FACULTAD 4

Sistema de gestión para las listas de verificación en las actividades de calidad.

Trabajo de diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor:

Daniela Oramas Romero

Tutores:

M. Sc. Aymara Marín Díaz

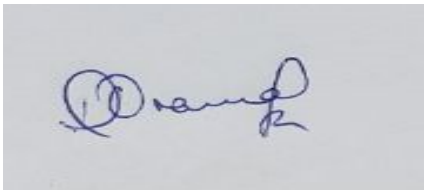
Dr. C. Yaimí Trujillo Casañola

La Habana, 27 de noviembre de 2023

Año 65 de la Revolución

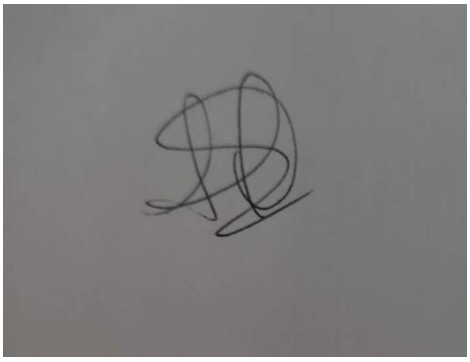
Declaración de autoría

El autor del trabajo de diploma con título **"Sistema de gestión para las listas de verificación en actividades de calidad."** concede a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la investigación, con carácter exclusivo. De forma similar se declara como único autor de su contenido. Para que así conste firma la presente a los 27 días del mes de noviembre del año 2023



_____

Firma del Autor



_____

Firma del Tutor

_____

Firma del Tutor

Datos de contacto

Aymara Marín Díaz: Se graduó en Ingeniería en Ciencias Informáticas en el año 2008. Alcanzó el título de Máster en Calidad de Software en el año 2016. En la actualidad ostenta la categoría docente de Profesor Auxiliar. Se especializa en los temas de Ingeniería de Software, mejora de procesos y pruebas de software. Cuenta con más de 15 publicaciones internacionales en el área de la calidad de software, tanto en su control como en su aseguramiento. Se encuentra certificada como: Certificación Internacional Tester of Fundation Level de ISTQB, Certificación Internacional Advanced Level, Test Manager de ISTQB. Se ha desempeñado en los roles de planificador, analista, probador, administrador de la calidad, gestor de pruebas y líder de proyecto. Tiene experiencia en el trato con clientes por los roles y cargos desempeñados.

Yaimí Trujillo Casañola: Doctora en Ciencias Técnicas de la Universidad de las Ciencias Informáticas (UCI) desde marzo del 2015. Profesora Titular desde noviembre del 2017 en la disciplina de Ingeniería y Gestión de Software. Desde su fundación se vincula al Centro Nacional de Calidad de Software (CALISOFT), posteriormente es Directora de Calidad de Software hasta el 2021, actualmente se desempeña como Decana de la Facultad 4. Tiene más de 50 artículos publicados en Revistas y Memorias de Eventos. Es árbitro de la Revista Cubana de Ciencias Informáticas (RCCI), de la Revista de I+D Tecnológico de la Universidad Tecnológica de Panamá, Revista Chilena de Ingeniería. Es Coordinadora de la Maestría de Calidad de Software. Posee las Certificaciones Internacionales ISTQB® Certified Tester – Foundation Level y Certified Tester, Advanced Level, Test Manager con el International Software Quality Institute (iSQI).

Dedicatoria

Dedico este logro con amor a mi madre y abuela, quienes siempre confiaron en mí y fueron pilares fundamentales para hacer posible este triunfo. A mi novio, cuyo anhelo por vivir este momento no se pudo realizar, este logro es un homenaje a nuestros sueños compartidos. Y, sobre todo, dedico este logro a mi abuelo, estoy segura de que, desde donde quiera que esté, está muy orgulloso de mi.

Agradecimiento

En este momento de logro, miro hacia atrás estos cinco años que se han ido volando y me gustaría aprovechar esta oportunidad para expresar mis agradecimientos a todas las personas que han sido fundamentales en este camino para lograr esta meta.

Agradecerle primeramente a mi mamá y mi abuela, sin ustedes nada de esto hubiera sido posible. Fueron mi fuerza, mi sostén, mis ganas de no rendirme y de seguir adelante. Abuela, qué sería de mi sin tu presencia constante en mi vida, tú que siempre te has desvivido por mí y me impulsas a ser mejor cada día, gracias, abuela, por ser mi roca, mi confidente y mi mayor defensora. Mamá, gracias a ti que me hiciste una persona de bien, me enseñaste todo lo que sé, y todo lo que soy te lo debo a ti, le doy gracias todos los días a dios por tenerte a ti como la mejor madre del mundo. A mi hermana, siempre me has visto como un ejemplo a seguir, si alguna vez te he inspirado, puedo decirte con certeza que tú también me has inspirado a ser una persona mejor.

A mi novio, tras tantos años compartiendo un camino lleno de sueños, hoy quiero agradecerte cada vez que me dijiste "tú puedes", "lo vas a lograr" o "todo va a salir bien", pues esas palabras de aliento me impulsaron a que nunca me rindiera y a creer en mí misma. Quiero agradecerte por la paciencia infinita que me tuviste, por ser mi maestro, mi mejor amigo y mi compañero de vida. Gracias por creer en mí incluso cuando yo dudaba, por levantarme cuando tropecé y por celebrar cada logro como si fuera tuyo. Simplemente, mil gracias.

A mi papá, gracias por tu constante apoyo a lo largo de mi vida, a pesar de la distancia que nos separa, agradezco infinitamente que siempre hayas estado ahí.

Agradecerles a mis tíos Titi y Edelio, y a mi padrastro, a los tres, por el amor y el apoyo que me han brindado a lo largo de los años.

A mi segunda familia, mi suegra y mi abuela Ladys, que me han acogido con los brazos abiertos y me han tratado como una hija, gracias por estar siempre presente para mí.

A mi "Family UCI", mi relación más tóxica, mi enana, gracias por aparecer en mi vida en el momento preciso, incluso sin darte cuenta de cuánto te necesitaba. Eres mi compañía favorita, la única persona capaz de sacarme de quicio y al mismo tiempo, en un abrir y cerrar de ojos, dibujar una sonrisa en mi rostro. Es curioso cómo logras despertar en mí la contradicción de querer matarte y, al instante, preguntarme qué sería de mí sin ti. Gracias por las peleas, las lágrimas pero sobre todo por los días incansables de risas, esas noches que no me dejaste durmiendo sola y por quererme tanto como lo haces. Mi hijo malcriado, pollo, gracias por cada sonrisa que me sacaste a pesar de parecer imposible, por aguantar todas mis peleas, aunque la mayoría te las merecías y especialmente gracias por no dejarme sola. A los dos, nunca me voy a arrepentir de haberme juntado con ustedes, porque en ese momento empezó la mejor amistad que tengo hoy.

A mis amigos en general, todos los que de una forma u otra han hecho que este logro sea posible, en especial a Leya, gracias por hacer que mi primer año fuera tan extraordinario, Quique por no dejar que me rindiera, mis puchis, Rosmery, Chacon, Roberto, Alejandro, Mario gracias por toda su ayuda y compañía, Laura por siempre darme ese empujoncito de tú sí puedes, a Jose por todo lo que has hecho por mí y no dejarme desamparada cuando más lo necesitaba. Gracias a mis compadres mejor conocidos como la familia "García", por ser mi escape del estrés. Mi grupo 4302 y mi gente del 15, gracias por todo lo que

hemos compartido juntos, tanto las fiestas como los repasos. Mis amigos del Pre Yanet, Fito, Sheila, y mi "Padre" Daniel como cariñosamente le digo, gracias por siempre haber confiado en mí.

A todos esos profesores que se dedicaron a enseñarme todo lo que sabían, a todos los que me guiaron por el mejor camino para lograr este triunfo.

A mi tutora, a pesar de contar con poco tiempo, gracias siempre por tu dedicación y apoyo.

Y para concluir me gustaría hacerlo con una frase de Pablo López: "Entre la felicidad y la desesperación la carretera es muy corta", gracias a todos los que de una forma u otra han acompañado mi camino hacia la felicidad.

Resumen

Las listas de verificación son herramientas utilizadas para controlar y verificar el cumplimiento de requisitos y funciones específicas, así como recopilar datos e información relacionados con el software. En la Universidad de Ciencias Informáticas, dicha técnica se emplea como parte de las actividades de calidad, sin embargo, realizar estas listas manualmente puede ser tedioso y requerir gran cantidad de tiempo, atención y esfuerzo. Además, genera trabajo adicional, retrasos e incluso afecta la calidad de las actividades. Como consecuencia, surge la necesidad de crear un sistema para la gestión de las mismas, con el objetivo de disminuir el esfuerzo en las actividades de calidad. El sistema propuesto se basa en buenas prácticas de ingeniería de software y estándares de calidad reconocidos, utilizando técnicas de desarrollo ágil. El desarrollo del mismo ha sido guiado por las especificaciones que propone la metodología de desarrollo de software SCRUM, la cual facilita el análisis, el diseño, la implementación y la validación de las funcionalidades del sistema. Para el desarrollo de la propuesta de solución, se selecciona Python en su versión 3.11 como lenguaje de programación, Visual Paradigm como herramienta de modelado y SQLite como gestor de base de datos. La validación del mismo se realiza mediante pruebas estructurales, funcionales, no funcionales y asociadas al cambio para garantizar su calidad y conformidad con los requisitos establecidos.

PALABRAS CLAVE: calidad; calidad de software; listas de verificación

Abstract

Checklists are tools used to monitor and verify compliance with specific requirements and functions, as well as collect software-related data and information. At the University of Computer Sciences, this technique is used as part of quality activities; however, making these lists manually can be tedious and require a large amount of time, attention and effort. In addition, it generates additional work, delays and even affects the quality of activities. As a consequence, the need arises to create a system for their management, with the aim of reducing the effort in quality activities. The proposed system is based on good software engineering practices and recognized quality standards, using agile development techniques. Its development has been guided by the specifications proposed by the SCRUM software development methodology, which facilitates the analysis, design, implementation and validation of the system's functionalities. For the development of the solution proposal, Python version 3.11 is selected as the programming language, Visual Paradigm as the modeling tool and SQLite as the database manager. Its validation is carried out through structural, functional, non-functional and change-associated tests to guarantee its quality and compliance with the established requirements.

KEYWORDS: checklists; quality; software quality

Tabla de contenidos

Índice de tablas

Índice de figuras

Introducción

La era digital, con el tiempo, se ha convertido en un componente cada vez más relevante en el mundo moderno, transformando grandemente la forma de interactuar el ser humano con el mundo (Cueva Gaibor, 2020). La ingeniería de software, una disciplina muy importante en la vida actual, es un atributo clave dentro del entorno digital. A lo largo de la historia esta ha seguido un enfoque metódico, ordenado y medible para el desarrollo, funcionamiento y mantenimiento del software, con el propósito de lograr que este sea de excelencia en términos de calidad (Marín Díaz et al., 2020). Según los autores Gómez Palomo y Moraleda Gil (2020) se define como un conjunto de programas y datos que se utilizan para coordinar y controlar el comportamiento de un sistema informático.

Como resultado del uso y aumento de la demanda del software, se ha hecho fundamental la necesidad de mantenerlo seguro, estable y eficaz. Sin embargo, en la industria se siguen generando y desplegando productos que no cumplen con los estándares de calidad esperados. De acuerdo con Pressman y Maxim (2019) es el cumplimiento de la especificación original de un producto, vinculado a la satisfacción del usuario, brindando un valor visible a los mismos. Dirigiendo este concepto al campo del software, los profesores Carrizo y Alfaro (2018) definen a la calidad de software como el cumplimiento de los requisitos conceptuales por parte del mismo, obteniéndose mediante mejoras paulatinas en el proceso de producción, mantenimiento y gestión del mismo.

La calidad se gestiona mediante un conjunto de acciones y herramientas diseñadas para asegurar que los productos o servicios de una empresa cumplan con los requisitos establecidos y satisfagan las necesidades y expectativas de los clientes. Esto implica la mejora continua de los procesos de una organización. Ambos conceptos están vinculados, como para mejorar los estándares de los productos y servicios ofrecidos por una organización, es necesario identificar y eliminar las ineficiencias y defectos en los procesos. Las empresas que son capaces de mejorar sus sistemas y ofrecer software de alta calidad tienen una ventaja competitiva en el mercado (Defeo & Juran, 2016).

Este proceso de gestión conlleva una evaluación de la calidad de software, la cual es realizada a lo largo del ciclo de desarrollo de software, siendo una actividad comúnmente practicada por profesionales de variada experiencia y de distintos campos. Los procedimientos pueden variar en cada organización, pero lo importante es que estén escritos, personalizados, adaptados a los

procesos de la organización y que sean cumplidos (Paz, 2016). Como se enunció anteriormente, éstas pueden ser realizadas en distintos momentos como en el desarrollo, una vez construido, antes de adquirir o al implantar. Cualquiera sea el caso, la evaluación de la calidad comienza a partir de la definición y declaración de los requerimientos de calidad del software, sean explícitos o implícitos. Esto es válido dado que uno de sus objetivos es comprobar que dichos requerimientos sean satisfechos (Segura et al., 2008).

Las pruebas verifican dinámicamente el comportamiento de un programa contra el comportamiento esperado, usando un conjunto finito de casos de prueba, seleccionados de manera adecuada (Paz, 2016). Estas chequean si el software cumple o no con los requerimientos propios establecidos (Flores Castillejo, 2020). Esta actividad permite reducir riesgos en las aplicaciones, y lograr que se identifiquen los defectos en las etapas más tempranas durante el desarrollo de software.

El proceso de prueba de software tiene tres etapas principales: el desarrollo de los casos de prueba, la ejecución de estos casos de prueba y el análisis de los resultados de la ejecución. Este conjunto de pasos tiene como objetivo asegurar la calidad de un sistema mediante el uso de pruebas enfocadas a validar distintos aspectos de la calidad del sistema, descritos de manera detallada en el estándar ISO/IEEE 25010-2011 (Chinarro Morales, 2019). Se derivan de estas las técnicas de pruebas. Una técnica de prueba juega un papel importante en las pruebas de software. Mientras realiza la caja negra la prueba, el probador no tiene ningún conocimiento de diseño y sin acceso al código fuente. Este solo tiene conocimiento de la arquitectura del sistema. Esta técnica se utiliza para garantizar que todas las entradas que necesita el sistema sean aceptadas de la manera especificada y proporcionar la salida correcta (Myers et al., 2011).

Estos documentos son conocidos como artefactos, de los cuales existen muchos tipos. Sin embargo, esta investigación se enfoca en las listas de verificación, pues son los artefactos seleccionados para estudiar (Nazar et al., 2016). Estas listas de verificación permiten a los revisores evaluar los estudios en función de un conjunto de componentes de calidad predefinidos. La lista de verificación es una lista de chequeo y posibles preguntas a desarrollar en las entrevistas que especifican dónde buscar, qué buscar y cómo buscar, diseñadas en correspondencia con los criterios de evaluación con el objetivo de valorar la conformidad de los proyectos con dichos criterios (ISTQB, 2023).

A nivel global, se utilizan en las auditorías internas a los proyectos de desarrollo de software como plan de muestreo y administrador del tiempo, con la meta de proporcionar evidencias de que se realizó la auditoría y lo que se trató en la misma (EALDE Business School, 2020). Además, son empleadas en la medicina, específicamente en el campo de la cirugía (WHO Patient Safety & World Health Organization, 2009) y en el manejo del inventario de los hospitales con el objetivo de facilitar el almacenamiento y uso de los recursos de la institución (Gehlbach & Artino, 2018). Las mismas son consideradas una manera sencilla y fácil de recordarle a los profesionales, sin importar cual sea su rama, de lo que ya conocen, pero pueden olvidar fácilmente.

La Universidad de las Ciencias Informáticas es un centro docente productivo donde se desarrolla software para diversos sectores como la ofimática y la medicina. Cuenta con varios centros de desarrollo en ejecución dentro de la institución, entre los que se encuentran el Centro de Innovación y Desarrollo para Internet (CIDI), el Centro de Informatización de la Gestión Documental (CIGED), entre tantos otros (Universidad de las Ciencias Informáticas, 2023). Según la organización Top Universities, la universidad alcanzó el puesto número 123 en el Ranking de Universidades Latinoamericanas del 2021 (QS Quacquarelli Symonds Limited, 2021). Los resultados de la institución hablan por sí solos, tanto a nivel de producción de software, como a nivel de mentor y guía en el desarrollo de las ciencias informáticas en Cuba (Gulín-González, 2022). Con dichos elementos se valida la valía de la institución a nivel nacional a la hora de la creación de productos de software.

Los productos desplegados por la universidad son de calidad, evidenciado en el recibimiento del nivel 2 del Modelo de Madurez de Capacidad Integrada, CMMI por sus siglas en inglés (Falcón Márquez, 2015), lo cual avala a la institución como un centro con experiencia en la producción de software, desde su organización inicial hasta la revisión de la calidad final. Las listas de verificación son una herramienta valiosa en la gestión de esta en los productos elaborados en los diferentes centros de la institución. A través de la creación y utilización de listas de verificación específicas para los distintos tipos de productos, se pueden identificar los requisitos y criterios de calidad que deben cumplirse para garantizar que los productos sean consistentes y cumplan con las expectativas de los clientes en las diferentes etapas del proceso de desarrollo de software (Tague, 2005).

El Departamento de Calidad de Software utiliza las listas de verificación, resultando en un aumento de la efectividad con el fin de garantizar la organización y el éxito en la ejecución de proyectos y tareas. Estas tienen un formato definido por la institución en Excel. Al ser realizadas

de forma manual se pueden introducir defectos tanto en el proceso de confección como en su aplicación, y generan retrabajo, tiempo e incluso retraso, llegando a afectar la calidad de las actividades.

A partir de la situación problemática anterior, se define como **problema de investigación**: ¿Cómo disminuir el esfuerzo en la realización de actividades de calidad ejecutadas en la actividad productiva de la Universidad de las Ciencias Informáticas?

Por lo que se plantea como **objeto de estudio:** Los sistemas de gestión de la información, enmarcado en el **campo de acción**, el proceso de gestión de la información de las listas de verificación.

Para dar solución al problema de investigación planteado, se propone como **objetivo general** desarrollar un sistema de gestión de la información de las listas de verificación de manera que disminuya el esfuerzo dedicado a las actividades de calidad.

De este objetivo se descomponen como **objetivos específicos** siguientes:

1. Definir el marco teórico de la investigación mediante el estudio y el análisis de los principales referentes teóricos para el desarrollo de la solución.
2. Definir los requisitos para un sistema que gestione la creación y utilización de las listas de verificación.
3. Implementar un sistema que gestione la creación y utilización de listas de verificación utilizadas en las actividades de calidad.
4. Validar el sistema propuesto.

De acuerdo a todo lo antes expuesto se propone la siguiente idea a defender:

El desarrollo de un sistema web que gestione la creación y utilización de las listas de verificación permitirá la disminución del esfuerzo dedicado a las actividades de calidad.

Métodos teóricos

Histórico-lógico: En la primera parte de la investigación se desarrollará un estudio del estado del arte de la problemática; así como se analizarán las ventajas y desventajas de cada una de las herramientas utilizadas actualmente para la gestión de listas de verificación.

Método analítico–sintético: Se utilizará para el análisis de la información existente sobre: listas de verificación para actividades de calidad y las herramientas a emplear en el desarrollo del sistema. Luego como parte de la aplicación del método se determinarán los puntos esenciales de cada elemento objeto de análisis y se sintetizan en aspectos relevantes para la investigación.

Métodos empíricos

Entrevista: Se utilizará la entrevista como una conversación planificada con especialistas y revisores, revisores líderes de la Universidad de las Ciencias Informáticas para obtener información acerca del problema en cuestión. Su uso constituye un medio para el conocimiento cualitativo de las características particulares de un proceso y puede influir en el posterior análisis y diseño del producto de software que contribuirá a la disminución del esfuerzo dedicado a las actividades de calidad.

El siguiente trabajo de diploma está estructurado en 3 capítulos, a continuación, se muestra una breve descripción de cada uno de ellos:

Capítulo 1 denominado “Fundamentación teórica”, donde se incluyen los resultados del estudio sobre el estado del arte de herramientas existentes en Cuba y en el resto del mundo para la gestión de listas de verificación; así como el análisis de la metodología, tecnologías, técnicas y herramientas a utilizar en el desarrollo de la aplicación. Se exponen además los conceptos fundamentales para una mejor comprensión de la investigación.

Capítulo 2 denominado “Características del sistema”, donde se expone la propuesta del sistema, así como se enumeran los requisitos funcionales y no funcionales del mismo. Se modelan y describen las historias de usuarios.

Capítulo 3 denominado “Implementación y Pruebas”, se presentan los estándares de codificación, destacando las mejores prácticas para asegurar la calidad en el proceso de desarrollo de software. También se aplican los niveles de pruebas de componente, de sistema y de aceptación que permiten comprobar el correcto desarrollo de las funcionalidades implementadas.

Capítulo 1 - Fundamentación teórica

Introducción

El presente capítulo define un marco conceptual referente a la gestión de listas de verificación, conceptos y funciones del mismo. Se establecen comparaciones entre los sistemas homólogos existentes. También se describen las herramientas, tecnologías y la metodología de desarrollo de software a utilizar que deberán ser aplicadas posteriormente en el desarrollo de la propuesta de solución.

1.1 Conceptos asociados al tema

Con el fin de facilitar la comprensión de esta investigación, a continuación, se presentan conceptos relacionados que ayudarán a clarificar el contexto.

1.1.1 Calidad

La calidad es un concepto complejo y multifacético que puede ser difícil de definir de manera precisa. El Dr. Pressman la define como "la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente" (Pressman & Maxim, 2019). La norma ISO 9000:2000 describe la calidad como: “el grado en el que un conjunto de características (rango diferenciador) inherentes cumple con los requisitos (necesidad o expectativa establecida, generalmente implícita u obligatoria)” (López Rivero, 2013). La RAE (Diccionario de la Real Academia de la Lengua, 2018), define la calidad como: “Propiedad o conjunto de propiedades inherentes a algo, que permiten juzgar su valor”. Esta definición está orientada al mercado (Aizprua et al., 202).

El reconocido autor Philip Crosby la definió de una manera más profunda: "El primer supuesto erróneo es que calidad significa bueno, lujoso, brillo o peso. La palabra "calidad" es usada para darle el significado relativo a frases como "buena calidad", "mala calidad" y ahora "calidad de vida". Calidad de vida es un cliché porque cada receptor asume que el orador dice exactamente lo que él (ella) "el receptor", quiere decir. Esa es precisamente la razón por la que definimos calidad como "Conformidad con requerimientos", si así es como lo vamos a manejar…” (Hoyer & Hoyer, 2001). Mientras que Juran la definió como “aquellas características de un producto que

se basan en las necesidades del cliente y que por eso brindan la satisfacción del producto" (Juran & Defeo, 2016).

La calidad implica cumplir con requisitos, estándares y características establecidas, así como satisfacer las necesidades y expectativas del cliente. Es un concepto amplio y subjetivo que puede variar según el contexto y las perspectivas involucradas. La comprensión y búsqueda de la calidad es fundamental en diversos campos, incluido el desarrollo de software, la gestión de proyectos y la satisfacción del cliente.

1.1.2 Calidad de software

Según Pressman la calidad del software no es más que: "La medida en que un sistema, componente o proceso cumple los requisitos explícitamente especificados y las expectativas del usuario o cliente. La calidad del software incluye características tales como la corrección, confiabilidad, eficiencia, facilidad de uso, mantenibilidad y portabilidad" (Pressman & Maxim, 2019).

Sin embargo, la IEEE define la calidad de software como "el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario" haciendo énfasis en los requisitos específicos del sistema y en la satisfacción del cliente (Villanueva Rosas & Muñoz, 2020). La norma ISO 25010, la define como "el grado en que dicho producto satisface los requisitos de sus usuarios aportando de esta manera un valor" (ISO 25010, 2011).

La calidad se refiere a un conjunto de propiedades inherentes a algo que permiten juzgar su valor. En el caso del software, esta implica cumplir con los requisitos explícitamente especificados y las expectativas del usuario o cliente, así como ofrecer características como corrección, confiabilidad, eficiencia, facilidad de uso, mantenibilidad y portabilidad.

1.1.3 Lista de verificación

Conforme a la Organización Panamericana de la Salud (OPS), las listas de verificación son una herramienta útil para la mejora continua, debido a que permiten recopilar información importante sobre los procesos clave de una empresa durante una revisión o auditoría. Estas listas son documentos que contienen una serie de verificaciones o preguntas que se deben realizar para asegurarse de que se están cumpliendo los estándares y procedimientos establecidos (ISTQB,

2023,). Es importante que estas listas sean sencillas y fáciles de usar y entender para que la persona que las utilice pueda llevar a cabo una secuencia ordenada de observaciones o verificaciones. Además, estas listas también pueden usarse como método de recolección de datos durante la auditoría (Saavedra Saavedra, 2022).

Es una lista conteniendo preguntas sobre la evaluación del software y sus diferentes aristas, para garantizar la calidad, las cuales son evaluadas según su cumplimiento y desempeño (ISOTools, 2022)

Las listas de verificación son herramientas importantes para la mejora continua y la evaluación de procesos, incluyendo la evaluación de la calidad del software. Permiten recopilar información clave, asegurar el cumplimiento de estándares y procedimientos, y facilitar la identificación de áreas de mejora. La simplicidad y facilidad de uso son características fundamentales de las listas de verificación.

El conocimiento de estos conceptos capacita para evaluar y mejorar la calidad en el trabajo, especialmente en el desarrollo de software. Las listas de verificación son una herramienta valiosa que te permite recopilar información, identificar áreas de mejora y garantizar el cumplimiento de estándares, contribuyendo así a la entrega de productos o servicios de mayor calidad y satisfacción para tus clientes.

1.2 Análisis de sistemas homólogos

El software a crear es factible a nivel de mercado, contando con características únicas como la integración con el formato Excel, la facilidad de uso de la web y el uso libre de costo. Con el objetivo de identificar otras características principales que debe tener un sistema de gestión de listas de verificación, se realizó un estudio de este apartado en distintos sistemas informáticos actualmente disponibles en el mercado, donde se identificó para mejorar la calidad y la eficiencia en los procesos empresariales la integración de una plataforma de gestión de calidad como Qualio.

Qualio: Es una plataforma diseñada para mejorar la calidad y la eficiencia en diferentes procesos empresariales, permitiendo la creación y personalización de listas de verificación de calidad, la automatización de tareas y flujos de trabajo, la gestión del cumplimiento normativo y la integración con otras herramientas empresariales como Slack. Al integrar Qualio en el sistema

empresarial, se pueden lograr mejoras significativas en la calidad y la eficiencia al automatizar tareas, mejorar la gestión del cumplimiento normativo y facilitar la colaboración y la comunicación entre los equipos (Díaz, 2016).

Qualio, anteriormente llamado ZenDOC, es un software para llevar a cabo la gestión de un sistema de calidad especialmente destinado a compañías orientadas a las ciencias de la vida, permitiendo gestionar el control de documentos, la formación, las acciones preventivas y correctivas, las auditorías, los proveedores y las quejas. Qualio se enfoca en la documentación y en la formación, pues posee una sección en la que se listan los documentos existentes, otra sección de espacio de trabajo a través de la cual se gestionan los documentos y las formaciones, y por último una sección de reportes donde se muestran los reportes correspondientes a las formaciones realizadas, las actividades, las revisiones, las auditorías y el control de cambios. En cuanto a la gestión de documentos. No brinda funcionalidades específicas para la gestión de hallazgos que faciliten su análisis, como la generación de gráficos o el envío de notificaciones ante fechas de revisión o verificación de eficacia que se aproximen. De hecho, no permite el envío de notificaciones en ningún caso pues el usuario solo se entera de las acciones a llevar a cabo entrando a la sección correspondiente. Así mismo, tampoco brinda la posibilidad de gestionar los indicadores y riesgos, y no posee un calendario (Díaz, 2016).

Otro sistema que es válido destacar es Checklist.gg, una herramienta de gestión de listas de verificación impulsado por Inteligencia Artificial (IA) que está diseñada para ayudar a las organizaciones en sus procesos (checklist.gg, 2023). A pesar de ser una propuesta cautivadora, cuenta con la limitación de que su acceso es de pago, y solo permite generar 3 listas de verificación.

Un sistema adicional es el de Staff.Wiki, que genera listas de verificación dependiendo de la descripción que se le introduzca. Funciona integrado en el sitio, que está enfocado en el desarrollo de wikis y de artículos dentro de las mismas. Por lo tanto, no es factible para su uso en este sistema pues utiliza un enfoque distinto al necesario para la solución (Staff.Wiki, 2023).

Tabla 1 - Comparación de sistemas homólogos

Sistemas	**Pago**	**Funcionalidades**	**Integración con el formato excel**

Qualio	X	Creación y personalización de listas de verificación	No
Checklist.gg	X	Genera listas de verificación	No
Staff.Wiki	X	Administrar listas de verificación	No

Luego del estudio a los sistemas homólogos se concluye que ninguno resuelve la situación a desarrollar, dado que son sistemas de pago y ninguno se enfoca en listas de verificación de calidad en el ámbito del software, por lo cual se ve necesario realizar el sistema en su totalidad. Se tendrán en cuenta características y comportamientos de varias de ellas para el desarrollo de la solución informática, tales como el administrar listas de verificación y algunos de sus formularios al gestionar las listas de verificación.

1.3 Fundamentación del proceso de software a desarrollar

La propuesta de solución que se plantea se enfoca en el uso del entorno web como un medio para gestionar de manera más eficiente las listas de verificación. El entorno web ofrece una serie de beneficios que lo hacen ideal para este propósito, tales como su amplia disponibilidad y accesibilidad, su versatilidad para trabajar con diferentes plataformas y dispositivos, y su capacidad para trabajar en tiempo real.

1.3.1 Metodología de desarrollo de software

Las metodologías de desarrollo de software surgieron para mejorar la eficiencia y la calidad de los proyectos de software. Estas se refieren a un conjunto de procedimientos lógicos y racionales que se utilizan para alcanzar un objetivo específico que requiere habilidades y conocimientos específicos. Es una etapa específica dentro de un trabajo o proyecto en la que se parte de una base teórica y se seleccionan técnicas o métodos concretos para guiar el proceso y lograr los objetivos establecidos. La metodología consiste en el conjunto de métodos que se utilizan en una actividad particular con el fin de formalizarla y optimizarla. Define los pasos a seguir y cómo llevarlos a cabo para completar una tarea de manera eficiente (Hernández Fariñas, 2022).

Existen dos tipos principales, las metodologías tradicionales y las metodologías ágiles. Las metodologías tradicionales se enfocan en la documentación exhaustiva durante todo el ciclo del proyecto, mientras que las ágiles priorizan la capacidad de respuesta frente a los cambios, la confianza en las habilidades del equipo, y una mayor colaboración y dinamismo en los métodos de trabajo. Aunque las tradicionales todavía tienen un papel importante, las ágiles se han vuelto cada vez más populares debido a su enfoque en la adaptabilidad y la colaboración. Por consiguiente, para un proyecto de software, sería más beneficioso utilizar una metodología ágil para garantizar una mayor eficiencia y calidad en el proceso de desarrollo (Echevarría Nieves, 2021).

Considerando las características del proyecto y del equipo de desarrollo, es recomendable utilizar una metodología ágil. Las metodologías ágiles son enfoques flexibles y colaborativos que se adaptan bien a entornos donde los requisitos son cambiantes, se busca una entrega rápida y se fomenta la colaboración y retroalimentación continua. Entre las principales metodologías ágiles se encuentran las siguientes:

- Scrum: Es un enfoque de gestión de proyectos que se centra en la colaboración efectiva de equipos y en la entrega de valor en incrementos. Para lograr esto, Scrum utiliza una estructura definida que incluye roles, artefactos y eventos específicos (Schwaber & Sutherland, 2020).
- Programación Extrema (XP): Aborda el desafío de lidiar con requisitos en constante cambio a lo largo del desarrollo de software. Esta se basa en principios probados en la ingeniería de software y busca proporcionar una estructura sólida para el desarrollo de software de calidad. Se enfoca en la colaboración cercana entre el equipo de desarrollo y los clientes, la simplicidad en el diseño y la implementación, la retroalimentación continua y la capacidad de respuesta a los cambios. XP también promueve la realización de pruebas frecuentes y la integración continua para garantizar la calidad del software (Bautista-Villegas, 2022).
- Crystal Metodologías: Es altamente flexible y abierta, lo que la convierte en una de las metodologías más ágiles disponibles. Sin embargo, a su vez, esto también la convierte en una de las más difíciles de comprender, entender y aplicar, raíz a que su éxito depende en gran medida de la experiencia y habilidades de quienes la utilizan (Ibarra Corona & Escudero-Nahón, 2021).

- Kanban: Es una metodología que utiliza tarjetas visuales para gestionar el trabajo. La palabra "Kanban" proviene del japonés y se refiere a estas tarjetas. Es una metodología especialmente útil cuando se trabaja con productos que están abiertos a cambios y en situaciones donde la planificación del trabajo es difícil (León Ardilla, 2020).
- Running Lean: Es una guía valiosa y una herramienta práctica tanto para emprendedores, directivos, propietarios de pequeños negocios, desarrolladores y programadores, como para cualquier persona interesada en iniciar un negocio. Esta metodología ofrece un enfoque eficiente y ágil para analizar nuevas ideas de productos y crear productos exitosos. Se basa en un proceso sistemático que permite iterar desde un plan inicial (Plan A) hacia un plan que realmente funcione, evitando quedarse sin recursos en el proceso (León Ardilla, 2020).

Metodología SCRUM:

Esta metodología es la más común y aplicada. Si bien es cierto que no da recomendaciones específicas para cada situación ayuda en gran medida a que se solucionen los posibles inconvenientes y a que se favorezca la productividad. Es un método eficaz que con práctica continua permite que la organización funcione con éxito y que los clientes y usuarios lo aprecien, principalmente en entornos (León Ardilla, 2020).

Roles de SCRUM (Schwaber & Sutherland, 2020):

- Product Owner: Es la única persona autorizada para decidir las funcionalidades y características que tendrá el producto. Es quien representa al cliente y usuarios del software.
- Scrum Máster: Es quien interactúa con el equipo, el cliente y los gestores. Garantiza el funcionamiento de los procesos y la metodología. Debe ser miembro del equipo y trabajar a la par. Coordina los encuentros diarios y se encarga de eliminar obstáculos.
- Scrum Team: Es el equipo de desarrolladores multidisciplinario, integrado por programadores, diseñadores, arquitectos y testers, que en forma auto-organizada son los encargados de desarrollar el producto.

Artefactos de SCRUM (Schwaber & Sutherland, 2020):

Product Backlog (Listado de Requisitos): Es una lista priorizada de todas las cosas que se necesitan hacer en un proyecto. El dueño del producto es responsable de definir, actualizar y organizar los requisitos en función de su valor, riesgo, prioridad y necesidad.

Sprint Backlog: Es un conjunto de elementos seleccionados del Product Backlog para trabajar durante un período de tiempo específico, llamado Sprint. El equipo de desarrollo divide estos elementos en tareas más pequeñas y las organiza en el Sprint Backlog. Esta lista es la guía para el trabajo diario del equipo durante el Sprint.

Incremento: Es el resultado tangible y potencialmente entregable del trabajo realizado durante un Sprint. Es una versión mejorada y completa del producto que incluye todas las funcionalidades y características desarrolladas hasta el momento.

Artefactos a usar:

- En las metodologías ágiles, como Scrum o Kanban, el levantamiento de requisitos es un proceso continuo que se lleva a cabo a lo largo del desarrollo. En estas metodologías, los requisitos se pueden modificar o añadir a medida que se avanza en el desarrollo, lo que permite adaptarse a las necesidades cambiantes de los usuarios y clientes. Este levantamiento es una buena práctica que debe aplicarse en cualquier metodología de desarrollo de software.
- Las historias de usuarios son el principal artefacto de Scrum, pero también son un elemento fundamental de XP. Se utilizan para definir el alcance del trabajo que se realizará durante cada sprint.
- Product Backlog.
- Sprint Backlog.

El uso de esta metodología ágil tiene como ventajas (García Avila, 2022):

- La satisfacción del cliente es un aspecto fundamental en las metodologías ágiles, pues involucran al cliente como parte integral del equipo de trabajo y lo comprometen con el resultado final. Esto marca una gran diferencia en comparación con las metodologías tradicionales, donde el cliente es considerado como una entidad separada que se encarga de proporcionar requisitos o necesidades funcionales, y luego aprueba documentos extensos que no volverá a revisar hasta que el producto esté terminado.

- La simplicidad es una característica clave de Scrum, ya que proporciona una estructura clara y definida para cada evento. Cada evento en Scrum está cuidadosamente identificado, lo que incluye quiénes participan, cuál es el objetivo, cuánto tiempo debe tomar y cuál es el resultado esperado. Esta claridad y especificidad facilitan a los miembros del equipo la comprensión y adopción de la metodología.
- La inspección es un elemento destacado en Scrum y se encuentra presente en tres de sus eventos principales: la reunión diaria, la revisión del sprint y la retrospectiva del sprint. Estos eventos están diseñados específicamente para fomentar la inspección y garantizar la mejora continua del equipo.
- La adaptación es una de las mejores cualidades de la metodología Scrum, ya que se destaca por su capacidad de responder a los cambios en las características del producto. Esta flexibilidad es una de las principales diferencias que la distingue de otras metodologías, ya que permite realizar cambios en cualquier momento, incluso durante el desarrollo de las diferentes iteraciones o Sprints, siempre y cuando no afecten la entrega acordada.

En lugar de desarrollar todo el proyecto de una sola vez, Scrum divide el proyecto en partes más pequeñas llamadas sprints. Cada sprint tiene una duración fija y se enfoca en entregar un conjunto de funcionalidades del proyecto. Al final de cada sprint, el equipo revisa el progreso y ajusta el plan para el siguiente sprint. La idea central de Scrum es que el equipo trabaje junto de forma colaborativa y se comunique de manera efectiva. Scrum se enfoca en la entrega de valor al cliente de manera incremental y en la mejora continua del proceso de desarrollo (García Avila, 2022).

1.4 Herramientas y tecnologías

El uso de herramientas mejora la calidad en el desarrollo de un proyecto de software. Se utiliza un conjunto variado de ellas, cada una cumpliendo un rol definitivo en el proceso. Entre las opciones seleccionadas se encuentra Visual Studio Code, un editor de código fuente gratuito y de código abierto. Se eligen estas herramientas por la facilidad de implementación que ofrece Python y la adaptabilidad que posee para la web. Además, todas utilizan el mismo lenguaje, para facilitar el trabajo del autor.

1.4.1 Herramienta CASE

La herramienta CASE a utilizar para generar los distintos artefactos es el Visual Paradigm, en su versión 8.0, muy útil para modelar todas las representaciones ingenieriles requeridas. Además, destaca por su diseño centrado en el modelado de los distintos diagramas orientados al negocio. Visual Paradigm es una herramienta de modelado de software que permite crear diferentes tipos de diagramas UML y otros modelos de diseño para diferentes fases del ciclo de vida del software. Es una herramienta muy completa y potente que te permite modelar y diseñar sistemas de software de manera visual y colaborativa (Visual Paradigm, 2023).

Con Visual Paradigm, puedes crear una gran variedad de diagramas, desde los más simples como diagramas de clases o de casos de uso, hasta los más complejos como diagramas de secuencia o de actividad. Además, puedes generar código a partir de tus modelos de diseño, lo que te permite ahorrar tiempo y reducir errores de implementación. Otra característica importante de Visual Paradigm es su capacidad para trabajar en equipo, lo que permite la colaboración de diferentes miembros del equipo de desarrollo en la creación y mantenimiento de los modelos de diseño. También permite la integración con otras herramientas de desarrollo de software, como IDEs y sistemas de control de versiones (Visual Paradigm, 2023).

1.4.2 Editor de código

Visual Studio Code es un editor de código fuente ligero pero potente que se ejecuta en su escritorio y está disponible para Windows, macOS y Linux. Viene con soporte incorporado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes y tiempos de ejecución (como C ++, C #, Java, Python, PHP, Go, .NET) (Microsoft, 2023).

- Es gratuito y de código abierto: Es un software gratuito y de código abierto, lo que significa que cualquier persona puede utilizarlo, modificarlo y distribuirlo sin costo alguno.

- Amplia gama de extensiones: Cuenta con una gran cantidad de extensiones que pueden ampliar sus capacidades y personalizar su entorno de desarrollo. Estas extensiones pueden ayudarlo a trabajar con diferentes lenguajes de programación, depurar su código, administrar sus repositorios de código fuente y mucho más.

- Integración con Git: Tiene una integración profunda con Git, lo que significa que puede administrar sus repositorios de código fuente directamente desde el editor. Esto le permite

realizar operaciones de Git como confirmaciones, ramificación y fusión dentro del editor sin tener que cambiar a una herramienta de línea de comandos.

- Depuración fácil y rápida: Ofrece una experiencia de depuración fácil y rápida con un depurador incorporado. Puede establecer puntos de interrupción, inspeccionar variables y evaluar expresiones en tiempo real mientras depura su código.

- Multiplataforma: Está disponible en varias plataformas, incluyendo Windows, macOS y Linux, lo que lo hace fácilmente accesible para cualquier desarrollador independientemente del sistema operativo que utilice.

1.4.3 Lenguaje de programación

Un lenguaje de programación es un medio de comunicación entre los programadores y las computadoras, utilizado para crear software, aplicaciones, sitios web, scripts y otras secuencias de instrucciones que pueden ser ejecutadas por los sistemas informáticos (García Avila, 2019). Dentro de los lenguajes de programación más usados son:

- **Java:** Java es un lenguaje de programación moderno y orientado a objetos que se usa ampliamente en varios campos, desde la educación hasta la industria. Es conocido por sus características, como la abstracción de datos, la programación multinúcleo, la gestión de subprocesos y la independencia de la plataforma, que lo diferencian de otros lenguajes. Java se usa a menudo en entornos educativos para enseñar conceptos de programación. Por ejemplo, un estudio diseñó una aplicación para enseñar programación Java usando el Marco de gamificación de octoálisis, que mostró impactos positivos en la participación y el aprendizaje de los estudiantes (Waworuntu, 2021).
- **C#:** Es un lenguaje de programación de alto nivel, orientado a objetos y multiplataforma, que se inspira en el lenguaje Java y forma parte del entorno .NET de Microsoft. Se usa para desarrollar aplicaciones web, móviles, de escritorio, juegos o servicios web. Es un lenguaje moderno y fácil de aprender, que ofrece muchas ventajas como la seguridad de tipos, la recolección automática de basura o el manejo de excepciones (Kodigo, 2023).
- **Python:** Es un lenguaje de programación versátil, de alto nivel y multiplataforma, conocido por su facilidad de uso, legibilidad y flexibilidad. Se utiliza para desarrollar una amplia gama de aplicaciones, como web, móviles, de escritorio, científicas, de inteligencia artificial y análisis de datos. Es altamente valorado tanto por las empresas como por los

desarrolladores debido a su popularidad y demanda en el mercado. Además, cuenta con una comunidad activa y una amplia variedad de bibliotecas y frameworks disponibles (Academia, 2022).

Tras evaluar varios lenguajes de programación, se optó por seleccionar Python en su versión 3.11 como el lenguaje para implementar la solución propuesta. Esta elección se basó en varias razones: Python es un lenguaje de alto nivel, gratuito y cuenta con una amplia gama de bibliotecas de código abierto. Además, cuenta con una comunidad de desarrolladores muy activa que ofrece apoyo tanto a principiantes que tienen dudas iniciales como a expertos que pueden compartir experiencias valiosas de otros desarrolladores con mayor experiencia. A continuación, se aborda más sobre este lenguaje:

Python es un lenguaje de programación multiparadigma, lo que significa que soporta varios estilos de programación, entre ellos la orientación a objetos, la programación imperativa, la programación orientada a aspectos y, en menor medida, la programación funcional. Al ser interpretado, los programas escritos en este se ejecutan directamente sin necesidad de compilar. Además, utiliza tipado dinámico, lo que permite asignar valores de diferentes tipos a una misma variable en tiempo de ejecución. Por otro lado, Python es un lenguaje fuertemente tipado, lo que significa que los tipos de datos de las variables deben ser definidos y respetados en tiempo de ejecución. Por último, este es multiplataforma, lo que significa que puede ser utilizado en diferentes sistemas operativos, incluyendo Windows, macOS, Linux y otros sistemas Unix (García Avila, 2022).

Ventajas del uso de Python:

- Versatilidad: Es un lenguaje de programación muy versátil que permite crear todo tipo de programas, desde aplicaciones web hasta análisis de datos y aprendizaje automático.
- Multiplataforma: Es compatible con una amplia variedad de sistemas operativos, incluyendo Windows, macOS, Linux y otros sistemas Unix, lo que lo convierte en un lenguaje de programación multiplataforma.
- Fácil de aprender: Es muy fácil de aprender gracias a su sintaxis clara y concisa, lo que permite a los principiantes comenzar a escribir programas en poco tiempo.

- Orientado a objetos: Es un lenguaje de programación orientado a objetos que proporciona una manera sencilla de crear programas con componentes reutilizables. Además, Python también permite la programación imperativa, programación funcional y programación orientada a aspectos.

- Amplia gama de bibliotecas: Cuenta con una amplia variedad de bibliotecas y módulos disponibles que permiten extender su funcionalidad básica a cualquier campo, desde la ciencia de datos hasta la inteligencia artificial y el aprendizaje automático (García Avila, 2022).

1.4.4 Lenguajes de desarrollo web

Adicionalmente se emplean otros lenguajes enfocados al desarrollo web como HTML, CSS y Javascript. A continuación, se abordará sobre estos lenguajes.

HTML: Es un lenguaje de marcado utilizado para crear y diseñar páginas web que permite presentar información de manera estructurada y atractiva. Este lenguaje permite la creación de enlaces o vínculos que llevan a otros recursos relacionados, como otras páginas web o archivos multimedia, lo que permite a los usuarios navegar fácilmente por diferentes fuentes de información. Además, HTML permite la inclusión de diferentes elementos multimedia, como imágenes, videos y sonidos, que enriquecen la presentación y la experiencia del usuario al interactuar con la página web (Casado Vara, 2019).

CSS (Hojas de Estilo en Cascada): Es un conjunto de reglas y estándares establecidos por el W3C para definir cómo se presenta el contenido en una página web creada con HTML o XHTML. Permite la definición del aspecto visual de los elementos individuales o grupos de elementos dentro de una página web en diferentes dispositivos y medios, como pantallas de computadoras, tabletas y dispositivos móviles (Collazo Baños, 2019).

Ventajas:

- Simplifica el código de las páginas web.

- Optimiza el rendimiento de los navegadores.

- Permite dar una apariencia homogénea a un sitio web al aplicar los mismos estilos a todas sus páginas.

- Amplía las posibilidades de presentación de HTML al permitir mucho más control.
- Permite que los usuarios con necesidades especiales creen sus propias hojas de estilo para ver el contenido según sus preferencias.

Javascript: Es un lenguaje de programación ligero que puede ser interpretado o compilado en tiempo real. Se considera un lenguaje de programación de "primera clase", lo que significa que es muy versátil y útil para una amplia variedad de aplicaciones. Aunque es conocido principalmente por su uso en la creación de secuencias de comandos para páginas web, se utiliza en muchos otros entornos, incluyendo Node.js, Apache CouchDB y Adobe Acrobat. JavaScript es un lenguaje de programación basado en prototipos, lo que significa que los objetos se crean a partir de otros objetos existentes. Es un lenguaje multiparadigma, lo que significa que admite varios estilos de programación, incluyendo programación orientada a objetos, programación imperativa y programación declarativa. También es un lenguaje dinámico, lo que significa que las variables no necesitan ser declaradas antes de su uso, y es un lenguaje de un solo hilo, lo que significa que solo puede ejecutar una tarea a la vez (Mozilla Corporation, 2023).

Tiene la gran ventaja de poder ser utilizado en cualquier página web sin necesidad de instalar otro programa adicional. Este lenguaje ofrece una amplia variedad de posibilidades, desde la creación de pequeños programas que se pueden integrar en páginas web hasta la creación de programas más grandes y complejos orientados a objetos. JavaScript es capaz de crear efectos visuales y de interactuar con los usuarios de una página web, lo que permite una experiencia de usuario más dinámica y atractiva. Por ejemplo, se pueden crear botones interactivos, animaciones, formularios dinámicos y otras características que mejoran la experiencia del usuario en la página web (Lima Torres, 2020).

1.4.5 Marco de trabajo para el desarrollo de la solución informática

Un framework es una estructura o conjunto de herramientas que proporciona una base para desarrollar un proyecto con objetivos específicos. Funciona como una plantilla que sirve como punto de partida para la organización y desarrollo de software. El uso de estos puede simplificar significativamente tareas y procesos, lo cual es especialmente beneficioso para los profesionales digitales, ya que les permite ser más ágiles y productivos. Por lo general, los programadores los utilizan para acelerar el trabajo, fomentar la colaboración, reducir errores y obtener resultados de mayor calidad. Sin embargo, no se limitan al departamento de tecnología de la información, ya

que en el ámbito digital existen frameworks para prácticamente todo: desde definir el proceso de compra de un cliente hasta implementar mejoras en productos digitales para aumentar las conversiones (Noa, 2022). Entre los frameworks disponibles que facilitan el desarrollo de esta aplicación de manera sólida y eficiente, se optó por utilizar los siguientes:

Django: Es el framework más utilizado para el desarrollo de aplicaciones web en Python. Este es ampliamente reconocido y adoptado en la comunidad de desarrollo debido a su enfoque en la productividad y la seguridad. Ofrece una amplia gama de características y funcionalidades, como un ORM (Object-Relational Mapping) integrado, un sistema de enrutamiento y vistas, autenticación de usuarios, administración de bases de datos y una gran cantidad de bibliotecas adicionales. Además, se destaca por su capacidad para manejar proyectos web de gran escala, gracias a su arquitectura escalable y su enfoque en la reutilización de código. Aunque existen otros frameworks populares en Python, como Flask y Pyramid, Django se considera el framework líder y preferido para el desarrollo de aplicaciones web en Python. Por esto para el desarrollo back-end, se utiliza Django, en la versión 4.2.0(Vincent, 2023):

- Abstrae a los programadores de los problemas comunes del desarrollo web y acelera las tareas más frecuentes en la programación (García Avila, 2022).
- Ofrece plantillas que ayudan a separar el contenido de la presentación y evitan manipular la lógica de negocio al realizar cambios en la apariencia de la página (García Avila, 2022).
- Incluye varias aplicaciones comunes a todos los sitios web, como la autenticación de usuarios y la administración del contenido del sitio (García Avila, 2022).
- Se basa en el patrón Modelo-Vista-Plantilla (MVT por sus siglas en inglés), el cual utiliza modelos como bases de datos, las vistas como controladoras y las plantillas como la comunicación interactiva con el usuario. Con su uso, toma menos tiempo el desarrollo de las aplicaciones web (Gore et al., 2021).

Bootstrap: Es el más popular Front-End Framework de diseño Responsive de código abierto creado por Mark Otto y Jacbod Thornton de Twitter, compuesto por HTML, CSS y JavaScript que sirve como estructura de inicio en la producción de aplicaciones web, simplificando este largo proceso y controlando la parte del Front en los sitios. Bootstrap integra una combinación entre CSS para el diseño y el estilo que se le puede dar a una página web teniendo como resultado una interfaz elegante e interactiva con el usuario. Bootstrap cuenta con una documentación

extensa y detallada, en la cual se encontrará ejemplos fáciles donde se entenderá el uso de los componentes y el diseño web. Además, por ser un framework altamente adoptado, se encontrará más documentación en foros, blogs donde serán los propios desarrolladores los que guíen en base a la experiencia adquirida (SARZOSA, 2018).

1.4.6 Gestor de bases de datos

El gestor de bases de datos a utilizar en la solución es SQLite3. Es el gestor de base de datos integrado en Django, que utiliza el lenguaje SQL para generar consultas (Django Software Foundation, 2023).

(González 2019) define que “SQLite es un sistema de gestión de bases de datos relacional, famoso por su pequeño tamaño. A diferencia de otros sistemas de gestión cliente-servidor el motor de SQLite no es un proceso independiente lo que hace que la latencia sea menor y el acceso más eficiente. Debido a su facilidad de uso, su pequeño tamaño y su versatilidad, SQLite es utilizado en una gran variedad de 16 aplicaciones, entre ellas Mozilla Firefox o Skype. Su uso ha sido muy popular en las aplicaciones para smartphones con sistema operativo Android y iOS” (Lima Torres, 2020).

SQLite es un sistema de gestión de bases de datos que se distingue por no necesitar un servidor para manipular las peticiones a la base de datos. Esto significa que se puede operar directamente desde el dispositivo donde se encuentra almacenada la base de datos, sin necesidad de una infraestructura de red o un servidor dedicado. Además, otra característica destacada de SQLite es que no requiere de administración, configuración o mantenimiento, lo que simplifica su uso y reduce la complejidad del proceso de gestión de la base de datos (SQLite Consortium, 2023). Se puede afirmar que SQLite es un gestor de bases de datos más rápido y ligero que MySQL y PostgreSQL. Además, es compatible con muchas plataformas y está disponible de forma gratuita debido a que es de dominio público. Esto significa que no hay costos de licencias o de uso, lo que lo hace atractivo para aquellos que buscan una solución de base de datos económica y fácil de implementar (Lima Torres, 2020).

Conclusiones parciales

- Con el análisis de los sistemas homólogos existentes se comprobó que ninguno de estos sistemas da solución al problema planteado, pero aportan información oportuna en cuanto a la tendencia y diseño del software.
- El estudio de la metodología permitió identificar el camino a seguir más eficiente en el desarrollo del software.
- Se identificó todas las herramientas a manejar como son: lenguaje Python 3.11 para el servidor con Django 4.2.5 cómo framework, HTML 5, CSS 3 y JavaScript como lenguajes del lado del cliente, Visual Paradigm for UML 8.0 para el modelado de los artefactos, Visual Studio Code 1.82.3 como IDE y SQlite3 para gestionar la base de datos.

Capítulo 2 - Características del sistema

Introducción

En el presente capítulo se detalla la propuesta del sistema. Además, se enumeran los requisitos funcionales y no funcionales del sistema. Se describe la solución propuesta para abordar un problema específico, junto con sus características. También se identifican los artefactos, como diagramas y documentos, que se crearán durante el análisis y diseño para apoyar el desarrollo de las nuevas funcionalidades que se van a agregar. Todo esto se hace utilizando una metodología específica, lo que ayuda a asegurar un enfoque estructurado y coherente para el desarrollo del proyecto.

2.1 Propuesta del sistema

Como propuesta de sistema se define un software que es para una plataforma web que permita a los usuarios gestionar sus listas de verificación de manera centralizada y cómoda. La plataforma permitirá a los usuarios crear y modificar listas de verificación, así como asignar las mismas a diferentes miembros del equipo. Además, contará con una serie de herramientas y opciones que permitirán a los usuarios trabajar de manera más eficiente y colaborativa, lo que resultará en una mayor productividad y mejores resultados. Entre las funcionalidades de esta plataforma se encuentran crear diferentes tipos de listas de verificación, que abarque cada una de las características de la calidad.

2.2 Product Backlog

El Product Backlog es una lista dinámica y ordenada que contiene las mejoras necesarias para el producto. Es una lista priorizada de todas las cosas que se necesitan hacer en un proyecto. Es la única fuente de trabajo para el equipo Scrum y representa los elementos que deben ser completados. Los elementos del Product Backlog que el equipo Scrum puede finalizar en un Sprint se consideran listos para ser seleccionados. Estos elementos se vuelven más claros y comprensibles después de pasar por el proceso de refinamiento. El refinamiento del Product Backlog implica descomponer y definir con mayor detalle los elementos, dividiéndolos en partes más pequeñas y precisas (Schwaber & Sutherland, 2020).

Durante el proceso de refinamiento del Product Backlog, se utiliza la buena práctica de levantamiento de requisitos, además de la técnica de las historias de usuario para desglosar y definir mejor los elementos.

2.2.1 Requisitos del sistema

Los requisitos para un sistema son las descripciones de lo que el sistema debe lograr, incluyendo los servicios que ofrece y las restricciones en su funcionamiento. Estos requisitos son el reflejo de las necesidades de los clientes en cuanto a lo que esperan del sistema. Una vez que se hayan realizado las descripciones de los procesos y se haya creado un modelo conceptual, es posible identificar y describir los requisitos funcionales del sistema. Estos requisitos funcionales se refieren a las acciones y comportamientos específicos que el sistema debe llevar a cabo para cumplir con las necesidades de los clientes (Echevarría Nieves, 2021).

2.2.2 Requisitos funcionales

En la tabla 2, se muestran numerados los requisitos funcionales del sistema. De cada uno se brinda una descripción seguido del nivel de prioridad en el sistema y la complejidad que conlleva su implementación. Estos requisitos funcionales se obtuvieron del estudio de sistemas homólogos y mediante la realización de una entrevista a los expertos de más de 10 años en la industria y en temas de calidad.

Tabla 2 - Requisitos funcionales del sistema

No.	Nombre	Descripción	Prioridad	Complejidad
RF 1	Adicionar pregunta	El sistema debe permitir adicionar una pregunta a la lista de verificación (etapas de desarrollo, tipos de proyecto, características, evaluar proceso o producto)	Alta	Alta
RF 2	Modificar pregunta	El sistema debe permitir modificar una pregunta de la lista de verificación	Alta	Alta

RF 3	Listar preguntas	El sistema debe permitir listar algunas o todas las preguntas de la lista de verificación	Media	Media
RF 4	Eliminar pregunta	El sistema debe permitir eliminar una o varias preguntas de la lista de verificación	Media	Baja
RF 5	Eliminar notificaciones	El sistema debe permitir eliminar todas las notificaciones	Media	Baja
RF 6	Registrar los resultados	El sistema debe guardar los resultados de las preguntas de las listas de verificación	Alta	Alta
RF 7	Adicionar usuario	El sistema debe permitir adicionar un nuevo usuario con los permisos correspondientes	Media	Media
RF 8	Modificar usuario	El sistema debe permitir modificar los datos y los permisos de un usuario existente	Media	Media
RF 9	Eliminar usuario	El sistema debe permitir eliminar un usuario	Media	Baja
RF 10	Listar usuarios	El sistema debe permitir listar algunos o todos los usuarios que existen	Media	Media
RF 11	Buscar usuario	El sistema debe permitir la búsqueda de un usuario específico dependiendo del usuario o nivel de permiso introducido	Media	Media
RF 12	Autenticar usuario	El sistema debe permitir al usuario autenticarse para poder operar en él	Alta	Media

RF 13	Adicionar producto	El sistema debe permitir adicionar un producto de software	Alta	Alta
RF 14	Modificar producto	El sistema debe permitir modificar los datos almacenados de un producto de software existente	Alta	Alta
RF 15	Eliminar producto	El sistema debe permitir eliminar un producto de software	Media	Baja
RF 16	Listar productos	El sistema debe permitir listar algunos o todos los productos de software existentes	Media	Baja
RF 17	Buscar producto	El sistema debe permitir la búsqueda de un producto de software específico dependiendo de un valor introducido	Media	Media
RF 18	Adicionar Lista de Verificación	El sistema debe permitir adicionar una lista de verificación	Alta	Alta
RF 19	Modificar lista de verificación	El sistema debe permitir modificar el nombre y el requisito de una lista de verificación	Media	Alta
RF 20	Eliminar lista de verificación	El sistema debe permitir la eliminación de la lista de verificación	Media	Baja
RF 21	Listar lista de verificación	El sistema debe permitir listar algunas o todas las listas de verificación	Media	Baja
RF 22	Buscar lista de verificación	El sistema debe permitir la búsqueda de una lista de verificación específica	Media	Baja

		dependiendo de un valor introducido		
RF 23	Asignar lista de verificación	El sistema debe permitir que un usuario con cierto rol asigne listas de verificación a otros usuarios	Alta	Alta
RF 24	Mostrar notificación de asignación	El sistema debe mostrar una notificación a un revisor cuando a este sea asignada una lista de verificación	Media	Baja
RF 25	Chequear listas de verificación asignadas	El sistema debe permitir revisar las listas de verificación que están siendo llenadas por los revisores	Media	Alta
RF 26	Cambiar contraseña	El sistema debe permitir a los usuarios cambiar su contraseña	Baja	Baja
RF 27	Crear nueva lista de verificación	El sistema debe permitir crear una lista de verificación nueva	Alta	Alta

2.2.3 Requisitos no funcionales

Requisitos de usabilidad

En la tabla 3, se muestran los requisitos de usabilidad recomendados para el sistema. Entre los datos se encuentran la edad de los usuarios que usarán el sistema, su ocupación, el tipo de discapacidad que puedan poseer, la experiencia con la aplicación informática y la experiencia profesional que se sugiere que posean los mismos.

Tabla 3 - Requisitos de usabilidad del sistema

#	Sexo	Edad	Nivel de escolaridad	Ocupación	Experiencia profesional	Experiencia con la aplicación informática	Tipo de discapacidad	Otras
1	M, F	25 - 60	Universit	Profesor	2 años	1 mes	ninguna	

			ario					

- Tipo de aplicación informática: aplicación web.
- Finalidad: sistema para gestionar las listas de verificación en la UCI.
- Ambiente: la computadora personal debe contar con un procesador dual – Core o superior, con una memoria RAM de 2 Gigabytes como mínimo. Debe ser posible su ejecución sobre los sistemas operativos Windows y Linux, con un tiempo de respuesta inferior a los 4 segundos.

RnF 1. Accesibilidad – El sistema debe permitir el acceso del usuario de acuerdo con su rol.

Requisito de Rendimiento

RnF 2. Tiempo de respuesta – El sistema debe responder en 2 segundos a cada operación.

Requisitos de Software

RnF 3. Plataforma – El sistema debe ser multiplataforma.

RnF 4. Programa – El sistema debe ser ejecutado en un navegador web como Mozilla Firefox, Safari o Google Chrome.

Requisitos de Hardware

RnF 5. Microprocesador – El sistema debe ejecutarse en una computadora con un procesador dual – Core o superior.

RnF 6. Memoria RAM – El sistema debe ejecutarse en una computadora con una memoria RAM mínima de 2 Gigabytes.

Requisito de Seguridad

RnF 7. Autenticación – El sistema debe permitir acceso a los usuarios registrados en la base de datos y que se autentiquen correctamente.

Requisitos de Restricciones del diseño y la implementación

RnF 8. Base de datos – El sistema debe utilizar como base de datos SQlite 3.

RnF 9. Experiencia de usuario – El sistema debe ser diseñado de manera simple e intuitiva.

RnF 10. Uso de nomenclatura – El sistema debe programarse utilizando la nomenclatura *camelCase*.

Requisito de Apariencia o Interfaz externa

RnF 11. Letra – El sistema debe ser implementado con una fuente entendible para las personas con problemas de accesibilidad como personas con mala visión, con un trazado sencillo y legible.

2.2.4 Planificación de sprint

Los sprints son el núcleo central de la metodología Scrum, ya que permiten convertir ideas en valor tangible. Son eventos de duración establecida, generalmente de un mes o menos, que brindan coherencia y enfoque al equipo. Un nuevo sprint comienza inmediatamente después de la finalización del sprint anterior. Durante los sprints, todo el trabajo necesario para alcanzar los objetivos del producto se lleva a cabo, incluyendo la planificación del sprint, las reuniones diarias de seguimiento (daily scrums), la revisión del sprint y la retrospectiva del sprint. Estos eventos se realizan dentro de los sprints con el fin de maximizar la eficiencia y la entrega de valor al cliente (Schwaber & Sutherland, 2020). A continuación, se muestran las tablas 4 y 5 donde se representa la planificación de los dos sprints llevados a cabo:

Tabla 4 - Planificación del Sprint 1. Fuente: Elaboración propia

ID	***Tareas***	***Estado***	***Tiempo estimado***	***Tiempo Inicial***	***Tiempo final***
T1	Reunión de planificación conjunta entre el equipo de desarrollo y el propietario del producto.	Resuelta	48h	2/05/2023	4/05/2023
T2	Configurar el ambiente de desarrollo con la integración de los marcos	Resuelta	48h	4/05/2023	6/05/2023

	de trabajo seleccionados para el desarrollo.				
T3	Implementar los CRUD Gestionar listas de verificación y Gestionar usuarios.	Resuelta	96h	7/05/2023	11/05/2023
T4	Implementar los CRUD Gestionar productos y Gestionar preguntas.	Resuelta	120h	12/05/2023	17/05/2023
T5	Implementar funcionalidades: Cambiar contraseña y mostrar notificación de asignación.	Resuelta	24h	18/05/2023	19/05/2023
T6	Revisar el sprint	Resuelta	24h	19/05/2023	20/05/2023

Tabla 5 - Planificación del Sprint 2. Fuente: Elaboración propia

ID	***Tareas***	***Estado***	***Tiempo estimado***	***Tiempo Inicial***	***Tiempo final***
T1	Reunión de planificación conjunta entre el equipo de desarrollo y el propietario del producto.	Resuelta	24h	20/05/2023	21/05/2023

T2	Implementar funcionalidad: Registrar resultados.	Resuelta	24h	22/05/2023	23/05/2023
T3	Implementar funcionalidad: Crear una nueva lista de verificación	Resuelta	24h	23/05/2023	24/05/2023
T4	Implementar funcionalidad: Chequear lista de verificación	Resuelta	24h	24/05/2023	25/05/2023
T5	Revisar el sprint	Resuelta	24h	25/05/2023	26/05/2023

2.2.5 Historias de usuario

Cuando se realiza la especificación de requisitos, la metodología Scrum requiere utilizar historias de usuario, una técnica para encapsular los requisitos funcionales del sistema. Con esta, se describe la salida esperada del sistema y cómo se ve beneficiado el usuario. Se redacta en un lenguaje natural y sencillo para evitar confusiones con los usuarios finales, ya que estos no dominan el lenguaje técnico. A continuación, la tabla 6 - Historia de Usuario número 13 y la tabla 7 - Historia de Usuario número 10 detallan a los requisitos Adicionar producto y Listar usuario. El resto de las historias de usuario se encuentran en el anexo 2.

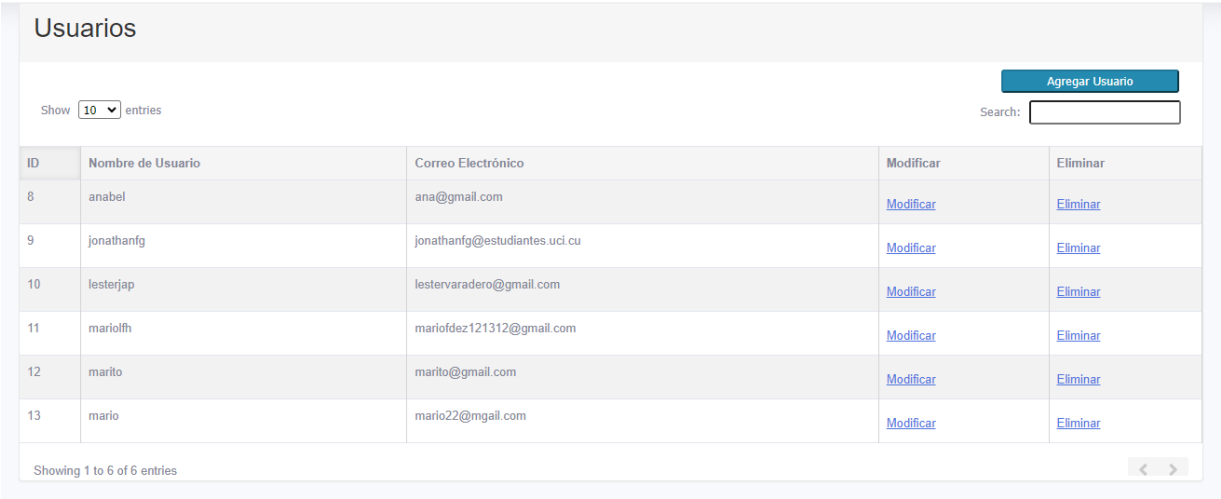
Tabla 6 - Historia de Usuario número 13

Historia de Usuario	
Número: 10	Nombre: Adicionar Producto
Usuario: Revisor jefe	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alto

Puntos estimados: 1	Iteración asignada: 2
Programador responsable: Daniela Oramas Romero	
Descripción: Al autenticarse el usuario, en la interfaz principal selecciona la opción Productos, se pulsa el botón Agregar Producto. Se muestra una ventana con los campos Nombre del Producto, Versión, Fecha Límite, Centro de Origen e Iteración, seguido por los botones aceptar y cancelar.	
Observaciones: - Si los datos están incompletos o incorrectos se señalan los campos en cuestión dando la posibilidad al revisor jefe de realizar nuevamente la acción en cuestión.	
Interfaz: 	

Tabla 7 - Historia de Usuario número 10

Historia de Usuario

Número: 10	Nombre: Listar Usuario
Usuario: Revisor jefe	
Prioridad en negocio: Media	Riesgo en desarrollo: Media
Puntos estimados: 0.5	Iteración asignada: 1
Programador responsable: Daniela Oramas Romero	
Descripción: Al autenticarse el usuario, en la interfaz principal selecciona la opción Usuarios, y a continuación se muestra una vista con un listado de todos los usuarios y con los campos ID, Nombre de Usuario, Correo Electrónico y las opciones eliminar, modificar y adicionar.	
Observaciones: Se muestra toda la información de los usuarios.	
Interfaz: 	

2.3 Modelado de datos

Un modelo de datos es un esquema que detalla las expresiones permitidas por el propio modelo, anuncia las reglas y definiciones esenciales de los datos a los usuarios. Describe la semántica a través de tablas representadas por una tecnología de manipulación de datos tal como el lenguaje SQL. Lo conciertan entidades, atributos y relaciones donde se pueden realizar un conjunto de operaciones que permiten especificar consultas y actualizaciones de la base de datos (Ramírez Carreño, 2022). A continuación, se muestra la figura 1 que corresponde al modelado de datos.

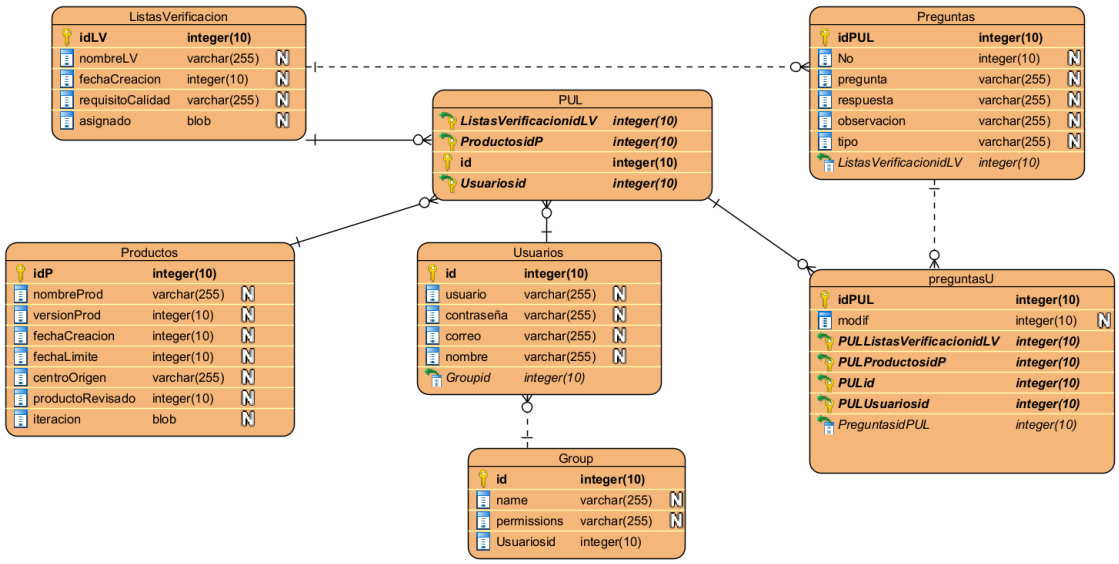


Figura 1 - Modelo de datos. Fuente: Elaboración propia

El modelo anterior representa visualmente la información que se registra en el sistema, así como las conexiones que existen entre los datos. Muestra cómo se almacenarán los datos del negocio de manera duradera en la base de datos.

2.4 Diseño de la propuesta de solución

En el desarrollo de esta disciplina, se crea un modelo del sistema que considera su arquitectura para asegurar que cumpla con los requisitos funcionales y no funcionales. Este proceso permite que los elementos modelados sirvan como base para la etapa de implementación.

2.4.1 Diseño de la arquitectura

Para garantizar la calidad de un software, es importante establecer una arquitectura desde el inicio que describa los principios fundamentales del sistema y garantice su robustez y escalabilidad. La arquitectura de software define la estructura del sistema, compuesta por componentes con funciones específicas que interactúan entre sí (García Avila, 2022).

2.4.2 Arquitectura de software

La arquitectura de software proporciona una visión abstracta de alto nivel de los componentes y la relación entre ellos, lo que permite abordar la reutilización y la evolución del código. Además, hay enfoques de desarrollo de software que se centran en mapear la idea del dominio del negocio en los artefactos del software, lo cual tiene un valor estratégico al identificar el problema relevante y construir arquitecturas que conecten la implementación con un modelo en evolución de la idea principal del negocio (Cambarieri et al., 2020).

El sistema se diseñará e implementará siguiendo una arquitectura cliente/servidor, un enfoque ampliamente reconocido y utilizado en el desarrollo de aplicaciones. Esta es un modelo en el que los dispositivos o procesos en una red se dividen en dos roles: cliente y servidor. En este modelo, los clientes envían solicitudes a los servidores, que a su vez proporcionan los recursos o servicios solicitados. Un ejemplo común de esta arquitectura son las aplicaciones web, donde los navegadores actúan como clientes que solicitan y reciben contenido de los servidores web (Roa Bánquez et al., 2019).

Es esencial comprender las características fundamentales de las arquitecturas cliente/servidor antes de aprender a desarrollar aplicaciones web. En este modelo, los clientes son responsables de iniciar las solicitudes y presentar la interfaz de usuario, mientras que los servidores se encargan de procesar las solicitudes, administrar los datos y proporcionar respuestas a los clientes. Esta arquitectura ofrece varias ventajas, como (Roa Bánquez et al., 2019):

- La capacidad de separar las funciones de la aplicación según su servicio, lo que permite ubicar cada función en la plataforma más adecuada para su ejecución.
- También facilita la modularidad y el desarrollo de componentes reutilizables.
- Aprovecha las redes de ordenadores para permitir que múltiples procesadores ejecutan partes distribuidas de una misma aplicación, lo que permite la concurrencia de procesos.
- Migrar aplicaciones de un procesador a otro con cambios mínimos en los programas.
- Ofrece escalabilidad a las aplicaciones.

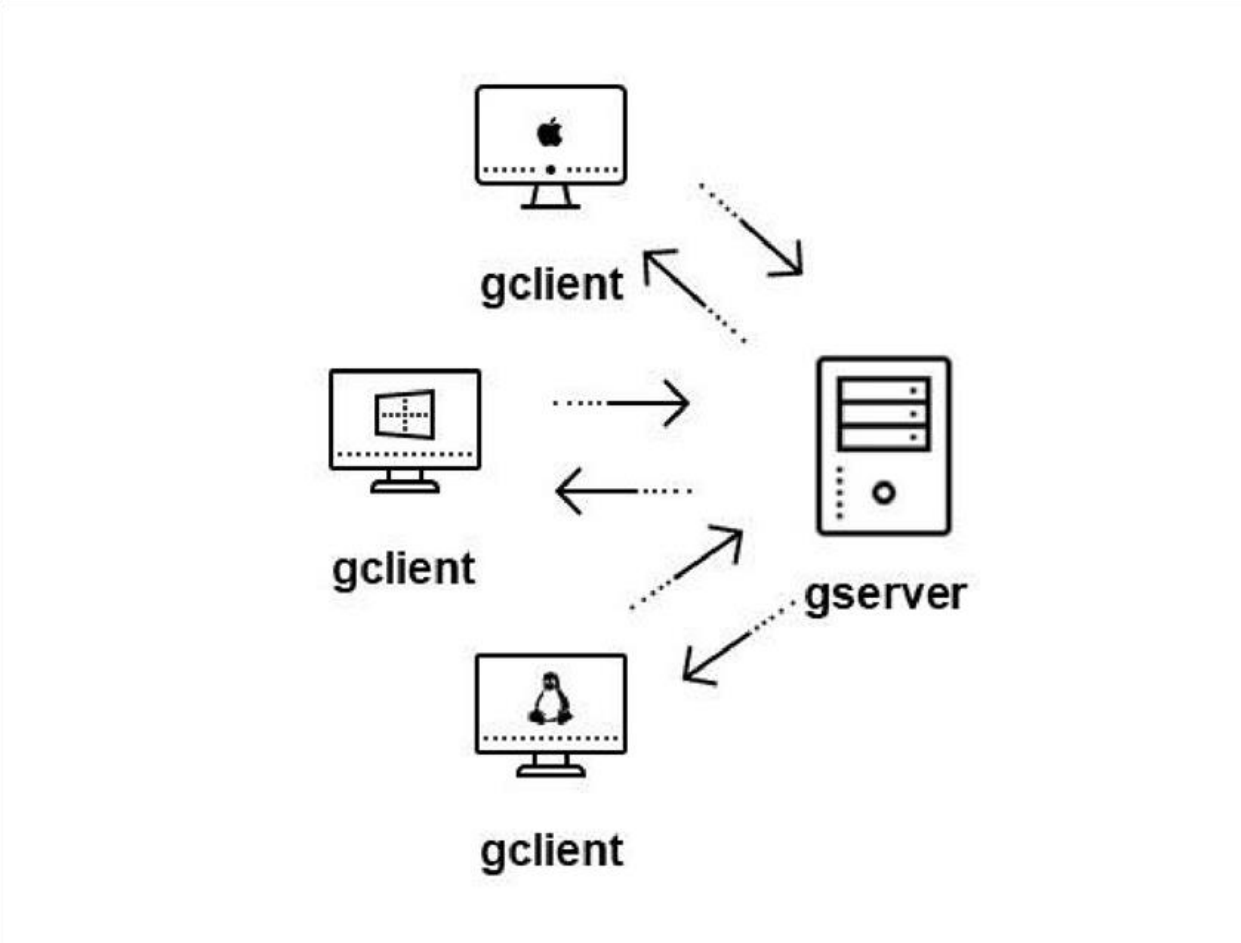


Figura 2 - Arquitectura basada en el estilo arquitectural Cliente-Servidor. Fuente: Elaboración propia*, con íconos disponibles libre de costo (Icons8 LLC, 2023)*

2.4.3 Arquitectura de software

La arquitectura de software proporciona una visión abstracta de alto nivel de los componentes y la relación entre ellos, lo que permite abordar la reutilización y la evolución del código. Además, hay enfoques de desarrollo de software que se centran en mapear la idea del dominio del negocio en los artefactos del software, lo cual tiene un valor estratégico al identificar el problema relevante y construir arquitecturas que conecten la implementación con un modelo en evolución de la idea principal del negocio (Cambarieri et al., 2020).

2.4.4 Patrón arquitectónico

Para el modelado del sistema se utilizará el patrón arquitectónico Modelo-Vista-Plantilla (MVT), tal como lo define Django como framework de desarrollo. Este patrón permite organizar el

sistema en paquetes individuales, sin que existan dependencias entre el modelo y las plantillas (García Avila, 2022).

El patrón arquitectónico Modelo-Vista-Plantilla (MVT) es una especificación del patrón Modelo Vista Controlador (MVC). En este patrón (García Avila, 2022):

- La capa Modelo: Es responsable de acceder a la base de datos y contiene toda la información relacionada con los datos, como su acceso, validación, comportamiento y relaciones.
- La capa Vista: Es la capa de lógica de negocios, encargada de acceder al modelo y delegar su funcionalidad a la plantilla adecuada.
- La capa Template: Es la capa de presentación y se enfoca en la apariencia visual de la página web o documento.

Para representar la arquitectura del sistema MVT, se utiliza un diagrama de paquetes que agrupa elementos relacionados de manera semántica y muestra las dependencias entre ellos. Este diagrama que se muestra a continuación visualiza la estructura del sistema y cómo los diferentes componentes se interconectan para cumplir con los requisitos del mismo.

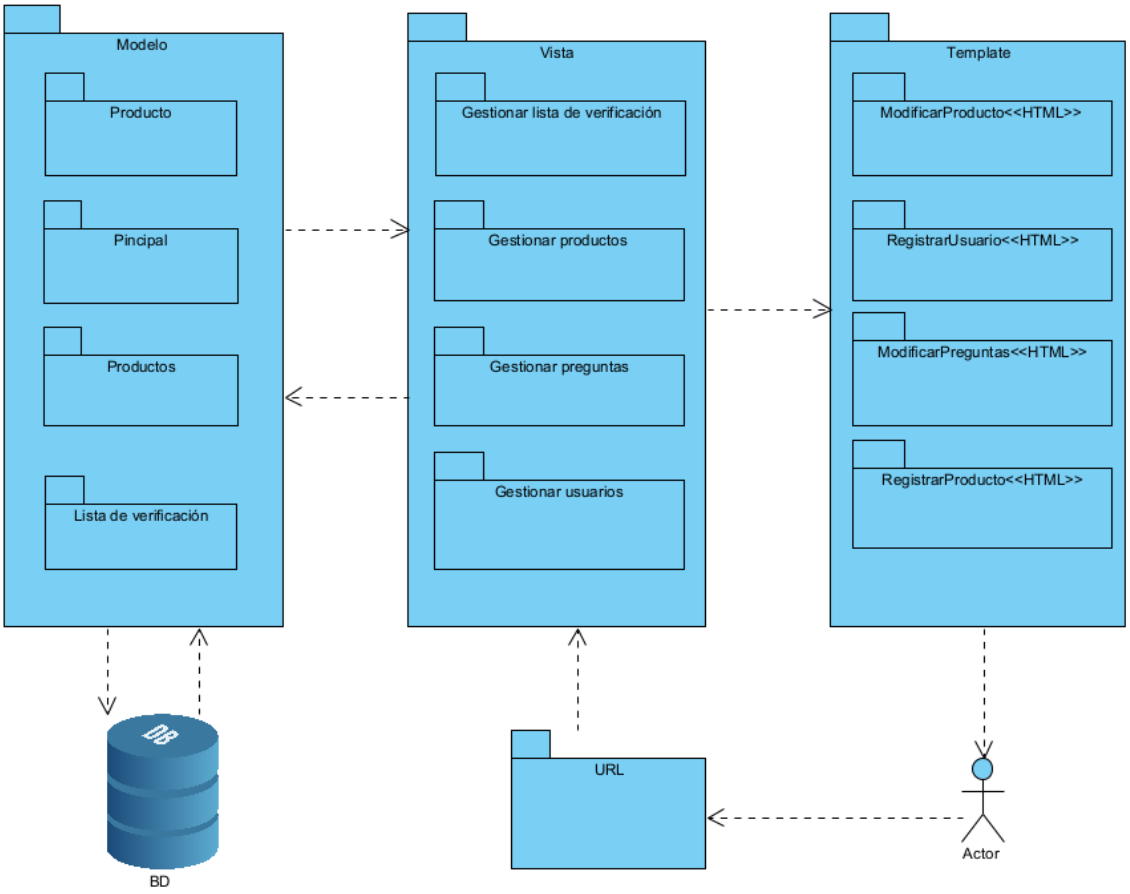


Figura 3 - Diagrama de paquetes. Modelado de la arquitectura. Fuente: Elaboración propia

2.5 Patrones de Diseño

Los patrones de diseño son soluciones genéricas a problemas comunes en el desarrollo de software que han sido probadas y validadas por la comunidad de desarrolladores de software. Proporcionan una forma estructurada y consistente de resolver problemas de diseño y promueven la flexibilidad, la elegancia y la reutilización del código. Para lograr un mejor diseño de la solución, se utilizan patrones de diseño específicos que son apropiados para el problema en cuestión. Estos facilitan una guía para la implementación del código, la estructura de las clases y objetos, la interacción entre ellos y la organización del sistema. Al seguirlos, se puede mejorar la calidad del diseño de la solución y reducir los errores y problemas de mantenimiento del código (García Avila, 2022).

2.5.1 Patrones GOF (Gang of Four)

Se hicieron populares en la industria del software gracias al libro "Design Patterns: Elements of Reusable Object-Oriented Programming", escrito por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides en 1995. Ofrecen estructuras de clases e interfaces para resolver problemas comunes de diseño de software de manera escalable, flexible y rápida. Los patrones GoF proporcionan una serie de soluciones genéricas para problemas comunes en el desarrollo de software, lo que permite a los desarrolladores ampliar su vocabulario de diseño y aprender nuevos estilos de programación orientada a objetos. Estos se enfocan en la estructura y organización de las clases y objetos en un sistema y están diseñados para ser reutilizables en diferentes contextos y aplicaciones. Al aplicarlos, se pueden desarrollar aplicaciones más robustas, escalables y flexibles en menos tiempo y con menos errores (Grau Rodríguez, 2019).

Según los autores, los patrones GoF se dividen en tres grupos principales (Grau Rodríguez, 2019):

Patrones de diseño estructurales: Estos patrones se enfocan en el proceso de instanciación de objetos y proporcionan la ventaja de depender más en la composición de objetos que en la herencia de clases. se utilizó el siguiente tipo de patrón estructural (Refactoring.Guru, 2023):

Decorator: Es una técnica para añadir funcionalidades a objetos existentes, sin modificar su estructura interna. Esto se logra colocando el objeto dentro de uno o más objetos encapsuladores especiales que contienen las funcionalidades adicionales. Este método pertenece a la clase abstracta View, padre de todas las vistas, que contienen un decorador para permitir agregar funcionalidades dinámicamente, se empleó para garantizar que los usuarios estén autenticados y puedan acceder a la información del sistema.

```python
@method_decorator(user_passes_test(lambda u: u.is_authenticated and u.is_staff), name = 'dispatch')
class listarUsuario(ListView):
    model = User
    template_name = 'ListarUsuarios.html'
    def get_context_data(self, **kwargs):
        return super().get_context_data(**kwargs)
```

Figura 4 - Decorador implementado sobre la clase listarUsuario

Patrones de diseño de comportamiento: Estos patrones se enfocan en facilitar el diseño de sistemas, identificando formas sencillas de relacionar entidades. Describen maneras de

componer objetos para agregar nuevas funcionalidades a dichos objetos, de los cuales se utilizó (Refactoring.Guru, 2023):

Template Method: Es una técnica de diseño de comportamiento que permite definir el esqueleto de un algoritmo en una clase base, pero permite que las subclases proporcionen implementaciones específicas de ciertos pasos del algoritmo sin cambiar su estructura general.

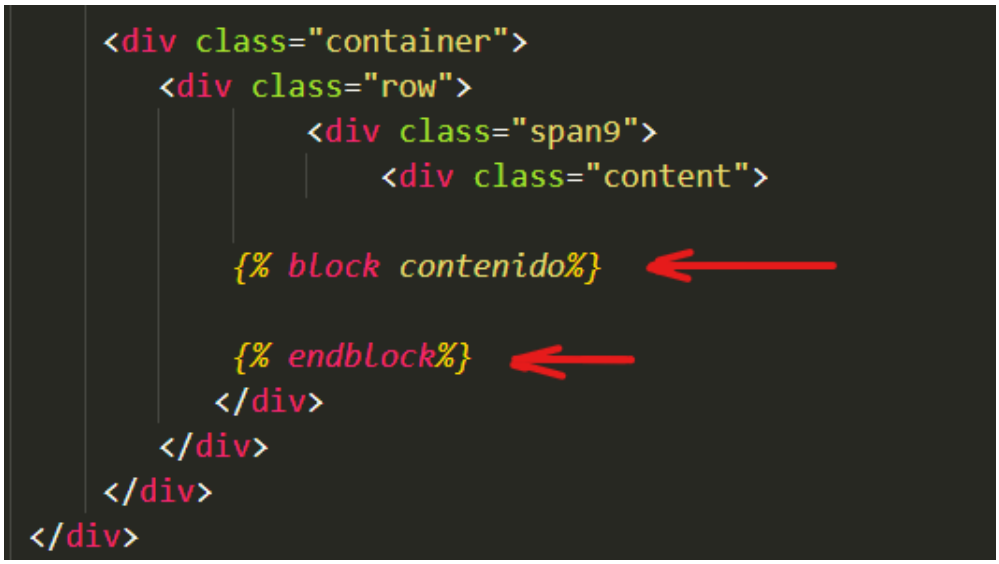


Figura 5 - Etiquetas Block en un archivo HTML, indicando a Django que inserte otro archivo HTML en ese espacio mediante el Template Method

Patrones de diseño creacionales: Estos patrones se enfocan en la creación de objetos y proporcionan soluciones para crear objetos de manera más flexible y eficiente. Se enfocan en cómo se inicializan y crean los objetos, lo que permite a los desarrolladores crear objetos de manera más fácil y rápida, de los diferentes tipos de patrones creacionales que existen se trabajó (Refactoring.Guru, 2023):

Builder: Es una técnica de diseño creacional que nos permite construir objetos complejos paso a paso, de manera que podemos producir distintos tipos y representaciones del objeto utilizando el mismo código de construcción.

```python
class producto(models.Model):
    idP = models.AutoField(auto_created=True,  primary_key=True, serialize=False, verbose_name='ID')
    nombreProd =  models.CharField(max_length=40)
    versionProd = models.CharField(max_length=40)
    fechaCreacion =  models.DateTimeField(auto_now_add=True)
    fechaLimite = models.DateField()
    centroOrigen = models.CharField(max_length=700)
    productoRevisado = models.BooleanField()
    iteracion = models.IntegerField()
```

Figura 6 - Modelo de la clase Producto, donde Django usa Builder automáticamente

2.5.2 Patrones GRASP

Los patrones GRASP General Responsibility Assignment Software Patterns (Patrones Generales de Software para Asignar Responsabilidades en sus siglas en inglés) describen los principios fundamentales para asignar responsabilidades a los objetos. Luego de un estudio de la bibliografía, y de acuerdo con lo expresado por autores como Larman (1998) y Duncan (2012), existen 9 patrones GRASP, entre los que se encuentran los mostrados a continuación.

- Experto: se encarga de asignar las responsabilidades necesarias al experto en la información, ósea, a la clase que cuenta con toda la información necesaria para cumplir con dichas responsabilidades. Se ve evidenciado en la clase Producto()
- Bajo Acoplamiento: se encarga de conceder las responsabilidades de tal forma que las clases sean capaces de comunicarse con el menor número de clases posible. Se evidencia un bajo acoplamiento en los modelos producto(), listaVerificacion(), preguntas(), PUL() y preguntasU(). Esto se debe a que estos modelos parecen tener una dependencia mínima entre sí y están enfocados en representar entidades o conceptos específicos.
- Alta Cohesión: se encarga de conceder a las clases responsabilidades que operen en una misma área de la aplicación y que no sean muy complejas. Se utiliza en la vista listarPreguntas() para mantener la responsabilidad y la funcionalidad relacionadas en un solo lugar. Esto ayuda a mejorar la legibilidad, el mantenimiento y la escalabilidad del código.

La utilización de estos patrones resultó en un diseño sencillo, pensado para la creación de un repositorio robusto, fácil de entender, mantener y ampliar; aumentando la capacidad de reutilización de sus componentes y conservando el encapsulamiento de la información.

Conclusiones parciales

Después de presentar la propuesta de solución derivada de la investigación, se han extraído las siguientes conclusiones:

- La definición de los requisitos funcionales permitió identificar las funcionalidades clave que el sistema debe ofrecer, brindando una visión clara de las acciones y características necesarias para cumplir con los objetivos del proyecto.
- La arquitectura modelo-vista-template y los patrones de diseño establecieron la estructura y las pautas necesarias para construir la solución de manera sólida y coherente.
- El diseño del modelo de datos describe la organización y estructura de la base de datos, lo cual garantizó una gestión adecuada y eficiente de los datos.

Capítulo 3 - Implementación y Pruebas

Introducción

En este capítulo, se llevará a cabo la implementación de la solución propuesta, es decir, convertir el diseño en un sistema funcional. Este se centrará en los elementos claves del proceso de pruebas. Se establecerá la estrategia de pruebas, se abordan los tipos de pruebas realizadas y se analizarán los resultados obtenidos de las mismas para evaluar la calidad de la solución propuesta.

3.1 Estándares de codificación

Un estándar de codificación abarca todos los aspectos relacionados con la creación del código fuente de un software. Su objetivo principal es establecer directrices que aseguren que la estructura del código sea uniforme y altamente legible (Rodríguez, 2020). Para poder desarrollar de manera legible y ordenada al implementar un software de aplicación, es necesario aplicar algunos estándares en la nomenclatura de la codificación al momento de programar. Uno de los estilos en la nomenclatura de código de programación a usar en este sistema es el Snake Case el cual se caracteriza por que cada una de las palabras se separa por un guion bajo (_). Es común en los nombres de variables y funciones de lenguajes como C, aunque también Ruby y Python lo adaptaron (Cotrina De La Cruz & Marquez Zavaleta, 2021). A continuación, se explican y ejemplifican algunas de las reglas de este estándar de codificación utilizados en la propuesta de solución.

• La declaración de importación debe escribirse en líneas separadas y colocarse al principio del archivo (Rodríguez, 2020):

```
from django.contrib import admin
from django.urls import path, include
from  Web import views
from django.contrib.auth.views import LogoutView
```

Figura 7 - Importación de Django. Fuente: Elaboración propia

• Nombres de funciones y miembros de clases: Las funciones deben nombrarse con letras minúsculas y las palabras separadas por un guion bajo (_) (Rodríguez, 2020):

```python
class agregarUsuario(CreateView):
    model = User
    template_name = 'agregar.html'
    form_class = agregarUserForm
    success_url = reverse_lazy('Usuarios')
    def form_valid(self, form):
        form.instance.email = self.request.POST.get("email")
        return super().form_valid(form)
    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['Elemento'] = 'Usuario'
        return context
```

Figura 8 - Clase agregarUsuario. Fuente: Elaboración propia

• Los nombres de los atributos, variables y parámetros tendrán todas las letras en minúsculas y usarán el guion bajo como delimitador entre palabras (Rodríguez, 2020):

```python
model = producto
template_name = 'ListarProductos.html'
```

Figura 9 - Variables. Fuente: Elaboración propia

3.2 Pruebas de software

Las pruebas de software son un conjunto de actividades que buscan evaluar y verificar la calidad, funcionalidad y rendimiento de un producto o aplicación de software (IBM, 2023). Estas permiten identificar errores, defectos y problemas de funcionamiento en las aplicaciones, así como verificar si cumplen con los requisitos y expectativas del usuario. Existen diferentes tipos de pruebas de software, como las funcionales, las de rendimiento, las de usabilidad, las de integración, entre otras (Serna López et al., 2020). Además, pueden ser manuales, donde un tester interactúa directamente con la aplicación, o automatizadas, donde se utilizan herramientas y marcos de trabajo para ejecutarlas de manera automática y reducir la intervención humana en tareas repetitivas o complejas (Gordillo Agudelo et al., 2021).

3.2.1 Estrategia de pruebas

Una estrategia de prueba de software es un plan detallado que integra diversas técnicas de diseño de casos de prueba en una serie de pasos bien organizados. Es una táctica que

proporciona una estructura clara para la prueba del software, incluyendo la planificación, diseño, ejecución y evaluación de las pruebas. La estrategia actúa como un mapa que guía y define cuándo y cómo se deben llevar a cabo los diferentes pasos de la prueba, así como la cantidad de esfuerzo, tiempo y recursos necesarios. Esta garantiza una prueba exhaustiva y eficiente del software, asegurando su calidad y rendimiento. El objetivo de esta es asegurar una construcción adecuada del software (Cremé Zerquera, 2022). En la tabla 8, que se presenta a continuación se muestra la estrategia de prueba que se trazó para la validación de este sistema:

Tabla 8 - Tipos y niveles de Pruebas

Niveles	Tipo de Prueba	Objetivo
Componente o unidad	Pruebas estructurales o unitarias	● Detectar defectos. ● Comprobar si una unidad, es decir un módulo, función o método funciona como debería.
Sistema	● Pruebas funcionales ● Pruebas no funcionales ● Pruebas asociadas al cambio	● Detectar defectos. ● Asegurar que los requisitos para los que se crea el producto se cumplen.
Aceptación	● Funcionalidades (historias de usuario) ● Pruebas funcionales(sistema)	● El cliente acepte el producto. ● Validar con el cliente el cumplimiento de los requisitos del sistema.

3.2.2 Pruebas funcionales

Las pruebas funcionales son un tipo de pruebas que se realizan en software para evaluar su funcionalidad y asegurar su calidad. Estas pueden ser de diferentes tipos, como unitarias, de integración, de regulación, entre otras. Se realizan para reducir los errores en el software y se aplica para verificar si este satisface las expectativas del usuario. Para realizarlas oficialmente, es necesario contar con una metodología y herramientas adecuadas que permitan validar los requisitos funcionales, corregir los errores y mejorar la calidad. Además, existen herramientas libres y de pago para realizarlas en el software (Colorado Rivera, 2020).

El proceso para ejecutar este tipo de pruebas es el siguiente (García Avila, 2022):

- Analizar los requisitos y sus especificaciones.

• Seleccionar entradas válidas y no válidas de acuerdo con las especificaciones.

• Determinar las salidas esperadas para cada entrada.

• Diseñar los casos de pruebas con las entradas seleccionadas.

• Ejecutar los casos de prueba.

• Comparar las salidas encontradas con las salidas esperadas.

• Determinar si el funcionamiento del software en prueba es apropiado.

Se mostrará a continuación la tabla 9 que describe el diseño de casos de pruebas de la funcionalidad Gestionar Usuario y la tabla 10 que contiene las descripciones de las variables de las mismas.

Tabla 9 - Diseño de casos de prueba de la funcionalidad Gestionar Usuario, en su sesión "Registrar Usuario"

Sección 1: Registrar Usuario							
Id del escenario	**Escenario**	**Usuario**	**Contraseña**	**Rectificar Contraseña**	**Correo**	**Respuesta del sistema**	**Resultado de la prueba**
EC 1.1	Entrada de datos válidos	V	V	V	V	Se crea un nuevo usuario y muestra el mensaje "Registro satisfactorio"	Satisfactorio
		daniela or	micontrase ña1234	micontras eña1234	danielaor@ gmail.com		
EC 1.2	Entrada de datos inválidos	I	V	V	V	No se crea el usuario, señala el campo con datos inválidos y muestra el mensaje "Utiliza un formato que coincida con el solicitado".	Satisfactorio
		*****	micontrase ña1234	micontras eña1234	danielaor@ gmail.com		

EC 1.3	Campo vacío	V	N/A	V	V	No se crea el usuario, señala el campo con datos inválidos y muestra el mensaje "Por favor, utilice el formato correcto"	Satisfactorio
		daniela or		micontraseña1234	danielaor@gmail.com		
EC 1.4	Usuario con longitud menor a 4 caracteres	I	V	V	V	No crea el usuario y muestra el mensaje "Aumenta la longitud de este texto a 4 caracteres o más (actualmente, el texto tiene X caracteres)"	Satisfactorio
		mar	micontraseña1234	micontraseña1234	danielaor@gmail.com		
EC 1.5	Contraseña con longitud menor a 8 caracteres	V	I	I	V	No crea el usuario y muestra el mensaje "Aumenta la longitud de este texto a 8 caracteres o más (actualmente, el texto tiene X caracteres)"	Satisfactorio
		daniela or	ola	ola	danielaor@gmail.com		
EC 1.6	Usuario Existente	V	V	V	V	No crea el usuario y muestra el mensaje "El usuario ya existe"	Satisfactorio
		daniela or	micontraseña1234	micontraseña1234	danielaor@gmail.com		

EC 1.7	"Rectificar contraseña" distinto a "contraseña"	V	V	I	V	No crea al usuario y muestra el mensaje "Las contraseñas no son iguales"	Satisfactorio
		daniela or	micontrase ña1234	micontras eña1243	danielaor@ gmail.com		
EC 1.8	Se cancela la operación	N/A	N/A	N/A	N/A	No crea el usuario, se ocultan los campos para crear un usuario y se ve solamente el listado de usuarios	Satisfactorio

Tabla 10 - Descripción de variables

No.	Nombre del campo	Clasificación	Valor Nulo	Descripción
1	Usuario	Campo de texto	No	Permite solo caracteres (a-z, 0- 9). No se permiten cadenas compuestas. Solamente por los caracteres (., ¿), estos no se permiten al inicio y al final de la cadena, ni uno al lado del otro.
2	Contraseña	Campo de texto	No	Permite todos los caracteres.
3	Rectificar contraseña	Campo de texto	No	Permite todos los caracteres.
4	Correo	Campo de texto	No	Tiene que tener obligatoriamente un (@) luego de esto el dominio.

Resultado de las pruebas funcionales

Las pruebas funcionales se centran en probar el comportamiento de un sistema o una solución en relación con los requisitos funcionales establecidos. Estas pruebas implican diseñar casos de prueba que representen diferentes escenarios y situaciones que podrían ocurrir en el uso real del sistema. En cada caso de prueba, se utilizan datos válidos e inválidos como entrada para evaluar cómo el sistema responde y si cumple con los requisitos establecidos. Se seleccionan cuidadosamente los valores de entrada para abarcar una amplia gama de posibilidades, pero sin

crear un número excesivo de casos de prueba. En la primera iteración se detectaron 14 no conformidades, de ellas 9 no conformidades son de error de idioma, que fueron resueltas a través de la traducción de las interfaces y 5 no conformidades de modificar el usuario, resueltas creando una nueva clase en el formulario para el usuario. En la segunda iteración se encontró un total de 5 no conformidades de error de idioma. Todas las no conformidades encontradas se solucionaron como se muestra en el gráfico 10 que se muestra a continuación:

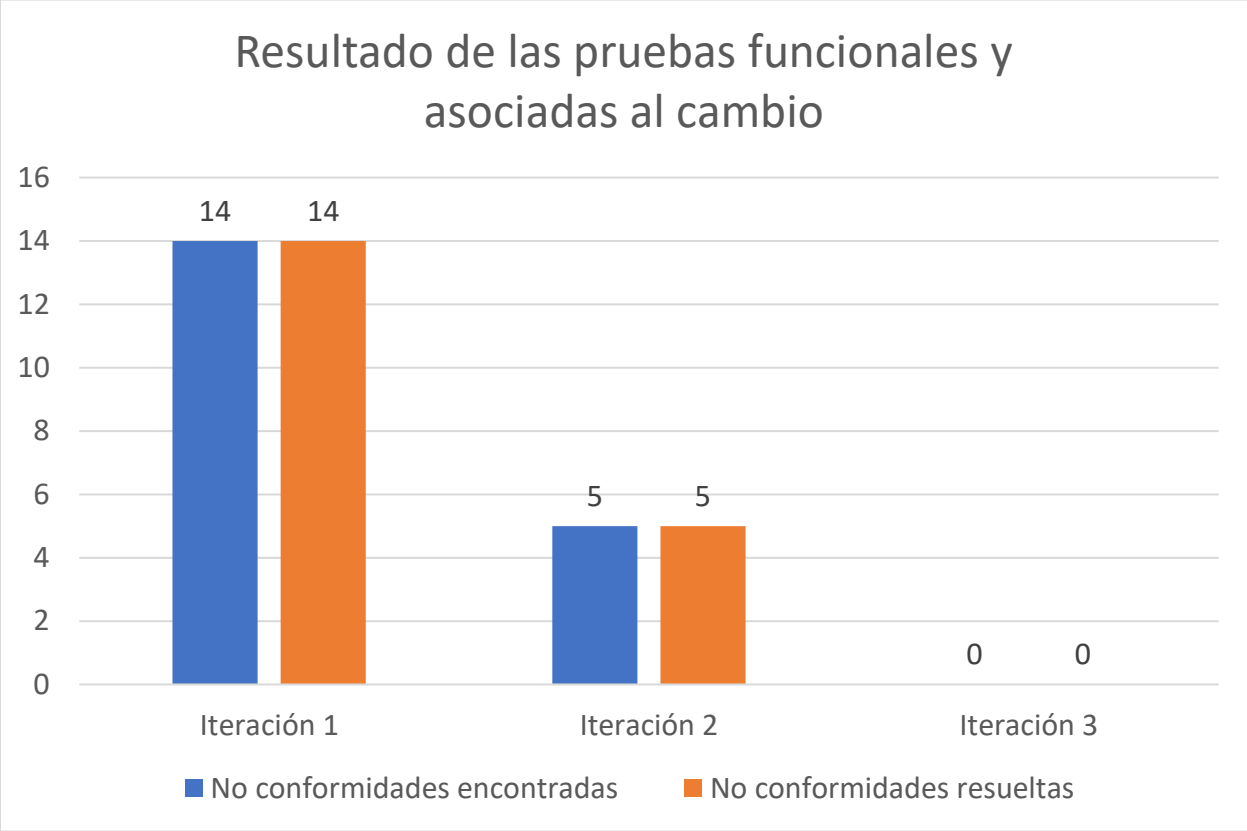


Figura 10 - Gráfico de los resultados de las pruebas funcionales y asociadas al cambio. Fuente: Elaboración propia

3.2.3 Pruebas no funcionales

Las pruebas no funcionales evalúan el "modo" en que el sistema opera, incluyendo, entre otras, pruebas de rendimiento, pruebas de carga, pruebas de estrés, pruebas de usabilidad, pruebas de mantenibilidad, pruebas de confiabilidad y pruebas de portabilidad. Estas pruebas no funcionales se pueden llevar a cabo en todos los niveles de pruebas (Marin Diaz, 2020). En este sistema se validan los requisitos no funcionales mediante el uso de la técnica de calidad listas

de verificación. A continuación, se muestra la figura 11 que muestra las respuestas a la lista de verificación:

Lista de verificación de fiabilidad

No.	REQUISITOS	Evaluación	OBSERVACIONES
	ADAPTABILIDAD		
1	Dependencia de hardware.	**Sí**	
2	Dependencia de software.	**Sí**	
3	Idioma estándar.	**Sí**	
4	Comunicación del sistema con cada uno de los entornos objetivo.	**Sí**	
5	Encapsulación de dependencia.	**Sí**	
6	Representación de dependencias en múltiples sistemas.	**Sí**	
	INSTALABILIDAD		
7	Requisitos previos para la instalación del software.	**Sí**	
8	Requisitos del navegador para el uso de una aplicación.	**Sí**	
9	Requisitos de memoria o RAM.	**Sí**	
10	Requisitos de almacenamiento del disco duro.	**Sí**	
11	Requisitos del sistema operativo para la instalación.	**Sí**	
12	Requisitos para ejecutarse en diferentes dispositivos.	**Sí**	
13	Procedimiento de instalación.	**Sí**	

Figura 11 - Lista de Verificación de fiabilidad. Fuente: Elaboración propia

3.2.4 Pruebas unitarias

Las pruebas unitarias son un tipo de prueba funcional que se enfoca en evaluar el funcionamiento de cada unidad o componente individual de un software de manera aislada. Estas pruebas se realizan para verificar que cada unidad de código cumpla con su función correctamente y que no existan errores en su implementación. Se realizan generalmente por los desarrolladores del software y se pueden automatizar para facilitar su ejecución y reducir el tiempo de prueba. Las utilizadas en este trabajo se enfocan exclusivamente en el enfoque de caja blanca. Estas se centran en la verificación de las unidades más pequeñas del diseño, como módulos, clases y métodos. Se basan en una descripción detallada del diseño como referencia para probar los caminos de control críticos y detectar posibles errores en la unidad (Rojas Robert et al., 2019).

Las pruebas unitarias se implementan con el objetivo de mejorar la calidad del software desarrollado por un equipo. Una vez que se han cumplido todos los requisitos lógicos de una historia y se han ejecutado casos de prueba para garantizar el correcto funcionamiento, el código de la función puede ser agregado al repositorio de integración. Sin embargo, esto sólo ocurre

después de verificar nuevamente la historia para asegurarse de que todo esté en orden. Una vez que se ha realizado esta verificación, se procede al desarrollo estable del repositorio y, finalmente, se realiza la integración del código (Marcillo Sánchez & Román Barrezueta, 2022).

En este proyecto, se realizan pruebas unitarias utilizando la herramienta llamada "test.py". Este enfoque permite evaluar el comportamiento del sistema y verificar que las funcionalidades cumplieran con los requisitos establecidos. Estas fueron efectivas para garantizar la calidad y confiabilidad del sistema. Mediante la realización del sistema se le fueron aplicando las pruebas unitarias a las clases del modelo. A continuación, se presentan algunas funcionalidades del sistema a la cual se les realiza una prueba unitaria:

```python
class ProductoModelTest(TestCase):
    def setUp(self):
        self.producto = producto.objects.create(
            nombreProd='Producto de Prueba',
            versionProd='1.0',
            fechaLimite=timezone.now().date(),
            centroOrigen='Centro de Origen',
            productoRevisado=True,
            iteracion=1
        )

    def test_str_representation(self):
        self.assertEqual(str(self.producto), 'Producto de Prueba')

    def test_fields(self):
        self.assertEqual(self.producto.nombreProd, 'Producto de Prueba')
        self.assertEqual(self.producto.versionProd, '1.0')
        self.assertTrue(self.producto.fechaCreacion)
        self.assertEqual(self.producto.fechaLimite, timezone.now().date())
        self.assertEqual(self.producto.centroOrigen, 'Centro de Origen')
        self.assertTrue(self.producto.productoRevisado)
        self.assertEqual(self.producto.iteracion, 1)
```

Figura 12 - Caso de prueba ProductoModelTest de la clase producto(). Fuente: Elaboración propia

```python
class ListaVerificacionModelTest(TestCase):
    def setUp(self):
        self.lista_verificacion = listaVerificacion.objects.create(
            nombreLV='Lista de Verificación de Prueba',
            fechaCreacion=timezone.now().date(),
            requisitoCalidad='Requisito de Calidad',
            asignado=False
        )

    def test_str_representation(self):
        self.assertEqual(str(self.lista_verificacion), 'Lista de Verificación de Prueba')

    def test_fields(self):
        self.assertEqual(self.lista_verificacion.nombreLV, 'Lista de Verificación de Prueba')
        self.assertTrue(self.lista_verificacion.fechaCreacion)
        self.assertEqual(self.lista_verificacion.requisitoCalidad, 'Requisito de Calidad')
        self.assertFalse(self.lista_verificacion.asignado)
```

Figura 13 - Caso de prueba ListaVerificacionModelTest de la clase listaVerificacion(). Fuente: Elaboración propia

Resultado de las pruebas unitarias

La utilización del conjunto de pruebas de Django permitió ejecutar las principales funcionalidades de forma controlada, evaluando su comportamiento frente a los resultados esperados. A medida que se ejecutaban las pruebas, los errores que surgían se iban corrigiendo para lograr una efectividad del 100% en cada prueba.

```
PS E:\aportes 230528\ServerLocation> python manage.py test
Found 21 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.....................
----------------------------------------------------------------------
Ran 21 tests in 5.153s

OK
Destroying test database for alias 'default'...
```

Ln 31, Col 21 Spaces: 4 UTF-8 LF HTML Go Live Prettier

Figura 14 - Resultado de las pruebas funcionales. Fuente: Elaboración propia

3.2.5 Pruebas de aceptación

Las pruebas de aceptación se enfocan en el comportamiento y capacidades del sistema o producto en un entorno real. En este nivel, es importante tener en cuenta los siguientes aspectos (Marín Díaz et al., 2020):

1. Se verifica la idoneidad del sistema para su uso por parte de los usuarios de negocio.

2. Las pruebas se llevan a cabo utilizando el entorno del cliente.

3. El entorno del cliente puede revelar posibles fallos o errores.

4. Algunas formas comunes de pruebas de aceptación incluyen la prueba de aceptación del usuario (UAT), la prueba de aceptación operativa (OAT), la prueba de cumplimiento contractual o regulatorio, y las pruebas alfa y beta.

Resultado de las pruebas de aceptación, pruebas funcionales. (Historias de usuarios, Requisitos):

Durante la etapa de aceptación, se llevaron a cabo las siguientes acciones junto con el cliente:

1. Validación de que el sistema está completo y funcionará conforme a lo previsto.

2. Verificación de los comportamientos funcionales y no funcionales del sistema.

3. Aseguramiento de que se cumplen los requisitos de los usuarios, las operaciones y los aspectos legales o reglamentarios.

A continuación, se muestra el gráfico 15 que muestra el cumplimiento de los requisitos:

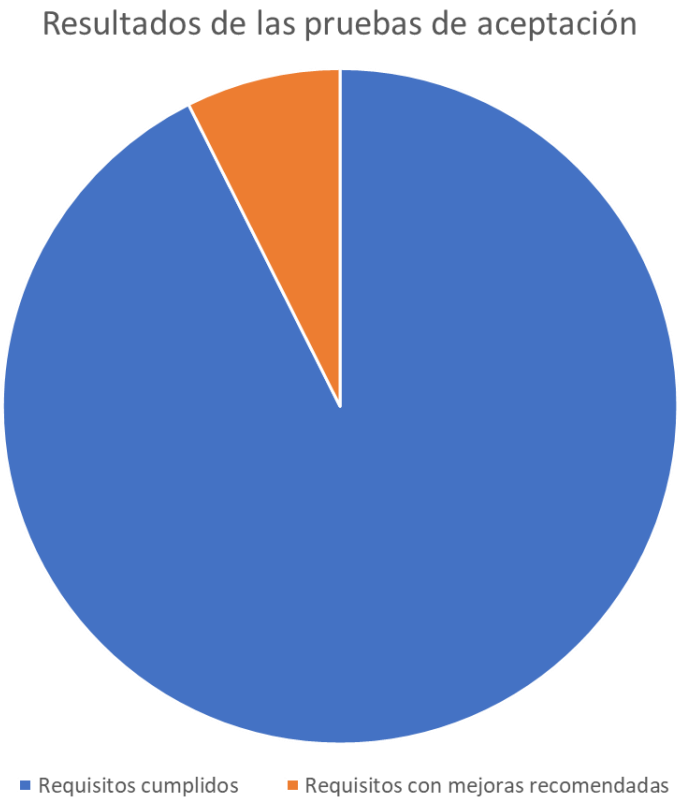


Figura 15 - Resultado de las pruebas de aceptación. Fuente: Elaboración propia

Conclusiones parciales

- Durante el desarrollo del proceso de implementación, la aplicación de estándares de codificación desempeñó un papel importante al lograr que el código fuente fuera uniforme y legible de manera efectiva.
- Fue posible identificar y corregir errores que estaban presentes en el sitio web.
- Estas pruebas permitieron detectar las debilidades y fallos en el sistema, lo que a su vez permitió garantizar y confirmar la solidez y resistencia del sistema.

Conclusiones Finales

Al finalizar este trabajo de diploma, se logró alcanzar tanto el objetivo general establecido como los objetivos de investigación planificados. Como resultado de este estudio, se han obtenido las siguientes conclusiones:

- El análisis de los sistemas homólogos, permitió identificar que estos no pueden ser utilizados como propuesta de solución en la Universidad de las Ciencias Informáticas, sin embargo, de los mismos se identificaron funcionalidades aplicables a la propuesta de solución.
- La selección de las herramientas, la metodología y las técnicas propiciaron la implementación del sistema, cumpliendo de esta forma con el objetivo general.
- El levantamiento de requisitos permitió identificar 27 requisitos funcionales, y modelar estos en 27 historias de usuarios.
- Con la implementación del sistema se creó un producto de software que permite la creación y utilización de las listas de verificación.
- La validación del sistema se llevó a cabo mediante la ejecución de diversas pruebas de software, las cuales permitieron evaluar su funcionalidad. Estas pruebas contribuyeron a garantizar la calidad y la solidez de la solución propuesta.

Recomendaciones

Después de finalizar el desarrollo del sistema propuesto, se sugiere considerar las siguientes recomendaciones:

- Incorporar al sistema la funcionalidad de análisis estadísticos de las respuestas a las preguntas.
- Incorporar al sistema técnicas de IA para la personalización de las listas de verificación.

Referencias Bibliográficas

Academia. (2022). *¿Para qué se utiliza JavaScript?* Kodigo. Retrieved octubre 29, 2023, from https://kodigo.org/para-que-se-utiliza-javascript/

Aizprua, S., Ortega, A., & Von Chong, L. (2021, febrero 4). Calidad del software una perspectiva continua. *Revista Científica Centros*, *8*(2), 120-134. Retrieved mayo 17, 2023, from https://revistasvip.up.ac.pa/index.php/revista_cientifica_centros/article/view/304

Alfonso Benítez, D. (2017, junio). *Herramienta para generar productos de trabajos de la metodología variación AUP-UCI*. Cuba. Retrieved mayo 9, 2023, from https://repositorio.uci.cu/jspui/handle/123456789/8141

Bautista-Villegas, E. (2022, enero 25). Metodologías agiles XP y Scrum, empleadas para el desarrollo de páginas web, bajo MVC, con lenguaje PHP y framework Laravel. *Revista Amazonía Digital*, *1*(1), 7. Google Scholar. 10.55873/rad.v1i1.168

Burnstein, I. (2003). *Practical Software Testing: A Process-Oriented Approach*. Springer. 10.1007/0-387-21658-8_12

Cambarieri, M., Difabio, F., & García Martínez, N. (2020). Implementación de una Arquitectura de Software guiada por el Dominio. *XXI Simposio Argentino de Ingeniería de Software (ASSE 2020)-JAIIO 49 (Modalidad virtual)*. http://sedici.unlp.edu.ar/bitstream/handle/10915/115198/Documento_completo.pdf-PDFA.pdf?sequence=1&isAllowed=y

Carrizo, D., & Alfaro, A. (2018, marzo). Método de aseguramiento de la calidad en una metodología de desarrollo de software: un enfoque práctico. *Ingeniare. Revista chilena de ingeniería*, *26*(1), 114-129. 10.4067/S0718-33052018000100114

Casado Vara, R. C. (2019). Introducción a HTML. (Ediciones Universidad de Salamanca (España)). https://gredos.usal.es/handle/10366/139647

Checklist.com B.V. (2023). *Checklist*. Checklist Templates App & Online Maker - Checklist.com. Retrieved abril 4, 2023, from https://checklist.com

checklist.gg. (2023). *checklist.gg*. checklist.gg. Retrieved junio 25, 2023, from https://checklist.gg

Chinarro Morales, E. J. (2019). Definición e implementación del proceso de pruebas de software basado en la NTP-ISO/IEC 12207: 2016 aplicado a una empresa consultora de software. https://cybertesis.unmsm.edu.pe/bitstream/handle/20.500.12672/10587/Chinarro_me.pdf?sequence=1&isAllowed=y

Collazo Baños, M. D. (2019, junio). Microservicios para la obtención de información de los productos que oferta la red de ventas minorista en Cuba basado en la arquitectura de microservicios. Retrieved junio 24, 2023, from https://repositorio.uci.cu/jspui/handle/123456789/10231

Colorado Rivera, L. P. (2020). Automatización de pruebas funcionales, un complemento para la calidad del software. https://www.semanticscholar.org/paper/Automatizaci%C3%B3n-de-pruebas-funcionales,-un-para-la-Rivera-Paola/f2c0e2cc97120bcd061570032bb572d04b280e45

Cotrina De La Cruz, N. A., & Márquez Zavaleta, E. E. (2021). *Formato para tesis*. Repositorio UPN. Retrieved octubre 30, 2023, from https://repositorio.upn.edu.pe/bitstream/handle/11537/28293/Cotrina%20De%20La%20Cruz%20%e2%80%8bNicky%20Alejandro%20-%20%20Marquez%20Zavaleta%20Eduard%20Elias.pdf?sequence=1&isAllowed=y

Cremé Zerquera, S. (2022). Sistema web de apoyo a la enseñanza de los algoritmos de clasificación no supervisada del Reconocimiento Lógico Combinatorio de Patrones.

Universidad de las Ciencias Informáticas. Facultad de Ciencias y Tecnologías …. https://repositorio.uci.cu/jspui/bitstream/123456789/10641/1/TD_09950_22.pdf

Cueva Gaibor, D. A. (2020). Transformación digital en la universidad actual. *Revista Conrado*, *16*(77), 483-489. http://scielo.sld.cu/scielo.php?pid=s1990-86442020000600483&script=sci_arttext&tlng=en

Defeo, J. A., & Juran, J. M. (2016). *Juran's Quality Handbook: The Complete Guide to Performance Excellence, Seventh Edition* (J. A. Defeo, Ed.; Seventh edition ed.). McGraw-Hill Education.

Díaz, D. (2016). *MSGC-Herramienta para el mantenimiento de SGC certificado bajo ISO 9001*. Universidad Nacional de La Plata. http://sedici.unlp.edu.ar/handle/10915/59020

Django Software Foundation. (2023). *Databases*. Django documentation. Retrieved Junio 19, 2023, from https://docs.djangoproject.com/en/4.2/ref/databases/#sqlite-notes

Doist Inc. (2023). *Todoist*. Todoist | A To-Do List to Organize Your Work & Life. Retrieved marzo 19, 2023, from https://todoist.com/

Duncan, D. (2012, noviembre 16). GRASP Patterns. 32. https://home.cs.colorado.edu/~kena/classes/5448/f12/presentation-materials/duncan.pdf

EALDE Business School. (2020, diciembre 4). *Lista de verificación de auditoria ISO 9001: Cómo elaborarla*. EALDE Business School. Retrieved marzo 3, 2023, from https://www.ealde.es/lista-verificacion-auditoria-iso-9001/

Echevarría Nieves, Z. (2021, noviembre). Análisis y diseño de una aplicación móvil para informar sobre covid-19 en aguada de pasajeros. *Universidad de las Ciencias Informáticas. Facultad 1*. https://repositorio.uci.cu/jspui/bitstream/123456789/10655/1/TD_09795_21.pdf

Falcón Márquez, O. R. (2015, octubre 15). *Proceso productivo de la UCI evaluado con CMMI nivel 2*. Universidad de las Ciencias Informáticas. Retrieved marzo 17, 2023, from https://www.uci.cu/proceso-productivo-de-la-uci-evaluado-con-cmmi-nivel-2

Flores Castillejo, J. O. (2020). Sistema web open source vue Js para el proceso de pruebas de calidad de software en la empresa NextPerience. *Universidad César Vallejo*.

García Avila, J. (2019, marzo 9). *Módulo Fototeca de la Biblioteca Virtual del Centro de Información para la Prensa*. Repositorio UCi. Retrieved octubre 28, 2023, from https://repositorio.uci.cu/jspui/bitstream/123456789/10500/1/TD_09918_22.pdf

García Avila, J. (2022, junio). Módulo Fototeca de la Biblioteca Virtual del Centro de Información para la Prensa. https://repositorio.uci.cu/jspui/handle/123456789/10500

Gehlbach, H., & Artino, A. R. (2018). The survey checklist (manifesto). *Academic Medicine*, *93*(3), 360-366.

Gómez Palomo, S. R., & Moraleda Gil, E. (2020). *Aproximación a la ingeniería del software*. Centro de Estudios Ramón Areces.

Gordillo Agudelo, C. E., Florian-Gaviria, B., & Aristizábal, E. M. (2021, diciembre 29). Estudio de plataformas de monitoreo para seleccionar la pila tecnológica base de un sistema de analíticas especializado para pruebas de software. *INGENIERÍA Y COMPETITIVIDAD*, *24*(1). https://doi.org/10.25100/iyc.v24i1.11086

Gore, H., Singh, R. K., Singh, A., Singh, A. P., Shabaz, M., Singh, B. K., & Jagota, V. (2021). Django: Web development simple & fast. *Annals of the Romanian Society for Cell Biology*, *25*(6), 4576-4585. https://www.annalsofrscb.ro/index.php/journal/article/download/6301/4788

Grau Rodríguez, D. (2019). Sistema inteligente para el agrupamiento de patrones de diseño de recursos educativos en lenguajes potenciales. (Universidad de las Ciencias Informáticas. Facultad 4). https://repositorio.uci.cu/jspui/handle/123456789/10116

Gulín-González, J. (2022, febrero 5). Achievement and challenges of the Cuban Science, Technology, and Innovation System: A perspective on computational science. *International Journal of Quantum Chemistry*, *122*(3). 10.1002/qua.26837

Hernández Fariñas, L. (2022, diciembre). Plataforma Web para la Vigilancia Tecnológica. Retrieved Junio 20, 2023, from https://repositorio.uci.cu/jspui/handle/123456789/10614

Hoyer, R. W., & Hoyer, B. (2001). ¿Qué es calidad? *Revista Quality Progress*, *34*(2). https://www.academia.edu/download/38262413/Que_es_calidad.pdf

Ibarra Corona, M. A., & Escudero-Nahón, A. (2021). Metodologías de Ingeniería de Software para el Diseño y Desarrollo de Plataformas de Tecnología Educativa. *Academia Journals, Chetumal*, 368-371. Google Scholar. Retrieved junio 18, 2023, from https://www.researchgate.net/profile/Alexandro-Escudero-Nahon/publication/352397970_Metodologias_de_Ingenieria_de_Software_para_el_Diseno_y_Desarrollo_de_Plataformas_de_Tecnologia_Educativa/links/60c83724a6fdcc57ed055e87/Metodologias-de-Ingenieria-de-Softw

IBM. (2023). *¿Qué es la prueba de software y cómo funciona?* IBM. Retrieved agosto 25, 2023, from https://www.ibm.com/es-es/topics/software-testing

Icons8 LLC. (2023). *Network symbols and icons in Dotted Style, PNG, SVG*. Icons8. Retrieved septiembre 14, 2023, from https://icons8.com/icon/set/network/dotty

ISO 25010. (2011). *ISO 25010*. iso25000.com. Retrieved noviembre 10, 2023, from https://iso25000.com/index.php/normas-iso-25000/iso-25010

ISTQB. (2023). *Certified Tester Foundation Level Syllabus*.

Juran, J. M., & Defeo, J. A. (2016). *Juran's Quality Handbook: The Complete Guide to Performance Excellence, Seventh Edition* (J. A. Defeo, Ed.). McGraw-Hill Education.

Kodigo. (2023). *Los 5 lenguajes de programación más utilizados en la actualidad*. Kodigo.org. Retrieved noviembre 10, 2023, from https://kodigo.org/los-5-lenguajes-de-programacion-mas-utilizados-durante-el-2022-y-2023/

Larman, C. (1998). *Applying UML and patterns* (Vol. 2). Prentice Hall Englewood Cliffs, NJ.

León Ardilla, K. Y. (2020). Metodologías ágiles como herramientas fundamentales para el desarrollo de emprendimientos. 72. Retrieved junio 15, 2023, from https://repository.unad.edu.co/bitstream/handle/10596/33613/kyleona.pdf?sequence=3&isAllowed=y

Lima Torres, S. (2020, septiembre). Componente de revisión de estándar de arquitectura de datos para el gestor SQLite. *UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS. FACULTAD 3. XETID*. https://repositorio.uci.cu/jspui/handle/123456789/10481

López Rivero, E. (2013, junio). *Sistema Basado en Casos para contribuir a la disminución de los costos de la calidad de los proyectos del Centro de Telemática.* La Habana, Cuba. Retrieved abril 21, 2023, from https://repositorio.uci.cu/jspui/handle/ident/8046

Marcillo Sánchez, P. M., & Román Barrezueta, L. D. (2022). Análisis de la información generada para mantener la escalabilidad y persistencia del proceso de desarrollo de software. *Serie Científica de la Universidad de las Ciencias Informáticas*, *15*(8), 193-227. https://dialnet.unirioja.es/servlet/articulo?codigo=8955513

Marín Díaz, A., Casañola Trujillo, Y., & Buedo Hidalgo, D. (2020, septiembre). Estrategia de pruebas para organizaciones desarrolladoras de software. *Revista Cubana de Ciencias Informáticas*, *14*(3), 83-104. http://scielo.sld.cu/scielo.php?script=sci_abstract&pid=S2227-18992020000300083&lng=es&nrm=iso&tlng=en

Marín Diaz, A., Trujillo Casañola, Y., & Buedo Hidalgo, D. (2020). *Estrategia de pruebas para organizaciones desarrolladoras de software*.

Microsoft. (2023). *Visual Studio Code* (Versión 1.79) [Software]. https://code.visualstudio.com

Mozilla Corporation. (2023, febrero 9). *JavaScript | MDN*. MDN Web Docs. Retrieved junio 25, 2023, from https://developer.mozilla.org/es/docs/Web/JavaScript

Myers, G. J., Sandler, C., & Badgett, T. (2011). *The Art of Software Testing*. Wiley.

Nazar, N., Hu, Y., & Jiang, H. (2016, septiembre). Summarizing software artifacts: A literature review. *Journal of Computer Science and Technology*, *31*(5), 883-909. https://doi.org/10.1007/s11390-016-1671-1

Noa. (2022, septiembre 22). *Framework: qué es, para qué sirve y algunos ejemplos*. UNIR FP. Retrieved octubre 29, 2023, from https://unirfp.unir.net/revista/ingenieria-y-tecnologia/framework/

Paz, J. A. M. (2016). Análisis del proceso de pruebas de calidad de software. *Ingeniería solidaria*, *12*(20), 163-176.

Pressman, R. S., & Maxim, B. R. (2019). *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education.

Protogerou, C., & Hagger, M. S. (2020, diciembre). A checklist to assess the quality of survey studies in psychology. *Methods in Psychology*, *3*, 100031. 10.1016/j.metip.2020.100031

QS Quacquarelli Symonds Limited. (2021). *Universidad de las Ciencias Informáticas : Rankings, Fees & Courses Details*. Top Universities. Retrieved marzo 16, 2023, from https://www.topuniversities.com/universities/universidad-de-las-ciencias-informaticas

Ramírez Carreño, R. J. (2022). Aplicación informática para la gestión de información de la Práctica Profesional en el Centro de Software Libre. Retrieved junio 23, 2023, from https://repositorio.uci.cu/jspui/handle/123456789/10465

Refactoring.Guru. (2023). *Patrones estructurales*. Refactoring.Guru. Retrieved June 23, 2023, from https://refactoring.guru/es/design-patterns/

Roa Bánquez, K., Martínez Barrera, r., & Cabrera Martínez, C. (2019). Experiencias en el aula virtual como mediación pedagógica para el apoyo al aprendizaje en el espacio académico de lenguaje cliente servidor. *CITAS*, *5*(1), 109-121. Retrieved junio 25, 2023, from https://revistas.usantotomas.edu.co/index.php/citas/article/view/6075/5792

Rodríguez, A. (n.d.). *"SEIE. Sistema de Evaluación Integral Estudiantil"*. Repositorio Digital. Retrieved octubre 12, 2023, from https://repositorio.uci.cu/jspui/bitstream/123456789/10529/1/TD_09646_20.pdf

Rodríguez, A. (2020). *"SEIE. Sistema de Evaluación Integral Estudiantil"*. Repositorio Digital. Retrieved octubre 12, 2023, from https://repositorio.uci.cu/jspui/bitstream/123456789/10529/1/TD_09646_20.pdf

Rodríguez Baena, L. (2013). *Cascading Style Sheets*. Interacción Persona-Ordenador. http://www.colimbo.net/documentos/documentacion/fipo/FIPO06-CSS.pdf

Rojas Robert, D. M., Pérez Morales, Z., & Delgado Dapena, M. D. (2019). Generador de valores interesantes para casos de pruebas unitarias. *Ingeniería Industrial*, *40*(2), 183-193. https://www.semanticscholar.org/paper/Generador-de-valores-interesantes-para-casos-de-Rojas-Robert-P%C3%A9rez-Morales/3e2f11b5125e903666e3fc177f69717e8c16b3c6

Saavedra Saavedra, Z. J. (2022). Implementación de mejoras en la gestión de calidad e inocuidad de una empresa productora de mezclas alimenticias en seco. 97. Retrieved mayo 16, 2023, from http://repositorio.lamolina.edu.pe/bitstream/handle/20.500.12996/5587/saavedra-saavedra-zoila-janeth.pdf?sequence=1&isAllowed=y

SARZOSA, C. E. (2018, septiembre 27). *UNIVERSIDAD TÉCNICA DEL NORTE*. UNIVERSIDAD TÉCNICA DEL NORTE. Retrieved noviembre 1, 2023, from

http://repositorio.utn.edu.ec/bitstream/123456789/8641/1/04%20ISC%20484%20TRABAJO%20DE%20GRADO.pdf

Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide - The Definitive Guide to Scrum: The Rules of the Game*. https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf

Segura, A., Vidal, C., & Prieto, M. (2008). Evaluación de la Calidad del Software para el Aprendizaje. *X Simposio Internacional de Informática Educativa SIIE*, 59-64.

Serna López, G. A., Jaramillo Terán, L. M., Upegui Giraldo, G. d. J., Gómez Giraldo, R., & Bueno Loaiza, Y. P. (2020). Piloto de automatización de pruebas de software en la Unidad de Servicios Tecnológicos del Centro de Servicios y Gestión Empresarial (CESGE) del SENA. *Revista CINTEX*, *25*, 45-50. https://www.semanticscholar.org/paper/Piloto-de-automatizaci%C3%B3n-de-pruebas-de-software-en-L%C2%B4%C3%B3pez-Ter%C3%A1n/7a16a9799869198a77bc2c5464599ddacdf3491f

SQLite Consortium. (2023, mayo). *SQLite*. SQLite Home Page. Retrieved junio 22, 2023, from https://www.sqlite.org/index.html

Staff.Wiki. (2023). *Using AI To Create Checklists - Staff.Wiki*. Staff Wiki. Retrieved junio 25, 2023, from https://staff.wiki/267274,Page,using_ai_to_create_checklists,KB.aspx

Tague, N. R. (2005). *The Quality Toolbox*. ASQ Quality Press.

Universidad de las Ciencias Informáticas. (2023). *Centros de Desarrollo*. Universidad de las Ciencias Informáticas. Retrieved marzo 15, 2023, from https://www.uci.cu/investigacion-y-desarrollo/centros-de-desarrollo

Villanueva Rosas, E., & Muñoz, M. (2020). *"Propuesta De Un Marco De Trabajo: Para El Aseguramiento De La Calidad Del Proceso Y Producto De Software Para Entidades Muy Pequeñas,"*. IEEE Xplore.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9390146&isnumber=9390092

Vincent, W. S. (2023). *Django for Beginners: Build websites with Python and Django*. WelcomeToCode. https://books.google.com.cu/books?id=GVxwDwAAQBAJ&printsec=frontcover&hl=es&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

Visual Paradigm. (2023). *Visual Paradigm* (Versión 8.0) [Software]. https://www.visual-paradigm.com

Waworuntu, A. (2021, junio 9). *View of Vol 8 No 1 (2021): IJNMT (International Journal of New Media Technology)*. journal umn. Retrieved octubre 28, 2023, from https://ejournals.umn.ac.id/index.php/IJNMT/issue/view/202/IJNMTJune2021

WHO Patient Safety & World Health Organization. (2009). WHO guidelines for safe surgery 2009: safe surgery saves lives. *Safe surgery saves lives*, (WHO/IER/PSP/2008.08-1E), 124. https://apps.who.int/iris/handle/10665/44185

Anexos

Anexo 1: Guía de entrevista:

Nombre de la entrevistadora: Daniela Oramas Romero

Centro de trabajo: Universidad de Ciencias Informáticas (UCI)

Nombre de entrevistada: M. Sc. Aymara Marín Díaz

Centro de trabajo: Universidad de Ciencias Informáticas (UCI)

1. Cantidad de años en la industria de software
2. Cargo que desempeña
3. ¿Ha realizado levantamiento de requisitos?
4. ¿Ha participado en revisiones donde tenga que aplicar listas de verificación?
5. A su juicio, ¿qué proceso le llevó más tiempo el levantamiento de requisitos o el llenado de las listas de verificación?
6. ¿Si fuera a realizar un sistema de gestión de listas de verificación que requerimientos obtendrían?

Anexo 2: Historias de Usuario

Tabla 11 - Historia de Usuario número 1

Historia de Usuario	
Número: 1	Nombre: Adicionar Pregunta
Usuario: Revisor jefe	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alto
Puntos estimados: 1	Iteración asignada: 2
Programador responsable: Daniela Oramas Romero	
Descripción: Al autenticarse el usuario, en la interfaz principal selecciona la opción Listas de verificación, se presiona el ícono que se relaciona a la opción Ver, al final de la vista se pulsa	

el signo de más. Se muestran los campos No, Pregunta, Respuesta y Observación.
Observaciones: - Si los datos están incompletos o incorrectos se señalan los campos en cuestión dando la posibilidad al revisor jefe de realizar nuevamente la acción en cuestión. - Si selecciona la opción Cancelar regresará al listar pregunta.

Tabla 12 - Historia de Usuario número 2

Historia de Usuario	
Número: 2	Nombre: Modificar Pregunta
Usuario: Revisor jefe	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alto
Puntos estimados: 1	Iteración asignada: 2
Programador responsable: Daniela Oramas Romero	
Descripción: Al autenticarse el usuario, en la interfaz principal selecciona la opción Listas de verificación, se presiona el ícono que se relaciona a la opción Ver, en el lateral de cada pregunta está la opción de modificar. Se muestran los campos No y Pregunta.	
Observaciones: - Si los datos están incompletos o incorrectos se señalan los campos en cuestión dando la posibilidad al revisor jefe de realizar nuevamente la acción en cuestión. - Si selecciona la opción Cancelar regresará al listar pregunta.	

Tabla 13 - Historia de Usuario número 3

Historia de Usuario	
Número: 3	Nombre: Listar Pregunta
Usuario: Revisor jefe	
Prioridad en negocio: Media	Riesgo en desarrollo: Medio

Puntos estimados: 1	Iteración asignada: 1
Programador responsable: Daniela Oramas Romero	
Descripción: Al autenticarse el usuario, en la interfaz principal selecciona la opción Listas de verificación, se presiona el ícono que se relaciona a la opción Ver, y a continuación se muestra una vista con un listado de todas las preguntas más los campos No, Pregunta, Respuesta y Observación.	
Observaciones: Se muestra toda la información de las preguntas	

Tabla 14 - Historia de Usuario número 4

Historia de Usuario	
Número: 4	Nombre: Eliminar Pregunta
Usuario: Revisor jefe	
Prioridad en negocio: Alta	Riesgo en desarrollo: Bajo
Puntos estimados: 1	Iteración asignada: 2
Programador responsable: Daniela Oramas Romero	
Descripción: Al autenticarse el usuario, en la interfaz principal selecciona la opción Listas de verificación, se presiona el ícono que se relaciona a la opción Ver, en el lateral de cada pregunta está la opción de eliminar. Al seleccionar la pregunta aparece un mensaje de confirmación donde debe seleccionar el botón eliminar	
Observaciones: El revisor jefe puede cancelar la opción en cualquier momento	

Tabla 15 - Historia de Usuario número 5

Historia de Usuario	
Número: 5	Nombre: Eliminar notificaciones

Usuario: Revisor jefe	
Prioridad en negocio: Alta	Riesgo en desarrollo: Bajo
Puntos estimados: 1	Iteración asignada: 2
Programador responsable: Daniela Oramas Romero	
Descripción: Al autenticarse el usuario, en la interfaz principal se muestra una campana que en caso de que esté de color azul.	
Observaciones: El revisor jefe puede buscar cualquier palabra de la pregunta.	

Tabla 16 - Historia de Usuario número 6

Historia de Usuario	
Número: 6	Nombre: Registrar los resultados
Usuario: Revisor	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alto
Puntos estimados: 1	Iteración asignada: 2
Programador responsable: Daniela Oramas Romero	
Descripción: Al autenticarse el usuario, en la interfaz principal selecciona la opción Listas de verificación, se presiona el ícono que se relaciona a la opción Ver, donde se muestra la vista de preguntas, al completar esta lista de verificación el revisor selecciona la opción guardar para así conservar todas las respuestas a las preguntas	
Observaciones: El revisor puede guardar estos datos, aunque no haya terminado	

Tabla 17 - Historia de Usuario número 7

Historia de Usuario	
Número: 7	Nombre: Adicionar Usuario

Usuario: Revisor jefe	
Prioridad en negocio: Alta	Riesgo en desarrollo: Medio
Puntos estimados: 1	Iteración asignada: 2
Programador responsable: Daniela Oramas Romero	
Descripción: Al autenticarse el usuario, en la interfaz principal selecciona la opción Usuarios, y a continuación se muestra una vista con un listado de todos los usuarios, se selecciona la opción Adicionar Usuario y se muestran los campos Usuario, Contraseña, Repetir contraseña y Correo, y los botones enviar y cancelar.	
Observaciones: - Si los datos están incompletos o incorrectos se señalan los campos en cuestión dando la posibilidad de realizar nuevamente la acción en cuestión. - Si selecciona la opción Cancelar regresará al listar usuario.	

Tabla 18 - Historia de Usuario número 8

Historia de Usuario	
Número: 8	Nombre: Modificar Usuario
Usuario: Revisor jefe	
Prioridad en negocio: Alta	Riesgo en desarrollo: Medio
Puntos estimados: 2	Iteración asignada: 2
Programador responsable: Daniela Oramas Romero	
Descripción: Al autenticarse el usuario, en la interfaz principal selecciona la opción Usuarios, y a continuación se muestra una vista con un listado de todos los usuarios, al lado de cada usuario se muestra un ícono el cual identifica la opción modificar, y se muestran los campos Nombre de Usuario, y Dirección de correo electrónico, seguido de los botones enviar y cancelar.	
Observaciones: - Si los datos están incompletos o incorrectos se señalan los campos en cuestión dando la posibilidad al revisor jefe de realizar nuevamente la acción en cuestión. - Si selecciona la opción Cancelar regresará al listar usuario. - La contraseña no se puede modificar.	

Tabla 19 - Historia de Usuario número 9

Historia de Usuario	
Número: 9	Nombre: Eliminar Usuario
Usuario: Revisor jefe	
Prioridad en negocio: Media	Riesgo en desarrollo: Bajo
Puntos estimados: 1	Iteración asignada: 1
Programador responsable: Daniela Oramas Romero	
Descripción: Al autenticarse el usuario, en la interfaz principal selecciona la opción Usuarios, y a continuación se muestra una vista con un listado de todos los usuarios, al lado de cada usuario se muestra un ícono el cual identifica la opción eliminar. Al seleccionar el usuario aparece un mensaje de confirmación donde debe seleccionar el botón eliminar.	
Observaciones: Se podrá cancelar esta opción	

Tabla 20 - Historia de Usuario número 11

Historia de Usuario	
Número: 11	Nombre: Buscar Usuario
Usuario: Revisor jefe	
Prioridad en negocio: Media	Riesgo en desarrollo: Medio
Puntos estimados: 1	Iteración asignada: 1
Programador responsable: Daniela Oramas Romero	
Descripción: Al autenticarse el usuario, en la interfaz principal selecciona la opción Usuarios, y a continuación se muestra una vista con un listado de todos los usuarios, donde se muestra una barra de buscar.	
Observaciones: Se puede buscar tanto por nombre de usuario como por correo.	

Tabla 21 - Historia de Usuario número 12

Historia de Usuario	
Número: 12	Nombre: Autenticar Usuario
Usuario: Revisor jefe, revisor	
Prioridad en negocio: Media	Riesgo en desarrollo: Medio
Puntos estimados: 1	Iteración asignada: 1
Programador responsable: Daniela Oramas Romero	
Descripción: Al encontrarse en la página del autenticar el usuario tendrá que poner su usuario y contraseña y tocar el botón acceder.	
Observaciones: - Si los datos están incompletos o incorrectos se señalan los campos en cuestión dando la posibilidad al revisor jefe de realizar nuevamente la acción en cuestión.	

Tabla 22 - Historia de Usuario número 14

Historia de Usuario	
Número: 14	Nombre: Modificar Producto
Usuario: Revisor jefe	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alto
Puntos estimados: 1	Iteración asignada: 2
Programador responsable: Daniela Oramas Romero	
Descripción: Al autenticarse el usuario, en la interfaz principal selecciona la opción Productos, se pulsa el ícono del lateral de cada producto que corresponde al modificar. Al seleccionar el producto puede modificar y se muestra una ventana con los campos Nombre del Producto, Versión, Fecha Límite, Centro de Origen e Iteración, seguido por los botones aceptar y cancelar.	
Observaciones: - Si los datos están incompletos o incorrectos se señalan los campos en cuestión dando la	

posibilidad al revisor jefe de realizar nuevamente la acción en cuestión. - El revisor jefe puede cancelar la opción en cualquier momento.

Tabla 23 - Historia de Usuario número 15

Historia de Usuario	
Número: 15	Nombre: Eliminar Producto
Usuario: Revisor jefe	
Prioridad en negocio: Alta	Riesgo en desarrollo: Bajo
Puntos estimados: 1	Iteración asignada: 2
Programador responsable: Daniela Oramas Romero	
Descripción: Al autenticarse el usuario, en la interfaz principal selecciona la opción Productos, se pulsa el ícono del lateral de cada producto que corresponde al eliminar. Al seleccionar el producto aparece un mensaje de confirmación donde debe seleccionar el botón eliminar.	
Observaciones: El revisor jefe puede cancelar la opción en cualquier momento.	

Tabla 24 - Historia de Usuario número 16

Historia de Usuario	
Número: 16	Nombre: Listar Producto
Usuario: Revisor jefe	
Prioridad en negocio: Alta	Riesgo en desarrollo: Bajo
Puntos estimados: 1	Iteración asignada: 2
Programador responsable: Daniela Oramas Romero	
Descripción: Al autenticarse el usuario, en la interfaz principal selecciona la opción Productos, y a continuación se muestra una vista con un listado de todos los productos.	

Observaciones: Se muestra toda la información de los productos.

Tabla 25 - Historia de Usuario número 17

Historia de Usuario	
Número: 17	Nombre: Buscar Producto
Usuario: Revisor jefe	
Prioridad en negocio: Alta	Riesgo en desarrollo: Medio
Puntos estimados: 1	Iteración asignada: 2
Programador responsable: Daniela Oramas Romero	
Descripción: Al autenticarse el usuario, en la interfaz principal selecciona la opción Productos, se ve una barra de buscar y ahí se introduce la información del producto, y se irán mostrando los productos que coinciden.	
Observaciones: Se puede buscar con cualquier información del producto.	

Tabla 26 - Historia de Usuario número 18

Historia de Usuario	
Número: 18	Nombre: Adicionar Lista de Verificación
Usuario: Revisor jefe	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alto
Puntos estimados: 4	Iteración asignada: 3
Programador responsable: Daniela Oramas Romero	
Descripción: Al autenticarse el usuario, en la interfaz principal se mostrará la vista de listas de verificación, selecciona el botón Adicionar Lista de Verificación y se muestra una ventana con los campos Nombre de la lista, Requisito de Calidad y Archivo con las listas de verificación,	

más los botones de enviar y cancelar.
Observaciones: - Si los datos están incompletos o incorrectos se señalan los campos en cuestión dando la posibilidad al revisor jefe de realizar nuevamente la acción en cuestión. - Solo admite documentos con las extensiones (.xlsx). - Se podrá cancelar este procedimiento en cualquier momento.

Tabla 27 - Historia de Usuario número 19

Historia de Usuario	
Número: 19	Nombre: Modificar Lista de Verificación
Usuario: Revisor jefe	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alto
Puntos estimados: 4	Iteración asignada: 3
Programador responsable: Daniela Oramas Romero	
Descripción: Al autenticarse el usuario, en la interfaz principal se mostrará la vista de listas de verificación, selecciona el ícono que corresponde al modificar y se muestra una ventana con los campos Nombre de la lista y Requisito de Calidad.	
Observaciones: - Si los datos están incompletos o incorrectos se señalan los campos en cuestión dando la posibilidad al revisor jefe de realizar nuevamente la acción en cuestión. - Se podrá cancelar este procedimiento en cualquier momento.	

Tabla 28 - Historia de Usuario número 20

Historia de Usuario	
Número: 20	Nombre: Eliminar Lista de Verificación
Usuario: Revisor jefe	
Prioridad en negocio: Alta	Riesgo en desarrollo: Bajo

Puntos estimados: 2	Iteración asignada: 3
Programador responsable: Daniela Oramas Romero	
Descripción: Al autenticarse el usuario, en la interfaz principal se mostrará la vista de listas de verificación, selecciona el ícono que corresponde al eliminar, se mostrará un mensaje de confirmación.	
Observaciones: Se podrá cancelar este procedimiento en cualquier momento.	

Tabla 29 - Historia de Usuario número 21

Historia de Usuario	
Número: 21	Nombre: Listar Lista de Verificación
Usuario: Revisor, revisor jefe	
Prioridad en negocio: Baja	Riesgo en desarrollo: Bajo
Puntos estimados: 1	Iteración asignada: 1
Programador responsable: Daniela Oramas Romero	
Descripción: Al autenticarse el usuario, en la interfaz principal se mostrará la vista de listas de verificación.	
Observaciones: Se muestra toda la información de las listas de verificación.	

Tabla 30 - Historia de Usuario número 22

Historia de Usuario	
Número: 22	Nombre: Buscar Lista de Verificación
Usuario: Revisor jefe, revisor	
Prioridad en negocio: Baja	Riesgo en desarrollo: Bajo

Puntos estimados: 1	Iteración asignada: 1
Programador responsable: Daniela Oramas Romero	
Descripción: Al autenticarse el usuario, en la interfaz principal se mostrará la vista de listas de verificación, en la barra de buscar se introduce la información de las listas de verificación.	
Observaciones: Se puede buscar por cualquier información.	

Tabla 31 - Historia de Usuario número 23

Historia de Usuario	
Número: 23	Nombre: Asignar Lista de Verificación
Usuario: Revisor jefe	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alto
Puntos estimados: 4	Iteración asignada: 3
Programador responsable: Daniela Oramas Romero	
Descripción: Al autenticarse el usuario, en la interfaz principal se mostrará la vista de listas de verificación, selecciona el botón Asignar a la lista de verificación que se quiera asignar.	
Observaciones: Se podrá cancelar este procedimiento en cualquier momento.	

Tabla 32 - Historia de Usuario número 24

Historia de Usuario	
Número: 24	Nombre: Mostrar notificación de asignación
Usuario: Revisor	
Prioridad en negocio: Baja	Riesgo en desarrollo: Bajo
Puntos estimados: 1	Iteración asignada: 2

Programador responsable: Daniela Oramas Romero
Descripción: Al autenticarse el usuario, en la interfaz principal se mostrará la vista de listas de verificación y se mostrará un ícono de campana donde al dar click se mostrarán todas las notificaciones.
Observaciones: Se podrá cancelar este procedimiento en cualquier momento.

Tabla 33 - Historia de Usuario número 25

Historia de Usuario	
Número: 25	Nombre: Chequear listas de verificación asignadas
Usuario: Revisor jefe	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alto
Puntos estimados: 1	Iteración asignada: 2
Programador responsable: Daniela Oramas Romero	
Descripción: Al autenticarse el usuario, en la interfaz principal se mostrará la vista de listas de verificación, se selecciona la opción usuarios y en asignadas seleccionar al usuario que se quiera revisar. Ahí se mostrarán todas las listas asignadas a ese usuario y se elige la lista de verificación a chequear.	
Observaciones: -No se podrá editar ningún resultado.	

Tabla 34 - Historia de Usuario número 26

Historia de Usuario	
Número: 26	Nombre: Cambiar contraseña
Usuario: Revisor jefe	
Prioridad en negocio: Baja	Riesgo en desarrollo: Bajo

Puntos estimados: 0.5	Iteración asignada: 1
Programador responsable: Daniela Oramas Romero	
Descripción: Al autenticarse el usuario, en la interfaz principal se muestra en la esquina superior el nombre de usuario, al seleccionarlo se muestra las opciones Cambiar contraseña y Salir, se selecciona la opción Cambiar contraseña y se muestran los campos Contraseña antigua, Contraseña nueva y Contraseña nueva (Confirmación).	
Observaciones: - Si las contraseñas no coinciden se muestra un mensaje “Las contraseñas no coinciden”. - La contraseña no puede ser similar al nombre de usuario.	

Tabla 35 - Historia de Usuario número 27

Historia de Usuario	
Número: 27	Nombre: Crear nueva lista de verificación
Usuario: Revisor jefe	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alto
Puntos estimados: 1	Iteración asignada: 1
Programador responsable: Daniela Oramas Romero	
Descripción: Al autenticarse el usuario, en la interfaz principal se mostrará la vista de listas de verificación, selecciona el botón Crear Nueva Lista de Verificación y se muestra una ventana con los campos Nombre de la lista, Requisito de Calidad y Archivo, más los botones de enviar y cancelar. Luego se toca la opción ver y ahí se añade las preguntas que se desee.	
Observaciones: - Si las contraseñas no coinciden se muestra un mensaje “Las contraseñas no coinciden”. - La contraseña no puede ser similar al nombre de usuario.	

Anexo 3: Acta de aceptación

UCI Universidad de las Ciencias Informáticas

Acta de aceptación

ACTA DE ACEPTACIÓN

En cumplimiento de la investigación: Sistema para la gestión de listas de verificación en actividades de calidad, se hace constar que el cliente acepta los productos que se relacionan a continuación:

Listado de productos que serán aceptados:

- Requisitos
- Historias de usuarios

Entrega	**Recibe**
Nombre y apellidos: Daniela Oramas Romero	**Nombre y apellidos:** Aymara Marin Diaz
Cargo: Equipo de proyecto	**Cargo:** Cliente
Firma:	**Firma:**

Fecha: 20/2/2023.

Figura 16 - Acta de aceptación (Requisitos, historias de usuario)

UCI Universidad de las Ciencias Informáticas

Acta de aceptación

ACTA DE ACEPTACIÓN

En cumplimiento de la investigación Sistema para la gestión de listas de verificación en actividades de calidad, se hace constar que el cliente acepta los productos que se relacionan a continuación:

Listado de productos que serán aceptados:

- MaNagerList v1.0 (Sistema Web)

Entrega	Recibe
Nombre y apellidos: Daniela Oramas Romero	**Nombre y apellidos:** Aymara Marin Diaz
Cargo: Equipo de proyecto	**Cargo:** Cliente
Firma:	**Firma:**

Fecha: 30/11/2023

Figura 17 - Acta de aceptación