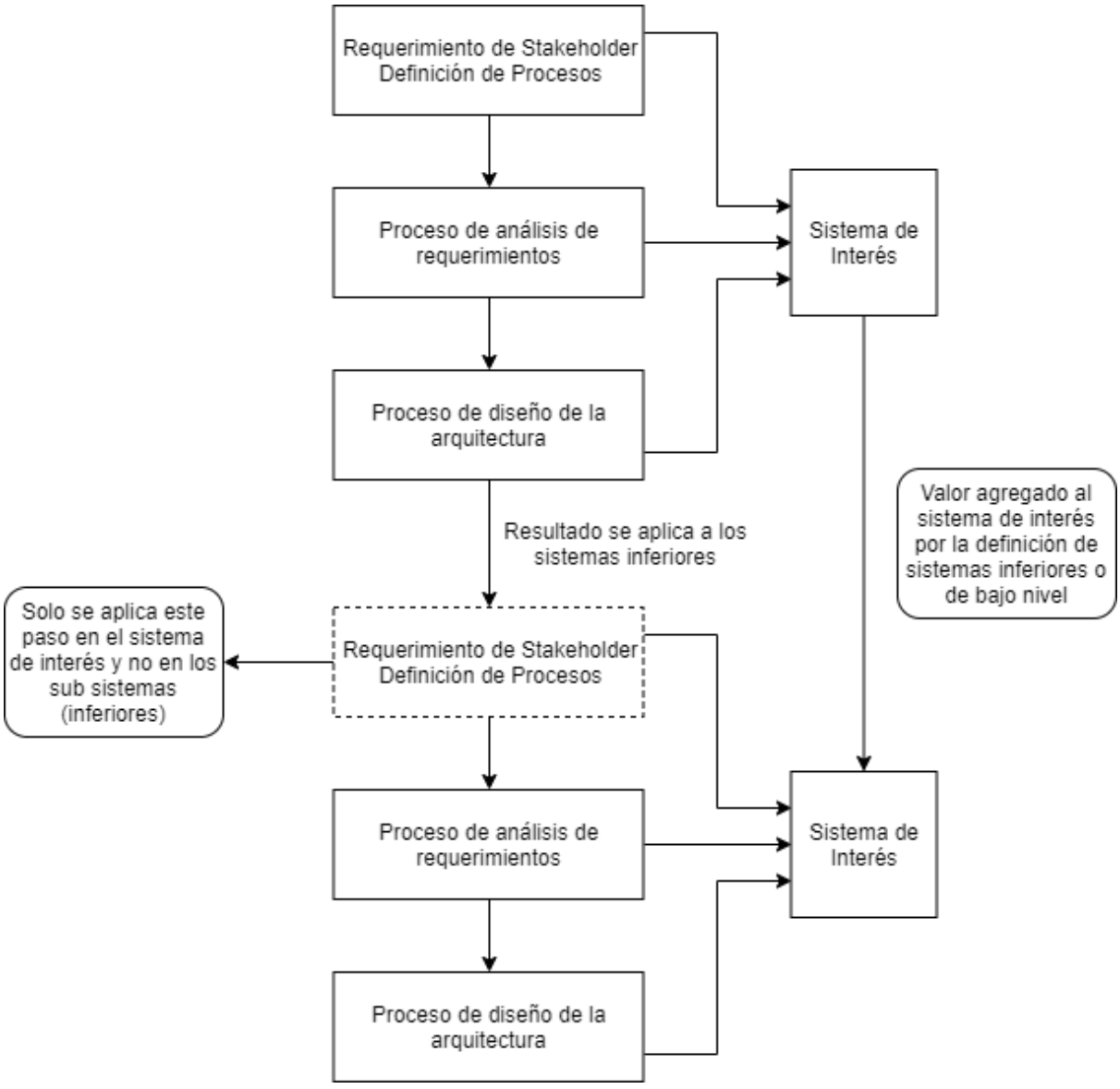


Figura 11. Aplicación recursiva de procesos ISO/IEEE 29148:2011

2.7.1.2.1 Definición de Requerimientos de Stakeholders

El objetivo de este proceso es definir los requerimientos del sistema que provean los servicios necesitados por los stakeholders en un entorno definido. El resultado de este proceso es el conjunto de las características del sistema y su contexto de uso, restricciones del sistema,

trazabilidad de requerimientos a necesidades, definición de requerimientos de stakeholders, y validación de estos requerimientos (ISO/IEC/IEEE, 2011, pág. 19).

2.7.1.2.2 Análisis de Requerimientos

El objetivo de este proceso es transformar todos los requerimientos de los stakeholders a una visión técnica del sistema que pueda entregar los servicios necesitados. El resultado de la aplicación de este proceso es un conjunto de las características, atributos, y requerimientos funcionales y de rendimiento, restricciones para el diseño de la arquitectura y los medios para realizar el sistema, integridad y trazabilidad de requerimientos del sistema a requerimientos de stakeholders, y definición de una base para determinar la satisfacción con los requerimientos (ISO/IEC/IEEE, 2011, pág. 27).

2.7.1.2.3 Otros procesos relacionados con Requerimientos Técnicos

Los procesos adicionales incluidos son el **Diseño de Arquitectura** cuyo objetivo es sintetizar una solución que satisfaga los requerimientos del sistema, la **Verificación** cuyo objetivo es confirmar que los requerimientos de diseño son cumplidos y completados en el sistema, y la Validación cuyo objetivo es proveer evidencia objetiva de que los servicios provistos por el sistema están en conformidad con los requerimientos de los stakeholders. (ISO/IEC/IEEE, 2011, págs. 33 - 36)

2.7.1.3 Documentos Finales

Una vez ejecutados los procesos definidos por el estándar, el resultado será un conjunto de documentos que contienen toda la información obtenida respecto al desarrollo del sistema. Los documentos son Especificación de Requerimientos de Stakeholders, Especificación de

Requerimientos del Sistema, y Especificación de requerimientos de software (ISO/IEC/IEEE, 2011, pág. 41).

2.7.2 ISO/IEC/IEEE 12207:2017

Es un estándar para Ingeniería de Sistemas y Software, y Procesos del Ciclo de Vida. Su enfoque es ofrecer procesos a ser ejecutados durante la adquisición, desarrollo, operación, mantenimiento, etc. del sistema con el propósito de alcanzar la satisfacción del cliente. Este estándar ofrece diversos procesos que pueden ser empleados para definir, controlar, y mejorar los procesos del ciclo de vida del software dentro de una organización o proyecto (ISO/IEC/IEEE, 2017, págs. vii - 1).

2.7.2.1 Sistema de Software

Es un sistema en el que el software es fundamental o de mayor impacto para las partes interesadas. El objetivo es proveer productos o servicios en ambientes predeterminados para beneficio de los usuarios y stakeholders. Su estructura está compuesta por elementos que interactúan entre sí para cumplir con los requerimientos especificados (ISO/IEC/IEEE, 2017, págs. 13 - 15).

2.7.2.2 Procesos

Los procesos definidos por este estándar se clasifican en cuatro grupos que son: Procesos de Acuerdos, Procesos Organizacionales para Habilitación de Proyectos, Procesos de Gestión Técnica, y Procesos Técnicos (ISO/IEC/IEEE, 2017, pág. 24).

2.7.2.2.1 Procesos de Acuerdos

Los procesos definidos dentro de esta agrupación son: Proceso de Adquisición, y Proceso de Suministro (ISO/IEC/IEEE, 2017, pág. 24).

2.7.2.2.2 Procesos Organizacionales para Habilitación de Proyectos

Los procesos definidos dentro de esta agrupación son: Proceso de Gestión del Modelo del Ciclo de Vida, Proceso de Gestión de Infraestructura, Proceso de Gestión de Portafolio, Proceso de Gestión de Recurso Humano, Proceso de Gestión Calidad, y Proceso de Gestión de Conocimiento (ISO/IEC/IEEE, 2017, págs. 28, 29).

2.7.2.2.3 Procesos de Gestión Técnica

Los procesos definidos dentro de esta agrupación son: Proceso de Planificación de Proyecto, Proceso de Control y Evaluación de Proyecto, Proceso de Gestión de Decisiones, Proceso de Gestión de Riesgos, Proceso de Gestión de Configuración, Proceso de Gestión de Información, Proceso de Medición, y Proceso de Aseguramiento de la Calidad (ISO/IEC/IEEE, 2017, págs. 37, 38).

2.7.2.2.4 Procesos Técnicos

Los procesos definidos dentro de esta agrupación son: Análisis de la Misión o Negocio, Proceso de Definición de Requerimientos y Necesidades de Stakeholders, Proceso de Definición de Requerimientos de Sistemas y Software, Proceso de Definición de Arquitectura, Proceso de Definición de Diseño, Proceso de Análisis de Sistema, Proceso de Implementación, Proceso de Integración, Proceso de Verificación, Proceso de Transición, Proceso de Validación, Proceso de

Operación, Proceso de Mantenimiento, y Proceso de Eliminación (ISO/IEC/IEEE, 2017, págs. 55, 56).

CAPITULO III - ELABORACIÓN DE LA METODOLOGÍA PROPUESTA

3.1 FACTORES CRÍTICOS

El objetivo de la creación de una propuesta metodológica es obtener un software de calidad mediante la selección de actividades específicas de las metodologías revisadas previamente, junto con la integración de procesos definidos en los estándares ISO/IEEE. Y como se mencionó anteriormente, la calidad del software dependerá del cumplimiento de los requerimientos definidos y será medida de acuerdo con las principales métricas para el desarrollo del software.

Para esto se ha identificado factores críticos que se enfocan en el cumplimiento de estas métricas. Los elementos son: tiempo de desarrollo, correcta definición de requerimientos, comprensión de requerimientos, y modularidad del software.

3.1.1 Tiempo de Desarrollo

La evolución de las metodologías ágiles ha permitido una mayor interacción del usuario con el proceso de desarrollo del software mediante la visualización de los avances y evolución del producto de software. Por esta razón, la estimación del tiempo de desarrollo ya no solamente debe realizarse para el proyecto en su totalidad, sino también para un conjunto de requerimientos que podrán ser agrupados, cumplidos y presentados al cliente de forma continua.

SCRUM, una metodología ágil que plantea la ejecución del trabajo en períodos de tiempo para cumplir objetivos definidos, permite que el software sea revisado, evaluado, y validado durante la ejecución del proyecto de manera incremental. Así, se podrá identificar y corregir errores,

redefinir procesos, e incluir nuevos requerimientos, optimizando el uso de recursos durante el desarrollo y reduciendo el impacto en el tiempo que estos factores tendrían al ser identificados al final de la implementación.

El enfoque de este factor es cumplir con las métricas de calidad de **eficiencia**, al generar un software en el tiempo esperado y con la optimización de los recursos disponibles; y **capacidad de prueba**, al realizar validaciones periódicas de los avances alcanzados a lo largo del tiempo.

3.1.2 Correcta Definición de Requerimientos

En capítulos anteriores se mencionó la importancia de una correcta definición de requerimientos y su impacto en la calidad final del producto. Es indispensable que los requerimientos expresen claramente las necesidades del cliente respecto del software que será implementado.

Por esta razón, los procesos de Definición de Requerimientos del Sistema, y Definición de Requerimientos del Software del estándar ISO/IEEE 29148 ayudarán con el objetivo de determinar clara y objetivamente los requerimientos de manera organizada y bajo procesos que han sido validados a través del tiempo.

El enfoque de este factor es cumplir con las métricas de calidad de **confiabilidad** y **exactitud**, al generar un software estable que tenga el comportamiento esperado y produzca el resultado correcto de acuerdo con las definiciones de requerimientos.

3.1.3 Comprensión de Requerimientos

Una vez realizada la definición de requerimientos es indispensable que la comprensión de estos sea la misma para cada miembro del equipo. UML (Lenguaje de Modelado Unificado)

permitirá mantener un estándar para el modelado de diagramas, en donde cada elemento mantendrá su significado durante las diferentes fases de desarrollo y a lo largo del sistema.

UWE, metodología de desarrollo web basada en UML, permitirá esta estandarización, facilitando la comprensión de los diferentes requerimientos, sus interacciones, y relaciones mediante el uso de diferentes diagramas durante el ciclo de vida del software, que están definidos en el modelo de su metodología.

El enfoque de este factor es cumplir con las métricas de calidad de **usabilidad**; para facilitar la interacción del usuario en el sistema; y **reusabilidad**; al permitir el uso de diferentes diagramas y componentes a lo largo del sistema.

3.1.4 Modularidad del Software

La modularidad del software se define como el grado en el que un sistema está compuesto de diferentes componentes de manera que la modificación de uno tiene un impacto mínimo para el resto (ISO/IEC/IEEE, 2010).

Para lograr que un sistema sea modular se necesita realizar una correcta definición y diseño de la arquitectura. Los estándares ISO/IEC/IEEE 29148 e ISO/IEC/IEEE 12207, junto a sus procesos para la definición de la arquitectura de un sistema permitirán la creación de un sistema modular.

El enfoque de este factor es cumplir con las métricas de calidad de **modificabilidad**; facilitando la modificación del sistema en el caso de que existan cambios en los requerimientos; **portabilidad**; al permitir que el sistema pueda ser utilizado en diversas computadoras y sistemas operativos; y **mantenibilidad**, permitiendo una codificación estructurada y de fácil comprensión.

3.2 HIPÓTESIS Y TESIS

En base a los factores críticos identificados se han elaborado hipótesis con sus respectivas tesis que se presentan en la siguiente tabla.

Tabla 9

Hipótesis y tesis a partir de estándares y metodologías

Hipótesis	Tésis
¿Podrá disminuir el tiempo de desarrollo la inclusión de SCRUM en la metodología?	Scrum permitirá reducir tiempos de desarrollo por su propuesta de definición periódica de objetivos; que mediante la creación de una estrategia de ejecución y delimitación de alcance (priorizando el tiempo y alineación con los requerimientos levantados) podrá optimizar el uso de recursos en el cumplimiento de este objetivo sin desviaciones de la meta trazada.
¿Podrá la aplicación de estándares durante el levantamiento de requerimientos optimizar tiempo y recursos en el proceso de desarrollo del proyecto, generando un sistema más apegado a los requerimientos de los stakeholders?	La aplicación de los procedimientos, especificados en los estándares, durante el levantamiento de requerimientos permitirá generar desarrollos incrementales objetivos, que en conjunto formarán un producto final satisfactorio para los stakeholders y en conformidad a sus requerimientos.
¿Podrá el uso del lenguaje UML, sugerido por UWE, facilitar la comprensión del sistema por los miembros del equipo?	La aplicación de un lenguaje claro, estandarizado, y con elementos predefinidos permitirá la compresión del sistema, en los diferentes niveles, transmitiendo la misma interpretación y significado a todo el equipo, y evitando así las confusiones.
¿Podrá la aplicación de estándares en la definición de la arquitectura mejorar la modularidad del software?	La definición de la arquitectura mediante el uso de procedimientos definidos en los estándares mejorará la modularidad del sistema permitiendo optimizar y crear y estandarizar componentes (usados en distintas situaciones), y agilizar el mantenimiento de cada módulo.

3.3 DESARROLLO

Para la ejecución de la metodología propuesta se definen dos fases; primero la fase de desarrollo del proyecto para describir las actividades a realizar durante este proceso, y segundo, la fase de aplicación del modelo metodológico.

3.3.1 Desarrollo del Proyecto

La fase de desarrollo del proyecto describe el proceso que se llevará a cabo desde el inicio del proyecto hasta su culminación. Las diferentes actividades que serán ejecutadas se describen a continuación.

La fase de desarrollo del proyecto iniciará con una entrevista al cliente para determinar las diferentes necesidades y requerimientos del sistema. Una vez obtenidos los insumos iniciales, se procederá a realizar un modelo de contexto con el fin de plasmar los requerimientos de manera visual. Este modelo será explicado al cliente para validar la información obtenida durante la entrevista, y con su aprobación se procederá a la aplicación del modelo metodológico.

Durante la ejecución de la segunda fase, se podrá agendar reuniones con el cliente para revisiones y avances del sistema. Estas serán periódicas y tendrán como objetivo presentar la evolución del sistema a lo largo del tiempo, y refinar detalles que no estén completamente claros.

Una vez completado el sistema se realizará una presentación final al cliente para mostrar el completo funcionamiento del sistema, validar la funcionalidad requerida, y dar por completado el proyecto.

3.3.2 Modelo Metodológico

Mediante el análisis previo de las metodologías de desarrollo y estándares ISO/IEEE, junto con la identificación de los diferentes factores críticos, se propone combinar las metodologías UWE y SCRUM durante el ciclo de vida de desarrollo del sistema, y aplicar procesos específicos de los estándares a las diferentes fases de desarrollo del sistema como se muestra en el siguiente gráfico.

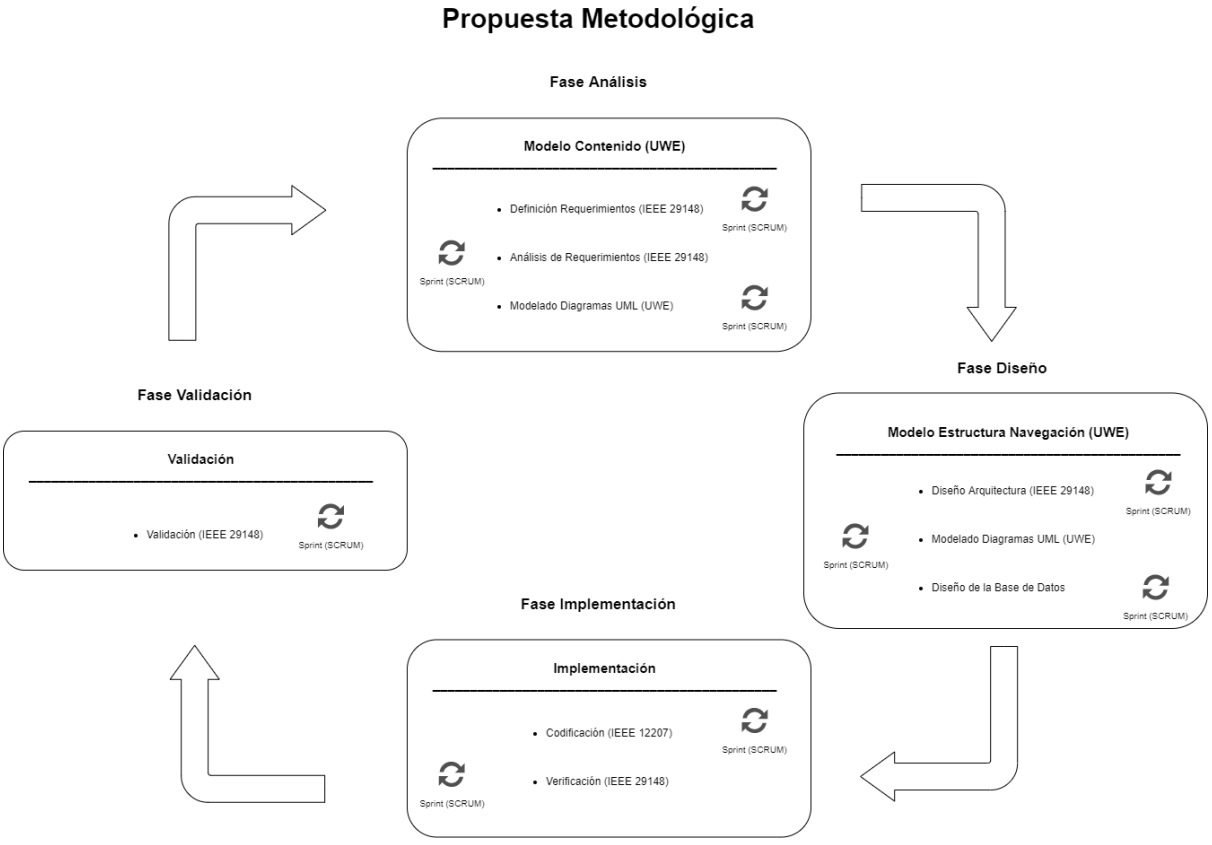


Figura 12. Propuesta Metodológica

Durante la fase de Análisis se propone realizar la definición de requerimientos utilizando el proceso del estándar IEEE 29148, el análisis de requerimientos utilizando el proceso del estándar

IEEE 29148, y realizar el modelado de los diagramas de casos de uso (UWE) y el modelo de contenido (UWE). Para cada elemento a ser desarrollado se cuenta con uno o varios Sprints (SCRUM) donde se podrán definir los objetivos a alcanzar hasta finalizar con cada elemento.

En la fase de Diseño se plantea diseñar la arquitectura utilizando el proceso del estándar IEEE 29148, y realizar los diagramas de navegación y de procesos (UWE). Cada elemento podrá ser desarrollado en uno o varios Sprints (SCRUM) de acuerdo con la necesidad.

En la fase de implementación se propone realizar la codificación utilizando procesos del estándar IEEE 12207. Una vez terminado el desarrollo se procederá a la verificación utilizando el proceso del estándar IEEE 29148. Durante el desarrollo y la verificación se podrá hacer uso de los Sprints (SCRUM) necesarios.

Finalmente, para la fase de validación se plantea la aplicación de un proceso definido en el estándar IEEE 29148 durante la ejecución de uno o varios Sprints (SCRUM).

En el caso de que existan nuevos requerimientos, estos podrán ser añadidos desde la fase de análisis, utilizando uno o varios Sprints (SCRUM) según sea necesario.

3.3.2.1 Fase Análisis

Etapa inicial para la definición de los requerimientos, y creación de diagramas y demás insumos necesarios para el desarrollo del sistema.

3.3.2.1.1 Definición de Requerimientos

Aplicación del proceso de definición de requerimientos del estándar ISO/IEEE 29148 durante uno o varios Sprints de SCRUM.

3.3.2.1.2 Análisis de Requerimientos

Aplicación del proceso de análisis de requerimientos del estándar ISO/IEEE 29148 durante uno o varios Sprints de SCRUM.

3.3.2.1.3 Modelado de Diagramas UML

Creación del modelo de contenido requerido por UWE a partir de la creación de los diagramas de casos de uso en notación UML durante uno o varios Sprints de SCRUM.

3.3.2.2 Fase Diseño

Etapa de elaboración de la arquitectura y diagramas descriptivos del funcionamiento que tendrá el sistema.

3.3.2.2.1 Diseño de Arquitectura

Aplicación del proceso de definición de arquitectura del estándar ISO/IEEE 29148 durante uno o varios Sprints de SCRUM.

3.3.2.2.2 Modelado de Diagramas UML

Creación de los modelos UML de navegación y de procesos requeridos por UWE utilizando como insumo el modelo de contenido, obtenido de la fase previa, durante uno o varios Sprints de SCRUM.

3.3.2.2.3 Diseño Base de Datos

A partir del modelo de contenido se puede diagramar el esquema de la base de datos, y el script SQL para su creación.

3.3.2.3 Fase Implementación

Etapa de generación de código para materializar los procesos anteriores de análisis de requerimientos y de diseño del sistema.

3.3.2.3.1 Definición de Estrategia

Aplicación de la actividad de definición de estrategia descrito en el proceso de implementación del estándar ISO/IEEE 12207 durante un sprint de SCRUM.

3.3.2.3.2 Codificación

Aplicación de procesos de implementación del estándar ISO/IEEE 12207 durante varios Sprints de SCRUM para la codificación del Sistema de Gestión de Conjunto Habitacional, en base a los requerimientos, diagramas, y modelos obtenidos de las fases previas.

3.3.2.3.3 Verificación

Aplicación del proceso de verificación del estándar ISO/IEEE 29148 durante uno o varios Sprints de SCRUM.

3.3.2.4 Fase Validación

Etapa de revisiones finales y de conformidad a lo acordado al inicio del desarrollo del proyecto.

3.3.2.4.1 Validación

Aplicación del proceso de validación del estándar ISO/IEEE 29148 durante un Sprint de SCRUM.

CAPITULO IV - DESARROLLO DEL SISTEMA BAJO LA METODOLOGÍA PROPUESTA

4.1 DESCRIPCIÓN DEL SISTEMA

4.1.1 Entrevista

La metodología propuesta plantea el uso de una entrevista para el levantamiento inicial de requerimientos. El resultado de esta actividad permitirá definir un modelo de contexto, que será la base para la definición de requerimientos, creación de flujos de procesos y definición de la arquitectura del sistema. La entrevista se fundamentó en identificar el problema existente, las características y procesos del negocio, las regulaciones y restricciones existentes, y los encargados de la ejecución de las diferentes actividades del proceso.

El formato de la entrevista que se utilizó para la entrevista se encuentra en el **Anexo 1**. Mediante el uso de esta herramienta se determinó lo siguiente. El proyecto consiste en la elaboración un sistema que permita gestionar y administrar unidades habitacionales que pertenezcan a un conjunto a un edificio.

El ingreso de la información al sistema será realizado por un único usuario Administrador, quién es el responsable de la gestión del conjunto habitacional. La información que será ingresada pertenecerá al conjunto habitacional, unidades habitacionales, áreas comunales, condóminos, y cuotas, y será utilizada para fines legales de transacciones con los bienes inmuebles.

El proceso de gestión de conjuntos habitacionales está normado por la ley de propiedad horizontal. También existe un reglamento interno que no puede contradecir a esta ley, pero que

permite definir normativas internas que regulan acciones realizadas por los condóminos dentro del conjunto habitacional para garantizar el buen uso de las instalaciones, la seguridad de las personas, entre otros.

Finalmente, la información almacenada en el sistema permitirá al administrador tomar decisiones con el fin de garantizar la correcta operatividad del conjunto habitacional con todos sus elementos.

4.1.2 Modelo de Contexto

El modelo de contexto fue desarrollado considerando que debe existir una gestión o administración de varios componentes, los cuales, al ser utilizados en conjunto, formarán la solución a las necesidades planteadas por el cliente.

Con la información obtenida en la entrevista se identificó que el sistema debe consistir de la automatización de los procesos de gestión de conjunto habitacional, de unidades habitacionales, de propietarios, de arrendatarios, de áreas comunales, y de cuotas. Mediante el uso de toda la información ingresada se podrá evidenciar el historial de pagos de cuotas ordinarias y extraordinarias, unidades habitacionales en mora, entre otros reportes, que facilitan el trabajo del administrador del conjunto.

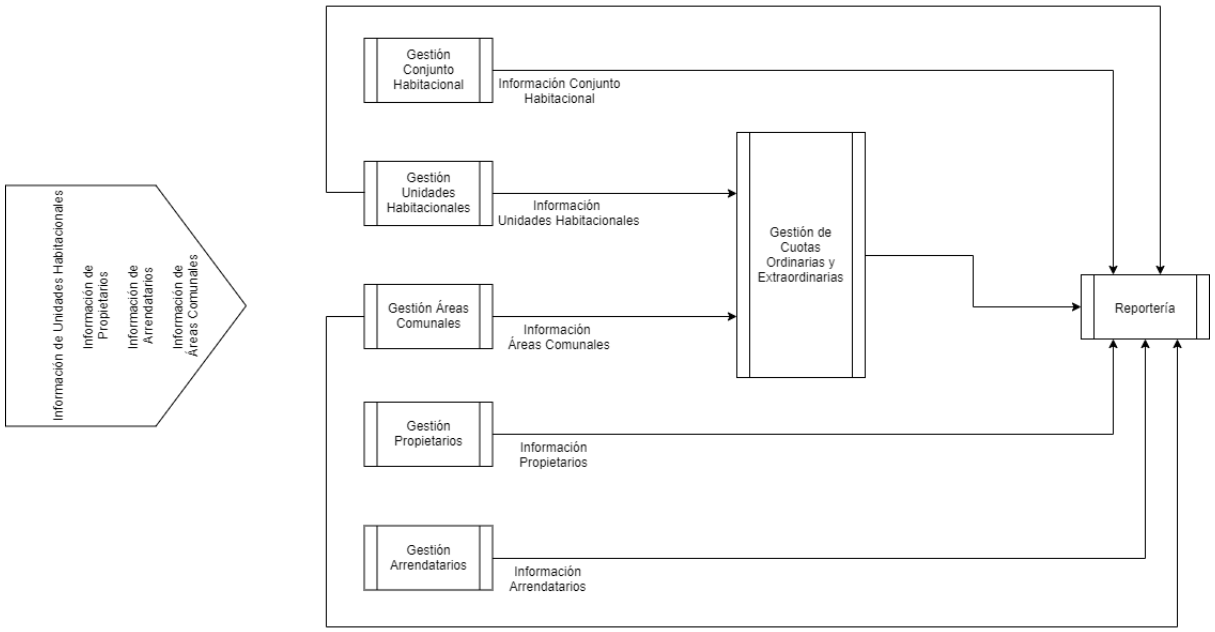


Figura 13. Modelo de Contexto

4.1.3 Identificación del Problema

Eduardo de Jesús Cerecer (Ibarra, 2016), arquitecto de ITESO – Universidad Jesuita de Guadalajara, y Félix Bombarolo (CF+S, 1997), máster en hábitat y vivienda, coinciden en la existencia de diversos problemas de gestión dentro de los conjuntos habitacionales, y proponen la implementación de buenas prácticas para mejorar la organización y coordinación de las diferentes actividades ejecutadas.

En base a encuestas realizadas en diferentes urbanizaciones y edificios ubicados en diferentes partes de la ciudad (Conjunto San Martín – Sector Rumipamba, Torre San Francisco – Sector 6 de Diciembre, Edificio Bujase – Sector La Gasca, Urbanización La Granja – Sector Mariana de Jesús.) se ha identificado que la gestión de conjuntos habitacionales, adicional a los problemas de

gestión, utiliza procesos netamente manuales, sin ningún control, y sin ninguna supervisión. El administrador, quien es la persona encargada de realizar todas las actividades referentes registro de condóminos, áreas comunales, cuotas, entre otros, registra la información, en el mejor de los casos en hojas de cálculo, y en el peor de los casos en hojas de papel.

Por estas razones, la implementación de un Sistema de Conjunto Habitacional, que permita el registro de información de todas estas actividades, facilitará la gestión realizada por el administrador y pondrá a disposición toda la información referente al control de servicios, áreas comunales, unidades habitacionales, y personas.

4.1.4 Objetivo

El objetivo de implementar un sistema de Gestión de Conjunto Habitacional es definir condiciones y características de desarrollo que permitan validar la metodología propuesta en este proyecto. Adicionalmente se deberá identificar las ventajas y desventajas de una las diferentes metodologías, en contraste con la metodología propuesta, a fin de evidenciar los factores críticos de cambio o afectación sobre el proceso de desarrollo.

4.1.5 Alcance

El sistema de Gestión de Conjunto Habitacional constará de la gestión de conjuntos, de unidades, de propietarios, de arrendatarios, de áreas comunales, y de cuotas. El registro será realizado por el Administrador del conjunto, quién podrá visualizar toda la información en el sistema, y socializarla de acuerdo con sus necesidades y por los medios apropiados según la normativa del conjunto. Adicionalmente existirán reportes para la visualización de las cuotas

pagadas o adeudadas por las unidades habitacionales, visualización de los diferentes propietarios y arrendatarios asignados a unidad habitacional, y visualización del estado de las diferentes áreas comunales.

4.1.6 Supuestos y Riesgos

El desarrollo del Sistema de Gestión de Conjunto Habitacional se basa en los siguientes supuestos obtenidos de la entrevista inicial realizada al cliente. Primero, el administrador será el responsable de ingresar y gestionar la información de forma manual a lo largo del sistema. Segundo, existirá solamente un administrador por cada conjunto, sin existir la posibilidad de que la misma persona esté encargada de varios conjuntos. Tercero, el conjunto habitacional debe estar regulado por una normativa interna donde necesariamente se defina el valor de las cuotas mensuales (alícuotas) proporcionales que serán cobradas a cada unidad habitacional. Finalmente, la información ingresada por el administrador será fiel y coherente, evitando por todos los medios necesarios cualquier tipo de fraude o ilegalidad.

De la misma manera algunos de los posibles riesgos existentes durante el uso del Sistema de Gestión de Conjunto Habitacional se describen a continuación. Primero, la información almacenada en el sistema puede no reflejar la situación real del conjunto debido a la mala gestión del Administrador, ya sea por omisión, o cualquier tipo de ilegalidad. Segundo, las normativas internas de diferentes conjuntos pueden no ser compatibles con los requerimientos iniciales forzando la adaptación del sistema a nuevas políticas de administración. Tercero, el administrador podría ser una persona no acostumbrada al uso de tecnología; en este caso computadoras, por lo que la gestión del conjunto se tornaría aún mas complicada. Finalmente, los condóminos podrían no estar de acuerdo en el uso del sistema después de poco tiempo de uso,

por lo que la información almacenada no sería útil para revisar la situación real del conjunto habitacional.

4.2 METODOLOGÍA PROPUESTA

4.2.1 Fase Análisis

4.2.1.1 Definición de Requerimientos

4.2.1.1.1 Documento de Especificación de Requerimientos del Sistema (SyRS – ISO/IEEE 29148)

4.2.1.1.1.1 Propósito del Sistema

El sistema de Gestión de Conjunto Habitacional es desarrollado con el fin de facilitar las actividades y responsabilidades del administrador de un conjunto, y obtener información pertinente respecto a los propietarios, arrendatarios, unidades habitacionales, áreas comunales, y cuotas, para la toma de decisiones.

4.2.1.1.1.2 Alcance

El sistema de Gestión de Conjunto Habitacional constará de la gestión de conjuntos, de unidades, de propietarios, de arrendatarios, de áreas comunales, y de cuotas. Adicionalmente se podrá visualizar información en formato de reporte respecto a la gestión realizada por el Administrador. Estos reportes incluirán la visualización de las cuotas pagadas o adeudadas por las unidades habitacionales, visualización de los diferentes propietarios y arrendatarios asignados a unidad habitacional, y la visualización del estado de las diferentes áreas comunales.

4.2.1.1.1.3 Descripción General del Sistema

4.2.1.1.1.3.1 Contexto del Sistema

El sistema de gestión de Conjunto Habitacional consiste en la automatización de los procesos de gestión de conjunto habitacional, de unidades habitacionales, de propietarios, de arrendatarios, de áreas comunales, y de cuotas. Mediante el uso de toda la información ingresada se podrá evidenciar el historial de pagos de cuotas ordinarias y extraordinarias, unidades habitacionales en mora, entre otros reportes, que facilitan el trabajo del administrador del conjunto. (Referirse a la **Figura 13**).

4.2.1.1.1.4 Requerimientos del Sistema

4.2.1.1.1.4.1 Requerimientos Funcionales

Los requerimientos funcionales del Sistema de Gestión de Conjunto Habitacional se muestran en la siguiente tabla.

Tabla 10
Requerimientos Funcionales del Sistema

Requerimientos Funcionales	
No. Requerimiento Funcional	**Descripción Requerimiento Funcional**
RF1	El administrador podrá registrar la información del Conjunto Habitacional para la cual se generarán sus asociaciones requeridas.
RF2	El administrador podrá editar la información del Conjunto Habitacional para actualizar los datos que sean necesarios.
RF3	El administrador podrá visualizar la información del Conjunto Habitacional.
RF4	El administrador podrá registrar Unidades Habitacionales asociadas al conjunto habitacional.
RF5	El administrador podrá editar la información de Unidades Habitacionales para corregir errores, o actualizar información de propietarios, arrendatarios, cuotas, etc.

Continúa

RF6	El administrador podrá visualizar la información de Unidades Habitacionales.
RF7	El administrador podrá registrar Propietarios asociados a una o varias unidades habitacionales.
RF8	El administrador podrá editar la información de Propietarios para corregir o actualizar datos.
RF9	El administrador podrá visualizar la información de Propietarios.
RF10	El administrador podrá eliminar Propietarios en el caso de que la unidad habitacional haya cambiado de propietario.
RF11	El administrador podrá registrar Arrendatarios asociados a una o varias unidades habitacionales.
RF12	El administrador podrá editar la información de Arrendatarios para corregir o actualizar datos.
RF13	El administrador podrá visualizar la información de Arrendatarios.
RF14	El administrador podrá eliminar Arrendatarios en el caso de que la unidad habitacional haya sido desocupada o arrendada a una persona diferente.
RF15	El administrador podrá registrar Áreas Comunales asociadas al conjunto habitacional.
RF16	El administrador podrá editar la información de Áreas Comunales para corregir o actualizar datos.
RF17	El administrador podrá visualizar la información de Áreas Comunales.
RF18	El administrador podrá registrar Cuotas relacionadas a las unidades habitacionales. En el caso de alícuota, el cálculo del valor será automático de acuerdo con la unidad habitacional asociada.
RF19	El administrador podrá editar la información de Cuotas mientras estas no hayan sido pagadas o eliminadas.
RF20	El administrador podrá visualizar la información de Cuotas.
RF21	El administrador podrá eliminar Cuotas.
RF22	El administrador podrá visualizar Reportes con la información correspondiente a la gestión del conjunto habitacional.

4.2.1.1.1.4.2 Requerimientos No Funcionales

Los requerimientos no funcionales del Sistema de Gestión de Conjunto Habitacional se muestran en la siguiente tabla.

Tabla 11
Requerimientos No Funcionales del Sistema

Requerimientos No Funcionales	
Requerimiento No Funcional	**Descripción Requerimiento No Funcional**
Desempeño	Corto tiempo de respuesta de las peticiones del usuario al servidor.
Escalabilidad	Fácil agregación de nuevos módulos en el caso de ser necesario a futuro.
Disponibilidad	Sistema accesible a través de un navegador y conexión a internet en cualquier momento.
Confiabilidad	Información consistente y coherente dentro de todo el sistema.
Tolerancia a Fallos	Información resguardada en el caso de fallas del servidor o del sistema.
Mantenibilidad	Actualizaciones del software, librerías, ambiente, etc. veloces y transparentes para el usuario.

4.2.1.1.1.4.3 Características Físicas

Debido a los requerimientos de disponibilidad, el sistema de gestión de Conjunto Habitacional deberá ser de tipo y web, y será instalado en un servidor en la nube, para que el usuario Administrador pueda conectarse a través de un navegador y conexión a internet.

4.2.1.1.1.4.4 Seguridad del Sistema

El sistema de gestión de Conjunto Habitacional tendrá acceso restringido a personas no autorizadas. El control se realizará mediante un usuario con su respectiva contraseña que posea el rol de administrador del conjunto.

Adicionalmente, el sistema contará con un certificado de seguridad (HTTPS) para ofrecer trasferencia de datos encriptada y segura.

4.2.1.2 Análisis de Requerimientos

4.2.1.2.1 Documento de Especificación de Requerimientos del Software (SRS – ISO/IEEE 29148)

4.2.1.2.1.1 Propósito del Software

El propósito de la implementación del sistema de gestión de Conjunto Habitacional es automatizar el proceso de gestión que, en muchos casos, es manejado de forma manual. De esta manera se mejorará la eficiencia de la gestión realizada por el administrador del conjunto.

4.2.1.2.1.2 Alcance

El sistema de Gestión de Conjunto Habitacional constará de la gestión de conjuntos, de unidades, de propietarios, de arrendatarios, de áreas comunales, y de cuotas. Adicionalmente se podrá visualizar información en formato de reporte respecto a la gestión realizada por el Administrador. Estos reportes incluirán la visualización de las cuotas pagadas o adeudadas por las unidades habitacionales, visualización de los diferentes propietarios y arrendatarios asignados a unidad habitacional, y la visualización del estado de las diferentes áreas comunales.

4.2.1.2.1.3 Funciones del Sistema

El sistema de gestión de Conjunto Habitacional contará con las siguientes funciones que serán realizadas por el administrador del conjunto.

El usuario Administrador podrá acceder al sistema para gestionar todos los procesos. De igual manera el usuario podrá finalizar la sesión para salir del sistema.

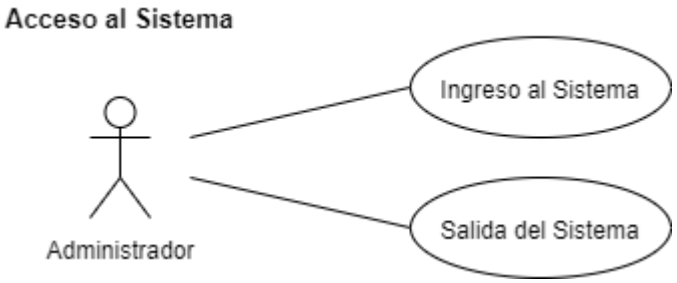


Figura 14. Diagrama de Caso de Uso: Acceso al Sistema

La gestión del conjunto permitirá al Administrador el ingreso, modificación y visualización de toda la información respecto al conjunto.

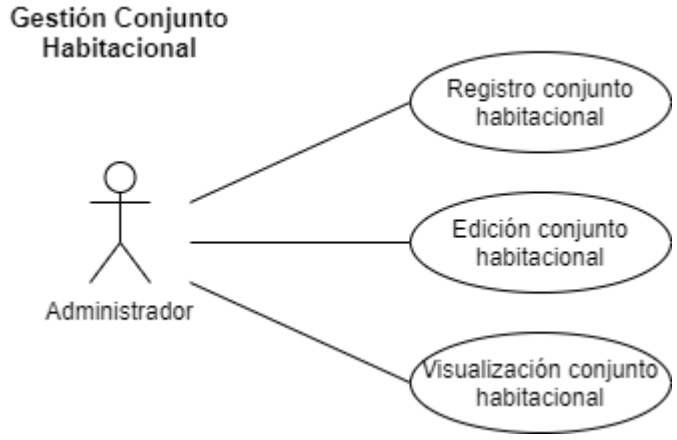


Figura 15. Diagrama de Caso de Uso: Gestión Conjunto Habitacional

La gestión de unidades habitacionales permitirá al Administrador la creación, edición, y visualización de unidades habitacionales, junto con su asignación a los respectivos propietarios y arrendatarios.

Figura 16. Diagrama de Caso de Uso: Gestión Unidades Habitacionales

La gestión de propietarios y arrendatarios permitirá al Administrador el ingreso, edición, visualización, y eliminación de la información de estas personas.

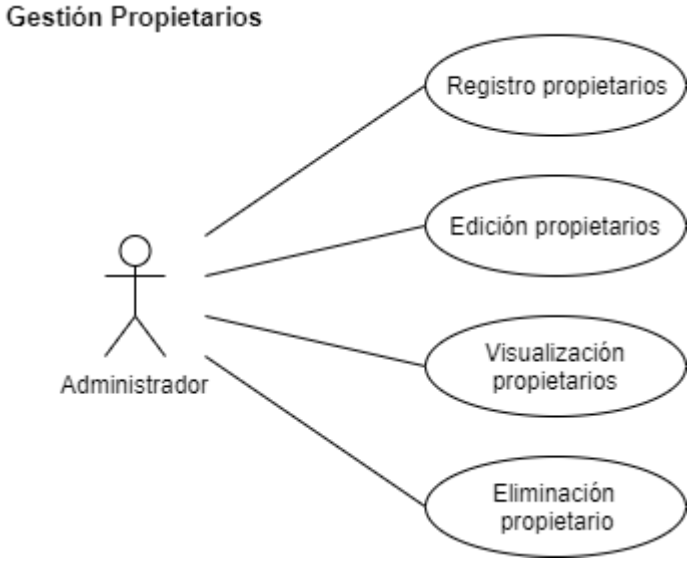


Figura 17. Diagrama de Caso de Uso: Gestión Propietarios

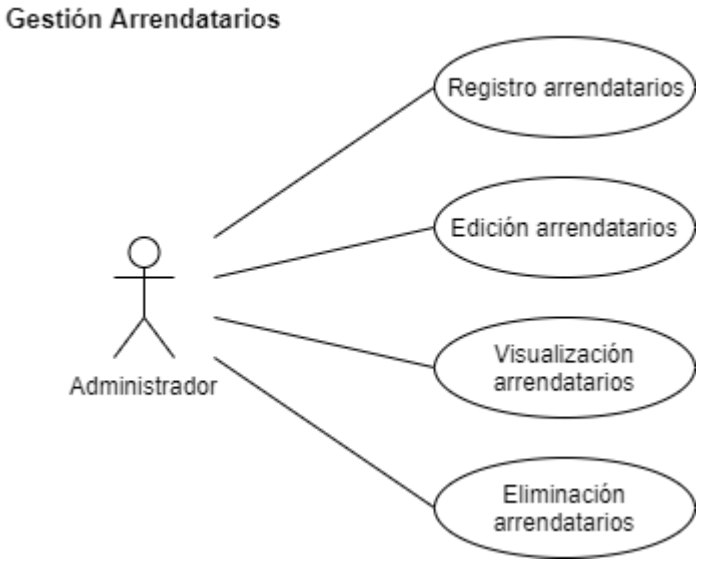


Figura 18. Diagrama de Caso de Uso: Gestión Arrendatarios

La gestión de áreas comunales permitirá al Administrador la creación, modificación, y visualización de la información de áreas comunales.

Figura 19. Diagrama de Caso de Uso: Gestión Áreas Comunales

La gestión de cuotas permitirá al Administrador crear, modificar, y visualizar cuotas de tipo ordinaria y extraordinaria, junto con la asignación a las unidades habitacionales correspondientes.

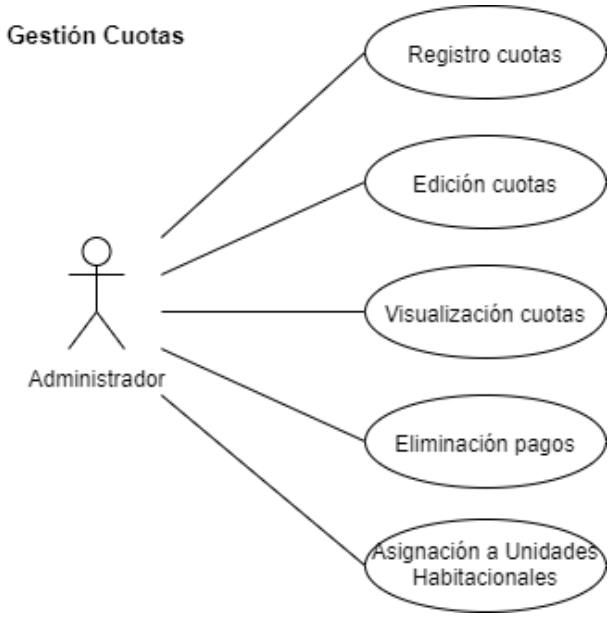


Figura 20. Diagrama de Caso de Uso: Gestión Cuotas

La visualización de reportes permitirá al Administrador acceder a diferentes vistas en formato de tabla donde se mostrará información de las cuotas pagadas o adeudadas por las unidades habitacionales, los diferentes propietarios y arrendatarios asignados a unidad habitacional, y el estado de las diferentes áreas comunales.

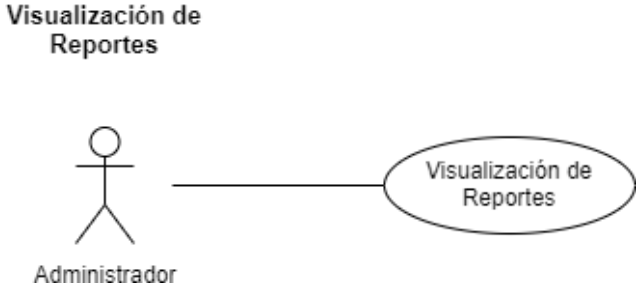


Figura 21. Diagrama de Caso de Uso: Visualización Reportes

4.2.1.2.1.4 Limitaciones

El sistema de gestión de Conjunto Habitacional contará con algunas restricciones. Primero, la única persona con acceso a la información será el administrador, y este será quién decida la manera apropiada de socializarla. Toda la información será almacenada en una base de datos, y los registros de cada tabla no podrán ser eliminados completamente. Para las funciones de eliminación se utilizará únicamente borrado lógico, es decir, que permanecerán en la base de datos con estado inactivo, y no podrán ser modificadas, solamente visualizadas bajo reportes específicos. Las acciones realizadas por el administrador son de entera responsabilidad de este y deberán ser reguladas por la normativa interna del conjunto.

4.2.1.2.1.5 Requerimientos Específicos

4.2.1.2.1.5.1 Funciones

Para el registro de información en el sistema se tomarán en cuenta las validaciones correspondientes para los campos específicos de texto, números, fechas, y demás formatos a utilizar.

La secuencia de operación del sistema será: primero la creación del conjunto habitacional, segundo la creación de áreas comunales o unidades habitacionales. A continuación, la creación de arrendatarios y/o propietarios para ser asignados a una unidad habitacional. Finalmente, la creación y asignación de cuotas para cada unidad habitacional. Para la visualización de los diferentes reportes, la única condición será tener suficiente información en la base de datos correspondiente a lo que se desea observar.

La secuencia de operación para cada elemento del sistema será, primero la creación para habilitar la visualización, edición, y eliminación. La funcionalidad de edición y eliminación, en los casos que corresponda, estarán únicamente habilitadas dependiendo del estado de los registros de acuerdo con los requerimientos del sistema.

En el caso de fallos en el servidor, se presentarán alertas específicas en el sistema indicando el error ocurrido, de manera que el usuario administrador pueda contactar al equipo de soporte.

4.2.1.2.1.5.2 Requerimientos de Desempeño

El sistema de gestión de Conjunto Habitacional permitirá la existencia de varias sesiones simultáneas para el usuario administrador, permitiéndole gestionar la información del lugar que sea más conveniente para este.

El procesamiento de la información en cuanto a la funcionalidad de creación, edición, visualización, y eliminación de registros únicos será inmediata. El tiempo procesamiento de múltiples registros incrementará de acuerdo con la cantidad de información almacenada.

El usuario administrador del sistema deberá esperar a que se completen las transacciones para poder continuar con el uso del sistema.

4.2.1.2.1.5.3 Restricciones de Diseño

El sistema de gestión de Conjunto Habitacional deberá ser diseñado para ser compatible con los navegadores web de computadores de escritorio, o portátiles, más comunes del mercado.

4.2.1.3 Modelado Diagramas UML

4.2.1.3.1 Diagramas de Casos de Uso

La diagramación de los casos de uso del sistema de gestión de Conjunto Habitacional se encuentra dentro del documento de Especificación de Requerimientos del Software (SRS) bajo la sección de Funciones del Sistema.

4.2.1.3.2 Modelo de Contenido (UWE)

La **Figura 22** presenta todas las entidades y sus asociaciones de acuerdo a los requerimientos definidos previamente. El Conjunto Habitacional tiene varias áreas comunales y varias unidades habitacionales asociadas. Las Áreas Comunales están asociadas solamente a un conjunto. Las Unidades Habitacionales están asociadas solamente a un conjunto, pero pueden asociarse con varios propietarios y varios arrendatarios a lo largo del tiempo. Los Propietarios y Arrendatarios pueden estar asociados a varias unidades habitacionales. Y por último, las Cuotas están relacionadas específicamente con las unidades habitacionales independientemente del propietario o arrendatario vigente.

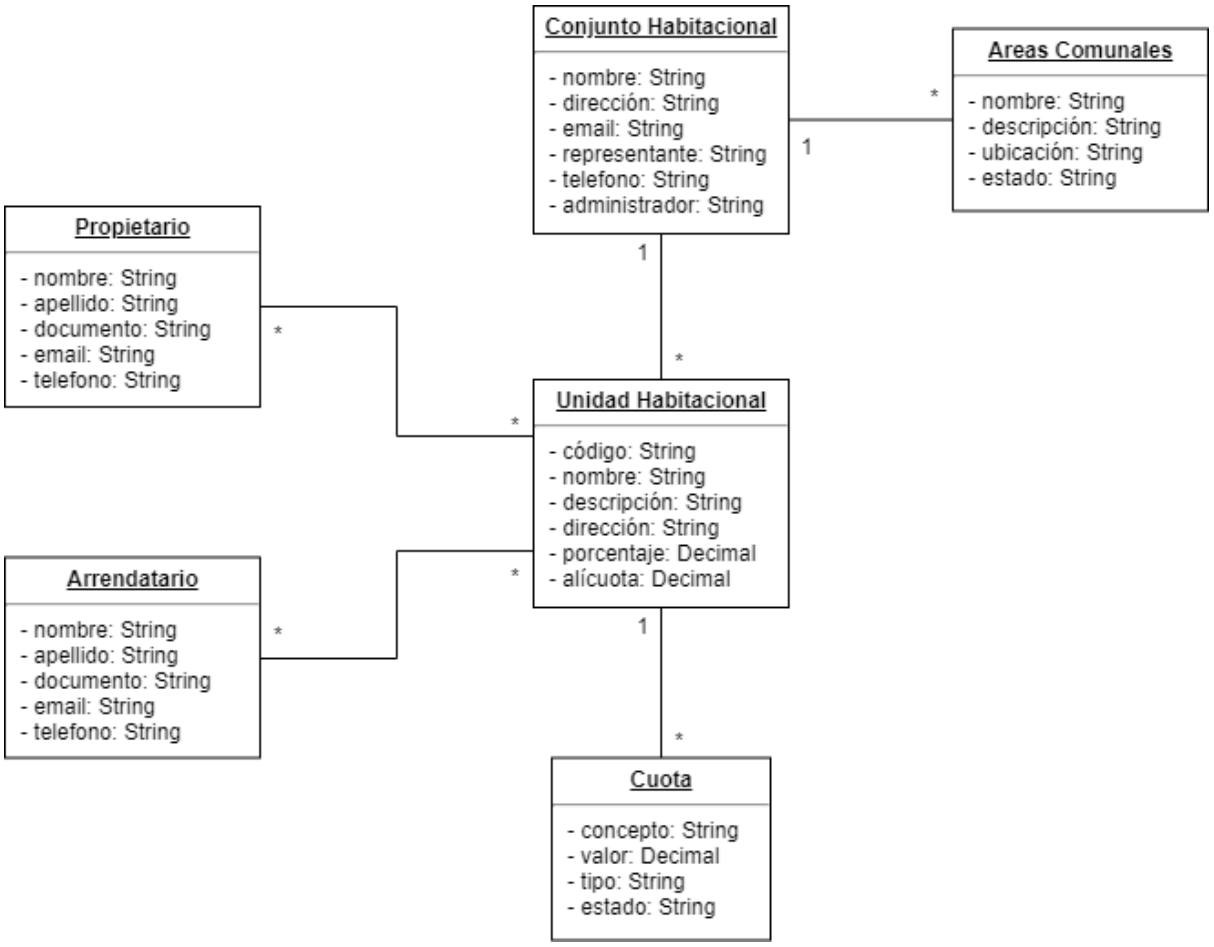


Figura 22. Modelo de Contenido (UWE)

4.2.2 Fase Diseño

4.2.2.1 Diseño de Arquitectura

4.2.2.1.1 Arquitectura Física

El sistema de gestión de Conjunto Habitacional estará almacenado en una instancia de Amazon EC2. El acceso al sistema se lo realizará mediante el uso de un navegador web con conexión a internet. Amazon Route 53 redireccionará el dominio hacia la instancia EC2. Todas las peticiones que lleguen a la instancia serán procesadas mediante la arquitectura del software de la **Figura 24**. Finalmente, las peticiones de consultas o manipulación de datos serán enviadas desde el sistema ubicado en la instancia EC2 hacia Amazon RDS donde se encuentra el esquema de base de datos relacional.

Figura 23. Arquitectura Física

4.2.2.1.2 Arquitectura del Software

La **Figura 24** refleja la arquitectura del sistema de software y su estructura de archivos. La arquitectura base es la misma utilizada por el framework Laravel. El proceso inicia con una petición del usuario, la cuál habiendo llegado al servidor es dirigida por una ruta hacia un controlador específico. El controlador es el encargado de recibir la petición y retornar una respuesta. Para esto, el controlar utiliza la capa de procesos según la acción requerida del usuario para consultas o transacciones con la base de datos. El Proceso se comunica con el repositorio para hacer la consulta o transacción. Y finalmente el repositorio, mediante el uso del modelo de persistencia obtiene la información requerida y lo retorna al proceso. El proceso retorna al controlador, y finalmente el controlador envía la respuesta al usuario.

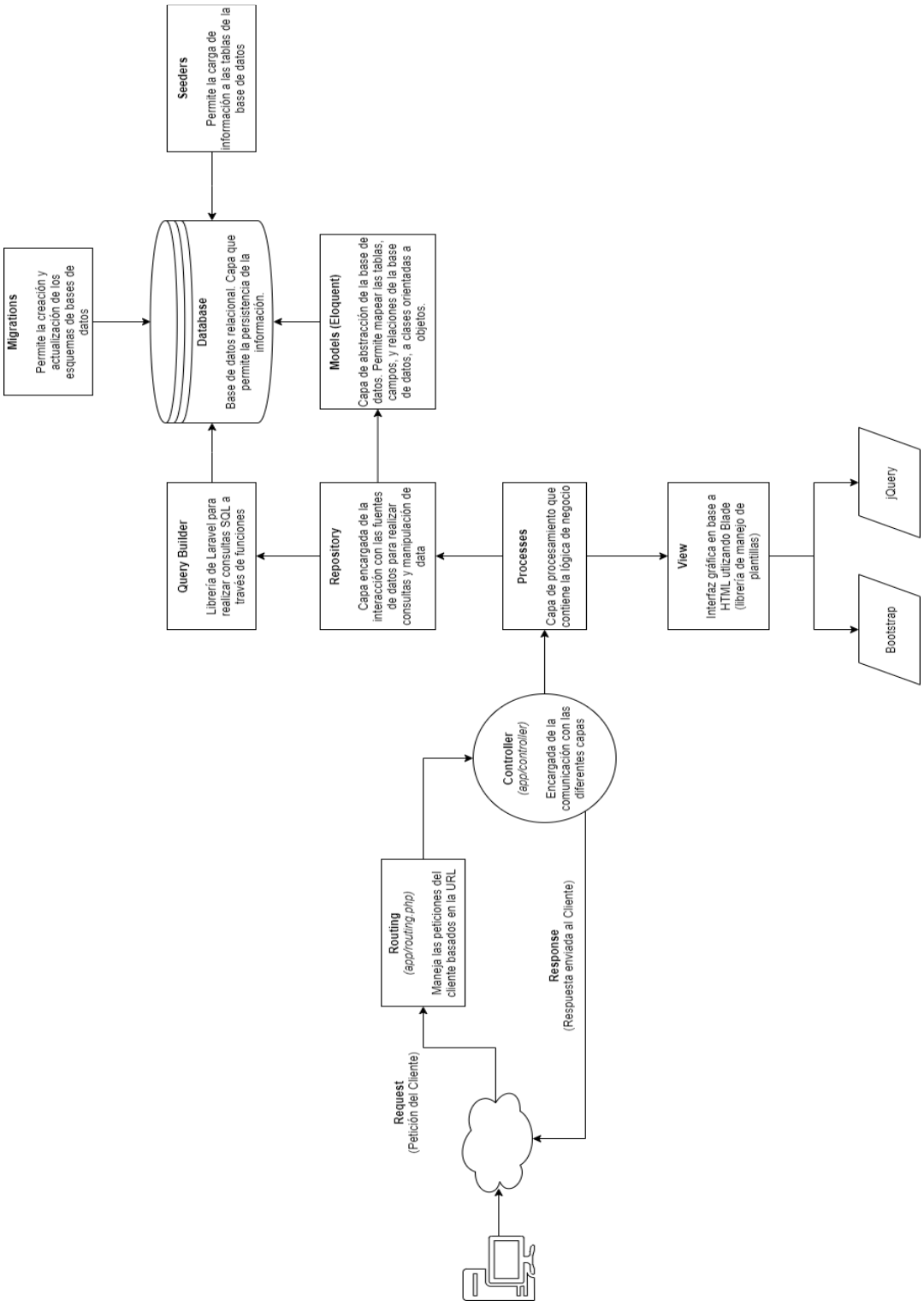


Figura 24. Arquitectura del Software

Para el sistema de gestión de Conjunto Habitacional se han agregado capas adicionales para mejorar la organización y comprensión del código, convirtiendo a la arquitectura en n capas. La estructura de archivos manejados se detalla a continuación.

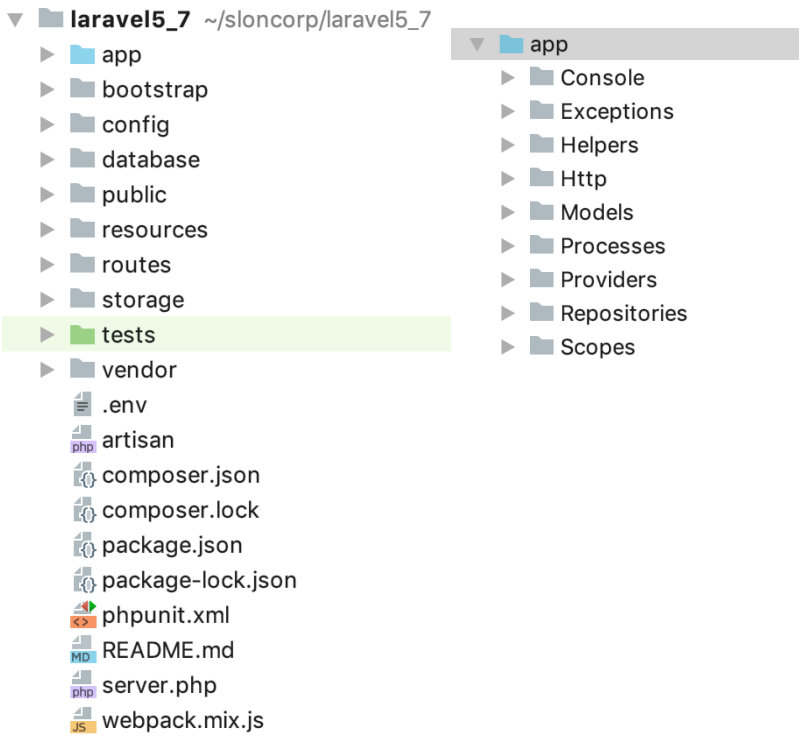


Figura 25. Estructura principal de archivos

4.2.2.2 Modelado Diagramas UML

4.2.2.2.1 Modelo Navegación (UWE)

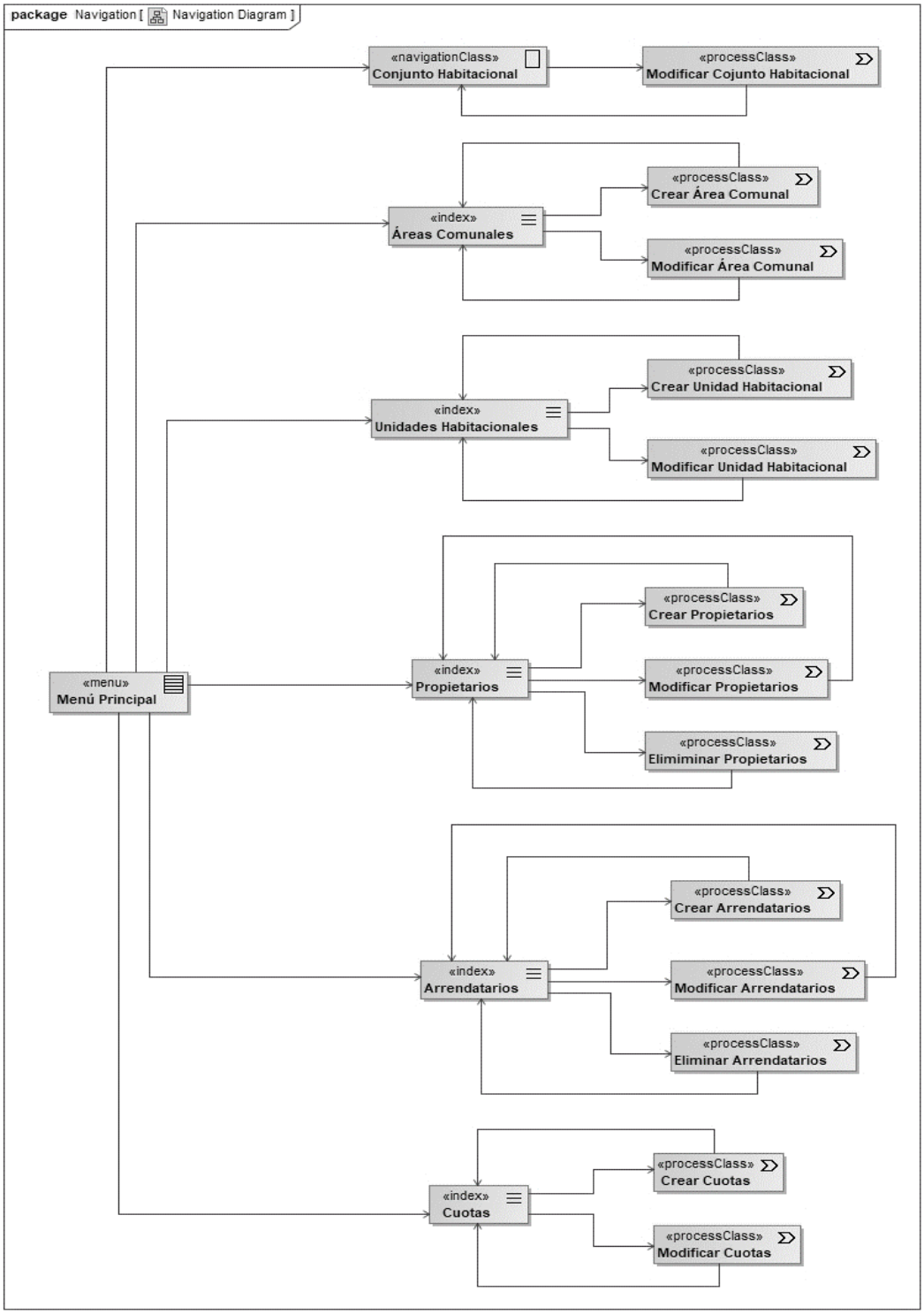


Figura 26. Modelo Navegación (UWE)

4.2.2.2.2 Modelo Estructura Procesos (UWE)

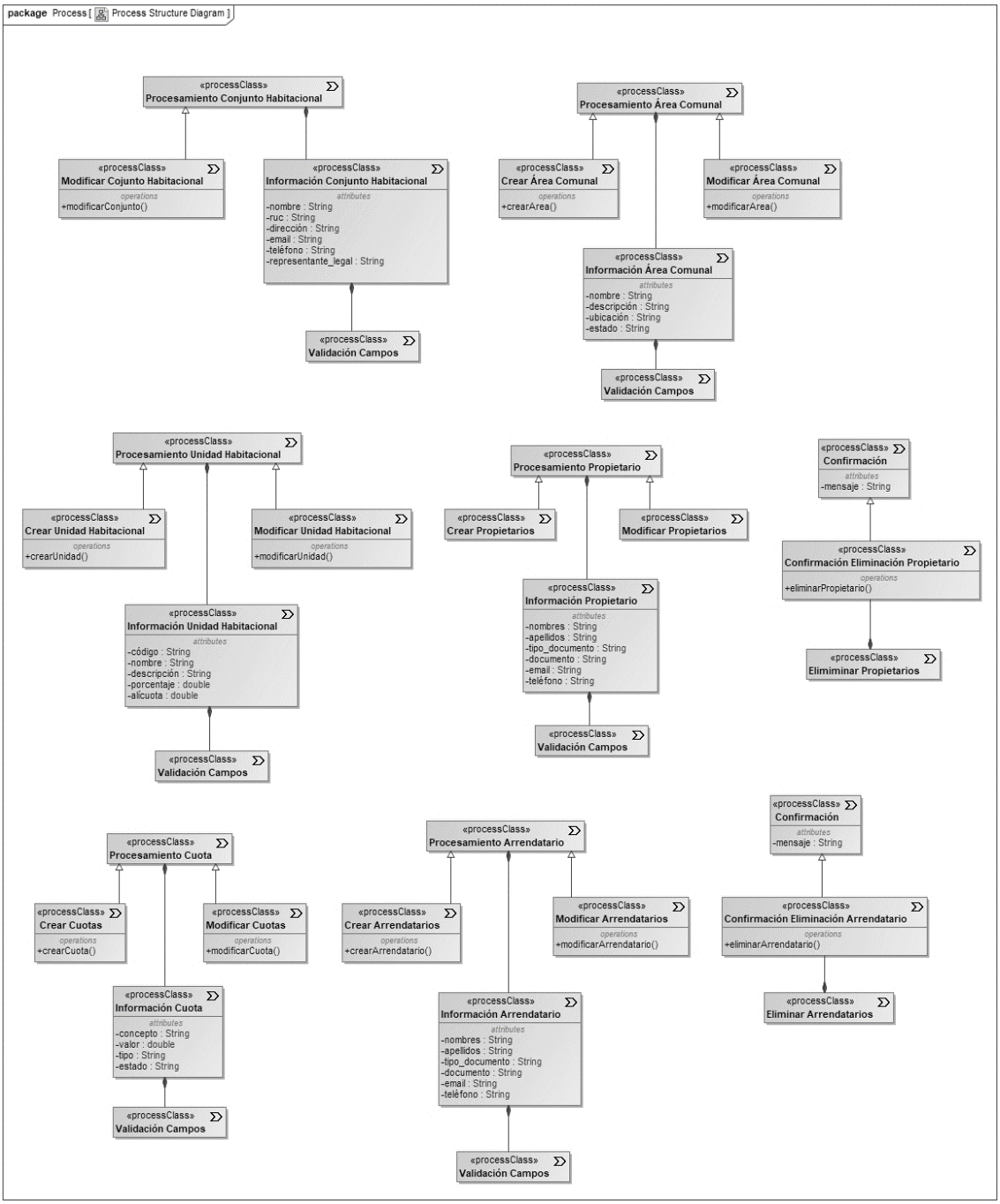


Figura 27. Estructura Procesos (UWE)

4.2.2.2.3 Flujo de Procesos (UWE)

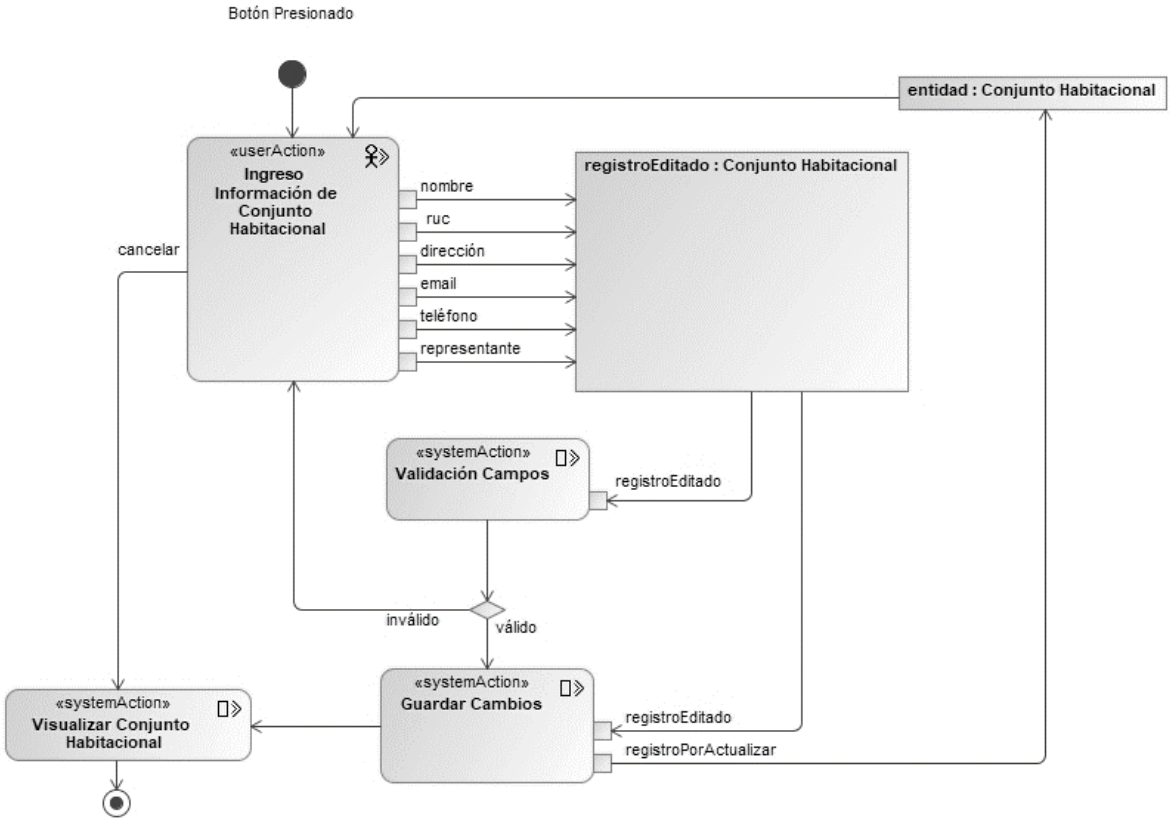


Figura 28. Flujo Modificación Conjunto Habitacional

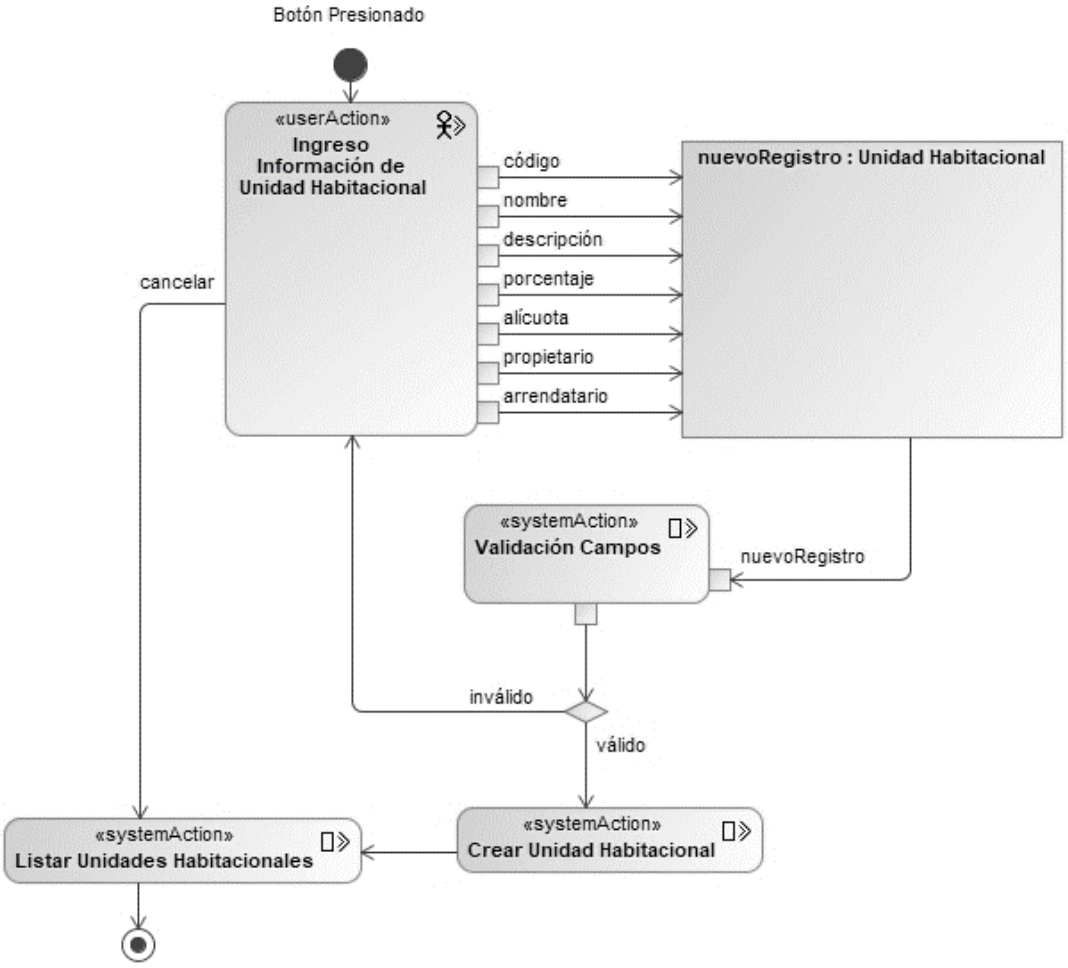


Figura 29. Flujo Creación Unidad Habitacional

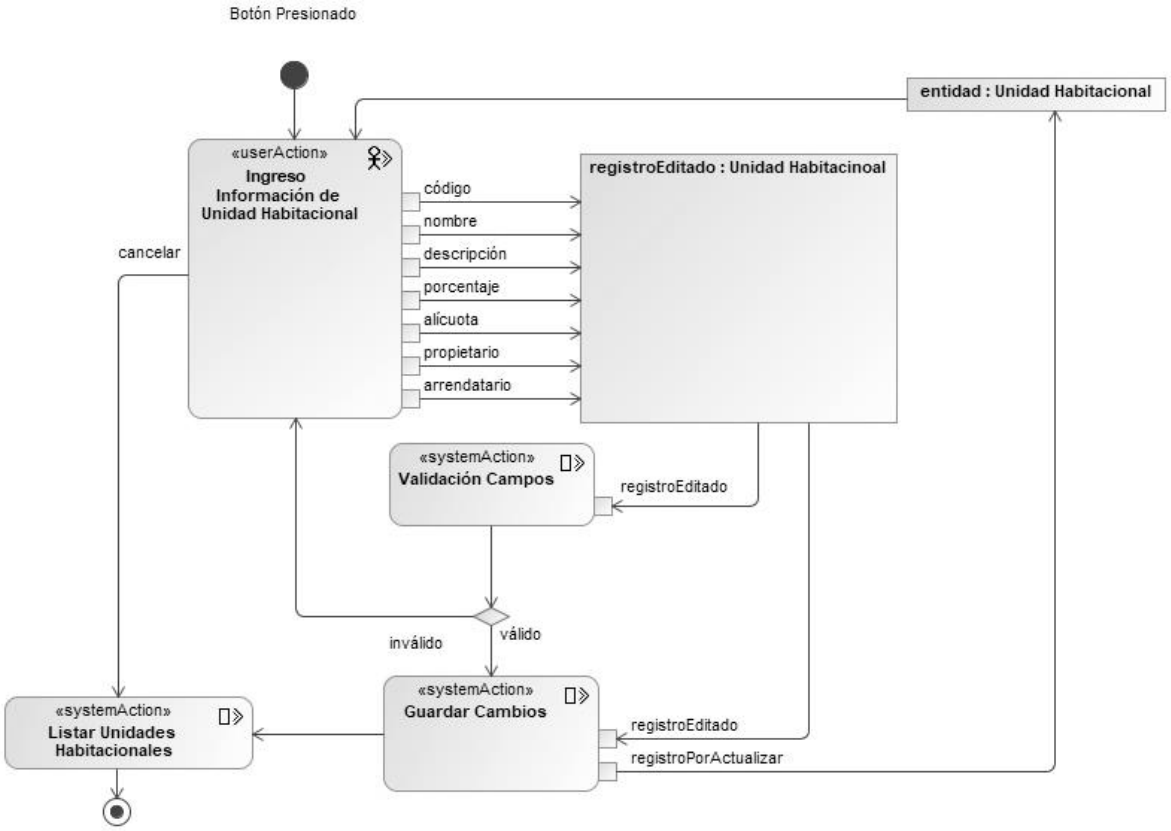


Figura 30. Flujo Modificación Unidad Habitacional

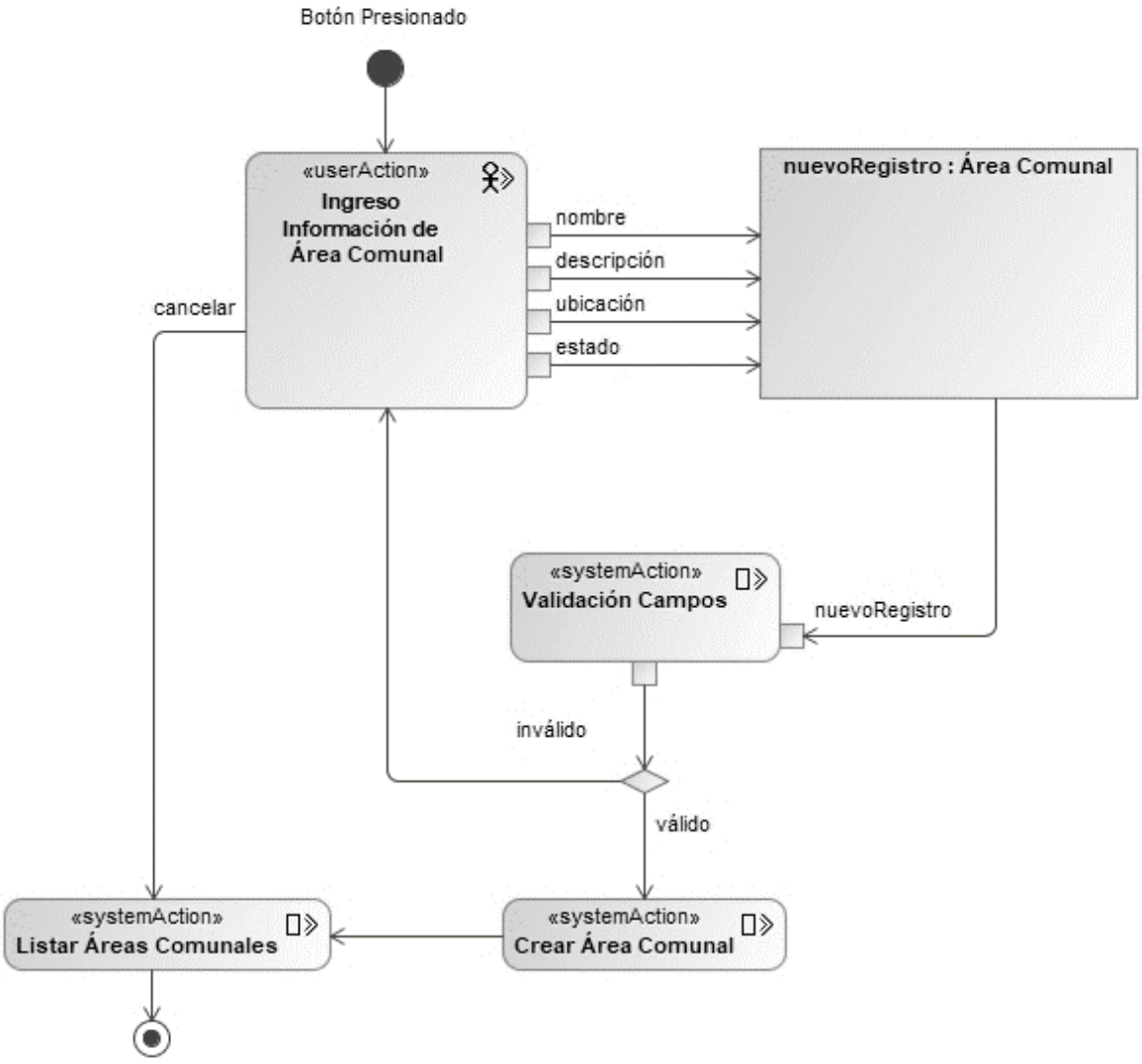


Figura 31. Flujo Creación Área Comunal

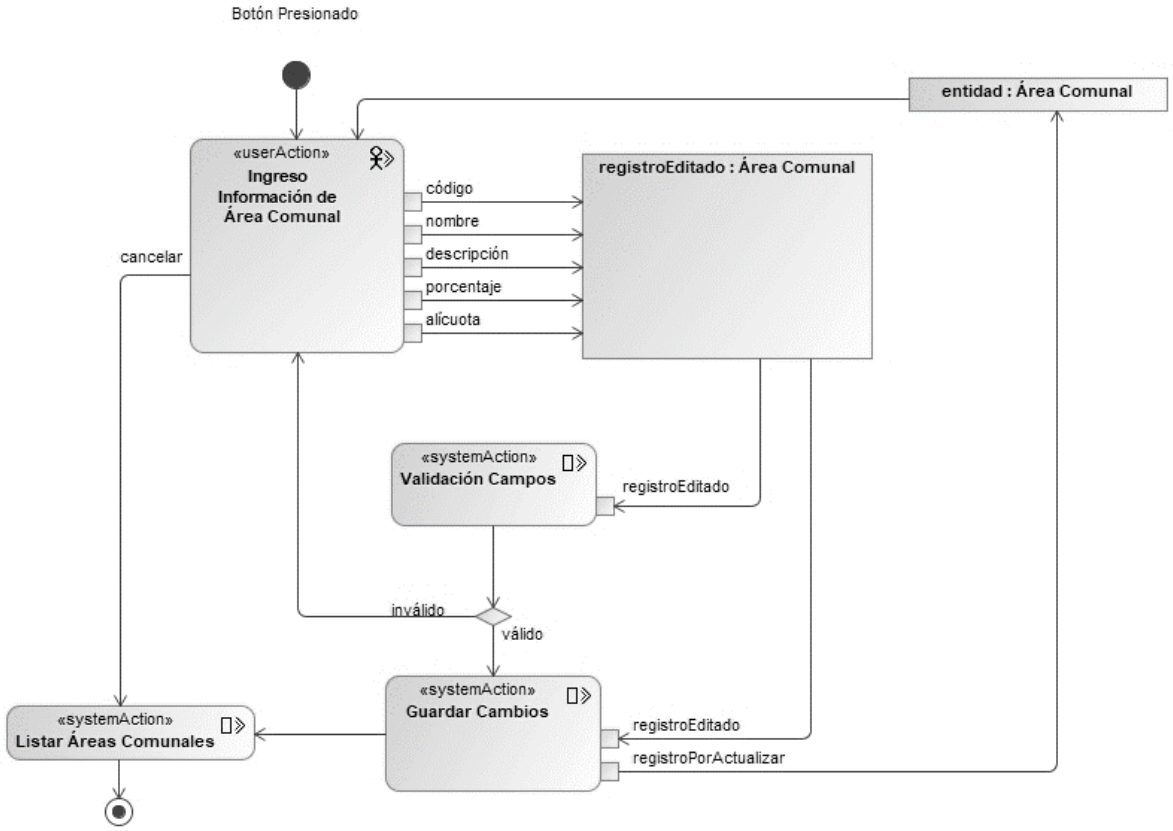


Figura 32. Flujo Modificación Área Comunal

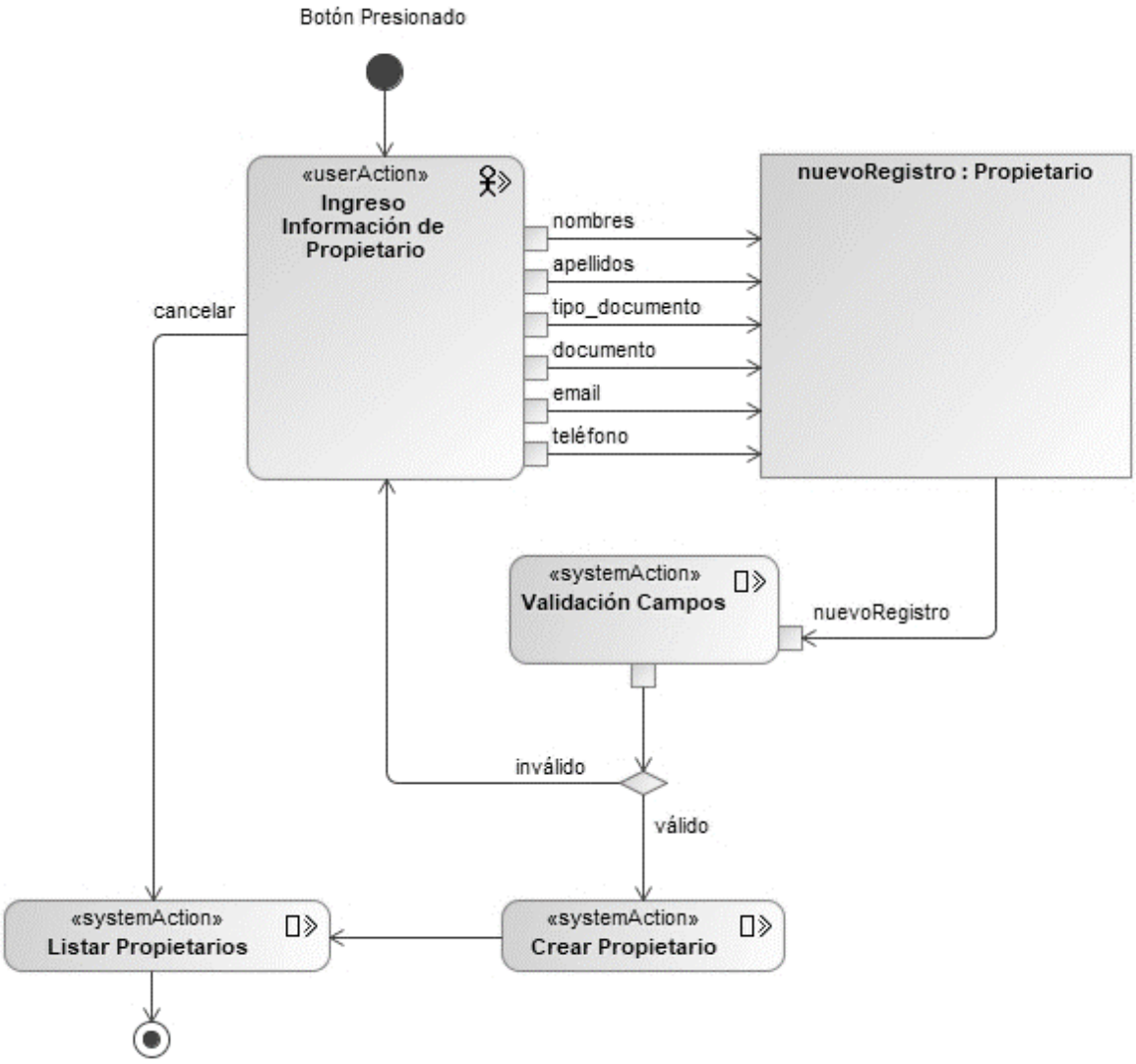


Figura 33. Flujo Creación Propietario

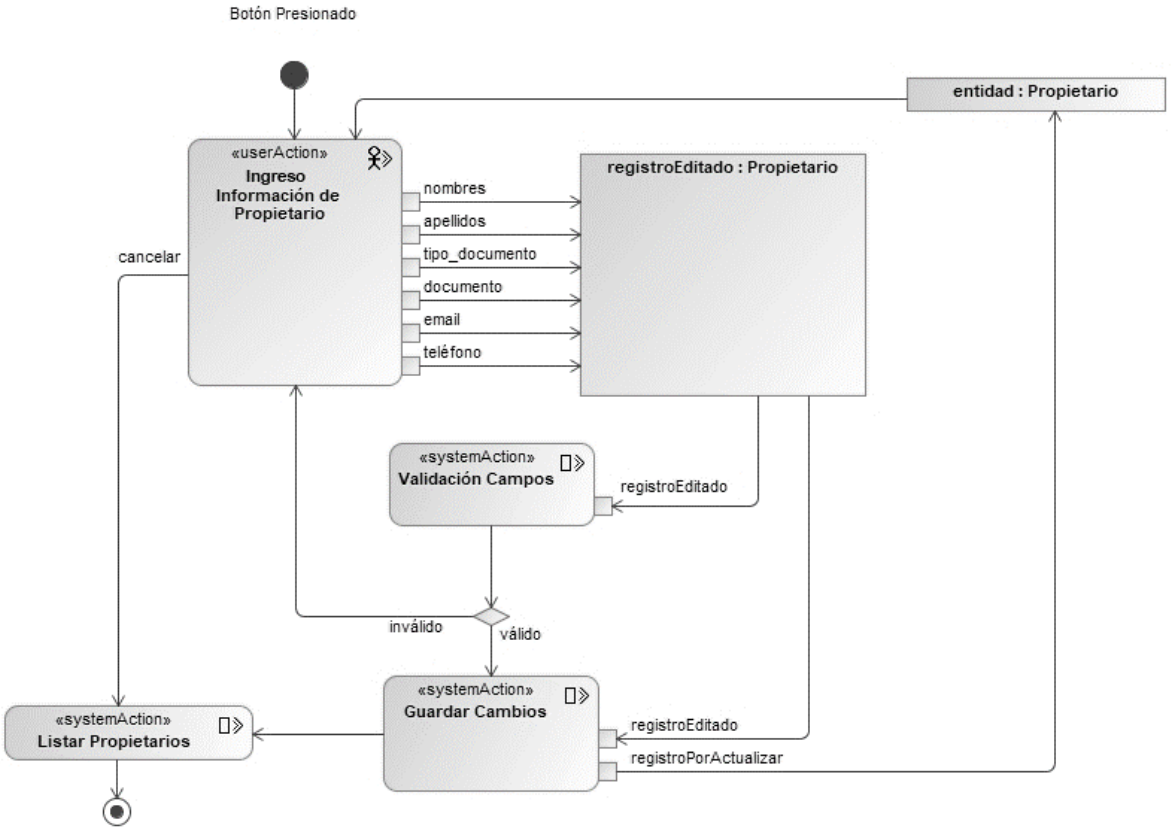


Figura 34. Flujo Modificación Propietario

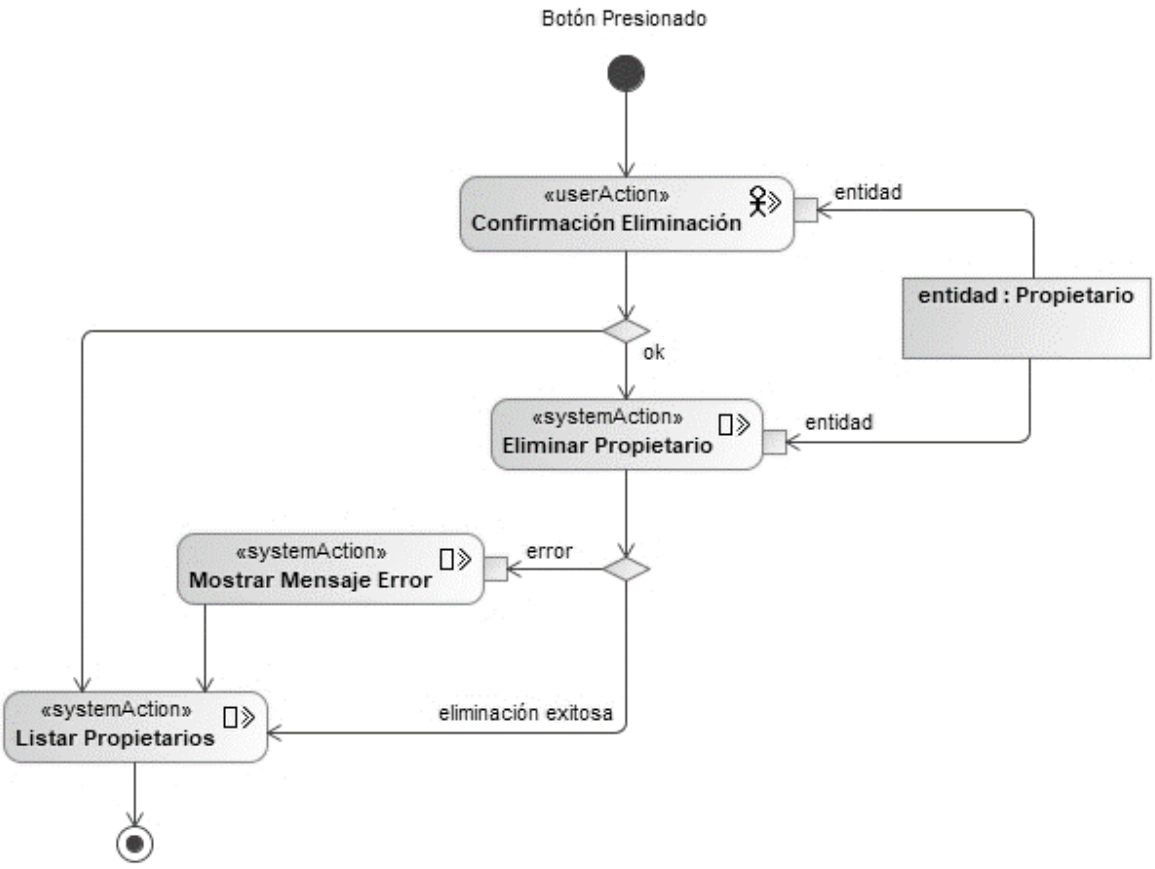


Figura 35. Flujo Eliminación Propietario

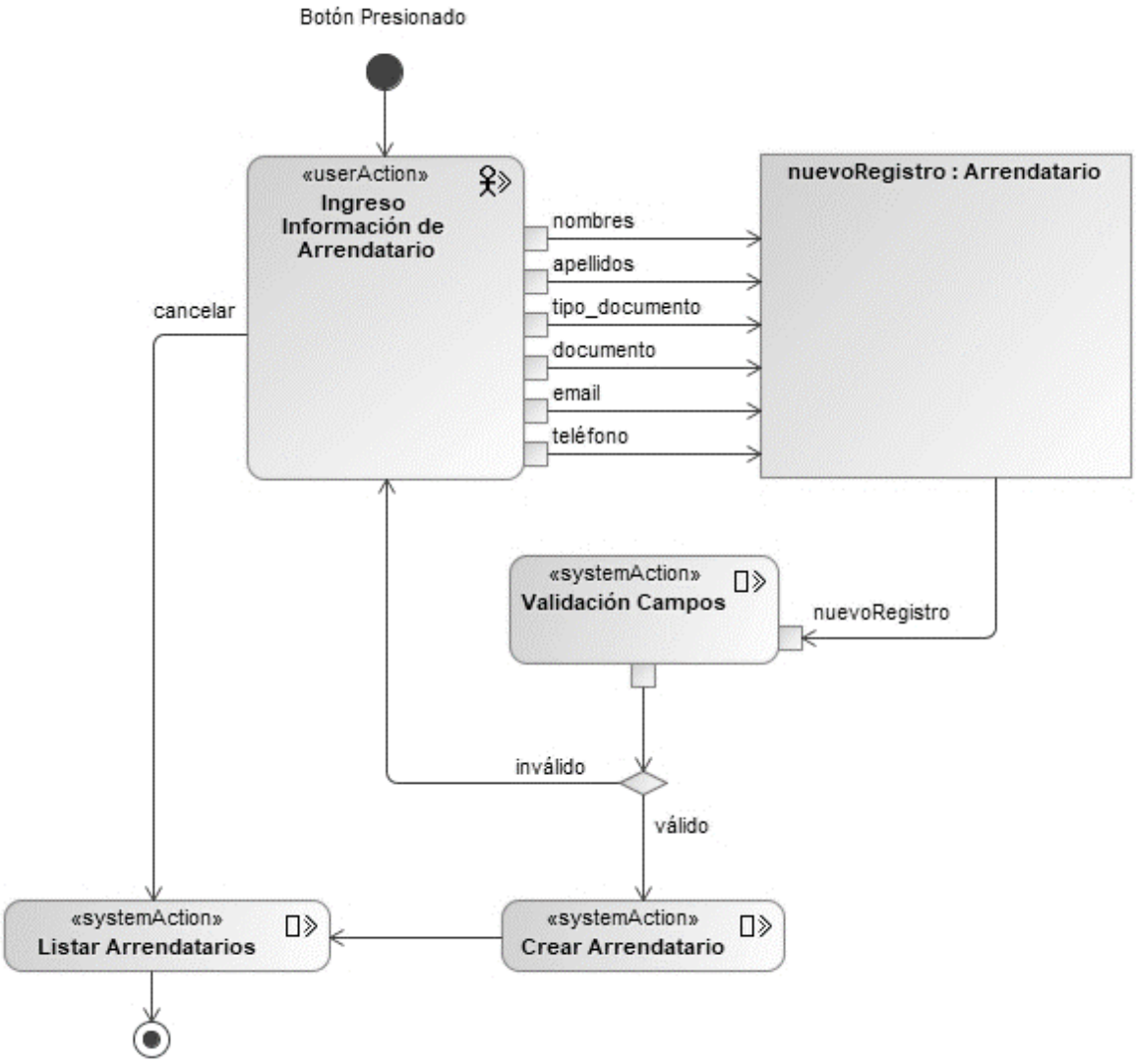


Figura 36. Flujo Creación Arrendatario

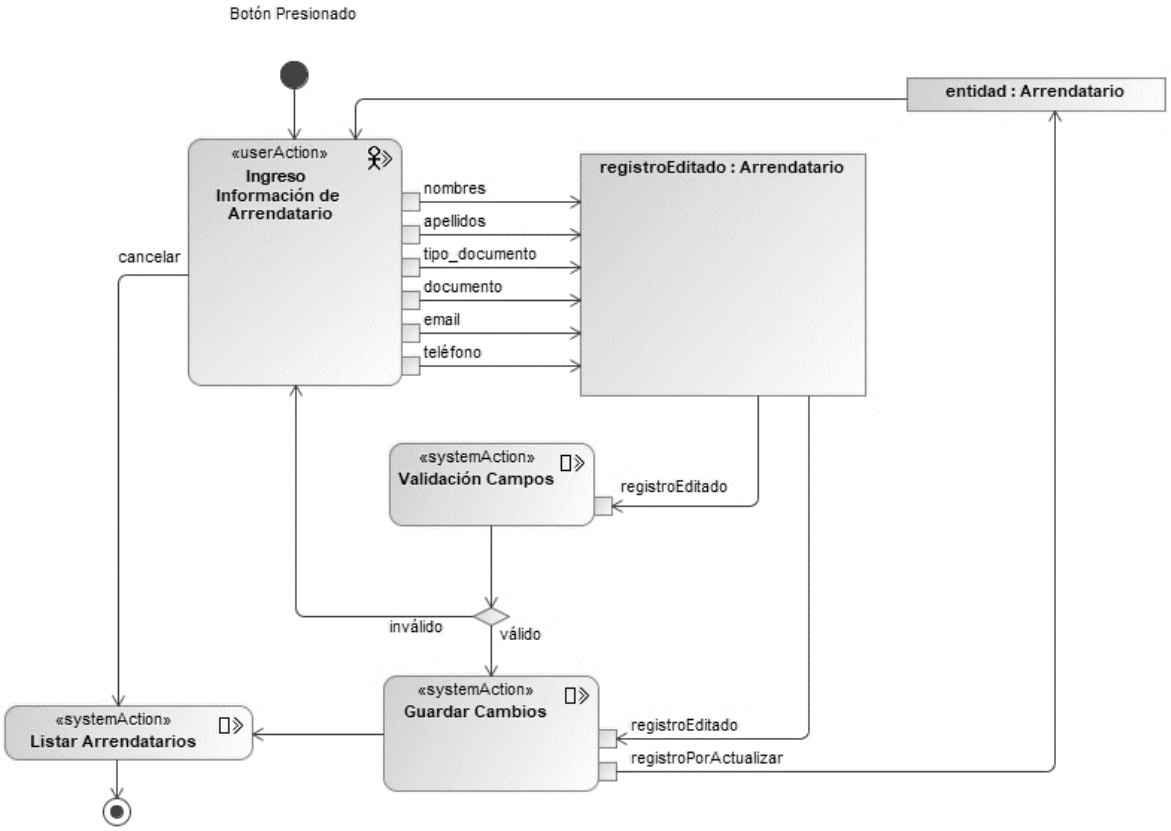


Figura 37. Flujo Modificación Arrendatario

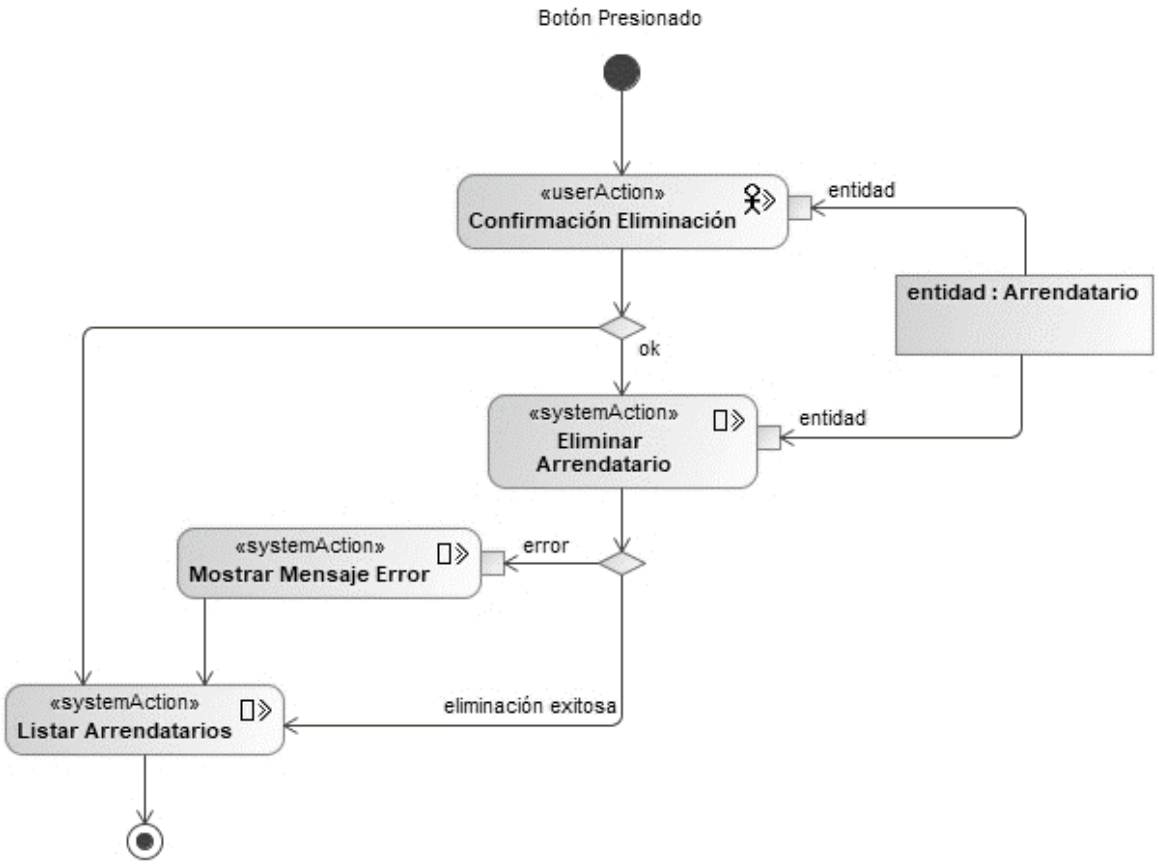


Figura 38. Flujo Eliminación Arrendatario

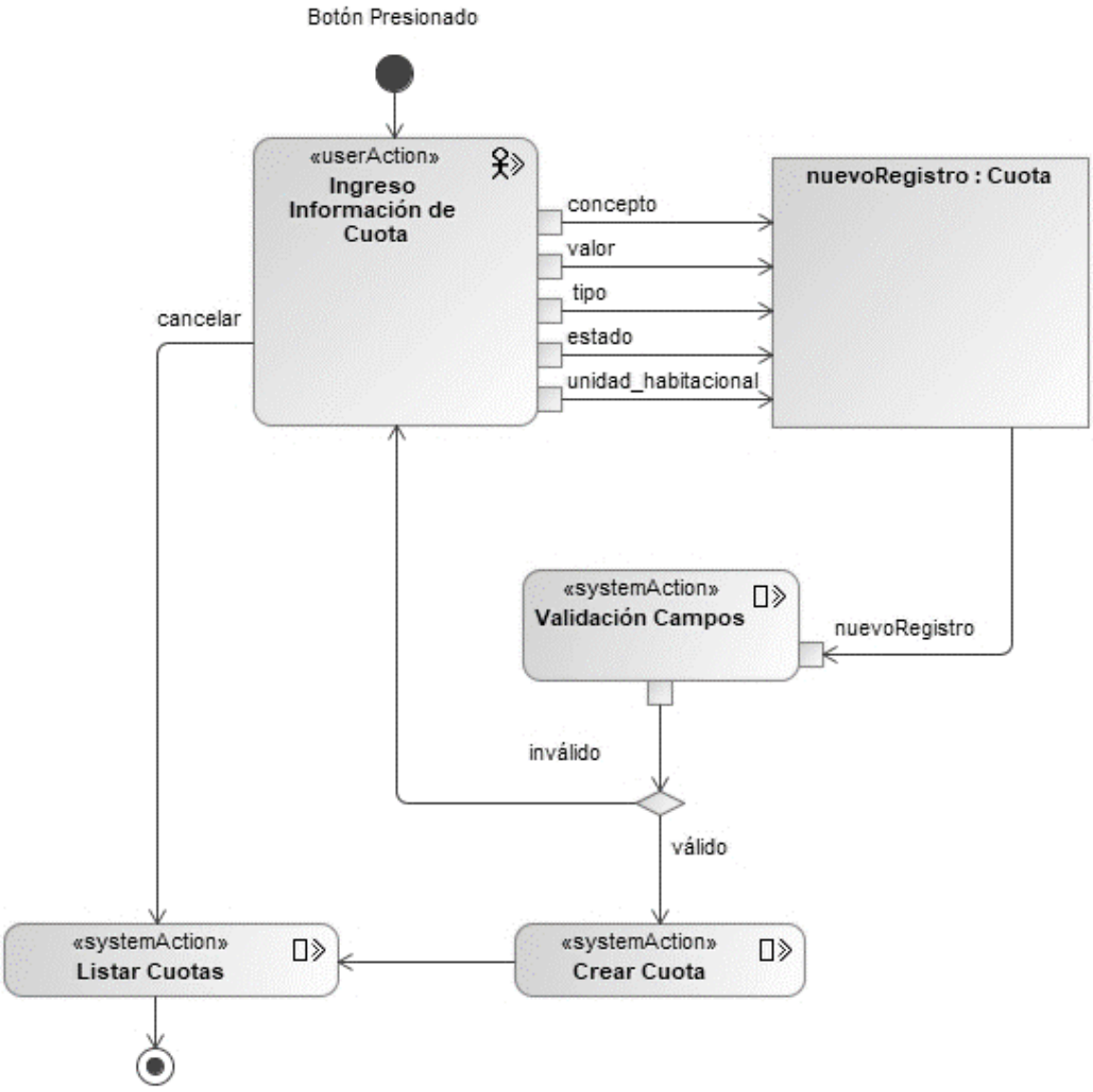


Figura 39. Flujo Creación Cuotas

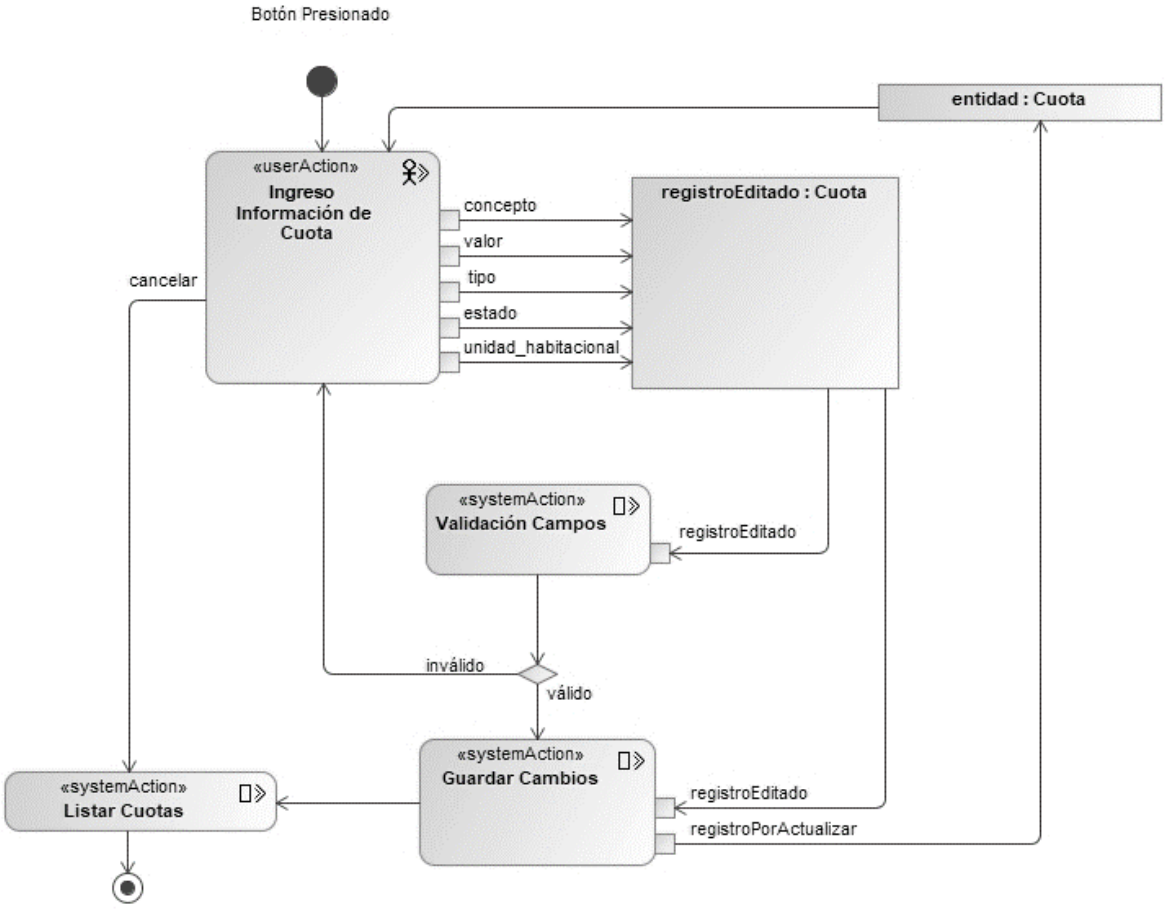


Figura 40. Flujo Modificación Cuotas

4.2.2.3 Diseño Base de Datos

4.2.2.3.1 Diagrama Base de Datos

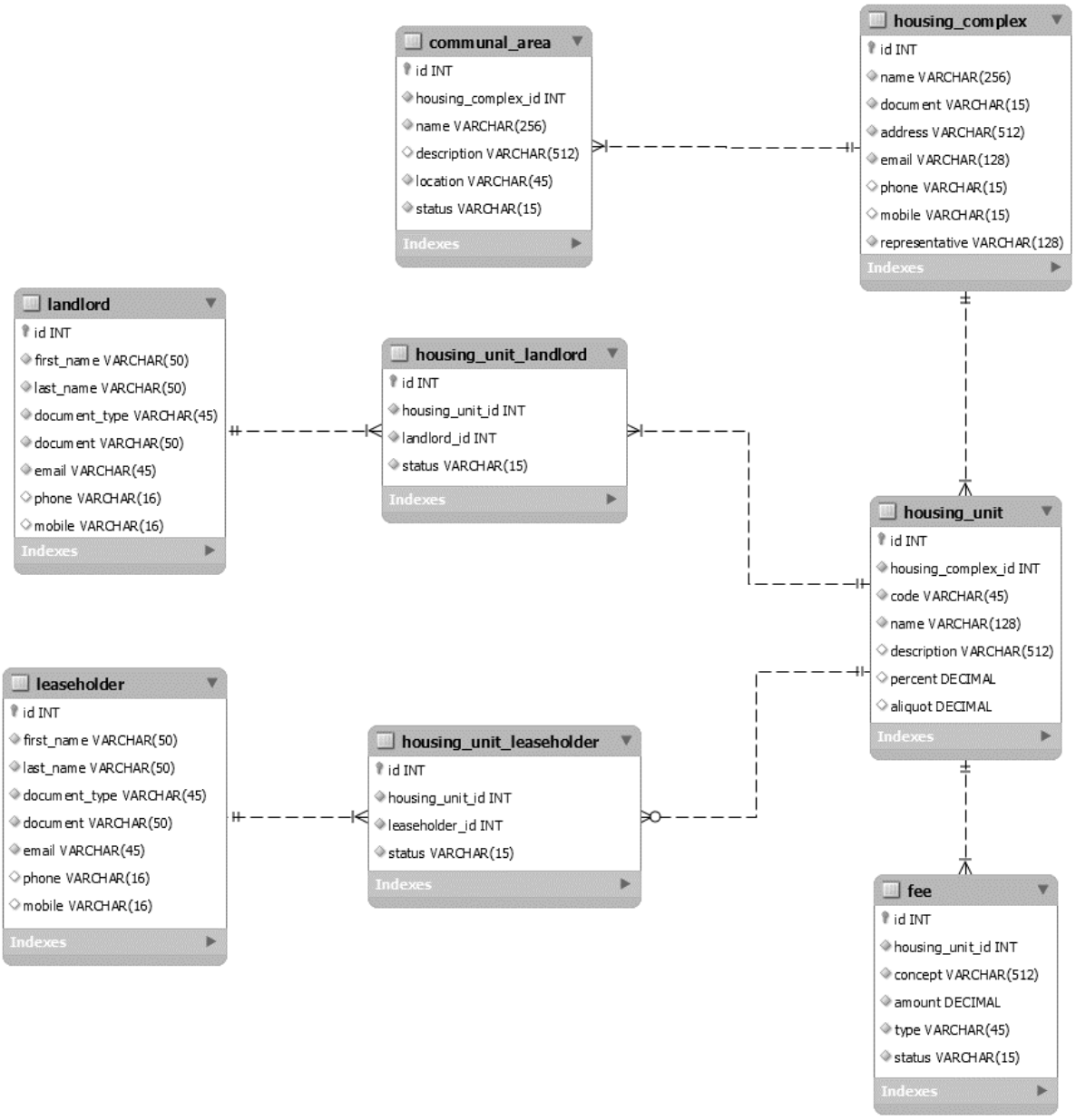


Figura 41. Esquema de Base de Datos

4.2.2.3.2 Script Base de Datos

Referirse al **Anexo 2**.

4.2.3 Fase Implementación

4.2.3.1 Definición Estrategia

Para la implementación del sistema de gestión de Conjunto Habitacional se utilizará la siguiente estrategia. En el ambiente local se trabajará con Homestead, que es una máquina virtual empaquetada con el lenguaje de programación a utilizar (PHP), servidor web configurado, y demás herramientas que se necesitarán en el desarrollo (Colaboradores de Laravel, 2020).

El uso de Homestead es requerido para simular el ambiente real en donde el sistema será desplegado después de la implementación por su sistema operativo Ubuntu 18.04 basado en Unix. La instancia EC2 donde el sistema será desplegado es un servidor con sistema operativo basado en Unix.

Para el back-end, desarrollado en PHP, se utilizará el estándar PSR-4, el cual se encuentra precargado en el IDE de desarrollo PHP Storm. Para el front-end el lenguaje principal será Javascript, y el estándar que se utilizará es ECMAScript 6 (ES-6).

Para el control de versiones del código fuente se utilizará Git, y para la gestión del repositorio, se utilizará Bitbucket.

La gestión de tareas se manejará en la herramienta web Jira. El flujo de una tarea empieza con estado Por Hacer. Una vez iniciada su estado cambia a En Progreso. Al haber sido terminada la tarea, su estado cambiará a Control de Calidad. Si cumple con los requerimientos adecuados, la

tarea podrá ser marcada como Finalizada, de lo contrario volverá a Por Hacer, junto con la descripción de los errores o cambios que se deberán realizar.

4.2.3.2 Codificación

Referirse al CD adjunto.

4.2.3.3 Verificación

La verificación de la funcionalidad del sistema se ejecutó periódicamente durante la finalización de cada Sprint en base a sus tareas definida. A continuación se muestran los reportes de todos los Sprints utilizados para la implementación del sistema. Adicionalmente, cada proceso de verificación consta con un documento de verificación de funcionalidad que se encuentran en la sección de anexos.

Sprint 1: Definición de Requerimientos de Software y de Sistema

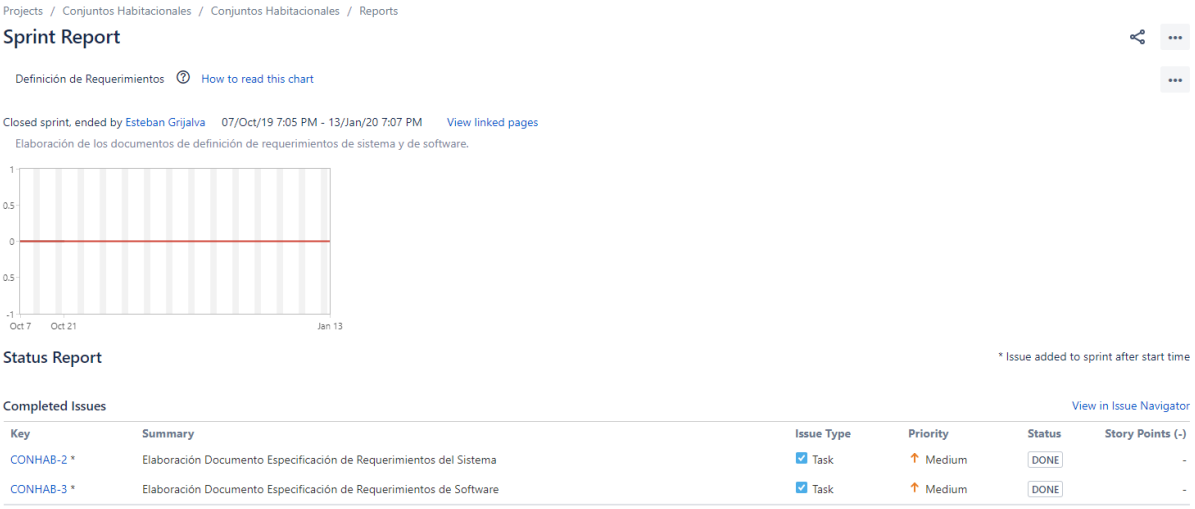


Figura 42. Reporte Sprint 1

Sprint 2: Modelo de Cotenido y Diagramas de Casos de Uso

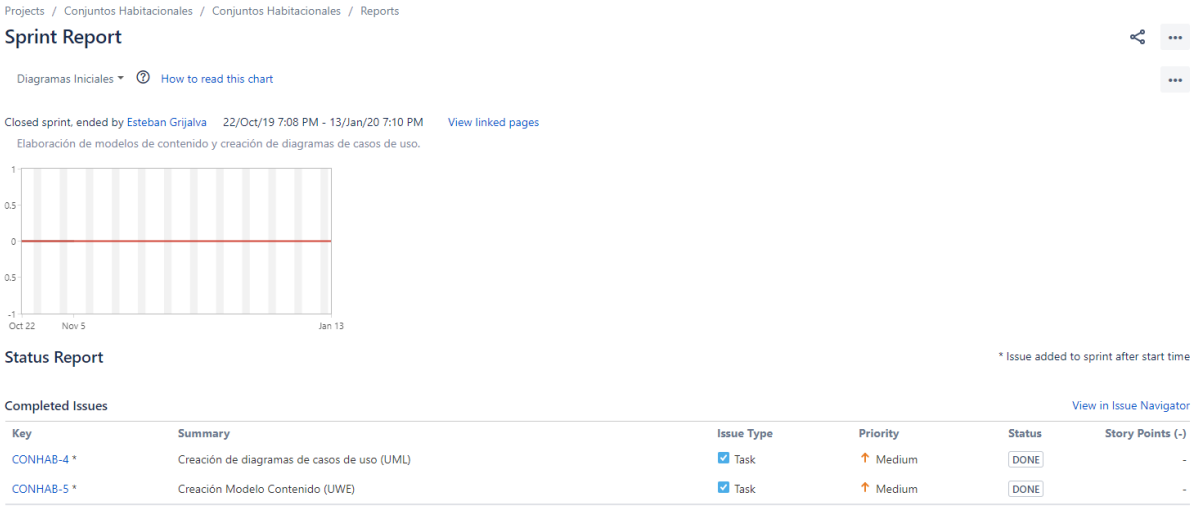


Figura 43. Reporte Sprint 2

Sprint 3: Diagramas de Navegación, Procesos y Arquitectura.

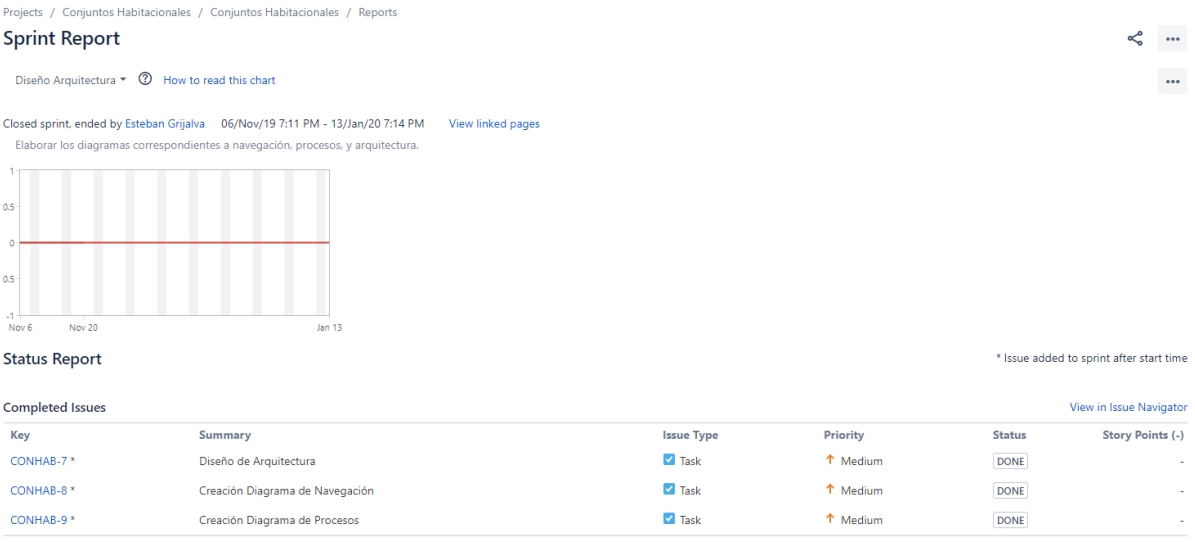


Figura 44. Reporte Sprint 3

Sprint 4: Configuración inicial del Sistema, y creación de Migraciones.

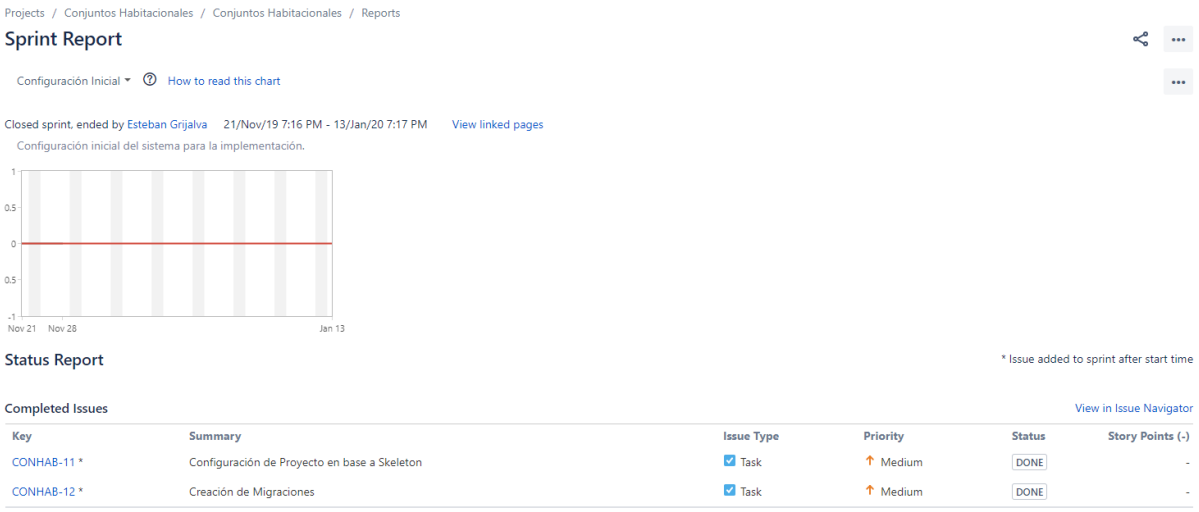


Figura 45. Reporte Sprint 4

Sprint 5: Funcionalidad para Conjunto Habitacional y Áreas Comunales

Verificación**:** referirse al **Anexo 3**.

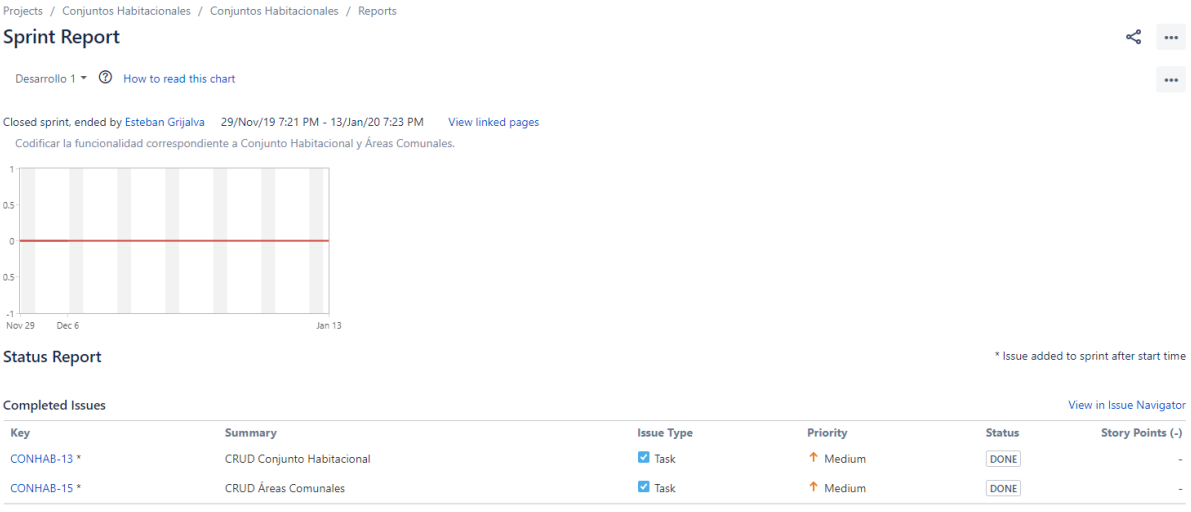
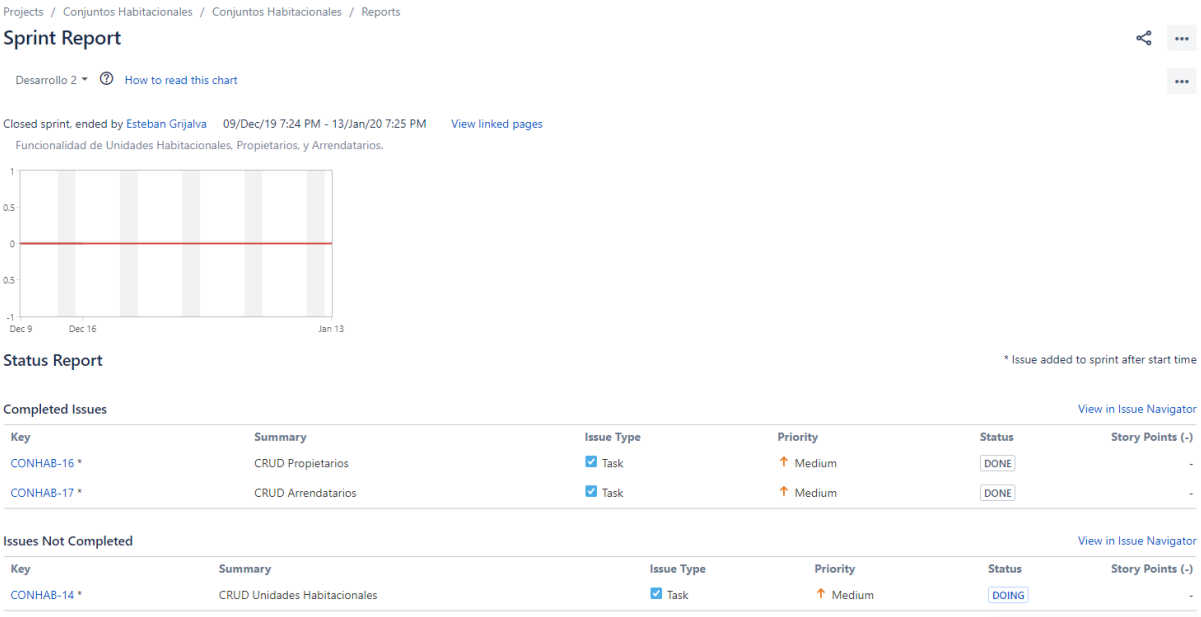


Figura 46. Reporte Sprint 5

Sprint 6: Funcionalidad Arrendatarios, Propietarios, y Unidades Habitacionales.

Verificación**:** referirse al **Anexo 4**.



Projects / Conjuntos Habitacionales / Conjuntos Habitacionales / Reports

Sprint Report

Desarrollo 2 How to read this chart

Closed sprint, ended by Esteban Grijalva 09/Dec/19 7:24 PM - 13/Jan/20 7:25 PM View linked pages

Funcionalidad de Unidades Habitacionales, Propietarios, y Arrendatarios.

1 0.5 0 0.5 -1

Dec 9 Dec 16 Jan 13

Status Report

* Issue added to sprint after start time

Completed Issues View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (-)
CONHAB-16 *	CRUD Propietarios	Task	Medium	DONE	-
CONHAB-17 *	CRUD Arrendatarios	Task	Medium	DONE	-

Issues Not Completed View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (-)
CONHAB-14 *	CRUD Unidades Habitacionales	Task	Medium	DOING	-

Figura 47. Reporte Sprint 6

Sprint 7: Corrección Unidades Habitacionales y funcionalidad de Cuotas.

Verificación: referirse al **Anexo 5**.

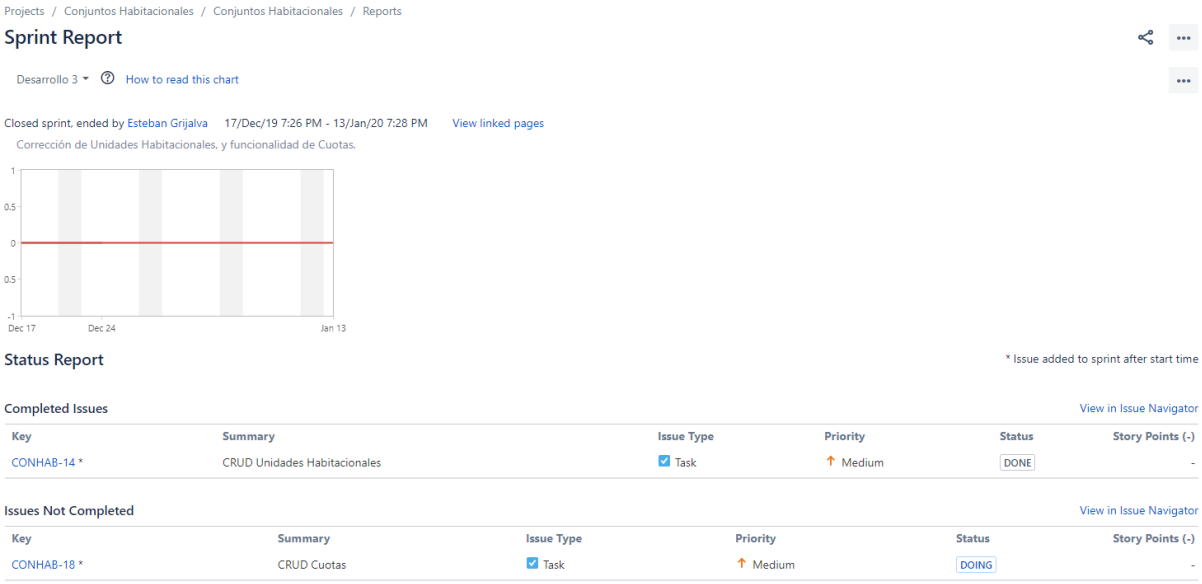


Figura 48. Reporte Sprint 7

Sprint 8: Asignación de Cuotas y Reportes.

Verificación**:** referirse al **Anexo 6**.

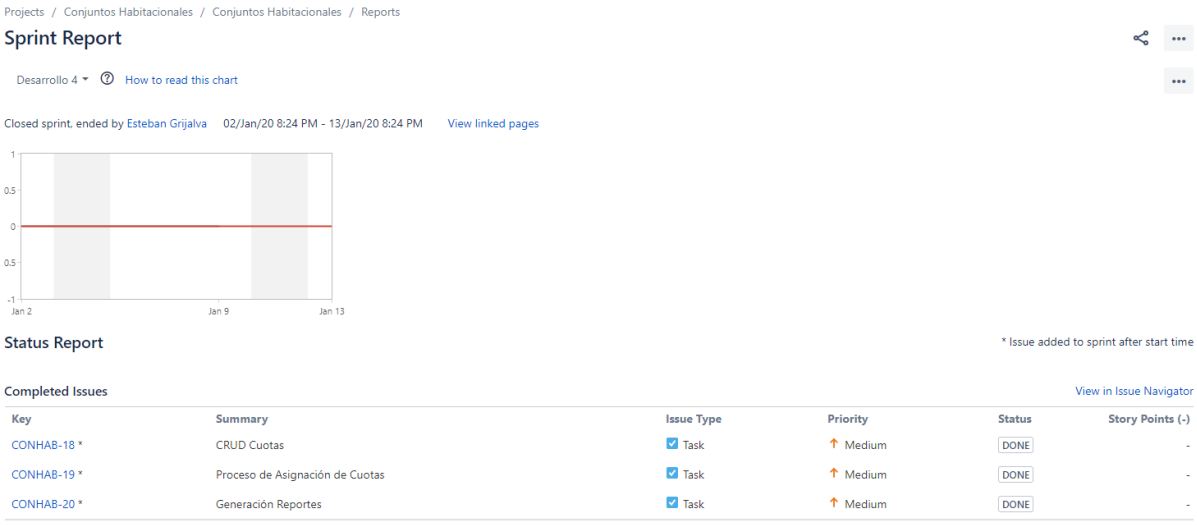


Figura 49. Reporte Sprint 8

4.2.4 Fase Validación

4.2.4.1 Validación

El documento de validación del software se encuentra en el **Anexo 7**. Para este proceso primero se realizó el despliegue del sistema funcional en la arquitectura diseñada de acuerdo con los requerimientos. Después, se agendó una reunión con el cliente final, en donde se realizó pruebas a toda la funcionalidad y sus validaciones, determinando que el desarrollo cumplió con los requerimientos solicitados.

CAPITULO V - ANÁLISIS DE RESULTADOS DE LA METODOLOGÍA PROPUESTA

5.1 FASE ANÁLISIS

5.1.1 Definición de Requerimientos

Durante esta actividad se elaboró un documento que describe los requerimientos del sistema. Este documento se basó en la entrevista, y el modelo de contexto validado por el cliente. Como resultado se pudieron obtener los requerimientos funcionales y no funcionales, y las características físicas y de seguridad del sistema.

Se identificó, sin embargo, que la definición de los requerimientos funcionales realizado de forma independiente de sus respectivos diagramas de casos de uso, pueden llevar a confusiones entre los miembros del equipo. Si bien los diagramas mencionados fueron realizados en las siguientes actividades, resultaría más efectivo modelarlos simultáneamente con la definición de los requerimientos funcionales. De esta manera se podrá tener una mejor comprensión general y específica de la funcionalidad que deberá tener el sistema al momento de su implementación.

Esta actividad fue realizada durante la ejecución de un Sprint. No hubo necesidad de extender el tiempo y el objetivo del sprint fue la creación del documento de definición de requerimientos del sistema.

5.1.2 Análisis de Requerimientos

En el análisis de requerimientos se obtuvo como resultado la definición del alcance del sistema, sus funciones, limitaciones, y especificidades plasmadas en un documento. Adicionalmente, dentro de las funciones del sistema, se realizaron los diagramas de casos que describen la interacción del usuario con el sistema, y sus actividades.

Como se mencionó en el resultado de la actividad de definición de requerimientos, es necesario la presencia tanto de los requerimientos funcionales, como de los diagramas descriptivos de casos de uso para evitar confusiones. Los documentos de definición de requerimientos y análisis de requerimientos podrían ser fusionados para obtener mejores resultados en cuanto a la comprensión de requerimientos.

Esta actividad fue realizada durante la ejecución de un Sprint. No hubo necesidad de extender el tiempo y el objetivo del sprint fue la creación del documento de definición de requerimientos del software.

5.1.3 Modelado Diagramas Uml

5.1.3.1 Diagramas de Casos de Uso

El objetivo de la diagramación de casos de uso es describir de forma gráfica la funcionalidad del sistema respecto de la interacción entre el usuario y el sistema. Esta actividad fue realizada dentro del análisis de requerimientos, por lo tanto, no deberá constar como independiente. Para esta actividad no fue necesario crear un Sprint.

5.1.3.2 Modelo de Contenido (Uwe)

El modelo de contenido de la metodología UWE contiene la descripción de todas las entidades identificadas en los requerimientos, junto con las relaciones que mantienen entre sí. La creación de este modelo facilita la visualización de la interacción de los elementos que tendrá el sistema.

El modelo de contenido es utilizado como insumo a continuación en la fase de diseño para la elaboración del diagrama de base de datos.

Esta actividad fue realizada durante la ejecución de un Sprint. No hubo necesidad de extender el tiempo y el objetivo del sprint fue la creación del modelo de contenido.

5.2 FASE DISEÑO

5.2.1 Diseño de Arquitectura

5.2.1.1 Arquitectura Física

El objetivo del diseño de la arquitectura física fue obtener el diagrama de los componentes físicos que serán utilizados al momento de desplegar el sistema. Este diagrama se basó en la definición de las características y requerimientos físicos, al igual que en las restricciones especificadas en la fase de análisis.

La arquitectura física no es definida dentro de la metodología UWE ni tampoco en SCRUM. Sin embargo, los diferentes estándares utilizados proponen realizarlo; probando su utilidad al momento de socializar la información con el equipo al igual que para efectos de explicación y presentación al cliente.

Esta actividad fue realizada durante la ejecución de un Sprint. No hubo necesidad de extender el tiempo y el objetivo del sprint fue la elaboración del diagrama de arquitectura física.

5.2.1.2 Arquitectura del Software

El resultado de la ejecución de esta actividad fue obtener el diagrama de arquitectura del software. En este se detalla todo el flujo de procesos que realizará el software partiendo desde las peticiones de usuario, continuando con el procesamiento de información, y finalizando con las respuestas del servidor.

Si bien este diagrama no es requerido para las metodologías UWE y SCRUM, tiene un rol muy importante para los procesos de los estándares aplicados. La utilidad de la arquitectura del software es mostrar al equipo encargado de la codificación, las diferentes capas de procesamiento que existirán dando a conocer, de esta manera, la estructura de programación y el manejo del código.

La definición de esta arquitectura tiene un rol muy importante en la futura mantenibilidad del sistema. Si el código es modular y está correctamente estructurado, será mucho más fácil mantenerlo a futuro.

Esta actividad fue realizada durante la ejecución de un Sprint. No hubo necesidad de extender el tiempo y el objetivo del sprint fue la elaboración del diagrama de arquitectura del software.

5.2.2 Modelado Diagramas UML

5.2.2.1 Modelo Navegación (UWE)

Durante esta actividad se obtuvo como resultado el modelo de navegación. Este diagrama expresa de manera general, y como su nombre lo indica, la navegación que se implementará para el usuario en el sistema. En este modelo intervienen los diferentes procesos que el sistema contiene y las entidades con las que cada uno de estos se relaciona.

El modelo de navegación permite visualizar el flujo de acciones que el usuario realizará durante la gestión de las entidades definidas previamente en el modelo de contenido. No obstante, puede ser un limitante o restrictivo durante la fase de implementación ya que reduce para las posibilidades de diseño de las interfaces que el sistema tendrá.

Esta actividad fue realizada durante la ejecución de un Sprint. No hubo necesidad de extender el tiempo y el objetivo del sprint fue la elaboración del modelo navegacional.

5.2.2.2 Modelo Estructura Procesos (UWE)

El modelo de estructura de procesos es un requerimiento de la metodología UWE para continuar con la transformación de modelos en la que se basa. En esta estructura se puede observar todos los procesos descritos para cada entidad junto a nuevos procesos y elementos que intervienen para completar una actividad (por ejemplo, validaciones o confirmaciones).

Este modelo es un requerimiento muy útil para el flujo de procesos que se describe en el siguiente punto. Sin embargo, la metodología propuesta no está basada en la transformación de modelos al nivel mismo nivel que UWE, por lo que se pudo haber evitado la creación de este diagrama, y enfocarse en la creación de los siguientes.

Esta actividad fue realizada durante la ejecución de un Sprint. No hubo necesidad de extender el tiempo y el objetivo del sprint fue la elaboración del modelo de estructura de procesos.

5.2.2.3 Flujo de Procesos (UWE)

El diagrama de flujo de procesos es el resultado de la transformación de los modelos navegacional y estructura de procesos. Este diagrama describe todos los procesos para cada entidad descritos de inicio a fin. Se puede observar también todos los flujos existentes durante la ejecución de cada proceso y los diferentes resultados que se presentarán.

El flujo de procesos es una herramienta excelente al momento de describir el funcionamiento de un proceso, y ayudará a clarificar cualquier duda existente sobre el requerimiento. Es

necesario contemplar todos los posibles flujos del proceso para cada elemento, con el fin de disminuir la cantidad de adaptaciones o cambios durante la fase de implementación.

Después de la diagramación del flujo de procesos, se identificó que el insumo principal y necesario es el modelo de navegación. De esta manera se pueda mejorar la metodología propuesta, eliminando un elemento, y por lo tanto, obteniendo más tiempo para las actividades restantes.

Esta actividad fue realizada durante la ejecución de un Sprint. No hubo necesidad de extender el tiempo y el objetivo del sprint fue la elaboración del modelo de flujo de procesos.

5.2.3 Diseño Base de Datos

Para la creación del diagrama de diseño de base de datos el insumo más importante fue el modelo de contenido. En base a este modelo se pudo identificar todas las entidades, con sus relaciones. Adicionalmente, se añadieron nuevos campos necesarios para la estructura de la base de datos.

Si bien, ninguna metodología de las investigadas especifica directamente la generación de un diseño de base de datos, es indispensable generar este insumo. Uno de los requerimientos del proyecto era almacenar la información, por esta razón este elemento es obligatorio.

Esta actividad fue realizada durante la ejecución de un Sprint. No hubo necesidad de extender el tiempo y el objetivo del sprint fue la elaboración del diseño de la base de datos.

5.3 FASE IMPLEMENTACIÓN

5.3.1 Definición Estrategia

Para la creación de la estrategia de implementación fue necesario el uso de la definición de requerimientos del sistema y requerimientos del software. De esta manera se pudo definir la forma en la que el sistema iba a ser desarrollado; sus configuraciones, su control de versiones, los estándares a ser utilizados, el flujo de las tareas, entre otros. Este es parte del proceso de implementación definido por la norma ISO/IEEE 12207.

El objetivo de la estrategia de implementación es determinar el proceso por el cual el software será desarrollado. Esta información es vital para el equipo encargado de la codificación, ya que define y limita las acciones a ser realizadas, forzando a los programadores a alinearse con el proceso para la implementación de los requerimientos en el sistema.

Se recomienda la creación de un documento formal conforme a las políticas de la empresa para la definición formal de la estrategia de implementación.

Esta actividad fue realizada durante la ejecución de un Sprint. No hubo necesidad de extender el tiempo y el objetivo del sprint fue la definición de la estrategia de implementación.

5.3.2 Codificación

El resultado de la codificación es el producto de software que incluye toda la funcionalidad analizada y diseñada. En este proceso se utiliza la estrategia de implementación definida en la actividad anterior.

Durante la ejecución de esta actividad se identificó que la correcta especificación de requerimientos y la correcta diagramación del flujo de procesos, facilitó la comprensión de las tareas a ser desarrolladas por el equipo. Sin embargo, durante la implementación, se encontraron falencias en los flujos de los procesos y se tuvieron que tomar decisiones correctivas para poder cumplir con los requerimientos asociados.

Esta actividad fue realizada durante la ejecución de varios Sprints. Al finalizar la ejecución del primero, se pudo evidenciar que las tareas asignadas contemplaban mucho mas tiempo y esfuerzo que lo que se había estimado. Por lo que las tareas que estaban pendientes, en proceso, o en control de calidad, tuvieron que ser movidas a un nuevo Sprint. En este segundo Sprint la planificación se realizó de mejor manera para poder corregir errores y finalizar tareas pendientes, junto con la adición de tareas nuevas que podrían ser realizadas durante el tiempo definido.

La correcta planificación de las tareas a ser desarrolladas en un Sprint para cumplir con un objetivo específico es una tarea complicada, por lo que la persona a cargo de manejar el proyecto deberá tomar las mejores decisiones respecto a que actividades se deben priorizar, y cómo manejar los tiempos y recursos para no retrasar el desarrollo del sistema.

5.3.3 Verificación

El resultado de la verificación es un acta donde el cliente certifica que el requerimiento consta en el sistema. Este proceso se realizó varias veces durante la codificación. Al completarse cada Sprint planificado se agendaba una reunión con el cliente para validar las tareas realizadas.

En este proceso se identificaron algunas falencias en la comprensión de los requerimientos del sistema. El cliente explicó algunos requerimientos nuevamente, y solventó algunas confusiones

existentes. Todas las modificaciones se planificaron para incluirse en los Sprints posteriores en la actividad de codificación.

Hay que tomar en cuenta que existe un riesgo al realizar verificaciones periódicas. El cliente podrá verse tentado a cambiar o añadir requerimientos; por lo que se deberá analizar, y viabilizar o rechazar estas nuevas solicitudes de acuerdo con los acuerdos previamente definidos.

Se recomienda añadir a la metodología la validación de los requerimientos del sistema previo a la codificación del sistema, para evitar, en la medida de lo posible, la confusión o no comprensión de los requerimientos.

5.4 FASE VALIDACIÓN

5.4.1 Validación

Este proceso consta de una reunión con el cliente para comprobar que todos los requerimientos hayan sido implementados con las validaciones requeridas, y que el funcionamiento del sistema sea integral de acuerdo con lo solicitado por el cliente.

El resultado de este proceso es un documento de aceptación del sistema, firmado por ambas partes, que certifica que el proyecto ha sido implementado de manera correcta.

Para la validación fue necesario crear el ambiente físico definido en la arquitectura, en donde se realizó la configuración y despliegue del sistema.

Esta actividad fue realizada durante la ejecución de un Sprint. No hubo necesidad de extender el tiempo y el objetivo del sprint fue el despliegue del sistema de acuerdo con la arquitectura

definida, junto con la verificación y aprobación del sistema por parte del cliente reflejado en el documento de aceptación.

CAPITULO VI - CONCLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES

6.1.1 Conclusiones del Proyecto de Investigación

- La investigación teórica de los temas pertinentes, en fuentes confiables de información, es fundamental para el desarrollo del proyecto, ya que permite obtener conocimientos clasificados y ordenados que serán usados como punto de partida hacia el cumplimiento del objetivo planteado.
- Es importante recurrir a varias fuentes de información confiables para obtener diferentes criterios y poder generar un aprendizaje. Pueden existir diversas opiniones o clasificaciones de conceptos; pero la combinación o integración de estos, brindará un valor agregado al conocimiento adquirido.
- La investigación teórica extensa sobre un tema en específico desarrolla la capacidad de análisis y decisión sobre la aceptación de diversos conceptos, evaluando y adoptando únicamente aquellos que mantengan coherencia o que sean aplicables al proyecto en desarrollo.

6.1.2 Conclusiones de la Aplicación de l Metodología Propuesta

- La definición correcta de los requerimientos tiene un rol fundamental al momento de aplicar cualquier metodología, ya que el desarrollo del proyecto se fundamentará en la información recompilada.
- La comparación de diferentes metodologías de desarrollo de software permitió adquirir una visión más clara de los objetivos y necesidades que cada una busca cumplir o satisfacer, y de esta manera se pudo utilizar los elementos que se podrían alinear en una nueva metodología.

- La metodología propuesta debió haber incluido más procesos provenientes de los estándares y/o metodologías evaluadas, ya que se identificaron algunas falencias al momento de su aplicación en diferentes fases.
- El resultado de la aplicación de la metodología para el sistema de Gestión de Conjunto Habitacional fue en su mayoría exitoso considerando su característica de aplicativo web y su alcance no tan extenso.
- La aplicación de los procesos definidos por los estándares ISO/IEEE 29148 e ISO/IEEE 12207, si bien ocupan una buena cantidad de tiempo y recursos, probaron ser de utilidad para satisfacer necesidades que las metodologías no contemplan.
- Para el desarrollo de una propuesta metodológica es indispensable identificar los factores críticos junto con sus métricas asociadas, ya que estos elementos son los que definen la dirección y el enfoque que la metodología tendrá.

6.1.3 Conclusiones Operativas

- Durante el transcurso de la carrera se estudiaron diferentes estándares y metodologías; siendo el objetivo la aplicación específica de cada uno de estos a un sistema. Sin embargo, en el campo laboral no se respeta necesariamente todos los elementos de las metodologías y estándares, sino se busca la adaptación a los procesos de la empresa. Por lo tanto, es necesario fomentar el uso de diversas herramientas para obtener un resultado integral que cumpla con las necesidades de cada proyecto.
- Las asignaturas que comprenden todo el proceso de Ingeniería de Software, en el ámbito laboral, han probado ser un complemento necesario al conocimiento formativo de profesionales de esta carrera.

6.2 RECOMENDACIONES

- Se recomienda motivar a los estudiantes de la carrera a desarrollar habilidades de investigación, que serán muy necesarias para su desenvolvimiento profesional.
- Se recomienda aplicar la metodología propuesta a sistemas con un alcance mayor para determinar su efectividad.
- Se recomienda aplicar la metodología propuesta a sistemas que no sean orientados a la web para aprobar o descartar su uso efectivo en software de diferentes características.
- Se recomienda actualizar la metodología propuesta en base a los resultados obtenidos de su aplicación e incluyendo procesos adicionales de los estándares ISO/IEEE 29148 e ISO/IEEE.
- Se recomienda enfatizar la aplicación de estándares y metodologías en el desarrollo de proyectos software de las asignaturas de la carrera, para que de esta manera los estudiantes estén adquieran el conocimiento necesario para implementar software con un buen nivel de calidad.

ANEXOS

BIBLIOGRAFÍA

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., . . . Thomas, D. (2001). *Agile Manifesto*. Retrieved from http://agilemanifesto.org/iso/es/principles.html

Braude, E. J., & Bernstein, M. E. (2016). *Software Engineering Modern Approaches.* Long Grove: Waveland Press, Inc.

CF+S, C. d. (1997, Abril 30). *Biblioteca CF+S*. Retrieved from http://habitat.aq.upm.es/iah/cepal/a007.html

Charland, A., & LeRoux, B. (2011). Mobile Application Development: Web vs. Native. *Communications of the ACM*, 49-53.

Colaboradores de Laravel. (2020). *Laravel Homestead*. Retrieved from Laravel: https://laravel.com/docs/6.x/homestead

Dick, J., Hull, E., & Jackson, K. (2017). *Requirements Engineering.* Cham: Springer.

Ibarra, E. d. (2016, Enero). *ReI*. Retrieved from Repositorio Institucional del ITESO: https://rei.iteso.mx/bitstream/handle/11117/3223/CASO%20DE%20ESTUDIO%20-%20DISE%C3%91O%20DE%20CONJUNTOS%20HABITACIONALES%20EN%20PUER.pdf?sequence=2

IEEE. (2017). *Online Browsing Platform (OBP).* Retrieved from ISO/IEC/IEEE 15939:2017: https://www.iso.org/obp/ui/#iso:std:iso-iec-ieee:15939:ed-1:v1:en:term:3.10

IEEE Computer Society. (2005). *IEEE Std 1220.* New York: The Institute of Electrical and Electronics Engineers, Inc.

IEEE Computer Society. (2014). *IEEE Standard for Software Quality Assurance Processes.* New York: IEEE Computer Society.

IEEE Computer Society. (2014). *SWEBOK V3.0 Guide to the Software Engineering Body of Knowledge.* IEEE Computer Society.

ISO. (1994). *ISO 8402 Quality Management and Quality Assurance — Vocabulary.* ISO.

ISO. (2018). *ISO Online Browsing Platform (OBP)*. Retrieved from ISO 18229:2018: https://www.iso.org/obp/ui/#iso:std:iso:18229:ed-1:v1:en:term:3.13

ISO/IEC. (1999). *Online Browsing Platform (OBP).* Retrieved from ISO/IEC 14776-411:1999: https://www.iso.org/obp/ui/#iso:std:iso-iec:14776:-411:ed-1:v1:en:term:3.1.9

ISO/IEC. (2007). *ISO 15939 Software Engineering - Software Measurement Process.* Geneva: ISO/IEC.

ISO/IEC. (2013). *ISO Online Browsing Platform (OBP)*. Retrieved from ISO/IEC 20944-1:2013: https://www.iso.org/obp/ui/#iso:std:iso-iec:20944:-1:ed-1:v1:en:term:3.1.3.2

ISO/IEC. (2013). *ISO Online Browsing Platform (OBP)*. Retrieved from ISO/IEC 20944-1:2013: https://www.iso.org/obp/ui/#iso:std:iso-iec:20944:-1:ed-1:v1:en:term:3.1.3.9

ISO/IEC/IEEE. (2010). *ISO/IEC/IEEE 24765:2010 Systems and software engineering - Vocabulary.* Switzerland: ISO.

ISO/IEC/IEEE. (2011). *ISO/IEC/IEEE 29148.* Geneva: ISO/IEC/IEEE.

ISO/IEC/IEEE. (2017). *ISO/IEC/IEEE 12207 Systems and software engineering — Software life cycle processes.* Geneva: ISO/IEC/IEEE.

ISO/IEC/IEEE. (2017). *Online Browsing Platform (OBP).* Retrieved from ISO/IEC/IEEE 24765:2017 Systems and software engineering — Vocabulary: https://www.iso.org/obp/ui/#iso:std:iso-iec-ieee:24765:ed-2:v1:en:term:3.3790

Koch, N. (2007). Classification of Model Transformation Techniques used in UML-based Web Engineering. *IET Softw*, 98-111.

Laplante, P. A. (2009). *Requirements Engineering for Software and Systems.* Boca Raton: CRC Press.

Leach, R. J. (2016). *Introduction to Software Engineering.* Boca Raton: CRC Press.

Mall, R. (2014). *Fundamentals of Software Engineering.* Delhi: PHI Learning Private Limited.

Muller, G. (2011). *Fundamentals of Requirements Engineering.* Kongsberg: CRC Press.

Nanda, V. (2005). *Quality Management System Handbook for Product Development Companies.* Boca Raton: CRC Press.

Offutt, J. (2002). Quality Attributes of Web Software Applications. *IEEE SOFTWARE* , 25 - 32.

Rossi, G., Pastor, O., Schwabe, D., & Olsina, L. (2008). *Web Engineering: Modelling and Implementing Web Applications.* London: Springer.

Schulmeyer, G. G. (2008). *Handbook of Software Quality Assurance*. Norwood: ARTECH HOUSE, INC.

Schwaber, K., & Sutherland, J. (2017, November). *Scrum Guides.* Retrieved from The Scrum Guide: https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf

SeaLights. (2018). *Sealights*. Retrieved 01 23, 2019, from https://www.sealights.io/software-development-metrics/11-scrum-metrics-and-their-value-to-scrum-teams/

The Institute of Electrical and Electronics Engineers . (1990). *IEEE Standard Glossary of Software Engineering Terminology.* New York.

The Standish Group. (2014). *Project Smart.* Retrieved 01 19, 2019, from The Standish Group Report CHAOS: https://www.projectsmart.co.uk/white-papers/chaos-report.pdf

UML-Based Web Engineering: An Approach Based on Standards. (2008). In N. Koch, A. Knapp, G. Zhang, & H. Baumeister, *Web Engineering: Modelling and Implementing Web Applications* (pp. 157 - 191). Springer.

Zave, P., & Jackson, M. (1997). Four Dark Corners of Requirements Engineering. *ACM Transactions on Software Engineering and Methodology, 6*(1), 1-30.