Tecnológico de Monterrey
Planeación de sistemas de software (Gpo 101)

Plan de Calidad
Avance para ajustar la estrategia

Sofía Cantú Talamantes	A01571120
Ozner Axel Leyva Mariscal	A01742377
Carolina Nicole González Leal	A01284948
Roberto José García Ríos	A01284731
Diego Gutiérrez Treviño	A01284841

15-03-2024

Índice

1. Introducción

Este documento presenta el Plan de Calidad para el proyecto KnowX. El propósito del presente plan es definir los productos, prácticas y acuerdos bajo los cuales se llevarán a cabo las actividades de aseguramiento de calidad en el proyecto. Este plan tiene como alcance todos los productos, subproductos y procesos que puedan ser generados durante la ejecución del mismo.

2. Roles y responsables del plan de calidad

Los roles y responsabilidades relacionados al proceso de calidad dentro del desarrollo del proyecto se presentan en la siguiente tabla.

Roles	**Responsabilidad**	**Colaborador (Quienes)**
Desarrollador	Cumplir con las historias de usuario asignadas. Cumplir en tiempo y forma para cada sprint y entrega. Cada tarea necesita una prueba unitaria como mínimo. Cada historia de usuario necesita una prueba por pares como mínimo.	**Diego** **Roberto** **Carolina** **Sofia** **Ozner**
Code Review	Cada PR será aprobado por lo menos por un miembro del equipo, y se debe registrar quién lo aprobó. Antes de hacer merge se debe de revisar por dos personas.	**Diego** **Roberto** **Carolina** **Sofia** **Ozner**
Auditor	Una vez realizada la documentación y/o el proyecto revisará que esté todo correcto. Se tendrán reuniones con los desarrolladores, para revisar que se cumpla con los requerimientos.	**Profesores**

Revisor	Al final de cada sprint, entrará a la versión actual y correrá los tests definidos para asegurarse que la demo a mostrar es viable.	**Sofía**
Quality Manager	Se encarga de asegurar la calidad del sistema, haciendo pruebas regulares y siguiendo a los desarrolladores y los cambios realizados durante el sprint.	**Diego**

3. Estándares de calidad aplicables

A continuación se presentan los estándares de los cuales se va a apegar el proyecto a desarrollar.

Tipo de Estándar	Estándar	Productos de Ingeniería de software a los que se aplicará
Estándar de especificación de requerimientos	**IEEE830 y IEEE29148**	**Documento de Requerimientos de Software**
Estándar de procesos de ciclo de vida del software	**ISO/IEC 12207**	**Documento de Procesos de Ciclo de Vida del Software**
Estándar de calidad del producto software y la experiencia del usuario	**ISO/IEC 25010**	**Documento de Calidad del Producto de Software y la Experiencia del Usuario**
Estándar de diseño	**IEEE1016**	**Documento de Diseño y Arquitectura**

Accesibilidad web	**WCAG 2.1:**	**Documento para la Accesibilidad considerada en el Diseño Web.**
Estándar de documentación de pruebas	**IEEE29119-3**	**Documento de Plan de Calidad y Casos de Prueba**

4. Prácticas de calidad

A continuación se describen aquellas prácticas y acciones que serán aplicadas en el desarrollo de los productos involucrados con la finalidad de alcanzar los estándares de calidad esperados del proyecto.

Prácticas	**Descripción y Alcance**
Revisión individual	Todo producto y subproducto proveniente del desarrollo del proyecto debe de ser revisado por lo menos por el Quality Manager y Revisor, antes de ser integrado a la solución principal. Las integraciones por parte de los desarrolladores serán realizadas por funcionalidades de pull requests en GitHub, por lo que serán ejecutadas asincrónicamente. En caso de requerir una revisión particular, se organizan sesiones de máximo 20 minutos de revisión por funcionalidad para garantizar la productividad en el desarrollo del código. Las sesiones deben de realizarse entre por lo menos tres miembros del equipo, para garantizar la calidad del entregable. Code reviews - Se agrega una pequeña descripción con los componentes generados y/o cambios hechos. - Se hará revisión con tests generados con la herramienta CYPRESS. - Hasta que todas las pruebas sean exitosas, se podrá hacer push hacía la rama principal del proyecto con autorización de otro miembro del equipo.

Revisión por pares	Todas las funcionalidades que impliquen la integración de múltiples componentes deben de ser realizadas y revisadas en pares de desarrolladores. Ellos también se encargarán de garantizar la resolución de cualquier conflicto de unión que pueda surgir en un push a una rama.
	
	Code reviews
	
	- Cada historia de usuario debe de generar un branch
	  - Los nombres de branch serán: H001/DescripcionDeFuncionAAgregar
	  - En caso de ser necesario crear una sub-rama para un criterio de aceptación en particular, la nomenclatura será: CA01/DescripcionDeCriterio
	  - Cada commit realizado sobre las branches, deberán contener una explicación concisa de lo realizado.
	  - Al hacer un pull request, se debe de revisar que no haya errores por dos o más miembros del equipo
	  - Antes de hacer pull request, todas las pruebas unitarias previamente validadas tienen que ser exitosas.
	- Se tendrán que dar validación por dos desarrolladores y se agregará en la descripción los nombres de las personas que revisaron.

Control de Versiones y Ambientes	Toda actualización, inserción o eliminación del código del proyecto debe de realizarse fuera del entorno principal de trabajo, y debe de ser revisado antes de integrarse.
	
	Existen tres ambientes de trabajo:
	
	1) Local: Desarrollo local
	  a) Existirá una branch por cada Historia de usuario generada. Esa branch deberá contar con descripciones de modificación, y branch secundarias para cada feature que se requiera de esa historia de usuario.
	2) Branch Dev: Ambiente de pruebas.
	  a) Para poder hacer merge con las branches principales, se tuvieron que hacer las pruebas de manera individual en cada branch.
	  b) En caso de existir un problema de integración, los conflictos de unión deben de resolverse entre los miembros del equipo cuyas actualizaciones causan disrupción. En caso de que no se resuelva, el Quality Manager se encargará de resolver cada caso.
	3) Branch Main: Ambiente de producción (el más actualizado).

	a) Para poder hacer merge con Main, tendrá que ser autorizado por dos integrantes del equipo, el Quality Manager y Revisor. Se usará el editor de código Visual Studio Code para el desarrollo de la aplicación.
Pruebas	Por cada Historia de Usuario - Debe existir una prueba automatizada de integración. - Puede complementarse con una prueba End-to-End. - Configuración para que se corran las pruebas al hacer un push a la branch destino. - Una descripción del cumplimiento de cada criterio de aceptación. Por cada Historia de Usuario que involucre almacenamiento de datos - Debe existir una prueba demostrando conectividad entre base de datos y servidor. - Comprobar que se están realizando los procesos requeridos en la base de datos (creación, actualización, borrado y lectura) - Registro de los cambios hechos sobre las bases de datos. Para cada Historia de Usuario que involucre el uso del back-end - Debe de existir una prueba comprobando el funcionamiento de la ruta utilizada Los resultados de las pruebas se almacenan en el REPORTE DE PRUEBAS y cualquier error grave detectado se abre un “Issue” en github explicando el problema y la prueba fallida. - El issue debe de mencionar la prueba y la historia de usuario a la que corresponde, además de un camino de prueba que causa el error. Para que una prueba se considere exitosa debe de obtener los resultados esperados estipulados en el documento de CASOS DE PRUEBA

Nomenclatura y Estándares de codificación	Para el desarrollo en Javascript XML (JSX) de la aplicación en Next.js, se seguirá el estándar de JavaScript Standard Style. - Para los nombres de las variables se utilizarán nombres descriptivos. - Las variables estarán escritas en notación camello. - Las variables globales y predefinidas estarán escritas totalmente en mayúsculas. - Las funciones estarán escritas en notación Pascal. - Las funciones contendrán un comentario donde se define el funcionamiento en inglés. - Las variables y funciones estarán en inglés. Para el desarrollo en Python del servicio de backend, se utilizarán los mismos estándares. Cada pull request deberá contener: - Un nombre conformado por su número de HU acompañado con una breve descripción de los cambios implementados. - Un comentario con un desglose de la funcionalidad implementada y las pruebas necesarias para comprobar su ejecución. - Registro de las revisiones realizadas por otros miembros del equipo, garantizando que los cambios sean correctos.

Dentro del calendario de trabajo se deberán incluir las actividades correspondientes a las actividades de calidad definidas para los productos del proyecto y del ciclo. Diariamente, en la reunión interna de avance del proyecto, se deberá validar el trabajo hecho contra los parámetros establecidos para el plan de calidad, y si no se está cumpliendo definir acciones correctivas para lograr la meta comprometida.

Quincenalmente, con las sesiones de avance con los auditores y stakeholders, se harán pruebas de requerimientos, validando que el producto este acorde a la visión establecida. Cada vez que se inicie y se termine el proceso de generación de un producto, el autor debe validar que se esté creando de acuerdo a las prácticas establecidas en esta estrategia.

5. Estrategia de pruebas

Tipos de pruebas consideradas, su alcance y su aplicación:

- Prueba unitaria para cada requerimiento: Se verifica que cada componente del sistema funcione de manera correcta.
- Pruebas de integración (funcionalidad): Se enfocan en probar las interfaces y la comunicación entre los componentes ya probados unitariamente.

- Pruebas end-to-end: Estas pruebas son ejecutadas por el cliente, y prueban todo el flujo de punta a punta del desarrollo.
- Pruebas de aceptación: consisten en garantizar que los requerimientos sean cumplidos.

Tipo de prueba	**Automatización**	**#Ciclos**
Prueba unitaria	Sí	Por integración de código
Prueba de integración	Sí	Cierre de sprints
Prueba end-to-end	Parcialmente	Revisión con cliente
Pruebas de aceptación	No	Cierre de sprint

6. Entregables de calidad

Consideramos que la preparación de los siguientes productos son indispensables para la conclusión del proyecto. Establecemos a continuación el nombre del reporte y la frecuencia en la que se aplicará durante el proyecto.

Entregable	**Información**
Casos de pruebas y reporte de pruebas	• Contendrá los escenarios y casos de prueba a aplicar para cada componente. • Serán aplicables a funcionalidades completas, módulos y producto terminado. • Contendrá tanto el diseño de las pruebas como los resultados esperados y resultados obtenidos.
Reporte de recorrido	• Contendrá los resultados del recorrido que se realice a los funcionalidades seleccionadas

Informe de Rendimiento	• Contendrá un análisis de tiempos de carga y eficiencia del sistema.
informe de Usabilidad	• Contendrá la evaluación de la facilidad de uso y accesibilidad.
Reporte de Inspección	• Contendrá los resultados de la inspección formal que se realice a las funcionalidades seleccionadas.

7. Tipos de severidades

A continuación se identifican los tipos de severidades que puedan tomar en el desarrollo del proyecto, representando una posibilidad de riesgo que impactan y obstaculizan el progreso del mismo. A continuación se presentan las severidades identificadas en el proyecto:

Tipo	Descripción	Descripción detallada
D (💀)	**Desastroso**	Provoca un mal funcionamiento en más de dos funcionalidades claves de la aplicación
G (🔥)	**Grave**	Provoca un mal funcionamiento en una o dos funcionalidades claves de la aplicación
P (🥺)	**Preocupante**	Provoca un mal funcionamiento de una o más secciones no claves de la aplicación.
T (😐)	**Tolerable**	Provoca un mal funcionamiento de una sección no clave de la aplicación.
C (☝🤓)	**Controlable**	Provoca un error que no interfiere con el funcionamiento de la aplicación.

8. Tipos de defectos

En la siguiente tabla se encuentran los tipos de defectos que son posibles a presentarse durante el desarrollo del producto. Los enlistamos de menor a mayor de acuerdo a su impacto en el proyecto. En el tipo 10, el menor, existen los defectos de entendimiento, los cuales fácilmente pueden ser explicados y corregidos. Por otro lado existe el tipo 50, donde sí existe una falla en el cumplimiento de los estándares de requerimientos definidos, el defecto tiene un impacto mucho mayor en el proyecto.

Tipo	Nombre	Descripción
10	**Entendimiento**	No se entendió el requerimiento
20	**Eficiencia**	El código es ineficiente
30	**Casos Especiales**	No se tiene la funcionalidad esperada en casos especiales
40	**Funcionalidad**	No se tiene la funcionalidad esperada
50	**Estándares**	Falla en el cumplimiento de los estándares definidos

9. Métricas del proyecto

A continuación se encuentran los elementos más importantes que se considerarán al avanzar en el proyecto. Cumplir positivamente con ellas es esencial para considerar que el proyecto ha sido un éxito.

Métrica	Consideración

Satisfacción del SocioFormador	• A través de preguntas y análisis de retroalimentación.
Tiempo de Respuesta	• Medición de la velocidad de la aplicación.
Cobertura de Pruebas	• Porcentaje de requerimientos y código cubierto por las pruebas.

10. Casos de prueba y métricas del proyecto

10.1 Contexto de caso de prueba

Se han definido los casos de prueba que definirán si el requerimiento cumple con la función a tener, así mismo, el proceso que implica su ejecución, al igual que el resultado que se espera tener.

10.2 Contexto de métricas del proyecto

El detalle de las métricas definidas para el proyecto, la forma de calcularlas, la frecuencia de su revisión y su valor actual se encuentra en el documento Matriz de Métricas.

10.3 Presentación de información

A través de la siguiente liga, puede acceder a la información antes mencionada:

CasosDePrueba/Métricas_LosCroods

11. Liga al Github

La siguiente liga muestra el repositorio actual del equipo en Github, el cual fue proporcionado por Wizeline:

https://github.com/wizelineacademy/itesm-socioformador-feb-jun-2024-Croods