

Universidad ORT Uruguay

Facultad de Ingeniería

Ingeniería de Software 2

Obligatorio 1

Tomás Dilema (209316)

Jorge Bruno Fernández (221961)

Juan Ignacio Olivera (223139)

Entregado como requisito de la materia Ingeniería de
Software 2

13 de octubre de 2020

[https://www.overleaf.com/read/mnjzrcmගdgcj](https://www.overleaf.com/read/mnjzrcmഗdgcj)

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 1, 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. Planificación y análisis de el estado de calidad inicial del software y productos asociados	3
1.1. Cobertura de las pruebas unitarias	4
1.1.1. Análisis de las pruebas unitarias	6
1.2. Calidad de los casos de prueba funcionales	9
1.2.1. Casos de prueba Registro de Profesional	9
1.2.2. Casos de prueba Registro de Usuario	10
1.3. Calidad de código y cumplimiento de estándares	11
1.3.1. Problemas detectados por SonarQube	11
1.3.2. Problemas detectados por EasyPMD	13
1.3.3. Problemas detectados por JavaHints	14
1.3.4. Problemas de usabilidad	14
1.3.5. Resultados obtenidos de la misión principal	21
2. Solicitudes de cambios para los cambios pedidos	23
2.1. Proceso de gestión de los cambios	23
2.2. Tabla con las solicitudes de cambio	24
2.3. Evidencia de uso de un repositorio	33
3. Análisis de impacto de los cambios a realizar	34
4. Plan de proyecto de mantenimiento	42
4.1. Identificación de interesados	42
4.1.1. Matriz poder/predecibilidad	43
4.2. Definición de objetivos del proyecto	43
4.3. Definición del alcance del producto y del proyecto	44
4.3.1. Definición del alcance del proyecto y lista de entregables	45
4.3.2. Diccionario de EDT	48
4.3.3. Alcance del producto	57
4.4. Estimación del esfuerzo	63
4.5. Especificación del ciclo de vida	65
4.6. Cronograma de trabajo	66
4.6.1. Relación con el modelo de ciclo de vida especificado	66
4.7. Estimación de costos	67
4.8. Plan de calidad	70
4.9. Plan de métricas	71

4.9.1. Desarrollo de conjunto de metas	71
4.9.2. Desarrollo de generación de preguntas, medición e implementación	72
4.10. Plan de RRHH	73
4.11. Plan de gestión de riesgos	75
4.11.1. Plan de contingencia para los riesgos mas prioritarios	76
Bibliografía	78

1. Planificación y análisis de el estado de calidad inicial del software y productos asociados

En el siguiente capítulo se muestra evidencia del análisis del estado de calidad inicial del software y productos asociados. Para este análisis exhaustivo a del estado de calidad tuvimos en cuenta los Factores de la calidad de McCall. A continuación, se presentan los errores que tenía el código y los respectivos cambios realizados por el equipo de mantenimiento.



Figura 1.1: Atributos de calidad 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.
- 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.

Es importante mencionar que al definir el alcance de este análisis se tomó en cuenta el código fuente. Este mismo criterio aplica para la cobertura de las pruebas unitarias que se vieron afectadas cuando no se ignoraba dicho código.

1.1. Cobertura de las pruebas unitarias

Realizado un análisis con la herramienta Java Code Coverage (JaCoCo) se puede observar que la cobertura de las pruebas unitarias se corresponde con la obtenida en el punto 2.2.2 de la documentación del proyecto.

dominio

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cqty	Missed	Lines	Missed	Methods	Missed	Classes
Usuario		70%		67%	16	41	18	96	12	35	0	1
Sistema		69%		80%	10	25	41	108	8	20	0	1
Mensaje		0%		n/a	10	10	20	20	10	10	1	1
PlanDeAlimentacion		42%		n/a	16	17	24	34	16	17	0	1
Alimento		81%		50%	7	16	15	39	6	15	0	1
ComidaPorDia		62%		50%	9	13	11	27	4	8	0	1
Profesional.Pais		97%		n/a	2	4	0	5	2	4	0	1
Alimento.TipoAlimento		94%		n/a	2	4	0	4	2	4	0	1
Alimento.Nutrientes		91%		n/a	2	4	0	3	2	4	0	1
Persona		89%		100%	3	15	3	32	3	12	0	1
Usuario.Nacionalidades		98%		n/a	1	4	0	6	1	4	0	1
Usuario.Restricciones		93%		n/a	1	4	0	2	1	4	0	1
Usuario.Preferencias		93%		n/a	1	4	0	2	1	4	0	1
Sistema.tipoUsuario		90%		n/a	1	4	0	2	1	4	0	1
Profesional		99%		50%	3	15	1	33	1	13	0	1
Total	513 of 2.691	81%	14 of 44	68%	84	180	133	413	70	158	1	15

Figura 1.2: Resultados JaCoCo clase Dominio

Usuario

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	CQty	Missed	Lines
• inicializoListaNacionalidades()	0%	n/a	1	1	2	2	2	
• setProfesionalAsignado(Profesional)	0%	n/a	1	1	2	2	2	
• setCasillaDeEntrada(ArrayList)	0%	n/a	1	1	2	2	2	
• setPreferenciasAlimentarias(Usuario_Preferencias)	0%	n/a	1	1	2	2	2	
• setRestricciones(Usuario_Restricciones)	0%	n/a	1	1	2	2	2	
• setPlan(PlanDeAlimentacion)	0%	n/a	1	1	2	2	2	
• getProfesionalAsignado()	0%	n/a	1	1	1	1	1	
• getCasillaDeEntrada()	0%	n/a	1	1	1	1	1	
• getAlimentacion()	0%	n/a	1	1	1	1	1	
• getPreferenciasAlimentarias()	0%	n/a	1	1	1	1	1	
• getRestricciones()	0%	n/a	1	1	1	1	1	
• getPlan()	0%	n/a	1	1	1	1	1	

Figura 1.3: Resultados JaCoCo clase de UsuarioDominio

Como se puede apreciar en la figura 1.3 la clase de dominio tiene pruebas sin realizar, esto impacta directamente al nivel de cobertura de las pruebas unitarias.

El resultado de una sesión de cobertura también es directamente visible en los editores de fuente de Java. El analizador de cobertura EclEmma tiene Un código de color personalizable que resalta las líneas total, parcialmente y no cubiertas. Esto funciona tanto para su propio código fuente como para fuentes adjuntas a bibliotecas externas instrumentadas.

El EclEmma se pudo apreciar que existen caminos no cubiertos, el marcado en rojo existe camino que no entro y no se evaluó ni siquiera y el marcado en amarillo porque evaluó la condición pero no fue cubierto totalmente.

```
//Metodo para validar que el dato sea numerico
public boolean pidoDatoNumerico(int dato, int min, int max) {
    int datoAVerificar = 0;
    boolean pidiendo = false;
    try {
        datoAVerificar = dato;
        if ((datoAVerificar >= min) && (datoAVerificar <= max)) {
            pidiendo = true;
        }
    } catch (NumberFormatException ex) {
    }
    return pidiendo;
}
```

Figura 1.4: Resultados EclEmma

Como se puedo apreciar en la figura 1.4 el metodo “public boolean pidoDatoNumerico(int dato,int min,int max)” no llego a evaluar totalmente todo los casos, existen caminos no cubiertos.

Luego se encontraron errores de codificación con el programa SonarQube que se analizaran a continuación.

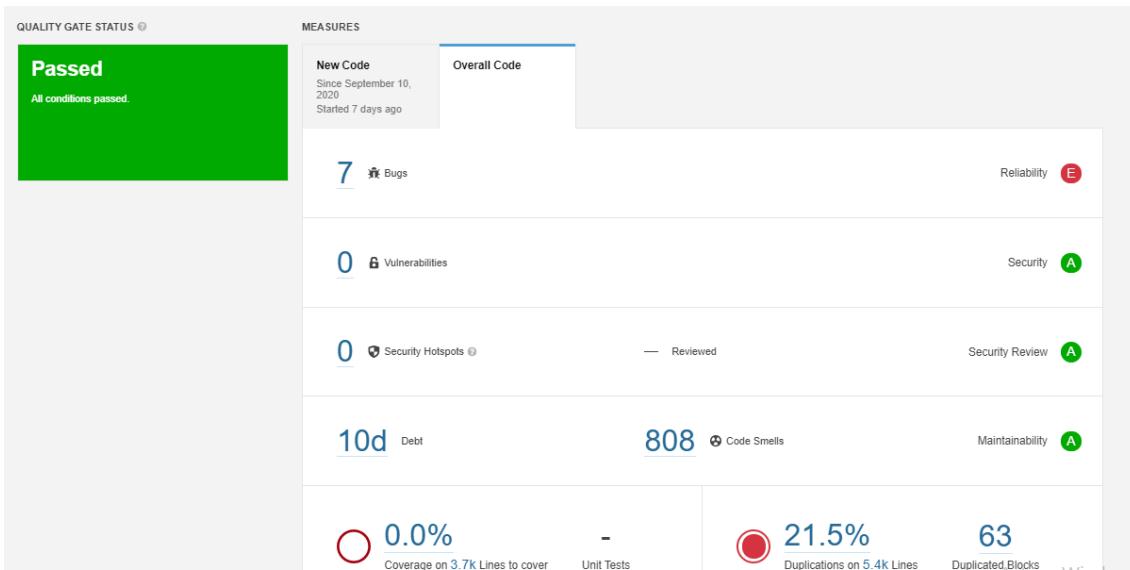


Figura 1.5: Resultados SonarQube

1.1.1. Análisis de las pruebas unitarias

Se realizo un análisis de código de las pruebas unitarias teniendo en cuenta el estándar ATRIP que refieren a que cada prueba sea:

- Automáticas
- Completas
- Repetibles
- Independientes
- Profesionales.

Este análisis se hizo para cada clase probadora del dominio. Como se puede observar en las prueba de JUnit, no todas cumplen con el conjunto automatizado de pruebas que constan de tres partes:

- 1. Una parte de configuración, en la cual se inicializa el sistema con el caso de prueba, esto es, las entradas y salidas esperadas.
- 2. Una parte de llamada (call), en la cual se llama al objeto o al método que se va a probar.
- 3. Una parte de declaración, en la cual se compara el resultado de la llamada con el resultado esperado. Si la información se evalúa como verdadera, la prueba tuvo éxito; pero si resulta falsa, entonces fracasó.

En las pruebas de JUnit existen discrepancias y omisiones al conjunto de pasos establecidos mencionados anteriormente, además del mal manejo de los *fixtures*. En

algunas clases se puede observar que no se inicializan los datos de entrada ni los de salida que posteriormente se utilizaran en los métodos de prueba utilizando el fixture `@Before`. Además estos fixtures son métodos que están vacíos esto sería uno de los ítem de la hediondez de la programación. Un ejemplo de esto se encuentra en la figura 1.6.

Todas las clases de test

```
@Before
public void setUp() {
}
```

Figura 1.6: Fixture Before vacío en clase test

Clase AlimentoTest.

```
@Test
public void testGetTipoCereal() {
    System.out.println("getTipo");
    Alimento instance = new Alimento();
    instance.setTipo(Alimento.TipoAlimento.Cereal);
    Alimento.TipoAlimento expResult = Alimento.TipoAlimento.Cereal;
    Alimento.TipoAlimento result = instance.getTipo();
    assertEquals(expResult, result);
}
```

Figura 1.7: Error println en clase Test

En la figura 1.7 se puede observar como existen salidas por consola (`println`) en las clases de prueba, esto es un error porque estamos consumiendo recurso (en este caso de la consola) donde no tiene nada de sentido y no esta pactado con la business logic y no interactúa directamente con el usuario por ser clases de prueba.

```

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 {

    public MensajeTest() {
    }

    @BeforeClass
    public static void setUpClass() {
    }

    @AfterClass
    public static void tearDownClass() {
    }

    @Before
    public void setUp() {
    }

    @After
    public void tearDown() {
    }
}

```

Figura 1.8: Error existencia de lazy class en clase Test

Existencia de una lazy class donde se ve que los métodos están vacíos, no existen atributos de clases, solo imports, los métodos fixtures vacíos y el constructor sin parámetro sin inicialización de ningún objeto o atributo de clase. Es un tipo de lazy class inline, esta es una clase no hace nada y no es responsable de nada, y que si tienen una planificabilidad de responsabilidad que puede volver operativa para ella pero en el momento no se ve reflejado en las prueba unitarias.

Clase PersonaTest.

```
public class PersonaImpl extends Persona {  
  
    public PersonaImpl() {  
        super("", "", "", "", null);  
    }  
}
```

Figura 1.9: Error de Manipulación de métodos Dominio

Existe un problema con la manipulación de dominio, una clase de prueba unitaria no puede modificar el esqueleto del dominio y ni agregar otra “lógica de negocio” en su clase de test.

1.2. Calidad de los casos de prueba funcionales

Para verificar la calidad de los casos de prueba, se realizo un estudio al análisis de caja negra realizado por los desarrolladores. Teniendo como parámetros por cada escenario:

- **OK:** Coincide con lo registrado.
- **Casi OK:** Coincide en el funcionamiento, pero el mensaje no.
- **No OK:** No coincide el funcionamiento.

1.2.1. Casos de prueba Registro de Profesional

Escenario	Nombre	Apellido	Nombre Usuario	Fecha Nacimiento	Nombre Titulo	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 vacio
3	Joaquin		JoacoP	20-01-1995	Nutricionista	25-06-2017	Uruguay	El apellido no puede ser vacio
4	Joaquin	Pascual		20-01-1995	Nutricionista	25-06-2017	Uruguay	El nombre de usuario no puede ser vacio
5	Joaquin	Pascual	JoacoP ya existente en el sistema	20-01-1995	Nutricionista	25-06-2017	Uruguay	Nombre de usuario no valido
6	Joaquin	Pascual	JoacoP		Nutricionista	25-06-2017	Uruguay	Fecha de nacimiento no valida
7	Joaquin	Pascual	JoacoP	20-01-1995		25-06-2017	Uruguay	El nombre de usuario no puede ser vacio
8	Joaquin	Pascual	JoacoP	20-01-1995	Nutricionista		Uruguay	Fecha de graduación no valida

Figura 1.10: Tabla de Casos de Prueba: Registro de Profesional

- Escenario 1 OK.
- Escenario 2 OK.
- Escenario 3 No OK: No muestra mensaje de error al no ingresar apellido.
- Escenario 4 casi OK: No muestra mensaje exacto, pero informa que ocurrió un error.
- Escenario 5 OK
- Escenario 6 OK
- Escenario 7 casi OK: No muestra mensaje exacto, pero informa que ocurrió un error.
- Escenario 8 OK

1.2.2. Casos de prueba Registro de Usuario

Escenario	Nombre	Apellido	Nombre Usuario	Nacionalidad	Fecha Nacimiento	Altura	Peso	Sexo	Restricciones	Preferencias	Resultado
1	Natalia	Elvas	ElvNati2	Uruguaya	20-03-1998	160	50	F	Ninguna	Ninguna	Usuario registrado correctamente
2		Elvas	ElvNati2	Uruguaya	20-03-1998	160	50	F	Ninguna	Ninguna	El nombre no puede ser vacío
3	Natalia		ElvNati2	Uruguaya	20-03-1998	160	50	F	Ninguna	Ninguna	El apellido no puede ser vacío
4	Natalia	Elvas		Uruguaya	20-03-1998	160	50	F	Ninguna	Ninguna	El nombre de usuario no puede ser vacío
5	Natalia	Elvas	JoaicolP	Uruguaya	20-03-1998	160	50	F	Ninguna	Ninguna	Nombre de usuario no válido
6	Natalia	Elvas	ElvNati2	Uruguaya		160	50	F	Ninguna	Ninguna	Fecha de nacimiento no válida
7	Natalia	Elvas	ElvNati2	Uruguaya	20-03-1998		50	F	Ninguna	Ninguna	La altura no puede ser vacía
8	Natalia	Elvas	ElvNati2	Uruguaya	20-03-1998	300	50	F	Ninguna	Ninguna	El dato debe de estar entre 0 y 265
8	Natalia	Elvas	ElvNati2	Uruguaya	20-03-1998	-1	50	F	Ninguna	Ninguna	El dato debe de estar entre 0 y 265
8	Natalia	Elvas	ElvNati2	Uruguaya	20-03-1998	a	50	F	Ninguna	Ninguna	Debe ingresar un dato numérico
9	Natalia	Elvas	ElvNati2	Uruguaya	20-03-1998	160		F	Ninguna	Ninguna	El peso no puede ser vacío
10	Natalia	Elvas	ElvNati2	Uruguaya	20-03-1998	160	600	F	Ninguna	Ninguna	El dato debe de estar entre 0 y 265
10	Natalia	Elvas	ElvNati2	Uruguaya	20-03-1998	160	-1	F	Ninguna	Ninguna	El dato debe de estar entre 0 y 265
10	Natalia	Elvas	ElvNati2	Uruguaya	20-03-1998	160	a	F	Ninguna	Ninguna	Debe ingresar un dato numérico

Figura 1.11: Tabla de Casos de Prueba: Registro de Usuario

- Escenario 1 OK.
- Escenario 2 No OK: no muestra mensaje y permite registrar usuario.
- Escenario 3 OK.
- Escenario 4 casi OK: No muestra mensaje exacto, pero informa que ocurrió un error.

- Escenario 5 OK.
- Escenario 6 casi OK: No muestra mensaje de error, pero informa que ocurrió un error.
- Escenario 7 casi OK: No muestra mensaje exacto, pero informa que ocurrió un error.
- Escenario 8.1 casi OK: No muestra mensaje correcto, pero informa que ocurrió un error.
- Escenario 8.2 casi OK: No muestra mensaje correcto, pero informa que ocurrió un error.
- Escenario 8.3 casi OK: No muestra mensaje correcto, pero informa que ocurrió un error.
- Escenario 9 casi OK: No muestra mensaje exacto, pero informa que ocurrió un error.
- Escenario 10.1 OK.
- Escenario 10.2 casi OK: No muestra mensaje correcto, pero informa que ocurrió un error.
- Escenario 10.3 casi OK: No muestra mensaje correcto, pero informa que ocurrió un error.

1.3. Calidad de código y cumplimiento de estándares

Tras analizar el código de la aplicación con diversas herramientas análisis estadístico, se pudo apreciar que en ciertas partes de la misma existían múltiples incumplimientos con el code conventions de Java, así como faltas de buenas prácticas de programador 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. Y por último ciertos problemas de optimización de usabilidad y no un buen manejo de las heurísticas de Jakob Nielsen. Entre otros problemas que se van a listar a continuación en esta sección:

1.3.1. Problemas detectados por SonarQube

Usando el SonarQube observamos problemas encontrados en el código, agrupados en ciertas categorías, las cuales se indicaran a continuación.

Tipos de problemas

Hay tres tipos de problemas:

- **Bug:** Un error de codificación que va a romper el código y debe ser reparado inmediatamente.
- **Vulnerability:** Un punto en el código que esta abierto a ataque.
- **Code Smell:** Un problema de mantenibilidad que hace al código confuso y difícil de mantener.

Severidad de los problemas

Cada problema tiene una de cinco severidades:

- **BLOCKER:** Bug con una alta probabilidad de impactar en el comportamiento de la aplicación en producción: pérdida de memoria, conexión JDBC no cerrada, etc. El código DEBE ser reparado inmediatamente.
- **CRITICAL:** Puede ser o un bug con una baja probabilidad de impactar en el comportamiento de la aplicación en producción o un problema que representa una falla de seguridad: un bloque catch vacío, inyección de SQL, etc. El código DEBE ser revisado inmediatamente.
- **MAJOR:** Falla de calidad que puede impactar fuertemente en la productividad del desarrollador: partes de código no cubiertas, bloques duplicados, parámetros sin uso, etc.
- **MINOR:** Falla de calidad que puede afectar levemente en la productividad del desarrollador: las líneas no deben ser demasiado largas, las sentencias “switch” deben tener al menos 3 casos, etc.
- **INFO:** Ni un error ni una falla de calidad, solo un descubrimiento.

The screenshot shows a code review interface with a red vertical bar on the left. The code snippet is as follows:

```
public void guardarSistema() {
    try {
        ObjectOutputStream out = new ObjectOutputStream
            (new FileOutputStream("sis.ser"));
        ...
        out.writeObject(listaAlimentos);
        out.writeObject(listaUsuarios);
        out.writeObject(listaProfesionales);
        out.flush();
        out.close();
    } catch (IOException ex) {
    }
}
```

A callout box highlights the line `ObjectOutputStream out = new ObjectOutputStream` with the text: "Use try-with-resources or close this 'ObjectOutputStream' in a 'finally' clause. Why is this an issue?". Below the code, there is a status bar with: "Bug ▾ ⚠ Blocker ▾ Open ▾ Not assigned ▾ 5min effort Comment" and "1 hour ago ▾ L129 🔍 No tags ▾".

Figura 1.12: Ejemplo de bug de tipo BLOCKER: Usar sentencia try-with-resources.

En la figura 1.11 se muestra un bug de tipo BLOCKER llamado “use try-with-resources or close this ‘ObjectOutputStream’ in a ‘finally’ clause”. La sentencia try-with-resources es una sentencia try que declara uno o más recursos. Un recurso es un objeto que debe ser cerrado después de que el programa termina de usarlo. La sentencia try-with-resources asegura que cada recurso fue cerrado al final de la sentencia. Cualquier objeto que implementa java.lang.AutoCloseable, que incluye todos los objetos que implementan java.io.Closeable, puede ser usado como recurso.

public enum TipoAlimento {
 Fruta, Cereal, Legumbre, CarnesBlancas, CarnesRojas, Vegetales,
}

Rename this constant name to match the regular expression '^A-Z][A-Z0-9]*(_[A-Z0-9]+)*\$'.
Why is this an issue?
Code Smell Critical Open Not assigned 2min effort
No tags

Figura 1.13: Ejemplo de Code Smell de tipo CRITICAL: Renombrar constante.

En la figura 1.12 se muestra un Code Smell de tipo CRITICAL llamado “Rename this constant to match the regular expression”. Esto ocurre ya que cuando una variable global almacena un valor constante el cual se usará en todo el proyecto, esa variable debe escribirse en mayúsculas y separando las palabras con “_” (barra baja).

45

1.3.2. Problemas detectados por EasyPMD

Description	File	Location
1 - Overridable method 'inicializaListaTiposDeUsuario' called during object construction	Sistema.java	...entacionSaludable/src/dominio/Sistema.java
1 - Overridable method 'inicializaListaTipoDeUsuario' called during object construction	Sistema.java	...entacionSaludable/src/dominio/Sistema.java
1 - Overridable method 'inicializaListaEnum' called during object construction	Usuario.java	...entacionSaludable/src/dominio/Usuario.java
1 - Overridable method 'inicializarLista' called during object construction	Usuario.java	...entacionSaludable/src/dominio/Usuario.java
1 - Overridable method 'actualizarLista' called during object construction	PanelCambioDeUsuario....ble/src/interfaz/PanelCambioDeUsuario.java	
1 - Overridable method 'actualizarLista' called during object construction	PanelCambioDeUsuario....ble/src/interfaz/PanelCambioDeUsuario.java	
1 - Overridable method 'actualizarLista' called during object construction	PanelCambioDeUsuario....ble/src/interfaz/PanelCambioDeUsuario.java	
1 - Overridable method 'actualizarLista' called during object construction	PanelCambioDeUsuario....ble/src/interfaz/PanelCambioDeUsuario.java	

Figura 1.14: Ejemplo de problema de EasyPMD: Overridable method called during object constructor.

En la figura 1.14 se muestra un problema bajo el nombre “Overridable method called during object constructor”. Esto significa que los constructores no deben invocar métodos reemplazables, directa o indirectamente. Si viola esta regla, el programa fallará. Ejemplo si la clase de sistema tendría una subclase está extiende de sistema lo que pasaría es lo siguiente cuando se instancian. El constructor de la superclase (Sistema) se ejecutará antes que el constructor de la subclase, por lo que el método de reemplazo en la subclase se invocará antes de que se ejecute el constructor de la subclase.

```
import java.io.Serializable;
import java.util.Objects;
import java.util.Optional;

public class Alimento implements Serializable {

    //Atributos
    private static final long serialVersionUID = 6106269076155338045L;
    private String nombre;
    private TipoAlimento tipo;
    private TipoAlimento[] listaEnumTipoAlimento = inicializaEnumTipoAlimento();
    private Nutrientes[] listaEnumNutrientes = inicializaEnumNutrientes();
    private boolean[] listaNutrientesSeleccionados;
```

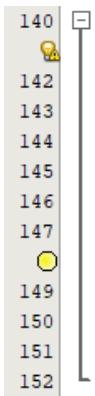
3 - Avoid variables with short names like p
3 - Avoid excessively long variable names like listaEnumTipoAlimento
3 - Avoid excessively long variable names like listaEnumNutrientes
3 - Avoid excessively long variable names like listaNutrientesSeleccionados
3 - Avoid excessively long variable names like listaEnumNutrientes
3 - Avoid excessively long variable names like listaEnumTipoAlimento
3 - Avoid excessively long variable names like listaNutrientesSeleccionados
3 - Ensure you override both equals() and hashCode()
3 - Ensure you override both equals() and hashCode()
3 - It is somewhat confusing to have a field name matching the declaring c
3 - Avoid variables with short names like p
3 - Abstract classes should be named AbstractXXX

Figura 1.15: Ejemplo de problema de EasyPMD: Avoid excessively long variable names.

En la figura 1.15 se muestra un problema bajo el nombre “avoid excessively long variable names”. Esto significa que los nombres de las variables no pueden ser muy

largos. Esto es debido a una regla que dice que el largo máximo de una variable es de 20 caracteres o números cercanos, ya que eso dificulta el entendimiento del código.

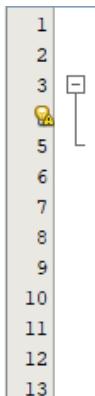
1.3.3. Problemas detectados por JavaHints



```
140     public boolean pidoDatoNumerico(int dato, int min, int max) {
141         int datoAVerificar = 0;
142         boolean pidiendo = false;
143         try {
144             datoAVerificar = dato;
145             if ((datoAVerificar >= min) && (datoAVerificar <= max)) {
146                 pidiendo = true;
147             }
148         } catch (NumberFormatException ex) {
149         }
150         return pidiendo;
151     }
152 }
```

Figura 1.16: Ejemplo de problema de JavaHints: The assigned value is never used.

Como se puede observar en la figura 1.16, indica el problema “The assigned value is never used” subrayando el 0 en amarillo. Este problema significa que el valor asignado a una variable se cambia dentro del código antes de ser usado o no se realiza ninguna operación con el mismo, volviéndolo redundante.



```
1 package dominio;
2
3 import java.io.Serializable;
4 import java.lang.reflect.Array;
5 import java.util.ArrayList;
6
7 public class PlanDeAlimentacion implements Serializable{
8
9     //Atributos
10    private static final long serialVersionUID = 6106269076155338045L;
11    private Usuario usuario;
12    private ArrayList<Alimento> listaLunes;
13    private ArrayList<Alimento> listaMartes;
```

Figura 1.17: Ejemplo de problema de JavaHints: Unused import.

Como se puede observar en la figura 1.17, se indica el problema “Unused import” subrayando el fragmento de código “import java.lang.reflect.Array;” en amarillo. Este problema significa que esa importación de librería no es usada en esa clase, lo cual puede dificultar el entendimiento de la misma.

1.3.4. Problemas de usabilidad

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 provocan una desmejora de experiencia por parte del usuario en cual puede a llegar realizar un manejo erróneo del sistema.

Interfaz principal

Al presionar el botón de inicio nos aparece la pantalla principal. Encontramos faltantes y mal uso de las siguientes heurísticas para la interfaz principal:

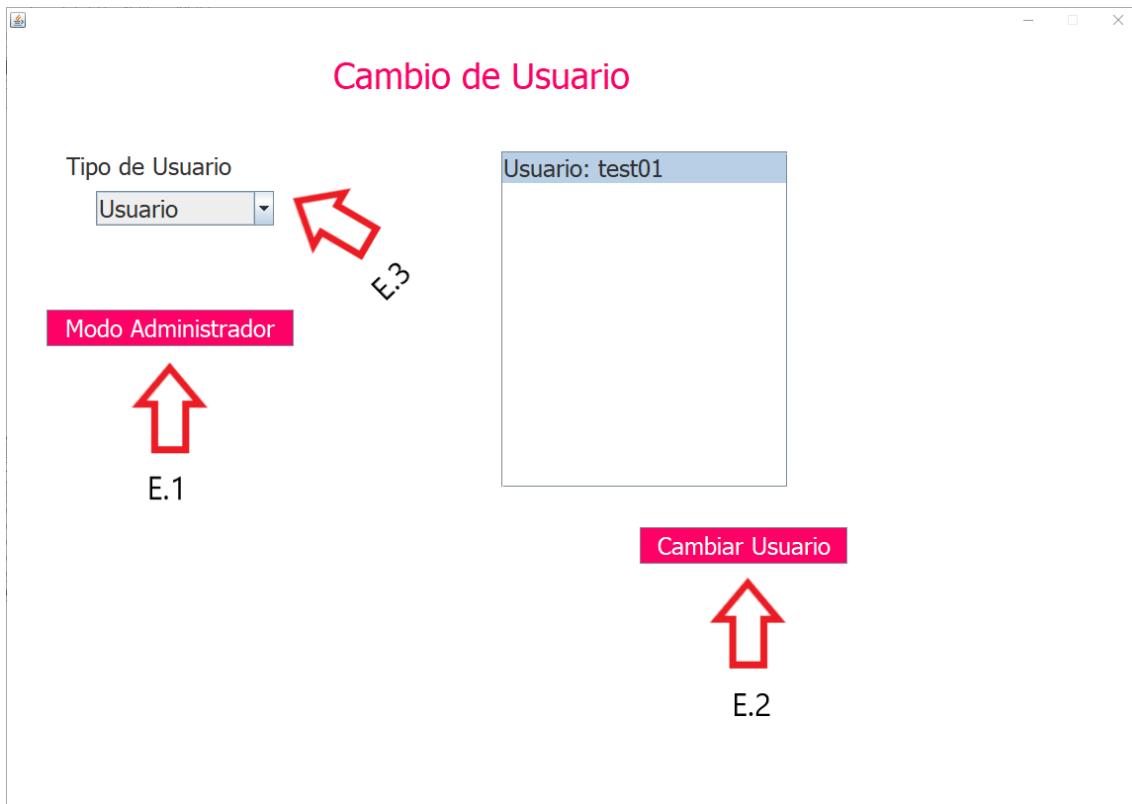


Figura 1.18: Interfaz principal

- **Diálogos estéticos y diseño minimalista:** La interfaz 1.18 posee un problema de características de despliegue, los elementos se encuentran dispersado por el tamaño de la interfaz mas bien el “empaquetado”, el mal manejo de los alignment de los objetos de textos, botones y etiquetas.
- **Relación entre el sistema y el mundo real:** Los usuarios no deberían cuestionarse si es usuario o profesional, debería haber un estándar que diga el rol que cumple el usuario que va utilizar el sistema, ejemplo el ComboBox (E.3) tiene mal definido los roles, acá claramente existe un error de nomenclatura de dominio. Además, el botón login esta con un nombre diferente al adecuado (E.2) para un sistema de este tipo, en verdad de cambiar usuario un botón que su texto dijera iniciar sesión.
- **Control y libertad de usuario:** A veces, el usuario se equivoca, es normal, en un caso que el usuario no quiera conectarse al sistema porque justo le

ocurrió un incidente no existe una posibilidad de salir, solamente mediante los iconos de salida que nuestro sistema operativo nos proporciona.

- **Flexibilidad y eficiencia de uso:** Se tiene que tener un sistema preparado para todo tipo de usuario ya previamente estipulado mediante los procesos de software, no existe diferenciación de ciertos roles que tiene los diferentes usuarios por lo tanto no cumple los requisitos de ser un sistema preparado para cualquiera que utilice este programa.
- **Prevención de errores:** La prevención de errores no se cumple, otra vez existe un error conceptual del dominio del sistema, el botón administrador(E.1) le da la posibilidad de acceder a ciertos privilegios a un usuario con privilegio menor. Esto ultimo no cumple con ciertos estándares de calidad de software, como la fiabilidad y la funcionalidad.

Interfaces Registro

Siguiendo el flujo de la aplicación al presionar click sobre el modo Administrador vemos un panel del lado izquierdo de la interfaz donde existe los diferentes registros: Usuario, Profesional, Alimento, Cambiar Usuario.

A continuación se presentan las siguientes heurísticas que no se aplicaron:

E.1 E.2

Registro Usuario

Foto de Perfil:

Nombre: test02

Apellido: testeando El apellido no puede ser vacío

Nombre de Usuario: test02.

Nacionalidad: Uruguaya

Fecha de Nacimiento: 12/09/2020

Altura: 140 cm El dato debe estar entre:0y265

Peso: 80 kg El peso no puede estar vacío

Sexo: Masculino Femenino

Restricciones: Celíaco Intolerante a la lactosa Diabético
 Hipertensión

Preferencias: Vegano Vegetariano Macrobiótico Orgánico

Usuario registrado correctamente

Aceptar

Figura 1.19: Interfaz Registro Usuario



Figura 1.20: Interfaz Registro Profesional



Figura 1.21: Interfaz Registro de Alimento

- **Relación entre el sistema y el mundo real:** El sistema tiene que “hablar” el lenguaje del usuario con palabras o frases que a éste le sean familiares y que pueda reconocer con facilidad algunas etiquetas. El mal uso de esta heurística se encuentra evidenciado en la interfaz 1.14 de registro de usuario:

- **Registro usuario:** En la interfaz registro de usuario, el mal uso de las

nomenclaturas de las etiquetas puede hacer confundir al usuario, por eso se recomienda utilizar otro nombre para las etiquetas. Para esta interfaz se cambiaría el nombre restricciones (E1) por enfermedades y el de preferencias (E2), por tipo de alimentación.

- **Diseño estético y minimalista:** En la interfaces de registro, en los puntos (E1) estas siguen la metodología de no mostrar pop-ups, en cambio si en labels como se específico en el ESRE del proyecto, pero al no mostrar pop-ups se sobrecarga la interfaz de elementos visuales. La heurística de Nielsen numero 8 dice no recargar el diseño, buscar interfaces que carguen rápido, lo principal es comunicar, no decorar.

Interfaces de Redacción



Figura 1.22: Interfaz Redacción consulta hacia al Usuario

- **Control y libertad del usuario:** Al hacer el envío de la consulta, se envía la consulta pero no existe ningún mensaje de diálogo o pop-up en el cual avise al usuario si esta seguro en hacer el envío de ese mensaje. Tenemos que darle al usuario la posibilidad de subsanar el error.
- **Visibilidad del estado del sistema:** En la interfaz 1.19 componente E1, el usuario no recibe ninguna retroalimentación de que ha hecho el envío de la consulta correctamente.

Interfaces de Consulta

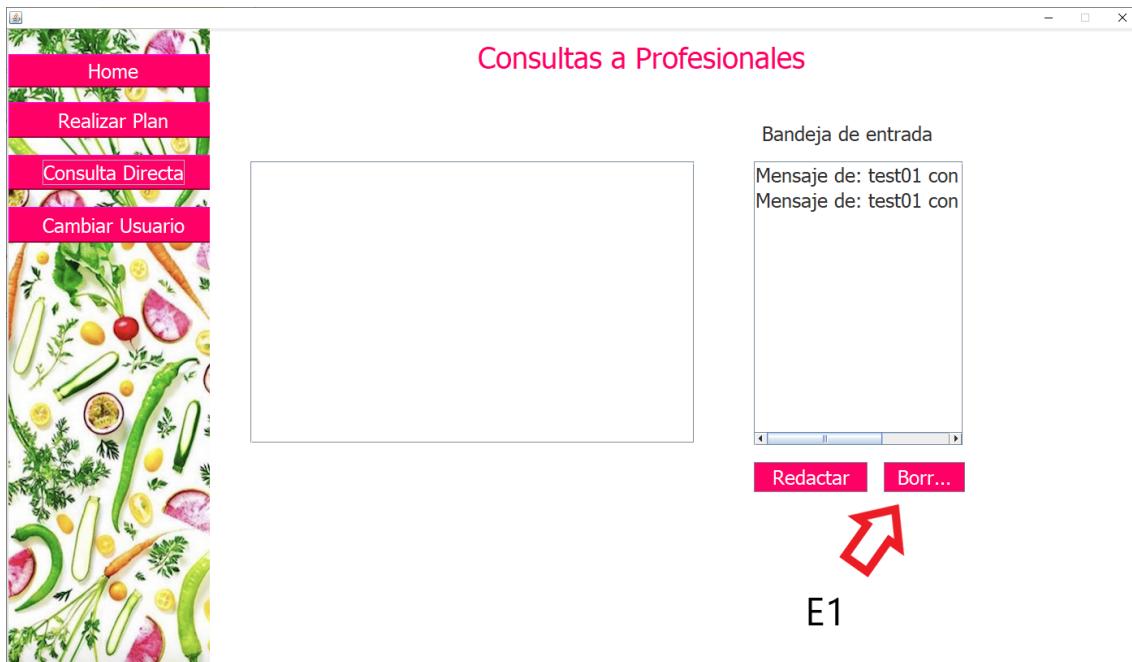


Figura 1.23: Interfaz de Casilla del Profesional

- **Relación entre el sistema y el mundo real:** En la figura 1.20 se ve claramente como la casilla del profesional tiene el botón borrar(E1) con el nombre no completo, siguiendo esta heurística las imágenes, botones, iconos, etc.. tienen que seguir un orden lógico sin la posibilidad de que el usuario se equivoque.

Interfaces de Agregar Comida

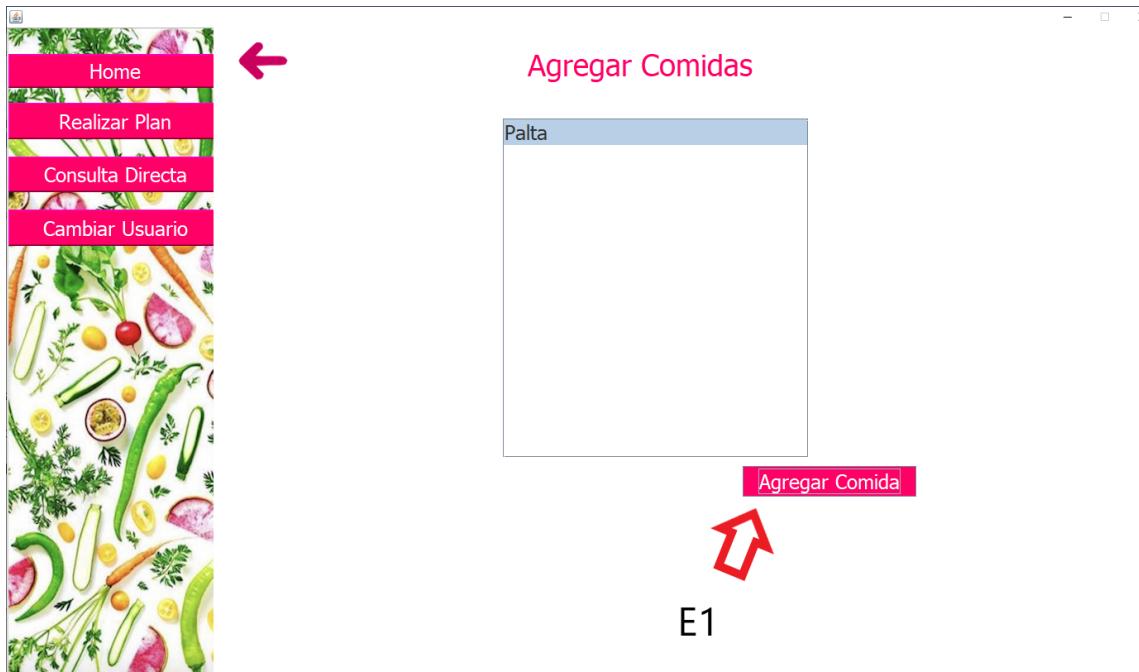


Figura 1.24: Interfaz Alta de comida



Figura 1.25: Interfaz Alta de comida a dieta

- **Visibilidad del estado del sistema:** En la interfaz 1.21 componente E1, el usuario no recibe ninguna retroalimentación de que ha hecho el alta de una comida.

- **Reconocer antes que recordar:** Como se puede observar al agregar un alimento no nos da la posibilidad de agregar cuantas veces se quiere agregar ese alimento, por ejemplo la existencia de un campo numérico que diga cuantas paltas quiero agregar, en verdad de ir agregando de a uno.
- **Control y libertad del usuario:** En la interfaz falta algún botón que permita quitar el alimento que no se quiere agregar, se necesita implementar alguna posibilidad de subsanar el error y no sentirse frustrado por no poder realizar algo.
- **Consistencia y estándares:** Otro punto que no se tiene en cuenta es seguir los convenios establecidos de usabilidad. Un ejemplo de esto es el objeto E1, con respecto al panel izquierdo no se ve una paleta de colores clara de uso de estándares, el rojo se debería usar para alguna acción de eliminar, dar de baja, o desetiquetar. Y un color por ejemplo verde para los botones de dar de alta, agregar o seleccionar.

1.3.5. Resultados obtenidos de la misión principal

Durante las sesiones, se identificaron un mal uso de las heurísticas de Nielsen esta ya fueron mencionadas anteriormente, a continuación una breve reseña de lo explorado:

- **Mal uso de estándares:** Un mal uso de estándares de usabilidad es que no existe un buen uso de paleta de colores para los elementos, por ejemplo verde para retroalimentar el sistema cuando algo esta válido.
- **Problema de ortografía:** Un problema de ortografía puede desencadenar problemas al sistema y al usuario, puede llevar a este ultimo a realizar acciones erróneas. El sistema tiene que tener conceptos que sean familiares al usuario.
- **Problema de visibilidad de los elementos:** El sistema presenta botones con un tamaño poco adecuado, no son visibles, realizar acciones que no deberían, además de otros elementos faltantes que pueden llevar a cabo un reconocimiento antes que un recuerdo.
- **Mensajes excesivos o escaso:** Los mensajes excesivos con etiquetas recargan el diseño de la interfaz haciéndolo poco estético, el uso de pop-ups puede lleva a cabo un buen uso.
- **Falta de ayuda y documentación:** Falta de documentación de como funciona un plan de alimentación, en que afectaría un ingreso de una comida a un plan. Creemos que el usuario puede verse perdido en el sistema y este necesita algunos ítems de que puede beneficiarse y/o perjudicarse realizando ciertas acciones.

Además subdividimos las pruebas en las siguientes categorías y en cada una se ingreso una métrica:

- **Interactividad:** ¿Los mecanismos de interacción (por ejemplo, menús desplegables, botones, punteros) son fáciles de entender y usar? Porcentaje resultante: 60/100
- **Plantilla:** ¿Los mecanismos de navegación, contenido y funciones se colocan de forma que el usuario pueda encontrarlos rápidamente? Porcentaje resultante: 20/100
- **Legibilidad:** ¿El texto está bien escrito y es comprensible? ¿Las representaciones gráficas se entienden con facilidad? Porcentaje resultante: 40/100
- **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? Porcentaje resultante: 15/100
- **Características de despliegue:** ¿La aplicación usa de manera óptima el tamaño y la resolución de la pantalla? Porcentaje resultante: 30/100
- **Sensibilidad temporal:** ¿Las características, funciones y contenido importantes pueden usarse o adquirir en forma oportuna? Porcentaje resultante: 40/100
- **Personalización:** ¿La aplicación se adapta a las necesidades específicas de diferentes categorías de usuario o de usuarios individuales? Porcentaje resultante: 30/100
- **Accesibilidad:** ¿La aplicación es accesible a personas que tienen discapacidades? Porcentaje resultante: 10/100

Realizamos dos sesiones de 30 minutos cada una. Los testers a cargo fueron:

- **Dilema, Tomas** Fecha de realización: 20-09-20
- **Fernandez, Bruno** Fecha de ealización: 19-09-20

Tenemos que conectar con el usuario.

2. Solicitudes de cambios para los cambios pedidos

2.1. Proceso de gestión de los cambios

Para el proceso de gestión de los cambios planificamos estos objetivos:

- Restringir los cambios a dichos elementos.
- Auditar los cambios a estos elementos.
- Gestionamos la configuración de estos elementos.

Los factores de éxito que se definió:

- El proyecto estará claramente definido.
- La Dirección cree firmemente en el cambio.
- Comunicamos claramente la visión y objetivos que se persiguen.
- Facilitamos el acceso de todos los implicados a la información.
- Desarrollamos e implantamos un excelente plan de comunicación eficaz.
- Implicamos a las personas. Conseguir la participación proactiva de los equipos.
- Logramos un compromiso de los implicados.
- Definimos claramente los objetivos de las personas y de la organización.
- Creamos un equipo de promoción y seguimiento dentro de la organización.
- Tenemos un plan de incentivos para su desarrollo. Si no es así, conseguimos que las personas se impliquen y encuentren atractivo el proyecto. Correcta gestión de las expectativas.
- Desarrollamos planes de formación como parte del plan de gestión del cambio.

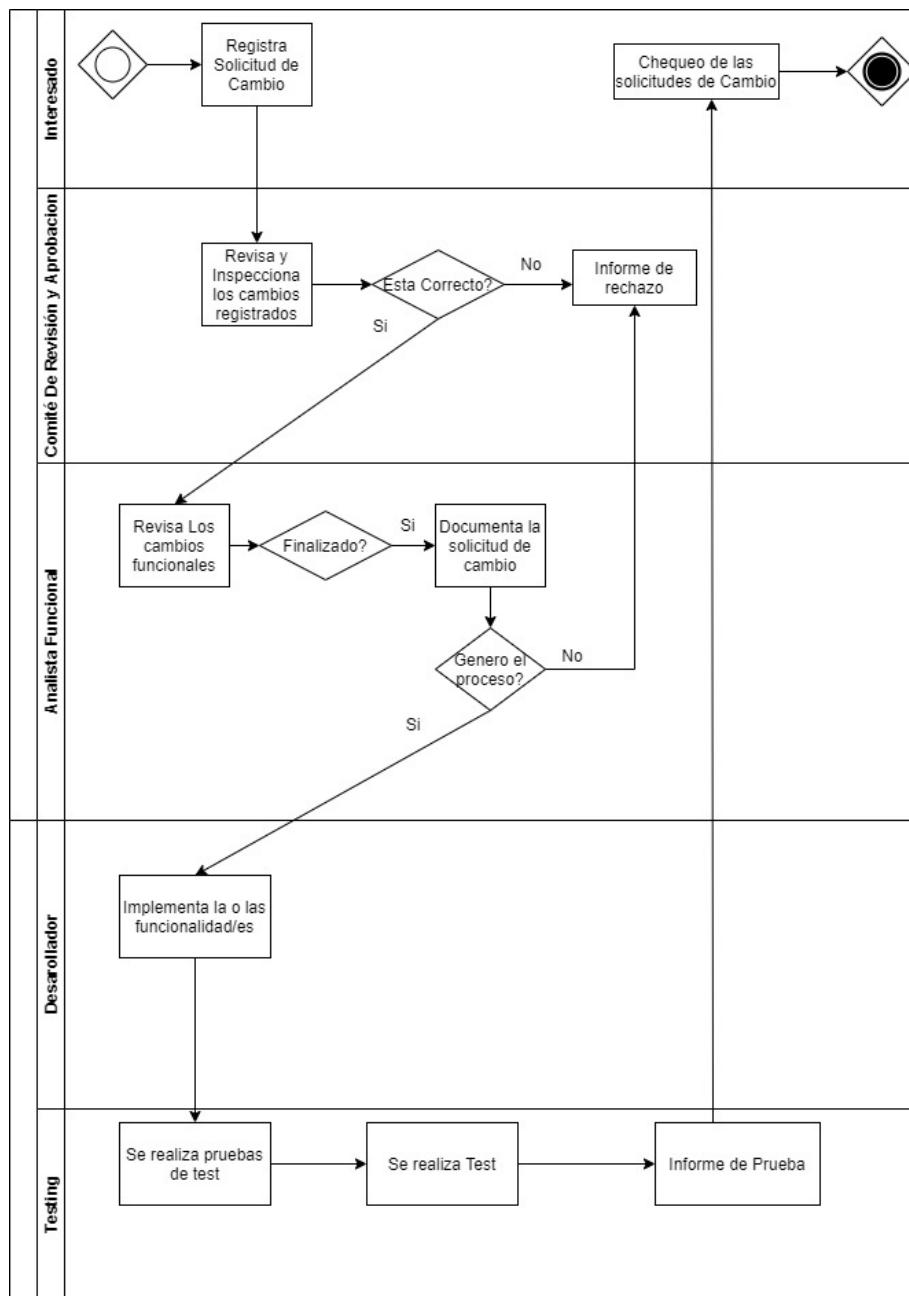


Figura 2.1: Diagrama SWIMLINE solicitud de cambio

2.2. Tabla con las solicitudes de cambio

ID	1
Cambio solicitado	Interfaz Login y Cerrar sesión.
Descripción del problema	La interfaz sin un login y un la funcionalidad para el sistema es un problema grande, uno de estos problemas es que este puede ver datos de otros usuarios. Ingresar en el modo administrador y el management de parte del dominio, además de sin la funcionalidad cerrar sesión. Y darle al usuario el control y la libertad.
Detalles del solicitante	Crear un login es importante para el sistema, un modo administrador y una interfaz sin cerrar sesión puede ser caótico para cualquier tipo de usuario.
¿Es una solicitud de cambio previamente solicitado?	NO
¿Cuándo fue solicitado este cambio?	18/9/2020
Razones y justificación	Es importante que la interfaz tenga un login y un cerrar sesión para que el sistema tenga un manejo óptimo, seguro y confiable.
Áreas afectadas	Interfaz y código.
Impacto de la no implementación del cambio propuesto	Puede haber un manejo erróneo de información, datos que ciertos interesados no quieren dar a conocer, un sistema que pueda ser inseguro y poca un mal equilibrio de la libertad del usuario.

ID	2
Cambio solicitado	Ayudas Visuales.
Descripción del problema	Las ayudas visuales actuales son muy precarias y se vuelve difícil la navegabilidad en la interfaz.
Detalles del solicitante	Ayudas visuales para saber en cuál de las opciones principales nos encontramos parados.
¿Es una solicitud de cambio previamente solicitado?	NO
¿Cuándo fue solicitado este cambio?	24/9/2020
Razones y justificación	Es importante que la interfaz tenga buenas ayudas visuales para que la misma sea más intuitiva y facilita la navegabilidad.
Áreas afectadas	Interfaz y código.
Impacto de la no implementación del cambio propuesto	Se torna difícil la navegabilidad a través de la interfaz, no se entiende bien en qué opción principal nos encontramos.

ID	3
Cambio solicitado	Eliminar Consulta Vacía.
Descripción del problema	Un usuario es capaz de enviar una consulta vacía a otro usuario.
Detalles del solicitante	Quitar en el sistema la posibilidad de enviar consultas vacías (sin texto) a otro usuario.
¿Es una solicitud de cambio previamente solicitado?	NO
¿Cuándo fue solicitado este cambio?	24/9/2020
Razones y justificación	Es importante que se quite la posibilidad de enviar consultas vacías, genera inconsistencia y no lo hace un sistema robusto.
Áreas afectadas	código.
Impacto de la no implementación del cambio propuesto	Le quita ciertas calidades del software al sistema, la robustez, fiabilidad y consistencia, el impacto de la no implementación puede ser un gran mal uso de memoria.

ID	4
Cambio solicitado	Mejorar los mensajes de retroalimentación.
Descripción del problema	Al realizar alguna funcionalidad, los usuarios que usan el sistema reciben poca información si cometieron algún error.
Detalles del solicitante	Mejorar el sistema de mensaje a los usuarios.
¿Es una solicitud de cambio previamente solicitado?	NO
¿Cuándo fue solicitado este cambio?	25/9/2020
Razones y justificación	Es importante seguir las heurísticas de Nielsen y la visibilidad del sistema nos genera un sistema robusto, sin esta mejora se vería afectada esta cualidad.
Áreas afectadas	Interfaz y código.
Impacto de la no implementación del cambio propuesto	Al no tener implementado estos mensajes de retroalimentación, el usuario queda perdido y no recibe una buena comunicación del sistema. Empeora la relación con el sistema y capaz no pueda realizar su tarea correctamente.

ID	5
Cambio solicitado	Mejorar la calidad de código y cumplimiento de estándares.
Descripción del problema	El equipo de SQA realizó un análisis exhaustivo, es imprescindible tener una mejora de esta apartado para un buen mantenimiento a futuro para este sistema.
Detalles del solicitante	Quita la posibilidad de que la aplicación sea mantenible.
¿Es una solicitud de cambio previamente solicitado?	NO
¿Cuándo fue solicitado este cambio?	25/9/2020
Razones y justificación	Es importante que el sistema pueda ser mantenible, dificulta la realización de cambio en el código como de alguna funcionalidad en el sistema.
Áreas afectadas	Código.
Impacto de la no implementación del cambio propuesto	Le quita ciertas calidades del software al sistema, la robustez, fiabilidad y consistencia, el impacto de la no implementación puede ser un gran mal uso de memoria.

ID	6
Cambio solicitado	Mejorar la cobertura de código.
Descripción del problema	El equipo de SQA realizó un análisis, mediante las herramientas de cobertura (JaCoCo) y un análisis del setup de pruebas y se encontró que no se llegó al punto deseado de cobertura.
Detalles del solicitante	Mejorar la cobertura del código para que tenga un mejor para que se llegue a una buena certificación de seguridad.
¿Es una solicitud de cambio previamente solicitado?	NO
¿Cuándo fue solicitado este cambio?	25/9/2020
Razones y justificación	Es importante que el sistema pueda llegar a casi un 100 % de cobertura para tener una certificación de seguridad.
Áreas afectadas	Código.
Impacto de la no implementación del cambio propuesto	Le quita ciertas calidades del software al sistema, el no seguir un estándar ATRIP, el no realizar todos los caminos posibles, puede llegar a generar posibles bugs a futuro.

ID	7
Cambio solicitado	Reporte de funcionalidades del sistema.
Descripción del problema	La documentación actual tiene una mala representación del sistema, no es una representación real de la aplicación, existen ítems faltantes, lo que se ejecuta no es lo que está documentado, pruebas incompletas, o erróneas.
Detalles del solicitante	Hacer reporte de funcionalidades del sistema: las que coinciden con las especificaciones, las que no coinciden, las que faltan, las que pudieron ser agregadas/modificadas/reparadas y las que no.
¿Es una solicitud de cambio previamente solicitado?	SI
¿Cuándo fue solicitado este cambio?	26/9/2020
Razones y justificación	Mejorar la documentación mejora notoriamente la calidad del proyecto, si no se tiene una buena especificación de funcionalidades, el sistema puede quedar perder trazabilidad, bajar la calidad y quedar obsoleto.
Áreas afectadas	Documentación.
Impacto de la no implementación del cambio propuesto	Disminuye notoriamente la calidad de la aplicación y del sistema como su operación pudiendo potencialmente disminuir la cantidad de usuarios que puedan usar la aplicación.

ID	8
Cambio solicitado	Implementar funcionalidad de listas Agregar Comida a la dieta.
Descripción del problema	El usuario puede agregar cualquier comida a la dieta esto es incorrecto, se llegó a un acuerdo que existan dos listas de alimentos ingeridos, una que la controle el profesional que recomendó para la dieta y el otro que pueda flexibilizar el usuario
Detalles del solicitante	Cambio en la funcionalidad de Agregar comida a la dieta.
¿Es una solicitud de cambio previamente solicitado?	NO
¿Cuándo fue solicitado este cambio?	20/10/2020
Razones y justificación	Es importante implementar esta funcionalidad para que el sistema siga una mejora en la lógica a la realidad planteada, los profesionales puedan llevar un buen seguimiento del paciente, y que el paciente lleve un mejor control de sus comidas sin sobrepasar a las recomendadas del paciente.
Áreas afectadas	Interfaz y código.
Impacto de la no implementación del cambio propuesto	Dificulta la división de responsabilidad en la aplicación, difícil seguimiento de los planes de alimentación.

ID	9
Cambio solicitado	Implementación de la autenticación mediante contraseña, la cual debe ser alfanumérica de más de 8 caracteres.
Descripción del problema	El sistema tiene que poseer una autenticación mediante contraseña, en la cual el largo tiene que ser más de 8 caracteres.
Detalles del solicitante	Se debe implementar la autenticación mediante contraseña, la cual debe ser alfanumérica de más de 8 caracteres.
¿Es una solicitud de cambio previamente solicitado?	SI
¿Cuándo fue solicitado este cambio?	13/10/2020
Razones y justificación	Tener una autenticación alfanumérica mejora la seguridad y confiabilidad del sistema y también es una seguridad para el propio usuario de sus propios datos.
Áreas afectadas	Código.
Impacto de la no implementación del cambio propuesto	Disminuye notoriamente la calidad de la aplicación a nivel de seguridad.

ID	10
Cambio solicitado	Informar sobre el usuario autenticado en la aplicación en cada ventana.
Descripción del problema	El sistema debe informar todo el tiempo que el usuario este autenticado en cada ventana.
Detalles del solicitante	Informar sobre el usuario autenticado en la aplicación en cada ventana.
¿Es una solicitud de cambio previamente solicitado?	SI
¿Cuándo fue solicitado este cambio?	13/10/2020
Razones y justificación	Al no Informar sobre el usuario autenticado en la aplicación en cada ventana, el usuario pierde el estado de visibilidad, puede verse confundido y afectar a su navegabilidad y posteriormente a la tarea que esta realizando.
Áreas afectadas	Código.
Impacto de la no implementación del cambio propuesto	Disminuye notoriamente la calidad de la aplicación a nivel de robustez, performance y consistencia.

ID	11
Cambio solicitado	Implementación/solución al menos 6 items de reporte de defectos especificados en la documentación.
Descripción del problema	El equipo de SQA junto a la solicitud de cambios de los autores originales del proyecto mencionaron que al menos 6 items de la documentación en la sección Reporte de defectos encontrados en cual su estado es: No arreglado tiene que estar implementados.
Detalles del solicitante	Implementar/solucionar al menos 6 items de la sección 2.5.2 de la documentación.
¿Es una solicitud de cambio previamente solicitado?	SI
¿Cuándo fue solicitado este cambio?	13/10/2020
Razones y justificación	Se llego a un acuerdo que cada reporte de defectos encontrado por las pruebas exploratorias por el equipo de SQA y los autores del proyecto estén implementas y arreglado.
Áreas afectadas	Código.
Impacto de la no implementación del cambio propuesto	Disminuye notoriamente a toda calidad de la aplicación.

ID	12
Cambio solicitado	Implementación: el alimento tenga un atributo de calorías,foto y descripción.
Descripción del problema	Se llega mediante los clientes y los autores del proyecto que alimento tenga un atributo llamado calorías, foto, descripción para llevar un conteo, y trazabilidad de cuantas calorías se lleva ingeridas, y una descripción sobre alguna ventajas que puede tener el consumir este alimento y foto para la trazabilidad.
Detalles del solicitante	Implementar atributos calorías, foto, descripción en la clase alimento.
¿Es una solicitud de cambio previamente solicitado?	NO
¿Cuándo fue solicitado este cambio?	22/10/2020
Razones y justificación	Se llegó a un acuerdo que un atributo calórico, foro y descripción en la clase alimento para llevar un conteo de cuantas calorías se lleva ingeridas.
Áreas afectadas	Código.
Impacto de la no implementación del cambio propuesto	El no tener un atributo calórica, foto y descripción le quita importancia a ciertas funcionalidades del plan, seguimiento de la dieta de un paciente.

ID	13
Cambio solicitado	Implementación refresh de listas.
Descripción del problema	Las listas no actualizan sus datos cuando son agregados.
Detalles del solicitante	Implementar el refresh de consultas en las listas de la aplicación, implementando un botón de refresh para realizar esta nueva funcionalidad.
¿Es una solicitud de cambio previamente solicitado?	NO
¿Cuándo fue solicitado este cambio?	22/10/2020
Razones y justificación	Sin un refresh (actualización) pierde consistencia la aplicación, además de ciertas heurísticas generando mala usabilidad para el usuario.
Áreas afectadas	Interfaz y Código.
Impacto de la no implementación del cambio propuesto	El no tener un refresh complica la navegabilidad de los usuarios.

ID	14
Cambio solicitado	Implementación Profesional Agregar Comida
Descripción del problema	Implementar que el profesional pueda agregar alimentos y notifique con una lista de solicitudes de alimento al administrador del sistema.
Detalles del solicitante	Implementar que el profesional pueda agregar alimentos y notifique al administrador del sistema.
¿Es una solicitud de cambio previamente solicitado?	NO
¿Cuándo fue solicitado este cambio?	22/10/2020
Razones y justificación	El profesional tenga la posibilidad de agregar un alimento al sistema para realizar un los planes de alimentación y seguimiento de los mismos.
Áreas afectadas Interfaz	Código.
Impacto de la no implementación del cambio propuesto	El profesional tenga la posibilidad de agregar un alimento al sistema para realizar un los planes de alimentación y seguimiento de los mismos sin esto le quita privilegios al profesional y además le impide realizar las tareas de planes y seguimiento correspondientes.

Además para cada tabla de solicitud de cambio se realizo el tipo de mantenimiento que se realizara. A continuación se detallara para cada id que tipo de mantenimiento se realizara:

- **Preventivo:** ID: 5, 6.
- **Adaptativo:** ID: 2.
- **Correctivo** ID: 7.
- **Perfectivo:** ID: 3, 8, 1, 9, 10, 11, 12, 13, 14.

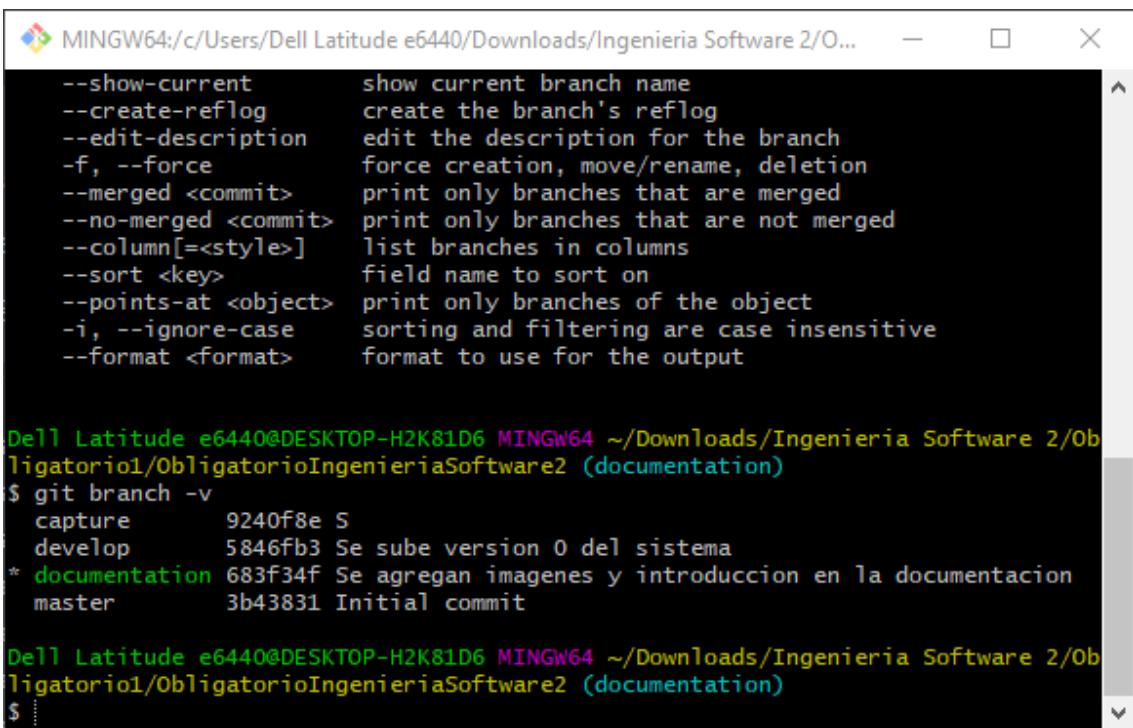
Definición del tipo de mantenimiento:

- **Perfectivo:** Para mejorar el rendimiento del producto.
- **Adaptativo:** Para mantener el software usable en un entorno operativo que ha cambiado.
- **Correctivo:** Para la reparación de fallas de desarrollo descubiertas en producción.
- **Preventivo:** Para mejorar la estructura de un programa, reducir su complejidad o hacerlo más fácil de entender.

2.3. Evidencia de uso de un repositorio

Dentro del equipo de SQA utilizamos un repositorio donde creamos 4 branches, la branch master, donde se guardaran las versiones estables del proyecto, la branch develop, donde se realiza todo el proceso de desarrollo del mantenimiento, la branch documentation, donde se guardará toda el plan de aseguramiento de calidad, análisis de calidad de la documentación, y análisis del código, y la branch image, donde se guardaran todas las imágenes relacionadas con el proyecto.

- **Branch master:** En esta branch subimos las versiones estables de la documentación ubicado en el archivo pdf. Esta branch fue de las últimas en ser actualizada, ya que todo el desarrollo de la mantención se guardó en otra branch y recién se subió cuando se tenía una versión estable.
- **Branch develop:** Aquí en esta rama se creo para la implementaciones de las funcionalidades provistas y las que no.
- **Branch documentation:** En esta branch se subió todo lo relacionado con el exhaustivo análisis de calidad, los planes de mantenimiento, el análisis del estado de la calidad y las solicitudes de cambio.
- **Branch image:** En esta branch se fue subieron todas las imágenes relacionadas al proyecto (EDT, tabla de solicitudes, especificación del ciclo de vida usado, screens de las interfaces, etc..), las cuales fueron vinculadas a este documento desde esta misma branch.



The screenshot shows a terminal window with the following content:

```
MINGW64:/c/Users/Dell Latitude e6440/Downloads/Ingenieria Software 2/O...
--show-current      show current branch name
--create-reflog    create the branch's reflog
--edit-description edit the description for the branch
-f, --force        force creation, move/ rename, deletion
--merged <commit> print only branches that are merged
--no-merged <commit> print only branches that are not merged
--column[=<style>] list branches in columns
--sort <key>       field name to sort on
--points-at <object> print only branches of the object
-i, --ignore-case  sorting and filtering are case insensitive
--format <format>  format to use for the output

Dell Latitude e6440@DESKTOP-H2K81D6 MINGW64 ~/Downloads/Ingenieria Software 2/O...
ligatorio1/ObligatorioIngenieriaSoftware2 (documentation)
$ git branch -v
capture      9240f8e S
develop      5846fb3 Se sube version 0 del sistema
* documentation 683f34f Se agregan imagenes y introduccion en la documentacion
master       3b43831 Initial commit

Dell Latitude e6440@DESKTOP-H2K81D6 MINGW64 ~/Downloads/Ingenieria Software 2/O...
ligatorio1/ObligatorioIngenieriaSoftware2 (documentation)
$
```

Figura 2.2: Evidencia de las ramas utilizadas en el obligatorio

3. Análisis de impacto de los cambios a realizar

El proceso de análisis de impacto: El proceso comenzó analizando las Solicitudes de Cambio (SC), el código fuente y la documentación asociada para identificar un conjunto inicial, llamado Starting Impact Set (SIS), de objetos de software que probablemente se vean afectados por el cambio requerido. Se identifico cada una de estas solicitudes de cambio para poder realizar la trazabilidad hacia adelante como hacia atrás durante todo el desarrollo de esta documentación.

Funcionalidades del producto de software:

- F1: Login.
- F2: Alta, baja y modificación de usuarios.
- F3: Alta, baja y modificación de alimentos.
- F4: Solicitud de alta, baja y modificación de alimentos por el profesional.
- F5: Pedir plan.
- F6: Enviar plan.
- F7: Modificar perfil usuario.
- F8: Modificar perfil profesional.
- F9: Redactar consulta.
- F10: Enviar consulta.
- F11: Refresh de listas.
- F12: Ver historial de comidas ingeridas.
- F13: Agregar comida ingerida diaria.
- F14: Ver bandeja de entrada.
- F15: Cerrar sesión.

- **ID: 1** La aplicación debería tener un login para que los distintos usuarios tengan su información guardada, segura y confiable, el sistema sea confiable, utilizar un Single Sign-On (Inicio de Sesión Único) que se utiliza para varias interfaces o utilizar un Simple Sign-in.

Ubicación del cambio: PanelPrincipal.java-void iniciarSesion()

Nivel de documentación	FUNCIONES DEL PRODUCTO SOFTWARE															Análisis de impacto vertical	ID 1
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15		
Especificación de requerimientos																	
Especificación de diseño																	
Casos de prueba																	
Código fuente	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
Análisis de impacto horizontal	IMPACTA TODO EL CODIGO DE LA INTERFAZ SIN INTERCEDER EN EL DOMINIO																

Figura 3.1: Tabla de análisis de impacto ID 1

- Duración: 0.8 días
- Recursos asignados: 2: developer, tester
- Esfuerzo estimado: 1.6 días x recursos
- **ID: 2** Para poder efectuar este cambio, el equipo de SQA tiene que llegar a un acuerdo con el equipo de diseño del proyecto sobre que seria lo adecuado, reparar, agregar, modificar en las distintas interfaces del programa. El ingreso de elementos interoperables, reusables y portables, como la implementación de JavaFX que cuenta con elementos para el diseño de interfaz gráfica GUI modernos y dinámicos como animaciones, vistas web, contenido multimedia y lo más importante estilos basados en hojas de estilos CSS, para el diseño de estas interfaces gráficas. Además cuenta con herramientas como el uso de FXML que es un lenguaje de tipo declarativo basado en XML, contando también con un editor gráfico de interfaz llamado SceneBuilder el cual permite construir mediante el sistema de arrastrar y soltar los elementos en el FXML. Entre otras mejoras que se le realiza a nivel de estética y usabilidad a las interfaces ubicadas en el package interfaz.

Ubicación del cambio: Todo el package interfaz

Nivel de documentación	FUNCIONES DEL PRODUCTO SOFTWARE															Análisis de impacto vertical	ID 2
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15		
Especificación de requerimientos																	
Especificación de diseño																	
Casos de prueba	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
Código fuente	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
Análisis de impacto horizontal	IMPACTA TODO EL CODIGO DE LA INTERFAZ SIN INTERCEDER EN EL DOMINIO																

Figura 3.2: Tabla de análisis de impacto ID 2

- Duración: 3 días

- Recursos asignados: 3: developer, diseñador y tester
- Esfuerzo estimado: 9 días x recursos
- **ID: 3** Cualquier usuario puede enviar una consulta sin tener un mensaje escrito. Quitar este “bug” impacta directamente a una mejora de la robustez del sistema. Esto va a impactar a los casos de prueba, se tienen que realizar nuevos casos de prueba para este arreglo.

Ubicación del cambio: clase Mensaje- void setMensaje(String mensaje)

	FUNCIONES DEL PRODUCTO SOFTWARE															Analisis de impacto vertical	ID 3
Nivel de documentación	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15		
Especificación de requerimientos																	
Especificación de diseño																	
Casos de prueba										X	X						
Código fuente										X	X						
Analisis de impacto horizontal																	

Figura 3.3: Tabla de análisis de impacto ID 3

- Duración: 0.1 día
- Recursos asignados: 1: developer
- Esfuerzo estimado: 0.1 días x recursos
- **ID: 4** Una mejora de como el usuario recibe mensaje de retroalimentación correcto, consistentes y bien explicados, como la implementación de los de distintos mensajes que sean relevantes al flujo por el cual esta pasando el usuario, para así darle suficiente feedback como para que sepa que esta pasando en cada momento.

Ubicación del cambio: Todo el package interfaz

	FUNCIONES DEL PRODUCTO SOFTWARE															Analisis de impacto vertical	ID 4
Nivel de documentación	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15		
Especificación de requerimientos																	
Especificación de diseño																	
Casos de prueba																	
Código fuente	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
Analisis de impacto horizontal	IMPACTA TODO EL CODIGO DE LA INTERFAZ																

Figura 3.4: Tabla de análisis de impacto ID 4

- Duración: 1 día
- Recursos asignados: 2: developer, tester
- Esfuerzo estimado: 2 días x recursos

- **ID: 5** Sobre el estado y el informe que se hizo sobre la calidad de código y cumplimiento de estándares es importante realizar cambios y mejoras tanto en el dominio, handlers, interfaces y clasesTest.

Ubicación del cambio: Todo el código del sistema

	FUNCIONES DEL PRODUCTO SOFTWARE															Analisis de impacto vertical	ID 5
Nivel de documentación	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15		
Especificación de requerimientos																	
Especificación de diseño	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
Casos de prueba	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
Código fuente	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
Analisis de impacto horizontal	IMPACTA TODO EL CODIGO DE LA APLICACIÓN																

Figura 3.5: Tabla de análisis de impacto ID 5

- Duración: 3 días
- Recursos asignados: 1: developer
- Esfuerzo estimado: 3 días x recursos
- **ID: 6** Sobre el estado y el informe que se hizo sobre la cobertura de código, las pruebas unitarias necesitan un mejor diseño de las pruebas (impacta a la especificación de diseño), mejorar hasta llegar a cubrir todos los caminos posibles y completar las pruebas faltantes.

Ubicación del cambio: Todo las clases test dentro del testPackage

	FUNCIONES DEL PRODUCTO SOFTWARE															Analisis de impacto vertical	ID 6
Nivel de documentación	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15		
Especificación de requerimientos																	
Especificación de diseño																	
Casos de prueba																	
Código fuente	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
Analisis de impacto horizontal	IMPACTA EN TODAS LAS CLASES TEST																

Figura 3.6: Tabla de análisis de impacto ID 6

- Duración: 3 días
- Recursos asignados: 2: developer, tester
- Esfuerzo estimado: 6 días x recursos
- **ID: 7** Este cambio necesariamente va generar un impacto directo en el documento ESRE (especificación de requerimientos), encontrar especificaciones que no coinciden o que faltan, o que se agregan o se modifican o se arreglan. Todo lo mencionado anteriormente tiene que aparecer reflejado en el documento ESRE. Al cliente no se lo puede dejar del lado sobre los reportes de las funcionalidades a implementar y/o mejorar, no se le puede entregar al cliente

un sistema que no cumpla con lo especificado con las funcionalidades. Este reporte hay que tener en cuenta que también puede generar cambios a nivel de código. Un estándar de cualquier proyecto sólido es que cada requerimiento que se prometió tiene que aparecer en el ESRE y en la aplicación. Para ello para cada especificación de la especificación de requerimientos hay que hacer una revisión y ver porque no se esta cumpliendo y luego repararlo.

Ubicación del cambio: ESRE, especificación de diseño, casos de prueba, código fuente

Nivel de documentación	FUNCIONES DEL PRODUCTO SOFTWARE															Análisis de impacto vertical	ID 7
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15		
Especificación de requerimientos	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
Especificación de diseño	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
Casos de prueba	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
Código fuente	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
Análisis de impacto horizontal	IMPACTA TODO EL SISTEMA																

Figura 3.7: Tabla de análisis de impacto ID 7

- Duración: 3 días
- Recursos asignados: 2: gerente de SQA, developer
- Esfuerzo estimado: 6 días x recursos
- **ID: 8** Para realizar este cambio, se tiene que especificar en el ESRE, el cliente tenga un reporte de esta funcionalidad, realizar nuevos casos de prueba, el usuario pueda agregar la comida a la dieta pero que le lance un alerta o reporte al profesional, además que le aparezca al usuario las comidas del plan de dieta. Por eso se decidió que se llevaran dos listas, en una el usuario tendrá las comidas que quiere agregar a la ingeridas diarias y la otra los alimento ingeridos que tiene el plan y se vayan sacando de esa lista.

Ubicación del cambio: ESRE, especificación de diseño, casos de prueba, código fuente: PanelDietaDiariaUsuario.java

Nivel de documentación	FUNCIONES DEL PRODUCTO SOFTWARE															Análisis de impacto vertical	ID 8
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15		
Especificación de requerimientos													X				
Especificación de diseño													X				
Casos de prueba													X				
Código fuente													X				
Análisis de impacto horizontal																	

Figura 3.8: Tabla de análisis de impacto ID 8

- Duración: 1 día
- Recursos asignados: 2: developer, tester

- Esfuerzo estimado: 2 días x recursos
- **ID: 9** Este cambio, se realiza junto al registro de los usuarios, esto va afectar a los casos de prueba sobre el registro de usuario.
Ubicación del cambio: ESRE, especificación de diseño, casos de prueba y escenario de caso de uso 2.4.2, código fuente: PanelRegistroUsuario.java PanelRegistroProfesional.java

Nivel de documentación	FUNCIONES DEL PRODUCTO SOFTWARE															Análisis de impacto vertical	ID 9
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15		
Especificación de requerimientos																	
Especificación de diseño																	
Casos de prueba	X	X															
Código fuente	X	X															
Análisis de impacto horizontal																	

Figura 3.9: Tabla de análisis de impacto ID 9

- Duración: 0.5 días
- Recursos asignados: 2: developer, tester
- Esfuerzo estimado: 1 días x recursos
- **ID: 10** Informar sobre el usuario autenticado en la aplicación en cada ventana. El sistema debe informar todo el tiempo que el usuario este autenticado en cada ventana.

Ubicación del cambio: código fuente: todas las interfaces

Nivel de documentación	FUNCIONES DEL PRODUCTO SOFTWARE															Análisis de impacto vertical	ID 10
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15		
Especificación de requerimientos																	
Especificación de diseño																	
Casos de prueba																	
Código fuente	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
Análisis de impacto horizontal	IMPACTA TODO EL CODIGO DE LA INTERFAZ																

Figura 3.10: Tabla de análisis de impacto ID 10

- Duración: 1 día
- Recursos asignados: 1: developer
- Esfuerzo estimado: 1 días x recursos
- **ID: 11** Implementación/solución de al menos 6 items de reporte de defectos especificados en la documentación pero se llegó a un acuerdo con el cliente que: **todo reporte que este en estado “no arreglado” sea reparado e implementado.**

ESRE, especificación de diseño, casos de prueba, código fuente:
En cada funcionalidad de cada implementación.

Nivel de documentación	FUNCIONES DEL PRODUCTO SOFTWARE														Análisis de impacto vertical ID 11
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F16
Especificación de requerimientos	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Especificación de diseño	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Casos de prueba	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Código fuente	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Ánálisis de impacto horizontal	IMPACTA A TODO EL SISTEMA POR SER CAMBIO GRANDES DESDE LA ESPECIFICACION DE REQUERIMIENTOS HASTA EL CODIGO FUENTE														

Figura 3.11: Tabla de análisis de impacto ID 11

- Duración: 6 días
- Recursos asignados: 2: developer, tester
- Esfuerzo estimado: 12 días x recursos
- **ID: 12** Informar sobre el usuario autenticado en la aplicación en cada ventana. El sistema debe informar todo el tiempo que el usuario este autenticado en cada ventana

Ubicación del cambio: código fuente: todas las interfaces

Nivel de documentación	FUNCIONES DEL PRODUCTO SOFTWARE														Análisis de impacto vertical ID 12
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
Especificación de requerimientos			X	X								X			
Especificación de diseño				X	X							X			
Casos de prueba					X	X						X			
Código fuente						X	X					X			
Ánálisis de impacto horizontal															

Figura 3.12: Tabla de análisis de impacto ID 12

- Duración: 0.8 días
- Recursos asignados: 2: developer, tester
- Esfuerzo estimado: 1.6 días x recursos
- **ID: 13** Para realizar este cambio, se necesita agregar un botón de refresh en las interfaces.

Ubicación del cambio: nuevos casos de prueba, código fuente: PanelVerHistorialDeUsuario.java PanelConsultaProfesional.java, PanelConsultaDesdeProfesional.java

	FUNCIONES DEL PRODUCTO SOFTWARE															Análisis de impacto vertical ID 13	
Nivel de documentación	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15		
Especificación de requerimientos																	
Especificación de diseño																	
Casos de prueba										X	X		X				
Código fuente											X	X		X			
Análisis de impacto horizontal																	

Figura 3.13: Tabla de análisis de impacto ID 13

- Duración: 0.9 días
- Recursos asignados: 2: developer, tester
- Esfuerzo estimado: 1.8 días x recursos
- **ID: 14** Para realizar esta solicitud hay que modificar parte del dominio del sistema, darle más privilegios al profesional sobre el manejo de los alimentos, cambio en la clase sistema, modificación de la interfaz del root, y del profesional.

Ubicación del cambio: código fuente: Sistema.java, PanelHome-Profesional.java

	FUNCIONES DEL PRODUCTO SOFTWARE															Análisis de impacto vertical ID 14	
Nivel de documentación	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15		
Especificación de requerimientos										X	X						
Especificación de diseño										X	X						
Casos de prueba										X	X						
Código fuente											X	X					
Análisis de impacto horizontal																	

Figura 3.14: Tabla de análisis de impacto ID 14

- Duración: 1 día
- Recursos asignados: 2: developer, tester
- Esfuerzo estimado: 2 días x recursos

4. Plan de proyecto de mantenimiento

4.1. Identificación de interesados

En esta sección se realiza una muestra de como se llego a la identificación de los distintos interesados, se entiende como interesado a un individuo, grupo u organización que puede afectar, verse afectado, o percibirse a sí mismo como afectado por una decisión, actividad o resultado del proyecto.

Para ello utilizamos la técnica de análisis de interesados y el registro de los mismos. Permitiendo así identificar intereses, expectativas y la influencia de los interesados, y relacionarlos con el propósito del proyecto.

Nombre	Posicion(Cargo)	Roll(en Proyecto)	Requerimientos	Informacion de contacto	Expectativas	Influencia	Clasificacion
Ignacio Olivera	Director de proyecto	Responsable SQA	Documentar y Refactorizacion del codigo	nachoOliveraR@hotmail.com	Llegar al tiempo estimado	Alta	Interno
Tomas Dilema	Director de proyecto y Analista Funcional	Responsable SQA	Documentar y Refactorizacion del codigo	tomasDilema@gmail.com	Llegar al tiempo estimado	Alta	Interno
Jorge Bruno Fernandez	Director de proyecto	Responsable SQA	Documentar y Refactorizacion del codigo	brunfernandezrod@hotmail.com	Llegar al tiempo estimado	Alta	Interno
Gerardo Maturro	Cliente del sistema	Cliente Principal	Recibir el producto con la calidad deseada	94366788	Recibir el proyecto y producto final	Alta	Externo
Sebastian Peralta	Cliente del sistema	Cliente Principal	Recibir el producto con la calidad deseada	speralta78@gmail.com	Recibir el proyecto y producto final	Alta	Externo
Autores originales del proyecto			—	—	—	Media	Externo
MSP (Ministerio de Salud Publica)	Ministerio de Salud	Ente Regulatorio	—	https://www.gub.uy/ministerio-salud-publica/	—	Media	Externo
Nutricionista Stefanie Heguy	Competidor	Competencia	—	www.Nutriev.org	—	Media	Externo
Juan Toledo	Consultor de tecnologias		Asesoramiento y Gestión en base a la experiencia previa en proyectos de este estilo	JuanToledo1998@hotmail.com	—	Media	Externo
Profesional	Profesional de Alimentacion	Usuario Profesional	Producto final	—	Tener el producto final con las funcionalidades correctamente	Alta	Externo
Usuario que quiera realizar un seguimiento de una alimentacion saludable	Paciente	Usuario	Producto final	—	Tener el producto final con las funcionalidades correctamente	Media	Externo
Root	Usuario Root (ABM) Y gestion de usuario	Usuario Administrador	Producto final	—	Tener el producto final con las funcionalidades correctamente	Alta	Externo

Figura 4.1: Interesados Lista

4.1.1. Matriz poder/predecibilidad

Para el análisis se diseña la matriz poder/predecibilidad (“interés”), este agrupa a los interesados dependiendo su nivel de autoridad a que se le denomina (“poder”) y su nivel de preocupación (“interés”) con respecto a los resultados del proyecto.

Poder	Alto	Autores originales del Proyecto MSP (Ministerio de Salud Pública)	Gerardo Maturro Sebastián Peralta Jorge Bruno Fernandez Ignacio Olivera Tomas Dilema Profesional Usuario Root
	Bajo	Juan Toledo	Paciente
		Bajo	Alto
Predecibilidad			

Figura 4.2: Interesados Matriz poder/predecibilidad

Las estrategia que se usa para realizar el seguimiento y el control de los interesados es la siguiente:

- **Gestionar de cerca:** Poder: Alto Predecibilidad: Alto
- **Mantener satisfecho:** Poder: Alto Predecibilidad:Bajo
- **Informar:** Poder: Bajo Predecibilidad: Alto
- **Monitorear:** Poder: Bajo Predecibilidad: Bajo

4.2. Definición de objetivos del proyecto

Los objetivos de proyecto serán nuestra guía para definir la estrategia con la que debemos trabajar durante a evolución del proyecto. Por eso, es clave que cumplan con una serie de características básicas para poder alcanzarlo. Se opto por seguir los objetivos SMART, este es aquel con el que se definen las metas que debe seguir y alcanzar los objetivos para los que se da forma una estrategia. El modelo de gestión SMART es uno de los más conocidos a la hora de definir y calcular el alcance de un proyecto. Este sigue cinco principios a seguir para construir las metas:

- **Specific** (Especifico)
- **Measurable** (Medible)
- **Achievable** (Alcanzable)

- **Realistic** (Realista)
- **Timely** (Definido en un plazo de tiempo determinado)

A continuación se presentaran los objetivos tenidos en cuenta utilizando las metas SMART:

1. Alcanzar una cobertura de los métodos en la herramienta de análisis de cobertura JaCoCo del 95 % en un plazo de 2 hora y 30 minutos.
2. Reducir en un 24 % la complejidad cognitiva del código llevándolas de 53 a 45 en un plazo de 3 horas.
3. Capturar correctamente las excepciones para la serialización correcta llevando de 2 a 0 la cantidad de bloques try-catch-finally que no están implementados correctamente, en un plazo de 30 minutos.
4. Capturar correctamente las excepciones críticas de objetos nulos llevando 2 a 0 la cantidad de bloques try-catch que no están implementados en un plazo de 45 minutos.
5. Eliminar los unused imports del código de la aplicación, llevándolos de 28 a 0 en 45 minutos.
6. Corregir todos los casos de prueba que no coinciden con el flujo real de la aplicación, llevándolos de 11 a 0 en 2 horas.
7. Crear una clase de constantes llevando la cantidad de literales 36 a 0 en 1 hora.
8. Implementar los fixtures de las pruebas unitarias (@Before, @After) llevando 34 a 0 la cantidad no implementada en 2 horas.
9. Corregir todos los incumplimientos de las heurísticas de Nielsen en la interfaz, llevándolos de 10 a 0, para mejorar la experiencia de usuario, en un plazo de 2 horas.
10. Quitar todas las variables no usadas en el código llevándolos de 20 a 0, para mejorar la mantenibilidad del código y posteriormente a la aplicación en 1 hora.
11. Corregir todos los errores “overridable method called during object construction” llevándolos de 10 a 0, para mejorar y evitar hacer caer a la aplicación, en un plazo de 2 horas y media.

4.3. Definición del alcance del producto y del proyecto

El plan de gestión del alcance define el modo en que los equipos del proyecto han de determinar el tipo de requisitos que es necesario recopilar para el proyecto. Por

eso para comenzar a definir el alcance del producto y del proyecto se creo un plan de alcance.

Como se planificó en anterioridad un ciclo de vida predictivo como el cascada, los entregables del proyecto se definen al comienzo del proyecto y cualquier cambio en el alcance es gestionado en forma progresiva.

Como se elaboro el plan de gestión del alcance: En la gestión del alcance del proyecto se incluye todos los procesos necesarios para garantizar que el proyecto incluya todo (y únicamente todo) el trabajo requerido para completarlo con éxito. El objetivo principal de la gestión del alcance del proyecto es definir y controlar qué se incluye y qué no se incluye en el proyecto.

- 1.1 Recopilar requisitos. Se recopilan los requisitos, lo cual consiste en definir y documentar las necesidades de los interesados a fin de cumplir con los objetivos del proyecto. El éxito del proyecto depende directamente del cuidado que se tenga en obtener y gestionar los requisitos del proyecto y del producto.
- 1.2 Definir el alcance. Se definió el alcance del proyecto el cual es el proceso que consiste en desarrollar una descripción detallada del proyecto y del producto. La preparación de una declaración detallada del alcance del proyecto es fundamental para su éxito, y se elabora a partir de los entregables principales, los supuestos y las restricciones que se documentan durante el inicio del proyecto.
- 1.3 Crear la EDT.

Se creo la EDT, la cual detalla el proceso de subdividir los entregables del proyecto y el trabajo del proyecto en componentes más pequeños y más fáciles de manejar. La estructura de desglose del trabajo (EDT) es una descomposición jerárquica, basada en los entregables del trabajo que debe ejecutar el equipo del proyecto para lograr los objetivos del proyecto y crear los entregables requeridos.

4.3.1. Definición del alcance del proyecto y lista de entregables

Nombre del proyecto: Alimentación Saludable
Fecha última actualización: -
Preparado por: -

- **Breve descripción del proyecto:** Alimentación Saludable es un software que permite a un usuario y a un profesional interactuar, permite a un profesional indicarle una dieta a un usuario y el usuario indica las comidas que comió en la semana.
- **Alcance del producto:** Usuarios, profesionales, y ciertas comidas detalladas.
- **Entregables:** Definidos en la subsección de abajo.

- **Criterio de aceptación:** Actualizar dietas, altas, bajas y modificaciones de datos, cambios de rol.
- **Exclusiones:** Roles externos a usuario y profesional, comidas no agregadas por profesionales.
- **Supuestos:** Los usuarios son personas comunes y los profesionales son médicos o nutricionistas certificados.
- **Restricciones:** Un usuario no puede acceder como profesional y viceversa, no se pueden seleccionar comidas no ingresadas en el sistema.
- **Riesgos preliminares identificados:** Los usuarios pueden mentir en sus dietas, los profesionales pueden crear dietas no aptas para los usuarios.
- **Requisitos de aprobación:** Tanto los usuarios como los profesionales deben mostrarse conformes con el software y sus funcionalidades, y el mismo debe respetar los estándares y objetivos determinados por la empresa.

Lista de entregables

Se realizo una lista de entregables desarrollando para cada tarea una descripción,criterio,modalidad,plazo de la realización dicha tarea, y el responsable a cargo de ese entregable:

- **E1:** Documento del estado inicial de la calidad del código (estándares, Code Smells), de las pruebas unitarias y casos de prueba funcionales.
Criterio: Calidad de código (criterios de las herramientas), pruebas funcionales criterios OK, Casi OK, No OK.
Modalidad: Pruebas exploratorias, herramientas de análisis estático, JaCoCo, pruebas de caja negra.
Plazo: 9 días.
Responsables: Gerente SQA, Project Manager, Senior Developer.
- **E2:** Documento del estado inicial de la calidad la interfaz.
Criterio: Heurísticas de Nielsen, categorías y métrica.
Modalidad: En base a las heurísticas de Nielsen se llevo a cabo un análisis por cada interfaz del programa, además subdividimos las pruebas en categorías y ingresamos una métrica de 0 a 100 % dependiendo cada prueba.
Plazo: 5 días.
Responsables: Gerente SQA, Project Manager, Diseñador gráfico.
- **E3:** Plan de proyecto de aseguramiento y mantenimiento del sistema.
Criterio: -
Modalidad: Se definió alcance y objetivos del proyecto y sus respectivas estimaciones. Se utilizo modelo ciclo de vida en cascada.
Plazo: 8 Días.
Responsables: Gerente SQA, Project Manager.

- **E4:** Elaboración y mejora del documento ERS (Especificación de Requerimientos del Software).
 - Criterio: -
 - Modalidad: Elaboración con autores originales del proyecto.
 - Plazo: -
 - Responsables: Gerente SQA, Project Manager

- **E5:** Mejora visuales, mejora de usabilidad (utilizando métricas de Nielsen como referencia).
 - Criterio: Heurísticas de Nielsen.
 - Modalidad: En base a las Heurísticas de Nielsen se va a proporcionar las modificaciones necesarias.
 - Plazo: Se encuentra en la estimación de esfuerzo.
 - Responsables :Senior developer, Junior developer, Diseñador gráfico.

- **E6:** Código mejorado, refactorizado y todas las funcionalidades estipuladas implementadas.
 - Criterio: Estándar de codificación Java, buenas prácticas de codificación, evitar Code Smells, implementar patrones y corregir errores.
 - Modalidad: Corregir errores y agregar elementos nuevos.
 - Plazo: Se encuentra en la estimación de esfuerzo.
 - Responsables: Senior developer, Junior developer, Tester.

- **E7:** Diseño de pruebas unitarias de las nuevas funcionalidades y reDiseño de las pruebas unitarias ya existentes.
 - Criterio: Estándar de codificación Java, buenas prácticas de codificación, evitar Code Smells, buen manejo de pruebas ATRIP.
 - Modalidad: Corregir errores y agregar pruebas.
 - Plazo: Se encuentra en la estimación de esfuerzo.
 - Responsables: Tester

- **E8:** Reporte final de proyecto.

El proceso de crear la estructura de desglose del trabajo (EDT) o WBS (work breakdown structure), consiste en dividir al proyecto en menores componentes para facilitar la planificación del proyecto. La EDT nos hace organizar y definir el alcance total del proyecto y representa el trabajo especificado en el enunciado del alcance del proyecto aprobado y vigente.

A continuación se presenta la EDT desarrollada por el equipo para este proyecto:

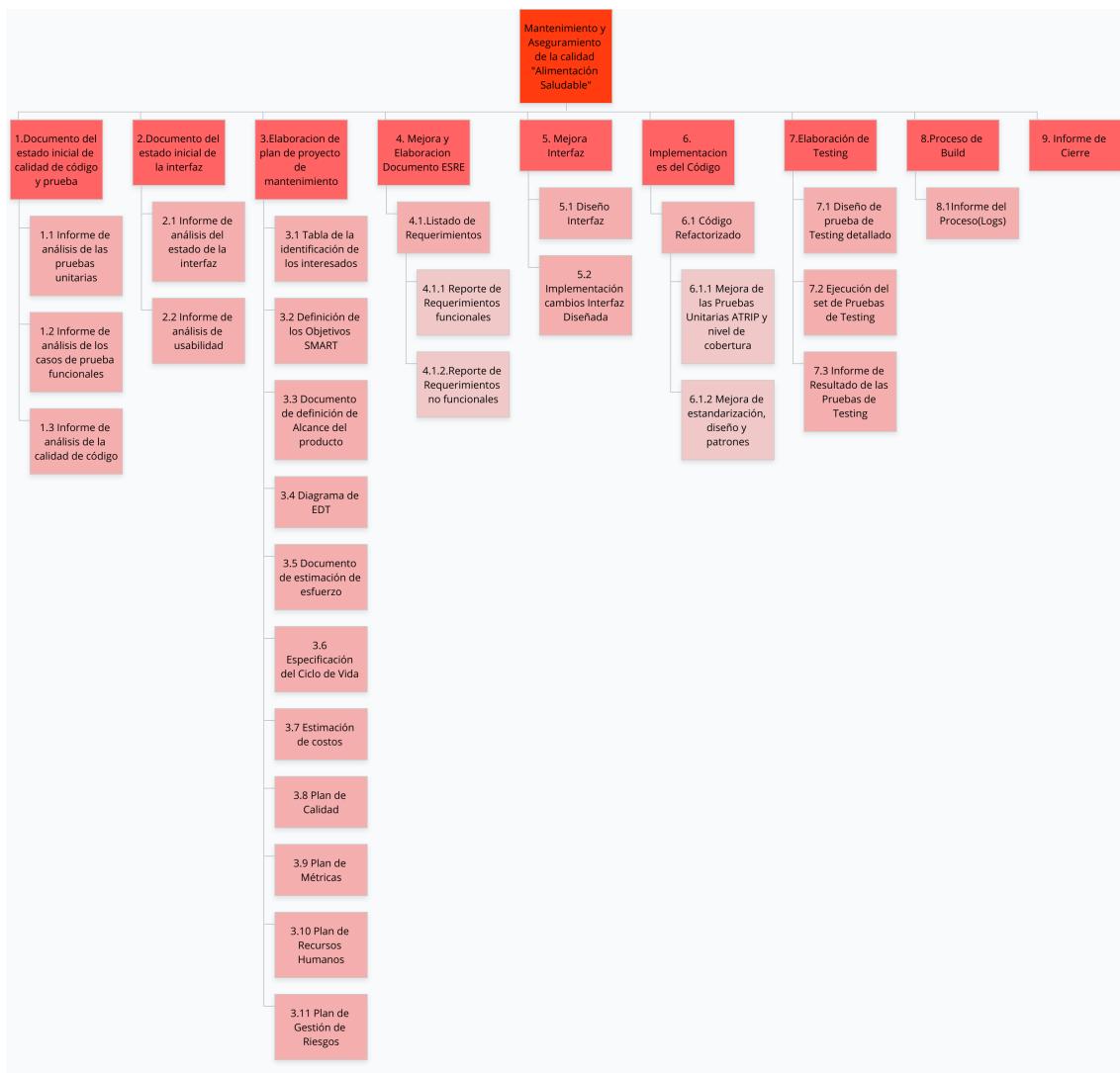


Figura 4.3: Diagrama EDT

4.3.2. Diccionario de EDT

ID	1.1
Cuenta control	1
Ultima actualización	8/10/2020
Responsable	Bruno Fernández.
Descripción	Informe del estado inicial de las pruebas unitarias post análisis de código o evidencia de la incorrectitud.
Criterio de aceptación	Informe de la correctitud de las pruebas unitarias (ATRIP) o evidencia de la incorrectitud.
Recursos asignados	Gerente SQA, Project Manager.
Cantidad personas	2
Entregables	Informe en formato documento 302, índice.
Duración	5 horas.

ID	1.2
Cuenta control	1
Ultima actualización	8/10/2020
Responsable	Tomas Dilema.
Descripción	Informe de análisis del estado inicial de los casos de prueba funcionales. Reproducción del caso de prueba para verificar su correctitud o evidencia de la incorrectitud.
Criterio de aceptación	Informe de análisis con los casos de prueba que no verificarán el uso real de la aplicación.
Recursos asignados	Gerentes SQA, Proyect Manager.
Cantidad personas	2
Entregables	Informe en formato documento 302, índice.
Duración	5 horas.

ID	1.3
Cuenta control	1
Ultima actualización	10/10/2020
Responsable	Bruno Fernandez.
Descripción	Informe de análisis del estado inicial del código. Verificación del cumplimiento de estándares y reporte de bugs.
Criterio de aceptación	Evidencia de la aplicación de al menos 3 herramientas de software para medir la calidad de código y el cumplimiento de estándares.
Recursos asignados	Gerentes SQA, Proyect Manager.
Cantidad personas	2
Entregables	Informe en formato documento 302.
Duración	5 horas.

ID	2.1
Cuenta control	2
Ultima actualización	12/10/2020
Responsable	Bruno Fernández.
Descripción	Informe de análisis del estado inicial de la interfaz.
Criterio de aceptación	Cumplimiento al menos de 80 % - 100 % de las métricas estipuladas para la interfaz.
Recursos asignados	Gerentes SQA, Proyect Manager.
Cantidad personas	2
Entregables	Informe en formato documento 302, índice.
Duración	5 horas.

ID	2.2
Cuenta control	2
Ultima actualización	11/10/2020
Responsable	Tomas Dilema.
Descripción	Informe de análisis de usabilidad de la interfaz.
Criterio de aceptación	Cumplimiento de las métricas de Nielsen.
Recursos asignados	Gerente SQA, Diseñador gráfico.
Cantidad personas	2
Entregables	Informe en formato documento 302.
Duración	5 horas.

ID	3.1
Cuenta control	3
Ultima actualización	11/10/2020
Responsable	Bruno Fernández.
Descripción	Tabla de la identificación de interesados.
Criterio de aceptación	Reconocimiento total de las partes afectadas en el proyecto, tanto como el equipo principal como externos al equipo.
Recursos asignados	Gerente SQA, Diseñador gráfico.
Cantidad personas	2
Entregables	Informe en formato documento 302, sección lista de interesados.
Duración	5 horas.

ID	3.2
Cuenta control	3
Ultima actualización	11/10/2020
Responsable	Juan Olivera.
Descripción	Definición de los objetivos SMART.
Criterio de aceptación	Objetivos bien definidos según el criterio SMART.
Recursos asignados	Gerente SQA, Proyect Manager.
Cantidad personas	2
Entregables	Documento de SQA 302, Sección Objetivos SMART.
Duración	5 horas.

ID	3.3
Cuenta control	3
Ultima actualización	11/10/2020
Responsable	Tomas Dilema.
Descripción	Documento de definición de alcance del producto.
Criterio de aceptación	Documento con evidencia de análisis de producto para traducir las descripciones de alto nivel en entregables tangibles.
Recursos asignados	Gerente SQA, Project manager.
Cantidad personas	2
Entregables	Informe en formato documento 302.
Duración	5 horas.

ID	3.4
Cuenta control	3
Ultima actualización	11/10/2020
Responsable	Tomas Dilema.
Descripción	Diagrama de EDT.
Criterio de aceptación	Esquema con desglose jerárquico de entregables, de no más de 3 niveles.
Recursos asignados	Project manager.
Cantidad personas	2
Entregables	Informe en formato documento 302, sección diagrama EDT.
Duración	5 horas.

ID	3.5
Cuenta control	3.
Ultima actualización	11/10/2020
Responsable	Bruno Fernández.
Descripción	Documento de estimaciones de esfuerzo.
Criterio de aceptación	Todas las tareas deben tener asociada una estimación del esfuerzo.
Recursos asignados	Gerente SQA, Project manager.
Cantidad personas	2
Entregables	Informe en formato documento 302, sección estimación de esfuerzo.
Duración	5 horas.

ID	3.6
Cuenta control	3
Ultima actualización	11/10/2020
Responsable	Bruno Fernández.
Descripción	Especificación de ciclo de vida.
Criterio de aceptación	Características principales de un modelo de ciclo de vida predictivo.
Recursos asignados	Project manager.
Cantidad personas	1
Entregables	Informe en formato documento 302, sección especificación de ciclo de vida.
Duración	5 horas.

ID	3.7
Cuenta control	3
Ultima actualización	11/10/2020
Responsable	Tomas Dilema.
Descripción	Estimación de costos.
Criterio de aceptación	El presupuesto inicial debe surgir de la técnica estimación por tres valores.
Recursos asignados	Project manager.
Cantidad personas	1
Entregables	Informe en formato documento 302, sección estimación de costos.
Duración	5 horas.

ID	3.8
Cuenta control	3
Ultima actualización	11/10/2020
Responsable	Bruno Fernández.
Descripción	Plan de aseguramiento calidad de software. Especifica qué procedimientos y recursos asociados deben aplicarse, quién debe aplicarlos y cuándo deben aplicarse.
Criterio de aceptación	El documento debe seguir el plan de SQA según la IEEE.
Recursos asignados	Gerente de SQA y Project manager.
Cantidad personas	2
Entregables	Informe en formato documento 302, SQA de software.
Duración	5 horas.

ID	3.9
Cuenta control	3
Ultima actualización	10/10/2020
Responsable	Juan Olivera.
Descripción	Plan de métricas en base al análisis de código. Se definen atributos de calidad para medir las mejoras al final del proyecto.
Criterio de aceptación	Inclusión de todas las métricas encontradas a partir del análisis del código con su estado inicial y final definido.
Recursos asignados	Project manager, Senior developer.
Cantidad personas	2
Entregables	Informe en formato documento 302, sección plan de métricas.
Duración	5 horas.

ID	3.10
Cuenta control	3
Ultima actualización	10/10/2020
Responsable	Tomas Dilema.
Descripción	Descripción del plan de RRHH. La manera en que se tratarán y estructurarán, en el ámbito del proyecto, los roles y responsabilidades.
Criterio de aceptación	El plan debe incluir todas las tareas a ser realizadas y su responsable asociado más la remuneración de este último.
Recursos asignados	Project manager.
Cantidad personas	1
Entregables	Informe en formato documento 302, sección plan de RHH y responsabilidades.
Duración	5 horas.

ID	3.11
Cuenta control	3
Ultima actualización	10/10/2020
Responsable	Tomas Dilema.
Descripción	Plan de gestión de riesgos. Identificación y análisis de riesgos para dar respuestas que disminuyan el impacto negativo.
Criterio de aceptación	Informe de un análisis exhaustivo que indique la detección o no de posibles riesgos y su impacto.
Recursos asignados	Project manager, Gerente de SQA
Cantidad personas	1
Entregables	Informe en formato documento 302, sección plan de Gestión de riesgos.
Duración	5 horas.

ID	4.1.1
Cuenta control	4.1
Ultima actualización	10/10/2020
Responsable	Bruno Fernández.
Descripción	Especificación de todos los requerimientos no funcionales, ya existentes y agregados.
Criterio de aceptación	Correcta definición del requerimiento, acorde a las restricciones de la aplicación.
Recursos asignados	Project manager, Senior developer.
Cantidad personas	2
Entregables	Informe en formato documento 302, sección plan de métricas.
Duración	5 horas.

ID	4.1.2
Cuenta control	4.1
Ultima actualización	10/10/2020
Responsable	Bruno Fernández.
Descripción	Especificación de todos los requerimientos funcionales, ya existentes y agregados.
Criterio de aceptación	Correcta definición del requerimiento, acorde a las restricciones de la aplicación.
Recursos asignados	1 persona.
Duración	1 hora.

ID	5.1
Cuenta control	5
Ultima actualización	10/10/2020
Responsable	Bruno Fernández.
Descripción	Especificación del diseño de la interfaz, se realizan bocetos, prototipos de los cambios requeridos.
Criterio de aceptación	Correcto diseño, uso de una buena paleta de colores, buen diseño en base a las 10 heurísticas de Nielsen.
Recursos asignados	Diseñador gráfico.
Cantidad personas	1
Entregables	Boceto realizado a lápiz, y después en la herramienta de software prototype.
Duración	5 horas.

ID	5.2
Cuenta control	5.1
Ultima actualización	10/10/2020
Responsable	Juan Olivera.
Descripción	Se implementan los cambios solicitados en el punto anterior en la interfaz.
Criterio de aceptación	Correcta definición del requerimiento, acorde a las restricciones de la aplicación.
Recursos asignados	1 persona.
Cantidad personas	1
Entregables	Informe en formato documento 302, sección especificación de ciclo de vida.
Duración	5 horas.

ID	6.1.1
Cuenta control	6.1
Ultima actualización	08/10/2020
Responsable	Tomas Dilema.
Descripción	Se mejoran todas las pruebas unitarias contenidas en el código, aumentando así el nivel de cobertura.
Criterio de aceptación	Correcta definición del requerimiento, acorde a las restricciones de la aplicación.
Recursos asignados	1 persona.
Cantidad personas	1
Entregables	Informe en formato documento 302, sección especificación de ciclo de vida.
Duración	5 horas.

ID	6.1.2
Cuenta control	6.1
Ultima actualización	10/10/2020
Responsable	Bruno Fernández.
Descripción	Se mejora la estandarización del código, el diseño de las pruebas y los patrones de las mismas.
Criterio de aceptación	Correcta definición del requerimiento, acorde a las restricciones de la aplicación.
Recursos asignados	1 persona.
Cantidad personas	1
Entregables	Informe en formato documento 302, sección especificación de ciclo de vida.
Duración	5 horas.

ID	7.1
Cuenta control	7
Ultima actualización	10/10/2020
Responsable	Bruno Fernández.
Descripción	Se diseñan pruebas de testing detalladas acordes al proyecto.
Criterio de aceptación	Correcta definición del requerimiento, acorde a las restricciones de la aplicación.
Recursos asignados	1 persona.
Cantidad personas	1
Entregables	Informe en formato documento 302, sección especificación de ciclo de vida.
Duración	5 horas.

ID	7.2
Cuenta control	7
Ultima actualización	10/10/2020
Responsable	Bruno Fernández.
Descripción	Se ejecutan las pruebas detalladas en el punto anterior para obtener sus respectivos resultados.
Criterio de aceptación	Correcta definición del requerimiento, acorde a las restricciones de la aplicación.
Recursos asignados	1 persona.
Cantidad personas	1
Entregables	Informe en formato documento 302, sección especificación de ciclo de vida.
Duración	5 horas.

ID	7.3
Cuenta control	7
Ultima actualización	10/10/2020
Responsable	Bruno Fernández.
Descripción	Se realiza un informe de los resultados obtenidos en el punto anterior.
Criterio de aceptación	Correcta definición del requerimiento, acorde a las restricciones de la aplicación.
Recursos asignados	1 persona.
Cantidad personas	1
Entregables	Informe en formato documento 302, sección especificación de ciclo de vida.
Duración	5 horas.

ID	8.1
Cuenta control	8
Ultima actualización	10/10/2020
Responsable	Tomas Dilema.
Descripción	Se detalla en un archivo de registro la compilación del build del gestor de construcción Maven.
Criterio de aceptación	Correcta definición del requerimiento, acorde a las restricciones de la aplicación.
Recursos asignados	1 persona.
Cantidad personas	1
Entregables	Informe en formato documento 302, sección especificación de ciclo de vida.
Duración	5 horas.

ID	9
Cuenta control	9
Ultima actualización	10/10/2020
Responsable	Bruno Fernández.
Descripción	Se realiza la actividad de puesta en marcha del sistema y un pedido de informe del mismo.
Criterio de aceptación	Correcta definición del requerimiento, acorde a las restricciones de la aplicación.
Recursos asignados	1 persona.
Cantidad personas	1
Entregables	Informe en formato documento 302, sección especificación de ciclo de vida.
Duración	5 horas.

4.3.3. Alcance del producto

En esta sección se definen como las características que debe cumplir el producto resultante del proyecto. Ya sea considerando su diseño, su función o su composición; el punto clave es que el alcance del producto se refiere a los entregables. El alcance del producto no solo aplica a productos o entregables físicos, sino que también es aplicable a servicios; cuando estos sean el objetivo final del proyecto. En el caso de un servicio, este concepto se centra en definir las tareas y responsabilidades del personal.

A continuación se listaran los requerimientos funcionales y no funcionales:

Requerimientos Funcionales

ID	RF1
Fecha de actualización	30/09/2020
Descripción	Se realiza la implementación del código de interfaz login y cerrar sesión.
Objetivo	Es importante que la interfaz tenga un login y cerrar sesión para que sistema tenga el manejo correcto.
Prioridad	Alta.
Criterio de aceptación	Finalización de todos los cambio implementado correctamente.

Tabla 4.1: Tabla Requerimiento funcional

ID	RF2
Fecha de actualización	30/09/2020
Descripción	El sistema podrá validar las consultas vacías.
Objetivo	Es importante que la interfaz valide las consultas vacías para que sea consistente y no genere nulos y sobrecargue el sistema de serializacion y posible base de datos.
Prioridad	Media.
Criterio de aceptación	Condiciones correctamente finalizadas e implementadas.

Tabla 4.2: Tabla Requerimiento funcional

ID	RF3
Fecha de actualización	30/09/2020
Descripción	El sistema permitirá un uso de listas para que el usuario pueda agregar comida a la dieta del plan recomendado por el profesional.
Objetivo	Implementar que el usuario pueda agregar comida a la dieta del plan recomendado.
Prioridad	Media.
Criterio de aceptación	Cuando la funcionalidad este implementada y 100 % funcional.

Tabla 4.3: Tabla Requerimiento funcional

ID	RF4
Fecha de actualización	30/09/2020
Descripción	El sistema informe sobre el usuario autenticado en la aplicación en cada ventana.
Objetivo	Implementación en el código del cambio propuesto.
Prioridad	Media.
Criterio de aceptación	Cuando la funcionalidad este implementada y 100 % funcional.

Tabla 4.4: Tabla Requerimiento funcional

ID	RF5
Fecha de actualización	30/09/2020
Descripción	El sistema permita al profesional agregar comida al sistema.
Objetivo	Implementación en el código del cambio propuesto enviando una lista de solicitud de alimento al usuario root.
Prioridad	Media.
Criterio de aceptación	Cuando la funcionalidad este implementada y 100 % funcional.

Tabla 4.5: Tabla Requerimiento funcional

ID	RF6
Fecha de actualización	30/09/2020
Descripción	El sistema permita el ingreso del principal nutriente.
Objetivo	Implementación en el código del cambio propuesto.
Prioridad	Baja.
Criterio de aceptación	Cuando la funcionalidad este implementada y 100 % funcional.

Tabla 4.6: Tabla Requerimiento funcional

ID	RF7
Fecha de actualización	30/09/2020
Descripción	El sistema permita el ingreso de registro de una contraseña con largo mayor a 8 dígitos .
Objetivo	Implementación en el código del cambio propuesto.
Prioridad	Alta.
Criterio de aceptación	Cuando implementación correcta de la contraseña con la validación de largo correspondiente.

Tabla 4.7: Tabla Requerimiento funcional

Requerimientos No Funcionales

ID	RNF1
Fecha de actualización	1/09/2020
Descripción	El sistema debe ser capaz de cerrar todas las ventas al momento del botón de salir.
Objetivo	Realizar la implementación a tiempo del shutdown de las ventanas correctamente sin que quede en el hilo del S.O.
Prioridad	Alta.
Criterio de aceptación	Shutdown óptimo de todas las ventana en el terminal que esta operando.

Tabla 4.8: Tabla Requerimiento funcional

ID	RNF2
Fecha de actualización	1/09/2020
Descripción	El sistema debe ser capaz de cerrar todas las ventas al momento del botón de salir.
Objetivo	Realizar la implementación a tiempo del shutdown de las ventanas correctamente sin que quede en el hilo del S.O.
Prioridad	Alta.
Criterio de aceptación	Shutdown óptimo de todas las ventana en el terminal que esta operando.

Tabla 4.9: Tabla Requerimiento funcional

ID	RNF3
Fecha de actualización	2/09/2020
Descripción	El sistema debe ser capaz de cerrar todas las ventas al momento del botón de salir.
Objetivo	Realizar la implementación a tiempo del shutdown de las ventanas correctamente sin que quede en el hilo del S.O.
Prioridad	Alta.
Criterio de aceptación	Shutdown óptimo de todas las ventanas en el terminal que esté operando.

Tabla 4.10: Tabla Requerimiento funcional

ID	RNF4
Fecha de actualización	1/09/2020
Descripción	Aumentar a el nivel de ayudas visuales de la interfaz para mejorar navegabilidad.
Objetivo	Realizar la implementación de las ayudas visuales siguiendo las heurísticas de Nielsen hasta a un nivel de navegabilidad mas que aceptable.
Prioridad	Alta.
Criterio de aceptación	Cuando se respeten todas las heurísticas de Nielsen y el nivel de navegabilidad.

Tabla 4.11: Tabla Requerimiento no funcional

ID	RNF5
Fecha de actualización	1/09/2020
Descripción	Mejorar los mensajes de retro alimentación.
Objetivo	Realizar todos los mensajes correctos con los popUps que corresponda.
Prioridad	Alta.
Criterio de aceptación	Arreglar y implementar correctamente todos los mensajes de retroalimentación.

Tabla 4.12: Tabla Requerimiento no funcional

ID	RNF6
Fecha de actualización	1/09/2020
Descripción	Realizar reporte de funcionalidades.
Objetivo	Realizar todos los reportes de funcionalidades.
Prioridad	Alta.
Criterio de aceptación	Realizar un reporte de todas las funcionalidades del sistema que coincidan con la especificación, las agregadas, modificadas, eliminadas y reparadas.

Tabla 4.13: Tabla Requerimiento no funcional

ID	RNF7
Fecha de actualización	1/09/2020
Descripción	Corregir las clases Test mal implementadas.
Objetivo	Corregir las clases de las pruebas unitarias mal implementadas, lazyClass, mala estandarización ATRIP y de Java.
Prioridad	Alta.
Criterio de aceptación	Realizar mejora de las pruebas unitarias para llegar a un mejor nivel de cobertura.

Tabla 4.14: Tabla Requerimiento no funcional

ID	RNF8
Fecha de actualización	1/09/2020
Descripción	Corregir y Realizas las pruebas funcionales .
Objetivo	Corregir y Realizas las pruebas funcionales que coinciden con el flujo real de la aplicación.
Prioridad	Alta.
Criterio de aceptación	Cuando todos los casos de prueba funcional representen un flujo real de la interfaz.

Tabla 4.15: Tabla Requerimiento no funcional

ID	RNF9
Fecha de actualización	1/09/2020
Descripción	Disminuir la cantidad de bugs dentro del sistema.
Objetivo	Disminuir los bugs existentes del sistema como la complejidad cognitiva .
Prioridad	Alta.
Criterio de aceptación	Cuando el sistema adquiera una correctitud adecuada.

Tabla 4.16: Tabla Requerimiento no funcional

ID	RNF10
Fecha de actualización	1/09/2020
Descripción	Implementar los cambios previstos en la interfaz.
Objetivo	Implementar todos los cambios previstos en la interfaz de la aplicación, botones, listas, labels, etc...
Prioridad	Alta.
Criterio de aceptación	Cuando se completen todos los cambios previstos.

Tabla 4.17: Tabla Requerimiento no funcional

4.4. Estimación del esfuerzo

ACTIVIDADES PLANIFICADAS PARA LA REALIZACION DEL MANTENIMIENTO DEL PROYECTO ALIMENTACION SALUDABLE										
ID	ID.Unidad de Trabajo	Nombre Actividad	Descripción	Actividad Siguiente	Típos de Habilidades	Requeridas	Duración (Días)	Cantidad Personal	Estimación de Esfuerzo	Restriciones
RFI1.0	5.1	Realizar Boeteo de los cambios de la interfaz	Se realiza un prototipo en aplicación Prototype	Actividad posterior ID:RFP1.1	Desarrollador	3	1	3	Cumplimiento de estandares y metricas	El profesional cumple con las peticiones establecidas
RFI1.1	5.2	Implementación de los cambios en base a las heuristicas de Nielsen	Se completa la implementación acorde con los cambios establecidos	Actividad posterior ID:RFP1.2	Desarrollador	4	1	4	Ajustar modificaciones extraidas de boeteo	El desarrollador realiza los cambios en interfaz acorde con las modificaciones del boeteo
RFI1.2	5.2	Implementación de ayuda visuales	Se agrega una guia visual para facilitar el manejo en el proxima	Actividad posterior ID:RFP1.3	Desarrollador	2	1	2	Cumplimiento de estandares y metricas	El desarrollador realiza los cambios en tiempo y condiciones
RFI1.3	5.2	Implementación Interfaz Principal Login	Se agrega la interfaz login para brindar mayor seguridad al programa	Actividad posterior ID:RFP1.4	Desarrollador, Junior Desarrollador	1	2	2	Cumplimiento de estandares y metricas	El desarrollador agrega el login en el tiempo establecido
RFI1.4	5.2	Implementación del cambio interfaz Profesional	Se adiciona cambio interfaz profesional por adicion de funciones	Actividad posterior ID:RFP1.0	Desarrollador, Junior Desarrollador	3	2	6	Cumplimiento de estandares y metricas	El desarrollador modifica esta funcionalidad en el tiempo establecido cumpliendo las restricciones
RFI1.5	5.2	Implementar funcionalidad de listas Agregar Comida a la dieta.	Se crean dos listas de alimentos, el usuario agrega los alimentos que ingiro y los del plan del profesional a cargo	Actividad posterior ID:RFP1.0	Desarrollador, Junior Desarrollador	2	2	4	Cumplimiento de estandares y metricas	El desarrollador agrega la funcionalidad cumpliendo las restricciones en tiempo.
RFI1.6	5.2	Implementación refresh de listas	Se agrega boton "refresh"	Actividad posterior ID:RFP1.0	Desarrollador, Junior Desarrollador	1	2	2	Cumplimiento de estandares y metricas	El desarrollador agrega la funcionalidad respondiendo las restricciones en tiempo.
RFI1.7	5.2	Mejorar los mensajes de retroalimentación.	Se implementan mensajes de alerta PopUps	Actividad posterior ID:RFP1.0	Desarrollador	2	1	2	Cumplimiento de estandares y metricas	El desarrollador cumple con las restricciones en tiempo.
RFPU1.0	6.1.1	Mejorar la cobertura de código	Alcanzar una cobertura de 90% de acuerdo a la implementación de analisis de cobertura de Code coverage 90 %	Actividad posterior ID:RFP1.1	Senior Desarrollador	3	1	3	Cumplimiento de estandares y metricas	El desarrollador logra la mejor cobertura establecida
RFPU1.1	6.1.1	Implementar las pruebas para nuevas funcionalidades	A raiz de la implementacion de las nuevas funcionalidades se realizaran sus respectivas pruebas unitarias	Actividad posterior ID:RFP1.2	Senior Desarrollador y Tester	7	2	14	Cumplimiento de estandares y estandar ATRIP	El desarrollador logra sin errores la implementacion de las pruebas unitarias
RFC1.0	6.1.2	Disminuir la cantidad de bloques catch	Error bloqueante, no avisa al usuario.	Actividad posterior ID:RFP1.3	Senior Desarrollador	0.5	1	0.5	Cumplimiento de estandares de codificacion	El desarrollador corrige exitosamente los errores en tiempo.
RFC1.1	6.1.2	Mejorar la estandarizacion de java	Corrección del codigo que no cumpla con la estandarizacion	Actividad posterior ID:RFP1.4	Senior Desarrollador	5.5	1	5.5	Cumplimiento de estandares de codificacion	El desarrollador ejecuta la corrección con exito
RFC1.2	6.1.2	Disminuir la cantidad de metodos muertos	Corrección de metodos que no son llamados en ningún momento	Actividad posterior ID:RFP1.0	Junior Desarrollador	1	1	1	Cumplimiento de estandares de codificacion	El desarrollador ejecuta la corrección con exito
RFC1.3	6.1.2	Disminuir la cantidad de variables inutilizadas	Corrección de variables que no son llamadas en ningún momento	Actividad posterior ID:RFP1.0	Junior Desarrollador	0.4	1	0.4	Cumplimiento de estandares de codificacion	El desarrollador ejecuta la corrección con exito
RFC1.4	6.1.2	Eliminar unused import	Corrección de imports no utilizados	Actividad posterior ID:RFP1.0	Junior Desarrollador	0.2	1	0.2	Cumplimiento de estandares de codificacion	El desarrollador ejecuta la corrección con exito
RFC1.5	6.1.2	Crear clase de constantes	Se implementa clase de constantes llevando la constante a interfaces de 3.0 y 4.0	Actividad posterior ID:RFP1.0	Senior Desarrollador	0.5	1	0.5	Cumplimiento de estandares de codificacion	El desarrollador crea la clase sin ningún error
RFC1.6	6.1.2	Disminuir la complejidad cognitiva	Disminuir la complejidad cognitiva para mejorar la memorabilidad a futuro	Actividad posterior ID:RFP1.0	Senior Desarrollador, Junior Desarrollador	5	2	10	Cumplimiento de estandares de codificacion	El desarrollador Senior o Junior tienen conocimiento de reducción de la complejidad
RFC1.7	6.1.2	Implementar las funcionalidades previstas	Se agrega al programa las funcionalidades nuevas previstas	Actividad posterior ID:RFT1.0	Junior Desarrollador, Senior Desarrollador	9	2	18	Cumplimiento de estandares de codificacion	La interfaz tiene que tener los cambios realizados para poder cumplir con las necesidades
RFT1.0	7.1	Planificación y Diseño de Pruebas	Se detallaran las Pruebas a Realizar	Actividad posterior ID:RFT1.1	Tester	4	1	4	Buenos procedimientos en el diseño de pruebas	El tester diseña las pruebas en forma existente en el tiempo estimado
RFT1.1	7.2	Ejecución de Set de pruebas clases de equivalencias	Se ejecutan las pruebas de clases de equivalencia	Actividad posterior ID:RFT1.2	Tester y Junior Desarrollador	1	2	2	Buenos procedimientos en la ejecución de las pruebas	El tester ejecuta las pruebas en forma existente en el tiempo estimado
RFT1.2	7.3	Ejecución de Set de pruebas partición de equivalencias	Se ejecutan las pruebas de partición de equivalencia con sus análisis de valores límites	Actividad posterior ID:RFT1.5	Tester y Junior Desarrollador	1	2	2	Buenos procedimientos en la ejecución de las pruebas	El tester ejecuta las pruebas en forma existente en el tiempo estimado
RFT1.3	7.4	Ejecución y Set de Pruebas valores	Se ejecutan las pruebas de valores límites	Actividad posterior ID:RFT1.5	Tester y Junior Desarrollador	1.5	2	3	Buenos procedimientos en la ejecución de las pruebas	El tester ejecuta las pruebas en forma existente en el tiempo estimado
RFT1.4	7.5	Ejecución y Set de Pruebas Testing exploratorio	Se ejecutan las pruebas Testing Exploratorio	Actividad posterior ID:RFT1.5	Tester y Junior Desarrollador	1.5	2	3	Buenos procedimientos en la ejecución de las pruebas	Todas las pruebas ejecutadas completadas y ejecutadas a tiempo
RFT1.5	7.6	Informe de Pruebas	Se realiza un informe detallado de las pruebas ejecutadas	Actividad posterior ID:RBD1.0	Tester	2	1	2	Buena documentación y un reporte completo de las pruebas	Todas las pruebas ejecutadas completadas y ejecutadas a tiempo
RBD1.0	8.1	Realización de builder y informe de logs	Se realiza el proceso de build de maven y el informe correspondiente	Actividad posterior ID:RFP1.0	Senior Desarrollador	0.8	1	0.8	Informe detallado del log de maven	Todas las actividades anteriormente realizadas con exito
RIF1.0	9	Informe de Cierre	Se realiza un informe de toda la planificación del proyecto	-	Project Manager y Gerente SQA	5	2	10	Informe de la puesta en marcha de la aplicación	Todas las actividades anteriormente realizadas con exito

Figura 4.4: Lista de actividades

El procedimiento de asignación de recursos fue por las habilidades correspondientes que necesita esa actividad.

- **Criterio para la asignación de las actividades de desarrollo de diseño de interfaz:**
 - Para la actividad de desarrollo de diseño de interfaz se necesita tener conocimientos de diseño de frontends, diseño en aplicaciones de escritorio, conocedor del paquete de JavaFX para las solicitudes cambio. El junior developer tiene experiencias pequeñas en diseño de frontends, conocimiento de java interfaces.
Roles: Diseñador, Junior developer.
- **Criterio para la asignación de las actividades de desarrollo de las pruebas unitarias:**
 - Para la actividad de desarrollo de las pruebas unitarias es necesario tener conocimiento de estandarización java, diseño de pruebas ATRIP, conocimiento de dependencia externa Junit.
Roles: Senior developer, tester.
- **Criterio para la asignación de actividades de desarrollo de las implementaciones de código:**
 - Buenas prácticas de programador, realización de patrones de diseño, estandarización de java, ser buen Clean Coder.
Roles: Tester, Junior developer, Senior developer, Tester.
- **Criterio para la asignación de actividades de desarrollo de testing:**
 - Para la realización del desarrollo de testing, se tiene que tener conocimiento de pruebas exploratorias, clases de equivalencia, clases de valores límites, y análisis de valores límites y buena documentación de reporte de las pruebas y defectos encontrados.
Roles: Tester, Junior developer, Senior developer.
- **Criterio para la asignación de Build:**
 - Para la realización del build es necesario tener a cargo alguien que tenga conocimiento de todo el proyecto como lo es el gerente de SQA, además de tener conocimiento de archivos XML, las distintas versiones y las dependencias que tiene en las bibliotecas externas como JavaFX o Junit.
Roles: Senior Developer, Gerente SQA
- **Criterio para la actividad de informe de cierre y puesta en marcha:**
 - Para la realización del informe que recoge el cierre de los elementos desarrollados durante la vida del Proyecto se asigna al Project Manager con el objetivo de evaluar el resultado de los trabajos y resumir todo lo sucedido en el proyecto. Brindando así la información de si el proyecto obtuvo o no los resultados previstos, en caso negativo, también incluye un análisis de las razones de ello.

4.5. Especificación del ciclo de vida

Para tener un plan de aseguramiento y realizar los cambios definidos del proyecto y poder realizar los objetivos SMART, nos basamos en un marco de referencia básico para dirigir el proyecto. Tomamos como marco un ciclo de vida predictivo. Estos son aquellos en los cuales el alcance del proyecto, el tiempo y costo requeridos para lograr dicho alcance, se determinan en las fases tempranas del ciclo de vida del proyecto. Nosotros optamos por este ciclo de vida por los siguientes motivos:

- El producto a mejorar se comprendió bien, no fue necesario realizar reuniones exhaustivas ni análisis para comprender el programa.
- Existió de antemano una base práctica significativa en la industria.
- Los objetivos especificados de calidad son claros, trazables y detallados.
- No existe la necesidad de hacer prototipos, ni cambios incrementales, ni tampoco pequeñas entregas.
- Los cambios del alcance se realizan cuidadosamente, el cronograma, el presupuesto, conformación del equipo del proyecto y los riesgos tomados.
- El cliente no le sirve una entrega parcial o con funcionalidades parcial de ese producto.

El ciclo de vida en cascada fue para el equipo de mantenimiento una mejor opción como baseline del proyecto.

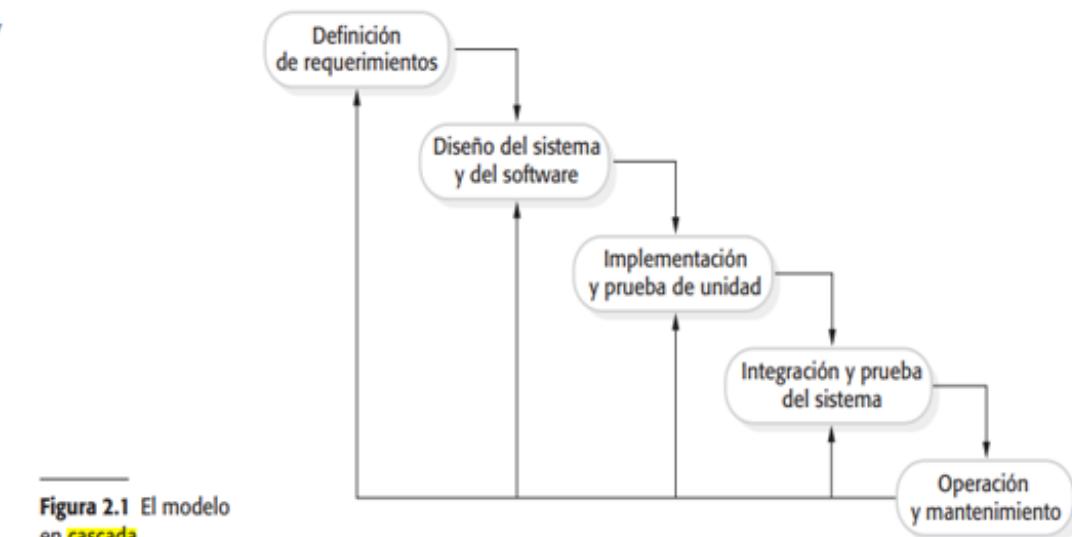


Figura 4.5: Ciclo de vida en cascada (WaterFall)

4.6. Cronograma de trabajo

Cada personal trabaja 3 horas por día, además las actividades que no pueden arrancar en paralelo sucede porque no pueden arrancar hasta que termine la anterior y además pueden no tener los recursos suficientes (la cantidad de personal) para realizar esa tarea, ejemplo: Un senior developer puede estar encargado de una tarea entonces no puede realizar otra tarea hasta que finalice esa.

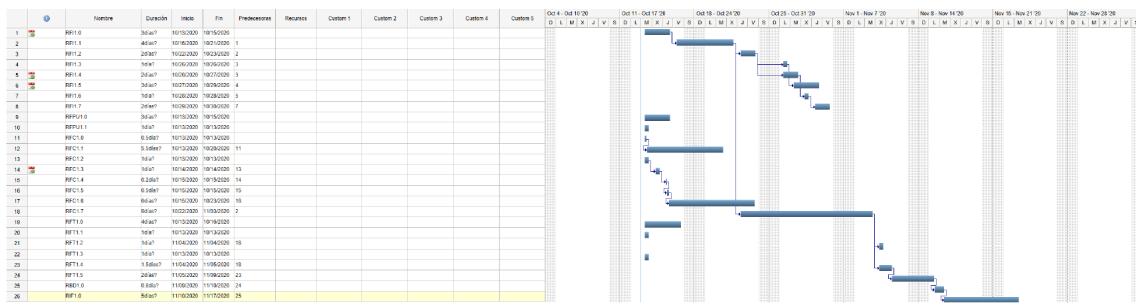


Figura 4.6: Diagrama de Gantt

Camino critico: Se trata de un algoritmo que calcula el orden y los plazos en la planificación de proyectos. Conocer la ruta crítica es clave para todo Project Manager, quien utiliza herramientas para conocer cuáles serán las actividades o tareas que debe realizar su equipo para que nada se retrase y se cumpla con los plazos estimados.

Además, cuando aplicamos el camino crítico en la planificación de proyectos estaremos ayudando a nuestro equipo a trabajar de una manera más lógica. Si aprovechamos todos los recursos y tenemos en cuenta gráficas de consumo que nos informen del esfuerzo aplicado en cada actividad, planificaremos de una manera más eficiente y próxima a la realidad.

Y es que uno de los mayores errores de muchos Project Managers es que su planificación no corresponde, en muchos casos, con la realidad de situaciones de riesgos e incertidumbres.

4.6.1. Relación con el modelo de ciclo de vida especificado

Todas las actividades que tienen el tiempo estimado, como se menciono anteriormente, tuvieron en cuenta los recursos asignados y la duración estimada. Como se dijo un personal no puede comenzar con otra tarea si tiene una tarea asignada pero puede comenzarla si tiene otro personal asignado.

Al ser un ciclo de vida predictivo, el alcance del proyecto, el tiempo y costo requeridos para lograr dicho alcance, se determinaron en las fases tempranas del ciclo de vida del proyecto.

Como se puede visualizar en el Gantt, la etapa de ejecución es la que demora mas y en gran medida afecta a los costos. En el Gantt las etapas de testing y builder son un claro ejemplo de que queda en evidencia que son caminos críticos y actividades criticas de alta prioridad. Esta etapa seria la integración y prueba del sistema. Las

etapas de operaciones y mantenimiento: Son de testing:pruebas con usuarios, caja negra y además toda la puesta en marcha, informes y logs del build del proyecto.

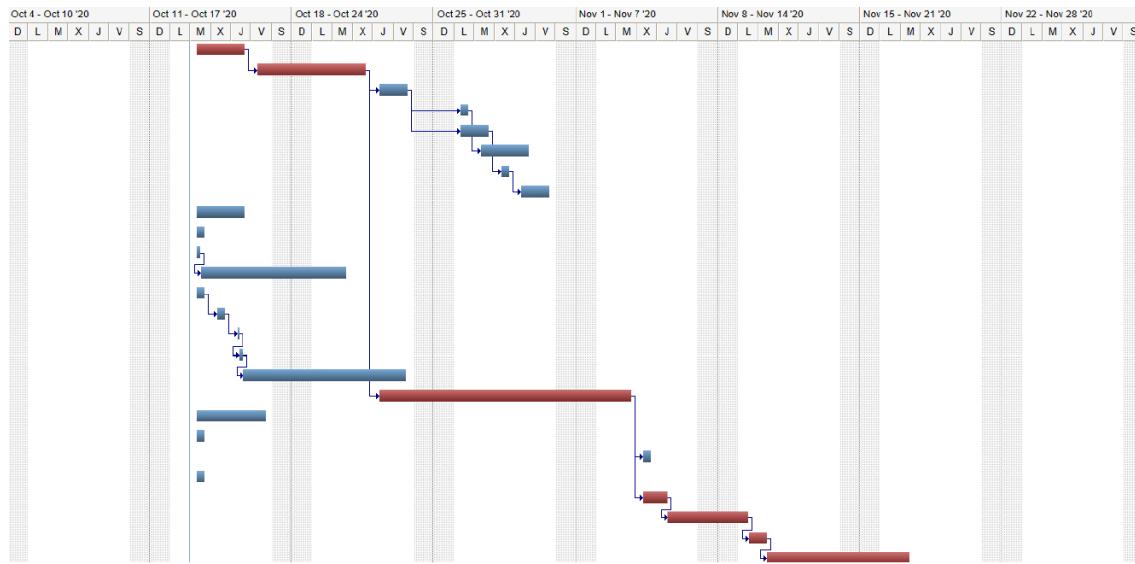


Figura 4.7: Diagrama Gantt con caminos críticos

4.7. Estimación de costos

ID	ID.Unidad de Trabajo	Nombre Actividad	Tipos de Habilidades Requeridas	Duración (Días)	Cantidad Persona	Estimacion Salario por Hora(\$)	Costo(\$)	
RFL1.0	5.1	Realización Set de Pruebas de la interfaz	Desarrollador	3*3h	1	9	19	171
RFL1.1	5.2	Implementación de los cambios en base a las Heurísticas de Nielsen	Desarrollador	4*3h	1	12	29	228
RFL1.2	5.2	Implementación de prueba visual	Desarrollador	2*3h	1	6	19	114
RFL1.3	5.2	Implementación Interfaz Principal Login	Desarrollador,Junior Desarrollador	1*3h	2	6	32	192
RFL1.4	5.2	Implementación del cambio interfaz Profesional	Desarrollador,Junior Desarrollador	3*3h	2	18	32	576
RFL1.5	5.2	Implementar funcionalidad de listas Agregar Comida a la dieta.	Desarrollador,Junior Desarrollador	2*3h	2	12	32	384
RFL1.6	5.2	Implementación de la funcionalidad de listas.	Desarrollador,Junior Desarrollador	1*3h	2	6	32	192
RFL1.7	5.2	Implementación de la funcionalidad de listas.	Desarrollador	2*3h	1	6	29	174
RPU1.0	6.1.1	Mejorar la cobertura de código.	Senior Developer	3*3h	1	9	18	162
RPU1.1	6.1.1	Implementar las pruebas para nuevas funcionalidades	Senior Developer y Tester	7*3h	2	42	32	1344
RFL1.0	6.1.2	Disminuir la cantidad de bloques catch mal y no implementados	Senior Developer	0,5*3h	1	1,5	18	27
RFL1.1	6.1.2	Mejora de la estandarización de java	Senior Developer	5,5*3h	1	16,5	18	297
RFL1.2	6.1.2	Disminuir la cantidad de métodos muertos	Junior Developer	1*3h	1	1,3	13	16,9
RFL1.3	6.1.2	Disminuir la cantidad de variables innecesarias	Junior Developer	0,5*3h	1	1,5	13	19,5
RFL1.4	6.1.2	Eliminación de imports	Junior Developer	0,2*3h	1	0,6	13	7,8
RFL1.5	6.1.2	Crear clase de constantes	Junior Developer	0,5*3h	1	1,5	13	19,5
RFL1.6	6.1.2	Disminuir la complejidad cognitiva	Senior Developer,Junior Developer	5*3h	2	30	31	930
RFL1.7	6.1.2	Implementar las funcionalidades previstas	Senior Developer,Junior Developer	9*3h	2	54	31	1674
RFT1.0	7.1	Planeación y Diseño de Pruebas	Tester	4*3h	1	12	14	168
RFT1.1	7.2	Ejecución y Set de Pruebas de equivalencias	Tester y Junior Developer	1*3h	2	6	27	162
RFT1.2	7.2	Ejecución de Set de pruebas parciales de equivalencias	Tester y Junior Developer	1*3h	2	6	27	162
RFT1.3	7.4	Ejecución y Set de Pruebas valores	Tester y Junior Developer	1,5*3h	2	9	27	243
RFT1.4	7.5	Ejecución y Set de Pruebas Testing exploratorio	Tester y Junior Developer	1,5*3h	2	9	27	243
RFT1.5	7.6	Informe de Pruebas	Tester	2*3h	1	6	14	84
RBD1.0	8.1	Realización de builder e Informe de logs	Senior Developer,Gerente SQA	0,8*3h	2	4,8	40	192
RIF1.0	9	Informe de Cierre	Project Manager y Gerente SQA	5*3h	2	30	51	1530

Figura 4.8: Tabla de estimación de costos

Se realizó el presupuesto de cada actividad, calculando que cada personal trabaja 3h por día, teniendo en cuenta la duración de la actividad y los recursos asignados (cantidad de personas) realizamos el producto que como resultado nos dio la estimación. Del mismo modo realizamos la estimación de costo como consecuencia del resultado del producto de salario*estimación dando como resultado el costo total de la actividad. El precio total de las actividades es: 9191,8\$.

Realizamos un flujo de caja que es la acumulación neta de activos líquidos en el periodo de 6 meses estipulados en el proyecto en total por lo tanto, constituye un indicador importante de la liquidez de nuestro equipo de SQA.

A continuación se mostrara el flujo de caja realizado expresado en dólares:

Alimentacion Saludable						
	MES					
	1	2	3	4	5	6
Sueldos	4.260	4.260	4.260	4.260	4.260	4.260
Horas extras	0	1.300	500	2.500	300	5.000
Alquiler	1.000	1.000	1.000	1.000	1.000	1.000
Servicio De Internet	58	58	58	58	58	58
Reserva Cont.						3.200
Totales	5.318	6.618	5.818	7.818	5.618	44.708
Porcentaje	11,9%	14,8%	13,0%	17,5%	12,6%	23,1%
Presup.Acum.	5.318	11.936	17.754	25.572	31.190	41.508
	11,9%	26,7%	39,7%	57,2%	69,8%	92,8%
Reserva C.G.						7.400
Ganancia						13.412
Precio						65.520
						30%
Plan de pago	20%	5%	5%	5%	5%	60%
Ingresos	13.104	3.276	3.276	3.276	3.276	39.312
Acumulado	13.104	16.380	19.656	22.932	26.208	65.520
Caja	7.786	4.444	1.902	-2.640	-4.982	24.012

Figura 4.9: Flujo de caja

Como servicio de internet fue contratado el paquete premium con velocidad:

- Hasta 240 Mbps de bajada.
- Hasta 24 Mbps de subida.
- Costo: 58\$.

El plan de pagos en el primer mes se nos dará el 20 % para las distintas instalaciones de las distintas terminales del equipo de trabajo, la contratación de servicio de internet y el alquiler como el traslado de los distintos recursos. A los siguientes meses se nos dará el 5 %, al finalizar se nos da el 60 % del costo total del proyecto.

Costo del alquiler= 1000\$ por mes

El sueldo por mes equivale a= 4260\$

El sueldo total= 25580\$

Diagrama de Flujo de Caja:

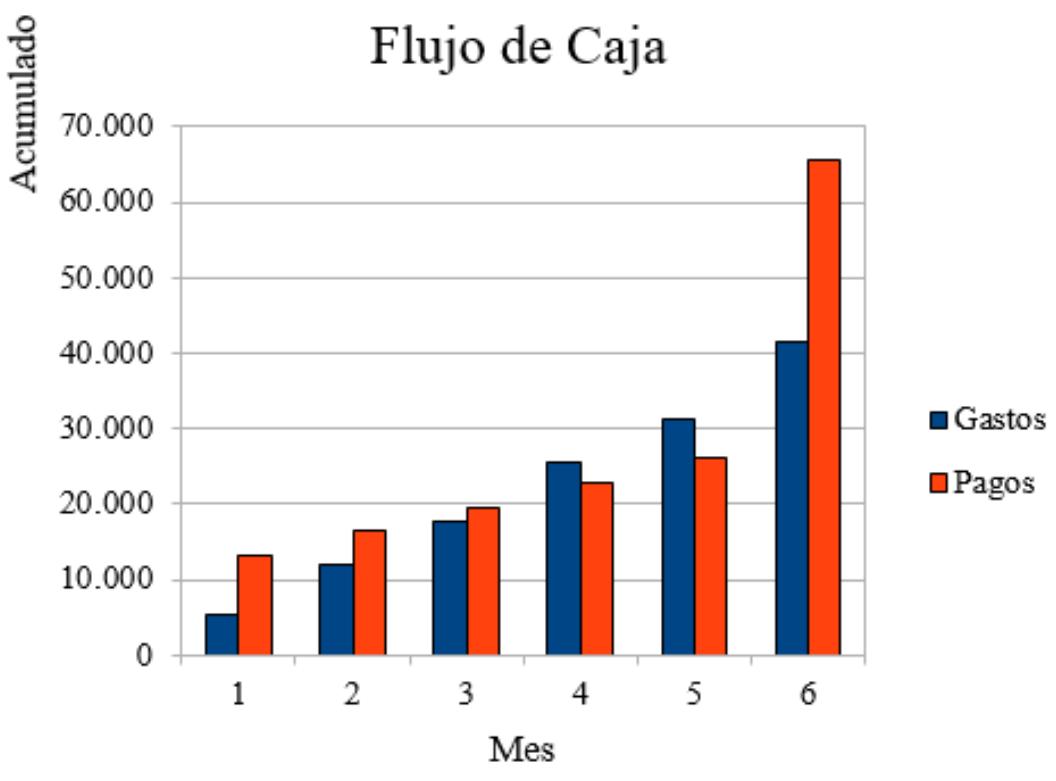


Figura 4.10: Diagrama de Flujo de Caja Gastos/Pagos

A continuación se presenta el diagrama del presupuesto acumulado:

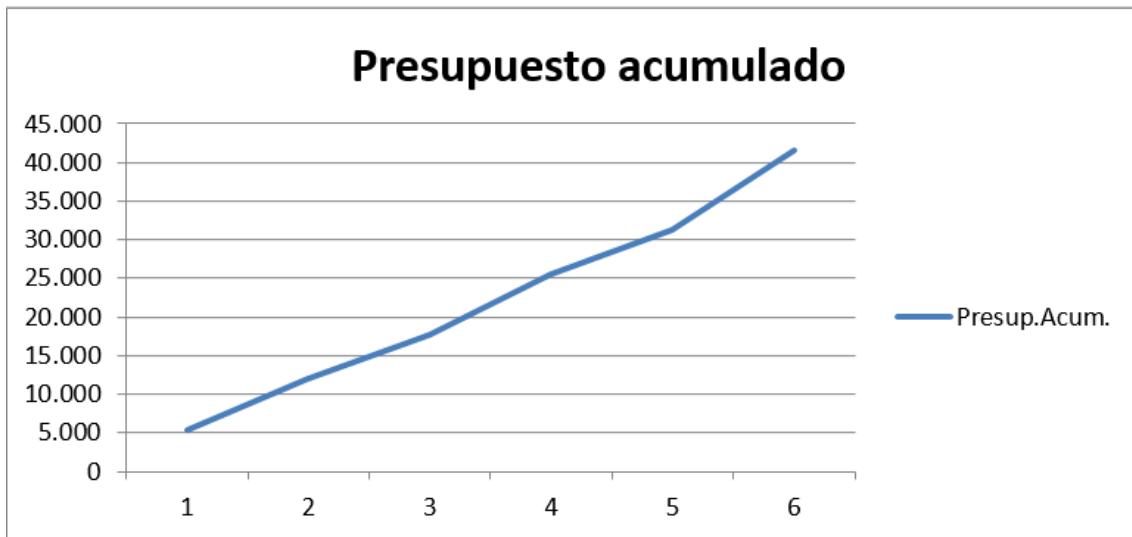


Figura 4.11: Diagrama de Presupuesto acumulado

4.8. Plan de calidad

En esta sección el personal SQA especifica que procedimientos y recursos asociados deben aplicarse, quien debe aplicarlos y cuando deben aplicarse a un proyecto, producto, proceso o contrato específico. Cada proyecto de desarrollo y mantenimiento debe tener un plan de calidad especificando sus metas, tareas de SQA a desarrollar, los estándares a utilizar, los procedimientos y las definiciones de los roles y sus responsabilidades. Para este proyecto se comprueba con anticipación si se cumplen todos los objetivos iniciales o si no se cumplen.

▪ Calidad del código

1. Para asegurar la calidad del código los desarrolladores encargados de las tareas del desarrollo del código y refactorización de código (ya sean senior o junior) deben acoplarse a las buenas prácticas de programador, estandarización del código, reglas de Clean Code, así como también para la implementación de nuevas funcionalidades. Se cumple todos los objetivos si sigue todas las prácticas necesarias y la correspondiente estandarización de código para el lenguaje JAVA.
2. Para realizar el aseguramiento e la calidad debemos definirnos al menos 3 herramienta de analizadores estáticos((e.g. Easy PMD, JavaHints, SonarQube). Para SonarQube se eligió la cantidad de bugs para cumplir totalmente la calidad y los objetivos iniciales. Para una cantidad de 100 bugs con ninguno de ellos critico se le asigna un aceptable, para una cantidad mayor de 100 con ningún bug critico debe mejorar y gestionar, y para mayor de 100 bugs con alguna criticidad calidad insatisfecha. En cambio para Easy PMD los errores críticos todo en 0, eliminar variables no usadas 100 % es calidad buena, el 90 % se considera aceptable menor a 90 % calidad insatisfecha.

▪ Calidad de las pruebas unitarias

1. Para asegurar la calidad de las pruebas unitarias se deberá diseñar las pruebas para las implementaciones nuevas, refactorizar las existentes y rediseñarlas de ser necesario siguiendo el estándar ATRIP.
2. Se deberá utilizar al menos una herramienta de cobertura que garantice una cobertura de al menos igual a la previa (e.g.JaCoCo). Se tomara como aceptable un porcentaje mayor a 95 %, a mejorar un porcentaje de 90 % a 95 % , y una cobertura menor a 90 % critica y calidad insatisfecha.

▪ Calidad de los casos de uso y de prueba

1. Se deberán modificar los casos de uso y de prueba para que coincidan con el flujo real de la aplicación y para los cambios realizados de las implementaciones nuevas se considerara solamente que estén todos los casos de uso y de prueba correcto.

- **Calidad de la interfaz**
- Los diseñadores gráficos deberán seguir las heurísticas de Nielsen, consideramos como resultado aceptable, que el sistema cumpla con al menos 8 de estas heurísticas, e inaceptable y calidad insatisfecha de 7 o menos.
- Se deberá validar la usabilidad de la interfaz liberando una versión para que los usuarios puedan realizar pruebas y walkthrough y puntuar dentro del rango 1-10 (siendo 1 el la peor calidad y 10 el mejor). El criterio de aceptación/atributo de calidad para la interfaz sería una retroalimentación de al menos 7.5 puntos por parte del usuario. Se considera el sistema como aceptable cuando cumple todos los objetivos pedidos, un buena correctitud del funcionamiento, puede mejorar y gestionar de cerca aquellos cambios solicitados solamente por el cliente con objetivos básicos de funcionamiento pero no del todo estandarizado y correcto. Y como de mala calidad aquel que no cumpla los cambios propuestos, principios de calidad del software entre otros.

4.9. Plan de métricas

Para el plan de métricas establecemos primeros las metas:

1. Desarrollamos un conjunto de metas corporativas, de la división y del proyecto de negocio que estén asociados a un conjunto de medidas de productividad y calidad.
2. Generamos preguntas: Generamos las preguntas que definen objetivos de la manera más completa y cuantificable posible.
3. Especificamos las medidas: Especificar las medidas necesarias a ser recolectadas para contestar las preguntas y seguir la evolución del proceso y producto con respecto a las metas.
4. Preparamos una recolección de datos: Desarrollar mecanismos para la recolección de datos.
5. Recolectar, validar y analizar los datos para la toma de decisiones: Recoger, validar y analizar los datos en tiempo real, para proporcionar la realimentación de proyectos en una acción correctiva.
6. Analizamos los datos para el logro de los objetivos y el aprendizaje: Analizamos los datos una vez alcanzado una meta para determinar el grado de conformidad y hacer las recomendaciones para mejoras futuras.

4.9.1. Desarrollo de conjunto de metas

1. Llegar al nivel de cobertura del 95 %.
2. Reducir las complejidad cognitiva del código.

3. Llegar a corregir los todos bugs del SonarQube y completar los 808 Code Smells en los tiempo estimados.
4. Corregir los incumplimientos de las heurísticas de Nielsen.
5. Realizar los cambios de la interfaz con buena calidad de usabilidad.
6. Llegar a tiempo a realizar las pruebas.
7. Realizar un correcto desarrollo de build.
8. Realizar todas las implementaciones con la calidad adecuada.
9. Puesta en marcha correcta en las distintas plataformas.

4.9.2. Desarrollo de generación de preguntas, medición e implementación

Cada pregunta se relaciona con el objetivo SMART definidos en el orden que se encuentra en la sección 4.2 definición objetivos del proyecto. Por lo tanto el primer ítem de este apartado corresponderá al primer objetivo SMART y así sucesivamente para las siguientes generación de preguntas.

- Pregunta: ¿Existe un nivel de cobertura adecuado para tener una certificación de calidad de código?
Medición: Porcentaje de nivel de cobertura de las pruebas unitarias.
Herramienta y/o mecanismo: Medir mediante la herramienta JaCoCo Coverage.
- Pregunta: ¿La complejidad cognitiva es la adecuada para la mantenibilidad a futuro de la aplicación?
Medición: Porcentaje de nivel de complejidad cognitiva.
Implementación y/o mecanismo: Analizador estático de código SonarQube.
- ¿No existen imports de paquetes no usados en el código?
Medición: Cantidad numérica de paquetes no usados.
Implementación y/o mecanismo: Analizador estático de código SonarQube.
- ¿Los casos de prueba coinciden perfectamente con el flujo real de la aplicación?
Medición: Cantidad numérica de pruebas coincidentes.
Implementación y/o mecanismo: Análisis de documentación.
- ¿Las constantes están definidas correctamente y eficientemente?
Medición:
Implementación y/o mecanismo:
- ¿Las pruebas unitarias están diseñadas y plasmadas correctamente, con su estándar de codificación y su procedimiento ATRIP correctamente?
Medición: Cantidad de clases Test con mala implementaciones.
Implementación: Análisis ATRIP y análisis de estándar de Java Codificación.

- ¿La interfaz de usuario Promueve correctamente las heurísticas de Nielsen?
Medición: Cantidad numérica de heurísticas no correctamente implementadas.
Implementación y/o mecanismo: Análisis de prueba de usabilidad usando heurísticas de Nielsen.
- ¿El código tiene un buen uso definición de variables?
Medición: Cantidad numérica de variables no correctamente inicializadas.
Implementación y/o mecanismo: Utilizando la herramienta JavaHints.
- ¿Existe un buen manejo de constructores y destructores en el código?
Medición: Cantidad de errores no críticos de constructores por clase.
Implementación y/o mecanismo: Analizador estático de código SonarQube.

4.10. Plan de RRHH

El plan de RRHH (recursos humanos) es el proceso de identificar y documentar los roles dentro de un proyecto, las responsabilidades, las habilidades requeridas y las relaciones de comunicación, así como también de como de crear un plan para la gestión personal.

El proceso de estimación de salarios consiste en desarrollar una estimación aproximada de los recursos monetarios necesarios para completar las actividades del proyecto.

La técnica utilizados fue: Estimación por tres valores (cE).

- Más probable (cM)
- Optimista (cO)
- Pesimista (cP)

La distribución que se realizó fue la triangular: $cE = (cO + cM + cP)/3$. Calculo de las horas estimadas por ROL:

Tabla estimada por ROL			
ROL	MAS PROBABLE (CM)	OPTIMISTA (CO)	PESIMISTA (CP)
Project Manager	30\$	20\$	38\$
TOTAL			
29\$			

Tabla 4.18: Tabla de estimación triangular por Project Manager

Tabla estimada por ROL			
ROL	MAS PROBABLE (CM)	OPTIMISTA (CO)	PESIMISTA (CP)
Gerente SQA	20\$	12\$	35\$
TOTAL			
22\$			

Tabla 4.19: Tabla de estimación triangular por Gerente SQA

Tabla estimada por ROL			
ROL	MAS PROBABLE (CM)	OPTIMISTA (CO)	PESIMISTA (CP)
Senior developer	15\$	10\$	30\$
TOTAL			
18\$			

Tabla 4.20: Tabla de estimación triangular por Senior developer

Tabla estimada por ROL			
ROL	MAS PROBABLE (CM)	OPTIMISTA (CO)	PESIMISTA (CP)
Junior developer	10\$	5\$	24\$
TOTAL			
13\$			

Tabla 4.21: Tabla de estimación triangular por Junior developer

Tabla estimada por ROL			
ROL	MAS PROBABLE (CM)	OPTIMISTA (CO)	PESIMISTA (CP)
Diseñador gráfico	15\$	9\$	33\$
TOTAL			
19\$			

Tabla 4.22: Tabla de estimación triangular por Diseñador gráfico

Tabla estimada por ROL			
ROL	MAS PROBABLE (CM)	OPTIMISTA (CO)	PESIMISTA (CP)
Tester	12\$	5\$	25\$
TOTAL			
14\$			

Tabla 4.23: Tabla de estimación triangular por Tester

La planificación de los recursos humanos es importante porque se utiliza para determinar e identificar aquellos recursos humanos, valga la redundancia, que posean las habilidades requeridas para el éxito del proyecto. Existen diversos formatos para documentar los roles y las responsabilidades de los miembros del equipo: jerárquico, matricial y tipo texto. Por eso se optó por una técnica la cual nos proporciona una correcta planificación de RRHH, se eligió el estilo de diagrama matricial, en particular, una matriz de asignación de responsabilidades (RAM). A continuación se muestran las asignaciones de responsabilidad RAM.

ROL	RH designado	Cantidad de RRHH designados	Salario por hora estimado
Project Manager	Ignacio Olivera	1	29\$
Gerente SQA	Tomas Dilema	1	22\$
Senior developer	Bruno Fernandez	1	18\$
Junior developer	Ignacio Olivera Tomas Dilema	2	13\$
Diseñador gráfico	Bruno Fernandez	1	19\$
Tester	Ignacio Olivera Tomas Dilema	2	14\$

Tabla 4.24: Tabla RAM, división por ROL

4.11. Plan de gestión 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 un 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).

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

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

Id	Fecha de Ingreso	Componente	Especificación	Consecuencia	Tipo	Probabilidad	Impacto	Magnitud	Ubicación de Impacto
1	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
2	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
3	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
4	11/10/2020	Gestión	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
5	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	3	Diagrama de Cronogramas de Trabajo
6	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
7	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
8	11/10/2020	Gestión	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
9	11/10/2020	Gestión	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
10	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
11	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
12	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
13	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
14	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
15	11/10/2020	Proyecto	Los Recursos asignado no están 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
16	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	0.6	3	1.8	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 4.12: Tabla de riesgos.

4.11.1. 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 Gantt necesitará 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 será necesario modificar la de actividades para agregar la tarea nueva y su estimación correspondiente de esfuerzo. Esto causará 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 habrá que readjustar el presupuesto inicial ya que ahora tenemos una nueva tarea que implica más horas de trabajo y más 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 los 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.

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>