**Plan de aseguramiento de la calidad del software**

Aseguramiento de la calidad de software

Manuel Antonio Cituk Martínez

Jorge Teodoro Dawn Rodríguez

Geovanna Fernanda De La Cruz Medina

Alma Angélica Ordoñez Sánchez

Verónica Marilyn Rivera Manzanero

Mérida Yuc, 11 de Mayo 2023

Índice

[**Introducción 3**](#_lpnxao1h5u4u)

[**Alcance 4**](#_paymse5ajvee)

[**Propósito 4**](#_ap3h2vgvanf3)

[**Descripción general del documento 4**](#_q074i0psdcjs)

[Documentos 5](#_mdb3swymm2bo)

[**Documentos referenciados 6**](#_mp20pwfs0mcg)

[**Gestión 6**](#_kboqlz22xiii)

[Organización 7](#_2vvkhbks4c1n)

[Descripción de la unidad de SQA 7](#_bbufpiwh44kv)

[Desarrollo y mantenimiento SQA 8](#_cjcub0h30jpl)

[Roles en la unidad de SQA 9](#_o81zx8fhibyc)

[Tamaño de la unidad SQA 12](#_xaaou4j38s5f)

[Resolución de conflictos 12](#_2sr7izukyfci)

[Tareas 13](#_ant1natath6v)

[Tareas y responsabilidades de los miembros de los equipos de la empresa 17](#_57jjxkrgd8qv)

[**Documentación 19**](#_ezb92yos52a9)

[Plan de verificacion y validacion de software 19](#_pp8f5woggmgb)

[Informe de verificacion y validacion: 19](#_gzo0nz3vftmh)

[Requerimientos de software 22](#_nj3p3f38a60)

[**Estándares, prácticas, convenciones y métricas 29**](#_rmueu55u8y5)

[**Revisiones y auditorías 32**](#_8rbvmlwkvi29)

[**Pruebas 37**](#_u5cahqjec3zx)

[**Informe de problemas y acciones correctivas de SQA 42**](#_5y1cqwqunikf)

[**Herramientas, técnicas y metodologías 44**](#_rfblm6q8bbnp)

[Herramientas 44](#_nvpbxmd3x4li)

[Técnicas 46](#_3cz8qgv3lag6)

[Metodologías 47](#_49a7ncxma1n3)

[**Control de código 48**](#_d4glxp2dnqvt)

[**Control de multimedia 50**](#_jc984n68lnzh)

[**Control de proveedores 50**](#_a5b8ox22zp6j)

[**Capacitación continua 51**](#_pmu8nz10wj6a)

[**Gestión de riesgos 52**](#_g5jzqwuqzjc3)

[**ANEXOS 54**](#_fe7a64tce6)

# Introducción

El siguiente documento contiene el Plan de Aseguramiento de Calidad del Software para el desarrollo de la aplicación móvil dirigida hacia las mujeres que la empresa **FemmeCode** realizó. Dicha aplicación móvil tendrá como propósito brindar seguridad a las mujeres ante posibles escenarios de violencia a través de un botón de emergencia que en cuestión de segundos alerta a las autoridades. De igual forma, la aplicación brindará asesoría legal al momento de que las mujeres presenten una denuncia y a través de foros podrán compartir sus experiencias con otras mujeres.

# Alcance

El presente documento se encuentra dirigido a todo el equipo de desarrollo de software, pruebas y mantenimiento. De igual forma, va dirigido hacia todas aquellas personas que tengan interés en el desarrollo de la aplicación, ya que fue elaborado para establecer un marco de trabajo en donde se hace referencia a los documentos del desarrollo de software para asegurar la calidad de los mismos.

# Propósito

El SQAP, también conocido como Plan de Aseguramiento de la Calidad del Software, fue concebido para establecer un marco de trabajo riguroso y estructurado que permita garantizar la calidad del software en todo su ciclo de vida.

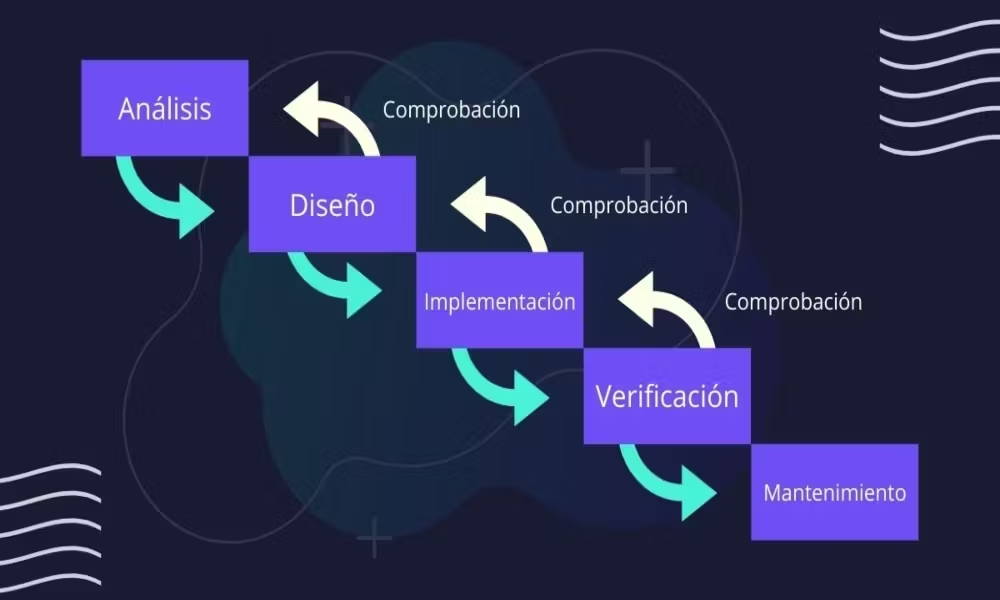
Este documento describe minuciosamente los procedimientos y directrices necesarias para que todos los actores involucrados en el proceso de desarrollo de software puedan producir, recopilar y validar la evidencia necesaria para demostrar que el software desarrollado cumple con los requisitos establecidos en términos de calidad y funcionalidad. En esencia, el SQAP es una hoja de ruta detallada y sofisticada que garantiza que el software producido cumpla con los más altos estándares de calidad y se ajuste a las expectativas del cliente y de los usuarios finales.

# Descripción general del documento

Este documento contiene:

1. Las fases del ciclo de desarrollo de software
2. Procesos de la documentación
3. Verificación y validación
4. Proceso de acciones preventivas y correctivas

El modelo del ciclo de desarrollo de software a usar será el de “cascada”. En este modelo, el proceso de desarrollo de software se divide en una serie de fases secuenciales y bien definidas, en las que cada fase debe completarse antes de pasar a la siguiente. El modelo toma su nombre de la forma en que el software fluye a través de las 4 fases (requisitos, diseño, construcción, pruebas y mantenimiento) de arriba hacia abajo, como una cascada.



*Imagen1.0 Descripción del modelo de ciclo de vida en cascada*

**Fases del ciclo de desarrollo del software.**

## Documentos

A continuación, se mencionarán aquellos documentos que forman parte de las fases del ciclo de vida de acuerdo a lo establecido en nuestro [modelo de procesos](https://github.com/ver0rivera/ProcessModel-SQA) basado en el Moprosoft.

Fase de requisitos:

* Documento de especificación de requisitos
* Plan de pruebas del sistema
* Manual de usuario

Fase de diseño:

* Documento de análisis y diseño
* Plan de desarrollo
* Registro de rastro
* Reporte de actividades

Fase de construcción:

* Documentos de control
* Documento de costos de control
* Análisis y diseño de la construcción del código
* Pruebas unitarias
* Registro de rastro
* Revisiones
* Reporte de actividades

Fase de pruebas:

* Plan de desarrollo
* Plan de pruebas de integración
* Manual de operaciones
* Manual de usuario

Fase de mantenimiento:

* Manual de mantenimiento
* Reporte de lo aprendido
* Reporte de mediciones

# Documentos referenciados

Software Quality Assurance- Libro que aborda los conceptos fundamentales y prácticas relacionadas con el aseguramiento de calidad del software. El libro incluye una introducción a los principios básicos de la ingeniería de software y cubre una amplia gama de temas relacionados con la calidad del software, como la planificación y la gestión de la calidad, la evaluación y mejora de procesos, la verificación y validación, la gestión de la configuración y la garantía de calidad del producto.

**Anexo “Software Quality Assurance”**

IEEE Standard for Software Quality Assurance Plans. - Describe los requisitos para la planificación y el contenido de un plan de aseguramiento de la calidad del software (SQA). El estándar establece las actividades y tareas que deben realizarse durante el desarrollo del software para garantizar que se cumplan los requisitos de calidad.

1.- IEEE Std 730™-2002

2.- IEEE Std 730™-2014

3.- IEEE Std 730.1-1995

**Anexo “Estándares para el aseguramiento de la calidad”**

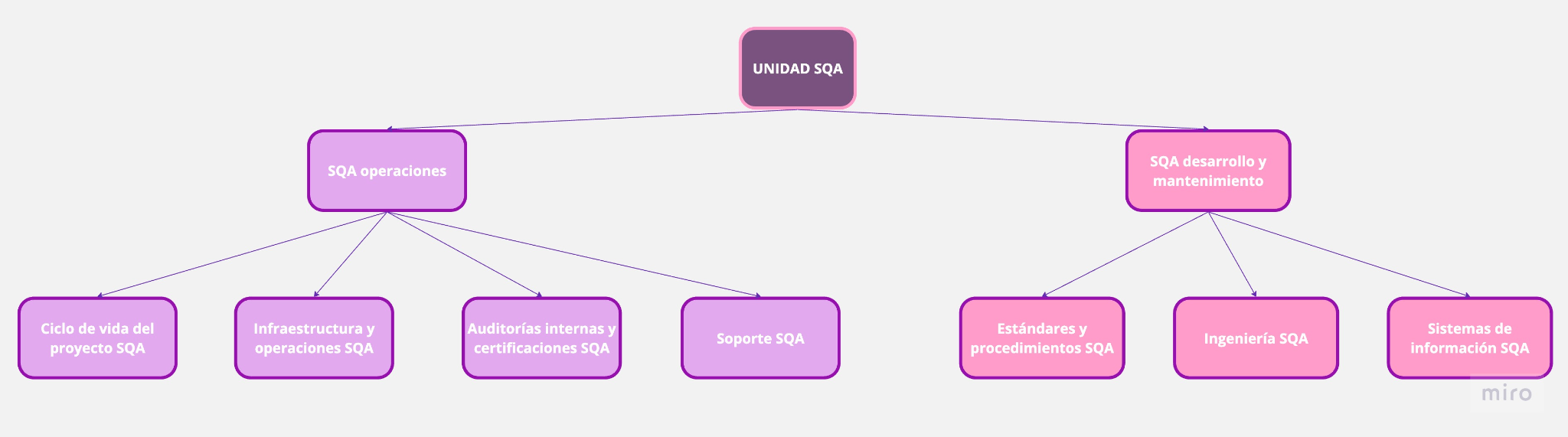
# 

# Gestión

## Organización

La unidad de SQA, o Aseguramiento de la Calidad del Software, es un equipo de trabajo que se encarga de supervisar y garantizar que el software desarrollado cumpla con los estándares de calidad establecidos. Esta unidad se encarga de realizar pruebas, evaluaciones y auditorías de todo el proceso de desarrollo del software para asegurarse de que se estén siguiendo los procedimientos adecuados y se estén cumpliendo los requisitos establecidos.

La unidad de SQA está compuesta por un conjunto de actores que trabajan en colaboración para lograr este objetivo. Cada actor tiene establecido un rol y participación definido.



*Imagen 2.0 Unidad de SQA-FemmeCode*

Como se puede observar en la imagen 2.0, los departamentos contienen unidades que tienen funciones y tareas definidas de acuerdo con el libro Software Quality Assurance.

**Anexo “Software Quality Assurance”**

## Descripción de la unidad de SQA

OPERACIONES SQA

* Unidad del ciclo de vida del proyecto SQA

Se define como un equipo multidisciplinario que se encarga de asegurar la calidad del software a lo largo de todo el ciclo de vida del proyecto.

* Unidad de infraestructura y operaciones SQA

Se describe como una unidad dentro de la organización de SQA que se enfoca en asegurar la calidad de los procesos de infraestructura y operaciones de TI, que son necesarios para mantener y hacer funcionar el software en un ambiente de producción.

* Unidad de auditorías internas y certificaciones SQA

Se encarga de auditar, evaluar y certificar la calidad de los procesos y productos de software de la organización, para garantizar que se cumplan los estándares y requisitos de calidad establecidos por la organización y por los organismos reguladores externos.

* Soporte SQA

Proporciona apoyo y recursos para la implementación efectiva de los procesos y estándares de calidad de software en toda la organización.

## Desarrollo y mantenimiento SQA

* Estándares y procedimientos SQA

Se encarga de establecer, documentar y mantener los estándares y procedimientos de calidad de software para toda la organización.

* Ingeniería SQA

Se enfoca en garantizar que los productos de software se encuentren diseñados y construidos de acuerdo con los estándares y procedimientos de calidad establecidos.

* Sistemas de información SQA

establecer, mejorar y mantener un sistema de información dedicado a la gestión de la calidad del software.

## Roles en la unidad de SQA

* Ejecutivo SQA

El papel del Ejecutivo de SQA es liderar y dirigir la implementación de SQA en la organización. Este es un papel de liderazgo estratégico que se enfoca en establecer y mantener políticas y estándares de SQA en toda la organización.

Gerente de SQA.

* Líder del equipo

Encargado de coordinar todas las actividades de SQA, desde la planificación y definición de estrategias hasta la ejecución y seguimiento de las pruebas y la documentación.

* Ejecutor de estándares

Localiza desviaciones de estándares, normas y procedimientos, de igual forma participa en el desarrollo y actualización de estos mismos.

* Especialista en pruebas

Diseña y ejecuta planes de pruebas para asegurarse de que el software funcione correctamente antes de ser entregado al cliente. También puede realizar pruebas de rendimiento y de seguridad.

* Responsable de documentación

Documenta todo el proceso de desarrollo de software, incluyendo los requisitos, el diseño, la implementación, las pruebas y la documentación del usuario.

* Desarrollador de software

Crea el software siguiendo los estándares de calidad establecidos. El trabajo del desarrollador es evaluado y supervisado por los analistas de calidad y los especialistas en pruebas.

* Auditor de calidad

Se encarga de realizar auditorías del proceso de desarrollo de software y de la documentación correspondiente, para asegurarse de que se están siguiendo las políticas y procedimientos adecuados.

* Comité de acciones correctivas (CAB)

Comité que se encarga de analizar y resolver los problemas relacionados con la calidad del software en una organización. El objetivo del CAB es identificar las causas raíz de los problemas de calidad, y tomar medidas correctivas para evitar que estos problemas se repitan en el futuro.

* Junta de control de cambios (CCB)

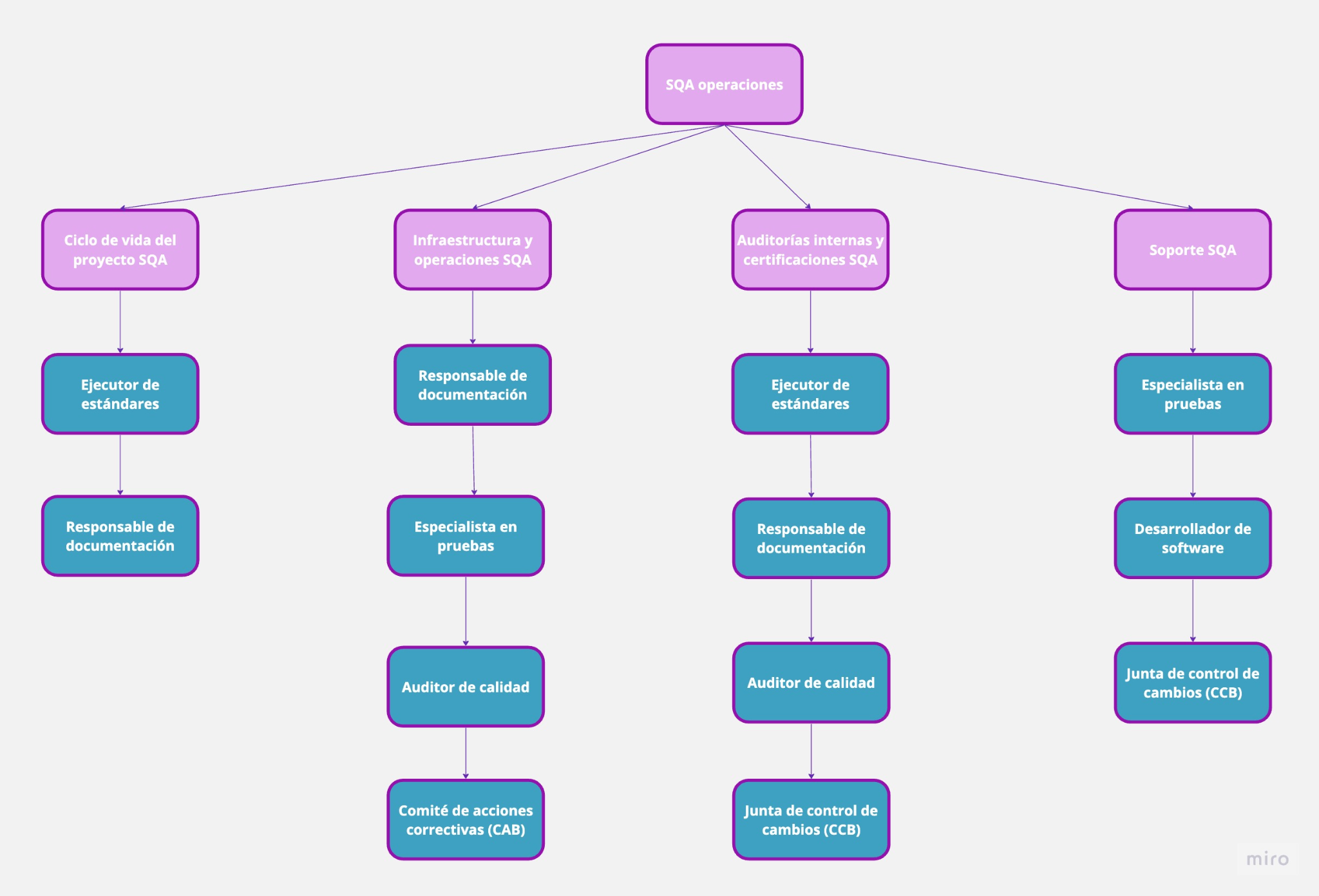
Personas responsables de revisar, evaluar y aprobar o rechazar cualquier cambio propuesto en el software o en los procesos relacionados con la calidad del software. La junta está compuesta por expertos en el área de desarrollo de software, gestión de proyectos y control de calidad.

* Fideicomisarios

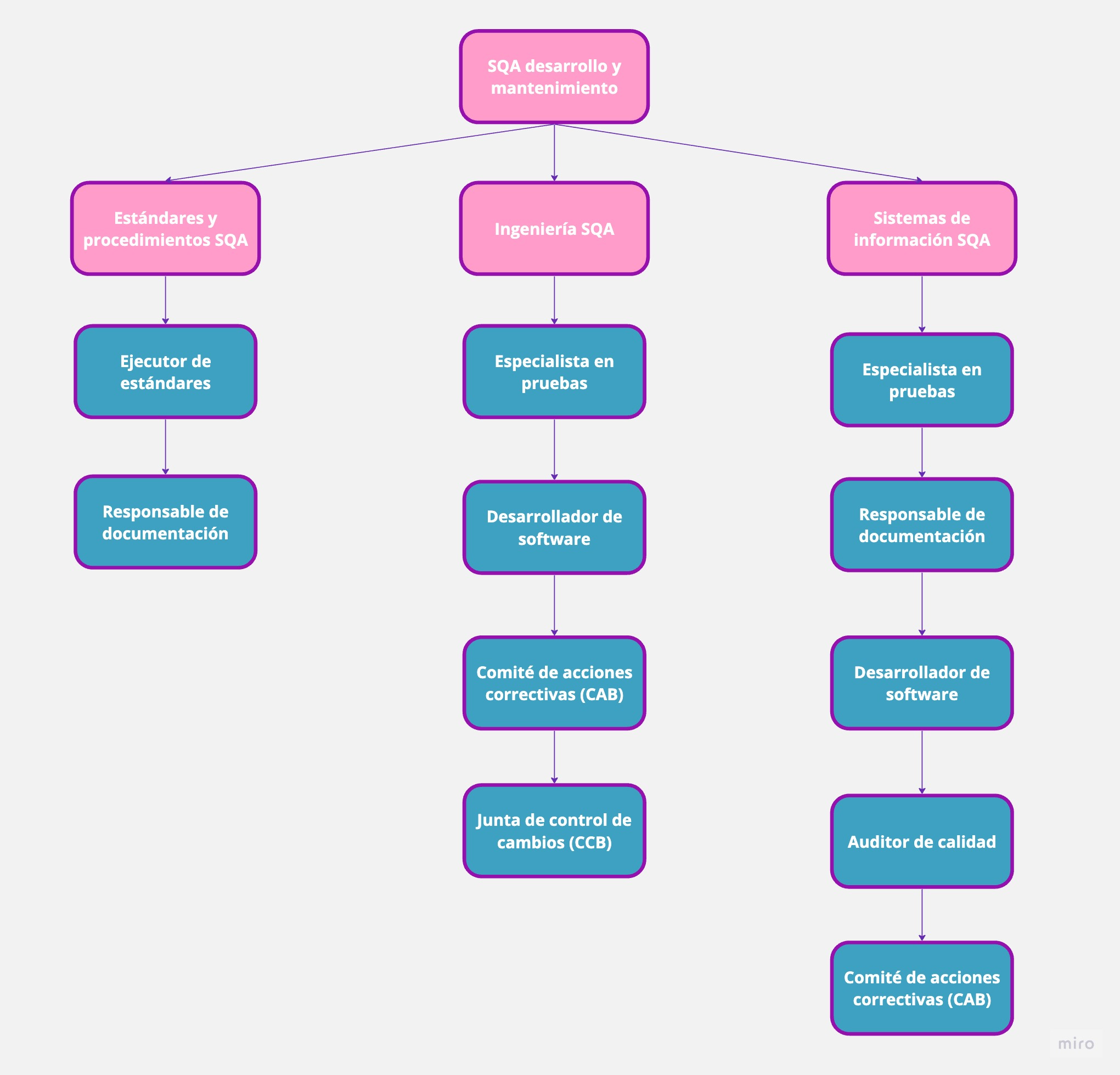
Agentes de SQA que brindan el apoyo interno necesario para implementar con éxito los componentes de SQA.

El establecimiento de los roles y la definición de los mismos se llevó a cabo con base en el libro Software Quality Assurance.

**Anexo “Software Quality Assurance”**



*Imagen 3.0 Roles del departamento de operaciones SQA*



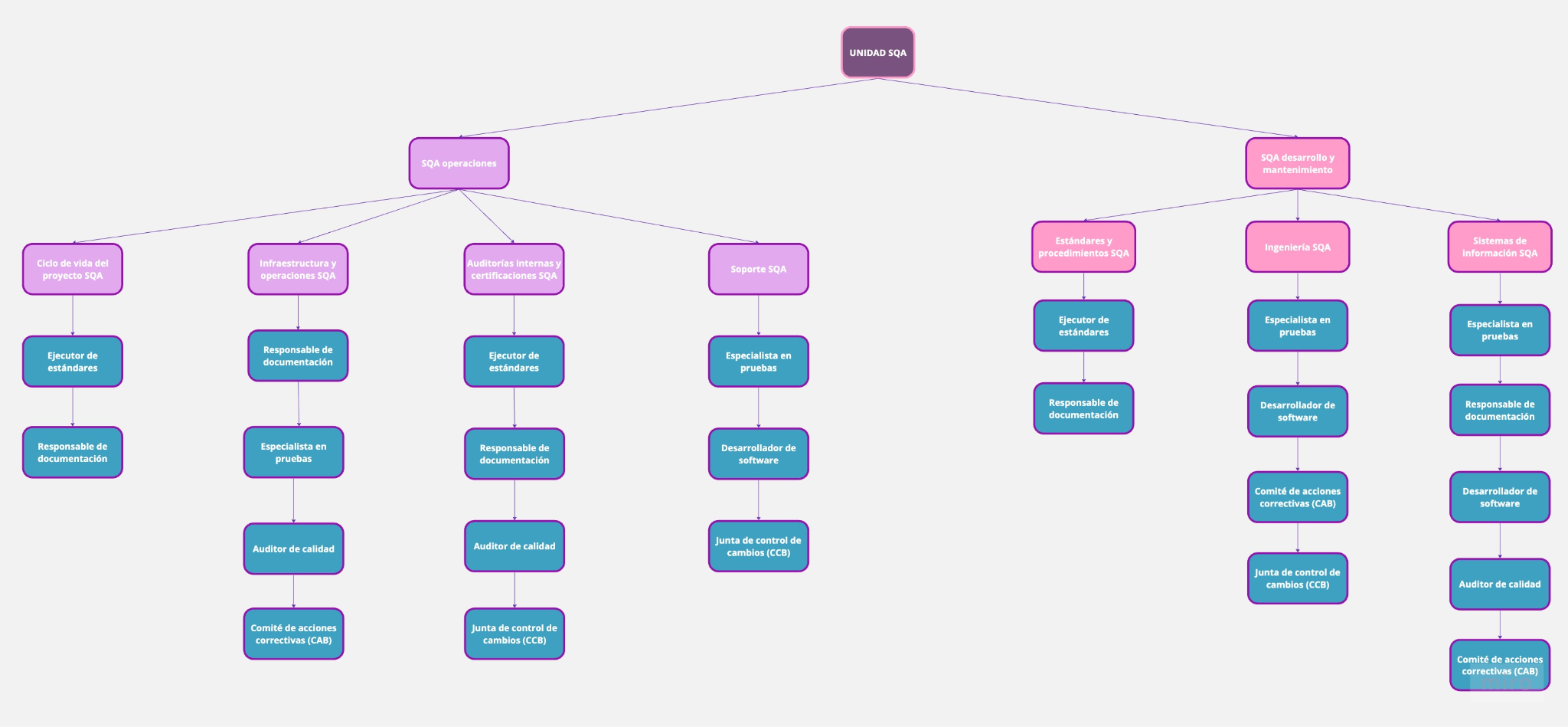
*Imagen 4.0 Roles del departamento de desarrollo y mantenimiento SQA*

## Tamaño de la unidad SQA

Como se puede observar en la imagen 5.0, la unidad de SQA para la empresa Femmecode, se encuentra dividida de la siguiente manera:

|  |  |
| --- | --- |
| **Ejecutivo de SQA** | 1 |
| **Departamentos** | 2 |
| **Gerente de unidad SQA** | 2 (uno por cada departamento) |
| **Unidad de operaciones** | 4 |
| **Unidad de desarrollo y mantenimiento** | 3 |

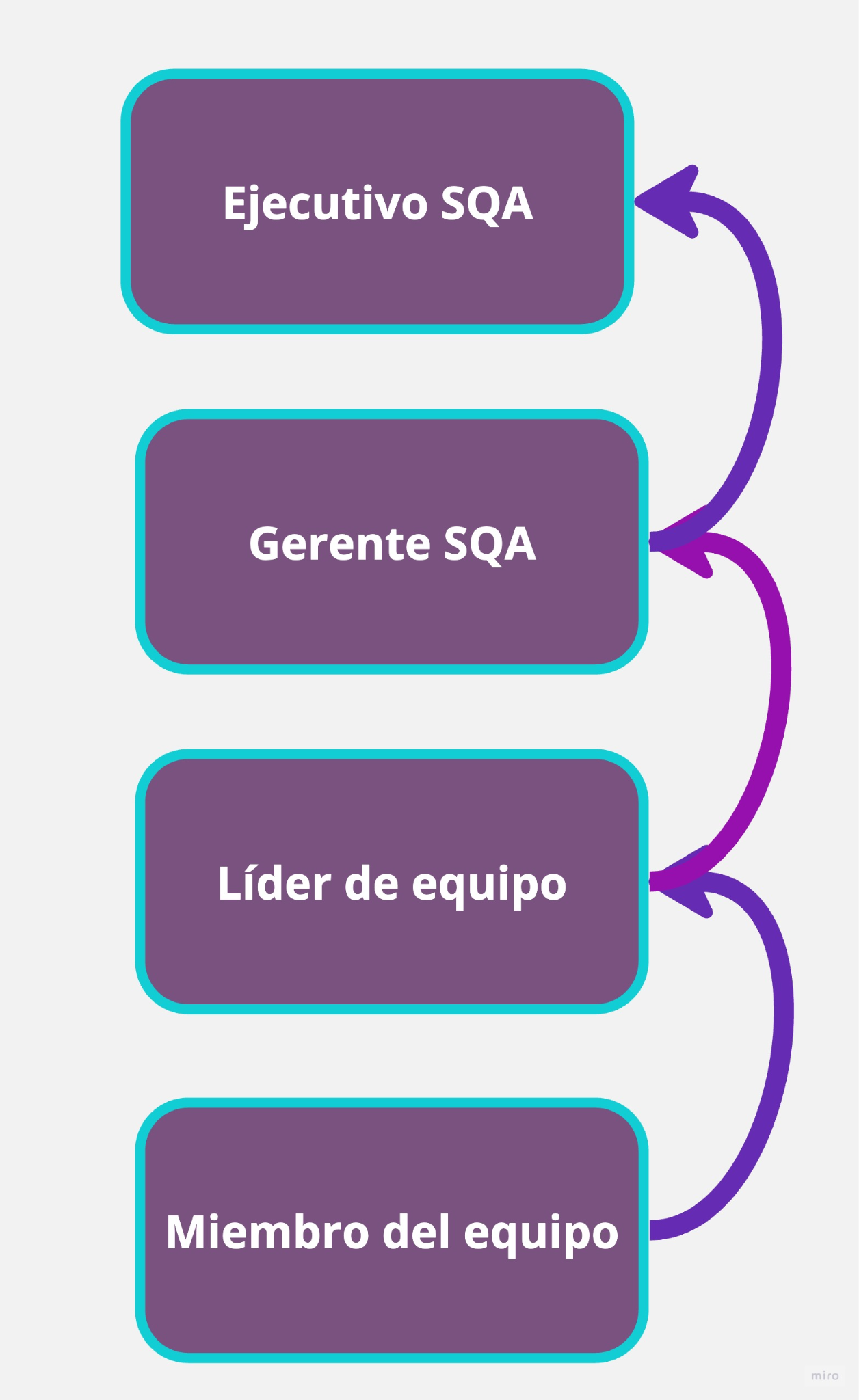
*Tabla de divisiones de la unidad de SQA-Femme*



*Imagen 5.0 Organigrama completo unidad de SQA-FemmeCode*

## Resolución de conflictos

De acuerdo con la jerarquía previamente establecida, es muy importante mencionar que la empresa FemmeCode manejará la resolución de conflictos de manera eficiente notificando a los miembros más altos de la organización para lograr encontrar la mejor solución. Imagen 6.0.



*Imagen 6.0 Diagrama de resolución de conflictos- FemmeCode*

## Tareas

**Tareas relacionadas con la unidad de SQA**

|  |  |
| --- | --- |
| **Ejecutivo SQA** | |
| **Planificación** | * Planificar y actualizar el sistema de gestión de la calidad del software de la organización. * Preparar planes de desarrollo de sistemas SQA recomendados para los departamentos de desarrollo y mantenimiento de software. |
| **Administración** | * Gestionar las actividades del equipo SQA. * Monitorea la implementación del programa de actividades de SQA. * Denominar miembros del equipo, miembros del comité de SQA y fideicomisarios de SQA. |
| **Contacto con clientes y organismos externos** | * Representar a la organización ante organismos externos en temas de calidad del software. * Realizar cuestiones organizativas de SQA, preparación del material solicitado, etc. Para consideración de la alta dirección. |
| **Actividades profesionales** | * Participar en comités conjuntos de proyectos y en revisiones formales de diseño. * Revisar y aprobar desviaciones de las especificaciones. * Consultar con gerentes de proyecto y líderes de equipo. |
| **Subunidad de operaciones SQA** | |
| **Control del ciclo de vida del proyecto** | Participar activamente en las actividades de SQA del equipo del proyecto, donde se requieren contribuciones profesionales. |
| **Operaciones de infraestructura** | Publicar de versiones actualizadas de:   * Procedimientos * Instrucciones de trabajo * Plantillas * Listas de verificación. |
| **Auditorías y certificación** | Auditorías internas de SQA :   * Elaborar programas anuales para auditorías internas SQA. * Brindar seguimiento a las correcciones y mejoras a realizar por los equipos auditados y otras unidades. * Preparar informes resumidos periódicos sobre el estado de los resultados de la auditoría, incluidas las recomendaciones de mejora.   Las auditorías SQA de subcontratistas y proveedores:   * Elaborar el programa anual de auditorías SQA de subcontratistas y proveedores. * Brindar seguimiento a las correcciones y mejoras a realizar por parte de los subcontratistas y proveedores auditados. * Recopilar datos sobre el desempeño de subcontratistas y proveedores de fuentes internas y externas. * Evaluar periódicamente los sistemas de SQA de los proveedores y subcontratistas certificados de la organización en función de los informes de auditoría y la información recopilada de otras fuentes internas y externas.   Auditorías externas:   * Coordinar el contenido y cronograma de la auditoría de certificación. * Elaborar documentos especificados por los organismos de certificación. * Realizar instrucciones para los equipos auditados y realización de los preparativos necesarios para las auditorías de certificación. * Participar en auditorías de certificación. * Asegurarse de que se realicen las correcciones y mejoras requeridas.   Auditorías por parte de los clientes de la organización:   * Coordinar el contenido y cronograma de la auditoría de certificación. * Elaborar documentos especificados por los organismos de certificación. * Realizar la instrucción de los equipos auditados y realización de los preparativos necesarios para las auditorías de certificación. * Participar en auditorías de certificación. * Asegurarse de que se realicen las correcciones y mejoras requeridas. |
| **Tareas de apoyo** | El apoyo que necesitan gira en torno a la implementación de procedimientos SQA:   * Elaborar de planes de proyectos y planes de calidad de proyectos. |
| **Subunidad de desarrollo y mantenimiento SQA** | |
| **Estándares y procedimientos** | * Preparar un programa anual para el desarrollo de nuevos procedimientos y actualizaciones de procedimientos, incluyendo. * Hacerse responsable por el desarrollo de nuevos procedimientos y actualizaciones de procedimientos. * Dar seguimiento de desarrollos y cambios en SQA y estándares de ingeniería de software; introducción de procedimientos adicionales y cambios relevantes para la organización. * Iniciar actualizaciones y adaptaciones de procedimientos en respuesta a cambios en los estándares profesionales, incluida la adopción o eliminación de estándares aplicados por la organización. |
| **Desarrollo y mantenimiento de ingeniería** | * Probar aspectos de calidad y productividad con respecto a nuevas herramientas de desarrollo y nuevas versiones de herramientas de desarrollo actualmente utilizadas. * Realizar evaluaciones de la calidad y productividad de nuevos métodos de desarrollo y mantenimiento y mejoras de métodos * Proveer de soporte tecnológico a los comités del CAB durante el análisis de fallas en el desarrollo de software y formulación de propuestas de solución. * Trabajar en el desarrollo de soluciones a las dificultades enfrentadas en la aplicación de herramientas y métodos de desarrollo de software actualmente utilizados. |
| **Sistemas de información** | * Desarrollar sistemas de información de SQA que faciliten el procesamiento de la información proporcionada por las unidades de desarrollo y mantenimiento de software. * Actualizar los sistemas de información de SQA. * Desarrollo y mantenimiento de SQA Internet de la organización /sitio intranet. |
| **Fideicomisarios SQA** | Tareas relacionadas con la unidad:   * Apoyar los intentos de sus colegas para resolver las dificultades que surjan en la implementación de los procedimientos de calidad del software. * Ayudar a su gerente de unidad a realizar sus tareas de SQA * Promover el cumplimiento y monitorear la implementación de los procedimientos SQA y las instrucciones de trabajo por parte de los colegas. * Reportar eventos de incumplimiento sustanciales y sistemáticos a la unidad SQA. * Reportar fallas severas de calidad del software a la unidad SQA.   Tareas relacionadas con la organización:   * Iniciar cambios y actualizaciones de procedimientos SQA e instrucciones de trabajo en toda la organización. * Iniciar mejoras en toda la organización de los procesos y aplicaciones de desarrollo y mantenimiento para el CAB para soluciones a fallas recurrentes observadas en sus unidades. * Identificar las necesidades de capacitación en SQA de toda la organización y proponer un programa de instrucción o capacitación adecuado para que lo lleve a cabo la unidad de SQA. |
| **Comités: Control de cambios y acciones correctivas** | |
| **Tareas** | Deberán ocuparse de casos específicos de interés más general:   * Actualizar un procedimiento específico * Análisis y solución de una falla de software * Elaboración métricas de software para un proceso o producto objetivo * Actualizar los costos de calidad de software y métodos de recopilación de datos para un producto específico |

Las tareas y responsabilidades descritas previamente se llevaron a cabo con base en el libro Software Quality Assurance.

**Anexo “Software Quality Assurance”**

## Tareas y responsabilidades de los miembros de los equipos de la empresa

Los roles involucrados en el desarrollo de todo el proyecto se encuentran definidos en nuestro modelo de procesos en la sección de -Prácticas-

**Anexo “Modelo de Procesos”**

Deberán seguir sus responsabilidades en tiempo y forma de acuerdo a los hitos del desarrollo del proyecto.

Fase de Requerimientos:

* Finalización de la especificación de requisitos
* Revisión y aprobación de la especificación de requisitos
* Creación del plan de pruebas de requerimientos
* Finalización de la revisión de pruebas de requerimientos
* Aprobación de pruebas de requerimientos

Fase de Diseño:

* Finalización del diseño de arquitectura
* Finalización del diseño detallado de la aplicación
* Revisión y aprobación del diseño de arquitectura
* Revisión y aprobación del diseño detallado de la aplicación
* Creación del plan de pruebas de diseño

Fase de Codificación:

* Finalización de la implementación del código
* Revisión del código por pares
* Pruebas unitarias completadas
* Finalización de la revisión del código por pares
* Aprobación del código

Fase de Pruebas:

* Finalización de las pruebas de integración
* Finalización de las pruebas de sistema
* Revisión y aprobación del plan de pruebas de aceptación del usuario
* Pruebas de aceptación del usuario completadas
* Aprobación de las pruebas de aceptación del usuario

Fase de Mantenimiento:

* Identificación y resolución de errores de software reportados por los usuarios
* Actualizaciones de software programadas y completadas
* Revisiones y actualizaciones de documentación de software
* Revisión y actualización del plan de pruebas de mantenimiento
* Aprobación del plan de pruebas de mantenimiento actualizado

# Documentación

Todos los documentos descritos deberán ser sometidos a una revisión formal en donde se utilicen como referencia los siguientes estándares:

* Requerimientos de software: IEEE Std 830-1998
* Descripción de diseño de software: IEEE std 1016 - 2009
* Plan de verificación y validación de software: IEEE std 1012-2016
* Informe de Verificación y Validación del Software: IEEE std 1012-2016
* Documentación del usuario: IEEE std 1063 - 1987
* Plan de Gestión de la Configuración del Software: IEEE 828-2012
* Manual de mantenimiento: ISO/IEC/IEEE 14764-2019

## Plan de verificación y validación de software

El plan, se puede encontrar dando click en el enlace:

[Plan de verificacion y validacion de software](https://docs.google.com/document/d/1rTIo8jJU609Ic-J7Rk5-6io3I9lAiWCcdZ0v2inrGzw/edit?usp=sharing)

## Informe de verificación y validación

En este documento, se deben resumir los resultados y hallazgos obtenidos durante el proceso de verificación y validación del software. Se debe proporcionar una visión general de las actividades realizadas, los criterios de verificación y validación utilizados y los resultados obtenidos en cada etapa del proceso.

Se realizará un informe completo que deberá incluir los siguientes elementos:

1. Introducción: Describe el propósito y el alcance del informe, así como la metodología y los criterios de verificación y validación utilizados.
2. Descripción del software: Proporciona una visión general del software que se ha desarrollado, incluyendo sus características, funcionalidades y requisitos clave.
3. Actividades de verificación: Resume las actividades realizadas durante la verificación del software, como revisiones de requisitos, diseño y código, pruebas unitarias, pruebas de integración, etc. Se describen los enfoques, métodos y herramientas utilizados durante estas actividades.
4. Resultados de la verificación: Presenta los resultados obtenidos durante las actividades de verificación, incluyendo los problemas identificados, las correcciones realizadas y la conformidad del software con los requisitos establecidos. También se pueden incluir métricas de calidad y resumen de cobertura de pruebas.
5. Actividades de validación: Resume las actividades realizadas durante la validación del software, como pruebas funcionales, pruebas de rendimiento, pruebas de usabilidad, pruebas de seguridad, etc. Se describen los enfoques, escenarios de prueba y resultados obtenidos durante estas actividades.
6. Resultados de la validación: Presenta los resultados obtenidos durante las actividades de validación, incluyendo la conformidad del software con los requisitos de funcionamiento, rendimiento, usabilidad, seguridad, etc. Se destacan los problemas identificados y las mejoras implementadas.
7. Conclusiones: Resume las conclusiones generales obtenidas a partir de los resultados de verificación y validación. Puede incluir recomendaciones para futuras mejoras y acciones correctivas.
8. Anexos: Incluye cualquier documentación adicional relevante, como casos de prueba, registros de errores, capturas de pantalla, evidencias de pruebas, etc.

El objetivo de realizar un informe completo es documentar y comunicar de manera efectiva todos los aspectos relacionados con la verificación y validación del software. Al crear un informe completo, se busca cumplir con varios propósitos importantes:

* Comunicación clara: El informe completo permite comunicar de manera clara y detallada cómo se ha llevado a cabo la verificación y validación del software. Proporciona una descripción exhaustiva de las actividades realizadas, los criterios utilizados, los resultados obtenidos y las conclusiones alcanzadas. Esto es esencial para que los interesados, incluidas las autoridades y los usuarios, comprendan el proceso y la calidad del software desarrollado.
* Transparencia y confianza: Al elaborar un informe completo, se demuestra transparencia en el proceso de desarrollo y en las medidas tomadas para garantizar la seguridad de las mujeres. Proporciona evidencia de que se han seguido procedimientos adecuados y se ha realizado una verificación exhaustiva para garantizar que la aplicación cumpla con los requisitos y estándares establecidos. Esto genera confianza tanto en las autoridades como en las usuarias de la aplicación.
* Evaluación y mejora continua: El informe completo sirve como punto de referencia para evaluar la eficacia del desarrollo y la implementación de la aplicación. Permite identificar áreas de mejora y posibles problemas que requieren atención adicional. Al documentar los resultados y los desafíos encontrados, se pueden tomar medidas correctivas y realizar mejoras continuas en el software para garantizar su óptimo funcionamiento y seguridad.
* Cumplimiento normativo y legal: En muchos casos, las aplicaciones que tienen un propósito específico, como la seguridad de las mujeres, deben cumplir con ciertos requisitos normativos y legales. El informe completo puede servir como evidencia de que se han seguido los protocolos y los estándares requeridos. También puede facilitar el proceso de evaluación y aprobación por parte de las autoridades pertinentes.

**Descripción del diseño de software**

**Interfaces de hardware.**

No aplica.

**Interfaces de software.**

Productos de software requeridos:

React Native.

Para el desarrollo de la aplicación. Permite a los desarrolladores crear aplicaciones móviles para iOS, Android y la web utilizando el mismo código de React, una biblioteca de JavaScript para crear interfaces de usuario.

Servidor de aplicaciones:

Base de datos NoSQL (No Structured Query Language) MongoDB .

La aplicación necesita una base de datos para almacenar la información de las cuentas que creen las mujeres con sus datos personales.

Rest API.

La aplicación necesita de esta API para el acceso a los datos.

**Interfaces con otras aplicaciones desarrolladas.**

Google Maps API

La aplicación permitirá guardar la ubicación de las mujeres al momento de activar el botón de ayuda para enviarla a las autoridades y guardarla en la cuenta de la aplicación.

**Interfases de comunicación**

No aplica

**Restricciones de memoria.**

No aplica

**Operaciones.**

Requerimientos para que el software funcione correctamente:

* El software deberá registrar las credenciales de las usuarias al momento que creen sus cuentas, para ello deberá realizar la conexión a la base de datos.
* El software deberá permanecer como “Sesión iniciada” en todo momento a menos que la usuaria decida cerrar sesión, esto quiere decir que deberá permanecer en funcionamiento en todo momento.
* El software deberá reaccionar de manera inmediata cuando el botón de ayuda se quede presionado por más de 3 seg, estableciendo la conexión con la base de datos para almacenar la ubicación detectada.
* El software deberá funcionar para enviar la alerta a las autoridades independientemente cuente con conexión a internet o no.

**Requerimientos de adaptación del sitio.**

No aplica.

Para asegurar la calidad del diseño de software previamente mencionado, se hará uso del estándar IEEE 1016-2009 "Estándar para la documentación del diseño de software". El cuál, establece los requisitos para la documentación del diseño de software y proporciona directrices para la creación de documentos de diseño que sean completos, coherentes y precisos. De igual forma define los elementos que deben incluirse en la documentación del diseño de software, tales como la descripción del sistema, la arquitectura del sistema, los detalles de la interfaz, los algoritmos y estructuras de datos, y la descripción detallada de cada componente del sistema.

**Ver anexo “Estándar documentación del diseño”**

**Plan de gestión de la configuración de software.**

Con el fin de mantener una gestión efectiva de la configuración de software al identificar, controlar e implementar cambios en los ítems, en nuestro Modelo de Procesos de software se estableció un documento de “Administración de la configuración de software” que contiene los pasos a seguir para gestionar toda la documentación.

**Ver anexo “Administración de la configuración”**

Este mismo documento menciona que la gestión de control de versiones y cambios, implementará el modelo de evolución lineal. De igual forma implementa una plantilla de solicitud de cambios que deberá aprobar aquella persona destinada a la revisión de los productos.

**Manual de mantenimiento del Software.**

El Manual de Mantenimiento contendrá información detallada sobre cómo realizar el mantenimiento del software de manera efectiva. Aquí se mencionan los elementos mínimos que se deben incluir:

**Introducción:**

1. Objetivo del manual: Una descripción general del propósito del manual de mantenimiento y a quién está destinado.
2. Descripción del software: Información sobre el software, incluyendo su propósito, funciones principales y tecnologías utilizadas.

**Arquitectura y diseño**:

1. Visión general de la arquitectura: Una descripción de alto nivel de la arquitectura del software, incluyendo los componentes y su interacción.
2. Diseño del software: Información detallada sobre el diseño del software, como los diagramas de clase, diagramas de secuencia u otros diagramas relevantes.

**Componentes y módulos:**

1. Descripción de los componentes: Una lista de los diferentes componentes o módulos del software, junto con su función y relación con otros componentes.
2. Dependencias y bibliotecas: Información sobre las bibliotecas externas o dependencias utilizadas por el software, incluyendo versiones y configuraciones relevantes.

**Procedimientos de mantenimiento:**

1. Identificación y seguimiento de problemas: Explicación de los procesos para identificar y registrar problemas o errores en el software, y cómo realizar un seguimiento de ellos.
2. Gestión de cambios: Procesos y directrices para administrar cambios en el software, incluyendo solicitudes de cambio, versionado y control de versiones.
3. Mantenimiento preventivo: Recomendaciones y mejores prácticas para prevenir problemas futuros, como limpieza de código, optimización y actualizaciones regulares.

**Proceso de corrección y actualización:**

1. Resolución de problemas: Pasos detallados para identificar, diagnosticar y corregir problemas o errores en el software.
2. Actualizaciones y parches: Instrucciones sobre cómo aplicar actualizaciones y parches al software, asegurando la compatibilidad y minimizando el impacto en la funcionalidad existente.

**Pruebas de mantenimiento:**

1. Estrategia de pruebas: Descripción de la estrategia y enfoque para realizar pruebas durante el mantenimiento del software.
2. Casos de prueba: Ejemplos de casos de prueba relevantes para verificar la corrección y el rendimiento después de los cambios o actualizaciones.

**Documentación relacionada**:

1. Documentos de diseño y especificaciones: Referencias a otros documentos relevantes, como documentos de diseño, especificaciones técnicas o manuales de usuario.

**Contacto y soporte:**

1. Información de contacto: Detalles de las personas o equipos responsables del mantenimiento del software, junto con información de contacto para obtener soporte técnico o hacer consultas.

**Manual de usuario**

El manual de usuario debe contener información relevante y útil para que los usuarios puedan comprender y utilizar adecuadamente un producto o sistema. A continuación se presentan los elementos que deberá de contener como mínimo el manual de usuario:

**Introducción:**

1. Descripción del producto: Una breve descripción del producto o sistema, incluyendo su propósito y funcionalidades principales.
2. Audiencia objetivo: Indicación del tipo de usuarios para los que está destinado el manual y el nivel de conocimientos previos que se espera.

**Instrucciones de inicio:**

* 1. Requisitos del sistema: Los requisitos de hardware, software y cualquier otra configuración necesaria para utilizar el producto.
  2. Instalación y configuración: Pasos detallados para instalar y configurar el producto, incluyendo la activación, si es necesario.

**Visión general del producto:**

1. Interfaz de usuario: Descripción de los elementos de la interfaz, como menús, botones, paneles y otras características visuales.
2. Navegación: Instrucciones sobre cómo moverse por el producto, cambiar entre pantallas, acceder a diferentes funciones y realizar acciones comunes.

**Instrucciones de uso:**

1. Funcionalidades principales: Descripción detallada de las principales funciones y características del producto, con instrucciones paso a paso sobre cómo utilizarlas.
2. Casos de uso: Ejemplos prácticos de cómo utilizar el producto en diferentes escenarios o situaciones.
3. Flujo de trabajo: Descripción de los procesos o flujos de trabajo comunes que se pueden realizar con el producto, junto con los pasos necesarios para completarlos.

**Solución de problemas y preguntas frecuentes:**

1. Mensajes de error: Explicación de los mensajes de error comunes y posibles soluciones.
2. Problemas comunes: Lista de problemas o situaciones que los usuarios pueden encontrar y cómo resolverlos.
3. Preguntas frecuentes: Respuestas a preguntas comunes que los usuarios pueden tener sobre el producto.

**Referencia rápida y glosario:**

1. Resumen de comandos o funciones clave: Una lista o tabla que enumera las funciones o comandos principales y su descripción breve.
2. Glosario de términos: Definiciones de términos técnicos o específicos utilizados en el manual o relacionados con el producto.

**Información adicional:**

1. Actualizaciones y versiones anteriores: Información sobre actualizaciones de software y cómo acceder a versiones anteriores si es relevante.
2. Soporte y contacto: Información de contacto para recibir soporte técnico o realizar consultas relacionadas con el producto.

# Estándares, prácticas, convenciones y métricas

La empresa “FemmeCode” utiliza múltiples estándares, prácticas y convenciones para asegurar la calidad de sus procesos. Aquí se listan los diferentes recursos utilizados junto con una breve descripción.

**Anexo “Plan de SQA”**

* **Documentación** 
  + ISO/IEC/IEEE 12207: Este estándar proporciona un marco para el ciclo de vida del software y establece los requisitos para la documentación necesaria en cada fase del ciclo de vida del software. Esto incluye documentación de requisitos, diseño, implementación, pruebas y mantenimiento. La información sobre la documentación se puede encontrar a partir de la página 44 cómo “6.1 Documentation process”
  + IEEE 829: Este es un estándar para la documentación de pruebas de software. Proporciona un marco para documentar diferentes tipos de pruebas de software, como pruebas de unidad, pruebas de integración y pruebas de sistema.

**Anexo “Recursos para la documentación”**

* **Requisitos:**
  + IEEE Std 830-1998: Este estándar, titulado "IEEE Recommended Practice for Software Requirements Specifications", proporciona directrices detalladas para la documentación de los requisitos de software. Describe la estructura y el contenido de una especificación de requisitos, incluyendo secciones como introducción, descripción general del sistema, requisitos específicos, entre otros.
  + ISO/IEC/IEEE 29148: Este estándar conjunto proporciona directrices para el proceso de ingeniería de requisitos. Define los conceptos, procesos y actividades relacionados con la gestión de requisitos, incluyendo la identificación, análisis, especificación y validación de los requisitos de software.

**Anexo “Recursos para los requisitos”**

* **Estructura lógica:**
  + ISO/IEC/IEEE 42010:2011: Este es un estándar internacional que establece los requisitos para la descripción de la arquitectura de software, incluyendo los modelos, vistas, componentes y relaciones.
  + "Clean Architecture: A Craftsman's Guide to Software Structure and Design" de Robert C. Martin: Este libro explora los principios y prácticas de la arquitectura de software limpia y cómo diseñar una estructura lógica efectiva para aplicaciones empresariales.

**Ver anexo “Recursos para la estructura lógica”**

* **Codificación:**

Para la codificación de la aplicación móvil, se utilizará React Native, framework que permite crear aplicaciones móviles multiplataforma utilizando JavaScript como lenguaje de programación principal. React Native utiliza componentes nativos de las plataformas móviles (Android e iOS) para ofrecer una experiencia de usuario más nativa. Es por eso que se necesitan los siguientes estándares:

* + Google JavaScript Style Guide: Establecido por Google, este estándar ofrece directrices claras y concisas para el desarrollo en JavaScript. Incluye recomendaciones sobre la organización del código, la indentación, los nombres de variables, los comentarios y las mejores prácticas de programación.

**Ver Anexo “Google JavaScript Style Guide”**

* + “Code Complete: A Practical Handbook of Software Construction” de Steve McConnell: Este libro aborda todos los aspectos del desarrollo de software, incluyendo la codificación, y ofrece prácticas recomendadas para crear un código de alta calidad.

**Anexo “Recursos para la codificación”**

* **Comentarios**
  + Google JavaScript Style Guide: La guía de estilo de Google para JavaScript incluye recomendaciones sobre cómo escribir y utilizar comentarios en JavaScript de manera clara y efectiva. Esta guía proporciona pautas claras sobre la estructura y el formato de los comentarios, así como su uso adecuado en diferentes contextos.

**Ver Anexo “Google JavaScript Style Guide”**

* **Pruebas**
  + ISO/IEC/IEEE 29119: Este estándar define un conjunto de estándares para el proceso de prueba de software. Proporciona directrices sobre la planificación de pruebas, diseño y ejecución de casos de prueba, gestión de incidentes, entre otros aspectos relacionados con las pruebas de software.
  + IEEE 829-2008: Este estándar describe la estructura y el contenido de los documentos de prueba. Proporciona pautas para la documentación de pruebas, incluyendo la descripción de casos de prueba, resultados de pruebas, procedimientos de prueba y otros aspectos relacionados.
  + IEEE 1008-1987: define un enfoque sistemático y documentado para las pruebas unitarias.

**Anexo “Recursos para las pruebas”**

* **Mantenimiento**
  + ISO/IEC 14764: Este estándar establece procesos y actividades para el mantenimiento de software. Define prácticas recomendadas para el mantenimiento correctivo, adaptativo, perfectivo y preventivo. También aborda aspectos como la documentación, la gestión de configuración y la gestión de la calidad en el mantenimiento de software.

**Anexo “Recursos para el mantenimiento”**

# Revisiones y auditorías

De acuerdo con el libro "Software Quality Assurance" de Daniel Galin, las revisiones y auditorías son actividades importantes dentro del plan de aseguramiento de la calidad de software y tienen los siguientes objetivos:

1. Mejorar la calidad del software: Las revisiones y auditorías permiten detectar errores y defectos en el software y corregirlos antes de que el software sea entregado al cliente. De esta manera, se garantiza que el software cumpla con los estándares de calidad requeridos y que funcione correctamente.
2. Ahorrar tiempo y reducir costos: Las revisiones y auditorías permiten detectar y corregir errores temprano en el proceso de desarrollo, lo que reduce los costos y el tiempo de desarrollo.
3. Identificar problemas de diseño: Las revisiones y auditorías permiten identificar problemas de diseño y arquitectura del software, lo que permite corregirlos antes de que sean codificados. De esta manera, se evita la introducción de errores en el código y se mejora la calidad del software.
4. Mejorar la productividad: Las revisiones y auditorías pueden mejorar la productividad de los desarrolladores, ya que les permite aprender de sus errores y mejorar sus habilidades. Además, permite compartir el conocimiento y la experiencia entre los miembros del equipo.
5. Cumplimiento con los estándares y regulaciones: Las auditorías permiten garantizar que el software cumpla con los estándares y regulaciones requeridos para su uso en determinados sectores, como la banca, la salud y el gobierno.

A continuación, se mencionarán las fases del ciclo de desarrollo de software para especificar de qué manera se llevarán a cabo las revisiones y/o auditorías en cada una.

Fase de requisitos

Revisión por pares: Documento de especificación de requerimientos.

Dicho documento será sometido a una revisión donde el equipo encargado de la revisión examinará el trabajo y proporcionará comentarios y sugerencias para la mejora de este.

Las revisiones se llevarán a cabo con base en el estándar IEEE 1028-2008 "Estándar para la Revisión de Software" (**Ver Anexo “Revisiones IEEE 1028**”) ya que proporciona lineamientos para la planificación, ejecución y gestión de revisiones y auditorías de software.

Los objetivos de la revisión de requerimientos serán los siguientes:

Evaluar la calidad de los requerimientos: La revisión de requerimientos tiene como objetivo asegurar que los requerimientos sean claros, completos, consistentes, verificables y que cumplan con las necesidades del usuario y los objetivos del proyecto.

Identificar problemas de diseño temprano: La revisión de requerimientos permite detectar problemas de diseño temprano en el ciclo de vida del software, lo que puede ahorrar tiempo y costos en el futuro.

Mejorar la comunicación: La revisión de requerimientos promueve una mejor comunicación entre los miembros del equipo y con los interesados en el proyecto.

Identificar posibles riesgos: La revisión de requerimientos ayuda a identificar posibles riesgos y problemas en el proyecto antes de que se inicien las actividades de diseño y desarrollo.

Mejorar la calidad del software: La revisión de requerimientos ayuda a garantizar que el software cumpla con los requisitos del usuario y los objetivos del proyecto.

**Auditoría:** Documento de especificación de requerimientos.

Las auditorías, son revisiones formales que se realizan después de que el software ha sido completado. En este caso, una vez terminado el documento de requerimientos se someterá a una auditoría.

El objetivo de la auditoría de requisitos es asegurarse de que los requisitos establecidos son completos, consistentes, correctos, verificables y medibles.

La auditoría de requisitos también puede identificar requisitos faltantes o contradictorios, y puede ayudar a establecer la base para la planificación de las revisiones posteriores del diseño y el código.

El estándar IEEE 1028 establece que los miembros del equipo de auditoría de requisitos deben tener experiencia en la revisión de requisitos y deben ser independientes del equipo de desarrollo de software. Durante la auditoría de requisitos, se debe revisar la documentación de requisitos, incluyendo los documentos de especificación de requisitos y otros artefactos relacionados, para asegurarse de que cumplen con los criterios de auditoría establecidos. También establece que la auditoría de requisitos debe incluir la revisión de requisitos funcionales y no funcionales, así como de los requisitos de rendimiento, seguridad, usabilidad y otros requisitos específicos del dominio de la aplicación. Además, se debe revisar si los requisitos cumplen con los estándares, políticas y prácticas de la organización, así como con las regulaciones y normas aplicables.

Fase de diseño

Auditoría: Documento de análisis y diseño

Dicho documento se someterá a la auditoría de revisión formal que tendrá como objetivos:

Identificar y corregir errores: el principal objetivo de la auditoría de diseño es identificar y corregir errores y defectos en el diseño del software antes de que se comience la implementación. Esto ayuda a prevenir costosos errores y problemas en el software final.

Mejorar la calidad del software: al identificar y corregir errores en la fase de diseño, se mejora la calidad del software y se reduce la probabilidad de errores en el software final.

Validar los requisitos: la revisión de diseño también puede ayudar a validar que el diseño cumple con los requisitos del software. Esto asegura que el software final cumpla con los requisitos del usuario.

Identificar riesgos: la revisión de diseño también puede ayudar a identificar riesgos potenciales en el diseño del software, como problemas de rendimiento o problemas de seguridad.

Fase de construcción

Todo lo relacionado con la fase de construcción del código, incluyendo las revisiones se encuentra descrito detalladamente en nuestro modelo de procesos en la sección -Actividades- en donde menciona que el código será sometido a una revisión por pares cada que se muestren avances del proyecto.

**Anexo “Modelo de Procesos”**

Posteriormente a las revisiones, el código se someterá a una auditoría la cuál tendrá como objetivo mejorar la eficiencia del proceso de desarrollo de software, ya que pueden ayudar a detectar errores temprano en el ciclo de vida del software y, por tanto, reducir el tiempo y el costo de corrección de errores más adelante en el ciclo de vida. Durante esta revisión, se busca verificar si el código cumple con los estándares de codificación, se encuentra bien documentado, es legible y entendible, y se ajusta a los requerimientos especificados.

Fase de pruebas

Revisión por pares: Plan de pruebas

El objetivo principal de las revisiones de las pruebas es garantizar que la documentación de prueba esté completa, correcta y clara, y que los casos de prueba cubran los requerimientos y objetivos de prueba establecidos.

Durante las revisiones de las pruebas, los revisores examinan la documentación de prueba, incluyendo los casos de prueba y los informes de resultados de prueba, para identificar errores, inconsistencias y omisiones. Los revisores también pueden verificar que los casos de prueba sean adecuados y efectivos para cumplir con los objetivos de prueba establecidos. Las revisiones de las pruebas también pueden ayudar a mejorar la eficiencia del proceso de pruebas al permitir la identificación temprana de problemas en la documentación de prueba, lo que reduce el tiempo y el costo de realizar correcciones en las pruebas más adelante en el ciclo de vida del software.

Fase de mantenimiento:

Revisión por pares: Manual de mantenimiento

El objetivo principal de las revisiones de mantenimiento es garantizar que los cambios realizados en el software durante la fase de mantenimiento cumplan con los objetivos de mantenimiento y los requisitos del usuario, y que se hayan implementado de manera efectiva y sin errores.

Durante las revisiones de mantenimiento, los revisores examinan los cambios realizados en el software, la documentación de mantenimiento, los registros de errores y cualquier otra documentación relevante para evaluar si se han abordado adecuadamente los problemas de mantenimiento y si se han cumplido los objetivos establecidos. Las revisiones de mantenimiento también pueden ayudar a mejorar la eficiencia del proceso de mantenimiento al permitir la identificación temprana de problemas en los cambios realizados, lo que reduce el tiempo y el costo de realizar correcciones en las fases posteriores del ciclo de vida del software.

# Pruebas

Durante el desarrollo de la aplicación móvil, se realizan diferentes tipos de pruebas dinámicas para garantizar su calidad y funcionamiento correcto. Estas serán lideradas por el Especialista en pruebas

**Pruebas unitarias:**

Planificación:

1. Se identifican las unidades de código que serán sometidas a pruebas unitarias.
2. Se establecen los objetivos específicos de las pruebas unitarias, como verificar la funcionalidad de las unidades o validar su comportamiento ante situaciones límite.
3. Se diseñan los casos de prueba que cubren diferentes escenarios y condiciones para cada unidad.
4. Se asignan los recursos necesarios para llevar a cabo las pruebas unitarias, como personal, herramientas y entornos de pruebas.

Ejecución:

1. Se realizan en un entorno de pruebas adecuado para ejecutar las pruebas unitarias sin que comprometa al sistema.
2. Se ejecutan los casos de prueba diseñados anteriormente para cada unidad de código.
3. Se registran los resultados de cada prueba, incluyendo los resultados esperados y los obtenidos, así como cualquier anomalía o error encontrado durante la ejecución.

Evaluación de resultados:

1. Se analizan los resultados de las pruebas unitarias para determinar si las unidades de código cumplen con los criterios de aceptación.
2. Se identifican los defectos, errores o anomalías encontrados durante las pruebas y se registran
3. Se genera un informe que resume los resultados de las pruebas unitarias, incluyendo los casos de prueba ejecutados, los resultados obtenidos y los defectos identificados.
4. Se toman decisiones basadas en los resultados de las pruebas unitarias, como la corrección de defectos encontrados, la revisión del diseño de las unidades de código o la validación de la funcionalidad implementada.

**Pruebas de integración:**

Planificación:

1. Identificación de componentes: Se identifican los componentes de software que serán integrados y probados en conjunto.
2. Se establecen los objetivos específicos de las pruebas de integración, como verificar la interoperabilidad de los componentes y detectar posibles problemas de interfaz.
3. Se diseñan los casos de prueba que cubren diferentes combinaciones y escenarios de integración entre los componentes.

Ejecución:

1. Se establece el entorno de pruebas adecuado para ejecutar las pruebas de integración, incluyendo la configuración y conexión de los componentes.
2. Se ejecutan los casos de prueba diseñados anteriormente para verificar la integración entre los componentes.
3. Se registran los resultados de cada prueba, incluyendo los resultados esperados y los obtenidos, así como cualquier problema o anomalía encontrado durante la ejecución.

Evaluación de resultados:

1. Se analizan los resultados de las pruebas de integración para determinar si los componentes se integran de manera adecuada y cumplen con los criterios de aceptación establecidos.
2. Se identifican y documentan los problemas de integración, como problemas de comunicación, incompatibilidades o fallos en la interacción entre los componentes.
3. Se genera un informe que resume los resultados de las pruebas de integración, incluyendo los casos de prueba ejecutados, los resultados obtenidos y los problemas de integración identificados.
4. Se toman decisiones basadas en los resultados de las pruebas de integración.

**Pruebas de seguridad**

Planificación:

1. Se identifican los requisitos de seguridad relevantes para el sistema o la aplicación en desarrollo.
2. Se realiza un análisis de riesgos para identificar las posibles amenazas y vulnerabilidades que podrían afectar la seguridad del software.
3. Se establecen los objetivos específicos de las pruebas de seguridad, como la identificación de vulnerabilidades, la evaluación de la resistencia a ataques o la verificación del cumplimiento de estándares de seguridad.
4. Se diseñan escenarios de prueba que simulan ataques o situaciones de seguridad específicas para evaluar la robustez del software.

Ejecución:

1. Se ejecutan los escenarios de prueba diseñados para evaluar la seguridad del software.
2. Se buscan y se identifican posibles vulnerabilidades, brechas de seguridad o debilidades en el sistema.
3. En algunos casos, se pueden realizar simulaciones de ataques reales o intentos de explotación para evaluar la resistencia del software ante amenazas conocidas.
4. Se recopila evidencia detallada de las pruebas realizadas, incluyendo capturas de pantalla, registros de eventos y cualquier información relevante relacionada con la seguridad.

Evaluación de resultados:

1. Se analizan los resultados de las pruebas de seguridad para evaluar el nivel de seguridad del software y determinar si se cumplen los requisitos y estándares de seguridad establecidos.
2. Se priorizan las vulnerabilidades identificadas según su gravedad, impacto potencial y riesgo para el sistema.
3. Se genera un informe que resume los resultados de las pruebas de seguridad, incluyendo las vulnerabilidades encontradas, sus implicaciones y recomendaciones para mitigar los riesgos de seguridad.
4. Se toman decisiones basadas en los resultados de las pruebas de seguridad, como la implementación de medidas de mitigación, corrección de vulnerabilidades o ajustes en la arquitectura del software para mejorar la seguridad.

**Pruebas de aceptación:**

Planificación:

1. Se identifican los requisitos funcionales y no funcionales que deben cumplirse para que el software sea aceptado por el cliente o usuario final.
2. Se diseñan los casos de prueba que reflejan los escenarios de uso reales del software y que verifican si los requisitos de aceptación se cumplen.
3. Se establece el entorno necesario para realizar las pruebas de aceptación, incluyendo configuraciones de hardware, software y datos relevantes.

Ejecución:

1. Se ejecutan los casos de prueba diseñados previamente, proporcionando los datos de entrada y registrando las salidas y resultados obtenidos.
2. Se verifica si el software cumple con los requisitos de aceptación establecidos y si se comporta según lo esperado por el cliente o usuario final.
3. Si se encuentran desviaciones o defectos durante las pruebas, se registran en un sistema de seguimiento de problemas para su posterior corrección.

Evaluación de resultados:

1. Se analizan los resultados de las pruebas de aceptación para determinar si el software cumple con los requisitos acordados y si satisface las expectativas del cliente o usuario final.
2. Se verifica que todas las funciones y características importantes del software estén implementadas y funcionen correctamente.
3. Se valida si el software cumple con los estándares y criterios establecidos en el proceso de desarrollo y en los requisitos de aceptación.
4. Se genera un informe que resume los resultados de las pruebas de aceptación, incluyendo los casos de prueba ejecutados, los resultados obtenidos, los defectos identificados y las observaciones relevantes.

Es importante mencionar que, durante la realización de cada fase de las diferentes pruebas, se deberán de llenar las plantillas correspondientes:

* Plan de prueba de software
* Descripción de la prueba de software
* Reporte de la prueba de software

**Anexo “Plantillas de pruebas”**

La descripción de las responsabilidades administrativas y organizacionales se encuentra en el apartado [“Gestión”](#_kboqlz22xiii) en donde se describe los roles de la unidad SQA, entre ellos, el Especialista de pruebas

La descripción de las herramientas de ingeniería de software asistida por computadora (CASE) se encuentra en el apartado [“Herramientas, técnicas y metodologías”](#_rfblm6q8bbnp) de este mismo documento

El método que utilizará el elemento organizativo SQA para verificar y validar los planes de prueba serán las revisiones por pares, en la que se realizará una revisión exhaustiva del plan de prueba por parte de personas con experiencia en pruebas de software. Durante esta revisión, se buscará identificar posibles deficiencias, errores o inconsistencias en el plan de prueba, así como asegurarse de que se cumplen los estándares y lineamientos establecidos.

# Informe de problemas y acciones correctivas de SQA

Acciones preventivas:

En el modelo CAPA(Corrective and preventive actions), las acciones preventivas se refieren a las medidas tomadas para evitar la aparición de problemas o no conformidades en el futuro. Estas acciones se centran en identificar y abordar las causas subyacentes que podrían llevar a problemas en el proceso de desarrollo de software o en el producto final.

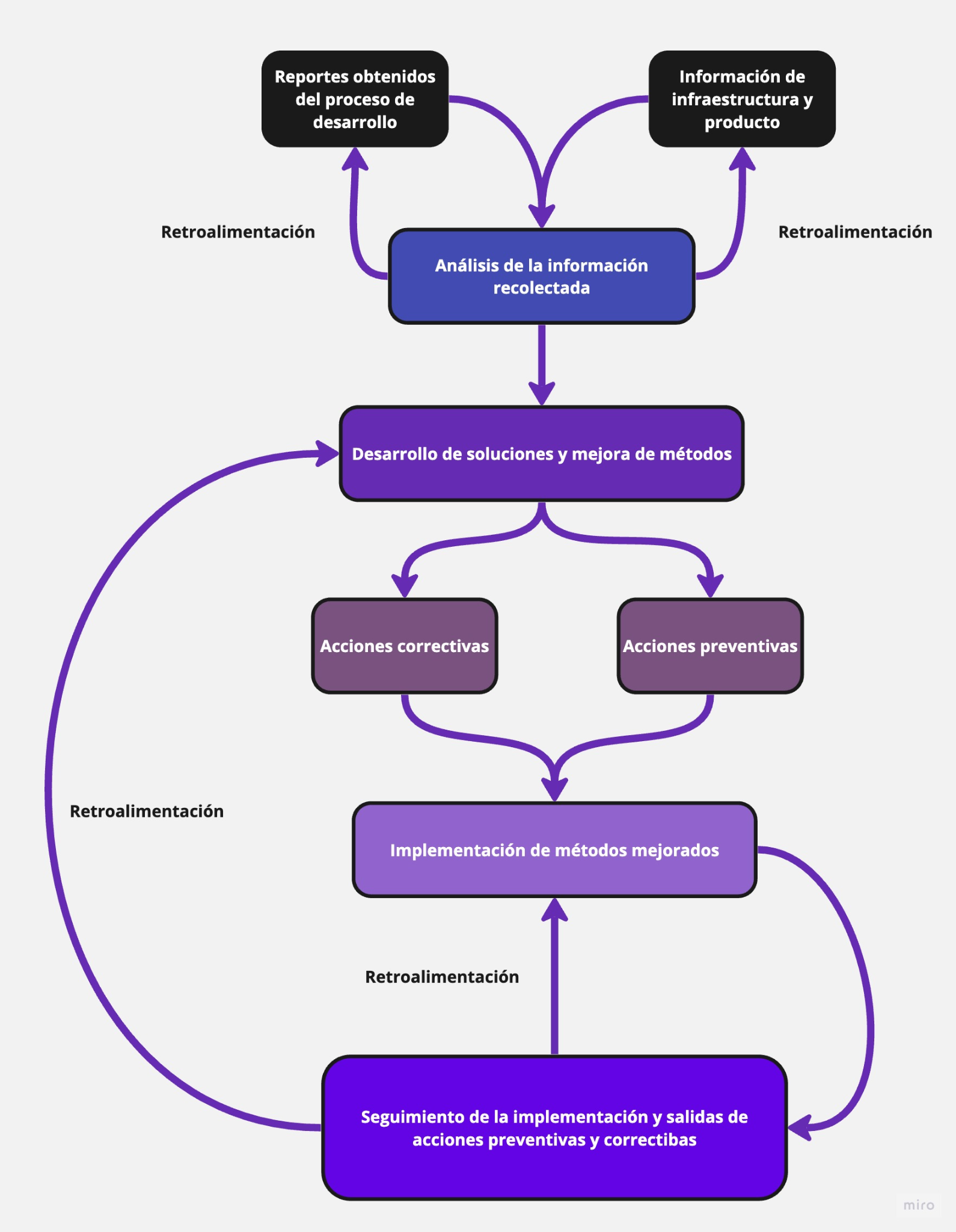
Acciones correctivas:

En el modelo CAPA (Corrective and preventive actions), las acciones correctivas se refieren a las medidas tomadas para abordar y corregir un problema o no conformidad identificados en el proceso de desarrollo de software o en el producto de software. Estas acciones se centran en eliminar las causas raíz del problema y evitar su recurrencia en el futuro.

En la **imagen 7.0** se observa el modelo a seguir de la empresa FemmeCode para la implementación de sus acciones preventivas y correctivas (basado en CAPA).

La información anterior de acuerdo con en el libro Software Quality Assurance.

**Anexo “Software Quality Assurance”**

****

*Imagen 7.0 Modelo de acciones preventivas y correctivas FemmeCode*

Reportes

El reporte de acciones preventivas y correctivas debe ser documentado y registrado para asegurar que se hayan implementado y evaluado efectivamente las acciones necesarias para prevenir o corregir problemas o no conformidades en el sistema de gestión de calidad de la organización.

La norma ISO 9001:2015 establece los requisitos para implementar y mantener un sistema de gestión de calidad en una organización. También establece requisitos específicos para la implementación de acciones correctivas y preventivas y los pasos a seguir para realizar un reporte de lo anterior.

Este reporte debe incluir una descripción detallada del problema o la no conformidad, la causa raíz identificada, las acciones tomadas para prevenir o corregir el problema y la evaluación de la efectividad de las acciones.

**Anexo “Norma Acciones preventivas y correctivas”**

Los miembros de la empresa elaborarán estos reportes siguiendo una plantilla elaborada con base en el estándar previamente descrito para registrar aquellos posibles errores encontrados en cualquiera de las fases del ciclo de vida del software, que estén relacionados a las acciones preventivas o correctivas implementadas.

**Anexo “Plantilla reporte”**

# Herramientas, técnicas y metodologías

## Herramientas

FemmeCode hace uso de una gran variedad de herramientas durante el ciclo de vida del proyecto. Estas herramientas son los IDEs utilizados, las herramientas de detección de errores, para la implementación de pruebas, entre otras actividades. A continuación se explican las herramientas que se utilizan junto con sus funcionalidades.

* Figma: Figma es una herramienta de diseño de interfaces de usuario (UI) y experiencia de usuario (UX) basada en la nube. Permite a los diseñadores crear, colaborar y compartir diseños de aplicaciones móviles y web de manera eficiente. Figma ha ganado popularidad debido a su enfoque en la colaboración en tiempo real y su capacidad para funcionar en diferentes sistemas operativos, lo que facilita el trabajo en equipo y la comunicación fluida entre diseñadores, desarrolladores y otros stakeholders del proyecto.
* Visual Studio: Es un IDE de Microsoft que ofrece soporte para el desarrollo multiplataforma de aplicaciones móviles. Es un IDE liviano y altamente personalizable que ofrece soporte para JavaScript, TypeScript y React Native. Proporciona características útiles como resaltado de sintaxis, finalización de código, depuración y control de versiones. Visual Studio ofrece soporte para múltiples lenguajes de programación y plataformas, lo que lo convierte en una opción versátil para el desarrollo de software. *(Anexo “Documentación Visual Studio”)*
* Git: Es un sistema de control de versiones distribuido ampliamente utilizado en el desarrollo de software. Sirve para gestionar y controlar los cambios en el código fuente y otros archivos de un proyecto a lo largo del tiempo. *(Anexo “Documentación Git”)*
* React Native: React Native es un framework de desarrollo de aplicaciones móviles creado por Facebook. Se basa en React, una biblioteca de JavaScript para construir interfaces de usuario. La principal ventaja de React Native es la capacidad de crear aplicaciones móviles multiplataforma, lo que significa que el mismo código puede ejecutarse tanto en dispositivos iOS como en dispositivos Android. *(Anexo “Documentación React Native”)*
* Jest: Es un marco de pruebas unitarias ampliamente utilizado en el desarrollo de software. Está diseñado principalmente para pruebas en JavaScript, pero también se puede utilizar con otros lenguajes de programación. Jest facilita la escritura, ejecución y organización de pruebas unitarias al proporcionar una amplia gama de funciones y herramientas. *(Anexo: “Documentación Jest”)*
* Appium: Appium es un marco de pruebas de integración de software diseñado específicamente para aplicaciones móviles. Su principal objetivo es facilitar la automatización de pruebas en dispositivos móviles y emuladores, lo que permite probar la funcionalidad de una aplicación en un entorno integrado. También se puede utilizar para realizar pruebas de interfaz de usuario, ya que admite la interacción con la interfaz de usuario de aplicaciones móviles en diferentes plataformas. *(Anexo: “Documentación Appium”)*
* Apache JMeter: Una herramienta de código abierto para realizar pruebas de carga y rendimiento de aplicaciones móviles. Es utilizada para realizar pruebas de rendimiento y carga en aplicaciones web y servicios de red. Está diseñada para simular el comportamiento de múltiples usuarios y medir el rendimiento y la capacidad de respuesta de un sistema en diferentes condiciones de carga. *(Anexo: “Documentación JMeter”)*
* BrowserStack: Una plataforma en la nube que permite realizar pruebas de compatibilidad en diferentes dispositivos y navegadores, incluyendo pruebas en aplicaciones móviles. *(Anexo: “Documentación BrowserStack”)*
* OWASP ZAP: Una herramienta de pruebas de seguridad de aplicaciones web de código abierto que también puede utilizarse para pruebas de seguridad en aplicaciones móviles. *(Anexo: “Documentación OWASP ZAP”)*
* UserTesting: Una plataforma que permite realizar pruebas de usabilidad con usuarios reales y obtener retroalimentación directa sobre la experiencia del usuario. *(Anexo “Documentación UserTesting”)*
* MongoDB: Es una base de datos NoSQL (No Structured Query Language) que se caracteriza por ser orientada a documentos. En lugar de utilizar tablas y filas como en las bases de datos relacionales, MongoDB almacena los datos en documentos JSON flexibles con una estructura similar a los objetos en la programación. *(Anexo “Documentación MongoDB”)*
* Node.js: Node.js es un entorno de ejecución de código JavaScript basado en el motor V8 de Google Chrome. A diferencia del uso tradicional de JavaScript en el navegador web, Node.js permite ejecutar código JavaScript en el servidor. *(Anexo “Documentación Node.js”)*
* Express.js: Express.js es un framework minimalista y flexible para el desarrollo de aplicaciones web en Node.js. Proporciona una capa de abstracción sobre las funcionalidades básicas de Node.js, lo que facilita la creación de servidores web y el desarrollo de aplicaciones web y API RESTful.
* Google Drive: Es un servicio de almacenamiento en la nube proporcionado por Google. Permite a los usuarios almacenar, sincronizar y compartir archivos y carpetas en línea. Con Google Drive, los usuarios pueden acceder a sus archivos desde cualquier dispositivo con conexión a Internet.
* Windows: Windows ofrece un entorno de trabajo intuitivo y fácil de usar, con iconos, ventanas y menús que permiten la ejecución de aplicaciones, la gestión de archivos y la navegación por internet, entre otras funciones.

## Técnicas

En cuanto a las técnicas que se emplean por parte de la empresa FemmeCode se consideran diferentes factores que se presentan posteriormente.

* Revisión del uso de estándares: Se establecerá un proceso para revisar regularmente el cumplimiento de los estándares de codificación, diseño y documentación establecidos en el proyecto. Se designan responsables para realizar revisiones periódicas y se establecerán criterios de revisión claros para evaluar la conformidad.
* Inspecciones de software: Se llevarán a cabo inspecciones formales del código fuente y de la documentación del software. Se establecerán reuniones periódicas donde el equipo de desarrollo revisará y discutirá el código y la documentación para identificar posibles errores, deficiencias o violaciones de estándares.
* Seguimiento de requisitos: Se implementará un sistema para rastrear los requisitos del software desde su definición inicial hasta su implementación y verificación. Se utilizará una herramienta de seguimiento de requisitos para registrar los cambios, asignar responsabilidades y garantizar que los requisitos se cumplan correctamente.
* Verificación de requisitos y diseño: Se realizarán revisiones técnicas formales para verificar que los requisitos y el diseño del software se hayan implementado correctamente. Se evaluará la coherencia, completitud y consistencia de los requisitos y se verificará que el diseño cumpla con los requisitos establecidos.
* Mediciones y evaluaciones de confiabilidad: Se recopilarán datos y métricas relacionadas con la confiabilidad del software, como el tiempo promedio entre fallas, el porcentaje de fallas corregidas, etc. Estas mediciones se utilizarán para evaluar la confiabilidad del software y tomar medidas correctivas si es necesario.

## Metodologías

Principalmente se hace uso de dos metodologías, la primera es del desarrollo en cascada que será utilizada durante el ciclo de vida del proyecto en general, mientras que conjuntamente igual se usará la metodología de desarrollo basado en prototipos.

Cuando se combinan la metodología de desarrollo en cascada y la metodología de desarrollo basado en prototipos, se logra una colaboración efectiva que beneficia el proceso de desarrollo de software. En esta combinación conjunta, ambas metodologías se complementan y se refuerzan mutuamente.

Durante la fase de análisis, la metodología en cascada se encarga de recopilar de manera estructurada los requisitos iniciales del proyecto. Por otro lado, la metodología de desarrollo basado en prototipos permite crear prototipos básicos que representan las funcionalidades clave, obteniendo así una validación temprana de los requisitos y retroalimentación de los usuarios.

En la fase de diseño, la metodología en cascada desarrolla un diseño preliminar del sistema, mientras que la metodología de desarrollo basado en prototipos va más allá al crear prototipos interactivos que reflejan las interfaces y la experiencia de usuario. Estos prototipos se presentan a los usuarios para obtener retroalimentación y mejorar el diseño.

Durante la fase de desarrollo, la metodología en cascada guía el desarrollo secuencial del software, siguiendo las etapas de implementación, pruebas y despliegue. Al mismo tiempo, la metodología de desarrollo basado en prototipos se aplica mediante iteraciones adicionales de prototipado, donde se desarrollan prototipos más avanzados y completos para validar y refinar funcionalidades específicas antes de su implementación definitiva.

En la fase de pruebas y retroalimentación, la metodología en cascada realiza pruebas exhaustivas del software, tanto a nivel unitario como de integración. La metodología de desarrollo basado en prototipos continúa aportando valor al permitir obtener retroalimentación adicional durante las pruebas a partir de los prototipos más avanzados. Esta retroalimentación se utiliza para realizar ajustes y mejoras en el software.

Finalmente, en la fase mantenimiento, la metodología en cascada se encarga de implementar el software en el entorno de producción y llevar a cabo su mantenimiento continuo. Si se requieren cambios o mejoras adicionales en el producto final, la metodología de desarrollo basado en prototipos sigue siendo útil, ya que permite realizar iteraciones de prototipado en etapas posteriores.

# Control de código

Para el control del código, la empresa FemmeCode hace uso de diferentes herramientas y prácticas para asegurar la calidad en la creación de una aplicación móvil. Los siguientes puntos describen cómo se lleva a cabo el control de código por parte de la empresa.

* Estándares de codificación.

Como se han establecido estándares de codificación por parte de la empresa, es importante verificar que en la fase de desarrollo del código estos se hayan seguido de manera adecuada y no se estén violando los estándares. Para verificar lo anterior se establecen revisiones periódicas del código por parte de otros miembros del equipo. Durante estas revisiones, los revisores pueden verificar el cumplimiento de los estándares de codificación establecidos y brindar retroalimentación sobre posibles mejoras. Esto ayuda a garantizar que el código cumpla con las pautas y sea coherente en todo el proyecto.

* Uso de Git para el control de versiones.

Git es una herramienta de control de versiones de código ampliamente utilizada que proporciona varios beneficios para el control y gestión de versiones en un proyecto de desarrollo de software. A continuación, se detallan algunas formas en las que Git es de utilidad para el control de versiones de código:

* + Historial de cambios: Con Git se registran todos los cambios realizados en el código a lo largo del tiempo, lo que permite rastrear y revisar el historial de modificaciones. Esto proporciona transparencia y trazabilidad de los cambios realizados, lo que facilita la colaboración entre los miembros del equipo de desarrollo de FemmeCode y la identificación de la causa de problemas o errores.
  + Ramificación y fusión: Se crean ramas independientes del código principal, lo que permite trabajar en paralelo en diferentes características o solucionar problemas sin afectar la rama principal. Esto facilita el desarrollo de nuevas funcionalidades sin interrupciones y la gestión de correcciones de errores de manera aislada. Posteriormente, es posible fusionar los cambios de una rama a otra, lo que facilita la integración de las modificaciones y la creación de nuevas versiones del software.
  + Etiquetado y versionado: Con Git se hace el etiquetado de versiones específicas del código, lo que facilita el seguimiento e identificación de versiones de lanzamiento o hitos importantes en el proyecto. Estas etiquetas pueden ayudar a mantener un registro claro de las diferentes versiones del software y permiten un control preciso sobre qué versión se está utilizando o desplegando en un entorno determinado.

Para el etiquetado y versionado del código se siguen los siguientes pasos:

1. La empresa FemmeCode sigue el esquema de versionado semántico, que son 3 número separador por puntos de la forma “A.B.C”, donde la A representa los cambios mayores, B nuevas funcionalidades que se hayan integrado y C es para las correcciones de errores.
2. Se crea un nuevo repositorio de Git para el nuevo proyecto, donde se les da acceso a los miembros del equipo.
3. Cuando se tenga la primera versión estable del software se crea una etiqueta para que esta sea la versión inicial, por lo que se pone el número de versión que en este caso la etiqueta sería “V1.0.0”.
4. Conforme se avance en el proyecto y se vayan agregando nuevos cambios y mejoras, se deben seguir etiquetando las versiones conforme se ha establecido en el punto número uno.
5. Se pueden administrar las etiquetas, se renombran, eliminan o mueven dentro del repositorio. Para conocer más información sobre en control de versiones y etiquetado se revisa la documentación de Git, que se encuentra en el anexo “Documentación Git”.

# Control de multimedia

El control de multimedia para el proyecto de software implica la gestión de los elementos multimedia utilizados para la creación de la aplicación, como lo son las imágenes, videos, sonidos, iconos, gráficos, entre otros recursos visuales y auditivos. Por lo mismo se hace uso de Google Drive, donde se almacenarán estos recursos, como se describe a continuación.

1. Primero se crea una carpeta donde se encontrarán todos los recursos dentro una división de carpetas para mantener una organización.
2. Antes de guardar algún elemento a las carpetas se deben tener establecidos los requisitos de calidad de multimedia. Esto se refiere a incluir aspectos como resolución, calidad visual, tamaño de archivo, formato de archivo compatible, tiempo de carga, adecuación al contexto de uso, entre otros.
3. Se crean las carpetas para organizar los elementos dentro la carpeta principal del proyecto con los siguientes nombres:

* Borrador-Prototipo: Se tendrán los prototipos de las interfaces gráficas como borrador que servirá de base para hacer el prototipado posteriormente.
* Prototipo: Se incluyen las imágenes del prototipo realizado con Figma.
* Iconos: En esta carpeta se guardan todos los iconos utilizados para el diseño de la interfaz gráfica.
* Imágenes: Imágenes utilizadas en la interfaz gráfica.
* Sonidos: Recursos de sonido con lo que contará la aplicación.
* Presentaciones: En caso de crear presentaciones para presentar avances del proyecto, se encontrarán dentro de esta carpeta.

*Imagen 8.0 Organización de carpetas en Google Drive*

# Control de proveedores

*No aplica*, ya que solo se hará uso exclusivamente de software libre, por lo que no será necesario gestionar y evaluar a proveedores externos.

# Capacitación continua

La capacitación del personal involucrado en las actividades de aseguramiento de la calidad del software es importante para asegurar la calidad de los productos de software entregados. Además, ayuda a garantizar que el personal tenga las habilidades y conocimientos necesarios para realizar sus tareas de manera efectiva y eficiente, y para asegurar que se sigan las mejores prácticas de la industria.

La capacitación continua también puede mejorar la motivación del personal, ya que les permite desarrollar habilidades y conocimientos adicionales que pueden ser útiles en su carrera. Cabe destacar que puede ser una forma de demostrar la competencia del personal ante los clientes y otros interesados, lo que puede mejorar la confianza en el equipo de aseguramiento de calidad del software y, en última instancia, en la calidad del software entregado.

La norma **ISO/IEC 12207:1997** establece la necesidad de un plan de capacitación continua para el personal involucrado en el desarrollo de software y aseguramiento de calidad. Este plan de capacitación tiene como objetivo garantizar que el personal tenga las habilidades y conocimientos necesarios para cumplir con sus responsabilidades de manera efectiva.

El plan de capacitación continua debe incluir los siguientes elementos:

* Identificación de necesidades de capacitación: se debe realizar una evaluación de las habilidades y conocimientos actuales del personal, para identificar las áreas que necesitan mejora.
* Objetivos de capacitación: una vez identificadas las áreas de mejora, se deben establecer objetivos de capacitación claros y medibles.
* Diseño de la capacitación: se deben seleccionar y diseñar cursos, talleres o sesiones de capacitación que sean relevantes para los objetivos establecidos.
* Implementación de la capacitación: se debe asegurar que el personal asista a las sesiones de capacitación y que el contenido sea entregado de manera efectiva.
* Evaluación de la capacitación: después de cada sesión de capacitación, se debe evaluar el impacto en el desempeño del personal y en la mejora de las habilidades y conocimientos identificados en el paso 1.
* Actualización del plan de capacitación: en función de los resultados de la evaluación, se deben actualizar los objetivos y el diseño de la capacitación.

**Ver anexo “Norma ISO/IEC 12207:1997”**

# Gestión de riesgos

El estándar ISO/IEC 12207:2008 proporciona una guía para la gestión de riesgos en el desarrollo y mantenimiento de software.

Para establecer un proceso de gestión de riesgos se deben de incluir los siguientes elementos:

1. Se deben identificar los riesgos que podrían afectar la calidad del software, para lo cual se pueden utilizar técnicas como el análisis de riesgos, la revisión de la literatura, la consulta con expertos, entre otras.
2. Se debe realizar un análisis de cada uno de los riesgos identificados, para determinar su probabilidad de ocurrencia y su impacto en el proyecto de software.
3. Se deben evaluar los riesgos identificados y analizados para determinar su importancia relativa y establecer prioridades para su tratamiento.
4. Se deben definir y aplicar estrategias para tratar los riesgos identificados, entre las que se incluyen la aceptación del riesgo, la reducción del riesgo, la transferencia del riesgo o la evitación del riesgo.
5. Se deben establecer mecanismos para monitorear y controlar los riesgos identificados y los tratamientos aplicados, y se deben tomar medidas adicionales si es necesario.

Una vez establecido el proceso de gestión de riesgos, se deben llevar a cabo las siguientes actividades:

1. **Identificar los riesgos.**

Específicos del proyecto de software, considerando las características del producto, el entorno, las restricciones del proyecto y los intereses de las partes interesadas.

1. **Evaluar y analizar los riesgos identificados.**

Considerando su impacto potencial en el proyecto y su probabilidad de ocurrencia.

1. **Seleccionar las estrategias de tratamiento de riesgos.**

Lo anterior considerando el nivel de riesgo aceptable y las limitaciones del proyecto.

1. **Definir planes de contingencia para los riesgos críticos.**
2. **Monitorear y controlar continuamente los riesgos.**

Lo anterior mediante la revisión periódica de los planes de gestión de riesgos y la implementación de medidas adicionales si es necesario.

Es importante destacar que la gestión de riesgos es un proceso continuo y dinámico, que debe ser revisado y actualizado regularmente durante todo el ciclo de vida del software.

**Ver anexo “Norma ISO/IEC 12207:1997”**

# ANEXOS

**“Plan de SQA”**

Link:

<https://github.com/ver0rivera/Plan-SQA>

**“Software Quality Assurance”**

Link:

<https://github.com/ver0rivera/Plan-SQA/blob/main/Est%C3%A1ndares/Software%20Quality%20Assurance%20From%20Theory%20to%20Implementation.pdf>

**“Modelo de Procesos”**

Link:

<https://github.com/ver0rivera/ProcessModel-SQA/blob/main/SCI-01_versiones/Pendientes/SCI_01_V3.0.docx.pdf>

**Norma acciones preventivas y correctivas**

Link:

<https://github.com/ver0rivera/Plan-SQA/blob/main/Est%C3%A1ndares/CAPA/ISO_IEC%20-%20ISO%209001_2015%20Quality%20management%20systems%20%E2%80%94%20Requirements-ISO%20copyright%20office%20(2015)%20(1).pdf>

**Anexo “Plantilla reporte”**

Link:

<https://github.com/ver0rivera/Plan-SQA/blob/main/Plantillas/Reportes/ReporteCAPA_Plantilla.pdf>

**“Documentación Git”**

Link:<https://www.git-scm.com/doc>

**“Revisiones IEEE 1028**”

Link:

<https://github.com/ver0rivera/Plan-SQA/blob/main/Est%C3%A1ndares/Revisiones/IEEE%20-%20IEEE%20Std%201028-2008%20Standard%20for%20Software%20Reviews%20and%20Audits%20(2008)%20.pdf>

**“Plantillas de pruebas”**

Link: <https://github.com/ver0rivera/Plan-SQA/tree/main/Plantillas/Reportes/Plantillas%20de%20pruebas>

**“Norma ISO/IEC 12207:1997”**

Link:

<https://github.com/ver0rivera/Plan-SQA/blob/main/Est%C3%A1ndares/Capacitacion/-%20IEEE-EIA-IEC-ISO-12207-2-Institute%20of%20Electrical%20%26%20Electronics%20Enginee%20(1998).pdf>

**“Administración de la configuración”**

Link: <https://github.com/ver0rivera/ProcessModel-SQA/blob/main/Documentos/Administraci%C3%B3n%20de%20la%20Configuraci%C3%B3n%20de%20Software.pdf>

**“Estándares para el aseguramiento de la calidad”**

Link:<https://github.com/ver0rivera/Plan-SQA/tree/main/DocumentosReferenciados>

**“Documentación Jest”**

Link:<https://jestjs.io/docs/getting-started>

**“Documentación Appium”**

Link:<http://appium.io/docs/en/2.0/>

**“Documentación JMeter”**

Link:<https://jmeter.apache.org/usermanual/index.html>

**“Documentación BrowserStack”**

Link:<https://www.browserstack.com/docs/>

**“Documentación OWASP ZAP”**

Link:<https://www.zaproxy.org/docs/>

**“Documentación React Native”**

Link:<https://reactnative.dev/docs/getting-started>

**“Documentación Visual Studio”**

Link:<https://code.visualstudio.com/docs>

**“Documentación MongoDB”**

Link:<https://www.mongodb.com/docs/>

**“Documentación Node.js”**

Link:<https://nodejs.org/en/docs>

**“Documentación UserTesting”**

Link: <https://app.usertesting.com/docs/Welcome>

**“Estándar documentación del diseño”**

Link:

<https://github.com/ver0rivera/Plan-SQA/blob/main/Est%C3%A1ndares/Dise%C3%B1o/IEEE-1016-2009.pdf>

**“Recursos para la estructura lógica”**

Link: <https://github.com/ver0rivera/Plan-SQA/tree/main/Est%C3%A1ndares/Estructura%20logica>

**“Recursos para la codificación”**

Link: <https://github.com/ver0rivera/Plan-SQA/tree/main/Est%C3%A1ndares/Codificacion>

**“Recursos para la documentación”**

Link: <https://github.com/ver0rivera/Plan-SQA/tree/main/Est%C3%A1ndares/Documentacion>

**“Recursos para los requisitos”**

Link: <https://github.com/ver0rivera/Plan-SQA/tree/main/Est%C3%A1ndares/Requisitos>

**“Recursos para las métricas”**

Link: <https://github.com/ver0rivera/Plan-SQA/tree/main/Est%C3%A1ndares/Metricas>

**“Google JavaScript Style Guide”**

Link:<https://google.github.io/styleguide/jsguide.html>

**“Recursos para el mantenimiento”**

Link:<https://github.com/ver0rivera/Plan-SQA/tree/main/Est%C3%A1ndares/Mantenimiento>

**“Recursos para las pruebas”**

Link:<https://github.com/ver0rivera/Plan-SQA/tree/main/Est%C3%A1ndares/Pruebas>