Aplicación móvil, para implementar métricas de productividad en la empresa Creatibot en Otavalo

Plan de SQA

Versión 1.0

Historia de revisiones

Fecha	Versión	Descripción	Autor
07/03/2025	1.0	Plan de SQA	Stalin Quilumbango

Contenido

1. PROPÓSITO....3

2. REFERENCIAS....3

3. GESTIÓN....3

3.1. ORGANIZACIÓN....4
3.2. ACTIVIDADES....4
3.2.1. Ciclo de vida del software cubierto por el Plan....4
3.2.2. Actividades de calidad a realizarse....5
3.2.3. Revisar cada producto....5
3.2.4. Revisar el ajuste al proceso....5
3.2.5. Realizar Revisión Técnica Formal (RTF)....6
3.2.6. Asegurar que las desviaciones son documentadas....6
3.2.7. Relaciones entre las actividades de SQA y la planificación....6
3.3. RESPONSABLES....7

4. DOCUMENTACIÓN....7

4.1. PROPÓSITO....7
4.2. DOCUMENTACIÓN MÍNIMA REQUERIDA....7
4.2.1. Especificación de requerimientos del software....7
4.2.2. Descripción del diseño del software....8
4.2.3. Plan de Verificación & Validación....9
4.2.4. Reportes de Verificación & Validación....9
4.2.5. Documentación de usuario....9
4.2.6. Plan de Gestión de configuración....9
4.3. OTROS DOCUMENTOS....10

5. ESTÁNDARES, PRÁCTICAS, CONVENCIONES Y MÉTRICAS....10

5.1. ESTÁNDAR DE DOCUMENTACIÓN....11
5.2. ESTÁNDAR DE VERIFICACIÓN Y PRÁCTICAS....12
5.3. OTROS ESTÁNDARES....13

6. REVISIONES Y AUDITORÍAS....14

6.1. OBJETIVO....14
6.2. REQUERIMIENTOS MÍNIMOS....14
6.2.1. Revisión de requerimientos....15
6.2.2. Revisión de diseño preliminar....15
6.2.3. Revisión de diseño crítico....15
6.2.4. Revisión del Plan de Verificación & Validación....15
6.2.5. Auditoría funcional....15
6.2.6. Auditoría física....15
6.2.7. Auditorías internas al proceso....15
6.2.8. Revisiones de gestión....15
6.2.9. Revisión del Plan de gestión de configuración....16
6.2.10. Revisión Post Mortem....16
6.2.11. Agenda....16
6.3. OTRAS REVISIONES....17
6.3.1. Revisión de documentación de usuario....17

7. VERIFICACIÓN....17

8. REPORTE DE PROBLEMAS Y ACCIONES CORRECTIVAS....18

9. HERRAMIENTAS, TÉCNICAS Y METODOLOGÍAS....19

10. GESTIÓN DE RIESGOS....21

1. Propósito

El presente Plan de Calidad tiene como objetivo principal establecer los lineamientos y directrices que garantizarán la calidad en el desarrollo de la aplicación móvil para la implementación de métricas de productividad en los equipos de desarrollo de software de Creatibot, una Pyme ubicada en Otavalo.

El alcance de este Plan de Calidad se enfoca en la fase de desarrollo del software, incluyendo la especificación de requisitos, diseño, implementación, pruebas y validación del sistema. Sin embargo, no abarca la fase de mantenimiento y soporte posterior a la implementación del software en el entorno productivo.

El software que se está desarrollando será utilizado para medir la eficiencia del equipo de desarrollo, el tiempo de desarrollo y la calidad del software, proporcionando a Creatibot herramientas para la toma de decisiones estratégicas y la optimización del rendimiento de sus equipos de trabajo.

Los elementos del software que serán cubiertos por este Plan incluyen:

- Requisitos funcionales y no funcionales.
- Arquitectura del software.
- Codificación y buenas prácticas de desarrollo.
- Estrategias y criterios de pruebas.
- Procedimientos de validación y verificación.
- Documentación del sistema.

A través de este Plan de Calidad, se busca garantizar que el desarrollo de la aplicación móvil cumpla con los estándares de calidad establecidos, promoviendo la mejora continua en el proceso de desarrollo de software dentro de Creatibot.

2. Referencias

[1]ANSI/IEEE Std 730.1-1989, IEEE Standard for Software Quality Assurance Plans.

3. Gestión

Para asegurar la correcta aplicación del Plan de Calidad, se establece la siguiente estructura organizativa y responsabilidades dentro del equipo de desarrollo:

- **Gerente del Proyecto:** Responsable de supervisar el cumplimiento del Plan de Calidad, asignar recursos y tomar decisiones estratégicas.
- **Líder Técnico:** Encargado de definir estándares de desarrollo, revisar el código y asegurar la implementación de buenas prácticas.
- **Equipo de Desarrollo:** Responsable de implementar las funcionalidades del software siguiendo los lineamientos de calidad establecidos.
- **Equipo de Pruebas:** Encargado de diseñar, ejecutar y documentar pruebas para validar la calidad del software.
- **Responsable de Documentación:** Asegura que toda la documentación del proyecto esté actualizada y alineada con el Plan de Calidad.

Las principales actividades dentro de la gestión de calidad incluyen:

- Definición y monitoreo de indicadores clave de calidad.
- Revisión periódica del código y auditorías técnicas.
- Implementación de pruebas automatizadas y manuales.

- Evaluación continua del desempeño del equipo y del producto.
- Revisión y actualización del Plan de Calidad conforme avanza el desarrollo del proyecto.

Esta estructura y conjunto de actividades buscan garantizar que la aplicación móvil cumpla con los estándares de calidad definidos, optimizando la eficiencia del equipo de desarrollo de Creatibot.

3.1. Organización

El equipo de trabajo se organiza en diferentes líneas de trabajo que tienen influencia y control sobre la calidad del software. Estas líneas de trabajo están estructuradas de la siguiente manera:

- **Línea de Gestión del Proyecto:** Encabezada por el Gerente del Proyecto, quien supervisa la planificación, ejecución y control del desarrollo, asegurando el cumplimiento de los estándares de calidad.
- **Línea de Desarrollo:** Liderada por el Líder Técnico, esta línea se encarga de la implementación del software siguiendo buenas prácticas de codificación y arquitecturas de software establecidas.
- **Línea de Pruebas y Validación:** Integrada por el equipo de pruebas, su función principal es diseñar y ejecutar pruebas de software para detectar errores y asegurar el correcto funcionamiento del sistema.
- **Línea de Documentación y Control de Calidad:** Esta línea garantiza que todos los procesos, requisitos y pruebas sean documentados adecuadamente, proporcionando evidencia del cumplimiento de los estándares de calidad.

Las dependencias e interacciones entre estas líneas de trabajo son clave para garantizar la calidad del software. La Línea de Desarrollo colabora estrechamente con la Línea de Pruebas y Validación para detectar y corregir errores de manera temprana. A su vez, la Línea de Documentación y Control de Calidad trabaja en conjunto con todas las demás líneas para asegurar la trazabilidad y transparencia de los procesos.

Esta organización permite un flujo de trabajo estructurado, asegurando que cada etapa del desarrollo del software cumpla con los criterios de calidad definidos en este Plan.

3.2. Actividades

3.2.1. Ciclo de vida del software cubierto por el Plan

El Plan de Calidad cubre las siguientes etapas del ciclo de vida del desarrollo de software:

- **Etapa de Requerimientos y Análisis:** Identificación y documentación de los requisitos funcionales y no funcionales, así como la validación de su viabilidad.
- **Etapa de Diseño:** Definición de la arquitectura del software, modelos de datos y estructura del sistema.
- **Etapa de Implementación:** Desarrollo del código fuente siguiendo estándares de calidad y buenas prácticas de programación.
- **Etapa de Pruebas:** Realización de pruebas unitarias, de integración y de sistema para validar la calidad y funcionalidad del software.
- **Etapa de Validación y Entrega:** Verificación del cumplimiento de requisitos y aprobación del software para su implementación.

Los productos del proyecto que estarán sujetos a revisiones de calidad incluyen:

- Especificaciones de requisitos del software.
- Diseños de arquitectura y modelos de datos.
- Código fuente y documentación técnica.
- Casos de prueba y resultados de pruebas ejecutadas.
- Informes de validación y aprobación del software.

Estas revisiones de calidad permitirán asegurar que cada etapa del ciclo de vida del software cumpla con los estándares definidos en este Plan.

3.2.2. Actividades de calidad a realizarse

Las tareas a ser llevadas a cabo deberán reflejar las evaluaciones a realizar, los estándares a seguir, los productos a revisar, los procedimientos a seguir en la elaboración de los distintos productos y los procedimientos para informar de los defectos detectados a sus responsables y realizar el seguimiento de los mismos hasta su corrección.

Las actividades que se realizarán son:

- Revisar cada producto
- Revisar el ajuste al proceso
- Realizar Revisión Técnica Formal (RTF)
- Asegurar que las desviaciones son documentadas.

3.2.3. Revisar cada producto

En esta actividad se revisan los productos que se definieron como claves para verificar en el Plan de calidad.

Se debe verificar que no queden correcciones sin resolver en los informes de revisión previos, si se encuentra alguna no resuelta, debe ser incluida en la siguiente revisión. Se revisan los productos contra los estándares, utilizando la checklist definida para el producto.

Se debe identificar, documentar y seguir la pista a las desviaciones encontradas y verificar que se hayan realizado las correcciones.

Como salida se obtiene el Informe de revisión de SQA, este informe debe ser distribuido a los responsables del producto y se debe asegurar de que son concientes de desviaciones o discrepancias encontradas.

3.2.4. Revisar el ajuste al proceso

En esta actividad se revisan los productos que se definieron como claves para verificar el cumplimiento de las actividades definidas en el proceso. Con el fin de asegurar la calidad en el producto final del desarrollo, se deben llevar a cabo revisiones sobre los productos durante todo el ciclo de vida del software.

Se debe recoger la información necesaria de cada producto, buscando hacia atrás los productos previos que deberían haberse generado, para poder establecer los criterios de revisión y evaluar si el producto cumple con las especificaciones.

Esta información se obtiene de los siguientes documentos:

Plan del Proyecto, Plan de la iteración, Plan de Verificación.

Antes de comenzar, se debe verificar en los informes de revisión previos que todas las desviaciones fueron corregidas, si no es así, las faltantes se incluyen para ser evaluadas.

Como salida se obtiene el Informe de revisión de SQA correspondiente a la evaluación de ajuste al Proceso, este informe debe ser distribuido a los responsables de las actividades y se debe asegurar de que son concientes de desviaciones o discrepancias encontradas.

3.2.5. Realizar Revisión Técnica Formal (RTF)

El objetivo de la RTF es descubrir errores en la función, la lógica ó la implementación de cualquier producto del software, verificar que satisface sus especificaciones, que se ajusta a los estándares establecidos, señalando las posibles desviaciones detectadas. Es un proceso de revisión riguroso, su objetivo es llegar a detectar lo antes posible, los posibles defectos o desviaciones en los productos que se van generando a lo largo del desarrollo. Por esta característica se adopta esta práctica para productos que son de especial importancia.

En la reunión participan el responsable de SQA e integrantes del equipo de desarrollo.

Se debe convocar a la reunión formalmente a los involucrados, informar del material que ellos deben preparar por adelantado, llevar una lista de preguntas y dudas que surgen del estudio del producto a ser revisado.

La duración de la reunión no debe ser mayor a dos horas.

Como salida se obtiene el Informe de RTF.

3.2.6. Asegurar que las desviaciones son documentadas

Las desviaciones encontradas en las actividades y en los productos deben ser documentadas y ser manejadas de acuerdo a un procedimiento establecido.

Se debe chequear que los responsables de cada plan los modifiquen cada vez que sea necesario, basados en las desviaciones encontradas.

3.2.7. Relaciones entre las actividades de SQA y la planificación

A continuación, se presenta una lista de las actividades de aseguramiento de la calidad del software (SQA) y su planificación en semanas:

Actividad	**Semana cuando se realiza**
Revisión y validación de requisitos del software.	Semana 1-2
Definición de estándares de calidad y buenas prácticas de desarrollo.	Semana 3-4
Implementación de auditorías de código y revisiones técnicas.	Semana 5-6
Ejecución de pruebas unitarias y de integración.	Semana 7-8
Evaluación del cumplimiento de criterios de calidad en el software desarrollado.	Semana 9-10
Validación final del producto y documentación de resultados de	Semana 11-12

pruebas.	
Revisión del Plan de Calidad y mejoras para futuros desarrollos.	Semana 13

3.3. Responsables

Responsables

Cada actividad identificada cuenta con un responsable específico:

- **Gerente del Proyecto:** Supervisión general del cumplimiento del Plan de Calidad.
- **Líder Técnico:** Definición y aseguramiento de estándares de calidad en el desarrollo.
- **Equipo de Desarrollo:** Implementación de funcionalidades siguiendo estándares definidos.
- **Equipo de Pruebas:** Ejecución de pruebas y validación de calidad.
- **Responsable de Documentación:** Gestión y actualización de la documentación del proyecto.

4. Documentación

4.1. Propósito

Identificación de la documentación relativa a desarrollo, Verificación & Validación, uso y mantenimiento del software.

Establecer como los documentos van a ser revisados para chequear consistencia: se confirman criterio e identificación de las revisiones.

4.2. Documentación mínima requerida

La documentación mínima es la requerida para asegurar que la implementación logrará satisfacer los requerimientos.

4.2.1. Especificación de requerimientos del software

El documento de especificación de requerimientos deberá describir, de forma clara y precisa, cada uno de los requerimientos esenciales del software además de las interfaces externas.

El cliente deberá obtener como resultado del proyecto una especificación adecuada a sus necesidades en el área de alcance del proyecto, de acuerdo al compromiso inicial del trabajo y a los cambios que este haya sufrido a lo largo del proyecto, que cubra aquellos aspectos que se haya acordado detallar con el cliente.

La especificación debe:

- Ser completa :

  a. Externa, respecto al alcance acordado.

  b. Internamente, no deben existir elementos sin especificar.
- Ser consistente, no pueden haber elementos contradictorios.
- Ser no ambigua, todo término referido al área de aplicación debe estar definido en un glosario.
- Ser verificable, debe ser posible verificar siguiendo un método definido, si el producto final cumple o no con cada requerimiento.

- Estar acompañada de un detalle de los procedimientos adecuados para verificar si el producto cumple o no con los requerimientos.
- Incluir requerimientos de calidad del producto a construir.

Los requerimientos de calidad del producto a construir son considerados dentro de atributos específicos del software que tienen incidencia sobre la calidad en el uso' y se detallan a continuación:

Funcionalidad

a. adecuación a las necesidades
b. precisión de los resultados
c. interoperabilidad
d. seguridad de los datos

Confiabilidad

a. madurez
b. tolerancia a faltas
c. recuperabilidad (Ver si aplica)

Usabilidad

a. comprensible
b. aprendible
c. operable
d. atractivo

Eficiencia

a. comportamiento respecto al tiempo (Ver si aplica)
b. utilización de recursos

Mantenibilidad

a. analizable
b. modificable
c. estable, no se producen efectos inesperados luego de modificaciones
d. verificable

Portabilidad

a. adaptable (Ver si aplica)
b. instalable
c. co-existencia
d. reemplazante (Ver si aplica)

Cada uno de estos atributos debe cumplir con las normas y regulaciones aplicables a cada uno.

4.2.2. Descripción del diseño del software

El documento de diseño especifica como el software será construido para satisfacer los requerimientos.

Deberá describir los componentes y subcomponentes del diseño del software, incluyendo interfaces internas. Este documento deberá ser elaborado primero como Preliminar y luego será gradualmente extendido hasta llegar a obtener el Detallado.

El cliente deberá obtener como resultado del proyecto el diseño de un producto de software que cubra aquellos aspectos que se haya acordado con el cliente incorporar al diseño, en función de la importancia que estos presenten y de sus conexiones lógicas.

El diseño debe:

- Corresponder a los requerimientos a incorporar:
  a. Todo elemento del diseño debe contribuir a algún requerimiento
  b. La implementación de todo requerimiento a incorporar debe estar contemplada en por lo menos un elemento del diseño.
- Ser consistente con la calidad del producto

4.2.3. Plan de Verificación & Validación

El Plan de V & V deberá identificar y describir los métodos a ser utilizados en:

- La verificación de que:
  a. los requerimientos descritos en el documento de requerimientos han sido aprobados por una autoridad apropiada. En este caso sería que cumplan con el acuerdo logrado entre el cliente y el equipo.
  b. los requerimientos descritos en el documento de requerimientos son implementados en el diseño expresado en el documento de diseño.
  c. el diseño expresado en el documento de diseño esta implementado en código.
- Validar que el código, cuando es ejecutado, se adecua a los requerimientos expresados en el documento de requerimientos.

4.2.4. Reportes de Verificación & Validación

Estos documentos deben especificar los resultados de la ejecución de los procesos descritos en el Plan de V & V.

4.2.5. Documentación de usuario

La documentación de usuario debe especificar y describir los datos y entradas de control requeridos, así como la secuencia de entradas, opciones, limitaciones de programa y otros elementos necesarios para la ejecución exitosa del software.

Todos los errores deben ser identificados y las acciones correctivas descritas.

Como resultado del proyecto el cliente obtendrá una documentación para el usuario de acuerdo a los requerimientos específicos del proyecto.

4.2.6. Plan de Gestión de configuración

El Plan de gestión de configuración debe contener métodos para identificar componentes de software, control e implementación de cambios, y registro y reporte del estado de los cambios implementados.

4.3. Otros documentos

Esta sección incluye otros documentos relevantes para la calidad del producto:

- Plan de Desarrollo.
- Plan de Proyecto.
- Manual de estándares y procedimientos.
- Documentación de especificaciones de requisitos.
- Informes de auditoría y control de calidad.

5. Estándares, prácticas, convenciones y métricas

Estándares de Calidad

Se adoptarán estándares internacionales para evaluar la calidad del software, asegurando que los productos desarrollados sean eficientes, mantenibles y funcionales.

- **ISO/IEC 25010:** Para evaluar la calidad del software en dimensiones como funcionalidad, eficiencia, mantenibilidad y usabilidad.
- **ISO/IEC 9126:** Para medir la calidad interna y externa del software a partir de características específicas.
- **IEEE 730 (Software Quality Assurance Plans):** Para establecer lineamientos en el proceso de aseguramiento de la calidad.

Prácticas de Desarrollo

- Se aplicarán prácticas recomendadas en metodologías ágiles para mejorar la eficiencia y efectividad en los procesos de desarrollo:
- **Scrum:** Para la gestión iterativa de los proyectos, promoviendo la mejora continua y la entrega incremental del software.
- **Code Review:** Revisión de código colaborativa para asegurar calidad, detección temprana de errores y cumplimiento de buenas prácticas.
- **Continuous Integration / Continuous Deployment (CI/CD):** Para la integración frecuente de código y despliegue automatizado, reduciendo errores en producción.
- **Testing Automatizado:** Implementación de pruebas unitarias y de integración para verificar la estabilidad del software.

Convenciones de Desarrollo

- Para estandarizar el trabajo dentro del equipo, se seguirán convenciones de codificación y documentación:
- **Convenciones de Codificación:**
- Uso de **Google Java Style Guide** para código en Java.
- Aplicación de **PEP 8** para código en Python.
- Uso de estándares de **Airbnb JavaScript Style Guide** para desarrollo en JavaScript.
- **Gestión de Versiones:**
- Uso de **GitFlow** como estrategia de ramificación en Git.
- Commits descriptivos siguiendo la convención **Conventional Commits**.
- **Documentación:**
- Uso de **Markdown y JSDoc** para documentación técnica del código.

- Aplicación de **PlantUML** para diagramas de arquitectura y flujos de trabajo.

Métricas de Productividad y Calidad

- Para evaluar el desempeño del equipo y la calidad del software, se emplearán métricas cuantificables:
- **Métricas de Productividad:**
- **Velocidad del equipo:** Número de historias de usuario completadas en cada sprint.
- **Tiempo de desarrollo:** Cantidad de horas invertidas en la implementación de cada funcionalidad.
- **Lead Time:** Tiempo desde la solicitud de una tarea hasta su entrega final.
- **Métricas de Calidad del Código:**
- **Cobertura de pruebas:** Porcentaje de código cubierto por pruebas unitarias y de integración.
- **Defect Density:** Número de errores detectados por cada mil líneas de código.
- **Technical Debt Ratio:** Medida del esfuerzo necesario para corregir problemas técnicos en el código base.

Monitoreo y Aseguramiento del Cumplimiento

- Para garantizar la implementación y cumplimiento de estos estándares y métricas, se utilizarán los siguientes mecanismos:
- **Herramientas de Monitoreo:**
- **SonarQube:** Para análisis de calidad del código y detección de deuda técnica.
- **Jenkins/GitHub Actions:** Para integración y despliegue continuo.
- **JIRA/Trello:** Para la gestión de tareas y medición de velocidad del equipo.
- **Google Forms/Encuestas en Confluence:** Para evaluar la percepción de la productividad y calidad del equipo.
- **Revisiones Periódicas:**
- **Reuniones de retrospectiva:** Evaluación al final de cada sprint sobre el desempeño y oportunidades de mejora.
- **Auditorías internas de código:** Revisión semestral de adherencia a estándares de codificación y documentación.
- **Evaluaciones de métricas:** Análisis trimestral de los indicadores de productividad y calidad para detectar tendencias y aplicar mejoras.

5.1. Estándar de documentación

Como estándares de documentación se definirán dos documentos:

- Estándar de documentación técnica y
- Estándar de documentación de usuario.

La documentación técnica del producto debe:

- Ser adecuada para que un grupo independiente del de desarrollo pueda encarar el mantenimiento del producto.
- Incluir fuentes, Modelos de Casos de Uso, Objetos

Para la escritura de documentos se han definido plantillas para ser utilizadas en la elaboración de entregables.

En estas plantillas se definen:

- encabezado y pie de página.
- fuente y tamaño de fuente para estilo normal
- fuente y tamaño de fuente para los títulos a utilizar
- datos mínimos que se deben incluir: fecha, versión y responsables.

Documentación Técnica

- **IEEE 830 (Software Requirements Specification - SRS):** Para la documentación de requerimientos del software.
- **IEEE 1016 (Software Design Description - SDD):** Para la especificación detallada de la arquitectura y diseño del software.
- **ISO/IEC 26514:** Para la documentación de la interfaz de usuario y guías de usuario.
- **PlantUML:** Para la generación de diagramas UML en la documentación del sistema.

Documentación del Código

- **Javadoc (para Java), Doxygen (para C++) y Pydoc (para Python):** Para la generación automática de documentación del código.
- **JSDoc:** Para la documentación de código JavaScript y TypeScript.
- **Markdown:** Para la documentación ligera dentro de los repositorios y en plataformas como GitHub o Confluence.

Gestión y Control de Documentación

- **GitHub Wiki o Confluence:** Para la documentación colaborativa y en línea.
- **Normas APA 7.ª edición:** Para la citación y referencias bibliográficas en informes técnicos.
- **Plantillas estándar de Creatibot:** Para garantizar uniformidad en documentos de especificación, diseño y pruebas.

5.2. Estándar de verificación y prácticas

Se utilizan las prácticas definidas en el Plan de Verificación y Validación.

Como estándar se utiliza el documento de:

Std 1012-1986 IEEE Standard for Software Verification and Validation Plans.

Estándares de Verificación

La verificación del software es fundamental para garantizar su calidad antes de la implementación. Se utilizarán los siguientes estándares:

- **ISO/IEC 25010:** Para evaluar la calidad del producto en términos de funcionalidad, rendimiento, seguridad y mantenibilidad.
- **IEEE 1012 (Verification & Validation - V&V):** Para definir los procesos de verificación y validación en el ciclo de vida del software.
- **ISTQB (International Software Testing Qualifications Board):** Para aplicar mejores prácticas en la verificación y pruebas de software.
- **ISO/IEC 12207:** Para estandarizar los procesos del ciclo de vida del software, incluyendo verificación y aseguramiento de la calidad.

Métodos de Verificación Utilizados

Se emplearán diferentes métodos para verificar la calidad y correcto funcionamiento del software:

- **Revisión de Código (Code Review):** Evaluación manual del código fuente para detectar defectos y mejorar la calidad del código.
- **Pruebas Estáticas:** Análisis del código sin ejecutarlo, mediante herramientas como linters y análisis de calidad.
- **Pruebas Dinámicas:** Ejecución del software para verificar su comportamiento bajo diferentes condiciones.

Prácticas de Verificación

- **Integración Continua (CI):** Uso de herramientas como **Jenkins** o **GitHub Actions** para automatizar la verificación del código en cada cambio.
- **Análisis Estático del Código:** Implementación de herramientas como **SonarQube**, **ESLint**, **Pylint** y **Checkstyle** para detectar errores sin ejecutar el código.
- **Pruebas Unitarias:** Uso de frameworks como **JUnit (Java), PyTest (Python), Mocha (JavaScript)** para validar el correcto funcionamiento de los módulos del software.
- **Pruebas de Seguridad:** Aplicación de herramientas como **OWASP ZAP** para detectar vulnerabilidades en el software.
- **Evaluación de Métricas de Código:** Aplicación de métricas de calidad como **complejidad ciclomática**, **cobertura de código** y **densidad de defectos**.

Prácticas de Validación

- **Pruebas Funcionales:** Validación de que el software cumple con los requerimientos establecidos utilizando herramientas como Selenium.
- **Pruebas de Rendimiento:** Evaluación de tiempos de respuesta y escalabilidad con **JMeter** o **Gatling**.
- **Pruebas de Usabilidad:** Recolección de feedback de usuarios mediante encuestas y pruebas controladas.
- **Revisión de Requisitos:** Validación de que los requisitos documentados cumplen con las necesidades del cliente y usuarios finales.

5.3. Otros Estándares

Estándares de Seguridad

Para garantizar la seguridad del software y la protección de los datos, se aplicarán los siguientes estándares:

- **OWASP Top 10:** Lineamientos para identificar y mitigar vulnerabilidades en aplicaciones web.
- **ISO/IEC 27001:** Estándar internacional para la gestión de la seguridad de la información.
- **NIST Cybersecurity Framework:** Directrices para la gestión de riesgos de ciberseguridad.
- **GDPR (Reglamento General de Protección de Datos):** Para la protección de datos personales en cumplimiento con normativas internacionales.

6. Revisiones y auditorías

6.1. Objetivo

Definición de las revisiones y auditorías técnicas y de gestión que se realizarán.

Especificación de cómo serán llevadas a cabo dichas revisiones y auditorías.

6.2. Requerimientos mínimos

Se realizarán revisiones periódicas en diferentes fases del desarrollo del software para detectar errores, mejorar la calidad del código y garantizar la alineación con los objetivos del proyecto.

Tipos de Revisiones y Frecuencia

Tipo de Revisión	Descripción	Frecuencia	Responsable
Revisión de Código (Code Review)	Evaluación manual del código fuente para detectar errores y mejorar su calidad.	En cada fusión de código (Pull Request).	Desarrolladores senior / Líder técnico.
Revisión de Requisitos	Verificación de que los requisitos documentados son claros, completos y viables.	Inicio de cada sprint.	Product Owner / Líder de proyecto.
Revisión de Arquitectura	Evaluación de la estructura del sistema para garantizar escalabilidad y mantenibilidad.	Cada dos sprints.	Arquitecto de software / Líder técnico.
Revisión de Diseño de UI/UX	Validación de que la interfaz de usuario sigue estándares de usabilidad y accesibilidad.	Antes del desarrollo de nuevas funcionalidades.	Diseñador UI/UX / Desarrolladores frontend.
Revisión de Pruebas	Verificación de cobertura de pruebas y efectividad en la detección de errores.	En cada entrega de sprint.	QA Tester / Líder técnico.

Tipos de Auditorías y Frecuencia

Tipo de Auditoría	Descripción	Frecuencia	Responsable
Auditoría de Calidad del Código	Evaluación del código fuente con herramientas automáticas (SonarQube, ESLint, Pylint, etc.).	Mensual.	Líder técnico / QA Tester.
Auditoría de Seguridad	Identificación de vulnerabilidades en el sistema utilizando OWASP ZAP y pruebas de penetración.	Trimestral.	Equipo de seguridad / DevOps.
Auditoría de Cumplimiento de Procesos	Evaluación del seguimiento de estándares de desarrollo y gestión de proyectos (Scrum, GitFlow, CI/CD).	Trimestral.	Líder de proyecto / Equipo de calidad.
Auditoría de Productividad	Análisis de métricas de productividad del equipo (velocidad, lead time, defect density).	Trimestral.	Líder de proyecto / Scrum Master.

Agenda para la Realización de Revisiones y Auditorías

Actividad	Frecuencia	Día Sugerido	Responsables
Revisión de Código	En cada Pull Request	Variable según sprint	Desarrolladores / Líder técnico
Revisión de Requisitos	Inicio de cada sprint	Lunes de la primera semana del sprint	Product Owner / Líder de proyecto
Revisión de UI/UX	Antes del desarrollo de nuevas funcionalidades	Antes de cada sprint	Diseñador UI/UX / Desarrolladores frontend
Revisión de Pruebas	Al cierre de cada sprint	Último día del sprint	QA Tester / Líder técnico
Auditoría de Código	Mensual	Primer viernes del mes	Líder técnico / QA Tester
Auditoría de Seguridad	Trimestral	Última semana del trimestre	Equipo de seguridad / DevOps
Auditoría de Procesos	Trimestral	Última semana del trimestre	Líder de proyecto / Equipo de calidad
Auditoría de Productividad	Trimestral	Primera semana del trimestre	Líder de proyecto / Scrum Master

6.2.1. Revisión de requerimientos

Esta revisión se realiza para asegurar que se cumplió con los requerimientos especificados por el Cliente.

6.2.2. Revisión de diseño preliminar

Esta revisión se realiza para asegurar la consistencia y suficiencia técnica del diseño preliminar del software.

6.2.3. Revisión de diseño crítico

Esta revisión se realiza para asegurar la consistencia del diseño detallado con la especificación de requerimientos.

6.2.4. Revisión del Plan de Verificación & Validación

Esta revisión se realiza para asegurar la consistencia y completitud de los métodos especificados en el Plan de V & V.

6.2.5. Auditoría funcional

Esta auditoría se realiza previa a la liberación del software, para verificar que todos los requerimientos especificados en el documento de requerimientos fueron cumplidos.

6.2.6. Auditoría física

Esta revisión se realiza para verificar que el software y la documentación son consistentes y están aptos para la liberación.

6.2.7. Auditorías internas al proceso

Estas auditorías son para verificar la consistencia: del código versus el documento de diseño, especificaciones de interfase, implementaciones de diseño versus requerimientos funcionales, requerimientos funcionales versus descripciones de testeo.

6.2.8. Revisiones de gestión

Estas revisiones se realizan periódicamente para asegurar la ejecución de todas las actividades identificadas en este Plan. Deben realizarse por una persona ajena al grupo de trabajo (en caso de que sea posible).

6.2.9. Revisión del Plan de gestión de configuración

Esta revisión se realiza para asegurar la consistencia y completitud de los métodos especificados en el Plan de gestión de configuración.

6.2.10. Revisión Post Mortem

Esta revisión se realiza al concluir el proyecto para especificar las actividades de desarrollo implementadas durante el proyecto y para proveer recomendaciones.

6.2.11. Agenda

Para garantizar el cumplimiento de los estándares de calidad y productividad en Creatibot, se establece la siguiente agenda para la ejecución de revisiones y auditorías. Esta agenda permitirá un seguimiento estructurado y periódico de las actividades de verificación y aseguramiento de calidad en el desarrollo del software.

Agenda de Revisiones

Actividad	Frecuencia	Día Sugerido	Responsables
Revisión de Código (Code Review)	En cada Pull Request	Variable según sprint	Desarrolladores / Líder técnico
Revisión de Requisitos	Inicio de cada sprint	Lunes de la primera semana del sprint	Product Owner / Líder de proyecto
Revisión de UI/UX	Antes del desarrollo de nuevas funcionalidades	Antes de cada sprint	Diseñador UI/UX / Desarrolladores frontend
Revisión de Arquitectura	Cada dos sprints	Viernes de la segunda semana del sprint	Arquitecto de software / Líder técnico
Revisión de Pruebas	Al cierre de cada sprint	Último día del sprint	QA Tester / Líder técnico

Agenda de Auditorías

Actividad	Frecuencia	Día Sugerido	Responsables
Auditoría de Calidad del Código	Mensual	Primer viernes del mes	Líder técnico / QA Tester
Auditoría de Seguridad	Trimestral	Última semana del trimestre	Equipo de seguridad / DevOps
Auditoría de Cumplimiento de Procesos	Trimestral	Última semana del trimestre	Líder de proyecto / Equipo de calidad
Auditoría de Productividad	Trimestral	Primera semana del trimestre	Líder de proyecto / Scrum Master

Consideraciones para la Ejecución de Revisiones y Auditorías

- Todas las revisiones y auditorías deberán ser documentadas en informes que incluyan hallazgos, recomendaciones y acciones correctivas.
- Los resultados de auditorías trimestrales serán presentados en reuniones estratégicas para definir mejoras en procesos y metodologías.

- Las revisiones de código deberán realizarse antes de fusionar cualquier cambio en la rama principal del repositorio.
- Se utilizarán herramientas de gestión como JIRA, Trello o Notion para programar y dar seguimiento a las actividades de revisión y auditoría.
- Las auditorías de seguridad incluirán pruebas de penetración y análisis de vulnerabilidades con herramientas como OWASP ZAP.

6.3. Otras revisiones

6.3.1. Revisión de documentación de usuario

Se revisa la completitud, claridad, correctitud y aplicación de uso.

7. Verificación

En esta sección se identifican las verificaciones adicionales que no fueron contempladas en el Plan de Verificación y Validación (V&V) del software. Además, se especifican los métodos a ser utilizados para garantizar la calidad del desarrollo y el cumplimiento de los requisitos definidos.

Verificaciones Adicionales

A continuación, se detallan las verificaciones complementarias que se implementarán para asegurar la estabilidad, funcionalidad y seguridad del software desarrollado en Creatibot.

Tipo de Verificación	Descripción	Método de Verificación	Frecuencia	Responsable
Verificación de Integración Continua	Garantizar que las nuevas implementaciones no generen fallos en el sistema.	Uso de pipelines de CI/CD (GitHub Actions, GitLab CI, Jenkins).	En cada commit o fusión de código.	Equipo de desarrollo / DevOps.
Verificación de Performance y Escalabilidad	Evaluar tiempos de respuesta y carga bajo diferentes condiciones.	Pruebas de carga y estrés con JMeter, Gatling o Locust.	Mensual y antes de cada lanzamiento.	QA Tester / DevOps.
Verificación de Seguridad	Identificación de vulnerabilidades en el código y la infraestructura.	Análisis con SonarQube, OWASP ZAP y pruebas de penetración.	Trimestral.	Equipo de seguridad / DevOps.
Verificación de Accesibilidad	Validar cumplimiento con estándares de accesibilidad (WCAG 2.1).	Evaluación con AXE, Lighthouse y revisiones manuales.	Antes del lanzamiento de cada nueva funcionalidad.	Diseñador UI/UX / QA Tester.

Métodos de Verificación

Los métodos de verificación a aplicar incluyen:

1. **Revisiones de Código:** Evaluación manual del código por desarrolladores senior para detectar errores y mejoras.
2. **Pruebas Automatizadas:** Implementación de pruebas unitarias, de integración y end-to-end con Selenium, Jest o Cypress.
3. **Pruebas de Carga y Estrés:** Simulación de múltiples usuarios simultáneos para evaluar rendimiento y escalabilidad.

4. **Análisis de Seguridad:** Escaneos de vulnerabilidades con herramientas automatizadas y pruebas de penetración manuales.
5. **Evaluaciones de Accesibilidad:** Uso de herramientas automáticas y revisiones con usuarios con necesidades especiales.
6. **Encuestas de Usabilidad:** Obtención de feedback cualitativo mediante cuestionarios y entrevistas a usuarios.
7. **Monitoreo en Producción:** Seguimiento continuo de métricas de desempeño con herramientas como New Relic o Datadog.

8. Reporte de problemas y acciones correctivas

Esta sección detalla las prácticas y procedimientos que se seguirán para el **reporte, seguimiento y resolución** de los problemas detectados durante el desarrollo del software en Creatibot. También se especifican los responsables de la implementación de las acciones correctivas.

Prácticas y Procedimientos para el Reporte de Problemas

El proceso de reporte de problemas se gestionará mediante un **sistema de seguimiento de incidencias**, como **JIRA, Trello o GitHub Issues**, asegurando una trazabilidad efectiva.

1. **Detección del Problema:**
    - Cualquier miembro del equipo puede identificar un problema y deberá documentarlo en la herramienta de gestión.
2. **Registro del Problema:**
    - Se generará un ticket que contendrá la siguiente información mínima:
        - **Título:** Breve descripción del problema.
        - **Descripción:** Explicación detallada del problema, incluyendo evidencia (logs, capturas de pantalla, errores).
        - **Prioridad:** Baja, Media, Alta, Crítica.
        - **Categoría:** Código, UI/UX, Seguridad, Performance, Base de Datos, Infraestructura.
        - **Estado:** Abierto, En progreso, Resuelto, Cerrado.
        - **Responsable:** Persona asignada para la solución.
3. **Asignación del Problema:**
    - El líder técnico o el Scrum Master asignará el problema al miembro del equipo más adecuado según su categoría y complejidad.
4. **Análisis y Diagnóstico:**
    - Se investigará el problema, verificando su causa raíz y posibles soluciones.
5. **Desarrollo de la Solución:**
    - Se implementará la corrección del problema siguiendo las prácticas establecidas de calidad.
6. **Pruebas y Validación:**
    - El equipo de QA verificará que el problema se haya solucionado y que no afecte otras funcionalidades.
7. **Cierre del Ticket:**
    - Si las pruebas son exitosas, el problema se marca como "Cerrado" en el sistema de gestión.

Seguimiento y Resolución de Problemas

Para garantizar que los problemas sean resueltos de manera efectiva, se implementarán las siguientes estrategias:

- **Revisiones Semanales:** Se realizará una reunión cada semana para revisar el estado de los problemas abiertos y priorizar su solución.
- **Análisis de Tendencias:** Se analizarán los problemas recurrentes para identificar patrones y aplicar soluciones preventivas.
- **Documentación de Soluciones:** Se registrarán las soluciones implementadas en una base de conocimiento para referencia futura.

Acciones Correctivas

Si se identifican fallas recurrentes o críticas, se tomarán acciones correctivas para evitar su repetición:

Problema Detectado	Acción Correctiva	Responsable
Errores frecuentes en el código	Revisión de estándares de codificación y formación del equipo	Líder técnico / Desarrolladores
Problemas de rendimiento	Optimización del código y pruebas de carga adicionales	DevOps / QA
Vulnerabilidades de seguridad	Implementación de auditorías de seguridad más frecuentes	Equipo de seguridad
Deficiencias en la UI/UX	Realización de pruebas de usuario y ajustes en diseño	Diseñador UI/UX
Falta de documentación	Refuerzo en la documentación del proyecto	Todo el equipo
Problemas en la gestión del proyecto	Evaluación de metodologías ágiles y ajuste de procesos	Scrum Master / Líder de Proyecto

Responsables del Reporte y Acciones Correctivas

Rol	Responsabilidades
Desarrolladores	Reportar y corregir problemas de código.
QA Tester	Identificar errores en pruebas y validar correcciones.
Líder Técnico	Supervisar la solución de problemas críticos y asegurar la calidad.
Scrum Master	Priorizar la resolución de problemas en cada sprint.
Equipo de Seguridad	Identificar y mitigar vulnerabilidades.
Product Owner	Verificar que las correcciones alineen con los requisitos del producto.

9. Herramientas, técnicas y metodologías

En esta sección se identifican las **herramientas de software, técnicas y metodologías** que se emplearán para el aseguramiento de calidad en el desarrollo de la aplicación móvil para métricas de productividad en Creatibot. Estas herramientas respaldarán las actividades de verificación, validación, pruebas y gestión del proyecto, asegurando la eficiencia y calidad del software desarrollado.

Herramientas de Software

A continuación, se detallan las herramientas utilizadas para cada aspecto del aseguramiento de calidad:

Área	Herramienta	Propósito
Gestión de Proyectos	Jira, Trello, Notion	Seguimiento de tareas, gestión de backlog, planificación de sprints.
Control de Versiones	Git, GitHub, GitLab	Gestión de código fuente, colaboración y control de cambios.
Integración y Despliegue Continuo (CI/CD)	GitHub Actions, Jenkins, GitLab CI	Automatización de pruebas, compilación y despliegue.
Pruebas Automatizadas	Jest, Selenium, Cypress, JUnit	Pruebas unitarias, de integración y end-to-end.
Revisión de Código	SonarQube, ESLint, Prettier, Checkstyle	Análisis estático de código, aseguramiento de estándares.
Pruebas de Performance	JMeter, Gatling, Locust	Evaluación de tiempos de respuesta y escalabilidad.
Monitoreo y Logueo	New Relic, Datadog, ELK Stack	Análisis de rendimiento y detección de fallos en producción.
Pruebas de Seguridad	OWASP ZAP, SonarQube, Snyk	Identificación de vulnerabilidades en el código y la infraestructura.
Gestión de Documentación	Confluence, Google Docs, Notion	Creación y mantenimiento de documentación técnica y de usuario.

Técnicas de Aseguramiento de Calidad

Para garantizar la calidad del software, se aplicarán las siguientes técnicas:

1. **Revisión de Código:** Evaluación manual y con herramientas de análisis estático para detectar errores y mejoras.
2. **Pruebas Automatizadas:** Implementación de pruebas unitarias, de integración y end-to-end para reducir errores en producción.
3. **Pruebas de Carga y Estrés:** Simulación de múltiples usuarios para validar la capacidad y estabilidad del sistema.
4. **Análisis de Seguridad:** Uso de escáneres de vulnerabilidades y pruebas de penetración manuales para detectar riesgos.
5. **Monitoreo en Producción:** Implementación de herramientas de observabilidad para detectar y solucionar problemas en tiempo real.
6. **Encuestas de Percepción:** Uso de cuestionarios para evaluar la satisfacción y usabilidad del software en los equipos de desarrollo.

Metodologías de Desarrollo y Calidad

Se adoptarán metodologías ágiles y marcos de referencia para garantizar un desarrollo iterativo y de calidad:

Metodología	Descripción
Scrum	Se utilizará para la planificación y ejecución de sprints, con reuniones diarias de seguimiento.
Kanban	Aplicado para la gestión visual del flujo de trabajo en el equipo de desarrollo.
TDD (Test-Driven Development)	Desarrollo basado en pruebas para asegurar la funcionalidad desde el inicio.
CI/CD (Integración y Despliegue Continuo)	Implementación de pipelines automatizados para pruebas y despliegues frecuentes.
ISO/IEC 25010	Aplicado para medir la calidad del software en términos de funcionalidad, confiabilidad y usabilidad.

10. Gestión de riesgos

En esta sección se describen los métodos y procedimientos utilizados para la identificación, monitoreo y control de los riesgos durante el desarrollo del proyecto de la aplicación móvil para métricas de productividad en Creatibot.

Los riesgos identificados, así como sus estrategias de mitigación, monitoreo y planes de contingencia, estarán documentados en el **Documento de Gestión de Riesgos**, al cual se podrá hacer referencia para un análisis más detallado.

Métodos y Procedimientos para la Gestión de Riesgos

Para la gestión de riesgos se seguirá el **ciclo de vida de gestión de riesgos** basado en las mejores prácticas del **PMBOK (Project Management Body of Knowledge)** y la **norma ISO 31000**, que incluye:

1. **Identificación de Riesgos:** Se detectan posibles eventos que puedan afectar negativamente el desarrollo del software.
2. **Análisis y Evaluación de Riesgos:** Se determinan la probabilidad y el impacto de cada riesgo identificado.
3. **Planificación de Respuesta:** Se establecen estrategias de mitigación y contingencia para cada riesgo.
4. **Monitoreo y Control de Riesgos:** Se supervisan los riesgos durante todo el ciclo de vida del proyecto.

Para documentar los riesgos, se utilizará una **Matriz de Riesgos**, la cual incluirá:

- **Descripción del riesgo**
- **Categoría del riesgo (técnico, operativo, financiero, organizacional, externo, etc.)**
- **Probabilidad de ocurrencia (baja, media, alta)**
- **Impacto en el proyecto (leve, moderado, crítico)**
- **Estrategia de mitigación**
- **Plan de contingencia**

Principales Riesgos Identificados

A continuación, se listan algunos de los riesgos identificados junto con sus estrategias de mitigación y contingencia:

ID	Riesgo Identificado	Probabilidad	Impacto	Estrategia de Mitigación	Plan de Contingencia
R1	Retrasos en el desarrollo por falta de recursos humanos	Media	Alto	Planificación detallada, asignación eficiente de recursos, y posible contratación de personal externo.	Reasignación de tareas y ajuste en cronograma.
R2	Errores críticos en el software en fase de producción	Baja	Crítico	Pruebas automatizadas y manuales, uso de integración continua.	Implementación de parches rápidos y rollbacks en caso de fallos graves.
R3	Falta de adopción del sistema por parte del equipo de desarrollo de Creatibot	Media	Alto	Capacitación y sesiones de inducción para los usuarios.	Soporte técnico y mejora en la experiencia de usuario.
R4	Pérdida de datos por fallos en el servidor o almacenamiento	Baja	Crítico	Implementación de backups automáticos y redundancia en la infraestructura.	Restauración de datos desde copias de seguridad y análisis post-mortem.

Monitoreo y Control de Riesgos

El monitoreo de riesgos se realizará mediante las siguientes estrategias:

- **Reuniones de seguimiento semanales** para revisar el estado de los riesgos y aplicar medidas correctivas.
- **Uso de herramientas de gestión de riesgos** como **JIRA, Notion o Trello**, donde cada riesgo se registrará y actualizará periódicamente.
- **Indicadores de riesgo**, como el número de incidencias detectadas, retrasos en tareas clave y métricas de seguridad.
- **Revisión y actualización periódica de la Matriz de Riesgos**, asegurando su alineación con el estado real del proyecto.

Referencia al Documento de Gestión de Riesgos

Para un análisis detallado de cada riesgo, sus estrategias de mitigación, planes de contingencia y responsables, se podrá consultar el **Documento de Gestión de Riesgos**, el cual se actualizará de manera continua durante el desarrollo del proyecto.

Este enfoque permitirá minimizar los impactos negativos en el desarrollo del software, garantizando la continuidad del proyecto y la entrega de un producto de calidad en Creatibot.