

Universidad ORT Uruguay

Facultad de Ingeniería

Ingeniería de Software 2

Obligatorio 2

Marco Fiorito (227548)

Matias Salles Dulan (201701)

16 de junio de 2020

Índice general

1. Informe de estado del proyecto	2
1.1. Logros obtenidos en el proyecto	2
1.2. Logros planificados pero no completados	4
1.3. Causa raíz de las variaciones	4
1.4. Fondos gastados	5
1.5. Situación del presupuesto del proyecto	5
1.6. Situación de los riesgos identificados	7
1.6.1. Riesgos	7
1.6.2. Plan de respuesta	7
2. Estado de calidad del software	9
2.1. Analizadores de Código	9
2.1.1. SonarQube	10
2.1.2. PMD	11
2.1.3. FindBugs	12
2.2. Cobertura de pruebas unitarias	13
2.3. Análisis de usabilidad	14
2.3.1. Resultado	18
3. Análisis de documentación	19
4. Medición de la Calidad Final	20
4.0.1. Resultado	23
4.0.2. Glosario de Factores de Calidad de Mc Call	24
5. Plan de Metricas	25
6. Reflexión final	28
6.1. Analisis objetivos SMART	28
6.2. Analisis del equipo	29

1. Informe de estado del proyecto

Periodo del informe: 13/05/2020 - 15/06/2020

En este capítulo se analizarán los logros obtenidos y costos generados luego de la finalización del proyecto de mantenimiento.

1.1. Logros obtenidos en el proyecto

A continuación se listarán los logros obtenidos en las primeras dos semanas del proyecto.

- **Mejora en la calidad global del código:** (Completo)
 - Corrección de Bugs de Baja Prioridad (TC1)
 - Corrección de Bugs de Alta Prioridad (TC2)
 - Corrección de Vulnerabilidades (TC3)
 - Corrección de Code Smells de Baja Prioridad (TC4)
 - Corrección de Code Smells de Alta Prioridad (TC5)
- **Correcciones en el funcionamiento del sistema:**
 - Tener Proyecto de NetBeans sin Errores (TN1)
 - Corrección de Solicitud Plan de Alimentación (TN2, TN3)
 - Corrección de Visualización Ingestas (TN5)
- **Mejora en la usabilidad global del sistema:**
 - Tener visibilidad de que usuario realizó el inicio de sesión (TU7)
 - Tener un orden lógico y natural de los campos del login (TU6)
 - Capacidad de tener una resolución diferente a la que tiene por defecto el programa (TU8/SC2)
- **Nuevas funcionalidades:** (Adelantado)
 - Capacidad de salir del programa desde cualquier pantalla a la que tiene por defecto el programa (SC1)
- **Mejora en la cobertura de pruebas unitarias:** N.A

- **Documentación actual renovada:** N.A

A continuación se listarán los logros obtenidos en las ultimas dos semanas del proyecto.

- **Mejora en la calidad global del código:** (Completo)
- **Correcciones en el funcionamiento del sistema:** (Completo)
 - Corrección de Creación Plan de Alimentación (TNU2)
- **Mejora en la usabilidad global del sistema:** (Completo)
 - Unificación en la forma que se presentan las imágenes en la aplicación (TNU1)
- **Nuevas funcionalidades:** (Completo)
 - Capacidad de cargar un set de datos al entrar al sistema (TU3)
 - Capacidad de eliminar alimentos ingeridos y registrados (Recortado del alcance)
 - Posibilidad de subir imágenes .jpg (Recortado del alcance)
- **Mejora en la cobertura de pruebas unitarias:** (Completado)
 - No existen tests unitarios que fallen (TP2)
 - Aumento de porcentaje de cobertura del código (TP1)
- **Documentación actual renovada:** (Completado)
 - Requerimientos funcionales formulados (TD2)
 - Casos de uso formulados (TD4,TD5)
 - Requerimientos no funcionales formulados (TD3) - (Extra, no estaba en el alcance)

1.2. Logros planificados pero no completados

Los siguientes paquetes de trabajo fueron recortados del alcance como parte del plan de respuesta (**PR1**):

- Capacidad de eliminar alimentos ingeridos y registrados
- Posibilidad de subir imágenes .jpg

1.3. Causa raíz de las variaciones

Vale la pena resaltar que varias de estas variaciones son las mismas que para el informe de avance, dado que en el segundo período no hemos sufrido ningún desvío, aunque si sufrimos un riesgo el cual es mencionado mas adelante en la sección **Situación de los riesgos identificados**.

En este período hemos realizado cuatro reuniones de aproximadamente una hora que funcionaron como retrospectiva de cada semana. A partir de estas reuniones hemos concluido algunas razones por las cuales el desarrollo se ha atrasado y no hemos logrado cumplir con los objetivos para este período.

A continuación se listarán estas razones.

- Mala organización del código: este punto ha provocado una curva de adaptación mayor al proyecto al que teníamos esperado, esto ha atrasado cambios en el código que teníamos planificado para el entregable **Correcciones en el Funcionamiento del Sistema y Mejora en la Usabilidad Global del Sistema**.
- Mala organización de las clases de interfaz: los nombres y la distribución de los componentes de UI no son consistentes buena provocando mucho más trabajo y esfuerzo en cada cambio dado que nos obliga a mejorar la calidad de muchas cosas.
- Mala calidad del código: este es un punto que ya teníamos identificado pero que de todas formas subestimamos, al adentrarnos más en el código notamos que cada pequeño cambio implicaba un trabajo mayor al esperado ya que métodos realizaban distintas cosas de las que su nombre implicaba, los nombres de las variables no eran correctos entre otros.
- División de responsabilidades: las clases de la UI realizan muchas cosas, los desarrolladores de este proyecto hicieron que una misma pantalla tenga paneles de diferentes pantallas superpuestos lo que hace mucho más difícil la mantenibilidad y la facilidad de agregar nuevas pantallas.

1.4. Fondos gastados

En la carpeta 'Obligatorio 2' incluida en la entrega se encuentra un archivo 'Registro de Esfuerzo.txt' utilizado para llevar registro del mismo, también incluido en el repositorio utilizado:

[Link al Repositorio](#)

Si utilizamos los costos por hora incluidos en la entrega del obligatorio 1 podemos obtener el AC:

- Desarrollador - 61hs 15min = 915.75 US\$
- Product Manager - 15hs = 300 US\$
- QA - 13hs = 156 US\$

Total (AC): 1371,75 US\$

1.5. Situación del presupuesto del proyecto

El EV hasta el momento siguiendo el diccionario de la EDT es el siguiente:

- Desarrollador - 76hs = 1140 US\$
- Product Manager - 12hs = 240 US\$
- QA - 29hs = 348 US\$

Total(EV): 1728 US\$.

Si lo comparamos con el BAC (1908 US\$) incluido en el obligatorio 1 (presupuesto inicial) y le restamos el EV vemos que el resultado es equivalente al costo de los dos paquetes de trabajo que se quitaron del alcance:

BAC - EV = 1908 - 1728 = 180 US\$

Corroborando:

- Capacidad de eliminar alimentos ingeridos y registrados: 9hs de Desarrollador
- Posibilidad de subir imágenes .jpg: 3hs de Desarrollador

Total tareas no completadas = 180 US\$

Entonces con el EV y el AC podemos calcular el CPI como EV/AC:

$$\text{CPI} = 1728/1371 = 1,26$$

Esto implica que a pesar de haber gastado mas en el primer periodo como se mostró en el informe de avance, en la segunda mitad gastamos bastante menos de lo planeado con lo que termino dando un CPI mayor a 0.

También podemos calcular el SPI como EV/PV siendo el PV = BAC dado que el proyecto ha finalizado.

$$\text{SPI} = 1728/1908 = 0,9$$

Esto implica que nos quedamos apenas por detrás en el cronograma, lo cual se justifica con el plan de respuesta (**PR1**), en donde se quitó del alcance los paquetes de trabajo mencionados anteriormente.

Finalmente, podemos concluir que se gasto menos de lo planeado, en particular $(\text{BAC} - \text{AC} = 573 \text{ US\$})$, y si quitamos las tareas recortadas del alcance y calculamos nuevamente utilizando un nuevo BAC $(\text{BAC2} - \text{AC} = 357 \text{ US\$})$ podemos ver que en cuanto a costos el proyecto fue exitoso, mientras que en cuanto al cronograma dos paquetes de trabajo no se realizaron como se justifico anteriormente.

1.6. Situación de los riesgos identificados

A continuación se incluirán los riesgos que surgieron en el periodo del proyecto y fueron identificados por el equipo, así como el plan de respuesta para cada uno de ellos.

1.6.1. Riesgos

(R1) Como hemos comentado anteriormente uno de los riesgos que hemos tenido y sabemos que vamos a tener en el próximo período son las malas prácticas de programación con las cuales fue realizado el sistema, lo cual hace muy difícil implementar cambios y/o agregar nuevas funcionalidades a la aplicación, llevando a un atraso en el cronograma.

(R2) La solución definida en una tarea (asociada a la resolución de pantalla) no era la más óptima o requería de más trabajo que el esperado, que junto a la mala calidad del código hizo que esta misma sea inviable.

(R3) Luego de realizar una evaluación con EVM el proyecto parece que costaría mas de lo planeado.

(R4) No es posible testear algún método del dominio.

1.6.2. Plan de respuesta

(PR1) Consiste en realizar refactor de las clases en donde el equipo opera más frecuentemente, lo cual implica horas de trabajo en las cuales no avanzamos en el cronograma. Por esto mismo se decidió combinar este plan de respuesta con **(PR3)**.

(PR2) Consiste en solucionar el defecto asociado a la tarea de otra forma. En este caso nos ocurrió con el entregable (Capacidad de tener una resolución diferente a la que tiene por defecto el programa (TU8/SC2)) por lo que elegimos cambiar la resolución del programa a 1280x720 en vez de ofrecer esta misma y la anterior, ya que 1280x720 es soportada por la gran mayoría de monitores, generalmente a partir de las 13 pulgadas.

(PR3) Retirar del alcance 1 o 2 sub-entregables de prioridad Baja y Media. En este caso elegimos retirar los siguientes sub-entregables:

- Posibilidad de subir imágenes .jpg (SC6) - Prioridad BAJA
- Capacidad de eliminar alimentos ingeridos y registrados (TU2) - Prioridad MEDIA

(**PR4**) En nuestro caso nos ocurrió con el método `guardarDatosSistemas()` como se explica luego en la sección de **Cobertura de pruebas unitarias**, por lo que decidimos considerarlo como aceptado igualmente.

2. Estado de calidad del software

En este capítulo analizaremos el estado de calidad del software luego de la finalización del proyecto de mantenimiento.

2.1. Analizadores de Código

A continuación se incluirá los resultados de analizar el código del proyecto utilizando SonarQube, PMD y FindBugs, con una breve explicación sobre los nuevos resultados en comparación a los obtenidos antes del comienzo el proyecto de mantenimiento.

2.1.1. SonarQube

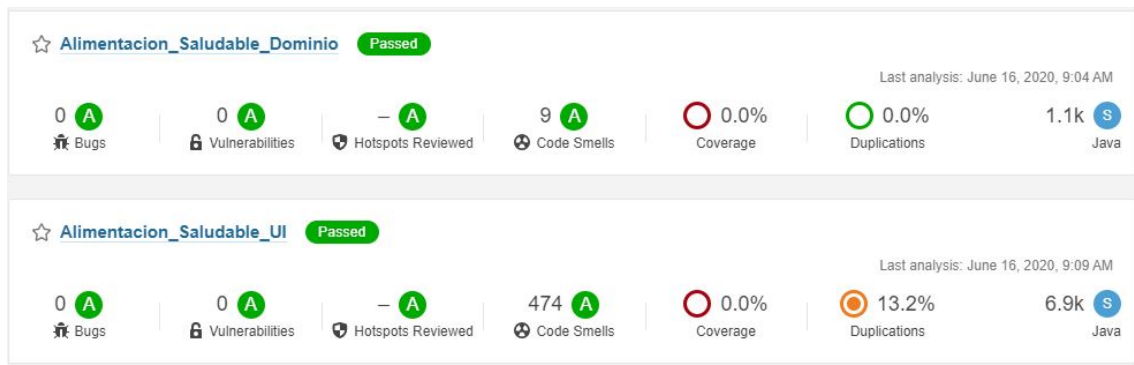
SonarQube es una plataforma para evaluar código fuente que usa diversas herramientas de análisis estático de código fuente como Checkstyle, PMD, etc. Informa sobre código duplicado, estándares de codificación, cobertura de código, complejidad ciclomática, errores potenciales, comentarios y diseño del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

Si comparamos con el análisis realizado antes del comienzo del proyecto de mantenimiento podemos ver lo siguiente:

■ Dominio:

- Paso de haber 18 bugs a 0.
- Paso de haber 10 vulnerabilidades a 0.
- Paso de haber 121 code smells a 9.

■ UI:

- Paso de haber 504 code smells a 474
- Paso de haber 16.3 % de duplicaciones a 13.2 %
(esto se debe al código auto-generado por NetBeans)

2.1.2. 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	2
LocalVariableNamingConventions	2
UnnecessaryLocalBeforeReturn	5
UnnecessaryFullyQualifiedName	82
ForLoopCanBeForeach	14
PositionLiteralsFirstInComparisons	43
UnusedImports	1
UnusedFormalParameter	106
UselessParentheses	2
MethodNamingConventions	2
ControlStatementBraces	1
SimplifyBooleanReturns	3
PackageCase	2
SingularField	229
UseCollectionIsEmpty	2
UnusedLocalVariable	2

PMD Report - Todo el proyecto

Si comparamos con el análisis realizado antes del comienzo del proyecto de mantenimiento podemos ver lo que no hay mucha diferencia como en SonarQube, si sumamos todas las violaciones a las reglas obtenemos que paso de haber 547 violaciones a 499. A pesar de ser una mejora, mirando detalladamente el reporte pudimos ver que una gran parte de las violaciones restantes vienen de código auto-generado por NetBeans en la UI, por lo que realizamos el mismo test solo sobre el dominio para ver los resultados.

Summary

Rule name	Number of violations
SimplifyBooleanReturns	3
ForLoopCanBeForeach	3

PMD Report - Dominio

Como podemos ver en el paquete de dominio hay un total de solamente 6 violaciones a las reglas, por lo que se asimila al resultado de SonarQube.

2.1.3. FindBugs

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

Los resultados fueron los siguientes:

Warning Type	Number
Bad practice Warnings	5
Dodgy code Warnings	3
Total	8

FindBugs Report

Si comparamos con el análisis realizado antes del comienzo del proyecto de mantenimiento podemos ver lo siguiente:

- Paso de haber 34 Bad practice warnings a 5.
- Paso de haber 3 Correctness warnings a 0.
- Paso de haber 2 Malicious code vulnerability warnings a 0.
- Paso de haber 5 Dodgy code warnings a 3

Esto se asimila al reporte de SonarQube donde hubo una mejora notable, si sumamos la cantidad de bugs vemos que paso de haber 44 a 8.

2.2. Cobertura de pruebas unitarias

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.

Para este proyecto nos hemos planteado la tarea el subir la cobertura de las pruebas unitarias a un 90 %.

La cobertura de las pruebas unitarias es un valor que indica que tanto del código se esta probando efectivamente en las pruebas (cobertura de sentencias), así mismo, la cobertura de ramas nos indica que porcentaje de las ramas se están probando.

Estos valores pueden obtenerse fácilmente utilizando la herramienta JaCo Coverage.

JaCoCoverage analysis of project "AlimentacionSaludable" (powered by JaCoCo from EcjEmma) > dominio

dominio

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
Sistema		89%		64%	52	134	10	274	1	43	0	1
Persona		63%		50%	14	29	12	38	2	12	0	1
Persona.TipoPersona		0%		n/a	4	4	3	3	4	4	1	1
Sistema.Paises		97%		n/a	2	4	0	4	2	4	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
Conversacion		98%		81%	3	22	1	42	0	14	0	1
Usuario		100%		100%	0	31	0	70	0	17	0	1
PlanAlimentacion		100%		100%	0	26	0	46	0	18	0	1
Profesional		100%		100%	0	21	0	45	0	10	0	1
Alimento		100%		100%	0	16	0	30	0	12	0	1
Ingesta		100%		100%	0	12	0	23	0	8	0	1
InformacionMensaje		100%		100%	0	13	0	23	0	9	0	1
ComposicionAlimento		100%		100%	0	6	0	12	0	5	0	1
Total	300 of 3.208	91%	85 of 324	74%	83	334	26	618	17	172	1	16

JaCoCoverage Report

Como se muestra en la figura, hemos logrado el 91 % en la cobertura de sentencias y el 74 % en la cobertura de ramas, esto se debe a que no se encontró la forma de testear el método guardarDatosSistema() el cual es el responsable de serializar los datos a persistir en memoria secundaria, el cual es responsable por 23 missed branches, por lo que se llegaría al 90 % en caso que JaCoCoverage no tuviera en cuenta este método.

2.3. 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.

Análisis previo: 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.

Análisis nuevo: Se soluciono el bug respecto a la sobre-escritura de las solicitudes de plan de alimentación.
Cabe destacar que en el alcance del proyecto no incluimos otras mejoras aplicadas a este factor.

Calificación anterior: 4

Calificación nueva: 3

■ 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.

Análisis previo: 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, colocándose 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.

Análisis nuevo: Mediante la reestructuración de algunas pantallas como lo fueron el login y creación plan de alimentación y agregado de labels logramos mejorar dicho factor en gran medida.

Calificación anterior: 1

Calificación nueva: 0

■ Control y Libertad del Usuario

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

Análisis previo: 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).

Análisis nuevo: No hemos implementado una por una las contra-partes del sistema pero si hemos implementado la posibilidad de eliminar los datos del sistema logrando así una leve mejora en este factor.

Calificación anterior: 4

Calificación nueva: 3

■ 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.

Análisis previo: 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).

Análisis nuevo: A pesar de no haber cambiado los iconos del sistema hemos agregado labels que indican la acción de cada uno de ellos, por lo que no queda duda al realizar una acción.

Calificación anterior: 1

Calificación nueva: 0

■ 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.

Análisis previo: El sistema no muestra los campos obligatorios en los formularios de registro ni presenta el login como una acción 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 File-Chooser para subir una imagen (png).

Análisis nuevo: Hemos mejorado en el inicio de sesión en dos pasos, controlando una de las fuentes de errores mas frecuente.

Calificación anterior: 2

Calificación nueva: 1

■ 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.

Análisis previo: 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.

Análisis nuevo: Hemos agregado labels a todos los iconos del sistema y la posibilidad de visualizar el nombre completo del usuario/profesional que inicio sesión.

Calificación anterior: 2

Calificación nueva: 0

■ 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.

Análisis previo: 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.

Análisis nuevo: Hemos implementado la posibilidad de cerrar la aplicación desde cualquier pantalla y mejoras en la resolución. A pesar de no haber mejorado el punto sobre las fechas consideramos que los otros aspectos tienen mayor peso, por lo tanto entendemos que la mejora en este punto fue completa.

Calificación anterior: 3

Calificación nueva: 0

■ 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.

Análisis previo: 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.

Análisis nuevo: Consideramos que en este punto solamente mejoramos en cuanto al agregado de los labels a los iconos para explicitar sus funcionalidades.

Calificación anterior: 2

Calificación nueva: 1

■ 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.

Análisis previo: 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.

Análisis nuevo: No hay variaciones sobre el análisis previo.

Calificación anterior: 0

Calificación nueva: 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

Análisis previo: 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.

Análisis nuevo: Se ha mejorado la ubicación del botón para obtener ayuda, colocándolo en la parte derecha-superior de la pantalla, también al mejorar el punto relacionado a la resolución de pantalla este mismo se puede visualizar en cualquier monitor.

Calificación anterior: 2

Calificación nueva: 0

2.3.1. Resultado

Heurística	Anterior	Nueva
Visibilidad del Estado del Sistema	4	3
Vinculo entre el Sistema y el Mundo Real	1	0
Control y Libertad del Usuario	4	3
Consistencia y Estándares	1	0
Prevención de Errores	2	1
Reconocimiento sobre Memoria	2	0
Flexibilidad y eficiencia de uso	3	0
Diseño minimalista	2	1
Ayuda para Reconocer, Diagnosticar y Recuperarse de Errores	0	0
Ayuda y Documentación	2	0
Promedio de usabilidad inicial: 2,1		
Promedio de usabilidad final: 0.8		

El resultado muestra que hemos mejorado notablemente la usabilidad del sistema.

3. Análisis de documentación

Como parte de los entregables que hemos definidos en el alcance del proyecto estaban decidimos crear nuevamente la documentación del sistema en donde:

- Se formulan los requerimientos funcionales de forma correcta
- Se formulan los requerimientos no funcionales de forma correcta.
- Se formulan los Casos de uso de forma correcta con capturas de pantalla correspondientes a cada uno de ellos.
- Se eliminan las secciones que no aportan valor, contienen errores o quedan desactualizadas (Log de Versiones, Reporte de defectos, Sesiones de prueba), permitiendo extender en un futuro dicha documentación con facilidad sin generar confusiones debido a estas secciones.

Estos cambios fueron incluidos en el documento incluido en la carpeta del obligatorio 2 llamado:

'Alimentación Saludable - Nueva Documentación'

4. Medición de la Calidad Final

Finalmente evaluaremos nuevamente la calidad del proyecto, utilizando como insumo los defectos restantes listados en la documentación del obligatorio 1, para esto 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.

Corrección	
Puntos Débiles	Puntos Fuertes
Código y Funcionalidades: N.A	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, aunque ahora es compatible con la mayor parte de monitores en el mercado. <i>Defecto/s U9</i>	
Documentación: N.A	
Pruebas Unitarias: N.A	
<u>Calificación:1</u>	

Confiabilidad	
Puntos Débiles	Puntos Fuertes
Código y Funcionalidades: N.A	
Usabilidad: - Resolución única de la interfaz gráfica, incompatible con varias computadoras, aunque ahora es compatible con la mayor parte de monitores en el mercado. <i>Defecto/s U9</i>	
Documentación: N.A	
Pruebas Unitarias: N.A	
<u>Calificación:1</u>	

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, aunque ahora es compatible con la mayor parte de monitores en el mercado. <i>Defecto/s U9</i> - Faltan contra partes de todas las funcionalidades del sistema. <i>Defecto/s U3</i> 	
Documentación: N.A	
Pruebas Unitarias: N.A	
Calificación:2	

Facilidad de Mantenimiento	
Puntos Débiles	Puntos Fuertes
Código y Funcionalidades: N.A	
Usabilidad: N.A	
Documentación: N.A	
Pruebas Unitarias: N.A	
Calificación:0	

Flexibilidad	
Puntos Débiles	Puntos Fuertes
Código y Funcionalidades: N.A	
Usabilidad: N.A	
Documentación: N.A	
Pruebas Unitarias: N.A	
<u>Calificación:0</u>	

Facilidad de Prueba	
Puntos Débiles	Puntos Fuertes
Código y Funcionalidades: N.A	
Usabilidad: N.A	
Documentación: N.A	
Pruebas Unitarias: N.A	
<u>Calificación:0</u>	

4.0.1. Resultado

Factor de calidad	Inicial	Final
Corrección	4	1
Confiabilidad	4	1
Facilidad de Uso	3	2
Facilidad de Mantenimiento	2	0
Flexibilidad	2	0
Facilidad de Prueba	3	0
<u>Promedio de calidad inicial: 3</u>		
<u>Promedio de calidad inicial: 0.7</u>		

Podemos concluir que la calidad del sistema ha sufrido una mejora notable pasando de estar a un punto menos del máximo a estar a menos de un punto del mínimo.

4.0.2. 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.















5. Plan de Metricas

En el correr del proyecto realizamos las siguientes actividades en el plan de métricas:

- **(1):** No realizamos un merge hasta que el Pull Request sea aprobado por el otro compañero.
- **(2):** Adjuntamos capturas de los analizadores de código en la gran mayoría de los Pull Request en donde se resuelve alguna tarea, ya que en algunos los cambios realizados son mínimos y no impactan o solo impactan en code smells de código auto-generado por NetBeans.
- **(3):** Llevamos cuenta de las regresiones que ocurren en JIRA y el .txt en GitHub.
- **(4):** Evaluamos las tareas marcadas como Done como QA para tratar de encontrar posibles bugs. (Gracias a esto encontramos un error en una solución marcada como done, y aplicamos un Fix, Pull Request #17)
- **(5):** Llevamos tracking de cuando las tareas fueron realizadas mediante el software de JIRA.
- **(6):** Realizar tests de cobertura de pruebas unitarias con los entregables que apliquen.

A continuación, se incluyen algunas evidencia de lo mencionado:

- **1 y 4:** Se incluye una imagen a continuación
- **2:** Se puede ver en todo el repositorio, a modo de ejemplo:
<https://github.com/Marcoo09/ID2/pull/23>
- **3:** Se encuentra en el archivo 'Registro de Esfuerzo.txt' como se menciona anteriormente
- **5:** Se incluye una imagen a continuacion
- **6:** Link al único Pull Request que aplica dicha métrica:
<https://github.com/Marcoo09/ID2/pull/22>

<input type="checkbox"/>		0 Open	<input checked="" type="checkbox"/>	26 Closed	Author ▾	Label ▾	Projects ▾	Milestones ▾	Reviews ▾	Assignee ▾	Sort ▾
<input type="checkbox"/>		Finalización Proyecto de Mantenimiento								1	
		#26 by matiassalles99 was merged 15 hours ago									
<input type="checkbox"/>		Registro de esfuerzo de pm								1	
		#25 by Marcoo09 was merged 19 hours ago • Approved									
<input type="checkbox"/>		Requerimientos funcionales y casos de uso								1	
		#24 by Marcoo09 was merged 3 days ago • Approved									
<input type="checkbox"/>		Corrección creación plan alimentación								1	
		#23 by matiassalles99 was merged 4 days ago • Approved									
<input type="checkbox"/>		Mejorar cobertura								1	
		#22 by Marcoo09 was merged 4 days ago									
<input type="checkbox"/>		Solucionar test que fallan								1	
		#21 by Marcoo09 was merged 6 days ago • Approved									
<input type="checkbox"/>		Unificacion en el tamaño de imagenes en la interfaz								3	
		#20 by matiassalles99 was merged 8 days ago • Approved									
<input type="checkbox"/>		Carga de datos de prueba y eliminacion de los mismos implementada								1	
		#19 by matiassalles99 was merged 8 days ago • Approved									
<input type="checkbox"/>		actualizar horas de pm y qa								1	
		#18 by Marcoo09 was merged 16 days ago									
<input type="checkbox"/>		Fix null pointer exception en login								1	
		#17 by Marcoo09 was merged 16 days ago • Approved									
<input type="checkbox"/>		Feature/cambios pantalla login								1	
		#16 by Marcoo09 was merged 16 days ago • Approved									
<input type="checkbox"/>		Feature/tener visibilidad de usuario logueado								1	
		#15 by Marcoo09 was merged 16 days ago • Approved									
<input type="checkbox"/>		Agregar boton de cerrar en todas las pantallas								1	
		#14 by Marcoo09 was merged 16 days ago • Approved									

1 y 4

Projects / Id2 / ID2-32 / ID2-6

TC2

[Attach](#) [Create subtask](#) [Link issue](#) [▼](#) [...](#)

Description

Existen 7 bugs de alta prioridad encontrados mediante las distintas herramientas, entre estos se encuentran:

Alta

- NullPointerException could be thrown
- Useless self-assignment
- Unwritten field
- Use try-with-resources or close FileOutputStream in a finally clause

Environment

None

Activity

Show: [Comments](#) [History](#) [Work log](#)

[MS](#)

Pro tip: press **M** to comment

1 [Share](#) [...](#)

[Done](#) [✓ Done](#)

Assignee

[MS](#) Matias Salles

Reporter

[MS](#) Matias Salles

Labels

None

Original Estimate

1h

Time tracking

[🕒](#) No time logged 1m remaining

Epic Link

[Corrección de Bugs de Alta...](#)

Sprint

None [+1](#)

Priority

[↑](#) High

BigTemplate

[Open BigTemplate](#)

[Show 3 more fields](#)

Components, Fix versions and Affects versions

Created May 10, 2020, 10:22 PM
Updated 8 days ago

[Configure](#)

6. Reflexión final

6.1. Analisis objetivos SMART

A continuación se realizará un análisis de los objetivos propuestos para este proyecto y si se han logrado cumplir.

En este proyecto se definieron los siguientes objetivos:

- **Objetivo:** 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'.
Conclusión: Hemos cumplido 21 de 23 tareas propuestas (91.3 %), por lo tanto hemos cumplido con este objetivo.
- **Objetivo:** 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.
Conclusión: A lo largo de los distintos entregables hemos testeado el programa (por parte de los QAs) y hemos identificado solamente una regresión (NullPointerException al intentar loguearse), por lo tanto hemos cumplido con este objetivo.
- **Objetivo:** Involucrar al cliente al final de cada entregable para recibir un feedback del mismo.
Conclusión: Hemos cumplido este objetivo dado que al final de cada sub-entregable realizamos la entrega al cliente para validar el mismo.
- **Objetivo:** 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.
Conclusión: Consideramos que este objetivo cumplido arduamente dado que hemos realizado refactoring del código, de las pruebas unitarias y un mecanismo de reviews cruzadas junto a análisis de código incluidos en los Pull Request (si aplica) para asegurar la calidad dando como resultado un producto mantenible a diferencia del entregado inicialmente.

6.2. Analisis del equipo

Consideramos que más allá de la corta duración del proyecto en este tiempo hemos crecido como equipo logrando una buena química y sincronización en el trabajo.

En las primeras dos semanas principalmente nos costó avanzar en paralelo y acostumbrarnos al proyecto pero con el tiempo supimos entender quien debía tomar cada tarea en particular (más allá de lo que estaba asignado) para maximizar el tiempo y productividad y así lograr los objetivos planteados.

Un ejemplo claro de esto es que en la segunda mitad del proyecto identificamos como fortaleza de uno de los integrantes del equipo que estaba ya acostumbrado al código del proyecto por ende iba a tener una velocidad mayor en las distintas implementaciones de funcionalidades, como conclusión de esto el Product Manager tomo como decisión asignar las tareas de pruebas unitarias y documentación al otro integrante dado que de esta forma se lograría maximizar el tiempo y el trabajo en paralelo lo cual resulto en costos menores a los planeados.

Consideramos que esto fue una decisión acertada que nos hizo llegar en tiempo y fecha con los entregables planificados.

Con respecto a los elementos de gestión de software hemos sido organizados tanto con Github como con Jira.

En Github hemos seguido la metodología Git Flow y hemos realizados reviews cruzados mediante Pull Requests buscando la mejor calidad del código lo que ha previsto de errores o baja de calidad global.

Con el correcto uso de Jira hemos tenido claro en todo momento que tareas tenía asignado quien y en que estado estaba (To Do, In Progress, Done). Esto nos ha ayudado principalmente en los momentos que no estábamos trabajando ambos al mismo tiempo.

Con respecto a los roles desempeñados, hemos intentado ser coherentes y cada uno luego de desarrollar una funcionalidad realizaba la validación de su correcto funcionamiento.

Como conclusión creemos que llegamos a un buen resultado y que hemos tenido un buen trabajo en equipo.

Nos supimos organizar correctamente y trabajar como equipo para enfrentar los distintos riegos y desafíos que se han presentado en el correr del proyecto.