

Universidad ORT Uruguay

Facultad de Ingeniería

Ingeniería de Software 2

Obligatorio 1

Marco Fiorito (227548)

Matias Salles Dulan(201701)

13 de mayo de 2020

Declaraciones de autoría

Nosotros, Marco Fiorito y Matias Salles Dulan, declaramos que el trabajo que se presenta en esa obra es de nuestra propia mano. Podemos asegurar que:

- La obra fue producida en su totalidad mientras realizábamos Ingeniería de Software 2.
- Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad.
- Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra.
- En la obra, hemos acusado recibo de las ayudas recibidas.
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, hemos explicado claramente qué fue contribuido por otros, y qué fue contribuido por nosotros.
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.

Índice general

1. Estado de calidad del software	4
1.1. Analizadores de Código	4
1.1.1. SonarQube	5
1.1.2. Tamaño del Proyecto	5
1.1.3. PMD	6
1.1.4. FindBugs	7
1.2. Cobertura de Pruebas Unitarias	8
1.2.1. JaCoCoverage	8
1.3. Análisis de Usabilidad	9
1.3.1. Resultado	12
1.4. Análisis de la documentación	13
1.4.1. Resumen de log de versiones	13
1.4.2. RF/RNF	13
1.4.3. Casos de uso	13
1.4.4. Sesiones de pruebas	14
1.4.5. Reporte de defectos	14
1.5. Resumen de Defectos y Tareas Asociadas	16
1.5.1. Analizadores de código	16
1.5.2. Cobertura de pruebas unitarias	18
1.5.3. Usabilidad	18
1.5.4. Documentación	21
1.5.5. Nuevos defectos encontrados	22
1.5.6. Mixtos	24
1.5.7. Glosario de Codificación de los Defectos y Tareas	25
2. Análisis de cambios	26
2.1. Solicitudes de cambio	26
2.2. Análisis de Impacto	33
2.3. Glosario de Funcionalidades	36
3. Proyecto de mantenimiento	38
3.1. Identificación de interesados	39
3.2. Gestión de interesados	41
3.3. Definición de objetivos SMART del proyecto	43
3.4. Roles	44
3.5. Definición de alcance del proyecto	45

3.5.1.	Proyect Scope Statement (Pliegue de condiciones)	45
3.5.2.	Estructura de descomposición del trabajo (EDT)	48
3.5.3.	Diccionario del EDT y Estimación de Esfuerzo	49
3.6.	Cronograma de trabajo.	54
3.7.	Presupuesto inicial	55
3.8.	Plan de calidad.	56
3.8.1.	Actividades de aseguramiento y control de calidad	56
3.8.2.	Estándares a utilizar	58
3.8.3.	Calidad Inicial	58
3.8.4.	Resultado	64
3.8.5.	Glosario de Factores de Calidad de Mc Call	66
3.9.	Plan de Métricas y GQM	67
3.10.	Ciclo de vida.	69

1. Estado de calidad del software

En este capítulo se busca evaluar el estado de calidad del software utilizando varias herramientas para medir distintas partes del mismo (código, interfaz de usuario, documentación, etc.).

Dada dicha información procederemos a listar los defectos encontrados por cada sección, diseñando una solución para cada uno de ellos las cuales se agruparan en tareas. Esto luego nos servirá como insumo para definir las solicitudes de cambio y luego definir el alcance del proyecto.

1.1. Analizadores de Código

La correctitud en el código junto a la aplicación de estándares tienen un gran impacto en la mantenibilidad, comprensión y legibilidad del proyecto, por esto mismo, en esta sección utilizaremos diversos analizadores de código como SonarQube, PMD y FindBugs apuntando a conocer mas sobre dichos factores.

1.1.1. SonarQube

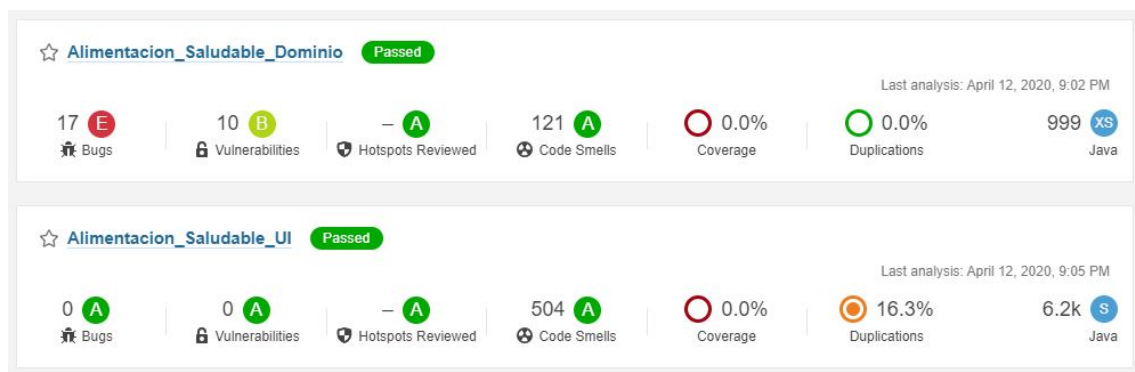
SonarQube es una plataforma para evaluar código fuente que usa diversas herramientas de análisis estático de código fuente como Checkstyle, PMD, etc. Informa sobre código duplicado, estándares de codificación, cobertura de código, complejidad ciclomática, errores potenciales, comentarios y diseño del software.

Para comprender mejor el estado de calidad del software separamos el código en 2 proyectos de SonarQube para analizar el dominio y la interfaz gráfica de forma separada.

El principal motivo de esta decisión fue el resultado del reporte de SonarQube en cuanto a las líneas duplicadas cuando analizábamos el código por completo, ya que las líneas auto-generadas por NetBeans de Java Swing afectaban los resultados globales, y excluir la interfaz implicaría no analizar todo el código que no sea auto-generado. Por ello, al separar el análisis en dos proyectos de SonarQube, podemos entender mejor de donde vienen los bugs, errores, líneas duplicadas, etc.

(En este caso excluiríamos a SonarQube como fuente para analizar la cobertura de código ya que usaremos JaCoCoverage para ello.)

Los resultados fueron los siguientes:



SonarQube Report

A pesar de la cantidad de bugs y vulnerabilidades que SonarQube clasifica como 'E' y 'B' respectivamente, SonarQube clasificó la mantenibilidad de ambos proyectos como 'A', la cual se calcula teniendo en cuenta el porcentaje de 'Code Smells' y la Deuda Técnica.

1.1.2. Tamaño del Proyecto

En el reporte anterior podemos ver que el dominio esta compuesto por solamente 1000 líneas de código aproximadamente y la UI por 6200 líneas de código aproximadamente, de las cuales varias son auto-generadas por NetBeans para inicializar los componentes de Java Swing. Teniendo en cuenta esto mismo, podemos calificar al proyecto como 'chico' respecto a su tamaño en cantidad de líneas de código.

1.1.3. PMD

PMD es una herramienta open source que dado un set de reglas analiza el código estático de la aplicación y genera reportes de issues encontrados siguiendo el set de reglas. Los reportes por lo general dejan en evidencia código ineficiente, malos hábitos de programación los cuales pueden reducir la mantenibilidad del programa si se acumulan.

Para dicho análisis utilizamos el set de reglas estándar de PMD el cual incluye todo lo necesario para encontrar CodeSmells, violaciones del estándar y código que puede ser mejorado entre otros.

Los resultados fueron los siguientes:

Summary

Rule name	Number of violations
UseUtilityClass	1
LooseCoupling	63
OverrideBothEqualsAndHashCode	5
FormalParameterNamingConventions	2
UnnecessaryLocalBeforeReturn	5
IdempotentOperations	1
UnnecessaryFullyQualifiedName	78
ForLoopCanBeForeach	11
PositionLiteralsFirstInComparisons	43
UnusedImports	11
CloseResource	1
UselessParentheses	6
UnusedFormalParameter	98
MethodNamingConventions	2
UselessOverridingMethod	2
PackageCase	1
SingularField	213
EmptyCatchBlock	1
UseCollectionIsEmpty	2
UnusedLocalVariable	1

PMD Report

A diferencia de SonarQube, PMD carece de una interfaz gráfica que muestre una visión global sobre el código analizado. Aun así, PMD nos permite ver de forma organizada la cantidad de violaciones por cada regla. A partir del resultado pudimos concluir que varias de esas violaciones pueden ser rápidamente eliminadas, como UselessParentheses, UnusedImports, ForLoopCanBeForeach entre otros, ya que el tiempo necesario para solucionar dichas violaciones no es prolongado.

1.1.4. FindBugs

FindBugs es un programa open source cuyo objetivo es buscar bugs en programas escritos en código Java utilizando análisis estático similar a PMD.

Al analizar el código FindBugs reportó 44 warnings de los siguientes tipos:

Summary

Warning Type	Number
Bad practice Warnings	34
Correctness Warnings	3
Malicious code vulnerability Warnings	2
Dodgy code Warnings	5
Total	44

FindBugs Report

Al igual que PMD carece de una interfaz gráfica como PMD. Analizando en detalle los bugs encontrados muchos de ellos pertenecen a un mismo tipo, por ejemplo:

- `UI_INHERITANCE_UNSAFE_GETRESOURCE` que puede ser arreglado simplemente haciendo que las clases involucradas en el bug sean 'final'.
- `NP_EQUALS_SHOULD_HANDLE_NULL_ARGUMENT` que se solucionarían chequeando por null en los `equals()`.

Por lo que al igual que con PMD, se podría eliminar varios de estos warnings en poco tiempo ya que las soluciones serían simples.

1.2. Cobertura de Pruebas Unitarias


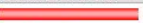




Las pruebas unitarias nos permiten encontrar varios errores en la lógica de negocio del producto por lo que se considera una parte critica del desarrollo y es importante mantener un alto porcentaje de cobertura en todo momento. A su vez realizar pruebas unitarias durante el desarrollo ayuda a prevenir bugs de regresión y busca garantizar que nuevas funcionalidades no introduzcan nuevos bugs (busca garantizar aunque nunca se está completamente seguro).

Cabe destacar que al ejecutar las pruebas unitarias en el proyecto, 97 de ellas pasan, mientras que 14 fallan y 15 generan error.

1.2.1. JaCoCoverage

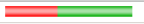
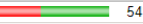

























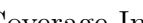

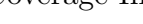
JaCoCoverage es una herramienta que analiza la cobertura de las pruebas unitarias de cada archivo dentro de la aplicación. Esta herramienta es útil a la hora de evaluar si el código está bien testado.

Luego de realizar el test de cobertura utilizando esta herramienta obtuvimos como resultado que el paquete que debe ser testado (dominio) tuvo un 61 % en cobertura de sentencias y un 57 % en cobertura de ramas. Dichos resultados no satisfacen el estándar de 90 % de cobertura mencionado en las quality gates de SonarQube entre otros.

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
interfaz		0%		0%	649	649	4,856	4,856	400	400	114	114
dominio		61%		57%	108	277	156	532	36	155	1	15
inicioSistema		0%		0%	6	6	10	10	2	2	1	1
Total	27,044 of 28,693	6%	579 of 718	19%	763	932	5,022	5,398	438	557	116	130

JaCoCoverage Report

dominio

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
Sistema		59%		54%	58	117	108	273	11	42	0	1
Sistema.Paises		0%		n/a	4	4	4	4	4	4	1	1
Usuario		43%		25%	12	21	20	45	6	15	0	1
Conversacion		53%		50%	9	21	14	39	4	13	0	1
Persona		87%		78%	4	21	3	30	1	12	0	1
InformacionMensaje		81%		n/a	1	9	4	19	1	9	0	1
Sistema.DiasDeLaSemana		90%		n/a	2	4	0	2	2	4	0	1
Sistema.Preferencias		87%		n/a	2	4	0	2	2	4	0	1
Sistema.Restricciones		84%		n/a	2	4	0	2	2	4	0	1
Sistema.IngestasPorDia		81%		n/a	2	4	0	2	2	4	0	1
Alimento		93%		90%	2	16	2	29	1	11	0	1
Ingesta		98%		83%	2	13	1	21	0	7	0	1
PlanAlimentacion		100%		75%	2	17	0	29	0	13	0	1
Profesional		100%		75%	2	12	0	23	0	8	0	1
ComposicionAlimento		100%		100%	0	6	0	12	0	5	0	1
Total	1,045 of 2,692	61%	97 of 236	59%	104	273	156	532	36	155	1	15

JaCoCoverage In-Depth Report

1.3. Análisis de Usabilidad

La usabilidad de un sistema es un atributo de calidad que permite evaluar que tan fácil de usar es este, considerando eficacia, eficiencia y satisfacción de usuario. Al ser dichas evaluaciones subjetivas a cada usuario utilizaremos las heurísticas de Nielsen las cuales nos dan una referencia clara. Utilizando el sistema de valoraciones de Nielsen también es posible cuantificar la usabilidad del sistema. Para esto se consideran 10 aspectos del sistema a evaluar, y a cada uno se le atribuye una puntuación de 0 a 4, representando 0 un buen uso de las practicas pautadas por las heurísticas de Nielsen y 4 un fallo grave en el principio heurístico. Aplicando dicho sistema al software a evaluar obtenemos lo siguiente:

■ Visibilidad del Estado del Sistema

El sistema siempre debe mantener al usuario informado de lo que ocurre, dando feedback en un tiempo razonable.

El sistema nos permite saber en que sección estamos mediante el cambio de color en los iconos, aunque el mismo podría ser mas explicito en los pasos intermedios de las funcionalidades (ej: aclarar que se debe elegir un profesional de la lista para crear una nueva consulta), o con los pasos a seguir cuando no hay alimentos y/o profesionales, ya que un usuario nuevo puede no saber que hacer en esa situación. También, existen bugs en el sistema afectan dicho factor, por ej: si ya habíamos solicitado un plan de alimentación y queremos solicitar otro, se sobre escribe el anterior y no se genera uno nuevo, sin aviso ni consulta hacia el usuario.

Calificación: 4

■ Vinculo entre el Sistema y el Mundo Real

El sistema debe utilizar un lenguaje que el usuario comprenda, con conceptos que le sean familiares. La información se presenta en un orden natural y lógico.

El sistema presenta un lenguaje familiar y sencillo de entender incluso para el usuario no profesional (ej: Consultas Pendientes, Ingresar un alimento, etc.), aunque el login presenta la información de forma confusa, donde el botón para registrarse no especifica su función, colocandose debajo de una lista de los usuarios registrados (que tampoco especifica que dicho lista son los usuarios registrados) donde uno esperaría el botón para iniciar sesión y no un registro.

Calificación: 1

■ Control y Libertad del Usuario

Se debe ofrecer soporte para que el usuario pueda deshacer o rehacer acciones sin miedo a equivocarse.

Faltan contra-partes para todas de las funciones del sistema, como (eliminar alimento, eliminar usuario/profesional, borrar/cancelar consulta, volver para atrás luego de querer crear una consulta con un profesional nueva en caso de haber seleccionado dicha función por error, entre otros).

Calificación: 4

■ Consistencia y Estándares

Los iconos, palabras deben tener un significado para el usuario y deben ser usados de forma consistente en todo el sistema siguiendo estándares.

El sistema utiliza terminología coherente y iconos universales para la mayoría de las funciones, aun así, el icono para registrar un profesional entre otros no logra distinguirse y podría ser mas explicito con algún otro icono (por ejemplo un birrete).

Calificación: 1

■ Prevención de Errores

Se debe tratar de minimizar la ocurrencia de errores, guiar al usuario para que en lo posible no cometa los mismos, en vez de simplemente marcarlos.

El sistema no muestra los campos obligatorios en los formularios de registro ni presenta el login como una accion de 2 pasos (seleccionar usuario, iniciar sesión), ya que nos podemos equivocar al seleccionar la lista y necesitaríamos volver al menú principal para recién ahí seleccionar el usuario nuevamente. Aun así se muestra la única extensión permitida desde el FileChooser para subir una imagen (png).

Calificación: 2

■ Reconocimiento sobre Memoria

Se debe priorizar el reconocimiento sobre la memoria, esto quiere decir minimizar la cantidad de cosas que el usuario debe recordar para utilizar el sistema.

El sistema podría especificar la funcionalidad a la que hace referencia cada icono con un label, así no tener que memorizar que hace cada uno de ellos. También se podría mostrar en pantalla el usuario/profesional el cual ha iniciado sesión.

Calificación: 2

■ **Flexibilidad y eficiencia de uso**

Se debe poder navegar por el sistema de forma intuitiva, así como proveer atajos para minimizar el tiempo gastado en navegar por el mismo.

El sistema provee una barra lateral con acceso a todas las funciones y un icono para volver al menú de inicio. A pesar de eso, no se permite cerrar el sistema desde otra pantalla que no sea el menú de inicio, así como la resolución de la interfaz gráfica no es óptima, en computadoras que no disponen de resoluciones altas no se logra acceder a botones/funciones del programa por lo que anula el uso de estos usuarios.

También resulta incomodo seleccionar una fecha de nacimiento ya que hay que regresar hacia atrás en los años manualmente haciendo click en un botón, en vez de elegir el año en una grilla.

Calificación: 3

■ **Diseño minimalista**

La interfaz no debe mostrar información que no sea estrictamente relevante o que no tenga utilidad, ya que esto disminuye la visibilidad de la información que si es relevante.

El sistema es minimalista, al punto en el cual como se menciono anteriormente, se podrían agregar labels a los iconos para explicitar sus funcionalidades. También, se podría utilizar una paleta de colores mas relacionada a la temática del sistema (blancos, negros y verdes por ejemplo) que también facilitaría la visualización del sistema en general, así como la satisfacción de un conjunto mayor de usuarios.

Calificación: 2

■ **Ayuda para Reconocer, Diagnosticar y Recuperarse de Errores**

Se debe dar un mensaje claro al usuario en caso de errores, sugiriendo un curso de acción para solucionar el problema.

El sistema da mensajes claros y precisos en los formularios, en caso de no haber ingresado la información necesaria, no encontramos otras fuentes de errores posibles.

Calificación: 0

▪ Ayuda y Documentación

Se debe proveer al usuario de una documentación fácil de encontrar, la cual debe estar centrada en las tareas del usuario. Esta debe ser concisa y contener los pasos a realizar por el usuario para determinada acción

El sistema provee ayuda al usuario y profesional una vez que el mismo inicia sesión, aun así, el acceso a dicha ayuda parece escondido, difícil de encontrar e inaccesible en algunas resoluciones de pantalla (se coloca en la parte derecha-inferior de la pantalla). También, no provee ayuda en la pantalla de login.

Calificación: 2

1.3.1. Resultado

Heurística	Calificación
Visibilidad del Estado del Sistema	4
Vínculo entre el Sistema y el Mundo Real	1
Control y Libertad del Usuario	4
Consistencia y Estándares	1
Prevención de Errores	2
Reconocimiento sobre Memoria	2
Flexibilidad y eficiencia de uso	3
Diseño minimalista	2
Ayuda para Reconocer, Diagnosticar y Recuperarse de Errores	0
Ayuda y Documentación	2
<u>Promedio de usabilidad inicial: 2,1</u>	

El resultado nos muestra que la usabilidad del sistema necesita ser mejorada, a pesar de no ser crítica (2-3 de promedio), no está cerca de ser óptima, por lo que clasificaríamos la misma con una prioridad de media-alta.

1.4. Análisis de la documentación

En esta sección se hará un análisis de la documentación del proyecto dado. En el mismo se reportaran nuevos errores encontrados que no se mencionan en la documentación, correctitud y formato del documento entre otros.

1.4.1. Resumen de log de versiones

En la documentación aparece una sección de resumen de log de versiones en la cual se lista el estado y características que tenia el software en cada versión específica. Se documenta desde la versión 0.0.1 a la 1.3.0, cuando en las siguientes secciones (por ejemplo sesiones de pruebas) se hace referencia a versiones 2.0.0, 3.0.0 entre otras, por lo que deja claro que hacen falta dichas versiones en la documentación

1.4.2. RF/RNF

La documentación no tiene una sección clara listando requerimientos funcionales sino que mezcla funcionalidades con 'casos de usos' (no correctos, ver siguiente sección) relacionados a el. En caso de analizarlos como si fueran requerimientos funcionales ignorando los casos de uso, les estaría haciendo falta una descripción clara del requerimiento, prioridad y numero para hacer referencia al mismo.

En cuanto a los requerimientos no funcionales, no hay nada similar a ello en la documentación.

1.4.3. Casos de uso

Como se menciona anteriormente, los 'casos de uso (referidos como casos de prueba en la documentación)' se encuentran mezclados con un lista de funcionalidades aun así les hace falta un nombre, descripción, actores, prioridad, post-condiciones, referencias a requerimientos funcionales y no funcionales, y cursos alternativos, ya que no hace referencia a un curso normal pero solo especifica la salida esperada con todas las acciones correctas de parte del usuario.

También, cabe destacar que en la documentación los casos de prueba empiezan en el punto 'D', cuando en otras secciones (por ejemplo sesiones de pruebas), se hace referencia a casos de prueba 1.A.1, 1.A.2, 1.B.1, 1.C.1 entre otros.

A continuación se listan los errores encontrados en los casos de uso mediante pruebas exploratorias:

- En el punto A de FUNCIONALIDADES (ROL DE USUARIO) hace falta una pre-condición en la cual especifique que es necesario tener alimentos registrados por un profesional en el sistema, o en su defecto un curso alternativo que documente que sucede en este curso.
- En los Casos 3 y 4 del punto A de FUNCIONALIDADES (ROL DE USUARIO) no queda claro si se puede enviar o no mensajes vacíos, aunque, analizando la funcionalidad y comparando con aplicaciones que tienen funcionalidades

similares (por ejemplo: Whatsapp) claramente no se debería poder, por lo tanto debería existir una pre-condición, o en su defecto un curso alternativo que documente que sucede en este curso.

- En el Caso 2 del punto C de FUNCIONALIDADES (ROL DE USUARIO) no tiene sentido comentar como NOTA que la tarea se puede realizar de forma iterativa dado que luego de realizarla una vez se navega a una pantalla que informa al usuario que la tarea fue exitosa y debe realizar todo el curso normal nuevamente, así como aclarar que no hay limite en las ingestas ingresadas por el usuario, ya que no es relevante en cuanto a un caso de uso, lo cual podría estar aclarado en un RF.
- El Caso 3 del punto C de FUNCIONALIDADES (ROL DE USUARIO) es incorrecto (no se puede dejar mas de un campo sin llenar) dado que solamente sucede este error cuando el usuario no llena el campo correspondiente al alimento. Si el mismo vacía intencionalmente el campo fecha (dado que tiene como valor por defecto el día de hoy) el error no ocurre, ya que el sistema lo ingresa igualmente con la fecha del día de hoy.
- En el Caso 3 del punto B de FUNCIONALIDADES (ROL PROFESIONAL) hace falta especificar que actualmente no es necesario llenar todos los días para poder finalizar la tarea exitosamente.

1.4.4. Sesiones de pruebas

Las sesiones de pruebas son claras y correctas, las mismas incluyen fecha, versión, entorno de prueba, duración y resultados concretos.

Aun así, encontramos un caso de uso, que a pesar de tener en cuenta la falta de versiones en el 'resumen de log de versiones', hace referencia a una funcionalidad que no esta presente en el producto entregado: 'Planes pendientes', por lo que la sesión de prueba no refleja ningún comportamiento que aplique al sistema analizado por nosotros. La prueba de uso relacionada es la siguiente:

Sesión 10: Asociada al caso de prueba 2.B.4

Fecha: 19/11/17

Versión: 3.0.0

Duración: 17 minutos

Entorno específico: 3

Resultado: Ingreso a la opción plan de alimentación del menú y presiono el botón para consultar los planes pendientes de ese usuario. Como no existe ninguno, se despliega en pantalla el respectivo cartel ...

1.4.5. Reporte de defectos

El reporte de defectos es correcto y claro, se titula y describe claramente cada defecto, teniendo una severidad definida y un estatus en cuanto a si fue realizada o

no. También se incluyen posibles mejoras a la usabilidad del sistema e interfaz de usuario.

A pesar de esto, encontramos algunos defectos que presentan inconsistencias entre los mimos y el comportamiento del sistema:

- En las posibles mejoras en relación a la usabilidad del sistema se comenta que fue realizada una mejora con respecto a la heurística de ayuda y documentación del sistema, pero, consideramos que esta mejora está parcialmente realizada dado que solamente agregan las ayudas en las pantallas de inicio de cada usuario y no en cada pantalla en particular.
- En los errores del sistema se comenta que existe un defecto no realizado que consiste en que un profesional no puede visualizar el segundo plan de alimentación del mismo usuario (previamente habiendo completado el primero), en cambio, testeando esto nos funciona correctamente.
- En las posibles mejoras en relación a la usabilidad se menciona una mejora relacionado a que se mantiene una fecha seleccionada en diferentes usuarios al entrar a una pantalla, además de que la descripción no es clara, tampoco logramos reproducirlo.
- En las posibles mejoras en relación a la usabilidad se menciona una mejora relacionado a colocar por defecto la fecha de hoy al intentar ingresar una ingesta, especifican que esto no fue realizado, pero, verificando notamos que si funciona de esta manera.

También encontramos varios errores nuevos (no listados en el reporte de defectos), los cuales se listaran en la sección de resumen de defectos.

1.5. Resumen de Defectos y Tareas Asociadas

A continuación se listaran los **nuevos** defectos encontrados y los derivados de las secciones anteriores, a los mismos se les asociara un código a cada defecto (ver glosario), prioridad (alta, media, baja), una descripción, una tarea asociada la cual apunta a solucionar dichos defectos y un código a cada tarea (ver glosario).

1.5.1. Analizadores de código

Código	Descripción	Prioridad
C1	Existen 23 bugs de baja prioridad encontrados mediante las distintas herramientas, entre estos se encuentran: <ul style="list-style-type: none">• Override hashCode() & Equals()• Add type test to Equals() method• Declare serialVersionUID to serializable classes• Usage of GetResource may be unsafe if class is extended	Baja
TC1	Se deberá eliminar al menos 18 bugs de baja prioridad de los 23 encontrados.	

Código	Descripción	Prioridad
C2	Existen 7 bugs de alta prioridad encontrados mediante las distintas herramientas, entre estos se encuentran: <ul style="list-style-type: none">• NullPointerException could be thrown• Useless self-assignment• Unwritten field• Use try-with-resources or close FileOutputStream in a finally clause	Alta
TC2	Se deberá eliminar al menos 4 bugs de alta prioridad de los 7 encontrados.	

Código	Descripción	Prioridad
C3	Existen 13 vulnerabilidades encontrados mediante las distintas herramientas, entre estos se encuentran: <ul style="list-style-type: none">• Make variable static final or non-public• May expose internal representation by returning/incorporating reference to mutable object	Media
TC3	Se deberá eliminar las 13 vulnerabilidades encontradas.	

Código	Descripción	Prioridad
C4	<p>Existen aproximadamente 500 code smells de baja prioridad encontrados mediante las distintas herramientas, entre estos se encuentran:</p> <ul style="list-style-type: none"> • Unused imports • Use utility class • Loose Coupling • Unncessary local before return • Idempotent operations • Unnecessary fully qualified name • For loop can be for each • Poisition literals first in comparisons • Close resource • Useless Parentheses • Unused formal parameter • Useless overriding method • Package Case • Singular Field • Unused local variable 	Baja
TC4	Se deberá eliminar al menos 250 de los 500 code smells de baja prioridad encontrados.	

Código	Descripción	Prioridad
C5	<p>Existen aproximadamente 80 code smells de alta prioridad encontrados mediante las distintas herramientas, entre estos se encuentran:</p> <ul style="list-style-type: none"> • Empty catch block • Redundant null check • Method with more than 7 parameters • Merge if statement with enclosing one • Remove expression which always evaluates to true • Rename constant to match regular expression • Formal parameter naming conventions • Method naming conventions 	Alta
TC5	Se deberá eliminar al menos 40 de los 80 code smells de alta prioridad encontrados.	

1.5.2. Cobertura de pruebas unitarias

Código	Descripción	Prioridad
P1	Existe una cobertura de sentencias del 61 % y 57 % de cobertura de ramas, lo cual no es aceptable frente al estándar del 90 %.	Alta
TP1	Se deberá mejorar la cobertura de pruebas unitarias hasta al menos un 90 %.	

Código	Descripción	Prioridad
P2	Existen 14 tests que fallan y 15 que generan errores.	Alta
TP2	Se deberá solucionar los 14 tests que fallan y los 15 que generan errores.	

1.5.3. Usabilidad

Código	Descripción	Prioridad
U1	Falta de especificación en iconos, acciones a seguir/acciones intermedias.	Alta
TU1	Se deberá agregar una breve explicación de que acción se espera del usuario tome en todos los cursos alternativos, así como especificar con labels la funcionalidad de cada icono.	

Código	Descripción	Prioridad
U2	Faltan contra partes de todas las funcionalidades del sistema.	Alta
TU2	Se propone una solución parcial la cual implemente la posibilidad de revertir (eliminando) alimentos ingerido y alimentos ingresados al sistema, estas dos funcionalidades a nuestro entender son las mas usadas y propicias a errores.	

Código	Descripción	Prioridad
U3	El programa no presenta datos existentes al iniciarse, lo que dificulta su uso teniendo que ingresar todo manualmente, incluso los alimentos.	Media
TU3	Se deberá poder cargar alimentos de consumo diario al sistema sin la necesidad de que un profesional los provea, así como un usuario y profesional 'demo' para facilitar el uso del sistema.	

Código	Descripción	Prioridad
U4	El icono de registrar a un profesional no representa su funcionalidad.	Baja
TU4	Se deberá cambiar el icono de registrar un profesional, se propone el uso de un birrete con un '+' como icono.	

Código	Descripción	Prioridad
U5	No se muestran los campos obligatorios en los formularios de registro.	Baja
TU5	Se deberá mostrar un '*' rojo al lado de los campos obligatorios en los formularios.	

Código	Descripción	Prioridad
U6	El login no presenta la información en un orden natural/lógico.	Media
U7	No existe una seleccion de usuario/profesional seguida de un boton de login, fomenta la chance de errores al registrarse como un usuario/profesional.	Baja
TU6	Se deberá reordenar los campos en esta pantalla siguiendo los estándares de diseño de pantallas de onboarding.	

Código	Descripción	Prioridad
U8	No se muestra en ninguna parte del sistema el nombre del usuario/profesional con el cual se realizo el inicio de sesión.	Baja
TU7	Se deberá mostrar el nombre del usuario/profesional con el cual se realizo el inicio de sesión.	

Código	Descripción	Prioridad
U9	La resolución de la pantalla es única y no satisface a todas las computadoras, haciendo difícil el uso y prueba del sistema dado que puede dejar fuera funcionalidades del sistema.	Alta
TU8	Se deberá poder elegir entre dos resoluciones.	

Código	Descripción	Prioridad
U10	El calendario provisto no es optimo/cómodo al elegir la fecha de nacimiento, en la cual se debe ir hacia atrás en los años manualmente haciendo click en un botón, en vez de mostrar una grilla con los años a elegir.	Baja
TU9	Se deberá indagar sobre las distintas librerías de calendarios disponibles para java y en caso de encontrar otra mas óptima cambiar la actual.	

Código	Descripción	Prioridad
U11	La paleta de colores elegida no facilita la visualización del sistema ni transmite de lo que trata el sistema.	Baja
TU10	Se deberá cambiar la paleta de colores de acorde a la temática de alimentación saludable (por ejemplo blanco, verde y negro).	

Código	Descripción	Prioridad
U12	La ayuda que se provee al usuario es difícil de encontrar, además no se presenta ayuda en la sección del login.	Alta
TU11	Se deberá cambiar la posición de el icono de ayuda o resaltarlo, también se deberá extender dicha ayuda al menú del login y registro.	

1.5.4. Documentación

Código	Descripción	Prioridad
D1	Se hacen referencias en la documentación a versiones que no aparecen en el resumen de log de versiones.	Alta
TD1	Se deberá eliminar el log de versiones y las referencias al mismo, ya que no aporta ni disponemos de la información necesaria para completarlo.	

Código	Descripción	Prioridad
D2	No se especifican claramente los RF, listando algunas funcionalidades del sistema mezcladas con 'casos de uso' de forma no correcta.	Media
TD2	Se deberá extraer las funcionalidades del sistema existentes y documentar apropiadamente cada una de ellas.	

Código	Descripción	Prioridad
D3	No se especifican los RNF.	Alta
TD3	Se deberá agregar las RNF que identifique el equipo que el sistema debería cumplir, ya que las RNF iniciales a las cuales los desarrolladores de la aplicación apuntaban no están documentadas.	

Código	Descripción	Prioridad
D4	Los 'casos de uso' embebidos en el listado de funcionalidades no muestran un formato correcto, en donde les hace falta un numero, nombre, descripción, actores, prioridad, post-condiciones, referencias a RF y RNF y cursos alternativos.	Media
TD4	Se deberá documentar los casos de uso existentes y los nuevos identificados.	

Código	Descripción	Prioridad
D5	Existen 5 casos de uso que presentan inconsistencias entre el mismo y el comportamiento del sistema (Listados en la sección de Casos de Uso de Análisis de la Documentación).	Alta
TD5	Se deberán eliminar dichas inconsistencias y formular correctamente dichos casos de uso.	

Código	Descripción	Prioridad
D6	En las sesiones de prueba se hace referencia a una funcionalidad que no esta presente en el proyecto entregado.	Baja
TD6	Se deberán quitar dicha sesión de prueba de la documentación.	

Código	Descripción	Prioridad
D7	Existen 4 defectos listados en la documentación que presentan inconsistencias entre el mismo y el comportamiento del sistema (Listados en la sección de Reporte de Defectos de Analisis de la Documentacion).	Alta
TD7	Se deberán quitar y/o modificar los defectos listados en la documentacion que presenten inconsistencias.	

1.5.5. Nuevos defectos encontrados

Código	Descripción	Prioridad
N1	NullPointerException al agregar un alimento ingerido cuando ejecutamos el proyecto desde Netbeans, pero no en el ejecutable.	Alta
TN1	Se deberá solucionar el NullPointerException que se genera al agregar un alimento ingerido cuando se ejecuta el el programa desde NetBeans.	

Código	Descripción	Prioridad
N2	Si no hay alimentos registrados y se solicita un plan de alimentación, no aparece un error y no le llega la solicitud a los profesionales.	Alta
TN2	Se deberá notificar al usuario al solicitar un plan de alimentación si es que no existen alimentos registrados en el sistema.	

Código	Descripción	Prioridad
N3	Si solicitamos más de un plan de alimentación con el mismo usuario antes que nos acepten el primero se sobre-escribe el mismo.	Alta
TN3	Se deberá permitir solicitar solo un plan de alimentación de parte del usuario mientras que el mismo no sea completado y enviado por el profesional. Luego de que el profesional complete y envíe el plan de alimentación solicitado, el usuario podrá solicitar uno nuevo.	

Código	Descripción	Prioridad
N4	Siendo un profesional, al seleccionar uno de los planes solicitados por un usuario se muestran los datos del mismo a la derecha, pero si apretamos en otra sección del menú y volvemos, los datos a la derecha aparecen incompletos (falta fecha de nacimiento y nombre).	Baja
TN4	Se deberá evitar la pérdida de información al cambiar de una ventana a otra en la sección de planes de alimentación solicitados.	

Código	Descripción	Prioridad
N5	En la vista previa del plan de alimentación solicitado, en la lista de ingestas aparece el primer alimento ingerido por el usuario múltiples veces, a pesar de haber ingerido varios distintos, cuando se realiza el plan eso se arregla. También ocurre lo mismo al visualizar el perfil del usuario desde las consultas pendientes.	Alta
TN5	Se deberá poder visualizar correctamente los alimentos ingeridos por el usuario en cualquier parte del sistema.	

Código	Descripción	Prioridad
N6	Si hay alimentos registrados, cuando un usuario trata de ingresar un alimento ingerido, si no se llena ningún campo y presiona crear, el sistema muestra una pantalla que indica que no hay alimentos registrados, en vez de mostrar que no se completaron los campos necesarios.	Alta
TN6	Cuando se presiona registrar alimento ingerido sin seleccionar ningún alimento, a pesar de existir varios registrados, se deberá notificar al usuario de los campos no llenados en lugar de mostrar que no hay alimentos registrados.	

1.5.6. Mixtos

Código	Descripción	Prioridad
U13	Al subir una imagen ya sea para en el ingreso de alimentos, registro de usuario o profesional, la misma se recorta de una forma no deseada.	Media
N8	Al visualizar un usuario (con una imagen asociada) en consultas pendientes la imagen del mismo no se ve correctamente, recortándose de forma distinta que en las otras secciones (registro de usuario, planes de alimentación pendientes).	Media
N9	Al visualizar un profesional (con una imagen asociada) en consulta con profesional la imagen del mismo no se ve correctamente, recortándose de forma distinta que en las otras secciones (registro de profesional).	Media
N10	Al visualizar un usuario (sin una imagen asociada) en consultas pendientes el sistema nos muestra un recuadro blanco sin significado, lo cual no especifica que dicho usuario no tiene una imagen asociada.	Baja
N11	Al visualizar un usuario (con una imagen asociada) en planes de alimentación pendientes, en caso de que la imagen tenga un tamaño 'grande', la misma se vera recortada y ocupando todo el espacio donde se deberían mostrar los datos del usuario y la opción para crear el plan de alimentación, dejando así dicha pantalla sin uso.	Alta
TUN1	Se deberá poder visualizar la imagen sin importar su tamaño o hacer un recorte que se mantenga consistente a través de la aplicación.	

Código	Descripción	Prioridad
U14	El orden de los campos al agregar un plan de alimentación de parte de un profesional no es lógico.	Media
N7	Como profesional se permite enviar un plan de alimentación sin completar todos los días, esto no refleja la necesidad de la funcionalidad y tampoco lo especificado.	Alta
TUN2	Se deberá reestructurar y agregar controles a dicha pantalla para que sea mas clara y ordenada, facilitando su uso y permitiendo que el profesional pueda saber que dias ya fueron completados y cuales no.	

1.5.7. Glosario de Codificación de los Defectos y Tareas

Los códigos asignados a cada defecto siguen la siguiente nomenclatura:

- **C** para los defectos relacionados al análisis del código.
- **P**[número] para los defectos relacionados a las pruebas unitarias.
- **U** para los defectos relacionados al análisis de usabilidad.
- **D** para los defectos relacionados al análisis de la documentación.
- **N** para los nuevos defectos encontrados en las funcionalidades del sistema.
- **T**[área/s asociadas] para las tareas propuestas como solución a los defectos.

2. Análisis de cambios

En este capítulo definiremos las solicitudes de cambio y analizaremos el impacto que generan los mismos, lo cual luego será de gran ayuda a la hora de estimar los costos correspondientes.

2.1. Solicitudes de cambio

A continuación se listarán los cambios solicitados por el cliente y los derivados de las tareas/defectos definidas en el capítulo anterior.

Numero de solicitud	1
Origen	Cliente
Tipo	<input checked="" type="checkbox"/> Nuevo requisito <input type="checkbox"/> Cambio en el requisito <input type="checkbox"/> Cambio en el diseño <input type="checkbox"/> Problemas en el software <input type="checkbox"/> Error en la documentación <input type="checkbox"/> Sugerencia de mejora <input type="checkbox"/> Otro
Prioridad	<input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Descripción	Se debe poder salir de la aplicación en cualquiera de sus ventanas
Estado	Aprobado
Fecha de Aprobación	4/20/2020

Numero de solicitud	2
Origen	Cliente, TU7
Tipo	<input checked="" type="checkbox"/> Nuevo requisito <input type="checkbox"/> Cambio en el requisito <input type="checkbox"/> Cambio en el diseño <input checked="" type="checkbox"/> Problemas en el software <input type="checkbox"/> Error en la documentación <input type="checkbox"/> Sugerencia de mejora <input type="checkbox"/> Otro
Prioridad	<input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Descripción	La única resolución disponible no es compatible con todas las computadoras, dejando funcionalidades del sistema inaccesibles al usuario. Se debe poder elegir entre dos resoluciones.
Estado	Aprobado
Fecha de Aprobación	4/20/2020

Numero de solicitud	3
Origen	Cliente
Tipo	<input type="checkbox"/> Nuevo requisito <input type="checkbox"/> Cambio en el requisito <input type="checkbox"/> Cambio en el diseño <input type="checkbox"/> Problemas en el software <input type="checkbox"/> Error en la documentación <input checked="" type="checkbox"/> Sugerencia de mejora <input type="checkbox"/> Otro
Prioridad	<input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Descripción	Las fechas a lo largo de los distintos formularios de la aplicación no son validadas, por lo tanto por defecto muestran el día de hoy. Se deberá agregar validaciones en las fechas de nacimiento, graduación, entre otras.
Estado	Aprobado
Fecha de Aprobación	4/20/2020

Numero de solicitud	4
Origen	Cliente, TUN2
Tipo	<input type="checkbox"/> Nuevo requisito <input type="checkbox"/> Cambio en el requisito <input type="checkbox"/> Cambio en el diseño <input checked="" type="checkbox"/> Problemas en el software <input type="checkbox"/> Error en la documentación <input checked="" type="checkbox"/> Sugerencia de mejora <input type="checkbox"/> Otro
Prioridad	<input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Descripción	Se deberá reestructurar y agregar controles a dicha pantalla para que sea mas clara y ordenada, facilitando su uso y permitiendo que el profesional pueda saber que dias ya fueron completados y cuales no.
Estado	Aprobado
Fecha de Aprobación	4/20/2020

Numero de solicitud	5
Origen	Cliente, TUN1
Tipo	<input type="checkbox"/> Nuevo requisito <input type="checkbox"/> Cambio en el requisito <input type="checkbox"/> Cambio en el diseño <input checked="" type="checkbox"/> Problemas en el software <input type="checkbox"/> Error en la documentación <input type="checkbox"/> Sugerencia de mejora <input type="checkbox"/> Otro
Prioridad	<input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Descripción	Se deberá solucionar subir el recorte indeseado de imágenes que excedan determinado tamaño en el perfil de Usuario, Profesional e imágenes de alimentos.
Estado	Aprobado
Fecha de Aprobación	4/20/2020

Numero de solicitud	6
Origen	Cliente
Tipo	<input type="checkbox"/> Nuevo requisito <input checked="" type="checkbox"/> Cambio en el requisito <input type="checkbox"/> Cambio en el diseño <input type="checkbox"/> Problemas en el software <input type="checkbox"/> Error en la documentación <input checked="" type="checkbox"/> Sugerencia de mejora <input type="checkbox"/> Otro
Prioridad	<input type="checkbox"/> Alta <input type="checkbox"/> Media <input checked="" type="checkbox"/> Baja
Descripción	Se deberá agregar la posibilidad de subir imágenes en formato jpg.
Estado	Aprobado
Fecha de Aprobación	4/20/2020

Numero de solicitud	7
Origen	Cliente
Tipo	<input type="checkbox"/> Nuevo requisito <input type="checkbox"/> Cambio en el requisito <input type="checkbox"/> Cambio en el diseño <input type="checkbox"/> Problemas en el software <input type="checkbox"/> Error en la documentación <input checked="" type="checkbox"/> Sugerencia de mejora <input type="checkbox"/> Otro
Prioridad	<input type="checkbox"/> Alta <input type="checkbox"/> Media <input checked="" type="checkbox"/> Baja
Descripción	Se mejorara la interfaz gráfica del filechooser provisto al subir una imagen al sistema, dado que esté no tiene una UI acorde al programa, ni al sistema operativo en el cual se está ejecutando.
Estado	Aprobado
Fecha de Aprobación	4/20/2020

Numero de solicitud	8
Origen	TC1 - TC5
Tipo	<input type="checkbox"/> Nuevo requisito <input type="checkbox"/> Cambio en el requisito <input type="checkbox"/> Cambio en el diseño <input checked="" type="checkbox"/> Problemas en el software <input type="checkbox"/> Error en la documentación <input checked="" type="checkbox"/> Sugerencia de mejora <input type="checkbox"/> Otro
Prioridad	<input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Descripción	Se deberá mejorar la calidad global del código, eliminando los bugs, vulnerabilidades y code smells correspondientes a cada tarea.
Estado	Aprobado
Fecha de Aprobación	4/20/2020

Numero de solicitud	9
Origen	TP1, TP2
Tipo	<input type="checkbox"/> Nuevo requisito <input type="checkbox"/> Cambio en el requisito <input type="checkbox"/> Cambio en el diseño <input checked="" type="checkbox"/> Problemas en el software <input type="checkbox"/> Error en la documentación <input checked="" type="checkbox"/> Sugerencia de mejora <input type="checkbox"/> Otro
Prioridad	<input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Descripción	Se deberá mejorar la cobertura de pruebas unitarias al menos a un 90 % y corregir las que generan errores o fallan.
Estado	Aprobado
Fecha de Aprobación	4/20/2020

Numero de solicitud	10
Origen	TU1 - TU11
Tipo	<input type="checkbox"/> Nuevo requisito <input type="checkbox"/> Cambio en el requisito <input type="checkbox"/> Cambio en el diseño <input checked="" type="checkbox"/> Problemas en el software <input type="checkbox"/> Error en la documentación <input checked="" type="checkbox"/> Sugerencia de mejora <input type="checkbox"/> Otro
Prioridad	<input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Descripción	Se deberá mejorar la usabilidad global del sistema, siguiendo las indicaciones de las tareas asociadas.
Estado	Aprobado
Fecha de Aprobación	4/20/2020

Numero de solicitud	11
Origen	TD1 - TD7
Tipo	<input type="checkbox"/> Nuevo requisito <input type="checkbox"/> Cambio en el requisito <input type="checkbox"/> Cambio en el diseño <input type="checkbox"/> Problemas en el software <input checked="" type="checkbox"/> Error en la documentación <input type="checkbox"/> Sugerencia de mejora <input type="checkbox"/> Otro
Prioridad	<input type="checkbox"/> Alta <input checked="" type="checkbox"/> Media <input type="checkbox"/> Baja
Descripción	Se deberá renovar la documentación actual del sistema, siguiendo las indicaciones de las tareas asociadas.
Estado	Aprobado
Fecha de Aprobación	4/20/2020

Numero de solicitud	12
Origen	TN1 - TN6
Tipo	<input type="checkbox"/> Nuevo requisito <input type="checkbox"/> Cambio en el requisito <input type="checkbox"/> Cambio en el diseño <input checked="" type="checkbox"/> Problemas en el software <input type="checkbox"/> Error en la documentación <input type="checkbox"/> Sugerencia de mejora <input type="checkbox"/> Otro
Prioridad	<input checked="" type="checkbox"/> Alta <input type="checkbox"/> Media <input type="checkbox"/> Baja
Descripción	Se deberá mejorar la confiabilidad del sistema, eliminando los defectos encontrados mediante las indicaciones de las tareas asociadas.
Estado	Aprobado
Fecha de Aprobación	4/20/2020

2.2. Análisis de Impacto

A continuación se listan los análisis de impacto de cada solicitud de cambio correspondiente. Es importante aclarar, que algunas solicitudes de cambio (en específico la n°9 y n°11) no impactaban en ninguna funcionalidad existente del sistema (n°11) o no era posible medir su impacto objetivamente antes de empezar con el proyecto de mantenimiento, ya que podría afectar a todas como no afectar a ninguna (n°9). Esto ultimo se debe al impacto de los de los otros cambios en las funcionalidades nuevas y existentes lo cual afectaría el análisis de cobertura y quizás invalide tests existentes.

Solicitud n° 1	Funciones del producto de software																
Nivel de documentación	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17
Espec. de Req.																	X
Espec. de Diseño																	X
Casos de Prueba																	X
Código Fuente																	X

F17 hace referencia a la nueva funcionalidad la cual se menciona en la solicitud de cambio: Se debe poder salir de la aplicación en cualquier de sus ventanas.

Solicitud n° 2	Funciones del producto de software																
Nivel de documentación	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F18
Espec. de Req.																	X
Espec. de Diseño																	X
Casos de Prueba																	X
Código Fuente																	X

F18 hace referencia a la nueva funcionalidad la cual se menciona en la solicitud de cambio: Se debe poder elegir entre una resolución mas y la existente.

Solicitud nº 3	Funciones del producto de software															
Nivel de documentación	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16
Espec. de Req.	X	X							X							
Espec. de Diseño																
Casos de Prueba	X	X							X							
Código Fuente	X	X							X							

Solicitud nº 4	Funciones del producto de software															
Nivel de documentación	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16
Espec. de Req.											X					
Espec. de Diseño											X					
Casos de Prueba											X					
Código Fuente											X					

Solicitud nº 5	Funciones del producto de software															
Nivel de documentación	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16
Espec. de Req.																
Espec. de Diseño																
Casos de Prueba																
Código Fuente	X	X						X							X	

Solicitud nº 6	Funciones del producto de software															
Nivel de documentación	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16
Espec. de Req.	X	X						X								
Espec. de Diseño																
Casos de Prueba																
Código Fuente	X	X						X								

Solicitud nº 7	Funciones del producto de software															
Nivel de documentación	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16
Espec. de Req.																
Espec. de Diseño																
Casos de Prueba																
Código Fuente	X	X						X								

Solicitud nº 8	Funciones del producto de software															
Nivel de documentación	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16
Espec. de Req.																
Espec. de Diseño																
Casos de Prueba																
Código Fuente	X	X	X	X				X			X	X				X

Para el análisis de impacto anterior (nº8) se utilizaron las mismas herramientas de análisis de código mencionadas en el primer capítulo para encontrar las clases las cuales contenían dichos bugs/vulnerabilidades/code smells para así obtener un análisis mas preciso.

Solicitud nº 10	Funciones del producto de software																	
Nivel de documentación	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F19/20	F21
Espec. de Req.																	X	X
Espec. de Diseño																	X	X
Casos de Prueba								X									X	X
Código Fuente		X	X	X				X									X	X

F19/20 hace referencia a una nueva funcionalidad la cual esta incluida en las tareas asociadas a la solicitud de cambio: Se debe poder borrar alimentos ingeridos y alimentos registrados.

F21 hace referencia a una nueva funcionalidad la cual esta incluida en las tareas asociadas a la solicitud de cambio: Se debe poder cargar un set de datos de prueba para utilizar el sistema sin necesidad de ingresar todo a mano.

Solicitud n° 12	Funciones del producto de software															
Nivel de documentación	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16
Espec. de Req.																
Espec. de Diseño																
Casos de Prueba									X	X						
Código Fuente									X	X	X			X	X	

2.3. Glosario de Funcionalidades

El siguiente listado hace referencia a las funcionalidades nuevas y existentes en el sistema. Las funcionalidades existentes fueron extraídas mediante pruebas exploratorias, ya que en la documentación, como explica el defecto D2, solo se mencionan alguna de ellas.

- F1: Registrar Usuario
- F2: Registrar Profesional
- F3: Iniciar Sesión como Usuario
- F4: Iniciar Sesión como Profesional
- F5: Crear Consulta con profesional
- F6: Enviar Mensaje a un profesional
- F7: Enviar mensaje a un usuario
- F8: Ingresar nuevo alimento
- F9: Ingresar alimento ingerido
- F10: Solicitar un plan de alimentación
- F11: Crear plan de alimentación
- F12: Enviar plan de alimentación
- F13: Ver los planes de alimentación disponibles
- F14: Ver consultas pendientes
- F15: Ver perfil de usuario/profesional
- F16: Persistencia de datos (Serialización)
- **F17**: Se debe poder salir de la aplicación en cualquier de sus ventanas

- [F18](#): Se debe poder elegir entre una resolución mas y la existente
- [F19/20](#): Se debe poder borrar alimentos ingeridos y alimentos registrados
- [F21](#): Se debe poder cargar un set de datos de prueba para utilizar el sistema sin necesidad de ingresar todo a mano

3. Proyecto de mantenimiento

Luego de recaudar toda la información necesaria en los capítulos anteriores, procederemos a plantear nuestro proyecto de mantenimiento.

En este capítulo identificaremos los interesados y formularemos un plan de gestión para los mismos, luego definiremos los objetivos y alcance del producto para obtener un presupuesto inicial y un cronograma a seguir en la próxima etapa del trabajo. Finalizaremos definiendo el plan de calidad, métricas y el ciclo de vida del proyecto a seguir.

Cabe destacar que realizaremos el trabajo de mantenimiento en GitHub utilizando la metodología de trabajo GitFlow. En el zip podrán encontrar también las imágenes y reportes incluidos en la documentación en caso de querer visualizar mejor alguna de ellas.

(Link al repositorio)

3.1. Identificación de interesados

Luego de analizar el mercado, logramos identificar ciertos interesados los cuales presentaremos en la siguiente tabla. La misma incluirá la información del mismo, interés, poder y predecibilidad, tipo de interés (negativo o positivo) y una tarea asociada (ver codificación a continuación de la tabla).

Interesados	Interés	Poder	Predecibilidad	Tipo de Interés	Tarea
Personas con problemas de salud	Alto	Alto	Alto	Positivo	1
Personas sin problemas de salud	Medio-Bajo	Medio-Bajo	Medio-Bajo	Positivo	3
Deportistas que quieren mejorar su dieta	Medio-Bajo	Medio-Alto	Medio-Alto	Positivo	2
Nutricionistas o afines sin trabajo	Medio-Bajo	Alto	Media-Alto	Positivo	2
Nutricionistas o afines con trabajo	Medio-Alto	Medio-Alto	Medio-Bajo	Positivo	1
Cliente	Alto	Alto	Alto	Positivo	1
Mutualistas/Consultoras	Bajo	Alto	Bajo	Negativo	3
Equipo	Alto	Alto	Alto	Positivo	1

Codificación en la tarea de los interesados:

- **1:** Involucrar y atraer activamente
- **2:** Involucrar y mantener satisfechos
- **3:** Mantener informados
- **4:** Monitorear (esfuerzo mínimo)

A continuación se pasará a especificar más en detalle cada interesado y explicar los datos de la tabla anterior.

- **Personas con problemas de salud:**

Las personas con problemas de salud son los principales consumidores de nuestro producto dado que al tener restricciones alimenticias probablemente necesiten solicitar un plan de alimentación y hacer uso de la mayoría de las funcionalidades del sistema. Por esto mismo, si estas personas se encuentran sin posibilidad de concurrir a un centro médico ya sea por motivo económico, falta de tiempo, viaje entre otros, la no-necesidad de concurrir a una mutualista, coloca a nuestra plataforma como una buena alternativa. Por esto mismo se califico a estas personas con un interés/poder/predecibilidad Alto.

- **Personas sin problemas de salud:** Estos usuarios tienen interés/poder/-predecibilidad Medio-Bajo ya que podrían dejar de usar el sistema al no necesitarlo de la misma forma que otros interesados o desconocer de el mismo en primer lugar, aun así nos interesa su opinión para agregarle valor al sistema y obtener la atención de los mismos.
- **Deportistas que quieren mejorar su dieta:** Estos usuarios o posibles usuarios tienen un interés Medio-Bajo ya que deben concurrir a un profesional desde hace tiempo o propio del centro de deporte donde entrenan.
El poder Medio-Alto esta dado por su influencia en las personas, ya que un deportista utilice nuestra plataforma agrega valor y confiabilidad a la misma. Por ultimo, la predecibilidad Medio-Alto se debe a que al menos que nuestro sistema le aporte algún beneficio a dichos interesados, no presentarían motivos para utilizarlo.
- **Nutricionistas o afines sin trabajo:** Estos tienen interés Medio-Bajo dado que al estar sin trabajo la opción de usar el sistema no debería ser de su interés ya que no genera ningún ingreso ni experiencia verificable. Tienen un poder Alto ya que mediante encuestas o entrevistas remuneradas son nuestra principal fuente de información del lado de los profesionales. Su predecibilidad Medio-Alto se debe a que no tienen motivos para no ser parte de dichas encuestas/entrevistas.
- **Nutricionistas o afines con trabajo:** Estos tienen interés/poder Medio-Alto ya que consideramos que al estar trabajando posee de ingresos fijos y disponen de la posibilidad de ayudar a otras personas a voluntad mediante nuestra plataforma. Aun así, como muestra la predecibilidad la misma es Medio-Bajo, ya que depende de cada uno si desea ser parte de la plataforma y ayudar a voluntad.
- **Cliente:** El cliente es el más interesado, ya sea para mejorar el producto, captar mas usuarios o mantener felices a los actuales.
Tiene un poder Alto al ser el inversor en este proyecto y una predecibilidad Alta dado que ya sabemos sus expectativas y la duración proyecto.
- **Mutualistas/Consultoras:** Estos tienen interés Bajo ya que nuestro producto de momento no presenta ningún riesgo a su base de clientes ni tampoco agrega valor a sus organizaciones. Su poder Alto se debe a la posibilidad de prohibir el uso de nuestro sistema dentro de su organización y profesionales. Predecibilidad Bajo ya que a pesar de no ser competencia ni riesgo en el momento, no sabemos sus planes.
- **Equipo:** Nosotros como equipo tenemos interés en el proyecto y deseamos que sea exitoso, tenemos poder Alto ya que nuestras decisiones y sugerencias pueden influir en el resultado y tenemos una predecibilidad Alta ya que al ser un equipo pequeño hay más posibilidad de que estemos de acuerdo en la mayoría de las decisiones.

3.2. Gestión de interesados

Dada la siguiente matriz de evaluación de participación daremos lugar a una descripción breve sobre como involucrar a cada uno de los interesados en el proyecto.

Interesados	Desconocedor	Reticente	Neutral	Partidario	Líder
Usuarios con problemas de salud				D C	
Usuarios sin problemas de salud	C			D	
Deportistas que quieren mejorar su dieta			C	D	
Nutricionistas o afines sin trabajo			C		D
Nutricionistas o afines con trabajo			C	D	
Cliente					D C
Mutualistas/Consultoras		C		D	
Equipo					D C

- **Personas con problemas de salud:**

Las personas con problemas de salud son los principales consumidores de nuestro producto, por ello se les deberá informar e involucrar constantemente para evaluar los cambios en el software (nuevas funcionalidades, cambios en la interfaz gráfica entre otros) y así poder identificar posibles mejoras. Así se procura lograr que estos mismos se mantengan como nuestro publico principal y como consecuencia incentivar a mas profesionales a unirse a la plataforma.

- **Personas sin problemas de salud:**

Estas personas al no tener una necesidad clara son poco concurrentes a la plataforma o desconocedores de la misma. Por esto se los informará de forma no frecuente, para poder captar atención de posibles usuarios en algunos casos. Agregar funcionalidades que se adapten a sus necesidades o campañas de marketing se muestran como posibles opciones pero quedan por fuera de este proyecto de mantenimiento.

- **Deportistas que quieren mejorar su dieta:**

Los deportistas que quieren mejorar su dieta tienen motivo para usar nuestra plataforma, aun así no es la única opción existente, por esto mismo se recomienda involucrarlos activamente luego de la finalización de este proyecto de mantenimiento, ya que a diferencia de otras personas con problema de salud, usualmente concurren a un centro de deportes los cuales ya presentan profesionales a cargo. Por esto mismo, aunque sean neutrales al producto, luego del proyecto de mantenimiento se informara a los mismos sobre la plataforma, y

así involucrarlos en un posible proyecto futuro con una adaptación del sistema a los mismos en mente.

- **Nutricionistas o afines sin trabajo:**

Al no estar actualmente trabajando, la propuesta de utilizar nuestro sistema no se muestra como una opción viable ya que no genera ingresos. Aun así, su opinión, ya sea con entrevistas o encuestas pagas son una fuerte fuente de información para la mejora del producto, por esto mismo se les comunicará de forma frecuente para evaluar cambios en el software y posibles mejoras en el mismo. También recomendamos para un futuro proyecto, evaluar la forma de generar valor ya sea generando experiencia verificable como profesional o ingreso (quizás con propagandas) para que estos profesionales se conviertan en posibles usuarios concurrentes del sistema.

- **Nutricionistas o afines con trabajo:**

Al estar trabajando actualmente, debemos captar la atención dentro de este grupo de interesados a los que deseen ayudar a las personas que ya sea por motivo económico, falta de tiempo, viaje entre otros, no puedan asistir a un centro médico. Por esto mismo se les comunicará sobre cambios en el software y datos sobre la cantidad de usuarios (no profesionales) ya registrados, con la intención de poder captar la atención de los mismos y en lo posible obtener una base de profesionales registrados mayor.

- **Cliente:**

Al cliente se lo involucrara semana a semana para evaluar el avance en el proyecto, validar los entregables y discutir posibles mejoras/conflictos.

- **Mutualistas/Consultoras:**

Al no presentar un riesgo para estas mismas actualmente , no se planea informar ni involucrar a estas organizaciones en el proyecto, pero si proveer información en caso de ser solicitada. Aun así se recomienda a futuro evaluar la posibilidad de agregar valor al sistema de forma tal que los profesionales trabajando para dicha organización puedan atender a pacientes de dicha organización en forma remota y estas mismas pasen a ser interesados positivos y posibles inversores.

- **Equipo:**

Dentro del equipo se realizara reuniones semanales para evaluar el avance del proyecto y resolver cualquier posible desacuerdo.

3.3. Definición de objetivos SMART del proyecto

A continuación se listarán los objetivos SMART del proyecto.

- Corregir para el final del proyecto al menos un 75 % de las tareas que se especificaron en el primer capítulo en 'Resumen de Defectos y Tareas Asociadas'.
- Tener menos de 15 regresiones a lo largo del proyecto. Con regresiones nos referimos a funcionalidades que estaban funcionando o que en algún momento arreglamos y vuelvan a fallar de un momento a otro.
- Involucrar al cliente al final de cada entregable para recibir un feedback del mismo.
- En ningún momento del proyecto comprometer la calidad del código por la necesidad de llegar a tiempo a una fecha. De esta forma nos aseguraremos que en el futuro ya seamos nosotros u otro equipo, el código estará en un buen estado para seguir trabajando en él.

3.4. Roles

A continuación se listan los roles del equipo de mantenimiento junto al costo por hora correspondiente:

- Product Manager - 20 US\$ hora
- Desarrollador Semi-Senior - 15 US\$ hora
- QA Semi-Senior - 12 US\$ hora

El product manager es el responsable de la estrategia, planificación, ejecución y lanzamiento de un producto. Durante el desarrollo al final de cada entregable será el encargado de gestionar el avance, comunicarse con el cliente, identificar posibles problemas y situaciones que bloquean el avance, ya sea técnicas o requerimientos formulados incorrectamente, de esta forma se buscará asegurar el correcto cumplimiento de los objetivos del proyecto.

El desarrollador será el encargado de implementar las funcionalidades, bugs y mejoras de calidad del código en el proyecto, además de esto, deberá ayudar al product manager para identificar posibles riesgos en el trabajo futuro para que el equipo pueda prevenirlos o atacarlos de la mejor forma y así lograr los objetivos a corto y por ende a largo plazo (en nuestro caso cumplir el alcance del proyecto en tiempo y forma).

El QA será el encargado de gestionar el plan de pruebas, este deberá estar involucrado en los requerimientos del sistema y deberá ser la barrera entre el producto que los desarrolladores construyen y el producto que se le entrega al cliente, para que de esta forma el producto entregado tenga el mayor valor y la menor cantidad de regresiones posibles, asegurando así la calidad del trabajo realizado.

A lo largo del proyecto tendremos 2 integrantes de cada rol, ambos seremos desarrolladores y QA, y a su vez seremos product manager la mitad del proyecto cada uno.

La idea que tenemos es que a medida que avanzan las semanas y el producto avanza ir testeando el producto como rol de QA intentando que el mismo desarrollador no testee su propio trabajo, por esta razón 2 QA y 2 desarrolladores. A su vez necesitaremos alguien que gestione como avanza el proyecto, por esto el product manager.

Resumiendo el equipo se compone por:

- Marco Fiorito - Product Manager/Desarrollador Semi-Senior/QA Semi Senior
- Matias Salles - Product Manager/Desarrollador Semi-Senior/QA Semi Senior

3.5. Definición de alcance del proyecto

En la siguiente sección definiremos el alcance del producto incluyendo el pliegue de condiciones del proyecto (project scope statement), la estructura de descomposición del trabajo (EDT/WBS) y el diccionario de la misma donde incluiremos una estimación del esfuerzo por cada paquete/entregable y asignaremos al mismo el rol correspondiente dentro del equipo de trabajo (Ver roles).

3.5.1. Project Scope Statement (Pliegue de condiciones)

■ Product scope description:

El producto-resultado pretende ser una mejora notable sobre el sistema existente, siendo en su nueva versión apto para el uso diario por los interesados que apuntan al uso del mismo (principalmente nutricionistas, deportistas y usuarios con problemas de salud).

La mejora no solo apunta a lograr un sistema confiable con un correcto funcionamiento en las funcionalidades ofrecidas, sino que también se pretende lograr un sistema fácil de usar sin importar edad/tipo de interesado y fácil de mantener gracias a una alta cobertura de pruebas unitarias, buena calidad de código y una documentación renovada que provee una rápida comprensión del proyecto a nuevos/futuros integrantes del mismo u otro equipo de mantenimiento.

■ Project deliverables:

El proyecto constará de 6 entregables principales los cuales serán entregados en sus correspondientes fechas:

- Mejora en la calidad global del código - 19/5/2020
- Correcciones en funcionamiento del sistema - 24/5/2020
- Mejora en la usabilidad global del sistema - 29/5/2020
- Nuevas Funcionalidades - 5/6/2020
- Mejora en la cobertura de pruebas unitarias - 12/6/2020
- Renovación de la documentación del sistema - 15/6/2020

La suma de estos 6 entregables conforman la entrega final en el día 16/6/2020

■ **Project acceptance criteria:**

Sera indispensable que la entrega final cuente con los 6 entregables mencionados anteriormente, así mismo, a continuación listaremos los criterios de aceptación que debe cumplir cada uno de ellos para que los mismos cumplan con la expectativa descripta en el product scope description.

- Mejora en la calidad global del código:
Las herramientas de análisis de código en conjunto (SonarQube, PMD y FindBugs) deberán:
 - Reportar menos de 250 code smells de baja prioridad.
 - Reportar menos de 40 code smells de alta prioridad.
 - Reportar 0 vulnerabilidades.
 - Reportar menos de 5 bugs de baja prioridad.
 - Reportar menos de 3 bugs de alta prioridad.
- Correcciones en funcionamiento del sistema:
Se deberán:
 - Eliminar todos los errores encontrados relacionados a la solicitud/-creación de los planes de alimentación.
 - Poder visualizar correctamente las ingestas de los usuarios en cualquier sección del programa.
 - Poder ejecutar el programa desde NetBeans sin que se generen excepciones.
- Mejora en la usabilidad global del sistema:
Se deberá:
 - Ofrecer otra resolución (preferentemente 1280x720) menor a 1360x768.
 - Reestructurar la pantalla de login siguiendo el estándar de las pantallas onboarding.
 - Visualizar las imágenes subidas de la misma forma en todas las secciones del programa.
 - Tener visibilidad del usuario que haya iniciado sesión dentro del sistema
- Nuevas Funcionalidades:
Se deberá:
 - Poder salir del programa desde cualquier ventana.
 - Poder cargar un set de datos de prueba desde el login.
 - Poder subir fotos jpg.
 - Poder seleccionar tanto un alimento ingerido como un alimento registrado para eliminarlo del sistema.
- Mejora en la cobertura de pruebas unitarias:
Mediante la herramienta JaCoCoverage, el sistema deberá:

- Presentar una cobertura de pruebas unitarias de un 90 % como mínimo, tanto en sentencias como ramas
 - Que no existan test unitarios que estén fallando.
- Renovación de la documentación del sistema:
Se deberá:
 - Documentar los RF nuevamente.
 - Documentar los casos de uso del sistema nuevamente siendo los mismos exhaustivos y coherentes con el funcionamiento del sistema.

Es importante mencionar que en caso de ser posible, cada entregable incluirá una mayor cantidad de mejoras, agregándole valor al sistema para el cliente.

■ Project constraints:

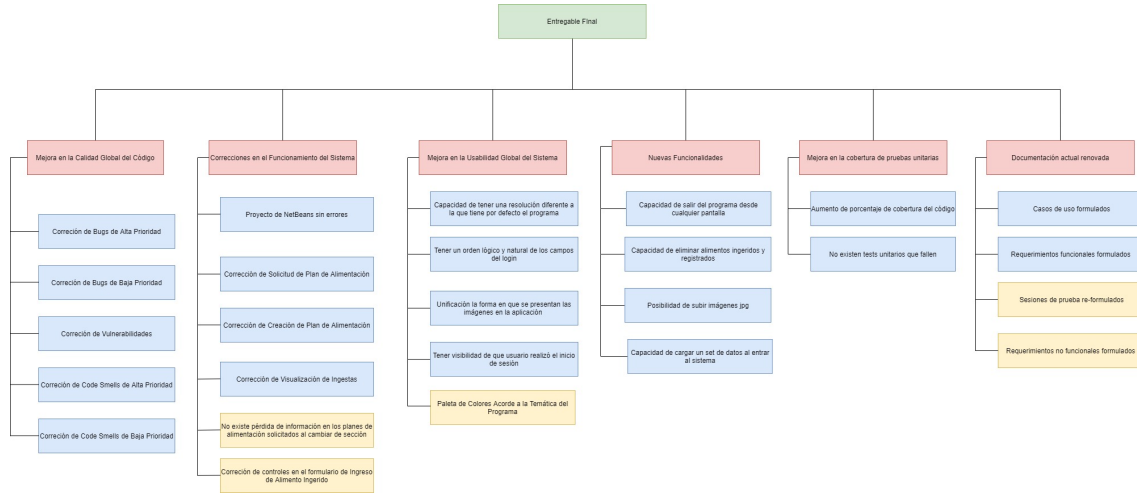
El proyecto presenta una fecha de finalización fija sin posible extensión (16/6/2020)

■ Project Assumptions

- Los requerimientos actuales del proyecto no cambiarán durante el mes que dura el proyecto, evitando tener que cambiar radicalmente el plan de mantenimiento semana a semana y realizar instancias de pre-planeación, definiendo lo que se ejecutará cada semana durante ese mes.
- El ambiente de testing será provisto por el cliente, de esta forma no tendremos que configurar nada para testear el sistema.
- Las herramientas de integración continua para realizar los releases a producción y a testing serán provistas por el cliente, ahorrándonos así tiempo de configuración y comunicación al cliente y testers sobre los pasos a seguir para hacer uso del sistema.

3.5.2. Estructura de descomposición del trabajo (EDT)

A continuación se incluye la estructura de descomposición del trabajo mediante el siguiente diagrama:



La imagen se encuentra en el zip y en el repositorio de trabajo mencionado al principio del capítulo para una mejor visualización, aun así, al acercarse a la imagen se puede visualizar correctamente el EDT.

Los paquetes amarillos hacen referencia a entregables los cuales no entran en el alcance del producto pero se incluirán en la entrega final en caso de ser posible, por esto mismo no tendrán un diccionario asociado.

3.5.3. Diccionario del EDT y Estimación de Esfuerzo

A continuación se incluirá un diccionario por cada entregable incluido en el EDT, en el mismo se incluirá las restricciones de dicho entregable, un estimación de horas para cada tarea asociada al entregable y el rol asignado a cada tarea. Los entregables de mayor jerarquía serán la suma de los entregables mas chicos, mas las horas de QA para testear el entregable (si aplica) y las horas del product manager para evaluar los resultados obtenidos y comunicar con el cliente entre otros, por esto mismo estos diccionarios de EDT no incluirán las tareas y roles.

Nombre de paquete: Corrección de Bugs de Alta Prioridad		
Restricciones:	Inicio: 13/5/2020 - Fin: 19/5/2020	
Tareas/Solicitudes	Horas Estimadas	Roles Asignado
TC2	1	Desarrollador

Nombre de paquete: Corrección de Bugs de Baja Prioridad		
Restricciones:	Inicio: 13/5/2020 - Fin: 19/5/2020	
Tareas/Solicitudes	Horas Estimadas	Roles Asignados
TC1	1	Desarrollador

Nombre de paquete: Corrección de Vulnerabilidades		
Restricciones:	Inicio: 13/5/2020 - Fin: 19/5/2020	
Tareas/Solicitudes	Horas Estimadas	Roles Asignados
TC3	2	Desarrollador

Nombre de paquete: Corrección de Code Smells de Alta Prioridad		
Restricciones:	Inicio: 13/5/2020 - Fin: 19/5/2020	
Tareas/Solicitudes	Horas Estimadas	Roles Asignados
TC5	4	Desarrollador

Nombre de paquete: Corrección de Code Smells de Baja Prioridad		
Restricciones:	Inicio: 13/5/2020 - Fin: 19/5/2020	
Tareas/Solicitudes	Horas Estimadas	Roles Asignados
TC4	4	Desarrollador

Nombre de paquete: Mejora en la Calidad Global del Código	
Restricciones:	Inicio: 13/5/2020 - Fin: 19/5/2020
Horas Desarrollador:	12
Horas Product Manager:	2
Horas Estimadas:	14

Nombre de paquete: Proyecto de NetBeans sin Errores		
Restricciones:	Inicio: 20/5/2020 - Fin: 24/5/2020	
Tareas/Solicitudes	Horas Estimadas	Roles Asignados
TN1	1	Desarrollador

Nombre de paquete: Corrección de Solicitud Plan de Alimentación		
Restricciones:	Inicio: 20/5/2020 - Fin: 24/5/2020	
Tareas/Solicitudes	Horas Estimadas	Roles Asignados
TN2	1	Desarrollador
TN3	2	Desarrollador

Nombre de paquete: Corrección de Creación Plan de Alimentación		
Restricciones:	Inicio: 20/5/2020 - Fin: 24/5/2020	
Tareas/Solicitudes	Horas Estimadas	Roles Asignados
TNU2	9	Desarrollador

Nombre de paquete: Corrección de Visualización Ingestas		
Restricciones:	Inicio: 20/5/2020 - Fin: 24/5/2020	
Tareas/Solicitudes	Horas Estimadas	Roles Asignados
TN5	2	Desarrollador

Nombre de paquete: Correcciones en el Funcionamiento del Sistema	
Restricciones:	Inicio: 20/5/2020 - Fin: 24/5/2020
Horas Desarrollador:	15
Horas QA:	3
Horas Product Manager:	2
Horas Estimadas:	20

Nombre de paquete: <u>Capacidad de tener una resolución diferente a la que tiene por defecto el programa</u>		
Restricciones:	Inicio: 25/5/2020 - Fin: 29/5/2020	
Tareas/Solicitudes	Horas Estimadas	Roles Asignados
TU8/SC2	10	Desarrollador

Nombre de paquete: <u>Tener un orden lógico y natural de los campos del login</u>		
Restricciones:	Inicio: 25/5/2020 - Fin: 29/5/2020	
Tareas/Solicitudes	Horas Estimadas	Roles Asignados
TU6	5	Desarrollador

Nombre de paquete: <u>Unificación de la forma en que se presentan las imágenes en la aplicación</u>		
Restricciones:	Inicio: 25/5/2020 - Fin: 29/5/2020	
Tareas/Solicitudes	Horas Estimadas	Roles Asignados
TUN1	6	Desarrollador

Nombre de paquete: <u>Tener visibilidad de que usuario realizó el inicio de sesión</u>		
Restricciones:	Inicio: 25/5/2020 - Fin: 29/5/2020	
Tareas/Solicitudes	Horas Estimadas	Roles Asignados
TU7	1	Desarrollador

Nombre de paquete: <u>Mejora en la Usabilidad Global del Sistema</u>	
Restricciones:	Inicio: 25/5/2020 - Fin: 29/5/2020
Horas Desarrollador:	22
Horas QA:	3
Horas Product Manager:	2
Horas Estimadas:	27

Nombre de paquete: <u>Capacidad de salir del programa desde cualquier pantalla</u>		
Restricciones:	Inicio: 30/5/2020 - Fin: 5/6/2020	
Tareas/Solicitudes	Horas Estimadas	Roles Asignados
SC1	5	Desarrollador

Nombre de paquete: <u>Capacidad de eliminar alimentos ingeridos y registrados</u>		
Restricciones:	Inicio: 30/5/2020 - Fin: 5/6/2020	
Tareas/Solicitudes	Horas Estimadas	Roles Asignados
TU2	9	Desarrollador

Nombre de paquete: <u>Posibilidad de subir imágenes jpg</u>		
Restricciones:	Inicio: 30/5/2020 - Fin: 5/6/2020	
Tareas/Solicitudes	Horas Estimadas	Roles Asignados
SC6	3	Desarrollador

Nombre de paquete: <u>Capacidad de cargar un set de datos al entrar al sistema</u>		
Restricciones:	Inicio: 30/5/2020 - Fin: 5/6/2020	
Tareas/Solicitudes	Horas Estimadas	Roles Asignados
TU3	6	Desarrollador

Nombre de paquete: <u>Nuevas Funcionalidades</u>	
Restricciones:	Inicio: 30/5/2020 - Fin: 5/6/2020
Horas Desarrollador:	23
Horas QA:	3
Horas Product Manager:	2
Horas Estimadas:	28

Nombre de paquete: <u>Aumento de porcentaje de cobertura del código</u>		
Restricciones:	Inicio: 6/6/2020 - Fin: 12/6/2020	
Tareas/Solicitudes	Horas Estimadas	Roles Asignados
TP1	12	Desarrollador

Nombre de paquete: <u>No existen tests unitarios que fallen</u>		
Restricciones:	Inicio: 6/6/2020 - Fin: 12/6/2020	
Tareas/Solicitudes	Horas Estimadas	Roles Asignados
TP2	4	Desarrollador

Nombre de paquete: <u>Mejora en la cobertura de pruebas unitarias</u>	
Restricciones:	Inicio: 6/6/2020 - Fin: 12/6/2020
Horas Desarrollador:	16
Horas Product Manager:	2
Horas Estimadas:	19

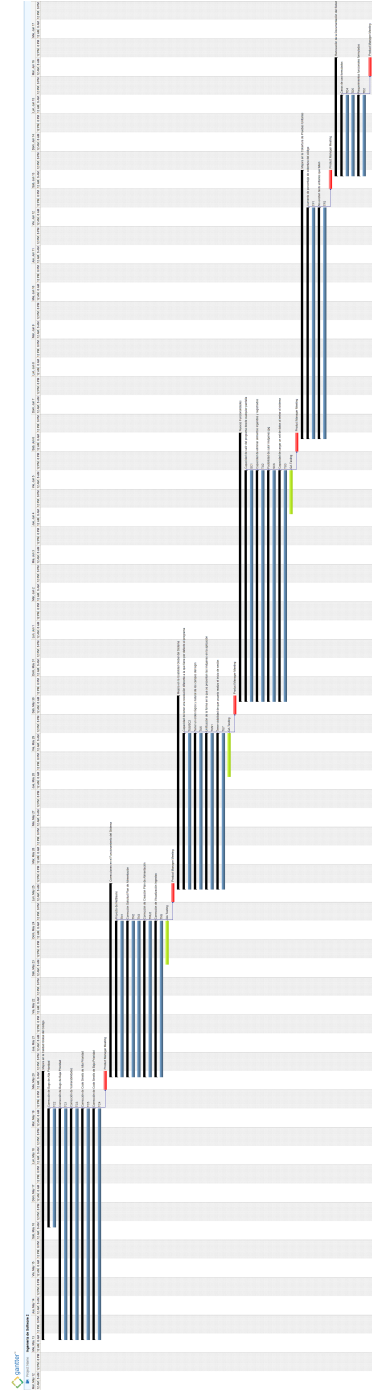
Nombre de paquete: <u>Casos de uso formulados</u>		
Restricciones:	Inicio: 13/6/2020 - Fin: 15/6/2020	
Tareas/Solicitudes	Horas Estimadas	Roles Asignados
TD4	6	QA
TD5	4	QA

Nombre de paquete: <u>Requerimientos funcionales formulados</u>		
Restricciones:	Inicio: 13/6/2020 - Fin: 15/6/2020	
Tareas/Solicitudes	Horas Estimadas	Roles Asignados
TD2	10	QA

Nombre de paquete: <u>Documentación actual renovada</u>	
Restricciones:	Inicio: 13/6/2020 - Fin: 15/6/2020
Horas QA:	20
Horas Product Manager:	2
Horas Estimadas:	22

3.6. Cronograma de trabajo.

A continuación se incluye el Gantt formulado para la siguiente etapa. Cabe destacar que se incluye la imagen en el zip y el repositorio mencionado en el principio del capitulo en caso de no poder visualizar correctamente el mismo aun acercando la imagen varias veces. Las tareas dentro de cada entregable se muestran en forma paralela ya que no tienen predecesores. Finalmente, se medirá el tiempo tomado para resolver cada tarea, así luego podremos comparar con la estimación inicial.



3.7. Presupuesto inicial

A continuación se muestra el cálculo del presupuesto inicial utilizando los datos anteriores definidos en la estimación de costos.

Rol	Horas Totales	Costo por hora (US\$)	Costo Total (US\$)
Desarrollador	88	15	1320
Product Manager	12	20	240
QA	29	12	348
Presupuesto inicial: US\$ 1908			

3.8. Plan de calidad.

En esta sección especificaremos el plan de calidad que se ejecutará durante este proyecto, serán especificadas las prácticas y estándares a utilizar.

Como el proyecto tiene una corta duración utilizaremos un proceso ágil de trabajo, por esta razón el plan de calidad deberá adaptarse al proceso de trabajo para que ambos estén alineados y al final del proyecto lograr cumplir con lo prometido al cliente en tiempo y forma.

3.8.1. Actividades de aseguramiento y control de calidad

A continuación se detallarán las practicas a realizar en el transcurso del proyecto:

Prácticas	Desarrollo	Encargado	Objetivos y fundamentos
Planificación de proceso de software	Realizar un proceso organizado y riguroso de software	Equipo	El objetivo es saber en cada momento del proyecto que hacer, que falta hacer y como realizarlo, así como también estar preparados frente a los problemas que pueden surgir
Diseño simple	Realizar el código lo más simple que se pueda	Desarrolladores	Se busca que el código sea simple para que cada uno de los desarrolladores entienda perfecto lo que se esta haciendo y no pierda tiempo en entenderlo.
Pruebas Unitarias	Realizar pruebas a medida que se desarrolla el sistema siendo una restricción para dar como finalizada una funcionalidad o bug que estén realizadas.	Desarrolladores	Se busca que durante el desarrollo se realicen pruebas para encontrar errores lo más temprano posible y así disminuir el re-trabajo a futuro y las regresiones.
Refactoring	Mientras se realiza el desarrollo tratar de mejorar el código constantemente.	Desarrolladores	Se busca que el diseño del código se mantenga simple, que la cantidad de errores y code smells disminuya y los programadores puedan desarrollar más rápido a futuro.

Prácticas	Desarrollo	Encargado	Objetivos y fundamentos
Integración Continua	Integrar continuamente las distintas partes desarrolladas por los distintos desarrolladores semana a semana	Desarrolladores	Las integraciones continuas apuntan a disminuir los errores generados en la etapa de integración y que el cliente tenga a su disposición versiones para probar.
Conocimiento del equipo sobre el código	Se busca que todos los desarrolladores conozcan el código, una práctica que vamos a seguir son realizar reviews de código cruzados para encontrar mejoras, siguiendo rigurosamente metodología de trabajo propuesta por GitFlow y Pull Requests cruzados.	Desarrolladores	Se busca que los programadores conozcan todo el código desarrollado, esto hace más ágil y más flexible al proceso de desarrollo.
25 horas máximo semanales desarrollando	Cantidad máxima de horas semanales que pueden trabajar los desarrolladores	Desarrolladores	Para evitar cansancio y disminuir errores, no permitiremos trabajar más de 25 horas semanales desarrollando
5 horas máximo semanales testeando	Cantidad máxima de horas semanales realizando test manuales.	QA	Para evitar cansancio pensando además que en nuestro equipo ambos tenemos roles de QA y Desarrolladores, dedicaremos como máximo 5 horas semanales a testear.
Test de regresión	Realizaremos test de regresión cada dos semanas.	QA	Para que durante el proyecto no se introduzcan regresiones realizaremos 2 test de regresión, uno a mitad del proyecto y uno al final para el último entregable.
Smoke test	Realizaremos smoke test al final de cada semana.	QA	Con el objetivo de que las nuevas funcionalidades no introduzcan nuevos bugs o introduzcan un bug sobre una funcionalidad crítica, realizaremos un smoke test al final de cada semana para no hacer un entregable con fallas críticas.

3.8.2. Estándares a utilizar

Durante el proyecto vamos a seguir los estándares de calidad de código de desarrollo en Java y utilizando analizadores de código como Sonarqube y PMD para medir corroborar el cumplimiento de dichos estándares y evaluar el estado/mantenibilidad del código.

A esto le sumaremos los reviews cruzados en los Pull Requests adjuntando los reportes de los analizadores mencionados anteriormente.

3.8.3. Calidad Inicial

Finalmente, para evaluar la calidad inicial del proyecto, utilizando como insumo los defectos listados en la primera parte nos concentraremos en los siguientes factores de calidad de Mc Call que consideramos mas importantes para este proyecto:

- Corrección
- Confiabilidad
- Facilidad de Uso
- Facilidad de Mantenimiento
- Flexibilidad
- Facilidad de Prueba

Para cada uno de ellos se mostrara una tabla incluyendo los defectos principales que influyen en cada factor. Finalmente se calificara cada factor de 0 a 4, donde el 0 representa un buen uso de las practicas pautadas y 4 un fallo grave.

Cabe destacar que:

- El resultado final de este análisis se tuvo en cuenta al definir el alcance del proyecto
- Al final del proyecto se evaluarán nuevamente dichos factores.

Corrección	
Puntos Débiles	Puntos Fuertes
Código y Funcionalidades: - Posibilidad de crear un plan alimenticio no completo no refleja la realidad (Falta de controles). <i>Defecto/s N7</i>	Código y Funcionalidades: - A pesar de existir otros defectos en el comportamiento del sistema, no se encontró otro que hiciera que el mismo no refleje la realidad.
Usabilidad: - Resolución única de la interfaz gráfica, incompatible con varias computadoras. <i>Defecto/s U9</i>	
Documentación: - Falta de requerimientos funcionales y no funcionales. <i>Defecto/s D2, D3</i> - Incorrecto e incompleto formato de casos de uso. <i>Defecto/s D4</i> - Inconsistencias en algunos: defectos listados, casos de uso, log de versiones sesiones de prueba (no coinciden con el comportamiento del sistema). <i>Defecto/s D1, D5, D6, D7</i>	
Pruebas Unitarias: - La cobertura de pruebas unitarias no es aceptable respecto al estándar del 90 %. <i>Defecto/s P1</i> - Existen tests entregados que fallan y otros que generan errores. <i>Defecto/s P2</i>	
Calificación:4	

En las tablas siguientes, podremos observar que los puntos débiles listados principalmente en: código, usabilidad y pruebas unitarias, están incluidos en otros factores de calidad (confiabilidad y facilidad de prueba específicamente), por lo que una mejora en los factores mencionados llevaría a una mejora en la corrección, la cual fallaría principalmente en la documentación.

Confiabilidad	
Puntos Débiles	Puntos Fuertes
Código y Funcionalidades: <ul style="list-style-type: none"> - Posibilidad de crear un plan alimenticio no completo. (Falta de controles). <i>Defecto/s N7</i> - (*)Nuevos defectos encontrados, no mencionados en la documentación. <i>Defecto/s N1 - N11</i> - La gran cantidad de defectos listados en la documentación no corregidos. - Inconsistencia entre el proyecto de Net-Beans y el ejecutable. <i>Defecto/s N1</i> - Los bugs, vulnerabilidades, codesmells que fueron clasificados como críticos o de riesgo alto. <i>Defecto/s C2, C5</i> 	Código y Funcionalidades: <ul style="list-style-type: none"> - Los resultados de los analizadores de código no presentaron una gran cantidad de bugs, code smells, que fueran críticos para el funcionamiento del sistema.
Usabilidad: <ul style="list-style-type: none"> - Resolución única de la interfaz gráfica, incompatible con varias computadoras. <i>Defecto/s U9</i> - Recorte no deseado de las imágenes de alimentos, usuarios y profesionales. <i>Defecto/s U13, N9, N10, N11</i> 	
Documentación: <ul style="list-style-type: none"> - Idem (*) - Inconsistencias en algunos: defectos listados, casos de uso, log de versiones sesiones de prueba (no coinciden con el comportamiento del sistema). <i>Defecto/s D1, D5, D6, D7</i> 	
Pruebas Unitarias: <ul style="list-style-type: none"> - Existen tests que fallan y otros que generan errores. <i>Defecto/s P2</i> 	
Calificación:4	

Facilidad de Uso	
Puntos Débiles	Puntos Fuertes
Código y Funcionalidades: N.A	
Usabilidad: <ul style="list-style-type: none"> - Falta de especificación en los pasos intermedios de las funcionalidades o cuando realizamos cursos alternativos. <i>Defecto/s U1</i> - Resolución única de la interfaz gráfica, incompatible con varias computadoras. <i>Defecto/s U9</i> - Iconos y listas que no especifican su propósito. <i>Defecto/s U1, U5</i> - Falta de tips/ayuda en la sección del login y difícil de encontrar en las otras secciones. <i>Defecto/s U12</i> - Falta de especificación de datos obligatorios antes de enviar un formulario. <i>Defecto/s U6</i> - Orden no-lógico para la creación de un plan alimenticio de parte de un profesional. <i>Defecto/s U14</i> - Faltan contra partes de todas las funcionalidades del sistema. <i>Defecto/s U3</i> 	Usabilidad: <ul style="list-style-type: none"> - El programa presenta salidas claras en casos de falta de datos al enviar un formulario.
Documentación: <ul style="list-style-type: none"> - La falta e impropia documentacion de los RF y casos de uso hace difícil utilizar el sistema. <i>Defecto/s D2, D4</i> 	
Pruebas Unitarias: N.A	
Calificación:3	

Facilidad de Mantenimiento	
Puntos Débiles	Puntos Fuertes
Código y Funcionalidades: - Los comportamientos incorrectos ya existentes pueden influir en un 'arreglo', terminando en una cadena de bugs,errores,etc. que deben ser arreglados para para llevar a cabo la corrección propuesta en primer lugar. <i>Defecto/s N1 - N11 y los mencionados en la documentación del sistema (6.2)</i>	Código y Funcionalidades: - El dominio esta compuesto por solamente 1000 lineas de código aproximadamente y la UI por 4200 aproximadamente, de las cuales una gran parte son para la inicialización de los componentes de JavaSwing, lo que facilita localizar nuevos problemas o existentes. - Los resultados de los analizadores de código no presentaron una gran cantidad de bugs, code smells, que fueran críticos para el funcionamiento del sistema.
Usabilidad: N.A	
Documentación: N.A	
Pruebas Unitarias: -La baja cobertura de pruebas unitarias no nos permite verificar con seguridad que luego de una corrección las funcionalidades que ya disponía el sistema siguen su correcto funcionamiento sin ser afectadas por nuestro cambio. <i>Defecto/s P1</i> - Existen tests entregados que fallan y otros que generan errores. <i>Defecto/s P2</i>	
Calificación:2	

Flexibilidad	
Puntos Débiles	Puntos Fuertes
Código y Funcionalidades: - Los comportamientos no deseados en distintas partes de la aplicación pueden influir en los cambios propuesto dado que para hacer ciertas funcionalidades se necesitará solucionar errores actuales de la aplicación, además, al realizar cambios sobre código con defectos hay más probabilidad de introducir nuevos bugs. <i>Defecto/s N1 - N11 y los mencionados en la documentación del sistema (6.2)</i>	Código y Funcionalidades: - El dominio esta compuesto por solamente 1000 lineas de código aproximadamente y la UI por 4200 aproximadamente, de las cuales una gran parte son para la inicialización de los componentes de JavaSwing, lo que facilita modificar funciones existentes. - Los resultados de los analizadores de código no presentaron una gran cantidad de bugs, code smells, que fueran críticos para el funcionamiento del sistema.
Usabilidad: N.A	
Documentación: N.A	
Pruebas Unitarias: -La baja cobertura de pruebas unitarias no nos permite verificar con seguridad que luego de un cambio/corrección/funcionalidad nueva entre otras, las funcionalidades que ya disponía el sistema siguen su correcto funcionamiento sin ser afectadas por nuestro cambio. <i>Defecto/s P1</i>	
<u>Calificación:2</u>	

Se puede observar que los defectos que influyen a la flexibilidad también están incluidos en la facilidad de mantenimiento, por lo que una mejora en la facilidad de mantenimiento del sistema tendría implícita una mejora en la flexibilidad del mismo.

Facilidad de Prueba	
Puntos Débiles	Puntos Fuertes
Código y Funcionalidades: N.A	
Usabilidad: - El programa no presenta ningún set de datos de prueba que se pueda acceder de forma fácil para realizar pruebas del programa en distintos estados del mismo, por lo cual cada prueba exploratoria que se quiera realizar totalmente independientes de las demás pruebas hay que volver al estado inicial del programa perdiendo todos los datos anteriores, eliminando el archivo serializado o agregando a mano todos los datos nuevamente. <i>Defecto/s U4</i>	
Documentación: N.A	
Pruebas Unitarias: - La cobertura de pruebas unitarias no es aceptable respecto al estándar del 90 %, lo cual dificulta la tarea de asegurar que el sistema o una funcionalidad específica presenta un funcionamiento correcto. <i>Defecto/s nº P1</i> - Existen tests entregados que fallan y otros que generan errores. <i>Defecto/s P2</i>	
Calificación:3	

3.8.4. Resultado

Factor de calidad	Calificación
Correccion	4
Confiabilidad	4
Facilidad de Uso	3
Facilidad de Mantenimiento	2
Flexibilidad	2
Facilidad de Prueba	3
Promedio de calidad inicial: 3	

Podemos concluir que la calidad del sistema es baja y su mejora es crítica, ya que el resultado promedia 1 punto menos que el puntaje mínimo

Teniendo en cuenta el resultado anterior, fijaremos dos factores principales y tres secundarios (en los cuales también se definirán prioridades) a mejorar.

Principales:

- Confiabilidad
- Facilidad de Prueba

Secundarios:

1. Facilidad de Uso
2. Facilidad de Mantenimiento
3. Correctitud

Los factores principales fueron seleccionados tal que dichas mejoras apunten a que el sistema presente un funcionamiento correcto de sus funcionalidades principales para poder ser usado correctamente, y a facilitar un mantenimiento futuro del mismo (agregar funcionalidades, encontrar errores/bugs fácilmente entre otros, comprensión del sistema mediante la documentación).

Los factores secundarios se seleccionaron principalmente con una mejora en la usabilidad en mente, luego que el dominio sea mejorado para una mayor robustez del sistema. También se tuvo en cuenta los defectos solapados (que afectan a mas de un factor), para obtener el resultado mas óptimo posible luego de las mejoras.

Por esto mismo, si bien se procura mejorar la documentación, no se tratará con total formalidad (documentando sesiones de prueba, requerimientos no funcionales entre otros). Con esto lograremos priorizar el correcto funcionamiento del sistema, ya que mejorar de forma extensa la documentación no corrige los defectos del mismo, aun así, se procurara crear una documentación que sea correcta, sin inconsistencias respecto al funcionamiento del sistema y que permita obtener una rápida comprensión del programa a quienes lo lean.

3.8.5. Glosario de Factores de Calidad de Mc Call

- **Corrección:** Grado en el que un software satisface sus especificaciones y cumple con los objetivos de la misión del cliente.
- **Confiabilidad:** Grado en el que se espera que un software cumpla con su función y con la precisión requerida.
- **Facilidad de Uso:** Esfuerzo necesario para aprender, operar y preparar los datos de entrada de un programa e interpretar su salida.
- **Facilidad de Mantenimiento:** Esfuerzo necesario para localizar y corregir un error en un programa.
- **Flexibilidad:** Esfuerzo necesario para modificar un programa en operación.
- **Facilidad de Prueba:** Esfuerzo que demanda probar un programa con el fin de asegurar que realiza su función.

3.9. Plan de Métricas y GQM

A continuación se listará cada objetivo especificando las preguntas, de donde luego se derivarán métricas para luego definir la implementación/realización de cada una de ellas a lo largo de la ejecución del proyecto.

- **Objetivo:** No comprometer la calidad del proyecto en caso de no llegar a tiempo con las entregas.
- **Preguntas:**
 - ¿Cómo medimos la calidad?
 - ¿Cómo sabemos si llegamos a tiempo?
 - ¿Consideramos que la calidad del proyecto aumentó?
- **Métricas:**
 - **M1:** Cantidad de bugs, vulnerabilidades y code smells con SonarQube, PMD y FindBugs.
 - **M2:** Cobertura de pruebas unitarias sobre las funciones/clases agregadas/modificadas mediante JaCoCoverage.
 - **M3:** Comparación entre la fecha de finalización de cada entregable y la fecha en la cual damos por finalizado dicho entregable.
 - **M4:** Comparación entre la cantidad de horas estimadas en cada entregable y la cantidad real.
 - **M5:** Evaluar que las tareas marcadas como 'Done' cumplen con su criterio de aceptación (la solución definida).
 - **M6:** Cantidad de tareas hechas del entregable vigente - Cantidad de tareas en el entregable.
 - **M7:** Cantidad de Pull Requests revisados por el otro integrante del equipo.
- **Implementación:**
 - Analizar al finalizar las tareas el código con los analizadores mencionados en **M1**.
 - Analizar al finalizar las tareas la cobertura de pruebas unitarias relacionadas con las funciones/clases agregadas/modificadas (si aplica).
 - Registrar el inicio y fin de cada tarea durante el desarrollo.
 - Registrar el inicio y fin de cada entregable durante el desarrollo.
 - Llevar registro de las veces que no ha existido revisión en GitHub del código por otro desarrollador.
 - Confirmar que la tarea realizada cumple con su criterio de aceptación antes de etiquetarla como 'Done'.

- Revisar cada dos días la cantidad de tareas hechas del entregable vigente para evaluar el estado actual de progreso.
 - Esperar a que el otro integrante del equipo realice una review sobre el Pull Request antes de realizar el Merge.
-

▪ **Objetivo:** Tener menos de 15 regresiones a lo largo del proyecto.

▪ **Preguntas:**

- ¿Cómo sabemos cuando ocurre una regresión?
- ¿Por qué ocurrió la regresión?

▪ **Métricas:**

- **M8:** Cantidad de veces que se marca como 'In Progress' o 'To Do' una tarea que ya estaba marcada como 'Done'.

▪ **Implementación:**

- Utilizaremos Jira para definir las tareas en 'tarjetas', las mismas podrán una de las cuatro etiquetas (To do, In Progress, In Manual Test, Done). Cuando una tarjeta en estado Done pasa a In Progress o To Do aumentaremos en uno la cantidad de regresiones durante la ejecución del proyecto.
-

▪ **Objetivo:** Haber llevado a cabo al final del proyecto al menos un 75 % de las tareas

▪ **Preguntas:**

- ¿Cómo sabemos el porcentaje de las tareas realizadas?

▪ **Métricas:**

- **M9:**

$$\%TareasRealizadas = \frac{TareasDone \cdot 100}{TareasDefinidas} \quad (3.1)$$

▪ **Implementación:**

- Analizar al final de cada entregable (3.1) para evaluar como se esta ejecutando el proyecto.

3.10. Ciclo de vida.

El análisis realizado en las últimas secciones nos ha ayudado a seleccionar que ciclo de vida es el que se ajusta más a nuestro proyecto de mantenimiento y por ende es el que vamos a utilizar.

Decidimos que la mejor opción es un ciclo de vida en espiral porque este proyecto tendrá una duración corta y tenemos que tener mecanismos para gestionar los riesgos adecuadamente, ya que una mala gestión de riesgos puede repercutir en un atraso en el proyecto y por ende un atraso en los entregables prometidos al cliente.

Consideramos que las iteraciones recurrentes en definir objetivos, evaluar riesgos, desarrollo/ validación y planificación nos van a ayudar a identificar riesgos de forma temprana semana a semana, o al final de cada entregable, y así poder agilizar el proyecto o gestionar cambios en el cronograma.

Este proyecto lo vamos a dividir en 6 fases (uno por cada entregable) y al final de cada una tendremos una instancia de retrospectiva para ver como nos fue en esta fase tanto en los objetivos como en cantidad de valor que se agrego al producto, y así verificar que lo que está planeado para la siguiente fase no cambió de prioridad por nuevos riesgos en el desarrollo.

De esta forma creemos que vamos a llegar al final del proyecto con el mayor valor en el producto posible y lograr cumplir las expectativas del cliente.

Bibliografía

- [1] R. S. Pressman, *Software Engineering: A Practitioner's Approach*, eight ed. McGraw-Hill, 2014.
- [2] I. Sommerville, *Software Engineering*, 10th ed. Pearson, 2015.
- [3] Universidad ORT Uruguay. (2013) Documento 302 - Facultad de Ingeniería. [Online]. Available: <http://www.ort.edu.uy/fi/pdf/documento302facultaddeingenieria.pdf>
- [4] ——. Elaboración de planes de calidad en proyectos de software. [Online]. Available: http://sedici.unlp.edu.ar/bitstream/handle/10915/23062/Documento_completo.pdf?sequence=1&isAllowed=y