PLAN DE ASEGURAMIENTO DE CALIDAD

25 DE ABRIL DE 2016

Kwan Tecnología

Calle 45 N. 351 Emiliano Zapata Oriente

C.P. 97144

01 999 988 9298

**PREFACIO**

Este documento contiene el Plan de Aseguramiento de la Calidad (SQA, por sus siglas en inglés) para la empresa Kwan. Este documento ha sido adaptado del documento SQA Plan Template, TM-SQA-01, v2.0.

**REGISTRO DE CAMBIOS**

**\*A** – AÑADIDO **M** – MODIFICADO **D** – ELIMINADO

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N. DE VERSIÓN | FECHA | N. DE FIGURA, TABLA O PÁRRAFO | **A\*** **M** **D** | TÍTULO O DESCRIPCIÓN BREVE | N. DE PETICIÓN DE CAMBIO |
|  |  |  |  |  |  |
| 001 |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**TABLA DE CONTENIDOS**

**Sección**

[SECCIÓN 1. PROPÓSITO 5](#_Toc449540088)

[1.1 Alcance 5](#_Toc449540089)

[1.2 Lista de acrónimos y definiciones 5](#_Toc449540090)

[1.3 Vista general del documento 5](#_Toc449540091)

[1.4 Documentos de referencia 6](#_Toc449540092)

[SECCIÓN 2. GESTIÓN 7](#_Toc449540093)

[2.1 Organización 7](#_Toc449540094)

[2.2 Recursos 8](#_Toc449540095)

[2.3 Procesos 8](#_Toc449540096)

[SECCIÓN 3. TAREAS DE SQA 14](#_Toc449540097)

[3.1 Tarea: 14](#_Toc449540098)

[SECCIÓN 4. ESTÁNDARES, PRÁCTICAS, CONVENCIONES Y MÉTRICAS 15](#_Toc449540099)

[4.1 Métricas 15](#_Toc449540100)

[SECCIÓN 5. PRUEBAS 16](#_Toc449540101)

[SECCIÓN 6. REPORTE DE PROBLEMAS Y RESOLUCIÓN 17](#_Toc449540102)

[6.1 Reporte del proceso de auditoría 17](#_Toc449540103)

[SECCION 7. HERRAMIENTAS, TECNICAS Y METODOLOGIA. 20](#_Toc449540104)

[7.1 Herramientas 20](#_Toc449540105)

[7.2 Técnicas 20](#_Toc449540106)

[SECCIÓN 8. CONTROL DE CÓDIGO FUENTE 21](#_Toc449540107)

SECCIÓN 1. PROPÓSITO

El propósito de este plan es definir la organización de Aseguramiento de la Calidad de Software (SQA, por sus siglas en inglés) para la empresa Kwan, las tareas y responsabilidades de SQA; proveer documentos de referencia y guías para realizar las actividades de SQA; proveer estándares, prácticas y convenciones utilizadas al realizar las actividades de SQA; y proveer las herramientas, técnicas y metodologías para necesarias para apoyar a las actividades de SQA, con base al IEEE Std. 730-1998, Standard for Software Quality Assurance Plans [1].

1.1 Alcance

Este plan establece las actividades de SQA realizadas durante el ciclo de vida de los proyectos de Kwan, enfocándose en las etapas de codificación y pruebas. No estando contempladas, por salirse del alcance del proyecto, las etapas relacionadas con el desarrollo de requisitos, diseño y el mantenimiento del producto de software.

El objetivo del plan SQA es verificar que todo el software y documentación entregable cumpla con los requisitos técnicos establecidos.

1.2 Lista de acrónimos y definiciones

A continuación, se muestra una lista de los acrónimos y definiciones usadas en este documento:

* **Back-end:** Parte del software que procesa la entrada desde el front-end.
* **Bug:** Error o fallo en un programa de computador o sistema de software que desencadena un resultado indeseado.
* **Framework:** Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software.
* **Front-end:** Parte del software que interactúa con el o los usuarios.
* **GP:** Gestión del Proyecto.
* **SQA:** Aseguramiento de la Calidad del Software.

1.3 Vista general del documento

Este documento identifica la organización y procedimientos a ser usados para ejecutar las actividades relacionadas al plan de SQA como se especifica en IEEE Std. 730-1998, IEEE Standard for Software Quality Assurance Plans, referencia (1) y el IEEE Std 730.1-1995, IEEE Guide for SQA Planning, referencia (2). A continuación, se muestra una vista general de las secciones del documento.

La sección 1 describe el propósito y contenido del plan SQA, las relaciones del plan con otros planes y lista todos los acrónimos y documentos referenciados en este plan.

La sección 2 describe cada elemento de la organización que influye en la calidad del software.

La sección 3 describe las tareas de SQA.

La sección 4 identifica los estándares, prácticas y convenciones.

La sección 5 describe la involucración del SQA en las pruebas.

La sección 6 describe el reporte de problemas y las acciones correctivas.

La sección 7 describe herramientas de SQA, técnicas y metodologías.

La sección 8 describe la herramienta de manejo de configuración usada para el control de código.

1.4 Documentos de referencia

[1] IEEE Std 730-1998, IEEE Standard for Software Quality Assurance Planning.

[2] IEEE Std 730.1-1995, IEEE Guide for Software Quality Assurance Planning.

[3] Modelo McCall – Criterios de calidad

[4] Modelo ISO-9126 – Criterios de calidad

[5] Modelo ISO-25000 – Criterios de calidad

[6] IEEE Std 1016 -1998 – Prácticas recomendadas para el diseño de software.

[7] IEEE Std 1008 - 1987 – Estándar para pruebas unitarias de software.

[8] IEEE Std 1061 - 1998 – Estándar para una metodología para la calidad de software.

[9] Kwan Tec – Plan de desarrollo de software.

[10] IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications

[11] Jira – <https://es.atlassian.com/software/jira> – Herramienta utilizada para el seguimiento de proyectos de equipo de software.

[12] Mocha – <https://mochajs.org/> – Framework de pruebas para JavaScript.

[13] Cordova – <https://cordova.apache.org/> – Framework para el desarrollo de aplicaciones móviles.

[14] Protractor – [http://angular.github.io/protractor/#/](http://angular.github.io/protractor/) – Framework de pruebas end-to-end para aplicaciones en AngularJS.

[15] Karma – <https://karma-runner.github.io/0.13/index.html> – Corredor de pruebas para JavaScript.

SECCIÓN 2. GESTIÓN

Esta sección describe los elementos de la organización que tienen influencia en la calidad de software.

2.1 Organización

El encargado del área de gestión de calidad en el proyecto es el responsable de realizar la gestión que asegura que el proceso establecido sea realmente implementado y que los productos de ese proceso cumplan con los criterios de calidad establecidos en este plan. Las disciplinas de gestión brindan soporte a las disciplinas básicas (Requerimientos, Análisis, Diseño, Implementación, Implantación y Verificación) y se realizan en forma paralela a ellas.

La Figura 2-1 muestra la organización del equipo de desarrollo.

Figura 2-1. Organización del equipo de desarrollo

A continuación, se describen los grupos que influyen y controlan la calidad de software con sus respectivas responsabilidades.

1. Líder del proyecto es responsable de:
   * Informar al gerente del proyecto.
   * Administrar al equipo de trabajo.
   * Asignar las tareas.
   * Revisar productos.
   * Revisar procesos que se consideren relevantes para la empresa.
   * Manipulación de las bases de datos.
   * Configuración de los servidores locales.
2. Responsable de requisitos es responsable de:
   * Analizar las necesidades del cliente.
   * Identificar las posibles mejoras y cambios al producto, así como la agregación de nuevas características.
   * Dividir los requisitos en iteraciones.
   * Apoyar al líder del proyecto en la asignación de tareas en la herramienta Jira.
3. Líder de front-end es responsable de:
   * Diseñar los aspectos gráficos de las páginas o aplicaciones tales como bosquejos y prototipos.
   * Aceptar o rechazar los módulos o avances que entreguen los desarrolladores de front-end.
   * Verificar aspectos básicos de un código mantenible tales como nombres correctos de variables, indentación y procedimientos en los métodos que soportan el nombre de estos para front-end.
4. Líder de back-end es responsable de:
   * Mantener las bases de datos funcionales.
   * Aceptar o rechazar los módulos o avances que entreguen los desarrolladores de back-end.
   * Verificar aspectos básicos de un código mantenible tales como nombres correctos de variables, indentación y procedimientos en los métodos que soportan el nombre de estos para back-end.
5. Desarrollador es responsable de:
   * Codificar las características específicas del área designada por el líder del proyecto.
   * Probar su propio código según sus propios criterios.
6. Responsable de pruebas es responsable de:
   * Diseñar el plan de pruebas.
   * Define patrones de aceptación.
   * Define casos de prueba.

2.2 Recursos

**2.2 Instalaciones y equipamiento**

El equipo de SQA tendrá acceso a las instalaciones y equipos definidos continuación para realizar funciones tales como evaluar los productos software, documentación y realizar las revisiones:

* 7 Monitores
* 3 Servidores
* Conexión a internet (inalámbrico o cableado)
* Fuente de corriente eléctrica
* Lápices
* Hojas de papel

**2.2.2 Personal**

El personal está familiarizado y es capaz de aplicar los estándares y pautas definidas en este documento, lo que implica el conocimiento previo de este plan. El personal también está familiarizado con los planes de desarrollo de software, plan de administración de la configuración, así como con las actividades relacionadas con el desarrollo de software, de diseño, codificación, pruebas y requerimientos. Es imprescindible que el personal tenga una cultura de trabajo en equipo. El personal requerido para las actividades de SQA son los siguientes:

* 1 Líder de proyecto
* 1 Responsable de requisitos
* 1 Líder de front-end
* 1 Líder de back-end
* 3 Desarrolladores
* 1 Responsable de pruebas
* 1 Gerente del proyecto

2.3 Procesos

/// Introducción

**2.3.1 Obtención de requisitos**

Propósito: Obtener posibles funcionalidades del sistema software.

Roles involucrados:

* Gerente del proyecto.

Actividades:

1. El gerente del proyecto contacta con el cliente vía correo electrónico para determinar fecha y lugar en el cual se reunirán.
2. El gerente del proyecto y el cliente se reúnen en la fecha y lugar acordados para dialogar las características del software.
3. El gerente del proyecto escucha las peticiones del cliente y toma apuntes en notas durante la plática.

Criterios de salida:

* Lista de posibles requisitos.

**2.3.2 Consolidación de requisitos**

Propósito: Definir las funcionalidades que tendrá el sistema de software.

Roles involucrados:

* Gerente del proyecto.

Herramientas utilizadas:

* Jira.

Criterios de entrada:

* Lista de posibles requisitos.

Actividades:

1. El gerente de proyecto analiza las peticiones por parte del cliente con base en las notas previas y experiencia para determinar la lista de requisitos.
2. El gerente de proyecto prioriza la lista de requisitos resultantes de acuerdo a tres criterios establecidos con base en su experiencia:
   1. Estabilidad del sistema.
   2. Características que permitan la adopción rápida del sistema.
   3. Características que faciliten la expansión del sistema.
3. El gerente de proyecto crea un ticket por cada requisito en la herramienta Jira, incluyendo las tareas involucradas para cada requisito, el tiempo necesario y la prioridad con base en su experiencia.

Criterios de salida:

* Lista de posibles requisitos.
* Tickets en Jira.

**2.3.3 Asignación de tareas**

Propósito: Asignar tickets con las tareas a cada desarrollador.

Roles involucrados:

* Gerente del proyecto.
* Desarrollador (back-end o front-end).

Herramientas utilizadas:

* Jira.

Criterios de entrada:

* Tickets con las tareas a realizar hechos en Jira.

Actividades:

1. El gerente del proyecto revisa los tickets en Jira para determinar a qué área pertenece el requisito (back-end o front-end).
2. El gerente del proyecto asigna las tareas en base a su experiencia y a las capacidades de cada desarrollador. Esta actividad es realizada en una junta semanal.
3. El desarrollador revisa las tareas de los tickets que se le hayan asignado. De tener duda, se consultará con el gerente del proyecto.
4. El gerente del proyecto junto con los desarrolladores involucrados acuerda los tiempos para la finalización y el alcance del requisito.

Criterios de salida:

* Lista de requisitos actualizada.

**2.3.4 Análisis de tareas**

Propósito: Identificar características a codificar para definir un orden en el cual serán desarrolladas.

Roles involucrados:

* Desarrollador (back-end o front-end).

Herramientas utilizadas:

* Jira.

Criterios de entrada:

* Tickets con las tareas a realizar hechos en Jira.
* Lista de asignación\*.

Actividades:

1. El desarrollador identifica cuáles son las tareas que se le han asignado
2. El desarrollador ordena las tareas que se le han asignado de acuerdo a la prioridad que se le ha asignado en el ticket.
3. El desarrollador crea una lista con el orden en el que se van a desarrollar las características.

Criterios de salida:

* Lista de características a desarrollar con prioridad asignada de acuerdo a cada desarrollador.

**2.3.5 Codificación**

Propósito: Desarrollar características identificadas usando los lenguajes de programación asignados según el área (back-end, front-end).

Roles involucrados:

* Desarrollador (back-end o front-end).
* Líder del proyecto.

Herramientas utilizadas:

* IDE.
* Frameworks.
* GitHub.

Criterios de entrada:

* Tickets con las tareas a realizar hechos en Jira.
* Lista de características a desarrollar con prioridad asignada de acuerdo a cada desarrollador.

Actividades:

1. El desarrollador crea módulos atómicos, de forma que un módulo sea una funcionalidad.
   1. Dudas generadas durante el desarrollo se solucionan con el líder del proyecto por medio de mensajes en la herramienta Slack o por correo electrónico.
2. El desarrollador al terminar un módulo procede a realizar las pruebas correspondientes.
3. El desarrollador marca el ticket como completado y actualiza su rama de codificación.
4. El desarrollador comunica al líder de su área el avance del proyecto.
5. El líder del proyecto valida los avances y actualiza la rama principal.

Criterios de salida:

* Tickets marcados como completados.
* Tickets marcados como incompletos.
* Módulos codificados y verificados.

**2.3.6 Pruebas (back-end)**

Propósito: Verificación del código desarrollado.

Roles involucrados:

* Tester.

Herramientas utilizadas:

* Mocha.

Criterios de entrada:

* Módulos o funcionalidades ya codificadas.
* Los tickets con las actividades a realizar que son definidas en la herramienta Jira.

Actividades:

1. El desarrollador analiza los requisitos con base a la complejidad de estos para determinar qué aspectos se van a probar.
   1. Los aspectos a tener en cuenta son: elementos de todo el proyecto involucrados, prioridad del requisito, numero de características por requisito.
2. El tester comienza a realizar las pruebas con ayuda de los frameworks.
3. El tester analiza el resultado de la prueba.
4. El tester actualiza los repositorios con los nuevos módulos o funcionalides.

Criterios de salida:

* Prototipo funcional.

**2.3.7 Pruebas (front-end)**

Propósito: Verificación del código desarrollado.

Roles involucrados:

* Tester

Herramientas utilizadas:

* Protractor.
* Karma.

Criterios de entrada:

* Módulos o funcionalidades ya codificadas.
* Los tickets con las actividades a realizar que son definidas en la herramienta Jira.

Actividades:

1. El tester analiza los requisitos con base a la complejidad de estos para determinar qué aspectos se van a probar.
   1. Los aspectos a tener en cuenta son: elementos de todo el proyecto involucrados, prioridad del requisito, numero de características por requisito.
2. El tester comienza a realizar las pruebas con ayuda de los frameworks.
3. El tester analiza el resultado de la prueba.
4. El tester actualiza los repositorios con los nuevos módulos o funcionalides.

Criterios de salida:

* Prototipo funcional.

**2.3.8 Verificación del estado del proyecto**

Propósito: Verificación de los avances realizados.

Roles involucrados:

* Gestor del proyecto.

Criterios de entrada:

* Tickets con las actividades a realizar hechos en Jira.
* Lista de asignación.
* Código desarrollado.
* Prototipo funcional.

Actividades:

1. El GP analiza a que desarrollador se le asignó cada funcionalidad.
2. El GP determina con base en los tickets y en el código entregado si se cumplieron las metas de la iteración.
3. El GP determina:
4. El GP identifica anomalías y decide su nivel de estado crítico.
5. El GP asigna las nuevas tareas con base en los resultados mostrados.

Criterios de salida:

* Lista de características a desarrollar.
* El producto de software validado.
* Lista de errores o situaciones sin resolver.

SECCIÓN 3. TAREAS DE SQA

Las tareas de SQA se realizan con relación a qué actividades de desarrollo de software se estén llevando a cabo. Una o más tareas de SQA pueden realizarse concurrentemente hasta que una tarea haya sido completada. Se le considera completa a una tarea cuando los reportes requeridos hayan sido satisfactoriamente completados o los ítems de acción hayan sido cerrados. Las siguientes tareas se realizarán como parte del rol de SQA y requerirán de la coordinación y cooperación de la mayoría del equipo.

3.1 Tarea:

**3. n TAREA: REVISIONES Y AUDITORIAS**

En esta sección se recomienda el uso de personal externo (Software Testing and Continuous Quality Improvement 2nd Edition by William E. Lewis - Auerbach Publications), este personal verificara el estado del proyecto, el progreso y los problemas en el proyecto,

Debido a que la función del SQA es parte integral del éxito del proyecto, el personal comunicara libremente sus resultados al Gerente del proyecto y el equipo del proyecto. El método para la presentación de informes de cumplimiento y las áreas problemáticas se comunicarán en un informe documentando el área problemática o la falta de cumplimiento.

SECCIÓN 4. ESTÁNDARES, PRÁCTICAS, CONVENCIONES Y MÉTRICAS

Para verificar la entrega de un producto de alta calidad, todo individuo asignado al proyecto participará en el aseguramiento de la calidad. La referencia [ ] define los procedimientos mediante los cuales el equipo de desarrollo verificará la calidad del producto durante el proceso de desarrollo. El resto de esta sección describe los procedimientos usados por SQA para verificar que las actividades de aseguramiento de la calidad de este plan y los estándares, prácticas, convenciones y métricas sean cumplidas.

// Realizar plan de codificación

4.1 Métricas

Las siguientes mediciones se harán y se utilizarán para determinar la calendarización de las actividades a lo largo del proyecto.

// TODO

SECCIÓN 5. PRUEBAS

Las actividades de pruebas realizadas en los proyectos de Kwan incluyen:

1. Pruebas unitarias
2. Pruebas de caja negra
3. Pruebas end-to-end
4. Pruebas de integración

El líder del proyecto designará a una persona como el responsable de pruebas, quien se encargará de elaborar un plan de pruebas para el proyecto que se desarrollará.

Todo personal que desarrolle algún ticket del proyecto realizará las siguientes actividades de pruebas necesarias del software:

* Realizar las pruebas unitarias.
* Realizar las pruebas de acuerdo al plan de pruebas.
* Documentar los resultados de las pruebas.
* Recomendar acciones correctivas si se encontraron defectos en los módulos probados.
* Enviar los reportes de las pruebas al responsable de pruebas.

La Figura 6-1 ilustra el Flujo del Proceso de Pruebas. Forma parte de las responsabilidades de SQA verificar que los resultados de las pruebas sean registrados y evaluados.

SECCIÓN 6. REPORTE DE PROBLEMAS Y RESOLUCIÓN

En esta sección se describe el reporte y control del sistema utilizado por el personal de calidad para registrar y analizar las discrepancias encontradas, así como para monitorear la implementación de las acciones correctivas. Los formatos (ISO 19011, auditoria interna) utilizados para realizar los reportes se describen más adelante en esta sección.

6.1 Reporte del proceso de auditoría

El personal de calidad reportará el resultado de las auditorias y las recomendaciones proporcionadas. Este reporte se usa para asegurarse que el proceso:

1. Se está siguiendo de manera correcta y se está trabajando de forma efectiva

2. Se está siguiendo, pero no se está trabajando de manera efectiva

3. No se está siguiendo

**6.1.1 Presentación del reporte del proceso de auditoria**

El proceso de Reporte de Auditorias está dirigido hacia el Líder del Proyecto el cual utilizara los reportes de las siguientes maneras:

1. Para saber si los procesos de desarrollo son acatados y si son efectivos para el cumplimiento de las metas del proyecto. Cuando sea necesario el Líder del proyecto puede iniciar cambios a los procesos, mediante los procedimientos establecidos, para que los procesos queden estables.

2. Para indicar el acuerdo, desacuerdo, o el aplazamiento de las recomendaciones hechas en el Proceso de Reporte de Auditoría. En caso de que el Líder del proyecto indica desacuerdo con las recomendaciones registradas en el proceso de reporte de auditoría, la disposición final de recomendaciones del informe se hace por el Gerente del proyecto.

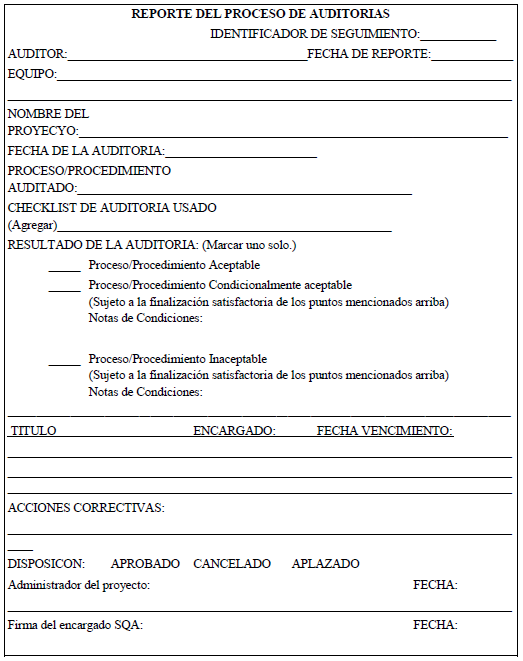


Figura 7-1. Reporte del proceso de Auditorias

**6.1.2 Procedimiento de Escalamiento para la Resolución de no concurrencia en el Proceso de reporte de Auditoría**

Al encontrase un problema de calidad en algún elemento de trabajo ya sea documento, código o producto de software se tendrá que realizar lo siguiente:

1. Primero se tratará con el creador de ese elemento,

a. Si existen problemas de desacuerdos en la resolución de del problema, el personal de calidad tendrá que notificar al Líder del proyecto para que este tome cartas en el asunto y de una solución al problema.

b. Si el Líder del proyecto no da una solución, se escalará el problema al Gerente el proyecto y él tomará la decisión final.

El personal de calidad conservará el registro original de las conclusiones tomadas y la resolución posterior en sus expedientes de auditoría.

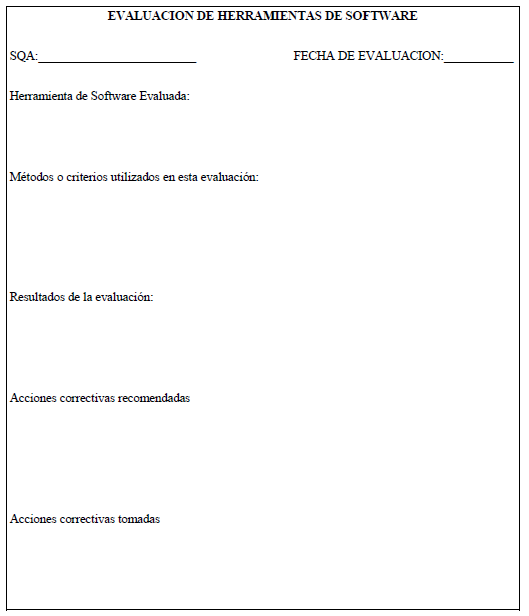


Figura 7-2. Evaluación de herramientas de software

SECCION 7. HERRAMIENTAS, TECNICAS Y METODOLOGIA.

//Utilidades del sistema operativo, depuradores, documentos de ayuda, checklist, analizadores de estructuras, analizadores de código, auditorias de estándares, monitoreo de rendimiento, software de desarrollo, matrices de seguimiento.

7.1 Herramientas

Utilidades del sistema operativo, depuradores, documentos de ayuda, checklist, analizadores de estructuras, analizadores de código, auditorias de estándares, monitoreo de rendimiento, software de desarrollo, matrices de seguimiento de software, pruebas de generadores de casos.

Lenguajes: PHP, HTML5, CSS, JavaScript, XML, Java, J2EE.

Herramientas de diagramas UML: Astah.

Herramientas de bases de datos: MYSQL, MongoDB .

Herramienta de procesamiento de texto: Microsoft Word, Notepad++

Herramientas de apoyo: Internet, AngularJS, Ant, Maven, Gradle.

Herramientas de desarrollo: NetBeans, Dreamweaver, IntelliJ IDEA, Eclipse, SWT, JFace, EMF, GEF, GMF.

7.2 Técnicas

En las técnicas se incluirá la revisión de uso de estándares, inspecciones de software, rastreo de requerimientos, verificación y validación de diseño y requerimientos, mediciones y evaluaciones de fiabilidad, análisis de lógica de negocio.

Estándares: Codificación de Lenguajes, UML, Diseño de BD Lógicas y Físicas

Programación en Pares.

Programación orientada a objetos y Programación orientada a eventos.

7.3 Metodologías

Estas son un grupo de técnicas y herramientas. Estas metodologías se deben de documentar para completar la tarea o actividad y proporcionar una descripción del proceso que se va a usar

Paradigma de programación: Orientado a Objetos y Orientado a Eventos.

SECCIÓN 8. CONTROL DE CÓDIGO FUENTE

El propósito de esta sección es definir los métodos e instalaciones usadas para mantener, almacenar, resguardar y documentar las versiones del código fuente identificado durante todas las fases del proceso de ciclo de vida software. Se utilizará la herramienta Git.

**Para el correcto control del código fuente se tendrán que considerar los siguientes aspectos:**

Las leyendas que se utilizarán para las ramas serán las siguientes:

|  |  |  |
| --- | --- | --- |
| **Instancia** | **Rama** | **Descripción, Instrucciones, Notas** |
| Estable | Stable | Acepta la unión de las ramas "Trabajando" y "Revisión". |
| Trabajando | Master | Acepta la unión de las ramas "Características/Problemas" y "Revisión". |
| Características / Problemas | Topic-\* | Siempre deriva de la rama "Trabajando". |
| Revisión | Hotfix-\* | Siempre deriva de la rama "Estable". |

Figura \*-\*. Tabla de definición de ramas

**Ramas principales**

En el proyecto software siempre deberán existir al menos dos ramas, la rama principal denominada “Master” que contendrá la última versión y la rama secundaria denominada “Stable” que contendrá la última versión que es estable, es decir que todas sus características han sido probadas y verificadas. La rama “Stable” no deberá ser manipulada hasta que el código sea verificado en la revisión técnica.

**Ramas de soporte**

En el proyecto las ramas de soporte son usadas como auxiliar en el desarrollo paralelo entre los desarrolladores, facilitar el rastreo de características y asistir en la reparación de defectos. Estas ramas deberán tener un tiempo de vida corto ya que se eliminarán eventualmente, el tiempo de vida dependerá de que tipo de rama sea y las características involucradas.

Los tipos de ramas que se usaran son:

* Rama de características
* Rama de bug
* Rama de Revisión

*Rama de características*

Estas ramas son usadas cuando se desarrolla una nueva característica, el tiempo que tardara en lanzarse la característica dependerá de las características de esta. Sin importar cuando se finalice la rama, está siempre hará unión con la rama master.

* Debe derivar de master
* Debe unirse con master
* La convención del nombre será: feature-<#identificador de la característica dado en Jira>

*Rama de bug*

La rama de bug difiere de las demás ramas semánticamente. Estas ramas solo deben ser creadas cuando exista un bug en un módulo que deba ser reparado y unido en la siguiente iteración, por esta razón la vida de una rama de bug no debe ser mayor al tiempo de la iteración. Sin importar cuando se finalice la rama, está siempre hará unión con la rama master

* Debe derivar de master
* Debe unirse con master
* La convención del nombre será: bug-<# número de bug identificado en el proceso de desarrollo>

*Rama de revisión*

Una rama de revisión proviene de la necesidad de actuar de inmediato mediante un estado no deseado de una modulo en desarrollo. Además, debido a la urgencia, no se requiere una revisión para ser integrado durante un la siguiente iteracion. Debido a estos requisitos, una rama de revisiones siempre debe derivar a partir de una rama estable. Esto por dos razones

* El desarrollo de la rama master puede continuar mientras la revisión está siendo dirigida.
* Una rama estable representa lo que aún está en producción