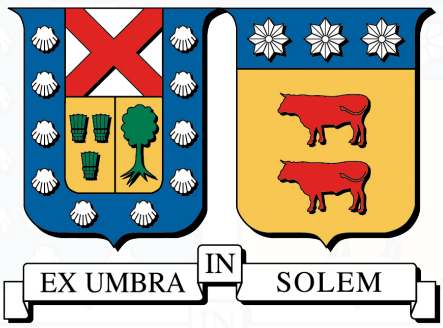
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO - CHILE



PLAN DE CALIDAD PARA GERPRIN

GABRIEL ARÍSTIDES CAMILO SAN MARTÍN

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO DE EJECUCIÓN EN INFORMÁTICA

PROFESOR GUÍA	:	MARCELLO VISCONTI ZAMORA
PROFESOR CORREFERENTE	:	LIOUBOV DOMBROVSKAIA

ABRIL 2019

EX UMBRA
IN
SOLEM

RESUMEN EJECUTIVO

Gerprin es un sistema integral de seguridad antirrobo para vehículos, desarrollado por *Duosoft*. *Gerprin* provee al propietario del automóvil la seguridad y confianza necesaria para poder aparcar y saber que su automóvil no será robado, mediante la implementación de un sistema a bordo dotado de cortacorriente que se activa utilizando biometría y un módulo *GPS* que alerta en caso de movimientos no autorizados, además, el sistema cuenta con una aplicación para dispositivos móviles que permite configurar y controlar el sistema abordo.

El presente trabajo busca documentar los recursos, actores y tiempos que deben ser empleados en un determinado proceso con el fin de asegurar cualidades objetivas que se cumplir por el producto final.

ABSTRACT

Gerprin is an integral anti-theft security system for vehicles, developed by *Duosoft* that ensure the owners to park with the security and confidence. *Gerprin* counts with an onboard system equipped with a short-circuit that is activated using biometrics and a *GPS* module that alerts in case of unauthorized movements, furthermore it has a mobile application in order to facilitate the follow-up of the system.

The present work is looking forward to document the resources, actors and times that should be used in a certain process to ensure objective qualities that are met by the final product.

EX UMBRA
IN
SOLEM

Índice de Contenidos

Índice de Tablas

EX UMBRA
IN
SOLEM

Índice de Figuras

EX UMBRA
IN
SOLEM

1 | Introducción

Considerando el prototipo de *Gerprin* desarrollado para la XXIII Feria de Software de la Universidad Técnica Federico Santa María en año 2015, se busca crear un plan de calidad para el desarrollo de una nueva versión de este con propósitos comerciales, y así lograr un producto con estándares de calidad adecuados al mercado y asegurar un correcto funcionamiento de la nueva versión del sistema.

Es menester comenzar contextualizando el problema que se busca solucionar mediante la invención de *Gerprin*, antes de proceder con una descripción preliminar del plan de calidad que se busca realizar.

1.1. Resumen del proyecto

El Informe Anual entregado por Carabineros de Chile (Chile, 2018) contabilizó un total de 16.678 robos de vehículos motorizados entre los meses de enero y agosto. Esto significa que se sustrajo un vehículo, en nuestro país, cada 21 minutos.

En respuesta a lo anterior, nace *Gerprin*, él busca eliminar la inseguridad en los conductores y permitir que puedan aparcar en cualquier lugar sin miedo a ser víctimas del robo de su vehículo. Para lograr esto, fue propuesto un sistema de seguridad vehicular integral que cuenta con tres aspectos. En primer lugar, un sistema a bordo de corta corriente controlado con un lector biométrico para que ningún extraño pueda operar el vehículo. En segundo lugar, un sistema centralizado de monitoreo de la posición *GPS* del vehículo que genera alertas al cliente cuando se detectan movimientos no autorizados. Y finalmente, se desarrolló una aplicación móvil que le permitirá a nuestros clientes monitorear la posición de su vehículo y controlar el sistema a bordo, permitiendo agregar, modificar o eliminar usuarios de una forma fácil y rápida.

Se buscó solucionar una de las necesidades más básicas del ser humano: la seguridad. No sólo se creó un gadget antirrobo, sino que gestó un completo sistema de protección para el automóvil, el cual entregará confianza y tranquilidad a los usuarios cuando deban estacionar sus vehículos en lugares que no conocen o incluso en su propio hogar. Con *Gerprin* de monitoreo podrán saber en todo momento cual es la posición de su automóvil, mientras que, con el sistema biométrico, tendrán la certeza de que nadie podrá utilizar el vehículo sin su autorización.

En último lugar, para comprender la envergadura del sistema es necesario señalar las diferencias con los productos del mercado. Señalar, además, que ya existen sistemas de corta corriente, incluso los hay operados con biometría; Los sistemas de monitoreo *GPS* ya se han masificado y especialmente en el área de transporte comercial. *Gerprin* representa un sistema de seguridad integral que combina todos los aspectos anteriores: la seguridad a bordo (corta corriente con biometría), monitoreo satelital centralizado (*GPS*) y el control de estos mediante una aplicación móvil fácil de utilizar.

1.2. Propósito

Como fue mencionado al comienzo del presente documento, este trabajo busca realizar una segunda iteración en el desarrollo de *Gerprin*, con el fin de generar una versión comercializable del sistema. Utilizando como punto de partida la primera iteración diseñada y desarrollada para la XXIII feria de software de la Universidad Técnica Federico Santa María.

En el presente plan de calidad se tiene como propósito establecer los procesos y acciones para el aseguramiento de calidad, definiendo la estructura para el desarrollo del sistema, las gestiones necesarias para dicho desarrollo, las herramientas y técnicas a emplear y las pruebas necesarias para asegurar la calidad esperada.

1.3. Alcance

El plan de calidad abordará las distintas etapas de ingeniería de software, desde un punto de vista práctico, enfocado a los productos y desarrollo de los pasos de análisis de requerimientos, diseño, desarrollo pruebas y documentación.

Dada la existencia previa de un software funcional desarrollado el año 2015 que cumple con los requisitos funcionales especificados para dicha aplicación, el presente plan de calidad se enfocará en los requisitos no funcionales.

2 | Requisitos no funcionales

Se hace menester comenzar el plan de calidad definiendo los “atributos de calidad”, los indicadores de medición y los valores de validación de estos. Lo anterior, con el fin de definir los márgenes generales en el desarrollo de la aplicación y el nivel de integración del sistema con los atributos antes mencionados.

2.1. Definición de atributos de calidad

A continuación, se procederá con la presentación y definición de los requisitos no funcionales que definirán el lineamiento del desarrollo de la aplicación.

Fiabilidad:

Se debe garantizar la respuesta de la aplicación ante una acción del usuario y además que la información entregada sea correcta.

Usabilidad:

Facilidad en el uso de la aplicación, reflejándose en la rapidez para aprender el funcionamiento del sistema y recordar este aprendizaje.

Adapabilidad:

Capacidad de la aplicación de funcionar en distintos entornos informáticos.

Temporización:

Debe emplear el menor tiempo posible en entregar una respuesta visible al usuario ante una acción de este.

Soportar alta carga:

El Sistema debe ser capaz de funcionar incluso en momentos de alto tráfico.

Disponibilidad:

El sistema debe encontrarse disponible para su acceso o consulta el mayor tiempo posible.

Seguridad:

Protección de la infraestructura e información contenida en el sistema.

Escalabilidad:

Capacidad del sistema de aumentar la carga de trabajo o sus funciones sin ver afectado el funcionamiento del resto de funcionalidades y tampoco disminuir su calidad.

Mantenibilidad:

Esfuerzo y tiempo dedicado al soporte del sistema.

2.2. Objetivos Cuantificables

En la presente sección se establecerán los indicadores de cada uno de los atributos de calidad definidos en el apartado anterior. Además, se establecen los valores mínimos y máximos de los indicadores de los atributos de calidad, con el propósito de establecer de forma objetiva el cumplimiento requisitos no funcionales deseados en el sistema.

2.2.1. Fiabilidad

Es necesario que la información entregada por cualquiera de los distintos módulos que forman *Gerprin* sea legible por los módulos restantes y de este modo no tener perdidas en la comunicación entre distintas plataformas. Las solicitudes y respuesta deben ser verídicas y corresponder a acciones reales desencadenadas por los actores de la aplicación.

Para lo anterior es necesario que el sistema de comunicación que se desarrollará no presente perdidas de información, sin embargo, se deben fijar estándares distintos para cada módulo respecto a la fiabilidad de los datos, debido a que la información que maneja la aplicación tiene distintos grados de importancia, incluso permitiendo perdida parcial de esta en determinadas condiciones.

Comenzando con el sistema a bordo, es necesaria la comunicación fiable con la *API* de *Gerprin* al momento de realizar activación o desactivación del cortacorriente mediante la aplicación móvil, pero no lo es para establecer una conexión fiable cada vez que el módulo contenedor del *GPS* envía la posición. Por este motivo el envío de la ubicación se realiza mediante protocolo *UDP*[1] , a diferencia del resto de las funciones del sistema a bordo que deben emplear *TCP*[2].

Por otro lado, la aplicación móvil y la *API* desarrollada, deben contar con una conexión fiable de comunicación, esto debido a la información critica que es manejada por estos módulos.

2.2.1.1. Objetivos cuantificables

- Funcionalidades de la aplicación móvil sin errores.
- Servicios entregados por la *API* sin errores.
- Envió de posición de forma precisa e intervalos regulares sin errores.
- Funcionalidad de activación y desactivación de relé de forma remota sin errores.

2.2.1.2. Criterios de medición.

Tabla 2.1: Validación para los indicadores de fiabilidad.

Criterio	Medición
Funcionalidades de la aplicación móvil sin errores.	95 % de efectividad sin errores.
Servicios entregados por la *API* sin errores.	95 % de respuestas sin errores.
Envió de posición de forma precisa e intervalos regulares sin errores.	75 % de posiciones almacenadas con un error de ±20 metros.
Funcionalidad de activación y desactivación de relé de forma remota sin errores.	95 % de efectividad.

[1]https://www.ietf.org/rfc/rfc768.txt
[2]https://tools.ietf.org/html/rfc793

Los criterios de validacion se basan en la experiencia de aplicaciones similares. Es necesario considerar una conexión a internet estable que permita el envío y recepción de información.

2.2.2. Usabilidad

Es necesario para el módulo de la aplicación móvil del sistema contar con una interfaz que entregue una curva de aprendizaje empinada, es decir, que permita comprender el funcionamiento de la aplicación móvil en un corto periodo de tiempo.

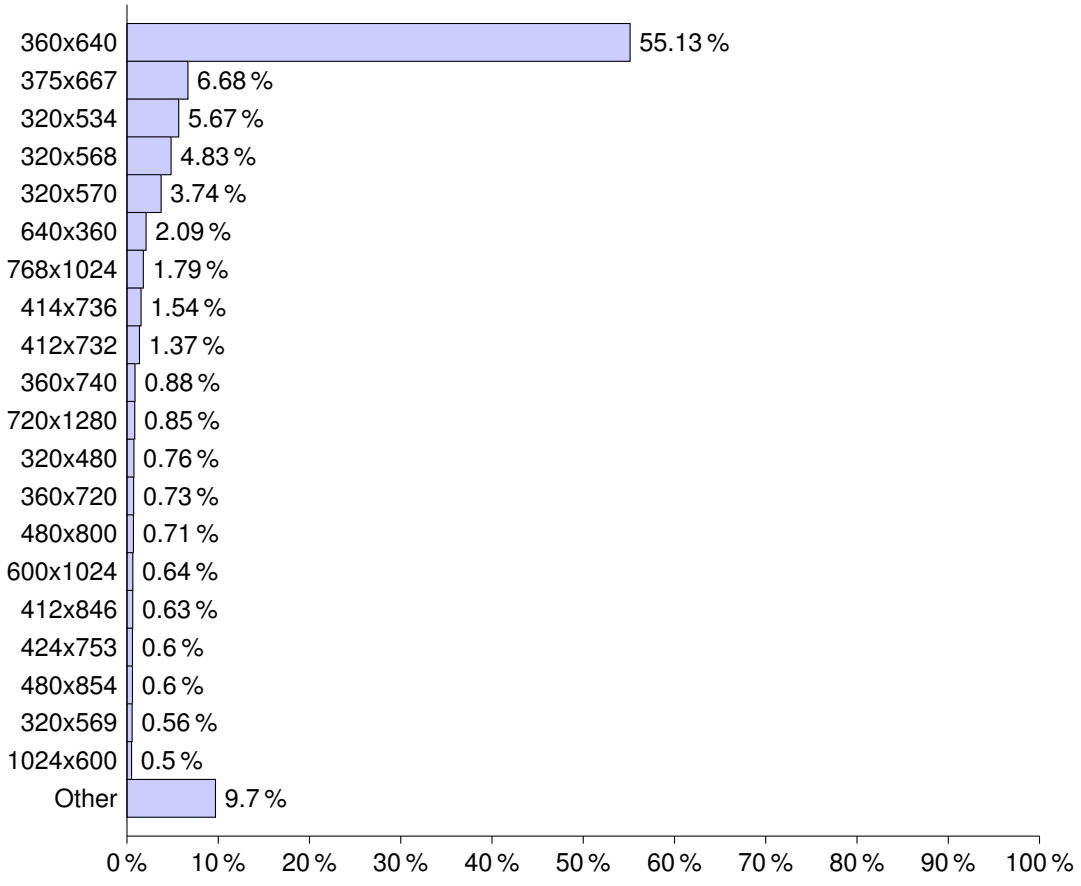
Para esto es necesario que la interfaz aplique colores y símbolos coherentes con las acciones a realizar o información que muestra, Información ordenada y fácil de entender.

Dado que el sistema será desarrollado para dispositivos móviles es importante que sea responsivo. Por otro lado, el tamaño de la pantalla del dispositivo, si bien, no es importante en términos de la distribución de los elementos a mostrar, si lo es en términos de usabilidad, es decir, que se muestre adecuadamente los elementos en la pantalla y con un tamaño adecuado. Por este motivo se considera que debe adecuarse a la mayor cantidad de dispositivos.

Considerando que el primer *iPhone* contaba con un tamaño de pantalla de 3,5 pulgadas y que estas van en aumento, se considerará esta como el tamaño mínimo de pantalla.

Respecto a la resolución mínima de pantalla, se puede apreciar en el gráfico 2.1 que aproximadamente el 74 % de los usuarios de dispositivos móviles en chile emplean una resolución mayor a 360 x 640 píxeles.

Figura 2.1: Screen Resolution, Market Share in Chile Sept 2017 - Sept 2018
http://gs.statcounter.com/screen-resolution-stats/mobile-tablet/chile/#monthly-201709-201809-bar.



2.2.2.1. Objetivos cuantificables

- Dimensiones de la página adaptables a los tamaños de pantalla de los dispositivos.
- Dimensiones de la página adaptables distintas resoluciones.

2.2.2.2. Criterios de Validación

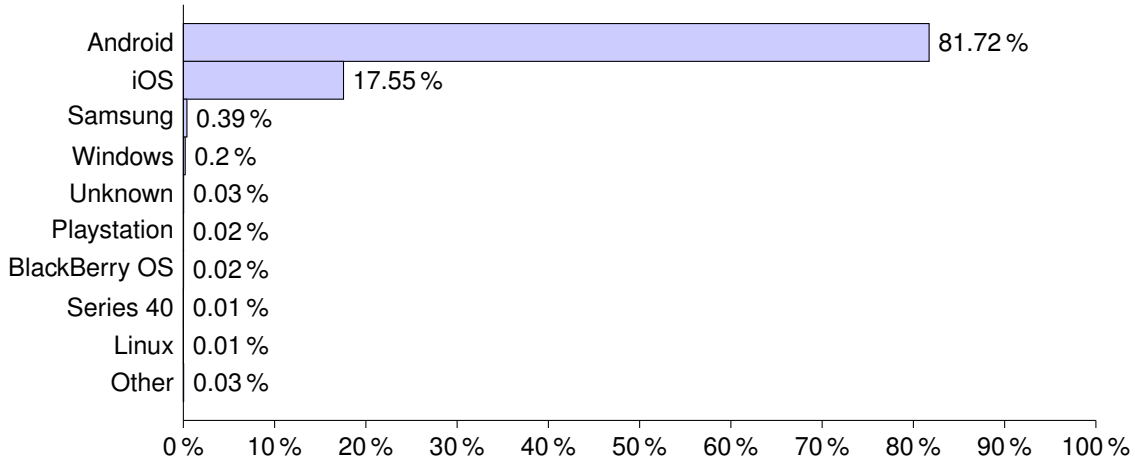
Tabla 2.2: Validación para los indicadores de usabilidad.

Criterio	Medición
Tamaño pantalla mínimo.	3,5 pulgadas.
Resolución mínima.	360x640 pixeles.

2.2.3. Adapabilidad

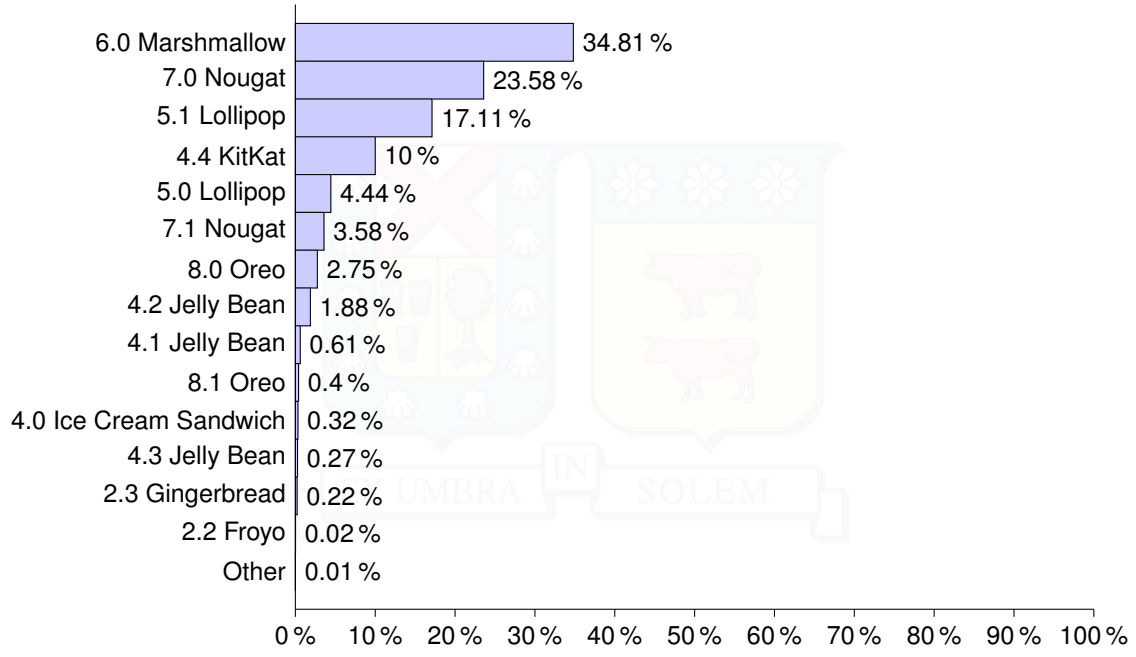
En vista de la existencia de distintas plataformas móviles, es necesario asegurar un correcto funcionamiento con los sistemas operativos más comunes de estas. A continuación, se puede observar en la gráfica 2.2 el detalle en porcentajes del mercado que poseen los sistemas operativos más populares del último año.

Figura 2.2: OS, Market Share in Chile Sept 2017 - Sept 2018
http://gs.statcounter.com/os-market-share/mobile-tablet/chile/#monthly-201709-201809-bar.



Como se puede apreciar en el gráfico 2.2, los sistemas más populares corresponden a *Android* e *iOS*, que en conjunto corresponden al 95,26 % del mercado. Teniendo en cuenta estas cifras y el amplio segmento que abarcan estos dos sistemas, el desarrollo de *Gerprin* se concentrará en dichas plataformas. Para el caso de *Android*, el uso de las distintas versiones se muestra en el grafico 2.3.

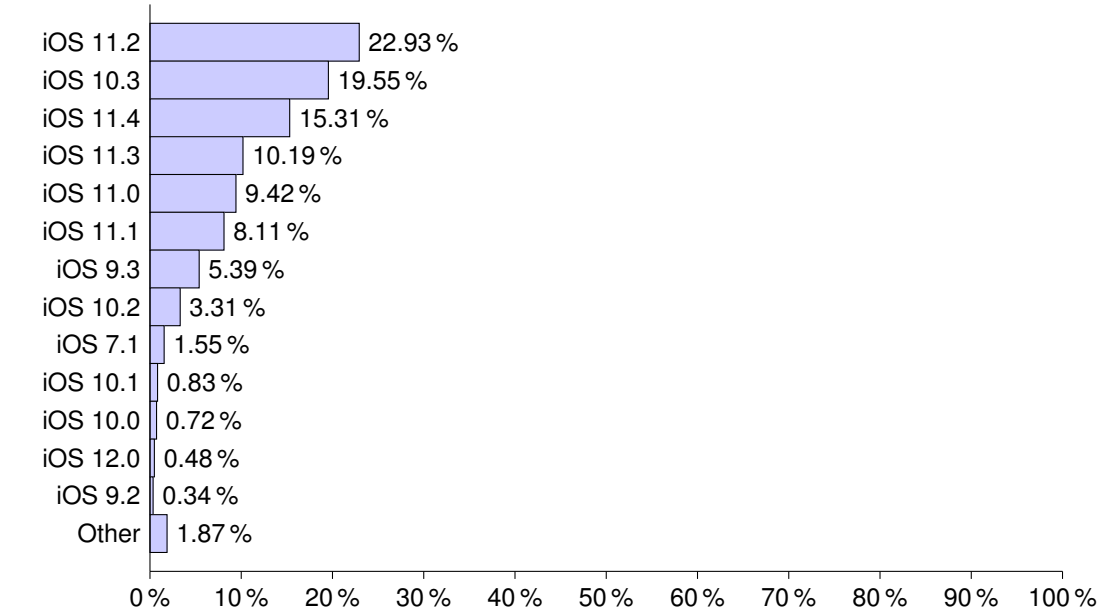
Figura 2.3: Android Version, Market Share in Chile Sept 2017 - Sept 2018
http://gs.statcounter.com/android-version-market-share/mobile-tablet/chile/#monthly-201709-201809-bar.



Si bien, corresponde a un segmento importante el uso de *Android KitKat* (10 %) es desde la versión de la *API* de *Android* 21, empleada por google en el desarrollo de *Android Lollipop*, que se integra nativamente *Material Design* que es ampliamente recomendado para el diseño de aplicaciones. Debido a este motivo, se considerá el desarrollo enfocado en sistemas *Android* 5.0 o mayores.

Por otro lado, las versiones de *iOS* y el porcentaje de su participación en el mercado se muestran en el gráfico 2.4.

Figura 2.4: iOS Version, Market Share in Chile Sept 2017 - Sept 2018
http://gs.statcounter.com/ios-version-market-share/mobile-tablet/chile/#monthly-201709-201809-bar.



En base al porcentaje de uso de las distintas versiones de *iOS* del gráfico 2.4, se considera un desarrollo para la versión 10.2 o superiores.

2.2.3.1. Objetivos cuantificables

- Correcta ejecución en una versión minima de *Android*.
- Correcta ejecución en una versión minima de *iOS*.

2.2.3.2. Criterios de Validación

Tabla 2.3: Validación para los indicadores de adapabilidad.

Criterio	Medición
versión *Android*.	5.0 o superior.
versión *iOS*.	10.2 o superiores.

2.2.4. Temporización

Se considera la velocidad de conexión del sistema a bordo de 9,6 kbps correspondiente a tecnología *GSM* y una de hasta 2 mbps para una red *3G* para la aplicación móvil, como la velocidad necesaria para un óptimo funcionamiento del sistema.

Debido a que *Gerprin* es una aplicación basada en servicios, es necesario que estos tengan una baja taza de espera al consultar al módulo que contiene la *API*.

2.2.4.1. Objetivos cuantificables

- Rápida carga de plantillas.
- Rápida Respuesta de los servicios.

2.2.4.2. Criterios de Validación

Tabla 2.4: Validación para los indicadores de temporización.

Criterio	Medición
Tiempo de respuesta.	inferior a 3 segundos.
Compresión *gzip*	Si

Los criterios de validación se basan en la experiencia de aplicaciones similares.

2.2.5. Soportar alta carga

Debido al alto tráfico del servidor central de la aplicación y que esta basa su arquitectura en servicios provistos por una *API*, es de suma importancia que cuente con el soporte necesario para responder a una gran cantidad de solicitudes en un corto periodo de tiempo.

Mediante herramientas como la consola de *Google Play* o *Google Analitycs* se pueden obtener estadísticas del uso del sistema, tales como tiempo medio de navegación, número de terminales con la aplicación instalada, entre otros factores.

Esta información se emplea para calcular el promedio de usuarios conectados y se estima, en base a aplicaciones similares, que cada uno de ellos realizará una consulta al servidor cada 2 segundos.

Es importante considerar que el número de usuarios varia dependiendo de la popularidad de *Gerprin*. Debe ser obtenido inicialmente mediante un estudio de mercado y posteriormente corregida con datos empíricos.

2.2.5.1. Objetivos cuantificables

- Cantidad de solicitudes por segundo.

2.2.5.2. Criterios de Validación

Tabla 2.5: Validación para los indicadores de soporte de alta carga.

Criterio	Medición
Cantidad de solicitudes por segundo.	1 por segundo cada 2 usuarios.

Los criterios de validación se basan en la experiencia de aplicaciones similares.

2.2.6. Disponibilidad

Debido a la importancia de la aplicación es completamente necesario que esta se encuentre disponible el mayor tiempo posible.

Los usuarios deben tener plena confianza en que la plataforma se encuentra disponible y enviará las notificaciones *push* si el sistema detecta un intento de robo, ya sea un intento de encender el vehículo por un usuario no autorizado o si este se desplazó sin autorización.

Además, debe estar disponible en casos de ser necesario desactivar el cortacorriente de forma remota sin una huella digital cuando ocurra una emergencia.

Es deseable, además, que la aplicación se encuentre offline por periodos no mayores a 4 horas en caso de presentar algún problema.

Basado en criterios de experiencias en el desarrollo de aplicaciones similares, es necesario que esta no presente un tiempo offline mayor a la suma de 12 horas mensuales y no puede ser en intervalos mayores a 4 horas desde reportado el problema.

2.2.6.1. Clúster de servidores

La aplicación debe encontrarse en un *Cluster* de servidores, es decir, debe encontrarse en varios computadores respondiendo las solicitudes como uno solo. Esto, con el objeto de dividir la carga de procesamiento de los servidores, evitar la falta de disponibilidad por fallas de hardware y permitir escalar la cantidad de carga que puede aguantar el servicio de forma rápida y fácil. Por esta razón el sistema deberá contar con un mínimo de dos instancias de máquinas con la posibilidad de aumentar dicha cantidad según la demanda.

2.2.6.2. Objetivos cuantificables

- Tiempo online total.
- Máximo tiempo offline seguido.

2.2.6.3. Criterios de Validación

Tabla 2.6: Validación para los indicadores de disponibilidad.

Criterio	Medición
Tiempo online total.	98 % del tiempo se debe encontrar online.
Máximo tiempo offline seguido.	Intervalos no mayores a 4 horas.

Los criterios de validación se basan en la experiencia de aplicaciones similares.

2.2.7. Seguridad

Debido a la naturaleza del sistema y la información sensible que esta puede manejar, es necesario implementar roles para los usuarios con el propósito de establecer un acceso consistente y seguro a la información contenida en la plataforma.

Para implementar lo anterior es necesario establecer cuentas de usuarios protegidas por contraseñas que deben ser almacenadas de forma encriptada y un token o *API Key* que permita realizar consultas seguras a la API del sistema.

Es necesario, además, contemplar ataques de tipo *SQL injection*[3] y de fuerza bruta.

2.2.7.1. Limitar puntos de acceso al servidor

Es necesario establecer la arquitectura del servidor a usar. Se empleará un servidor *AWS EC2* y los puertos abiertos son:

- Conexión SSH: puerto 22.
- Puerto API: 80.

2.2.7.2. Objetivos cuantificables

- Seguridad de sesión.
- Limitar puntos de acceso al servidor.

[3] http://www.revistasbolivianas.org.bo/pdf/rits/n8/n8a17.pdf

2.2.7.3. Criterios de Validación

Tabla 2.7: Validación para los indicadores de Seguridad.

Criterio	Medición
Máximo de intentos fallidos.	10.
Ataque fuerza bruta.	Se añade Captcha al formulario al superar máximo de intentos fallidos.
Ataque inyección SQL.	No permitir instrucciones de base de datos en ningún formulario.
Requisitos al crear contraseña.	Mínimo 8 caracteres con al menos un digito.
Encriptar contraseña.	Si, empleando método blowfish.

Los criterios de validación se basan en la experiencia de aplicaciones similares.

2.2.8. Escalabilidad

Es necesario que el sistema permita añadir nuevas funcionalidades, modificar las ya implementadas, o bien, aumentar la capacidad de trabajo sin afectar el rendimiento de este.

Para lograr lo anterior es necesario plantear una serie de puntos que permitan agregar o modificar funciones del sistema sin afecten a otras.

2.2.8.1. Objetivos cuantificables

- Encapsulamiento de los módulos.
- Facilitar escalabilidad de la arquitectura del servidor.

2.2.8.2. Criterios de Validación

Encapsulamiento de los módulos

Lo primero que debe ser tomado en cuenta es la creación del sistema de forma modular. Esto se logra encapsulando funciones, para que trabajen de manera privada y que solamente puedan ser llamadas por otras funciones sin modificar la forma en que realizan sus procesos. De este modo, se pueden modificar de forma independiente.

Para este desarrollo, se considera que la aplicación se divide en tres módulos que trabajan juntos, pero su funcionamiento se encuentra encapsulado. El primer módulo es una aplicación móvil desarrollada de forma nativa para *Android* y *iOS*, el segundo módulo es una *API* instalada en un servidor *AWS EC2*. Y finalmente un módulo Arduino a bordo del vehículo.

Encapsulamiento de los módulos

Otro punto importante que tomar en cuenta es la escalabilidad vertical y horizontal de la arquitectura del servidor. La escalabilidad vertical permite aumentar la capacidad física de almacenamiento y procesamiento, aumentando las características del hardware donde se encuentra el sistema. Mientras que la horizontal permite aumentar el número de nodos que conforman el *Cluster* del servidor. Para el caso de *Gerprin*, *AWS EC2* permite la rápida implementación de ambos métodos.

2.2.9. Mantenibilidad

ISO/IEC 25010[4] define la mantenibilidad como el grado de efectividad o eficiencia con la que un producto o sistema puede ser modificado. Para logar reducir el trabajo de mantenibilidad del *software* existen una variedad de atributos que pueden ser integrados para facilitar la labor.

2.2.9.1. Objetivos cuantificables

- Comentarios en el código fuente.
- Estandarización del nombre.

2.2.9.2. Criterios de Validación

Comentarios en el código fuente

Es necesaria la existencia de comentarios dentro del código fuente del programa. Se busca que mínimamente se encuentren comentarios de cada función y método indicando valores de entrada, salida y una descripción de las acciones que realiza. Además, se esperan comentarios explicativos de cada clase indicando el significado de sus atributos.

Estandarización del nombre

Otra buena práctica para facilitar el mantenimiento es la estandarización de nombres siguiendo la convención que existe para este propósito para *PHP*[5].

[4] I. S. Organization, "ISO/IEC 25010," in Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and Software Quality Models, ed, 2011

[5] http://pear.php.net/manual/en/standards.naming.php

3 | Modelo de desarrollo

3.1. Elección del modelo

Para el desarrollo de la presente aplicación será empleada *Extreme Programming (XP)* que corresponde a una de las metodologías basadas en el manifiesto ágil [6]. Bajo la lógica de un desarrollo iterativo e incremental se desarrollará la aplicación en ciclos o iteraciones que añaden funcionalidades al sistema, comenzando por los requerimientos más críticos o básicos.

Debido a la naturaleza de la aplicación (enfocada a servicios), el desarrollo iterativo se adapta muy bien al proceso de creación del sistema. Esto debido a que las distintas funcionalidades de la aplicación pueden ser desarrolladas de forma individual unas de otras y en iteraciones distintas. El segundo principio del desarrollo ágil[7] toma una importancia vital en la creación de funciones que pueden ser *testeables* en etapas tempranas del proyecto.

Según lo dicho anteriormente, la frase apócrifa *divide et impera* toma relevancia al dividir el problema completo en tres módulos e incluso en funciones encapsuladas que pueden ser desarrolladas de forma individual.

La capacidad iterativa de las metodologías agiles se complementa con la posibilidad de desarrollar funciones de forma paralela, de este modo se puede reducir el tiempo de desarrollo entre iteraciones. Si bien, esta característica no es aplicable a la totalidad del proyecto si puede ser aplicada a funciones que no tengan una dependencia de otras.

Hasta el momento se han nombrado dos ventajas propias de todas las metodologías que sigan la lógica del desarrollo ágil. *Extreme Programming* entrega ventajas propias que se adaptan a lo esperado y aseguran un *software* de calidad.

Las pruebas unitarias continuas, facilitadas por el desarrollo iterativo permiten probar funcionalidades completas de forma temprana y a su vez, la corrección temprana en caso de identificar fallas.

El *Test-Driven Development (TDD)* corresponde a un *framework* de *Extreme Programming* que se basa en la definición de pruebas unitarias antes del desarrollo de la aplicación y la implementación de estas en paralelo al trabajo de codificación del sistema, de este modo aseguramos el cumplimiento de los requerimientos definidos con el cliente. Es importante notar que el proceso de pruebas unitarias de las funciones terminadas y el desarrollo de otras iteraciones puede ser gestado en paralelo permitiendo reducir el tiempo de desarrollo sin desmedro de la calidad final del *software*.

Test-Driven Development sigue la siguiente estructura:

- Diseño de la arquitectura de alto nivel.
- Codificación
  - Prueba unitaria

[6] http://agilemanifesto.org/iso/es/manifesto.html
[7] http://agilemanifesto.org/iso/es/principles.html

- Codificación
- Refactorización

- Testing

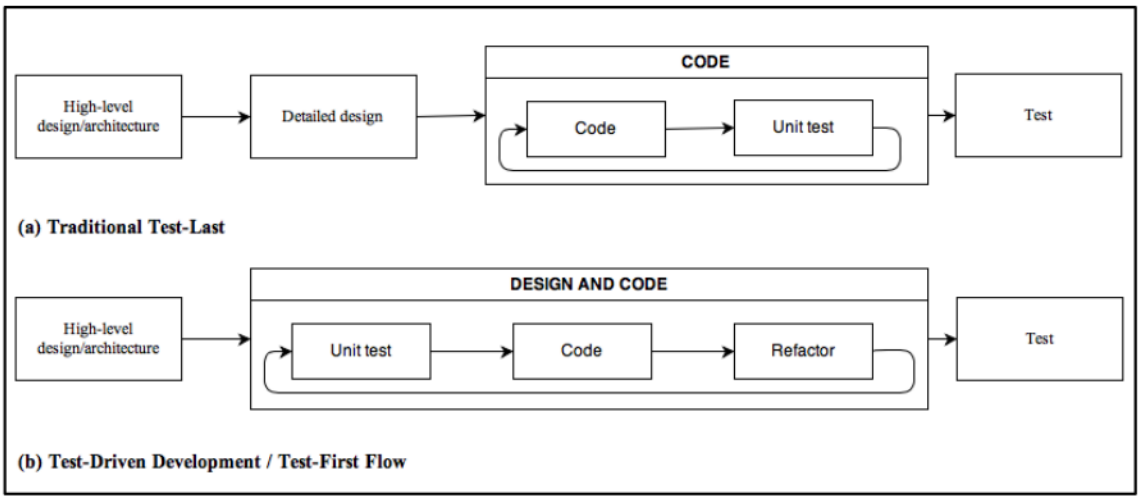


Figura 3.1: Model of Janzen and Saiedian (2008) - Traditional Development vs. TDD

La figura 3.1 muestra la secuencia de los procesos según *Test-Driven Development* (a) en contraposición con la secuencia tradicional (b).

En base a la metodología seguida, los comentarios dentro del código son esenciales para una buena refactorización de este, permitiendo que sea legible. Como fue explicado, debido al ciclo continuo de prueba, codificación y refactorización es necesaria una rápida comprensión del algoritmo y su funcionalidad. Esta secuencia cíclica facilita reescribir el código de una mejor forma, optimizando no solo el funcionamiento de este si no que eliminando bloques innecesarios o duplicados.

Por otro lado, una de las desventajas de emplear *TDD* es que debido a su lógica de diseño emergente carece de un esquema de diseño detallado del sistema completo, que podría ser obtenido con otras metodologías, como por ejemplo, un enfoque de diseño orientado a objetos (*OOD*). Para el caso particular de *Gerprin*, no es un punto crucial, debido a que el sistema fue desarrollado con anterioridad y se conocen las interacciones entre los módulos y las funciones que los componen de sobremanera.

3.2. Definición de actividades

El ciclo de vida de un proyecto desarrollado empleando XP está compuesto de seis fases[8]:

1. Exploración
2. Planificación de las entregas
3. Iteraciones
4. Integración y pruebas
5. Aceptación y entrega
6. Muerte del proyecto

[8] http://www.cyta.com.ar/ta0502/v5n2a1.htm

3.2.1. Exploración

Exploración es la actividad inicial del ciclo de vida del desarrollo de un proyecto empleando *XP* como metodología. Consiste en la realización de las primeras interacciones con el proyecto, tomando conocimiento en líneas generales de las historias de usuario del *software* a desarrollar.

Es importante en esta etapa definir las herramientas, metodologías, canales de comunicación y *framework* que se utilizaran, además, realizar un primer acercamiento de estas al equipo de trabajo, para familiarizarse con su uso.

3.2.2. Planificación de las entregas

En la etapa de planificación de la entrega se deben definir las prioridades de las historias de usuario y el esfuerzo que debe ser empleado en el desarrollo de dicha historia. Lo anterior, con el objetivo de determinar el orden y contenido de los entregables.

Para lo anterior se debe elaborar un informe con las historias de usuario que serán desarrolladas en cada entregable y la estimación del tiempo que tardará en ser completado. El documento debe definir la fecha de finalización de cada una de las iteraciones programadas separadas en plazos de no más de tres semanas.

Esta etapa se debe repetir antes de cada iteración para definir el contenido del siguiente entregable y corregir, de ser necesario, las estimaciones realizadas en iteraciones anteriores.

Con el fin de obtener una retroalimentación del tiempo de trabajo y poder corregir a tiempo problemas de atraso que se estén produciendo, se empleará un sistema de puntaje para medir el tiempo de desarrollo de las historias donde un punto equivale a una semana de trabajo de una persona. De este modo cada historia tiene un estimado de trabajo y el valor real de la labor realizada, buscando que dichos valores sean lo más similar posible

3.2.3. Iteraciones

Las iteraciones corresponden a varias entregas que se realizan de forma periódica basadas en el documento generado en la etapa anterior.

En la primera iteración se debe realizar un diseño de la arquitectura del sistema, definiendo a grandes rasgos la relación entre las partes que lo componen. Lo anterior se logra realizando un breve análisis de las historias de usuario que fuercen el uso de una arquitectura especifica.

Debido a que la lógica de trabajo se basa en diseño emergente, no es necesario un gran detalle en el diseño, esto debido a que se debe consdierar que dicho diseño puede verse afectado por futuras decisiones en el desarrollo.

En esta etapa se adopta el flujo de trabajo de *TDD*, donde antes de comenzar los trabajos de programación se escriben las pruebas unitarias que deben ser aprobadas por los distintos procesos a desarrollar. Dichas pruebas deben ser realizadas en paralelo a la codificación del programa, de esta forma, cada prueba da lugar a una refactorización del código, permitiendo que el diseño emerja desde la optimización del mismo.

Para comprender de una mejor forma esta lógica de trabajo se debe entender que cada prueba que dé a conocer un error del programa es una prueba exitosa, esto debido a que nos permite conocer los errores antes de que estos pacen al proceso de producción.

3.2.4. Integración y pruebas

La etapa de Integración y pruebas consiste en la instalación de la aplicación en su ubicación final, donde se realizarán pruebas de integración que permiten comprobar que las configuraciones y relaciones

entre las partes del sistema funcionen correctamente.

Junto con lo anterior, se deben considerar nuevas funcionalidades para la aplicación que no fueron incluidas en las fases anteriores y que son deseables para nuevas versiones.

3.2.5. Aceptación y entrega

Para esta etapa la aplicación debe ser entregada mientras se encuentra en los servidores de producción, con el objetivo de resolver situaciones que presentaron problemas en las pruebas de integración y desarrollar las funcionalidades que fueron solicitadas en la etapa anterior.

3.2.6. Muerte del proyecto

Debido a que el proyecto no recibirá más cambios, en esta etapa se debe desarrollar un manual de usuario explicando las funcionalidades del sistema que contenga el propósito de cada función, como usarlas correctamente, los valores configurables, su significado y las restricciones de estos.

Además, se deberá realizar un documento de diseño de la arquitectura final del sistema con el propósito de documentar el funcionamiento de este en caso de futuras mejoras.

3.3. Productos de Trabajo

El manifiesto ágil, en el segundo elemento que se ha aprendido a valorar dice “Software funcionando sobre documentación extensiva”. Debido a esto los productos de trabajo generados por cada una de las etapas van enfocadas a un software funcional más que a la documentación.

Sin embargo, el mismo manifiesta concluye “aunque valoramos los elementos de la derecha (documentación), valoramos más los de la izquierda (código funcionando)”. Los productos de trabajo que no corresponden a código funcional, es decir, corresponden a documentos, planes de trabajo, manuales o especificaciones deben encontrarse en proporción al trabajo a realizar.

A continuación, se detallan los productos de trabajo que deben ser generados:

1. Especificación de Requerimientos
2. Planificación de proyecto
3. Plan de pruebas
4. Especificación del Sistema
5. Plan de Aseguramiento de Calidad (*SQA*)
6. Manual de usuario

3.3.1. Especificación de Requerimientos

Para comprender de mejor forma el sistema que se busca construir, es necesario realizar un proceso de documentación esquemática de los requerimientos que forman parte del sistema.

La especificación de requerimiento es un documento que mediante la declaración y definición de los requerimientos funcionales y no funcionales del sistema describe el comportamiento completo de este.

Basado en Garcia (2015), el siguiente esquema muestra el índice de contenidos del documento.

1. **Introducción**
   - *a*) Propósito del documento
   - *b*) Alcance del producto
   - *c*) Definiciones, acrónimos y abreviaturas
   - *d*) Referencias
   - *e*) Descripción del resto del documento
2. **Descripción general**
   - *a*) Perspectiva del producto
   - *b*) Funciones del producto
   - *c*) Características del usuario
   - *d*) Restricciones generales
   - *e*) Suposiciones y dependencias
3. **Requerimientos Específicos**

   Incluye los requerimientos funcionales, no funcionales y de interfaz. Obviamente esta es la parte más sustancial del documento, pero debido a la amplia variedad en la práctica organizacional, no es apropiado definir una estructura estándar para esta sección. Los requerimientos pueden documentar las interfaces externas, describir la funcionalidad y el rendimiento del sistema, especificar los requerimientos lógicos de la base de datos, las restricciones de diseño, las propiedades emergentes del sistema y las características de calidad.
4. **Apéndice**
5. **Índice**

3.3.2. Planificación de proyecto

Antes de comenzar a escribir código, es necesario planear las distintas entregas y el contenido de estas. El documento debe contener las historias de usuarios que serán incluidas en cada iteración y las fechas de entrega de cada una de estas.

Para lograr establecer una correcta planificación de las iteraciones, se deben escribir todas las historias de usuario del sistema, para luego valorarlas en termino de complejidad e importancia, estableciendo una ruta crítica de las actividades a realizar.

Se debe notar que debido a las características del trabajo de desarrollo de *software* es necesario ser flexible sobre el contenido de cada entrega, considerando cambios solicitados por el cliente, corregir errores detectados en iteraciones anteriores, entre otras actividades que pueden ser incluidas en iteraciones futuras. Debido a esto es necesario contemplar modificar la planificación con cada nueva iteración.

El esquema muestra el índice de contenido del producto de trabajo, basado en Garcia (2015)

1. **Introducción**
   - *a*) Objetivo
   - *b*) Alcance

2. **Estimación de esfuerzo y duración**
   a) Estimación de esfuerzo
3. **Organización del Proyecto**
   Describe la forma en que el equipo de desarrollo está organizado, la gente involucrada y sus roles.
4. **Planificación**
   Descomposición detallada de las actividades. Señala el *hardware* y *software* de ayuda requeridos para llevar a cabo el desarrollo. Describe la división del proyecto en actividades e identifica hitos asociados a cada actividad.
5. **Estrategia del seguimiento**
6. **Interfaz externa al proyecto**
7. **Identificación y análisis de los riesgos**

3.3.3. Plan de pruebas

Uno de los ejes fundamentales al trabajar bajo *TDD* son las pruebas constantes que deben ser realizadas para asegurar un correcto funcionamiento. El plan de pruebas, al igual que la planificación de iteraciones, es un documento que sufrirá modificaciones y será escrito a lo largo del desarrollo del proyecto.

Ante de comenzar la labor de codificación de cada iteración, se deben escribir las distintas pruebas que confirmaran la integración y funcionamiento correcto de cada entregable.

Antes de desarrollar cualquier función se deben establecer las siguientes pruebas.

Pruebas unitarias:

Son pruebas que aseguran el correcto funcionamiento de una unidad específica del sistema.

Pruebas de integración:

Pruebas que comprueban que la correcta integración de todos los elementos unitarios que componen el sistema.

Pruebas de aceptación:

Las pruebas de aceptación son realizadas en la última iteración y prueban el sistema completo antes de ser instalado en ambiente productivo.

Pruebas de sistema:

Las pruebas de sistema realizan una comparación entre los objetivos originales del sistema, aquellos planteados en la toma de requisitos, y los procesos, actividades y rutinas del sistema desarrollado.

El esquema muestra el índice de contenido del producto de trabajo, basado en Garcia (2015)

1. **Introducción**
   a) Objetivo
   b) Alcance
2. **Definición de elementos a probar**
3. **Definición de la estrategia a utilizar**

Describe los recursos a emplear. Especificación de técnica que se usará. Herramientas utilizadas. Criterio indicador de éxito de las pruebas.

4. **Equipo de pruebas**

3.3.4. Especificación del Sistema

Debido a que la labor de *software* es un trabajo realizado en equipo, es necesario dejar plasmada la arquitectura del sistema y explicar a grandes rasgos el funcionamiento de los distintos procesos. Esto con el objetivo de ayudar a futuros desarrolladores que realicen labores de mantención o actualizaciones al sistema.

Las especificaciones del sistema deben contener la arquitectura y descripción de las funciones que componen cada una de las distintas partes.

El esquema muestra el índice de contenido del producto de trabajo, basado en Garcia (2015)

1. **Introducción**
   a) Objetivo
   b) Alcance
2. **Identificación de necesidades** Describe el análisis técnico y económico. Se debe realizar un análisis para evaluar su visibilidad
3. **clasificación de funciones**
   a) Funciones de *Software*
   b) Funciones de *Hardware*
   c) Funciones del personal
4. **Restricciones del sistema**
5. **Esquema relacional del sistema**

3.3.5. Plan de Aseguramiento de Calidad (*SQA*)

Es necesario asegurar la calidad del *software* desarrollado, garantizando el cumplimiento de los distintos requerimientos desarrollados.

El documento es un resumen de las actividades desarrolladas y la confirmación de su cumplimiento, mediante la validación de los resultados. En el plan de aseguramiento de calidad se debe especificar los resultados obtenidos y compararlos con los resultados esperados.

El documento debe contener las modificaciones, revisiones y demás actividades que se realizaron en caso de no obtener los resultados esperados.

El esquema muestra el índice de contenido del producto de trabajo, basado en Garcia (2015)

1. **Introducción**
   a) Objetivo
   b) Alcance

c) Definiciones

d) Resumen

2. **Gestión SQA**

a) Actividades

b) Resultado obtenido en las actividades

c) Revisiones y auditorías

3.3.6. Manual de usuario

Final mente el manual de usuario es un documento que tiene como objetivo describir el funcionamiento del sistema al usuario final.

El documento debe explicar el funcionamiento y propósito de cada una de las historias de usuario que componen el sistema. De este modo el usuario comprende el propósito y uso correcto de *software* y *hardware*.

En la sección final del documento debe incluir la forma de contacto al área de soporte, con el fin de aclarar las consultas más específicas.

El esquema muestra el índice de contenido del producto de trabajo, basado en Garcia (2015)

1. **Prefacio**

a) Resumen

b) Cómo usar el manual

2. **Índice**
3. **Modelo del sistema**
4. **Funciones principales del sistema**
5. **Sección de preguntas**

a) preguntas frecuentes

6. **Contáctenos**

3.4. Hitos del desarrollo (Atributos de Calidad)

3.4.1. Por actividades

Tabla 3.1: Hitos del desarrollo por actividades.

Actividad	**Atributo**
Exploración	Se deben respetar los plazos de entrega del proyecto. Fiabilidad
Planificación de las entregas	Debe ser fiable, respetando los plazos y fechas definidos junto al cliente y al mismo tiempo, debe ser flexible, permitiendo modificaciones y actualizaciones. El cliente debe estar de acuerdo con las fechas de entrega y el contenido de estas.
Iteraciones	El trabajo realizado en esta etapa, debe ser eficiente y cumplir en un 100 % con lo establecido en las actividades anteriores. El producto debe ser fiable, mantenible, multiplataforma y adaptable. Respecto a los entregables, el cliente debe aceptar el 100 % de las funcionalidades del sistema.
Integración y pruebas	Al igual que en la etapa anterior, el trabajo debe ser eficiente, además, debe ser mantenible y el producto debe ser fiable. Se deben corregir el 100 % de las fallas encontradas en la actividad anterior.
Aceptación y entrega	El trabajo debe ser adaptable y escalable. El proceso de instalación debe ser eficiente y fiable. En esta etapa el 100 % de las funcionalidades del sistema deben ser aceptadas por el cliente.
Muerte del proyecto	El 100 % de las funciones deben encontrarse documentadas, tanto en lenguaje técnico como en lenguaje común. En caso de encontrar fallas estas deben ser solucionadas en un plazo máximo de 8 horas hábiles. Las tareas realizadas en esta etapa deben ser mantenibles, eficientes y adaptables.

3.4.2. Por producto

Tabla 3.2: Hitos del desarrollo por productos.

Producto	Atributo
Especificación de Requerimientos	El documento debe contemplar el 100 % de los requerimientos acordados con el cliente. Debe ser fiable y escalable.
Planificación de proyecto	Mantenible y adaptable. Debe contener el 100 % de las actividades a realizadas.
Plan de pruebas	Mantenible, adaptable y fiable. Debe contener pruebas unitarias, de integración, compatibilidad y aceptación.
Especificación del Sistema	Fiable, adaptable y mantenible. Debe ser aprobado en un 100 % por el equipo de desarrollo y el cliente.
Plan de Aseguramiento de Calidad (SQA)	Mantenible, adaptable, fiable, seguro, escalable. Debe comprobar en un 100 % la calidad del sistema.
Manual de usuario	Fiable. Debe ser comprensible en un 100 % por el cliente, debe contener un índice y no superar la extensión de 40 paginas.

3.5. Puntos de revisión

Los puntos de revisión en cada una de las distintas etapas son los siguientes:

1. Exploración
   - *a*) Revisión y aprobación de las especificaciones de requerimientos
   - *b*) Revisión y aprobación de las historias de usuario (preliminar)
2. Planificación de las entregas
   - *a*) Revisión y aprobación de la actualización de las historias de usuario (se actualiza en cada iteración)
   - *b*) Revisión y aprobación de la actualización de la planificación de la Iteración (se actualiza en cada iteración)
   - *c*) Revisión y aprobación de plan de pruebas unitarias, de integración y de sistema (se actualiza en cada iteración)
3. Iteraciones
   - *a*) Revisión y aprobación de la actualización del modelo del modelo entidad relación
   - *b*) Revisión y aprobación del código (se actualiza en cada iteración)
   - *c*) Revisión y aprobación de la actualización del documento de especificaciones del sistema (se actualiza en cada iteración)
   - *d*) Revisión y aprobación de las pruebas unitarias, de aceptación y de integración.

e) Revisión y aprobación de documentación de resultado asociado a las pruebas unitarias, de aceptación y de integración.

f) Revisión de la actualización Plan de Aseguramiento de Calidad (se actualiza en cada iteración)

4. Integración y pruebas

a) Revisión y aprobación del proceso de integración.

b) Revisión y aprobación de la prueba de integración.

c) Revisión y aprobación de documentación de resultado asociado a las pruebas de integración.

d) Revisión y aprobación del documento de especificaciones del sistema

e) Revisión de la actualización Plan de Aseguramiento de Calidad (se actualiza en cada iteración).

5. Aceptación y entrega.

a) Revisión y aprobación de modificaciones en el código del sistema.

b) Revisión y aprobación de las especificaciones del sistema asociada a las modificaciones.

c) Revisión de la actualización Plan de Aseguramiento de Calidad.

6. Muerte del proyecto.

a) Revisión y aprobación del manual de usuario

b) Revisión de la actualización Plan de Aseguramiento de Calidad.

EX UMBRA
IN
SOLEM

4 | Gestion de calidad

4.1. Equipo de trabajo

A continuación, se presenta un esquema que muestra la estructura del equipo de trabajo basado en Herramientas específicas para SQA y SCM (2000)

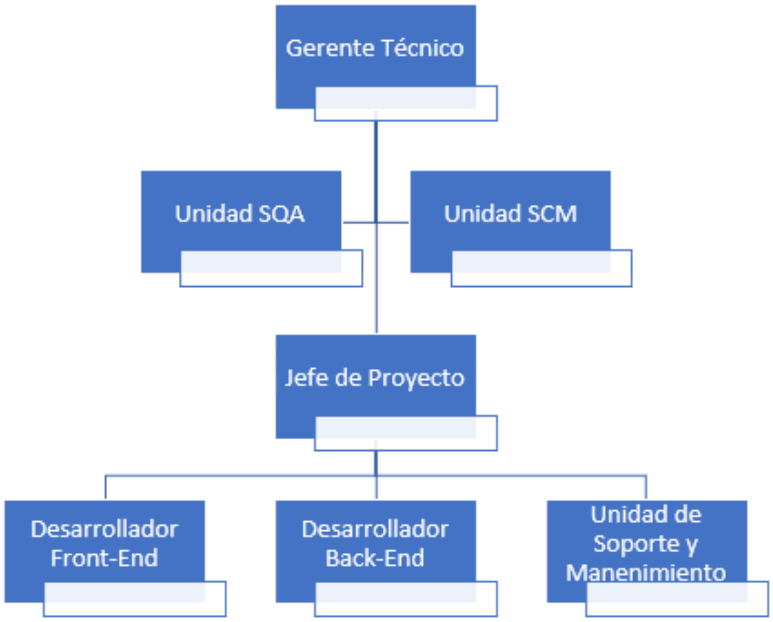


Figura 4.1: Equipo de trabajo

Según Don Wells, autor de las reglas de *Extreme Programming*, el equipo debe tener dos características transversales:

- **Coraje**: Se debe ser sincero respecto al progreso y plazos del proyecto. Los trabajadores forman parte de un equipo, por lo que no deben temer a nada
- **Respeto**: Se debe respetar a todos los miembros del equipo y clientes. Cada trabajador realiza un aporte valioso al equipo, por lo que debe ser respetado.

4.2. Recursos

4.2.1. Personal

4.2.1.1. Gerente Técnico

Las responsabilidades del gerente técnico son:

- Establecer y seleccionar el lineamiento de un programa de calidad para el desarrollo de *software*.
- Revisar y aprobar el plan de *SQA* para cada proyecto.
- Facilitar al equipo los medios para la realización de las actividades asociadas al plan de *SQA*.
- Monitorear las actividades de *SQA*.

4.2.1.2. Unidad *SQA*

Son los encargados de generar el plan de aseguramiento de calidad que incluyen el aseguramiento de calidad de entregables y la documentación. Si bien, son los encargados de realizar el plan de calidad, no son encargados de velar por la realización de las actividades asociadas al plan o de prácticas, procesos y procedimientos. Dicha responsabilidad recae en el jefe de proyecto y encargados de cada área.

Las responsabilidades de la unidad de *SQA* son:

- Establecer los estándares de calidad.
- Observar, participar y verificar que las revisiones de calidad se realicen correctamente.
- Velar por la correcta realización de las distintas pruebas y procedimientos establecidos de acuerdo con el plan de pruebas.
- Crear un plan de identificación de defectos para el proceso de desarrollo y apoyar a áreas de gestión en perfeccionar estos.

4.2.1.3. Unidad *SCM*

La unidad de Gestión de Configuración del *Software* es la encargada de mantener la integridad del producto a lo largo de todo el desarrollo del proyecto, mediante un control de cambios adecuado para el desarrollo.

Las responsabilidades de *SCM* asociadas a *SQA* son:

- Revisar y comentar el plan de *SQA* para el proyecto.
- Implementar las actividades de calidad de acuerdo con el plan de *SQA*.
- Resolver los problemas detectados por *SQA* relacionados con *SCM*.
- Implementar las prácticas, procesos y procedimientos definidos en el plan de *SCM* y en otros planes o documentos complementarios.

4.2.1.4. Jefe de Proyecto

Las responsabilidades del jefe de proyecto son:

- Establecer un programa de calidad para el proyecto de desarrollo de *software* de acuerdo con las políticas organizacionales.

- Identificar las actividades de *SQA* requeridas para el proyecto.
- Revisar y aprobar el plan de *SQA* para el proyecto.
- Identificar los participantes de las actividades de *SQA*.
- Implementar las actividades de *SQA* de acuerdo con el plan.
- Monitorear las actividades de *SQA* planificadas en el plan.
- Identificar los factores de calidad para la implementación del software.
- Identificar, desarrollar y mantener la documentación del proyecto.

4.2.1.5. Desarrolladores *Front-End*, Desarrolladores *Back-End* y Soporte y mantenimiento

El área de soporte y mantenimiento es el encargado de realizar correcciones y solucionar problemas relacionados con el adecuado funcionamiento del sistema una vez este se encuentre en ambientes productivos. Esto se traduce en identificar causas de errores mediante el registro de actividades de la aplicación y dar soporte a consultas del cliente.

Por otro lado, el área de desarrollo se divide en desarrolladores *Front-End* y *Back-End* que hace referencia a la capa del sistema sobre la cual realizan su trabajo de desarrollo.

El desarrollador *Front-End* trabaja sobre la capa de presentación de datos, es decir, la que interactúa directamente con el usuario. Mientras el desarrollador *Back-End* trabaja sobre la capa de acceso de datos que se encarga de procesar la información proveniente de la capa de presentación

Sus responsabilidades con *SQA* son:

- Revisar y entregar sus observaciones sobre el plan de *SQA* para el proyecto.
- Implementar las actividades de *SQA* de acuerdo con el plan.
- Participar de la solución de los problemas detectados por las actividades de *SQA* que sean de su competencia.
- Implementar las prácticas, procesos y procedimientos definidos en el plan de proyecto y en otros planes o documentos complementarios.

4.2.2. Infraestructura

Para un buen desarrollo del sistema, además de los recursos humanos y la realización de las actividades de *SQA* es necesario contar una infraestructura que pueda satisfacer las necesidades del equipo para la realización del software.

4.2.2.1. Oficina

Es necesaria una oficina que posea áreas de trabajo y salas de reuniones. Respecto a las áreas de trabajo estas deben contar con escritorios y sillas ergonómicas recomendadas para trabajo de escritorio que cuenten con altura variable y soporte para brazos y espalda. Los escritorios deben tener un largo no inferior a un metro y un ancho no inferior a 70 cm.

Respecto a las oficinas de reuniones, esta debe tenera disposición un televisor que permita conexión a un equipo (*notebook* o dispositivo móvil) o en su defecto un proyector.

Es necesario que en la oficina existan los servicios básicos como luz y agua potable. Finalmente es necesario contar con una conexión de internet no menor de 50 Mbps.

4.2.2.2. Equipamiento de trabajo

Es necesario mínimo de 5 equipos de trabajo enfocados en el desarrollo.

- **Procesador**: Intel Core i7-7700.
- **Memoria**: 12GB DDR4 (2400 MHz).
- **Almacenamiento**: Unidad SSD 240GB Lectura 540 MB/s Escritura 465 MB/s.
- **Pantalla**: Dos pantallas de mínimo 17".

Además, es necesario 3 equipos Laptop para uso administrativo.

- **Procesador**: Intel Core i5 7200U.
- **Memoria**: 8GB DDR4 (2400 MHz).
- **Almacenamiento**: HDD 1TB (5400rpm).

4.2.3. Actividades

El área de *SQA*, con el objetivo de asegurar la calidad de forma continua, se debe encargar de varias actividades a lo largo del proceso de desarrollo de *Gerprin*. Es importante que *SQA* realice las definiciones de estándares, revisión de productos y procesos, pruebas y análisis de defectos hasta la entrega exitosa de las últimas modificaciones y posteriores documentaciones del sistema. Para esto, debe trabajar en conjunto con las áreas de desarrollo, tanto *Front-End*, *Back-End*, unidades de soporte, mantenimiento e implantación.

Las siguientes actividades se encuentran basadas en Herramientas específicas para el grupo de *SQA* Herramientas específicas para SQA y SCM (2000).

4.2.3.1. Evaluación de la selección de los productos de trabajo

El *SQA* debe asistir al jefe de proyectos en la elaboración de los estándares y guías aplicables que conforman los distintos artefactos necesarios para la planificación inicial del proyecto. Entre los elementos que deben ser considerados son herramientas por utilizar, diseño preliminar y *mockup* de la interfaz del sistema, entre otros.

4.2.3.2. Evaluación de las herramientas

En esta actividad el *SQA* tiene la responsabilidad de evaluar las herramientas que serán empleadas para el desarrollo del sistema, considerando herramientas para pruebas, evaluaciones de rendimiento de la aplicación y herramientas de administración de proyectos.

Por otro lado, junto con el jefe de proyecto deben ser decididas las herramientas empleadas directamente en el proceso de desarrollo como *frameworks*, herramientas de despliegue de aplicaciones, entre otros.

4.2.3.3. Evaluación de la planificación y el monitoreo del proyecto

SQA es responsable de la elaboración del plan de aseguramiento de calidad, la elaboración e identificación de guías y estándares que puedan ser aplicados a las iteraciones y entregables.

4.2.3.4. Evaluación de la especificación de requerimientos

Respecto a las especificaciones de requerimiento *SQA* debe realizar las siguientes actividades:

- Verificar que se cumplan correctamente las actividades y procesos de la fase de exploración.
- Garantizar que se revisaron adecuadamente los entregables (especificación del sistema y de requerimientos) de la fase de exploración.
- Asegurar la inclusión de la corrección de los entregables según las observaciones realizadas en el proceso de revisión.
- Corroborar que estén expresados y documentados los requerimientos funcionales, técnicos, operacionales y de interfaz, de manera tal que puedan ser verificados en el producto final.

4.2.3.5. Evaluación del diseño

SQA es responsable de:

- Garantizar que se revisaron adecuadamente los entregables (diseño preliminar, diseño detallado, plan de pruebas, especificación de casos y procedimientos de prueba) de la fase de planificación de las entregas e iteraciones.
- Asegurar la inclusión de la corrección de los entregables según las observaciones realizadas en el proceso de revisión.

4.2.3.6. Evaluación de la implementación y de la prueba de unidad

SQA debe:

- Garantizar que el proceso de codificación, las revisiones asociadas y la prueba de unidad sean conducidos de acuerdo a lo señalado en el plan de pruebas.
- Asegurar la inclusión de la corrección de los entregables según las observaciones realizadas en el proceso de revisión.
- Verificar la implementación de las acciones correctivas derivadas de la prueba de unidad.
- Comprobar la utilización de la especificación de procedimientos y casos de prueba durante la prueba de unidad.
- Corroborar la documentación del código y de los resultados de la prueba de unidad.

4.2.3.7. Evaluación de la integración y prueba

SQA es responsable de:

- Verificar que el proceso de integración y las actividades de prueba sean realizadas conforme al plan de proyecto, el diseño, el plan de prueba y los estándares y procedimientos establecidos.
- Asegurar que la prueba de integración haya sido completada satisfactoriamente, que sus resultados fueron registrados y divulgados y que las acciones correctivas derivadas de ella fueron implementadas.
- Corroborara el desarrollo adecuado de las pruebas de aceptación y del sistema.
- Monitorear las actividades de prueba y certificar sus resultados.
- Revisar las pruebas.

4.2.3.8. Evaluación del producto antes de su liberación

Se deben realizar las evaluaciones del producto terminado y la documentación. Para esto es necesario que el área de *SQA* sea participe de las auditorias funcionales y físicas.

4.2.3.9. Evaluación del proceso de revisión

SQA es responsable de garantizar que todos los entregables sean revisados y corregidos en caso de ser necesario y analizar los problemas encontrados de una forma sistémica, identificando causas, impactos y frecuencia de ocurrencia.

4.2.3.10. Evaluación de las acciones correctivas

SQA es responsable de establecer acciones preventivas y monitorear que estas sean implementadas de forma correcta.

4.2.3.11. Evaluación del proceso de *SCM*

SQA debe:

- Revisar el plan de *SCM*.
- Asegurar la correcta identificación de los ítems de configuración.
- Garantizar un adecuado control de cambios.
- Corroborar que la contabilidad del estado de la configuración sea preparada oportunamente y que refleje la situación real de los ítems de configuración en relación con el proyecto.
- Comprobar la adherencia de las actividades de *SCM* al plan de *SCM*.
- Verificar el correcto funcionamiento de la librería del *software*.

4.2.3.12. Verificar la implementación de los procesos

SQA debe velar por la correcta implementación y cumplimiento de estándares y procesos definidos en las especificaciones de requisitos y planificación de iteraciones.

4.2.3.13. Establecer las auditorías

SQA es responsable en la institución por el desarrollo de las auditorías internas. Por lo tanto, debe gestionarlas de ser preciso.

Además, es su responsabilidad participar en la auditoría física y funcional.

4.2.3.14. Responsabilidades

El área de *SQA* es responsable de revisar que los productos de trabajo se adecuen a los estándares, procedimientos y al plan de proyecto, además de dar seguimiento a las actividades realizadas a lo largo del desarrollo del proyecto. Todo lo anteriormente observado debe ser informado al jefe de proyecto y de ser necesario al jefe de proyecto.

En la siguiente tabla se adjunta la matriz de responsabilidades Herramientas específicas para SQA y SCM (2000).

Tabla 4.1: Responsabilidades por actividades.

Actividad	**GT**	**JP**	**SQA**	**SCM**	**Analista**	**Front-End**	**Back-End**	**Tester**
Evaluación de la selección los productos de trabajo.	X	X	X	X	X			
Evaluación de las herramientas	X	X	X					
Evaluación de la planificación y el monitoreo del proyecto	X	X	X	X				
Evaluación de la especificación de requerimientos	X	X	X		X	X	X	X
Evaluación del diseño	X	X	X		X	X	X	X
Evaluación de la implementación y de la prueba de unidad			X			X	X	X
Evaluación de la integración y prueba			X			X	X	X
Evaluación del producto antes de su liberación		X	X		X	X	X	X
Evaluación del proceso de revisión		X	X					
Evaluación de las acciones correctivas	X	X	X	X				
Evaluación del proceso de *SCM*	X	X	X	X				
Verificar la implementación de los procesos		X	X					
Establecer las auditorías		X	X	X				

EX UMBRA
IN
SOLEM

5 | Herramientas, técnicas y metodologías

En el presente capitulo se identifican técnicas y herramientas que serán implementadas por el equipo de SQA para un aseguramiento de calidad eficiente y efectivo.

Para actividades de IT existen muchas herramientas y técnicas que permiten disminuir la cantidad de errores y facilitan la correcta ejecución de distintas actividades del desarrollo del software. A continuación, se procederá a indicar actividades que garanticen el progreso de la labor de SQA hacia un cumplimiento de los objetivos de este.

5.1. Técnicas

Una de las técnicas principales a emplear en el desarrollo del software corresponde a TDD, que fue descrita anteriormente (3.1). Esta técnica no solo permite reducir los errores de cada release, también permite una detección temprana de estos.

5.2. Herramientas

Serán empleadas distintas herramientas para las actividades de SQA. A continuación se detallan estas y el objetivo de su uso.

- El canal de comunicación formal será mediante correo electrónico. Adicionalmente se empleará Discord como un canal de comunicación secundario para realizar conversaciones de carácter inmediato.
- Se emplearán procesadores de texto online para la documentación de resultados tales como Microsoft office 365.
- Para el ingreso, edición, recolección y asignación de los defectos encontrados se empleará ActiveCollab. Además, permite la creación de carta Gantt y asignación de actividades.
- Para la generación de test automáticos y casos de pruebas se realizará con el software SoapUI.
- Para mejorar el uso de un sistema de control de versiones se empleará Git Flow, que permite estandarizar el flujo de trabajo de Git.

Finalmente se empleará como IDE de trabajo PyCharm que permite identificar de forma rápida errores comunes de programación como variables no definidas o código duplicado entre otras.

5.3. Revisiones

Las revisiones son una metodología empleada en SQA, utilizada para la detección temprana de desviaciones y defectos. El objetivo de las revisiones es visibilizar el proceso de desarrollo con el fin de realizar un monitoreo a productos y procesos.

SQA debe velar por la correcta realización y documentación de las revisiones. Adicionalmente es responsable por la correcta asignación, documentación y programación de las actividades que deben ser realizadas al detectar cualquier anomalía en estas.

Basado en Herramientas específicas para SQA y SCM (2000) se procede a describir los participantes y de forma seguida, las etapas que conforman parte del proceso de revisión.

5.3.1. Roles y responsabilidades

Los actores descritos a continuación deben guiarse por el checklist de la revisión adjunto en anexos.

5.3.1.1. Moderador

El moderador es el encargado de facilitar todo el proceso de revisión, debe liderar el resto del equipo, coordinar las actividades, trabajar con los jefes de proyecto de otras secciones, preparar las inspecciones coordinando al equipo de trabajo, determinar el alcance la de inspección e informar los productos de las actividades de inspección a jefes de proyecto.

5.3.1.2. Autor

Corresponde al autor del producto a ser inspeccionado. Debe preparar el producto para ser inspeccionado, realizar recomendaciones de personas que participen en el proceso de inspección, explicar el producto a ser inspeccionado, corregir defectos detectados y participar de las acciones correctivas posteriores.

5.3.1.3. Presentador

El presentador es el encargado de introducir el producto de trabajo al resto del equipo encargado de la inspección, debe comprender a cabalidad el producto con el objetivo de guiar al equipo de trabajo en el proceso de inspección.

5.3.1.4. Inspector

Es el encargado de inspeccionar el producto de trabajo e identificar errores y defectos. Se encarga del proceso de inspección según las instrucciones entregadas por el moderador.

5.3.1.5. Secretario

Se encarga de realizar el registro de las observaciones, identificando y clasificando los defectos encontrados, participa de la inspección en igualdad al resto del equipo y una vez finalizada la inspección debe leer cada defecto y su clasificación. Adicionalmente se encarga de anotar acciones correctivas y los plazos de estas.

5.3.1.6. Observador

El rol de observador es opcional. Se encarga de observar y apuntar la información de los participantes del proceso de revisión y del proceso mismo con el objetivo de poder realizar mejora continua a las actividades realizadas.

5.3.2. Etapas de Revisión

5.3.2.1. Planificación

Objetivo
El objetivo de esta etapa es verificar que el producto se encuentre en condiciones de ser inspeccionado y planificar las actividades para garantizar el éxito de la revisión.

Participantes
Moderador y el autor.

Criterios de entrada

1. El autor señala que el producto de trabajo está listo para la revisión.
2. La existencia de participantes apropiados y con disponibilidad de tiempo.

Actividades

1. Selección de los participantes y asignación de roles.
2. Determinar el tamaño del producto de trabajo por ser revisado.
3. Determinar los criterios que el producto debe satisfacer.
4. Establecer la necesidad de una reunión orientación.
5. Planificar la reunión de orientación y/o la inspección.
6. Preparación y distribución del paquete de revisión a los participantes.
7. Registro del tiempo usado durante la fase de planificación.

Criterios de salida
Cada participante completa satisfactoriamente la checklist de las tareas asignadas a la etapa de planificación.

5.3.2.2. Orientación (opcional)

Objetivo
El objetivo de esta reunión es instruir a los participantes sobre el producto de trabajo y reconocer el proceso de revisión que se aplicará a dicho producto.

Participantes
Moderador, autor y cualquier otro participante que requiera información sobre el producto o el proceso de revisión.

Criterios de entrada
Los criterios de salida de la etapa de planificación han sido completados satisfactoriamente.

Actividades

1. Descripción general del producto de trabajo sujeto a inspección.
2. Revisar la asignación de roles y el plan para la inspección.
3. Responder cualquier consulta.

Criterios de salida

1. Todos los participantes están preparados para proceder.
2. Cada participante completa satisfactoriamente la checklist de las tareas asignadas a la etapa de orientación.

5.3.2.3. Preparación

Objetivo
El objetivo de esta etapa es que cada miembro evalúe el producto de trabajo para detectar defectos, clasifique estos defectos y los documente.

Participantes
Moderador, presentador y cualquier participante bajo el rol de inspector.

Criterios de entrada
Los criterios de salida de la etapa de planificación y de la orientación han sido completados satisfactoriamente.

Actividades

1. Los inspectores revisan el producto de trabajo y registran los defectos detectados.
2. Clasificar todos los defectos.
3. El presentador se prepara para exponer el producto durante la reunión de inspección.
4. Entrega del informe de revisión.

Criterios de salida

1. Los defectos han sido correctamente documentados.
2. Todos los participantes están preparados para proceder en la inspección.
3. Cada participante completa satisfactoriamente la checklist de las tareas asignadas a la etapa de preparación.

5.3.2.4. Inspección

Objetivo

Los objetivos de esta etapa son que los participantes lleguen a un consenso respecto de los defectos que afectan al producto de trabajo, decidan si requiere de una revisión adicional, y que discutan y documenten sobre las lecciones aprendidas durante el proceso.

Participantes

Moderador, presentador, autor, inspector(es), secretario y, opcionalmente, un observador.

Criterios de entrada

Todos los participantes pueden participar de la inspección.

Actividades

1. Se enuncian los roles, el enfoque y las guías para llevar a cabo la inspección.
2. El presentador describe el producto de trabajo.
3. Se nombran los defectos globales que afectan la completitud del producto de trabajo.
4. Se nombran los defectos específicos por ser revisados y registrados.
5. Los errores de formatos son entregados al autor para acciones correctivas.
6. Asignación de las los defectos no resueltos.
7. Determinación sobre la necesidad de tiempo adicional.
8. Emplear la hora adicional para terminar las actividades restantes.
9. Establecer la necesidad de una revisión adicional.
10. Solicitar retroalimentación a los participantes en relación con la inspección.

Criterios de salida

1. Los defectos han sido registrados
2. Los defectos no resueltos han sido asignados.

5.3.2.5. Rework

Objetivo

El objetivo de esta etapa es corregir los defectos y resolver aquellos clasificados como no resueltos.

Participantes

Autor.

Criterios de entrada

Los criterios de salida de la etapa de inspección han sido completados satisfactoriamente.

Actividades

1. Estimar el tiempo requeridos para resolver los defectos no resueltos.
2. Corregir los defectos identificados en el producto de trabajo.
3. Resolver los defectos no resueltos del producto de trabajo.
4. Corregir los errores de forma identificados en el producto de trabajo.

Criterios de salida

Todas las correcciones han sido completadas.

5.3.2.6. Seguimiento

Objetivo

Los objetivos de esta etapa son asegurar que todos los defectos y los errores de forma han sido corregidos. Además, que se haya dado solución a los defectos no resueltos.

Participantes

Moderador y autor.

Criterios de entrada

Los criterios de salida de la etapa de rework han sido completados satisfactoriamente.

Actividades

1. El moderador confirma que todos los defectos y errores de forma hayan sido corregidos, y que a los defectos no resueltos se les haya dado solución.
2. Completar la documentación de la revisión.

Criterios de salida

1. Los templates de revisión han sido completados y distribuidos.
2. Todos los defectos no resueltos han sido incorporados al monitoreo del proceso.

5.3.3. Informe de revisión

Se adjunta en anexos una pauta sobre el informe de revisión.

5.4. Auditorias

Las auditorias tienen como fin comparar el estado actual de los productos de trabajo con el estado reportado, mediante la evaluación del cumplimiento de normas, estándares y otros acuerdos existentes.

El proceso de auditoria descrito en Herramientas específicas para SQA y SCM (2000) comienza cuando el iniciador identifica la necesidad de una auditoría. Esto da lugar a que un auditor elabore un plan de auditoría, el cual es expuesto a los demás auditores y a la institución auditada durante la reunión de orientación. Ya definido el curso de la auditoría los auditores pueden comenzar con la evaluación. Para ello se presentan en la institución auditada para entrevistar a los desarrolladores, revisar la documentación asociada

a los procesos examinados e inspeccionar los productos. Con la información recopilada, el auditor entrega las observaciones y conclusiones preliminares a la institución auditada en la reunión de cierre. Después de la discusión de estos resultados, el auditor desarrolla un informe de auditoría. Con este último la institución está en condiciones de definir las acciones correctivas y monitorear su implantación.

Las auditorias pueden ser realizadas tanto por un equipo formado por personas pertenecientes a la organización (auditorías internas) como por un equipo externo (auditorías externas). Adicionalmente, estas pueden ser realizadas en cualquier etapa del desarrollo del software variando en el tipo de auditoria y el uso que se le data la información generada.

5.4.1. Roles y responsabilidades

5.4.1.1. Iniciador

Es quien inicia y aprueba las auditorias, es responsable de indicar las directrices básicas que debe tener la auditoria, es decir, definir el propósito, los criterios a evaluar, el curso que se seguirá y cuáles serán los productos y/o procesos a ser auditados. Además, se debe encargar de revisar los productos generados y dirigir las acciones correctivas.

5.4.1.2. Moderador (Líder del equipo auditor)

Es el responsable de asegurar el logro de los objetivos trazados. Debe trazar los planes de la auditoria, definir el equipo auditor y dirigirlo a lo largo de todo el proceso.

5.4.1.3. Auditores

Son los encargados de examinar los productos y procesos según el plan de auditoria, registrando todas las observaciones y comunicándolas al moderador.

5.4.1.4. Institución auditada

Corresponde a la institución a ser auditada. Debe nombrar sus representantes y facilitar toda la información necesaria a los auditores.

5.4.1.5. Secretario

Es el encargado de registrar todas las anomalías, decisiones, recomendaciones y conclusiones realizadas.

5.4.1.6. Nivel de gestión

Corresponde a los directivos del nivel de gestión, si bien, no son parte explicita del equipo de auditoria deben programar las auditorias y las normas de esta, facilitar toda la información requerida y entregar entrenamiento y orientación al equipo auditado.

5.4.2. Etapas de la auditoria

5.4.2.1. Planificación

Objetivo

El objetivo principal de esta etapa es establecer el ámbito y los recursos para la auditoría, como también, planificar sus actividades.

Participantes

Iniciador, moderador (líder del equipo auditor).

Criterios de entrada

1. Una autoridad competente ha autorizado la auditoría.
2. Se encuentra disponible la información requerida para esta etapa de la auditoría.

Actividades

1. El iniciador decide la necesidad de una auditoría.
2. El moderador debe comprender el objetivo del proyecto de desarrollo de software y los productos producidos.
3. El moderador debe informarse sobre el estado de avance del proyecto.
4. Definir el ámbito de la auditoría.
5. Desarrollar un checklist para la auditoría.
6. El moderador debe presentar el plan de auditoría al iniciador para su corrección y aprobación.
7. El iniciador notifica a la institución auditada sobre el desarrollo de una auditoría.
8. El auditor selecciona los miembros del equipo de auditoría.

Criterios de salida

1. El plan de auditoría ha sido aprobado.
2. La institución auditada ha sido notificada sobre la futura auditoría.

5.4.2.2. Reunión de Orientación

Objetivo

El propósito de esta etapa es clarificar el contenido del plan de auditoría a los miembros de la institución auditada y corroborar que el equipo de auditoría comprende los objetivos del proyecto.

Participantes

Iniciador, moderador, auditores, secretario, institución auditada.

Criterios de entrada

El plan de auditoría ha sido aprobado.

Actividades

1. El moderador explica a los demás el contenido del plan de auditoría.
2. La institución auditada presenta el proyecto a los auditores.
3. Se resuelven las dudas planteadas por las partes.

Criterios de salida

1. El equipo de auditoría comprende los objetivos del proyecto.
2. La institución auditada comprende el plan de auditoría.

5.4.2.3. Evaluación

La presente etapa se divide en 3 sub etapas

Site visit

Objetivo

El propósito de esta etapa es comprobar que los productos requeridos están siendo desarrollados de acuerdo a los estándares aplicables, que el proceso se ajusta a los procedimientos definidos y que los reportes del estado del proyecto reflejan su situación actual.

Participantes

Auditores, institución auditada.

Criterios de entrada

1. El equipo de auditoría comprende los objetivos del proyecto.
2. La institución auditada comprende el plan de auditoría.
3. Los recursos solicitados en el plan de auditoría se encuentran disponibles.

Actividades

1. El equipo de auditoría comprende los objetivos del proyecto.
2. La institución auditada comprende el plan de auditoría.
3. Los recursos solicitados en el plan de auditoría se encuentran disponibles.
4. Los auditores entrevistan al equipo desarrollador.
5. Los auditores examinan los registros del proyecto.
6. Los auditores examinan los productos de trabajo.

Criterios de salida

1. Los auditores han recopilado información suficiente sobre los productos/procesos auditados.
2. Todas las observaciones han sido debidamente registradas.

Reunión de cierre

Objetivo

El propósito de esta reunión es crear una instancia en que los auditores presenten los resultados (al nivel de observaciones) de la evaluación a la institución auditada para permitir a esta última manifestar su opinión frente a ellas, clarificar cualquier mal interpretación en la que los auditores hayan incurrido e indicar posibles omisiones importantes dentro de la etapa previa.

Participantes

Iniciador, moderador, auditores, institución auditada, secretario.

Criterios de entrada

1. Los auditores han recopilado información suficiente sobre los productos/procesos auditados.
2. Todas las observaciones han sido debidamente registradas.

Actividades

1. Los auditores informan sobre el estado del proceso de auditoría.
2. Los auditores exponen las observaciones preliminares.
3. La institución auditada se manifiesta ante las observaciones.

Criterios de salida

1. Los auditores han expuesto las conclusiones y recomendaciones preliminares.
2. Se han resuelto todas las observaciones hechas por la institución auditada.
3. Se ha llegado a acuerdo en relación con los resultados de la auditoría.

Informe de Resultados

Objetivo

El objetivo de la presente etapa es desarrollar y entregar un informe sobre los resultados de la auditoría.

Participantes

Moderador, auditores.

Criterios de entrada

La reunión de término ha finalizado con éxito.

Actividades

1. El moderador prepara un informe de auditoría.
2. El moderador entrega el informe de auditoría al iniciador.
3. El iniciador recibe y distribuye el informe de auditoría.

Criterios de salida
El informe de auditoría fue entregado al iniciador.

5.4.2.4. Seguimiento

Objetivo
El propósito del seguimiento es que el iniciador junto a la institución auditada identifique las acciones correctivas necesarias para eliminar o prevenir las disconformidades para su posterior implantación.

Participantes
Iniciador, institución auditada.

Criterios de entrada
El informe de auditoría fue entregado al iniciador.

Actividades

1. El iniciador y la institución auditada identifican acciones correctivas para eliminar o prevenir las disconformidades.
2. Implementación de las acciones correctivas definidas.
3. Verificar la implantación de las acciones correctivas.

Criterios de salida
La institución auditada está comprometida con el seguimiento de las acciones correctivas iniciadas en pro de resolver las disconformidades detectadas durante la auditoría.

5.4.3. Informe de Auditoria

Se adjunta en anexos una pauta sobre el informe de auditoria.

5.5. Checklist

Se adjuntan las plantillas de los documentos de checklist en anexos.

EX UMBRA
IN
SOLEM

6 | Pruebas

Según IEEE (2004), "las pruebas de software çonsisten en verificar el comportamiento de un programa dinámico a través de un grupo finito de casos de prueba, debidamente seleccionados del, típicamente, ámbito de ejecuciones infinito, en relación al comportamiento esperado".

Es importante mencionar que bajo una lógica de TDD, las actividades de pruebas son realizadas a lo largo de todo el proyecto e intercaladas con el resto de las actividades de desarrollo, tomando en cuenta la calidad del software a lo largo de toda la vida del proyecto.

6.1. Estructura de las pruebas

Para la implementación y creación de las pruebas se deben seguir una serie de etapas que garantizan que estas tengan éxito al encontrar los posibles errores en el sistema. A continuación, se detallan las etapas del proceso de prueba del sistema.

6.1.1. Planificación

Esta etapa debe comenzar junto con la planificación del proyecto. Los detalles del plan de pruebas deben ser desarrollados después de la aprobación de la especificación de requerimientos.

En esta etapa se deben definir todos los elementos necesarios para posteriormente llevar acabo las pruebas, esto implica definir los niveles, tipos, metodologías, requerimientos, valores de aprobación y rechazo, recursos materiales y humano, además de responsables del proceso de pruebas.

Las pruebas que serán realizadas deben encontrarse trazadas con los requerimientos previamente aprobados, esto con el fin de asegurar el cumplimiento de todo lo establecido en la toma de requerimientos.

Una vez el documento con el plan de pruebas se encuentre terminado debe ser sometido a revisión y corregido para resolver discrepancias con el plan de proyecto, considerando los plazos y recomendaciones.

6.1.2. Especificación

Basado en el plan de pruebas desarrollado en la etapa anterior se debe describir con mayor rigor cada uno de los test que se realizaran identificando el propósito, los métodos y contenidos de las entradas y salidas.

Es importante detallar con exactitud los métodos y contenido de *input* y *output* del sistema, describiendo los tipos y volúmenes de datos; rangos de capacidad y tiempos de respuesta; métodos que serán utilizados para ingresar y recibir la información; y la traza con los requerimientos para aprobar o rechazar una funcionalidad específica del sistema.

Junto a lo anterior, es necesario definir los métodos y herramientas que deben utilizarse y describir el proceso de análisis de los resultados. Considerando que en este punto el sistema se encuentra en una etapa temprana de desarrollo, los niveles de prueba deben ser descritos en el siguiente orden: sistema, aceptación, integración y unitarias.

Al igual que en la etapa anterior, el documento debe ser revisado, corregido de ser necesario y aprobado. Las pruebas de sistema realizan una comparación entre los objetivos originales del sistema, aquellos planteados en la toma de requisitos, y los procesos, actividades y rutinas del sistema desarrollado.

6.1.3. Ejecución

Basado en la especificación de las pruebas descritas en la etapa anterior, se debe realizar la ejecución de cada uno de los test, registrando los resultados y estableciendo, para cada prueba, el éxito o fracaso en base a los criterios ya definidos.

En caso de que el sistema no apruebe alguna de las pruebas, el equipo de desarrollo debe encargarse de realizar las correcciones necesarias y posteriormente se debe comprobar la correctitud del sistema completo. Una vez el sistema apruebe todas las pruebas, se debe seguir con la siguiente actividad definida en el plan de pruebas.

Las pruebas se deben realizar en el siguiente orden.

- **Pruebas unitarias:** En paralelo a la etapa de codificación.
- **Pruebas de integración:** En paralelo a la etapa de integración.
- **Pruebas del sistema:** Previo a la entrega del sistema.
- **Pruebas de aceptación:** Antes de la etapa de aceptación y entrega.

Únicamente las pruebas unitarias son responsabilidad del equipo de desarrollo, el resto debe ser realizadas por un equipo distinto, responsable de estas.

6.1.4. Análisis de resultados

Basado en los resultados obtenidos, se debe realizar un análisis que permita identificar los defectos y sus posibles causas, lo anterior con el objetivo de establecer acciones correctivas y evitar propagación y repetición de errores.

Es importante recalcar que no se deben buscar responsabilidades individuales, únicamente asociar el fallo con un proceso en el desarrollo.

6.1.5. Completación

Para dar conclusión al proceso de prueba, la unidad responsable debe preparar los elementos necesarios para su posterior uso y realizar la documentación del proceso realizado.

6.2. Pruebas sobre el sistema *Gerprin*

A continuación, se describen los tipos de pruebas que deben ser realizadas sobre el sistema *Gerprin*.

6.2.1. Pruebas unitarias

Las pruebas unitarias tienen como objetivo asegurar que cada una de las funciones codificadas para el sistema realicen de forma correcta el proceso para el cual fueron creadas. Este tipo de pruebas se realiza bajo la lógica de "caja blanca", debido a la necesidad de probar las excepciones, casos de error, validaciones de los datos de entrada y mensajes posibles de respuesta.

Si bien las pruebas unitarias son responsabilidad del equipo de desarrollo, la instancia final de estas debe ser realizadas por personas distintas a quienes se encargaron de codificar la función que debe ser probada.

Basado en la metodología TDD, descrita con anterioridad, el tipo de pruebas descritas en esta sección deben ser realizadas en paralelo al proceso de codificación y marcan el fin de cada una de las iteraciones de esta etapa al ser aprobadas.

Se debe particionar cada uno de los procesos en unidades funcionales y definir las pruebas para los casos de éxito y posibles caminos alternativos. Las pruebas unitarias concluyen una vez se probaron las funciones codificadas hasta el momento, es decir, las completadas en la misma iteración e iteraciones anteriores del desarrollo.

6.2.1.1. Herramientas

La gran mayoría de *framwork* permiten la automatización de pruebas unitarias para agilizar el proceso y reducir errores humanos. Adicionalmente, software como *Postman* o *SoapUI* permiten hacer consultas mediante *SOAP* o *REST* de forma automática, convirtiéndose en herramientas muy útiles en esta etapa de pruebas.

6.2.1.2. Documentación

Las pruebas unitarias, al igual que el resto, deben ser correctamente documentadas, tanto su planificación como los resultados obtenidos. Adicionalmente, para este tipo de pruebas se debe documentar la traza con los requisitos funcionales, especificando explícitamente que a cuál requisito corresponde la función que se encuentra probando.

Las plantillas para la documentación de las pruebas unitarias se encuentran en anexos.

6.2.1.3. Ejemplo de prueba

A continuación, se presenta un ejemplo de prueba unitaria utilizando *Postman* para la comprobación de las respuestas de la *API* y el correcto funcionamiento de esta. Las respuestas aquí esperadas deben ser redefinidas y aceptadas por el equipo de desarrollo como las esperadas por el sistema, debido a que estas se encuentran basadas en el primer desarrollo de Gerprin y no en las nuevas funcionalidades que serán creadas.

Para agilizar el proceso se deben automatizar las pruebas unitarias que se ejecutaran, esto debido a la gran cantidad de funcionalidades que componen un sistema con las características de *Gerprin*. *Postman* permite validar la información recibida de una solicitud tipo *REST* mediante *Javascript*. Se debe registrar en la documentación pertinente el código utilizado para validar las solicitudes en la sección correspondiente.

La siguiente prueba se realizará sobre la función de *update_user* que permite modificar los datos del usuario que corresponden a nombre, apellido y rut. Para realizar la prueba se deben encadenar dos solicitudes, la primera de tipo *POST* a *update_user*, que permite modificar los datos del usuario y la segunda de tipo *GET* a *get_user*, que retorna la información de un usuario entregando el *ID* de este, posteriormente se realizara nuevamente ambas solicitudes con nuevos datos para comprobar que efectivamente se realizó la modificación

de la información y no fue una coincidencia que la información inicial del usuario coincidiera con la primera prueba.

El proceso de creación del tests automático debe ser realizado antes del proceso de pruebas, pero se añadirá en el siguiente cuadro con el propósito de ejemplificar de mejor forma las pruebas unitarias.

Tabla 6.1: Ejemplo pruebas unitarias

Prueba Unitaria: Actualizar Usuario Caso exitoso. **ID: 10**
Caso de prueba: Update User
Requisito Funcional asociado: Fiabilidad
Propósito: Se probará que las solicitudes de *update_user* acepten correctamente el formato de los datos de entrada y retornen el resultado esperado.
Prerrequisitos: se debe encontrar creadas las variables globales que utilizara Postman . Las variables a crear son: ▪ *url_base*: Corresponde a la ruta base donde se encuentra montado el sistema, por ejemplo, "localhost:8080/" ▪ *apikey*: clave que permite el uso de la *API*. ▪ *id_user*: Número identificatorio del usuario que será modificado. ▪ *name_user*: Nombre del usuario que será modificado. ▪ *lastname_user*: Apellido del usuario que será modificado. ▪ *rut_user*: Rut del usuario que será modificado. ▪ *name_user_2*: Nombre del usuario que será modificado, distinto a la variable *name_user*. ▪ *Lastname_user_2*: Apellido del usuario que será modificado, distinto a la variable *lastname_user*. ▪ *rut_use_2*: Rut del usuario que será modificado, distinto a la variable *rut_user*. ▪ *debe* tener instalado *Postman v7.2.2* y el sistema en un entorno local, de *playground* o *staged*
Datos de entrada: se debe encontrar creadas las variables globales que utilizara Postman . Las variables a crear son: ▪ Primera solicitud • url: {{url_base}}update_user • Método: POST • head ◦ X-API-Key: {{apikey}} ◦ Content-Type: application/json • body ◦ id: {{id}} ◦ name_user: {{name_user}} ◦ lastname_user: {{lastname_user}} ◦ rut_user: {{rut_user}} ▪ Segunda solicitud • url: {{url_base}}get_user • Método: GET • head ◦ X-API-Key: {{apikey}} ◦ Content-Type: application/json • body ◦ id: {{id}}

- Tercera solicitud
  - url: {{url_base}}update_user
  - Método: POST
  - head
    - X-API-Key: {{apikey}}
    - Content-Type: application/json
  - body
    - id: {{id}}
    - name_user: {{name_user_2}}
    - lastname_user: {{lastname_user_2}}
    - rut_user: {{rut_user_2}}
- Cuarta solicitud
  - url: {{url_base}}get_user
  - Método: GET
  - head
    - X-API-Key: {{apikey}}
    - Content-Type: application/json
  - body
    - id: {{id}}

Pasos:

- Crear o importar las variables globales a utilizar
- En caso de no encontrarse creada la colección de solicitudes debe ser creada.
- Se crea la colección en la pestaña *"Collections"* con nombre *"Actualizar Usuario Caso exitoso"*
- Se crean las request de la colección
  - Creamos la solicitud en el menu desplegable de la colección (click con boton derecho sobre la colección, luego en *"add request"*). Se añade nombre y descripción.
  - Abrir la solicitud e ingresar los parámetros de url, método, *head* y *body*
  - En la pestaña test se añade el código que comprueba que los resultados son correctos.
- Correr colección de solicitudes

Resultados esperados: Se espera que la totalidad de los resultados sean *"OK"*
Solicitud 1:

```
tests["Status code is 200"] = responseCode.code === 200;
var data = JSON.parse(responseBody);
tests["Message: " + data.message] = data.message === "User successfully update";
tests["Id user update"] = responseBody.has("id") === pm.environment.get("id_user");
```

Solicitud 2:

```
tests["Status code is 200"] = responseCode.code === 200;
var data = JSON.parse(responseBody);
tests["Id User"] = responseBody.has("id") === pm.environment.get("id_user");
tests["Name User"] = responseBody.has("name") === pm.environment.get("name_user");
tests["Lastname User"] = responseBody.has("lastname") ===
pm.environment.get("lastname_user");
tests["Rut User"] = responseBody.has("rut") === pm.environment.get("rut_user");
```

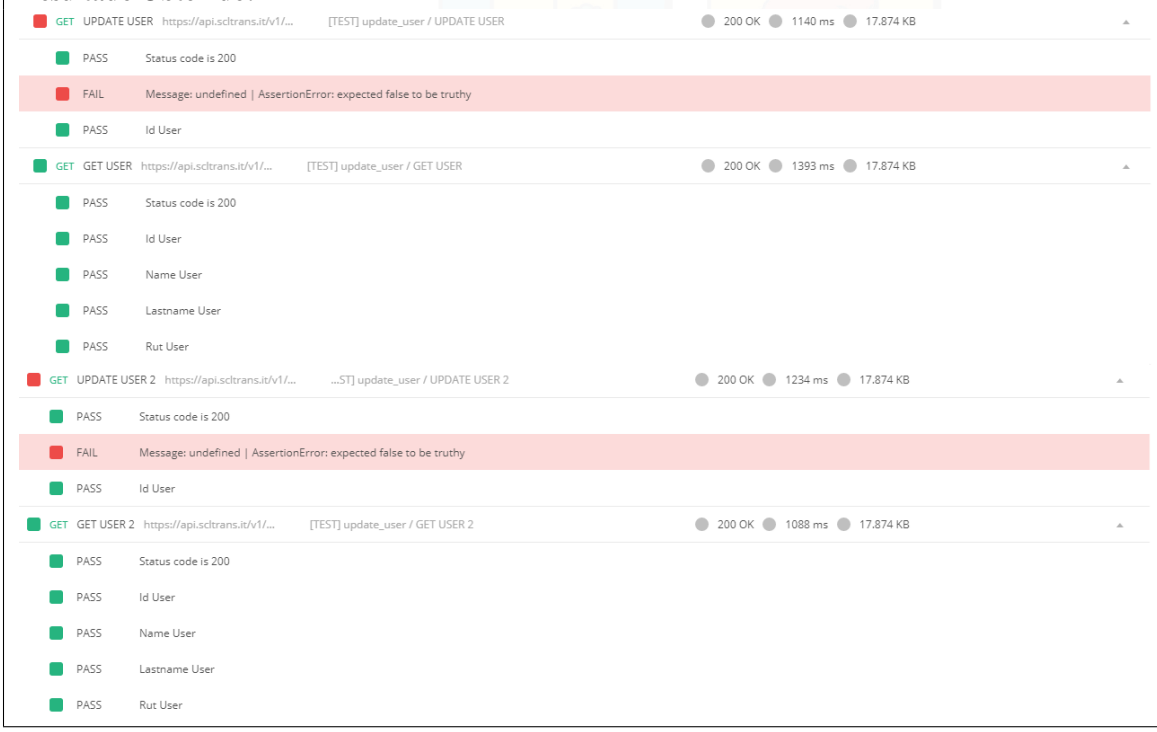
Solicitud 3:

```
tests["Status code is 200"] = responseCode.code === 200;
var data = JSON.parse(responseBody);
tests["Message: " + data.message] = data.message === "User successfully update";
tests["Id user update"] = responseBody.has("id") === pm.environment.get("id_user");
```

Solicitud 4:

```
tests["Status code is 200"] = responseCode.code === 200;
var data = JSON.parse(responseBody);
tests["Id User"] = responseBody.has("id") === pm.environment.get("id_user);
tests["Name User"] = responseBody.has("name") === pm.environment.get("name_user_2”);
tests["Lastname User"] = responseBody.has("lastname") ===
pm.environment.get("lastname_user_2”);
tests["Rut User"] = responseBody.has("rut") === pm.environment.get("rut_user”);
```

Resultado Obtenido:



6.2.2. Pruebas de integración

Las pruebas de integración aseguran el correcto funcionamiento entre dos o más unidades del sistema, para *Gerprin* es clave el acoplamiento entre los distintos módulos que lo componen para asegurar estándares de calidad mínimos asociados a mantenibilidad y fiabilidad de datos.

La integración entre las unidades es dependiente del ambiente en el cual se encuentre el sistema, por lo que se debe montar un entorno de preproducción, también conocido como ambiente de *staging* o de *QA*. Este ambiente debe tener las mismas características que producción para evitar futuros problemas de compatibilidad.

Este tipo de pruebas son realizadas por el área de *QA* y se encarga tanto de realizar las pruebas como de documentarlas. Por su parte, el equipo de desarrollo se debe encargar de entregar las instrucciones necesarias para levantar el servicio en *staging* y dar el soporte de ser necesario.

Las pruebas de integración seguirán la lógica *down-top*, es decir, se llamarán los módulos superiores (*API*) desde los módulos inferiores (aplicación móvil y modulo abordo). Se debe cuidar que la base de datos de prueba sea consistente y considerar que las soluciones deben ser de carácter global, debido a que afectan a la comunicación entre módulos, por ejemplo, si se decide cambiar la comunicación de *REST* a *SOAP*, se debe considerar hacerlo en todas las llamadas de dicho modulo para mantener la coherencia del sistema.

6.2.2.1. Herramientas

Para la creación de un entorno de pruebas es altamente recomendable emplear *Vagrant* o *Docker* para emular un servidor de producción, esto debido a que permiten crear maquinas virtuales o contenedores que de forma rápida pueden copiar las características de los servidores de producción.

6.2.2.2. Documentación

Respecto a la documentación de las pruebas de integración es necesario identificar las unidades que se pondrán a prueba.

Las plantillas para la documentación de las pruebas de integración se encuentran en anexos.

6.2.2.3. Ejemplo de prueba

El siguiente ejemplo prueba la misma funcionalidad antes propuesta, sin embargo, la prueba debe ser realizada desde la aplicación con interfaz gráfica. La importancia de la prueba de integración radica en la comunicación entre los distintos módulos.

Tabla 6.2: Ejemplo pruebas de integración.

Prueba de Integracion: Actualizar Usuario caso exitoso. **ID: 10**
Caso de prueba: Update User
Unidades relacionadas: Modulo API, Aplicación móvil y lector biométrico
Propósito: Se probará que los usuarios puedan modificar su información.
Prerrequisitos: El sistema debe encontrarse en ambiente de *staged*. La base de datos de prueba debe tener un usuario cargado con nombre, apellido y Rut distintos a los indicados en los datos de entrada
Datos de entrada: Se deben ingresar de forma manual los siguientes datos: ▪ Nombre “Elizabeth” ▪ Apellido “Sánchez” ▪ Rut “123456780”
Pasos: ▪ Realizar login en la aplicación ▪ Ir a mi perfil (esquina superior izquierda) ▪ Ir a editar datos ▪ Ingresar datos ▪ Click en guardar ▪ Comprobar que los datos fueron modificados en base de datos manejada por la API ▪ Comprobar que los datos se muestran modificados en la aplicación (perfil de usuario, datos que se muestran por defecto en la pantalla de editar usuario y detalle de usuario en pantalla de administrador) ▪ Comprobar que los datos se muestran modificados en la pantalla led del lector biométrico
Resultados esperados: El sistema debe redirigir a la página del perfil de usuario mostrando los nuevos datos e indicar que se guardaron exitosamente los datos.
Resultado obtenido: El sistema redirige a la página con los nuevos datos, pero no muestra mensaje indicando que el cambio fue exitoso.

La prueba anterior comprueba la correcta comunicación entre la aplicación móvil y la *API*, La *API* y base de datos, *API* y modulo a bordo.

6.2.3. Pruebas de sistema

Las pruebas de sistema son la última instancia para identificar errores antes de entregar el software al cliente, este proceso se realiza sobre el producto completo. Estas pruebas engloban un conjunto de tests que aseguran la correcta implementación de las reglas de negocio, para esto comprueba la navegación del sistema, el ingreso de datos, procesamiento y recuperación. Para el caso de *Gerprint*, se implementarán pruebas performance y de humo.

Las pruebas de performance intentan emular el tráfico normal de uso que se encontrara la aplicación, se debe realizar para cada una de las funciones del sistema. Mientras que las pruebas de humo corresponden a pruebas no exhaustivas del sistema completo, además, se debe considerar dentro de las pruebas de humo, que la aplicación cumpla con las condiciones de usabilidad para los tamaños de los distintos dispositivos especificados

6.2.3.1. Documentación

Las pruebas de performance deben documentar la cantidad de solicitudes por minuto realizadas a la aplicación, tiempo de respuesta y uso de ram y procesador del servidor.

Las plantillas para la documentación de las pruebas de humo se encuentran en anexos

6.2.3.2. Ejemplo de prueba

Las pruebas de sistema se hacen contra el documento de requisitos. Cada una de las pruebas realizadas debe hacer referencia a uno de los requerimientos identificados en la etapa de análisis.

Tabla 6.3: Ejemplo pruebas de hummo

Prueba de Humo: Actualizar Usuario caso exitoso. **ID: 10**
Caso de prueba: Update User
Requisito: Update User (ID: 10)
Prerrequisitos: El sistema debe encontrarse en ambiente de stage.
Datos de entrada: Se deben ingresar de forma manual los siguientes datos: ▪ Nombre “Elizabeth” ▪ Apellido “Sánchez” ▪ Rut “123456780”
Pasos: ▪ Realizar login en la aplicación ▪ Ir a mi perfil (esquina superior izquierda) ▪ Ir a editar datos ▪ Ingresar datos ▪ Click en guardar ▪ Comprobar que los datos fueron modificados en base de datos manejada por la API ▪ Comprobar que se ajuste a pantallas de 3,5 pulgadas con resolución de 360 x 640 o superior. ▪ Comprobar que los textos tengan un contraste adecuado respecto al fonda para su fácil lectura. ▪ Comprobar que los iconos tengan relación con la acción a realizar
Resultados esperados: El sistema debe redirigir a la página del perfil de usuario mostrando los nuevos datos e indicar que se guardaron exitosamente los datos.
Resultado obtenido: El cliente reporta que el sistema redirige a la página con los nuevos datos, pero no muestra mensaje indicando que el cambie fue exitoso.

Tabla 6.4: Ejemplo pruebas de performance

Prueba performance: Actualizar Usuario caso exitoso. **ID:**
Caso de prueba: Update User
Cantidad de solicitudes por minuto: 100 solicitudes por segundo
Propósito:
Prerrequisitos: se debe encontrar creadas las variables globales que utilizara Postman . Las variables a crear son: ▪ *url_base*: Corresponde a la ruta base donde se encuentra montado el sistema, por ejemplo, "localhost:8080/" ▪ *apikey*: clave que permite el uso de la *API*. ▪ *id_user*: Número identificatorio del usuario que será modificado. ▪ *name_user*: Nombre del usuario que será modificado. ▪ *lastname_user*: Apellido del usuario que será modificado. ▪ *rut_user*: Rut del usuario que será modificado. ▪ *name_user_2*: Nombre del usuario que será modificado, distinto a la variable *name_user*. ▪ *Lastname_user_2*: Apellido del usuario que será modificado, distinto a la variable *lastname_user*. ▪ *rut_use_2*: Rut del usuario que será modificado, distinto a la variable *rut_user*. ▪ *debe* tener instalado *Postman v7.2.2* y el sistema en un entorno local, de *playground* o *staged*
Datos de entrada: se debe encontrar creadas las variables globales que utilizara Postman . Las variables a crear son: ▪ Primera solicitud • url: {{url_base}}update_user • Método: POST • head ◦ X-API-Key: {{apikey}} ◦ Content-Type: application/json • body ◦ id: {{id}} ◦ name_user: {{name_user}} ◦ lastname_user: {{lastname_user}} ◦ rut_user: {{rut_user}}
Tiempo Promedio: 3,2 segundos

6.2.4. Pruebas de aceptación

Las pruebas de aceptación son realizadas por el cliente y tiene como objetivo validar que se cumplan los requisitos levantados. Estas pruebas se realizan bajo la lógica de "caja negra", debido a que no es relevante el funcionamiento interno de la aplicación para los usuarios finales. El cliente debe poseer el manual de usuario y documento de requisitos para completar las pruebas de aceptación.

6.2.4.1. Documentación

A diferencia de las pruebas anteriores para el presente tipo de pruebas, se debe llenar los cuadros de información únicamente si el cliente reporta un error en el funcionamiento del sistema.

6.2.4.2. Ejemplo de prueba

Tabla 6.5: Ejemplo pruebas de aceptación

Prueba de Aceptación: Actualizar Usuario caso exitoso. **ID: 10**
Caso de prueba: Update User
Reporte del cliente: Se reporto el error mediante plataforma de soporte ticket asociado: 0032 Al editar el usuario, el sistema no me muestra mensaje indicando que fue modificado exitosamente o que ocurrió un error, según se indica en el documento de requerimientos.
Prerrequisitos: El sistema debe encontrarse en ambiente de staged. La base de datos de prueba debe tener un usuario cargado con nombre, apellido y Rut distintos a los indicados en los datos de entrada.
Datos de entrada: Se deben ingresar de forma manual los siguientes datos: ▪ Nombre “Elizabeth” ▪ Apellido “Sánchez” ▪ Rut “123456780”
Pasos: ▪ Realizar login en la aplicación ▪ Ir a mi perfil (esquina superior izquierda) ▪ Ir a editar datos ▪ Ingresar datos ▪ Click en guardar ▪ Comprobar que se muestre mensaje indicando estado del cambio realizado, con uno de los siguientes dos mensajes: “Usuario modificado con éxito” o “Error al modificar usuario”
Resultados esperados: El sistema debe redirigir a la página del perfil de usuario mostrando los nuevos datos e indicar que se guardaron exitosamente los datos.
Resultado obtenido: El cliente reporta que el sistema redirige a la página con los nuevos datos, pero no muestra mensaje indicando que el cambie fue exitoso.

7 | Conclusiones

En la presente documento, se aborda el aseguramiento de la calidad de *Gerprin*, sin embargo, bajo un primer análisis se puede apreciar que es imposible asegurar objetivamente que un sistema posee una característica tan abstracta como el concepto de "calidad". Para poder asegurar la calidad de un sistema es fundamentar definir los indicadores que al ser cumplidos permiten definir *Gerprin* como un software de calidad.

La elección de los requisitos no funcionales y de los atributos de calidad es fundamental para el aseguramiento de esta, pues son estos, que, mediante su cumplimiento, definirán la calidad. En la selección de los indicadores que marcan la calidad de software se destaca la clasificación del modelo ISO 9126 que clasifica en 6 grandes grupos las características que debe tener un software para considerarse que es de "calidad". En el presente documento se consideraron dichos elementos, sin embargo, en la sección de requisitos no funcionales, se destacó un grupo reducido de indicadores sobre el resto debido a la experiencia en el desarrollo de software similares, donde primaban los atributos de calidad seleccionados por sobre otros.

Más allá de definir el concepto de calidad, al cual se busca llegar, es necesario establecer y definir un proceso por el cual se logre alcanzar los estándares impuestos. Por este motivo la segunda etapa de un plan de aseguramiento de calidad es definir las metodologías que serán empleadas para el desarrollo del software.

Como se hizo énfasis, *Gerprin* es un sistema que ya se encuentra desarrollado. En esta ocasión se apunta a una nueva versión, sin desconocer la actual. Por este motivo se pueden considerar metodologías menos ortodoxas, como el concepto de diseño emergente, que fue descrito en la sección correspondiente. Esta técnica corresponde a una característica de la metodología de *XP* que resalta por su gran flexibilidad en el desarrollo, sobre esto se aplica *TDD* que permite la temprana detección de errores, que sumado a la flexibilidad de *XP* permiten una pronta solución y evolución del software.

Asimismo, se implementó un proceso de *SQA* basado en lo descrito en Herramientas específicas para SQA y SCM (2000) que clasifica a las organizaciones que utilizan *SQA* de alguna u otra forma en el desarrollo de software según cuatro niveles. Los niveles de madurez se detallan en base a los procesos y subprocesos que se implementan. Para el sistema aquí presentado se internalizaron los estándares, procesos y actividades recomendadas hasta el segundo nivel, que representa el núcleo del aseguramiento de calidad. En el desarrollo de *Gerprin* se busca simplificar los procesos dejando de lado largas documentaciones o procesos que aumentan los tiempos de desarrollo.

El primer nivel de madurez se integró reconociendo la importancia del proceso de *SQA* y su integración al desarrollo del software. Mediante el presente documento se definen las acciones que procederán a integrar el proceso, es decir, de adoptan las revisiones y auditorias, que mediante checklist mantienen un control sobre la adherencia de los productos y procesos a los estándares y procedimientos establecidos.

El segundo nivel es lograr el compromiso de la organización, mediante políticas organizacionales, con el área de *SQA* para incorporar un plan de *SQA* en cada proyecto y establecer los responsables de las actividades de aseguramiento de calidad. Para *Gerprin*, se establece un área de *SQA* y los responsables de las actividades relacionadas al aseguramiento de la calidad.

Una vez realizado el proceso de desarrollo empleando las metodología, herramientas y esquema

organizacional definidas en el plan de calidad, es necesario comprobar que los indicadores definidos en las etapas iniciales del proyecto se cumplen. Mediante la etapa de pruebas se asegura que el sistema cumpla con los estándares impuestos que aseguran la calidad del producto final, En la sección correspondiente se detallan las pruebas que se realizan para asegurar el cumplimiento de todos los requerimientos.

No establecer estándares que deben ser cumplidos entrega un producto que puede o no tener la calidad necesaria para un uso productivo; lo mismo ocurre al no establecer las metodologías y herramientas, creando un camino que es incapaz de asegurar el cumplimiento de los estándares definidos inicialmente. Asegurar la calidad de un producto es una labor integral que debe considerar el proceso interno y no solo el resultado final, comprender que es lo necesario para obtener un software de calidad y los pasos para conseguirlo son las etapas tempranas en un desarrollo exitoso.

La aparición de nuevas metodologías que se adaptan a los procesos de *SQA* son esenciales en el desarrollo de cualquier producto. En el caso puntual del desarrollo de *Gerprin* la implementación de *TDD* sobre *XP* permite una detección temprana de los errores de codificación y tener espacios para gestionar una solución.

Lejos estamos de la época conocida como “crisis del software”, donde el desarrollo de sistemas era un proceso relativamente nuevo, sin metodologías y estándares que permitan asegurar la calidad del producto final. Herramientas como checklist, automatización de pruebas, máquinas virtuales, entre otras, no solo permiten asegurar lo que años antes no se podía, además lo logran con un menor esfuerzo humano, sin embargo, para lograr esto es necesario la planificación e integración de dichas técnicas y herramientas en el proceso de ingeniería de software.

Bibliografía

Balaguera, Yohn Daniel Amaya (2013). *Metodologías ágiles en el desarrollo de aplicaciones para dispositivos móviles. Estado actual*. Revista de Tecnología, Journal Technology, Volumen 12.

Blé Jurado, Carlos y colaboradores (2010). *Diseño Ágil con TDD*. lulu.com, 1° edición.

Espinoza, Carlos Ramírez (2017). *Plan de Calidad Para "Post Emergency Support System"*.

Garcia, Joaquin Alfonso Nuñez (2015). *Plan de Calidad Para "Huellas Uuniversitarias"*. 3.3.1, 3.3.2, 3.3.3, 3.3.4, 3.3.5, 3.3.6

Herramientas específicas para SQA y SCM (2000). https://www.inf.utfsm.cl/~visconti/herramientas/. [Acceso abril de 2019]. 4.1, 4.2.3, 4.2.3.14, 5.3, 5.4, 7

Joskowicz, José (2008). *Reglas y Prácticas en eXtreme Programming*. [Universidad de Vigo, España].

Marecos Brizuela, Pedro Celestino (2017). *Revisión sistemática sobre metodologías agiles en empresas de software*. Revista de la Facultad de Ciencias Aplicadas, Universidad Nacional de Pilar.

Pressman, Roger S (2010). *Ingeniería de Software, un enfoque práctico*. Editorial McGraw-Hill, 7° edición.

EX UMBRA
IN
SOLEM

A | Anexos

A.1. Informe de revisión

1 Información del Proyecto y del Producto	
Proyecto:	
Producto	**Tipo** ❒ Planes ❒ Código ❒ Esp. de Requerimientos ❒ Esp. de Pruebas ❒ Diseño ❒ Documentación Usuario

2 Inspector		
Nombre		**Rol** ❒ Moderador ❒ Secretario ❒ Presentador ❒ Observador ❒ Inspector
e-mail	**Fono**	

3 Resumen				
Recibido: __/__/____	**Entregado:** __/__/____	**Resumen de Defectos**		
Preparación			**Preparación**	**Inspección**
Fecha	**Horas**	**Mayores**		
		Menores		
		No resueltos		
Total Horas: ____		**Total**		

4 Observaciones
Tiempo de rework estimado: ____
Comentarios:

5 Defectos							
Nro.	Inicial	Ubicación	Descripción	Mayor	Menor	No Resuelto	Excluido

A.2. Informe de Auditoria

1 Identificación de la auditoría			
Proyecto			
Proyecto:			
Jefe de Proyecto:			
Fase del ciclo de vida	❒ Planificación	❒ Diseño	❒ Integración y pruebas
	❒ Esp. de Requerimientos	❒ Implementación	❒ Aceptación y entrega
			❒ Mantención
Auditoría			
Propósito y objetivos:			
Tipo de auditoría: ❒ Interna ❒ Externa		**Fecha inicio:** __/__/____	**Fecha término:** __/__/____
Participantes			
Institución auditada: _____		**Institución auditora:** _____	
Representantes		**Representantes**	
Iniciador		**Moderador**	
Otros representantes		**Auditor**	
		Auditor	
		Auditor	
		Auditor	

2 Objetos de auditoría

Procesos:

1.- _____ 6.- _____

2.- _____ 7.- _____

3.- _____ 8.- _____

4.- _____ 9.- _____

5.- _____ 10.- _____

Productos:

1.- _____ 6.- _____

2.- _____ 7.- _____

3.- _____ 8.- _____

4.- _____ 9.- _____

5.- _____ 10.- _____

3 Bases para la evaluación

Normas, estándares, guías, planes y procedimientos

1.- _____ 6.- _____

2.- _____ 7.- _____

3.- _____ 8.- _____

4.- _____ 9.- _____

5.- _____ 10.- _____

Criterios de evaluación

Criterio	Descripción	Nivel de satisfacción

Hoja __ de ___

Informe de Auditoría

4 Actividades de auditoría				
Actividad	**Fecha (Desde- Hasta)**		**Estado[1]**	**Observación**

[1]R: Realizada, I: Incompleta, NR: No realizada

5 Resumen de resultados				
Anomalías más relevantes				
Nro.	**Descripción**	**Ubicación**	**Prioridad[1]**	**Recomendación**
[1]A: Alta, M: Media; B: Baja				
Conclusiones				
Conclusiones:				

Estado de la auditoría

❒ Aprobada

❒ Condicionada _____

❒ Reprobada _____

Fecha auditoría de verificación: __/__/____

Tipo de auditoría: ❒ Interna ❒ Externa

Documentos anexados	
•	
•	
•	
•	
•	
•	
•	

_____

Firma del Moderador

_____

Firma del Inciador

Hoja __ de ___

Informe de Auditoría

6 Anomalías				
Nro.	**Descripción**	**Ubicación**	**Prioridad** [1]	**Recomendación**

[1] **A: Alta, M: Media; B: Baja**

A.3. Checklist

A.3.1. Checklist por actividades del proceso de desarrollo evaluados por QA

Exploración		
1. Identificación		
Proyecto:	**Fecha de inicio:** __/__/____ **Fecha de término:** __/__/____	
Nombre:		
E-mail:	**Fono:**	
2. Actividades		
	Sí	**No**
• ¿Se elaboró el plan de SQA?		
• ¿Se apoyó la elaboración del plan de proyecto?		
• ¿Se entregó el soporte solicitado por el personal de desarrollo?		
Auditoría (opcional)		
• ¿Se auditó el proceso de planificación?		
Reporte de problemas y acciones correctivas		
• ¿Se informó sobre el estado de avance de las actividades de SQA?		
• ¿Se informaron las desviaciones detectadas?		
Revisión		
• ¿Se evaluó la selección de los productos de trabajo??		
• ¿Se evaluó la selección de las herramientas?		
• ¿Se evaluó el plan de proyecto?		
• ¿Se evaluó el plan de SQA??		
• ¿Se evaluó el plan de SCM?		

Planificación de las entregas		
1. Identificación		
Proyecto:	**Fecha de inicio:** __/__/____ **Fecha de término:** __/__/____	
Nombre:		
E-mail:	**Fono:**	
2. Actividades		
	Sí	**No**
• ¿Se entregó el soporte solicitado por el personal de desarrollo?		
Auditoría (opcional)		
• ¿Se auditó el proceso de especificación de requerimientos?		
• ¿Se auditó el proceso de diseño?		
Reporte de problemas y acciones correctivas		
• ¿Se informó sobre el estado de avance de las actividades de SQA?		
• ¿Se informaron las desviaciones detectadas?		
Revisión		
• ¿Se evaluó la especificación de requerimientos?		
• ¿Se evaluó el diseño preliminar?		
• ¿Se evaluaron los planes de prueba (sistema, aceptación, integridad y unidad)?		
Prueba		
• Planificación de la prueba del sistema.		
• Planificación de la prueba de aceptación.		
• Planificación de la prueba de integración.		
• Planificación de la prueba de unidad.		

Iteraciones		
1. Identificación		
Proyecto:	**Fecha de inicio:** __/__/____ **Fecha de término:** __/__/____	
Nombre:		
E-mail:	**Fono:**	
2. Actividades		
	Sí	**No**
• ¿Se entregó el soporte solicitado por el personal de desarrollo?		
Auditoría (opcional)		
• ¿Se auditó el proceso de codificación?		
• ¿Se auditó la prueba de unidad?		
Reporte de problemas y acciones correctivas		
• ¿Se informó sobre el estado de avance de las actividades de SQA?		
• ¿Se informaron las desviaciones detectadas?		
Revisión		
• ¿Se evaluó la especificación de los casos de prueba?		
• ¿Se evaluó la especificación de los procedimientos de prueba?		
• ¿Se evaluó el código y su documentación?		
• ¿Se evaluaron los resultados de la prueba de unidad?		
Prueba		
• Especificación de casos de prueba.		
• Especificación de procedimientos de prueba.		
• Prueba de unidad		

Integración y pruebas		
1. Identificación		
Proyecto:	**Fecha de inicio:** __/__/____ **Fecha de término:** __/__/____	
Nombre:		
E-mail:	**Fono:**	
2. Actividades		
	Sí	**No**
• ¿Se entregó el soporte solicitado por el personal de desarrollo?		
Auditoría (opcional)		
• ¿Se auditó el proceso de integración?		
• ¿Se auditó el proceso de prueba o parte de él?		
Reporte de problemas y acciones correctivas		
• ¿Se informó sobre el estado de avance de las actividades de SQA?		
• ¿Se informaron las desviaciones detectadas?		
Revisión		
• ¿Se evaluó el proceso de integración?		
• ¿Se evaluó la prueba y sus resultados?		
Prueba		
• Prueba de integridad.		
• Prueba de aceptación.		
• Prueba del sistema.		
• Análisis y reporte de resultados		

Aceptación y entrega		
1. Identificación		
Proyecto:	**Fecha de inicio:** __/__/____ **Fecha de término:** __/__/____	
Nombre:		
E-mail:	**Fono:**	
2. Actividades		
	Sí	**No**
• ¿Se entregó el soporte solicitado por el personal de desarrollo?		
Auditoría		
• ¿Se realizaron las auditorías de la configuración funcional y física?		
Reporte de problemas y acciones correctivas		
• ¿Se informó sobre el estado de avance de las actividades de SQA?		
• ¿Se informaron las desviaciones detectadas?		
Revisión		
• ¿Se evaluó el software y su documentación?		

Muerte del proyecto			
1. Identificación			
Proyecto:		**Fecha de inicio:** __/__/____ **Fecha de término:** __/__/____	
Nombre:			
E-mail:		**Fono:**	
2. Actividades			
		Sí	**No**
• ¿Se planificaron las actividades de SQA asociadas a la muerte del proyecto?			
• ¿Se entregó el soporte solicitado por el personal de desarrollo?			
Auditoría (opcional)			
• ¿Se auditó el proceso de muerte del proyecto?			
Reporte de problemas y acciones correctivas			
• ¿Se informó sobre el estado de avance de las actividades de SQA?			
• ¿Se informaron las desviaciones detectadas?			
Revisión			
• ¿Se evaluó cada cambio introducido durante la mantención a nivel de análisis, diseño, e implementación?			
• ¿Se evaluó la documentación asociada a los cambios?			
• ¿Se evaluó la nueva versión/edición del software? ¿Y su documentación?			
Prueba			
• Planificación, especificación, desarrollo de las pruebas, análisis y reporte de resultados para los cambios introducidos durante la mantención.			

A.3.2. Checklist por actividades del proceso de desarrollo evaluados por QA

Checklist: Especificación de Requerimientos				
1. Identificación del proyecto y del producto				
Proyecto:				
Producto:				
2. Inspector				
Nombre		**Rol**		
e-mail	**Fono**	❒ Moderador	❒ Secretario	
		❒ Presentador	❒ Observador	
		❒ Inspector		
3. Checklist				
		Sí	**No**	**N/A**
Adherencia				
• ¿El documento se adhiere a los estándares establecidos?				
Claridad				
• ¿Los requerimientos son especificados en forma clara?				
• ¿Los requerimientos se encuentran libres de ambigüedades?				
• ¿La especificación de requerimientos se lee fácilmente?				
• ¿La terminología utilizada es consistente con la empleada por el cliente/usuario?				

Completitud			
• ¿Se describen todos los requerimientos y las restricciones?			
• ¿Se asigna prioridad a los requerimientos y las restricciones?			
• ¿Se define correctamente los criterios para asignar prioridades a los requerimientos?			
• ¿Se dimensiona el impacto del sistema sobre los usuarios, otros sistemas y su entorno?			
• ¿Se especifican todas las funciones necesarias y suficientes para completar los objetivos del sistema?			
• ¿Se describen las entradas/proceso/salidas necesarias y suficientes para cada función?			
• ¿Se establecen los tiempos de respuesta esperados por el usuario?			
• ¿Se definen formalmente todas las interfaces internas/externas del sistema?			
• ¿Se incluyen los requerimientos de interfaz entre el hardware, software y el usuario?			
• ¿Se definen los niveles de seguridad requeridos?			
• ¿Se especifica la confiabilidad incluyendo las consecuencias de las fallas del software, la información que debe ser protegida de estas fallas, la detección de errores y la recuperación?			
• ¿Se definen los criterios de éxito? ¿Se definen atributos de calidad que permitan medir los requerimientos? ¿Se les asigna un valor objetivo?			
• ¿Se definen métodos de prueba para cada requerimiento de software?			
• ¿Se definen los requerimientos de mantenibilidad especificando la escalabilidad del software, interfaces con otros sistemas, precisión, rendimiento, etc.?			
• ¿Se especifica el impacto del incumplimiento de los requerimientos?			
Consistencia			
• ¿Los requerimientos son consistentes entre ellos y con requerimientos de sistemas relacionados?			
• ¿Los requerimientos son consistentes con la especificación de requerimientos preliminar de la planificación?			
Facilidad de pruebas			
• ¿Es factible probar, demostrar o analizar el cumplimiento de los requerimientos?			
• ¿Los requerimientos son lo suficientemente precisos para facilitar la especificación de las pruebas?			
Factibilidad			
• ¿Es posible implementar los requerimientos con las técnicas, herramientas, recursos y personal definidos y bajo los costos y la calendarización estipulada?			
• ¿Es posible satisfacer los atributos de calidad definidos?			
• ¿Son factibles el diseño, implementación, mantención y operación del software?			

Checklist: Planes del Proyecto

1. Identificación del proyecto y del producto

Proyecto:

Producto:

2. Inspector

Nombre		**Rol**
e-mail	**Fono**	❒ Moderador ❒ Secretario ❒ Presentador ❒ Observador ❒ Inspector

3. Checklist

	Sí	No	N/A
Adherencia			
• ¿El documento se adhiere a los estándares establecidos?			
Claridad			
• ¿Se alcanza el propósito principal del plan y de sus secciones?			
• ¿Se encuentran claros y bien precisadas las asignaciones de recursos, la calendarización y los hitos?			
• ¿El plan es de fácil lectura?			
• ¿La terminología utilizada es consistente y comprensible por el jefe de proyectos y los desarrolladores?			
Completitud			
• ¿Se cuenta con una especificación de requerimientos preliminar adecuada para el estudio de soluciones factibles?			
• ¿Se informan los resultados del análisis de alternativas realizados? ¿Se justifican las decisiones tomadas?			
• ¿El proceso de desarrollo ha sido apropiadamente seleccionado? ¿Es explicado por procedimientos sobre su monitoreo y aplicación?			
• ¿Se han especificado las técnicas y herramientas necesarias y suficientes para las actividades de desarrollo?			
• ¿Se hallan completos la estructura organizacional, la asignación de recursos, la calendarización y los hitos del proyecto?			
• ¿Se identifica soluciones y planes adecuados para los riesgos?			
• ¿El proceso incluye: (1) costos, tamaño y esfuerzo, (2) revisiones, (3) métricas, (4) especificación de requerimientos, (5) diseño, (6) pruebas, (7) SQA, (8) SCM, (9) verificación y validación, (10) planificación del proyecto, (11) administración de riesgos, e (12) integración			
Correctitud			
• ¿La asignación de recursos y la calendarización establecida corresponden a las estimaciones de tamaño y esfuerzo?			
• ¿Existe información que valide y justifique las asignaciones y la calendarización?			
• ¿La calendarización está ausente de conflictos y "embotellamientos"?			
Mantenibilidad			
• ¿El documento presentado es fácilmente mantenible?			

Checklist: Diseño Preliminar

1. Identificación del proyecto y del producto

Proyecto:

Producto:

2. Inspector

Nombre		**Rol**
e-mail	**Fono**	❒ Moderador ❒ Secretario ❒ Presentador ❒ Observador ❒ Inspector

3. Checklist

	Sí	No	N/A
Adherencia			
• ¿El documento se adhiere a los estándares establecidos?			
• ¿El diseño fue desarrollado de acuerdo a las metodologías y técnicas predefinidas?			
Claridad			
• ¿El diseño representa claramente la arquitectura (flujos de datos, flujos de control e interfaces)?			
• ¿Se documentan todos los objetivos, suposiciones, restricciones, decisiones y dependencias de este diseño?			
• ¿La terminología utilizada es consistente con la empleada por los desarrolladores?			
Completitud			
• ¿Se encuentran claros los objetivos del diseño preliminar?			
• ¿Se incluye una descripción del procedimiento que se utilizó para desarrollar el diseño preliminar (técnicas, representación del diseño, etc.)?			
• ¿Existe una lista de las funciones que deben ser provistas por el software?			
• ¿La especificación de módulos cubre completamente la funcionalidad de los requerimientos del software?			
• ¿La especificación de los módulos contempla su funcionalidad, entradas, salidas, los criterios de ejecución y la interfaz con otros módulos?			
• ¿Existe un modelo de la interfaz entre el sistema y el usuario final: (1) descripción de los conocimientos técnicos del usuario, (2) información sobre la flexibilidad y adaptabilidad de la interfaz usuaria, (3) información sobre tutoriales, asistencia y manuales para el usuario, (4) tareas que el usuario deberá desempeñar, (5) y la apreciación del usuario con respecto a las tecnologías de la información?			
• ¿Se modelan todas las interfaces?			
• ¿Se diseña la interfaz considerando al usuario final?			
• ¿Se describen y justifica las estructuras de datos?			
• ¿Se especifica la organización y los contenidos de la base de datos?			
• ¿Se describen y justifican los algoritmos más relevantes?			
• ¿Se han identificado y analizado las rutas de ejecución críticas?			
Confiabilidad			
• ¿El diseño prevé la detección y recuperación de errores?			
• ¿Son descritas completamente las condiciones de error?			
Consistencia			
• ¿Se utilizan consistentemente los nombres de los elementos de datos, procedimientos y funciones a lo largo de la descripción y representación del diseño?			
• ¿El diseño representa el hardware, el software y el entorno del sistema?			
• ¿El diseño es consistente con la especificación de requerimientos?			

Facilidad de pruebas			
• ¿Es factible probar, demostrar o analizar que el diseño preliminar satisface los requerimientos?			
• ¿Es posible integrar y probar el software generado a partir de este diseño?			
Factibilidad			
• ¿El diseño es factible según la calendarización, el presupuesto y la tecnología disponibles?			
Mantenibilidad			
• ¿El diseño es modular?			
• ¿Los módulos tienen alta cohesión y bajo acoplamiento?			
Trazabilidad			
• ¿Es posible trazar el diseño con los requerimientos?			

Checklist: Plan de pruebas			
1. Identificación del proyecto y del producto			
Proyecto:			
Producto:			
2. Inspector			
Nombre		**Rol**	
e-mail	**Fono**	❒ Moderador ❒ Secretario ❒ Presentador ❒ Observador ❒ Inspector	
3. Checklist			
	Sí	**No**	**N/A**
Adherencia			
• ¿El documento se adhiere a los estándares establecidos?			
Claridad			
• ¿El plan es de fácil lectura?			
• ¿La terminología utilizada es consistente con la empleada por los desarrolladores?			
Completitud			
• ¿Se describen adecuadamente las funciones que serán probadas?			
• ¿Para todos los requerimientos considerados no testeables, se entrega una explicación?			
• ¿Se definen los tipos de pruebas apropiadamente?			
• ¿Se definen los criterios de éxito para todas las pruebas?			
• ¿Se establecen las condiciones sobre las cuales las pruebas serán interrumpidas?			
• ¿Se especifica claramente el orden de los pasos a seguir para la integración de las pruebas?			
• ¿Se definen correctamente las pruebas de regresión?			
• ¿Se prevé recopilar suficientes datos para la estimación de la confiabilidad del software?			
• ¿Se ha calendarizado la obtención/utilización de los recursos, métodos y herramientas necesarias para realizar las pruebas?			
• ¿Se describe la programación de la etapa de pruebas con el suficiente nivel de detalle?			
• ¿Se han definido los roles y responsabilidades para todos los individuos involucrados en las pruebas?			
• ¿Se menciona la participación de personal de SQA para la verificación de las actividades de prueba?			

Correctitud			
• ¿Son realistas los criterios de entrada y salida de las pruebas?			
• ¿El conjunto de casos de pruebas incluye la cobertura de entradas ilegales y conflictivas?			
• ¿El conjunto de casos de pruebas contempla el uso adecuado de los valores de entrada por defecto?			
• ¿El conjunto de casos de pruebas contiene un número apropiado de rutas de error?			
• ¿Son suficientes y adecuadas las pautas para ejecutar el plan de pruebas?			
Factibilidad			
• ¿Es posible realizar las actividades descritas en el plan con la calendarización, el presupuesto y la tecnología disponibles??			
Mantenibilidad			
• ¿Se contemplan en el plan de pruebas el manejo de los cambios que podrían ocurrir en la especificación de requerimientos, diseño o código?			
Trazabilidad			
• ¿Son los criterios de aceptación de las pruebas trazables con los requerimientos?			
• ¿El conjunto de casos d prueba contempla las interfaces definidas?			

Checklist: Código

1. Identificación del proyecto y del producto

Proyecto:

Producto:

2. Inspector

Nombre		**Rol**
e-mail	**Fono**	❒ Moderador ❒ Secretario ❒ Presentador ❒ Observador ❒ Inspector

3. Checklist

	Sí	No	N/A
Completitud			
• ¿El código es completo y preciso de acuerdo con la documentación del diseño?			
• ¿El código se integra y debugged para satisfacer la especificación del diseño?			
• ¿Se generan las bases de datos necesarias, incluyendo los datos iniciales?			
• ¿No existen variables, constantes y tipos de datos superfluos? ¿Están bien definidos y son correctamente referenciados?			
Consistencia			
• ¿El código es consistente con el diseño?			
• ¿Se utilizan siempre los mismos formatos, tipos de invocaciones y estructuras?			
Correctitud			
• ¿El código se adhiere a los estándares definidos?			
• ¿Se definen y utilizan correctamente todas las variables?			
• ¿Los comentarios son precisos?			
• ¿El número de parámetros de las invocaciones es el correcto?			

Facilidad de comprensión			
• ¿Los comentarios describen cada rutina en forma clara y completa?			
• ¿Existe código ambiguo o innecesario? Si es así, ¿está adecuadamente documentado?			
• ¿Se utilizan técnicas de formato para facilitar la comprensión (¿identación, espacios en blancos, etc.?			
• ¿Se ocupan convenciones mnemotécnicas para la denominación de las variables? ¿Los nombres reflejan el tipo de variables			
• ¿Los rangos válidos para las variables están definidos?			
• ¿El código utiliza ecuaciones matemáticas concordantes con la descripción de los algoritmos contenida en el diseño?			
Facilidad de pruebas			
• ¿Se suprime el uso de técnicas y prácticas que dificulten las pruebas?			
Mantenibilidad			
• ¿El código referencia simbólicamente las constantes para facilitar los cambios?			
• ¿Se incluyen diccionarios de datos y cross-references para mostrar el acceso a las variables y constantes dentro del programa?			
• ¿Los "subprogramas" cuentan con un único punto de entrada y salida?			
Previsibilidad			
• ¿El código fue escrito en un lenguaje con sintaxis y semántica bien definidas?			
• ¿El código evita utilizar los parámetros predefinidos por el lenguaje?			
• ¿El código se encuentra libre de loops infinitos?			
• ¿Se evita la recursividad?			
Robustez			
• ¿Se prevén los errores de ejecución como divisiones por cero, rangos no aceptables para las variables, stack overflow, entradas inválidas, etc.?			
Trazabilidad			
• ¿El código identifica cada programa de una única forma?			
• ¿El código puede ser fácilmente trazado con el diseño?			
• ¿El código contiene un histórico con los cambios que se le han realizado junto con las razones asociadas?			

Checklist: Procedimientos y casos de prueba			
1. Identificación del proyecto y del producto			
Proyecto:			
Producto:			
2. Inspector			
Nombre	**Rol**		
e-mail / **Fono**	❒ Moderador ❒ Secretario ❒ Presentador ❒ Observador ❒ Inspector		
3. Checklist			
	Sí	**No**	**N/A**
Adherencia			
• ¿El documento se adhiere a los estándares y procesos definidos?			
Claridad			
• ¿Las explicaciones sobre la ejecución de los procedimientos de prueba son claras y explícitas?			
• ¿Las instrucciones son entregadas como un conjunto ordenados de pasos por seguir?			
• ¿Los pasos del inicio y de los procedimientos de las pruebas son precisos y están libres de ambigüedades?			
• ¿Los criterios de éxito y fracaso son claros y no ambiguos?			

Completitud			
• ¿La función probada se describe con exactitud?			
• ¿La función probada corresponde a la última versión de dicha función?			
• ¿Se asocia cada requerimiento y las funciones asociadas a él con un procedimiento de prueba?			
• ¿El propósito de los procedimientos y casos de prueba es claro y preciso?			
• ¿Los procedimientos de prueba enumeran la precedencia de los casos de prueba?			
• ¿Los procedimientos de prueba especifican los equipos, el software y el personal requerido para los casos de prueba?			
• ¿El procedimiento describe las respuestas esperadas para cada caso de prueba?			
• ¿Los procedimientos indican como evaluar los resultados de los casos de prueba (criterios de éxito/fracaso de la prueba)?			
• ¿El procedimiento indica si es o no posible seguir con las pruebas ante la caída del sistema?			
• ¿Los casos de prueba validan la respuesta del sistema a entradas ilegales o conflictivas?			
Confiabilidad			
• ¿Se ha validado el equipo para las pruebas?			
• ¿Se han validado las pruebas?			
• ¿Se han verificado todas las entradas de datos?			
• ¿Se recopilan y documentan suficientes datos para la estimación de la confiabilidad del software?			
Consistencia			
• ¿Se identifican todas las dependencias entre los procedimientos?			
Correctitud			
• ¿Concuerdan los resultados de éxito definidos en los procedimientos con el comportamiento del sistema esperado?			
Facilidad de pruebas			
• ¿Es factible realizar las pruebas con el mínimo respaldo de los desarrolladores?			
Trazabilidad			
• ¿Los procedimientos de prueba indican todas las especificaciones, procedimientos, guías o manuales requeridos para su operación?			
• ¿Es visible la trazabilidad entre los requerimientos y la combinación de las pruebas?			

Checklist: Manual del usuario				
1. Identificación del proyecto y del producto				
Proyecto:				
Producto:				
2. Inspector				
Nombre		**Rol**		
e-mail	**Fono**	❒ Moderador	❒ Secretario	
		❒ Presentador	❒ Observador	
		❒ Inspector		
3. Checklist				
		Sí	**No**	**N/A**
Adherencia				
• ¿El documento se adhiere a los estándares definidos?				
Claridad				
• ¿La terminología utilizada es consistente con los conocimientos del usuario?				
• ¿La documentación es de fácil lectura y comprensión para el usuario final?				
• ¿El manual del usuario se encuentra a nivel del usuario final del software?				

Completitud			
• ¿El manual del usuario describe toda la funcionalidad del sistema?			
• ¿El manual del usuario entrega las referencias pertinentes al usuario?			
• ¿El manual del usuario detalla las interrelaciones entre las funciones del software?			
• ¿El manual del usuario describe la ayuda en línea que provee el sistema?			
• ¿El manual del usuario referencia los tutoriales disponibles para el sistema?			
• ¿El manual del usuario referencia las interfaces con otros sistemas?			
• ¿Se cita en el manual del usuario los estándares de performance acordados?			
Mantenibilidad			
• ¿El manual del usuario fue diseñado para facilitar su mantenibilidad?			

A.4. Checklist del Equipo de Revisión

***Checklist* del Moderador**		
1. Identificación		
Proyecto:	**Fecha de recepción:** __/__/____	
Producto:	**Fecha de entrega:** __/__/____	
Nombre:		
E-mail:	**Fono:**	
2. Checklist		
	Sí	**No**
• ¿Se registró el esfuerzo invertido en cada una de las etapas?		
Planificación		
• ¿Trabajó con el jefe de proyectos en la selección de los participantes y en la asignación de roles?		
• ¿Determinó el tamaño del producto y los criterios de aprobación?		
• ¿Determinó la necesidad de una reunión de orientación?		
• ¿Planificó la reunión de orientación y/o la de inspección?		
• ¿Creó y distribuyó el paquete de revisión?		
• ¿Confirmó la recepción del paquete de revisión con los participantes?		
Orientación		
• ¿Confirmó la comprensión del producto, explicó los roles asignados y respondió a cualquier consulta de los participantes?		
Preparación		
• ¿Se preparó para la inspección utilizando la checklist de inspector?		
• ¿Revisó la completitud de los Informes de Revisión?		
• ¿Determinó si todos los participantes se han preparado adecuadamente para la inspección?		
Inspección		
• ¿Enunció los roles, el enfoque y entregar pautas sobre la inspección?		
• ¿Al finalizar la reunión, decidió si se requiere de tiempo adicional?		
• ¿Al finalizar la reunión, asignó los defectos no resueltos?		
• ¿Al finalizar la reunión, decidió si una revisión adicional es necesaria?		
• ¿Al finalizar la reunión, solicitó retroalimentación a los participantes y al observador?		
• ¿Al finalizar la reunión, consultó al autor sobre el tiempo estimado para llevar a cabo las correcciones?		
Seguimiento		
• ¿Verificó que todos los defectos y errores de forma hayan sido corregidos satisfactoriamente?		
• ¿Informó y distribuyó los resultados de la revisión?		

Checklist del Autor		
1. Identificación		
Proyecto:	**Fecha de recepción:** __/__/____	
Producto:	**Fecha de entrega:** __/__/____	
Nombre:		
E-mail:	**Fono:**	
2. Checklist		
	Sí	**No**
• ¿Registró el esfuerzo invertido en cada una de las etapas?		
Planificación		
• ¿Confirmó la completitud del producto de trabajo para la revisión?		
• ¿Recomendó posibles participantes al moderador?		
• ¿Apoyó al moderador en determinar el tamaño del producto y los criterios de aprobación?		
• ¿Apoyó al moderador en la decisión de la necesidad de una reunión de orientación?		
Orientación		
• ¿Describió globalmente el producto de trabajo?		
• ¿Respondió a las consultas de los participantes?		
Inspección		
• ¿Reconoció los defectos detectados en el producto de trabajo?		
• ¿Entregó breves explicaciones técnicas sobre el producto?		
Rework		
• ¿Corrigió todos los defectos identificados en el producto?		
• ¿Corrigió todos los problemas de forma detectados en el producto?		
Seguimiento		
• ¿Verificó que todos los defectos hayan sido corregidos durante el rework?		
• ¿Cooperó con el moderador en caso de necesitarse acciones correctivas adicionales?		
• ¿Cooperó con el moderador y los inspectores en la revisión adicional si esta tiene lugar?		

Checklist del Inspector		
1. Identificación		
Proyecto:	**Fecha de recepción:** __/__/____	
Producto:	**Fecha de entrega:** __/__/____	
Nombre:		
E-mail:	**Fono:**	
2. Checklist		
	Sí	**No**
• ¿Registró el esfuerzo invertido en cada una de las etapas?		
Planificación		
• ¿Confirmó la habilidad para desempeñar el rol asignado?		
Orientación		
• Si se estimó necesario, asistió a la reunión de orientación, confirmó el rol y/o los múltiples roles asignados a la revisión.		
Preparación		
• ¿Confirmó la recepción del paquete de revisión y de cualquier información entregada por el moderador?		
• ¿Revisó los *templetes, checklist*, guías y estándares pertenecientes al paquete de revisión?		
• ¿Examinó el producto de trabajo según las indicaciones impartidas por el moderador?		
• ¿Clasificó y documentó los defectos identificados en el Informe de la Revisión?		
• ¿Hizo una lista con los errores de forma (léxico, sintaxis, semántica, etc.)?		
• ¿Entregó el informe de inspección al moderador?		
Inspección		
• ¿Nombró los defectos pertinentes durante la presentación del producto?		
• ¿Trabajó con los demás inspectores para alcanzar consenso sobre los defectos y su clasificación?		
• ¿Entregó la lista de errores de forma al autor?		
• ¿Expuso las lecciones aprendidas y propuso mejoras al moderador?		

Checklist del Presentador		
1. Identificación		
Proyecto:	**Fecha de recepción:** __/__/____	
Producto:	**Fecha de entrega:** __/__/____	
Nombre:		
E-mail:	**Fono:**	
2. Checklist		
	Sí	**No**
• ¿Registró el esfuerzo invertido en cada una de las etapas?		
Planificación		
• ¿Coordinó con el moderador las prioridades de la inspección de forma tal que se logren los objetivos dentro del lapso de tiempo de las 2 horas establecido para esta reunión?		
Orientación		
• Si se estimó necesario, asistió a la reunión de orientación, confirmó el rol y/o los múltiples roles asignados a la revisión.		
Preparación		
• ¿Revisó los *templates, checklist*, guías y estándares pertenecientes al paquete de revisión?		
• Si además cumplió el rol de inspector, se preparó para la inspección con la *checklist* de inspector		
• ¿Preparó la presentación del producto de trabajo?		
Inspección		
• ¿Siguió las pautas entregadas por el moderador al inicio de la reunión?		
• ¿Presentó el producto de trabajo a los participantes centrándose en los aspectos principales?		
• ¿Aseguró la participación equitativa de los miembros de la inspección?		
• ¿Participó como un inspector en igualdad de condiciones que los otros participantes?		

Checklist del Secretario		
1. Identificación		
Proyecto:	**Fecha de recepción:** __/__/____	
Producto:	**Fecha de entrega:** __/__/____	
Nombre:		
E-mail:	**Fono:**	
2. Checklist		
	Sí	**No**
• ¿Registró el esfuerzo invertido en cada una de las etapas?		
Planificación		
• ¿Coordinó con el moderador la mejor forma de registrar los defectos, acciones y lecciones aprendidas?		
Orientación		
• Si se estimó necesario, asistió a la reunión de orientación, confirmó el rol y/o los múltiples roles asignados a la revisión.		
Preparación		
• ¿Revisó los *templates, checklist*, guías y estándares pertenecientes al paquete de revisión?		
• Si además se cumplió el rol de inspector, se preparó para la inspección con la checklist de inspector		
• ¿Se preparó para inspección revisando los planes y reuniendo los recursos necesarios para un registro más oportuno de la inspección?		
Inspección		
• ¿Cuándo se llegó a consenso con relación a un defecto, registró la información oportuna?		
• ¿Advirtió a los otros participantes si su capacidad de registrar los acontecimientos se vio sobrepasada?		
• Al final de la inspección, leyó la información registrada para su verificación.		
• ¿Participó como un inspector en igualdad de condiciones que los otros participantes?		
• ¿Registró las acciones, asignaciones y fechas acordadas durante el término de la reunión?		
• ¿Registró las mejoras propuestas y las lecciones aprendidas expuestas según las indicaciones del moderador?		

Checklist del Observador			
1. Identificación			
Proyecto:		**Fecha de recepción:** __/__/____	
Producto:		**Fecha de entrega:** __/__/____	
Nombre:			
E-mail:		**Fono:**	
2. Checklist			
		Sí	**No**
• ¿Registró el esfuerzo invertido en cada una de las etapas?			
Planificación			
• ¿Coordinó con el moderador la mejor forma de capturar las lecciones aprendidas y de registrar las observaciones sin interrumpir el transcurso habitual de la inspección?			
Preparación			
• ¿Revisó la documentación y las bases de la revisión de acuerdo al tipo de producto de trabajo con el apoyo del moderador?			
Inspección			
• ¿Registró observaciones sobre las áreas que requieren de mejoramiento en relación a los roles individuales y sus interacciones?			
• ¿Registró las mayores diferencias entre las guías entregadas y la práctica real?			
• ¿Registró la adherencia al proceso definido?			
• ¿Preparó un resumen de las observaciones para entregarlo al moderador al finalizar la inspección?			

A.5. Checklist del Equipo de Auditoria

Checklist: SQA		
1. Identificación de la auditoría		
Institución auditada:		
Proyecto:	**Fase del ciclo de vida**	
Iniciador:	❒ Planificación	❒ Integración y pruebas
Tipo de auditoría: ❒ Interna	❒ Esp. de Requerimientos	❒ Aceptación y entrega
❒ Externa	❒ Diseño	❒ Mantención
	❒ Implementación	
2. Auditor		
Nombre		
e-mail	**Fono**	
3. Checklist		
	Sí	**No**
• ¿Se creó un plan de SQA como parte del plan del proyecto? ¿Se encuentra actualizado?		
• ¿El plan de SQA fue revisado y aprobado?		
• El plan de SQA incluye: (a) los requerimientos para SQA y las actividades que deben ser desarrolladas, (b) la calendarización de las actividades definidas, (c) los recursos requeridos, (d) la participación de SQA en el desarrollo de software, (e) la participación de SQA en el proceso de SCM, y (f) la participación de SQA en el proceso de pruebas.		
• ¿Existe evidencia sobre la implementación de las actividades de SQA?		

Checklist: Proceso de Documentación

1. Identificación de la auditoría

Institución auditada:

Proyecto:	**Fase del ciclo de vida**	
Iniciador:	❒ Planificación	❒ Integración y pruebas
Tipo de auditoría: ❒ Interna	❒ Esp. de Requerimientos	❒ Aceptación y entrega
❒ Externa	❒ Diseño	❒ Mantención
	❒ Implementación	

2. Auditor

Nombre	
e-mail	**Fono**

3. Checklist

	Sí	No
• ¿Existen estándares definidos para preparar la documentación de los productos de trabajo?		
• ¿La documentación existente se ajusta a dichos estándares?		
• ¿Existen procedimientos documentados para asegurar la adherencia a estos estándares?		
• ¿Estos procedimientos distinguen los cambios a los documentos bajo control de configuración del software? ¿Este tipo de cambios es revisado?		
• ¿El contenido de la documentación de los productos de trabajo es clara, concisa, completa y comprensible?		
• ¿Los miembros de las revisiones de esta documentación se encuentran lo suficientemente familiarizados con ella como para detectar inconsistencias fácilmente?		
• ¿Existe una autoridad competente para la aprobación de la documentación de los entregables (productos de trabajo)? ¿Es visible para los desarrolladores?		
• ¿Se entrega oportunamente la documentación solicitada por el cliente?		
• ¿Existen suficientes copias de los documentos?		
• ¿La documentación es desarrollada paralelamente a las otras actividades del desarrollo de software? ¿Refleja el estado real del proyecto y de los productos de trabajo?		

Checklist: SCM

1. Identificación de la auditoría

Institución auditada:

Proyecto:	**Fase del ciclo de vida**	
Iniciador:	❒ Planificación	❒ Integración y pruebas
Tipo de auditoría: ❒ Interna ❒ Externa	❒ Esp. de Requerimientos	❒ Aceptación y entrega
	❒ Diseño	❒ Mantención
	❒ Implementación	

2. Auditor

Nombre	
e-mail	**Fono**

3. Checklist

	Sí	No
• ¿Se preparó un plan de SCM? ¿Se encuentra actualizado?		
• ¿El plan de SCM fue revisado y aprobado?		
• ¿Se definen en el plan los mecanismos de selección e identificación de ítems de configuración? ¿Se define el esquema de versiones y revisiones?		
• ¿Los procedimientos de SCM son implementados adecuadamente? ¿Existen procedimientos para el acceso a la librería del software?		
• ¿Existe un grupo de SCM con responsabilidades bien definidas? ¿Cuenta con los recursos adecuados?		
• ¿Las líneas bases se ajustan a los requerimientos?		
• ¿Existen procedimientos para gestionar el control de cambios adecuadamente?		
• ¿Existe un CCB? ¿Quiénes pertenecen al él? ¿SQA forma parte del comité? ¿Existen procedimientos claros para sus actividades? ¿Sus actividades son monitoreadas?		
• ¿Se mantiene información sobre el estado de la configuración del software? ¿Actualizada?		
• ¿El plan de SCM contempla las auditorías FCA y PCA? ¿Se llevan a cabo?		

Checklist: Librería del software

1. Identificación de la auditoría

Institución auditada:

Proyecto:	**Fase del ciclo de vida**	
Iniciador:	❒ Planificación	❒ Integración y pruebas
Tipo de auditoría: ❒ Interna ❒ Externa	❒ Esp. de Requerimientos	❒ Aceptación y entrega
	❒ Diseño	❒ Mantención
	❒ Implementación	

2. Auditor

Nombre	
e-mail	**Fono**

3. Checklist

	Sí	No
• ¿Se ha establecido una librería del software? ¿Se ha asignado un responsable?		
• ¿Existen procedimientos adecuados para el acceso y la gestión de la librería del software?		
• ¿Se documentan apropiadamente las versiones de los productos de trabajo?		
• ¿Existe un índice de los tópicos de la librería del software? ¿Actualizado?		
• ¿Existe un registro del ingreso/salida (check in/chek out) de los entregables de la librería del software?		
• ¿Se asigna a cada ítem un identificador que refleje la versión y el tipo de producto de trabajo?		
• ¿Se controla la gestión de la librería del software? ¿Cómo?		

Checklist: Identificación y seguimiento de problemas		
1. Identificación de la auditoría		
Institución auditada:		
Proyecto:	**Fase del ciclo de vida**	
Iniciador:	❒ Planificación	❒ Integración y pruebas
Tipo de auditoría: ❒ Interna ❒ Externa	❒ Esp. de Requerimientos	❒ Aceptación y entrega
	❒ Diseño	❒ Mantención
	❒ Implementación	
2. Auditor		
Nombre		
e-mail	**Fono**	

3. Checklist	**Sí**	**No**
• ¿Existen procedimientos que aseguren la detección y corrección de los problemas y/o discrepancias detectadas?		
• ¿Se examinan los informes de problemas y de discrepancias para determinar las posibles causas?		
• ¿Se analiza la relación entre las diferentes actividades de desarrollo para prevenir disconformidades en los productos?		
• ¿Se definen y planifican acciones correctivas? ¿Se asignan los recursos adecuados?		
• ¿Las acciones correctivas son registradas y documentadas minuciosamente?		
• ¿Se revisan y monitorean las acciones correctivas para determinar su efectividad, completitud y complacencia respecto de los estándares?		
• ¿El nivel de gestión apoya las acciones correctivas?		
• ¿Los desarrolladores están de acuerdo en generar informes de problemas y de discrepancias? ¿Los utilizan?		

Checklist: Estado del proyecto		
4. Identificación de la auditoría		
Institución auditada:		
Proyecto:	**Fase del ciclo de vida**	
Iniciador:	❒ Planificación	❒ Integración y pruebas
Tipo de auditoría: ❒ Interna ❒ Externa	❒ Esp. de Requerimientos	❒ Aceptación y entrega
	❒ Diseño	❒ Mantención
	❒ Implementación	
5. Auditor		
Nombre		
e-mail	**Fono**	

6. Checklist	**Sí**	**No**
• ¿El estado real del proyecto concuerda con la planificación? ¿Si no es así, que tan grande es la brecha?		
• De acuerdo con el plan de proyecto: ¿cuál es el estado de las actividades, recursos, productos de trabajo, hitos?		
• Determinar: (a) fase de desarrollo actual, (b) estado de avance de las actividades, (c) conformación y organización del equipo desarrollador, (d) productos de trabajo, (e) hitos, y (f) resultados de las revisiones.		

A.6. Plantillas de pruebas

Prueba Unitaria:	**ID:**
Caso de prueba:	
Requisito Funcional asociado:	
Propósito:	
Prerrequisitos:	
Datos de entrada:	
Pasos:	
Resultados esperados:	
Resultado Obtenido:	

Prueba de Integración:	**ID:**
Caso de prueba:	
Unidades Relacionadas:	
Propósito:	
Prerrequisitos:	
Datos de entrada:	
Pasos:	
Resultados esperados:	
Resultado Obtenido:	

Prueba de Humo:	**ID:**
Caso de prueba:	
Requisito:	
Prerrequisitos:	
Datos de entrada:	
Pasos:	
Resultados esperados:	
Resultado Obtenido:	

Prueba de Performance:	**ID:**
Caso de prueba:	
Requisito:	
Cantidad de solicitudes por minuto:	
Prerrequisitos:	
Datos de entrada:	
Pasos:	
Tiempo promedio:	

Prueba de Aceptación:	**ID:**
Caso de prueba:	
Reporte del cliente:	
Prerrequisitos:	
Datos de entrada:	
Pasos:	
Resultados esperados:	
Resultado Obtenido:	