Tecnológico de Monterrey

Planeación de sistemas de software (Gpo 106)

Curso:

TC3004B.106

Campus:

Monterrey

Equipo Docente:

Jorge Alejandro Álvarez Bujanos
Adan Octavio Ruiz Martínez
Leonardo Salvador Gámez Peña
José Francisco Mendoza Bazán
Jorge Alberto Torres Bautista

Plan de Calidad

Equipo 4:

Eduardo Zentella Castillo	| A00835387
Cesar Ivan Hernandez Melendez	| A00829868
Jose David de la Garza Salas	| A00834760
Pablo Andrés Martínez Sánchez	| A01252489
Javier Eduardo Corrales Cardoza	| A01742328

Lugar y Fecha:

Monterrey, Nuevo León. 10 de Marzo de 2024.

Índice

1. Propósito del documento....3
1.1. Propósito y Alcance....3
1.2. Elementos del software....3
1.3. Proporción del ciclo de vida....3
2. Gestión....4
2.1. Organización....4
2.2. Tareas....4
2.3. Roles y Responsabilidades....4
2.4. Recursos estimados y garantía de calidad....4
3. Documentación....4
3.1. Propósito....4
3.2. Requisitos mínimos....4
3.3. Lista de Documentos a Revisar o Auditados....5
3.4. Criterios de Revisión....5
4. Estándares....5
4.1. Propósito....5
4.2. Estándares de documentación y de pruebas....6
5. Revisiones de Software y pruebas....7
5.1. Propósito....7
5.2. Revisión de software....7
5.3. Pruebas Unitarias....7
5.4. Pruebas de Integración....8
5.5. Pruebas de Sistema....8
5.6. Pruebas de Aceptación....9
6. Informe de Pruebas....10
6.1. Herramientas de informe....10
6.2. Prácticas y Procedimientos de Informe....10
7. Herramientas....10
8. Control de medios....11
9. Gestión de riesgos....11

Plan de aseguramiento de calidad de software

1. Propósito del documento

1.1. Propósito y Alcance

El propósito del plan de calidad es garantizar que el proyecto se desarrolle de manera adecuada y cumpla con los estándares establecidos por el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE), asegurando la calidad en todas las etapas del desarrollo del software. El alcance de este plan se limita únicamente al desarrollo del software, abarcando el frontend de la aplicación web, el servidor y la base de datos en Oracle Cloud Infrastructure.

1.2. Elementos del software

1.2.1. **Frontend de la Aplicación Web**: Desarrollado utilizando tecnologías como React para la interfaz de usuario y Node.js para la lógica del lado del cliente.
1.2.2. **Servidor**: Implementado en Oracle Cloud Infrastructure para garantizar la escalabilidad y disponibilidad del servicio.
1.2.3. **Base de Datos**: Utilizando Oracle Autonomous Database para almacenar y gestionar los datos de manera segura y eficiente.
1.2.4. **Conexión a la API de Telegram**: Integración con la API de Telegram para servicios de mensajería web seguros.

1.3. Proporción del ciclo de vida

El ciclo de vida del proyecto se extiende hasta la completa implementación y entrega del servicio. Se seguirán las siguientes fases del ciclo de vida del software:

1.3.1. **Requisitos**: Análisis y especificación de los requisitos del software.
1.3.2. **Diseño**: Diseño detallado scrum del frontend, servidor, base de datos y la integración con la API de Telegram.
1.3.3. **Implementación**: Desarrollo del software utilizando las tecnologías mencionadas, asegurando la calidad del código y su funcionalidad.
1.3.4. **Pruebas**: Realización de pruebas unitarias, de integración y de sistema para verificar que el software cumple con los requisitos establecidos.
1.3.5. **Despliegue**: Implementación del software en Oracle Cloud Infrastructure y configuración de los servicios necesarios.
1.3.6. **Entrega**: Puesta en producción del servicio y entrega al socio formador.

2. Gestión

2.1. Organización

Oracle México actuará como socio formador en el proyecto, brindando orientación y apoyo en el uso de Oracle Cloud Infrastructure y tecnologías relacionadas.

2.2. Tareas

Las tareas del equipo incluirán la planificación, desarrollo, pruebas y despliegue del software. Se seguirá un enfoque ágil, dividiendo el trabajo en iteraciones cortas y entregando incrementos funcionales de manera regular.

2.3. Roles y Responsabilidades

El equipo de desarrollo estará compuesto por:

Scrum Master (Jose): Liderar el equipo, eliminar obstáculos y asegurar el cumplimiento de los objetivos del proyecto.
Desarrolladores Frontend (XX y XX): Diseñar y desarrollar la interfaz de usuario utilizando tecnologías como React.
Desarrollador Base de Datos (Eduardo): Diseñar y gestionar la estructura de la base de datos en Oracle Autonomous Database, asegurando la integridad y seguridad de los datos.
Desarrollador Backend (XX): Implementar la lógica del servidor utilizando Node.js y gestionar la comunicación con la base de datos.

2.4. Recursos estimados y garantía de calidad

Se cuenta con un presupuesto de 5,000 MXN proporcionado por Oracle, así como herramientas de software gratuitas para el desarrollo del proyecto. Estos recursos serán utilizados para cubrir gastos operativos y cualquier necesidad relacionada con el desarrollo del software. Dado que este proyecto se considera estudiantil, la calidad del producto es crucial para la calificación final. Se garantizará un producto funcional y servible, siguiendo buenas prácticas de desarrollo de software y realizando pruebas exhaustivas para validar su funcionamiento. El cumplimiento de los estándares IEEE y la satisfacción del cliente serán prioritarios en la garantía de calidad del proyecto.

3. Documentación

3.1. Propósito

La documentación durante el proceso de desarrollo tiene como objetivo proporcionar una guía clara y detallada para el desarrollo, verificación, validación y uso del software. Los documentos principales que dirigirán el proceso son el Documento de Especificación de Requisitos de Software (SRS) y el Plan de Proyecto.

3.2. Requisitos mínimos

3.2.1. **Descripción de Requisitos del Software**: Detalla los requisitos funcionales y no funcionales del sistema, incluyendo la pantalla de login, el dashboard para desarrolladores y coordinadores, y la funcionalidad del chatbot.

3.2.2. **Descripción del Diseño de Software**: Describe la arquitectura y diseño del sistema, incluyendo la estructura de la aplicación web, la comunicación con la API y la implementación del chatbot.

3.2.3. **Planes de Verificación y Validación**: Establece los procedimientos y criterios para verificar y validar el software, asegurando que cumpla con los requisitos establecidos. Se incluirán pruebas unitarias, de integración y de sistema.

3.2.4. **Informe de Resultados**: Documenta los resultados de las pruebas realizadas, identificando cualquier problema o error encontrado y las acciones correctivas tomadas.

3.2.5. **Documentación del Usuario**: Proporciona instrucciones detalladas para el uso del software, incluyendo cómo iniciar sesión, navegar por el dashboard y utilizar las funciones del chatbot.

3.2.6. **Plan de Gestión de la Configuración del Software**: Detalla cómo se gestionará la configuración del software, incluyendo control de versiones, gestión de cambios y mantenimiento de la integridad del sistema.

3.3. Lista de Documentos a Revisar o Auditados

- Documento de Especificación de Requisitos de Software (SRS)
- Plan de Proyecto

3.4. Criterios de Revisión

- Los documentos deben estar completos y actualizados según el estado actual del proyecto.
- Los requisitos del software deben ser claros, específicos y coherentes.
- El diseño del software debe ser adecuado para cumplir con los requisitos establecidos.
- Los planes de verificación y validación deben ser exhaustivos y garantizar la calidad del software.
- El informe de resultados debe proporcionar una visión clara del estado y calidad del software, identificando cualquier problema encontrado y las acciones correctivas tomadas.
- La documentación del usuario debe ser clara, concisa y fácil de entender para los usuarios finales.
- El plan de gestión de la configuración del software debe garantizar un control adecuado de la configuración y cambios en el software a lo largo del ciclo de vida del proyecto.

4. Estándares

4.1. Propósito

El propósito de los estándares en el proyecto es establecer prácticas, convenciones y métricas que aseguren la calidad del software y la documentación asociada. Se aplicarán estándares de documentación y de pruebas para garantizar la claridad y eficacia del desarrollo del software.

4.2. Estándares de documentación y de pruebas

4.2.1. **Diseño de Codificación y Comentarios**: Se seguirá el estándar de utilizar variables en formato "lowerCamelCase" para nombres de variables y "UpperCamelCase" para nombres de clases. Los comentarios deberán seguir la convención de incluir el símbolo "#" seguido de un espacio antes del comentario, situado encima de la función, proporcionando una descripción clara de la funcionalidad básica de la misma.

4.2.2. **Prácticas y Estándares de Pruebas**: Se aplicarán prácticas de pruebas tanto dinámicas como estáticas para garantizar la calidad del software. Se realizarán pruebas de caja negra, caja blanca y caja gris para cubrir diferentes aspectos del sistema. Algunos ejemplos de prácticas de pruebas se presentan en la siguiente tabla:

Tipo de Prueba	Descripción
Pruebas Unitarias	Verifican el funcionamiento correcto de unidades individuales de código, como funciones o métodos.
Pruebas de Integración	Verifican que los diferentes componentes del sistema funcionen correctamente juntos.
Pruebas de Sistema	Prueban el sistema en su totalidad para garantizar que cumpla con los requisitos especificados.
Pruebas de Aceptación	Verifican que el sistema cumpla con los requisitos del usuario y esté listo para su despliegue.

4.2.3. **Estándar IEEE 730**: Se seguirá el estándar IEEE 730 para la calidad del software, que proporciona directrices para la planificación, supervisión y control del proceso de aseguramiento de la calidad del software. Este estándar asegura que se apliquen las mejores prácticas en todas las etapas del desarrollo del software para garantizar la calidad del producto final.

4.2.4. **Métricas del Producto y Proceso de Garantía de Calidad Seleccionada**: Se utilizarán métricas específicas del producto y del proceso para evaluar y mejorar la calidad del software. Algunas métricas pueden incluir la cantidad de defectos encontrados durante las pruebas, la cobertura de código alcanzada por las pruebas, el tiempo dedicado a actividades de garantía de calidad, entre otras.

- **Burndown Chart**: El Burndown Chart es una herramienta visual que muestra el progreso del equipo en la finalización de las tareas a lo largo del tiempo. Permite evaluar el ritmo y la productividad del equipo al comparar las tareas planificadas con las completadas. Un descenso constante en la gráfica indica un buen progreso, mientras que un estancamiento podría indicar posibles problemas en el proceso de desarrollo.

- **Cantidad de Defectos**: Esta métrica registra la cantidad de defectos encontrados durante las pruebas del software. Permite identificar áreas

problemáticas en el código y priorizar la resolución de problemas para mejorar la calidad del producto final.

- **Rendimiento del Sistema**: Se pueden medir métricas de rendimiento del sistema, como el tiempo de respuesta de la aplicación, la carga máxima que puede soportar o el consumo de recursos. Estas métricas son importantes para garantizar que el sistema cumpla con los requisitos de rendimiento establecidos.

5. Revisiones de Software y pruebas

5.1. Propósito

El propósito de las revisiones de software y pruebas es asegurar que el software cumpla con los requisitos establecidos y funcione de manera esperada, tanto a nivel funcional como no funcional. Se utilizará el estándar de versionamiento de software (X.X.X), donde la primera X representa la versión mayor, la segunda X la versión menor y la última X para cambios de funcionalidad o corrección de fallos.

5.2. Revisión de software

Verificar que las especificaciones del software aborden todos los requisitos del sistema, incluyendo la compatibilidad con múltiples navegadores para la aplicación web. Asegurarse de que el diseño arquitectónico abarque todos los componentes del sistema, incluyendo la aplicación web, la API y la base de datos, todos alojados en Oracle Cloud Infrastructure. Revisar el diseño detallado para confirmar que cumple con las especificaciones y requisitos del sistema, incluyendo la correcta implementación de la funcionalidad del chatbot y la seguridad de la información.

5.3. Pruebas Unitarias

Función/Método	Caso de Prueba	Resultado Esperado	Resultado Real
Inicio de Sesión	Usuario ingresa credenciales válidas	Acceso concedido al sistema	
	Usuario ingresa credenciales inválidas	Mensaje de error de credenciales incorrectas	
Creación de Tarea	Usuario crea una tarea individual	La tarea se crea correctamente en la base de datos	
Modificación de Tarea	Usuario modifica una tarea individual	La tarea se actualiza correctamente en la base de datos	
Eliminación de Tarea	Usuario elimina una tarea individual	La tarea se elimina correctamente de la base de datos	

Visualización de Tareas	Usuario visualiza sus tareas	Se muestran las tareas del usuario en la interfaz	
Creación de Tarea de Grupo	Coordinador crea una tarea de grupo	La tarea se crea correctamente en la base de datos	
Modificación de Tarea de Grupo	Coordinador modifica una tarea de grupo	La tarea se actualiza correctamente en la base de datos	
Eliminación de Tarea de Grupo	Coordinador elimina una tarea de grupo	La tarea se elimina correctamente de la base de datos	
Comunicación con API	Aplicación web se comunica correctamente con la API	Se reciben y envían datos de manera correcta	
Encriptación de Datos	Datos sensibles se encriptan correctamente	Los datos encriptados son seguros y legibles	

5.4. Pruebas de Integración

Componentes a integrar	Caso de Prueba	Resultado Esperado	Resultado Real
Aplicación Web y API	Aplicación web se comunica correctamente con la API	Integración sin errores y transferencia de datos exitosa	
Aplicación Web y Base de Datos	Aplicación web accede y actualiza datos en la base de datos	Integración sin errores y actualización correcta de datos	
API y Base de Datos	API accede y actualiza datos en la base de datos	Integración sin errores y actualización correcta de datos	

5.5. Pruebas de Sistema

Caso de Prueba	Resultado Esperado	Resultado Real
Navegación por la Aplicación Web	Los usuarios pueden navegar por la aplicación sin problemas	
Funcionalidad del Chatbot	Usuarios pueden agregar, modificar y eliminar tareas a través del chatbot	
Acceso a la API de Telegram	Comunicación exitosa con la API de Telegram para servicios de mensajería web seguros	
Acceso y Actualización de la Base de Datos	Acceso y actualización correcta de datos en la base de datos a través de la aplicación web y la API	
Seguridad de la Información	Datos encriptados para garantizar la seguridad de la información	

5.6. Pruebas de Aceptación

Caso de Prueba	Resultado Esperado	Resultado Real
Usuario inicia sesión con éxito	Acceso concedido al sistema	
Usuario crea, modifica y elimina tareas individuales	Tareas se crean, modifican y eliminan correctamente	
Coordinador crea, modifica y elimina tareas de grupo	Tareas de grupo se crean, modifican y eliminan correctamente	
Usuarios visualizan sus tareas correctamente	Se muestran las tareas asignadas correctamente	
Coordinadores visualizan todas las tareas correctamente	Se muestran todas las tareas del grupo correctamente	
Funcionalidad del Chatbot	Usuarios pueden interactuar con el chatbot de manera eficaz	

6. Informe de Pruebas

6.1. Herramientas de informe

El informe de pruebas es una parte crítica del proceso de desarrollo de software, que proporciona una visión clara del estado y la calidad del producto en desarrollo. En un entorno de trabajo ágil como Scrum, estas pruebas y los problemas detectados se abordarán durante los sprints, permitiendo un enfoque incremental para mejorar la calidad del producto. Para gestionar este proceso, se utilizará Azure DevOps como herramienta de gestión de proyectos y seguimiento.

6.2. Prácticas y Procedimientos de Informe

6.2.1. **Registro de Pruebas**: Se registrarán todas las pruebas realizadas durante el sprint, incluyendo pruebas unitarias, de integración, de sistema y de aceptación. Cada prueba deberá ser documentada con su descripción, resultados obtenidos y cualquier anomalía detectada.

6.2.2. **Seguimiento de Problemas**: Cualquier defecto o problema encontrado durante las pruebas será registrado como un "issue" en Azure DevOps. Se asignará un responsable para cada problema y se establecerá un seguimiento para su resolución.

6.2.3. **Priorización de Problemas**: Los problemas identificados serán priorizados según su impacto en el producto y su criticidad. Aquellos que afecten significativamente la funcionalidad o seguridad del sistema tendrán una alta prioridad y serán abordados rápidamente.

6.2.4. **Resolución de Problemas**: El equipo trabajará en conjunto para resolver los problemas detectados durante el sprint. Se asignarán tareas específicas a los desarrolladores responsables de abordar cada problema, y se establecerán plazos para su resolución.

6.2.5. **Trabajo incremental**: El trabajo en el proyecto se realizará de manera incremental, con entregas funcionales al final de cada sprint. Esto permitirá una retroalimentación temprana por parte del cliente y una mejora continua en la calidad del producto.

7. Herramientas

Para soportar la Garantía de Calidad de Software (SQA), es importante utilizar herramientas técnicas y metodologías que faciliten el proceso de desarrollo, pruebas y aseguramiento de la calidad del software. A continuación se presentan algunas herramientas y metodologías que serán útiles:

- **Azure DevOps**: Esta plataforma nos ofrece una amplia gama de herramientas para la gestión de proyectos ágiles, incluyendo seguimiento de tareas, control de versiones, integración continua y seguimiento de problemas. Esto nos permite trabajar de manera colaborativa y coordinada, facilitando la implementación de prácticas ágiles y la gestión eficiente del proceso de desarrollo.
- **Git/GitHub**: Git es un sistema de control de versiones distribuido ampliamente utilizado que nos permite colaborar en el desarrollo de software de manera eficiente. GitHub es una plataforma basada en Git que ofrece funcionalidades adicionales, como seguimiento de problemas, revisión de código y despliegue continuo. Estas

herramientas son fundamentales para gestionar el código fuente y mantener un registro histórico de los cambios realizados durante el desarrollo del software.

8. Control de medios

Para identificar el medio físico para cada producto del software y protegerlo de daños durante el proceso de desarrollo, se pueden implementar los siguientes métodos utilizando las instalaciones físicas de Oracle Cloud Infrastructure (OCI):

- **Asignación de Recursos Virtuales en OCI**: Utilizar los recursos virtuales disponibles en OCI para asignar y gestionar el espacio de almacenamiento necesario para los diferentes productos de software. Esto incluye la creación de máquinas virtuales, contenedores de almacenamiento y redes virtuales que alojarán y protegerán los componentes del software durante el proceso de desarrollo.
- **Uso de Recursos Redundante**: Emplear opciones de deployment redundante disponibles en OCI, como múltiples availability domain, almacenamiento de objetos y almacenamiento de archivos, para garantizar la disponibilidad y durabilidad del servicio. La redundancia de deployment ayuda a proteger el servicio contra posibles fallos de hardware o interrupciones del servicio.

9. Gestión de riesgos

El plan de gestión de riesgos es una parte crucial del proyecto, ya que nos permite identificar y mitigar los posibles problemas que podrían surgir durante el transcurso de este. A continuación, se detallarán los pasos a seguir para gestionar eficazmente los riesgos:

- **Identificación de Riesgos**: Se llevará a cabo una exhaustiva sesión de lluvia de ideas con el equipo para identificar todos los posibles riesgos que podrían afectar el proyecto. Esto incluirá riesgos técnicos, operativos, de recursos humanos y externos.
- **Evaluación de Riesgos**: Una vez identificados los riesgos, se evaluará cada uno en términos de su probabilidad de ocurrencia y su impacto potencial en el proyecto. Esto nos ayudará a determinar la prioridad de cada riesgo y a centrar nuestros esfuerzos en los más críticos.
- **Mitigación de Riesgos**: Se desarrollarán estrategias de mitigación para abordar los riesgos identificados y reducir su probabilidad de ocurrencia o su impacto en el proyecto. Estas estrategias pueden incluir acciones preventivas, transferencia de riesgos, mitigación de impacto y planes de contingencia.
- **Seguimiento y Control**: Se establecerá un sistema de seguimiento y control para monitorear continuamente los riesgos a lo largo del proyecto. Esto nos permitirá identificar cualquier cambio en la naturaleza de los riesgos y ajustar nuestras estrategias de mitigación según sea necesario.