



Plan de Calidad para Producto de Software “Toc&Talk!”

**Memoria para optar al título de:
Ingeniero de Ejecución en Informática**

Felipe Alexis Mancilla Sepúlveda

**Profesor Guía:
Marcello Alfredo Visconti Zamora**
**Profesor Correferente:
Luis Hevia Rodríguez**

Octubre 2017

Resumen

En el presente informe, se trabajara sobre la documentación del “Plan de Calidad” para una versión comercial de la plataforma web “Toc&Talk!”. El cual es una plataforma web donde dos personas puedan practicar un idioma hablando, mediante el uso de videollamadas, entregando así un espacio virtual para hablar el idioma que les interesa, y mejoren sus capacidades lingüísticas. Por ende, en este documento se explicaran los atributos de calidad que debe tener el sistema y especificar lo que se pide de cada una, definir los procesos de desarrollo con sus actividades, entregables y fijando puntos de revisión para evaluarlos, presentar la estructura del Área de QA, con las responsabilidades y actividades que tendrá cada integrante y concretar los casos de prueba que deberán ser desarrollados con el fin de cumplir el objetivo de este plan.

Abstract

In this report, we will work about the documentation of the “Quality Plan” for a comercial versión of the web plataform “Toc&Talk!”. It’s a web plataform where two people can practice a language, speaking it, using videocalls, thus providing a virtual space to talk the language that interests them, and improve their language skills.

Therefore, this document will explain the quality attributes that the system must have and specify what is required for each one, define the development processes with their activities, work products and setting review points to evaluate them, present the structure of the área of QA, with the responsibilities and activities that will have each member and specify the test cases that must be developed in order to satisfy the objective of this plan.

INDICE

1. INTRODUCCIÓN.....	5
1.1. Propósito.....	6
1.2. Alcance.....	6
2. REQUERIMIENTOS NO FUNCIONALES.....	7
2.1. Definiciones de requerimientos no funcionales.....	7
2.2. Objetivos Cuantificables.....	8
2.2.1. Usabilidad.....	8
2.2.2. Adaptabilidad.....	9
2.2.3. Seguridad.....	11
2.2.4. Modularidad.....	13
2.2.5. Disponibilidad.....	13
2.2.6. Recuperabilidad.....	14
2.2.7. Capacidad.....	15
2.2.8. Rendimiento.....	16
3. MODELO DE DESARROLLO.....	18
3.1. Elección de Modelo.....	18
3.2. Definición de Actividades.....	18
3.3. Definición de Entregables de Trabajo.....	19
3.3.1. Plan de Pruebas.....	19
3.3.2. Plan de Calidad.....	20
3.4. Atributos por Calidad.....	21
3.4.1. Por Actividades.....	21
3.4.2. Por Entregable.....	23
3.5. Puntos de Revisión.....	24
4. GESTION DE CALIDAD.....	26
4.1. Equipo de Trabajo.....	26
4.2. Recursos.....	27
4.2.1. Personal.....	27
4.2.2. Infraestructura.....	28
4.2.3. Actividades.....	29
4.2.4. Responsabilidades.....	32
5. ANALISIS DE RIESGOS.....	33
5.1. Identificación de Riesgos.....	33
5.2. Clasificación de Riesgos.....	35
5.3. Priorización de Riesgos.....	37

5.4. Control de Riesgos.....	38
6. HERRAMIENTAS TECNICAS Y REVISIONES.....	38
6.1. Tecnicas.....	38
6.2. Herramientas de desarrollo y puesta en marcha.....	39
6.3. Actividades y Revisiones.....	39
6.3.1. Revisiones e Inspecciones.....	40
6.3.2. Actividades.....	44
6.4. Registro y generación de informes.....	47
6.5. Revisión por Checklist.....	48
7. PRUEBAS.....	49
7.1. Estructura de Pruebas.....	49
7.1.1. Planificación.....	50
7.1.2. Especificación.....	51
7.1.3. Ejecución.....	52
7.1.4. Analisis de Resultados.....	52
7.1.5. Completación.....	53
7.2. Pruebas a Considerar.....	53
7.2.1. Protocolos de Acceso.....	53
7.2.2. Almacenamiento de datos privados por encriptación.....	53
7.2.3. Tiempo de acceso de la funcionalidad.....	53
7.2.4. Optimizar imágenes.....	54
7.2.5. Mantener orden del codigo.....	54
7.3. Casos de Pruebas.....	54
7.3.1. Pruebas por Atributo de Calidad.....	54
8. CONCLUSIONES.....	60
9. BIBLIOGRAFIA.....	64
10. ANEXOS.....	66

1. Introducción.

Esta plataforma nace como una herramienta para colaborar en el manejo de un idioma distinto al nativo; lo cual ha pasado de ser motivación personal a una necesidad fundamental

tanto en ámbitos profesionales y académicos, como de entretenimiento y ocio. No obstante, la mayoría de los servicios que ofrece el mercado para enfrentar esta situación, se enfoca en el manejo escrito del idioma, careciendo de herramientas para el desarrollo oral de éste; sumado a que los pocos servicios existentes se encuentran a un valor poco asequible para la población en general.

Para satisfacer esta necesidad, el equipo pretende crear una plataforma web y de fácil acceso, en donde las personas de distintos lugares del mundo que quieran practicar oralmente un idioma puedan hacerlo, proveyendo un servicio virtual en donde se permita desarrollar las habilidades de comunicación y expresión oral en base a la conversación hablada entre los mismos usuarios de manera gratuita. En el tema del cliente, este corresponde a los usuarios, que son aquellas personas que quieren desarrollar competencias lingüísticas en el ámbito oral en algún idioma de interés. El potencial grupo de usuarios a quienes va dirigida esta aplicación está compuesto por estudiantes, universitarios y profesionales en general.

Con este software se pretende crear un espacio para poder practicar un idioma oralmente, a través de una interfaz gráfica, sencilla y fácil de usar, compatible con diferentes dispositivos, permitiendo el acceso de los usuarios desde cualquier lugar donde puedan conectarse, resguardando la privacidad de sus datos. La forma en la que se logran establecer interacciones entre usuarios, es la funcionalidad más relevante que ofrece este servicio a disposición de su comunidad, teniendo presente que, para el usuario, el uso de esta plataforma no poseerá costo alguno.

Esta idea se distingue del resto de los servicios más conocidos, en que no existe dentro de ellos una plataforma que oriente dicha solución al desarrollo de las competencias lingüísticas orales necesarias para llevar a cabo una conversación fluida en un idioma que no es el nativo; dando la libertad de poder conectarse en cualquier momento y en cualquier lugar. Además, el usuario, por su parte, no será quien deba buscar con “quién” hablar, sino que sólo debe tener la voluntad para hacerlo, haciendo de su conversación una experiencia cómoda, grata y única. Finalmente, durante la etapa de la “Feria de Software”, se construye una primera versión funcional del sistema para su presentación en el primer feedback de los usuarios.

Para su versión comercial es fundamental que el prototipo pase por una gran cantidad de pruebas de diferente índole, con la finalidad de reducir el riesgo de

posibles errores (o bugs) durante su salida al mercado. Además se presentarán los riesgos críticos del proyecto, el plan de mitigación y contingencia de los doces más urgentes en resolver.

1.1. Propósito.

Para tener una versión comercial del software, es fundamental que se confeccione un documento de calidad donde se especifiquen los atributos de calidad del sistema, cómo se medirán, y cuál(es) será(n) su(s) criterio(s) de validación. Además de explicar el modelo de desarrollo, se profundizará en las actividades que se desarrollarán y los entregables de cada actividad. Finalmente se detallarán las técnicas que se deben aplicar y las pruebas que deben chequear para el correcto funcionamiento del sistema.

1.2. Alcance.

El alcance de este informe, será definir los atributos de calidad que posteriormente serán probados en el sistema, los procedimientos a trabajar y los documentos a utilizar para las revisiones de cada parte del software. Para esto, será necesario planificar el curso de acción para cada módulo a probar, los recursos a utilizar y el personal para la realización de estas actividades. Otro factor a considerar será documentar los riesgos asociados a cada fase de desarrollo, sus posibles consecuencias y cómo mitigarlas.

Finalmente especificar los puntos a revisar en cada chequeo de un área del sistema (calidad de los procesos), los productos obtenidos de cada proceso de desarrollo y su equipo de revisión; y la estructura que deben tener las pruebas.

2. Requerimientos no funcionales.

Los requerimientos no funcionales, o también llamados atributos de calidad, se refieren a las características inherentes de un sistema, es decir; los requisitos que se complementan con las funcionalidades del software para cumplir con el objetivo final que busque entregar un software de calidad que sea eficaz y eficiente, permitiendo cumplir con las expectativas del usuario.

2.1. Definición de los requerimientos no funcionales.

Para medir la calidad de un producto, es requerido saber estos requerimientos, por lo que para esta plataforma web “Toc&Talk!”, se definen sus atributos de calidad:

1. **Usabilidad:**

Cualidad de una página web o del programa informático que son sencillos de usar porque facilitan la lectura de los textos, descargan rápidamente la información y presentan funciones y menús sencillos, por lo que el usuario se encuentra cómodo con su uso.

2. **Adaptabilidad:**

Cualidad de un software de adaptarse a otros ambientes.

3. **Seguridad:**

Capacidad de un sistema para verificar el acceso y manejo de personas no autorizadas.

4. **Modularidad:**

Propiedad que permite subdividir una aplicación en partes más pequeñas (llamadas módulos), cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las partes restantes.

5. **Disponibilidad:**

Es el porcentaje del tiempo que un sistema es capaz de realizar las funciones para las que está diseñado. Para saber cuánto tiempo está activo y en funcionamiento.

6. **Recuperabilidad:**

Verificar si el software puede reasumir el funcionamiento y restaurar datos perdidos después de un fallo ocasional.

7. **Capacidad:**

Propiedad del software para almacenar datos y el orden en que los almacena.

8. Rendimiento:

Determinar lo rápido que realiza una tarea un sistema en condiciones particulares de esfuerzo. Puede servir para verificar y validar otros atributos de calidad, como la fiabilidad, escalabilidad, etc.

2.2. Objetivos Cuantificables

Para cada atributo de calidad mencionado anteriormente, se deben definir sus objetivos asociados, pues ayudarán a entregar una medición de manera correcta y el argumento de por qué son necesarios que aparezcan en el software.

Estos objetivos deben ser claros y concisos dentro del equipo de desarrollo, ya que de ellos se derivará al equipo de Testing, quienes estarán a cargo de realizar la validación de cada propósito a medir en dicho informe. También se fijarán los criterios de aceptación/validación de cada objetivo, esto de acuerdo a lo que el equipo requiera que se determine en el producto final.

1. Usabilidad:

En el producto final, se busca que la interfaz con el que interactúa el usuario, sea lo más fácil y rápido en entender, además de ser considerarlo atractivo ante sus ojos y de esa manera tener mas efectividad al mostrar las funcionalidades que tiene el sistema. De esto es fundamental para el equipo, pues como dicen “la primera impresión lo es todo”, queremos que el usuario se sienta cómodo con los colores y palabras que manejamos en la interfaz.

a. Objetivos:

- i. Tamaño y fuente de letra entendible para el usuario.
- ii. Colores atrayentes para el ojo humano.
- iii. Diseño simple y minimalista.
- iv. Mínimo uso de click para llegar a un fin.
- v. Mínimo tiempo de aprendizaje del sistema.

b. Criterios de validación:

- i. Tamaño y fuente de letra entendible para el usuario:

Criterio de Validación	Medición
Distancia de lectura correcta de la interfaz.	Máximo 50 cm

- ii. Colores atractivos para el ojo humano.

Criterio de Validación	Medición
Contraste de los colores.	Ningún color opaque a otro.

- iii. Diseño simple y minimalista.

Criterio de Validación	Medición
Cantidad de botones por vista.	Máximo 5 botones.
Cantidad de imágenes por vista	Máximo 2 imágenes.

- iv. Mínimo uso de click para llegar a un fin.

Criterio de Validación	Medición
Cantidad de click para llegar a una vista	Mínimo 2 click Máximo 3 click.
Tiempo para llegar a cualquier vista	Máximo 6 segundos.

- v. Mínimo tiempo de aprendizaje del sistema.

Criterio de Validación	Medición
Tiempo de aprendizaje.	Máximo 5 min.

2. Adaptabilidad:

Como misión de equipo, se busca que el producto sea de multiplataforma y que pueda adaptarse a ambientes extremos; tales como ser usado por miles de usuarios sin sobrecargarse. Lo anterior cobra relevancia ya que el principal riesgo es la cantidad de usuarios que usará este sistema, por lo que debemos tener una ambiente confiable para que puedan utilizar la videollamada entre cientos de usuarios.

a. Objetivos:

- i. Cantidad aceptable de usuarios concurrentes para lograr la finalidad el producto.
- ii. Las plataformas donde el sistema podrá desarrollarse.
- iii. Cantidad de idiomas en que el sistema estará definido.

b. Criterios de aceptación:

- i. Cantidad aceptable de usuarios concurrentes para lograr la finalidad el producto.

Criterio de Validación	Medición
Cantidad de usuarios concurrentes sin que el servidor caiga.	Mínimo 1000 usuarios
Cantidad de videollamadas concurrentes sin que el servidor caiga.	Mínimo 500 videollamadas.

- ii. Las plataformas donde el sistema podrá desarrollarse.

Criterio de Validación	Medición
Plataformas web	Funcione perfectamente en Google Chrome, Mozilla Firefox, Safari e IE.
Plataformas móviles	Funcione en cualquier smartphone con cámara. Sistema operativo: Android Versión > 4.0, IOS 7.

- iii. Cantidad de idiomas en que el sistema estará definido.

Criterio de Validación	Medición
Idiomas en que estará el sistema.	Máximo 5 idiomas. Español, Inglés, Francés, Italiano y Chino.

3. Seguridad:

Como el sistema trabajará con cuentas de información importante del usuario, es obligatorio que esos datos no se transparenten al público, y por lo tanto desarrollar un ambiente seguro para guardarlos en el servidor, como también filtrar ciertos datos para mostrar como identidad del usuario a los demás. También es necesario construir un servidor seguro para enfrentar posibles ataques como Malware o revisar el código de la página para saber cómo acceder a datos privilegiados, etc. Por ende, el equipo debe cumplir con ciertos criterios en base a las consultas que se haga desde la página web al servidor.

a. Objetivos:

- i. Inicio de sesión segura.
- ii. Seguridad en el registro de datos.
- iii. Seguridad en la recuperación de password.
- iv. Consultas seguras.
- v. Password seguro.
- vi. Seguridad en los servidores.
- vii. Protocolos aceptables.

b. Criterios de Validación:

- i. Inicio de sesión segura.

Criterio de Validación	Medición
Inicio de sesión segura.	Se pedirá un username y password para ingresar a su cuenta. Al fallar 3 veces, se le pedirá iniciar sesión en "5 min más". A la 5° vez, se bloqueará la cuenta pidiendo acceder a "Recuperación de contraseña".

- ii. Seguridad en el registro de datos.

Criterio de Validación	Medición
Registro de datos	Password debe aparecer en la BD encriptado bajo el algoritmo MD5. Respuesta a la consulta SQL no debe mostrar ningún dato

	en la dirección web.
--	----------------------

iii. Seguridad en la recuperación de password.

Criterio de Validación	Medición
Recuperación de Password Segura.	Se debe responder una pregunta secreta (especificada en “Registro Usuario”) y al responder correctamente, darle la opción de cambiar su contraseña y enviar al email su confirmación. De no ser bien respondida, entregar mensaje de error.

iv. Consultas seguras.

Criterio de Validación	Medición
Consultas de datos del usuario.	Sólo en la página dentro del tiempo de sesión del usuario. Sólo consultas HTTP POST.

v. Password seguro.

Criterio de Validación	Medición
Password Seguro	Debe contener al menos 8 caracteres. (Mínimo 1 dígito).

vi. Seguridad en los servidores.

Criterio de Validación	Medición
Seguridad en los servidores	Certificado SSL en cada servidor. Eliminación de Malware. Certificado de firma de código.

vii. Protocolos aceptables en el servidor.

Criterio de Validación	Medición
Protocolos aceptables en el servidor	HTTPS:443 HTTP: 80 SSH:20 P2P

4. **Modularidad:**

Para temas del soporte del sistema, se busca que el software en su conjunto este creado por módulos independientes entre sí. En base a esto, se podrían testear las funcionalidades generales de forma más eficiente, como también, ser más preciso y veloces en las soluciones de los problemas que pueda presentar en su funcionamiento. Además para temas de actualización del sistema, se seguirá desarrollando módulos con funciones similares entre sí, para acoplarlos de forma rápida en el sistema sin afectar a los otros módulos. Es por esta finalidad que cada “paquete de código” debe ser independiente entre sí, para facilitar los cambios que se pueda dar en uno, sin afectar a sus otras partes.

a. **Objetivos:**

i. Funciones en módulos.

b. **Criterios de Validación:**

i. Funciones en módulos.

Criterio de Validación	Medición
Funciones en un módulo	Máximo 5 funciones en un módulo.

5. **Disponibilidad:**

Un criterio importante de un sistema es su porcentaje de activo funcionamiento, ya que entre más tiempo estén sus funciones disponibles aumentará la experiencia del usuario frente al sistema y por lo tanto mejorará su uso durante su curso. Para cumplir esto, es necesario que el sistema maneje la mayor cantidad de consultas que lleguen de parte de los

usuarios, como también trabajar de manera paralela frente a una gran cantidad de usuarios concurrentes que utilizan recursos del sistema, como hacer las videollamadas. Y como uno de los riesgos es que no haya un usuario disponible para hablar durante la videollamada, bajos ciertos criterios, entonces el tiempo que debe estar activo el sistema debe llegar ser constante durante el tiempo.

a. Objetivos:

- i. Uptime del sistema.
- ii. Tiempo Efectivo durante las videollamadas.

b. Criterios de Validación:

- i. Uptime del sistema.

Criterio de Validación	Medición
Sistema Disponible	Uptime del sistema debe ser de 99%

- ii. Tiempo Efectivo durante las videollamadas.

Criterio de Validación	Medición
Tiempo Efectivo Videollamadas.	Máximo 45 minutos.

6. Recuperabilidad:

Una de las metas que debe tener un sistema, es su capacidad para restaurarse frente a un problema que afecta al funcionamiento local o global de su infraestructura. El tiempo en que reasume su actividad es fundamental para que el producto siga siendo utilizado, dado que actualmente la rapidez con la que obtiene respuesta el usuario frente a peticiones en temas de servicios (como aplicaciones móviles, web, etc.) es cada vez más corta y eso produce cierto rechazo a respuestas que no cumple con ese tiempo de proceso. Y finalmente, el tener un sistema sin uso por harto tiempo, produce un gran riesgo de que se “desencante” de la plataforma, pudiendo reaccionar en la baja de usuarios.

a. Objetivos:

- i. Tiempo de recuperación del sistema.

- ii. Recuperación de los datos frente a una falla en el servidor.
- iii. Respuesta frente a fallas.

b. Criterios de Validación:

- i. Tiempo de recuperación del sistema.

Criterio de Validación	Medición
Tiempo de recuperación del sistema.	Máximo 2 minutos desde la caída del servidor.

- ii. Recuperación de los datos frente a una falla del servidor.

Criterio de Validación	Medición
Recuperación de los datos frente a una falla.	Máximo 10-20 segundos para acceder a los datos de un servidor espejo cuando los demás fallan.

- iii. Respuesta frente a fallas.

Criterio de Validación	Medición
Respuesta frente a fallas	Mínimo 4 servidores “espejos” que tienen el mismo base de datos (BDD).

7. Capacidad:

Como la plataforma web debe almacenar una cantidad significativa por cada usuario (como la información de su perfil, su valoración, reportes, etc) y además se debe manejar con una gran cantidad de usuarios (mínimo 10^5 usuarios), entonces el tamaño de los servidores donde se guardarán los datos debe tener un espacio considerable (mínimo 100 GB). Además como se trabajará con una cantidad mínima de 1000 usuarios concurrentes, la rapidez para acceder a los datos dependerá únicamente de cómo se guarden en el sistema.

a. Objetivos:

- i. Tamaño de Servidor.

- ii. Base de datos.

b. Criterios de Validación:

- i. Tamaño de Servidor.

Criterio de Validación	Medición
Tamaño de Servidor.	Mínimo 100 GB.

- ii. Base de datos.

Criterio de Validación	Medición
Base de datos	Se provee de la base de datos NoSql MongoDB.

8. Rendimiento:

Como se mencionó anteriormente, la rapidez de respuesta que entregue el sistema frente a una petición del usuario es un factor importante que afecta la usabilidad del producto, por ende se hizo estudios para obtener datos sobre el “tiempo aceptable” de cada función, resolver un fallo y/o atender consultas de los usuarios.

a. Objetivos:

- i. Tiempo de respuesta.
- ii. Tiempo de resolución a fallas.
- iii. Tiempo de vida de una consulta del usuario.

b. Criterios de Validación:

- i. Tiempo de respuesta.

Criterio de Validación	Medición
Tiempo de respuesta.	Cada función del sistema tiene un tiempo de respuesta de 2 a 5 segundos (dependiendo de la transacción de la funcionalidad).

ii. Tiempo de resolución a fallas.

Criterio de Validación	Medición
Tiempo de resolución a fallas.	Para fallas que puede arreglar el servidor, de 5 segundos a 1 minuto máximo. Para fallas que deben arreglar los programadores, máximo 1 día.

iii. Tiempo de vida de una consulta del usuario.

Criterio de Validación	Medición
Tiempo de vida de una consulta y/o reclamo de un usuario.	Máximo 12 horas para una respuesta.

3. Modelo de Desarrollo.

3.1. Elección del Modelo.

Durante la fase de “Análisis del Software”, fue esencial la elección de un modelo de desarrollo de SW para el producto que el equipo buscaba construir; dado que en base a ello, se haría más efectivo la metodología planificada para construir el sistema. El modelo a seguir fue “Desarrollo en fases”, el cual consiste en el desarrollo de versiones del producto, donde en cada iteración hay una nueva versión mejorada y/o con nuevas funcionalidades del sistema. Este modelo estaba ad-hoc con lo que busca el equipo de desarrolladores, por las siguientes razones:

- 1. Metodología iterativo – incremental.**
- 2. Prototipos Temprano para mostrar al Cliente y/o usuario.**
- 3. Trabajo en Paralelo.**
- 4. Metas más definidas y alcanzables.**
- 5. Desarrollo Ágil.**

En base a estos puntos referidos, el desarrollo de la plataforma “Toc&Talk!” ha podido adaptarse a los factores externos y a las malas estimaciones de tiempo para algunas funcionalidades, terminando cada versión con éxito y mostrando el sello de calidad necesarios.

3.2. Definición de Actividades.

En el proyecto a construir se sigue una serie de actividades que lo guiarán; desde la licitación de la idea del producto hasta la finalización de éste. De esta forma, y como la ingeniería de SW menciona, existe un proceso de desarrollo de forma sistemática, eficaz y eficiente, para lograr construir un software de calidad.

Para lograr lo anterior, las actividades a realizar son:

- 1. Definición de la idea básica.**
- 2. Análisis de Requisitos (Funcionales y No funcionales).**
- 3. Planificación del proyecto.**
- 4. Análisis de Riesgos.**
- 5. Diseño del Software.**
- 6. Implementación.**
- 7. Operación.**

3.3. Definición de Entregables de Trabajo.

Para llevar a cabo cada fase del proceso que lleva el proyecto, es necesario definir los entregables (output) que tiene cada proceso, detallando la finalidad que tiene para el producto final. Los entregables de trabajo de cada actividad mencionada anteriormente son:

1. **Especificación de Requisitos Funcionales y Extra-Funcionales.**
2. **Plan de Proyecto.**
3. **Plan de Riesgo.**
4. **Plan de Pruebas.**
5. **Plan de Calidad (SQA).**
6. **Plan de Gestión de Configuración (SCM).**
7. **Manual del Usuario.**

Ahora bien, se debe mencionar que el entregable más importante para los stakeholders, es la versión final del software.

Para temas de SQA, solo se es necesario saber sobre el documento de pruebas y de calidad.

1. **Plan de Pruebas:**

En este entregable, se documentan los casos de pruebas que se hacen a las versiones liberadas del sistema. Este desarrollo debe ser complementario al desarrollar cada versión del sistema, pues a medida que se implementan las funcionalidades, estas deben ser probadas para mejorar la calidad que se espera del prototipo. Además, a medida que se hacen pruebas, se encuentran defectos de manera temprana y al corregirlas, se baja la tasa de posibles fallas post-lanzamiento. Por lo tanto, el objetivo de este documento es entregar información para encontrar defectos en el sistema y validar que los requisitos pedidos por el cliente se cumplan. Existen diferentes pruebas que se deben registrar tales como:

a. **Pruebas Unitarias:**

Son las pruebas que se hace a la codificación de una función no trivial o método de un módulo.

b. **Pruebas de Integración:**

Son las pruebas que se hacen al integrar uno o más funciones a un módulo.

- c. **Pruebas** **Funcionales:**
Son las pruebas que se hacen para validar el requerimiento funcional.
- d. **Pruebas** **de** **Regresión:**
Verificar que la nueva versión cumplen con las funcionalidades de la versión anterior. Además, se controlan posibles defectos introducidos en la corrección de versiones anteriores.
- e. **Pruebas** **de** **Rendimiento:**
Son las pruebas que se hacen para validar los requisitos extra-funcionales.
- f. **Pruebas** **de** **Aceptación:**
Son las pruebas que se hacen con los usuarios para verificar que cada cosa pedida por el cliente se proyecte en el trabajo del usuario.
- g. **Pruebas** **de** **Instalación:**
Son las pruebas que se hacen para saber lo que necesitará el cliente para instalar y configurar el nuevo sistema.

La estructura a seguir en el plan es:

1. **INTRODUCCIÓN**
 - 1.1. Objetivo
 - 1.2. Alcance
2. **DEFINICIÓN DE ELEMENTOS A PROBAR**
3. **DEFINICIÓN ESTRATEGIA A UTILIZAR**
4. Recursos a emplear. Especificación de la técnica que se usará. Herramientas utilizadas. Criterio indicador de éxito de las pruebas.
5. **EQUIPO DE PRUEBAS**

Estructura: Plan de Pruebas

2. **Plan** **de** **Calidad** **(SQA):**
En este entregable, se documentan los atributos de calidad que deben ser validados en el producto. Para eso se investigan los requisitos extra-funcionales del sistema, los recursos que se necesitarán para aplicar los procesos de aseguramiento de calidad y los criterios de validación definidos para verificar que la medición de estos atributos sean lo que haya pedido el cliente o que sean valores muy aproximados (casi iguales). Uno de los ejemplos que se trabajan en este plan es la “Disponibilidad” del sistema, en el cual es importante que su métrica está aproximado a 24 horas de los 7 días de la semana.
La estructura es la siguiente [5]:

1. INTRODUCCIÓN

1.1. Objetivo

1.2. Alcance

1.3. Definiciones

1.4. Resumen

2. GESTIÓN SQA

2.1. Actividades

2.2. Resultado obtenido en las Actividades

2.3. Revisiones y auditoría

Estructura: Plan de Calidad

3.4. Atributos de Calidad.

3.4.1. Por

Actividades.

En la siguiente tabla, se definen los atributos de calidad que debe cumplir cada actividad de desarrollo, definiendo las métricas pedidas y las especificaciones de cada atributo para la actividad.

Actividad	Atributo(s)
Definición de la idea básica.	Eficiente; vale decir, debe comprender el contexto del problema y argumentar por qué es la mejor solución de las alternativas planteadas.
Análisis de Requisitos (Funcionales y No funcionales).	Eficiente, adaptable y usables. Debe identificar y especificar cada tarea que hará el sistema, representando en su conjunto el 100% de la solución a la necesidad del cliente. Además deben ser adaptables en el tiempo, producto de los cambios que pueda realizar el cliente en en cualquier fase del proyecto. Y finalmente cada requisito aprobado por el cliente debe darle valor al sistema y asegurar su usabilidad en el producto final.
Planificación del proyecto.	Adaptable y fiable. Debe haber un margen de error de 2 a 5 días en cada actividad; y además debe ser adaptable a cualquier mala estimación de la tarea a realizar a un

	tiempo acordado entre el desarrollador y el cliente.
Análisis de Riesgos.	Eficiente, adaptable y fiable. Debe ser capaz de identificar los posibles riesgos del proyecto y generar estrategias para mitigar el 80% de cada situación negativa, como ser fácil de modificar un campo para mejorar la mitigación como ser fiables para confiar y tener un margen muy pequeño de error en la estimación.
Diseño del Software.	Modulable y adaptable. Deben existir interfaces intuitivas que trabajen con cada módulo del código, para tener una versión que se pueda adaptar al feedback del cliente, como también añadir diseños de interfaces cuando se trabajen con el siguiente módulo. Se debe diseñar el 100% del módulo para estar a disposición de las correcciones del cliente.
Implementación.	Eficiente, modulable, adaptable y testeables. Se debe implementar el 100% de cada funcionalidad diseñada en "Diseño de SW". Además, su estructura debe ser modular, donde cada tarea representa un módulo y sus subtarear submódulos. Y en base a esa modularidad, ser adaptables a correcciones. Se debe testear cada funcionalidad para cumplir el 100% del requisito.
Operación.	Eficiente, mantenible y segura. Debe estar operable en el 100% de los requisitos pedidos por el cliente, entregar soporte frente a fallas y/o consultas del sistema y tener un carácter seguro frente a las consultas de los datos privilegiados de los usuarios, donde sólo él puede hacerlo a sus propios datos.

3.4.2. Por Entregable.

En la siguiente tabla, se definen los atributos de calidad que debe cumplir cada Entregable que se obtiene de las actividades de desarrollo, definiendo las métricas pedidas y las especificaciones de cada atributo.

Entregable	Atributo(s)
Especificación de Requisitos Funcionales y Extra-Funcionales.	Eficiente, adaptable y usables. Debe documentar el 100% de los requisitos funcionales y extra-funcionales acordados con el cliente, complementándose con el diagrama de Casos de Usos y/o Modelo de Dominio. Cada requisito pertenece a una solución parcial del sistema y que demuestre que será usado. Además debe ser adaptable a posibles cambios constantes con los stakeholders durante su confección.
Plan de Proyecto.	Fiable y adaptable. Debe documentar el 100% de las actividades y subactividades, con un tiempo estimado a realizarlas, para así desarrollar el producto final. Debe existir un margen de error entre actividades de 2 a 5 días en caso de mal estimación.
Plan de Riesgo.	Eficiente, fiable y adaptable. Debe documentar el 100% de los posibles riesgos que podrían afectar al proyecto y las estrategias para mitigarlos. Debe tener una investigación previa, para mostrar fiabilidad en los puntos de mitigación de la probabilidad e impacto y ser adaptable durante el desarrollo en caso de mala estimación.
Plan de Pruebas.	Eficiente, modulable y fiable. Debe documentar el proceso de cada prueba: unitaria, integración, funcional, regresión, rendimiento (stress, capacidad, etc), aceptación e instalación. En cada prueba se debe trabajar con cada módulo

	del sistema. Además, se debe entregar el 100% de los casos pruebas para cada módulo. Será necesario realizar al menos 5 pruebas para cada funcionalidad.
Plan de Calidad (SQA).	Eficiente, seguro, adaptable y usable. Se debe documentar los atributos de calidad a definir y especificar al menos 2 métricas de cada una. Este documento debe representar el 100% de las características inherentes del sistema y de lo que busca el cliente.
Plan de Gestión de Configuración (SCM).	Eficiente, adaptable y fiable. Se debe documentar el 100% de las líneas de base (“baseline”) y cada control de cambio que suceda en el desarrollo del software. Debe tener un pequeño margen de error al momento de acordar el cambio con el desarrollador.
Manual del Usuario.	Eficiente, fiable y adaptable. Debe documentar el 100% de las funciones en lenguaje natural (de forma resumida y con imágenes), al menos registrar 15 preguntas frecuentes y el contacto del soporte.

3.5. Puntos de Revisión.

En cada etapa del proyecto, se deben registrar los hitos donde se aplican los puntos de revisiones para chequear que la actividad haya sido realizada y de esa manera se podrá inspeccionar y validar los diferentes estados del proyecto.

Para cada actividad del proceso de desarrollo, se indican los distintos hitos y puntos de revisión:

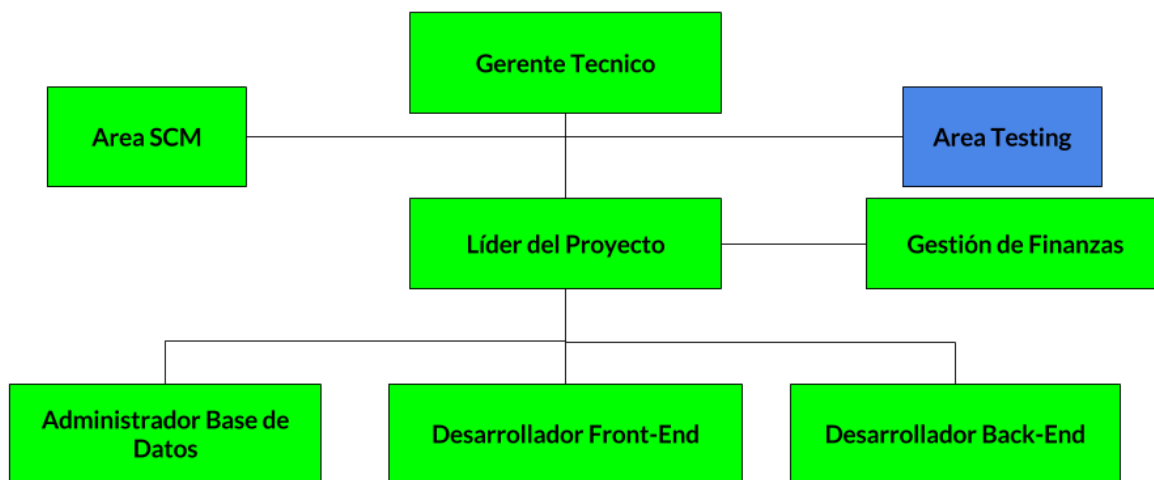
- ❖ **Definición de la idea básica**
 - Revisión y aprobación de la solución.
- ❖ **Análisis de Requisitos (Funcionales y No funcionales)**
 - Revisión y aprobación del documento “Especificación de Requisitos Funcionales y Extra-Funcionales.”
 - Reporte del estado y resultados del documento “Especificación de Requisitos Funcionales y Extra-Funcionales” del área QA.
- ❖ **Planificación del proyecto**
 - Revisión y aprobación del documento “Plan de Proyecto”.

- Revisión y aprobación del documento “Plan de Calidad (SQA)”.
- Revisión y aprobación del documento “Plan de Gestión de Configuración (SCM)”.
- ❖ **Análisis de Riesgos**
 - Revisión y aprobación del documento “Plan de Riesgo”
 - Reporte del estado y resultados del documento “Plan de Riesgo” del área QA.
- ❖ **Diseño del Software**
 - Revisión y aprobación del diseño de las interfaces.
 - Revisión y aprobación del diseño de las funciones.
 - Revisión y aprobación del modelo de base de datos.
 - Revisión y aprobación de los casos de pruebas unitarias, integración, funcional, regresión, rendimiento, aceptación e instalación.
 - Reporte del estado y resultados de las actividades anteriores de parte del área QA.
- ❖ **Implementación.**
 - Revisión y aprobación de las funcionalidades implementadas.
 - Revisión y aprobación de las interfaces implementadas.
 - Reporte de los resultados de los casos de pruebas unitarias, integración y funcional.
 - En caso de existir una versión anterior, reporte de los resultados de la prueba de regresión.
- ❖ **Pruebas de Calidad.**
 - Revisión y aprobación de los atributos de calidad.
 - Revisión y aprobación de las pruebas de rendimiento.
 - Reporte del estado de los resultados del “Plan de Calidad (SQA)”.
- ❖ **Operación.**
 - Reporte del estado de los resultados del “Plan de Gestión de Configuración (SCM)”
 - Revisión y aprobación de las modificaciones en las funciones.
 - Revisión y aprobación de las modificaciones en el diseño de interfaces.
 - Revisión y aprobación del documento “Manual del Usuario”.
 - Revisión y aprobación del equipo de soporte y los recursos.
 - Revisión y aprobación de la versión del sistema testeado.
 - Revisión y aprobación de las pruebas de aceptación.
 - Revisión y aprobación de las pruebas de instalación.
 - Reporte de estado de los resultados de las actividades anteriores en el área de QA.

4. Gestión de Calidad.

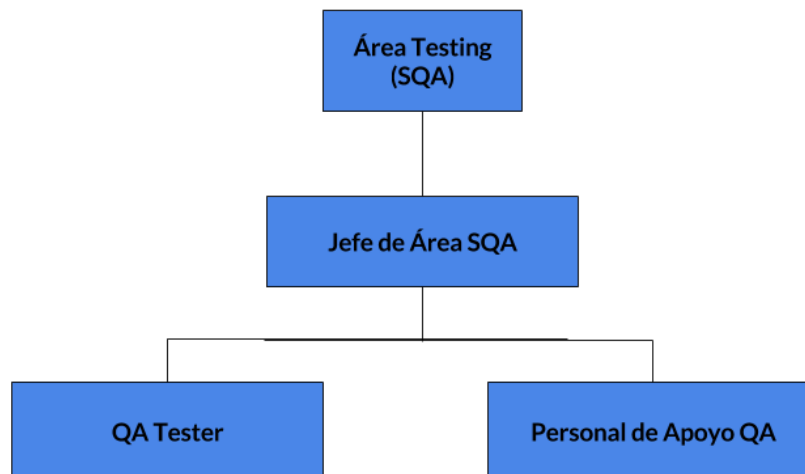
4.1. Equipo de Trabajo.

Para el desarrollo del proyecto, es necesario trabajar cada proceso de forma ágil, pues existen requerimientos funcionales complejos de diseñar en poco tiempo. Sumado a lo anterior, es importante recalcar que cada integrante del equipo debe tomar su rol con seriedad y responsabilidad, dado que se deben trabajar muchas áreas para entregar un producto de calidad y cada integrante está a cargo de un área en particular. Para ello, la estructura del equipo es la siguiente.



Equipo de Trabajo "Toc&Talk!"

Y para área de testing (QA), la estructura del equipo SQA es la siguiente.



Equipo del Área de Testing SQA "Toc&Talk!"

Las responsabilidades que tendrá cada cargo, se mencionara en el siguiente punto.

4.2. Recursos.

4.2.1. Personal.

El objetivo del equipo es levantar la infraestructura y lanzar el sistema idealmente sin errores post-lanzamiento. Es por ello que el área de testing es fundamental, ya que el esfuerzo requerido para las actividades de SQA representa el 50% del desarrollo total del proyecto. Con prioridad a las pruebas de stress y rendimiento de las videollamadas. Para ello, se debe contar con un Jefe del área SQA, QA Tester y personal de apoyo para las distintas actividades que aseguren la calidad. A continuación se describen las responsabilidades de cada uno:

1. Jefe del área SQA:

- a. Gestionar la unidad de SQA.
- b. Definir una estrategia para la implantación de las prácticas de SQA.
- c. Proveer el personal de SQA requerido para las actividades de desarrollo.
- d. Planificar las actividades de SQA.
- e. Informar a los niveles superiores sobre el estado del proceso y las actividades de SQA en los proyectos.
- f. Garantizar la capacitación del personal de SQA y los recursos necesarios para el desarrollo de sus actividades.
- g. Promover el mejoramiento continuo del proceso de SQA.
- h. Gestionar el desarrollo de herramientas para facilitar el proceso de SQA.
- i. Interactuar con todos los miembros de la organización.

2. QA Tester:

- a. Desarrollar las actividades de SQA planificadas para los proyectos.
- b. Participar en las revisiones, en la prueba, SCM, en el seguimiento, en la solución de problemas y en la elaboración de estándares.

- c. Auditar, monitorear, evaluar e informar sobre las actividades de desarrollo.
- d. Interactuar con los desarrolladores y la unidad de SCM.

3. **Personal de apoyo:**

- a. Desarrollar tareas administrativas para los QA testers.
- b. Completar los informes de SQA.
- c. Interactuar con las diferentes herramientas utilizadas para SQA durante un proyecto.
- d. Asistir al QA tester en las revisiones, la prueba y las auditorías.
- e. Interactuar con el personal de SCM según sea necesario.

4.2.2. Infraestructura.

Es necesario que el equipo de SQA que trabaje con las actividades de aseguramiento de calidad, cuente con una serie de herramientas con características específicas, para adecuarse al software y realizar sus labores de evaluación y auditoría eficazmente.

Dichas herramientas serán:

1. **Notebooks:**

Cada QA tester debe tener un notebook y una cantidad mínima de 3 a 4 personas para realizar las pruebas. Cabe mencionar que el sistema operativo del notebook debe ser Fedora 21, con la instalación del framework Stack MEAN, Apache versión 2.4.16 y la librería mocha.js para las pruebas. Como accesorios, debe contar con una cámara (Webcam) y micrófono incluidos para las videollamadas.

2. **Servidor:**

El servidor necesario debe soportar una gran cantidad de usuarios concurrente que ocupen la plataforma, y por lo tanto debe ser capaz de procesar muchas consultas en un tiempo aceptable, eso se ve reflejado en la cantidad RAM y nucleos que tenga para temas de trabajo en paralelo, debe soportar al menos Fedora 21 Server como SO y contar con una gran capacidad de almacenamiento para los datos del usuario.

3. **Lugar**

Físico:

Es necesario contar con un lugar físico que tenga electricidad e

iluminación para conectar los notebooks. Además se debe contar con mesones (de preferencia pegados al muro) y sillas para que puedan sentarse. También debe contar con una instalación de internet que tenga una velocidad de bajada de 10MB para automatizar las pruebas de acuerdo a la cantidad de QA tester que haya y poder aplicar las correcciones post-prueba. Finalmente debe contar con una velocidad de 5MB de subida para las videollamadas.

4.2.3. Actividades.

Las actividades que realiza el grupo de SQA tienen un solo objetivo, que es la realización de las pruebas para registrar las mediciones de los atributos de calidad acordados en el Documento “Especificación de Requisitos”. Idealmente se busca que cada tarea de los QA tester’s se haga en paralelo con el proceso de desarrollo de la versión del software. De esta manera a medida que se implementan las funciones y se hacen las revisiones de éstas para las pruebas, también se miden los atributos para mantener un control de ellos, evidenciando que hayan iteraciones del sistema e incrementación de funciones. Para ello se realizarán las siguientes evaluaciones [9].

1. ***Evaluación de la selección de los entregables de trabajo:***
El plan de proyecto identifica los productos de trabajo que deben ser desarrollados y evaluados, incluyendo los estándares y guías aplicables a su desarrollo. SQA debe asistir al jefe de proyectos en la selección de los estándares y guías aplicables a cada entregable.
2. ***Evaluación de las herramientas:***
SQA debe evaluar la selección de las herramientas existentes y adquiridas para el desarrollo. Éstas deben ser evaluadas según su funcionalidad, disponibilidad y facilidad de operación.
3. ***Evaluación de la planificación y el monitoreo del proyecto:***
Durante la planificación, SQA debe apoyar la identificación apropiada de guías y estándares aplicables a los entregables del proyecto y responsabilizarse de la elaboración del plan de SQA. Posteriormente, debe supervisar el cumplimiento del plan de proyecto.
4. ***Evaluación de la especificación de requerimientos:***
SQA debe:

- a. Comprobar la adherencia de los entregables a los estándares definidos en el plan de proyecto.
- b. Verificar la adherencia del proceso de especificación de requerimientos a los procedimientos definidos en el plan de proyecto.
- c. Garantizar que se revisaron adecuadamente los entregables (especificación del sistema y de requerimientos) de la fase de especificación de requerimientos.
- d. Asegurar la incorporación de los resultados de las revisiones en los entregables de esta fase.
- e. 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.

5. **Evaluación del diseño:**

SQA es responsable de:

- a. Comprobar la adherencia de los entregables a los estándares definidos en el plan de proyecto.
- b. Verificar la adherencia del proceso de diseño a los procedimientos definidos en el plan de proyecto.
- c. 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 diseño.
- d. Asegurar la incorporación de los resultados de las revisiones en los entregables de esta fase.

6. **Evaluación de la implementación y de la prueba de unidad:**

SQA debe:

- a. Garantizar que el proceso de codificación, las revisiones asociadas y la prueba de unidad sean conducidos de acuerdo a los estándares y procedimientos establecidos en el plan de proyecto.
- b. Asegurar la incorporación de los resultados de las revisiones en los entregables de esta fase.
- c. Verificar la implementación de las acciones correctivas derivadas de la prueba de unidad.

- d. Comprobar la utilización de la especificación de procedimientos y casos de prueba durante la prueba de unidad.
- e. Corroborar la documentación del código y de los resultados de la prueba de unidad.

7. Evaluación de la integración y prueba:

SQA es responsable de:

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

8. Evaluación del producto antes de su liberación:

SQA debe evaluar las actividades de preparación del producto final y su documentación para la entrega al cliente, para lo cual debe participar de la auditoría funcional y física. Además deberá entregar las métricas finales de los atributos de calidad definidos.

9. Evaluación del proceso de revisión:

SQA debe verificar que todo producto que se encuentre listo sea revisado y que las acciones correctivas identificadas durante la revisión sean implementadas.

10. Evaluación de las acciones correctivas:

SQA debe analizar los problemas detectados para determinar sus causas, impactos y frecuencia de ocurrencia, además para establecer acciones preventivas. SQA es responsable también de monitorear la adecuada implementación de las acciones correctivas derivadas de estos

problemas.

11. Evaluación del proceso de SCM:

SQA debe:

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

12. Verificar la implementación de los procesos:

SQA debe corroborar la adherencia de todos los procesos a los estándares y procedimientos definidos en el plan de proyecto.

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.

4.2.4. Responsabilidades.

La autoridad de SQA deriva del gerente técnico de la “**Toc&Talk!**” y su principal obligación es monitorear las actividades del proceso de desarrollo y revisar la adherencia de los productos de trabajo a los estándares, procedimientos y al plan del proyecto. Los resultados de este seguimiento deben ser informados al jefe de proyectos y, según sea aplicable, al Gerente Técnico.

En la siguiente tabla se adjunta una matriz de responsabilidades sobre las actividades de SQA y los interesados en cada actividad [9].

Actividad	Gerente Técnico	Jefe Proyecto	SQA	SCM	Analista	Diseñador	Desarrollador	QA Tester
Evaluación de la selección los entregables 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	O	X
Evaluación del diseño	X	X	X		X	X	O	X
Evaluación de la implementación y de la prueba de unidad			X			X	X	X
Evaluación de la integración y prueba			X			X	O	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				

5. Análisis de Riesgos.

5.1. Identificación de Riesgos.

Todo proyecto tiene riesgos que lo afectan de distintas maneras, algunas pueden ser perjudiciales para los recursos que tiene y otras pueden convertir al proyecto en algo imposible de llevarlo a cabo. Es por ello que es necesario analizar cada factor negativo que pueda hacer daño, estimando de acuerdo a la experiencia del equipo la probabilidad de que aparezca y el impacto que puede provocar. En base

a esto, se crea un plan de mitigación para disminuir la probabilidad y/o el impacto del riesgo. Entre los posibles riesgos de “Toc&Talk!” se encuentra(n):

Identificador	Descripción
R1	Que la implementación de la videollamada en el sistema no sea efectiva para tener una conversación de mínimo 30 minutos.
R2	Que la infraestructura del servidor no tenga la capacidad necesaria para guardar la información de forma segura de 1000 usuarios.
R3	Que el servidor entregue un rendimiento de tiempo de respuesta más allá de los 10 segundos.
R4	Que el servidor no pueda soportar a lo mínimo 1000 usuarios concurrentes.
R5	Que el algoritmo de búsqueda de personas no entregue un resultado para el usuario en menos de 10 segundos.
R6	Que no exista una cantidad mínima de usuarios de distintas lenguas para hacer la videollamada entre dos personas y así generar la conversación en el idioma que se busca aprender.
R7	Que los temas legales asociados a las cuentas de los usuarios y la privacidad de los datos, no sean aceptados antes del lanzamiento del producto.
R8	Que el diseño de la plataforma no parezca atractivo para el usuario (el color y/o la forma).
R9	Que las interfaces usuarias no sean intuitivas.
R10	Que exista obsolescencia pre parto, donde ya haya sido creada la plataforma.

R11	Que no exista suficiente difusión para atraer a más usuarios.
R12	Que el plazo del proyecto (1 año) no se cumpla por la mala estimación de los requisitos.
R13	Que aparezca una cantidad necesaria de bugs en el sistema para que se suspenda la operación del software.
R14	Que el presupuesto del proyecto no alcance para dejar el sistema en operación.
R15	Que los conocimientos sobre Javascript estén sobreestimados y se haga complejo desarrollarlo en ese lenguaje.
R16	Que existan comportamientos “extraños” de usuarios en las conversaciones que baje la cantidad de uso de la plataforma.

De entre todos los riesgos presentados, se enmarcan las que puedan afectar a los atributos de calidad definidos para el sistema, y por lo tanto las que se trabajaran durante el desarrollo de este documento.

5.2. Clasificación de Riesgos.

Ahora bien, es necesario clasificar cada riesgo enmarcado para descubrir la naturaleza de éste y de esa manera ordenarlos bajo el nombre “Tipos de Riesgo”. Por lo tanto, la clasificación queda de la siguiente forma [11]:

1. **Riesgos** **del** **Proyecto:**
Identifica los problemas potenciales de presupuesto, calendario, personal, recursos, etc.

Identificador	Descripción
R2	Que la infraestructura del servidor no tenga la capacidad necesaria para guardar la información de forma segura de 1000 usuarios.
R3	Que el servidor entregue un rendimiento de tiempo de respuesta más allá de los 10 segundos.
R4	Que el servidor no pueda soportar a lo mínimo 1000 usuarios concurrentes.
R6	Que no exista una cantidad mínima de usuarios de distintas lenguas para hacer la videollamada entre dos personas y así generar la conversación en el idioma que se busca aprender.

2. **Riesgos**

Técnicos:

Identificar posibles problemas de diseño, implementación, interfaz, verificación y mantenimiento.

Identificador	Descripción
R1	Que la implementación de la videollamada en el sistema no sea efectiva para tener una conversación de mínimo 30 min.
R8	Que el diseño de la plataforma no parezca atractivo para el usuario (el color y/o la forma).
R9	Que las interfaces usuarias no sean intuitivas.

3. **Riesgos**

de

Negocios:

Amenaza la viabilidad del proyecto, sin importar la naturaleza que tenga.

Identificador	Descripción
R10	Que exista obsolescencia pre parto, donde ya haya sido creada la plataforma.

5.3. Priorización de Riesgos.

Tras haber definido la clasificación de los riesgos para saber la naturaleza de cada uno; es necesario saber la prioridad que tienen. Para ello se consideran dos atributos:

1. **Probabilidad de que aparezca.**

- a. 3: Alta.
- b. 2: Media.
- c. 1: Baja.

2. **Impacto negativo que provoca.**

- a. 3: Alta.
- b. 2: Media.
- c. 1: Baja

Por ello, para darle un valor de orden desde los riesgos más urgentes en mitigar hasta los riesgos menores o indiferentes, se calcula la prioridad de cada riesgo de la siguiente forma:

$$\text{Probabilidad} * \text{Impacto}$$

Así, se obtiene la siguiente tabla:

Identificador	Probabilidad	Impacto	Prioridad(Probabilidad * Impacto)
R1	3	3	9
R4	2	2	4
R6	2	2	4
R8	1	3	3
R9	1	3	3
R2	1	3	3
R3	1	2	2
R10	1	1	1

Tras los resultados de la tabla anterior se concluye que los riesgos se deben mitigar de acuerdo a la prioridad y urgencia que tengan; vale decir que a mayor prioridad, mayor urgencia. Para ello se confecciona para cada riesgo, un “Plan de mitigación” el cual corresponde a la estrategia que se debe seguir para disminuir uno de los atributos (o idealmente los dos) y de esa forma se considera exitoso cuando en el proceso de desarrollo se aplica este plan y finalmente no se materializa ningún problema. En el caso de que la estrategia de mitigación sea inadecuada y se materialice el riesgo, existe otro plan complementario de la mitigación que es el “Plan de Contingencia” el cual corresponde a una nueva estrategia que provoca que el riesgo materializado no provoque la inviabilidad en el proyecto.

- 5.4. Control de Riesgos.
- Como se dijo anteriormente, cada riesgo tiene asociado un “Plan de Mitigación”, por ende para los primeros 6 riesgos prioritarios se adjunta sus estrategias respectivas (Ver Anexos punto 4).

6. Herramientas Técnicas y Revisiones.

6.1. Técnicas.

Para el desarrollo de la plataforma “Toc&Talk!”, se implementan las funciones en código JavaScript, usando el framework “Stack MEAN” el cual consiste en un conjunto de subsistema de software para el desarrollo de aplicaciones, como también en páginas web dinámicas. Para el área de SQA, se trabaja con dos framework que trabajan las pruebas y la automatización de ellas, estas herramientas consisten en:

1. **JasmineJS**, es un framework de testing de código abierto para Javascript, cuya finalidad es trabajar las pruebas unitarias [12].
2. **RedwoodHQ**, es un framework de automatización de pruebas de código abierto que trabaja para cualquier herramienta de que confecciona pruebas, permite hacer pruebas con datos automáticos y entregar resultados rápidos para evaluarlos [19].

Se eligen estas herramientas no sólo por su cualidad de “con un par de líneas puedes crear mucho” en comparación con otras plataformas; sino también por su rapidez de desempeño frente a múltiples consultas al servidor. Si bien, su curva de aprendizaje es medianamente alta, vale la pena estudiarlas por las grandes recomendaciones que tienen en relación al trabajar con “Stack MEAN”.

6.2. Herramientas para el desarrollo y puesta en marcha.

Se enumeran las especificaciones de cada herramienta a utilizar en la puesta de marcha del sistema.

- ❖ **Arquitectura de la Plataforma: Cloud Server (DonWeb).**
- ❖ **Servidor:**
 - Fedora 21 Server.
 - Soporte para MongoDB.
 - Soporte para NodeJS.
- ❖ **Lenguaje de Programación:**
 - Javascript.
- ❖ **Framework a utilizar:**
 - Stack MEAN.
- ❖ **Protocolos:**
 - WebRTC.
- ❖ **Librerías JS:**
 - Socket.io
 - PeerJS.
 - JasmineJS.
- ❖ **Lenguaje de Modelado:**
 - UML.
- ❖ **Control de versión:**
 - Git (Github).
- ❖ **IssueTracker:**
 - Github.
- ❖ **Medio para reuniones:**
 - Google Hangouts.
- ❖ **Plataformas QA:**
 - RedWoodHQ.

6.3. Actividades y Revisiones.

Tras haber especificado cada herramienta a utilizar para desarrollar el producto “Toc&Talk!”, el siguiente paso es comentar las actividades que se realizarán para

mejorar la calidad del sistema en el área de Testing (SQA), utilizando las herramientas y técnicas descritas anteriormente.

6.3.1. Revisiones e Inspecciones.

Las revisiones buscan descubrir de forma temprana, los defectos en los diferentes tipos de entregables que tiene cada proceso del proyecto. Su objetivo es actuar como filtro para mejorar tanto la documentación, como el diseño y la codificación del proyecto. Gracias a ello, habrá un gran impacto en la disminución de los costos de testing y mantención. Otras ventajas que tiene este ámbito es [15]:

- ❖ Detectar errores en la lógica, función e implementación.
- ❖ Verificar satisfacción de requerimientos.
- ❖ Asegurar cumplimiento de estándares.
- ❖ Fomentar uniformidad.
- ❖ Hacer proyectos más manejables.

Ahora bien, el equipo ha decidido basar las revisiones en la metodología de las “Inspecciones de Fagan”, el cual consiste en un método de revisiones agrupadas que previenen y reducen los diferentes tipos de defectos y por ende costos. Este proceso consta de 6 etapas:

1. **Planeación:**

Cuando el desarrollador completa su entregable de trabajo, se forma un grupo de inspección y se designa un moderador. El moderador asegura que el entregable satisfaga el criterio de inspección. Se le asignan diferentes roles a las personas que integran el grupo de inspección, así como la planificación de tiempos y recursos necesarios.

2. **Overview:**

Si los inspectores no están familiarizados con el desarrollo del proyecto, una vista general es necesaria en este momento. Este es un paso opcional, pero no menos importante ya que en esta etapa se dará al grupo de inspección un "background"(conocimiento general) y el contexto a cubrir por las inspecciones.

3. **Preparación:**

Los inspectores se preparan individualmente para la evaluación en la reunión, estudiando los entregables de trabajo y el material

relacionado. Aquí es aconsejable la utilización de listas de chequeos para ayudar a encontrar defectos comunes. El tiempo que pueda llevar esta etapa va a depender de cuán familiarizado esté el inspector con el trabajo que debe analizar.

4. **Inspección:**

En esta etapa los inspectores se reúnen para una puesta en común de su trabajo individual. El moderador deberá asegurarse que todos los inspectores estén suficientemente preparados. La persona designada como lector presenta el entregable de trabajo, interpretando o parafraseando el texto, mientras que cada participante observa en busca de defectos. Es recomendable que este examen no dure más de 2 horas ya que la atención en busca de defectos va disminuyendo con el tiempo. Al terminar con la reunión, el grupo determina si el producto es aceptado o debe ser retrabajado para una posterior inspección.

5. **Retrabajo:**

El autor corrige todos los defectos encontrados por los inspectores.

6. **Seguimiento:**

El moderador chequea las correcciones del autor. Si el moderador está satisfecho, la inspección está formalmente completa y el "producto de trabajo" es puesto bajo el control de configuración.

Como se mencionó anteriormente, cada integrante que trabaja en las revisiones, tiene un rol dentro de él que se explican a continuación:

1. **Moderador:**

El moderador es quien facilita todo el proceso de revisión, responsable de la dirección que tomará, y por lo tanto, un pilar fundamental. Algunas de sus responsabilidades son:

- a. Liderar, gestionar y coordinar el proceso de revisión.
- b. Guiarse por las pautas provistas en la checklist de revisión.
- c. Trabajar con el jefe de proyectos en la selección de los participantes, informarle sobre los recursos requeridos y asegurar que todos los participantes reciban la capacitación adecuada.
- d. Determinar el tamaño del producto de trabajo por inspeccionar.
- e. Determinar la necesidad de una reunión de orientación.
- f. Participar de la reunión de orientación.
- g. Preparar la inspección.

- h. Coordinar con el presentador las prioridades de la inspección.
- i. Planificar la inspección adecuadamente.

2. **Autor:**

Representa el desarrollador del producto de trabajo. Es quien prepara los entregables de trabajo para la revisión, de acoger y de realizar las correcciones de los defectos encontrados durante la revisión. Participa en todo el proceso de la revisión a excepción de de la preparación. Algunas de sus responsabilidades son:

- a. Preparar el producto de trabajo para el proceso de revisión y verificar su completitud de los datos.
- b. Guiarse por las pautas provistas en la checklist de revisión.
- c. Recomendar participantes para la inspección.
- d. Apoyar al moderador en la estimación del tamaño del producto.
- e. Entregar una explicación detallada sobre el producto y su documentación durante la reunión de orientación.
- f. Corregir los defectos detectados.

3. **Inspector:**

Es quien examina el producto de trabajo en la revisión. Cada inspector debe ser capaz de identificar y detectar errores. Son los responsables de hacer recomendaciones que pueden ser utilizadas para definir acciones correctivas y realizar mejoras en la etapa de revisión. Participa hasta la etapa de Inspección.

Algunas de sus responsabilidades son:

- a. Comprender el rol asignado. Participar en la reunión de orientación de ser necesario y/o consultar al moderador.
- b. Guiarse por las pautas provistas en la checklist de revisión.
- c. Examinar el producto de trabajo de acuerdo a las instrucciones provistas por el moderador.
- d. Identificar los errores de forma (léxico, puntuación semántica y gramática) durante la inspección.
- e. Completar el informe de revisión según las instrucciones entregadas por el moderador.
- f. Participar en la inspección según las instrucciones dadas por el moderador.
- g. Exponer posibles mejoras al proceso de revisión y las lecciones aprendidas durante este proceso.

4. **Presentador:**

Es el relator del producto durante la inspección. Apoya al moderador en el control de la reunión de inspección y es por ello que debe mantener la atención de los participantes en las áreas más relevantes del producto. El presentador participa en la etapa de preparación e inspección y puede hacer de inspector si fuera necesario.

Algunas de sus responsabilidades son:

- a. Coordinar la reunión de inspección en conjunto con el moderador.
- b. Guiarse por las pautas provistas en la checklist de revisión.
- c. Preparar con antelación la presentación del producto.
- d. Guiar la inspección del producto. El presentador debe centrar la atención de los inspectores en las secciones más relevantes del producto.
- e. Asegurar la participación equitativa de los miembros de la inspección.
- f. Participar como inspector en igualdad de condiciones que sus pares.
- g. Exponer posibles mejoras al proceso de revisión y las lecciones aprendidas durante este proceso.

5. **Secretario:**

Es quien lleva registro de las reuniones de inspección realizadas. Responsable de completar la documentación sobre los defectos y su clasificación, las acciones a asignarse y las lecciones aprendidas. El secretario participa en la etapa de inspección y puede hacer de inspector si es necesario.

Algunas de sus responsabilidades son:

- a. Guiarse por las pautas provistas en la checklist.
- b. Registrar la información pertinente a cada defecto utilizando el Informe de Revisión y los templates correspondientes.
- c. Leer cada defecto y su clasificación al final de la inspección.
- d. Participar como inspector en igualdad de condiciones que sus pares.
- e. Anotar las acciones correctivas y los plazos respectivos.
- f. Exponer posibles mejoras al proceso de revisión y las lecciones aprendidas durante este proceso.

6.3.2. Actividades de QA

Para explicar cada etapa del procedimiento de revisión, es necesario saber los “criterios de entrada” y “criterios de salida”,

❖ **Criterios de Entrada:**

- Las entradas al procedimiento de revisión están disponibles.
- Todos los participantes comprenden los roles y responsabilidades asignadas.
- Todos los participantes han confirmado la recepción del paquete de revisión.
- Todos los participantes comprenden el producto de trabajo.

❖ **Criterios de Salida:**

- Los templates de revisión han sido completados y distribuidos.
- Todos los defectos no resueltos han sido incorporados al monitoreo del proceso.
- Cada participante completa satisfactoriamente la checklist de las tareas asignadas a la etapa de preparación.

Sabiendo esto, se explican con detalle los procedimientos de cada etapa [9].

Etapa 1: Planificación	
Objetivos:	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 Autor
<u>Actividades</u>	1. El moderador debe seleccionar los participantes y asignarles roles específicos. 2. El moderador y el autor determinan el tamaño del producto de trabajo que debe ser revisado. 3. El moderador determina los criterios que el producto debe satisfacer. 4. El moderador establece la necesidad de una reunión de orientación.

5. El moderador planifica la reunión de orientación y/o la inspección.
6. El moderador prepara y distribuye el paquete de revisión a los participantes.
7. El moderador y el autor registran el tiempo utilizado durante la fase de planificación.

Etapa 2: Overview

Objetivos: El objetivo de esta etapa es reunir a los participantes e instruirlos sobre el producto de trabajo y reconocer el proceso de revisión.

Participantes: Moderador, Autor y cualquier otro participante que requiera información sobre el entregable de trabajo o proceso de revisión.

Actividades

1. El autor describe el entregable de trabajo sujeto a inspección.
2. Se revisa la asignación de roles y el plan para la inspección.
3. El moderador responde a cualquier consulta.

Etapa 3: Preparación

Objetivos: El objetivo de esta etapa es que cada miembro evalúe el entregable de trabajo para detectar defectos, los clasifique y los documente.

Participantes: Moderador, Presentador y cualquier participante bajo el rol de Inspector.

Actividades

1. Los inspectores revisan el entregable de trabajo y registran los defectos.
2. Cada defecto detectado es clasificado.
3. El presentador se prepara para exponer el producto durante la reunión de inspección.
4. Cada inspector entrega el informe de revisión.

Etapa 4: Inspección

Objetivos: El objetivo de esta etapa es que los participantes lleguen a un consenso respecto de los defectos que afectan al producto del trabajo, decidan si requiere de una revisión adicional, que discutan y documenten sobre las lecciones aprendidas del proceso.

Participantes: Moderador, Presentador, Autor, Inspector(es) y Secretario.

Actividades

1. El moderador enuncia los roles, el enfoque y las guías para la inspección.
2. El presentador describe el entregable de trabajo.
3. Se nombran y se discuten los defectos globales que afectan la completitud del entregable de trabajo.
4. Se nombra y discuten los defectos específicos por ser revisados y registrados.
5. Cada inspector entrega una lista con errores de formatos del autor.
6. El moderador asigna entre los participantes los defectos no resueltos.
7. Se determina la necesidad de tiempo adicional.
8. Se emplea la hora adicional para terminar las actividades restantes.
9. Se establece la necesidad de una revisión adicional.
10. El moderador solicita retroalimentación a los participantes en relación con la inspección.
11. El moderador solicita los informes de inspección.

Etapa 5: Retrabajo

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

Participantes: Autor, Moderador y opcionalmente, Jefe de proyecto u otros.

Actividades

1. Los participantes asignados estiman el tiempo requeridos para resolver los defectos no resueltos.
2. El autor corrige los defectos identificados en el entregable de trabajo.
3. Se resuelven los defectos no resueltos del entregable de trabajo.
4. El autor corrige los errores identificados en el entregable de trabajo.

Etapa 6: Seguimiento

Objetivos: El objetivo de esta etapa es asegurar que todos los defectos y los errores de forma, hayan sido corregidos; y que se haya dado solución a los defectos no resueltos.

Participantes: Moderador y Autor.

Actividades

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

6.4. Registro y generación de informes

Para el desarrollo de las actividades de SQA, es requerido de que haya la documentación de cada proceso hecho y los entregables (o productos) que se haya obtenido de cada etapa de desarrollo. Esto logrará de que cada producto sea revisado y corregido durante las actividades de calidad de la documentación, obteniendo un producto más refinado al momento de registrar el historial del proyecto (todos los documentos guardados para objetivos de información).

Para temas del “historial del proyecto”, es necesario dejar registro por cada avance, retraso o correcciones en la estructura en cada etapa del proceso de desarrollo, tanto en los procesos como los productos de ellos. Cada integrante debe tener claro que su interacción en el sistema, sea en documentos como

codificando, debe tener un registro, ya que mientras más información se maneje, mejor contexto del sistema se tiene.

6.5. Revisión por Checklist

Para especificar qué atributos de calidad queremos evaluar de las actividades de SQA, se elabora un checklist con los puntos específicos a pedir, para el cumplimiento de las prácticas y actividades de SQA durante el proceso de desarrollo de software. En base a esto, se obtendrá información fundamental para la creación de informes de revisión con las métricas pedidas, y de esa manera tener un contexto claro y legible de los requerimientos a cumplir en cada etapa.

Para este informe, se requiere revisar 3 áreas del desarrollo:

1. Revisión por actividades.
2. Revisión por Entregable de Trabajo.
3. Revisión por Equipo de revisión del proceso de software.

Por lo tanto los checklist para la revisión de cada área serían [9.1]:

6.5.1. Revisión por Actividades (Ver Anexos).

- 6.5.1.1. Checklist Planificación.
- 6.5.1.2. Checklist Análisis de Requisitos.
- 6.5.1.3. Checklist Diseño.
- 6.5.1.4. Checklist Implementación.
- 6.5.1.5. Checklist Integración y Pruebas.
- 6.5.1.6. Checklist Aceptación y Entrega.
- 6.5.1.7. Checklist Mantención.

6.5.2. Revisión por Entregable de Trabajo (Ver Anexos).

- 6.5.2.1 Checklist Plan de Proyecto.
- 6.5.2.2 Checklist Especificación de Requerimientos.
- 6.5.2.3 Checklist Diseño Preliminar.
- 6.5.2.4 Checklist Diseño Detallado.
- 6.5.2.5 Checklist Código.
- 6.5.2.6 Checklist Plan de Pruebas.
- 6.5.2.7 Checklist Procedimientos y casos de prueba.
- 6.5.2.8 Checklist Manual de Usuario.

6.5.3 Revisión por Equipo de Revisión (Ver Anexos).

- 6.5.3.1 Checklist Moderador.
- 6.5.3.2 Checklist Autor.
- 6.5.3.3 Checklist Inspector.
- 6.5.3.4 Checklist Presentador.
- 6.5.3.5 Checklist Secretario.

7. Pruebas

7.1 Estructura de las pruebas.

Una prueba es una actividad que evalúa los atributos y capacidad de un programa o sistema para determinar si cumple los resultados esperados. Esto implica verificar la satisfacción de los requerimientos (tanto funcionales como no funcionales) e identificar las diferencias que aparecen entre los resultados esperados y los obtenidos. Por ello, la prueba permitirá ejecutar un programa o sistema con la intención de encontrar defectos.

En el área de SQA, es muy importante realizar la mayor cantidad de pruebas que permitan los recursos que tenga el proyecto, ya que al detectar defectos en el código y mejorarlo, conlleva a que se minimice la cantidad de bugs (falla o comportamiento no adecuado) que puedan aparecer durante el periodo de “producción” del sistema.

Para cada prueba, se sigue un proceso que incluye cinco etapas [9.2]:

- 1. Planificación.**
- 2. Especificación**
- 3. Ejecución.**
- 4. Análisis de Resultados.**
- 5. Completación.**

En general, el proceso comienza identificando los tipos de pruebas a realizar, estimando los recursos y el esfuerzo requerido para ellas y programando las actividades a realizar. Posteriormente, para cada tipo de prueba se especifica un conjunto de casos de pruebas y los procedimientos respectivos. Sobre la base de la especificación, se ejecutan las pruebas y se registran los resultados, los cuales servirán para identificar defectos y sus posibles causas, así como también para

establecer acciones correctivas durante el análisis. Por último, se prepara el equipamiento y los casos de pruebas para usos posteriores, concluyendo la documentación de todo el proceso.

Para efecto de este proceso se consideran cuatro niveles de prueba [9.2]:

- 1. Unidad.**
- 2. Integración.**
- 3. Aceptación.**
- 4. Regresión (de ser necesario).**
- 5. Sistema.**

A continuación se describe las diferentes etapas del proceso de prueba:

7.1.1 Planificación.

La planificación de las actividades que se ejecutaran en las prueba comienza junto a la planificación. Durante ese periodo se identifican los esfuerzos (actividades) y recursos necesarios para la(s) prueba(s). No obstante, el desarrollo formal del plan de pruebas tiene lugar sólo después de la aprobación de la especificación de los requerimientos.

En base a la especificación de requerimientos y del plan de proyecto, el jefe del proyecto, la unidad responsable de prueba y su líder serán responsables de desarrollar el plan de pruebas. Para eso se deben realizar las siguientes actividades:

- ❖ Sobre la base de los requerimientos, definir los niveles, tipos y metodologías de prueba que serán utilizadas.
- ❖ Especificar los estándares y guías involucradas.
- ❖ Identificar el ambiente en que se llevarán a cabo las pruebas.
- ❖ Establecer los criterios de aprobación/reprobación para el sistema de software.
- ❖ Estimar el esfuerzo y los recursos necesarios.
- ❖ Asignar un responsable del proceso de prueba para el proyecto.
- ❖ Trazar los requerimientos con los tipos de pruebas para verificar su completitud.
- ❖ Programar los diferentes tipos de prueba, incluyendo tiempo para las correcciones y las pruebas de estas últimas.
- ❖ Desarrollar el plan según el template correspondiente.
- ❖ Terminado el plan de pruebas, someterlo a una revisión.
- ❖ Corregir las discrepancias e incorporar las recomendaciones.
- ❖ Aprobar, distribuir y poner bajo control de configuración del software, el plan

de pruebas.

7.1.2 Especificación.

En base al plan, se describen los diferentes tipos de pruebas para los diferentes casos; a nivel funcional (propósito) y a nivel detallado (procedimiento).

La descripción de los casos de pruebas involucra las siguientes actividades:

- ❖ Identificar el conjunto de casos de pruebas por cada tipo de prueba, especificando el propósito de cada caso.
- ❖ Definir para cada caso de prueba, el conjunto de entradas requeridas para satisfacer el propósito de ésta.
- ❖ Definir para cada caso de prueba, las salidas esperadas.
- ❖ Especificar los métodos de inserción/extracción para el test.
- ❖ Identificar los tipos y los volúmenes de datos de entrada/salida.
- ❖ Determinar criterios de evaluación para el análisis de resultados de las pruebas. Definir rangos, capacidades y tiempos que permitan establecer su aprobación/reprobación.
- ❖ Trazar los requerimientos de los casos de prueba para verificar su completitud.
- ❖ Documentar la descripción funcional de los casos de prueba por tipo de prueba. Someter a revisión este documento, corregirlo, aprobarlo y ponerlo bajo control de configuración del software.

La descripción detallada describe:

- ❖ Seleccionar los métodos y las herramientas para la ejecución de los casos de pruebas.
- ❖ Definir el procedimiento para proveer las entradas a cada caso de prueba.
- ❖ Definir criterios de aprobación/reprobación para cada etapa de los casos de prueba.
- ❖ Agrupar los datos de entrada.
- ❖ Definir procedimientos para el análisis de los resultados de las pruebas y para comparar los resultados actuales con los esperados.
- ❖ Documentar la descripción detallada de los casos de prueba. Someter a revisión este documento, corregirlo, aprobarlo y ponerlo bajo control de configuración del software.

7.1.3 Ejecución.

Teniendo lista la especificaciones de cada prueba y una vez que la codificación haya terminado, se comienza la ejecución de las pruebas. Para cada tipo, se ejecutan los diferentes casos de pruebas predefinidos, registrando los resultados obtenidos y mediante los criterios de aceptación/validación (resultado esperado), se chequea si la prueba cumplió o no con el propósito creado. Luego, se indican las correcciones necesarias, las cuales serán llevadas a cabo por el equipo desarrollador, para luego ser probados nuevamente con la finalidad de verificar su correctitud y prevenir la introducción de nuevos defectos. Finalmente, una autoridad competente evalúa la pertinencia de dar por finalizadas las actividades correspondientes a dicho tipo de prueba, pasando a la siguiente.

Los niveles de pruebas serán ejecutados secuencialmente en el siguiente orden:

- ❖ Pruebas de unidad al finalizar la codificación.
- ❖ Prueba de integración en paralelo con la etapa de integración.
- ❖ Prueba de aceptación durante la etapa de aceptación y entrega.
- ❖ Prueba de regresión, en caso de existir una versión anterior del sistema.
- ❖ Prueba del sistema previo a la entrega y operación del sistema.

La primera de las pruebas es responsabilidad del equipo desarrollador y las demás de la unidad responsable de la prueba (QA tester).

7.1.4 Análisis de Resultados.

En base a los resultados obtenidos de las pruebas, se realiza una identificación de los defectos y de sus posibles causas. Esto llevará a establecer responsabilidades asociadas al proceso de desarrollo y no a responsabilidades individuales.

Es fundamental determinar en qué fase el defecto fue encontrado y la causa (no-adherencia a los estándares y procedimientos, especificación imprecisa, etc). A partir de esta información, se deben proponer acciones para la corrección a las unidades involucradas, lo que permitirá que se minimice la cantidad de posibles defectos detectados en cada prueba. Esta acción permitirá bajar la probabilidad de que aparezca un defecto que no se haya contabilizado, entregando así la seguridad de cumplir con los estándares pedidos y la calidad del sistema (en su funcionalidad y en sus atributos).

7.1.5 Completación.

Para la finalización del proceso de prueba, es necesario que los encargados (QA tester) preparen el equipamiento y los casos de pruebas para su uso posterior; completando la documentación de este proceso. De esa manera, una entidad evaluará si se da por finalizado el proceso para proseguir al próximo tipo de prueba.

7.2 Pruebas a considerar.

Para “Toc&Talk!” es importante realizar algunas pruebas previas antes de cada caso de prueba (sea unitaria, de rendimiento, etc) que se ejecutara en el sistema, ya que se debe cumplir con un flujo de entrada y salida de usuarios de tal manera que los datos usados por ellos, estén completamente seguro y de que no existirá una filtración de ellos, es por eso que se hacen las siguientes pruebas:

7.2.1 Protocolos de acceso.

Para entrar a la plataforma mediante un navegador web y/o móvil (en caso de smartphones), se podrá acceder mediante el protocolo HTTPS (Hypertext Transfer Protocol Secure), y mediante el puerto 443 habilitado.

Para temas de acceder al servidor desde un computador remoto, será solo por el protocolo SSH (Secure Shell) a través del protocolo 22 habilitado.

7.2.2 Almacenamiento de datos privados por encriptación.

Para funcionalidades donde se deben guardar datos privados en el sistema, estas se harán mediante una encriptación MD5 (Message-Digest Algorithm 5) durante el proceso del registro de datos en la base de datos del servidor.

7.2.3 Tiempo de acceso de la funcionalidad.

Se debe tomar el tiempo desde que el navegador web pide los recursos de la página a acceder por la funcionalidad, hasta que el servidor termine de mandar lo pedido. De esa forma, se obtendrá información para plantear posibilidades para optimizar el tiempo resultante.

7.2.4 Optimizar imágenes.

Se debe tomar en cuenta el menor tamaño posible para visualizar una imagen, pues dependiendo del tamaño que tenga, será el tiempo que demore el servidor en enviarlo al navegador. Para eso se chequea que las imágenes este comprimida al momento de ser enviada, y de esa forma se ahorraran varios costos tanto de bytes de envio como del uso de datos limitados por parte del navegador (y mas importante de quien utilice datos móviles).

7.2.5 Mantener orden del codigo.

Por disciplina del equipo, como también para tener conocimiento de en que lugar van cada parte del codigo en el desarrollo de funcionalidades como de las vistas, se debe tener una estructura clara y especifica donde haya separación de las librerías CSS, JS, y otras más con los codificación de forma indentada. Se debe revisar que se cumpla la estructura definida por el equipo.

7.3 Casos de Pruebas.

Una vez terminado la estructura que tomara cada prueba y las pruebas “previas” que deben hacerse antes de ejecutar las pruebas para cada nivel, se comienza a elaborar los casos de pruebas de los requerimientos no funcionales cuyo objetivo es monitorear y controlar que las salidas esperadas de cada prueba sea en función al criterio de aceptación. Si bien, el equipo de SQA esta encargado de hacer tantas pruebas como tiempo y recursos dispone, para este informe, se detallara un caso de prueba para cada atributo de calidad.

7.3.1 Pruebas por Atributo de Calidad.

7.3.1.1 Usabilidad.

- **Caso de Prueba:** Tiempo de Aprendizaje.
- **Descripción:**
Tiempo en que un usuario le toma aprender a utilizar las funciones más importantes del sistema.

- **Condiciones Iniciales:**
 - Pruebas unitarias y de integración realizadas, y posterior corrección de los defectos encontrados.
 - Cantidad de 15 usuarios para usar el sistema.
 - Servidor corriendo en estado “development” (no en producción).
- **Entradas:**
 - Usuario “x” se registra con username “userX” (X indica el numero de usuario) y password “pass1234”.
 - Idioma a practicar “Ingles”.
- **Salidas Esperadas:**
 - Tomando el tiempo que demora hacer todas las entradas correctamente, debe registrarse una duración rápida y satisfactoria para el cliente.
- **Criterios de Evaluación de resultados:**
 - Tiempo máximo registrado para entender correctamente las funcionalidades importantes (Registrar, seleccionar idioma y buscar usuario), será de 5 minutos aproximadamente.
- **Procedimiento:**
 - Usuario “x” crea una cuenta con los datos entregados.
 - Usuario “x” accede a su perfil.
 - Usuario “x” elige idioma a hablar.
 - Usuario “x” da click en el logo de la plataforma para comenzar a buscar un match con otro usuario e ingresar a la videollamada.
 - Usuario “x” termina videollamada y vuelve a menú principal.

7.3.1.2 Adaptabilidad.

- **Caso de Prueba:** Cantidad aceptable de usuarios concurrentes.
- **Descripción:**
Cuota de usuarios necesarios que ocupen el sistema para que hayan una búsqueda rápida de usuarios para conversar, y no haya sobrecarga del servidor.
- **Condiciones Iniciales:**
 - Funcionalidad de la videollamada probada e

- integrada correctamente.
 - Tener instalado un programa para simular usuarios que ocupen la videollamada (En este caso, Apache Server).
 - Cuentas de usuarios creadas para pruebas.
 - Servidor corriendo en estado “development” (no en producción).
- **Entradas:**
 - Simular 1000 usuarios concurrentes.
- **Salidas Esperadas:**
 - Una gran cantidad de usuarios conectados al mismo tiempo.
 - Una gran cantidad de videollamadas realizadas con éxito.
 - Pagina web funcionando perfectamente.
- **Criterios de Evaluación de resultados:**
 - 1000 usuarios conectados y 500 videollamadas implementadas exitosamente.
 - Servidor funcionando sin caída.
- **Procedimiento:**
 - Usuarios “x” e “y” comienzan “búsqueda de usuario”.
 - Sistema le provee una videollamada para los dos.
 - Seguir hasta que todos los usuarios estén conectados.
 - Obtener tiempo de cada petición y la carga realizada (en paquetes de datos).

7.3.1.3 Seguridad.

- **Caso de Prueba:** Seguridad de registro de datos.
- **Descripción:**

Cuando un usuario registra sus datos personales con la finalidad de crear una cuenta, se establece la confianza de que tales datos no podrán ser accedidos por otra persona además del usuario, y se encriptan el password en caso de que sea así.
- **Condiciones Iniciales:**
 - Algoritmo de encriptación probado e integrado correctamente.
 - Funcionalidad de registro de usuario probado e

integrado correctamente.

➤ **Entradas:**

- 5 usuarios de diferentes edades para pruebas.
- Usuario “x” se registra con username “userX” (X indica el numero de usuario) y password “passYYYY” (YYYY representa el año de nacimiento).

➤ **Salidas Esperadas:**

- Finalizado el crear un usuario, la url de la página no debe contener datos personales.
- Mensaje exitoso de usuario creado.
- En la BD, password encriptado.

➤ **Criterios de Evaluación de resultados:**

- Password encriptado, de acuerdo a la clave privada definida en el algoritmo MD5.
- Metodo HTTP de respuesta para la creación de usuario debe ser POST.
- Usando el algoritmo y su clave, desencriptar el password y chequear que el mismo que puso el usuario de prueba.

➤ **Procedimiento:**

- Usuario “x” registra sus datos personales en la sección de “Registro Usuario”.
- Sistema toma los valores entregados e encripta el password.
- Se entrega un mensaje exitoso de registro.

7.3.1.4 Modularidad.

➤ **Caso de Prueba:** Funciones en modulos.

➤ **Descripción:**

Para tener pruebas unitarias mas rápida de probar y corregir, es necesario modularizar todas las funciones del sistema en diferentes archivos, con un nombre fácil de distinguir, y mantener el orden con el fin de saber más rápido donde puede estar un posible error, al debuggear el sistema.

➤ **Condiciones Iniciales:**

- Librería para leer archivos.
- Funcionalidades probadas e integradas.

➤ **Entradas:**

- Archivos de funciones.
- **Salidas Esperadas:**
 - Resultado de cantidad de funciones ordenadas y organizadas por un nombre que las representa.
- **Criterios de Evaluación de resultados:**
 - Deben existir 5 funciones en un modulo (archivo de funciones), para fomentar la modularidad.
- **Procedimiento:**
 - Prueba abre y lee un archivo “x”.
 - Prueba entrega cantidad de funciones encontradas.

7.3.1.5 Disponibilidad.

- **Caso de Prueba:** Uptime del sistema.
- **Descripción:**

Es el porcentaje donde el servidor esta activo durante un tiempo determinado, para demostrar que el servidor es confiable y que no se caera de forma constante, es necesario implementar un servidor que se maneje en diferentes ambientes.
- **Condiciones Iniciales:**
 - Servidor NodeJS implementado.
 - Programa instalado para simular usuarios.
 - Cuentas de usuario reales, creadas.
- **Entradas:**
 - 100 usuarios usando videollamadas.
 - 200 usuarios simulados, haciendo una tarea en específico, en cada tramo de 30 segundos durante 1 hora.
- **Salidas Esperadas:**
 - Servidor en estado activo.
 - Cantidad de usuarios minimo concurrentes.
- **Criterios de Evaluación de resultados:**
 - Servidor activo por 1 hora.
 - 300 Usuarios concurrentes minimo.
- **Procedimiento:**
 - Usuarios reales “x” e “y” acceden al sistema y hacen click en el icono de “Comenzar”.
 - Sistema les provee una videollamada para hablar durante 1 hora.

- Repetir los puntos anteriores hasta que estén todos conectados.
- Usuarios simulados (200) acceden el sistema y hacen una tarea específica.
- Repetir proceso anterior cada 30 segundos, hasta cumplir una hora.

7.3.1.6 Recuperabilidad.

➤ **Caso de Prueba:** Recuperación de datos.

➤ **Descripción:**

Al momento de fallar el servidor y este quede no operativo o se reinicie, existiran servidores espejos para acceder a la página web y a los datos, mientras el servidor que fallo vuelva a estar activo.

➤ **Condiciones Iniciales:**

- Servidores espejos.
- Función de redireccionamiento de servidor implementado.
- Datos del usuario creado.

➤ **Entradas:**

- Comando para apagar un servidor.
- 1 usuario para probar.

➤ **Salidas Esperadas:**

- Pagina web muestra ligero retraso en cargar alguna funcionalidad del sistema.
- Luego del retraso, pagina web cargada normalmente con las variables de sesión del usuario logueado.

➤ **Criterios de Evaluación de resultados:**

- Retraso en cargar la plataforma y los datos con un servidor espejo activo, en no más de 20 segundos.

➤ **Procedimiento:**

- Usuario “x” accede a la página, luego de hacer login.
- Se apagar el servidor definido.
- Usuario “x” hace click en algún botón del sistema.
- Sistema muestra página web de un servidor espejo luego de un tiempo de demora.

7.3.1.7 Capacidad.

- **Caso de Prueba:** Tamaño del Almacenamiento.
- **Descripción:**

El servidor debe contar con un disco duro que guarde los datos obtenidos por los usuarios, sea de cuenta como de mensajes, y para eso debe contar con un gran tamaño de almacenamiento.
- **Condiciones Iniciales:**
 - Funcionalidades para registrar un usuario y escribir mensaje a los amigos, probadas e implementadas.
 - Servidor con un disco duro de 100GB.
- **Entradas:**
 - Datos de usuarios.
 - Mensajes enviados al chat.
- **Salidas Esperadas:**
 - Usuarios creados y mensajes almacenados en la base de datos del servidor.
- **Criterios de Evaluación de resultados:**
 - Cantidad de usuarios creados y mensajes almacenados, que lleguen a 100 GB. De esa manera se sabra cuanto usuarios y mensajes puede almacenar.
- **Procedimiento:**
 - Usuario “x” crea una cuenta.
 - Sistema valida los datos y los guarda en el servidor.
 - Usuario “x” escribe un mensaje a un amigo definido por el sistema.
 - Repetir punto anterior hasta tener 100 mensajes escritos.
 - Repetir desde el primer punto, hasta llegar a los 10GB de almacenamiento.

8. Conclusiones

Desde la existencia del ser humano, la capacidad para comunicarse ha sido de vital importancia, tanto para la sobrevivencia como para la transmisión de ideas y/o emociones. En base a esto se origina el idioma, como sistema de comunicación verbal y gestual, en donde individuos pertenecientes a una comunidad humana pueden transmitir ideas más claras, articuladas en el lenguaje de manera unificada. Desde entonces, podemos decir que hablar un idioma es una necesidad fundamental que con el paso de los años ha ido adquiriendo nuevas formas tanto de expresión como de aprendizaje.

Hoy en día vivimos en un mundo globalizado donde es necesario establecer relaciones con personas de otros lugares del mundo. Para ello necesitamos “entendernos”, es decir, necesitamos estar familiarizados con el idioma que habla la otra persona. Esto no es una tarea fácil, ya que la cultura, por su parte, cumple una función importante en la evolución de un idioma, en donde el aprendizaje no sólo se basa en el aprender el lenguaje involucrado, sino que también, el hablarlo.

Hablar una lengua distinta a la nativa, ha pasado de ser una motivación personal a una necesidad fundamental, sobretodo en ámbitos profesionales y académicos, convirtiéndose en una herramienta valiosa para comunicar a personas de distintas partes del mundo. Sin embargo, el cómo aprenderlos ha sido y sigue siendo un problema que nos involucra a todos.

Millones de personas dedican mucho tiempo intentando aprender un segundo idioma a través de métodos tradicionales, que si bien permiten adquirir conocimientos sobre el lenguaje asociado al idioma (gramática, sintaxis, etc), no son suficientes para lograr que la gente mantenga una conversación fluida. Permitiéndonos identificar que el problema está en la carencia de herramientas y espacios para poder desarrollar todas estas competencias lingüísticas orales, necesarias para llevar a cabo expresiones de ideas claras y concisas.

La situación actual, para poder practicar otro idioma, se basa principalmente en el desarrollo de competencias lectoras y gramaticales del mismo, existiendo, en el mercado, una carencia de espacios y servicios que permitan llevar a cabo el desarrollo del idioma hablado, involucrando competencias que van más allá de las ya mencionadas, apuntando a aquellas que son orales y auditivas. En Chile, la gente que se desea instruir en algún otro idioma distinto al inglés cuenta con menos servicios aún (o simplemente no los tienen), realidad que se repite en otros países con otros idiomas también, señalando, además, que de los pocos servicios existentes en este rubro, la mayoría de ellos poseen costos no accesibles por usuarios comunes y corrientes, ya sea, porque no pueden pagarlo o simplemente por no cumplir las condiciones que el medio de pago requiere, teniendo en cuenta de que dichos servicios no poseen su foco en el entrenamiento ni la práctica del idioma hablado como tal, ni poseen las herramientas efectivas para lograr un nivel de comunicación suficientemente bueno como para expresarse fluidamente ante otro hablante.

Lo anterior nos conduce a interrogantes aún mayores como: ¿cómo una persona que no posee los recursos económicos suficientes puede llegar a desenvolverse en un idioma distinto al nativo de manera efectiva?, ¿qué sucede con aquellas personas que poseen un dominio sobre un idioma diferente y desean practicarlo?. La respuesta para todas estas interrogantes se puede resumir en que no existe un servicio de bajo costo económico que brinde un espacio para poder comunicarse de forma hablada con otra persona que tiene su mismo interés, lo que nos presenta un problema que es claro: hay una necesidad que no está satisfecha.

De esta forma, nace la idea de “Toc&Talk!”, donde la solución para satisfacer la necesidad es crear una plataforma móvil de fácil acceso, en donde las personas de distintos lugares del mundo que quieran aprender a expresarse de forma hablada en algún otro idioma, puedan hacerlo.

Para conseguir esto, se penso en proveer un servicio virtual en donde se permitan desarrollar no sólo aquellas habilidades de comunicación y de expresión oral, sino que también, competencias lingüísticas tanto para hablar como para escuchar, en base a la conversación hablada entre los mismos usuarios que constituirán dicha plataforma.

Si bien el uso de videollamadas sería un mecanismo de comunicación eficaz, en una primera instancia se enfocaran en la comunicación por voz, ya que puede darse el caso en que usuarios no quieran hablar con otros a causa del rechazo físico, tal como sucede en plataformas como Chatroulette y Omegle, con la diferencia de que su enfoque es “conversar con extraños”. A pesar de esto se pretende implementar este mecanismo y a su vez buscar una manera de evitar la discriminación que puede llegar a existir entre usuarios; por ejemplo, dando la opción de activar la cámara luego de transcurrida una cantidad necesaria de minutos hablando, lo que motivaría, a su vez, a que las conversaciones sean más prolongadas y efectivas.

Para las plataformas que manejan una gran cantidad de usuarios, como OpenEnglish, entre otros, han tenido que pasar por la fase de “Aseguramiento de Calidad”, donde al mejorar necesariamente la calidad del sistema, mejoran la experiencia del usuario. Es por esa razón, que se hacen diferentes pruebas unitarias de las funciones del sistema para corregir la mayor cantidad de defectos encontrados en el sistema, como también realizar pruebas de los requisitos extra-funcionales, que son atributos que proporcionan el aspecto del diseño de las funcionalidades. Además al mejorar la codificación se puede lograr sostener de mejor manera una gran cantidad de usuarios que utilicen los recursos de la plataforma. Por otra parte, se estandarizan diseños de interfaces de calidad, ya que se ha convertido en un desafío en la actualidad el darle mayor atributo de “usabilidad” a través de las interfaces que provee el sistema y de como se ha hecho fundamental aplicar investigación exhaustiva de plataformas conocidas para confeccionar diseño de vistas mas adhoc con lo que atrae más a los usuarios. Y en base a esto, “reutilizar” dichas planillas de diseño para la construcción de las vistas futuras.

Ante lo mencionado, es el proposito de este “Plan de Calidad” de la plataforma

“Toc&Talk!”, pues si se busca obtener su versión comercial, es indispensable que cumpla con los siguientes puntos.

1. Garantizar el cumplimiento de los requisitos funcionales y extra-funcionales pedidos por el cliente.
2. Identificar y estandarizar procesos para mejorar la calidad del software.
3. Definir los productos de trabajos que tendrá cada etapa de construcción del sistema.
4. Definir los recursos y el personal a capacitar para implementar este plan.
5. Mitigar los riesgos que afectan al proyecto.
6. Confeccionar informes de revisiones de cada producto de trabajo y/o versión desarrollada del software.

Al aplicar las actividades, los roles descritos y las responsabilidades que tiene cada uno, ayudara considerablemente a darle un sello de calidad al sistema, cumpliendo atributos de calidad como Usabilidad, Seguridad, Disponibilidad, etc. De esta forma no solo se mejora la eficiencia y eficacia del sistema, sino que también aplicamos procesos de calidad que entregara información para mejorar dichos procesos para una nueva iteración en la fase de SQA.

Finalmente, en este proceso de la memoria se ha podido conocer los estándares de calidad que son necesarios aplicar en el software y como aplicar cada atributo de calidad de forma específica. Además de la cantidad de información que se debe recopilar de muchas temas de investigación para elaborar un plan de calidad adecuado para la plataforma que se busca crear, como también los roles y actividades que deben cumplir por cada producto a revisar. Es necesario dejar aparte el área de SQA, del área de desarrollo, ya que abarca demasiados temas que deben cubrirse, y satisfacerse si es que se quiere presentar un producto con altos estándares de calidad en el mercado globalizado de hoy, por lo que la separación de esas áreas ayuda a trabajar de manera paralela y obtener mejores resultados en menos tiempo.

Hoy en día, las herramientas y metodologías que se trabajan en el área de SQA, son cada vez mayores, por lo que es importante para marcar el éxito del proyecto, el estudiarlas en su mayor parte y aplicarlas, pues al mejorar la calidad del proceso de pruebas eso lleva a mejorar la calidad del producto.

9. Bibliografía

- [1] ¿Qué es la Usabilidad?, Guía Digital Beta.
<https://www.google.cl/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8#q=Usabilidad>
- [2] Servidores Espejo, bitEndian Enginyers Informatics.
<http://www.bitendian.com/es/servidores-espejo/>
- [3] Desarrollo iterativo e incremental, Proyectosagiles.org
<https://proyectosagiles.org/desarrollo-iterativo-incremental/>
- [4] El ciclo de vida de proyectos de consultoría de negocios, Project Management Institute.
<https://americalatina.pmi.org/latam/KnowledgeCenter/Articles/~media/993705D14AEE4DCFB2E60EF6CDDFDBE9.ashx>
- [5] Memoria de Informática UTFSM, Joaquín Alfonso Nuñez García.
Plan de calidad para “Huellas Universitarias”, Memoria para optar al título de Ingeniero de Ejecución en Informática, Junio 2015.
- [6] Plan de proyecto de Software, Wikiversidad.
https://es.wikiversity.org/wiki/Plan_de_proyecto_software
- [7] Guía de Usuario, Wikipedia.
https://es.wikipedia.org/wiki/Gu%C3%ADa_del_usuario
- [8] Atributos de Calidad en el Desarrollo de Software, inShare.
<https://es.slideshare.net/gustavocuen77/atributos-de-calidad-en-el-desarrollo-de-software-30272613>
- [9] Herramientas específicas para SQA y SCM del Prof. Visconti, Depto de Informática UTFSM.
<https://www.inf.utfsm.cl/~visconti/herramientas/>
- [9.1] “Checklist de SQA” (Herramientas específicas para SQA y SCM del Prof. Visconti)
- [10] Líder de Proyecto, Ecured.
https://www.ecured.cu/L%C3%ADder_de_proyecto
- [11] Tipos de Riesgos, Gestión de Proyectos Software.
<https://sites.google.com/site/gestiondeproyectossoftware/unidad-3-planificacion-de-proyecto/3-5-1-tipos-de-riesgos>
- [12] Jasmine JS, Wikipedia.
[https://en.wikipedia.org/wiki/Jasmine_\(JavaScript_testing_framework\)](https://en.wikipedia.org/wiki/Jasmine_(JavaScript_testing_framework))

- [13] TDD como metodología de diseño de software, Blog Tecnología del desarrollo.
<https://www.paradigmadigital.com/dev/tdd-como-metodologia-de-diseno-de-software/>
- [14] Capitulo 2. ¿Qué es el desarrollo Dirigido por Tests? (TDD), LibrosWeb.
http://librosweb.es/libro/tdd/capitulo_2.html
- [15] Diapositivas Profesor Visconti ISW año 2015, Depto de Informatica UTFSM
<https://drive.google.com/drive/folders/0B8fM2nLvoHEfbklqUDVOOFdmMms?usp=sharing>
- [16] Inspecciones de Fagan, Wikipedia
https://es.wikipedia.org/wiki/Inspecci%C3%B3n_de_Fagan
- [17] Inspecciones de Software, Monografías.
<http://www.monografias.com/trabajos6/isof/isof.shtml>
- [18] Software Engineering 9th Edition, Ian Sommerville.
https://edisciplinas.usp.br/pluginfile.php/2150022/mod_resource/content/1/1429431793.203Software%20Engineering%20by%20Sommerville.pdf
Version Traducida (7th Edition):
http://zeus.inf.ucv.cl/~bcrawford/AULA_ICI_3242/Ingenieria%20del%20Software%207ma.%20Ed.%20-%20Ian%20Sommerville.pdf
- [19] RedWoodHQ, PrimaTest.
<http://redwoodhq.com/>

Anexos

1. Revisión por Actividades.

1.1. Checklist Planificación.

Planificación		
1. Identificación		
<u>Proyecto:</u>	<u>Fecha de inicio:</u> <u> </u> / <u> </u> / <u> </u> <u>Fecha de término:</u> <u> </u> / <u> </u> / <u> </u>	
<u>Nombre:</u>		
<u>E-mail:</u>	<u>Fono:</u>	
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?		

1.2. Checklist Análisis de Requisitos.

Especificación de Requerimientos		
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?		
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?		

1.3. Checklist Diseño.

Diseño		
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 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 el diseño preliminar?		
¿Se evaluó el diseño detallado?		
¿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.		

1.4. Checklist Implementación.

Implementación		
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		

1.5. Checklist Integración y Pruebas.

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.		

1.6. Checklist Aceptación y Entrega.

Aceptación y Entrega		
1. Identificación		
<u>Proyecto:</u>	<u>Fecha de inicio:</u> / / <u>Fecha de término:</u> / /	
<u>Nombre:</u>		
<u>E-mail:</u>	<u>Fono:</u>	
2. Actividades		
	Sí	No
¿Se entregó el soporte solicitado por el personal de desarrollo?		
Auditoría (opcional)		
¿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?		

1.7. Checklist Mantención.

Mantención	
1. Identificación	
<u>Proyecto:</u>	<u>Fecha de inicio:</u> / / <u>Fecha de término:</u> / /
<u>Nombre:</u>	
<u>E-mail:</u>	<u>Fono:</u>

2. Actividades		
	Sí	No
¿Se planificaron las actividades de SQA asociadas a la mantención?		
¿Se entregó el soporte solicitado por el personal de desarrollo?		
Auditoría (opcional)		
¿Se auditó el proceso de mantención?		
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.		

2. Revisión por Entregable de Trabajo.

2.1. Checklist Plan de Proyecto.

Checklist: Plan del Proyecto
1. Identificación del proyecto y del producto
<u>Proyecto:</u>

<u>Producto::</u>			
1. Inspector			
<u>Nombre:</u>			
<u>E-mail:</u>	<u>Fono:</u>	<u>Rol:</u>	
2. 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?			
Compleitud			
¿Se cuenta con una especificación de requerimientos preliminar adecuada para el estudio de soluciones factibles?			
¿Se informan los resultados de los 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?			

2.2. Checklist Especificación de Requerimientos.

Checklist: Especificación de Requerimientos			
1. Identificación del proyecto y del producto			
<u>Proyecto:</u>			
<u>Producto::</u>			
2. Inspector			
<u>Nombre:</u>			
<u>E-mail:</u>	<u>Fono:</u>	<u>Rol:</u>	
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?			
Compleitud			
¿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?			

2.3. Checklist Diseño Preliminar.

Checklist: Diseño Preliminar
2. Identificación del proyecto y del producto
<u>Proyecto:</u>
<u>Producto:</u>
2. Inspector

<u>Nombre:</u>			
<u>E-mail:</u>	<u>Fono:</u>	<u>Rol:</u>	
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?			
Compleitud			
• ¿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 justifican 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?			

2.4. Checklist Diseño Detallado.

Checklist: Diseño Detallado			
1. Identificación del proyecto y del producto			
<u>Proyecto:</u>			
<u>Producto:</u>			
2. Inspector			
<u>Nombre:</u>			
<u>E-mail:</u>	<u>Fono:</u>	<u>Rol:</u>	
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 flujos de datos, flujos de control e interfaces?			
¿La terminología utilizada es consistente con la empleada por los desarrolladores?			
Compleitud			
¿El diseño satisface completamente los requerimientos y su propósito?			
¿El diseño implementa los algoritmos previamente especificados?			

¿Se definen e inicializan todas las variables y constantes definidas?			
¿Se definen y verifican apropiadamente todas las entradas y salidas?			
¿Se especifican las unidades de medición, los rangos, la exactitud y la precisión de todos los parámetros?			
¿Se especifican las restricciones, como tiempo de procesamiento y tamaño, para cada unidad de diseño?			
¿Existe validación de las entradas, salidas, interfaces y resultados?			
¿Se utilizan mensajes de error significativos para los errores?			
Consistencia			
¿Se utilizan consistentemente los nombres de los elementos de datos a lo largo de la descripción y representación del diseño?			
¿El diseño de las interfaces es consistente entre sí y con el diseño preliminar?			
Facilidad de pruebas			
¿Se puede probar, demostrar y analizar que cada unidad de diseño cumple con los requerimientos establecidos?			
Factibilidad			
¿El diseño es factible según la calendarización, el presupuesto y la tecnología disponibles?			
Mantenibilidad			
¿Se ha reducido el nivel de complejidad del diseño?			
¿Las unidades de diseño son claras, comprensibles y modificables en relación con los requerimientos de mantención?			
Trazabilidad			
¿Es posible trazar el diseño con los requerimientos?			

2.5. Checklist Código.

Checklist: Código			
1. Identificación del proyecto y del producto			
<u>Proyecto:</u>			
<u>Producto:</u>			
2. Inspector			
<u>Nombre:</u>			
<u>E-mail:</u>	<u>Fono:</u>	<u>Rol:</u>	
3. Checklist			
	Sí	No	N/A
Compleitud			
¿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 previenen 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?			

2.6. Checklist Plan de Pruebas.

Checklist: Plan de Pruebas			
1. Identificación del proyecto y del producto			
<u>Proyecto:</u>			
<u>Producto:</u>			
2. Inspector			
<u>Nombre:</u>			
<u>E-mail:</u>	<u>Fono:</u>	<u>Rol:</u>	
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?			
Compleitud			
¿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 de prueba contempla las interfaces definidas?			

2.7. Checklist Procedimientos y casos de prueba.

Checklist: Procedimientos y casos de prueba			
1. Identificación del proyecto y del producto			
<u>Proyecto:</u>			
<u>Producto:</u>			
2. Inspector			
<u>Nombre:</u>			
<u>E-mail:</u>	<u>Fono:</u>	<u>Rol:</u>	
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 ordenado de pasos a 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?			
Compleitud			
¿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 cómo 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?			

2.8. Checklist Manual de Usuario.

Checklist: Manual de Usuario			
1. Identificación del proyecto y del producto			
<u>Proyecto:</u>			
<u>Producto:</u>			
2. Inspector			
<u>Nombre:</u>			
<u>E-mail:</u>	<u>Fono:</u>	<u>Rol:</u>	
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?			
Compleitud			
¿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 en 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?			

3. Revisión por Equipo de Revisión.

3.1. Checklist Moderador.

Checklist: Moderador	
1. Identificación	
<u>Proyecto:</u> <u>Producto:</u>	<u>Fecha de inicio:</u> / / <u>Fecha de término:</u> / /
<u>Nombre:</u>	
<u>E-mail:</u>	<u>Fono:</u>
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 era 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?		

3.2. Checklist Autor.

Checklist: Autor		
1. Identificación		
<u>Proyecto:</u> <u>Producto:</u>	<u>Fecha de inicio:</u> / / <u>Fecha de término:</u> / /	
<u>Nombre:</u>		
<u>E-mail:</u>	<u>Fono:</u>	
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 necesitar acciones correctivas adicionales?		
• ¿Cooperó con el moderador y los inspectores en la revisión adicional si esta tenía lugar?		

3.3. Checklist Inspector.

Checklist: Inspector		
1. Identificación		
<u>Proyecto:</u> <u>Producto:</u>	<u>Fecha de inicio:</u> / / <u>Fecha de término:</u> / /	
<u>Nombre:</u>		
<u>E-mail:</u>	<u>Fono:</u>	
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 templates, 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?		

3.4. Checklist Presentador.

Checklist: Presentador		
1. Identificación		
<u>Proyecto:</u> <u>Producto:</u>	<u>Fecha de inicio:</u> / / <u>Fecha de término:</u> / /	
<u>Nombre:</u>		
<u>E-mail:</u>	<u>Fono:</u>	
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?		

3.5. Checklist Secretario.

Checklist: Secretario		
1. Identificación		
<u>Proyecto:</u> <u>Producto:</u>	<u>Fecha de inicio:</u> / / <u>Fecha de término:</u> / /	
<u>Nombre:</u>		
<u>E-mail:</u>	<u>Fono:</u>	
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?		

4. Control de Riesgos.

Hoja de Control de Riesgo														
Nombre Proyecto: "Toc&Talk!"														
<table border="1"><tr><td><table border="1"><tr><td>ID: R1</td></tr><tr><td>Prioridad: 9</td></tr><tr><td>Probabilidad: 3 (Alta)</td></tr></table></td><td colspan="2">Declaración del Riesgo: Que la implementación de la videollamada en el sistema no sea efectiva para tener una conversación de mínimo 30 min.</td></tr><tr><td><table border="1"><tr><td>Impacto: 3 (Alta)</td></tr><tr><td>TimeFrame: Inminente</td></tr></table></td><td><table border="1"><tr><td>Origen: Equipo</td><td>Clasificación: Técnico</td></tr></table></td></tr></table>			<table border="1"><tr><td>ID: R1</td></tr><tr><td>Prioridad: 9</td></tr><tr><td>Probabilidad: 3 (Alta)</td></tr></table>	ID: R1	Prioridad: 9	Probabilidad: 3 (Alta)	Declaración del Riesgo: Que la implementación de la videollamada en el sistema no sea efectiva para tener una conversación de mínimo 30 min.		<table border="1"><tr><td>Impacto: 3 (Alta)</td></tr><tr><td>TimeFrame: Inminente</td></tr></table>	Impacto: 3 (Alta)	TimeFrame: Inminente	<table border="1"><tr><td>Origen: Equipo</td><td>Clasificación: Técnico</td></tr></table>	Origen: Equipo	Clasificación: Técnico
<table border="1"><tr><td>ID: R1</td></tr><tr><td>Prioridad: 9</td></tr><tr><td>Probabilidad: 3 (Alta)</td></tr></table>	ID: R1	Prioridad: 9	Probabilidad: 3 (Alta)	Declaración del Riesgo: Que la implementación de la videollamada en el sistema no sea efectiva para tener una conversación de mínimo 30 min.										
ID: R1														
Prioridad: 9														
Probabilidad: 3 (Alta)														
<table border="1"><tr><td>Impacto: 3 (Alta)</td></tr><tr><td>TimeFrame: Inminente</td></tr></table>	Impacto: 3 (Alta)	TimeFrame: Inminente	<table border="1"><tr><td>Origen: Equipo</td><td>Clasificación: Técnico</td></tr></table>	Origen: Equipo	Clasificación: Técnico									
Impacto: 3 (Alta)														
TimeFrame: Inminente														
Origen: Equipo	Clasificación: Técnico													
Contexto: El mayor riesgo de la plataforma radica en su principal funcionalidad, para Toc&Talk!, es la videollamada pues mediante el uso de este medio para tener conversaciones entre dos usuarios al azar, y crear una experiencia única donde puedan mejorar el aspecto lingüístico de cada uno.														
Estrategia de Mitigación: 1. Investigar sobre los protocolos que son utilizados para generar una conversación de voz entre usuarios, y como adaptarlos a nuestra plataforma. Además indagar sobre las distintas librerías de Javascript que trabajan con conexiones de llamadas. 2. Diseñar una interfaz como primer acercamiento y implementarla el primer demo. 3. Corregir errores de pruebas unitarias, de integración y funcional y mejorar el servicio Cliente-Servidor de la plataforma. 4. Probar la videollamada y medir el tiempo hasta que se logra saturar con una cantidad de usuarios definidos por tramo (0 a 10, 10 a 20, etc).														
Plan de Contingencia y Gatillo: 1. Investigar sobre cómo adaptar nuestra plataforma mediante el uso de API con servicios confiables como Skype, Google Hangouts, etc. 2. Diseñar una conexión de las API's de videollamadas (deseable) y llamada de voz (obligatoria) con el servicio elegido. Medir tiempo de saturación con una cantidad de usuarios por tramos. Gatillo: Diferencia de 5 días entre la fecha actual y el tiempo de plazo de entrega de la														

funcionalidad.

Hoja de Control de Riesgo

Nombre Proyecto: "Toc&Talk!"

ID: R4

Prioridad: 4

Probabilidad: 2 (Media)

Declaración del Riesgo:

Que el servidor no pueda soportar a lo mínimo 1000 usuarios concurrentes.

Impacto: 2 (Media)

TimeFrame: Largo Plazo

Origen:
Equipo

Clasificación:
Proyecto

Contexto: Para que el sistema sea considerado "útil", los atributos de calidad "Usabilidad" y "Disponibilidad" deben tener mucha influencia en el sistema, ya que necesario que existan una cantidad mínima de usuarios para que producto sea viable. Y como esté construida la infraestructura del servidor que dejara el sistema operando, será fundamental. Tratando de dar "servicio" a los múltiples usuarios que envían consultas al mismo tiempo o tiempos distintos.

Estrategia de Mitigación:

1. Diseñar casos de pruebas de rendimiento del servidor (para este caso, pruebas de stress).
2. Aplicar los casos de pruebas y analizar los resultados obtenidos frente a los criterios de aceptación.
3. Corregir los errores en caso de no obtener los resultados esperados.
4. En base a las revisiones, estudiar la cantidad de servidores a comprar.

Plan de Contingencia y Gatillo:

1. Estudiar el desempeño v/s costo de arrendar más servidores de lo usual.
2. Aplicar la elección deseada.
3. Diseñar pruebas de stress para verificar las salidas esperables v/s las obtenidas.

Gatillo: Servidor con la versión de prueba funcionando y no soportar más de 700 usuarios.

Hoja de Control de Riesgo

Nombre Proyecto: "Toc&Talk!"

ID: R6 Prioridad: 4 Probabilidad: 2 (Media)	Declaración del Riesgo: Que no exista una cantidad mínima de usuarios de distintas lenguas para hacer la videollamada entre dos personas y así generar la conversación en el idioma que se busca aprender.	
Impacto: 2 (Media) TimeFrame: Largo Plazo	Origen: Equipo	Clasificación: Proyecto

Contexto: Para que un usuario pueda conversar en otro idioma con algún usuario que tenga manejo en ese idioma, es necesario que al menos exista esa posibilidad. Entonces el sistema debe abarcar usuarios de distintos idiomas que busquen hablarlo.

Estrategia de Mitigación:

1. Definir pocos idiomas para conversar (máximo 3).
2. Idear estrategias de difusión para países que conversen esos idiomas como lengua

nativa. 3. Idear actividades en el sistema para probar la cantidad de gente de diferentes idiomas que usan la plataforma. 4. Chequear en la base de datos solamente los idiomas que tiene cada usuario y hacer un plan de acción frente a los resultados.
Plan de Contingencia y Gatillo: 1. Contratar profesores para los primeros días del lanzamiento del producto. Gatillo: Demo del producto lanzado y una cantidad de usuarios que hablen al menos 2 idiomas distintos, menor a 100.

Hoja de Control de Riesgo														
Nombre Proyecto: "Toc&Talk!"														
<table border="1"> <tr> <td> <table border="1"> <tr><td>ID: R8</td></tr> <tr><td>Prioridad: 3</td></tr> <tr><td>Probabilidad: 1 (Baja)</td></tr> </table> </td> <td colspan="2"> Declaración del Riesgo: Que el diseño de la plataforma no parezca atractivo para el usuario (el color y/o la forma). </td> </tr> <tr> <td> <table border="1"> <tr><td>Impacto: 3 (Alta)</td></tr> <tr><td>TimeFrame: Largo Plazo</td></tr> </table> </td> <td> <table border="1"> <tr> <td>Origen: Equipo</td> <td>Clasificación: Técnico</td> </tr> </table> </td> </tr> </table>			<table border="1"> <tr><td>ID: R8</td></tr> <tr><td>Prioridad: 3</td></tr> <tr><td>Probabilidad: 1 (Baja)</td></tr> </table>	ID: R8	Prioridad: 3	Probabilidad: 1 (Baja)	Declaración del Riesgo: Que el diseño de la plataforma no parezca atractivo para el usuario (el color y/o la forma).		<table border="1"> <tr><td>Impacto: 3 (Alta)</td></tr> <tr><td>TimeFrame: Largo Plazo</td></tr> </table>	Impacto: 3 (Alta)	TimeFrame: Largo Plazo	<table border="1"> <tr> <td>Origen: Equipo</td> <td>Clasificación: Técnico</td> </tr> </table>	Origen: Equipo	Clasificación: Técnico
<table border="1"> <tr><td>ID: R8</td></tr> <tr><td>Prioridad: 3</td></tr> <tr><td>Probabilidad: 1 (Baja)</td></tr> </table>	ID: R8	Prioridad: 3	Probabilidad: 1 (Baja)	Declaración del Riesgo: Que el diseño de la plataforma no parezca atractivo para el usuario (el color y/o la forma).										
ID: R8														
Prioridad: 3														
Probabilidad: 1 (Baja)														
<table border="1"> <tr><td>Impacto: 3 (Alta)</td></tr> <tr><td>TimeFrame: Largo Plazo</td></tr> </table>	Impacto: 3 (Alta)	TimeFrame: Largo Plazo	<table border="1"> <tr> <td>Origen: Equipo</td> <td>Clasificación: Técnico</td> </tr> </table>	Origen: Equipo	Clasificación: Técnico									
Impacto: 3 (Alta)														
TimeFrame: Largo Plazo														
Origen: Equipo	Clasificación: Técnico													
Contexto: Para que el sistema sea usable, el usuario debe sentirse cómodo con la forma y color de la plataforma. De no ser así, genera una actitud de rechazo y disminuyen sus motivaciones de mejorar lingüísticamente a través de la plataforma.														
Estrategia de Mitigación: 1. Diseñar diferentes alternativas de interfaces para cada funcionalidad. 2. Consultar con conocidos y expertos en temas de interfaces usuarias sobre los diseños														

- y registrar cada feedback que le hagan.
3. Corregir el diseño en caso de ser necesario (no se puede satisfacer el gusto de todos).
 4. Implementar diseño mejorado.

Plan de Contingencia y Gatillo:

1. Confeccionar una encuesta masiva para 200 usuarios con los diseños de las interfaces.
2. Aplicar dicha encuesta en el plazo de 5 días.
3. En base al feedback recibido, mejorar el diseño e implementarlo.

Gatillo: Muchas cuentas borradas en un tiempo determinado.

Hoja de Control de Riesgo

Nombre Proyecto: "Toc&Talk!"

<table border="1"> <tr> <td>ID: R9</td> </tr> <tr> <td>Prioridad: 3</td> </tr> <tr> <td>Probabilidad: 1 (Baja)</td> </tr> </table>	ID: R9	Prioridad: 3	Probabilidad: 1 (Baja)	Declaración del Riesgo: Que las interfaces usuarias no sean intuitivas.	
ID: R9					
Prioridad: 3					
Probabilidad: 1 (Baja)					
<table border="1"> <tr> <td>Impacto: 3 (Alta)</td> </tr> <tr> <td>TimeFrame: Inminente</td> </tr> </table>	Impacto: 3 (Alta)	TimeFrame: Inminente	<table border="1"> <tr> <td>Origen: Equipo</td> <td>Clasificación: Técnico</td> </tr> </table>	Origen: Equipo	Clasificación: Técnico
Impacto: 3 (Alta)					
TimeFrame: Inminente					
Origen: Equipo	Clasificación: Técnico				

Contexto: Una interfaz usuaria debe ser simple para el usuario, si se bombardea con muchos elementos, puede molestarle y rechazar el sistema. Otro problema puede ser si al usuario le cuesta trabajo utilizar las funciones del sistema.

Estrategia de Mitigación:

1. Investigar diferentes interfaces usuarias que sean muy intuitiva.
2. Estudiar las razones de su diseño y posteriormente, hacer un diseño de lo estudiado.

3. Implementar diseños y hacer pruebas alfas con pocos usuarios, con tiempo controlado, para acceder a distintas funciones.
4. Mediante el tiempo obtenido, hacer correcciones de diseño y volver a implementarlo.

Plan de Contingencia y Gatillo:

1. Contratar un experto en FrontEnd con experiencia en crear interfaces intuitivas.
2. En base al presupuesto, diseñar la mayor cantidad de vistas posibles.

Gatillo: Pruebas alfas y con tiempo de aprendizaje de más de 10 min con un sistema de pocas funcionalidades (entre 3 a 4).

Hoja de Control de Riesgo

Nombre Proyecto: "Toc&Talk!"

<table border="1"> <tr> <td>ID: R2</td> </tr> <tr> <td>Prioridad: 3</td> </tr> <tr> <td>Probabilidad: 1 (Baja)</td> </tr> </table>	ID: R2	Prioridad: 3	Probabilidad: 1 (Baja)	Declaración del Riesgo: Que la infraestructura del servidor no tenga la capacidad necesaria para guardar la información de forma segura de 1000 usuarios.	
ID: R2					
Prioridad: 3					
Probabilidad: 1 (Baja)					
<table border="1"> <tr> <td>Impacto: 3 (Alta)</td> </tr> <tr> <td>TimeFrame: Largo Plazo</td> </tr> </table>	Impacto: 3 (Alta)	TimeFrame: Largo Plazo	<table border="1"> <tr> <td>Origen: Equipo</td> <td>Clasificación: Proyecto</td> </tr> </table>	Origen: Equipo	Clasificación: Proyecto
Impacto: 3 (Alta)					
TimeFrame: Largo Plazo					
Origen: Equipo	Clasificación: Proyecto				

Contexto: Para la plataforma es fundamental que haya una cantidad grande de usuarios. Es por ello que es necesario tener una gran capacidad de almacenamiento al momento de guardar las cuentas de cada uno, además de los mensajes del chat y otras informaciones importantes.

Estrategia de Mitigación:

1. Arrendar un servidor con un mínimo de 100 GB de almacenamiento.
2. Hacer pruebas con cuentas "dummy" para chequear que pueda guardar al menos 1000

usuarios.

3. Hacer pruebas para chequear la cantidad máxima de usuarios que puede guardar.
4. Hacer pruebas con mensajes “dummy” para chequear la máxima cantidad de mensajes que puede guardar.

Plan de Contingencia y Gatillo:

1. Arrendar un servidor con un mínimo de 500 GB de almacenamiento.
2. Revisando el presupuesto del proyecto, cotizar servidores para suplir la gran carga de almacenamiento en los primeros días del lanzamiento.

Gatillo: 1 semana del lanzamiento al mercado y no haya servidores arrendados.