

Universidad ORT Uruguay

Facultad de Ingeniería

# Ingeniería de Software 1

## Obligatorio 2

Anibal André Hernández Depaz

193234

Entregado como requisito de la materia Ingeniería de  
Software 1

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

Profesor: Martín Solari  
2017

# Declaraciones de autoría

Yo, André Hernández , declaro que el trabajo que se presenta en esa obra es de mi propia mano. Puedo asegurar que:

- La obra fue producida en su totalidad mientras realizaba Ingeniería de Software 1;
- Cuando fue consultado el trabajo publicado por otros, se atribuyó con claridad;
- Cuando se citó obras de otros, se indicó las fuentes. Con excepción de estas citas, la obra es enteramente propia;
- En la obra, se ha acusado recibo de las ayudas recibidas;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, se ha explicado claramente que fue contribuido por otros, y que fue contribuido por quien escribe;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.

## **Resumen**

El proyecto esta orientado a el desarrollo de una aplicación, utilizando buenas prácticas tecnológicas y de gestión de la ingeniería de software, obteniendo un software de calidad. El propósito de la aplicación es permitir a los usuarios seguir un esquema de alimentación saludable avalado por profesionales de la alimentación. Se comenzó analizando todos los requerimientos funcionales y no funcionales, analizando y luego creado repositorios para un correcto versionado de la futura aplicación, estudiando los estándares correctos de codificación y usabilidad para luego comenzar a implementar el código de la aplicación. Se obtuvieron resultados exitosos, se pudo codificar la aplicación, reflejando requerimientos funcionales iniciales, tratando de obtener lo mas cercano a un software de calidad y permitiendo a los usuarios seguir un esquema de alimentación saludable aprobado por profesionales, como se planteo como propósito inicialmente. Finalmente, se realizan evaluaciones sobre la aplicación desarrollada, se evaluó la usabilidad del sistema y se realizaron pruebas funcionales conjunto a reporte de defectos explicando las técnicas utilizadas y validando procedimientos contra la aplicación.

# Índice general

<b>1. Versionado</b>	<b>2</b>
1.1. Repositorio utilizado . . . . .	2
1.2. Criterios de versionado . . . . .	3
1.3. Resumen del log de versiones . . . . .	3
<b>2. Codificación</b>	<b>5</b>
2.1. Estándar de codificación . . . . .	5
2.2. Pruebas unitarias . . . . .	10
2.3. Análisis de código . . . . .	11
<b>3. Interfaz de usuario y usabilidad</b>	<b>15</b>
3.1. Criterios de interfaz de usuario . . . . .	15
3.2. Evaluación de usabilidad . . . . .	21
<b>4. Pruebas funcionales</b>	<b>24</b>
4.1. Técnicas de prueba aplicadas . . . . .	24
4.2. Casos de prueba . . . . .	25
4.3. Sesiones de ejecución de pruebas . . . . .	34
<b>5. Reporte de defectos</b>	<b>42</b>
5.1. Definición de categorías de defectos . . . . .	42
5.2. Defectos encontrados por iteración . . . . .	43
5.3. Estado de calidad global . . . . .	44
<b>6. Reflexión</b>	<b>45</b>

# 1. Versionado

## 1.1. Repositorio utilizado

Se utiliza la herramienta GIT para el versionado de software; GIT es un software de control de versiones, gestionando los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo es decir a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración.

Por otra parte se utiliza GitHub, el cual nos permitirá administrar el versionado del proyecto desde la nube. Para la utilización de esta herramienta se creó una cuenta en <https://github.com/> creando automáticamente un perfil para la posterior creación de nuestro repositorio: projectAlimentacionSaludable. Cabe destacar que no se instaló ningún software extra para el manejo del repositorio, los procedimientos se realizaron todos por consola GIT.

A continuación se agrega al documento link al repositorio online utilizado, en donde se agrega al proyecto el usuario: martinsolari@gmail.com docente del curso para poder acceder al mismo.

### Link repositorio utilizado:

<https://github.com/AndreHernandezIngSoftware1/projectAlimentacionSaludable.git>

Por otra parte los Elementos de la Configuración de Software (ECS) incluidos en el repositorio serán los siguientes:

- **Programa:** Donde se podrá encontrar el código de fuente de todas las clases del sistema Aplicación Saludable.
- **Documentación:** Documentación de Ingeniería de Software realizada en LATEX en el sitio web: <https://www.overleaf.com>, con el contenido del desarrollo del sistema, fundamentando el buen uso de prácticas tecnológicas y de gestión de la ingeniería de software.

## 1.2. Criterios de versionado

Con respecto a los Elementos de la Configuración de Software (ECS) incluidos en el repositorio, tendremos dos ramas:

- **Develop**
- **Master**

En la rama **Develop** se encuentran todo el versionado de nuestro proyecto, realizado commits indicados con comentarios nemotécnicos para saber de qué se trata el cambio o avance en el proyecto. Las versiones que se encuentran en la rama **Develop** son versiones que se consideran estables para el sistema, pero no listas para pasar a producción o en este caso rama **Master**, ya que aun el sistema no cumple con normas de calidad, estándares de códigos, realización de pruebas funcionales y unitarias.

En la rama **Master** es la rama de producción, es decir, cuando se considere que el software aprueba pruebas unitarias, respeta estándares de codificación y se tiene un código robusto, es hora de pasar nuestro sistema a la rama **Master**, esta misma tendrá la versión producción del sistema, donde se podrá acceder a los distintos códigos de fuentes de las clases del sistema, el .JAR del sistema y además el documento de Ingeniería de Software.

## 1.3. Resumen del log de versiones

En esta sección del obligatorio se deja evidencia de la evolución del proyecto en el tiempo y del trabajo realizado:

```
Adrian@Adrian-PC MINGW64 ~/Desktop/projectAplicacionSaludable (develop)
$ git log
commit 96122e88983419345e2a3689c0665bf16a4989eb (HEAD -> develop, origin/develop)
Author: André <andrehernandez1994@gmail.com>
Date:   Wed Nov 22 15:01:48 2017 -0200

    Cambios en REGISTRO DE USUARIO y REGISTRO PROFESIONAL respecto al ingreso de foto

commit cb635111bc3229f9b57396c5b67e16c7f3e5f5fc
Author: André <andrehernandez1994@gmail.com>
Date:   Wed Nov 22 14:58:35 2017 -0200

    Se continua avanzando con pruebas JUnit en las clases del Dominio del sistema

commit 42d750c195a9651b7c2a914902406dfd4747a627
Author: André <andrehernandez1994@gmail.com>
Date:   Tue Nov 21 15:16:45 2017 -0200

    Se comienza a trabajar con JUnit - pruebas unitarias
```

Figura 1.1: Resumen de Log de versiones - Rama Develop - desde consola GIT

Se realizaron un total de 57 Commits en la rama Develop, entre las fechas 04/11/2017 - 22/11/2017. Cabe destacar que por seguridad se realizo un commit por cada avance que se considera importante con respecto a las funcionalidad del sistema.

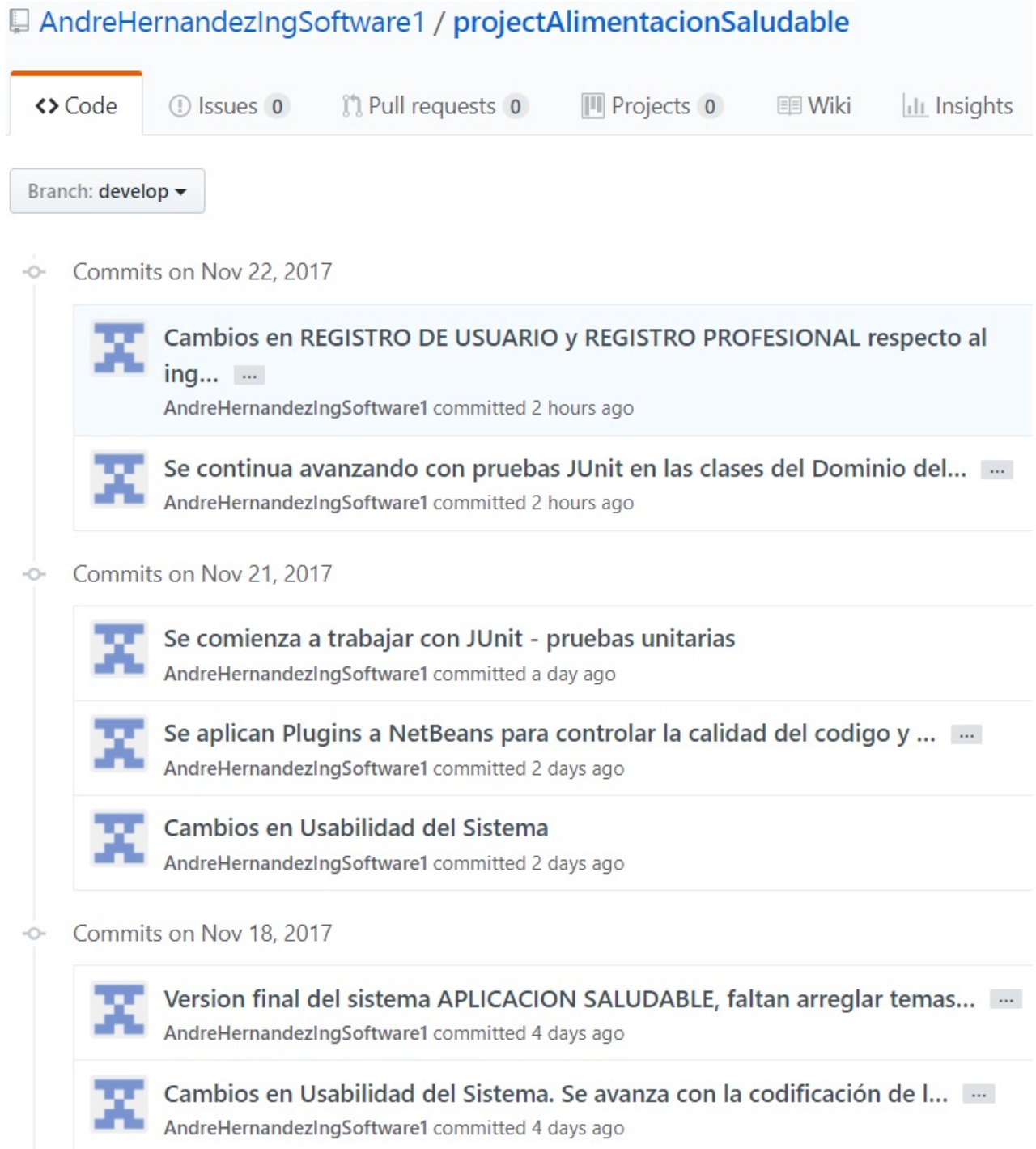


Figura 1.2: Resumen de Log de versiones - Rama Develop - desde GitHub

## 2. Codificación

### 2.1. Estándar de codificación

La forma de escribir código es propia de cada programador y completamente diferente a la forma de cualquier otro. De la forma de escribir usada depende la facilidad para entender y comprender a fondo el código; para definir esta forma de escribir código es aconsejable seguir un estándar de codificación. Los estándares de codificación son parte de las llamadas buenas practicas o mejores practicas, estas son un conjunto no formal de reglas, que han ido surgiendo en las distintas comunidades de desarrolladores con el paso del tiempo. [1]

Por otra parte, en el ámbito laboral informático, se utiliza la forma de trabajo en equipo, por lo que el estándar de codificación que se debe adoptar cuando formamos parte de un equipo de trabajo debe seguir determinadas características para que cualquier integrante del equipo pueda comprender el código y si es necesario realizar algún cambio, mejorar o depuración no pierda tiempo en comprender un código que esta mal indentado, sin buena legibilidad, sin adecuados comentarios, sin nombre de métodos o variables nemotecnias, entre otras normas de codificación que definen un estándar de codificación.

El correcto uso de un estándar de codificación a lo largo del desarrollo de un sistema informático nos asegura que nuestro código será de buena calidad. En el código de nuestro sistema Aplicación Saludable, se trata de optar por las buenas practicas de codificación, teniendo como fin la robustez y solidez del código, teniendo en cuenta el documento de ORACLE, Java Code Conventions [2].

A continuación se describen **buenas practicas** aplicadas al sistema Aplicacion Saludable:

- **Comentarios de comienzo:**

Java Code Conventions indica que todos los ficheros fuente deben comenzar con un comentario en el que se lista el nombre de la clase. Al comienzo de cada clase de todos los paquetes del sistema Aplicación Saludable, se indica en primer lugar el autor de la clase, seguido por una descripción de la metería y el requisito de entrega, luego el nombre de la clase y finalmente se describe la ultima fecha de edición de la clases correspondiente.



En el siguiente ejemplo mostramos los comentarios de comienzo de la clase Alimento:

```
// @author André Hernández — Numero de Estudiante: 193234
// SEGUNDO OBLIGATORIO — Ingenieria de Software I
// CALSE ALIMENTO
// 17/11/2017
```

- **Sentencias package e import:**

Java Code Conventions indica que la primera línea no-comentario de los ficheros fuente Java es la sentencia package. Después de esta, pueden seguir varias sentencias import. Las clases del sistema Aplicación Saludable con respecto a sentencias package e import, en el siguiente caso tenemos como package: Dominio y un import de: ArrayList:

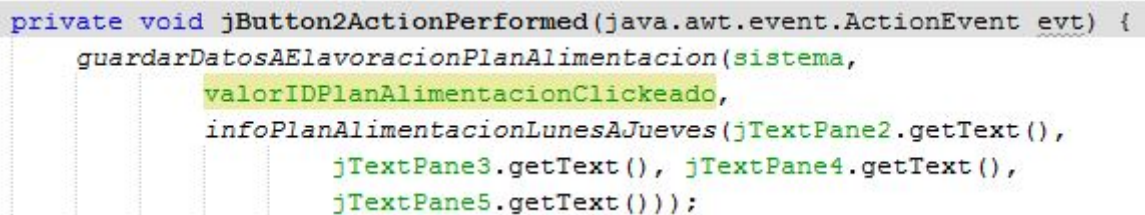
```
package Dominio;
import java.util.ArrayList;
```

- **Longitud de la línea:**

Java Code Conventions indica que se deben evitar las líneas de más de 80 caracteres, ya que no son manejadas bien por muchas terminales y herramientas.

- **Rompiendo líneas:**

Java Code Conventions indica, cuando una expresión no entre en una línea, romperla de acuerdo con estos principios: romper después de una coma, romper antes de un operador, este lo vemos reflejado en la siguiente imagen (Figura 2.1) tomada del método guardarDatosAElaboracionPlanAlimentacion() del código del sistema Aplicación Saludable:



```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    guardarDatosAElaboracionPlanAlimentacion(sistema,
        valorIDPlanAlimentacionClickeado,
        infoPlanAlimentacionLunesAJueves(jTextPane2.getText(),
            jTextPane3.getText(), jTextPane4.getText(),
            jTextPane5.getText()));
}
```

Figura 2.1: Cumplimiento Rompiendo líneas

Java Code Conventions indica también que se debe alinear la nueva línea con el comienzo de la expresión al mismo nivel de la línea, para para este proyecto se utiliza la indentación por defecto del IDE NetBeans version 8.2.

- **Formatos de los comentarios de implementación:**

Java Code Conventions indica que los programas pueden tener cuatro estilos de comentarios de implementación: de bloque, de una línea, de remolque, y de fin de línea. A lo largo del sistema Alimentación Saludable se utilizan los comentarios de fin de línea. Ejemplo:

```
//Habilitamos para que los Profesionales  
//puedan elaborar con detalle el Plan de Alimentacion
```

- **Declaraciones, Cantidad por línea:**

Java Code Conventions recomienda una declaración por línea, ya que facilita los comentarios. Lo vemos reflejado en la siguiente imagen (Figura 2.2) tomada de la clase Persona del código del sistema Aplicación Saludable:

```
public class Persona {  
  
    private String primerNombre;  
    private String segundoNombre;  
    private String primerApellido;  
    private String segundoApellido;  
    private Date fechaNacimiento;  
    private Icon fotoPerfil;  
}
```

Figura 2.2: Declaraciones, Cantidad por línea

- **Declaraciones de class e interfaces:**

Java Code Conventions al codificar clases e interfaces de Java, se siguen las siguientes reglas de formato: ningún espacio en blanco entre el nombre de un método y el paréntesis que abre su lista de parámetros. La llave de apertura aparece al final de la misma línea de la sentencia declaración. La llave de cierre empieza una nueva línea indentada para ajustarse a su sentencia de apertura correspondiente, excepto cuando no existen sentencias entre ambas, que debe aparecer inmediatamente después de la de apertura. Los métodos se separan con una línea en blanco. Lo vemos reflejado en la siguiente imagen (Figura 2.3) tomada del código del sistema Aplicación Saludable:

```
public static void cargarJListRegistro(JList lista,  
    String nuevoRegistro, DefaultListModel modelo) {  
    modelo.addElement(nuevoRegistro.replaceAll(" ", ""));  
    lista.setModel(modelo);  
}
```

Figura 2.3: Declaraciones de class e interfaces

- **Sentencias de retorno:**

Java Code Conventions indica que una sentencia return con un valor no debe usar paréntesis a menos que hagan el valor de retorno más obvio de alguna manera. Esto se refleja en el sistema Alimentación Saludable:

```
return datosProfesional;  
return sistema.getListProfesionales();
```

- **Sentencias simples:**

Java Code Conventions indica que cada línea debe contener como mucho una sentencia. Esto se refleja en el sistema Alimentación Saludable como podemos ver en la siguiente imagen (Figura 2.4):

```
ArrayList<String> listaInfoPlanesAlimentacion = new ArrayList<String>();  
listaInfoPlanesAlimentacion.add(planAlimentacionViernes);  
listaInfoPlanesAlimentacion.add(planAlimentacionSabado);  
listaInfoPlanesAlimentacion.add(planAlimentacionDomingo);  
return listaInfoPlanesAlimentacion;
```

Figura 2.4: Sentencias simples

- **Sentencias for:**

Sistema Alimentación Saludable cumple con el estándar para sentencias for de Java Code Conventions.

- **Sentencias switch:**

Sistema Alimentación Saludable cumple con el estandar para sentencias switch de Java Code Conventions.

- **Asignación de variables:**

Java Code Conventions indica que se debe evitar asignar el mismo valor a varias variables en la misma sentencia. Es difícil de leer. La correcta asignación se refleja en el sistema Alimentación Saludable, a continuación se describe un ejemplo:

```
Object[] objectPlanDeAlimentacion = new Object[2];  
int idPlanAlimentacion = planAlimentacion.getIdPlanAlimentacion();  
objectPlanDeAlimentacion[0] = idPlanAlimentacion;  
String solicitante = datosUsuarioSistema;  
objectPlanDeAlimentacion[1] = solicitante;
```

■ **Convenciones de nombre:**

Las convenciones de nombres hacen los programas más entendibles haciendolos más fácil de leer, por otra parte pueden dar información sobre la función de un indentificador.

- **Clases:** Los nombres de las clases deben ser sustantivos, cuando son compuestos tendrán la primera letra de cada palabra que lo forma en mayúsculas. Ver Figura 2.5.

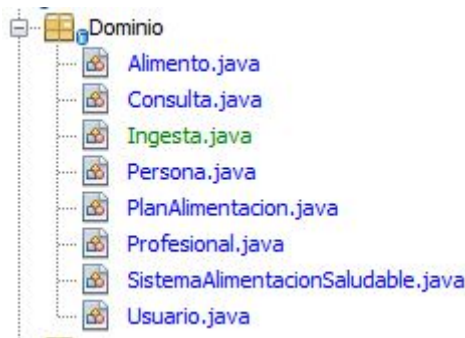


Figura 2.5: Convenciones de nombre - Clases

- **Métodos:** Los métodos deben ser verbos, cuando son compuestos tendrán la primera letra en minúscula, y la primera letra de las siguientes palabras que lo forma en mayúscula (CamelCase) <sup>1</sup>. Ver Figura 2.6.

```
public static String creacionJFileChooser  
  
public static void cargarJListRegistro  
  
public static boolean existeStringCargadoEnJList
```

Figura 2.6: Convenciones de nombre - Métodos

- **Variables:** Los nombres de las variables deben ser cortos pero con significado. La elección del nombre de una variable debe ser un mnemónico, designado para indicar a un observador casual su función.

```
String primerNombre;  
String segundoNombre;  
String primerApellido;  
String segundoApellido;  
Date fechaNacimiento;  
Icon fotoPerfil;
```

Figura 2.7: Convenciones de nombre - Variables

---

<sup>1</sup>Estilo de escritura. CamelCase se podría traducir como Mayúsculas/Minúsculas Camello.

## 2.2. Pruebas unitarias

Se realizaron diversas pruebas que abarcaron las clases del Dominio del sistema, para esto se utilizó la herramienta que IDE NetBeans nos proporciona, JUnit.

En el paquete Test Packages donde se ubica la codificación de las pruebas se crean dos paquetes, uno correspondiente a las clases del Dominio del sistema, llamado: Dominio y otro paquete llamado: Interfaz. El paquete Interfaz también se considera para la codificación de pruebas ya que esta clase del sistema contiene todos los métodos que se vinculan con las clases del dominio del sistema.

Cabe destacar que dentro del paquete Dominio se crea una clase llamada: CargaDeDatosTest, la misma contiene métodos que luego son llamados desde las clases codificadas para las pruebas unitarias, la finalidad de la creación de CargaDeDatosTest es mantener prolijidad en el código, llamado a determinados métodos cuando se necesite, dependiendo de la carga de datos que debemos realizar al codificar una prueba.

A-T-R-I-P demuestra las propiedades de una buena prueba unitaria, por lo que utilizo como características de las pruebas unitarias aplicadas a Aplicación Saludable:

- **Automático:** esta característica hace que la prueba unitaria se puede ejecutar de manera automática sin intervención de usuarios. Por otra parte, en el caso que se codifique un nuevo módulo en el sistema, cambios en el código, si la prueba unitaria sigue funcionando sin problemas, nos garantiza que este cambio en el código, no afectó algún otro módulo u funcionalidad del sistema.
- **Completo:** apunta a que las pruebas unitarias se deben realizar de la forma más completa, siempre tratando de llegar al 100 por ciento de cobertura de las clases del Sistema. En las pruebas unitarias realizadas a Aplicación Saludable, se pudo llegar al 100 por ciento de las clases del Dominio del sistema, pero con respecto a la clase InterfazAlimentacionSaludable contenedora de los métodos relacionados a las clases del Dominio, se seleccionaron las funcionalidades principales.
- **Repetible:** esta característica apunta a que los casos de pruebas se pueden ejecutar cuantas veces se considere necesario y el resultado que se obtiene cada vez que se lleva a cabo la ejecución, tiene que ser siempre el mismo.
- **Independiente:** apunta a que cada prueba es independiente del resto, ya sea por su codificación, por los resultados esperados y sus objetivos.
- **Profesional:** las pruebas codificadas en Aplicación Saludable cumplen con los **Estándares de Codificación** aplicados a lo largo del sistema, lo cual hace que las pruebas puedan tener un fácil mantenimiento si es necesario.

## 2.3. Análisis de código

El análisis de código es una metodología que establece, en base a una serie de reglas de estilo, si el código generado está bien formado y estructurado. Para ello hay herramientas de análisis que evalúan código y en base a las reglas de estilo que tenga aplicadas, nos reportará una serie de errores a corregir.

Lo que si tiene que quedar claro es la importancia de tener un código que esté validado por los estándares de codificación, a continuación se listan algunas de las razones:

- Código bien estructurado, documentado y fácil de mantener.
- Al trabajar con esta metodología, hacemos que la curva de aprendizaje de un nuevo componente del equipo sea menor.
- Más rapidez a la hora de resolver incidencias.
- El desarrollador adquirirá mejores prácticas a la hora de programar, lo que tendremos después, serán desarrolladores más profesionales en su trabajo.
- Refactorización del código mucho más simple.

Para el análisis del código de Aplicación Saludable usamos las siguientes herramientas, plugins instalados en nuestro IDE, NetBeans v8.2:

### 1. Checkstyle:

Genera informes sobre el grado de seguimiento del código a los estándares de codificación establecidos durante el desarrollo del sistema. Cada vez que alguna línea de código no conforme con respecto a las convenciones de Sun Microsystems - ORACLE, aparecerá una etiqueta a modo de aviso, si colocamos el ratón encima de la etiqueta, mostrará cuál es la convención que no se está siguiendo. La instalación de este Plugin en nuestro IDE, hizo que se corrigieran muchos aspectos de nuestro código, como por ejemplo la indentación, utilizar espacios o tabuladores, añadir comentarios, cómo organizar las llaves de los bloques de código, el tamaño máximo de la línea, import innecesarios, visibilidad de atributos de clases, nombrar convenciones de atributos y métodos, buenas prácticas de construcción de clase, secciones de código duplicadas.

### 2. FindBugs Integration:

Este Plugin se encarga de analizar el código en busca de posibles bugs y malas prácticas de codificación, emitiendo un informe de lo ocurrido.

### 3. JUnit:

La instalación de este Plugin permite realizar la ejecución de clases Java de manera controlada, en este caso clases del Dominio del sistema, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Su funcionamiento, en función de algún valor de entrada se

evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente. [3] El uso de este JUnit nos asegura como desarrolladores el incremento la estabilidad del software y te permiten escribir código más rápidamente incrementando su calidad.

#### 4. Jacocoverage:

JaCoCoverage funciona como un servicio adicional a JUnit, colorea todos los archivos java según la información de cobertura de las pruebas de unidad (Ver Figura 2.8). Esto nos permite identificar más fácilmente los caminos en nuestro código que no se cumplen y además tener un informe en porcentaje con respecto a las pruebas que estamos cubriendo. En la Figura 2.9 se puede apreciar el informe de JaCoCoverage en donde en la clase abstracta Persona esta pasando un porcentaje bajo de pruebas, por otra parte en la Figura 2.10 tenemos con éxito el cumplimiento del 100 porciento de pruebas cubiertas para la clase Alimento en este caso. En la Figura 2.11 se puede apreciar el informe JaCoCoverage desde la web para el paquete Dominio del sistema. La Figura 2.12, refleja el cumplimiento de 100 porciento de pruebas para todos los metodos de la clase Ingesta perteneciente al Dominio del sistema.

```
public Icon getFotoPerfil() {
    return fotoPerfil;
}

public void setPrimerNombre(String primerNombre) {
    this.primerNombre = primerNombre;
}
```

Figura 2.8: JUnit - Colorea archivos java según información de cobertura de pruebas

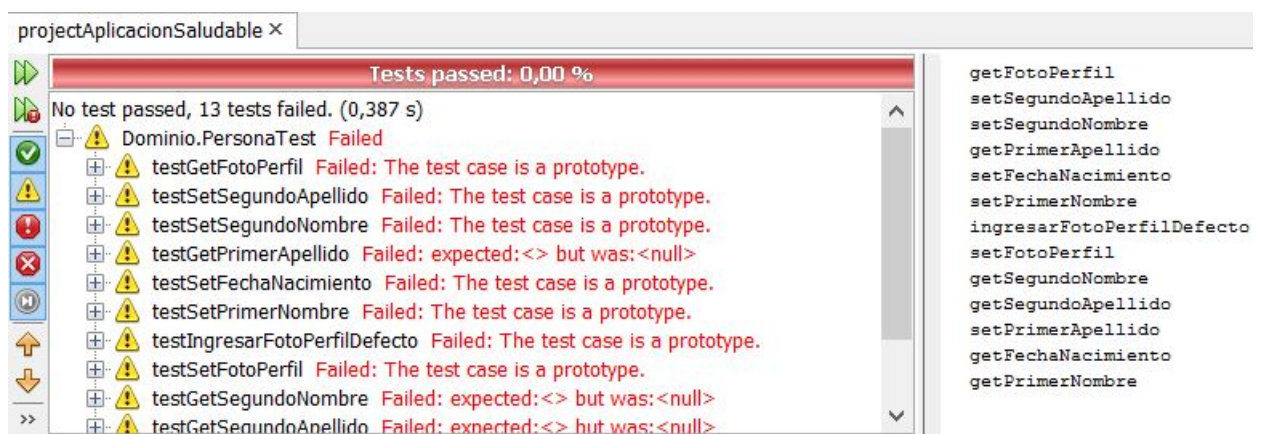


Figura 2.9: JUnit - Tests Passed 0,0 porciento



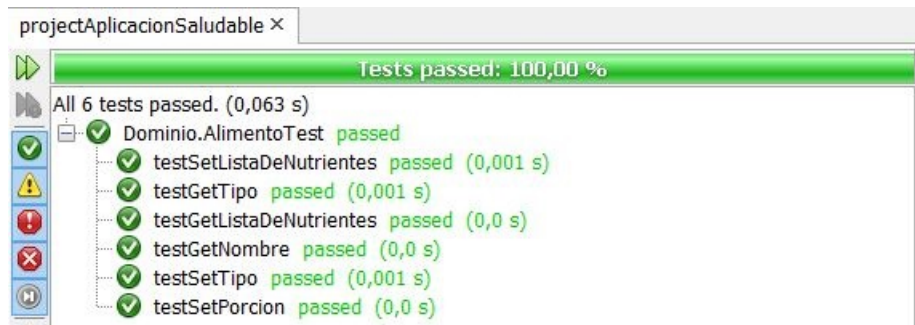


Figura 2.10: JUnit - Tests Passed 100 porciento

## Dominio

Element	Missed Instructions	Cov.
PlanAlimentacion	<div><div></div></div>	100%
SistemaAlimentacionSaludable	<div><div></div></div>	100%
Usuario	<div><div></div></div>	100%
Consulta	<div><div></div></div>	100%
Persona	<div><div></div></div>	100%
Alimento	<div><div></div></div>	100%
Profesional	<div><div></div></div>	100%
Ingesta	<div><div></div></div>	100%
Total	0 of 471	100%

Figura 2.11: JUnit - Informe JaCoCoverage

## Ingesta

Element	Missed Instructions	Cov.
setAlimentoIngerido(Alimento)	<div><div></div></div>	100%
setFechaIngesta(Date)	<div><div></div></div>	100%
Ingesta()	<div><div></div></div>	100%
getAlimentoIngerido()	<div><div></div></div>	100%
getFechaIngesta()	<div><div></div></div>	100%
Total	0 of 17	100%

Figura 2.12: JUnit -Ingesta JaCoCoverage



A continuación, en Figura 2.13 se puede apreciar como se cumple con la cobertura de pruebas unitarias en su 100 por ciento para métodos que se encuentran dentro de la clase InterfazAlimentacionSaludable y se relacionan con las clases del Dominio del sistema.

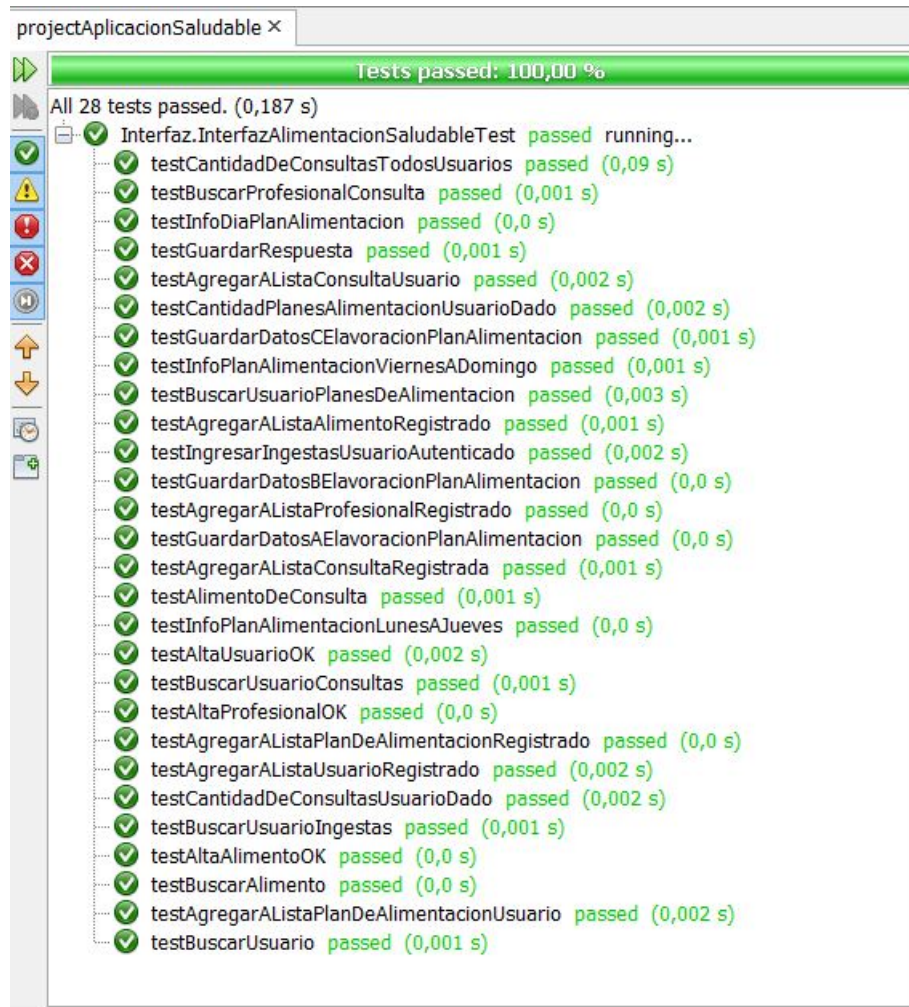


Figura 2.13: JUnit -Informe JaCoCoverage

## 3. Interfaz de usuario y usabilidad

### 3.1. Criterios de interfaz de usuario

Una interfaz de software es la parte de una aplicación que el usuario ve y con la cual interactúa. Está relacionada con la estructura, la arquitectura, y el código que hace el trabajo del software, pero no se confunde con ellos. Por otra parte la interfaz incluye las pantallas, ventanas, controles, menús, la documentación y el entrenamiento, en definitiva cualquier cosa que el usuario ve y con lo cual interactúa es parte de la interfaz. [4].

Es fundamental dedicar tiempo de un proyecto al diseño de la interfaz de Usuario de un sistema, aplicando buenas practicas de usabilidad y así podes llegar a obtener un sistema con una interfaz inteligente, fácil de aprender y usar. En donde permita a los usuarios hacer su trabajo o desempeñar una tarea en la manera que hace más sentido para ellos, en vez de tener que ajustarse al software, en definitiva el sistema se diseña específicamente para la gente que la usará.

Para comenzar el desarrollo de la Interfaz de Usuario del sistema Aplicación Saludable se comenzó a investigar sobre como construir una interfaz de usuario en donde se apliquen buenas practicas de usabilidad para una aplicaciones de escritorio profesionales:

- Diseño de Interfaces de Usuario Usables [5]
- Usabilidad - Agesic [6]
- Usabilidad, métodos y las técnicas[7]

A continuación se pasan a describir **principales conceptos**, luego de la investigación sobre usabilidad e interfaz gráfica de usuarios, que mas adelante se aplicaran como buenas practicas al sistema Aplicación Saludable:

### **El usuario no está utilizando tu aplicación**

La cuestión más básica a considerar en el diseño de interfaces de usuario es que el usuario no quiere utilizar tu aplicación. Quieren hacer su trabajo de la forma más sencilla y rápida posible, y la aplicación no es más que otra herramienta para ayudarles a lograrlo. Cuanto menos estorbe tu aplicación al usuario, mejor. El esfuerzo utilizado en usar tu aplicación es esfuerzo que no pueden utilizar en la tarea que están intentando realizar. Citas del libro de Alan Cooper:

“Imagina usuarios muy inteligentes pero muy ocupados”  
“No importa lo genial que sea tu interfaz, menos es más siempre”

### **Interferencias Innecesarias**

Cuando un usuario está trabajando, su atención está centrada en el trabajo que está realizando. Cada vez que tienen que concentrarse en la aplicación, les lleva tiempo el volver a centrarse en el trabajo. Por lo que se debería minimizar la cantidad de distracción y de interferencias por parte de tu aplicación. Cada aplicación tiene un elemento clave en la que centrarse, por lo que hay que hacer de este elemento clave el centro de la interfaz.

### **Utiliza la potencia de la computadora**

Cada vez que halla una decisión que tomar o trabajo que hacer, intentaremos que la interfaz lo haga por el usuario. Como por ejemplo, se tiene una aplicación en donde se tiene determinadas pestañas con diferentes funcionalidades en cada una de ellas, en cada de estas pestañas se debería comprobar el nombre de la funcionalidad y mostrar suficiente información para poder distinguirlas. De esa forma, se puede seleccionar entre multitud de funcionalidades diferentes del sistema sin tener que pensar demasiado.

### **Elementos fácil de distinguir y fácil de buscarlos**

Los elementos de la pantalla que hacen cosas distintas deberían ser fácilmente distinguibles unos de otros.

### **No elegir: lindo y fácil de usar**

Desde el punto de vista de la Usabilidad podemos afirmar que un sistema ya sea lindo o fácil de usar, uno no sustituye a otro en ningún caso: “Un gran sitio debe ser estéticamente exquisito y terriblemente fácil de usar. A la vez”.

### **Miro y entiendo**

Se trata de un nivel de interacción inconsciente, donde el usuario del sistema requiere de esfuerzo casi nulo para hacerse de información y conocimiento.

### **Leo y entiendo**

Leo y entiendo constituye el nivel siguiente de interacción, después de miro y entiendo. En este punto consiste en que el usuario del sistema lea el contenido de las etiquetas y textos, con la particularidad de que no necesita nada más que el texto que se lee para comprender el sentido del mismo.

**Principales atributos** que definen la usabilidad de un sistema interactivo:

- **Facilidad de aprendizaje:** minimizar el tiempo que se requiere desde el no conocimiento de una aplicación hasta su uso productivo.
- **Tiempo de respuesta:** capacidad del sistema de expresar los cambios de estado del usuario. Este factor es muy variable, ya que depende de las características que tenga la PC donde se encuentre el usuario.
- **Flexibilidad:** formas de intercambiar la información el usuario con el sistema. Aportar flexibilidad al sistema implica brindar control al usuario, capacidad de sustitución y capacidad de adaptación.
- **Robustez:** caracteriza la necesidad de que el usuario cumpla con sus objetivos y que disponga del asesoramiento necesario.
- **Recuperabilidad:** grado de facilidad que una aplicación permite al usuario para corregir una acción una vez está reconocido un error.
- **Sintetizabilidad:** este factor se caracteriza porque el usuario sea capaz de captar cuando ocurra algún cambio de operación en el sistema.
- **Consistencia:** es concepto clave en la usabilidad de un sistema informático. Es la capacidad de utilizar de la misma manera todos los mecanismos, sea cualquiera el momento que se necesite.
- **Disminución de la carga cognitiva:** los aspectos cognitivos de la interacción proporcionan la necesidad que tienen los usuarios de confiar más en los reconocimientos que en los recuerdos, es decir, el usuario no tiene que recordar abreviaciones y códigos muy complicados, por ejemplo.

### **Sistema Aplicación Saludable:**

Al ejecutar Aplicación Saludable nos encontramos con una ventana en modo pantalla completa, intentando captar toda la atención del usuario desde el inicio del sistema. Se decide el uso de pantalla completa para que todas las funcionalidades del sistema, se ejecuten en un ambiente, en este caso en una ventana controlada, evitando que se abran múltiples ventanas sin tener control de su superposición. Todas las funcionalidades que se ejecuten despliegan una ventana interna contenida sobre una ventana principal. Estas ventanas internas tienen la opción de minimizarse y maximizarse, quedando agrupadas en el extremo superior izquierdo, facilitando al usuario la interacción con el sistema, ya que no es necesario cerrar determinada funcionalidad para realizar otra. En la siguiente imagen (Figura 3.1) mostramos de que trata esta funcionalidad.



Figura 3.1: Funcionalidad Minimizar - Maximizar ventanas internas

El sistema cuenta con un menú principal ubicado en la parte superior del sistema, en donde aparecen las principales funcionalidades del sistema, las cual mencionamos a continuación:

- Registrar Usuario
- Registra Profesional
- Registrar Alimento
- Alimentos Ingeridos
- Consulta con Profesional
- Consultas de Usuarios
- Sugerencia de Plan de Alimentación
- Crear Plan de Alimentación
- Salir

Al momento de codificar la interfaz de usuario se pensó en cual era la mejor practica para que el usuario pueda tener todo a mano sin requerir de mucho esfuerzo, es aquí donde se decide implementar el sistema con un menú principal, facilitando la interacción del sistema con el usuario, mejorando la navegabilidad del sistema, pudiendo acceder siempre a cualquier funcionalidad sin importar cual sea la que se este ejecutando.

Por otra parte el menú principal cuenta con un Menú Ítem llamado **Rol del Sistema** el cual llamaremos **menú de autenticación**, donde muestra los diferentes roles del sistema: Administrador, Usuario y Profesional.

Cabe destacar que al inicio del sistema, el perfil autenticado es: Usuario, con el acceso a sus correspondientes funcionalidades.

- **Administrador:** El perfil administrador podrá registrar Usuarios y Profesionales, salir del sistema.
- **Usuario:** El perfil Usuario podrá registrar alimentos, alimentos ingeridos, realizar consultas con profesionales, solicitar y consultar planes de alimentación sugeridos por un profesional del sistema, salir del sistema.

- **Profesional:** El perfil Profesional podrá registrar alimentos, responder consultas elevadas por los usuarios del sistema, formular planes de alimentación y salir del sistema.

Si el perfil autenticado es Usuario o Profesionales, se carga una lista de todos los Usuarios, Profesionales registrados en el sistema, según corresponda. Esto lo podemos ver reflejado en el sistema si realizamos click sobre el **menú de Rol autenticado**. A su vez si realizamos click sobre cualquier de los Usuarios o Profesionales (listados según **menú de autenticación**) muestra automáticamente en el **menú información de autenticado**, ubicado en el menú principal del sistema el nombre-apellido en color rojo del Usuario o Profesional, indicado su foto de perfil si la tiene cargada en el sistema, conjunto a esto, se cargan en el menú principal del sistema las funcionalidades permitidas según corresponda la autenticación. En la siguiente imagen (Figura 3.1) se muestra el funcionamiento del sistema para rol del sistema Usuario y selección de perfil Usuario: Andre Hernandez.



Figura 3.2: Funcionamiento Autenticación - Aplicación Saludable

Para el menú principal del sistema se aplico el mismo tamaño, fuente y color de letra para todos los menú: Segoe UI 15 Bold, dejando fuera el **menú información de autenticado** donde se presenta en color rojo llamando la atención del usuario indicándole que usuario se encuentra autenticado en el sistema en ese momento. Se definió Segoe UI 15 Bold letra y sus características ya que se considera que su formato es fácil de leer facilitando su legibilidad para navegar por las diferentes funcionalidades del sistema sin que el usuario requiera de un esfuerzo extra. Con respecto a las funcionalidades, al realizar click sobre un ítem del menú principal, queda resaltado su contorno en color rojo, esto se hace para que el usuario siempre sepa en que sección del menú se encuentra y esta ejecutando en ese momento; al realizar click en otro ítem del menú, el anterior clikeado vuelve a su contorno original y se pinta de color rojo el contorno de la nueva funcionalidad clikeada. Ver Figura 3.3.



Figura 3.3: Posicionamiento según funcionalidad - Menú del Sistema

Con respecto a las ventanas internas:

- **Letra:** Se adopta el mismo tamaño, fuente y color de letra para todos los textos: Segoe UI Emoji 12 Bold.
- **Color de fondo:** RGB [211,234,216], verde tenue. Se selecciona este color para las ventanas internas ya que luego de una investigación sobre colores adecuados para aplicaciones teniendo en cuenta las preferencias de los usuarios, se llegó a que el color verde está asociado con la naturaleza, la tranquilidad y la salud [8], en este caso nuestra aplicación: Aplicación Saludable encaja con este perfil.
- **Movimiento:** El usuario puede desplazar las ventanas internas por toda la ventana principal hasta encontrar el lugar que le parezca más favorable o cómodo para el uso del sistema.

Todas las ventanas internas tienen la funcionalidad de minimizar y cerrar. Únicamente las ventanas internas correspondientes a las funcionalidades de Planes de Alimentación y Consulta a profesionales tienen la opción en donde el usuario puede maximizar, siempre dentro de la ventana principal del sistema, teniendo también visible el menú principal por si el usuario desea acceder a otras funcionalidades. Esto se define de esta manera, ya que son las funcionalidades del sistema en donde se maneja más flujo de información por parte del usuario y además son las funcionalidades principales del sistema.

Tanto en las ventanas internas referentes a Planes de Alimentación y Consulta a profesionales se maneja el uso de Layouts para que cuando el usuario maximice la ventana, los elementos que están dentro mantengan determinado orden y se reajusten al nuevo tamaño de ventana, sin causar efecto sobre el usuario.

Con respecto a los controles que se realizan en el sistema:

- El sistema no permite registrar Usuarios ni Profesionales sin ingresar los campos de Primer Nombre y Primer Apellido.
- El sistema no permite ingresar Profesionales sin ingresar el nombre del Título Profesional.
- El sistema no permite que se registren Usuarios o Profesionales con la misma combinación de Nombre-Apellido, ya que al realizar click sobre el **menú de Rol autenticado** y seleccionar un Usuario / Profesional según corresponda, se crea un perfil y el mismo debe ser único para el sistema.
- El sistema no permite que se registren alimentos con el mismo nombre, ya que generaría inconsistencia en su lógica interna.
- El sistema al momento de registrar un alimento en el sistema, se controla que la porción que se ingresa correspondiente al Alimento no sea menor a 0 gramos y tampoco mayor que 1000 gramos.

- El sistema no permite crear una consulta con profesional si no se completa el Titular de la consulta, el alimento consultado y el descripción de la consulta.
- El sistema no permite crear una solicitud de plan de alimentación si no se completa el campo observaciones.

Los controles que se realizan en el sistema de alguna manera tienen que ser vistos por el usuario a la hora de estar interactuando con el sistema, pero esto se debe hacer de manera cuidadosa; no siendo **Interferencias Innecesarias**. En Aplicación Saludable la manera de avisarle al usuario de que existe un control y que lo que esta queriendo ingresar en el sistema no es correcto o no esta validado, es remarcando el campo de ingreso en color rojo, ademas de eso, si se pasa el mouse por encima del campo, este le mostrara un mensaje informativo indicando lo que esta pasado y que debe hacer para solucionarlo. En la siguiente figura se muestra el Registro de Usuario, en donde se intenta ingresar un Usuario dejando vacío el campo Primer Apellido, Ver Figura 3.4.

The image shows a web form for user registration. The form has a light green background and a blue header bar with standard window controls (minimize, maximize, close). The fields are labeled as follows: 'Primer Nombre:' with the value 'Andre', 'Segundo Nombre:', 'Primer Apellido:', 'Segundo Apellido:', 'Nacionalidad:', and 'Fecha de Nacimiento:' with a calendar icon. The 'Primer Apellido:' field is highlighted with a red border, indicating an error. A yellow tooltip message 'Ingresa primer apellido para el Usuario' is displayed over the 'Segundo Apellido:' field. The right side of the form has a background image of wood.

Figura 3.4: Gestión de errores del Sistema

## 3.2. Evaluación de usabilidad

Para la evaluación de la usabilidad del sistema se tomaron en cuenta las 10 heurísticas de Jakob Nielsen [9].

A continuación pasaremos a comentar cada una de las 10 heurísticas de Jakob Nielsen aplicadas al sistema Alimentación Saludable:

- **Visibilidad:** Explica al usuario cuál es el estado del sistema en cada momento, y manténle informado de lo que está pasando. Aplicación Saludable, brinda un menú principal con las funcionalidades del sistema haciéndole saber al usuario donde se encuentra a cada momento remarcado la sección de menú en que se encuentra con color rojo.



- **Relación con la realidad:** Utiliza un lenguaje familiar y apropiado para los usuarios a los que te diriges, y organiza la información con un orden natural y lógico. La forma en que Aplicación Saludable transmite al Usuario los errores es conceptualmente amigable en ámbitos de usabilidad, remarcando con color rojo el campo en donde ocurre el problema, además indicando al pasar el mouse por encima, un mensaje informativo indicando que ocurre. Por otra parte la organización de la información y el lenguaje que se usa en el sistema, no es complicado ni difícil de entender. **Miro y Entiendo - Leo y Entiendo.**
- **Control y libertad:** Ofrece funciones de rehacer y deshacer que permitan al usuario tener el control de sus interacciones con libertad. Alimentación Saludable no permite rehacer y deshacer, pero si el usuario tiene control sobre lo que esta haciendo en el sistema, puede moverse dentro de las ventanas sin problemas, mover ventanas, acceder a varias funcionalidades a la vez.
- **Consistencia y estándares:** Establecer una convenciones lógicas y mantenerlas siempre, es decir: mismo lenguaje, mismo flujo de navegación. El sistema tiene el mismo flujo de navegación, para ir a determinada funcionalidad siempre se debe ir hasta el menú principal, luego se abre la ventana interna correspondiente a la funcionalidad y luego el usuario puede realizar sus tareas, siempre es el mismo proceso.
- **Prevención de errores:** Ayuda a los usuarios a evitar equivocarse antes de que cometan el error. Aplicación Saludable realiza varios controles mencionados anteriormente para que no se generen problemas en la lógica del sistema, además el sistema no debe perder robustez y consistencia. Como por ejemplo, ya hablado anteriormente el contorno rojo de los campos indicando lo que esta sucediendo y cual es el error cometido por el Usuario.
- **Reconocimiento:** Haz visible todo lo que sea posible, no esperes que los usuarios recuerden o memoricen información, muéstrala si es necesaria en el proceso, las instrucciones deben estar a la vista cuando sea necesario. El sistema cuenta con un menú principal donde todas las funcionalidades del sistema están a la vista, pudiendo realizar la operación que necesite al instante sin esfuerzo de cerrar o pausar funcionalidad en curso por el Usuario.
- **Flexibilidad:** Permite que el sistema pueda adaptarse a los usuarios frecuentes, diseña la realización de tareas avanzadas de manera fluida y eficiente. El sistema tiene aspectos de configuración, como por ejemplo las ventanas internas que se redimensionan, es el caso de consultas con profesionales y planes de Alimentación.
- **Estética y minimalismo:** Muestra sólo lo necesario y relevante en cada situación, no distraigas al usuario con información extra poco relevante. Aplicación Saludable muestra solo lo relévate para el Usuario que este autenticado en este momento. Si se realiza un cambio de autenticación en el sistema, todas las ventanas internas que estaban abiertas en una sección anterior, son cerradas para que el nuevo usuario autenticado tenga el sistema sin ventanas internas abiertas que no le interesan o no le correspondan.

- **Recuperarse de los errores:** Ayuda a los usuarios a reconocer y corregir sus errores, indica siempre el problema concreto que está ocurriendo y sugiere soluciones constructivas. Como se menciono anteriormente el sistema muestra el campo donde ocurre el error con contorno rojo, indicando ademas un mensaje informativo para que el usuario pueda recuperar del error sin problemas ni dificultades.
- **Ayuda y documentación:** La información de ayuda debe ser breve, concisa, fácil de buscar y enfocada a las tareas del usuario. El sistema no presenta información de ayuda al Usuario.

## 4. Pruebas funcionales

### 4.1. Técnicas de prueba aplicadas

Una prueba funcional es una prueba basada en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software. Las pruebas funcionales se hacen mediante el diseño de modelos de prueba que buscan evaluar cada una de las opciones con las que cuenta el software. En definitiva son pruebas específicas, concretas y exhaustivas para probar y validar que el software hace lo que debe hacer y sobre todo, lo que se ha especificado. [10].

Para el proyecto se aplicaron técnicas de **Caja Negra**, técnica de pruebas de software en la cual la funcionalidad se verifica sin tomar en cuenta la estructura interna de código, detalles de implementación o escenarios de ejecución internos en el software; se hace un enfoque solamente en las entradas y salidas del sistema, sin preocuparnos en tener conocimiento de la estructura interna del programa de software. Para obtener el detalle de cuáles deben ser esas entradas y salidas, nos basamos en los requerimientos de software y especificaciones funcionales [11].

- **Técnica de partición equivalente:** Ejemplo: si se están realizando pruebas con valores numéricos, dividir las pruebas por conjunto de números: numero negativos, numero negativos.
- **Técnica de valores limites:** Ejemplo: Si se esta realizado una prueba para probar valores limites del ingreso de un numero en un campo numérico, probar con sus limites, que pasa si pruebo con 0, con valores menores a 0, con el máximo permitido, con el mínimo permitido, con el ingreso de un numero mayor al máximo permitido, con el ingreso de un numero menor al mínimo permitido.
- **Técnica de desempeño:** Prueba el sistema en tiempo de ejecución, tiempo de respuesta en abrir determinada funcionalidad, tiempo de respuesta en cargar datos.
- **Técnica de integridad del sistema:** Consta de pruebas que integran todo el sistema, como por ejemplo realizar un acción que involucre mas de una funcionalidad del sistema, para chequear que todos los módulos tengan conexión entre ellos permitiendo la navegabilidad dentro del sistema.

## 4.2. Casos de prueba

### CP1 - Registrar Alimento

Tabla 4.1: CP1 - Registrar Alimento

Entrada de datos	Salida de datos	Resultado esperado
Nombre: Durazno Tipo: Fruta Porción: 15	Sistema permite la creación del Alimento con los siguientes datos: Nombre: Durazno Tipo: Fruta Porción: 15	Concuerda con Salida de datos. OK
Nombre: Tallarines Tipo: Pasta Porción: 150 Lista Nutrientes: Carbohidratos	Sistema permite la creación del Alimento con los siguientes datos: Nombre: Tallarines Tipo: Pasta Porción: 150 Lista de Nutrientes: Carbohidratos	Concuerda con Salida de datos. OK
Nombre: Manzana	Sistema no permite la creación del Alimento. Sistema marca contorno de los campo en color rojo: Nombre, Tipo, Porción indicando mensaje que se debe completar los campos para el ingreso.	Concuerda con Salida de datos. OK
Tipo: Fruta	Sistema no permite la creación del Alimento. Sistema marca contorno de los campo en color rojo: Nombre, Tipo, Porción indicando mensaje que se debe completar los campos para el ingreso.	Concuerda con Salida de datos. OK
Porción: 26	Sistema no permite la creación del Alimento. Sistema marca contorno de los campo en color rojo: Nombre, Tipo, Porción indicando mensaje que se debe completar los campos para el ingreso.	Concuerda con Salida de datos. OK
Escenario previo: Alimento creado en el sistema: Nombre: Manzana Tipo: Fruta Porción: 56 <hr/> Se ingresa en el sistema: Nombre: Manzana Tipo: Fruta Porción: 69	Sistema no permite la creación del Alimento. Sistema marcando el contorno en color rojo del campo Nombre indicando mensaje: Alimento existente en el Sistema.	Se espera que el Sistema no permita el ingreso del Alimento marcando el campo en color rojo Nombre y ademas manteniendo los datos en pantalla. NO OK.

Se aplican **Técnica de Partición Equivalente** y **Técnica de valores limites** para probar campo: Porción del Registro de Alimentos:

Tabla 4.2: Técnica de Partición Equivalente - Registro Alimento

Patrón de Equivalencia	Clases Validas	Clases No Validas
porcionA	(porcionA <= 1000 porcionA >= 1)	porcionA >1000 porcionA <1 porcionA tiene decimal porcionA vacia porcionA no numérico

Tabla 4.3: Técnica valores limites - Registro Alimentos

Entrada de datos	Salida de datos	Resultado esperado
Nombre: Durazno Tipo: Fruta Porción: -50	SDD1: Sistema no permite la creación del Alimento, marca contorno en color rojo del campo: Porción indicando mensaje: Porción ingresada incorrecta. La porción ingresada debe estar entre 1 gramo y mil gramos.	Concuerda con salida de datos. OK
Nombre: Pera Tipo: Fruta Porción: 0	Sistema se comporta al igual que SDD1.	Concuerda con salida de datos. OK
Nombre: Uva Tipo: Fruta Porción: 1	Sistema permite la creación del alimento con los siguientes datos: Nombre: Uva - Tipo: Fruta - Porción: 1	Concuerda con salida de datos. OK
Nombre: Anana Tipo: Fruta Porción: 360	Sistema permite la creación del alimento con los siguientes datos: Nombre: Anana - Tipo: Fruta - Porción: 360	Concuerda con salida de datos. OK
Nombre: Pera Tipo: Fruta Porción: 1000	Sistema permite la creación del alimento con los siguientes datos: Nombre: Pera -Tipo: Fruta - Porción: 1000	Concuerda con salida de datos. OK
Nombre: Pescado Tipo: Carne Porción: 1001	Sistema se comporta igual que SDD1	Concuerda con salida de datos. OK
Nombre: Asado Tipo: Carne Porción: 1923	Sistema se comporta igual que SDD1.	Concuerda con salida de datos. OK

## CP2 - Registrar Usuario

Tabla 4.4: CP1 - Registrar Usuario

Entrada de datos	Salida de datos	Resultado esperado
Primer Nombre: Andre Segundo Nombre: Fernando Primer Apellido: Hernandez Segundo Apellido: Aguirre	Sistema permite la creación del Usuario con los siguientes datos: Primer Nombre: Andre Segundo Nombre: Fernando Primer Apellido: Hernandez Segundo Apellido: Aguirre	Concuerda con Salida de datos. OK
Primer Nombre: Andre Segundo Nombre: Fernando Primer Apellido: Hernandez Segundo Apellido: Aguirre Lista de Preferencias: Arroz, Carne, Leche. Lista de Restricciones: Sal, Harina	Sistema permite la creación del Usuario con los siguientes datos: Primer Nombre: Andre Segundo Nombre: Fernando Primer Apellido: Hernandez Segundo Apellido: Aguirre Lista de Preferencias: Arroz, Carne, Leche. Lista de Restricciones: Sal, Harina	Concuerda con Salida de datos. OK
Segundo Nombre: Gerardo Primer Apellido: Benitez Fecha de nacimiento: 12/05/1996	Sistema no permite la creación del Usuario. Sistema marca contorno de campo en color rojo: Primer Nombre, indicando mensaje que se debe completar el campo Primer Nombre.	Concuerda con Salida de datos. OK
Primer Nombre: Alverto Segundo Nombre: Jose Segundo Apellido: Gerardo Lista de Preferencias: Arroz, Carne, Leche.	Sistema no permite la creación del Usuario. Sistema marca contorno de campo en color rojo: Primer Apellido, indicando mensaje que se debe completar el campo Primer Apellido.	Concuerda con Salida de datos. OK
Primer Nombre: Andre Primer Apellido: Hernandez Se ingresa foto de perfil	Sistema permite la creación del Usuario con los siguientes datos: Primer Nombre: André Primer Apellido: Hernández Se ingresa foto de perfil	Concuerda con Salida de datos. OK
Escenario previo: Usuario creado en el sistema: Primer Nombre: Andre Primer Apellido: Hernandez <hr/> Se ingresa en el sistema: Primer Nombre: Andre Primer Apellido: Hernandez Fecha de Nacimiento: 06/05/1985	Sistema no permite la creación del Usuario. Sistema marcando el contorno en color rojo de los campos Primer Nombre y Primer Apellido indicando mensaje: Usuario existente en el Sistema.	Concuerda con Salida de datos. OK

### CP3 - Registrar Profesional

Tabla 4.5: CP3 - Registrar Profesional

Entrada de datos	Salida de datos	Resultado esperado
Primer Nombre: Andre Segundo Nombre: Pedro Primer Apellido: Fernandez Segundo Apellido: Perez Nombre Título Profesional: Licenciado en Nutrición Fecha de Graduación: 01/06/2004	Sistema permite la creación del Profesional, con los siguientes datos: Primer Nombre: Andre Segundo Nombre: Pedro Primer Apellido: Fernandez Segundo Apellido: Perez Nombre Título Profesional: Licenciado en Nutrición. Fecha de Graduación: 01/06/2004	Concuerda con salida de datos. OK
Segundo Nombre: Andre Primer Apellido: Hernandez	Sistema no permite el ingreso del Profesional, marcado campo en color rojo Primer Nombre, indicando con mensaje: Ingrese primer nombre para el Profesional.	Concuerda con salida de datos. OK
Primer Nombre: Andre Segundo Nombre: Pedro Segundo Apellido: Fernandez Fecha de Nacimiento: 01/05/1987	Sistema no permite el ingreso del Profesional, marcando campo en color rojo Primer Apellido, indicando con mensaje: Ingrese primer apellido para el Profesional.	Concuerda con salida de datos. OK
Primer Nombre: Andre Primer Apellido: Fernandez Fecha de Graduación: 01/05/2005	Sistema no permite el ingreso del Profesional, marcando campo en color rojo Nombre Título Profesional, indicando con mensaje: Ingrese nombre de Título Profesional.	Concuerda con salida de datos. OK
Escenario previo Profesional creado en el sistema: Primer Nombre: Andre Primer Apellido: Hernandez Nombre Título Profesional: Licenciado en Nutrición. <hr/> Se ingresa en el sistema: Primer Nombre: Andre Primer Apellido: Hernandez Nombre Título Profesional: Deportologo	Sistema no permite la creación del Profesional, marcando en color rojo los campos Primer Nombre y Primer Apellido, indicando mensaje: Profesional existente en el sistema, manteniendo los datos en los campos para ser modificados.	Concuerda con salida de datos. OK.

## CP4 - Registrar Ingesta

Tabla 4.6: CP4 - Registrar Ingesta

Entrada de datos	Salida de datos	Resultado esperado
<p>Escenario previo:          Usuario Autenticado en el Sistema: Andre Hernandez</p> <hr/> <p>Se ingresan datos de la Ingesta:          Alimento Ingerido: Durazno          Fecha: 01/05/2017</p>	<p>Sistema permite el registro de la ingesta con los siguientes datos:          Alimento Ingerido: Durazno          Fecha: 01/05/2017          para el Usuario autenticado en el menú del sistema: Andre Hernandez, mostrando los datos ingresados en la tabla de Ingestas.</p>	<p>Concuerda con salida de datos.          OK</p>
<p>Escenario previo:          Usuario Autenticado en el Sistema: Andre Hernandez</p> <p>Ingesta registrada para el Usuario Andre Hernandez con los siguientes datos:          Alimento Ingerido: Pera          Fecha: 05/08/2017</p> <hr/> <p>Se ingresan datos de la Ingesta:          Alimento Ingerido: Pera          Fecha: 05/08/2017</p>	<p>Sistema permite el registro de la ingesta con los siguientes datos:          Alimento Ingerido: Pera          Fecha: 05/08/2017 para el Usuario Autenticado: Andre Hernandez, mostrando los datos ingresados en la tabla de Ingestas.</p>	<p>Lo que se espera es que el sistema informe al usuario que ya se registro esa ingesta para la fecha que esta intentando registrar; evitando el registro y no mostrado los datos en la tabla de Ingestas.          NO OK.</p>
<p>Escenario previo: Usuario Autenticado en el sistema: Andre Hernandez</p> <hr/> <p>Alimento Ingerido: Pera</p>	<p>Sistema no permite el registro de la Ingesta, marcando contorno del campo Fecha en color rojo, indicando con mensaje:          Debe seleccionar una fecha de Ingesta. No se cargan datos de ingesta en la tabla Ingesta.</p>	<p>Concuerda con salida de datos.          OK</p>
<p>Funcionalidad ordenar tabla de ingestas;</p> <p>Se realiza click sobre columna: Día de Ingesta de la tabla Ingesta.</p>	<p>Sistema se encarga de ordenar la tabla ingesta por la columna que fue clickeada, en este caso la columna: Día de Ingesta.</p>	<p>Concuerda con salida de datos.          OK.</p>

En este Registro de Ingesta tenemos la particularidad de tener un combo en donde se cargan automáticamente todos los Alimentos registrados previamente en el sistema cuando es ejecutada la funcionalidad de Registrar Ingesta. Aquí es donde se aplica la **Técnica de Desempeño** y también la **Técnica de Integridad del sistema**, ya que si registramos un nuevo Alimento desde la sección Registrar Alimento, sin cerrar nuestra funcionalidad Registrar Ingesta, al momento de hacer click



sobre el combo de Alimento Ingerido, nos carga el nuevo Alimento ingresado desde otra funcionalidad del sistema, por lo que tenemos una correcta comunicación entre los módulos del sistema.

### CP5 - Sugerencia de Plan de Alimentación

Tabla 4.7: CP5 - Sugerencia de Plan de Alimentación

Entrada de datos	Salida de datos	Resultado esperado
<p>Escenario previo:          Usuario Autenticado          en el Sistema:          Andre Hernandez</p> <hr/> <p>Observaciones: Me gustaría un Plan de Alimentación en base a grandes cantidades de Frutas.</p> <p>Click en Enviar Solicitud.</p>	<p>Sistema permite enviar la solicitud de creación de un Plan de Alimentación para el Usuario Autenticado: Andre Hernandez, en donde el campo Observaciones es: Me gustaria un Plan de Alimentación en base a grandes cantidades de Frutas.</p> <p>Se carga la solicitud del Plan de Alimentación en la tabla Historial de Planes de Alimentación.</p>	<p>Concuerda con salida de datos.          OK</p>
<p>Escenario previo:          Usuario Autenticado          en el Sistema:          Andre Hernandez</p> <hr/> <p>Click en boton          Enviar Solicitud.</p>	<p>Sistema no permite enviar la solicitud, marca el contorno del campo Observaciones en color rojo indicando mensaje: Ingrese Observaciones para la solicitud de un Plan de Alimentación.</p>	<p>Concuerda con salida de datos.          OK</p>
<p>Escenario previo:          Usuario Autenticado          en el Sistema: Andre Hernandez          Solicitud de Plan de Alimentación          enviada previamente.</p> <hr/> <p>Usuario realiza click sobre la solicitud de Plan de Alimentación mostrada en la tabla Historial de Planes de Alimentación.</p>	<p>Se cargan en panel de Detalles de Planes de Alimentación toda la información del plan de Alimentación que el usuario clickeo.</p>	<p>Concuerda con salida de datos.          OK</p>

## CP6 - Consulta con Profesional

Tabla 4.8: CP5 - Consulta con Profesional

Entrada de datos	Salida de datos	Resultado esperado
<p>Escenario previo:          Usuario Autenticado en el Sistema: Andre Hernandez</p> <hr/> <p>Titular de Consulta:          Dieta en base a frutas          Alimento Consultado: Pera          Descripción de la Consulta:          Dieta en base a frutas orientada al deporte</p>	<p>Sistema permite el ingreso de la consulta para el Usuario Autenticado: Andre Hernandez con los siguientes datos: Titular de la Consulta: Dieta en base a frutas. Descripción de la Consulta: Dieta en base a frutas orientada al deporte. Se cargan datos de la consulta ingresada en tabla Historial de Consultas.</p>	<p>Concuerda con salida de datos. OK</p>
<p>Escenario previo:          Usuario Autenticado en el Sistema: Andre Hernandez</p> <hr/> <p>Alimento Consultado: Pera          Descripción de la Consulta:          Dieta en base a carnes.</p>	<p>Sistema no permite el ingreso de la consulta para el Usuario Autenticado en el sistema: Andre Hernandez, marca contorno del campo Titular de la Consulta en color rojo indicando mensaje: Ingrese Titular de Consulta. No se limpian los campos de la pantalla. No se muestra Historial de Consultas con datos de la consulta.</p>	<p>Concuerda con salida de datos. OK.</p>
<p>Escenario previo:          Usuario Autenticado en el Sistema: Andre Hernandez</p> <hr/> <p>Titular de Consulta: Carnes          Alimento Consultado: Pera</p>	<p>Sistema no permite el ingreso de la consulta para el Usuario Autenticado en el sistema: Andre Hernandez, marca contorno del campo Descripción de Consulta en color rojo indicando mensaje: Ingrese Descripción de Consulta. No se limpian los campos de la pantalla. No se muestra Historial de Consultas con datos de la consulta.</p>	<p>Concuerda con salida de datos, OK.</p>
<p>Escenario previo:          Usuario Autenticado en el Sistema: Andre Hernandez          Consulta previa ingresada en el Sistema y mostrada en Historial de Consultas.</p> <hr/> <p>Se realiza click sobre la Consulta mostrada en la tabla Historial de Consultas.</p>	<p>El sistema muestra en el panel Detalle de Consultas, la información que tiene cargada la Consulta que fue clickeada, sin tener el Usuario Autenticado permiso para modificar esos datos.</p>	<p>Concuerda con salida de datos. OK.</p>

## CP7 - Consulta de Usuarios

Tabla 4.9: Consulta de Usuarios

Entrada de datos	Salida de datos	Resultado esperado
<p>Escenario previo: Profesional Autenticado en el Sistema: Andre Hernandez Consulta ingresada en el Sistema previamente por Usuario: Pedro Gonzalez</p> <hr/> <p>Profesional autenticado realiza click sobre la consulta en la tabla Historial de Consultas.</p>	<p>Sistema carga datos de la consulta clickeada por el profesional mostrando datos especificos de esa consulta en el panel Detalle de Consulta.</p>	<p>Concuerda con salida de datos. OK</p>
<p>Escenario previo: Profesional Autenticado en el Sistema: Andre Hernandez Consulta ingresada en el Sistema previamente por Usuario: Pedro Gonzalez Profesional realiza click sobre la consulta mostrada en la tabla Historial de Consultas</p> <hr/> <p>Profesional completa campo de Respuesta de Profesional para la consulta ingresando: Tiene que realizar comidas variadas a lo largo del día. Finalmente realiza click en guardar Respuesta.</p>	<p>Sistema guarda la siguiente respuesta del Profesional: Tienes que realizar comidas variadas durante el día. para el Usuario que creo la consulta: Pedro Gonzalez. Ademas de esto deja inhabilitado el campo de ingreso de Respuesta. Si Pedro Gonzalez entra con su perfil al sistema, vera que para esta consulta, tiene la respuesta del Profesional: Andre Hernandez, esto lo podra ver realizando click sobre la consulta en Historial de Consultas, en el panel Detalle de Consulta.</p>	<p>Concuerda con salida de datos. OK.</p>
<p>Escenario previo: Profesional Autenticado en el Sistema: Andre Hernandez Consulta ingresada en el Sistema previamente por Usuario: Pedro Gonzalez Profesional realiza click sobre la consulta mostrada en la tabla Historial de Consultas, en el Detalles de Consulta ingresa una Respuesta, guardándola.</p> <hr/> <p>Profesional presiona el boton Editar Respuesta y escribe una nueva respuesta: Carne</p>	<p>Sistema habilita el campo de ingreso de respuesta de Profesional, guardado para esa consulta clickeada creada por el Usuario: Pedro Gonzalez, la siguiente respuesta: Carne. Si Pedro Gonzalez entra con su perfil al sistema, vera que para esa consulta, tiene la respuesta del profesional Andre Hernandez, esto lo podra ver realizado click sobre la consulta en Historial de Consultas, en el panel Detalles de Consulta.</p>	<p>Concuerda con salida de datos, OK.</p>

Tabla 4.10: CP8 - Crear Plan de Alimentación

Entrada de datos	Salida de datos	Resultado esperado
<p>Escenario previo: Profesional Autenticado: Andre Hernandez. Plan de Alimentación solicitado por Usuario: Pedro Gonzalez</p> <hr/> <p>Profesional autenticado realiza click sobre el Plan de Alimentación en la tabla Historial de Planes de Alimentación.</p>	<p>Sistema carga datos del Plan de Alimentación clickeado por el profesional mostrando datos específicos de ese Plan de Alimentación en el panel Detalle de Plan de Alimentación.</p>	<p>Concuerda con salida de datos. OK</p>
<p>Escenario previo: Profesional Autenticado: Andre Hernandez. Plan de Alimentación solicitado: Pedro Gonzalez Profesional realiza click sobre el Plan de Alimentación mostrado en la tabla Historial de Planes de Alimentación.</p> <hr/> <p>Profesional ingresa fechas validas del Plan de Alimentación e ingresa para los 7 días de la semana, sugerencias de Alimentos, pudiendo dejar vacio un día de la semana. Fecha Desde: 20/01/2017 Fecha Hasta: 27/01/2017 Lunes: Arroz - Martes: Pollo Finalmente realiza click en guardar Plan de Alimentación.</p>	<p>Sistema guarda el plan de alimentación sugerido por el Profesional con la siguiente información: Fecha Desde: 20/01/2017 Fecha Hasta: 27/01/2017 Lunes: Arroz - Martes: Pollo para el Usuario solicitante: Pedro Gonzalez. Queda inhabilitado los campos de elaboración de plan de Alimentación. Pedro Gonzalez podrá ver la elaboración con detalles del Plan de Alimentación elaborado, si ingresa a su perfil en el Sistema, clickea el plan de Alimentación solicitado en la tabla Historial de Plan de Alimentación, panel Detalles de Plan de Alimentación.</p>	<p>Concuerda con salida de datos. OK.</p>
<p>Escenario previo: Profesional Autenticado: Andre Hernandez. Plan de Alimentación solicitado por: Pedro Gonzalez Profesional realiza click sobre el Plan de Alimentación mostrado en la tabla Historial de Plan de Alimentación, en el Detalles de Plan de Alimentación formula para los días de la semana, guardándolo.</p> <hr/> <p>Profesional presiona el boton Editar Respuesta y realiza cambios sobre el Plan de Alimentación: Lunes: pollo.</p>	<p>Sistema habilita el campo para formular plan de Alimentación guardado para ese plan de Alimentación clickeado por el Profesional, la siguiente respuesta: Lunes: pollo Si Pedro Gonzalez entra con su perfil al sistema, vera que para ese Plan de Alimentación, tiene la nueva elaboración del profesional Andre Hernandez, esto lo podra ver realizado click sobre el Plan de Alimentación en Historial de Planes de Alimentación, en Detalles de Planes de Alimentación..</p>	<p>Concuerda con salida de datos, OK.</p>

## 4.3. Sesiones de ejecución de pruebas

### CP1 - Registrar Alimento

- **Versión del sistema Aplicación Saludable:**  
Versión Final del producto ubicado en rama Máster
- **Configuración del entorno de trabajo:**  
IDE - NetBenas, versión 8.2  
JDK - 1.8.0-151  
JRE - 1.8.0-151
- **Fecha de realización de prueba:**  
Fecha: 22/11/2017
- **Hora de realización de prueba:**  
Hora: 23:30
- **Tester:**  
André Hernández

### CP1 - Registrar Alimento

Pruebas realizadas en los laboratorios de la Facultad ORT:

- **Versión del sistema Aplicación Saludable:**  
Versión Final del producto ubicado en rama Máster
- **Configuración del entorno de trabajo:**  
IDE - NetBenas, versión 8.2  
JDK - 1.8.0-51  
JRE - 1.8.0-51
- **Fecha de realización de prueba:**  
Fecha: 23/11/2017
- **Hora de realización de prueba:**  
Hora: 11:10
- **Tester:**  
André Hernández

### CP2 - Registrar Usuario

- **Versión del sistema Aplicación Saludable:**  
Versión Final del producto ubicado en rama Máster
- **Configuración del entorno de trabajo:**  
IDE - NetBenas, versión 8.2  
JDK - 1.8.0-151  
JRE - 1.8.0-151

- **Fecha de realización de prueba:**

Fecha: 22/11/2017

- **Hora de realización de prueba:**

Hora: 23:50

- **Tester:**

André Hernández

### **CP2 - Registrar Usuario**

Cuando se realizo esta prueba con el ambiente de trabajo que se describe a continuación se tuvo problemas, **no pudiendo ejecutar** la funcionalidad Registrar Usuario, luego de investigación se debe a que hay problemas con una librería que se uso en el proyecto para el manejo del ingreso de fechas: jcalendar-tz-1.3.3-4.jar.

- **Versión del sistema Aplicación Saludable:**

Versión Final del producto ubicado en rama Máster

- **Configuración del entorno de trabajo:**

IDE - NetBenas, versión 8.2

JDK - 1.8.0-111

JRE - 1.8.0-111

- **Fecha de realización de prueba:**

Fecha: 22/11/2017

- **Hora de realización de prueba:**

Hora: 00:00

- **Tester:**

André Hernández

### **CP2 - Registrar Usuario**

Pruebas realizadas en los laboratorios de la Facultad ORT:

- **Versión del sistema Aplicación Saludable:**

Versión Final del producto ubicado en rama Máster

- **Configuración del entorno de trabajo:**

IDE - NetBenas, versión 8.2

JDK - 1.8.0-51

JRE - 1.8.0-51

- **Fecha de realización de prueba:**

Fecha: 23/11/2017

- **Hora de realización de prueba:**

Hora: 11:25

- **Tester:**  
André Hernández

### **CP3 - Registrar Profesional**

- **Versión del sistema Aplicación Saludable:**  
Versión Final del producto ubicado en rama Máster
- **Configuración del entorno de trabajo:**  
IDE - NetBenas, versión 8.2  
JDK - 1.8.0-151  
JRE - 1.8.0-151
- **Fecha de realización de prueba:**  
Fecha: 23/11/2017
- **Hora de realización de prueba:**  
Hora: 00:40
- **Tester:**  
André Hernández

### **CP3 - Registrar Profesional**

Cuando se realizo esta prueba con el ambiente de trabajo que se describe a continuación se tuvo problemas, **no pudiendo ejecutar** la funcionalidad Registrar Profesional, luego de investigación se debe a que hay problemas con una librería que se uso en el proyecto para el manejo del ingreso de fechas: jcalendar-tz-1.3.3-4.jar.

- **Versión del sistema Aplicación Saludable:**  
Versión Final del producto ubicado en rama Máster
- **Configuración del entorno de trabajo:**  
IDE - NetBenas, versión 8.2  
JDK - 1.8.0-111  
JRE - 1.8.0-111
- **Fecha de realización de prueba:**  
Fecha: 23/11/2017
- **Hora de realización de prueba:**  
Hora: 01:00
- **Tester:**  
André Hernández

### **CP3 - Registrar Profesional**

Pruebas realizadas en los laboratorios de la Facultad ORT:

- **Versión del sistema Aplicación Saludable:**  
Versión Final del producto ubicado en rama Máster

- **Configuración del entorno de trabajo:**

IDE - NetBenas, versión 8.2

JDK - 1.8.0-51

JRE - 1.8.0-51

- **Fecha de realización de prueba:**

Fecha: 23/11/2017

- **Hora de realización de prueba:**

Hora: 11:50

- **Tester:**

André Hernández

#### **CP4 - Registrar Ingesta**

Cuando se realizo esta prueba con el ambiente de trabajo que se describe a continuación se tuvo problemas, **no pudiendo ejecutar** la funcionalidad Registrar Ingesta, luego de investigación se debe a que hay problemas con una librería que se uso en el proyecto para el manejo del ingreso de fechas: jcalendar-tz-1.3.3-4.jar.

- **Versión del sistema Aplicación Saludable:**

Versión Final del producto ubicado en rama Máster

- **Configuración del entorno de trabajo:**

IDE - NetBenas, versión 8.2

JDK - 1.8.0-111

JRE - 1.8.0-111

- **Fecha de realización de prueba:**

Fecha: 23/11/2017

- **Hora de realización de prueba:**

Hora: 01:00

- **Tester:**

André Hernández

#### **CP4 - Registrar Ingesta**

- **Versión del sistema Aplicación Saludable:**

Versión Final del producto ubicado en rama Máster

- **Configuración del entorno de trabajo:**

IDE - NetBenas, versión 8.2

JDK - 1.8.0-151

JRE - 1.8.0-151

- **Fecha de realización de prueba:**

Fecha: 23/11/2017



- **Hora de realización de prueba:**

Hora: 01:20

- **Tester:**

André Hernández

#### **CP4 - Registrar Ingesta**

Pruebas realizadas en los laboratorios de la Facultad ORT:

- **Versión del sistema Aplicación Saludable:**

Versión Final del producto ubicado en rama Máster

- **Configuración del entorno de trabajo:**

IDE - NetBenas, versión 8.2

JDK - 1.8.0-51

JRE - 1.8.0-51

- **Fecha de realización de prueba:**

Fecha: 23/11/2017

- **Hora de realización de prueba:**

Hora: 11:38

- **Tester:**

André Hernández

#### **CP5 - Sugerencia de Plan de Alimentación**

- **Versión del sistema Aplicación Saludable:**

Versión Final del producto ubicado en rama Máster

- **Configuración del entorno de trabajo:**

IDE - NetBenas, versión 8.2

JDK - 1.8.0-151

JRE - 1.8.0-151

- **Fecha de realización de prueba:**

Fecha: 23/11/2017

- **Hora de realización de prueba:**

Hora: 01:40

- **Tester:**

André Hernández

#### **CP5 - Sugerencia de Plan de Alimentación**

Pruebas realizadas en los laboratorios de la Facultad ORT:

- **Versión del sistema Aplicación Saludable:**

Versión Final del producto ubicado en rama Máster

■ **Configuración del entorno de trabajo:**

IDE - NetBenas, versión 8.2

JDK - 1.8.0-51

JRE - 1.8.0-51

■ **Fecha de realización de prueba:**

Fecha: 23/11/2017

■ **Hora de realización de prueba:**

Hora: 11:50

■ **Tester:**

André Hernández

**CP6 - Consulta con Profesional**

■ **Versión del sistema Aplicación Saludable:**

Versión Final del producto ubicado en rama Máster

■ **Configuración del entorno de trabajo:**

IDE - NetBenas, versión 8.2

JDK - 1.8.0-151

JRE - 1.8.0-151

■ **Fecha de realización de prueba:**

Fecha: 23/11/2017

■ **Hora de realización de prueba:**

Hora: 02:30

■ **Tester:**

André Hernández

**CP6 - Consulta con Profesional**

Pruebas realizadas en los laboratorios de la Facultad ORT:

■ **Versión del sistema Aplicación Saludable:**

Versión Final del producto ubicado en rama Máster

■ **Configuración del entorno de trabajo:**

IDE - NetBenas, versión 8.2

JDK - 1.8.0-51

JRE - 1.8.0-51

■ **Fecha de realización de prueba:**

Fecha: 23/11/2017

■ **Hora de realización de prueba:**

Hora: 12:10

- **Tester:**  
André Hernández

#### **CP7 - Consulta de Usuarios**

- **Versión del sistema Aplicación Saludable:**  
Versión Final del producto ubicado en rama Máster
- **Configuración del entorno de trabajo:**  
IDE - NetBenas, versión 8.2  
JDK - 1.8.0-151  
JRE - 1.8.0-151
- **Fecha de realización de prueba:**  
Fecha: 23/11/2017
- **Hora de realización de prueba:**  
Hora: 03:00
- **Tester:**  
André Hernández

#### **CP7 - Consulta de Usuarios**

Pruebas realizadas en los laboratorios de la Facultad ORT:

- **Versión del sistema Aplicación Saludable:**  
Versión Final del producto ubicado en rama Máster
- **Configuración del entorno de trabajo:**  
IDE - NetBenas, versión 8.2  
JDK - 1.8.0-51  
JRE - 1.8.0-51
- **Fecha de realización de prueba:**  
Fecha: 23/11/2017
- **Hora de realización de prueba:**  
Hora: 12:30
- **Tester:**  
André Hernández

#### **CP8 - Crear Plan de Alimentación**

- **Versión del sistema Aplicación Saludable:**  
Versión Final del producto ubicado en rama Máster
- **Configuración del entorno de trabajo:**  
IDE - NetBenas, versión 8.2  
JDK - 1.8.0-151  
JRE - 1.8.0-151

- **Fecha de realización de prueba:**

Fecha: 23/11/2017

- **Hora de realización de prueba:**

Hora: 04:00

- **Tester:**

André Hernández

### **CP8 - Crear Plan de Alimentación**

Pruebas realizadas en los laboratorios de la Facultad ORT:

- **Versión del sistema Aplicación Saludable:**

Versión Final del producto ubicado en rama Máster

- **Configuración del entorno de trabajo:**

IDE - NetBeans, versión 8.2

JDK - 1.8.0-51

JRE - 1.8.0-51

- **Fecha de realización de prueba:**

Fecha: 23/11/2017

- **Hora de realización de prueba:**

Hora: 13:50

- **Tester:**

André Hernández

## 5. Reporte de defectos

### 5.1. Definición de categorías de defectos

En esta sección se definen las categorías de defectos que se van a usar en el reporte, conjunto a su severidad [12].

#### Categorías de defectos:

- Defectos de Interfaz
- Defecto de Lógica del sistema
- Defecto de comunicación entre clases
- Defecto de usabilidad

#### Severidad de defectos:

- **Alta:** Bugs que se consideran que alteran de forma importante o detienen los flujos principales del sistema. Cabe destacar que uno de los factores claves por los que un sistema puede o no desplegarse en producción es debido a los incidentes de criticidad alta, por lo que es muy importante tener claro porque tiene esa clasificación. Por otra parte en caso de un mantenimiento del software serán los primeros incidentes en ser tratados, teniendo como finalidad llegar a la solución del bug reportado.
- **Media:** Las funcionalidades principales del sistema se ejecutan de forma completa, los reportes e interfaces se obtienen sin ningún fallo, pero hay que ejecutar más pasos de los necesarios para obtener el resultado final; falta de controles y validaciones.
- **Baja:** Orientados a presentación, diseño y navegabilidad, este tipo de bugs se tratan más de aquellos que no afectan ni un flujo principal del sistema, ni un flujo alternativo pero que puede ser algo molesto en la mayoría de los casos.

## 5.2. Defectos encontrados por iteración

### Titulo: Borrado de campos en pantalla Registro Alimento

- **Descripción:** Cuando se quiere ingresar un nuevo alimento en el sistema, que ya esta registrado previamente; el sistema no permite el ingreso del alimento, lo cual es correcto, pero borra los campos de la pantalla, lo que el sistema espera es que indique en color rojo el contorno del campo Nombre de Alimento, pero que no borre los datos de la pantalla del Registro de Alimento.
- **Severidad:** Media.
- **Categoría:** Defectos de Interfaz - Defecto de usabilidad.
- **Estado:** No corregido.

### Titulo: Ingreso de Consulta con Profesional - Campos de Ingreso

- **Descripción:** Cuando se quiere ingresar una Consulta, si en primer lugar se intenta ingresar dejando vacío el campo Titular de Consulta, el mismo queda en rojo y es correcto, pero en una segunda instancia, se completa el campo Titular de Consulta y se deja vacío el campo de Descripción de Consulta, intentando ingresar la consulta nuevamente. Aquí se puede ver el defecto, el campo de Titular de Consulta que fue completado sigue con el contorno en color rojo, el mismo tendría que estar validado correctamente, marcando solo el contorno rojo el campo que esta vacío en este caso lo es el de Descripción de la Consulta.
- **Severidad:** Media.
- **Categoría:** Defectos de Interfaz - Defecto de usabilidad
- **Estado:** No corregido.

### Titulo: Perfil de Usuarios y Profesionales

- **Descripción:** Se deben hacer varios pasos para poder acceder a un perfil de Usuario o Profesional.
- **Severidad:** Alta.
- **Categoría:** Defectos de Interfaz - Defecto de usabilidad
- **Estado:** No corregido.

### **Título: Registro de Ingestas de Alimentos**

- **Descripción:** Cuando se registra una Ingesta de alimento para el Usuario, se selecciona el nombre de alimento y la fecha de la ingesta; el sistema deja cargar para la misma fecha la ingesta de un mismo alimento, esto no debería ser así, realizando un control fecha de ingesta - alimento ingerido.
- **Severidad:** Alta.
- **Categoría:** Defectos de Interfaz - Defecto de usabilidad
- **Estado:** No corregido.

### **Título: Registro de Ingestas de Alimentos**

- **Descripción:** Cuando se Registra una Ingesta de alimento para el Usuario, se ingresa el nombre de alimento y la fecha de la ingesta; si se ingresa a la funcionalidad de Registrar Alimento y se registrar en el sistema un nuevo alimento, al volver a la funcionalidad Registro de Ingesta, el combo en donde están cargados todos los alimentos ingresados en el sistema, al hacer click demora en refrescarse, se deben realizar varios click para que se actualice con los nuevos datos del sistema.
- **Severidad:** Media.
- **Categoría:** Defectos de Interfaz - Defecto de usabilidad - Defecto de Lógica del sistema - Defecto de comunicación entre clases.
- **Estado:** No corregido.

## **5.3. Estado de calidad global**

A lo largo del proyecto se viene trabajando con firmeza en todo lo que tiene que ver con un sistema profesional y lo que implica llegar a conseguirlo, comenzando por las **Pruebas Unitaria**, luego por **Pruebas Funcionales** y finalmente elaborando un **Reporte de Defectos** del sistema se puede concluir que el sistema Aplicación Saludable presenta un estado de software bueno, si bien tiene varias mejoras para realizar, como por ejemplo en su usabilidad, su funcionamiento es optimo para el usuario.

## 6. Reflexión

La finalidad de este proyecto es desarrollar una aplicación que permita a los usuarios seguir un esquema de alimentación saludable avalado por profesionales de la alimentación; utilizando practicas tecnológicas y de gestión de la ingeniera de software, obteniendo como resultado un software de calidad y la aplicación de un conjunto de practicas profesionales.

Para llevar a cambo este proyecto en primera instancia se comenzó investigando sobre la codificación de sistema profesionales, para tener una idea mas robusta de como comenzar a diseñar nuestro sistema, Aplicación Saludable. Desde los inicios de la carrera se viene codificando código y también interfaces de usuarios, pero nunca teniendo en cuenta que tan importante es es la interfaz de usuario para los usuarios, nos cuesta ubicarnos en la cabeza del usuario, como programadores que somos, pero esta tarea es muy importante a la hora de diseñar un sistema, ya que los que determinan el éxito del desarrollo del software son los usuarios. Se investigaron buenas practicas con respecto a la usabilidad y diseños de interfaz para usuarios, se logro comprender la importancia de una interfaz de usuario, las características que debe cumplir y como hacer para que la aplicación tenga un impacto positivo sobres los usuarios sin demandar una gran exigencia para el uso de las funcionalidades del sistema. Se logro ubicarse por unos instantes en la cabeza de un usuario, el cual usaría Aplicación Saludable, de esta forma se logro comprender aun mas lo importante que es la usabilidad de un sistema, la cantidad de pasos que debo realizar para acceder a determinada funcionalidad del sistema, colores, mensajes de aviso que molestan la visión del usuario y lo desencajan del sistema, muchas características que quizá en algún otro momento no eran tan importantes a la hora de codificar un sistema, hoy en día se logro comprender que son de vital importancia.

El uso de un correcto estándar de codificación, si bien se han aprendido hasta el momento técnicas de como escribir un código legible y prolijo, en este proyecto se aplica un estándar de programación profesional, en este caso del lenguaje JAVA, también comprendiendo la importancia que tiene seguir dichos estándares, ya que si se quiere realizar un cambio en el código en cualquier parte del sistema, el correcto uso de un estándar de codificación facilitaría la compresión del código y se emplearía menos pienso y tiempo para efectuar el cambio.

Destaco también el uso de los repositorios remotos y locales, el uso de la herramienta GIT, que hasta el momento la forma que se empleaba para el archivado de versiones de un sistema no era muy buena, con GIT el desarrollador se asegura de



tener su código respaldado en un servidor teniendo un log de versiones, accediendo a commits realizados con sus respectivos comentarios, teniendo la opción de volver atrás en alguna versión si surgió algún problema, también facilitado el trabajo en equipo. Acá vuelvo a mencionar la importancia de seguir un correcto estándar de codificación para que los integrantes del equipo, no importa el cambio del sistema que se realice, si sigue el estándar de codificación, los integrantes podrán comprender el cambio sin dificultades.

Las pruebas unitarias y pruebas funcionales, estas secciones del proyecto son las que mas interesaron, conjunto a los reportes de defectos, ya que en mi caso no tenia mucho conocimiento sobre el área de testing. Tanto las pruebas unitarias como las pruebas funcionales son vitales a la hora de diseñar un sistema, en el ámbito laboral profesional se dedican muchas horas al testing de aplicaciones en busca de bugs y situaciones problemáticas. Estas pruebas son las que pueden llegar a descubrir errores en el código del sistema, descubrir que algo que se pensaba que funciona perfecto, no funciona como lo esperado. Por otra parte nos ubicamos en la cabeza de un tester, lo cual no es fácil en primera instancia, codificando pruebas, pero se pudo superar sin problemas. Con respecto a los reportes de defectos, un sistema siempre tiene defectos, el buen tester es el que encuentra mas defectos, luego de encontrados estos defectos requiere un cambio en el código y funcionalidades del sistema para corregir estos defectos, luego se realizan nuevamente pruebas unitarias y funcionales, es una cadena que lo que intenta hacer es asegurar la calidad global de software.

Para concluir, destaco el aprendizaje de buenas técnicas de Ingeniera de Software a la hora del diseño de un sistema, espero que este documento refleje las técnicas aplicadas con su correcta justificación.

Finalmente el diseño de Aplicación Saludable me hizo sentir por momentos desarrollador, por otros tester y sobre todo Usuario, ubicándome en diversas situaciones, lo cual me ayudo al desarrollo de un sistema robusto, estandarizado y con buenas practicas de usabilidad.

# Bibliografía

- [1] Merkury. Estándar de codificación.
- [2] S. M. I. Scott Hommel. Convenciones de código para el lenguaje de programación java.
- [3] W. L. E. libre. Junit.
- [4] L. y Rieman [1993]. Importancia de las interfaces de usuarios.
- [5] B. Roe. Diseño de interfaces de usuario usables: Una guía rápida para desarrolladores de software libre y de código abierto.
- [6] AGESIC. Usabilidad.
- [7] D. L. P. C. I. M. M. Bergues. Usabilidad, los métodos y las técnicas para la evaluación.
- [8] A. Mocholí. Diseño de apps y la importancia del color.
- [9] B. Martinez. La lógica de la usabilidad: las 10 heurísticas de jakob nielsen.
- [10] W. L. E. libre. Pruebas funcionales.
- [11] G. Terrera. Pruebas de caja negra.
- [12] C. C. S. de C.V. La clasificación de incidencias en el proceso de testing de software.