Universidad de las Ciencias Informáticas
Facultad CITEC

Aplicación web para la generación de instrumentos de recolección de datos

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor:

Marcelo Eduardo Exposito Blanco

Tutores:

Dr.C Yusniel Hidalgo Delgado

MSc. Yordanis García Leiva

Noviembre, 2022

“Año 64 del Triunfo de la Revolución”

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los _____ días del mes de _____ del año _____.

_____

Marcelo Eduardo Exposito Blanco

_____ _____

MSc. Yordanis García Leiva Dr.C Yusniel Hidalgo Delgado

AGRADECIMIENTOS

Quisiera agradecer a todas aquellas personas que de alguna forma u otra me han ayudado a afrontar todo el proceso que ha significado mi carrera y en particular mi tesis

En especial a mi padre que siempre ha sido mi apoyo moral y nunca ha dejado de confiar en mí, también tienen un lugar especial en mis agradecimientos dos personas que son el reflejo de que no solo tengo una madre sino dos: mi abuela que se ha mantenido firme a mi lado siendo una parte esencial de mi vida en todo momento y mi madre que aunque se encuentre a distancia siempre ha estado ahí para mi dándome apoyo y amor constante como solo una madre sabe hacer y en especial a una persona que a pesar de que ya no se encuentre entre nosotros es y será una motivación más que presente para graduarme y culminar mis estudios mi abuelo Exposito.

En todo este proceso universitario uno se encuentra con personas que marcan nuestra vida para siempre y que no se le puede dejar de agradecer porque han influido de manera directa o indirecta para poder llegar a este último paso, ese es el caso de mis amigos y compañeros a lo largo de estos años, en especial Raikol y David que desde el principio hemos formado una amistad que va a perdurar para siempre. Una persona que ha influido de manera muy directa en que este sueño de graduarme se haga realidad es mi amigo Yordanis que además de ser mi tutor ha sido como un guía para formarme como profesional.

También quisiera agradecer a mi tutor Yusniel que a pesar de las dificultades me acepto como su tesista y me ha ayudado crecer como ingeniero informático.

Un agradecimiento para todos los profesores que a lo largo de estos años me han brindado su conocimiento y ayuda para convertirme en mejor persona y un gran profesional

DEDICATORIA

Quisiera dedicarles esta tesis a todas las personas que confiaron en mi desde el principio de mi carrera y que siempre me apoyaron y me dieron ánimos para continuar a pesar de las dificultades.

Quiero hacer una dedicatoria especial a una persona que desde que era un niño confiaba y pedía que fuera un profesional y abogó siempre porque estudiara para que fuera alguien en la vida, ese es mi abuelo Exposito que este año lamentablemente falleció y no me pudo ver cumplir mi sueño y el suyo, pero sé que este donde este va a sentirse orgulloso por mi

También quiero dedicar esta tesis a mis padres y mis abuelos que siempre me inculcaron una ideología para formarme y prepararme para tener un título y así tener un futuro mejor esto también es para ellos

RESUMEN

Un instrumento de recolección de datos es un recurso que sirve al investigador para recolectar la información necesaria para desarrollar su proyecto investigativo. Los instrumentos de recolección de datos se utilizan en las investigaciones para extraer información de los eventos o fenómenos que se quieren analizar. Entre los instrumentos de recolección de datos se encuentran los cuestionarios. Los cuestionarios son aplicados con frecuencia en la actividad científica, así como en procesos de corte social, económico, político o de otra rama. La mayoría de los cuestionarios aplicados en Cuba se realizan con el uso del papel. Lo anterior provoca lentitud en el proceso de recopilación y análisis de los datos, así como una reducción en la variedad geográfica de la composición de la muestra. La presente investigación tiene como propósito desarrollar una aplicación web para la generación de instrumentos de recolección de datos que aumente la variedad geográfica de la composición de la muestra y la celeridad en el proceso. El desarrollo de la solución está guiado por el uso de la metodología de desarrollo de software Programación Extrema y la utilización de tecnologías de código abierto. El resultado de la investigación fue validado aplicando pruebas de software. Además, se verifica el cumplimiento del objetivo general de la investigación a partir de la definición de un conjunto de indicadores que permiten evaluar la relación causa-efecto de la variable independiente sobre las variables dependientes de la investigación.

PALABRAS CLAVE: aplicación web, cuestionario, instrumento, recolección de datos

ABSTRACT

A data collection instrument is a resource that is used by the researcher to collect the information needed to develop the research project. Data collection instruments are used in research to extract information about the events or phenomena to be analyzed. Data collection instruments include questionnaires. Questionnaires are frequently applied in scientific activity, as well as in social, economic, political or other processes. Most of the questionnaires used in Cuba are paper-based. This causes slowness in the process of data collection and analysis, as well as a reduction in the geographical variety of the composition of the sample. The purpose of this research is to develop a web application for the generation of data collection instruments that increases the geographical variety of the sample composition and the speed of the process. The development of the solution is guided by the use of the Extreme Programming software development methodology and the use of open source technologies. The result of the research was validated by applying software tests. In addition, the fulfilment of the general objective of the research is verified from the definition of a set of indicators that allow to evaluate the cause-effect relationship of the independent variable on the dependent variables of the research.

KEYWORDS: *data collection, instrument, questionnaire, web application*

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La estadística es una ciencia y una rama de las matemáticas a través de la cual se recolecta, analiza, describe y estudia una serie de datos, a fin de establecer comparaciones o variabilidades que permitan comprender un fenómeno en particular (Significados, 2022).

La estadística es una de las ramas de la matemática que constituye un método científico que pretende sacar conclusiones a partir de observaciones hechas en muestras tomadas. Permite reunir datos sobre un tema, agruparlos para una mejor comprensión y con ello, tomar mejores decisiones. A grandes rasgos esta se centra en la observación para la recolección de datos que posteriormente serán analizados y comparados con el fin de obtener un resultado (Significados, 2022).

Para el desarrollo de un análisis estadístico se utilizan los instrumentos de recolección de datos. Un instrumento de recolección de datos es un recurso que sirve al investigador para recolectar la información necesaria para desarrollar su proyecto investigativo. Los instrumentos de recolección de datos se utilizan en las investigaciones para extraer información de los eventos o fenómenos que se quieren analizar (tesisymasters, 2022).

Entre los instrumentos de recolección de datos se encuentran los cuestionarios. Los cuestionarios constituyen documentos formados por un conjunto de preguntas que deben estar redactadas de forma coherente, organizadas, secuenciadas y estructuradas, de acuerdo con una determinada planificación, con el fin de que sus respuestas puedan ofrecer toda la información necesaria. Los cuestionarios permiten extraer determinada información por medio de preguntas o cuestiones de un grupo de personas a las cuales se han seleccionado como muestra para realizar procesos de investigación o evaluación. Estos suelen aplicarse al público cuya opinión desea conocerse o someter a análisis. También se aplica para evaluar los conocimientos de la muestra poblacional seleccionada, mayormente estos últimos son aplicados en la rama de la educación (Meneses, 2016).

En Cuba se aplican los cuestionarios en varias esferas, con el fin de tomar decisiones. Los cuestionarios son aplicados con frecuencia en la actividad científica, así como en procesos de corte social, económico, político o de otra rama. La mayoría de los cuestionarios aplicados en Cuba se realizan con el uso del papel. Lo anterior provoca lentitud en el proceso de recopilación y análisis de los datos, así como una reducción en la variedad geográfica de la composición de la muestra, pues solo se puede llegar al personal que esté en ese momento al alcance del instrumento diseñado en una hoja de papel. Por otra parte, el procesamiento manual de los datos recolectados puede provocar errores en la obtención de las estadísticas del proceso.

A partir de la problemática antes descrita se genera la necesidad de resolver el siguiente **problema de investigación**: ¿cómo gestionar la recogida de información a través de instrumentos de recolección de datos aumentando la variedad geográfica de la composición de la muestra y la celeridad en el proceso? Esto nos lleva a tomar como **objeto de estudio:** instrumentos de recolección de datos.

Con el fin de solucionar el problema planteado se define como **objetivo general:** desarrollar una aplicación web para la generación de instrumentos de recolección de datos que aumente la variedad geográfica de la composición de la muestra y la celeridad en el proceso. Para dar cumplimiento al objetivo propuesto se han derivado un conjunto de **objetivos específicos:**

- Elaborar el marco teórico de la investigación mediante el estudio de los referentes teóricos sobre el desarrollo aplicaciones web para la captación de información.
- Realizar la identificación de los requisitos, análisis y diseño de la solución propuesta.
- Validar los resultados de la investigación aplicando técnicas, métricas y pruebas de software.

Enmarcándose en el **campo de acción:** instrumento de recolección de datos a través de la web. Definiéndose como **hipótesis:** si se desarrolla una aplicación web

para la generación de instrumentos de recolección de datos se aumenta la variedad geográfica de la composición de la muestra y la celeridad en el proceso.

Métodos científicos a emplear:

Los métodos científicos son empleados para explicar fenómenos, establecer relaciones entre los hechos, enunciar leyes que expliquen los fenómenos físicos del mundo, que permitan obtener con estos, conocimientos y aplicaciones útiles al hombre (Ceballos, 2004). En la presente investigación se hace uso de los métodos:

Métodos Teóricos:

Histórico-Lógico: permitió realizar un estudio de las herramientas utilizadas en la generación de instrumentos de recolección de datos existentes en el ámbito nacional como internacional.

Analítico-Sintético: permitió la realización del diseño teórico de la investigación, extrayéndose los elementos más importantes relacionados con los cuestionarios y encuestas, así como las herramientas y tecnologías.

Modelación: permitió la confección de los prototipos funcionales, el diseño arquitectónico y los diagramas de diseño.

Métodos Empíricos:

Observación: se utilizó a través del estudio realizado a diferentes sistemas dedicados al procesamiento de cuestionarios y encuestas que existen en el mundo, con el fin de determinar los elementos más comunes que están presentes en las mismas.

Medición: permitió medir la calidad de la especificación de los requisitos y el grado de ambigüedad de estos, además de obtener una medida de la calidad del diseño para su validación.

Entrevista: fue utilizado para comprender las necesidades del cliente, capturar los requisitos correspondientes a la aplicación y obtener información que apoye la realización de la investigación.

El contenido de este trabajo consta de tres capítulos, definidos de la siguiente forma:

Capítulo 1. Fundamentos y referentes teórico-metodológicos sobre el objeto de estudio.

En este capítulo se definen los conceptos básicos relacionados con la problemática y las tendencias actuales en cuanto a su campo de acción. Se realiza una descripción detallada de la metodología seleccionada para guiar el proceso de desarrollo de la presente investigación y la herramienta CASE utilizada. Además, se realiza una descripción de los lenguajes, herramientas, tecnologías y patrones de diseño utilizados en la solución.

Capítulo 2. Análisis, diseño e implementación de la solución propuesta: en el capítulo se describe el negocio a informatizar, se definen los requisitos funcionales y no funcionales. De la disciplina de análisis y diseño se ilustra y describe la arquitectura utilizada, así como el diagrama de clases del diseño, de igual forma se ejemplifica la utilización de patrones de diseño. De la disciplina de implementación se muestran los estándares de codificación que organizan la estructura del código.

Capítulo 3. Validación de la solución propuesta: se realiza la validación de los requisitos del sistema de acuerdo con los requisitos funcionales y no funcionales previamente planteados. Se muestran los resultados de la validación del diseño, así como de la ejecución de las pruebas de software en los cuatro niveles. También se demuestra cómo si se desarrolla una aplicación web para la generación de instrumentos de recolección de datos se aumenta la variedad geográfica de la composición de la muestra y la celeridad en el proceso.

CAPÍTULO 1: FUNDAMENTOS Y REFERENTES TEÓRICO-METODOLÓGICOS SOBRE EL OBJETO DE ESTUDIO

En este capítulo se describen conceptos relacionados con la problemática antes mencionada, las herramientas, lenguajes y tecnologías que se emplean en la construcción de la solución, así como aspectos esenciales que sirvan de soporte a la misma. Se describe la metodología seleccionada para guiar el proceso de desarrollo de la presente investigación.

1.1 Conceptos fundamentales asociados al dominio del problema

Para una mejor comprensión del tema de investigación, es necesario dominar las siguientes definiciones:

Recolección de datos: se refiere al enfoque sistemático de reunir y medir información de diversas fuentes a fin de obtener un panorama completo y preciso de una zona de interés. Esta permite a un individuo o empresa responder a preguntas relevantes, evaluar los resultados y anticipar mejor las probabilidades y tendencias futuras. La exactitud en la reunión de datos es esencial para garantizar la integridad de un estudio, las decisiones comerciales acertadas y la garantía de calidad (questionpro, 2022).

Instrumento de recolección de datos: es un recurso que sirve al investigador para recolectar la información necesaria para desarrollar su proyecto investigativo. Su principal característica es que vale para extraer datos directos de los fenómenos y/o población que se desea investigar (tesisymasters, 2022).

Existen varios tipos de instrumentos de recolección de datos como son:

- Entrevistas
- Observaciones
- Documentos y archivos

- Experimentos de laboratorio
- Cuestionarios en papel y cuestionarios online
- Formularios

Para el desarrollo de la presente investigación es importante dominar la definición de cuestionario y formulario.

Un cuestionario es aquel que plantea una serie de preguntas para extraer determinada información de un grupo de personas. Este permite recolectar información y datos para su tabulación, clasificación, descripción y análisis en un estudio o investigación. En este sentido, los cuestionarios pueden usarse como instrumentos de recolección de datos, o como herramienta de evaluación en el ámbito escolar (Significados, 2022) .

Los cuestionarios *online* o encuestas web o encuestas por internet constituyen un método de recopilación de datos en el que se envían cuestionarios a una muestra de encuestados y éstos pueden responder esta herramienta a través de la red**.** La razón principal del crecimiento visto en la implementación de las encuestas web es que son más económicas y tienen un gran alcance. A los encuestados se les pueden enviar encuestas a través de diversos medios, como el correo electrónico, páginas de internet y las redes sociales (questionpro, 2022) .

Por otra parte, los formularios constituyen una plantilla o página que cuenta con casilleros, o en su defecto espacios vacíos, los cuales están destinados para ser rellenados por algún individuo con alguna finalidad (definicionabc, 2022).

Uno de los medios a través de los cuales se pueden aplicar instrumentos de recolección de datos lo constituyen las redes sociales. Estas son plataformas digitales formadas por comunidades de individuos con intereses, actividades o relaciones en común. Las redes sociales permiten el contacto entre personas y funcionan como un medio para comunicarse e intercambiar información. Los individuos no necesariamente se tienen que conocer antes de entrar en contacto a

través de una red social, sino que pueden hacerlo a través de ella, y ese es uno de los mayores beneficios de las comunidades virtuales (concepto, s.f.) .

Las redes sociales son una gran fuente recolección de información para análisis de datos gracias a la gran interacción que posee los usuarios que en ellas interactúan. Para la integración de los instrumentos de recolección de datos a estas redes existen varios medios, uno muy efectivo son las aplicaciones web.

En la ingeniería de software se denomina aplicación web a aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de internet o de una intranet mediante un navegador. En otras palabras, es una aplicación (software) que se codifica en un lenguaje soportado por los navegadores web en la que se confía la ejecución al navegador (ecured.cu, s.f.).

1.2 Tendencias actuales

Con el propósito de buscar cuánto se ha avanzado en el desarrollo de soluciones para la generación de instrumentos de recolección de datos, se realizó un estudio de los siguientes sistemas:

Zoho Forms: es un creador de formularios en línea sin codificación, con función de arrastrar y soltar, utilizado para crear formularios para cada propósito. Te proporciona una variedad de opciones de personalización, admite asignaciones de tareas, flujos de trabajo de aprobación y también es receptivo a navegadores móviles. Zoho Forms se integra bien con los productos de Zoho, Salesforce y las aplicaciones de Google. También ofrece aplicaciones nativas para dispositivos iOS y Android que te permiten crear formularios y recopilar datos directamente desde tu dispositivo móvil, y también funciona sin conexión (capterra, s.f.).

Formsite: crea formularios .HTML[1] y encuestas web profesionales en línea. Elige entre más de 100 plantillas de formularios web preconstruidas que se pueden

[1] *HyperText Markup Language*

personalizar para inscripciones, reservas, pedidos seguros, encuestas a clientes y cobro de pagos. El editor con función de arrastrar y soltar permite diseñar cualquier tipo de formulario o encuesta web. Los formularios sensibles son estándar, lo que les da un aspecto excelente en todos los dispositivos (capterra, s.f.).

SurveyMonkey Enterprise: permite a las empresas registrar los comentarios críticos y actuar en base a ellos para impulsar el crecimiento y la innovación a escala con funciones avanzadas de seguridad y cumplimiento. Integra los comentarios de las encuestas con la pila tecnológica para fundamentar la toma de decisiones y automatizar los flujos de trabajo. Los productos, soluciones empresariales e integraciones de SurveyMonkey permiten a más de 335 000 organizaciones ofrecer mejores experiencias a sus clientes, aumentar la retención de empleados y potenciar el crecimiento y la innovación (capterra, s.f.).

Typeform: posee formularios interactivos y personalizables. Tiene integraciones con más de 120 aplicaciones permiten que este se integre sin problemas en un flujo de trabajo investigativo. Se caracteriza por su fácil implementación: sin programación ni incorporación. También ofrece seguridad de nivel empresarial para mantener los datos protegidos de sus usuarios (capterra, s.f.).

Google Forms: es una de las herramientas de Google que permite crear y publicar formularios personalizables que pueden compartirse entre muestras mediante una URL web. Esta posibilita que individuos y empresas de todos los tamaños produzcan formularios personalizables para realizar encuestas y generar gráficos en tiempo real con las respuestas (capterra, s.f.).

En la siguiente tabla se realiza un resumen del análisis realizado sobre las diferentes soluciones estudiadas. Para el análisis de estas soluciones se definen un conjunto de criterios que permiten comparar las mismas. La selección de los criterios se realiza a partir de las principales características que debe cumplir la solución propuesta.

Tabla 1. Comparación de las soluciones estudiadas. Fuente: elaboración propia

Sistemas/Indicadores	Software Libre	Exporta Información	Integración con Redes Sociales	Personalizable	Soberanía tecnológica
Zoho Forms	Sí	No	No	Sí	No
Formsite	No	Sí	Sí	Sí	No
SurveyMonkey Enterprise:	No	No	No	Sí	No
Typeform	Sí	No	Sí	Sí	No
Google Form	SÍ	SÍ	SÍ	SÍ	No

Después del análisis anterior, se llegó a la conclusión que de las soluciones existentes para la generación de instrumentos de recolección de datos en línea ninguna cumple en su totalidad con los indicadores definidos a partir de las necesidades que debe cubrir la solución que se propone. Lo anterior demuestra la importancia del desarrollo de la solución propuesta. Sin embargo, el estudio realizado, permitió tener en cuenta algunas de las características de estas soluciones para incorporarlas a la aplicación web que se propone.

1.3 Metodología de desarrollo de software

Las metodologías de desarrollo de software son un conjunto de técnicas y métodos organizativos que se aplican para diseñar soluciones de software. El objetivo de las distintas metodologías es el de intentar organizar los equipos de trabajo para que estos desarrollen las funciones de un programa de la mejor manera posible.

Las metodologías de desarrollo de software en la actualidad se dividen en dos enfoques: El enfoque tradicional y el ágil.

Las metodologías de desarrollo de software tradicionales se caracterizan por definir total y rígidamente los requisitos al inicio de los proyectos de ingeniería de software. Los ciclos de desarrollo son poco flexibles y no permiten realizar cambios.

En cambio, las metodologías ágiles se basan en la metodología incremental, en la que en cada ciclo de desarrollo se van agregando nuevas funcionalidades a la aplicación final. Sin embargo, los ciclos son mucho más cortos y rápidos, por lo que se van agregando pequeñas funcionalidades en lugar de grandes cambios.

En la presente investigación se decidió optar por una metodología con un enfoque más ágil debido a la complejidad de la aplicación a desarrollar y las necesidades del proyecto.

Las principales metodologías ágiles son:

- Kanban: metodología de trabajo inventada por la empresa de automóviles Toyota. Consiste en dividir las tareas en porciones mínimas y organizarlas en un tablero de trabajo dividido en tareas pendientes, en curso y finalizadas. De esta forma, se crea un flujo de trabajo muy visual basado en tareas prioritarias e incrementando el valor del producto.
- Scrum: es también una metodología incremental que divide los requisitos y tareas de forma similar a Kanban. Se itera sobre bloques de tiempos cortos y fijos (entre dos y cuatro semanas) para conseguir un resultado completo en cada iteración. Las etapas son: planificación de la iteración (*planning sprint*), ejecución (sprint), reunión diaria (daily meeting) y demostración de resultados (*sprint review*). Cada iteración por estas etapas se denomina también sprint.
- Lean: está configurado para que pequeños equipos de desarrollo muy capacitados elaboren cualquier tarea en poco tiempo. Los activos más importantes son las personas y su compromiso, relegando así a un segundo plano el tiempo y los costes. El aprendizaje, las reacciones rápidas y potenciar el equipo son fundamentales.
- Programación extrema (XP): es una metodología de desarrollo de software basada en las relaciones interpersonales, que se consideran la clave del éxito. Su principal objetivo es crear un buen ambiente de trabajo en equipo y que haya un *feedback* constante del cliente. El trabajo se basa en 12 conceptos: diseño sencillo, *testing*, refactorización y codificación con

estándares, propiedad colectiva del código, programación en parejas, integración continua, entregas semanales e integridad con el cliente, cliente *in situ*, entregas frecuentes y planificación.

Luego del análisis de las metodologías agiles se ha decidido utilizar en el desarrollo de la aplicación propuesta la metodología Programación Extrema (XP), ya que se adecua con los requerimientos exigidos. Debido al grado de adaptabilidad esta se basa en la constante comunicación y la retroalimentación. Uno de sus fines principales es el de construir un producto que vaya en línea con los requerimientos del cliente y las necesidades del proyecto. Esta metodología consta de 4 fases *(Solis, 2003)*:

- Planificación: consiste en realizar una recopilación de todos los requerimientos del proyecto, se crean las HU, el plan de iteraciones y el plan de entregas.

- Diseño: se basa en conseguir diseños simples y sencillos, con el objetivo de procurar hacerlo todo lo menos complicado posible para el cliente o usuario, se crean las tarjetas CRC, las cuales definen una clase expresando sus funcionalidades y las otras clases con las que colabora.

- Desarrollo: consiste en establecer una buena comunicación entre el equipo y el cliente, para que los desarrolladores puedan codificar todo lo necesario para el proyecto que se requiere. Además, se definen las tareas de ingeniería para que los desarrolladores tengan una guía para implementar todas las HU.

- Pruebas: consiste en comprobar el funcionamiento de la codificación que se halla implementado, garantizando la evaluación de las distintas tareas en las que ha sido divididas las HU.

1.4 Herramientas de Ingeniería del Software Asistidas por Computadoras

Las herramientas CASE comprenden un conjunto de programas de diferentes tipos empleados para ayudar a las actividades del proceso de desarrollo de software,

tales como: análisis de requerimientos, modelado de sistemas, depuración y pruebas *(Sommerville, 2005).*

Actualmente existen diversas herramientas CASE que soportan el lenguaje de modelado UML (Lenguaje Unificado de Modelado), dentro de las que se destaca *Visual Paradigm for UML (VP-UML)* por ser bastante completas, disponen de varios diagramas como los de clases, de objetos, de casos de uso del negocio y de paquetes, por la facilidad de uso para la creación de estos y por generar código a partir de los mismos. Visual Paradigm es gratuito, aunque se encuentra bajo una licencia que no permite su modificación o venta, ajustándose esta última a las políticas de soberanía tecnológica trazadas por el país.

Teniendo en cuenta la importancia del uso del software libre en Cuba, la existencia de una licencia UCI para el uso de Visual Paradigm y sus características, se selecciona, en su versión 8.0, para realizar el diseño UML de la herramienta propuesta en el presente trabajo, ya que posee varias ventajas, entre las que se encuentran: la posibilidad de utilizarla en múltiples plataformas, permite el modelado de bases de datos y su posterior exportación a código SQL (Lenguaje de Consulta Estructurado), es capaz de generar toda la documentación necesaria. Además, se puede utilizar en cualquier fase del desarrollo del software, contribuyendo esto a garantizar una mayor flexibilidad y agilidad para la adaptación al cambio.

1.5 Lenguajes de programación

Un lenguaje de programación es un lenguaje artificial que se utiliza para definir una secuencia de instrucciones para su procesamiento en un ordenador. Está formado por un conjunto de símbolos, palabras claves utilizables y por reglas gramaticales para construir sentencias sintáctica y semánticamente correctas *(Rodríguez, 2003)*.

De los lenguajes de programación que existen para desarrollar aplicaciones orientadas a la web se encuentran dos grupos fundamentales, la programación del lado del servidor y la programación del lado del cliente. Esta última incluye aquellos lenguajes que son interpretados por una aplicación cliente como el navegador web,

entre ellos se encuentra HTML (por sus siglas en inglés*, HyperText Markup Language*). Los lenguajes de programación del lado del servidor son reconocidos, ejecutados e interpretados por el propio servidor, el que se encarga de brindar información al cliente en un formato comprensible para él. Para el desarrollo de la solución propuesta se hace necesario utilizar los lenguajes que a continuación se describen, teniendo en cuenta que responden a las tecnologías y lenguajes definido por el grupo de investigación al cual tributa la presente tesis.

Python 3.10.5

Python se caracteriza principalmente como lenguaje por su sintaxis simple y expresiva, cuyo objetivo principal es aumentar la legibilidad y la facilidad de desarrollo. Es un lenguaje de programación versátil multiplataforma y multiparadigma que se destaca por su código legible y limpio. Una de las razones de su éxito es que cuenta con una licencia de código abierto que permite su utilización en cualquier escenario. Esto hace que sea uno de los lenguajes de iniciación de muchos programadores siendo impartido en escuelas y universidades de todo el mundo. Sumado a esto cuenta con grandes compañías que hacen de este un uso intensivo. Tal es el caso de Google, Facebook o Youtube, ya que permite, entre otras de sus características la automatización de procesos y ejecución de tareas tanto en entornos cliente como servidor (Robledano, 2019**)**.

JavaScript ES6

Es un lenguaje de programación ligero, interpretado por la mayoría de los navegadores y que les proporciona a las páginas web, efectos y funciones complementarias a las consideradas como estándar HTML. Este tipo de lenguaje de programación es de código abierto, por lo que cualquier persona puede utilizarlo sin comprar una licencia. Con frecuencia son empleados en los sitios web, para realizar acciones en el lado del cliente, estando centrado en el código fuente de la página web (Mozilla, 2020).

HTML 5.0

HTML es el lenguaje que se utiliza para crear las páginas web de internet. Es reconocido universalmente y permite publicar información de forma global. Define una estructura básica y un código HTML para la definición de contenido de una página web, como texto, imágenes, entre otros y se basa en la referenciación por hipertextos o enlaces entre páginas (Mozilla, 2019).

CSS 3.0

CSS (por sus siglas en inglés, *Cascading Style Sheets*) es un lenguaje para controlar la presentación de los documentos electrónicos definidos con HTML y XHTML (por sus siglas en inglés*, Extensible HyperText Markup Language*). CSS es la mejor forma de separar los contenidos de su presentación y es imprescindible para la creación de páginas web complejas. Mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes. Se utiliza para definir el aspecto de todos los contenidos, como: el color, tamaño y tipo de letra de los párrafos de texto, la separación entre titulares y párrafos, la tabulación con la que se muestran los elementos de una lista. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos. Funciona a base de reglas para las declaraciones sobre el estilo de uno o más elementos (Mozilla, 2019).

1.6 Marcos de trabajo

Desde el punto de vista del desarrollo de software, un marco de trabajo es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado (Alegsa, 2020).

Django 4.1

Django es un marco de trabajo de desarrollo web de código abierto, escrito en Python, que cumple en cierta medida el paradigma del Modelo Vista Controlador.

La meta fundamental de Django es facilitar la creación de sitios web complejos. Django pone énfasis en el re-uso, la conectividad y extensibilidad de componentes, del desarrollo rápido y del principio de DRY (del inglés *Don't Repeat Yourself*). Python es usado en todas las partes del marco de trabajo, incluso en configuraciones, archivos, y en los modelos de datos (ecured.cu, s.f.).

Para el desarrollo de la solución propuesta se utiliza el marco de trabajo antes descrito, teniendo en cuenta que responde a las tecnologías definidas por el grupo de investigación al cual tributa la presente tesis.

1.7 Editor de código

Los editores de código son programas que ayudan a gestionar el código fuente de los proyectos. Son ideales cuando se trabaja con diferentes lenguajes de programación, alternándolos o en un solo proyecto (por ejemplo, en un proyecto web es muy habitual combinar html, javascript, css, php entre otros).

Visual Studio Code 1.70.2

Visual Studio Code (VS Code) es un editor de código fuente desarrollado por Microsoft. Es software libre y multiplataforma, está disponible para Windows, GNU/Linux y macOS. VS Code tiene una buena integración con Git[2], cuenta con soporte para depuración de código, y dispone de un sin número de extensiones, que básicamente da la posibilidad de escribir y ejecutar código en cualquier lenguaje de programación.

Para el desarrollo de la solución propuesta se utiliza el editor de código antes descrito, teniendo en cuenta que responde a las tecnologías definidas por el grupo de investigación al cual tributa la presente tesis.

[2] Sistema de control de versiones distribuido

1.8 Sistema Gestor de Bases de Datos

Un Sistema Gestor de Base de Datos (SGBD) es un conjunto de programas, procedimientos y lenguajes que proporcionan las herramientas necesarias para trabajar con una base de datos. Incorpora una serie de funciones que permiten definir los registros, sus campos, sus relaciones, insertar, suprimir, modificar y consultar los datos *(Riquelme, 2010).*

SQlite 3.39.2

SQLite es una herramienta de software libre, que permite almacenar información en dispositivos empotrados de una forma sencilla, eficaz, potente, rápida. Implementa el estándar SQL92[3] y también agrega extensiones que facilitan su uso en cualquier ambiente de desarrollo. Esto permite que SQLite soporte desde las consultas más básicas hasta las más complejas del lenguaje SQL, y lo más importante es que se puede usar tanto en dispositivos móviles como en sistemas de escritorio, sin necesidad de realizar procesos complejos de importación y exportación de datos, ya que existe compatibilidad entre las diversas plataformas disponibles, haciendo que la portabilidad entre dispositivos y plataformas sea transparente (Rómmel, s.f.).

Para el desarrollo de la solución propuesta se utiliza el SGBD antes descrito, teniendo en cuenta que responde a las tecnologías definidas por el grupo de investigación al cual tributa la presente tesis.

1.9 Patrones de diseño

Los patrones de diseños son un conjunto de soluciones a problemas de diseño que ocurren una y otra vez en el desarrollo de software. Funcionan como una plantilla de solución donde una solución abstracta es adaptada por el usuario a su problema. Además, los patrones de diseño proveen un idioma común entre distintos lenguajes de programación que permite a arquitectos y desarrolladores comunicarse de

[3] Tercera revisión del lenguaje SQL

manera común y abordar problemas sin importar el lenguaje de programación utilizado (Rocha, 2018).

Objetivos de estos patrones**:**

- Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

Así mismo, no pretenden:

- Imponer ciertas alternativas de diseño frente a otras.
- Eliminar la creatividad inherente al proceso de diseño.
- No es obligatorio utilizar los patrones siempre, sólo en el caso de tener el mismo problema o similar que soluciona el patrón, siempre teniendo en cuenta que en un caso particular puede no ser aplicable. Abusar o forzar el uso de los patrones puede ser un error (Rouse M. , Tech Target, 2020).

En el capítulo 2 el autor de la presente investigación describe el uso de los patrones utilizados en el diseño e implementación de la aplicación web propuesta.

1.10 Pruebas

Uno de los elementos principales para certificar la calidad de una aplicación informática lo constituye el resultado de las pruebas que se practican. Para determinar el nivel de calidad se deben efectuar medidas o pruebas que permitan comprobar el grado de cumplimiento respecto a las especificaciones principales del sistema (Pressman R. , 2010). Teniendo en cuenta lo anterior a continuación se define el concepto de pruebas de software.

Pruebas de software: comprenden el conjunto de actividades que se realizan para identificar posibles fallos de funcionamiento, configuración o usabilidad de un programa o aplicación, por medio de pruebas sobre el comportamiento del mismo (Pressman R. , 2010).

Las pruebas de software se caracterizan por niveles o etapas**.** A continuación, se muestra una breve descripción de cada nivel o etapa de prueba:

- Pruebas modulares o a nivel de unidad: este tipo de pruebas son ejecutadas normalmente por el equipo de desarrollo, consisten en la ejecución de actividades que le permitan verificar al desarrollador que los componentes unitarios están codificados bajo condiciones de robustez, esto es, soportando el ingreso de datos erróneos o inesperados y demostrando así la capacidad de tratar errores de manera controlada. Adicionalmente, las pruebas sobre componentes unitarios, suelen denominarse pruebas de módulos o pruebas de clases, siendo la convención definida (Miguel Alejandro dias Rivera, Rafael Alejandro Aguilera Rodrígues, 2020).
- Pruebas de Integración: este tipo de pruebas son ejecutas por el equipo de desarrollo y consisten en la comprobación de que elementos del software que interactúan entre sí, funcionan de manera correcta (Londoño, 2005).
- Pruebas de Sistema: este tipo de pruebas deben ser ejecutadas idealmente por un equipo de pruebas ajeno al equipo de desarrollo, una buena práctica en este punto corresponde a la tercerización de esta responsabilidad. La obligación de este equipo, consiste en la ejecución de actividades de prueba en donde se debe verificar que la funcionalidad total de un sistema fue implementada de acuerdo a los documentos de especificación definidos en el proyecto. Los casos de prueba a diseñar en este nivel de pruebas, deben cubrir los aspectos funcionales y no funcionales del sistema. Para el diseño de los casos de prueba en este nivel, el equipo debe utilizar como bases de prueba entregables tales como: requerimientos iniciales, casos de uso, HU, diseños, manuales técnicos y de usuario final (Londoño Abad, 2005).

- Pruebas de Aceptación: independientemente de que se haya tercerizado el proceso de pruebas y así la firma responsable de estas actividades haya emitido un certificado de calidad sobre el sistema objeto de prueba, es indispensable, que el cliente designe a personal que haga parte de los procesos de negocio para la ejecución de pruebas de aceptación, es incluso recomendable, que los usuarios finales que participen en este proceso, sean independientes al personal que apoyó el proceso de desarrollo (Londoño Abad, 2005).

Para el desarrollo de la solución propuesta se adopta como estrategia de prueba aplicar los cuatro niveles, realizando pruebas funcionales con las técnicas que estas incluyen. En el capítulo 3 se realiza una descripción más amplia de la estrategia de prueba definida por el autor de la presente investigación.

1.11 Conclusiones parciales

El estudio de los conceptos asociados al dominio del problema, propicia un mejor entendimiento del objeto de estudio de la presente investigación. Por otra parte, el uso de métodos científicos, tanto teóricos como empíricos fue fundamental para la profundización del contenido del presente trabajo. El estudio de las tendencias actuales en el desarrollo de soluciones para la generación de instrumentos de recolección de datos en el ámbito nacional e internacional, constituyó un importante punto de referencia para conocer en mayor detalle las características de estos sistemas. Este estudio también permitió, definir algunas funcionalidades para el desarrollo de la solución propuesta y demostrar la necesidad de desarrollar la presente investigación. Por otra parte, el uso de las herramientas y tecnologías definidas por el grupo de investigación al cual tributa la presente tesis, permite que el autor del trabajo disminuya la curva de aprendizaje para el trabajo con las mismas, al existir tutoriales y experiencia en el equipo que facilita el uso de estas.

CAPÍTULO 2: ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE LA SOLUCIÓN PROPUESTA

En el presente capítulo se elabora una propuesta de solución, teniendo en cuenta las fases de planificación, diseño y desarrollo, definidas en la metodología XP. Se presenta la estructura del software a través de los artefactos propuestos en las fases de la metodología. Se realiza la definición de los Requisitos Funcionales (RF) y los Requisitos No Funcionales (RnF), se describen los mismos a través de Historias de usuario (HU). Por otra parte, en el capítulo también se describe la arquitectura sobre la cual se implementa la aplicación web, así como las clases del diseño y lo patrones de diseño empleados. De igual forma se exponen los estándares de codificación utilizados en la implementación de la solución.

2.1 Descripción del negocio

Entre los diferentes instrumentos de recolección de datos se encuentran los formularios. Los formularios son documentos físicos o digital que sirven para recopilar información de manera estructurada, para después almacenarla y ser utilizada con un fin en específico. Estos están conformados por casillas, los datos a solicitar varían de acuerdo con su objetivo. Los formularios pueden considerarse también como un instrumento de trabajo cuyo propósito es trasportar información para simplificar y facilitar el desarrollo de los procedimientos administrativos.

Las aplicaciones más usuales de los formularios son para la gestión de empresas e instituciones públicas. También son ampliamente utilizados en entornos científicos con el objeto de almacenar la información experimental. Se le pueden llamar formulario a un documento donde hay un conjunto de objetos y procedimientos que se utilizan como diseño de presentación al usuario.

Para el desarrollo de un formulario se deben tener en cuenta los siguientes pasos:

- Definir los objetivos del estudio: definir la necesidad de realizar el estudio, el tipo de información que se desea obtener y determinar las acciones a llevar acabo con los resultados obtenidos.
- Obtener retroalimentación sobre el formulario: saber que tan fácil le resulte al encuestado proporcionar información sobre el tema solicitado. Esto permite conocer si tienen problemas para entender las preguntas. Si les resulta difícil se debe encontrar una mejor manera de obtener la información.
- Necesidad de cada pregunta a elaborar: todas las preguntas del formulario deben ser relevantes para obtener los conocimientos que se buscan.
- Especificidad de las preguntas: las preguntas deben explorar una idea a la vez para garantizar que los encuestados puedan comprender lo que se está preguntando. Las preguntas vagas, generales y de varias partes pueden ser confusas y difíciles de responder.
- Adaptar el tipo de pregunta a la información que se desea obtener: usar preguntas de texto abierto, de opción múltiple, orden de rango, escala o suma constante según corresponda.
- Realizar una prueba previa del formulario: de esta manera se obtiene una detallada retroalimentación.
- Diseñar visualmente de forma correcta formulario: este paso tiene un impacto directo en el éxito del estudio. Cuanto mejor sea el diseño visual, más fácil será para los encuestados completar el formulario.

Teniendo en cuenta el negocio antes descrito, a continuación, se describen los requisitos funcionales y no funcionales definidos para el desarrollo de la aplicación web para la generación de instrumentos de recolección de datos.

2.2 Fases del proceso de desarrollo

2.2.1 Planificación

El ciclo de vida de un proyecto realizado con la metodología XP inicia con la fase de planificación. En esta fase los clientes plantean a grandes rasgos las Historias de Usuarios (HU) y los programadores realizan una estimación del esfuerzo necesario

de cada una de ellas. Además, se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Se deben incluir varias iteraciones para lograr una entrega. El resultado de esta fase es un Plan de entregas.

Requisitos funcionales

Los RF de un sistema, son aquellos que describen cualquier actividad que este deba realizar, en otras palabras, el comportamiento o función particular de un sistema o software cuando se cumplen ciertas condiciones. Por lo general, estos deben incluir funciones desempeñadas por pantallas específicas, descripciones de los flujos de trabajo a ser desempeñados por el sistema y otros requerimientos de negocio (Pressman R. , 2010).

A partir del negocio antes descrito e identificar las necesidades del cliente, se definieron un total de 18 requisitos funcionales. En la tabla 2 se describen cada uno de los requisitos funcionales definidos.

Tabla 2. Descripción de los requisitos identificados. **Fuente:** elaboración propia.

No.	Nombre	Descripción
RF1	Adicionar usuario	Permite adicionar un usuario al sistema.
RF2	Modificar usuario	Permite modificar un usuario ya existente en el sistema.
RF3	Desactivar usuario	Permite desactivar un usuario ya activo en el sistema.
RF4	Activar usuario	Permite activar un usuario ya no activo en el sistema.
RF5	Buscar usuario	Permite buscar un usuario ya existente en el sistema.
RF6	Adicionar rol	Permite adicionar un rol al sistema.
RF7	Modificar rol	Permite modificar un rol ya existente en el sistema.

RF8	Desactivar rol	Permite desactivar un rol ya activo en el sistema.
RF9	Activar rol	Permite activar un rol ya no activo en el sistema.
RF10	Buscar rol	Permite buscar un rol ya existente en el sistema.
RF11	Adicionar formulario	Permite adicionar un nuevo formulario al sistema.
RF12	Modificar formulario	Permite modificar un formulario ya creado en el sistema.
RF13	Eliminar formulario	Permite eliminar un formulario ya creado en el sistema.
RF14	Visualizar formulario	Permite visualizar un formulario ya creado en el sistema.
RF15	Responder formulario	Permite introducir los datos solicitados en el formulario y enviar los mismos.
RF16	Consolidar resultados	Permite obtener de forma cuantitativa los resultados de los datos recogidos en cada formulario, en función del tipo de encuesta aplicada.
RF17	Exportar Excel	Permite visualizar en formato Excel los resultados cuantitativos de los datos recogidos en cada formulario, en función del tipo de encuesta aplicada.
RF18	Compartir Formulario	Permite compartir en las redes los formularios diseñados en el sistema.

Requisitos no Funcionales

Los requisitos no funcionales (RnF) representan características generales y restricciones de la aplicación o sistema que se esté desarrollando. Suelen presentar dificultades en su definición dado que su conformidad o no conformidad podría ser sujeto de libre interpretación, por lo cual es recomendable acompañar su definición

con criterios de aceptación que se puedan medir (Pressman R. , 2010). A continuación, se listan los requisitos no funcionales definidos para el desarrollo de la solución propuesta:

- **Usabilidad**

RnF 1. Requisito de usabilidad. Se debe garantizar que la curva de aprendizaje del sistema sea baja. El usuario debe alcanzar su máxima productividad en el menor tiempo posible desde la primera vez de uso.

RnF 2. Requisito de usabilidad. En el sistema se deben visualizar todos los mensajes en idioma español.

RnF 3. Requisito de usabilidad. El sistema debe facilitar la interacción con el usuario. Se mostrarán mensajes al cancelar alguna operación relacionada con el registro o la modificación. Se alertará ante la posible ejecución de operaciones críticas.

RnF 4. Requisito de usabilidad. El sistema tendrá una interfaz simple. La tipografía utilizada debe ser uniforme, de un tamaño adecuado.

- **Rendimiento**

RnF 5. Requisito de rendimiento. Se debe garantizar tiempos de respuesta apropiados para la aplicación. Para ello se define como intervalo de tiempo de respuesta de 1 a 5 segundos, siendo este último tiempo el permitido a tareas que realizan consultas a la base de datos.

- **Soporte**

RnF 6. Requisito de soporte. Garantizar un sistema con capacidad de mantenimiento, reparable y escalable.

- **Seguridad**

RnF 7. Requisito de seguridad. Restringir el acceso al sistema a través de la gestión de usuarios y delimitar el acceso de cada usuario con permisos específicos.

RnF 8. Requisito de seguridad. Los niveles de acceso deben ser definidos por el administrador del sistema, los permisos serán basados en los roles definidos y el usuario no podrá gestionarlos.

RnF 9. Requisito de seguridad. Se utilizará un modelo de autorización basado en roles, a los que se les asignará permisos específicos y algunos adicionales en caso de ser necesario.

RnF 10. Requisito de seguridad. El sistema incluirá un módulo de gestión de usuarios y roles.

Historias de usuario

Las HU son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las HU es muy dinámico y flexible, en cualquier momento las HU pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas *(Letelier, et al., 2006).*

Para el desarrollo de la solución propuesta se diseñaron 18 HU. A continuación, en la tabla 3 se describe la HU “Adicionar formulario”, de prioridad alta, teniendo en cuenta el impacto que tiene en el desarrollo de la solución propuesta.

Tabla 3. Historia de usuario “Adicionar formulario”. **Fuente:** elaboración propia.

Historias de Usuario	
Número: 11	**Nombre de la HU:** Adicionar formulario
Programador: Marcelo Eduardo Exposito Blanco	**Iteración asignada:** 2da
Prioridad: Alta	**Tiempo estimado:** 32 horas
Riesgo en desarrollo:	**Tiempo real:** 32 horas
Descripción: permite al usuario registrar un nuevo formulario en el sistema, llenando los campos que muestra la interfaz. Todos los campos son obligatorios. A continuación, se describen cada uno de estos campos:	

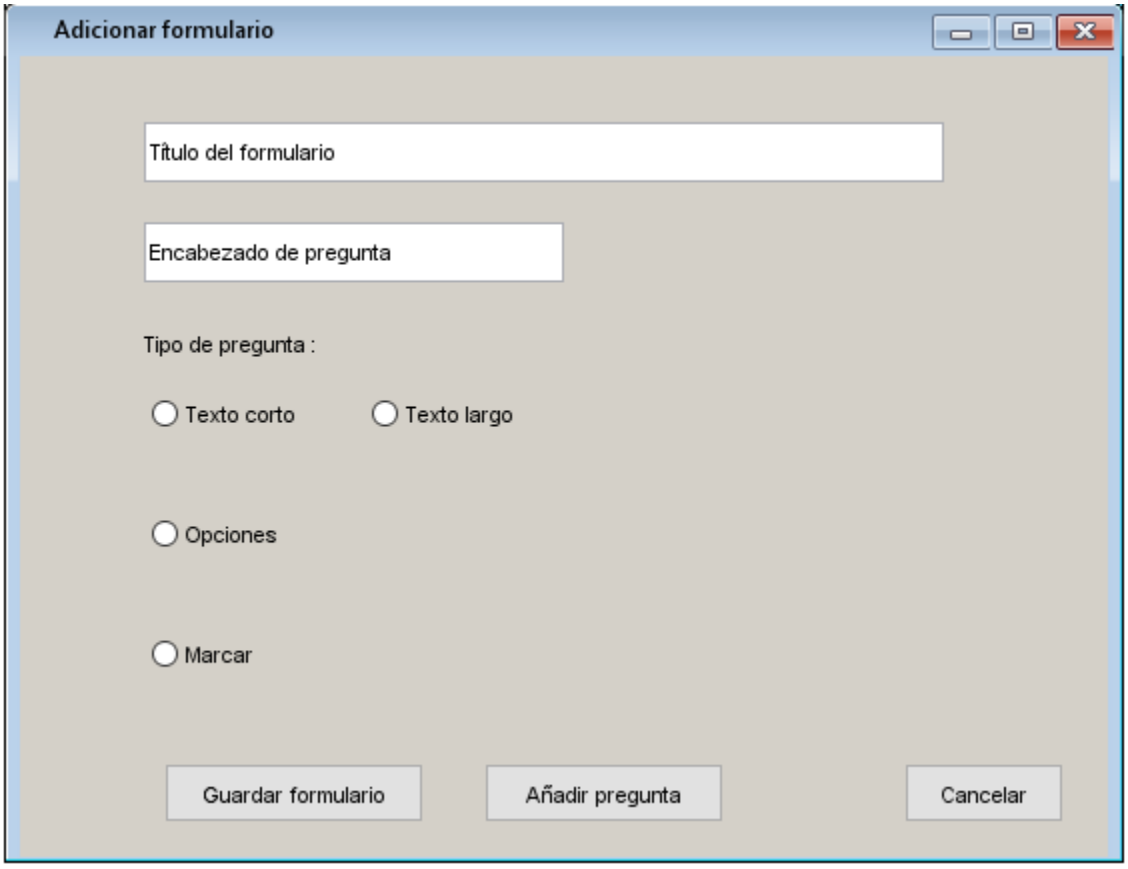
- Título del formulario: campo de tipo texto
- Encabezado de pregunta: campo de tipo texto
- Tipo de Pregunta: campo de selección

Una vez introducidos los datos el usuario selecciona la opción Guardar para guardar los datos, y el sistema muestra un mensaje de éxito al Adicionar el formulario.

La opción Cancelar permite al usuario cerrar la interfaz: Adicionar formulario.

Observaciones:

Prototipo de interfaz:



Estimación de esfuerzo por historia de usuario

Para la realización de la herramienta informática propuesta se efectuó una estimación de esfuerzo de cada una de las HU identificadas, arrojando los siguientes resultados:

Tabla 4. Estimación de esfuerzo por HU.
Fuente: elaboración propia.

Historias de usuario	Puntos de estimación (horas)
Adicionar usuario	25
Modificar usuario	25
Desactivar usuario	25
Activar usuario	26
Buscar usuario	20
Adicionar rol	25
Modificar rol	25
Desactivar rol	25
Activar rol	20
Buscar rol	20
Adicionar formulario	42
Modificar formulario	42
Eliminar formulario	42
Visualizar formulario	42

Responder formulario	42
Consolidar resultados	46
Exportar Excel	52
Compartir Formulario	32

Plan de iteraciones

Este plan incluye varias iteraciones sobre la solución antes de ser entregada. Además, define exactamente qué HU serán implementadas en cada iteración. Tomando como base cada una de las HU identificadas y el esfuerzo que se requiere para la realización de cada una de estas, se procede entonces a la realización de varias iteraciones para el desarrollo de la herramienta propuesta.

A continuación, se describen cada una de las iteraciones propuestas, donde la duración total de iteraciones en días, se obtiene a partir del esfuerzo en días estimado por el desarrollador para implementar cada HU:

Iteración #1: en esta iteración se implementaron las HU de la 1 a la 10, las cuales se refieren a la gestión de usuario y la gestión de roles.

Iteración #2: en esta iteración se implementaron las HU de la 11 a la 18, la cual tiene en cuenta la gestión de formularios, también se le adicionaron las funcionalidades Visualizar formulario, Responder formularios, Consolidar resultados, Exportar resultados a un documento Exel y Compartir en las redes sociales. Por otra parte, se corrigen los errores encontrados en la iteración anterior, obteniéndose una nueva versión de la solución.

A modo de resumen se presenta una tabla que muestra la duración de cada iteración, así como el orden en que serán implementadas las HU en cada una de ellas según la prioridad asignada por el cliente:

Tabla 5. Plan de duración de las iteraciones.
Fuente: elaboración propia.

Iteraciones	Historias de usuario	Duración
1	Adicionar usuario	236 horas
	Modificar usuario	
	Desactivar usuario	
	Activar usuario	
	Buscar usuario	
	Adicionar rol	
	Modificar rol	
	Desactivar rol	
	Activar rol	
	Buscar rol	
2	Adicionar formulario	340 horas
	Modificar formulario	
	Eliminar formulario	

Visualizar formulario
Responder formulario
Consolidar resultados
Exportar Excel
Compartir Formulario

Plan de entregas

Una vez identificadas por el cliente las HU y confeccionado por los desarrolladores el plan de iteración, se procede a la confección del plan de entrega, con la intención de que los clientes obtengan una estimación real del tiempo que conllevará la implementación de las funcionalidades que describe cada HU. Seguidamente se muestra el plan de entrega elaborado por el equipo de desarrollo para la fase de implementación:

Tabla 6. Plan de entregas.
Fuente: elaboración propia.

Iteración	Fecha de entrega
1	11/07/2022
2	19/10/2022

2.2.2 Diseño

La metodología de desarrollo de software XP propone que la implementación debe ejecutarse de forma iterativa e incremental, alcanzando al concluir cada iteración un producto funcional que debe ser examinado y mostrado al cliente. De esta forma se garantiza una constante retroalimentación entre los desarrolladores y clientes, posibilitando que los desarrolladores puedan ganar en claridad de lo que debe hacer el producto, apoyándose en la visión de los usuarios finales. A la hora de darle

cumplimiento a la actividad de diseñar, XP no especifica ninguna técnica de modelado, o sea, pueden utilizarse sencillos esquemas en pizarras, diagramas de clases usando UML o tarjetas CRC (Navia López & Vivero Díaz, 2014).

Arquitectura

La arquitectura de software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución *(IEEE, 2000)*.

La solución propuesta sigue una arquitectura de tipo Modelo Vista Controlador. Teniendo en cuenta que el marco de trabajo utilizada en la investigación es Django, la arquitectura antes mencionada es versionada a Modelo Vista Plantilla. Este estilo arquitectónico separa los datos de la aplicación en tres módulos, lo cuales se relacionan a continuación:

- Modelo: contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.

- Vista: compone la información que se envía al cliente y los mecanismos de interacción con éste.

- Plantilla (*templates*): decide como será mostrada la información, esta contiene las decisiones relacionadas a la presentación.

En el caso de Django el módulo Controlador está integrado al principio de funcionamiento del marco de trabajo. En la figura 1 se representa la arquitectura antes descrita en los componentes de la solución propuesta.

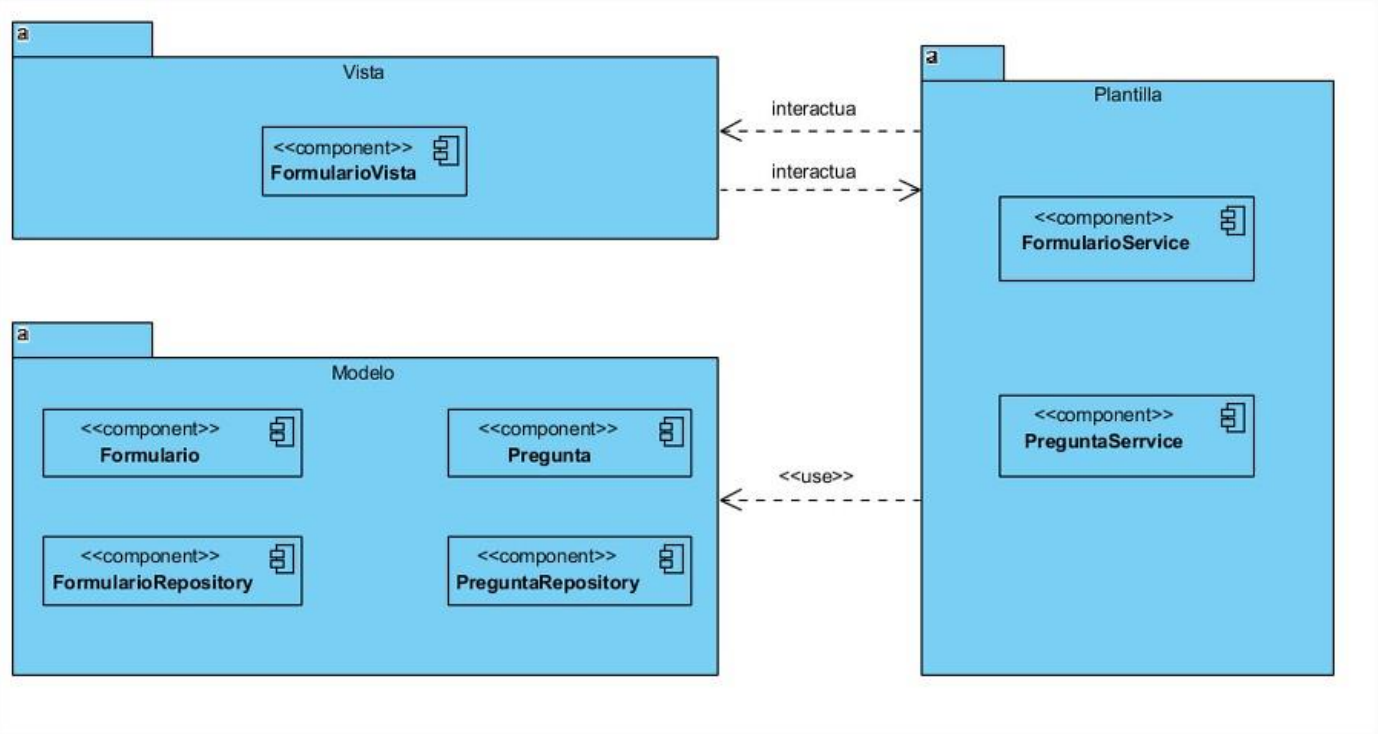


Figura 1. Representación del estilo arquitectónico Modelo Vista Plantilla.
Fuente: elaboración propia

Diagrama de clases del diseño

Un diagrama de clases de diseño muestra la especificación para las clases de software de una aplicación. Incluye clases, asociaciones y atributos, interfaces con sus operaciones y constantes, métodos, navegabilidad y dependencias. El diagrama de clases de diseño muestra definiciones de entidades software más que conceptos del mundo real. El diagrama de clases recoge las clases de objetos y sus asociaciones (UNAD, 2020).

En la figura 2, se muestra el diagrama de clases del diseño para la funcionalidad Adicionar formulario de la solución propuesta. En el mismo se evidencian los atributos, métodos y relaciones entre las clases.

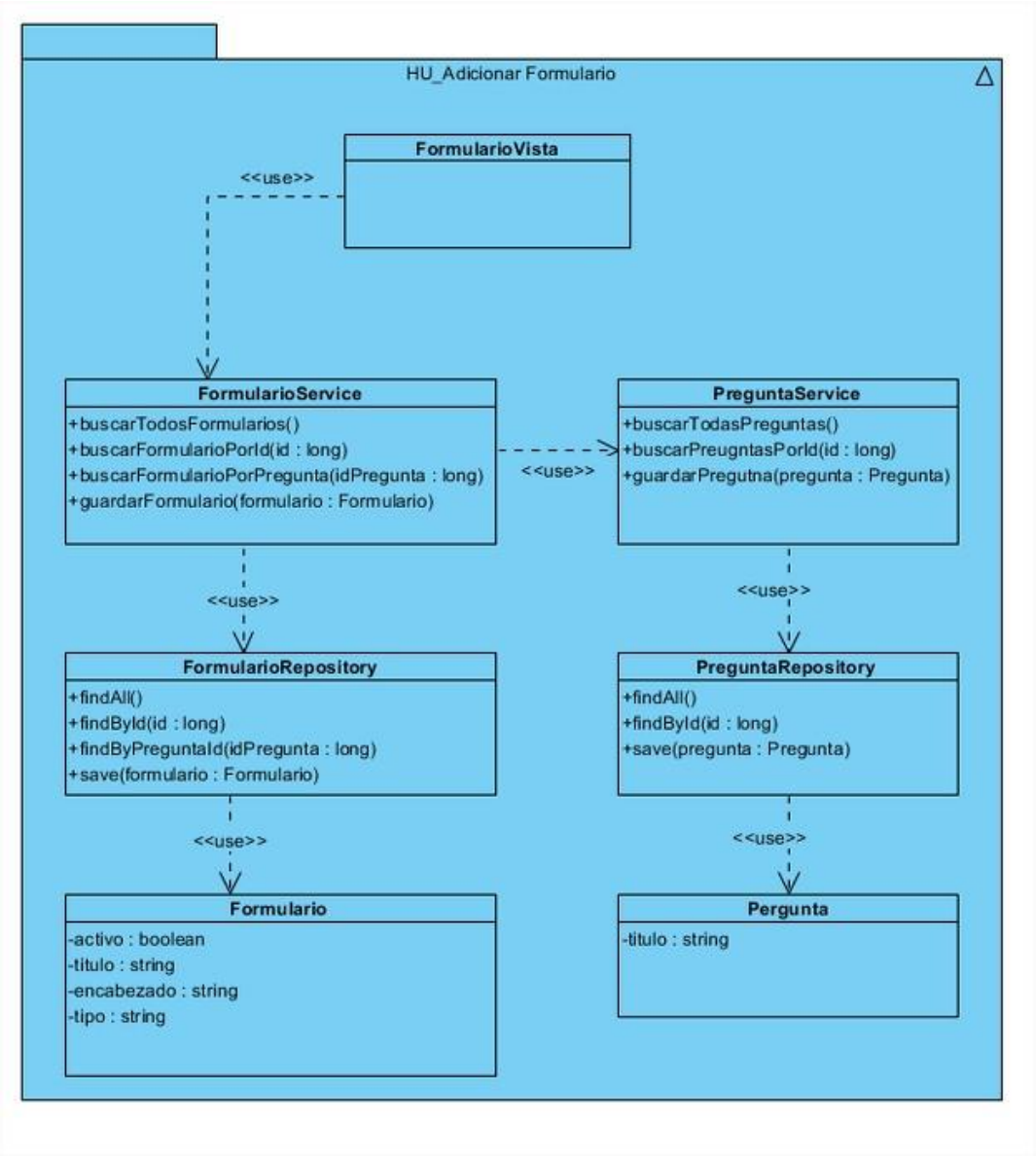


Figura 2. Diagrama de clases del diseño**.**
Fuente: elaboración propia

Patrones de diseño

Para diseñar la solución propuesta se emplearon un conjunto de patrones de diseño que constituyen soluciones simples a problemas específicos y comunes del diseño orientado a objetos. A continuación, se describen los patrones empleados:

Patrón bajo acoplamiento

Este patrón plantea la baja dependencia que debe existir entre las clases y está estrechamente relacionado con los patrones Experto o Alta Cohesión (Larman C. , 2016).

En la presente investigación el uso de este patrón se evidencia en el 80 % de las clases del diseño, las cuales tienen una baja dependencia una de otras, tal y como se demuestra en el Capítulo 3 con los resultados de la validación del diseño a partir de la aplicación de la métrica Relaciones entre Clases.

Patrón alta cohesión

Este patrón permite asignar responsabilidades de manera que la cohesión siga siendo alta (Larman C. , 2016).

En la presente investigación el uso de este patrón se evidencia en el 80 % de las clases del diseño, las cuales tienen una baja sobrecarga de responsabilidades, tal y como se demuestra en el Capítulo 3 con los resultados de la validación del diseño a partir de la aplicación de la métrica Tamaño Operacional de Clases.

Modelo de datos

El modelo de datos se utiliza para describir la estructura lógica y física de la información persistente gestionada por el sistema. Se utiliza para definir la correlación entre las clases de diseño persistentes y las estructuras de datos persistentes, y para definir las estructuras de datos persistentes (Lucid_Software_Inc, 2019). A continuación, se muestra el modelo de datos de la solución propuesta.

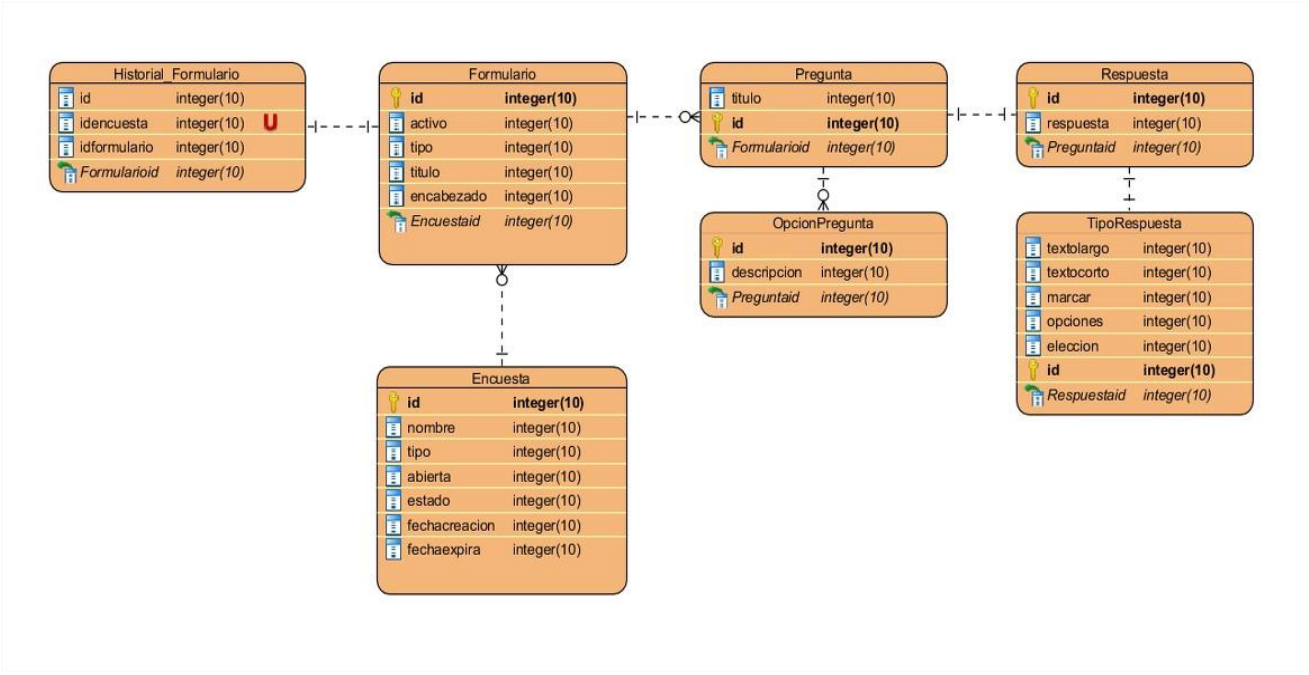


Figura 3. Modelo de datos de la solución propuesta.
Figura: elaboración propia.

2.2.3 Desarrollo

El desarrollo de la solución es la parte más importante dentro del desarrollo del software en la metodología XP. En esta fase las HU se descomponen en tareas de programación o ingeniería, que a su vez son convertidas en código. Además, propone una serie de prácticas que sirven para el desarrollo exitoso que se desee realizar. Se tiene en cuenta el desarrollo de las iteraciones según el plan de iteraciones definido, la entrega de la aplicación web en iteraciones pequeñas, un diseño simple y poco redundante del código con las funcionalidades necesarias.

Tareas de ingeniería

Una vez identificadas las HU, los programadores proceden a descomponer cada una en tareas específicas, las denominadas tareas de la programación que están escritas técnicamente y que darán solución a la HU correspondiente. Las tareas de ingeniería permiten a los desarrolladores obtener un nivel de detalle más avanzado por las HU. A continuación, se describe la Tarea de ingeniería 11, que tiene como objetivo implementar la HU Adicionar formulario:

Tabla 7. Tarea ingenieril para la HU # 11.
Fuente: elaboración propia.

Tarea de ingeniería	
Número de tarea: 11	**Historia de Usuario (No.11):** Adicionar formulario
Nombre de tarea: Implementar HU_ Adicionar formulario	
Tipo de tarea: Desarrollo	**Puntos estimados:** 5 días
Fecha de inicio: 12-07-2022	**Fecha de fin:** 16-07-2022
Programador responsable: Marcelo Eduardo Exposito Blanco	
Descripción: permite al usuario registrar un nuevo formulario en el sistema, llenando los campos que muestra la interfaz.	

Estándares de codificación

Los estándares de codificación constituyen buenas prácticas o conjunto de reglas no formales que han ido surgiendo en las comunidades de desarrolladores de software con el paso del tiempo. Estos tienen el propósito de obtener un código fuente más legible, portable, seguro, eficiente, robusto y cohesionado, permitiendo mayor velocidad en el desarrollo, mejor coordinación entre los equipos de trabajo. Además, logran que para un desarrollador sea más fácil integrarse a un proyecto ya comenzado, se eviten bugs[4] y se faciliten los procesos de mantenibilidad y revisión de código (Hommel S., 2019). Este grupo de buenas prácticas pueden ser definidas en dependencia del lenguaje o marco de trabajo a utilizar. A continuación, se describen los estándares de codificación utilizados en la solución propuesta:

- Los nombres de las clases comenzarán con mayúscula, en caso de ser un nombre compuesto las siguientes palabras se escribirán de igual forma. En la figura 4 se evidencia el uso de este estándar en la clase Formulario.

[4] Error de software que desencadena un resultado indeseado.

```
class Formulario (models.Model) :
    formId= 0
    titulo_form= models.CharField(max_length=200,verbose_name="Titulo del Formulario")
    encabezado= models.CharField(max_length=200,verbose_name="Encabezado de Formulario")
```

Figura 4. Estándar de codificación utilizado en el nombre de las clases.
Fuente: elaboración propia.

- Los identificadores para las variables y los parámetros se escribirán con letras en minúsculas y en caso de ser un nombre compuesto las siguientes palabras se escribirán en mayúscula. En la figura 4 se evidencia el uso de este estándar de codificación en definición de la variable *formId*.

- Tanto los nombres de los métodos como de las variables deben ser suficientemente descriptivos en cuanto a su función, y no deben exceder los 15 caracteres (ver figura 4).

2.3 Conclusiones parciales

El empleo de la metodología XP, permitió organizar el desarrollo de la solución y generar los productos de trabajo necesarios correspondientes a las fases de Planificación, Diseño y Desarrollo. La arquitectura diseñada a través del estilo arquitectónico Modelo-Vista-Plantilla, facilitó el establecimiento de una guía para la implementación de la solución. Los patrones de diseño y los estándares de codificación aplicados, permitieron establecer las pautas para la implementación, obteniéndose una solución con poca dependencia entre clases flexible al mantenimiento y a la introducción de cambios.

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

En el presente capítulo se muestran los resultados obtenidos luego de aplicar las técnicas de validación tanto al diseño del sistema como a la solución desarrollada. Documentándose los resultados alcanzados en cada caso. Para la validación del diseño se aplican métricas y para la validación de la solución se define una estrategia de prueba aplicando los niveles de unidad, integración, sistema y aceptación, con sus respectivos métodos y técnicas. Por otra parte, se realiza una verificación del cumplimiento del objetivo general de la investigación.

3.1. Validación de los requisitos

Técnicas de validación de requisitos

Con el objetivo de ratificar que los requisitos del software obtenidos definen el sistema que el cliente desea se llevó a cabo un proceso de validación de los mismos, para el cual se emplearon las siguientes técnicas:

Revisiones de los requisitos: se realizaron revisiones a cada uno de los requisitos por parte del equipo de desarrollo. Las revisiones internas generaron un total de 5 no conformidades de tipo técnicas y de ortografía, las cuales fueron corregidas satisfactoriamente.

Construcción de prototipos: se mostró al cliente un modelo ejecutable que permitió tener una visión preliminar de cómo se verían estos componentes en el sistema y a través de la interacción con los mismos se comprobó la satisfacción y aprobación del cliente, los mismos fueron aprobados satisfactoriamente.

Métricas aplicadas a los requisitos

Con el objetivo de medir la calidad de la especificación de los requisitos se aplicó la métrica Calidad de la especificación (CE).

Para obtener cuán entendibles y precisos son los requisitos, primeramente, se calcula el total de requisitos de la especificación como se muestra a continuación:

Nr: el total de requisitos de especificación.

Nf: cantidad de requisitos funcionales.

Nnf: cantidad de requisitos no funcionales.

$$\mathbf{Nr = Nf + Nnf}$$

Como resultado de la sustitución de los valores para el módulo de administración de la suite se obtiene:

$$Nr = 18 + 10$$

$$Nr = 28$$

Para determinar, finalmente, la Especificidad de los Requisitos (ER) o ausencia de ambigüedad en los mismos se realiza la siguiente operación:

$$\mathbf{ER = Nui / Nr}$$

Donde *Nui* es el número de requisitos para los cuales todos los revisores tuvieron interpretaciones idénticas. Mientras más cerca de 1 esté el valor de ER, menor será la ambigüedad.

Para el caso de los requisitos obtenidos para la aplicación web para la generación de instrumentos de recolección de datos, 3 requisitos produjeron contradicciones en las interpretaciones. Sustituyendo las variables se obtiene:

$$ER = 25/28$$

$$ER = 0.89$$

Arrojando un resultado final satisfactorio, indicando que el grado de ambigüedad de los requisitos es bajo (11%) ya que el 89% son entendibles. Los requisitos ambiguos fueron modificados y validados para garantizar una correcta interpretación.

3.2. Validación del diseño

Con el objetivo de comprobar la calidad del diseño se emplearon las métricas de diseño relaciones entre clases (RC) y tamaño operacional de clase (TOC).

Relaciones entre clases (RC)

Permite evaluar el acoplamiento, la complejidad de mantenimiento, la reutilización y la cantidad de pruebas de unidad necesarias para probar una clase teniendo en cuenta las relaciones existentes entre ellas (Pressman R. S., 2010)

La métrica RC está dada por el número de relaciones de uso de una clase con otra. El primer paso es evaluar un conjunto de atributos de calidad entre los que se encuentran el acoplamiento, complejidad de mantenimiento y reutilización de cada clase (Pressman R. S., Ingeniería del Software. Un enfoque práctico. Quinta Edición, 2002).

A continuación, se exponen los pasos que se llevaron a cabo para aplicar la métrica a todas las clases de la aplicación web propuesta en la presente investigación:

- Determinar la Cantidad de Relaciones de Uso (CRU) que poseen las clases a medir.
- Calcular el promedio de las CRU.
- Teniendo en cuenta los valores antes obtenidos se determina la incidencia de los atributos de calidad en cada una de las clases según los criterios expuestos en la siguiente tabla.

Tabla 8. Rango de valores para medir la afectación de los atributos de calidad de la métrica RC. **Fuente:** (Zapata, 2022).

Atributos de calidad	Clasificación	Criterio
Acoplamiento	Ninguna	CRU = 0
	Baja	CRU = 1
	Media	CRU = 2
	Alta	CRU > 2

Complejidad de mantenimiento	Baja	CRU = Promedio
	Media	Promedio < CRU < = 2* promedio
	Alta	CRU > 2* promedio
Reutilización	Baja	CRU > 2* promedio
	Media	Promedio < CRU < = 2* promedio
	Alta	CRU <= Promedio
Cantidad de pruebas	Baja	CRU <= Promedio
	Media	Promedio < CRU < = 2* promedio
	Alta	CRU > 2* promedio

En las figuras 5, 6, 7 y 8 se muestran los resultados de aplicar la métrica RC:

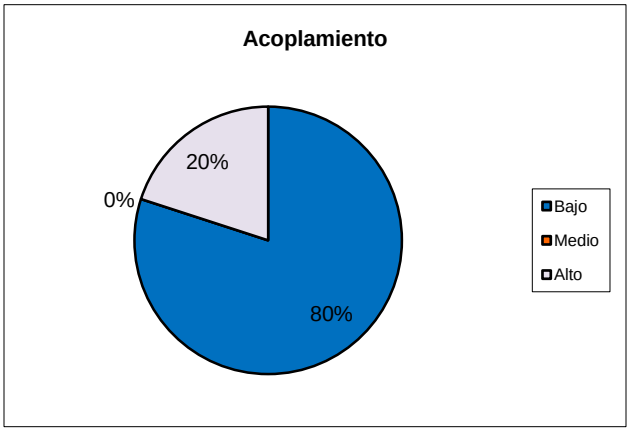


Figura 5. Representación en (%) de los resultados de la aplicación de la técnica RC al atributo de calidad de acoplamiento
Fuente: elaboración propia

Acoplamiento: según los resultados que se muestran, el 80 % posee un acoplamiento bajo y el 20% un acoplamiento alto, por lo que la mayoría de las clases poseen valores de acoplamiento bajos, lo que demuestra un correcto diseño de las clases en relación al atributo evaluado.

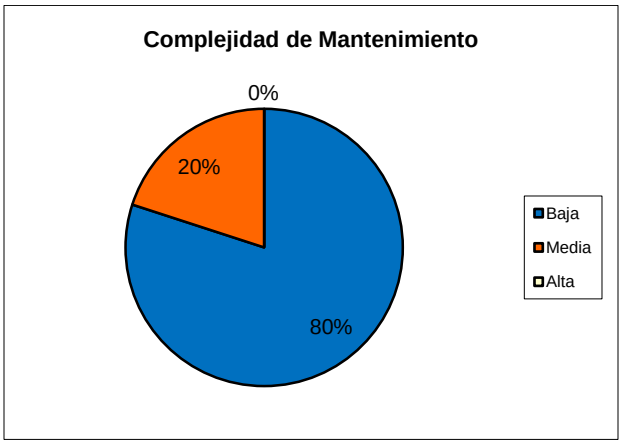


Figura 6. Representación en (%) de los resultados de la aplicación de la técnica RC al atributo de calidad de complejidad de mantenimiento.
Fuente: elaboración propia

Complejidad de mantenimiento: según los resultados que se muestran, el 80 % posee una complejidad de mantenimiento bajo y el 20% una complejidad de mantenimiento media, por lo que la mayoría de las clases poseen valores de complejidad de mantenimiento bajos, lo que demuestra un correcto diseño de las clases en relación al atributo evaluado.

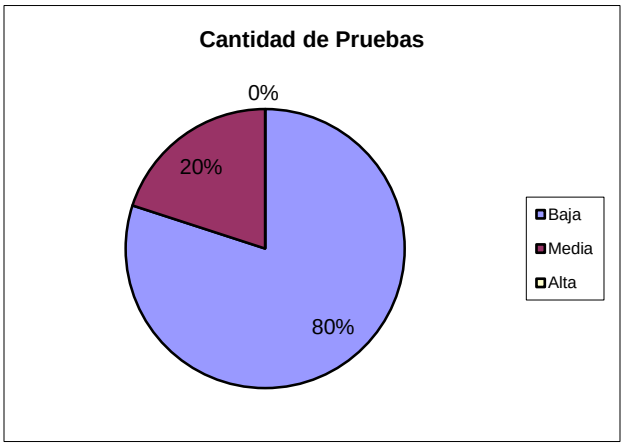


Figura 7. Representación en (%) de los resultados de la aplicación de la técnica RC al atributo de calidad de cantidad de pruebas.
Fuente: elaboración propia

Cantidad de pruebas: según los resultados que se muestran, el 80 % de las clases requieren de un bajo de esfuerzo a la hora de realizar cambios, rectificaciones y pruebas de software, mientras el 20 % requieren de un esfuerzo medio. Por lo que la mayoría de las clases requieren de un bajo de esfuerzo a la hora de realizar

cambios, rectificaciones y pruebas de software, lo que demuestra un correcto diseño de las clases en relación al atributo evaluado.

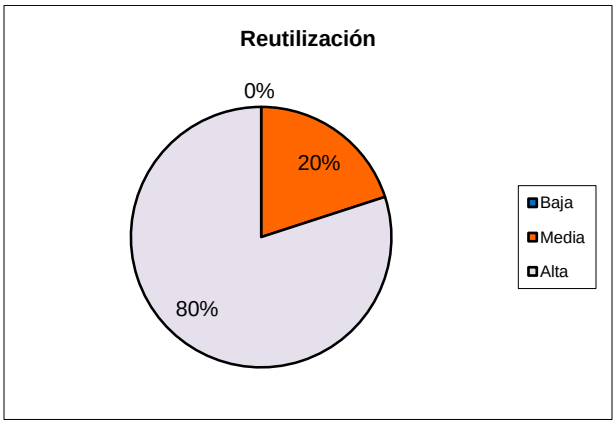


Figura 8. Representación en (%) de los resultados de la aplicación de la técnica RC al atributo de calidad de reutilización.
Fuente: elaboración propia

Reutilización: según los resultados que se muestran, el 80 % posee una reutilización alta y el 20% una reutilización media, por lo que la mayoría de las clases poseen valores de reutilización altos, lo que demuestra un correcto diseño de las clases en relación al atributo evaluado.

Teniendo en cuenta lo anterior se concluye que el diseño de la aplicación web para la generación de instrumentos de recolección de datos, propuesta en la presente investigación, alcanzó resultados positivos durante la evaluación de la métrica RC. Estos datos demuestran que las clases presentan un acoplamiento, una complejidad de mantenimiento y un esfuerzo para realizar pruebas bajo y en efecto presentan un alto grado de reutilización.

Tamaño Operacional de Clases (TOC)

Permite medir la responsabilidad, la complejidad de implementación y la reutilización de las clases del diseño. Es importante destacar que, para esta métrica, la responsabilidad y la complejidad son inversamente proporcionales a la reutilización, por lo que, a mayor responsabilidad y complejidad de implementación de una clase, menor es su nivel de reutilización (Pressman R. S., 2010).

La métrica TOC fue aplicada a cada una de las clases del diseño con el objetivo de medir la calidad de las mismas con respecto a su grado de responsabilidad, complejidad de implementación y reutilización (Pressman R. S., 2002).
A continuación, se explican los pasos que se llevaron a cabo para aplicar la métrica:

- Cálculo del umbral. El umbral se toma del tamaño general de una clase que se determina sumando todas las operaciones que posee.
- Calcular el promedio de los umbrales.

Teniendo en cuenta los valores antes obtenidos se determina la incidencia de los atributos de calidad en cada una de las clases, según los criterios expuestos en la tabla 9.

Tabla 9. Rango de valores para medir la afectación de los atributos de calidad de la métrica TOC. **Fuente:** (Zapata, 2022).

Atributos de calidad	Clasificación	Criterio
Responsabilidad	Baja	Umbral<= Promedio
	Media	Promedio < Umbral < = 2* promedio
	Alta	Umbral > 2* promedio
Complejidad de implementación	Baja	Umbral <= Promedio
	Media	Promedio < Umbral < = 2* promedio
	Alta	Umbral > 2* promedio
Reutilización	Baja	Umbral > 2* promedio
	Media	Promedio < Umbral < = 2* promedio
	Alta	Umbral <= Promedio

En las figuras 9, 10 y 11 se muestran los resultados de aplicar la métrica TOC:

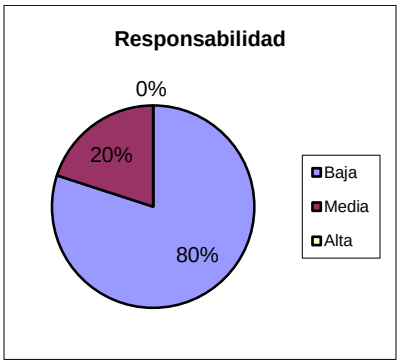


Figura 9. Representación en (%) de los resultados de la aplicación de la técnica TOC al atributo de calidad de responsabilidad.
Fuente: elaboración propia

Responsabilidad: según los resultados que se muestran, el 80 % posee una responsabilidad baja y el 20% una responsabilidad media, por lo que la mayoría de las clases poseen valores de responsabilidad bajos, lo que demuestra un correcto diseño de las clases en relación al atributo evaluado.

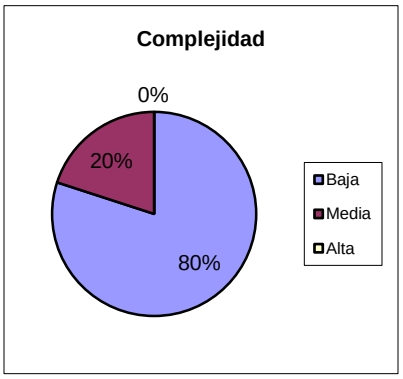


Figura 10. Representación en (%) de los resultados de la aplicación de la técnica TOC al atributo de calidad de complejidad.
Fuente: elaboración propia

Complejidad: según los resultados que se muestran, el 80 % posee una complejidad baja y el 20% una complejidad media, por lo que la mayoría de las clases poseen valores de complejidad bajos, lo que demuestra un correcto diseño de las clases en relación al atributo evaluado.

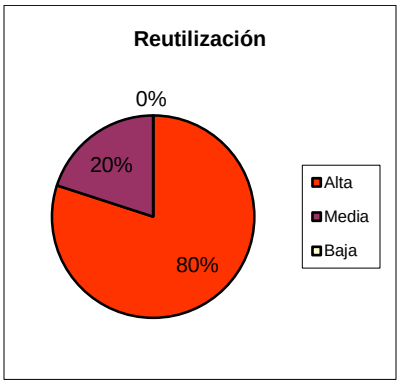


Figura 11. Representación en (%) de los resultados de la aplicación de la técnica TOC al atributo de calidad de reutilización.
Fuente: elaboración propia

Reutilización: según los resultados que se muestran, el 80 % posee una reutilización alta y el 20% una reutilización media, por lo que la mayoría de las clases poseen valores de reutilización altos, lo que demuestra un correcto diseño de las clases en relación al atributo evaluado.

Teniendo en cuenta lo anterior se concluye que el diseño de la aplicación web para la generación de instrumentos de recolección de datos, propuesta en la presente investigación, alcanzó resultados positivos durante la evaluación de la métrica TOC. Estos datos demuestran que las clases presentan una responsabilidad y complejidad baja, en efecto presentan un alto grado de reutilización.

3.3. Pruebas de software

Las pruebas de software son un conjunto de actividades que pueden ser planificadas con antelación y ejecutarse sistemáticamente durante la implementación o al finalizar el desarrollo del software. Estas se realizan para identificar posibles fallos de funcionamiento, configuración o usabilidad de un programa o aplicación. Las pruebas de software se ejecutan a partir de la aplicación de métodos y técnicas. La organización del proceso de pruebas se realiza a través de cuatro niveles: unidad, integración, sistema y aceptación (Pressman, 2010).

Para la validación la solución propuesta en la presente investigación se define una estrategia de prueba de software a nivel de unidad, integración, sistema y aceptación, con la aplicación de sus respectivos métodos y técnicas.

Pruebas a nivel de unidad

Las pruebas a nivel de unidad se concentran en el esfuerzo de verificación de la unidad más pequeña del diseño: el componente o módulo de software. Tomando como guía la descripción del diseño a nivel de componente, se prueban importantes caminos de control para describir errores dentro de los límites del módulo. El alcance restringido que se ha determinado para las pruebas de unidad limita la relativa complejidad de las pruebas y los errores que éstas descubren (Pressman, 2010). En el caso de la presente tesis en este nivel de prueba se aplicaron los métodos de caja blanca y caja negra.

Método de caja blanca

El método de caja blanca posibilita el desarrollo de casos de prueba que garanticen la ejecución, al menos una vez, de las rutas independientes (Pressman, 2010). En el caso de la presente tesis el método fue ejecutado aplicando la técnica de ruta básica.

La técnica de ruta básica tiene como objetivo comprobar que cada ruta se ejecute independiente de un componente o programa, obteniéndose una medida de la complejidad lógica del diseño. Esta técnica debe ser utilizada para evaluar la efectividad de los métodos asociados a una clase, con el objetivo de asegurar que cada ruta independiente sea ejecutada por lo menos una vez en el sistema (Pressman, 2010).

En la validación de la solución propuesta la técnica de ruta básica se aplicó a todos los métodos de las clases controladoras, teniendo en cuenta que estos agrupan las principales funcionalidades de la solución. A continuación, se describe la aplicación del método de caja blanca sobre la funcionalidad *questionForm.*

```
from forms import questionForm
                                    } 1
def add_question(request):
    if request.method == 'POST':} 2
        form = QuestionForm(request.POST) } 3
        if form.is_valid(): } 4
            form.save()
            return HttpResponseRedirect('/add_question/thanks/') } 5
    else:
        form = QuestionForm() } 6
    return render_to_response('books/add_question.html', {'form': form}) } 7
      } 8
```

Figura 12. Código sobre el cual se aplica la técnica de camino básico.
Fuente: elaboración propia

Una vez definido el código sobre el cual se aplica el método, los pasos a seguir para desarrollar la técnica de ruta básica son los siguientes:

1) Confeccionar el grafo de flujo: esta muestra el flujo de control lógico (Pressman, 2010). Está compuesto por los siguientes elementos:

- Nodos: son círculos que representan una o más sentencias procedimentales.
- Aristas: son flechas que representan el flujo de control y son análogas a las flechas del diagrama de flujo.
- Regiones: son las áreas delimitadas por aristas y nodos.

En la siguiente figura se muestra el grafo de flujo obtenido:

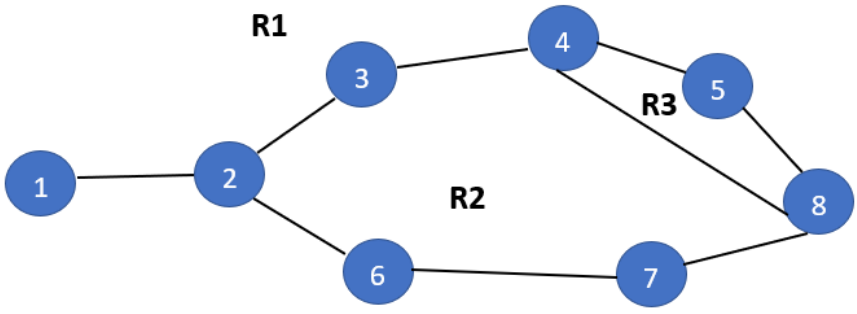


Figura 13. Grafo de la ruta básica de la funcionalidad *questionForm.*
Fuente: elaboración propia

2) Calcular la complejidad ciclomática:

El valor calculado por la complejidad ciclomática define el número de rutas independientes del conjunto básico de un programa y brinda una cota superior para el número de pruebas que se deben realizar a fin de asegurar que todos los enunciados se ejecutaron al menos una vez (Pressman, 2010).

La complejidad ciclomática se calcula de tres formas diferentes, las cuales deben llegar al mismo resultado para comprobar que el cálculo es el correcto (Pressman, 2010).

- El número de regiones del grafo de flujo corresponde a la complejidad ciclomática.
- La complejidad ciclomática V(G) para un grafo de flujo G se define como:
  V(G)=E-N+2
  Donde E es el número de aristas del gráfico de flujo y N el número de nodos del gráfico de flujo.
- La complejidad ciclomática V(G) para un grafo de flujo G también se define como
  V(G)=P+1
  Donde P es el número de nodos predicado (nodos de donde parten al menos dos aristas) contenidos en el grafo de flujo G.

En el grafo de flujo de la figura 13, la complejidad ciclomática puede calcularse usando cada una de las vías anteriormente descritas:

1. El grafo de flujo tiene 3 regiones, por tanto, V(G) = 3
2. V(G) = (9 aristas – 8 nodos) +2 = 3
3. V(G) = 2 nodos predicado + 1= 3

Por tanto, la complejidad ciclomática del grafo de flujo de la figura 13 es 3.

3) Determinar un conjunto básico de rutas linealmente independientes:

El valor de V(G) proporciona la cota superior sobre el número de rutas linealmente independientes a través de la estructura de control del programa (Pressman, 2010). En el caso de la funcionalidad *questionForm*, se definen 3 rutas básicas:

Ruta básica # 1: 1, 2, 3, 4, 5, 8
Ruta básica # 2: 1, 2, 3, 4, 8
Ruta básica # 3: 1, 2, 6, 7, 8

4) Obtención de casos de prueba (CP):

Una vez definidas las rutas, se procede a diseñar los casos de prueba para cada una de las rutas básicas obtenidas. A continuación, en la tabla 10 se presenta el caso de prueba definido para la ruta básica # 3.

Tabla 10. Caso de prueba de caja blanca para el camino básico #3.
Fuente: elaboración propia

Caso de prueba Camino Básico #3	
Descripción: este camino se ejecuta cuando la condicional que verifica si el *request.method* es diferente al POST.	
Entrada	ninguna
Resultado	Renderiza el visual del método *add_question*
Condiciones	Si el *request.method* es diferente al POST

Una vez ejecutados todos los casos de pruebas obtenidos con la técnica empleada, se concluye que los mismos fueron probados satisfactoriamente, corrigiéndose los hallazgos surgidos en una primera iteración y comprobándose su corrección en una segunda. Al concluir la prueba se demuestra que todas las funcionalidades de la aplicación web para la generación de instrumentos de recolección de datos se ejecutan satisfactoriamente, quedando libres de código repetido o innecesario.

Método de caja negra

El método de caja negra se centra en los requisitos funcionales del software. Es decir, permite al ingeniero de software derivar conjuntos de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales de un programa. El método de caja negra no es una opción frente a caja blanca. Es, en cambio, un enfoque complementario que tiene probabilidades de describir una clase diferente de errores de los que se identifican con los métodos de caja blanca (Pressman, 2010).

El método de caja negra se ejecuta a partir del desarrollo de pruebas funcionales, con la intención de identificar errores en las siguientes categorías (Pressman, 2010):

- Funciones incorrectas o faltantes.
- Errores de interfaz.
- Errores en estructuras de datos o en acceso a bases de datos externas.
- Errores de comportamiento o desempeño.
- Errores de inicialización y término.

Para desarrollar el método de caja negra se encuentra el uso de las técnicas (Pressman, 2010):

- Partición de equivalencia: divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

- Análisis de valores límites: prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

Con el propósito de evaluar a nivel de equipo de desarrollo el correcto funcionamiento y diseño de la aplicación web propuesta, se realizaron pruebas funcionales a nivel de unidad. Estas pruebas permitieron comprobar que la aplicación responde a cada uno de los RF y RNF sobre los cuales fue implementada. El desarrollo de estas pruebas se realizó aplicando el método de caja negra con el uso de las técnicas partición de equivalencia y análisis de valores

límites. Como herramientas de apoyo se utilizaron para el caso de las pruebas funcionales las Descripciones de Casos de Prueba (DCP). A continuación, en la figura 14 se muestran los resultados de las pruebas funcionales a nivel de unidad.

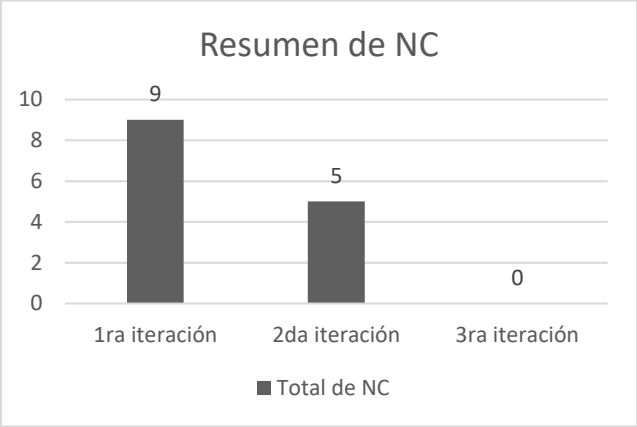


Figura 14. No conformidades detectadas al aplicar el método de caja negra a nivel de unidad.

Fuente: elaboración propia

Como muestra la figura 14, en la primera iteración se detectaron un total de 9 No Conformidades (NC). Las NC se clasifican en 4 de ortografía, 2 de redacción, 2 de funcionalidad y 1 de validación. Luego en una segunda iteración se identificaron 5 nuevas NC, distribuidas en 2 de validación, 1 de funcionalidad, así como 2 de consistencia y estándares. En la tercera iteración los resultados fueron satisfactorios, obteniéndose cero NC. Este resultado demuestra que las funcionalidades de la aplicación web para la generación de instrumentos de recolección de datos cumplen con cada uno de los RF y RNF.

Pruebas a nivel de integración y de sistema

Las pruebas a nivel de integración tienen como objetivo identificar errores introducidos por la combinación de programas o componentes probados unitariamente, para asegurar que la comunicación, enlaces y los datos compartidos ocurran apropiadamente. Se diseñan para descubrir errores o completitud en las especificaciones de las interfaces (Pressman, 2010).

Las pruebas a nivel de sistema tienen como objetivo verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las operaciones

apropiadas funcionando como un todo. Es similar a la prueba de integración, pero con un alcance más amplio (Pressman, 2010).

Con el propósito de evaluar la integración de los módulos de la aplicación propuesta y su correcto funcionamiento una vez acoplados, se realizaron pruebas funcionales a nivel de integración y de sistema, aplicando el método de caja negra con el uso de las técnicas partición de equivalencia y análisis de valores límites, utilizando las mismas DCP usadas en las pruebas a nivel de unidad. A continuación, en la figura 15 se muestran los resultados de la aplicación del método.

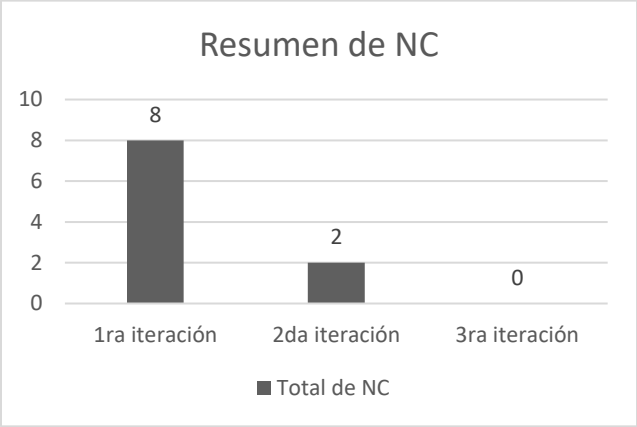


Figura 15. No conformidades detectadas al aplicar el método de caja negra a nivel de integración y sistema.
Fuente: elaboración propia

Como muestra la figura 15, en la primera iteración se detectaron un total de 8 No Conformidades (NC), clasificadas 5 de funcionalidad y 3 de validación. Luego en una segunda iteración surgieron 2 de consistencia y estándares. En la tercera iteración los resultados fueron satisfactorios, obteniéndose cero NC. Este resultado demuestra que los módulos de la aplicación web para la generación de instrumentos de recolección de datos se integraron satisfactoriamente y que el funcionamiento de la aplicación web es correcto.

Pruebas a nivel de aceptación

A la aplicación web para la generación de instrumentos de recolección de datos se le realizaron pruebas a nivel de aceptación, con el propósito de verificar que el

software está listo y cumple con cada una de las funcionalidades definidas con el cliente durante la etapa de levantamiento de requisitos. Luego de ejecutadas estas pruebas por los tutores de la presente investigación, quienes también ejercen como clientes, estos dictaminaron que la solución cumple con los requisitos definidos para su desarrollo.

3.4. Validación de los resultados de la investigación

Teniendo en cuenta que en la investigación realizada se define como idea a defender que: “si se desarrolla una aplicación web para la generación de instrumentos de recolección de datos se aumenta la variedad geográfica de la composición de la muestra y la celeridad en el proceso”. Para analizar la relación causa efecto entre la variable independiente “el desarrollo de una aplicación web” y las variables dependientes “aumenta la variedad geográfica de la composición de la muestra” y “celeridad en el proceso”, el autor de la presente investigación, define un conjunto de criterios de medida que permiten validar cómo a través de la aplicación web se logra la relación entre ambas variables. La obtención de estos criterios se realiza a partir de las principales deficiencias identificadas en la situación problemática que dan lugar al desarrollo de la solución propuesta.

Criterios de medida definidos:

- Formato en que son aplicados los instrumentos de recolección de datos: para el dominio de la investigación el autor define este criterio como la forma en que se aplican los cuestionarios, ejemplo con el uso del papel.
- Distribución geográfica de la muestra: se refiere a la ubicación del personal que se le va aplicar el instrumento de recolección de datos.

A continuación, en la tabla 11, se evalúan cada uno de los criterios de medida definidos anteriormente. La evaluación se realiza estableciendo un antes del desarrollo de la aplicación web y un después de obtenida la misma.

Tabla 11. Evaluación de los criterios de medidas definidos.
Fuente: elaboración propia

Criterios de medida	Antes de la aplicación	Después de la aplicación
Formato en que son aplicados los instrumentos de recolección de datos	Antes del desarrollo de la solución la mayoría de los instrumentos de recolección de datos se aplican manualmente con el uso del papel, lo que provoca demoras y posibles errores en el procesamiento de los resultados.	La aplicación web permite que los instrumentos de recolección de datos se puedan aplicar en formato digital, tanto a través de máquinas de escritorios, laptop o móviles, agilizando así el desarrollo del proceso y que la información de almacene en una base de datos.
Distribución geográfica de la muestra	En la actualidad con la aplicación de los instrumentos de recolección de datos a través del papel, las muestras de personas a responder el instrumento se limitan a la ubicación geográfica en la que se encuentre la persona encargada de aplicar el instrumento.	La aplicación de los instrumentos de recolección de datos a través de la solución propuesta, permite la participación de usuarios de diferentes ubicaciones, por tanto, la persona responsable de aplicar el instrumento solo necesitaría publicarlo en la web.

La comparación realizada en la tabla anterior, a través de los criterios de medida antes definidos, demuestra cómo utilizando la solución propuesta en la presente investigación, se aumenta la variedad geográfica de la composición de la muestra y la celeridad en el proceso.

3.5. Conclusiones parciales

El uso de métricas de validación del diseño certifica la obtención de una solución flexible al mantenimiento y a la introducción de cambios. La realización de pruebas internas a nivel de unidad, con el empleo del método de caja blanca y la técnica camino básico, certifican la correctitud del código de la solución, además el método de caja negra permitió mitigar las NC encontradas. La validación del resultado de la investigación, demuestra el cumplimiento de la relación causa efecto de la variable independiente “el desarrollo de una aplicación web” y las variables dependientes “aumenta la variedad geográfica de la composición de la muestra” y “celeridad en el proceso”.

CONCLUSIONES GENERALES

- El estudio de los referentes teóricos vinculados con soluciones informáticas para la generación de instrumentos de recolección de datos, permitió identificar características y buenas prácticas para ser incorporadas al desarrollo de la aplicación web propuesta.
- El empleo de técnicas de captura de requisitos, patrones de diseño y arquitectónicos, así como el uso de estándares de codificación en la implementación de la solución propuesta, permiten obtener una aplicación web flexible al mantenimiento y a la introducción de cambios.
- La validación de los RF, el diseño de la solución y el desarrollo de pruebas de software en los cuatro niveles, aplicando técnicas, métricas y métodos, permitió obtener una solución acorde las necesidades del cliente.
- Con la verificación de la relación causa efecto de la variable independiente sobre las variables dependientes de la investigación, se demuestra que con la aplicación web propuesta, se aumenta la variedad geográfica de la composición de la muestra y la celeridad en el proceso de recolección de datos.

RECOMENDACIONES

- Continuar con el desarrollo del sistema en función de sumar nuevas funcionalidades que permitan el cálculo de estadígrafos y la generación de gráficas a partir de los resultados consolidados.

BIBLIOGRAFÍA REFERENCIADA

Adrian Holovaty, J. K.-M. (n.d.). El libro de Django 1.0. In J. K.-M. Adrian Holovaty, *El libro de Django 1.0.*

Alegsa. (2020). *Alegsa*. Retrieved from http://www.alegsa.com.ar/Dic/framework.php

Álvarez, S. (2007). *Sistemas gestores de bases de datos. Introducción a este concepto y características especiales.* Retrieved from http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html

capterra. (n.d.). Retrieved from capterra.es: https://www.capterra.es/software/172452/zoho-forms

Ceballos, F. (2004). *Métodos de investigación científica.* Retrieved junio 15 , 2021

concepto. (n.d.). Retrieved from concepto.de: https://concepto.de/redes-sociales/#ixzz7VedY3RJE

definicionabc. (2022). Retrieved from definicionabc.com: https://www.definicionabc.com/general/formulario.php

ecured.cu. (n.d.). Retrieved from ecured: https://www.ecured.cu/Django

ecured.cu. (n.d.). Retrieved from ecured: https://www.ecured.cu/Aplicaci%C3%B3n_web

Hommel, S. (2019). *Estandares_de_codificacion_para_Java.*

IEEE. (2000). *Arquitectura Std 1471-2000.*

Larman. (2016).

Larman, C. (1999). *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México: PRENTICE HALL.

Larman, C. (2016). *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* (3ra ed.). Prentice-Hall.

Letelier, P., & Penadés, M. C. (2006). *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Retrieved from http://www.cyta.com.ar/ta0502/v5n2a1.htm

Londoño Abad. (2005).

Londoño Abad. (2005).

Londoño, J. H. (2005, abril 6). *Blogger*. Retrieved from http://ing-sw.blogspot.com/2005/04/tipos-de-pruebas-de-software.html

Lucid_Software_Inc. (2019). *Lucidchart*. Retrieved from https://www.lucidchart.com/pages/es

Meneses, J. (2016). *El cuestionario*.

Miguel Alejandro dias Rivera, Rafael Alejandro Aguilera Rodrígues. (2020). *Componentes para la gestión de alertas, mensajes y notificaciones en el Sistema de Gestión para la Atención a la Población.* La Habana.

Mozilla. (2019). *MDN web docs mozilla*. Retrieved from developer.mozilla.org/en-US/docs/Web/HTML

Mozilla. (2019). *MDN web docs mozilla*. Retrieved from developer.mozilla.org/en-US/docs/Web/CSS/CSS3

Mozilla. (2020). *Docs, MDN Web*. Retrieved from https://developer.mozilla.org/en-US/docs/Web/JavaScript

Navia López, A., & Vivero Díaz, Y. (2014). *Herramienta para obtener estadísticas del sistema gestor de base de datos PosgreSQL.*

Pressman, R. (2010). *Ingeniería del software.*

Pressman, R. S. (2001). Ingeniería del Software: una tecnología estratificada. In R. S. Pressman, *Ingeniería del Software. Un enfoque práctico. Quinta Edición.* España: McGraw Hill.

Pressman, R. S. (2002). *Ingeniería del Software. Un enfoque práctico. Quinta Edición.* Mc Graw Hill.

Pressman, R. S. (2003). *Ingeniería del Software. Un enfoque práctico. Sexta Edición.* Mc Graw Hill.

Pressman, R. S. (2010). *Ingeniería de Software. Un enfoque práctico* (Septima ed.). México DF: McGraw-Hill INTERAMERICA EDITORES.

Pressman, R. S. (2010). *Ingenieria de Software. Un enfoque práctico. Septima Edición.* Mexico, D.F: The Me Graw Hill Companies.

Riquelme, S. L. (2010). *Sistema Gestor de Base de Datos Relacionales*. Retrieved from http://www.fec.uh.cu/CUGIO/1%20acciones/Contenidos/BDR.pdf

Robledano, A. (2019, 9 23). *openwebinars.net*. Retrieved from openwebinars: https://openwebinars.net/blog/que-es-python/

Rocha, R. (2018). *Java EE 8 Design Patterns and Best Practices.* Birmingham: Packt.

Rodríguez Sala, J. J. (2003). *Introducción a la programación: teoría y práctica.* Editorial Club Universitario. Retrieved from ca: http://books.google.com.cu/books?id=nLMJsInMyBwC&pg=PA4&dq=lenguaje+de+programacion&hl=es#v=onepage&q=lenguaje%20de%20programacion&f=false

Rodríguez, J. J. (2003). Introducción a la programación: teoría y práctica. (9788484542742). Retrieved from s.l. : Editorial Club Universitario.

Rómmel, F. (n.d.). *sg.com.mx*. Retrieved from sg.com: https://sg.com.mx/revista/17/sqlite-la-base-datos-embebida

Rouse, M. (2020). *Tech Target*. Retrieved from https://searchsoftwarequality.techtarget.com/definition/pattern?_ga=2.163285817.444905919.1599000278-903531755.1599000278

significados. (n.d.). Retrieved from significados,com: https://www.significados.com/cuestionario/

Significados. (2022, junio 10). Retrieved from https://www.significados.com/estadistica/

Solis, M. C. (2003). *Una explicación de la programación extrema (XP).* Retrieved from http://www.willydev.net/descargas/prev/ExplicaXP.pdf

Sommerville, I. (2005). *Ingeniería del software. Séptima Edición.* Madrid. España: Pearson Educación. S. A.

tesisymasters. (2022). Retrieved from Tesis y Masters: https://tesisymasters.mx/instrumentos-de-recoleccion-de-datos/

UNAD. (2020, febrero 19). *stadium.unad.edu.co*. Retrieved from stadium.unad.edu.co: http://stadium.unad.edu.co/ovas/10596_9836/diagrama_de_clases_de_diseo.html

Zapata, M. (2022). Retrieved from sielo.sld.cu.