

Proyecto Final de Tecnicatura

Plan de Pruebas



The Boys

Integrantes:

ARIADNA DIAZ ROVETTA (ariadna.diaz@estudiantes.utec.edu.uy),
FEDERICO LOPEZ SENA (federico.lopez@estudiantes.utec.edu.uy),

ROBERT JOAQUIN SANTANA FERNANDEZ
(robert.santana@estudiantes.utec.edu.uy),

NAHUEL TORENA BORDÓN
(nahuel.torena@estudiantes.utec.edu.uy),

CHRISTOFER RYAN VAZQUEZ PERALTA
(christofer.vazquez@estudiantes.utec.edu.uy)

Historia de revisiones

Versión	Autor(es)	Descripción	Fecha
1.0	Nahuel Torená	Creación del documento	16/09/2022
1.1	Nahuel Torená	Adjunción de imágenes y diseño de suite de pruebas	23/09/2022
1.2	Nahuel Torená	Registro de ejecución de CPs	08/10/2022
1.3	Nahuel Torená	Cambio en base a FeedBack	17/08/2022
1.4	Nahuel Torená	Riesgos e impactos	19/08/2022
1.5	Nahuel Torená	Registro de ejecución	22/10/2022
1.6	Nahuel Torená	Corrección de documento	03/11/2022
1.7	Nahuel Torená	Corrección del documento	05/11/2022
1.8	Nahuel Torená	Actualización de cronograma y estrategia	09/11/2022
1.9	Nahuel Torená	Cambios en base a feedback	15/11/2022

2.0	Nahuel Torena	Ajustes Finales	20/11/2022
-----	---------------	-----------------	------------

1. Introducción	2
1.1 Propósito	2
1.2 Alcance	2
1.2.1 Componentes y características que serán probadas.	3
Aplicación web:	3
Aplicación móvil (Android):	3
1.2.2 Componentes, características y pruebas que están fuera del alcance.	4
1.2.3. Roles y responsabilidades	4
1.2.4. Manejo de Ambientes	5
1.3 Suposiciones y Restricciones	5
2. Técnicas de diseño y testeo	6
3. Cronograma de Testing	7
4. Estrategia	9
4.1.1. Ciclo de vida de casos de pruebas	11
4.1.2. Ciclo de vida de defectos	11
4.1.3. Criterios de aceptación	12
5. Riesgos	13
6. Herramientas	16
7. Métricas	17
8. Anexo	20

1. Introducción

1.1 Propósito

El objetivo de este plan es explicar cómo se realizará el cubrimiento de pruebas de ambas aplicaciones, tanto web como mobile, para el proyecto final de la tecnicatura del año 2022. Este documento tiene como propósito describir la estrategia a utilizar, qué técnicas se aplicaran, la metodología a aplicar, los módulos a probar, los riesgos que corre el área de testing, su posible impacto y su respectiva contingencia.

1.2 Alcance

Como anteriormente se mencionó, el testing será aplicado tanto a la aplicación web como a la mobile, por lo tanto el alcance cubre ambas aplicaciones, para cada aplicación se tiene un diferente conjunto de requerimientos a ser cubiertos, además diferentes para cada equipo de proyecto.

En base a estos requerimientos nuestro equipo abarcara pruebas que cubran en su totalidad las funcionalidades previstas y desarrolladas por el equipo de desarrollo, los componentes a ser cubiertos se detallaran a continuación. Los factores de la calidad no funcionales como el rendimiento, la seguridad informática y la usabilidad no se probarán en este proyecto en primera instancia, la prioridad es la calidad del producto.

1.2.1 Componentes y características que serán probadas.

Los componentes a ser probados por el equipo de testing como se comentó serán divididos por tipo de aplicación(Web y Android), las dividiremos en módulos para su entendimiento y comodidad del equipo:

Aplicación web:

- Login
- Acceso a funcionalidades según rol de usuario
- (RF-001) Gestión de Usuarios
- (RF-002-05) Registrar medicamento
- (RF-002-06) Listar medicamentos
- (RF-002-07) Eliminar medicamento
- (RF-002-08) Modificar medicamento
- (RF-002-17) Registrar ternera
- (RF-002-18) Listar terneras
- (RF-002-19) Eliminar ternera
- (RF-002-20) Modificar ternera
- (RF-003-02) Registrar alimentación
- (RF-003-05) Registrar temperatura de ternera
- (RF-002-21) Carga de datos históricos de enfermedad
- (RF-002-22) Listado de datos históricos de enfermedad

Aplicación móvil (Android):

- Login
- (RF-002-17) Registrar ternera
- (RF-002-18) Listar terneras
- (RF-002-21) Carga de datos históricos de enfermedad

- (RF-002-22) Listado de datos históricos de enfermedad

1.2.2 Componentes, características y pruebas que están fuera del alcance.

-Pruebas unitarias en sí no serán probadas por el equipo de testing en su plenitud ya que se espera que estas pruebas sean controladas en su mayoría por el equipo de desarrollo, algo muy básico que chequee su funcionamiento. El equipo de testing después decidirá dependiendo del tiempo disponible, tipo de funcionalidad y complejidad si es necesario revisarlo de forma unitaria.

-Pruebas de performance no serán realizadas debido al tiempo con el que cuenta el equipo para realizar el testing, por lo que, se concentraran los esfuerzos en las pruebas de integración y en el registro de casos de pruebas. Al ser un aplicación web y móvil de pocos recursos se espera que la performance no se vea afectada en gran medida.

-Pruebas automatizadas o de caja blanca, si bien contamos con conocimientos como para realizar pruebas automatizadas decidimos no quitar tiempo del equipo de testing en la creación de estas mismas y fortalecer las pruebas manuales y el diseño de los casos de prueba que aportaran mayor calidad al producto.

-Reporte de defectos anteriores a la aplicación web respecto de la aplicación de escritorio, en caso de ser arrastrado algún defecto desde esa lógica, se reportara y solucionara como un nuevo defecto.

-Las siguientes funcionalidades tampoco serán tenidas en cuenta ya que no serán desarrolladas por el equipo:

- (RF-002-01) Registrar alimento

- (RF-002-02) Listar alimentos
- (RF-002-03) Eliminar alimento
- (RF-002-04) Modificar alimento
- (RF-002-09) Registrar enfermedad
- (RF-002-10) Listar enfermedades
- (RF-002-11) Eliminar enfermedad
- (RF-002-12) Modificar enfermedad
- (RF-002-13) Crear guachera
- (RF-002-14) Listar guacheras
- (RF-002-15) Eliminar guachera
- (RF-002-16) Modificar guachera
- (RF-003-01) Registrar enfermedad detectada
- (RF-003-03) Registrar medicación
- (RF-003-04) Registrar peso de ternera

Estas funcionalidades serán desarrolladas por diferentes equipos pero quedarán fuera del alcance del nuestro.

1.2.3. Roles y responsabilidades

Si bien tener marcado roles para cada área y responsables ayuda en gran medida a la delegación de tarea, también creemos que como equipo deberíamos ser capaces de trabajar en todas las áreas independientemente de los roles y áreas que seamos asignados. Sin embargo, asignaremos los mismos como una simple formalidad a cumplir y para tener referencias, esto no indica que los integrantes del equipo no trabajen en las demás áreas de infraestructura, base de datos o programación.

- Gerente de pruebas: Nahuel Torená, encargado de seguir el cumplimiento de procesos, realizar reportes finales, destinar recursos y/o personal.
- Líder de equipo de Testing: Federico Lopez, encargado de asignar tareas o defectos, diseño de casos de pruebas y aprobación de casos y defectos.
- Tester: Ariadna Diaz, diseño y ejecución de pruebas, encargado del reporte de defectos.

Será responsabilidad de cada integrante mantener el sistema actualizado con las últimas funcionalidades desarrolladas de forma que en todo momento se hagan pruebas sobre ambientes estables y actualizados.

1.2.4. Manejo de Ambientes

Si bien en el manejo de ambiente se espera que se tenga un ambiente de pruebas donde realizar el testing y otro de desarrollo donde se hagan el desarrollo de funcionalidades, para nuestro proyecto se decidió utilizar un único ambiente, el ambiente será utilizado para la persona encargada de testing como ambiente de pruebas. En el caso de la persona encargada del desarrollo, se utilizará como ambiente de desarrollo.

En caso de cambiar roles, la base de datos está a disposición de cada integrante para ser reiniciada, borrando los valores y permitiendo el ingreso de valores desde 0, así como también el proyecto se puede instalar desde la última versión estable. Teniendo esto en cuenta se podrá realizar un manejo de ambientes adecuado teniendo en cuenta el equipo entero como un conjunto.

A su vez cuando una versión esté estable para ser testeada y sea liberada por el desarrollador se notificará al equipo de testing para que pueda descargar la nueva versión y comenzar con las pruebas. Los controles se irán realizando junto con los sprints, por ejemplo, en el Sprint 1, la liberación al equipo de Testing será la Versión 1, para el Sprint 2 la Versión 2, así sucesivamente. En caso de realizarse más versión durante un sprint, se agregará un número ascendente después del punto, ejemplo: Version 1.2/ Version 2.3.

Estas versiones servirán a la hora de notificar en Mantis los errores junto con el número de versión en el que se encontró y llevar un control de los mismos. También se anotará en que versiones se trabajó en este mismo documento para cada sprint y el equipo sabrá en todo momento la versión en la que se trabaja por lo diferentes medios de comunicación que se utilizan.

1.3 Suposiciones y Restricciones

Suposiciones:

- El ambiente de pruebas será en el ambiente local de cada uno de los integrantes, por lo que, cada set de prueba usado para realizar pruebas de la aplicación será responsabilidad de cada integrante.
- Respecto al versionado de la aplicación, será de vital importancia que cualquier cambio relacionado a la aplicación sea comunicado al equipo por cualquier vía de comunicación disponible para evitar conflictos a la hora del testing.

Restricciones:

- Las reuniones que requieran la participación completa del equipo se realizarán después de las 21:00 hs debido a las actividades laborales y/o académicas de los miembros del equipo.
- Diferentes capacidades del hardware de los integrantes, por lo que se puede dar ciertos problemas de performance que no estén relacionadas estrictamente con el código.

2. Técnicas de diseño y testeo

Dentro de las técnicas de diseño y testeo usaremos principalmente el testing de caja negra donde conociendo los valores esperados los compararemos con los obtenidos por el sistema y en base a esto comprobaremos el funcionamiento de nuestra aplicación. Se buscará con el testing de caja negra enfocarnos en las entradas y salidas del sistema sin preocuparnos por el código interno desde el lado del testing.

Mayoritariamente se usarán técnicas de clases de equivalencia y valores límites ya que por la naturaleza de la aplicación con estas pruebas podemos cubrir ampliamente todas las pruebas que requerimos. Los casos de prueba que utilizaran estas técnicas serán formados a partir de casos de uso brindados por el cliente donde en base a los requerimientos podemos crear casos que validen estas y aseguren que funcionen como deberían.

Como primer foco de pruebas será importante asegurarnos de que el ambiente de los integrantes del equipo local funcione correctamente por lo que las primeras pruebas deberán ser dirigidas a comprobar que la aplicación esté bien instalada para así poder trabajar en base a esta. Más detallado estará en los criterios de iniciación.

Se deben realizar pruebas de aceptación y de integración para asegurar que lo realizado está en línea con lo requerido, las pruebas de integración serán realizadas una vez que la funcionalidad desarrollada se adjunte con los demás componentes en la versión liberada a testear. En caso de que los incidentes encontrados no sean bloqueantes, se podrá dar como

Por último se realizarán pruebas de sistema cuando las pruebas de integración se hayan realizado y tengan un resultado exitoso para corroborar que funciona como un sistema unido y compacto, ya que puede darse el caso en que las funcionalidades integradas funcionan pero a la hora de ser un sistema unido no se comporte debidamente.

Con esto en mente, el registro de defectos y los casos de prueba serán actualizados a medida que se vayan realizando las pruebas de las funcionalidades en el transcurso de los sprints. Por último se realizará el informe de resultados ya acabando los sprints.

3. Cronograma de Testing

Fecha de Inicio y Final de Sprint	Sistema	Módulo	Funcionalidad	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5
Sprint 1- (05/09 - 24/09)	Web	Login	Login					
		Acceso a funcionalidades según rol de usuario	Acceso a funcionalidades según rol de usuario					
		Gestión de Usuarios	Alta de Usuario					
Prueba el equipo de Testing - Liberación día 22/09								
Sprint 2- (26/09 - 08/10)	Web	Gestión de usuarios	Modificación de Usuario					
			Listado de usuario					
			Borrado de Usuario					
	Móvil	Login	Login					
		Acceso a funcionalidades según rol de usuario	Acceso a funcionalidades según rol de usuario					
Prueba el equipo de Testing - Liberación día 06/10								
Sprint 3- (10/10 - 22/10)	Web	Gestión de Terneras	Registrar ternera					
			Listar terneras					

			Eliminar ternera					
			Modificar ternera					
Prueba el equipo de Testing - Liberación día 20/10								
Sprint 4- (24/10 - 05/11)	Web	Gestión de Medicamentos	Registrar medicamento					
			Listar medicamentos					
			Eliminar medicamento					
			Modificar medicamento					
Prueba el equipo de Testing - Liberación día 03/10								
Sprint 5- (07/11 - 21/11)	Móvil	Gestión de Terneras	Registrar ternera					
			Listar terneras					
		Gestión de enfermedades	Carga de datos históricos de enfermedad					
			Listado de datos históricos de enfermedad					
	Web	Gestión de enfermedades	Carga de datos históricos de enfermedad					

El cronograma anterior fue el planificado para el Testing divididos por módulos y funcionalidades de acuerdo a lo desarrollado por el equipo de programación. Cabe destacar que es un cronograma tentativo ya que se contempla que pueden existir atrasos en el área de desarrollo. Para la resolución de incidentes, no se tiene un tiempo específico ya que dependerá mucho del caso, lo ideal es resolverlo según vayan saliendo para el siguiente Sprint, por lo que, considerando esto, también pueden existir retrasos en caso de complicaciones en su resolución.

Los días Jueves de la segunda semana de cada sprint se espera que se tenga lista la versión a ser liberada al equipo de Testing para dejar un tiempo moderado para la realización de pruebas y la creación de documentos necesarios. Si el equipo de desarrollo

necesita mayor tiempo, se esperará que el tiempo se extienda hasta más tardar el Viernes por la noche, que si bien no es lo ideal, se espera que se pueda cumplir con ese tiempo extra.

En caso de verse atrasado por cualquier motivo el desarrollo o testing de una nueva funcionalidad, será detallado en el informe de resultados explicando el motivo de retraso y como se planea continuar intentando apegarse lo máximo que se pueda al cronograma inicial.

4. Estrategia

La estrategia para el testing será dividida por sprints y de acuerdo a las funcionalidades previstas a probar por el cronograma marcado además de que se mantendrá una estrecha comunicación con el área de desarrollo para saber el estado de las funcionalidades. Por cada Sprint se creará un Build en TestLink con un versionado de forma de realizar la ejecución de cada sprint en diferentes builds.

- Sprint 1: Se realizarán las lecturas conjuntas de los requerimientos necesarios de toda la aplicación junto a los compañeros. Una vez claros los requerimientos, se comenzará desde el área de testing a diseñar los casos de prueba para las funcionalidades previstas para el sprint 1 siempre basándose en los requerimientos del documento. Se prevé que habrá dificultades a la hora de la configuración de ambientes y la instalación de versión por lo que las funcionalidades a desarrollar y testear son considerablemente menores en este primer sprint. Se realizará las ejecuciones de los casos de pruebas diseñados anotando los resultados arrojados de las pruebas junto con el reporte en la herramienta Mantis de los incidentes encontrados.

- Sprint 2: Se corregirá el documento en caso de ser necesario por el feedback devuelto y de ser necesario también se modificarán los casos de pruebas existentes en TestLink. Se continuará con el diseño de los casos de prueba planeados para el módulo de 'Gestion de Usuarios' para las 3 funcionalidades restantes (Baja, modificación y listado de usuarios) y el diseño de casos para primeras funcionalidades de la aplicación móvil que incluye el Login, acceso por rol de usuario en una build nueva. Se planea realizar una revisión de los flujos anteriores teniendo en cuenta los incidentes marcados como resueltos al inicio de las pruebas. Para el caso de la aplicación web, se realizarán pruebas por funcionalidad de forma independiente, para luego realizar una revisión integrada de las funcionalidades como un módulo solo en el que se verificará que pueden funcionar de acuerdo a lo requerido en conjunto. En el caso de la aplicación móvil se harán pruebas similares a las que se hicieron para los mismos módulos de la aplicación web, con las modificaciones necesarias en las pruebas para adaptarse a un testing móvil. Por último se realizará la ejecución de pruebas y el informe mostrando los resultados obtenidos.

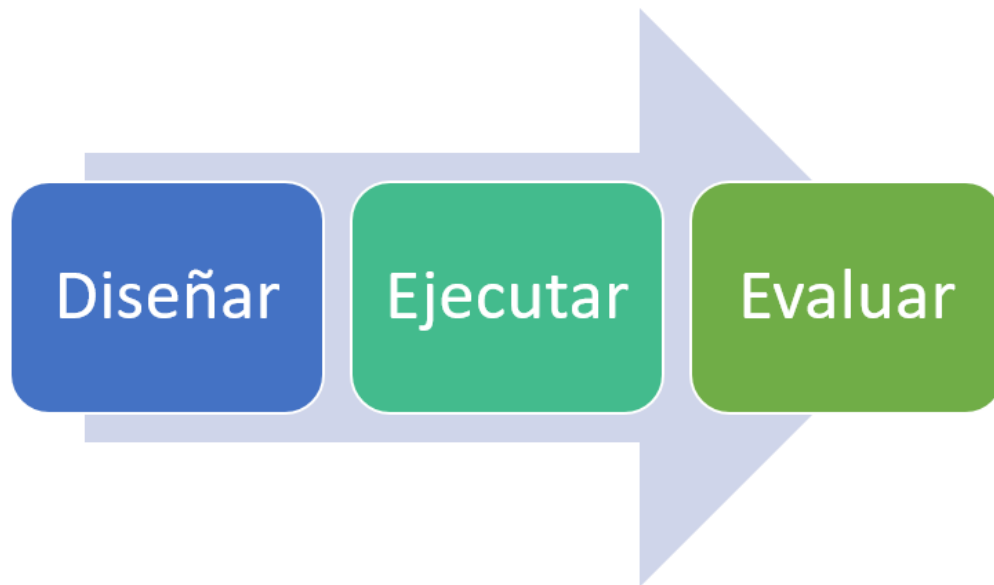
-Sprint 3: Para este sprint se espera realizar el módulo completo de terneras. Se comenzará por revisar los incidentes solucionados del anterior sprint. Se realizará la cobertura de casos de prueba por funcionalidad teniendo como referencia los requerimientos para realizar los casos bordes y de valores límites. La integración se verifica una vez cada funcionalidad nueva haya sido probada con un testing exploratorio. Una vez finalizado el diseño de casos de prueba se creará una nueva build donde se asignan a un tester los casos de prueba a ejecutar. Una vez finalizado, se registrarán los incidentes y se anotarán los resultados y conclusiones en el informe de resultados.

-Sprint 4: En este sprint se prevé continuar con el módulo de “gestión de medicamentos” que incluye las funcionalidades de dar de alta, baja, listar y modificación de medicamentos. Siguiendo la misma estrategia que el anterior sprint, se plantearía revisar los incidentes que hayan sido marcados como resueltos primero para darlos como resueltos o reabiertos en su contraparte. También como en el anterior sprint se buscará probar la integración de las 4 funcionalidades una vez hayan sido probadas por separado. También el diseño de los casos será realizado en un build con diferente versión y en caso tener que ajustarse los casos de prueba, se creará una nueva versión de los mismos con los ajustes necesarios. Por último se registrarán los incidentes y se verán reflejados junto con las conclusiones y gráficas en el informe de resultados.

-Sprint 5: Este será el último sprint y donde deberemos dar por finalizado todos los pendientes que queden, por lo que es importante que estemos dentro del cronograma planeado para poder cumplir. Se verificarán los incidentes que estén marcados como solucionados. Se realizará la ejecución de las últimas funcionalidades, primero el módulo de mantenimiento del tambo con sus 2 funcionalidades junto con la carga y listado de historicos, eso dentro de la aplicación web, para la aplicación móvil se realizará el diseño y ejecución del registro y listado de ternera junto con la carga y listado de historicos. Este sprint comparado con los demás tiene unos días más por lo que es el más cargado de los 5 sprints. Finalmente una vez realizada la ejecución de los últimos casos de prueba se adjuntan los resultados en el informe de resultados y se da cierre a todos los documentos.

4.1.1. Ciclo de vida de casos de pruebas

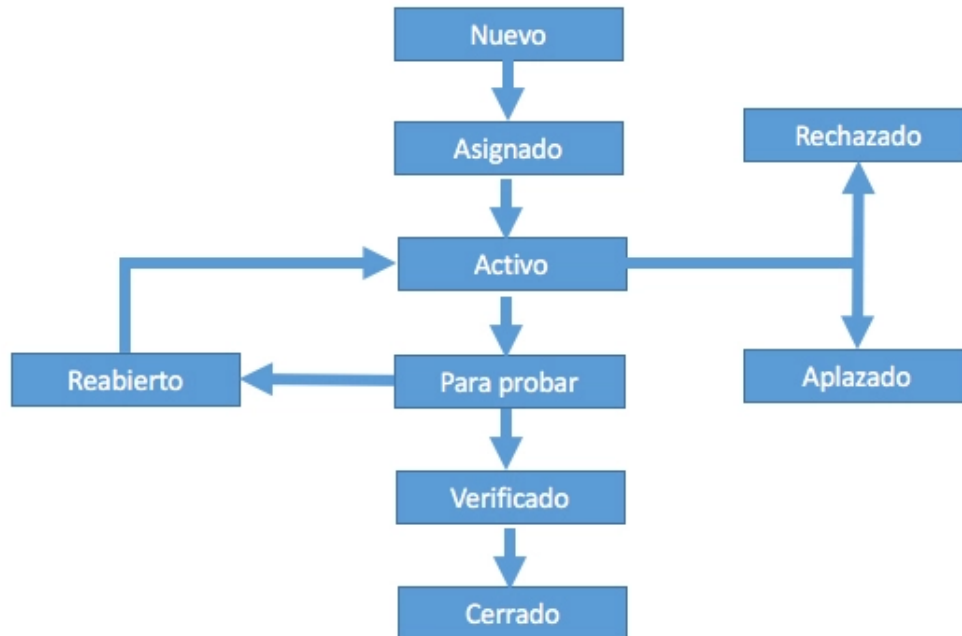
El proceso en el que llevaremos la realización de los casos de prueba son los pasos que se realizan dentro de la herramienta Testlink.



- Diseñar: en esta etapa se buscará diseñar los casos de prueba en base a los requerimientos de las funcionalidades correspondientes al sprint. El diseño se realizará directamente sobre Testlink en la suite de pruebas, aplica para ambas aplicaciones.
- Ejecutar: finalizada la etapa de diseño, se buscará ejecutar aquellas pruebas correspondientes al sprint en curso, aquellas pruebas que tengan el estado “fallo”, serán tomadas en cuenta como posible defecto y analizando cada caso serán tomadas como defectos a reportar en Mantis.
- Evaluar: se evaluará de forma continua las pruebas que se hayan diseñado, ejecutado, así como sus estados, de esta forma determinaremos el avance de las pruebas y sabremos si tenemos la aprobación para un pasaje a entregable.

4.1.2. Ciclo de vida de defectos

Los defectos que finalmente son reportados en Mantis seguirán un ciclo de vida representado por la siguiente imagen:



- **Nuevo:** defecto que fue reportado recientemente y no ha sido asignado a ningún desarrollador.
- **Asignado:** el defecto fue analizado y asignado a un desarrollador para su solución.
- **Activo:** defecto que está siendo tratado por un desarrollador, a la espera de una resolución por parte del desarrollador.
- **Para probar:** defecto que está listo para ser testeado por el equipo de testing.
- **Verificado:** el equipo de testing vio el defecto solucionado y dio su aprobación.
- **Cerrado:** defecto cerrado y solucionado comprobado por el equipo de testing.
- **Reabierto:** el defecto volvió a encontrarse por el equipo de testing, por lo que se reabre y se vuelve a marcar como activo para que el desarrollador previamente asignado lo vuelva a revisar.
- **Rechazado:** el defecto reportado no es considerado por el desarrollador como un defecto o por otro lado puede considerar que es un duplicado.
- **Aplazado:** luego del análisis del desarrollador se considera que la mejor opción es tratarlo en el siguiente sprint para aprovechar el tiempo.

4.1.3. Criterios de aceptación

Los criterios de aceptación que planteamos son pensados para la realidad de nuestro equipo y de proyecto buscando siempre un equilibrio entre el tiempo disponible y la mayor calidad posible, intentando evitar los riesgos identificados para evitar un atraso en las tareas.

-Criterios de inicio de testing: para este punto partiremos desde el momento en que el equipo de desarrollo considera una versión que está lista para ser testeada, a partir de ahí comienza el proceso de configuración del ambiente con la nueva versión, una vez finalizada la configuración se buscará probar de forma muy general si la aplicación quedó bien instalada en la máquina, para esto bastará una simple pruebas de humo hasta que el tester considere que es aceptable o que vea irregularidades.

-Criterios suspensión de pruebas: el criterio que manejaremos será para el caso que una funcionalidad sea bloqueante para continuar con las pruebas, como el caso de no poder acceder desde el login, para estos casos se suspenden las pruebas para su solventación y se volverán a ejecutar los criterios de inicio.

-Criterios de aceptación o rechazo: en pos de no comprometer los plazos de entrega pactados para cada sprint y apegarnos al cronograma de programación y testing, nuestro margen de aceptación no será tan estrictos como para tener un 100% de los casos de prueba sin errores o incidentes para su liberación. Por lo que consideramos ser más flexibles y en caso de tener más del 30% de casos fallados con prioridad alta, para ese caso se rechazaría la versión. Por lo demás, si en el informe de resultados, los casos de prueba fallados están por la mínima de 30% para severidad alta, se puede dar como aceptada la versión por parte del equipo de testing.

-Criterios de aceptación para cliente: finalmente cuando se tenga el producto completado, será necesario un criterio para aceptar o rechazar la liberación hacia el cliente, por lo que consideramos que un buen criterio sería no tener **ningún tipo de incidente de clase Alta y menos de 5 de clase media**, en caso de poseer más de esto, la versión sería rechazada y aplazada su liberación hasta corregido y alcanzó el estándar mínimo. En caso contrario y que se cumpla el estándar podrá ser liberado hacia el cliente notificando los errores conocidos.

5. Riesgos

Después de un análisis del proyecto en cuestión entre los integrantes del equipo tomando en cuenta, tiempo, capacidades individuales, generaciones, armado del equipo, conocimiento previo de la aplicación y posibles inconvenientes. Estos son algunos riesgos siendo objetivos que podrían llegar a suceder:

- Falta de trabajo en equipo previo(adaptación).
- El cambio en la plataforma a la hora de pasar a un servidor virtualizado para nuestras aplicaciones.
- Falta de comunicación entre los integrantes del equipo.
- Falta de tiempo para finalizar las funcionalidades de los sprints, generando “carry-over”.
- Compatibilidad entre todos los software que interactúan con las aplicaciones.
- Tiempo “muerto” de comprensión sobre la realidad planteada.
- Posibles retrasos ajenos al equipo de caída de servidores y/o sistemas.
- Mal entendimiento de funcionalidades.

Tabla 1. Valores de probabilidad, impacto en función de los riesgos y plan de mitigación para cada riesgo asociado.

Riesgo	Probabilidad	Impacto	Plan de Mitigación
1.Falta de trabajo en equipo previo(adaptación).	75%	5	Este riesgo depende enteramente del equipo de trabajo por lo que es recomendable comunicación fluida en el proyecto, para esto se realizarán llamadas periódicas en las que se comentará el avance que lleve cada persona, en estas instancias se buscará que todo el equipo participe aunque haya ocasiones en las que no se avance.
2.Cambio en la plataforma a la hora de pasar a un servidor virtualizado para	60%	4	Se estudiará previamente si el pasaje puede afectar alguna funcionalidad antes de ser realizado en caso de confirmarse la afección, se buscará dedicarle los 5 integrantes el tiempo de resolución fuera del horario donde cada uno desarrolla las demás áreas para compensar la afección. De igual manera se buscará diferentes formas en caso de que la elegida sea confirmada como afectuosa.

nuestras aplicacione s.			
3.Falta de comunicaci ón entre los integrantes del equipo.	90%	5	El riesgo de la comunicación tiene relación con el primer riesgo por lo que además de lo comentado en ese punto, también se utilizará herramientas como Excel y Trello para dejar evidenciado en lo que cada uno está trabajando. También se pide el ser activo en las conversaciones que se den en el grupo de WhatsApp aportando opiniones y comentarios. En caso de realizarse una reunión con los profesores, el grupo entero deberá estar presente, en caso contrario, se deberá notificar con anterioridad.
4.Falta de tiempo para finalizar las funcionalidades de los sprints, generando "carry-over".	85%	4	Se dividirán las tareas entre los diferentes miembros del equipo abordando las 4 áreas, en caso de que un área genere carry-over o crea que no llegara con las tareas asignadas, uno de los integrantes que esté más libre en ese momento o que vaya más avanzado brindara de su ayuda al área más comprometida. Esta ayuda no puede verse extendida por más de 2 días con el fin de cuidar su propia área.
5.Compatibilidad entre todos los software que interactúan con las aplicacione s.	70%	3	Ya habiendo establecido la aplicación en el área de desarrollo como estable se buscará utilizar las mismas versiones para cada programa, herramienta, librería y componentes para reducir al mínimo las posibilidades de fallo de compatibilidad entre las mismas.
6.Tiempo "muerto" de comprensión sobre la realidad planteada.	65%	3	Ya avanzado en el sprint se deberá comprender la realidad de forma clara para eso, en caso de que algún integrante tenga dudas sobre la misma, se realizará una única reunión del equipo completo donde se explicara la realidad entre los demás integrantes hasta que se comprenda completamente.
7.Posibles retrasos ajenos al equipo de caída de servidores	50%	2	Debido al tipo de caso (externo) no hay forma de saber con exactitud si se puede dar con frecuencia, en caso de darse, se intentará avanzar con las demás áreas como un refuerzo más. También se podrán realizar anotaciones(como casos de pruebas o reportes) en una planilla compartida de Excel donde se pueda seguir

y/o sistemas.			avanzando y una vez resuelto el problema, hacer la transición necesaria hacia la herramienta que corresponda.
8. Mal entendimiento de funcionalidades.	65%	3	El entendimiento de las funcionalidades debe ser claro por lo que se deberá prestar especial atención al feedback devuelto para evitar realizar funcionalidades equivocadas, a parte, se deberá consultar cualquier duda con el grupo de trabajo y en caso de duda general elevarla con el grupo tutores lo más pronto posible.

Tabla 3. Impacto

Catastrófico	5	de materializarse dañara el logro de los objetivos del proyecto
Mayor	4	de materializarse causaría una pérdida en el logro de los objetivos del proyecto
Moderado	3	de materializarse causaría daño medio en el logro de los objetivos del proyecto
Menores	2	de materializarse causaría daño a corto plazo pero no afectaría el logro de los objetivos del proyecto
Insignificante	1	Riesgo tiene un pequeño o nulo efecto en el proyecto

Si bien marcamos una cierta cantidad de riesgos, son sucesos que pueden llegar a suceder como no y pueden afectar el desempeño del equipo, también en base a estos riesgos se tomarán medidas entre los integrantes para reducir aún más la probabilidad de suceder. También existen riesgos que escapan de nuestro control como lo pueden ser la caída de servidores, sucesos personales de los integrantes o situaciones externas al proyecto, aun así estos pueden ser parte del mismo proyecto por lo cual se evalúan y toman medidas de contingencias necesarias.

6. Herramientas

Dentro de las herramientas que se usarán a la hora de realizar el testing se destacarán las 2 más importantes, Testlink y Mantis, también se usará Trello para la coordinación pero no es exclusiva del testing.

Testlink: herramienta donde se diseñará la suite de prueba donde se colocaran los casos de prueba desarrollados por el equipo de testing y en el que se ejecutarán los mismo permitiéndonos generar informes que mostrarán el avance realizado.

Mantis: en esta herramienta se reportan los defectos encontrados luego de haber ejecutados los casos de prueba en Testlink, aquí se le podrá asignar a alguien para su seguimiento, asignar prioridad, severidad y tener un control de errores.

7. Métricas

Las métricas utilizadas nos ayudarán a medir el avance y el cumplimiento de los requerimientos en este proyecto, en base a ellas podremos saber casi con exactitud en qué puntos estamos siendo fuertes, en cuales estamos más débiles y en base a eso enfocarnos en mejorar aquellas cosas que no están funcionando como deberían. También nos permitirán llevar un control más riguroso sobre las pruebas realizadas, su cubrimiento y que tan eficaces fuimos.

Indicadores de Métricas	
Código	001
Nombre	Incidentes por severidad
Detalles	Conocer la cantidad de incidentes en el periodo establecido reportado por el equipo y el nivel de severidad que se les asigna dentro de un sprint. Será extraído de Mantis en la sección de <u>Severidad</u> .
Valores estimativos	<p>Escenario óptimo: 0 caso severidad alta, hasta 2 de severidad baja/media.</p> <p>Escenario intermedio: 1 caso severidad alta y/o 3 severidad baja/media.</p> <p>Escenario de riesgo: 2+ caso de severidad alta y/o 5+ severidad baja/media.</p>
Responsable	Testing

Indicadores de Métricas	
Código	002
Nombre	Tiempo promedio de resolución por severidad. Será calculado con los tiempos sacados desde Mantis sobre la Actualización de Incidencias.
Detalles	<p>Conocer el tiempo en el que se resuelven las incidencias por nivel de severidad. Se toma como medida los días máximos en los que la incidencia queda resuelta.</p> <p>Cálculo: $Tr - Ti$</p> <p>Ti: Día y hora en el que el incidente se reportó. Tr: Día y hora en el que el incidente se resolvió.</p> <p>Ejemplo: Día: 16 hora: 21:00 Día: 15 hora: 14:00 Resultado: 1 día 7 horas.</p>
Valores estimativos	<p>Escenario óptimo: Severidad alta hasta 1 días Severidad media hasta 2 días Severidad baja hasta 4 días</p> <p>Escenario intermedio: Severidad alta hasta 2 días Severidad media hasta 3 días Severidad baja hasta 5 días</p> <p>Escenario de riesgo: Severidad alta más de 5 días Severidad media más de 7 días Severidad baja más de 10 días</p>
Responsable	Desarrollo-Testing

Indicadores de Métricas	
Código	003
Nombre	Funcionalidades realizadas. Las funcionalidades esperadas están colocadas en el cronograma por sprint y las realizadas serán brindadas por el equipo de desarrollo.
Detalles	Porcentaje de las funcionalidades realizadas que cumplen las especificaciones. Al finalizar el sprint se revisa cuantas funcionalidades requeridas fueron realizadas por el equipo en ese tiempo pactado.




	<p>Cálculo: Fr / Fe</p> <p>Fr: funcionalidades realizadas Fe: funcionalidades esperadas</p> <p>Ejemplo: $3/4 = 0.75 = 75\%$</p>
Valores estimativos	<p>Escenario óptimo: más de %90</p> <p>Escenario intermedio: entre %90 y %70</p> <p>Escenario de riesgo: menos de %70</p>
Responsable	Desarrollo

Indicadores de Métricas	
Código	004
Nombre	Casos de pruebas fallados. Esta información será extraída de TestLink al momento de terminar la ejecución de casos.
Detalles	<p>Cantidad de casos de pruebas fallados sobre el total de casos de pruebas en un sprint.</p> <p>Cálculo: Cf/Ct</p> <p>Cf: casos fallados. Ct: casos totales.</p> <p>Ejemplo: $20/40 = 0.5 = 50\%$</p>
Valores estimativos	<p>Escenario óptimo: menos del 5% del total de casos</p> <p>Escenario intermedio: entre %10 y %25 del total de casos</p> <p>Escenario de riesgo: más del % 25 del total de casos</p>
Responsable	Testing

Indicadores de Métricas	
Código	005
Nombre	Casos de pruebas totales por módulos. Esta información será extraída

	de las suites de pruebas en TestLink.
Detalles	Cantidad de casos de pruebas totales por módulos. Es un estimativo ya que habrá funcionalidades que no podrán cumplir con lo óptimo debido a su simpleza.
Valores estimativos	<p>Escenario óptimo: 15 o más casos de prueba por módulos.</p> <p>Escenario intermedio: Entre 7 y 12 casos de prueba por módulo</p> <p>Escenario de riesgo: 7 o menos casos de prueba por módulo.</p>
Responsable	Testing

8. Anexo

- Tabla sobre cronograma:
 Cronograma Testing
- Tabla sobre riesgos y impactos:
 Riesgos y mitigacion
- Casos de Uso:
 Casos de Uso