UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE QUÍMICA

APLICACIÓN DEL PENSAMIENTO BASADO EN RIESGOS PARA LA OPTIMIZACIÓN DEL TIEMPO EN LA ETAPA DE DESARROLLO PRELIMINAR DE UN SOFTWARE.

TRABAJO ESCRITO VÍA CURSOS DE EDUCACIÓN CONTINÚA

QUE PARA OBTENER EL TÍTULO DE

INGENIERO QUÍMICO METALÚRGICO

PRESENTA

ANGEL FRANCISCO ROMERO LOPEZ

Tutor: María Teresa Ávila Muñoz

Ciudad Universitaria CD.MX **2024**

JURADO ASIGNADO:

PRESIDENTE:	**Profesor:**	**MARIA TERESA AVILA MUÑOZ**
VOCAL:	**Profesor:**	**ESTEBAN HERNANDEZ TREJO**
SECRETARIO:	**Profesor:**	**ALEJANDRO HERNANDEZ MARTINEZ**
1er. SUPLENTE:	**Profesor:**	**JOSE SABINO SAMANO CASTILLO**
2° SUPLENTE:	**Profesor:**	**MARIA ROSA ISELA GASCON GUERRERO**

SITIO DONDE SE DESARROLLÓ EL TEMA: REMOTO (VIA GOOGLE MEET)

ASESOR DEL TEMA:

MARIA TERESA AVILA MUÑOZ

SUPERVISOR TÉCNICO:

SUSTENTANTE:

ANGEL FRANCISCO ROMERO

Índice

Objetivos 1
Conceptos y definiciones para el manejo del software 2
Segmentos clave del proyecto 10
Conceptos y definiciones sobre ISO 9001:2015 Sistemas de gestión de la calidad: Requisitos 13
Conceptos y definiciones de la ISO 31000:2018 Directrices para abordar los riesgos y oportunidades 16
Cuestiones externas e internas que afectan la etapa de desarrollo preliminar del software 21
Cuestiones externas que afectan la etapa de desarrollo preliminar del software 21
Cuestiones internas que afectan la etapa de desarrollo preliminar del software 25
Determinación de las partes interesadas en la etapa de desarrollo preliminar de software 28
Identificación del riesgo en los diagramas de flujo y en los archivos del software desarrollado 29
Diagramas de flujo 29
Archivos de código 31
Aplicación de la matriz de prioridades para establecer el nivel de riesgo 33
Valoración del riesgo 41
Acciones para abordar riesgos y oportunidades: Tratamiento del riesgo 42
Preparación e implementación de los planes para el tratamiento del riesgo 44
Resultados 53
Pensamiento basado en riesgos aplicado a las sub etapas del desarrollo preliminar del software 53
Pensamiento basado en riesgos aplicado a los segmentos de navegación, búsqueda y filtrado. 60
Archivos del software 67
Optimizado del tiempo de construcción en los segmentos clave 70
Matriz de riesgos residuales 71
Nuevos riesgos detectados a raíz de la implementación de acciones de tratamiento del riesgo 76
Conclusiones 76
Referencias bibliográficas 78
Anexo: Bibliografía 79

Objetivos

Objetivo general

Demostrar que aplicando el pensamiento basado en riesgos a la etapa de desarrollo preliminar de software es posible optimizar el tiempo de desarrollo.

Objetivos particulares

- Determinar los factores internos, factores externos y partes interesadas que afectan la etapa de desarrollo preliminar de software.
- Identificar los riesgos y su magnitud con ayuda de la matriz del análisis de riesgos y aplicando los apartados 4.1, 4.2 y 6.1 de la norma ISO 9001:2015 en conjunto con los apartados 5.4.1, y 6.4 y 6.5 de la norma ISO 31000:2018
- Implementar acciones a los segmentos de navegación, búsqueda y filtrado del software conforme a lo establecido en las normas ISO 9001:2015 e ISO 31000:2018.
- Comparar el tiempo de desarrollo de los segmentos de navegación, búsqueda y filtrado de la versión preliminar del software respecto de los desarrollados aplicando el pensamiento basado en riesgos.

Conceptos y definiciones para el manejo del software

El ciclo de vida de desarrollo de software

El ciclo de vida de desarrollo de software SDLC por sus siglas en inglés es una serie de fases o tareas definidas para diseñar y desarrollar un producto de software de manera coherente, eficiente y ordenada, de acuerdo con ISO, la Comisión Electrotécnica Internacional (IEC) y el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE), el ciclo tiene un enfoque en fases y estas se descomponen en sub etapas a ser completadas.

Etapas del ciclo de desarrollo de software (SDLC)

Si bien la mayoría de autores sostienen que las fases que integran al ciclo de vida de desarrollo de software dependen del modelo a emplear, Mc Leord (2009) y Chaitra (2019) sostienen que existen algunas fases en común en los modelos de ciclo de vida actuales, Mc Leord (2009) establece que un modelo de ciclo de vida para el desarrollo de software consiste en 4 etapas:

1. *Investigación preliminar - en esta fase se evalúa la factibilidad del proyecto, se evalúan los riesgos, se plantean objetivos y se establecen indicadores de desempeño, la etapa concluye cuando los desarrolladores obtienen autorización para proceder por parte de las partes interesadas.*

2. *Análisis - En esta etapa se determinan los requisitos funcionales de acuerdo a la organización y posteriormente se documentan los requisitos*

3. *Diseño - Una vez que se tienen los requisitos funcionales se procede a identificar y evaluar los diseños factibles, la etapa concluye cuando se selecciona el mejor diseño.*

4. *Construcción preliminar - Esta etapa se enfoca en construir una versión temprana en base al diseño seleccionado de acuerdo a los requisitos y es sometida a pruebas de aceptación, la etapa concluye cuando la versión es aprobada y entonces se sigue la ruta de despliegue.*

Por otro lado Chaitra (2019) sostiene que se el SDLC abarca 5 etapas o fases:

1. *Análisis de requisitos - En esta etapa se capturan todos los requisitos del cliente, se analizan y se establece un plan de proyecto*

2. *Diseño de software - en esta etapa se discuten los aspectos técnicos de las pates interesadas respecto al diseño, se establecen los criterios de selección y finalmente se selecciona el mejor diseño de software*

3. *Codificado de software - se realiza todo el código del proyecto en base a la selección del mejor diseño.*

4. *Etapa de pruebas de software - En esta etapa se realizan evaluaciones al software desarrollado con la finalidad de verificar que este cumpla con todas las especificaciones del plan de proyecto antes de entregarse como versión final al cliente.*

5. *Mantenimiento del software - el objetivo de esta etapa es arreglar los posibles errores que el producto pudiera presentar después de ser entregado, usualmente en forma de parche de versión o si es un problema mínimo se corregirá en la siguiente versión del producto.*

Ciclo de vida empleado para el desarrollo del software

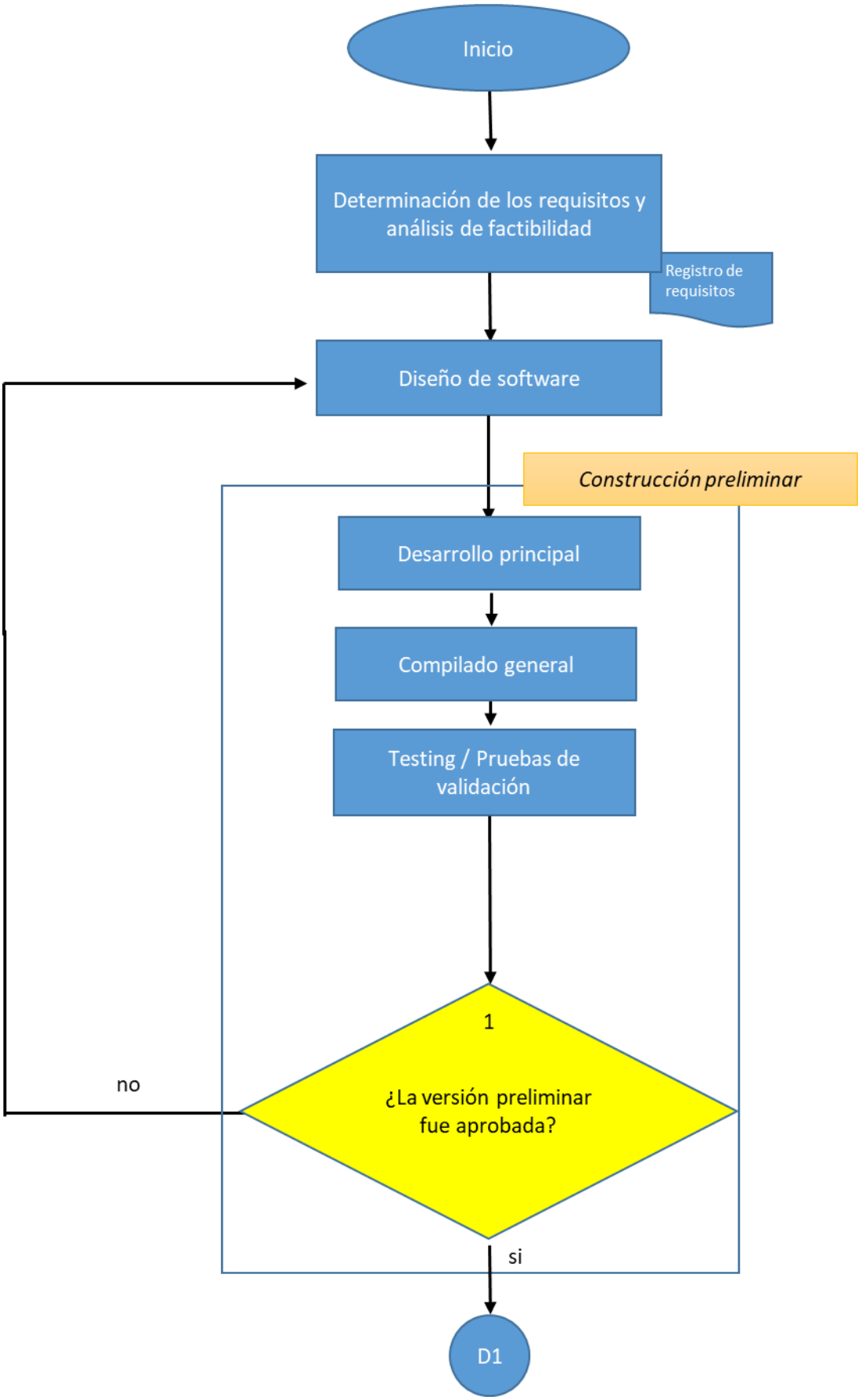
El modelo de ciclo de vida de desarrollo de software empleado contempla etapas similares a las descritas por Mc Leord (2009) y por Chaitra (2019).

A continuación se presenta la tabla 1 que describe las etapas que se emplearon para desarrollar el software.

Tabla 1. Etapas del ciclo de vida empleado para el desarrollo del software

Etapa	Descripción
Determinación de requisitos de cliente y análisis de factibilidad	• En esta fase se identifican los requisitos técnicos señalados por el cliente y se lleva a cabo levantamiento. • Se lleva a cabo el estudio completo de factibilidad. • Se acuerdan los requisitos técnicos en concordancia con la factibilidad.
Diseño del software	En esta etapa se establecen los objetivos y metas para el software y se diseñan varios modelos, la etapa concluye cuando se define el modelo a desarrollar.
Etapa de construcción preliminar	En esta etapa se construye el código del diseño elegido para desarrollar el software y se somete a diversas pruebas de aceptación por el equipo de desarrollo y por el cliente hasta llegar a una versión que cumpla con la funcionalidad y la estabilidad que se requiere
Lanzamiento y actualización	• En esta etapa se entrega el software final • Se implementa un plan de actualizaciones durante un periodo de tiempo acordado con el cliente con la finalidad de corregir defectos de acuerdo a ISO 9001:2015 apartado de control de cambios.

El diagrama 1 muestra el ciclo de vida empleado para el desarrollo del software.


Inicio
Determinación de los requisitos y análisis de factibilidad
Registro de requisitos
Diseño de software
Construcción preliminar
Desarrollo principal
Compilado general
Testing / Pruebas de validación
1
¿La versión preliminar fue aprobada?
no
si
D1

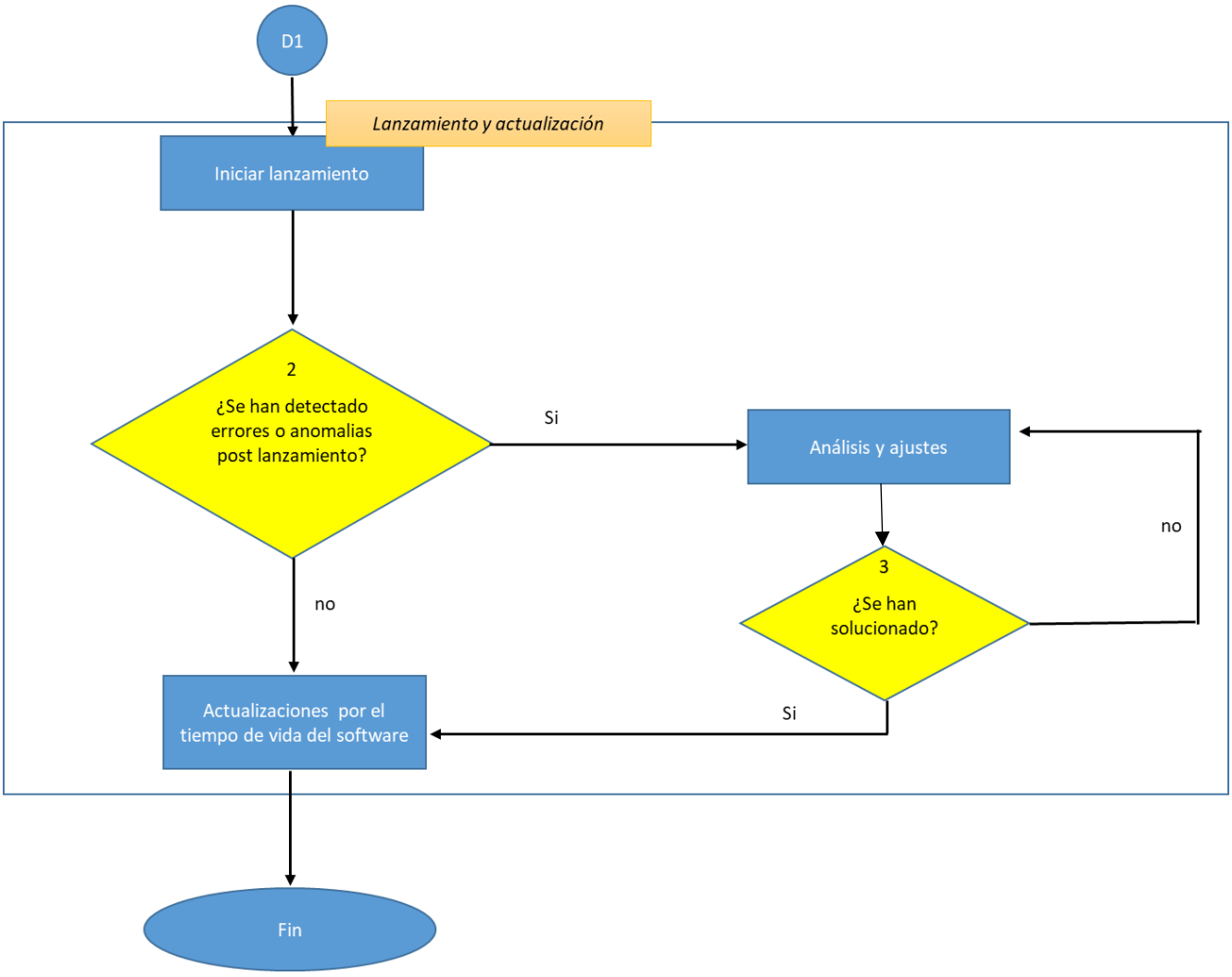


Diagrama 1. Flujo del ciclo de vida empleado para el desarrollo del software

La etapa de desarrollo preliminar

La etapa de desarrollo preliminar consiste en construir el código del diseño seleccionado para el software, es sometido a diversas pruebas por el equipo de desarrollo y por el cliente hasta que se llega a una versión que cumpla con la estabilidad y funcionalidad requerida.

La etapa de desarrollo preliminar está constituida por 3 sub etapas o sub fases mismas que se señalan a continuación:

Primera sub etapa: Desarrollo principal

El desarrollo principal consiste en construir el código para cada uno de los segmentos que integrarán el software de manera individual, una vez que son construidos se someten a una prueba de módulo para verificar que el código construido no presenta errores en su estructura, en caso de que el segmento no pase la prueba de unidad se debe reconstruir el código en base a los errores o deficiencias que se hayan detectado, esto se observa en el diagrama 2.

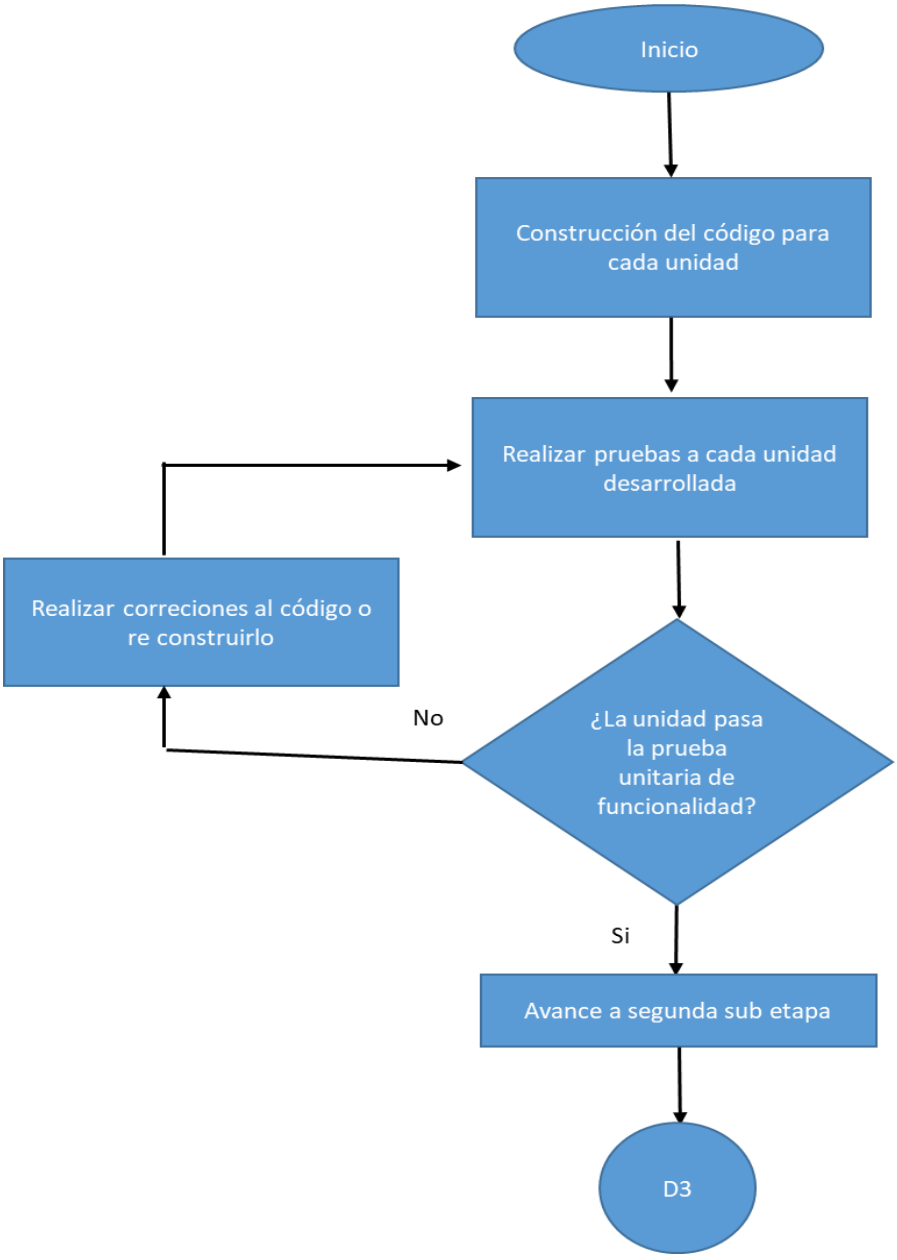


Diagrama 2. Flujo del desarrollo principal.

Segunda sub etapa: Compilado general

El compilado general tiene como finalidad probar de manera conjunta, la funcionalidad y la estabilidad de los segmentos desarrollados para comprobar que los segmentos se comunican de manera correcta y no presentan fugas de datos o cierres repentinos, cuando se presentan inconsistencias al compilar el sistema integrado durante la prueba se conduce un análisis de causas y se implementa una solución, una vez que se han solucionado los errores se vuelve a realizar el compilado general para asegurar la correcta funcionalidad de todo el sistema, esto se observa en el diagrama 3.

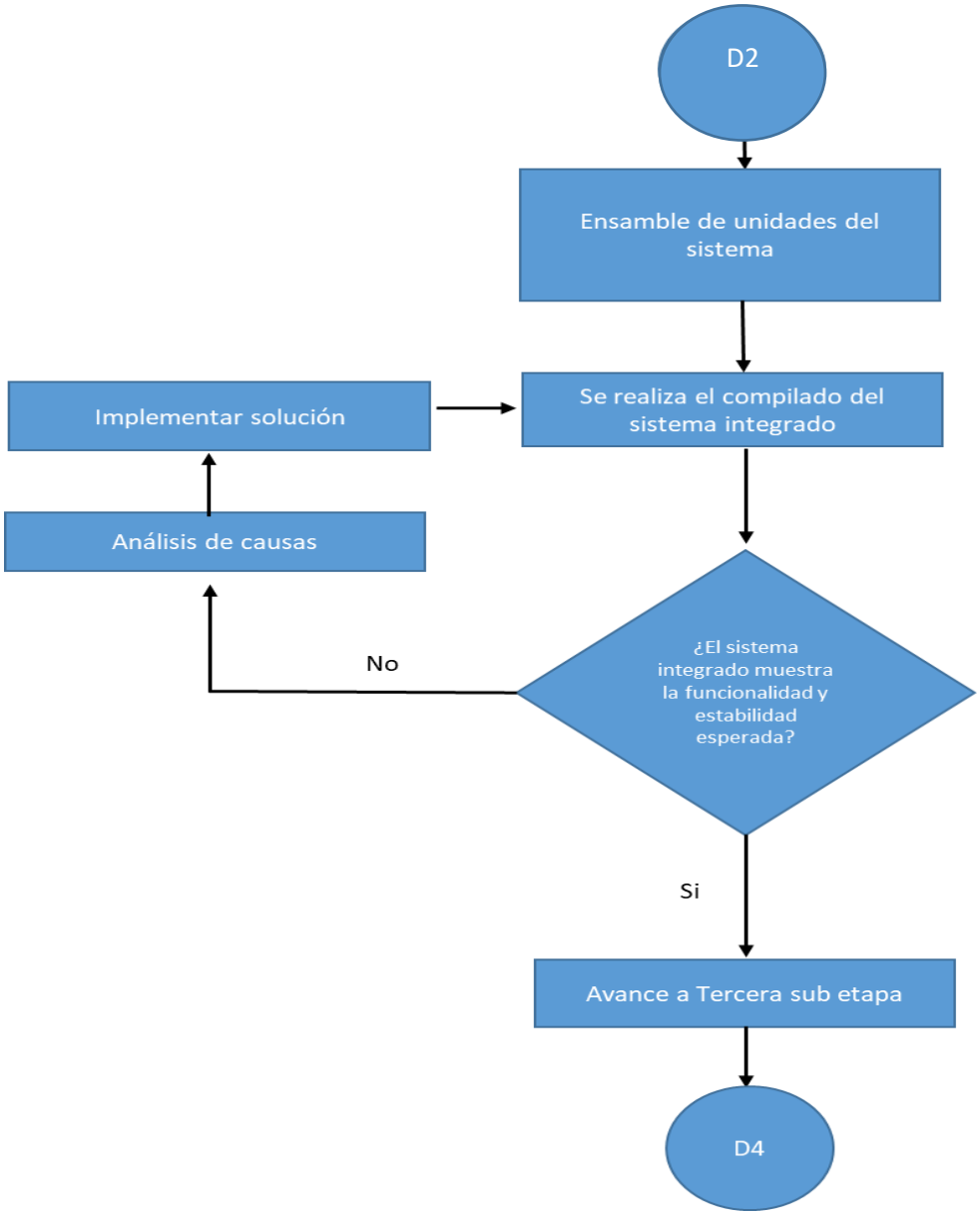


Diagrama 3. Flujo del compilado general

Tercera sub etapa: Pruebas de aceptación o validación

Las pruebas de aceptación o validación se llevan a cabo una vez que el sistema funciona de manera integral, estas pruebas tienen la finalidad de demostrar el cumplimiento de los requisitos del sistema en diversos casos para los cuales el software está pensado, el software debe pasar satisfactoriamente cada prueba para obtener aprobación y poder avanzar a la etapa de lanzamiento, en caso contrario se debe realizar un análisis de causas, rechazar la versión y regresar a la etapa 2 del ciclo de vida para reconstruir el software hasta que pase las pruebas de validación, esto se observa en el diagrama 4.

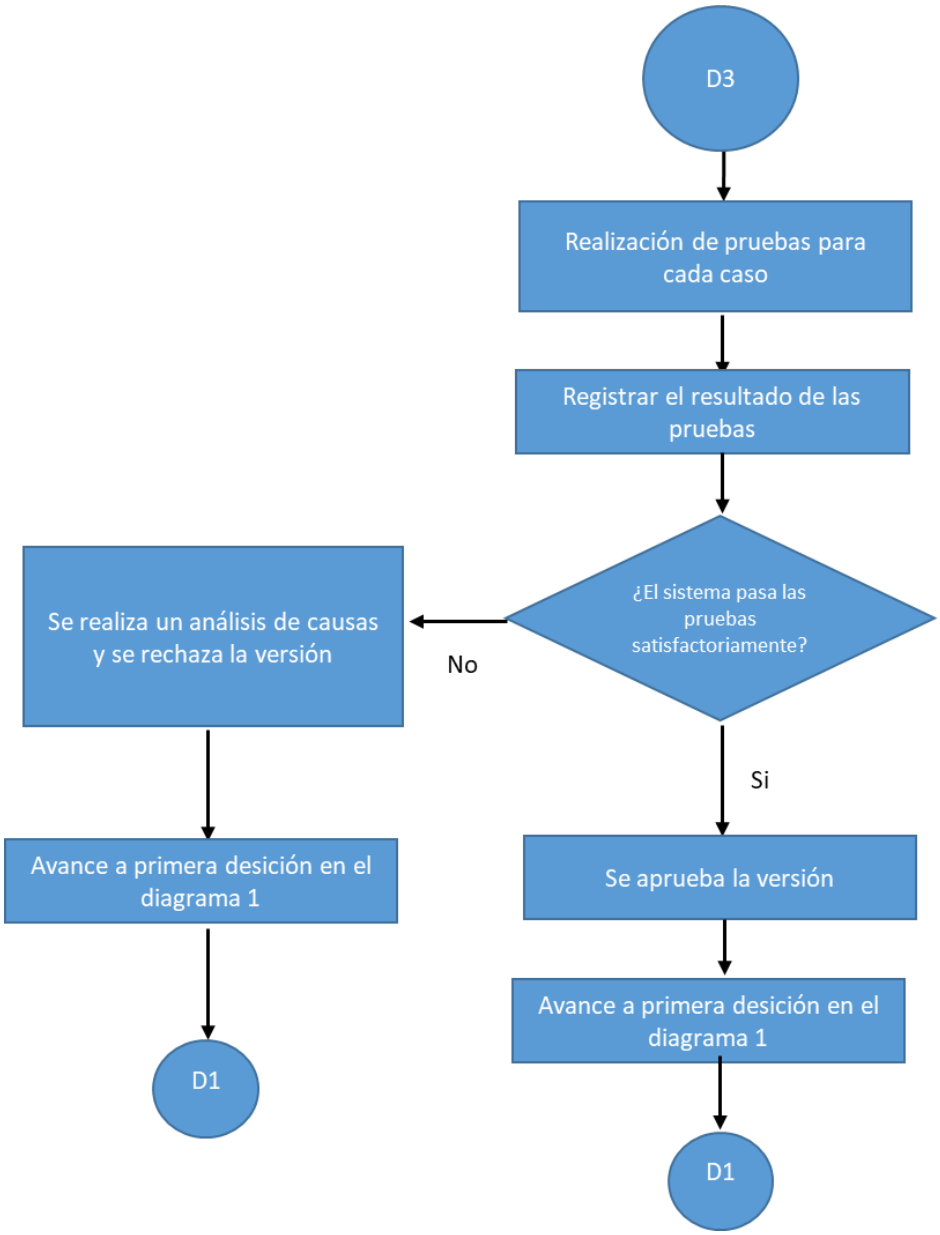


Diagrama 4. Flujo de las pruebas de aceptación o validación.

Segmentos clave del proyecto

Un segmento clave es una unidad de código la cual es parte fundamental para el correcto funcionamiento del software, estos segmentos pueden ser de navegación, procesado, almacenamiento, comunicación entre otros, en el caso de que uno o más segmentos estén ausentes o mal implementados en el software este podría no funcionar adecuadamente o no funcionar.

El software desarrollado cuenta con 3 segmentos clave los cuales se describen a continuación.

1. **Navegación**

Este segmento permite el movimiento a través de las diferentes secciones del software, es vital que exista un bloque, unidad o mapa de navegación dentro de los archivos del código para un correcto funcionamiento de la interfaz de usuario así como de la navegación interna en el software. Este segmento suele tener su propio archivo el cual esta enlazado a cada unidad del software.

En el caso de navegación, el software tiene implementado un mapa que representa el movimiento entre unidades, de acuerdo al diagrama 5:

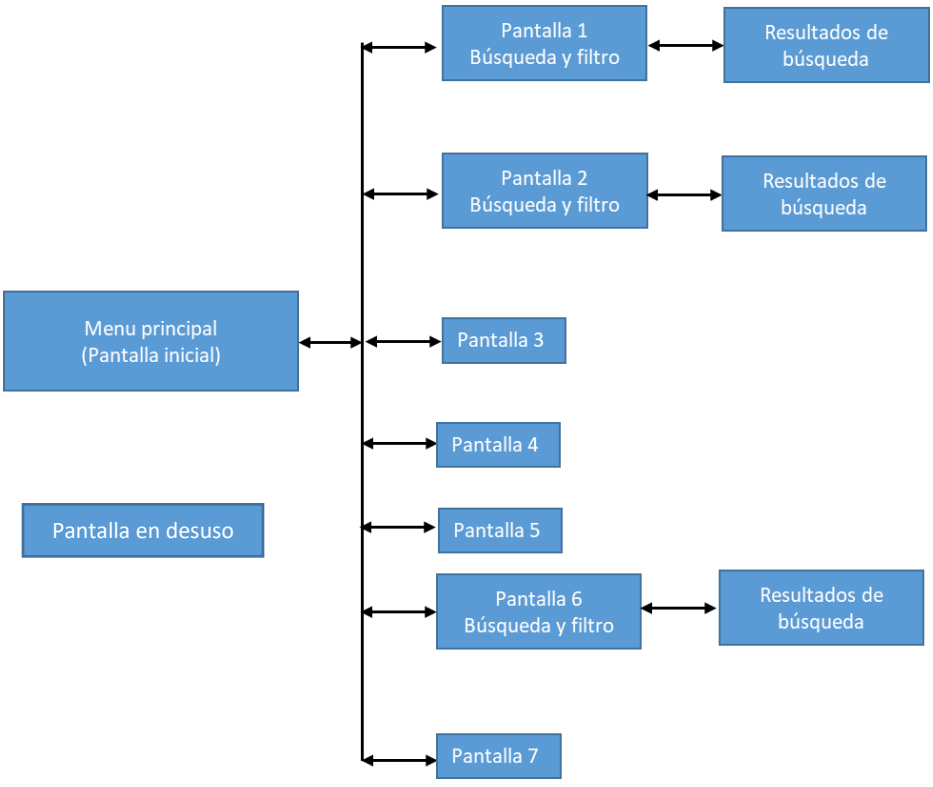


Diagrama 5. Mapa de navegación del software.

En el diagrama 5 las flechas dobles indican la navegación entre las diferentes pantallas a la pantalla principal.

2. Búsqueda

El software desarrollado tiene integradas bases de datos por lo cual es vital integrar un sistema de búsqueda adecuado, ingresar una solicitud y recibir un resultado o resultados de búsqueda. Este segmento está presente en cada archivo diseñado para recibir solicitudes de búsqueda en conjunto con el segmento de filtrado.

El segmento de búsqueda se encuentra implementado en 3 de los 6 segmentos que presenta el diagrama 5 mediante una barra de búsqueda. El diagrama 6 muestra el funcionamiento del segmento.

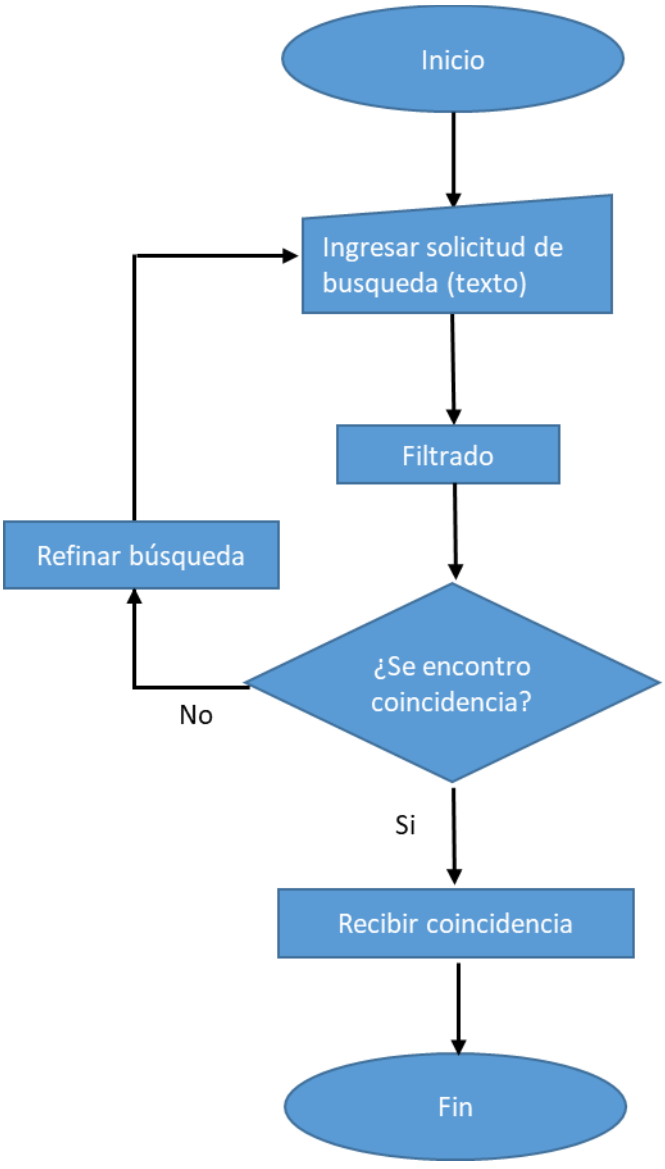


Diagrama 6. Flujo del segmento de búsqueda

3. Filtrado

En conjunto con el segmento de búsqueda, el segmento de filtrado se encarga de adecuar los posibles resultados de la búsqueda solicitada, sin un segmento de filtrado o mal implementado no se obtendría ningún resultado o serian inconsistentes con lo solicitado por el usuario. Este segmento está presente en cada archivo diseñado para recibir solicitudes de búsqueda en conjunto con el código de búsqueda, esto se observa en el diagrama 5.

El siguiente diagrama describe la operación del segmento de filtrado.

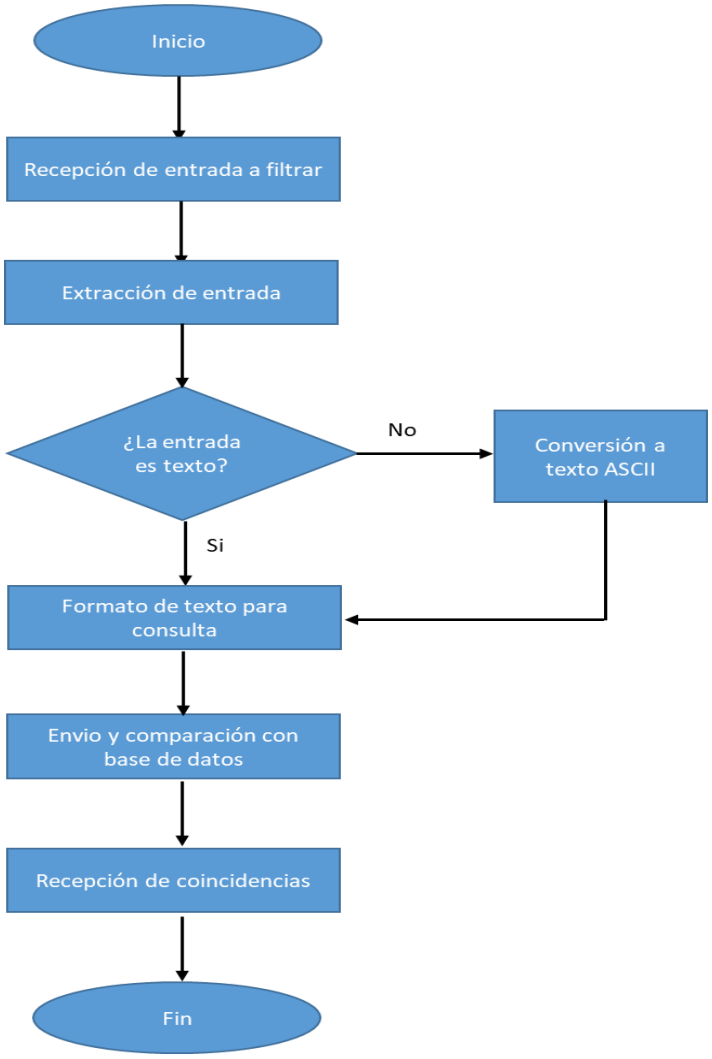


Diagrama 7. Flujo del segmento de filtrado

Nota: En cada una de las fases del proceso de filtrado se debe asegurar que la solicitud de entrada sea un texto en formato ASCII con la finalidad de que se ejecute correctamente el segmento

Conceptos y definiciones sobre ISO 9001:2015 Sistemas de gestión de la calidad: Requisitos

Definición de riesgo

En base a las normas ISO 9001:2015 y 9000:2015 el riesgo se define como el efecto de la incertidumbre sobre la consecución de los objetivos, este efecto puede ser positivo o negativo y con frecuencia se expresa como combinación de sucesos potenciales y sus consecuencias

Pensamiento basado en riesgos

El pensamiento basado en riesgos es un concepto que permite a la organización comprender su contexto, determinar los riesgos existentes como base para la planificación y actuar como herramienta preventiva, el concepto de acción preventiva se aborda en la normatividad ISO mediante el pensamiento basado en riesgos.

El pensamiento basado en riesgos está definido en 2 apartados principales:

1. Contexto de la organización y partes interesadas (Requisito 4.1 y 4.2 ISO 9001:2015)
2. Acciones para abordar riesgos y oportunidades(Requisito 6.1 ISO 9001:2015)

Contexto de la organización

Ambas normas, ISO 9001:2015 e ISO 31000:2018 establecen un concepto similar para abordar el concepto de contexto de la organización, la tabla 2 muestra ambos conceptos.

Tabla 2. Concepto Contexto de la organización abordado por las normas ISO 9001:2015 e ISO 31000:2018.

Norma	Contexto de la organización	Cuestiones internas	Cuestiones externas
ISO 9001:2015 Sistemas de gestión de la calidad: Requisitos Apartado 4.1 y anexo A4	La organización debe establecer las cuestiones externas e internas que afectan la capacidad de consecución de sus objetivos y de dar resultados, se debe dar seguimiento a estas cuestiones con la finalidad de adaptarse a las situaciones que estas pueden ocasionar	Respecto a las cuestiones internas estas pueden ser determinadas al considerar 1. Valores 2. Cultura interna 3. Conocimientos de la organización 4. Desempeño de la organización	De acuerdo con la norma ISO 9001:2015 las cuestiones externas pueden determinarse con más facilidad al considerar el entorno 1. Legal 2. Tecnológico 3. Competitivo 4. Mercadológico 5. Social 6. Cultural 7. Económico
ISO 31000:2018 Directrices para abordar riesgos y oportunidades Apartado 5.4.1	la organización debería analizar y comprender su contexto interno y externo como base para la planeación del riesgo	En el caso del contexto interno A. La identidad organizacional B. La estrategia, objetivos y políticas C. La cultura de la organización D. Normas, directrices y modelos adoptados o establecidos por la organización E. Datos, sistemas de información y flujos de información	Para identificar el contexto externo la norma ISO 31000:2018 establece los siguientes factores A. Factores sociales, económicos, políticos, culturales, legales, reglamentarios y tecnológicos a nivel internacional, nacional, regional o local en su defecto. B. Tendencias que afectan los objetivos de le empresa C. Las percepciones, necesidades y expectativas de las partes interesadas externas D. Las relaciones contractuales

Partes interesadas

Las partes interesadas son *todas aquellas entidades que pueden afectar, ser afectadas o percibirse como afectadas por una decisión o actividad*, por esto la organización debe determinar

1. Las partes interesadas pertinentes
2. Los requisitos pertinentes de estas partes interesadas

Además la organización debe dar seguimiento y documentar los requisitos pertinentes de las partes interesadas determinadas.

Acciones para abordar riesgos y oportunidades

De acuerdo con la norma ISO 9001:2015 la organización *deberá tomar en cuenta sus cuestiones internas y externas así como las partes interesadas pertinentes con la finalidad de determinar los riesgos y oportunidades presentes*, para:

1. *Aumentar los efectos deseables*
2. *Prevenir o reducir efectos no deseados*
3. *Lograr la mejora*

Además la organización debe planificar

1. *Las acciones para abordar riesgos y oportunidades*
2. *La integración e implementación de dichas acciones*
3. *La evaluación de la eficiencia de las acciones tomadas*

Las acciones que se determinen deben ser coherentes con la magnitud del riesgo detectado, algunas opciones que establece la norma ISO 9001:2015 para abordar riesgos son

1. *Evitar el riesgo*
2. *Asumir riesgos para perseguir una oportunidad*
3. *Eliminar la fuente del riesgo*
4. *Cambiar las probabilidades o consecuencias.*
5. *Compartir el riesgo*
6. *Mantener el riesgo mediante decisiones informadas*

Conceptos y definiciones de la ISO 31000:2018 Directrices para abordar los riesgos y oportunidades

Evaluación del riesgo

De acuerdo con la norma ISO 31000:2018 es conveniente que la organización lleve a cabo el proceso de evaluación del riesgo de manera sistemática e iterativa basándose en el conocimiento sobre las partes interesadas de manera que la evaluación sea la base para la planeación de sus actividades, está se compone de 3 secciones principales descritas en la tabla 3.

Tabla 3 Secciones que componen la evaluación del riesgo de acuerdo a la norma ISO 31000:2018 Directrices para abordar riesgos y oportunidades

Evaluación del riesgo		
Sección	Objetivo	Acciones a tomar en cuenta o apartados sugeridos por la norma para cada sección
Identificación del riesgo	Encontrar, reconocer y describir los riesgos que pueden beneficiar o perjudicar a la organización en el logro de sus objetivos independientemente de si las fuentes de riesgo están o no bajo su control.	1. Fuentes de riesgo tangibles e intangibles 2. Causas y eventos 3. Amenazas y oportunidades 4. Vulnerabilidades y capacidades 5. Cambios en el contexto interno y externo 6. Las consecuencias y su impacto en los objetivos 7. Limitaciones de conocimiento e información 8. Factores relacionados al tiempo
Análisis del riesgo	Comprender la naturaleza del riesgo, sus características y su nivel	1. Incertidumbres presentes 2. Fuentes de riesgo 3. Posibles consecuencias 4. Probabilidades 5. Eventos y escenarios 6. Controles

Evaluación del riesgo		
Sección	Objetivo	Acciones a tomar en cuenta o apartados sugeridos por la norma para cada sección
Valoración del riesgo	Apoyar la toma de decisiones al comparar los resultados obtenidos en el análisis de riesgo con los criterios de riesgos establecidos por la organización para determinar cuándo se requiere tomar acciones.	A. No hacer nada más B. Conducir un segundo estudio de análisis para comprender mejor el riesgo C. Considerar opciones para el tratamiento de riesgo D. Mantener los controles existentes E. Replantear los objetivos

Tratamiento del riesgo

En base a la norma ISO 31000:2018 el objetivo principal del tratamiento del riesgo es seleccionar e implementar opciones para abordar el riesgo mediante un proceso iterativo que contempla

A. *Formular y seleccionar es para el tratamiento del riesgo*
B. *Planificar e implementar el tratamiento del riesgo*
C. *Evaluar la eficacia del tratamiento*
D. *Decidir si el riesgo residual es aceptable*
E. *En caso de no serlo realizar un tratamiento adicional iterando este proceso*

Selección de acciones para el tratamiento del riesgo

La selección de opciones para abordar el riesgo implica analizar los beneficios potenciales de lograr los objetivos respecto del costo de implementación, esfuerzo o desventajas que pudieran implicar, cabe mencionar que las opciones para abordar riesgo no son excluyentes y es posible implementar varias opciones pero también no todas las opciones son viables en todas las circunstancias.

De acuerdo con la norma ISO 3100:2018 las opciones para el tratamiento del riesgo deberían *considerar los objetivos de la organización, partes interesadas, valores y percepciones*, además se debe tener en cuenta de que a pesar de la planeación de las opciones estas pueden no producir los resultados esperados o generar nuevas fuentes de riesgo que necesitará gestionarse por lo cual es necesario dar seguimiento y revisión a la implementación de opciones para gestionar el riesgo para asegurar que las distintas opciones del tratamiento sean y permanezcan eficaces. En caso de que no existan opciones para abordar los riesgos o que fueran insuficientes la norma indica que se debe llevar registro del riesgo de manera continua.

La norma ISO 31000:2018 sugiere algunas opciones para abordar el riesgo

A. *Evitar el riesgo al decidir no continuar o iniciar la actividad que genera el riesgo*
B. *Aceptar o aumentar el riesgo buscando una oportunidad*
C. *Eliminar la fuente del riesgo*
D. *Cambiar la probabilidad*
E. *Cambiar las consecuencias potenciales*
F. *Compartir el riesgo*
G. *Conservar el riesgo con base en una decisión informada*

Preparación e implementación de los planes para el tratamiento del riesgo

El objetivo de los planes de tratamiento de riesgos es establecer la forma en la que se implementaran las opciones elegidas para el tratamiento del riesgo, para esto es necesario dar seguimiento a dichos planes, integrar los planes de tratamiento a los planes de gestión de la organización y que las partes interesadas comprendan los planes.

La norma ISO 31000:2018 sugiere incluir los siguientes apartados en los panes de tratamiento.

Fundamento de la selección de opciones

Beneficios esperados

Responsables de la aprobación e implementación de los planes

Acciones propuestas

Recursos necesarios

Contingencias

Medidas del desempeño

Restricciones

Informes de seguimiento

Plazos previstos para la realización y finalización de acciones

Determinación del nivel de riesgo

Para realizar un análisis de riesgo se debe emplear una matriz de riesgos la cual permita visualizar, en base a la probabilidad de ocurrencia del riesgo y de las consecuencias de estos (impacto), el nivel de los riesgos analizados. De acuerdo con la OCDE es necesario establecer la probabilidad e impacto con ayuda de matrices para cada caso, para la probabilidad se tiene la matriz de probabilidad de riesgo de cumplimiento y para el impacto se tiene la calificación del impacto, ambas matrices deben ser individualizadas para cada caso de estudio.

Para el caso del software desarrollado se establecieron 3 niveles de probabilidad de en base a la OCDE y en función de su ocurrencia en las sub etapas del desarrollo del software, segmentos clave, archivos y diagramas de flujo, estos se ven reflejados en la tabla 4. En el caso del impacto, este se determina con ayuda de la tabla 5, donde este se describe en función de las consecuencias que hay sobre la etapa de desarrollo preliminar del software desarrollado y sobre la versión de software.

Tabla 4. Índice de probabilidad de ocurrencia de riesgo en la etapa de desarrollo preliminar del software

Índice de probabilidad	probabilidad	Descripción
1	baja	Se puede presentar rara vez en alguno de los apartados: diagramas de flujo, archivos del software, sub etapas o en los segmentos clave del software durante su construcción, compilado y funcionamiento.
2	media	Ocurre con frecuencia en los diagramas de flujo, archivos del software, sub etapas y en los segmentos clave del software durante su construcción, compilado y funcionamiento.
3	alta	Se presenta de forma constante en los diagramas de flujo, archivos del software, sub etapas y en los segmentos clave del software durante todas sus etapas de desarrollo.

Tabla 5. Índice de impacto en función de las consecuencias sobre la etapa de desarrollo preliminar de software

Valor del impacto	Nivel de impacto	Descripción del impacto
1	Relevante	Causa algunas interrupciones en el desarrollo de módulos y segmentos así como en la entrega de unidades para compilado reduciendo el tiempo para otras secciones.
2	Medio	Causa una disminución en el rendimiento global del software y anomalías en el funcionamiento de la versión preliminar del software retrasando la entrega de compilados para las pruebas de aceptación / validación.
3	Alto	Ocasiona errores graves en los segmentos clave del software y puede dejarlos inservibles provocando que la versión sea rechazada y se deba empezar desde el paso 2 del ciclo de vida, este tipo de riesgo puede causar la cancelación del proyecto, pérdida de clientes y pérdidas económicas.

De acuerdo con la OCDE la probabilidad y el impacto deben ser plasmados en una matriz de prioridades para establecer el nivel de riesgo al multiplicar el valor de la probabilidad por el del impacto, la matriz de prioridad se muestra en la tabla 6.

Tabla 6. Matriz de prioridad para la etapa de desarrollo preliminar en base a lo establecido en las tablas 5 y 6

Impacto / Probabilidad	Relevante (1)	Importante (2)	Prioritario (3)
Baja (1)	1	2	3
Media (2)	2	4	6
Alta (3)	3	6	9

Una vez determinada la matriz de prioridad, se define en la tabla 7 el tipo de gestión que se va a llevar a cabo para cada tipo de riesgo.

Tabla 7. Gestión en base al índice de riesgo.

Puntuación	Índice de riesgo	Tipo de gestión
Mayor o igual a 6	Prioritario	Cliente, alta dirección, responsables del desarrollo y gestión de procesos
Entre 2 y 4	Importante	Gestión por parte de los equipos de desarrollo
Igual a 1	Moderado	Gestión de procedimientos y procesos de rutina

Cuestiones externas e internas que afectan la etapa de desarrollo preliminar del software

Las normas ISO 9001:2015 e ISO 31000:2018 establecen que es necesario determinar las cuestiones internas y externas para determinar los riesgos y tomarlos como base para la planeación estratégica en logro de los objetivos, los puntos que establecen ambas normas se encuentran en la tabla 2.

Cuestiones externas que afectan la etapa de desarrollo preliminar del software

La tabla 8 muestra las cuestiones externas contempladas durante la etapa de desarrollo preliminar del software desarrollado, cabe mencionar que no se hizo uso ninguna herramienta para estudiar las cuestiones externas en esta etapa.

Tabla 8 Cuestiones externas tomadas en cuenta para la etapa de construcción preliminar del software desarrollado.

Tipo de cuestión externa (ISO9001:2015/31000:2018)	Cuestión	Relevancia
Legal	Licencia de uso de software de terceros	• Se emplean diversos softwares de terceros, para crear recursos visuales, generar archivos de interfaz o de servicio de bases de datos en la nube. • Es esencial contar con las licencias de uso para evitar conflictos legales tales cómo demandas o multas.
	Derechos de propiedad intelectual	• Registro de marca para la protección de la identidad del software • La protección del desarrollo del software • Evitar posibles infracciones hacia terceros por el uso parcial o total del software
Tecnológica	Aparición de nuevas tecnologías que modifican las prácticas actuales de desarrollo	• Dar seguimiento a este tipo de tecnologías permite a la organización preparar planes para abordar los riesgos y oportunidades que esta nueva tecnología pudiera ofrecer.
	Cambios en las tecnologías existentes para desarrollo de software	• Los métodos para construcción cambian o dejan de estar disponibles causando que se deban que buscar alternativas o teniendo que reconstruir segmentos enteros a causa de los cambios implementados. En escenarios más severos se deben dejar a sectores completos de público o clientes fuera debido a las restricciones que suponen los nuevos cambios.
	Actualizaciones en los métodos de construcción de código	• La manera en la que se construye el código puede cambiar de una actualización a otra. • Se implementan nuevas herramientas de desarrollo, nuevas funciones o se pueden eliminar librerías o funciones anteriores, por lo que dar seguimiento a los cambios idear planes estratégicos que minimicen los percances de cambios futuros o actuales.

Tomando en cuenta las cuestiones anteriores y las indicaciones de la norma ISO 9001:2015 y 31000:2018 se propone considerar las cuestiones de la tabla 9.

Tabla 9 Cuestiones externas propuestas en base a ISO 9001:2015 y 31000:2018

Tipo de cuestión externa (ISO9001:2015/31000:2018)	Cuestión	Relevancia
Sociocultural	Tendencias del consumidor y sus cambios	• Conocer las tendencias de los usuarios y clientes respecto de la arquitectura de software, interfaz de usuario y funciones permite crear productos que cumplan sus expectativas. • Es necesario informarse sobre estos aspectos con frecuencia para detectar e implementar oportunidades de mejora
Ecológicos	Regulaciones en el consumo energético	• Al existir una mala administración energética los costos de desarrollo del software incrementan.
Políticos	Políticas de privacidad y uso de datos	Dado que el software desarrollado se encuentra disponible en 40 países se requiere: • Amplio conocimiento sobre como la nación maneja la privacidad y uso de datos de sus ciudadanos ante las industrias tecnológicas. En México existe la Ley federal de protección de datos personales en posesión de particulares • Tener estos conocimientos permite a la industria adaptarse a las políticas de cada entidad federativa informando y solicitando permiso de uso a los usuarios o clientes
Económicos	Inflación	El precio de un proyecto puede llegar a ser afectado por fenómenos económicos a medida que pasa el tiempo, por lo que se pudieran establecer condiciones contractuales que permitan mantener un precio de proyecto justo-

En base a la tabla 9 se establece la tabla 10 análisis PESTEL

Tabla 10. Análisis PESTEL de las cuestiones externas que afectan la etapa de desarrollo preliminar propuestas

Políticos			Económicos		
Cuestión	Situación		Cuestión	Situación	
	Riesgo	Oportunidad		Riesgo	Oportunidad
Políticas de privacidad y uso de datos	x		Inflación	x	
Socioculturales			**Tecnológicos**		
Cuestión	Situación		Cuestión	Situación	
	Riesgo	Oportunidad		Riesgo	Oportunidad
Tendencias del consumidor y sus cambios		x	Aparición de nuevas tecnologías que modifican las prácticas actuales de desarrollo		x
			Cambios en las tecnologías existentes para desarrollo de software	x	
			Actualizaciones en los métodos de construcción de código	x	
Ecológicos			**Legales**		
Cuestión	Situación		Cuestión	Situación	
	Riesgo	Oportunidad		Riesgo	Oportunidad
Regulaciones en el consumo energético	x		Licencia de uso de software de terceros	x	
			Derechos de propiedad intelectual	x	

Cuestiones internas que afectan la etapa de desarrollo preliminar del software

Al igual que en el caso anterior ambas normas establecen que es necesario determinar las cuestiones internas para la planeación estratégica, en este caso las cuestiones internas tomadas en cuenta para el desarrollo del software se pueden ver en la tabla 11.

Tabla 11. Cuestiones internas consideradas para la etapa de construcción preliminar del software desarrollado

Cuestión identificada en base a ISO 9001:2015/31000:2018	Relevancia
Conocimientos de la organización	Ser conscientes de que conocimientos se tienen en la organización y en sus dependencias permite administrar el software para lograr los objetivos y metas planteados en el proyecto.
Desempeño del desarrollo del software	Dado que se maneja una gestión de cumplimiento de objetivos y metas, es importante: • Tener en cuenta indicadores de desempeño como por ejemplo objetivos cumplidos/objetivos totales o pruebas de unidad aprobadas/pruebas totales.
Estrategia y objetivos	Tener identificadas las metas y los objetivos así como las diversas opciones de cómo llegar a su cumplimiento permite: • Enfocarse en los problemas y sus soluciones para lograr los objetivos. • Visualizar alternativas para lograr objetivos

En base a la tabla 11 y considerando las sugerencias de la norma, se propone integrar 2 puntos más viéndose reflejados en la tabla 12

Tabla 12. Cuestiones internas propuestas en base a las indicaciones de ISO 9001:2015 e ISO 31000:2018.

Cuestión en base a ISO 9001:2015/31000:2018	Relevancia
Modelos y directrices desarrollados o empleados por la organización para el desarrollo del software	Tomar en cuenta los modelos desarrollados en la etapa de desarrollo preliminar permite: • Reducir el tiempo de construcción de código • Iterar sobre modelos anteriores para pulirlos y obtener mejor rendimiento
Flujos de información generados por la organización para el desarrollo del software	Los diagramas de flujo y flujos de información permiten: • Visualizar en orden las entradas, salidas, decisiones y procesos que debe cumplir el módulo • Establecer métodos y alternativas para construcción del software.

En base a las tablas 11 y 12 se establece la tabla 13 análisis FODA el cual solo contempla las cuestiones internas que afectan la etapa de desarrollo preliminar del software desarrollado.

Tabla 13. Análisis FODA sobre las cuestiones internas que afectan la etapa de desarrollo preliminar del software desarrollado.

Fortalezas	Debilidades
Estrategia y objetivos: la organización establece claramente los objetivos a cumplir durante la etapa de desarrollo preliminar lo que permite: 1. Determinar las opciones adecuadas para el logro de los objetivos 2. De ser necesario abordar los objetivos desde otra perspectiva dentro del contexto de desarrollo de software Desempeño del software: la organización tiene indicadores sólidos sobre el desempeño en la etapa de desarrollo preliminar como: 3. Pruebas unitarias aprobadas/pruebas totales 4. Tiempo de compilado 5. Cantidad de errores o advertencias/módulos totales	Conocimientos de la organización Debido a que las tecnologías cambian de forma frecuente la organización debería considerar implementar planes de capacitación o actualización constantes con el fin de: 1- Asegurar el uso correcto del software 2- identificar riesgos 3- Identificar oportunidades derivadas de los cambios tecnológicos.
Oportunidades	**Amenazas**
Modelos y directrices desarrollados o empleados por la organización en la etapa de desarrollo preliminar La organización si bien cuenta (de manera empírica) con modelos o directrices para abordar la etapa de desarrollo preliminar, debe plantear metodologías para implementar de manera sólida modelos y directrices que permitan: 1. Reducir el tiempo de desarrollo al emplear modelos y directrices establecidos y debidamente documentados. 2. Tener múltiples alternativas conocidas para su implementación de acuerdo a los objetivos de desarrollo.	Flujos de información generados por la organización Los flujos de información empleados para la etapa de desarrollo preliminar no consideran el manejo de excepciones para todos los casos que son vitales para el funcionamiento de los segmentos clave y solo cubren lo más evidente, los casos que no son completamente cubiertos en el manejo de excepciones son: 1. Cierre completo estructuras condicionales. 2. Ingreso de tipo de datos erróneo 3. Segmentos retirados que se conservan dentro del código del software

Determinación de las partes interesadas en la etapa de desarrollo preliminar de software

De acuerdo con la definición de la norma ISO 9001:2015 sobre las partes interesadas se ha establecido la tabla 14 para determinarlas

Tabla 14. Determinación de las partes interesadas en el desarrollo preliminar del software de acuerdo a la definición de la norma ISO 9001:2015.

Parte interesada	Forma en la cual afectan al desarrollo preliminar o se ven/perciben afectadas
Cliente	• Asigna el proyecto a la organización • Al recibir la versión preliminar, evalúa la factibilidad continuar el proyecto. • Evalúa la funcionalidad de la versión preliminar del software desarrollado.
Equipos desarrolladores	• Estiman el tiempo definido para el desarrollo del software. • Definen el plan de acción para el desarrollo y puesta en marcha del software de acuerdo con el tiempo establecido.
Organización desarrolladora	• Dar cumplimiento al plan de acción para asegurar el pago de los servicios. • Asegurar la asignación de recursos humanos necesarios para el desarrollo del software
Proveedores externos de herramientas de software	• Proporcionan el lenguaje y herramientas de programación para el desarrollo del software.
Usuario final	• Se perciben afectados por la protección de sus datos personales. • Determina si la interfaz es amigable.

Identificación del riesgo en los diagramas de flujo y en los archivos del software desarrollado

Diagramas de flujo

Los riesgos presentes en los diagramas de flujo tanto de las sub etapas así como de los segmentos clave se derivan principalmente de una planeación que no contempla los riesgos de la etapa de desarrollo preliminar del software, esto causa deficiencias en los diagramas de flujo del software.

De acuerdo con la norma ISO 31000:2018 la identificación del riesgo en los diagramas de flujo se hace tomando en cuenta:

a) Amenazas y oportunidades
b) Factores relacionados al tiempo
c) Las consecuencias sobre los objetivos

Tabla 15. Identificación del riesgo en los diagramas cómo función de las observaciones realizadas.

Sub etapa o segmento	Diagrama	Observaciones de acuerdo a ISO 31000:2018	Riesgos
Desarrollo principal	Diagrama 2 pág. 6	• No existe un proceso de revisión de correcciones que evite incorporar medidas que no arreglen los defectos	• Repetir el proceso más veces de las necesarias consumiendo más tiempo del establecido • Generar soluciones de poco impacto o inadecuadas que no arreglen los defectos detectados
Compilado general	Diagrama 3 pág. 7	• En la sub etapa de compilado general no se contempla la planeación para la implementación de acciones para solucionar los errores, únicamente se contempla en análisis y la implementación	• Se consume más tiempo debido a que no se tiene establecido un protocolo por lo que cada vez que ocurre todo debe ser pensado desde 0
Pruebas de aceptación o validación	Diagrama 4 pág. 8	• El diagrama solo contempla la realización de pruebas no se considera la planeación de casos de uso para el correcto desempeño en las pruebas	• El software puede no cumplir con los requisitos al no planearse correctamente los casos de prueba teniendo que reelaborar el diseño completo

Sub etapa o segmento	Diagrama	Observaciones de acuerdo a ISO 31000:2018	Riesgos
Navegación	Diagrama 5 pág. 9	• El árbol de navegación no contempla la movilidad entre pantallas de resultados pues esto se implementó sobre la marcha • No existe como tal un diagrama de flujo que describa cómo debe funcionar la navegación • El árbol de navegación es una representación gráfica del movimiento en el software y no un diagrama de flujo que describa el funcionamiento del módulo como tal	• el desarrollo de otras secciones se ve afectado por la implementación de características no requeridas, consumiendo tiempo y recursos • El código escrito puede no cumplir con la funcionalidad esperadas • Se consume más tiempo de desarrollo al establecer el funcionamiento de la navegación, retrasando el desarrollo de otras secciones
Búsqueda	Diagrama 6 pág. 10	• No se indica que la entrada de solicitudes de búsqueda se actualiza en tiempo real cada segundo	• Genera sesgos sobre la percepción del funcionamiento en el segmento, creando problemas de interacción
Filtrado	Diagrama 7 pág. 11	• el proceso de recepción de resultados corresponde al diagrama de búsqueda, el proceso que hace falta en la recolecta de coincidencias con la solicitud recibida	• Causa interrupciones constantes el desarrollo del segmento reduciendo el tiempo de desarrollo para los demás segmentos

Archivos de código

En el caso de los archivos de código de los segmentos clave los riesgos se derivan principalmente de la ejecución de los planes plasmados en forma de diagramas de flujo así como de las prácticas de los desarrolladores.

La tabla 16 muestra la identificación de los riesgos en los distintos archivos de código que conforman los 3 segmentos clave a partir de las observaciones realizadas en base a la norma ISO 31000:2018 y tomando en cuenta:

a) Amenazas y oportunidades
b) Factores relacionados al tiempo
c) Las consecuencias sobre los objetivos

Tabla 16. Identificación del riesgo en los archivos correspondientes a los segmentos de navegación, búsqueda y filtrado.

Segmento clave	Archivos propios de los segmentos	Observaciones en base a ISO 31000:2018	Riesgos potenciales
Navegación	Mapa de navegación (Navegador.xml)	• Existen archivos cuyos nombres son ambiguos en cuanto a su funcionalidad • Existe un segmento que forma parte del código del software que no es visible y no puede ser eliminado de forma directa en virtud de que afecta la operación del software • Existen herramientas de software de terceros que están incorporadas dentro del código, no son utilizadas y solo consumen recursos • Existe un segmento que ya no forma parte de la versión preliminar y no se es posible eliminarlo directamente de los archivos	• Genera confusión para localizar y hacer modificaciones consumiendo más tiempo del que debería • Disminuye el rendimiento global del software y consumir tiempo adicional para buscar soluciones • El código tiene más secciones de las que debería generando confusión en los desarrolladores • El software consume más recursos de los necesarios disminuyendo su eficiencia.
	Archivos dentro del paquete FRAGMENTS con extensión .KT	• Los archivos de funcionalidad son difíciles de identificar debido a sus nombres • Existen varios archivos en blanco los cuales no han sido eliminados	• Consume tiempo y genera confusión al equipo desarrollador aumentando el tiempo de las tareas relacionadas a esos archivos • Genera más código del necesario, consume tiempo

Segmento clave	Archivos propios de los segmentos	Observaciones en base a ISO 31000:2018	Riesgos potenciales
		• Para que los archivos sean estables se requiere escribir 2 secciones de código en un orden especifico, pero no existe documentación sobre cómo realizar esta operación	y aumenta el uso de memoria del software haciéndolo más pesado • Deja los segmentos con un orden distinto inservibles
Búsqueda	Archivos con el prefijo DATASource	• De acuerdo al requerimiento del cliente se desarrollaron 2 bases de datos que contienen información que hace más pesado el software y origina la disminución de su rendimiento global.	• Aumenta el tiempo de respuesta del software dando la impresión de mal funcionamiento • El funcionamiento anómalo del software causa un sesgo en las pruebas de aceptación que puede hacer que se descarte la versión.
	Archivos del sistema de búsqueda	• El sistema encargado de mostrar los resultados presenta una sintaxis que no está debidamente estructurada	• Podría causar fallos en el software como cierres inesperados, o congelaciones de pantalla
	Archivos de tipo LAYOUT con el sistema Search+Recycler	• Los archivos encargados de mostrar y presentar la información emplean herramientas de software de terceros inadecuados de funcionalidad • Los archivos no tienen una secuencia lógica de acuerdo a su función	• Causa un funcionamiento anormal del segmento de código • Consume más tiempo al dificultar la localización de los archivos de cada segmento
Filtrado	Archivos con terminación RecyclerAdapter	• El sistema de filtrado emplea métodos descontinuados para su funcionamiento • En todos los sistemas de filtrado de resultados existe una sección de código incompleta que genera cierres inesperados en algunos dispositivos	• El software dejaría de funcionar para versiones posteriores • Que el software no tenga la cobertura y funcionamiento requerido en los dispositivos

Aplicación de la matriz de prioridades para establecer el nivel de riesgo

En concordancia con el apartado 6.4.3 de la norma ISO 31000:2018 análisis del riesgo, se establecen las siguientes matrices haciendo uso de la matriz de prioridades, de probabilidad y de impacto. Las matrices 17 y 18 corresponden a las sub etapas de la etapa de desarrollo preliminar, a sus respectivos diagramas de flujo y a los archivos de código que integran el software desarrollado.

Con la finalidad de facilitar la comprensión de los criterios de riesgo se anexan nuevamente las tablas 6 y 7.

Tabla 6. Matriz de prioridad para la etapa de desarrollo preliminar en base a lo establecido en las tablas 5 y 6

Impacto / Probabilidad	Relevante (1)	Importante (2)	Prioritario (3)
Baja (1)	1	2	3
Media (2)	2	4	6
Alta (3)	3	6	9

Tabla 7. Gestión en base al índice de riesgo.

Puntuación	Índice de riesgo	Tipo de gestión
Mayor o igual a 6	Prioritario	Alta dirección, responsables del desarrollo y gestión de procesos
Entre 2 y 4	Importante	Gestión por parte de los equipos de desarrollo
Igual a 1	Moderado	Gestión de procedimientos y procesos de rutina

Tabla 17. Matriz de riesgos pertenecientes a las sub etapas y diagramas de flujo del software desarrollado

Observación	Riesgo detectado	Probabilidad	Impacto	Nivel del riesgo	Tipo de gestión
Desarrollo principal No existe un proceso de revisión de correcciones que evite incorporar medidas que no arreglen los defectos	Repetir el proceso más veces de las necesarias consumiendo más tiempo del establecido	2	1	Importante	Gestión de los equipos de desarrollo
	Generar soluciones de poco impacto o inadecuadas que no arreglen los defectos detectados	2	2	Importante	Gestión de los equipos de desarrollo
Compilado general En la sub etapa de compilado general no se contempla la planeación para la implementación de acciones para solucionar los errores, únicamente se contempla en análisis y la implementación	Se consume más tiempo debido a que no se tiene establecido un protocolo por lo que cada vez que ocurre todo debe ser pensado desde 0	1	2	Importante	Gestión de los equipos de desarrollo
Pruebas de validación/ aceptación El diagrama solo contempla la realización de pruebas no se considera la planeación de casos de uso para el correcto desempeño en las pruebas	El software puede no cumplir con los requisitos al no planearse correctamente los casos de prueba teniendo que rediseñar el diseño completo	1	3	Importante	Gestión de los equipos de desarrollo

Navegación El árbol de navegación no contempla la movilidad entre pantallas de resultados pues esto se implementó sobre la marcha	El desarrollo de los segmentos clave se ve afectado por la implementación de características no requeridas en otras secciones.	2	2	Importante	Gestión de los equipos de desarrollo
(mapa de navegación) No existe como tal un diagrama de flujo que describa cómo debe funcionar la navegación	El código escrito puede no cumplir con la funcionalidad esperadas	2	3	Prioritatrio	Alta dirección, responsables del desarrollo y gestión de procesos
El árbol de navegación es una representación gráfica del movimiento en el software y no un diagrama de flujo que describa el funcionamiento del módulo como tal	Se consume más tiempo de desarrollo en el segmento de navegación, retrasando el desarrollo de otras secciones	1	2	Importante	Gestión de los equipos de desarrollo

Búsqueda No se indica que la entrada de solicitudes de búsqueda se actualiza en tiempo real en el diagrama	Genera sesgos sobre el funcionamiento y desarrollo del segmento a causa de que el diagrama no es claro	1	2	Importante	Gestión de los equipos de desarrollo
Filtrado el proceso de recepción de resultados corresponde al diagrama de búsqueda, el proceso que hace falta en la recolecta de coincidencias con la solicitud recibida	Causa interrupciones constantes al desarrollo del segmento, reduciendo el tiempo de desarrollo para los demás segmentos	2	1	Importante	Gestión de los equipos de desarrollo

Tabla 18. Matriz de riesgos pertenecientes a los archivos propios del código del software desarrollado

Observación	Riesgo detectado	Probabilidad	Impacto	Nivel del riesgo	Tipo de gestión
Navegación (mapa de navegación) Existen archivos cuyos nombres son ambiguos en cuanto a su funcionalidad	Genera confusión para localizar y hacer modificaciones consumiendo más tiempo del que debería	3	1	Importante	Gestión de los equipos de desarrollo
Navegación (mapa de navegación) Existe un segmento que forma parte del código del software que no es visible y no puede ser eliminado de forma directa en virtud de que afecta la operación del software	Disminuye el rendimiento global del software y consumir tiempo adicional para buscar soluciones	2	2	Importante	Gestión de los equipos de desarrollo
Navegación (mapa de navegación) Existen herramientas de software de terceros que están incorporadas dentro del código, no son utilizadas y solo consumen recursos	El código tiene más secciones de las que debería generando confusión en los desarrolladores	2	2	Importante	Gestión de los equipos de desarrollo
Navegación (mapa de navegación) Existe un bloque de código aislado que ya no forma parte de la versión preliminar y no es posible eliminarlo directamente de los archivos	El segmento de navegación consume más recursos de los necesarios disminuyendo su eficiencia.	1	2	Importante	Gestión de los equipos de desarrollo

Navegación (fragments.kt) Los archivos de funcionalidad son difíciles de identificar debido a sus nombres	Genera confusión al equipo desarrollador aumentando el tiempo de las tareas relacionadas a esos archivos	2	1	Importante	Gestión de los equipos de desarrollo
Navegación (fragments.kt) Existen varios archivos en blanco los cuales no han sido eliminados	Genera más código del necesario, consume tiempo y aumenta el uso de memoria del software haciéndolo más pesado	2	1	Importante	Gestión de los equipos de desarrollo
Navegación (fragments.kt) Para que los archivos sean estables se requiere escribir 2 secciones de código en un orden especifico, pero no existe documentación sobre cómo realizar esta operación	Deja los segmentos con un orden distinto inservibles	3	3	Prioritario	Alta dirección, responsables del desarrollo y gestión de procesos
Búsqueda (bases de datos) De acuerdo al requerimiento del cliente se desarrollaron 2 bases de datos que contienen información que hace más pesado el software y origina la disminución de su rendimiento global.	Aumenta el tiempo de respuesta del software durante la búsqueda	1	1	Moderado	Gestión de procesos y procedimientos de rutina

	El funcionamiento anómalo ocasionado por las bases de datos más pesadas causa un sesgo en las pruebas de aceptación que puede hacer que se descarte la versión.	2	3	Prioritario	Alta dirección, responsables del desarrollo y gestión de procesos
Búsqueda (Sistema de búsqueda) El sistema encargado de mostrar los resultados presenta una sintaxis que no está debidamente estructurada	La sintaxis podría causar fallos en el software como cierres inesperados, o congelaciones de pantalla	2	3	Prioritario	Alta dirección, responsables del desarrollo y gestión de procesos
Búsqueda (Layouts) Los archivos no tienen una secuencia lógica de acuerdo a su función	Causa un funcionamiento anormal del segmento de código	2	2	Importante	Gestión de los equipos de desarrollo
Búsqueda (Layouts) Los archivos encargados de mostrar y presentar la información emplean herramientas de software de terceros inadecuados de funcionalidad	Disminuye la eficiencia del segmento de búsqueda, limita las funcionalidades del segmento y causa que los desarrolladores a buscar alternativas poco adecuadas	1	2	Importante	Gestión de los equipos de desarrollo

Filtrado (Archivos recicladores) El sistema de filtrado emplea métodos descontinuados para su funcionamiento	El filtro del software dejaría de funcionar para versiones posteriores	2	3	Prioritario	Alta dirección, responsables del desarrollo y gestión de procesos
Filtrado (Archivos recicladores) En todos los sistemas de filtrado de resultados existe una sección de código incompleta que genera cierres inesperados en algunos dispositivos	El software no tendría el funcionamiento requerido en los dispositivos y sería rechazado	1	3	Importante	Gestión de los equipos de desarrollo

Valoración del riesgo

Para llevar a cabo la valoración del riesgo es necesario tomar en cuenta los controles existentes o no existentes en la etapa de desarrollo preliminar, de acuerdo con la OCDE los tipos de control son:

1. *De dirección*: son instrumentos de guía para la consecución de resultados
2. *Detectivos*: Su función es identificar si se han producido resultados indeseables después de un evento
3. *Preventivos*: evitan o reducen la posibilidad de materialización de un riesgo
4. *Correctivos*: corrigen los resultados indeseables que se hayan producido

En el caso del software desarrollado los controles existentes son los siguientes:

1. Control de ajustes al código (Correctivo)
2. Análisis de causas (Preventivo)
3. Control de versiones (Preventivo)

La tabla 19 muestra los niveles de prioridad y de control empleados en el desarrollo del software, la tabla 20 muestra los criterios de acción en el desarrollo del software.

Tabla 19. Matriz de vulnerabilidad de los controles en la etapa preliminar del desarrollo del software

Prioridad inicial	3	3	6	9
	2	2	4	6
	1	1	2	3
		1	2	3
		Existentes y eficientes	Existen para casos concretos	No existen
		Valoración del control		

Tabla 20. Acciones para los controles según la puntuación obtenida en la tabla 19.

Puntuación	Acciones para los controles
Igual a 1	Mantener el control existente, se pueden conducir análisis para mejorar los controles cuando sea conveniente
Entre 2 y 4	Implementar acciones correctivas o preventivas en el control existente para ampliar su cobertura.
Mayor a 4	En base a los procesos que se lleven a cabo establecer un control de actividades de forma inmediata, tomando en cuenta los objetivos y alcance de la situación.

En base a la tabla 19 y 20 así como con la matriz de prioridad, se establece la tabla 21 que indica las opciones de tratamiento seleccionadas en base a la norma ISO 31000:2018 y acciones sugeridas para el tratamiento del riesgo en el software desarrollado.

Tabla 21. Opciones de tratamiento en base a la norma ISO 31000:2018 y acciones a emprender en base a la matriz de vulnerabilidad y a la valoración de los controles

Puntuación en base a la matriz de vulnerabilidad	Prioridad en base a la tabla 19	Opción (es) de tratamiento elegida (s)	Acciones para el tratamiento de riesgo tomando en cuenta los controles existentes
Igual a 1	Moderado	Conservar el riesgo	Conservar el riesgo mediante los controles existentes y conociendo su implicación
Entre 2 y 4	Importante	Reducir el riesgo	Formular acciones orientadas a reducir los riesgos existentes, ya sean dirigidas a cambiar los controles existentes o directamente sobre las etapas y segmentos clave del software desarrollado
Mayor o igual a 6	Prioritario	Reducir el riesgo	Implementar acciones orientadas a reducir el riesgo en la actividad/proceso a la brevedad., para proyectos futuros establecer controles teniendo en cuenta estos percances

Acciones para abordar riesgos y oportunidades: Tratamiento del riesgo

De acuerdo con la norma ISO 9001:2015 es necesario que se establezcan de forma ordenada y planificada las acciones para abordar los riesgos y oportunidades, en este caso se emplea la norma ISO 31000:2018 sección 6.5 tratamiento del riesgo, la sección contempla:

1. Formulación y selección de opciones para el tratamiento del riesgo
2. Planeación e implementación del tratamiento
3. Evaluación de las medidas tomadas y del riesgo residual
4. Toma de decisiones

Para el software desarrollado y en conjunto con la sección anterior de identificación del riesgo la tabla 22 establece las opciones y acciones a emprender en base al nivel de riesgo determinado y al nivel de eficiencia de los controles existentes, las opciones de tratamiento del riesgo se apegan estrictamente a las mencionadas en la norma ISO 31000:2018.

Tabla 22. Acciones y opciones de tratamiento del riesgo para el software desarrollado en base al nivel de riesgo determinado.

Nivel de riesgo	Opciones seleccionadas	Acciones a emprender
Prioritario	Reducir el riesgo	Sub etapas y diagramas de flujo: 1. Pruebas de valoración/aceptación: Establecer un método en colaboración con el cliente para determinar los casos para cada prueba y asegurar que las pruebas sean adecuadas Código del software 3. Navegación a. Crear un archivo de referencia de navegación donde se deje en claro el orden de las funciones b. Definir un diagrama de flujo del sistema de navegación que permita entender su funcionamiento 4. Búsqueda a) Bases de datos: Implementar herramientas de software de terceros que permitan usar la nube para almacenar las bases de datos más pesadas. b) Archivos del sistema de búsqueda: Establecer un formato de código estándar para la sintaxis de los sistemas de búsqueda 5. Filtrado Construir un filtro de referencia que se actualice periódicamente para mantener la estructura correcta.
Importante	Reducir el riesgo	Sub etapas y diagramas de flujo 1. Implementar una bitácora de revisión de cambios para dejar el registro del estado actual de las correcciones y cambios realizados. 2. Añadir al diagrama de flujo de la sub etapa de compilado general un proceso de planeación de soluciones que oriente a los desarrolladores en la corrección de los errores del software 3. Definir un diagrama de flujo que permita remover segmentos en desuso dentro del código en cualquiera de los segmentos 4. Reestructurar los diagramas de flujo cambiando los procesos que se encuentran fuera de lugar.

Nivel de riesgo	Opciones seleccionadas	Acciones a emprender
		Archivos de código 1. Establecer una nomenclatura adecuada así como sub carpetas para cada tipo de archivo en base al segmento clave al que pertenecen 2. Generar diagramas de flujo que indiquen como remover herramientas sin usarse. 3. Establecer de manera clara que funciones deberá realizar un segmento y seleccionar las herramientas de software de terceros más adecuadas mediante un análisis comparativo. 4. Asegurar que la estructura del código este completa
Moderado	Aceptar el riesgo	Se conservará y controlará el riesgo en base a una decisión informada sobre los efectos en el desarrollo del software

Preparación e implementación de los planes para el tratamiento del riesgo

De acuerdo con la norma ISO 31000:2018 apartado 6.5.3 se tomaron en cuenta los siguientes puntos para la planeación y la implementación de las acciones en el software desarrollado.

1. El índice del riesgo asociado con la sub etapa o archivo de código;
2. Las acciones propuestas;
3. Resultados y beneficios esperados.

La estructura de la planeación para el software desarrollado toma en cuenta los siguientes puntos:

1. Sub etapas o archivos de código
2. Clasificación del riesgo
3. Acciones a implementar
4. Pasos a seguir para la implementación de las acciones
5. Resultados
6. Beneficios esperados

La tabla 23 muestra la planeación para la implementación de los planes contemplando las acciones propuestas así como pasos a seguir y los beneficios que se esperan.

Tabla 23. Planeación para la implementación de las acciones al software desarrollado

Sub etapa o archivo de código	Clasificación	Acciones a implementar	Pasos a seguir	Resultado	Beneficios esperados
Pruebas de valoración/aceptación	Prioritario	Establecer un método en colaboración con el cliente para determinar los casos para cada prueba y asegurar que las pruebas sean adecuadas	1. Solicitar la participación del cliente en las pruebas 2. Acordar los aspectos de cada segmento a ser evaluados 3. Acordar los casos y condiciones para el software 4. Asegurarse de que se comprenden las condiciones, casos y aspectos a evaluar 5. En caso de encontrar alguna inconsistencia en las condiciones regresar al paso 2 6. Llevar a cabo las pruebas de aceptación / validación respecto a lo acordado	Método para establecer los casos de las pruebas de aceptación (diagramas, documentos, procedimientos)	Reducir la posibilidad de rechazo de versiones preliminares por fallos de en las pruebas de aceptación
Navegación	Prioritario	Crear un archivo de referencia de navegación donde se deje en claro el orden de las funciones	1. Definir un nombre claro para el archivo de referencia 2. Crear el archivo dentro del paquete NavigationMaestro 3. Establecer las pantallas a las que se le implementara la navegación 4. Construir el código del sistema de navegación y añadir comentarios que indiquen el orden de las funciones.	Archivo de referencia para el segmento de navegación	Reducir el tiempo de construcción del segmento de navegación en un 25% por lo menos

Tabla 23. Planeación para la implementación de las acciones al software desarrollado

Sub etapa o archivo de código	Clasificación	Acciones a implementar	Pasos a seguir	Resultado	Beneficios esperados
Búsqueda (Bases de datos)	Prioritario	Implementar herramientas de software de terceros que permitan usar la nube para almacenar las bases de datos más pesadas.	1. Analizar las diversas opciones existentes para almacenamiento en bases de datos. 2. Seleccionar la mejor opción en base a los datos a almacenar. 3. Buscar las dependencias o librerías pertinentes 4. Incorporarlas al código para poder emplear la nube	Definición de la herramienta de software a emplear	Previene el funcionamiento anómalo del software Elimina el tiempo de espera cuando se realiza una consulta
Búsqueda (Archivos del sistema de búsqueda)	Prioritario	Establecer un formato de código estándar para la sintaxis de los sistemas de búsqueda	1. Crear un paquete con nombre de fácil identificación 2. Crear un archivo de nombre AdaptadorMaestro(archivo maestro) 3. Codificar el archivo rescatando la estructura de los archivos actuales 4. Comentar el archivo (el software no interpretará ese texto) 5. Cuando sea necesario codificar un sistema de búsqueda, renombrar y hacer uso del archivo maestro 6. Cambiar los nombres de las variables dentro de la referencia para conectar el archivo de referencia al sistema	Definición de la sintaxis de los sistemas de búsqueda	Previene el uso de tiempo extra empleado en la construcción del código Evitar que se presenten funcionamientos anómalos

Tabla 23. Planeación para la implementación de las acciones al software desarrollado

Sub etapa o archivo de código	Clasificación	Acciones a implementar	Pasos a seguir	Resultado	Beneficios esperados
Filtrado	Prioritario	Construir un filtro de referencia que se actualice periódicamente para mantener la estructura funcional.	1. Crear dentro del paquete de búsqueda un archivo de nombre FiltroMaestro 2. Escribir el segmento de filtrado copiando el sistema actual 3. Sustituir los métodos que estén descontinuados por métodos actuales 4. "Comentar" el segmento de filtro de referencia 5. Cuando se requiera implementar un segmento de filtrado, copiar y pegar la referencia en el segmento. 6. Cambiar las variables de la referencia por las variables propias del segmento 7. Cada 6 meses revisar el filtro de referencia por si se han actualizado o descontinuado métodos	Asegurar el mantenimiento de la estructura correcta del software	Previene que el filtro deje de funcionar en versiones posteriores
Desarrollo principal	Prioritario	Implementar una bitácora de revisión de correcciones para dejar evidencia de las acciones realizadas	1. Seleccionar un archivo principal donde reportar las correcciones, su estado y fecha 2. En la parte superior después de los archivos importados, usar comentarios y el formato	Bitácora de cambios y estado de las soluciones propuestas	Controla los cambios realizados al código, proporcionando trazabilidad

Tabla 23. Planeación para la implementación de las acciones al software desarrollado

Sub etapa o archivo de código	Clasificación	Acciones a implementar	Pasos a seguir	Resultado	Beneficios esperados
			"*TODO*" para mostrar el estado de los cambios 3. EL registro en la bitácora debe tener: nombre, cargo, día y turno de modificaciones, estado de los cambios y de acuerdo al control de versiones ISO 9000:2015		Reduce los retrasos e la entrega de versiones por soluciones poco efectivas y difíciles de rastrear
Navegación	Importante	Definir un diagrama de flujo del sistema de navegación que permita entender su funcionamiento	1. Plasmar el funcionamiento empleando el software como referencia 2. Verificar las distintas funcionalidades con el código del sistema de navegación 3. Ensamblar el diagrama de acuerdo a lo analizado anteriormente	Diagrama de flujo sobre el sistema de navegación	Reduce al mínimo el tiempo que toma desarrollar segmentos de navegación
Compilado general	Importante	Definir un diagrama de flujo para la sub etapa de compilado general que oriente a los desarrolladores en la planeación de las soluciones para los errores del software	1. Identificar el error y el alcance del mismo 2. Identificar el segmento o segmentos que originan o que se piensa que originan el error 3. Identificar la alarmas del entorno (si las hay) 4. Comprobar sintaxis 5. Comprobar disponibilidad de herramientas de software de terceros o de librerías	Diagrama de flujo para la planeación de soluciones	Reduce el uso de tiempo extra en la planeación de soluciones de la sub etapa de compilado general

Tabla 23. Planeación para la implementación de las acciones al software desarrollado

Sub etapa o archivo de código	Clasificación	Acciones a implementar	Pasos a seguir	Resultado	Beneficios esperados
			6. Comprobar errores de escritura en el código		
Búsqueda	Importante	Añadir un mensaje emergente cada vez que se ingrese o modifique la solicitud de búsqueda para que sea consistente con el diagrama.	1. Ir al segmento de búsqueda 2. Integrar el código para que aparezca el mensaje Definir qué dirá el mensaje	Integración del código de mensaje emergente al sistema de búsqueda actual	Eliminación del sesgo sobre el funcionamiento del sistema de búsqueda
Archivos de navegación, búsqueda y filtrado	importante	Definir un diagrama de flujo que permita identificar segmentos en desuso dentro del código en cualquiera de los segmentos para ser eliminados	1. Identificar el segmento que se cree está en desuso 2. Emplear las herramientas del entorno para descubrir si hay conexiones restantes 3. En caso de no haber uso ninguno: 3.1 Emplear la herramienta de eliminación segura 3.2 En caso de que si, examinar las conexiones y determinar si efectivamente se trata de un segmento en desuso	Diagrama de flujo para la eliminación segura de archivos en desuso	Reduce al mínimo el tiempo que toma deshacerse de segmentos en desuso dentro del código Previene la disminución del rendimiento del software
Filtrado	Importante	Reestructurar los diagramas de flujo cambiando los procesos que se encuentran fuera de lugar	1. Examinar los diagramas que se cree contienen errores de diseño 2. Comparar el diagrama con el código del software para determinar si existen errores	Diagramas de flujo correctos respecto a los procesos que llevan a cabo	Reduce al mínimo el tiempo de codificación y de edición de los segmentos de filtrado

Tabla 23. Planeación para la implementación de las acciones al software desarrollado

Sub etapa o archivo de código	Clasificación	Acciones a implementar	Pasos a seguir	Resultado	Beneficios esperados
			3. En caso de que existan errores, corregir el diagrama para que coincida con el segmento desarrollado 3.		
Archivos de código de navegación, búsqueda y filtrado	importante	Establecer una nomenclatura adecuada así como sub carpetas para cada tipo de archivo en base al segmento clave al que pertenecen	1. Agrupar y nombrar provisionalmente los archivos de los segmentos mediante la herramienta REFACTOR 2. Crear cuando aplique, paquetes para clasificar los tipos de archivos 3. Elegir una nomenclatura clara para cada tipo de archivo 4. Aplicar cuando aplique los cambios con la herramienta REFACTOR	Archivos debidamente identificados y ordenados	Reduce al mínimo el tiempo de identificación de archivos de distintos segmentos
Búsqueda	Importante	Generar diagramas de flujo que indiquen como remover herramientas sin usarse.	1. Determinar que segmentos tienen herramientas sin usar 2. Identificar el color del texto 2.1 Si es gris se procede a la eliminación de ese texto 2.2 En caso contrario la herramienta se encuentra	Diagramas de flujo para la eliminación de herramientas de software de terceros	Evita que se tengan herramientas sin uso Previene el uso de recursos por parte de herramientas en desuso

Tabla 23. Planeación para la implementación de las acciones al software desarrollado

Sub etapa o archivo de código	Clasificación	Acciones a implementar	Pasos a seguir	Resultado	Beneficios esperados
			en uso y no se recomienda eliminarlo		
Filtrado	Importante	Establecer de manera clara que funciones deberá realizar un segmento y seleccionar las herramientas de software de terceros más adecuadas mediante un análisis comparativo.	1. Establecer el alcance del segmento 2. Listar todas las funciones que desempeñara el segmento 3. Listar las herramientas existentes compatibles con dichas funciones 4. Seleccionar un máximo de 3 herramientas compatibles con las funciones del segmento y comparar sus ventajas y desventajas	Definición de las herramientas de software de terceros más adecuadas para su función	Previene el uso de herramientas de software de terceros inadecuadas.
Filtrado	Importante	Asegurar que la estructura del código este completa	1. Listar los archivos que presentan una sección de filtrado 2. Inspeccionar la toma de decisiones 3. Cerrar la estructura del código Indicar la modificación de acuerdo a la norma ISO 9001:2015	Filtro con estructura completa	Previene el mal funcionamiento en los dispositivos

Tabla 23. Planeación para la implementación de las acciones al software desarrollado

Sub etapa o archivo de código	Clasificación	Acciones a implementar	Pasos a seguir	Resultado	Beneficios esperados
Búsqueda (Bases de datos	Moderado	Mantener el tamaño de archivo de las bases de datos debajo de cierto peso, cuando estas se acerquen al límite del tamaño de archivo subir a la nube	1. Establecer un tamaño de archivo el cual mantenga el tiempo de respuesta actual o menor 2. Definir previamente el tipo de archivos o texto que contendrán las bases de datos 3. En caso de emplear solo texto sintetizar la información lo más posible evitando la saturación por texto. 4. En caso de emplear imágenes convertirlas a SVG reduciendo al mínimo su peso, de no ser posible emplear PNG. Cuando el tamaño del archivo se acerque al límite emplear herramientas de la nube para subir la base de datos y evitar el aumento del tiempo de respuesta	Bases de datos funcionales	Mantener el tiempo de respuesta en el nivel actual.

Resultados

Pensamiento basado en riesgos aplicado a las sub etapas del desarrollo preliminar del software

Desarrollo principal

Para controlar la implementación de cambios y soluciones en la sub etapa de desarrollo principal se implementó una bitácora interna dentro del código del software (*Bitácora de cambios*) la cual siempre es el primer archivo en mostrarse a los desarrolladores, de esta forma proporcionando una entrada de los cambios realizados, fecha, turno, personal que realizo el cambio y estado de los cambios, gracias al formato "*TODO*" es posible resaltar estos datos y evitar la confusión de la bitácora con código, con este archivo además de dejarse registro de modificaciones se muestra cual es el estado del cambio en el software, en el caso de que un cambio no tenga el impacto esperado este es fácilmente identificable y rastreable. De esta manera se espera eliminar el tiempo extra empleado al no contar con algún tipo de control de cambios eliminando el riesgo de esta sub etapa.

Compilado general

En el caso de la sub etapa de compilado general se añadió el proceso de planeación de soluciones al diagrama de flujo con la finalidad de proporcionar guía para plantear soluciones adecuadas con la información obtenida del proceso anterior "Análisis de causas" y finalmente implementar dichas soluciones, con esto se espera eliminar el consumo de tiempo adicional al no tener un protocolo o guía de cómo proceder en la planeación de soluciones, por ende se evitaría la implementación de soluciones de bajo impacto, esto se observa en el diagrama 8.

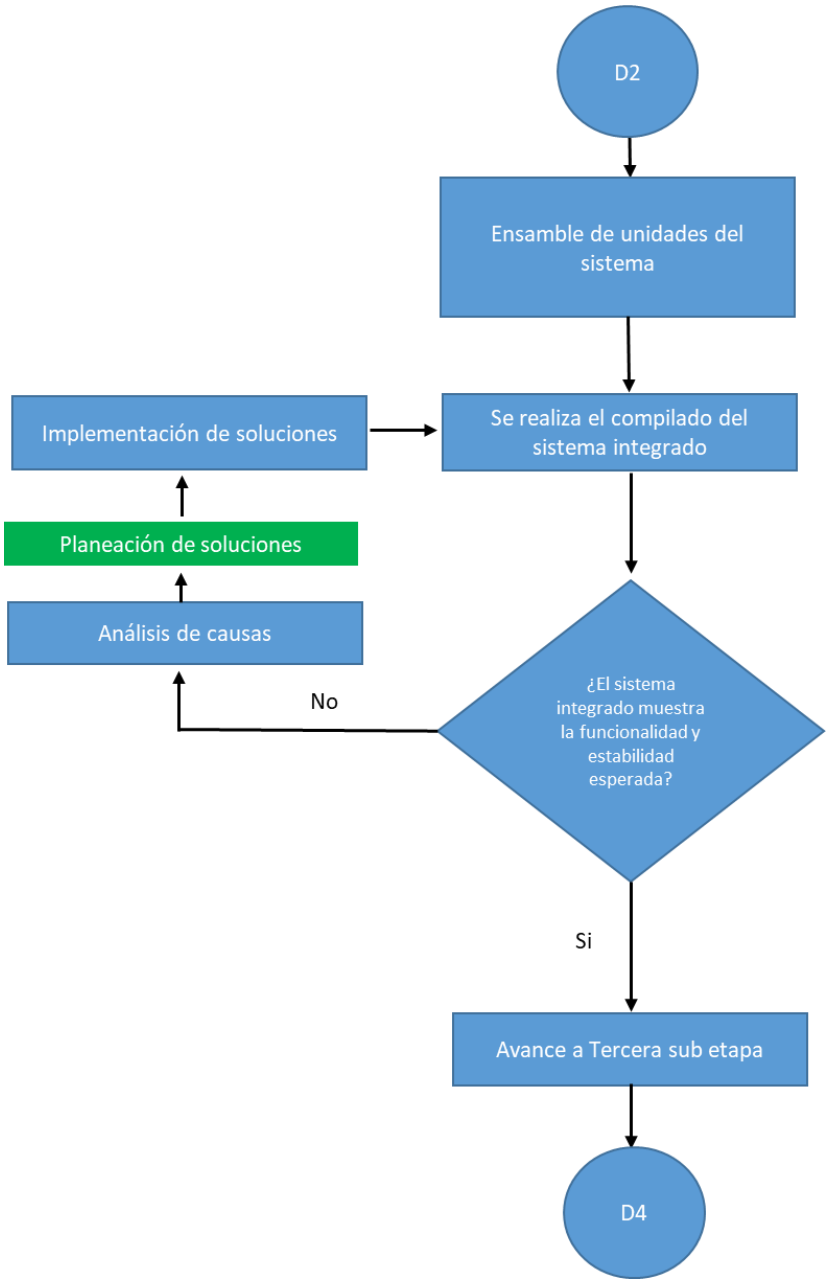
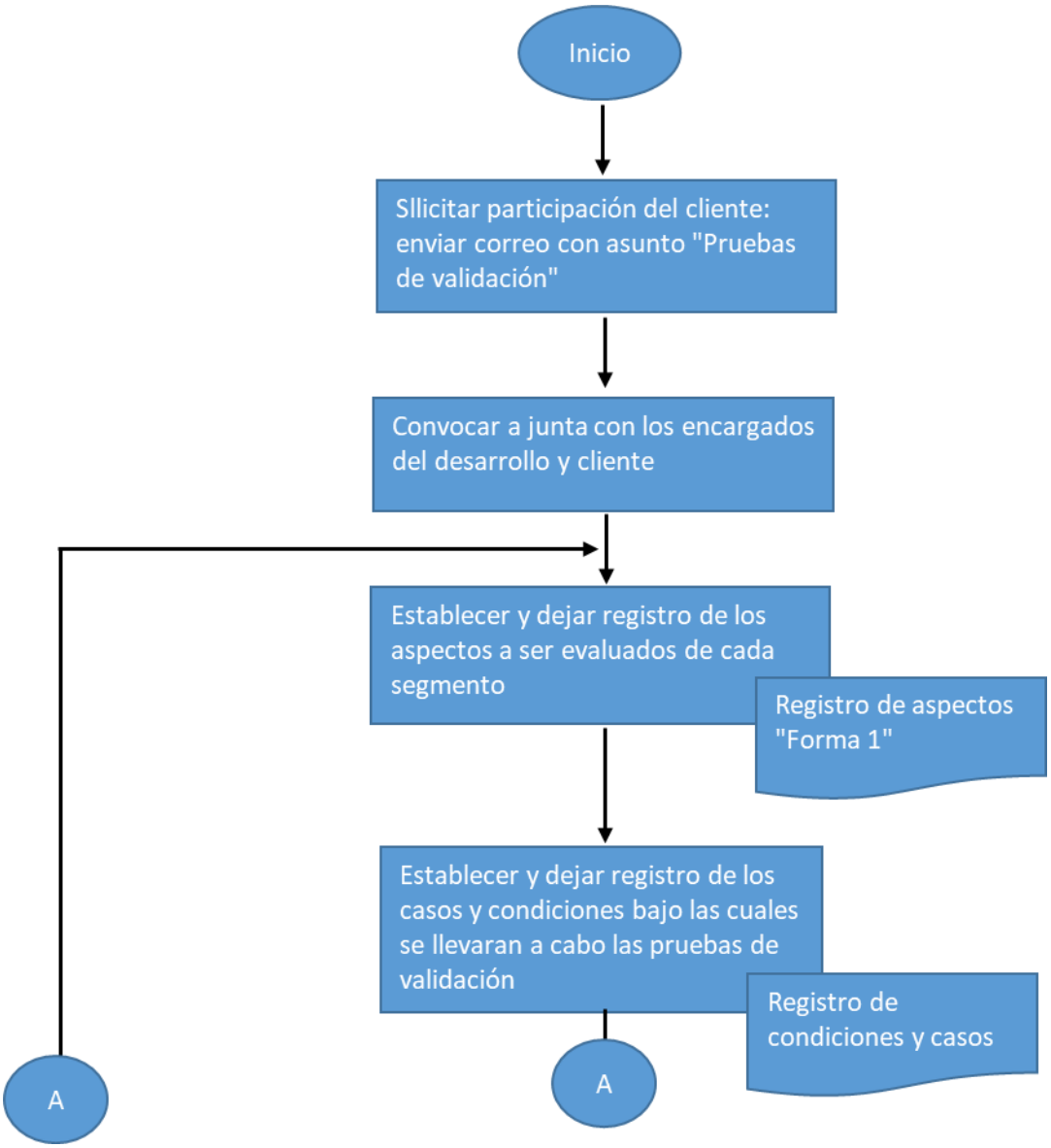


Diagrama 8. Diagrama de compilado general ajustado

Pruebas de aceptación / validación

Para eliminar el riesgo de rechazo de versiones del software a causa de fallo en las pruebas de aceptación / validación, se estableció un método que comprende la participación del cliente en conjunto con los desarrolladores, esto se plasmó en el diagrama de flujo 9 el cual comprende una reunión con el cliente, establecimiento de aspectos a evaluar, casos y condiciones de aprobación dejando registro de estos aspectos en los formatos I y II creados para este fin. Con este método se espera eliminar el rechazo de versiones.


Inicio
Sllicitar participación del cliente: enviar correo con asunto "Pruebas de validación"
Convocar a junta con los encargados del desarrollo y cliente
Establecer y dejar registro de los aspectos a ser evaluados de cada segmento
Registro de aspectos "Forma 1"
Establecer y dejar registro de los casos y condiciones bajo las cuales se llevaran a cabo las pruebas de validación
Registro de condiciones y casos
A
A

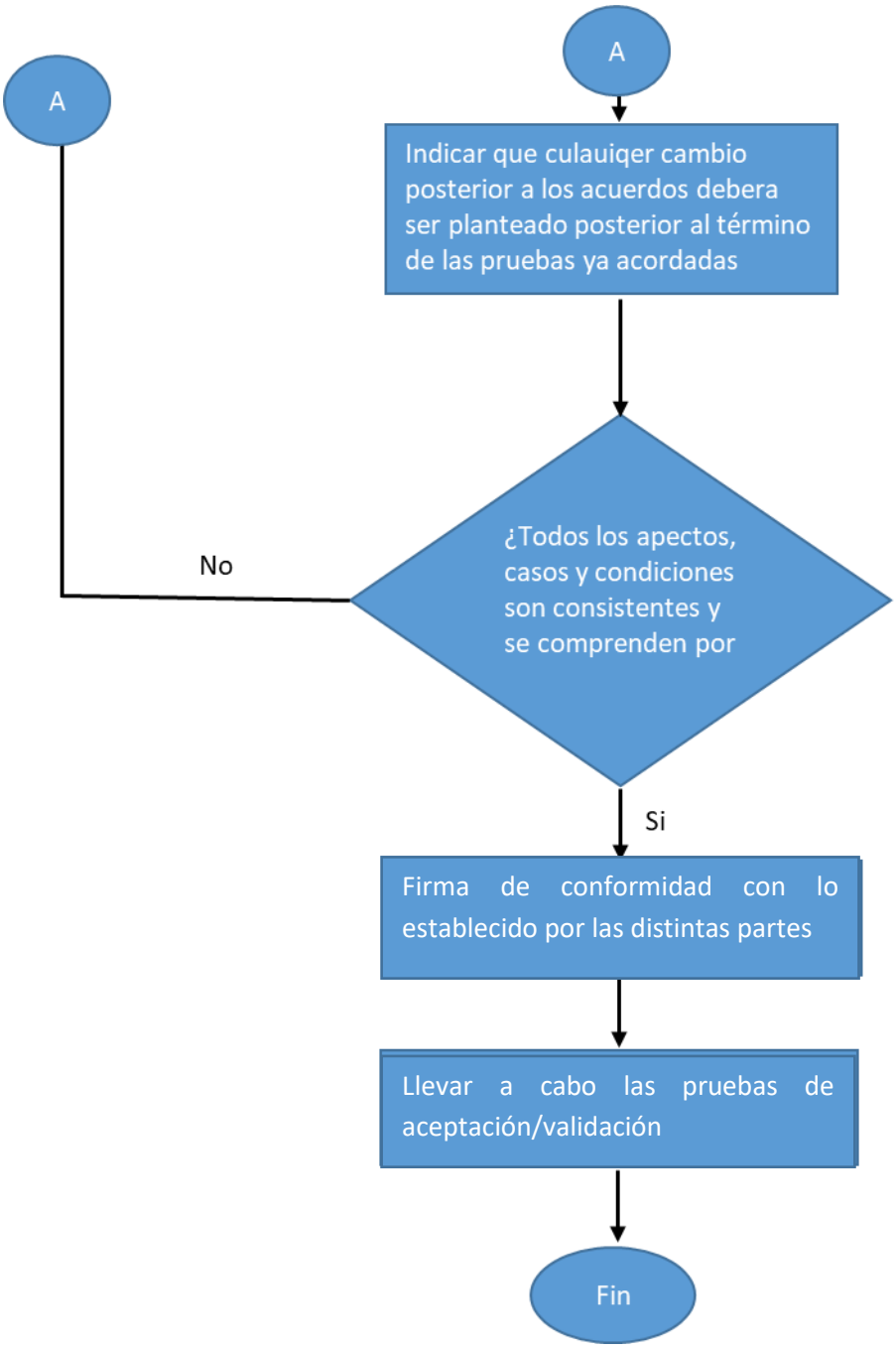


Diagrama 9. Método propuesto para prevenir rechazo de versiones

A continuación se presentan los formatos I y II que acompañan al diagrama

	Formato 1: Aspectos de evaluación por segmento	
	Página:	1 / 1

Instrucciones:

- Para cada segmento clave que compone el software indicar que aspectos se someterán a pruebas de aceptación/ validación, pueden ser por módulos (cuando aplique) o el segmento completo.
- Indicar sitio y fecha del acuerdo en su lugar correspondiente
- Cuando se complete la tabla de aspectos junto con la de casos y condiciones, recabar la firma de conformidad del cliente y guardar ambos registros como evidencia de conformidad

Tabla 1. Aspectos a evaluar de cada segmento clave del software v1.3.1.0

Segmento	Aspecto a evaluar en pruebas de aceptación
Navegación	
Búsqueda	
Filtrado	

Sitio: _____

Fecha del acuerdo: _____

_____

Firma de conformidad del cliente

Fecha de edición	
Elaborado por	

	Formato II: Definición de casos y condiciones para los segmentos de navegación, búsqueda y filtrado	
	Página:	1 / 2

Instrucciones:

- Para cada segmento clave, llenar la tabla de casos y condiciones para las pruebas de validación, en caso de necesitar más espacio para los casos y condiciones usar el reverso de estas hojas.
- Indicar la fecha y el sitio del acuerdo en su apartado correspondiente
- Recabar la firma de conformidad del cliente
- Adjuntar la forma II a la forma I y guardar como evidencia de conformidad.

Tabla 1. Casos y condiciones para el segmento de navegación

segmento a evaluar	Navegación
Aspecto/s a evaluar (mismos que en la forma I)	
Caso	Condiciones bajo las cuales se llevara a cabo la prueba

Tabla 2. Casos y condiciones para el segmento de búsqueda

segmento a evaluar	Búsqueda
Aspecto/s a evaluar (mismos que en la forma I)	
Caso	Condiciones bajo las cuales se llevara a cabo la prueba

Fecha de edición	
Elaborado por	

Tabla 3. Casos y condiciones para el segmento de filtrado

segmento a evaluar	Filtrado
Aspecto/s a evaluar (mismos que en la forma I)	
Caso	Condiciones bajo las cuales se llevara a cabo la prueba

Sitio: _____

Fecha del acuerdo: _____

_____

Firma de conformidad del cliente

Pensamiento basado en riesgos aplicado a los segmentos de navegación, búsqueda y filtrado.

Navegación

Archivos de referencia para la construcción de un sistema de navegación

Los archivos que componen el sistema de referencia son de 3 tipos, fragmento, árbol de navegación y la pantalla de inicio, para cada uno de estos se tiene un archivo maestro:

4. *FragmentoMaestro.kt*

Creado en base al requisito 7.5 de la norma ISO 9001:2015 este archivo cuenta con todas las instrucciones necesarias para acceder a cada función que el software posee. Este archivo permite reducir al mínimo el tiempo empleado para desarrollar funciones y pantallas para el software

5. *Navegador_maestro.xml*

Creado para conectar los fragmentos a una pantalla principal o interfaz principal, el árbol de navegación maestro permite sustituir y añadir los fragmentos actuales y futuros a una misma pantalla de inicio o interfaz principal reduciendo el tiempo destinado a este apartado al mínimo, esto se observa en la figura 11.

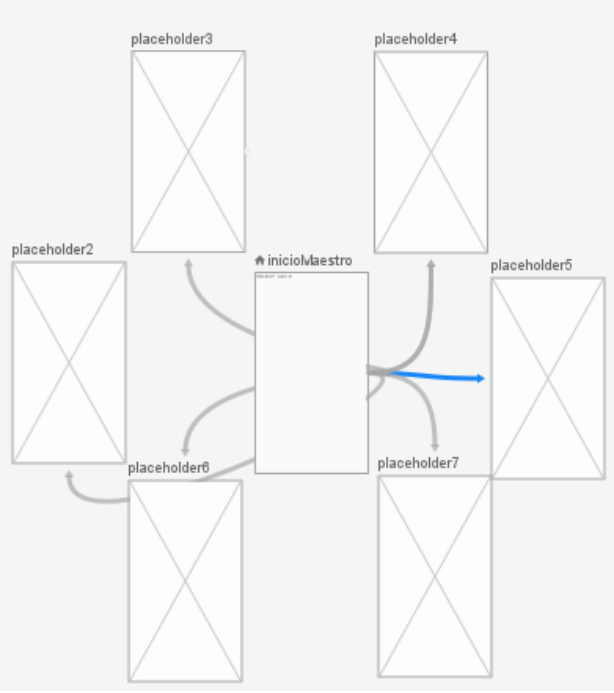


Figura 11. Esquema del navegador maestro

6. *InicioMaestro.kt*

Construido cómo un esqueleto para una pantalla o interfaz principal, permite conectar al árbol de navegación a un solo lugar desde dónde se puede acceder a todo el software, esto permite reducir al mínimo las tareas relacionadas a crear accesos para cada función del software una a una mediante un formato estándar.

Diagrama de flujo del funcionamiento del sistema de navegación (no árbol de navegación)

Para reducir la confusión y el tiempo asignado al desarrollo del segmento de navegación se construyó un diagrama de flujo que describe el funcionamiento del sistema completo para proporcionar una mejor comprensión del mismo, esto se observa en el diagrama 12.

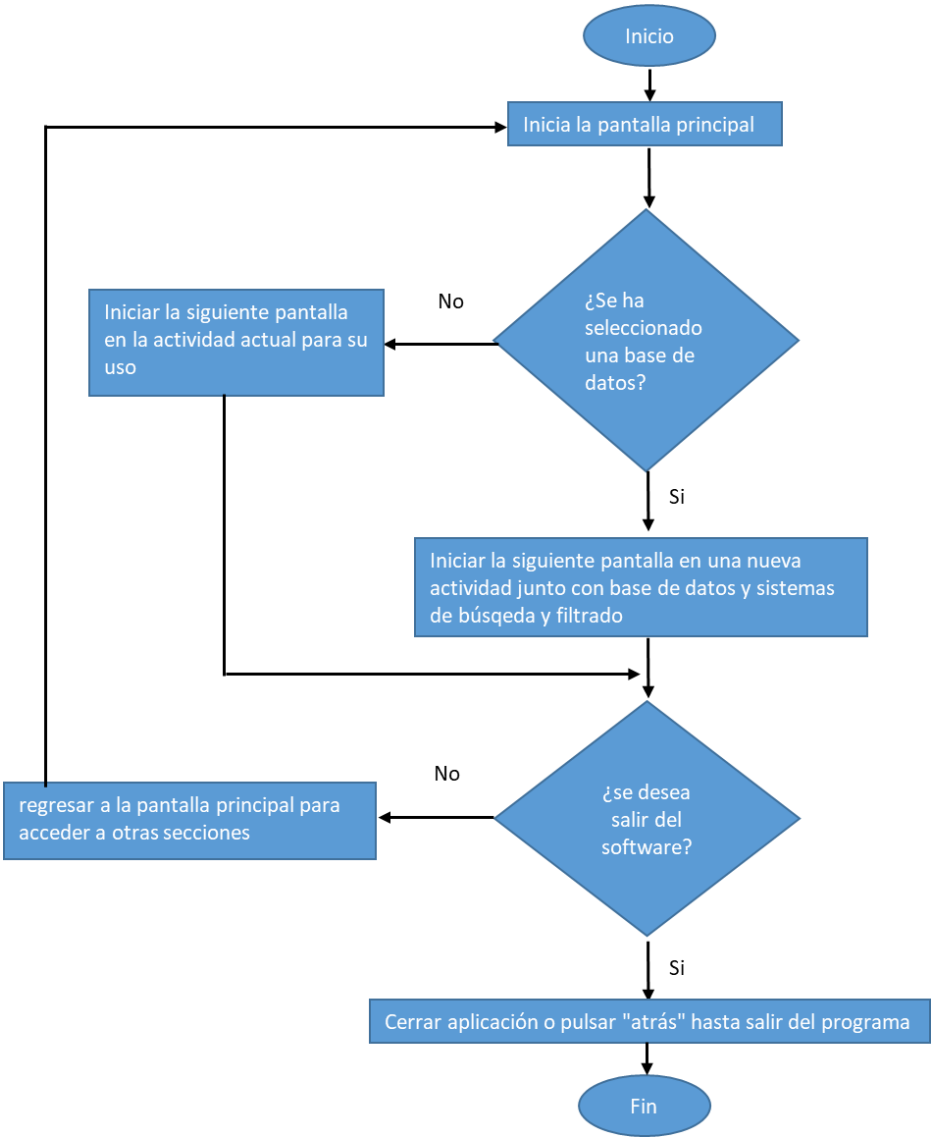


Diagrama 12. Funcionamiento del sistema de navegación

Búsqueda

Eliminación de sesgo sobre el funcionamiento del sistema de búsqueda

Para eliminar posibles sesgos sobre el funcionamiento del sistema de búsqueda se integró código que muestra un mensaje emergente indicando que el sistema se actualiza en tiempo real de acuerdo a la petición, con esto se espera eliminar cualquier posible sesgo sobre el segmento de búsqueda

Definición de la herramienta para bases de datos

Debido a que existen diferentes alternativas para administrar bases de datos, se realizó un análisis de estas herramientas para determinar cuál de ellas se adapta mejor al software con la finalidad de evitar funcionamiento anómalo de las bases a causa de emplear una herramienta poco adecuada y reducir el tiempo de respuesta de los segmentos de búsqueda. La tabla 24 muestra las herramientas tomadas en cuenta para este caso.

Tabla 24. Herramientas disponibles para migrar bases de datos a la nube

Base de datos	Ventajas	Desventajas
ROOM data Base	• Forma parte del ecosistema Android Jetpack el cual es empleado en el software • Es una librería / dependencia que se puede implementar en cualquier etapa del proyecto, cuando así sea decidido • Fácil de implementar y configurar • Permite alojar datos de forma local si así se requiere pero con la eficiencia de un sistema estructurado • Integra herramientas para automatizar la creación de bases de datos dentro del software	• Requiere conocimiento específico de SQLite y SQL • Actualizaciones manuales constantes • Exclusiva para los sistemas operativos Android • Las bases de datos pueden tornarse complejas de manejar
FIREBASE Data Base	• Es una librería / dependencia integrable al código • Multiplataforma, es posible emplearla en sistemas más allá de Android / google como: Apple, Microsoft, Linux etc. • Permite mantener los datos almacenado en la caché del dispositivo • No requiere conocimiento especializado en bases de datos • Contiene herramientas que ayudan al desarrollador a implementar bases de datos tipo FIREBASE	• El proyecto debe estar construido alrededor de la infraestructura FIREBASE • Los datos a almacenar deben seguir la estructura JSON forzosamente • Costosa para almacenar grandes volúmenes de datos • Seguridad compleja de codificar • Presenta limitaciones en cuanto a formatos de búsqueda, teniendo que adaptar el proyecto a ese formato.

Por el tipo de software desarrollado se ha elegido implementar ROOM DATABSE al sistema actual, con esto se espera eliminar tiempos excesivos de espera para realizar una consulta así como prevenir funcionamientos anómalos causados por no emplear una herramienta adecuada.

Construcción de un formato estándar para el código del sistema de búsqueda

Para reducir al mínimo el tiempo que se destina a incorporar un segmento de búsqueda al software y eliminar cualquier funcionamiento anómalo se construyó un archivo maestro (AdaptadorMaestro) en base al requisito 7.5 de la norma ISO 9001: 2015, el cual contiene todo el código necesario para incorporar la búsqueda de forma fácil y eficiente.

Eliminación de herramientas de software sin usar

Para prevenir que los archivos XML relacionados al sistema de búsqueda consuman más recursos de los necesarios se recomienda eliminar los textos de color gris en virtud de que son herramientas sin utilizar y solo consumen recursos que hacen más lentos los procesos.

Optimización de bases de datos actuales

Con el fin de tener bases de datos funcionales se crearon archivos de referencia del tipo clase de datos (“DataClassMaestro”) y base de datos (“DataSourceMaestro”). Se adaptaron las bases de datos existentes de acuerdo a los archivos de referencia creados, esto permitió su optimización en cuanto a tamaño y tipo de contenido, lo que permite su fácil manejo y modificación.

Filtrado

Ajuste de los diagramas de los segmentos de búsqueda y filtrado

Debido a que los diagramas de flujo correspondientes a los segmentos de búsqueda y filtrado presentaban errores en cuanto a las tareas que llevaban a cabo, se realizó una reasignación de sus funciones de acuerdo al código del software v1.3.1.0 de forma que lo que desempeña el software coincida con lo que hay en los diagramas, esto elimina el consumo de tiempo extra a causa de la confusión que generaban los diagramas anteriores, los ajustes se observan en los diagramas 13 y 14.

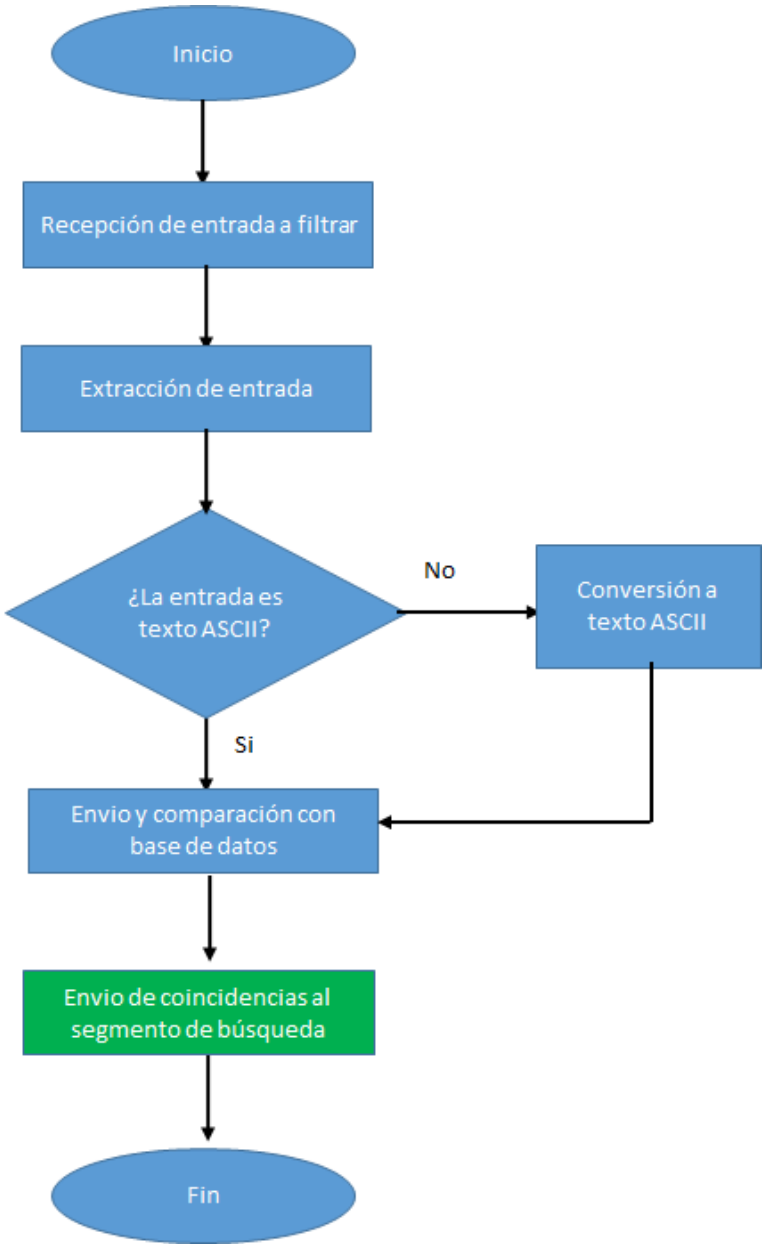


Figura 13. Diagrama de filtrado ajustado

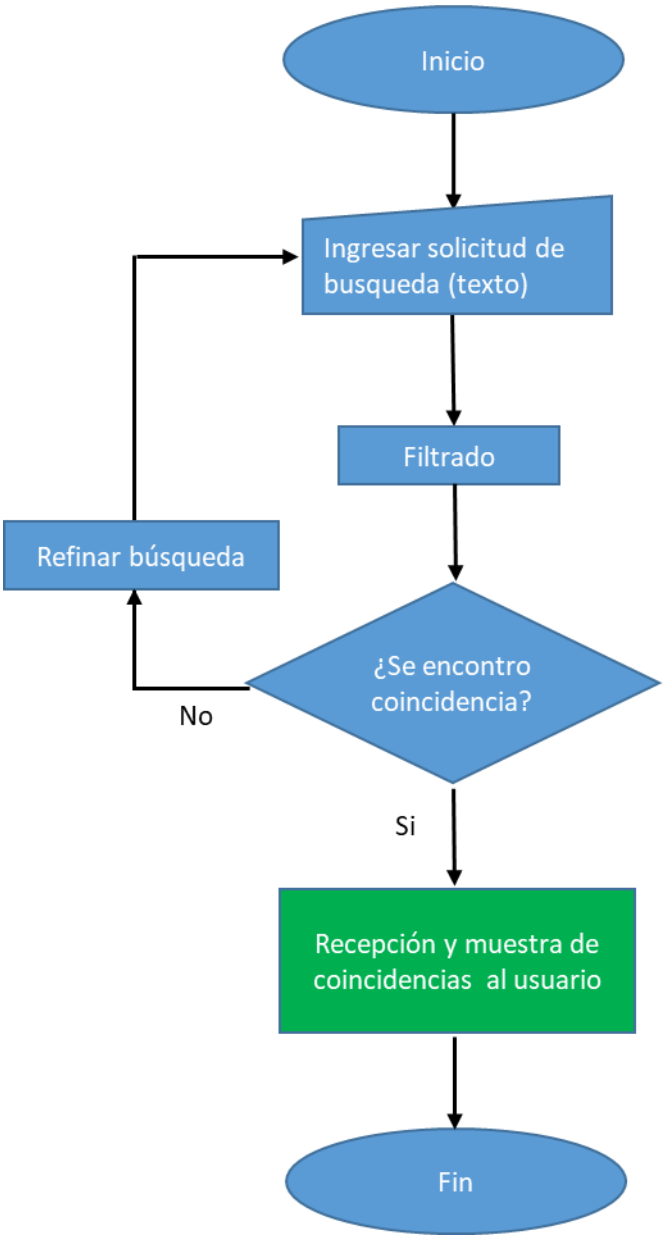


Figura 14. Diagrama de búsqueda ajustado

Selección de las herramientas de software para la construcción de filtros

Para prevenir el uso de herramientas de software poco adecuadas para la construcción de los filtros se llevó a cabo un análisis de las herramientas existentes, tomando en cuenta su costo, disponibilidad en plataformas, ventajas y desventajas para finalmente elegir la herramienta que fue empleada para la construcción de filtros maestros, esto se muestra en las tablas 25 y 26.

Tabla 25. Herramientas disponibles para la construcción de segmentos de filtrado

Segmento	Funciones	Herramientas compatibles
Filtrado	Recibe el texto del segmento de búsqueda, realiza comparaciones y envía los resultados al segmento de búsqueda para ser mostrados al usuario	• FIterable Android Jetpack • FIlter Library (H7) • FIlter Jetpack Compose

Tabla 26. Análisis de las herramientas disponibles en base a ventajas y desventajas

Herramienta	Ventajas	Desventajas
FIterable Android Jetpack	1. Se puede generar un archivo de referencia o clase propia. 2. Filtro compacto y claro de integrar 3. De sintaxis estándar por lo que su diseño es fácil 4. Soportado en toda la infraestructura de Google	• Requiere conocimiento sólido sobre el funcionamiento de filtrado de datos • Requiere conocimientos sólidos sobre "objetos acompañantes" para integrarse correctamente
FIlter Library (H7)	1. De fácil implementación en el proyecto actual 2. Reduce el tiempo de desarrollo manteniendo la estructura de código actual	• Sujeto a compra de licencias bajo condiciones de uso • Se desconoce si el desarrollador tiene permitido emplear sus dependencias en la infraestructura de Google
FIlter Jetpack Compose	1. Reduce el tiempo de desarrollo 2. Sintaxis menos complicada 3. Reduce el tiempo de compilación del proyecto 4. Elimina la necesidad de archivos XML	• Es vital tomar una capacitación en tecnologías COMPOSAABLE • Deja inútiles a los archivos XML actuales teniendo que migrar todo el contenido a esta nueva tecnología • El proyecto debe estar pensado para emplear esta nueva tecnología, no es posible integrarla una vez empleada otra herramienta.

Definición de la estructura del filtro

1. **Construcción de filtro de referencia**

Con la finalidad de reducir al mínimo el tiempo de construcción de segmentos de filtrado se construyó un filtro maestro, de esta forma y empleando la herramienta definida en el punto anterior los filtros son fáciles de implementar y actualizar

2. **Revisión de la estructura del código del filtro existente**

Para eliminar cualquier funcionamiento anómalo en el software se condujo una revisión de todos los archivos que tenían incorporado un sistema de filtrado, posteriormente se identificaron las partes incompletas y se ajustaron.

Archivos del software

Nomenclatura y agrupación

Con orden de reducir la confusión y el tiempo para encontrar archivos se agruparon todos los archivos de acuerdo a la función que desempeñan facilitando su localización, además se definió una nomenclatura estricta para cada tipo de archivo según su función y carpeta de agrupación esto puede verse en la tabla 27 y figura 15.

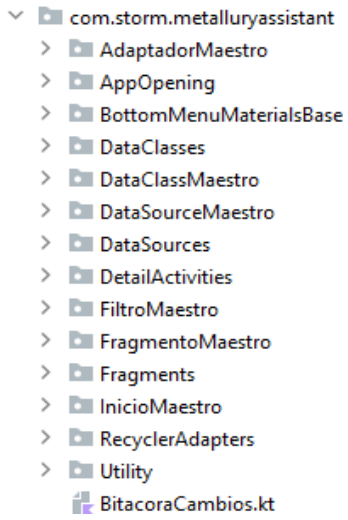


Figura 15. Agrupación por tipo de archivo

Tabla 27. Agrupación y nomenclatura de los archivos

Segmento	Tipo de archivo	Nomenclatura	Paquete
Navegación	Fragmentos	Iniciar con la función desempeñada y añadir la terminación “Fragment”	Fragmentos
Búsqueda	Adaptadores	Colocar una palabra relacionada bloque desarrollado y añadir la terminación “RecyclerAdapter”	RecyclerAdapters
	Base de datos	Iniciar el nombre de la clase de datos y añadir la terminación “DataSource”	DataSources
	Clase de datos	Iniciar con el prefijo “DataClass” y terminar con una palabra relacionada al bloque correspondiente	DataClasses
Filtrado	Filtros	El nombre del filtro coincide con el del adaptador al que se incorpora	AdapterRecycler

Eliminación de archivos en desuso del software desarrollado

Debido a que en la construcción del software se crearon archivos que dejaron de ser utilizados y con el fin de que los desarrolladores pudieran detectarlos se usó la herramienta “FIND USAGES” para estar en posibilidad de eliminarlos de forma segura y evitar la confusión de los desarrolladores y el consumo de recursos del software desarrollado, el procedimiento se observa en el diagrama 16.

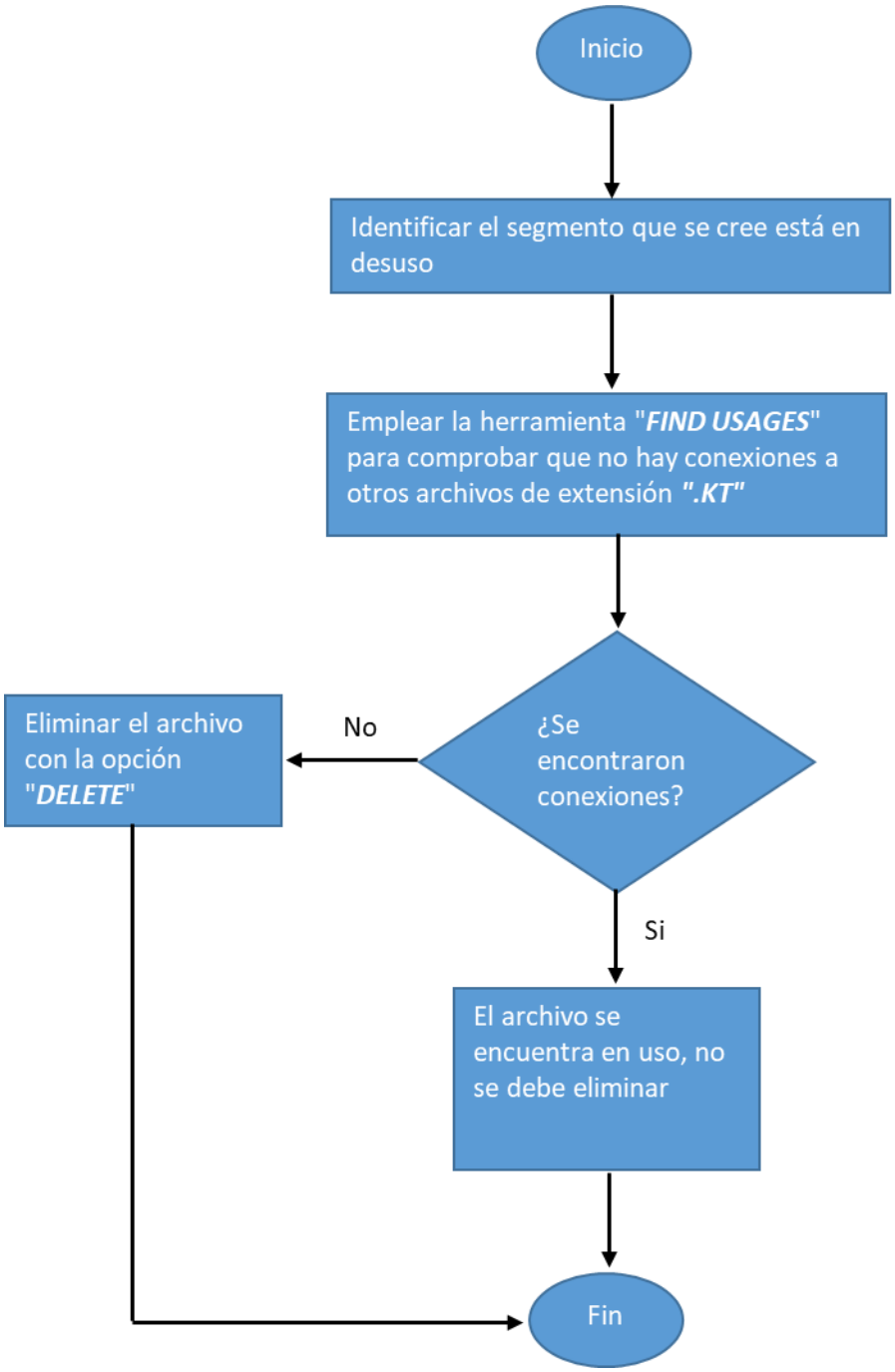


Diagrama 16. Eliminación de archivos en desuso

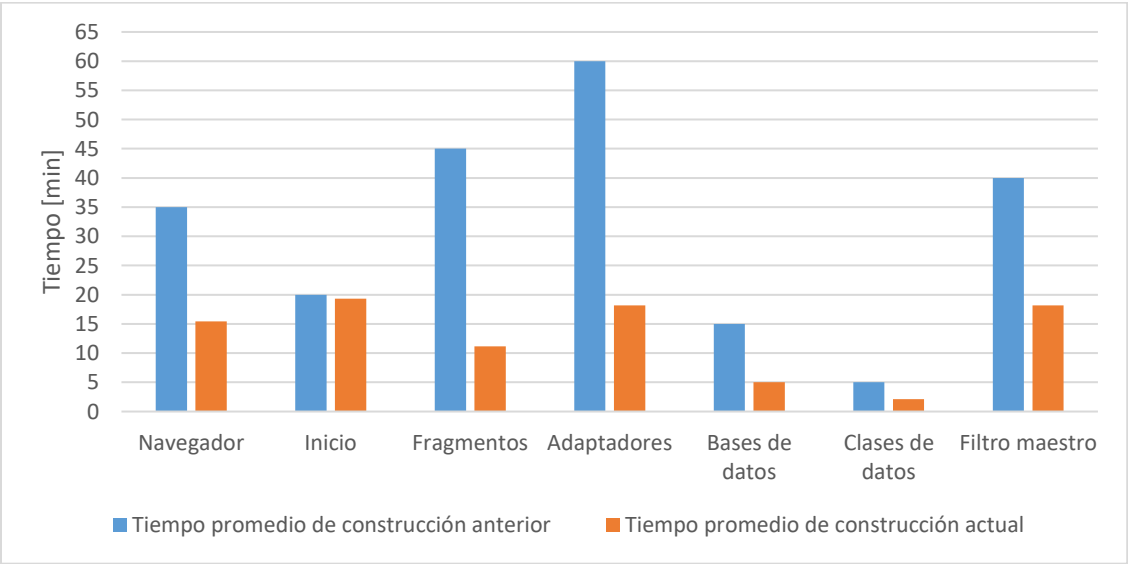
Optimizado del tiempo de construcción en los segmentos clave

Para medir la optimización del tiempo en la etapa de desarrollo preliminar del software se comparó el tiempo promedio de construcción de un segmento sin usar los archivos maestros contra el que toma construirlos usando los archivos maestros al aplicar el pensamiento basado en riesgos, la tabla 28 muestra los tiempos obtenidos para ambos métodos de construcción así como el % de tiempo reducido para cada segmento.

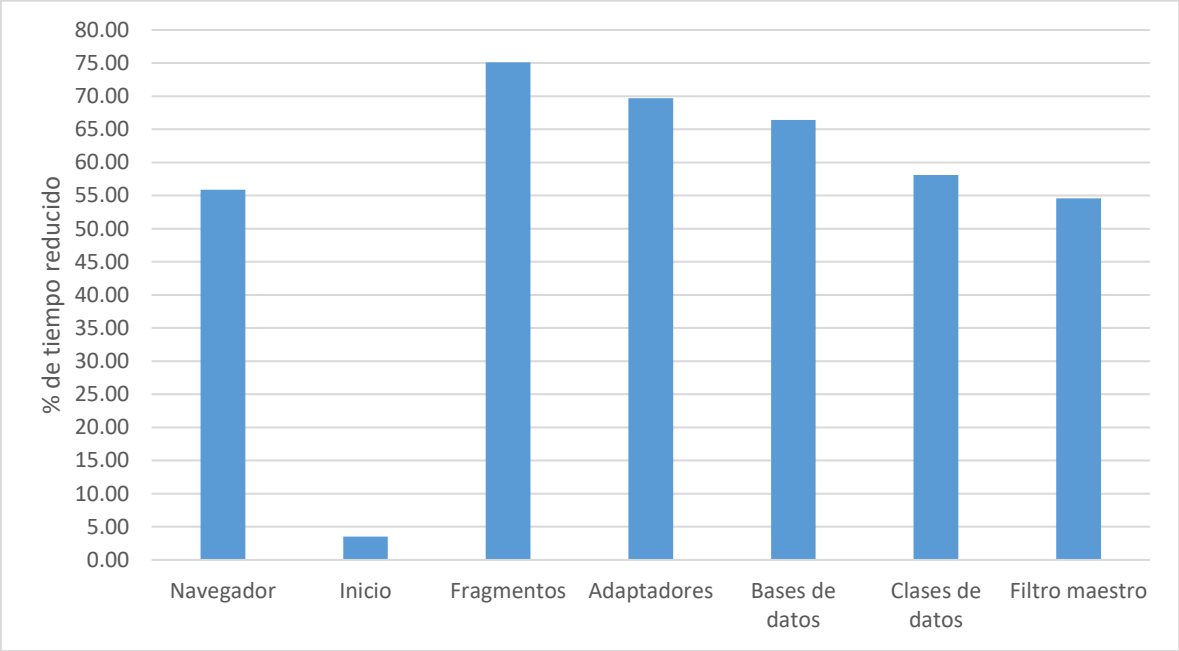
Tabla 28, Tiempos promedio de los segmentos clave y su % de reducción de tiempo de construcción promedio.

Segmento	Archivo	Tiempo promedio de construcción inicial [min.s]	Tiempo promedio de construcción actual [min.s]	% promedio de reducción de tiempo
Navegación	Navegador	35	15.43	55.91 %
	Inicio	20	19.31	3.5 %
	Fragmentos	45	11.19	75.14 %
Búsqueda	Adaptadores	60	18.17	69.71 %
	Bases de datos	15	5.04	66.43 %
	Clases de datos	5	2.095	58.10 %
Filtro	Filtro maestro	40	18.15	54.6 %
		Promedio de reducción de tiempo global:		54.8%

La figura 17 muestra el tiempo promedio de construcción sin emplear el pensamiento basado en riesgos contra tiempo de construcción empleando el pensamiento basado en riesgos



La figura 18 refleja el % de tiempo reducido para cada archivo de los segmentos clave



Matriz de riesgos residuales

Una vez detectados los riesgos en el desarrollo del software, se procedió a aplicar la gestión de riesgos, con la cual se logró cambiar la probabilidad y el impacto al implementar las acciones de tratamiento, los riesgos de tipo prioritario pasaron a ser moderados al igual que los riesgos de tipo importante, estos riesgos, de acuerdo con los resultados reflejados en la tabla 22 correspondiente a la sección de acciones para abordar riesgos y oportunidades el tratamiento del riesgo se conservarán y controlarán los riesgos remanentes.

Después de implementar las acciones establecidas en la sección del tratamiento del riesgo, se volvieron a evaluar los riesgos detectados en las sub etapas del desarrollo preliminar del software y en los archivos de código del software para determinar el nivel de riesgo que presentan posterior a la implementación de acciones para el tratamiento del riesgo, la tabla 29 muestra esta evaluación.

Tabla 29. Matriz de riesgos después de implementar las acciones de tratamiento del riesgo

Segmento/ Observación	Riesgo	Probabilidad	Impacto	Nivel del riesgo
Desarrollo principal No existe un proceso de revisión de correcciones que evite incorporar medidas que no arreglen los defectos	Repetir el proceso más veces de las necesarias consumiendo más tiempo del establecido	1	1	Moderado
	Generar soluciones de poco impacto o inadecuadas que no arreglen los defectos detectados	1	1	Moderado
Compilado general En la sub etapa de compilado general no se contempla la planeación para la implementación de acciones para solucionar los errores, únicamente se contempla en análisis y la implementación	Se consume más tiempo debido a que no se tiene establecido un protocolo por lo que cada vez que ocurre todo debe ser pensado desde 0	1	1	Moderado
Pruebas de validación El diagrama solo contempla la realización de pruebas no se considera la planeación de casos de uso para el correcto desempeño en las pruebas	El software puede no cumplir con los requisitos al no planearse correctamente los casos de prueba teniendo que reelaborar el diseño completo	1	1	Moderado
Navegación El árbol de navegación no contempla la movilidad entre pantallas de resultados pues esto se implementó sobre la marcha	El desarrollo de los segmentos clave se ve afectado por la implementación de características no requeridas en otras secciones.	1	1	Moderado

Existe un segmento que ya no forma parte de la versión actual y mantiene código sin usar	El software consume más recursos de los necesarios disminuyendo su eficiencia.	1	1	Moderado
El árbol de navegación es una representación gráfica del movimiento en el software y no un diagrama de flujo que describa el funcionamiento del módulo como tal	Se consume más tiempo de desarrollo al establecer el funcionamiento de la navegación, retrasando el desarrollo de otras secciones	1	1	Moderado
Existen archivos cuyos nombres son ambiguos en cuanto a su funcionalidad	Genera confusión para localizar y hacer modificaciones consumiendo más tiempo del que debería	1	1	Moderado
Existe un segmento que forma parte del código del software que no es visible y no puede ser eliminado de forma directa en virtud de que afecta la operación del software	Disminuye el rendimiento global del software y consumir tiempo adicional para buscar soluciones	1	1	Moderado
Existen herramientas de software de terceros que están incorporadas dentro del código, no son utilizadas y solo consumen recursos	El código tiene más secciones de las que debería generando confusión en los desarrolladores	1	1	Moderado
No existe como tal un diagrama de flujo que describa cómo debe funcionar la navegación	El código escrito puede no cumplir con la funcionalidad y versatilidad esperadas	1	1	Moderado
Los archivos de funcionalidad son difíciles de identificar debido a sus nombres	Consume tiempo y genera confusión al equipo desarrollador aumentando el tiempo de las tareas relacionadas a esos archivos	1	1	Moderado
Existen varios archivos en blanco los cuales no han sido eliminados	Genera más código del necesario, consume tiempo y aumenta el uso de memoria del software haciéndolo más pesado	1	1	Moderado

Para que los archivos sean estables se requiere escribir 2 secciones de código en un orden especifico, pero no existe documentación sobre cómo realizar esta operación	Deja los segmentos con un orden distinto inservibles	1	1	Moderado
Búsqueda No se indica que la entrada de solicitudes de búsqueda se actualiza en tiempo real cada segundo	Genera sesgos sobre la percepción del funcionamiento en el segmento.	1	1	Moderado
De acuerdo al requerimiento del cliente se desarrollaron 2 bases de datos que contienen información que hace más pesado el software y origina la disminución de su rendimiento global.	Aumenta el tiempo de respuesta del software dando la impresión de mal funcionamiento	1	1	Moderado
	El funcionamiento anómalo del software debido a las bases de datos causa un sesgo en las pruebas de aceptación que puede hacer que se descarte la versión.	1	1	Moderado
El sistema encargado de mostrar los resultados presenta una sintaxis que no está debidamente estructurada	La sintaxis podría causar fallos en el software como cierres inesperados, o congelaciones de pantalla	1	1	Moderado
Los archivos no tienen una secuencia lógica de acuerdo a su función	Causa un funcionamiento anormal del segmento de código	1	1	Moderado
Los archivos encargados de mostrar y presentar la información emplean herramientas de software de terceros inadecuados de funcionalidad	Disminuye la eficiencia del segmento de búsqueda, limita las funcionalidades del segmento y causa que los desarrolladores a buscar alternativas poco adecuadas	1	1	Moderado

Filtrado el proceso de recepción de resultados corresponde al diagrama de búsqueda, el proceso que hace falta en la recolecta de coincidencias con la solicitud recibida	Causa interrupciones constantes el desarrollo del segmento reduciendo el tiempo de desarrollo para los demás segmentos	1	1	Moderado
El sistema de filtrado emplea métodos descontinuados para su funcionamiento	El filtro del software dejaría de funcionar para versiones posteriores	1	1	Moderado
En todos los sistemas de filtrado de resultados existe una sección de código incompleta que genera cierres inesperados en algunos dispositivos	El software no tendría el funcionamiento requerido en los dispositivos y sería rechazado	1	1	Moderado
Archivos del sistema Se detectó que la sintaxis empleada para establecer comunicación entre archivos fue descontinuada, esto causó cierres inesperados en todos los segmentos	El software queda inservible	1	1	Moderado

Nuevos riesgos detectados a raíz de la implementación de acciones de tratamiento del riesgo

Durante la implementación de acciones para el tratamiento del riesgo se detectó que el proveedor externo de las herramientas que se utilizan en la construcción del software descontinuó un método para acceder a los segmentos clave del software mismo que era utilizado en varias etapas del desarrollo, originando cierres inesperados del sistema por lo que se procedió a actualizar los métodos de acceso afectados para asegurar su funcionalidad y reducir este riesgo a un nivel moderado.

Conclusiones

1. Se encontró que las cuestiones externas que afectan la etapa de desarrollo preliminar de software son:

    a. **Tecnológicas**
    La descontinuación de métodos de construcción de software, actualización de funciones del software, cambios en las prácticas de desarrollo y aparición de nuevas tecnologías para el desarrollo de software pueden cambiar sin previo aviso por lo que se debe dar seguimiento para evitar problemas en el desarrollo.
    b. **Legales**
    Las licencias de uso de software de terceros y los derechos de propiedad intelectual del software permiten el correcto desarrollo del software sin entrar en conflictos con terceros a la vez que se protege al software de usos indebidos.
    c. **Socioculturales**
    Dar seguimiento a las tendencias del consumidor y sus cambios permite desarrollar productos más amigables para el consumidor.
    d. **Políticas**
    Al establecer una política de privacidad y uso de datos acorde a las leyes, la organización evita problemas con el estado
    e. **Ecológicas**
    Estar al corriente de las regulaciones energéticas de su entorno se evitan sanciones e impacto negativos.
    f. **Económicas**
    Dar seguimiento a los cambios económicos del entorno permite a la organización establecer condiciones contractuales que mantengan los precios justos.

En el caso de las cuestiones internas que afectan la etapa de desarrollo preliminar del software se determinaron:

a. **Conocimiento de la organización**
   Con base en los conocimientos de la organización en el desarrollo se aceptan o no proyectos de diferentes naturalezas o de diversos grados de complejidad.
b. **Desempeño en el desarrollo del software**
   Establecer indicadores de desempeño en el desarrollo del software permite identificar áreas de mejora.
c. **Estrategia y objetivos del desarrollo**
   Permite optimizar el tiempo y los recursos empleados así como establecer alternativas y soluciones a los problemas que surjan durante el desarrollo del software.
d. **Modelos y directrices de la organización**
   Reduce el tiempo de construcción y mejora las prácticas de desarrollo al tomar en cuenta los modelos ya desarrollados.
e. **Flujos de información de la organización**
   Transmiten las instrucciones del proyecto, sus etapas y las funciones a desarrollar

Respecto de las partes interesadas, se encontraron las siguientes:

a. **Cliente**
   Decide si se asigna o no el proyecto, evalúa la factibilidad y funcionalidad del proyecto
b. **Equipos desarrolladores**
   Son los encargados de definir el plan de acción y el tiempo asignado para cada tarea del desarrollo preliminar
c. **Organización desarrolladora del software**
   Encargados de asignar los recursos humanos requeridos para el desarrollo del software
d. **Proveedores externos de herramientas de software**
   Responsables de proporcionar el lenguaje y entorno de programación para el desarrollo del software
e. **Usuario final**
   Se debe asegurar, la protección de sus datos personales así como una interfaz y uso amigable.

2. Los riesgos identificados son de 2 tipos, riesgos asociados a las sub etapas del desarrollo preliminar del software y riesgos asociados a los archivos que componen el código del software desarrollado, en ambos casos los niveles de riesgo fueron desde moderados hasta prioritarios.
3. De acuerdo con los requerimientos de las normas ISO 9000:2015 y 31000:2018 y con base a los niveles de riesgo identificados, las acciones implementadas para los riesgos prioritarios e importantes fueron dirigidas a

disminuir el riesgo, mientras que para el nivel moderado se consideró únicamente controlar el riesgo

4. En la integración de los archivos maestros se implementó la gestión del riesgo en el desarrollo del software y se reflejó una disminución del tiempo de construcción en los segmentos de navegación, búsqueda y filtrado, desde el 3.5% hasta el 75% de reducción del tiempo de construcción con una reducción de tiempo global del 54.8%.
5. En la implementación de acciones para el tratamiento del riesgo condujo a la detección de un nuevo riesgo, este se incluyó en la matriz de riesgos residuales y fue controlado.

Una vez aplicado el pensamiento basado en riesgos a la etapa de desarrollo preliminar de software de acuerdo con los requerimientos de las normas ISO 9001:2015 y ISO 31000:2018 se encontró, mediante la identificación y análisis de riesgo, que las principales fuentes de riesgo provienen de los diagramas de flujo y de los archivos propios del software los cuales presentaron riesgos desde relevantes hasta prioritarios mismos que fueron reducidos al nivel mínimo al implementar acciones para el tratamiento del riesgo.

Posteriormente al comparar los tiempos de desarrollo con pensamiento basado en riesgos se observó que el tiempo de desarrollo global de la etapa se reduce en un 54.8% respecto del tiempo original de desarrollo.

Referencias bibliográficas

1. Organización Internacional de Normalización. (2015). *Sistemas de gestión de calidad – Requisitos, Traducción oficial* (ISO 9001)
2. Organización Internacional de Normalización. (2018). *Gestión del riesgo – Directrices, Traducción Oficial* (ISO 31000)
3. Devi, R. (2012). Importance of Testing in Software Development Life Cycle. *5*, *3*(5), 1-5.
4. Chaitra, B., & Lesleeta, L. (2019). *A STUDY ON SOFTWARE DEVELOPMENT LIFE CYCLE*. *4*(1), 62-65.
5. McLeord, R., & Everett, G. (2007). *Software Testing: Testing Acroos the Entire Software Development Lifecycle*. IEEE Press.
6. Pressman, R. (2007). *Software Engineering A Practicioner´s Approach* (5ta edición). Mc Grawn Hill.
7. *Information Technology Policy*. (2017). Pennsylvania Office of Administration.
8. Langer, A. (2012). *Guide to Software development Designing and and Managing the Life Cycle*. Springer.

Anexo: Bibliografía

1. Manzoor, A., & Vivek, B. (2015). *A COMPARATIVE STUDY OF SOFTWARE DEVELOPMENT LIFE CYCLE MODELS*. *10*, *4*(10), 23-29.
2. Popa, M. (2010). *Audit Process during Projects for Development of New Mobile IT Applications*. *14*(3), 34-46.
3. Leigh, W., Barcia, R., & Omkar, C. (2015). *Enterprise class mobile application development—A complete lifecycle approach for mobile apps* (1st ed.). developerWork series IBM Press.
4. Olszewska, J., & Allison, I. (2020). *ODYSSEY: Software Development Life Cycle Ontology*. *2*(14), 301-310.