

Universidad ORT Uruguay

Facultad de Ingeniería

Ingeniería de Software 2

Obligatorio 2

Tomás Dilema (209316)

Jorge Bruno Fernández (221961)

Juan Ignacio Olivera (223139)

Entregado como requisito de la materia Ingeniería de
Software 2

7 de diciembre de 2020

<https://www.overleaf.com/read/mnjzrcmgdgcj>

Declaraciones de autoría

Nosotros, Tomás Dilema, Jorge Bruno Fernández y Juan Ignacio Olivera, 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.

Resumen

Para la materia Ingeniería de Software 2, obligatorio 2, se plantea un proyecto llamado “Alimentación Saludable”, el cual debemos analizar para encontrar los errores que contiene, cumplir con las solicitudes de cambio requeridas y documentar el impacto, costo, organización y errores de este programa y de los cambios solicitados.

Índice general

1. Ajustes del Plan de Riesgo	2
1.1. Plan de contingencia para los riesgos mas prioritarios	3
2. Cambios a lineas base	5
3. Reportes QC	9
3.1. Nuevos objetivos de calidad definidos:	10
3.1.1. Métricas de Usabilidad	11
4. Cambios al ESRE	13
4.1. Casos de Uso	14
4.2. Casos de Prueba	16
5. Prueba exploratoria Final	20
6. Reporte de los cambios implementados y de los que no lo fueron	22
7. Reporte del análisis del estado de calidad final del software y productos asociados	30
7.1. Cobertura de las pruebas unitarias	30
7.1.1. Calidad de código	33
7.1.2. Interfaz gráfica de usuario	36
8. Reflexiones	45
9. Anexo	46
9.1. Heurísticas de Nielsen	46
Bibliografía	48

1. Ajustes del Plan de Riesgo

Se adjunta la tabla que muestra el impacto de los riesgos:

	B	C	D	E	F	G	H	I	J
1	Fecha de Ingreso	Componente	Especificación	Consecuencia	Tipo	Probabilidad	Impacto	Magnitud	Ubicación de Impacto
2	11/10/2020	Organizacional	La organización se reestructura implicando que otro equipo tome el proyecto a mantener	Problemas legales con la organización	Amenaza	0.2	5	1	Contrato Organizacional
3	11/10/2020	Costos	Mala estimación de costos (que el costo se aleje del presupuesto inicial)	Retraso en las actividades en las que están involucrados, no llegar a cumplir las metas posibles	Amenaza	0.8	2	1.6	Presupuesto Inicial y Cronograma de Trabajo
4	11/10/2020	Cronograma	Mala asignación de tareas a los paquetes	Incrementar el tiempo de desarrollo	Amenaza	0.6	2	1.2	Diagrama de Cronogramas de Trabajo
5	11/10/2020	Gestion	Mala definición de objetivos	Necesidad de cambios o adecuaciones en la arquitectura propuesta, mejor estimación de tiempo y asignación de personal.	Amenaza	0.7	3	2.1	Diagrama de Cronogramas de Trabajo y Objetivos SMART
6	11/10/2020	Cronograma	Falta de tareas	Falta de tareas a definir, faltantes de tareas en el ciclo de vida especificado y EDT	Amenaza	0.6	3	1.8	Diagrama de Cronogramas de Trabajo
7	11/10/2020	Equipo de Proyecto	Renuncia de algún integrante del proyecto	Atraso del proyecto según integrante, no alcanzar las metas	Amenaza	0.3	5	1.5	Contratos de trabajo
8	11/10/2020	Equipo de Proyecto	Incapacidad de un RRHH (Ej: baja de enfermedad de algún integrante)	Atraso del proyecto, búsqueda de un reemplazante para esa actividad	Amenaza	0.5	4	2	Diagrama de Cronogramas de Trabajo
9	11/10/2020	Gestion	Mala identificación de los Stakeholders	Que deban implementarse dichos ajustes, malas especificaciones generación de problemas críticos durante el mantenimiento	Amenaza	0.4	3	1.2	Listado de Interesados
10	11/10/2020	Gestion	Mala especificación de los cambios a realizar	Retraso en las diferentes etapas estipuladas en la especificación	Amenaza	0.5	3	1.5	Todo el proyecto
11	11/10/2020	Interoperabilidad	No es posible coordinar el uso del sistema con los sistemas más nuevo	Retraso en la etapa de transferencia tecnológica (Migración)	Amenaza	0.9	4	3.6	Todo el proyecto
12	11/10/2020	Legal	Problemas de certificación del ministerio, interferencia con otra organización	Que el equipo de desarrollo no pueda comenzar a elaborar el mantenimiento.	Amenaza	0.2	6	1.2	Certificación de habilitación Ministerial
13	11/10/2020	Personal	Renuncia de algún integrante del proyecto	Atraso del proyecto según integrante	Amenaza	0.5	4	2	Plan de RRHH, y Contratos de trabajo
14	11/10/2020	Proyecto	Estimación del esfuerzo de construcción sea mayor al estimado	Cliente insatisfecho, posible aumento de costos	Amenaza	0.5	4	2	Documento de especificación de nuevos Requerimientos
15	11/10/2020	Proyecto	Estimación del esfuerzo de construcción sea menor al estimado	Personal con tiempo libre, búsqueda de posibles mejoras	Ventaja	0.5	3	1.5	Solicitudes de cambios
16	11/10/2020	Proyecto	Los recursos asignados no estén disponibles para desarrollar las actividades	Retraso de los tiempos de puesta en Marcha	Amenaza	0.4	4	1.6	Documento del estado de ejecución del sistema
17	11/10/2020	Proyecto	Pautas y procedimientos poco claros	Retraso en la etapa de Análisis y posteriormente a todo el desarrollo del proyecto	Amenaza	0.4	3	1.2	Documento de especificación de nuevos Requerimientos
18	12/10/2020	Proyecto	Se necesitan más ciclos de test de los planificados	Se retrasa el cronograma	Amenaza	0.7	4	2.8	Documento Testing
19	12/10/2020	Reusabilidad	No es posible reusar o reutilizar paquete nuevos de trabajo (ej: JAVAFX) para la mejora de la calidad del sistema	Retraso de la implantación.	Amenaza	0.7	3	2.1	Lista de cambios realizados, Checklist de objetivos
20	12/10/2020	Proyecto	No es posible realizar alguna nueva funcionalidad	Retraso de la implantación, cambios en el cronograma, posibles reuniones con consultores y cliente	Amenaza	0.7	4	2.8	Lista de cambios realizados, Diagrama de Cronograma de Trabajo

Figura 1.1: Tabla de riesgos.

El riesgo de un proyecto es un evento o condición incierta que, de producirse, tiene un efecto positivo o negativo en uno o más de los objetivos del proyecto, tales como el alcance, el cronograma, el costo y la calidad. Para el proyecto se realizó una tabla con cada recompilación de distintos tipos de riesgos, tanto sea de cronograma, tecnológico, costo o calidad que engloba a todo el ciclo de vida del mismo. Se les ingresa una métrica, la probabilidad que suceda ese riesgo, el impacto relacionado a la magnitud (magnitud = probabilidad * impacto).

A continuación se detallarán los distintos ajustes de riesgo y las distintas medidas que se tomaron cuando ocurrieron los distintos riesgos. También se detallarán los distintos planes de contingencia ejecutados cuando ocurrieron dichos riesgos. Los riesgos que ocurrieron son aquellos que están marcados con la probabilidad en color rojo.

- **Bajo Impacto:** Magnitud: de 1 a 1.5.
- **Medio Impacto:** Magnitud: de 1.6 a 2.
- **Alto Impacto:** Magnitud: de 2.1 a 5.

Nuestro plan de respuesta para el riesgo identificado fue basarse en documentación y en especialista pero se optó atrasar las actividades sucesoras, y no realizar los cambios a Java Fx y realizarlos en Java Swing. Actualmente no encontramos ningún riesgo encontrado que tenga alto impacto que no fuera descripto anteriormente.

1.1. Plan de contingencia para los riesgos mas prioritarios

Acción de mitigación y contención para realización de un cambio en el sistema

Si existe en el transcurso del proyecto algún problema a la hora de implementar los cambios especificados, se recomienda para el desarrollo de ese cambio o tarea, ponerlo en pausa. Asesorarse en base a documentación, consultores o otros tipos de herramientas, ya sea investigando o consultando, esto va implicar seguramente algún cambio en la tabla de actividades, la estimación del esfuerzo aumentara, el diagrama de Gantt necesitara ajustarse así como también la implicación directa sobre el costo del proyecto. Una vez realizadas estas actividades previas se procederá a implementar el cambio correspondiente con los nuevos conocimiento o herramientas adquiridas.

Acción de mitigación y contención para el faltantes de tareas y actividades

En caso de una tarea sera necesario modificar la de actividades para agregar la tarea nueva y su estimación correspondiente de esfuerzo. Esto causara la modificación de los diagramas de cronograma (Gantt) agregando la tarea nueva, con su duración, y posponiendo las tareas que le siguen a esta. Además habría que reajustar el presupuesto inicial ya que ahora tenemos una nueva tarea que implica mas horas de trabajo y mas personal a disposición de esa nueva tarea.

Acción de mitigación y contención de no ser posible coordinar el uso del sistema con los sistemas más nuevo

En caso de que el uso del sistema no pueda ser interoperable y portable en las distintas plataformas de las distintos terminales, se procederá a realizar consulta con expertos tecnológicos, se revisara el proceso de build y modificación de los diagramas de cronograma (Gantt). Además habría que reajustar el presupuesto inicial ya que ahora se revisara el proceso de build del sistema y la puesta en marcha.

Acción de mitigación y contención para mala definición de los objetivos

En el caso de haber definido mal un objetivo SMART o de agregar nuevos objetivos en el proyecto, esto causaría una redefinición del mismo y además provocaría probablemente la adición de nuevas tareas y actividades así como nuevas métricas. Esto modificará el Gantt, el plan de calidad, el plan de métricas y así como también aumentara la estimación de costo y el presupuesto.

Acción de mitigación y contención por si se necesitan más ciclos de test de los planificados

En el caso de haber definido mal el set de pruebas, o haber diseñado mal las pruebas unitarias, se necesitara modificar el Gantt, dejando a posterior actividades de build y de puesta en marcha así como los informes finales de los cambios realizados con éxito.

Esta acción se tuvo tomada en cuenta antes del primer registro de avance para poder seguir avanzando sobre las etapas de desarrollo del proyecto, estuvo mal identificada su nivel de impacto y probabilidad.

Acción de mitigación y contención Para la implementación de Java Fx

En el caso de haber problemas con la implementación de Java Fx, se reestructura el diagrama Gantt, se habla con el equipo de diseñadores, y se deja estipulado el uso de Java Swing.

2. Cambios a lineas base

En esta sección se encuentran las nuevas versiones de los planes, como el diagrama Gantt, y otros documentos que hayan sido modificados en base al análisis del reporte de avance.

Como fue mencionado anteriormente, una causa de las variaciones de ciertas tareas fue el no poder integrar Java Fx en el proyecto, ya que se intentó cambiar el JDK del proyecto además de su build de compilación, esta implementación no fue posible por los tiempos y los pocos recursos que se tuvo a la hora de realizar a cabo estas acciones. Se agregaron actividades del informe de cierre, como los ajustes del plan de riesgo, y el reporte del estado final de la aplicación.

En los reportes finales también existen las actividades que las tuvimos en cuenta como por ejemplo:

- Identificar las lecciones aprendidas.
- Gestionar el intercambio y la transferencia de conocimiento.
- Archivar la información del proyecto para su uso futuro por parte de la organización.

Como a su vez también tuvimos en cuenta los activos de los procesos de la organización que pueden influir en este proceso. Se opto por utilizar la técnica juicio experto. Nuestro personal realizo las actividades de cierre administrativo. Aseguraron que el cierre del proyecto o fase se realice de acuerdo con los estándares apropiados.

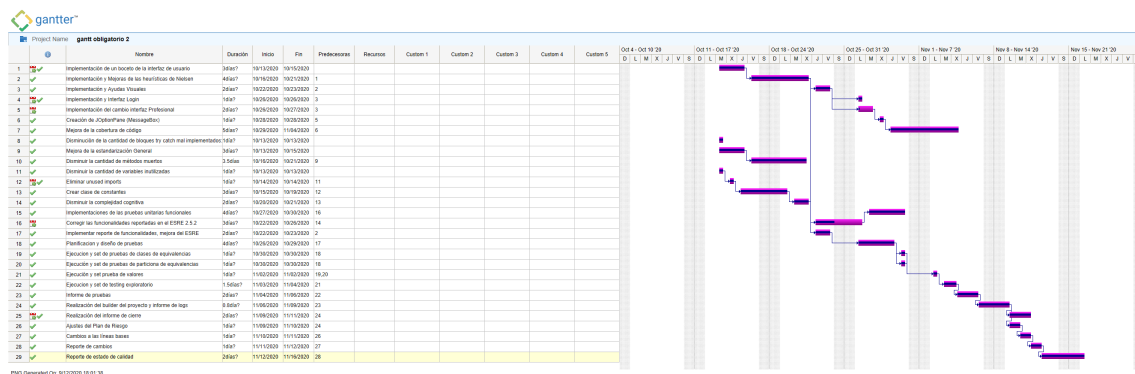


Figura 2.1: Diagrama Gantt completo




































Project Name		gantt obligatorio 2										
		Nombre	Duración	Inicio	Fin	Predecesoras	Recursos	Custom 1	Custom 2	Custom 3	Custom 4	Custom 5
1	 	Implementación de un boceto de la interfaz de usuario	3días?	10/13/2020	10/15/2020							
2		Implementación y Mejoras de las heurísticas de Nielsen	4días?	10/16/2020	10/21/2020	1						
3		Implementación y Ayudas Visuales	2días?	10/22/2020	10/23/2020	2						
4	 	Implementación y Interfaz Login	1día?	10/26/2020	10/26/2020	3						
5	 	Implementación del cambio interfaz Profesional	2días?	10/26/2020	10/27/2020	3						
6		Creación de JOptionPane (MessageBox)	1día?	10/28/2020	10/28/2020	5						
7		Mejora de la cobertura de código	5días?	10/29/2020	11/04/2020	6						
8		Disminución de la cantidad de bloques try catch mal implementados	1día?	10/13/2020	10/13/2020							
9		Mejora de la estandarización General	3días?	10/13/2020	10/15/2020							
10		Disminuir la cantidad de métodos muertos	3.5días	10/16/2020	10/21/2020	9						
11		Disminuir la cantidad de variables inutilizadas	1día?	10/13/2020	10/13/2020							
12	 	Eliminar unused imports	1día?	10/14/2020	10/14/2020	11						
13		Crear clase de constantes	3días?	10/15/2020	10/19/2020	12						
14		Disminuir la complejidad cognitiva	2días?	10/20/2020	10/21/2020	13						
15		Implementaciones de las pruebas unitarias funcionales	4días?	10/27/2020	10/30/2020	16						
16	 	Corregir las funcionalidades reportadas en el ESRE 2.5.2	3días?	10/22/2020	10/26/2020	14						
17		Implementar reporte de funcionalidades, mejora del ESRE	2días?	10/22/2020	10/23/2020	2						
18		Planificación y diseño de pruebas	4días?	10/26/2020	10/29/2020	17						
19		Ejecucion y set de pruebas de clases de equivalencias	1día?	10/30/2020	10/30/2020	18						
20		Ejecución y set de pruebas de particiona de equivalencias	1día?	10/30/2020	10/30/2020	18						
21		Ejecución y set prueba de valores	1día?	11/02/2020	11/02/2020	19,20						
22		Ejecucion y set de testing exploratorio	1.5días?	11/03/2020	11/04/2020	21						
23		Informe de pruebas	2días?	11/04/2020	11/06/2020	22						
24		Realización del bulider del proyecto y informe de logs	0.8día?	11/06/2020	11/09/2020	23						
25	 	Realización del informe de cierre	2días?	11/09/2020	11/11/2020	24						
26		Ajustes del Plan de Riesgo	1día?	11/09/2020	11/10/2020	24						
27		Cambios a las líneas bases	1día?	11/10/2020	11/11/2020	26						
28		Reporte de cambios	1día?	11/11/2020	11/12/2020	27						
29		Reporte de estado de calidad	2días?	11/12/2020	11/16/2020	28						

Figura 2.2: Lista de Actividades

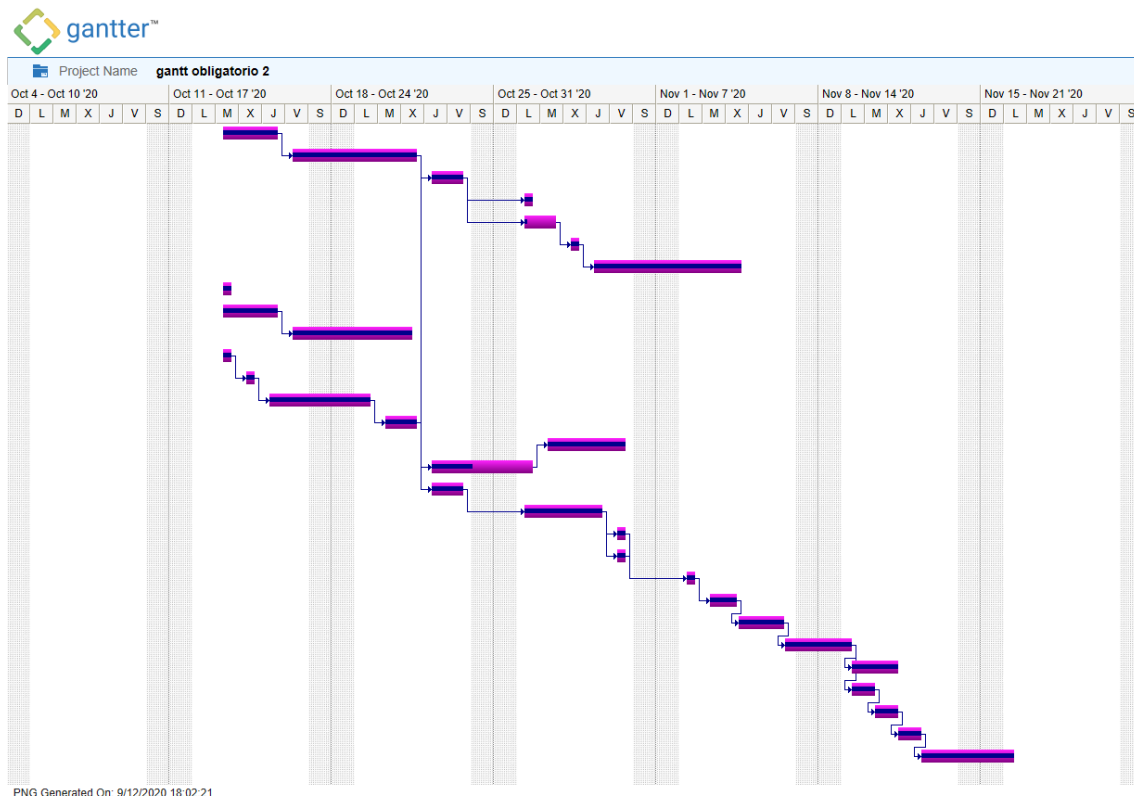


Figura 2.3: Diagrama Gantt

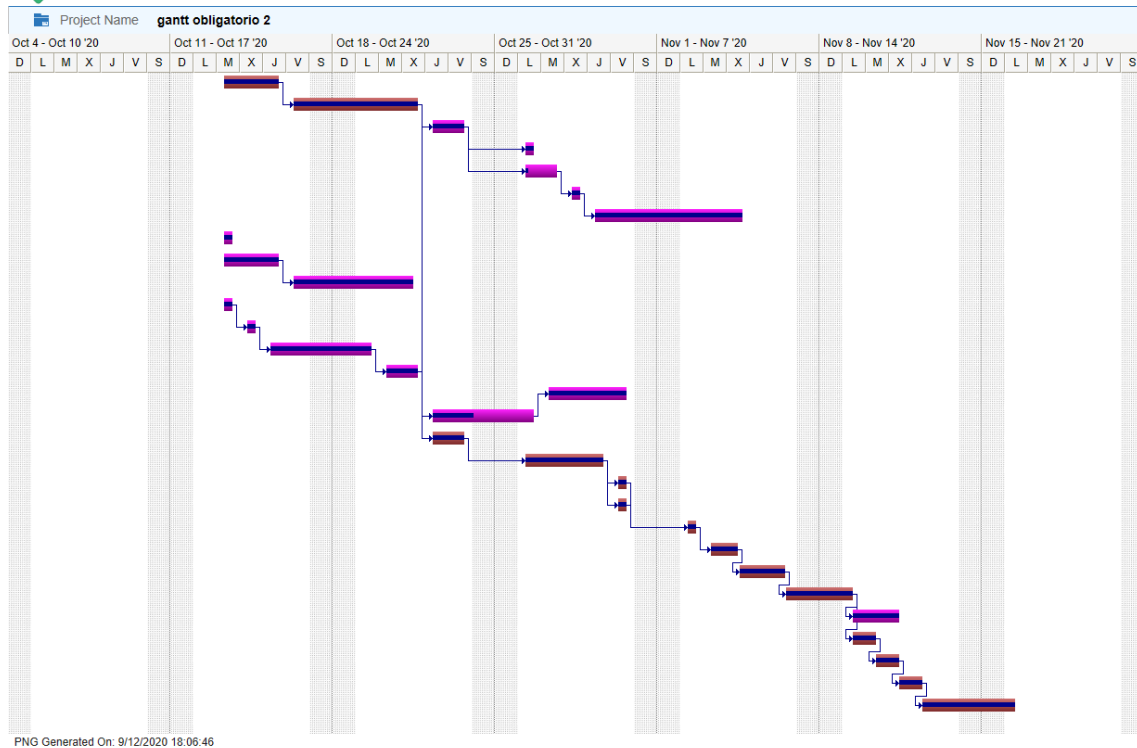


Figura 2.4: Diagrama Gantt- Ruta Critica



Figura 2.5: EDT - Diagrama

3. Reportes QC

Reporte Realizado el día 10 de noviembre. Realizado por: Project Manager:

- **Project Manager:** Ignacio Olivera
- **Gerente SQA:** Tomas Dilema
- **Aporte: SQA Developer:** Bruno Fernandez

Nombre Entregable	Esfuerzo en hs	Porcentaje de realización
Informe de avance	10	100 %
Mejora de documentación ESRE	10	40 %
Mejora Interfaz	20	98 %
Mejora de Codificación	20	70 %
Implementación de código	20	65 %
Elaboración de Testing	0	0 %
Informe de cierre	0	0 %
Reporte de Finalización	2	100 %

Tabla 3.1: Tabla Reporte QC

Reporte Realizado el día 28 de noviembre.

- **Project Manager:** Ignacio Olivera
- **Gerente SQA:** Tomas Dilema
- **Aporte: SQA Developer:** Bruno Fernandez

Nombre Entregable	Esfuerzo en hs	Porcentaje de realización
Informe de avance	10	100 %
Mejora de documentación ESRE	10	40 %
Mejora Interfaz	20	98 %
Mejora de Codificación	20	80 %
Implementación de código	40	98 %
Elaboración de Testing	2	70 %
Informe de cierre	1	50 %
Reporte de Finalización	2	100 %

Tabla 3.2: Tabla Reporte QC

3.1. Nuevos objetivos de calidad definidos:

En el siguiente capítulo se muestra evidencia de los objetivos definidos de calidad final del software y productos asociados. En este análisis exhaustivos del estado calidad final se tomo en cuenta los mismos atributos de calidad de software inicial, presentando en cuenta los factores de calidad de McCall.

- Corrección. Se baso en la corrección que es el grado en el que un programa satisface sus especificaciones y en el que cumple con los objetivos de la misión del cliente.
- Confiabilidad. El Grado en el que se espera que un programa cumpla con su función y con la precisión requerida.
- Eficiencia. Cantidad de recursos de cómputo y de código requeridos por un programa para llevar a cabo su función.
- Integridad. Grado en el que es posible controlar el acceso de personas no autorizadas al software o a los datos.
- Usabilidad. Esfuerzo que se requiere para aprender, operar, preparar las entradas e interpretar las salidas de un programa.
- Facilidad de recibir mantenimiento. Esfuerzo requerido para detectar y corregir un error en un programa (ésta es una definición muy limitada).
- Flexibilidad. Esfuerzo necesario para modificar un programa que ya opera. Susceptibilidad de someterse a pruebas. Esfuerzo que se requiere para probar un programa a fin de garantizar que realiza la función que se pretende.

- **Portabilidad.** Esfuerzo que se necesita para transferir el programa de un ambiente de sistema de hardware o software a otro.
- **Reusabilidad.** Grado en el que un programa (o partes de uno) pueden volverse a utilizar en otras aplicaciones (se relaciona con el empaque y el alcance de las funciones que lleva a cabo el programa).
- **Interoperabilidad.** Esfuerzo requerido para acoplar un sistema con otro.

A cada atributo se le asigna un porcentaje, lo definimos como nivel de calidad, un nivel bajo puede provocar:

- **Aumento de gastos**
- **Incremento de riesgos**
- **Ciertos grados de inoperabilidad.**
- **Provocación de una reducción de la confianza, tanto dentro como fuera de la organización.**

un nivel moderado puede provocar:

- **Niveles de degradación del producto en tiempo corto**
- **Aumento de gastos**
- **Nivel bajo:** 0 % a 50 %
- **Nivel moderado:** 50 % a 75 %
- **Nivel óptimo:** 75 % a 100 %

Estos factores son muy importantes para nosotros porque la calidad del productor es un “espejo” de la calidad del proyecto si el producto no presenta unos valores óptimos puede provocar lo que se menciono anteriormente.

3.1.1. Métricas de Usabilidad

Además agregamos las métricas de usabilidad categorizándolas cada una como se hizo en el estado inicial de calidad de la interfaz y en cada una se ingreso una métrica como:

- **Interactividad:** ¿Los mecanismos de interacción (por ejemplo, menús desplegables, botones, punteros) son fáciles de entender y usar?
- **Plantilla:** ¿Los mecanismos de navegación, contenido y funciones se colocan de forma que el usuario pueda encontrarlos rápidamente?
- **Legibilidad:** ¿El texto está bien escrito y es comprensible? ¿Las representaciones gráficas se entienden con facilidad?

- **Estética:** Estética: ¿La plantilla, color, fuente y características relacionadas facilitan el uso? ¿Los usuarios “se sienten cómodos” con la apariencia y el sentimiento de la aplicación?
- **Características de despliegue:** ¿La aplicación usa de manera óptima el tamaño y la resolución de la pantalla?
- **Sensibilidad temporal:** ¿Las características, funciones y contenido importantes pueden usarse o adquirirse en forma oportuna?
- **Personalización:** ¿La aplicación se adapta a las necesidades específicas de diferentes categorías de usuario o de usuarios individuales?
- **Accesibilidad:** ¿La aplicación es accesible a personas que tienen discapacidades?

Niveles definidos:

- **Nivel bajo:** 0 % a 50 %
- **Nivel moderado:** 50 % a 75 %
- **Nivel óptimo:** 75 % a 100 %

4. Cambios al ESRE

- Numero: RF1
- Nombre: Interfaz Login y Cerrar sesión
- Descripción: El sistema iniciara con una interfaz de Login, permitiendo a distintos tipos de usuarios ingresar a su cuenta cuenta.
- Prioridad: Importante
- Caso de uso: CU1

- Numero: RNF1
- Nombre: Ayudas Visuales
- Descripción: Ayudas visuales para saber en cual de las opciones principales nos encontramos parados.
- Prioridad: Deseable Caso de uso:

- Numero: RNF2
- Nombre: Eliminar Consultas Vacías
- Descripción: Quitar en el sistema la posibilidad de enviar consultas vacías (sin texto) a otro usuario. Prioridad: Deseable Caso de uso: CU 2

- Numero: RNF3
- Nombre: Implementación de contraseña para el ingreso de un usuario
- Descripción: Implementación la autenticación mediante contraseña, la cual debe ser alfanumérica de mas de 8 caracteres
- Prioridad: Importante
- Caso de uso: CU3

- Numero: RF2
- Nombre: Implementar
- Profesional agregar comida
- Descripción: Que el profesional pueda agregar comidas al sistema notificándose al administrador.
- Prioridad: Importante
- Caso de uso: CU4

4.1. Casos de Uso

- Identificador: CU 1
- Nombre: Interfaz Login y Cerrar sesión
- Descripción: El sistema iniciara con una interfaz de Login, permitiendo a distintos tipos de usuarios ingresar a su cuenta cuenta.
- Actores: Usuarios
- Precondiciones: Tener cuenta registrada.

En interfaz de Login:

Curso Normal:

- 1.El usuario ingresa el nombre de usuario.
- 2.El usuario ingresa contraseña
- 3.El usuario selecciona ingresar
- 4.Se ingresa a la cuenta del usuario

Curso Alternativo:

- 1.1 Se selecciona cerrar (cierra aplicación)
- 4.1 Usuario incorrecto
- 4.2 Contraseña incorrecta

- Identificador: CU 2
- Nombre: Eliminar consultas vacías Descripción: Quitar en el sistema la posibilidad de enviar consultas vacías (sin texto) a otro usuario.
- Actores: Usuarios

- Precondiciones: Tener cuenta registrada.

Curso Normal:

- 1.Ingreso la consulta
- 2.Consulta enviada

Curso alternativo:

- 2.1 No se ingresa ningún valor en el textbox, muestra mensaje “Consulta errónea verifique los campos”.

- Identificador: CU 3
- Nombre: Implementar contraseña para el registro de un usuario
- Descripción: Implementación la autenticación mediante contraseña, la cual debe ser alfanumérica de mas de 8 caracteres
- Actores: Usuarios
- Precondiciones:

Curso Normal

Interfaz de registro de usuario:

- 1.Ingreso Nombre de usuario
- 2.Ingreso apellido
- 3.Ingreso nombre de usuario
- 4.Ingreso contraseña
- 5.Ingreso nacionalidad
- 6.Ingreso Fecha nacimiento
- 7.Ingreso Altura
- 8.Ingreso Peso
- 9.Selecciono sexo
- 10.Selecciono preferencias
- 11.Se ingresa correctamente el usuario

Curso Alternativo:

- 2.1 Ingreso apellido menor a 8 caracteres, muestra mensaje “usuario no fue registrado correctamente”, y se muestra mensaje “contraseña no valida” en el textbox de contraseña.

- Identificador: CU 4
- Nombre: Agregar comida al sistema.
- Descripción: Que el profesional pueda agregar comidas al sistema notificándose al administrador.

- Actores: Profesional
- Precondiciones: Ser Profesional

Curso Normal:

- 1.Ingreso nombre del alimento
- 2.Selecciono tipo de alimento
- 3.Selecciono principales nutrientes

Curso Alternativo:

- 1.1 No se ingresa ningún nombre, al querer registrar muestra mensaje “Alimento no pudo ser registrado correctamente”
- 1.2 Se ingresa un nombre de alimento ya registrado, se muestra mensaje “Alimento no pudo ser registrado correctamente”.

4.2. Casos de Prueba

Caso de Prueba Nuevas funcionalidades:

Implementación Contraseña en registro de usuario y profesional:

Tomamos los mismos valores de las pruebas realizadas por los desarrolladores originales, corrigiendo sus errores bases y agregando los nuevos casos (Contraseña).

Registro de Usuario:

Escenario	Nombre	Apellido	Nombre Usuario	Nacionalidad	Fecha Nacimiento	Altura	Peso	Sexo	Restricciones	Preferencias	Resultado
1	Natalia	Elvas	ElvNatii	Uruguay	20-03-1998	160	50	F	Ninguna	Ninguna	Usuario registrado correctamente
2		Elvas	ElvNatii	Uruguay	20-03-1998	160	50	F	Ninguna	Ninguna	El nombre no puede ser vacío
3	Natalia		ElvNatii	Uruguay	20-03-1998	160	50	F	Ninguna	Ninguna	El apellido no puede ser vacío
4	Natalia	Elvas		Uruguay	20-03-1998	160	50	F	Ninguna	Ninguna	El nombre de usuario no puede ser vacío
5	Natalia	Elvas	JoacoP	Uruguay	20-03-1998	160	50	F	Ninguna	Ninguna	Nombre de usuario no válido
6	Natalia	Elvas	ElvNatii	Uruguay		160	50	F	Ninguna	Ninguna	Fecha de nacimiento no válida
7	Natalia	Elvas	ElvNatii	Uruguay	20-03-1998		50	F	Ninguna	Ninguna	La altura no puede ser vacía
8	Natalia	Elvas	ElvNatii	Uruguay	20-03-1998	300	50	F	Ninguna	Ninguna	El dato debe de estar entre 0 y 265
8	Natalia	Elvas	ElvNatii	Uruguay	20-03-1998	-1	50	F	Ninguna	Ninguna	El dato debe de estar entre 0 y 265
8	Natalia	Elvas	ElvNatii	Uruguay	20-03-1998	a	50	F	Ninguna	Ninguna	Debe ingresar un dato numérico
9	Natalia	Elvas	ElvNatii	Uruguay	20-03-1998	160		F	Ninguna	Ninguna	El peso no puede ser vacío
10	Natalia	Elvas	ElvNatii	Uruguay	20-03-1998	160	600	F	Ninguna	Ninguna	El dato debe de estar entre 0 y 265
10	Natalia	Elvas	ElvNatii	Uruguay	20-03-1998	160	-1	F	Ninguna	Ninguna	El dato debe de estar entre 0 y 265
10	Natalia	Elvas	ElvNatii	Uruguay	20-03-1998	160	a	F	Ninguna	Ninguna	Debe ingresar un dato numérico

Escenario	Nombre	Apellido	Nombre de Usuario	Contraseña	Nacionalidad	Fecha de nacimiento	Altura	Peso	Sexo	Restricciones	preferencias	resultados
11	Natalia	Elvas	ElvNatii		Uruguay	20/03/1998	160	50	F	Ninguna	Ninguna	No se pudo registrar el usuario correctamente, contraseña no válida
11	Natalia	Elvas	ElvNatii	ElvNati (Menor a 8 caracteres incorrecto)	Uruguay	20/03/1998	160	50	F	Ninguna	Ninguna	No se pudo registrar el usuario correctamente, contraseña no válida

- Análisis de Valores limites:

Contraseña:

- Limite superior negativo: 8
- Límite Inferior negativo: 0
- Límite inferior positivo: 9

Altura:

- Limite superior negativo: 264 inferior negativo: 0 inferior positivo: 1
- Limite superior positivo: 265

Peso:

- Limite superior negativo: 264
- Limite inferior negativo: 0
- Limite inferior positivo: 1
- Limite superior positivo: 265

Registro Profesional:

Escenario	Nombre	Apellido	Nombre Usuario	Fecha Nacimiento	Nombre Título	Fecha Graduación	País de Graduación	Resultado
1	Joaquin	Pascual	JoacoP	20-01-1995	Nutricionista	25-06-2017	Uruguay	Profesional registrado correctamente
2		Pascual	JoacoP	20-01-1995	Nutricionista	25-06-2017	Uruguay	El nombre no puede ser vacío
3	Joaquin		JoacoP	20-01-1995	Nutricionista	25-06-2017	Uruguay	El apellido no puede ser vacío
4	Joaquin	Pascual		20-01-1995	Nutricionista	25-06-2017	Uruguay	El nombre de usuario no puede ser vacío
5	Joaquin	Pascual	JoacoP ya existente en el sistema	20-01-1995	Nutricionista	25-06-2017	Uruguay	Nombre de usuario no válido
6	Joaquin	Pascual	JoacoP		Nutricionista	25-06-2017	Uruguay	Fecha de nacimiento no válida
7	Joaquin	Pascual	JoacoP	20-01-1995		25-06-2017	Uruguay	El nombre de usuario no puede ser vacío
8	Joaquin	Pascual	JoacoP	20-01-1995	Nutricionista		Uruguay	Fecha de graduación no válida

■ Análisis de Valores límites:

Contraseña (Cantidad de caracteres):

- Límite superior negativo: 8
- Límite Inferior negativo: 0
- Límite inferior positivo: 9

Caso de prueba Redactar Consulta:

Pruebas realizadas a error base encontrado, que permite realizar consultas vacías.

Escenario	Destinatario	Asunto	Mensaje	resultados
1	Andres	Dieta Marzo	Necesito reemplazar los lacteos	Consulta enviada
2	Andres		Necesito reemplazar los lacteos	Consulta erronea, verifique los campos
3	Andres	Dieta Marzo		Consulta erronea, verifique los campos
4	Andres			Consulta erronea, verifique los campos

Casos de Prueba para el Login:

Escenario	Destinatario	Asunto	Mensaje	resultados
1	Andres	Dieta Marzo	Necesito remplazar los lacteos	Consulta enviada
2	Andres		Necesito remplazar los lacteos	Consulta erronea, verifique los campos
3	Andres	Dieta Marzo		Consulta erronea, verifique los campos
4	Andres			Consulta erronea, verifique los campos

- Escenario 3: Se ingresa una contraseña invalida.
- Escenario 2: Se ingresa un usuario que no esta registrado en el sistema.
- Análisis de Valores limites:

Contraseña (Cantidad de caracteres):

- Límite superior negativo: 8
- Límite Inferior negativo: 0
- Límite inferior positivo: 9

Casos de Prueba para Registro de comida(Gestión de alimentos Profesional):

Escenario	Nombre	Tipo de alimentos	Principales nutrientes	resultados
1	Almendras	Cereal	*Fibra *Proteina	Alimento registrado correctamente
2	Almendras	Cereal	*Fibra *Proteina	Alimento registrado correctamente
3		Cereal	*Fibra *Proteina	Alimento no pudo ser registrado correctamente

- La entrada Tipo de alimentos es un componente de tipo Lista Desplegable
- La entrada Principales nutrientes es un CheckBox

5. Prueba exploratoria Final

Se realizan pruebas exploratorias en cual la misión principal es la de encontrar defectos y las faltas de aplicación de las 10 heurísticas de usabilidad de Jakob Nielsen, este mal manejo de las heurísticas pueden provocar una desmejora de experiencia por parte del usuario en cual puede a llegar realizar un manejo erróneo del sistema.

- **Misión Principal:** Validar el correcto funcionamiento del flujo de la aplicación.
- **Áreas:** OS-Windows 10
- **Fecha realización** 10/12/2020
- **Tiempo de Sesión** 50 minutos
- **Tester:** Bruno Fernandez
- **División de tareas:**
 - **Diseño y casos de prueba:** 88 %
 - **Investigación y Reporte de incidentes :** 2 %
 - **Armado de la sesión:** 10 %
 - **Misión vs Oportunidad:** 95 % - 5 %
- **Archivos de datos:** N/A
- **Notas de prueba:** Se verifica el correcto funcionamiento de los botones, labels, dropdowns, cierre de ventana, una buena persistencia, registros de los campos validos y mensajes de retro alimentación funcionando.
- **Riesgos:** N/A
- **Defectos:** BUG 1.0.0 Error en el botón perfil de profesional, no redirige a ningún panel.



Figura 5.1: Botón perfil de Profesional

- **Defectos:** BUG 1.0.1 Fechas no actualizadas.

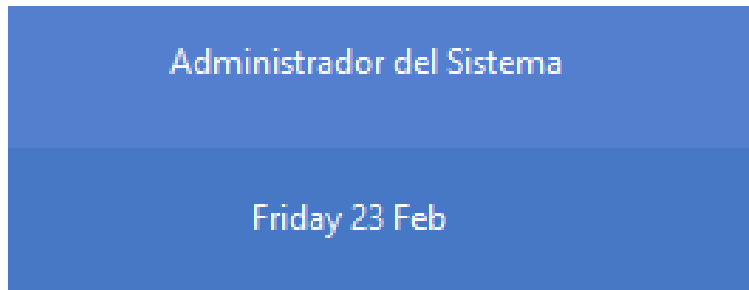


Figura 5.2: Label Fecha del día de hoy

- **Inconveniente:** N/A

6. Reporte de los cambios implementados y de los que no lo fueron

- **Ayudas visuales. Implementado** Se agregaron mensajes de confirmación, de error y de advertencia cada vez que el usuario realiza una acción importante para determinar si realmente desea hacerla, si ingreso bien lo campo y por consiguiente si realizo los pasos correctamente. Estas acciones van de la mano con las heurísticas 5, 9 y 10 de Nielsen.
- **ABM profesional. Implementado** El usuario administrador tiene la posibilidad de realizar altas, modificaciones y bajas de los profesionales, cambiar sus atributos y además consultarlos.

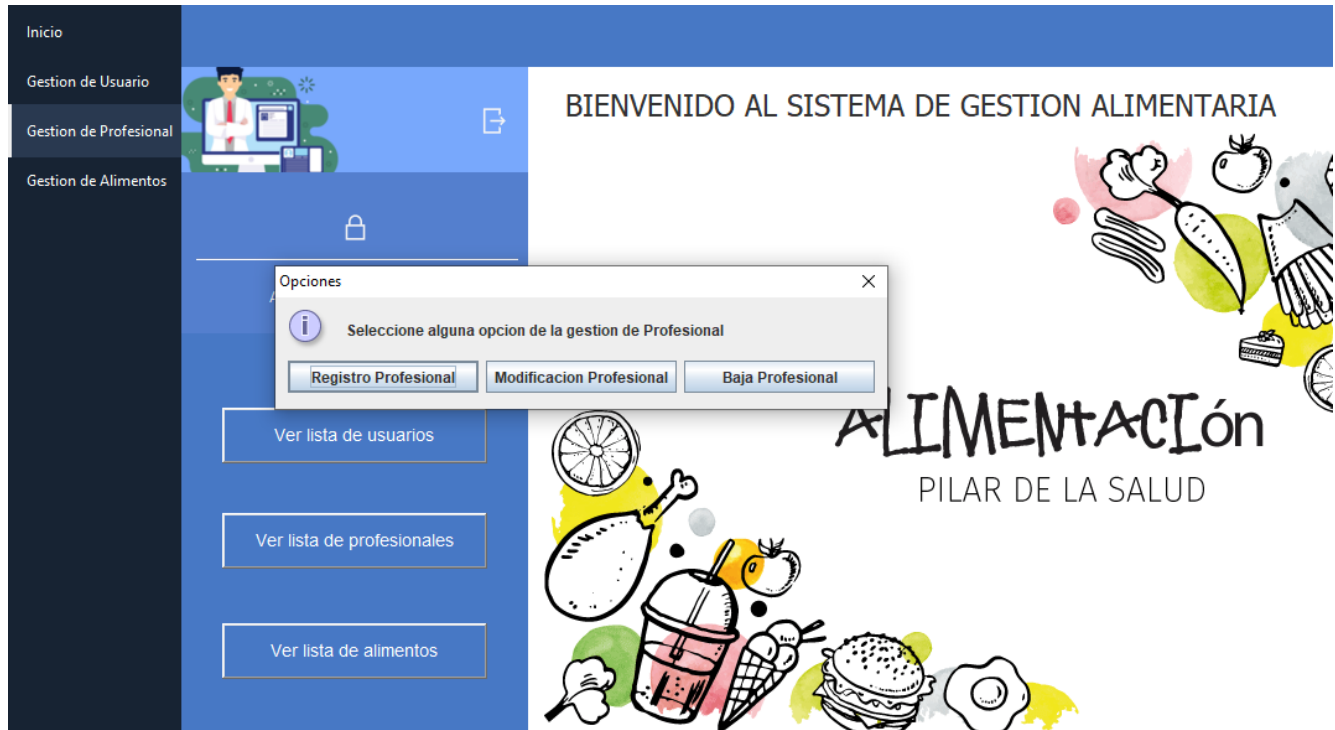


Figura 6.1: ABM profesional.

- **ABM usuario. Implementado** EL usuario administrador tiene la posibili-

dad de realizar altas, modificaciones y bajas de los usuarios, cambiar sus atributos y además consultarlos.

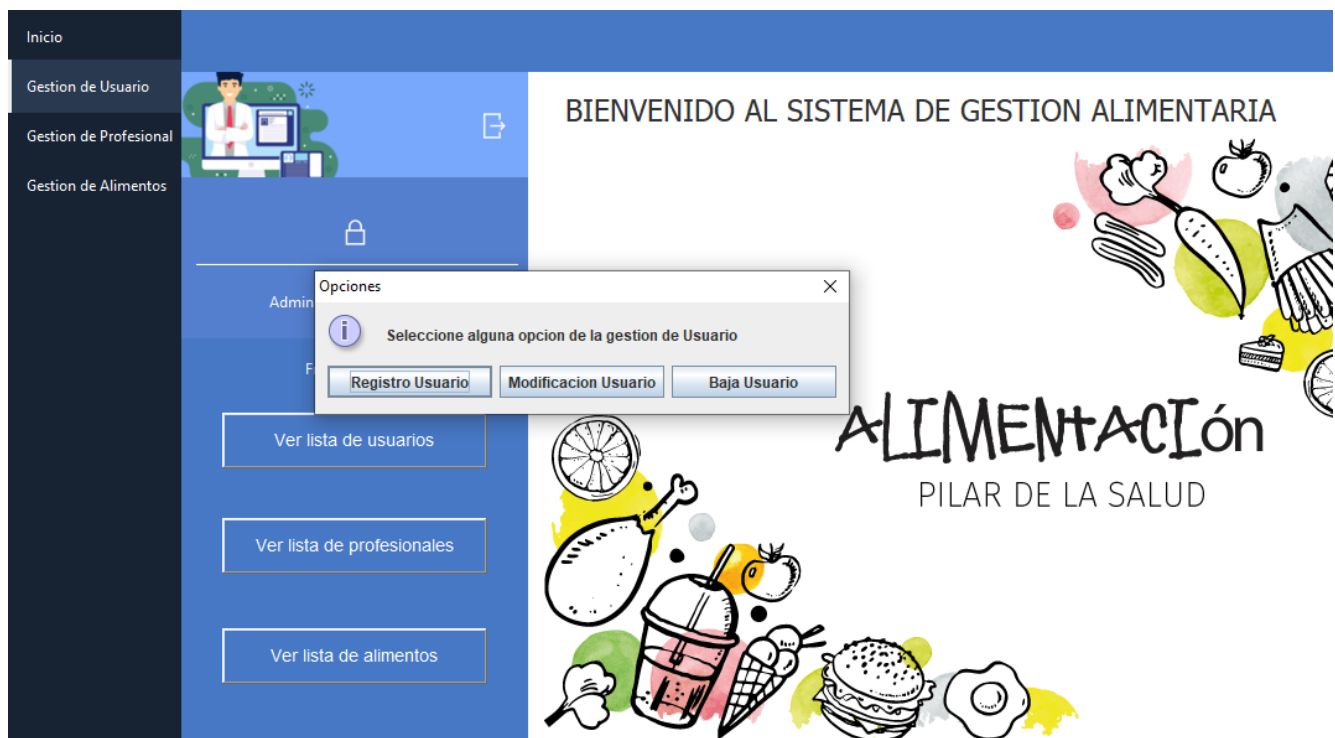


Figura 6.2: ABM usuario.

- **Consulta No Vacía.Implementado** Se elimino el "bug" que poseía la aplicación de realizar una consulta vacías, ahora además tampoco se podrá realizar una consulta con asunto vacío. Se implementaron estructura if y además popUS en caso de realizar consulta vacía.
- **Consulta Dobles.Implementado** Se elimino el "bug" que poseía la aplicación de realizar una consulta doubles, esto sucedía por que en el objeto sistema se le agregaba dos veces el mensaje, el mal manejo de las pruebas exploratorios y de testing basado en escenario pudo haber evitado estos errores.
- **Implementación de un password.Implementado** Todos los usuarios del sistema, tienen registrado un password, que les posibilita iniciar sesión al usuario, con esto el sistema adquiere mas robustez y mejora ese vacío de seguridad.
- **Mejorar la cobertura de código.Implementado** Fue implementado en un 93 % como fue estipulado en la métrica óptima que se estipulo para este aseguramiento de calidad. Se mejoro este apartado para determinar las partes críticas del código que no han sido comprobadas y las partes que ya lo fueron, además se utilizó como técnica de optimización, para llevar a cabo una eliminación de código muerto, más específicamente para detectar código inalcanzable, todo esto gracias al compilador.

- **Arreglar Test existentes.Implementado** Se llevo a cabo el arreglo del código de los test, ya que estos presentaban errores, tanto en los mensajes test, como plan de alimentación test, estos presentaban excepciones.

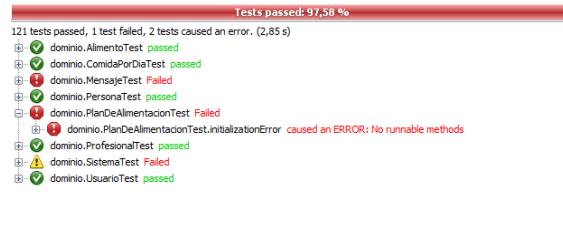


Figura 6.3: Tests sin arreglar.

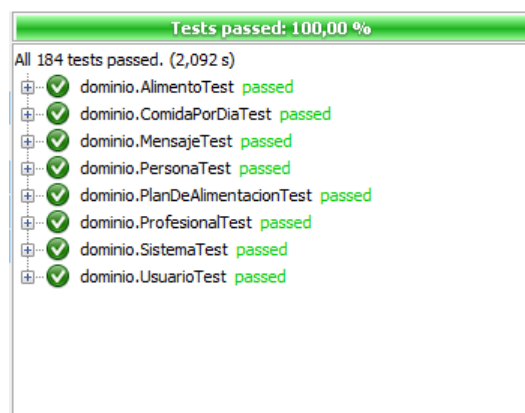


Figura 6.4: Tests nuevos implementados y corregidos.

- **Modificación de los paneles de los usuarios.Implementado** En nuestro plan de modificación de los paneles de los usuarios, en una primera instancia se tomo en cuenta las implementaciones de los objetos con Java F, este agrega toques modernos a su proyecto y hace que agregar animación y efectos especiales sea muy fácil. FX también es la opción para las personas que desarrollan aplicaciones móviles, por ende también se pensó en la escalabilidad, pero la vieja herramienta de compilación y el toolkit también casi un sistema viejo, no hizo imposible integrarlo. Al utilizar java Swing se apostó a un diseño mas juvenil, y integrado a las herramientas visuales del mercado, se fue por un estilo METRO UI, Donde el botón se convierte en ventana, y los objetos tiene un prevalencia geométrica.



Figura 6.5: Panel Administrador.

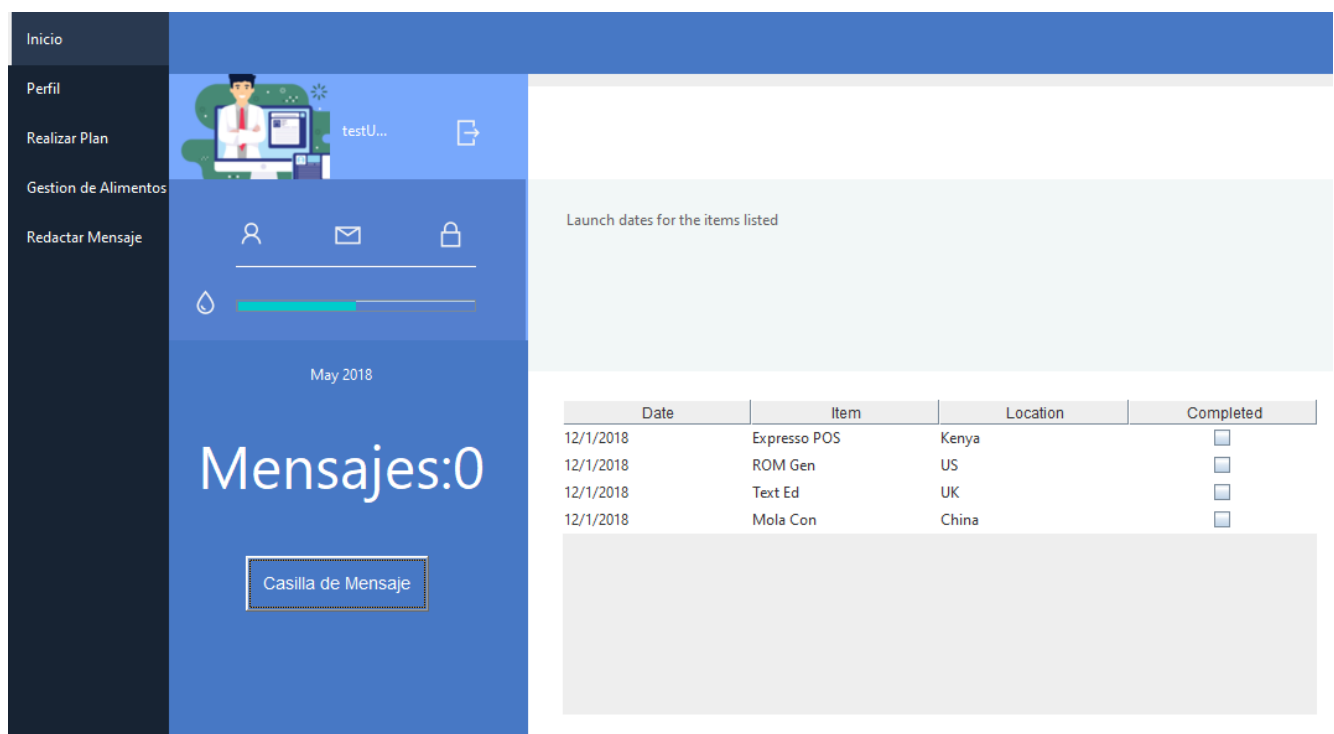


Figura 6.6: Panel Profesional.



Figura 6.7: Panel Usuario.

- **Reporte de funcionalidades del sistema. Implementado.** Para esta sección lo que se hizo fue revisar la documentación inicial en el primer periodo, es decir la especificación de requerimientos se revisaron, los requisitos y los casos de uso. Se modificaron la especificaciones de los distintos requerimientos (funcionales y no funcionales) los que coincidían se dejaron igual, si hubo modificaciones directas al requerimiento o este cambio debido a otro se modifico la especificación y se relabro y los nuevos cambios correspondientes a requerimientos, se elaboraron nuevas especificaciones de los mismos. En caso de ser necesario se modificaron los distintos casos de uso relacionados a los distintos requerimientos, esto sucede porque se cambio algún flujo de la interfaz y ya no se interactúan de la misma manera. Y de las nuevas funcionalidades implementadas se realizaron nuevos casos de uso.
- **Eliminación del BUG registrar Usuario. Implementado.** El sistema daba la posibilidad de registrar un usuario sin su nombre, esto le quita una gran parte de robustez, Robusto es un sistema que goza de buena salud y que brinda garantías de que va a continuar teniendo buena salud. Algunos síntomas de un sistema robusto son:

La capacidad de ser modificado sin introducir errores (opuesto a error prone)

Durabilidad del sistema funcionando correctamente (no aparecen errores aleatorios)

Claramente estos errores le quitaban esa cualidad al sistema, por ende realizo una revisión de los datos de ingreso, las estructuras y los métodos internos.

Existían campos que no se validaban correctamente.

Registro Usuario

Foto de Perfil:

Nombre:

Apellidos:

Nombre de Usuario:

Nacionalidad:

Fecha de Nacimiento:

Altura:

Peso:

Sexo: ☒ Masculino ☐ Femenino

Restricciones: ☐ Celíaco ☒ Intolerante a la lactosa ☐ Diabético ☐ Hipertensión

Preferencias: ☐ Vegano ☒ Vegetariano ☐ Macrobiótico ☐ Orgánico

Usuario registrado correctamente

Aceptar

Figura 6.8: Registro Usuario viejo.

Registro Usuario

Foto de Perfil:

Nombre:

Apellidos:

Nombre de Usuario:

Contraseña:

Nacionalidad:

Sexo: ☒ Masculino ☐ Femenino

Restricciones: ☐ Celíaco ☐ Intolerante a la lactosa ☐ Hipertensión ☐ Diabético

Preferencias: ☐ Orgánico ☐ Vegetariano ☐ Macrobiótico ☐ Vegano

Usuario no pudo ser registrado correctamente

Aceptar

Figura 6.9: Registro Usuario nuevo.

- **Mejora de la calidad de la codificación. Implementado un 60 por ciento** El sistema poseía métodos muertos, existía código idéntico o muy similar en más de una ubicación, clases que no hacen nada, como por ejemplo algunas clases de prueba, para nosotros es muy importante tener un código mantenible para posibles mejoras y futuras implementaciones, pensando en la escalabilidad y eficiencia del sistema. Por ende llevamos a cabo el proceso de refactoring, este significa hacer mejoras a un programa para frenar la degradación que va sufriendo por los cambios realizados. Por falta de tiempo no se pudo mejorar Excesivo uso de literales, como enums, donde cada clase poseía el mismo enum, código de constantes duplicado en el código.
- **Interfaz de Inicio de Sesión. Implementado** Se creo una interfaz que permite iniciar sesión a los diferentes usuarios, tanto administrador, profesional, usuario. Los diseñadores fueron por un estilo METRO Modern UI, este es el nombre de la interfaz de usuario, desarrollado por Microsoft, principalmente para su uso en Windows Phone. En términos generales, es una interfaz plana, con colores básicos y diseños geométricos, con una movilidad horizontal o vertical.

X

Bienvenido al Menu Principal de Alimentacion Saludable



The login form consists of two input fields. The first field is for the username, indicated by a person icon and a vertical line. The second field is for the password, indicated by a lock icon and a series of dots. Both fields have a green underline.

Login

Figura 6.10: Login Usuario.



Figura 6.11: Inicio de Sesión.

- **Mejorar la calidad general de la aplicación. Implementado** Este cambio tiene incluido otros cambios como el de mejorar las ayudas visuales, iconos en el sistema. Y hasta cierto punto las nuevas funcionalidades implementadas y las existentes. También fue mejorada la calidad de código que nos posibilita la mantenibilidad del código, la aplicación. Se completaron todos métodos tryCatch ahora en cada uno de estos métodos si ocurre el catch se despliega un mensaje al usuario dándole feedback. Fueron reducidas las líneas que superaban los 80 caracteres. Se borraron los métodos muertos, las lazy class, Se usaron las fixtures de las pruebas unitarias JUnit para mejorar la codificación de las mismas y mantener una mejor práctica de codificación de pruebas unitarias. Se cubrieron varias ramas de los métodos para verificar si estos métodos realizan las operaciones o la acción correcta.

7. Reporte del análisis del estado de calidad final del software y productos asociados

7.1. Cobertura de las pruebas unitarias

dominio

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Cla
Sistema		62%		100%	2	20	30	78	2	18	0	
Alimento		87%		62%	6	20	11	49	3	16	0	
Usuario		97%		67%	8	43	6	98	4	37	0	
Profesional.Pais		97%		n/a	2	4	0	5	2	4	0	
Alimento.TipoAlimento		94%		n/a	2	4	0	4	2	4	0	
Alimento.Nutrientes		91%		n/a	2	4	0	3	2	4	0	
Sistema.tipoUsuario		81%		n/a	2	4	0	2	2	4	0	
Usuario.Nacionalidades		98%		n/a	1	4	0	6	1	4	0	
Usuario.Restricciones		93%		n/a	1	4	0	2	1	4	0	
Usuario.Preferencias		93%		n/a	1	4	0	2	1	4	0	
Persona		97%		100%	1	17	1	35	1	14	0	
ComidaPorDia		98%		80%	2	13	1	27	0	8	0	
Profesional		100%		100%	0	16	0	34	0	14	0	
PlanDeAlimentacion		100%		n/a	0	17	0	34	0	17	0	
Mensaje		100%		n/a	0	10	0	20	0	10	0	
Total	186 of 2.629	93%	9 of 44	80%	30	184	49	399	21	162	0	

Figura 7.1: Cobertura - JaCoCoverage.

Como muestra la figura, la cobertura de las pruebas unitarias tras la implementación de las nuevas funcionalidades y la implementación de las nuevas pruebas a aumentado. Todavía existen ramas que faltan de cubrir, la clases sistema existen métodos como el de persistencia que no se pudo realizar por falta de tiempo, pero nunca nos alejamos del objetivo de llegar al 93 % de las ramas cubiertas. Esta mejora de las coberturas nos llevo un gran esfuerzo de recursos y tiempo pero adentro de las estipuladas, paralelamente se realizaron los métodos fixtures como se previo para mejorar la codificación y mantenibilidad de las pruebas de JUnit .Adicionalmente, vemos que esta herramienta puede ser sumamente útil para mejorar la calidad de nuestras pruebas y nos ayudan a asegurar que efectivamente estamos considerando todas las rutas y escenarios posibles en nuestra aplicación. Se llego al objetivo dentro de los plazos estimados, por lo tanto no hubo que recortar las tareas que

se consideraron menos relevantes. A futuro se recomienda mejorar los caminos de persistencia de la clase sistema, y poner el énfasis en el estudio, análisis y por consiguientemente mejora de la cobertura en esa clase, a su vez también se recomienda realizar las pruebas unitarias para los handlers de manejo de datos nuevos que se implementaron, estos a no ser de la clase de dominio no se tomaron en cuenta.

Sistema




















Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty
● cargarSistema()		0%		n/a	1	1
● guardarSistema()		0%		n/a	1	1
● pedidoDatoNumerico(int, int, int)		85%		100%	0	3
● Sistema()		100%		n/a	0	1
● Sistema(ArrayList, ArrayList, ArrayList, Sistema.tipoUsuario, HashMap)		100%		n/a	0	1
● inicializoListaTiposDeUsuario()		100%		n/a	0	1
● setListaTiposDeUsuario(Sistema.tipoUsuario[])		100%		n/a	0	1
● setUsuarioAdministrador(String, String)		100%		n/a	0	1
● getListaTiposDeUsuario()		100%		n/a	0	1
● setListaAlimentos(ArrayList)		100%		n/a	0	1
● setListaUsuarios(ArrayList)		100%		n/a	0	1
● setListaProfesionales(ArrayList)		100%		n/a	0	1
● setUsuarioActivo(Sistema.tipoUsuario)		100%		n/a	0	1
● getListaUsuarioAdministrador()		100%		n/a	0	1
● getListaAlimentos()		100%		n/a	0	1
● getListaUsuarios()		100%		n/a	0	1
● getListaProfesionales()		100%		n/a	0	1
● getUsuarioActivo()		100%		n/a	0	1
Total	90 of 234	62%	0 of 4	100%	2	20

Figura 7.2: Cobertura - Clase Sistema.

Mensaje











Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
● Mensaje()		100%		n/a	0	1	0	7	0	1
● toString()		100%		n/a	0	1	0	1	0	1
● setOrigen(Persona)		100%		n/a	0	1	0	2	0	1
● setDestino(Persona)		100%		n/a	0	1	0	2	0	1
● setAsunto(String)		100%		n/a	0	1	0	2	0	1
● setMensaje(String)		100%		n/a	0	1	0	2	0	1
● getOrigen()		100%		n/a	0	1	0	1	0	1
● getDestino()		100%		n/a	0	1	0	1	0	1
● getAsunto()		100%		n/a	0	1	0	1	0	1
● getMensaje()		100%		n/a	0	1	0	1	0	1
Total	0 of 63	100%	0 of 0	n/a	0	10	0	20	0	10

Figura 7.3: Cobertura - Clase Mensaje.

En la figura 6.3 se ve como el trabajo fue desarrollado y diseñado para cubrir todos los casos, esta clase su estado inicial de cobertura era de porcentaje era 0% por ende todo el porcentaje de las pruebas unitarias de las clases de dominio y iban a disminuir, y se tuvo que diseñar desde el principio para cubrir una gran parte de las ramas, pero finalmente se pudieron cubrir totalmente y que todas las pruebas estén correctamente entre el resultado esperado y obtenido.

```

package dominio;

import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;

public class MensajeTest {
    private static Mensaje mensaje;
    private static Usuario usuario;

    @Before
    public void setUp() throws Exception {
        usuario.setNombre("prueba");
        usuario.setNombreUsuario("pruebaTest");
    }

    @BeforeClass
    public static void setUpClass() {
        mensaje=new Mensaje();
        usuario=new Usuario();
    }

    @Test
    public void testEqualsAsuntoVacio() {
        System.out.println("EqualsAsuntoVacio()");
        mensaje.setAsunto("");
        boolean resultEsVacio = mensaje.getAsunto().isEmpty();
        assertTrue(resultEsVacio);
    }
}

```

Figura 7.4: Cobertura - Clase Mensaje.

```

@Test
public void testEqualsAsuntoNoVacio() {
    System.out.println("EqualsAsuntoNoVacio()");
    mensaje.setAsunto("test");
    boolean resultEsNoVacio = !mensaje.getAsunto().isEmpty();
    assertTrue(resultEsNoVacio);
}

@Test
public void testEqualsGetMensaje() {
    System.out.println("EqualsGetMensaje");
    String mensajeNuevo="Nuevo";
    mensaje.setMensaje(mensajeNuevo);
    String mensajeObtenido = mensaje.getMensaje();
    assertEquals(mensajeNuevo,mensajeObtenido);
}

@Test
public void testEqualsGetMensajeOk() {
    System.out.println("EqualsGetMensaje()");
    String mensajeNuevo="Nuevo";
    mensaje.setMensaje(mensajeNuevo);
    String mensajeObtenido = mensaje.getMensaje();
    boolean existeMensaje=!mensajeObtenido.isEmpty();
    assertTrue(existeMensaje);
}

```

Figura 7.5: Cobertura - Clase Mensaje.

7.1.1. Calidad de código

En esta sección, tras analizar el código final con las diversas herramientas de análisis estáticos, se pudo apreciar que en ciertas partes ya no existen los errores. Los errores mas comunes encontrados en el estado inicial.

- Unused Imports
- Variables no inicializadas
- Nomenclatura no adecuada.
- Métodos con Parámetros de mas de 7
- Excepciones Nulleables

- Excepciones Genéricos

Como se puede ver siguen existiendo en ciertas partes del código múltiples incumplimientos con el code convetions de Java, así como faltas de buenas practicas de programación descriptas en el libro de clean code. Falta de patrones de diseño, estos últimos ayudan a tener una mejor mantenibilidad del código y el ahorro de ciertos procesos del software. Pero si se pudo realizar una baja de métodos muertos, métodos largos bajando la complejidad cognitiva, mejorando los procedimientos try catch, de la serializacion y deserializacion del código, tanto en los métodos guardar sistema como cargar sistema. Quitar la responsabilidad de la clase sistema en ciertos métodos, aplicación de clases handlers, para gestión de login, y manejo de tipos de datos y validaciones.

Análisis de SonarQube

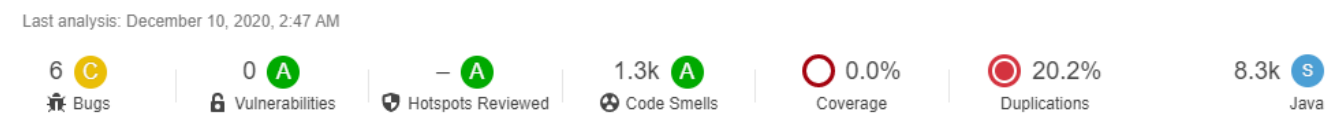


Figura 7.6: Análisis de SonarQube.

Como se puede apreciar sonarqube no dio la categoría C de bugs, se pudo bajar de 10 a 6 bugs críticos, quedaron pendientes la redefinicion del método equals de las clase persona, y alimento donde se tiene que poner el énfasis estrictamente en esos métodos, pueden generar errores a largo plazo. Por otra parte se bajo la duplicación de código del 30 % a 20 %

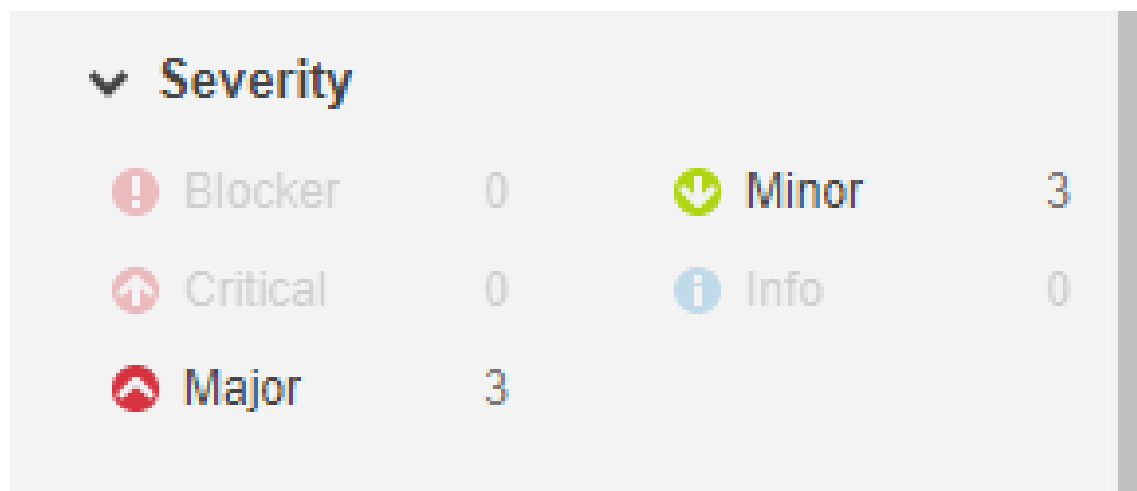


Figura 7.7: Severidad-Sonar Qube.

Análisis de EASY PMD

Se puso énfasis en los errores críticos que lanzaba, por una cuestión de tiempo, y además que estos errores eran muy críticos y llevaban un análisis mas exhaustivo,

como también el análisis y diseño de las ejecuciones de las pruebas JUnit.

	Description	File ▲	Location
Warning (93)			
✖	This anonymous inner class creation can be turned into a lambda ex...	InterfazL...	...ogin.java:113
✖	This anonymous inner class creation can be turned into a lambda ex...	InterfazL...	...ogin.java:134
✖	This anonymous inner class creation can be turned into a lambda ex...	PanelAgr...	...uario.java:60
✖	This anonymous inner class creation can be turned into a lambda ex...	PanelAgr...	...uario.java:72
✖	This anonymous inner class creation can be turned into a lambda ex...	PanelAgr...	...APlan.java:67
✖	This anonymous inner class creation can be turned into a lambda ex...	PanelAgr...	...APlan.java:86
✖	This anonymous inner class creation can be turned into a lambda ex...	PanelCo...	...sional.java:66
✖	This anonymous inner class creation can be turned into a lambda ex...	PanelCo...	...sional.java:82
✖	This anonymous inner class creation can be turned into a lambda ex...	PanelCo...	...sional.java:51
✖	This anonymous inner class creation can be turned into a lambda ex...	PanelCo...	...sional.java:86
✖	This anonymous inner class creation can be turned into a lambda ex...	PanelDie...	...uario.java:80
✖	This anonymous inner class creation can be turned into a lambda ex...	PanelDie...	...uario.java:94
✖	This anonymous inner class creation can be turned into a lambda ex...	PanelEdit...	...ional.java:122
✖	This anonymous inner class creation can be turned into a lambda ex...	PanelEdit...	...ional.java:145
✖	This anonymous inner class creation can be turned into a lambda ex...	PanelEdit...	...ional.java:225
✖	This anonymous inner class creation can be turned into a lambda ex...	PanelEdit...	...ional.java:239
✖	This anonymous inner class creation can be turned into a lambda ex...	PanelEdit...	...ional.java:253
✖	This anonymous inner class creation can be turned into a lambda ex...	PanelEdit...	...uario.java:143
✖	This anonymous inner class creation can be turned into a lambda ex...	PanelEdit...	...uario.java:176
✖	This anonymous inner class creation can be turned into a lambda ex...	PanelEdit...	...uario.java:213
✖	This anonymous inner class creation can be turned into a lambda ex...	PanelEdit...	...uario.java:250
✖	This anonymous inner class creation can be turned into a lambda ex...	PanelEdit...	...uario.java:266
✖	This anonymous inner class creation can be turned into a lambda ex...	PanelEdit...	...uario.java:277
✖	This anonymous inner class creation can be turned into a lambda ex...	PanelEdit...	...uario.java:288
✖	This anonymous inner class creation can be turned into a lambda ex...	PanelEdit...	...uario.java:299
✖	This anonymous inner class creation can be turned into a lambda ex...	PanelEdit...	...uario.java:315
✖	This anonymous inner class creation can be turned into a lambda ex...	PanelEdit...	...uario.java:326

Figura 7.8: Easy PMD- Analisis Final.

El analizador estático de Easy Pmd como se puede ver ya no nos muestra errores críticos como por ejemplo que nos mostraba el error en el estado inicial. “Overridable methodcalled during object constructor”. Esto significaba que los constructores no deben invocaban métodos reemplazables, directa o indirectamente. Se violaba esta regla, por ende el programa podría empezar a realizar fallas. El ejemplo que se trajo en la documentación de aseguramiento de calidad, era si la clase de sistema tendria una subclase, y esta extiende de sistema lo que pasaria es lo siguiente, cuando se instancian. El constructor de la superclase (Sistema) se ejecutara antes que el constructor de la subclase, por lo que el método de reemplazo en la subclase se

invocar a antes de que se ejecute el constructor de la superclase.

7.1.2. Interfaz gráfica de usuario

En esta sección se trata como se aplicaron las heurísticas de Nielsen para mejorar la experiencia de usuario tal y como se definió en las tareas. Además se les va asignar la métrica correspondiente sobre el estado final de usabilidad que definimos en el apartado de QC.

Interfaz de Inicio de Sesión

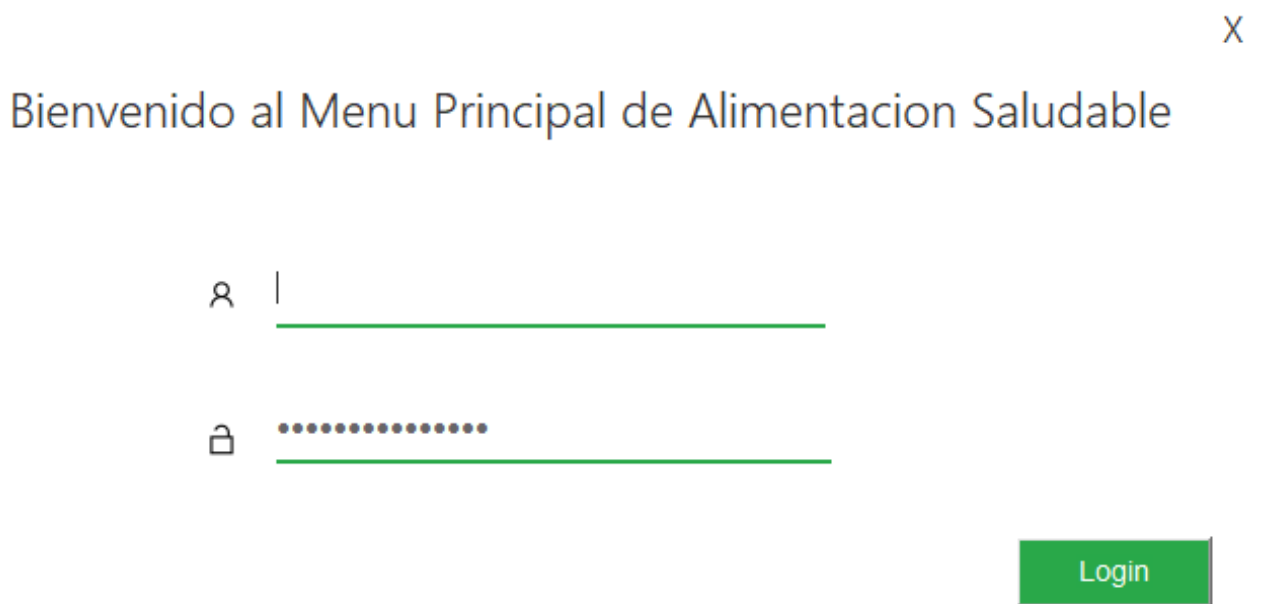


Figura 7.9: Interfaz de inicio de sesión.

Comparando y teniendo presente que la interfaz principal del estado del software inicial, era un panel con cambio de usuarios donde se evidenciaba el cambio de roles, aquí se intento mediante la solicitud de cambio planteada por nuestros clientes, realizar la interfaz de login, siguiendo las métricas de Nielsen, donde se siguiendo muchas heurísticas, visibilidad del estado del sistema, en caso de un error este informa al usuario que ingreso erróneamente los campos. Relación entre el mundo real y el sistema, esto queda evidenciado en el campo contraseña donde se puede visualizar que los puntos, son realmente los caracteres de la contraseña, se utiliza esto por seguridad en caso de que alguien lea, o visualice la pantalla. Tomando en cuenta que muchos sistemas utilizan este modo de visualizar implementar este estándar de visualización. El usuario va adquiriendo mas confiabilidad no solamente con la aplicación, sino también con el flujo del mismo. Ya no se plantea cosas, si esta bien o mal, simplemente las realiza de forma libre, y se comunica de forma “successfully” con el sistema.

Iconos y popUps Como dijimos cada acción u operación debería generar una respuesta perceptible por el usuario, desde el posicionamiento del cursor sobre un icono al envío de datos mediante formularios. Lo anterior también se aplica para cambios de estado como la liberación de un botón previamente bloqueado o el indicar que una opción ya ha sido seleccionada, por ejemplo, un hipervínculo. En general los tiempos de respuesta deben ser apropiados para cada tarea, si existen demoras mayores a 15 segundos en las respuestas del sistema el usuario debería informado del progreso en la concreción de la respuesta. Finalmente, un sistema cuenta con buena visibilidad si permite al usuario indicar verbalmente el estado y alternativas de acción en todo momento incluyendo la navegación, como se verá en la tercera heurística. Mencionado anteriormente lo que se implemento fueron los popUps y mejoras en los iconos del sistema, con esto el usuario sigue las convenciones del mundo real y otros sistemas, reconoce la acción que tiene que realizar y recibe el feedback adecuado para realizar sus actividades dentro del sistema.

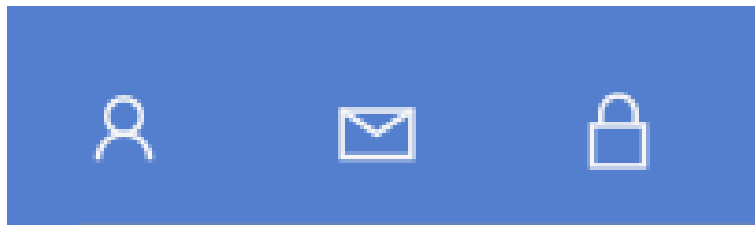


Figura 7.10: Iconos de los paneles.

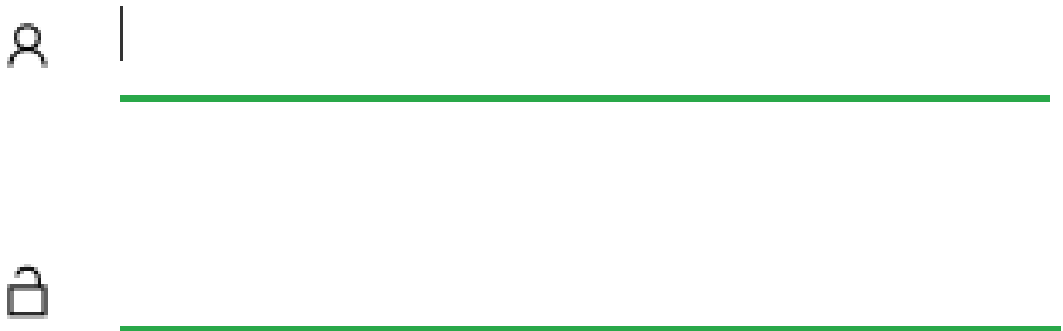


Figura 7.11: Iconos de los paneles.

Comparando y teniendo presente que la interfaz principal del estado del software inicial, era un panel con cambio de usuarios donde se evidenciaba el cambio de roles, aquí se intento mediante la solicitud de cambio planteada por nuestros clientes, realizar la interfaz de login, siguiendo las métricas de Nielsen, donde se siguiendo muchas heurísticas, visibilidad del estado del sistema, en caso de un error este informa al usuario que ingreso erróneamente los campos. Relación entre el mundo real y

el sistema, esto queda evidenciado en el campo contraseña donde se puede visualizar que los puntos, son realmente los caracteres de la contraseña, se utiliza esto por seguridad en caso de que alguien lea, o visualizar la pantalla. Tomando en cuenta que muchos sistemas utilizan este modo de visualizar implementar este estándar de visualización. El usuario va adquiriendo mas confiabilidad no solamente con la aplicación, sino también con el flujo del mismo. Ya no se plantea cosas, si esta bien o mal, simplemente las realiza de forma libre, y se comunica de forma “successfully” con el sistema.

Agregar/Modificar/Dar de Baja Usuario

Se agregaron los botones de Gestión de Usuario y Gestión Profesional, con esto previamente el sistema queda vulnerable en caso de algún error, en los registros, así como en los campos de usuario y de profesional, no existía un control por ende tenía muy poca seguridad en cuanto a la información, que pasa si un usuario se olvidaba la contraseña o por si algún motivo cambiaba la restricción de alimento que poseía, todo esto podría llevar a una degradación muy alta, un sistema poco robusto, y con poca seguridad.

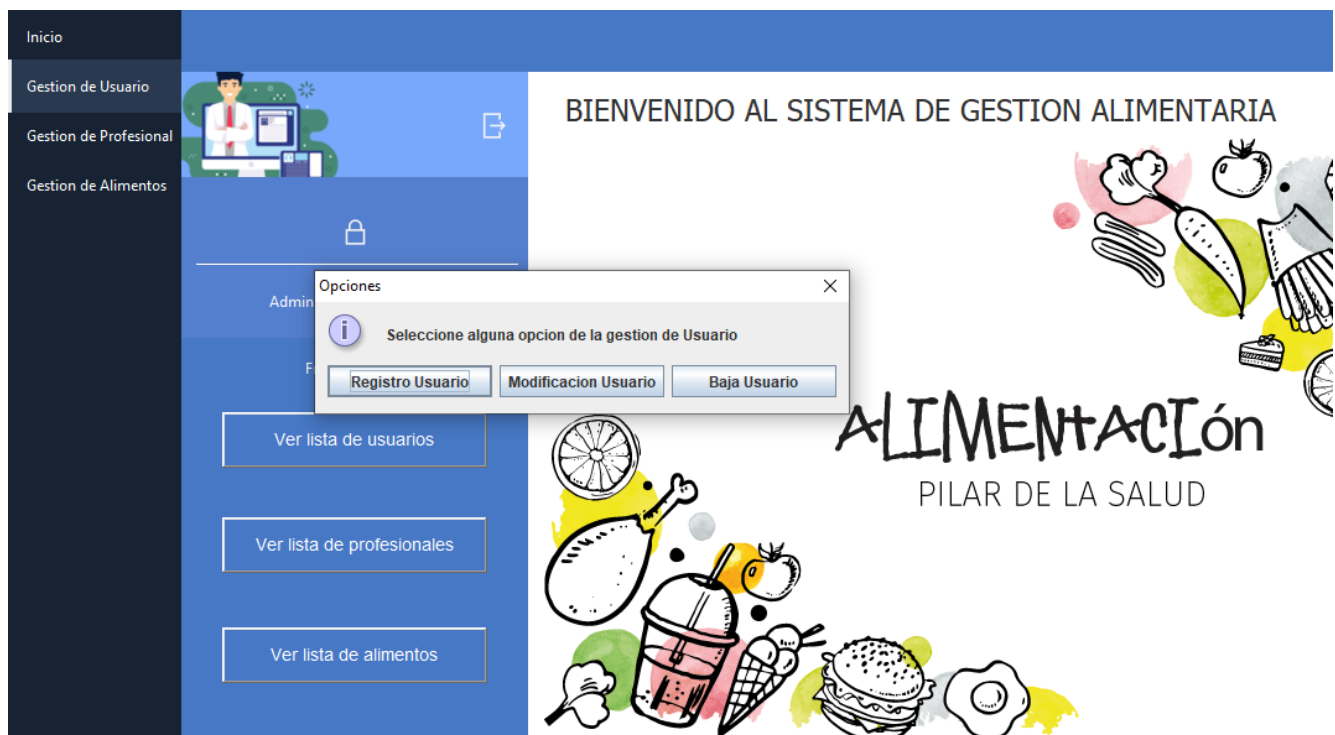


Figura 7.12: ABM de Usuario.

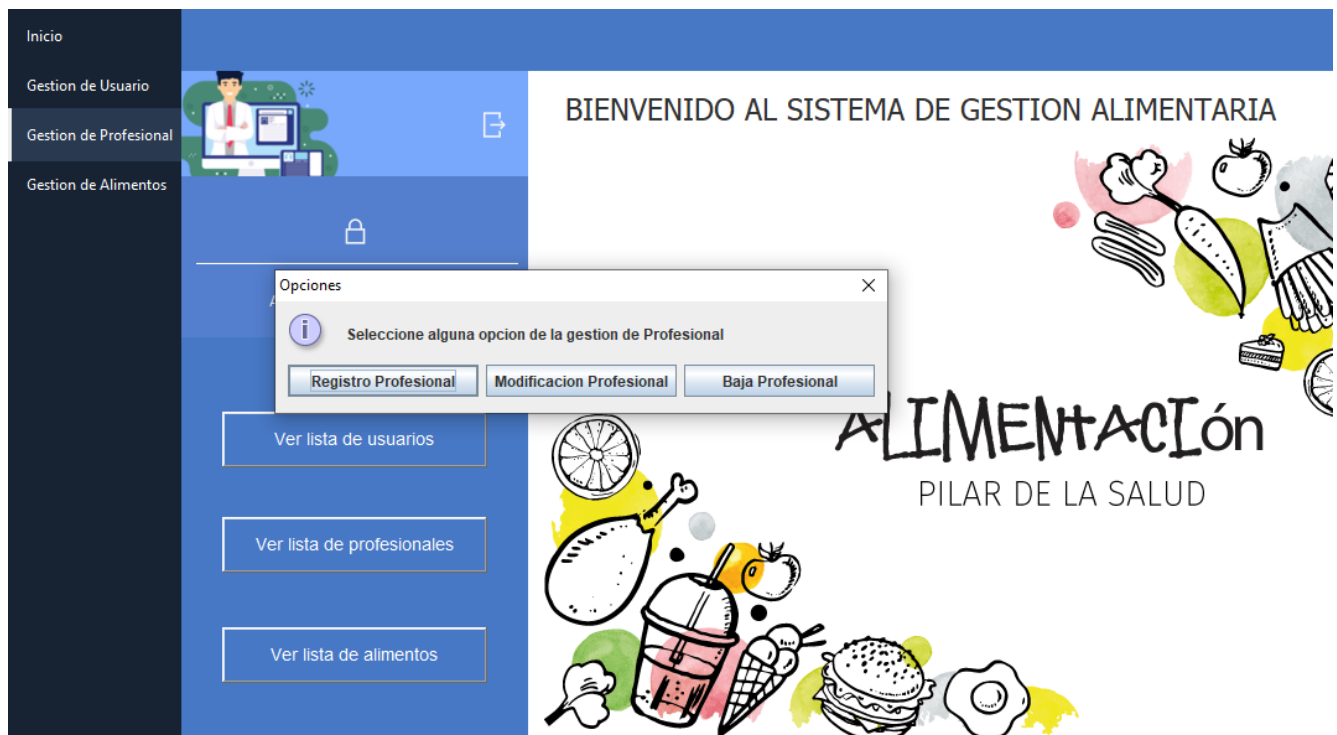


Figura 7.13: ABM de Profesional.

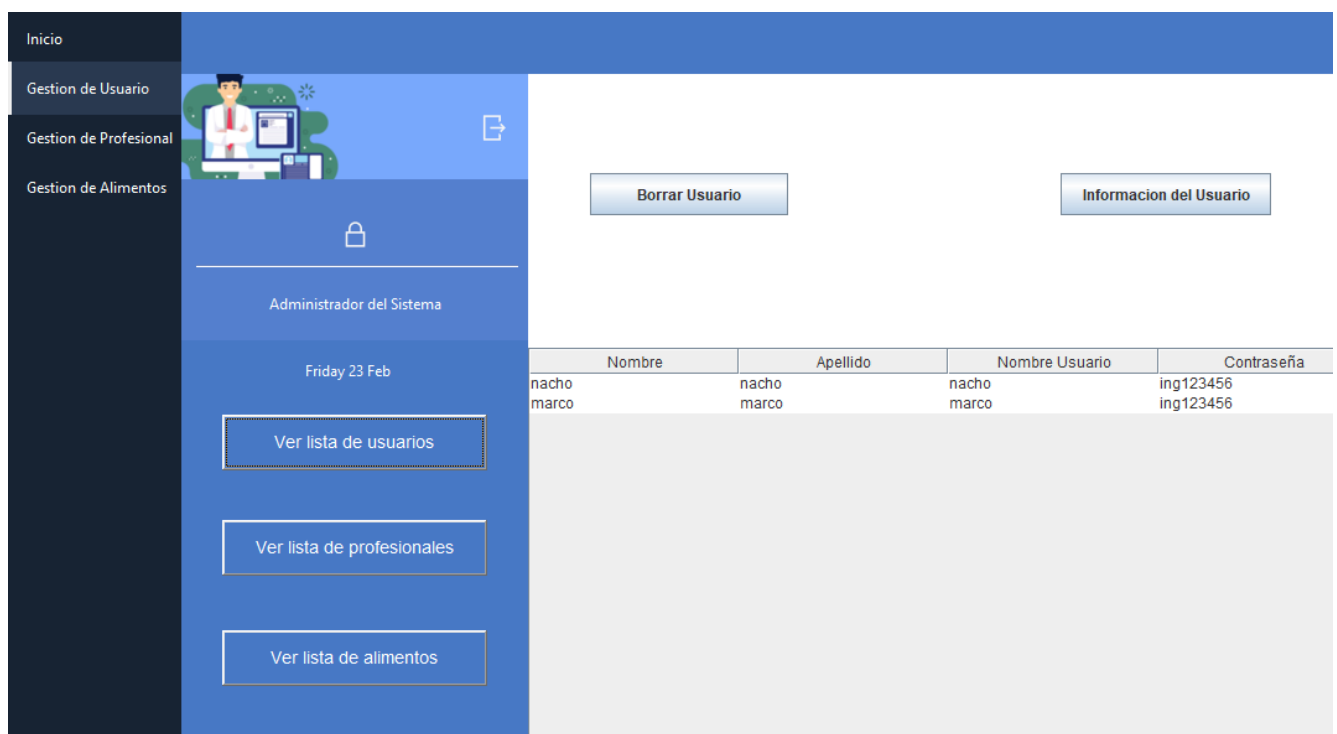


Figura 7.14: Baja Usuario.

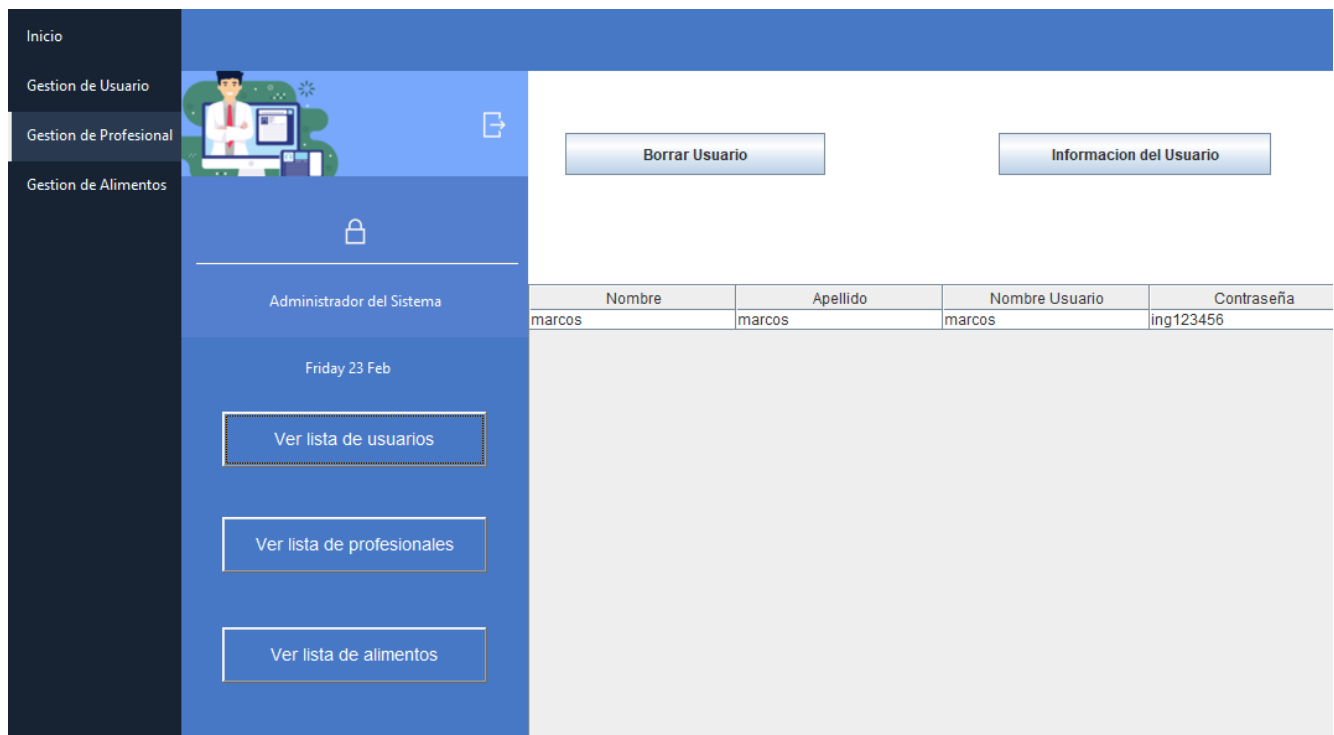


Figura 7.15: Baja Profesional.

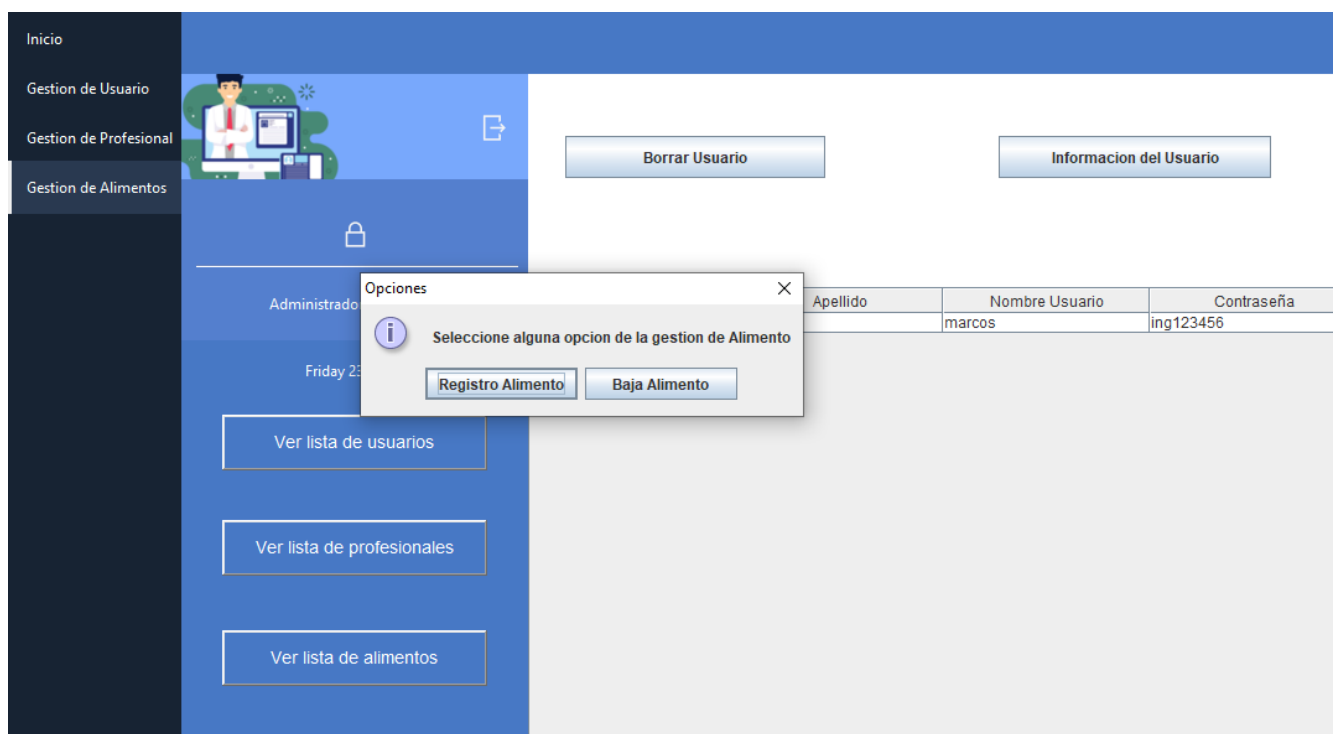


Figura 7.16: Alta y baja de Alimento.

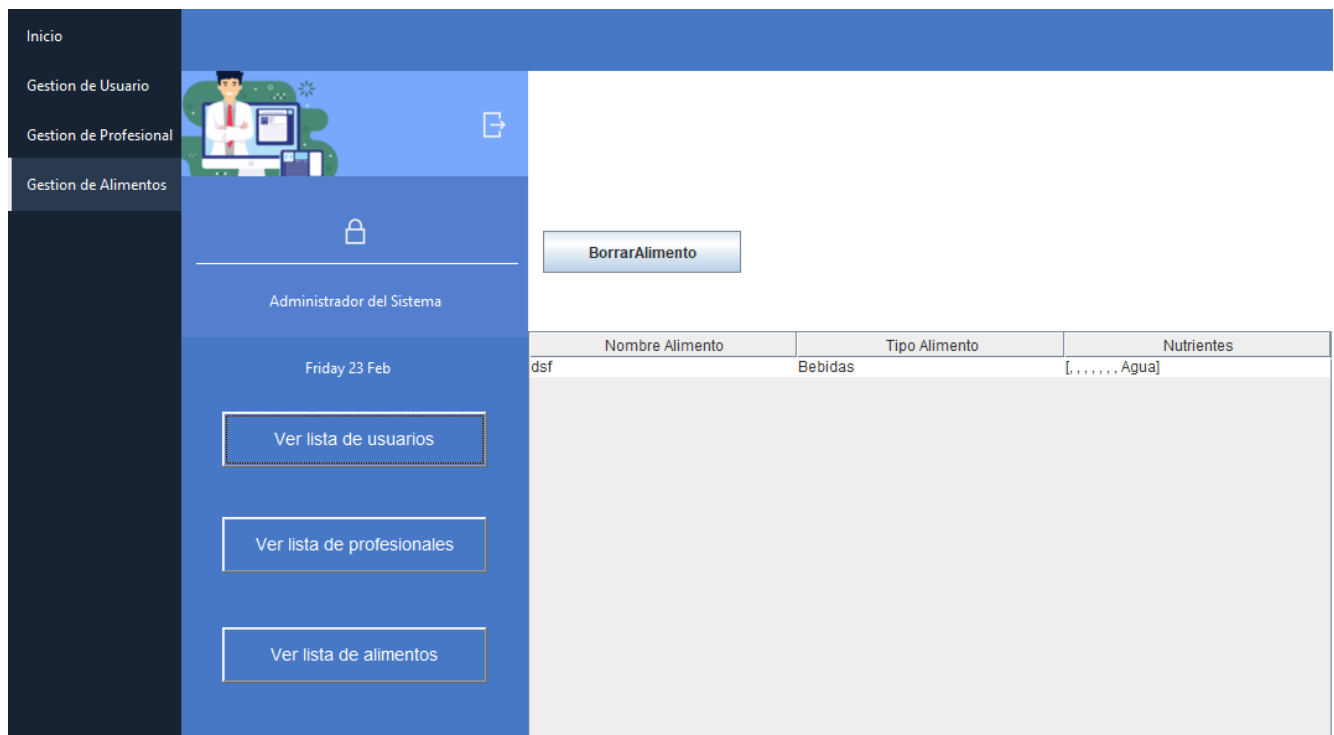


Figura 7.17: Baja Alimento.

Cada vez, el flujo de la aplicación va quedando claro, los roles se van diferenciando, donde para nosotros era importante realizar esta diferencia. El usuario administrador tiene que ser el encargado de solventar todos los problemas de la clase del dominio, si sucede algún error de gestión de los usuarios y del profesional, ahora con estas nuevas funcionalidades adquiere ese privilegio. Por ende la aplicación va adquiriendo confiabilidad, seguridad y robustez.

Retroalimentación del sistema

Cada vez que el usuario realiza una consulta, el usuario recibe un popUp de si se envió correctamente, o si le falta algún campo, en el estado inicial esto no estaba implementado, las funcionalidades de agregar alimento, usuario y profesional tenían estas implementaciones un poco erróneas. Estos conflictos no respetaban la heurística numero 1 de Nielsen, visibilidad del estado del sistema, el sistema debe darle feedback al usuario y que este ocurriendo lo correcto.

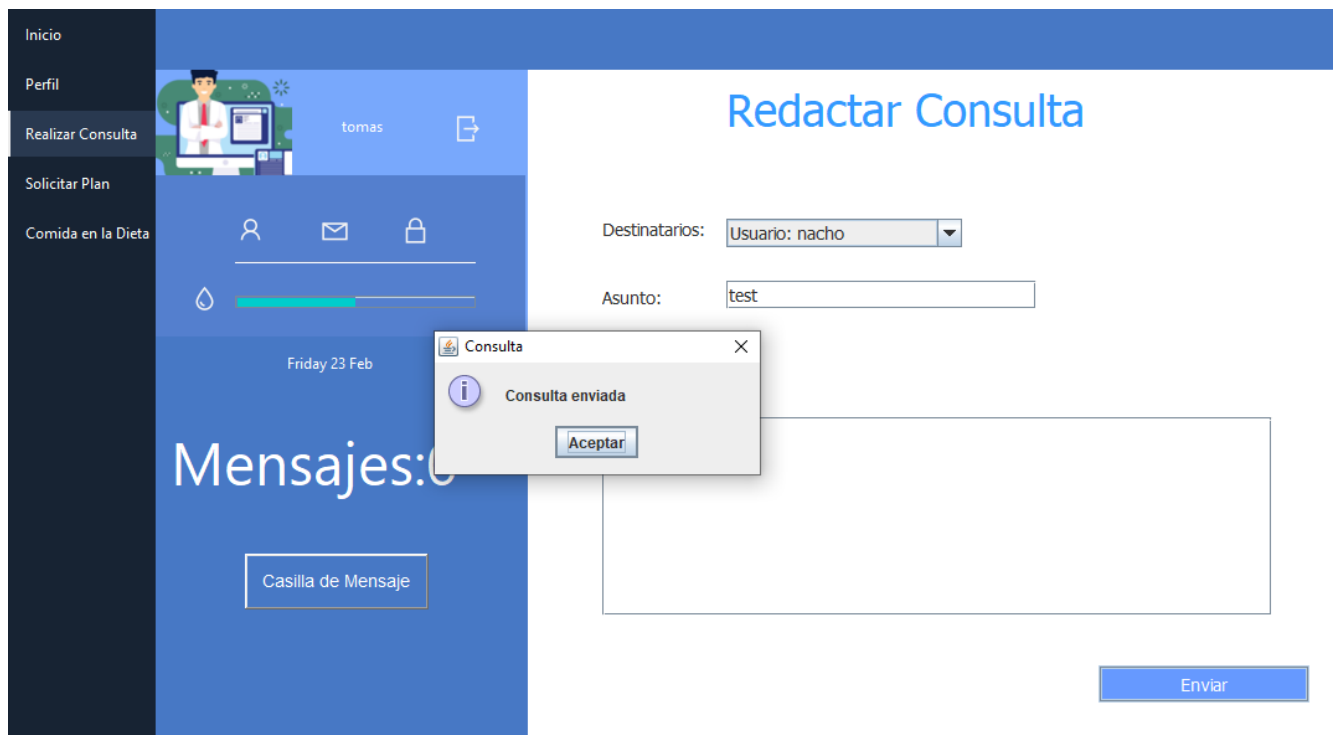


Figura 7.18: Consulta enviada OK.

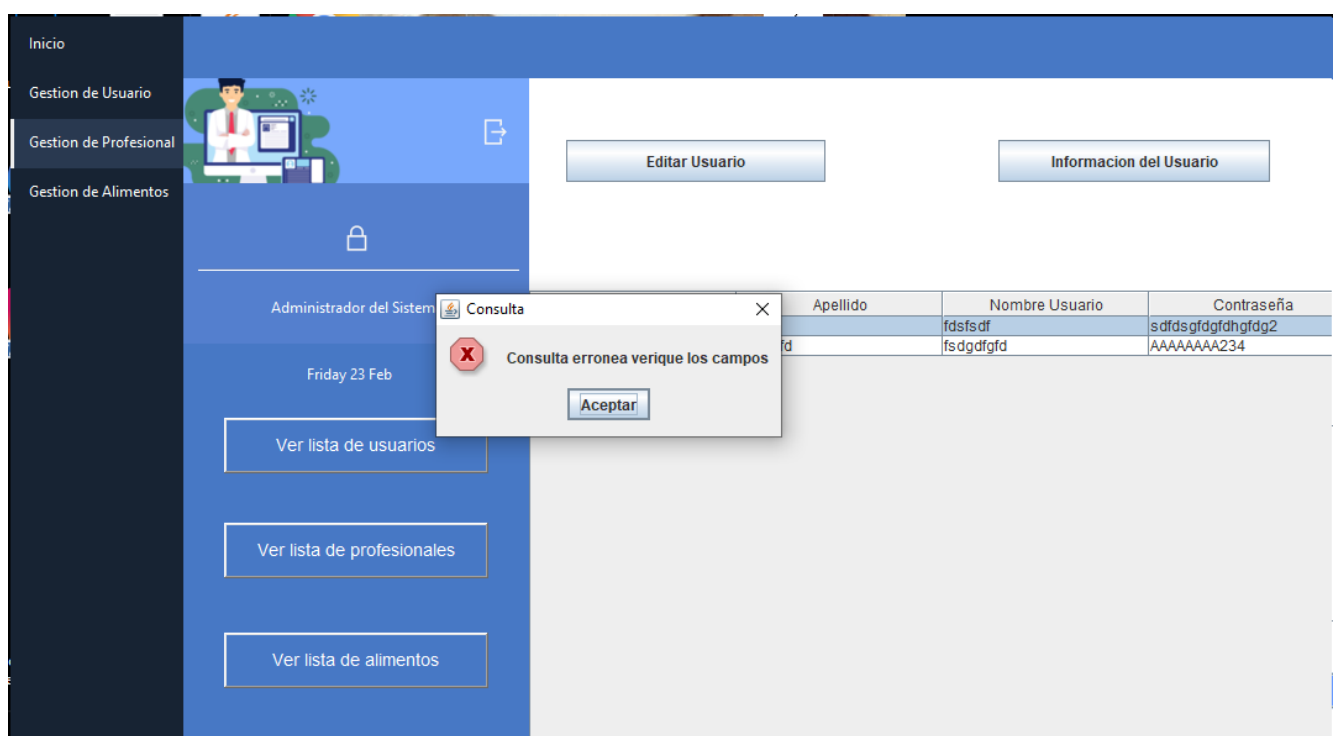


Figura 7.19: Consulta enviada ERROR.

Resultado de la métrica usabilidad:

- **Interactividad:** ¿Los mecanismos de interacción (por ejemplo, menús desplegables, botones, punteros) son fáciles de entender y usar? Con las implementaciones de iconos, listas desplegables como las listas de usuarios a eliminar o modificar, o también los alimentos, todas estas mejoras las calificamos como nivel óptimo para el sistema. Creemos que estas listas desplegables se pueden mejorar con alguna otra funcionalidad, dejamos pendiente este tema de estabilidad por falta de tiempo y presupuesto. Nivel: 75 %
- **Plantilla:** ¿Los mecanismos de navegación, contenido y funciones se colocan de forma que el usuario pueda encontrarlos rápidamente? Creemos que si, el usuario tiene todo una interfaz a su favor, el cambio de paneles se mantuvo, y se genero un mejor espaciado de contenido, tanto de botones, como etiquetas de mensaje e iconos interactivos. Nivel: 90 %
- **Estética:** ¿La plantilla, color, fuente y características relacionadas facilitan el uso? ¿Los usuarios “se sienten cómodos” con la apariencia y el sentimiento de la aplicación? Al no tener pruebas de regresión con usuarios se nos dificulto mucho este apartado de estética, nos hubiese gustado realizar un checklist a los usuarios o que escriban una pequeña reseña de que cambiarían de la estética con o si dejarían algún diseño. Nivel: 75 %
- **Características de despliegue:** ¿La aplicación usa de manera óptima el tamaño y la resolución de la pantalla? Creemos que si, el tamaño de la resolución de la aplicación con la pantalla es correcto y su centrado de todos los objetos y de la ventana fue el objetivo en este apartado. Nivel: 80 %
- **Personalización:** ¿La aplicación se adapta a las necesidades específicas de diferentes categorías de usuario o de usuarios individuales? Creemos que los diferentes roles se ven diferenciados y tienen sus características definidas, el saber que tarea tiene cada uno, que quiere saber el administrador, que le interesa al usuario y el profesional que le interesa de estos usuarios. Todo esto fue tomado en cuenta en este apartado. Nivel: 95 %

Resultado de la métrica general del sistema:

- Corrección. El software cumple con los objetivos principales del cliente, su forma de gestión de los alimentos, del historial, y de las comida ingeridas esta realizado, sin embargo el software presenta algunas fallas sobre esto, el profesional tiene que revisar que alimentos esta consumiendo el usuario, el usuario que tipo de profesional es el encargado de su plan de alimentación, todo esto son cosas que los stakeholders pueden quedar insatisfecho. Nivel de corrección: 60 %
- Confiabilidad. Este es un punto, en el que el programa cumple a raja tabla con las especificación, cumple un 100 % sus objetivos sin irse de las ramas. Nivel de confiabilidad: 90 %
- Eficiencia. Cantidad de recursos de cómputo y de código requeridos por un programa para llevar a cabo su función. Nivel de eficiencia: 90 %

- Integridad. Gracias a la implementación de la interfaz login, y el identificador de una contraseña, este apartado tiene un óptimo nivel. Nivel de Integridad: 89 %
- Usabilidad. En este item, como mencionamos anteriormente creemos que el usuario puede realizar el flujo normal, realizar las acciones correspondientes sin tener errores, sin que el precise ayuda en todo el momento sobre que esta haciendo el sistema, se le mantiene informado en cada momento, en cada acción. Nivel de usabilidad: 90 %
- Facilidad de recibir mantenimiento. En este apartado fue cuando se nos llamo, fue muy difícil las correcciones de todos los errores que presenta el sistema, las tecnologías avanzadas, la calidad de código no era la mas óptima y necesitaba verse para que no se degradara, la refactorización, el análisis mas exhaustivo del código, y la aplicación de patrones de diseño mas convencionales pueden llevar a que el sistema tenga una mejor facilidad de mantenimiento. Nivel de facilidad de recibir mantenimiento: 75 %
- Flexibilidad. Con los cambios que hicimos, las implementaciones de los handlers de validaciones y gestión de logueo, quitar la responsabilidad de muchas cosas de sistema, podemos decir que la flexibilidad aumento considerablemente. Nivel de flexibilidad: 86 %
- Portabilidad. La portabilidad es bastante alta, comparando con el sistema inicial se siguió manteniendo, se cambia el nombre del archivo de serialización por repositorio, para manejar mejor con los tipos de datos que se quieren guardar. Nivel de portabilidad: 86 %
- Reusabilidad. Este apartado fue muy importante para nosotros, se opto por cambiar un 50 % de todos los paneles pero reusando partes existentes de los que estaban, podemos decir que la reusabilidad es bastante elevada y comparada con el estado inicial mucho mas. Nivel de reusabilidad: 90 %
- Interoperabilidad. La capacidad de acoplar el sistema con otro es bastante baja, básicamente existe una franja muy pequeña, de pasar a un sistema a un sistema legacy, justamente este apartado esta en una franja critica, y creo que se merece un análisis exhaustivo, y poner el énfasis en este ítem. Nivel de interoperabilidad: 20 %

8. Reflexiones

Durante la realización de este obligatorio aprendimos como entender y analizar correctamente un proyecto externo, como corregirlo, y como medir el tiempo y el impacto de los cambios. Consideramos que este proyecto en particular es muy obsoleto, ya que la versión del JDK es muy antigua y difícil de actualizar, y la documentación estaba incompleta y errónea, por lo que debimos tratarlo como un sistema legacy, y trabajar con las condiciones en las que estaba sin buscar actualizarlo a una versión mas moderna. Las fortalezas del trabajo realizado fueron las mejoras realizadas a la interfaz, mejorando tanto en diseño como en calidad. También el aumento de la cobertura del código por medio de aumento y mejora de las pruebas unitarias ya existentes. En cambio, las debilidades del trabajo se reflejan en la inhabilidad del equipo de actualizar la versión del JDK del software, y el tiempo que tomo entender el comportamiento del mismo, resultando en demoras en el mantenimiento del mismo. También repercutió en el hecho de que tuvimos que adaptarnos a esa versión, y realizar los cambios planteados de forma diferente o no hacerlos directamente.

9. Anexo

9.1. Heurísticas de Nielsen

1. **Visibilidad del estado del sistema:** El sistema siempre debería mantener informados a los usuarios de lo que esta ocurriendo, a través de retroalimentación apropiada dentro de un tiempo razonable.
2. **Relación entre el sistema y el mundo real:** El sistema debería hablar el lenguaje de los usuarios mediante palabras, frases y conceptos que sean familiares al usuario, mas que con términos relacionados con el sistema. Seguir las convenciones del mundo real, haciendo que la información aparezca en un orden natural y lógico.
3. **Control y libertad del usuario:** Hay ocasiones en que los usuarios elegirán las funciones del sistema por error y necesitarían una “salida de emergencia” claramente marcada para dejar el estado no deseado al que accedieron, sin tener que pasar por una serie de pasos. Se deben apoyar las funciones de deshacer y rehacer.
4. **Consistencia y estándares:** Los usuarios no deberían cuestionarse si acciones, situaciones o palabras diferentes significan en realidad la misma cosa; siguen las convenciones establecidas.
5. **Prevención de errores:** Mucho mejor que un buen diseño de mensajes de error es realizar un diseño cuidadoso que prevenga la ocurrencia de problemas.
6. **Reconocimiento antes que recuerdo:** Se deben hacer visibles los objetos, acciones y opciones, El usuario no tendría que recordar la información que se le da en una parte del proceso, para seguir adelante. Las instrucciones para el uso del sistema deben estar a la vista o ser fácilmente recuperables cuando sea necesario.
7. **Flexibilidad y eficiencia de uso:** La presencia de aceleradores, que no son vistos por los usuarios novatos, puede ofrecer una interacción mas rápida a los usuarios expertos que la que el sistema puede proveer a los usuarios de todo tipo. Se debe permitir que los usuarios adapte el sistema para usos frecuentes.
8. **Estética y diseño minimalista:** Los diálogos no deben contener información que es irrelevante o poco usada. Cada unidad extra de información en un

dialogo, compite con las unidades de información relevante y disminuye su visibilidad relativa.

9. **Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de errores:** Los mensajes de error se deben entregar en un lenguaje claro y simple, indicando en forma precisa el problema y sugerir una solución constructiva al problema.
10. **Ayuda y documentación:** Incluso en los casos en que el sistema pueda ser usado sin documentación, podría ser necesario ofrecer ayuda y documentación. Dicha información debería ser fácil de buscar, estar enfocada en las tareas del usuario, con una lista concreta de pasos a desarrollar y no ser demasiado extensa.

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: <https://www.ort.edu.uy/fi/documentos/normas-especificas-para-la-presentacion-de-trabajos-finales-de-carrera.pdf>