# Plan SQA

## StockManager

Collareda Agustín y Frey Hugo

# Índice

# Plan SQA

## *Propósito*

El plan de aseguramiento de calidad es aplicable a todos los proyectos de desarrollo de software de los autores. Incluye los elementos del software que serán evaluados y su finalidad dentro del proyecto. Además, especifica qué etapas del ciclo de vida del software estarán cubiertas por el plan.

Por otro lado, se debe tener una noción fresca de la definición de la calidad del software. Esta tiene como objetivo brindar la confianza de que el producto final logrará satisfacer los requisitos del cliente.

## Referencias

Este documento es referenciado en el plan de gestión de riesgos y en el plan de gestión de configuraciones.

# *Gestión*

## Organización

El grupo conformado para la cátedra de Gestión de Calidad es:

- Agustin Collareda.
- Hugo Frey.

## Actividades

*Ciclo de vida del software cubierto por el Plan*

Este plan cubre las siguientes actividades del ciclo de vida:

- Gestión de proyecto.
- Relevamiento de requerimientos.
- Análisis de los requerimientos.
- Diseño del software.
- Implementación.
- Integración y pruebas.
- Mantenimiento.

*Actividades de calidad a realizarse*

Se debe revisar cada producto. Estos son:

- Especificación de requerimientos.
- Modelos representativos obtenidos a través del análisis.
- Diseño del producto software.
- Código fuente.
- Planes asociados a la calidad y la gestión.

Estos se deberán revisar y generar un informe el cual contenga las anomalías encontradas junto a las posibles correcciones que se podrían realizar y los artefactos que deberán ser revisados, es decir, los artefactos derivados de estos productos claves. A su vez, se debe verificar que se hayan realizado las correcciones de las anomalías encontradas.

Por otra parte, se deben 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 defectos o desviaciones en los productos que se van generando a lo largo del desarrollo.

Las reuniones que se llevan a cabo por estas RFT no deben durar más de dos horas y deberán ser planificadas con anterioridad, es decir, se deberá juntar todos los artefactos necesarios junto con una lista de chequeo la cual también deberá ser rellenada con anterioridad. En la reunión no se buscan soluciones sino informar las anomalías encontradas.

## *Documentación*

### Propósito

Se deben establecer los criterios de calidad y los métodos de revisión utilizados para comprobar el cumplimiento de los mismos de cada uno de los documentos claves del desarrollo del producto de software.

### Documentación mínima requerida

*Planificación en Trello*

El desarrollo de un sistema tendrá atrás una planificación que debe contener las tareas próximas que se deben realizar. Una tarea tiene asociado un título representativo con una posible descripción para más detalles. A su vez, cada tarea puede tener subtareas, siguiendo la filosofía de divide y vencerás, lo que permite al equipo poder visualizar el avance para tener mucha más motivación.

Todo esto se hará al estilo ágil con una pizarra en un tablero con la herramienta Trello. Cada miembro es responsable de hacer las tareas y completarlas en un tiempo establecido.

En el tablero se deberá seguir la siguiente nomenclatura:

T<ID>: <Tarea> - <Documento>

Un ejemplo de esto sería T01: Modificar - Plan de Gestión de Configuraciones.
Las tareas pueden ser crear, modificar, eliminar, implementar, entregar, mover y subir.

Dentro de la tarjeta se realizará un checklist que sigue la filosofía de divide y vencerás. Se establecen subtareas de la tarea principal con el fin de finalizarlo en el tiempo determinado.

*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:
- Externa, respecto al alcance acordado.
- Internamente, no deben existir elementos sin especificar.
- Ser consistente, no puede 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’.

Los atributos de calidad que se buscan son:

- Funcionalidad
- Adecuación a las necesidades
- Seguridad de los datos
- Confiabilidad
- Madurez

- Tolerancia a faltas
- Usabilidad
- Comprensible
- Aprendible
- Operable
- Eficiencia
- Mantenibilidad
- Modificable
- Estable, no se producen efectos inesperados luego de modificaciones
- Verificable

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

*Descripción del diseño del software*

El documento de diseño específica 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.
Algunos de los criterios para este documento son:

- Todo elemento del diseño debe contribuir a algún requerimiento.
- 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.

*Documentación de pruebas*

Esta documentación debe contener anexada las pruebas unitarias y ejecutadas las pruebas de validación con el fin de que se haga un proceso de pruebas correcto y efectivo para garantizar la confiabilidad del software.

En este caso, se cuenta con una planilla de excel en el cual se van ingresando las pruebas de validación, esta debe contar con un identificador para cada prueba, el responsable de la misma, una descripción de los hechos observados, una serie de pasos para verificar su uso y el estado.

A su vez, se rellenará un documento llamado informe de verificación y validación el cual garantiza que los errores descubiertos no se pierdan y puedan ser rastreados para poder corregirlos.

*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 ítems 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 un manual de usuario y un manual de instalación.

## *Plan de Gestión de configuración*

### Propósito

Este plan tiene como fin controlar la entrega y el cambio de los elementos a través del ciclo de vida del sistema, y almacenar el estado de los elementos y de las peticiones de cambio.

### Resumen

La Gestión de Configuración identifica los elementos de un proyecto de desarrollo de software (especificaciones, requisitos, arquitecturas, código, planes, etc.) proporcionando el control de los elementos identificados y la generación de informes de estado de la configuración, consiguiendo, al mismo tiempo, claridad de gestión, al asignar responsabilidades al personal encargado de las tareas de control a lo largo del ciclo de vida del producto.

## Organización y Responsabilidades

Al tratarse de un grupo pequeño de desarrollo, la administración de gestión de versiones será un trabajo realizado de forma cooperativa. Los cambios a realizar serán solicitados y tratados mediante comunicación directa entre los integrantes del equipo de desarrollo, ya sea por Whatsapp, Discord u otras herramientas de mensajería utilizadas.

Se anexa el “Plan de Gestión de Configuraciones_Collareda Agustín y Frey Hugo“ en que está más detallado.

# *Plan de pruebas*

## Propósito

El presente documento tiene como objetivo definir el Plan de Pruebas para el sistema StockManager, en el marco del proceso de verificación y validación del producto. Este plan establece el enfoque general que se seguirá para garantizar que el software cumpla con los requisitos funcionales y no funcionales definidos, y que se comporte de manera confiable bajo condiciones normales y excepcionales.

El propósito de este plan es proporcionar una visión clara y estructurada de las actividades de prueba a realizar, los recursos necesarios, los criterios de entrada y salida, los tipos de pruebas aplicables.

## Resumen

El plan abarca los casos de uso relacionados con la gestión de muebles, incluyendo su creación, modificación, eliminación y consulta, garantizando el cumplimiento de los requisitos funcionales establecidos.

Las pruebas se enfocan principalmente en pruebas unitarias automatizadas desarrolladas con PHPUnit, utilizando técnicas de caja blanca para pruebas internas y caja negra para validaciones funcionales. Dado que el sistema interactúa con una base de datos, se emplea la técnica de mocking para simular el comportamiento de los componentes externos y asegurar pruebas aisladas, confiables y repetibles.
Se definieron casos de prueba para los siguientes casos de uso:

- CU02: CrearMueble

- CU03: ModificarMueble
- CU04: EliminarMueble
- Casos complementarios para obtener muebles por ID

Cada caso de prueba incluye su descripción, entradas, salidas esperadas y criterios de éxito. Las pruebas contemplan tanto escenarios válidos como situaciones de error o entradas inválidas, evaluando la robustez del sistema ante condiciones normales y excepcionales.

El seguimiento y control se realiza mediante el análisis de resultados generados por PHPUnit y GitHub Actions, donde se registran los estados de ejecución de cada prueba. Esto permite detectar fallos tempranamente y garantizar la calidad continua del software.

Se anexa el “Plan de Pruebas_Collareda Agustín y Frey Hugo“ en que está más detallado.

## *Estándares y pautas a seguir*

Los autores se comprometen a seguir los siguientes estándares y pautas pactados:

- Se deberá seguir y aprender un estilo de codificación por cada lenguaje que se utilice, asegurando coherencia y legibilidad en el código.
- Se deberá documentar el código de manera clara y estructurada, utilizando herramientas y formatos estándar según el lenguaje de programación empleado.
- Los comentarios de código deben tener un formato asociado, incluyendo descripciones concisas de funciones, clases y métodos, así como etiquetas específicas para remarcar aspectos importantes.
- Realizar revisiones periódicas para verificar el cumplimiento de estándares, identificar errores y mejorar la calidad del producto software.
- Implementar pruebas para validar la funcionalidad del código y garantizar su fiabilidad.
- Utilizar control de versiones para gestionar cambios con el fin de que exista una recuperación ante una falla grave.
- Reutilizar código mediante el uso de patrones de diseño, bibliotecas y buenas prácticas de programación.
- Asegurar el cumplimiento de normas de seguridad y protección de datos en todas las etapas del desarrollo.
- Capacitarse en nuevas herramientas, lenguajes de programación y buenas prácticas de programación.
- Seguir un enfoque de mejora continua.

### Estilo de codificación (PHP)

Al programar con el lenguaje PHP, se seguirán las recomendaciones dadas por los estándares PSR-1, PSR-4 y PSR-12. Esto con el objetivo de lograr mantenibilidad y fiabilidad por parte del producto software.

Para garantizar el cumplimiento de este estándar, se optó por el uso de una herramienta llamada PHP_CodeSniffer la cual es opensource y se utiliza para detectar violaciones a estándares de codificación en archivos PHP. Algunas de las violaciones que detecta son: Indentación incorrecta, nombres de variables mal escritos, uso indebido de espacios, llaves mal ubicadas, etc. Cabe destacar que también cuenta con una

funcionalidad para solucionar la mayoría de problemas encontrados, esto se hace con el comando ***phpcbf***.

Para los proyectos elaborados en php de nuestro grupo, se va a seguir con el siguiente procedimiento. Una vez finalizado el código, se va a pasar por la herramienta mencionada anteriormente, los errores que se puedan solucionar automáticamente lo serán por la herramienta, los demás serán solucionados de manera manual y el estándar PSR-4 también será validado de esta manera ya que la herramienta vista no cuenta con este estándar para poder ser analizado. Para esto, se debe ver la estructura del loader generada en el archivo *composer.json* y según el estándar.

Para generar el archivo loader, se deberá utilizar el comando ***composer dump-autoload***.

## Usabilidad

Para verificar y validar la calidad del producto software desde la perspectiva del cliente, una de las características más importantes es la usabilidad. La dificultad que experimenta un usuario al utilizar un software puede perjudicar considerablemente el modelo de negocio planteado.

Para evaluar este aspecto, se tomaron como referencia las 10 reglas de Nielsen, que son un conjunto de principios generales para el diseño de interfaces que favorecen que el software sea intuitivo y fácil de usar.

Sobre esta base, se respondieron una serie de preguntas respecto al sistema y se clasificó la prioridad de los hallazgos mediante una escala del 1 al 5, considerando dos criterios:

- Facilidad de reparación: dónde 1 representa una corrección muy difícil y 5 representa una corrección muy sencilla.
- Severidad del problema: dónde 1 representa un problema leve y 5 representa un problema muy grave para el usuario.

Esta priorización permitió determinar qué aspectos deben resolverse primero para maximizar la usabilidad del producto y, por ende, su aceptación por parte del cliente.

## *Gestión de riesgos*

Un riesgo es un evento adverso que puede o no ocurrir el cual tiene asociado un impacto al proyecto. Este impacto a su vez se puede definir como qué tan grave fueron las pérdidas.

La gestión de riesgo puede seguir una estrategia proactiva o reactiva. En este caso se optó por la estrategia proactiva ya que las ventajas son superiores a las estrategias reactivas.

La gestión de riesgos se componen de las siguientes actividades:

- Identificación: En esta se obtiene la lista de riesgos preliminares, se busca todos los riesgos posibles que puedan ocurrir. Usualmente se tiene una lista de preguntas para determinar qué riesgos se tiene o por la experiencia del equipo también se conoce algunos riesgos.
- Análisis: En esta etapa se obtiene una lista de riesgos prioritarios. Para este caso, se analiza sobre tres ejes los riesgos, el impacto en una escala de despreciable, tolerable, serio y catastrófico. La probabilidad que se mide en porcentaje del 1 al 100 y la magnitud es medida en semanas.
- Planificación: En esta etapa se obtienen los planes de prevención, minimización y contingencia para los riesgos prioritarios. En este caso se seleccionan los riesgos con mayor magnitud, probabilidad e impacto y se realizan los correspondientes planes los cuales tienen asociados tareas concretas. Estos planes tienen como objetivo disminuir la probabilidad, el impacto y si ocurre como se debería manejar.
- Monitoreo: Esta etapa se va realizando periódicamente, al inicio de cada iteración, y tiene como fin revisar si se introdujo un nuevo riesgo, si se modificó el impacto, probabilidad o magnitud de los riesgos que fueron encontrados y si se debería volver a planificar los riesgos ya gestionados.

Se anexa el “plan de Gestión de Riesgos” que explica en más detalle estas etapas y cómo se realizará durante el proyecto.