

Universidad ORT Uruguay

Facultad de Ingeniería

Ingeniería de Software 1

Letra del Obligatorio 2

Santiago Toscanini

Sofía Tejerina

Entregado como requisito de la materia Ingeniería de
Software 1

25 de noviembre de 2019

Declaraciones de autoría

Nosotros, Santiago Toscanini y Sofía Tejerina , 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 1;
- 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

Este documento detalla información acerca de cómo desarrollamos una aplicación llamada EcoFood, que funciona a modo de aplicación de pedido de alimentos (PedidosYa, Rappi, Uber Eats, Glovo, etc). La aplicación está enfocada en una tienda que desea vender productos sin dañar al medio ambiente, esta misma será una web, donde los usuarios podrán acceder y realizar diversas acciones, como realizar una pre-venta de alimentos, realizar pedidos, imprimir sus facturas, entre otras.

Índice general

1. Versionado	2
1.1. Repositorio Utilizado	2
1.2. Criterio de versionado	4
1.3. Resumen de log de versiones	4
2. Codificación	8
2.1. Estandar de codificación	8
2.2. Pruebas unitarias	10
2.3. Análisis de código	17
3. Interfaz de usuario y usabilidad	18
3.1. Criterios de interfaz de usuario	18
3.2. Evaluación de usabilidad	19
4. Pruebas funcionales	22
4.1. Técnicas de prueba aplicadas	22
4.2. Casos de prueba	22
4.3. Sesiones de ejecución de pruebas	25
5. Reporte de defectos	34
5.1. Definición de categorías de defectos	34
5.2. Defectos encontrados por iteración	34
5.3. Estado de calidad global	36
6. Reflexión	37

1. Versionado

1.1. Repositorio Utilizado

Los sistemas de control de versiones son programas que tienen como objetivo controlar los cambios en el desarrollo de cualquier tipo de software, permitiendo conocer el estado actual de un proyecto, los cambios que se le han realizado a cualquiera de sus piezas, las personas que intervinieron en ellos, etc. Surge de la necesidad de mantener y llevar control del código que vamos programando, conservando sus distintos estados.

El versionado de este proyecto fue realizado en la plataforma online de GitHub y Git a nivel local.

Git

Es un sistema de control de versiones desarrollado por Linus Torvalds (el hombre que creó Linux). El sistema de control de versiones ayuda a registrar los cambios realizados al código. Registra quién realizó los cambios y puede restaurar el código borrado o modificado. Es un sistema de código abierto, relativamente nuevo que nos ofrece las mejores características en la actualidad.

Es multiplataforma, por lo que puede ser usado y crear repositorios locales (una copia local del código generado) en todos los sistemas operativos más comunes, Windows, Linux o Mac. Existen multitud de GUIs (Graphical User Interface o Interfaz de Usuario Gráfica) para trabajar con Git, no obstante para el aprendizaje se recomienda usarlo con línea de comandos, como lo usamos nosotros.

- Permite comparar el código de un archivo, de modo que podamos ver las diferencias entre versiones.
- Restaurar versiones antiguas.
- Fusionar cambios entre distintas versiones.
- Trabajar con distintas ramas de un proyecto, por ejemplo la de producción y desarrollo.

Es un sistema distribuido; en este tipo de sistemas cada uno de los integrantes del equipo mantiene una copia local del repositorio completo. Al disponer de un

repositorio local, se puede hacer commit (enviar cambios al sistema de control de versiones) en local, sin necesidad de estar conectado a Internet o cualquier otra red.

GitHub

GitHub es un servicio para hacer hosting de repositorios de software que se administra con Git. Digamos que en GitHub mantienes una copia de tus repositorios en la nube.

Es un sistema de gestión de proyectos y control de versiones de código, así como una plataforma de red social diseñada para desarrolladores.

Es un sitio web que usa Git para ofrecer a la comunidad de desarrolladores repositorios de software. En definitiva, es un sitio web pensado para hacer posible el compartir el código de una manera más fácil.

Ademas de alojar tu repositorio de código, te brinda herramientas muy útiles para el trabajo en equipo, dentro de un proyecto. Permite trabajar en colaboración con otras personas de todo el mundo, planificar proyectos y realizar un seguimiento del trabajo.

GitHub es también uno de los repositorios online más grandes de trabajo colaborativo en todo el mundo.

En él es gratuito alojar proyectos Open Source, lo que ha posibilitado que el número de proyectos no pare de crecer, y en estos momentos haya varios millones de repositorios y usuarios trabajando con esta herramienta.¹

En el repositorio guardaremos programas (código fuente y ejecutables), recursos (interfaz, gráficos), documentos (técnicos, administrativos) y estructuras de datos (base de datos, esquemas, archivos).

No usaremos herramientas de apoyo como Mantis debido a que es un proyecto muy pequeño y no tiene sentido usarla para el seguimiento de incidentes y solicitudes de modificaciones. Tampoco usaremos Jenkins, Hudson (trabajan con herramientas de control de versiones como CVS, Subversion, Git y Clearcase) para integrar cambios y detectar posibles errores antes de que ocurran, por la misma razón que explicamos antes, e sun proyecto muy pequeño todavía y lejos de ayudar a la fluidez correcta del proyecto hara que sea más torpe y dificulte su desarrollo.

¹<https://github.com/santiagotoscanini/Obligatorio.IS-Toscanini-Tejerina>

1.2. Criterio de versionado

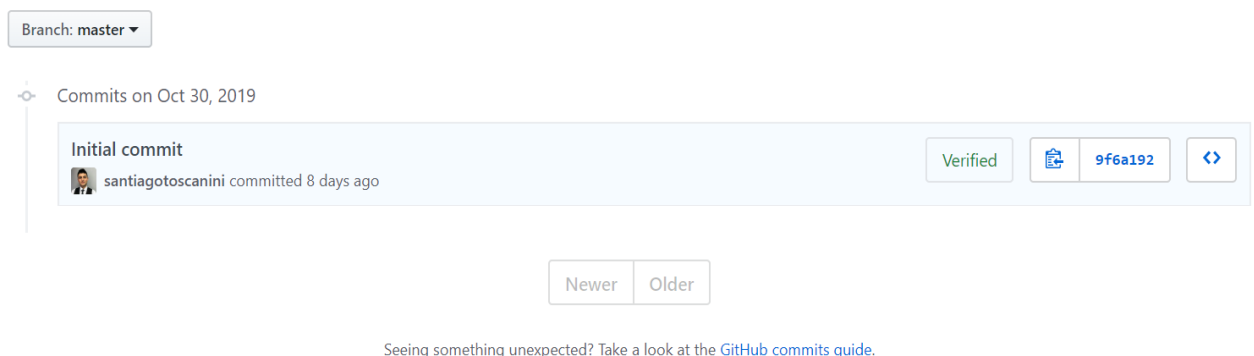
Un buen log de versión es fundamental para un proyecto, ya sea a gran escala como un proyecto de una o dos personas, como es este caso. Aplicar un buen uso de los recursos de control de versionado ayuda a identificar piezas de código, controlar modificaciones (propias o de los compañeros de proyecto) y registrar el estado de cada pieza de código y registrar las modificaciones. Aseguraremos la completitud, consistencia y correctitud del software. Nos aseguraremos de que no haya pérdida de información ni tiempo invertido.

Respecto al criterio de versionado, utilizamos dos ramas, master y develop, dentro de master colocaremos el código estable, y dentro de develop iremos desarrollando las nuevas funcionalidades para nuestro sistema, también se podría haber utilizado feature branch y así tener un mejor control de cada feature de nuestro sistema.













Respecto a los comentarios de los commits, intentamos que sean lo más descriptivos posible, dejando así la posibilidad de únicamente viendo el comentario del commit identificar que sección del programa se modificó.

1.3. Resumen de log de versiones

Primer commit en la rama Master GitHub



Commits en la rama Develop GitHub

Commits on Nov 5, 2019	
<p>Agregue la clase alimento, Envase y Factura, ademas agregue un nuevo ...</p> <p> SofiaTejerina committed 2 days ago</p>	<p> 3735451 </p>
<p>Adelanto de interfaz de pre-venta, falta mejorar el diseño y hacer la...</p> <p> SofiaTejerina committed 2 days ago</p>	<p> 0a80322 </p>
<p>Adelanto de pagina principal, se agregaron paneles de categorias, se ...</p> <p> SofiaTejerina committed 2 days ago</p>	<p> c926116 </p>
<p>Se crea pantalla principal para la aplicacion, y la clase base que ut...</p> <p> santiagotoscanini committed 3 days ago</p>	<p> 08c68d5 </p>
Commits on Oct 30, 2019	
<p>Initial commit</p> <p> santiagotoscanini committed 8 days ago</p>	<p>Verified  9f6a192 </p>

Commit en la consola de Git

```
$ git commit -m 'Sea realizan las pruebas unitarias de la clases de modelos'
[develop bb6f686] Sea realizan las pruebas unitarias de la clases de modelos
22 files changed, 1112 insertions(+), 808 deletions(-)
create mode 100644 lib/hamcrest/hamcrest-core-1.3.jar
create mode 100644 lib/junit_4/junit-4.12-javadoc.jar
create mode 100644 lib/junit_4/junit-4.12-sources.jar
create mode 100644 lib/junit_4/junit-4.12.jar
rewrite nbproject/private/private.xml (84%)
rewrite test/Modelos/AlimentoTest.java (69%)
create mode 100644 test/Modelos/ElementoCarritoTest.java
rewrite test/Modelos/FacturaTest.java (70%)
create mode 100644 test/Modelos/ParTest.java
rewrite test/Modelos/SistemaTest.java (72%)
rewrite test/Modelos/SucursalTest.java (74%)
rewrite test/Modelos/UsuarioTest.java (77%)
```

Push en la consola de Git

```
$ git push origin develop
Enumerating objects: 63, done.
Counting objects: 100% (63/63), done.
Delta compression using up to 4 threads
Compressing objects: 100% (34/34), done.
Writing objects: 100% (36/36), 1.27 MiB | 977.00 KiB/s, done.
Total 36 (delta 20), reused 0 (delta 0)
remote: Resolving deltas: 100% (20/20), completed with 19 local objects.
To github.com:santiagotoscanini/Obligatorio_IS-Toscanini-Tejerina.git
e29169c..bb6f686 develop -> develop
```


git status antes del add

```
Sofi@DESKTOP-UP77PNV MINGW64 ~/Documents/NetBeansProjects/Obligatorio_IS-Toscani
ni-Tejerina (develop)
$ git status
On branch develop
Your branch is up to date with 'origin/develop'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   build/built-jar.properties
        modified:   lib/nblibraries.properties
        modified:   nbproject/private/private.properties
        modified:   nbproject/private/private.xml
        modified:   nbproject/project.properties
        modified:   pdf/factura_0.pdf
        modified:   src/Modelos/ElementoCarrito.java
        modified:   src/Modelos/Factura.java
        modified:   src/Modelos/Sistema.java
        modified:   src/Modelos/Usuario.java
        modified:   src/Vistas/Main.java
        modified:   test/Modelos/AlimentoTest.java
        modified:   test/Modelos/FacturaTest.java
        modified:   test/Modelos/SistemaTest.java
        modified:   test/Modelos/SucursalTest.java
        modified:   test/Modelos/UsuarioTest.java

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        lib/hamcrest/
        lib/junit_4/
        test/Modelos/ElementoCarritoTest.java
        test/Modelos/ParTest.java

no changes added to commit (use "git add" and/or "git commit -a")
```

git status después del add

```
Sofi@DESKTOP-UP77PNV MINGW64 ~/Documents/NetBeansProjects/Obligatorio_IS-Toscanini-Tejerina (develop)
$ git status
On branch develop
Your branch is up to date with 'origin/develop'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   build/built-jar.properties
        new file:   lib/hamcrest/hamcrest-core-1.3.jar
        new file:   lib/junit_4/junit-4.12-javadoc.jar
        new file:   lib/junit_4/junit-4.12-sources.jar
        new file:   lib/junit_4/junit-4.12.jar
        modified:   lib/nblibraries.properties
        modified:   nbproject/private/private.properties
        modified:   nbproject/private/private.xml
        modified:   nbproject/project.properties
        modified:   pdf/factura_0.pdf
        modified:   src/Modelos/ElementoCarrito.java
        modified:   src/Modelos/Factura.java
        modified:   src/Modelos/Sistema.java
        modified:   src/Modelos/Usuario.java
        modified:   src/Vistas/Main.java
        modified:   test/Modelos/AlimentoTest.java
        new file:   test/Modelos/ElementoCarritoTest.java
        modified:   test/Modelos/FacturaTest.java
        new file:   test/Modelos/ParTest.java
        modified:   test/Modelos/SistemaTest.java
        modified:   test/Modelos/SucursalTest.java
        modified:   test/Modelos/UsuarioTest.java
```

git log en consola de Git

```

$ git log
commit bb6f6869a5e7a3b142418c293c007d5ff36501cf (HEAD -> develop, origin/develop)
Author: Sofia Tejerina <sofiatejerina23@gmail.com>
Date: Mon Nov 25 11:19:59 2019 -0300

    Sea realizan las pruebas unitarias de la clases de modelos

commit e29169c286e530040776645eae998e64fe4ffb77 (tag: v1.0)
Author: santiago toscanini <toscaninisantiago@gmail.com>
Date: Sun Nov 24 00:07:05 2019 -0300

    Funciona la opcion de ver mas vendidos, se arregla error al comprar producto desde el carrito

commit 0c3dffd9505d33c7053d0dbf687ae56abd7ab313
Author: santiago toscanini <toscaninisantiago@gmail.com>
Date: Sat Nov 23 18:52:53 2019 -0300

    Esta terminado la creacion del pdf de la factura

commit 9435955faff20c763c7f49f9c12a883d87e973a9
Author: santiago toscanini <toscaninisantiago@gmail.com>
Date: Wed Nov 20 00:07:59 2019 -0300

```

Registro de log en la consola de Git

```

ni-Tejerina (develop)
$ git log --oneline --graph
* d4b3673 (HEAD -> develop, origin/develop) Se agregan paneles para las diferentes acciones, se cambia la clase Sistema a una clase con atributos estaticos
* beb3c9c Se agregan todas las clases de la logica de negocio
* d0a11fe Cambios en clase Alimento, se crea carpeta de test
* c64f0fe Se agregan imagenes y se termina de cambiar los nombres de los componentes
* 3735451 Agregue la clase alimento, Envase y Factura, ademas agregue un nuevo atributo y metodos a la clase usuario (lista de facturas y metodos de modificacion de las listas)
* 0a80322 Adelanto de interfaz de pre-venta, falta mejorar el diseño y hacer la implementacion
* c926116 Adelanto de pagina principal, se agregaron paneles de categorias, se agregan paneles base para preventa, se renombraron elementos del JFrame
* 08c68d5 Se crea pantalla principal para la aplicacion, y la clase base que utilizaran los usuarios
* 9f6a192 (origin/master, origin/HEAD, master) Initial commit

```

2. Codificación

2.1. Estandar de codificación

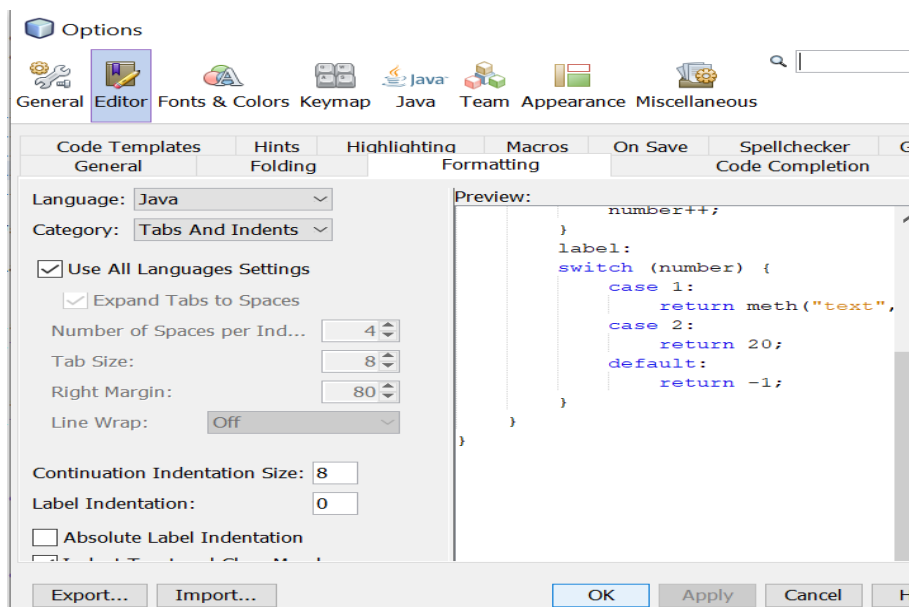
Como estandar de codificación usaremos Google Java Style¹ que se centra principalmente en las reglas rígidas y rápidas que seguimos universalmente.

Algunos ejemplos ilustrados con el código del proyecto 3R EcoShop

Caracteres de espacios en blanco

Además de la secuencia del terminador de línea, el carácter de espacio horizontal ASCII (0x20) es el único carácter de espacio en blanco que aparece en cualquier parte de un archivo fuente. Esto implica que:

- Todos los demás caracteres de espacios en blanco en cadenas y literales de caracteres se escapan.
- Los caracteres de tabulación no se usan para sangría.



¹<https://google.github.io/styleguide/javaguide.html>

Estructura del archivo fuente

Un archivo fuente consta de, en orden :

- Información de licencia o copyright, si está presente
- Declaración del paquete
- Declaraciones de importación
- Exactamente una clase de nivel superior
- Exactamente una línea en blanco separa cada sección que está presente.

Pedido y espaciado

Las importaciones se ordenan de la siguiente manera:

- Todas las importaciones estáticas en un solo bloque.
- Todas las importaciones no estáticas en un solo bloque.
- Si hay importaciones tanto estáticas como no estáticas, una sola línea en blanco separa los dos bloques.
- No hay otras líneas en blanco entre las declaraciones de importación.

```
1 package Modelos;
2
3 import static org.junit.Assert.assertEquals;
4 import static org.junit.Assert.fail;
5
6 import java.util.List;
7 import org.junit.After;
8 import org.junit.AfterClass;
9 import org.junit.Before;
10 import org.junit.BeforeClass;
11 import org.junit.Test;
12
13 public class UsuarioTest {
```

Sangría de bloque: +2 espacios

Cada vez que se abre un nuevo bloque o construcción similar a un bloque, la sangría aumenta en dos espacios. Cuando finaliza el bloque, la sangría vuelve al nivel de sangría anterior.

Una declaración por línea

Cada declaración es seguida por un salto de línea.

```
8     private String direccion;  
9     private int telefono;  
10    private final List<Factura> facturas;  
11
```

Una variable por declaración

Cada declaración de variable (campo o local) declara solo una variable: declaraciones como las que `int a, b;` no se utilizan.

Excepción: se aceptan múltiples declaraciones de variables en el encabezado de un `for` bucle.

2.2. Pruebas unitarias

Con el fin de asegurar una mejor calidad de código, recurrimos al uso de pruebas unitarias usando JUnits, las cuales deben ser simples y cortas porque buscamos probar una única cosa.

JUnits es un conjunto de bibliotecas que son utilizadas en programación para hacer pruebas unitarias en aplicaciones del lenguaje Java. Es un framework que permite evaluar si el funcionamiento de cada uno de los métodos de una clase se comporta de la manera esperada. En función de algún valor de entrada se evalúa el valor de retorno esperado, si la clase cumple con la especificación, significa que la clase pasó con éxito la prueba, de lo contrario se devolverá el fallo en el método correspondiente.

Es muy útil para controlar las pruebas de regresión, necesarias cuando una parte del código ha sido modificado y se desea comprobar si la clase cumple con los requerimientos anteriores y no se ha alterado su funcionalidad.

Aplicación de pruebas unitarias

Las pruebas unitarias se deben mantener en paralelo al código, respetando el estándar y las buenas prácticas de codificación mencionadas anteriormente.

- Uso correcto de los `assert`.
- Pruebas independientes unas de otras.
- Una prueba para cada función.

- No se exponen a probar implementaciones internas.

Las utilizaremos para encontrar errores en el funcionamiento de las funciones (de forma aislada), una por una y, eventualmente, conjuntos de funciones que se relacionan entre si.

Tipos de Assert

- `assertTrue(boolean test)`
`assertTrue(String message, boolean test)`
- `assertFalse(boolean test)`
`assertFalse(String message, boolean test)`
- `assertEquals(expected, actual)`
`assertEquals(String message, expected, actual)`
- `assertSame(Object expected, Object actual)`
`assertSame(String message, Object expected, Object actual)`
- `assertNotSame(Object expected, Object actual)`
`assertNotSame(String message, Object expected, Object actual)`
- `assertNull(Object object)`
`assertNull(String message, Object object)`
- `assertNotNull(Object object)`
`assertNotNull(String message, Object object)`
- `fail()`
`fail(String message)`

Las pruebas unitarias no dan un estado global de calidad, por lo cual complementamos con Pruebas Funcionales para generar un producto lo más adecuado posible a la usabilidad y las necesidades de los usuarios, que cumpliera con los requisitos funcionales establecidos.

Probamos las clases de Descripción del alimento

Datos válidos	Datos no válidos
Strings	int
	double
	float

Tabla 2.1: Clases de "Descripción" @

Entradas	Salidas Esperadas
Calabaza, apio, ajo, sal, condimento verde"	
50	"La descripción debe ser un String"
50.2	"La descripción debe ser un String"
43.23	"La descripción debe ser un String"

Tabla 2.2: Clases de "Descripción"@

Probamos las clases de Precio del alimento

Datos válidos	Datos no válidos
int	double
	float
	String

Tabla 2.3: Clases de "Precio"@

Entradas	Salidas Esperadas
69	
50.2	. ^{EI} precio debe ser un int"
50.2	. ^{EI} precio debe ser un int"
43.23	. ^{EI} precio debe ser un int"

Tabla 2.4: Clases de "Precio"@

Probamos las clases de Puntos de venta

Datos válidos	Datos no válidos
Strings	int
	double
	float

Tabla 2.5: Clases de "Dirección"@

Entradas	Salidas Esperadas
"18 de julio, 2341"	
53	"Dirección inválida"
56.2	"Dirección inválida"
42.23	"Dirección inválida"

Tabla 2.6: Clases de "Descripción"@

Datos válidos	Datos no válidos
Strings	int
	double
	float

Tabla 2.7: Clases de "Dirección" @

Entradas	Salidas Esperadas
"18 de julio, 2341"	
53	"Dirección inválida"
56.2	"Dirección inválida"
42.23	"Dirección inválida"
extra:	"Dirección inválida"

Tabla 2.8: Clases de "Descripción" @

Datos válidos	Datos no válidos
int	String
	double
	float

Tabla 2.9: Clases de "Teléfono" @

Entradas	Salidas Esperadas
4525678493	
"56565"	"Teléfono inválido"
56.2	"Teléfono inválido"
42.23	"Teléfono inválido"

Tabla 2.10: Clases de "Teléfono" @

Probamos Crear un Usuario

Datos válidos	Datos no válidos
Strings	int
	double
	float

Tabla 2.11: Clases de "Nombre" @

Entradas	Salidas Esperadas
Çarla Diaz"	
50	"Nombre inválido"
50.2	"Nombre inválido"
43.23	"Nombre inválido"
	"Nombre inválido"

Tabla 2.12: Clases de "Nombre"@

Datos válidos	Datos no válidos
Strings	int
	double
	float

Tabla 2.13: Clases de Çédula"@

Entradas	Salidas Esperadas
"1987695-0"	
50	Çédula inválida"
50.2	Çédula inválida"
43.23	Çédula inválida"
	Çédula inválida"

Tabla 2.14: Clases de Çédula"@

Datos válidos	Datos no válidos
Strings	int
	double
	float

Tabla 2.15: Clases de "Dirección"@

Entradas	Salidas Esperadas
"19 de Abril, 2345"	
50	"Dirección inválida"
50.2	"Dirección inválida"
43.23	"Dirección inválida"
	"Dirección inválida"

Tabla 2.16: Clases de "Dirección"@

Prueba de fecha y hora de envio

Datos válidos	Datos no válidos
Fechas de días posteriores al de realizada la compra	Día de la compra o anteriores
	Fechas que superen las tres semanas de realizada la compra
Horas entre las 8:00 am y las 17:00 pm	Horas entre las 17:00 pm y las 8:00 am.

Tabla 2.17: Datos de "Fecha y hora de entrega"@

Entradas	Salidas Esperadas
24 de noviembre	
23 noviembre	
6 de noviembre	
22 de noviembre	"Fecha inválida"
7 de noviembre	"Fecha inválida"
7:59 am	"Hora inválida"
8:00 am	
17:00 mp	
17:01 mp	"Hora inválida"
00:00 am	"Hora inválida"

Tabla 2.18: Datos de "Fecha y hora", día actual 22 de noviembre

Evidencia de realizacion de pruebas unitarias

```

@Test
public void testSetSegundoValor() {
    Integer expectedResult = 2;
    instance.setSegundoValor(2);
    assertEquals(expectedResult, instance.getSegundoValor());
}

/**
 * Test of compareTo method, of class Par.
 */
@Test
public void testCompareTo() {
    Par<Alimento, Integer> expectedResult = new Par(new Alimento("A1", "D3", 1, "FS"), 1);
    assertTrue(instance.compareTo(expectedResult) == 0);
}

```

```

/**
 * Test of equals method, of class Alimento.
 */
@Test
public void testEquals() {
    Alimento nuevo = new Alimento("n2", "d2", 1, "Frutos secos");
    assertFalse(nuevo.equals(instance));
}

/**
 * Test of toString method, of class Alimento.
 */
@Test
public void testToString() {
    String expectedResult = "1kg de frutillas";
    instance.setNombre("1kg de frutillas");
    assertEquals(expectedResult, instance.toString());
}

/**
 * Test of addSucursal method, of class Sistema.
 */
@Test
public void testAddSucursal() {
    Sucursal sucursal = new Sucursal("D1", 4245667);
    List<Sucursal> expectedResult = new ArrayList();
    expectedResult.add(sucursal);
    instance.addSucursal(sucursal);
    assertEquals(expectedResult, instance.getSucursales());
}

/**
 * Test of eliminarSucursalPorId method, of class Sistema.
 */
@Test
public void testEliminarSucursalPorId() {
    Sucursal sucursal = new Sucursal("D1", 4245667);
    List<Sucursal> expectedResult = new ArrayList();
    expectedResult.add(sucursal);
    instance.addSucursal(sucursal);
    expectedResult.remove(sucursal);
    instance.eliminarSucursalPorId(1);
    assertEquals(expectedResult, instance.getSucursales());
}

/**

```

2.3. Análisis de código

La mayor parte del esfuerzo de los desarrolladores se ocupa en mantener código ya existente, por lo que mantener un código claro y simple reduce esfuerzos a la hora de mantenerlos (y muchas frustraciones).

El desarrollo de software es un producto de equipo y el código es su herramienta de trabajo, un código claro es una herramienta buena que ayuda a mejorar el trabajo en equipo, cuanto más grande es el equipo y más áreas abarca (ingenieros de software, testers, desarrolladores, diseñadores) más importante se vuelve tener buenas herramientas de trabajo.

Los estándares de codificación son la base y guía de este requisito. En el apartado anterior planteamos un estándar elegido por nosotros para este sistema y ahora evaluaremos el producto final de código de la primer versión.

En la clase interfaz no se pudieron seguir al pie los estándares de codificación, buscaremos arreglar este defecto para hacer más clara su comprensión.

En cuanto a los comentarios, dejamos solo los necesarios y los referentes a los test unitarios, los otros no los vimos necesarios dado que el código está muy claro, ya que seguimos los estándares lo más posible, con nombres claros y métodos simples.

3. Interfaz de usuario y usabilidad

3.1. Criterios de interfaz de usuario

La interfaz de usuario es la forma de comunicación entre los seres humanos y el sistema, con el objetivo de lograr que la interfaz sea una extensión de la persona, es decir que sea fácil de aprender y de usar, evitando así sentimientos de frustración, seguimos un conjunto de criterios a la hora de diseñar la aplicación.

Consistencia

Uniformidad en la apariencia, colocación y comportamiento, reduciendo el esfuerzo de aprendizaje.

- Juntar funcionalidades similares.
- Usar diferentes estilos para elementos seleccionables.
- Centrar el objetivo.

Corrección de errores

El usuario debe tener posibilidad de cancelar una acción haciendo uso de un comando deshacer. Esto ayuda sobre todo a los nuevos usuarios.

Metáforas

Para simplificar descripciones que ejecutan cierta tarea se usa un elemento visual que representa la acción que se espera del usuario, a esto se le conoce como metáforas. Estas deben ser realistas y simples, asemejarse lo más posible a la vida real y que tengan significado para el usuario.

Ergonomía y estética

Los sistemas se vuelven más usables cuando indican su estado claramente, las posibles acciones que se pueden tomar y los posibles resultados. Creando una organización jerárquica, colocando información dentro de categorías

lógicas, logramos mejorar la usabilidad. Es necesario presentar y ocultar información según el contexto del sistema.

La ergonomía busca encontrar una distribución de los elementos más adecuada con el objetivo de que las personas ejecuten sus actividades más eficientemente. La estética busca que la interfaz sea agradable a la vista del usuario. Juntos estos conceptos se fortalecen entre sí.

Mejorar la experiencia

Lograr que los usuarios prefieran tus aplicación por sobre otras se ve potenciado con pequeños detalles de animación en la interfaz que la hagan más vistosa.

3.2. Evaluación de usabilidad

Como dijimos en la sección anterior, la interfaz de usuario es la forma de comunicación entre el usuario y el sistema, esta tiene tres dimensiones principales en las cuales tratamos de basarnos:

- Eficacia.
- Eficiencia.
- Satisfacción.

Una iterfaz de usuario que cumple con las métricas de la usabilidad es imprescindible si se busca lograr cumplir el objetivo de la aplicación.

Nos basaremos en la métrica de usabilidad de Nilsen para evaluar nuestra interfaz.

Métrica de Nilsen:

1. Tiempo de aprendizaje, que tan fácil es para el usuario realizar tareas básicas.
2. Eficacia, cual rápido se pueden realizar las tareas.
3. Tasa de errores, cuantos errores pueden cometer los usuarios y que tan fácilmente pueden recuperarse.
4. Retención, luego de un largo período sin uso de la aplicación; que tan fácil es volver a usarla.
5. Satisfacción, que tan placentero es el diseño del sistema.
6. Reconocimiento y no memorización.
7. Diseño minimalista.

8. Ayuda y documentación.

La idea inicial era una aplicación móvil, inspirada en aplicaciones que usamos todos los días, que se han vuelto muy populares y queridas entre los usuarios como Pedidos Ya, Pinterest, Instagram, Rappi, Mercado Libre y Tienda Mia. Tratamos de no perder el enfoque a móvil implementando una columna similar a los Menus de las aplicaciones móviles, pero siendo todo accedido directamente desde el inicio.

Para los puntos 1, 2 y 4, buscamos que todos lo que se puede hacer con la aplicación estuviera presente durante los flujos básicos de uso y que se puede llegar a todos los puntos de la aplicación, desde el inicio, en menos de 4 pasos.

Para el punto 3, nos aseguramos de tener un botón de volver al inicio en las pantallas que se van de él y al hacer la compra pueden tanto eliminar como agregar productos, o directamente cancelar la compra. En caso que se deba mostrar un mensaje de error tratamos que fuera lo más comunicativo posible, que no avasallara al usuario con un error sino que fuera más constructivo y amigable para la persona cuidando el tono.

Para saber que tan placentero es el diseño, el punto 5, le dimos a probar el sistema a 2 usuarios distintos.

La aplicación no esta pensada para hacerse sin atención 100 % dedicada, pero busca que el usuario se concentre más en comprar que en usar la aplicación, que para él usarla sea tan natural que lo haga sin tener que pararse a pensar cómo seguir.

Nos enfatizamos en una entrada de datos simple y organizada, donde los datos más importantes tengan una jerarquía mayor expresada en su tamaño, ubicación al centro de la pantalla y fácil acceso, con estos datos nos referimos al producto, que será el protagonista de nuestra tienda y lo que debe atrapar al usuario para consumir el servicio que la aplicación ofrece. En segundo lugar, toda la información que el sistema tiene disponible: ubicación de los locales, productos más comprados, estadísticas. No lo estamos quitando importancia pero entendemos que todos estos elementos son los que complementan y resaltan a los productos y a la tienda en sí.

Manteniendo una interfaz organizada, tratamos de tener como máximo cuatro tamaños de fuente y tres estilos de letra, como máximo cuatro colores que se complementaran positivamente.

Si bien no pudimos, nos gustaría para la siguiente versión mejorar la usabilidad de la interfaz para hacerla más imperceptible.^a su vez que atractiva para el usuario. También mejorar las pruebas de usabilidad sometiendola a más usuarios y en distintos contextos de uso.

En un principio la idea era tener un botón de ayuda, ya que lo vemos importante para el usuario, brindandole seguridad, por cuestiones de cálculos de tiempo aca-

bamos por quitarlo para la primer versión pero asegurándonos que la próxima este disponible.

Nos gustaría, a su vez que mejoramos los aspectos de usabilidad más apuntados a reducir los esfuerzos del usuario, mejorar el diseño de interfaz de usuario por uno más estético, moderno y minimalista, sin dejar de transmitir la idea de un tienda de Alimentos que promueve el cuidado ambiental.

Por último estamos buscando nuevas ideas que inviten a los consumidores a comprometerse con la propuesta de Reducir, Reciclar y Reutilizar, de una manera sana y satisfactoria.

4. Pruebas funcionales

4.1. Técnicas de prueba aplicadas

Pruebas del sistema

Validaremos el funcionamiento de todo el sistema y si este cumple con los requerimientos funcionales y no funcionales.

Aquí realizaremos las pruebas alfa y simulaciones hechas por los testers de lo que serían las pruebas beta para evaluar, previamente a la realización de las pruebas beta oficiales, la usabilidad. Las pruebas alfa se realizan en un entorno de desarrollo controlado y las beta en un ambiente operativo, desde otra computadora simulando el uso que le darán los usuarios.

Prueba de aceptación

Verificaremos que cumple con el objetivo de parte del usuario, un contraste entre las funcionalidades del sistema y los requerimientos del usuario.

Aquí realizaremos las pruebas beta, hechas por el propio usuario en su contexto de uso normal.

4.2. Casos de prueba

Prueba: Crear Producto

Datos válidos	Datos no válidos
Con todos los datos correctos	Sin el nombre
	Sin el dato de precio
	Sin la descripción
	Sin el nombre y la descripción
	Sin el nombre y el precio
	Sin el precio y la descripción
	Producto con mismo nombre y descripción que un producto ya existente

Tabla 4.1: 1.0 Crear productos

Entradas	Salidas Esperadas
"Sopa", Calabaza, apio, ajo, sal, condimento verde", 60	"Se agregó correctamente"
Calabaza, apio, ajo, sal, condimento verde", 60	"Falta el nombre del alimento"
"Sopa", Calabaza, apio, ajo, sal, condimento verde"	"Falta el precio del alimento"
"Sopa", 60	"Falta la descripción del Alimento"
60	"Faltan datos del alimento"
Calabaza, apio, ajo, sal, condimento verde"	"Faltan datos del alimento"
"Sopa"	"Faltan datos del alimento"
"Sopa", Calabaza, apio, ajo, sal, condimento verde", 55	"Ese alimento ya existe"

Tabla 4.2: 1.0 Crear productos

Prueba: Eliminar productos

- Elimino 1 producto.
- Elimino todos los productos.

Entradas	Salidas Esperadas
Preciona eliminar una vez	Desaparece el producto del carrito.
Elimina todos los productos.	"No tenemos alimentos disponibles, lo sentimos"

Tabla 4.3: 2.0 Eliminar Productos

Prueba: Seleccionar un envase

Entradas	Salidas Esperadas
Selecciona envase compostable	
Selecciona envase reutilizable	

Tabla 4.4: 3.0 Seleccionar envase

Prueba: Agrego punto de venta

Datos válidos	Datos no válidos
Con todos los datos correctos	Sin la dirección
	Sin el teléfono

Tabla 4.5: 4.0 Agregar punto de venta

Entradas	Salidas Esperadas
"18 de julio, 123", 23904940	"Se agregó correctamente"
386474848	"Falta el teléfono sucursal"
"18 de julio, 123"	"Falta el precio del alimento"

Tabla 4.6: 4.0 Agregar punto de venta

Prueba: Eliminar puntos de venta

- Elimino 1 punto de venta.
- Elimino todos los puntos de venta.

Entradas	Salidas Esperadas
Preciona eliminar una vez	Desaparece la sucursal de la lista.
Elimina todos los puntos de venta.	Quita todos los puntos de venta de la lista

Tabla 4.7: 5.0 Eliminar puntos de venta

Prueba: Crear venta

Para crear la venta el usuario no debe ingresar ningún dato, por defecto le quedan asignados datos, y dado que el único dato que debe ingresar serían la dirección de envío (al registrarse el usuario debe ingresar una dirección y al eliminar siempre le deja al menos una dirección), la fecha y la hora de entrega.

- Intenta comprar con el carrito vacío.

- Realiza una compra válida.
- Intenta comprar sin poner fecha y hora de envío.
- Intenta comprar sin haber seleccionado la dirección.

Entradas	Salidas Esperadas
Preciona comprar y cuando el carrito esta vacío.	."El carrito esta vacío"
Preciona comprar con al menos un elemento en el carrito.	"Su compra fue realizada con éxito"
Presiona comprar sin haber seleccionado la fecha y hora de entrega.	"Faltan datos para realizar la compra"
Presiona comprar sin haber seleccionado la dirección de entrega.	"Faltan datos para realizar la compra"

Tabla 4.8: 6.0 Crear Venta

4.3. Sesiones de ejecución de pruebas

Pruebas exploratorias: Flujo de Carrito

Tiempo: 15 minutos

Objetivos: Encontrar defectos en el flujo de carrito yendo hasta el y desde él a cualquier otra sección

Tester: Sofia Tejerina

Entorno: Computadora Medion, 8 de la mañana, con netBeans cerrado.

Version probada: 0.9, última versión antes de la definitiva de esta primera parte

Entradas	Salidas Obtenidas
ir de carrito a mis logros con elementos en el carrito	ok
ir de carrito a carrito con elementos	ok
ir de carrito a la tienda con elementos	ok
ir de carrito a mis direcciones con elementos	ok
ir de carrito a mis facturas con elementos	ok
mismas pruebas pero con carrito vacio	ok

Tabla 4.9: flujo de carrito

Pruebas exploratorias: Flujo de Mis logros

Tiempo: 10 minutos

Objetivos: Encontrar defectos en el flujo de Mis logros yendo hasta este y desde el a cualquier otra sección

Tester: Sofia Tejerina

Entorno: Computadora Medion, 8:30 de la mañana, con netBeans cerrado.

Version probada: 0.9, última versión antes de la definitiva de esta primera parte

Entradas	Salidas Obtenidas
ir de mis logros a el carrito	ok
ir de mis logros a mis logros	ok
ir de mis logros a la tienda	ok
ir de mis logros a mis direcciones	ok
ir de mis logrod a mis facturas	ok

Tabla 4.10: Flujo de logros

Pruebas exploratorias: Flujo de Mis direcciones

Tiempo: 15 minutos

Objetivos: Encontrar defectos en el flujo de Mis direcciones yendo hasta esta y desde ella a cualquier otra sección

Tester: Sofia Tejerina

Entorno: Computadora Medion, 8:50 de la mañana, con netBeans cerrado.

Version probada: 0.9, última versión antes de la definitiva de esta primera parte

Entradas	Salidas Obtenidas
ir de mis direcciones a el carrito	ok
ir de mis direcciones a mis logros	ok
ir de mis direcciones a la tienda	ok
ir de mis direcciones a mis direcciones	ok
ir de mis direcciones a mis facturas	ok

Tabla 4.11: flujo de Direcciones

Pruebas exploratorias: Flujo de Mis logros

Tiempo: 13 minutos

Objetivos: Encontrar defectos en el flujo de Mis logros yendo hasta este y desde él a cualquier otra sección

Tester: Sofia Tejerina

Entorno: Computadora Medion, 9:20 de la mañana, con netBeans cerrado.

Version probada: 0.9, última versión antes de la definitiva de esta primera parte

Entradas	Salidas Obtenidas
ir de mis recibos a mis logros	ok
ir de mis recibos a la tienda	ok
ir de mis recibos a mis direcciones	ok
ir de mis recibos a mis facturas	ok
ir de mis recibos a el carrito	no funciona

Tabla 4.12: flujo de recibos

Pruebas exploratorias: Flujo de alimentos

Tiempo: 16 minutos

Objetivos: Encontrar defectos en el flujo de Alimentos yendo hasta este y desde él a cualquier otra sección

Tester: Santiago Toscanini

Entorno: Computadora Medion, 9:40 de la mañana, con netBeans cerrado.

Version probada: 0.9, última versión antes de la definitiva de esta primera parte

Entradas	Salidas Obtenidas
ir de alimentos a el carrito	ok
ir de alimentos a mis logros	ok
ir de alimentos a la tienda	ok
ir de alimentos a mis direcciones	ok
ir de alimentos a mis facturas	ok
ir de alimentos a Mas vendidos	ok
ir de alimentos a Puntos de venta	ok
ir de alimentos a Estadisticas	ok
ir de alimentos a alimentos	ok

Tabla 4.13: flujo de Alimentos

Pruebas exploratorias: Flujo de Puntos de venta

Tiempo: 16 minutos

Objetivos: Encontrar defectos en el flujo de Puntos de venta yendo hasta este y desde él a cualquier otra sección

Tester: Santiago Toscanini

Entorno: Computadora Medion, 10:00 de la mañana, con netBeans cerrado.

Version probada: 0.9, última versión antes de la definitiva de esta primera parte

Entradas	Salidas Obtenidas
ir de Puntos de venta a el carrito	ok
ir de Puntos de venta a mis logros	ok
ir de Puntos de venta a la tienda	ok
ir de Puntos de venta a mis direcciones	ok
ir de Puntos de venta a mis facturas	ok
ir de Puntos de venta a mis alimentos	ok
ir de Puntos de venta a mas vendidos	ok
ir de Puntos de venta a puntos de venta	ok
ir de Puntos de venta a estadísticas	ok

Tabla 4.14: flujo de Puntos de venta

Pruebas exploratorias: Flujo de Estadísticas

Tiempo: 14 minutos

Objetivos: Encontrar defectos en el flujo de Estadísticas yendo hasta estas y desde ellas a cualquier otra sección

Tester: Santiago Toscanini

Entorno: Computadora Medion, 10:20 de la mañana, con netBeans cerrado.

Version probada: 0.9, última versión antes de la definitiva de esta primera parte

Entradas	Salidas Obtenidas
ir de Estadísticas a el carrito	ok
ir de Estadísticas a mis logros	ok
ir de Estadísticas a la tienda	ok
ir de Estadísticas a mis direcciones	ok
ir de Estadísticas a mis facturas	ok
ir de Estadísticas a alimentos	ok
ir de Estadísticas a mas vendidos	ok
ir de Estadísticas a puntos de venta	ok
ir de Estadísticas a estadísticas	ok

Tabla 4.15: flujo de Estadísticas

Pruebas exploratorias: Flujo de Frutos secos

Tiempo: 20 minutos

Objetivos: Encontrar defectos en el flujo de Frutos secos yendo hasta este y desde él a cualquier otra sección

Tester: Sofia Tejerina

Entorno: Computadora Medion, 10:40 de la mañana, con netBeans cerrado.

Version probada: 0.9, última versión antes de la definitiva de esta primera parte

Entradas	Salidas Obtenidas
ir de frutos secos a el carrito	ok
ir de frutos secos a mis logros	ok
ir de frutos secos a la tienda	ok
ir de frutos secos a mis direcciones	ok
ir de frutos secos a mis facturas	ok
ir de frutos secos a todo lo anterior con todos los datos de un nuevo alimento pero sin crearlo(se elimina la informacion)	ok
if de frutos secos a todo lo anterior pero habiendo creado un elemnto	ok

Tabla 4.16: flujo de Frutos secos

Pruebas exploratorias: Flujo de Especialidades

Tiempo: 20 minutos

Objetivos: Encontrar defectos en el flujo de Especialidades yendo hasta este y desde él a cualquier otra sección

Tester: Sofia Tejerina

Entorno: Computadora Medion, 11:05 de la mañana, con netBeans cerrado.

Version probada: 0.9, última versión antes de la definitiva de esta primera parte

Entradas	Salidas Obtenidas
ir de especialidades a el carrito	ok
ir de especialidades a mis logros	ok
ir de especialidades a la tienda	ok
ir de especialidades a mis direcciones	ok
ir de especialidades a mis facturas	ok
ir de especialidades a todo lo anterior con todos los datos de un nuevo alimento pero sin crearlo(se elimina la informacion)	ok
if de especialidades a todo lo anterior pero habiendo creado un elemnto	ok

Tabla 4.17: flujo de especialidades

Pruebas exploratorias: Flujo de Frutas congeladas

Tiempo: 20 minutos

Objetivos: Encontrar defectos en el flujo de Frutas Congeladas yendo hasta este y desde él a cualquier otra sección

Tester: Sofia Tejerina

Entorno: Computadora Medion, 11:30 de la mañana, con netBeans cerrado.

Version probada: 0.9, última versión antes de la definitiva de esta primera parte

Entradas	Salidas Obtenidas
ir de futas congeladas a el carrito	ok
ir de futas congeladas a mis logros	ok
ir de futas congeladas a la tienda	ok
ir de futas congeladas a mis direcciones	ok
ir de futas congeladas a mis facturas	ok
ir de futas congeladas a todo lo anterior con todos los datos de un nuevo alimento pero sin crearlo(se elimina la informacion)	ok
if de futas congeladas a todo lo anterior pero habiendo creado un elemnto	ok

Tabla 4.18: flujo de Frutas congeladas

Pruebas exploratorias: comportamiento Direcciones

Tiempo: 17 minutos

Objetivos: Encontrar defectos en eliminando y agregando direcciones a Mis direcciones.

Tester: Santiago Toscanini

Entorno: Computadora Medion, 11:55 de la mañana, con netBeans cerrado.

Version probada: 0.9, última versión antes de la definitiva de esta primera parte

Entradas	Salidas Obtenidas
Eliminar mis direcciones y que no te deje eliminar la ultima que quede	ok
Agregar 10 direcciones nuevas, ejecutar las pruebas de flujo de direcciones y volver	ok

Tabla 4.19: Eliminar mis direcciones

Pruebas exploratorias: comportamiento Direcciones

Tiempo: 22 minutos

Objetivos: Encontrar defectos en eliminando y agregando direcciones a Mis direcciones.

Tester: Santiago Toscanini

Entorno: Computadora Medion, 12:30 de la mañana, con netBeans cerrado.

Version probada: 0.9, última versión antes de la definitiva de esta primera parte

Entradas	Salidas Obtenidas
Eliminar todos los alimentos	ok
Agregar 3 elementos de cada tipo	ok
Agregar 3 alimentos de cada tipo, volver a hacer el flujo de alimentos y de cada clasificación	ok
Agregar 2 elementos de cada clasificación al carrito, eliminarlos del carrito y consultar si siguen en la base de datos	ok

Tabla 4.20: flujo de Eliminar alimentos

5. Reporte de defectos

5.1. Definición de categorías de defectos

Graves

Error en el cumplimiento básico de los requerimientos y errores que provoquen la caída del programa.

Medios

Errores que hagan al usuario sentir incómodo o irritado, dificultando su uso de la aplicación, falta de fluidez en el uso de la aplicación, no permitir al usuario retratarse, falta de mensajes de error.

Bajos

Errores que dificulten al usuario el uso de la aplicación sin provocarle ningún sentimiento de estrés, errores como no poder agrandar la pantalla, no tener respuesta en tiempo real de las estadísticas, no dejarle editar sino tener que eliminar y crear de cero, faltas de ortografía y errores de alineación.

5.2. Defectos encontrados por iteración

Cantidad de defectos encontrados:

Clasificación: Grave

Título: Error de eliminación de alimento.

Descripción: Si cliente agrega un producto al carrito y se elimina de la base de datos dicho producto, igualmente lo puede comprar.

Evidencia: Se agregó un elemento de cada tipo, se agregaron todos al carrito, luego se fueron eliminando uno a uno, al volver al carrito seguían estando, da error y provoca caída del programa.

Defectos encontrados por posibles usuarios de la aplicación

Clasificación: Medio

Título: Error de elección de botón.

Descripción: Si el usuario desea volver para atrás debe seleccionar volver a la tienda, pero eso lo confunde, también la ubicación del botón. Se vio como le usuario por inercia intentando ir para atras seleccionaba el botón "mis logros no "volver a la tienda".

Evidencia: Observación y comentarios del usuario.

Clasificación: Bajo

Título: Al cambiar de sistema operativo cambia la interfaz.

Descripción: El usuario probó la aplicación desde una computadora Mac y la interfaz estaba desalineada, las letras eran tan grandes que sobresalía del label, no dejando a la persona entender que decía.

Evidencia: Observación y comentarios del usuario.

Clasificación: Bajo

Título: Botón "Mis direcciones.^a parece siempre en gris.

Descripción: El usuario se vio confundido de que siempre apareciera como seleccionado el botón de "Mis direcciones.^{en} vez del botón que lo llevó al lugar donde se encuentra actualmente.

Evidencia: Observación y comentarios del usuario.

Clasificación: Bajo

Título: Botones desalineados.

Descripción: Se encontraron botones desalineados en los alimentos.

Evidencia: Observación y comentarios del usuario.

Clasificación: Bajo

Título: Texto del botón Carrito.^a veces se mueve para la izquierda del botón.

Descripción: Al seleccionar el botón .

Evidencia: Observación y comentarios del usuario.

5.3. Estado de calidad global

Lo bien logrado

Se logró realizar el cumplimiento de los requisitos funcionales de forma básica, con una fluidez de como máximo 4 clicks para ir de un lugar a otro de la aplicación.

Para Mejorar en la próxima versión

Realizar una depuración de código, en principio sacar todo aquello que sea redundante, lograr mayor claridad del mismo y verificar el correcto cumplimiento de los estándares de codificación.

Mejorar los mensajes de error, para que sean menos genéricos y más amigables para el usuario.

Re-diseñar la interfaz, por una más moderna y atractiva.

Reducir a máximo 3 clicks para ir de un mundo a otro de la aplicación.

Permitir el ingreso de fotos en la creación del alimento, permitir edición de elementos.

Lograr más uniformidad en los tamaños de fuente de la interfaz.

Realizar sesiones de pruebas exploratoria más largas y más a fondo. Pruebas unitarias con casos bordes. Pruebas de caja blanca además de mejores pruebas de caja negra

Para la proxima versión dejaremos que el usuario pueda editar su perfil, y que los alimentos ademas de ser editados tengan cada uno su propia imagen.

6. Reflexión

La implementación nuestra que promuebe el consumo sustentable fue permitirle ver al usuario su impacto ambiental dado sus compras online, esto da pie a múltiples futuras aplicaciones, entre ellas un ranking entre todos los usuarios de la tienda o descuentos para aquellos que superen ciertos valores.

Las pruebas que realizamos fueron en su mayoría pruebas de caja negra, las pruebas de caja blanca no quedaron documentadas porque las usamos de forma informal para encontrar errores que nos daban las pruebas de caja negra y probando todos los caminos de ejecución posibles, todas las alternativas, y finalmente arreglando los errores inmediatos.

Seguir el estandar de codificación fue una tarea sencilla, debido a que elegimos uno que no se alejó mucho de nuestra forma de programar, sin embargo al seguirlo estrictamente vimos que realmente nos ayudaba a ver más claro el código, ya fura para trabajar sobre algo que nuestro compañero habia hecho, como para realizar cambios o arreglos (cosa que nos pasó mucho al comienzo).

La interfaz y la interpretación de la letra fueron las cosas más complicadas y en las que más deberíamos hacer énfasis de aquí en adelante con el proyecto (nunca dejando de lado los otros aspectos como el código o el testing). No logramos quedar conformes con el producto final de la primer versión en estos aspectos. Cambiamos el diseño múltiples veces tratando de abarcar la mayor cantidad de aspectos de la métrica de usabilidad y terminamos perdiendo el enfoque, a la hora de programarlo se nos volvió muy largo y debimos recurrir a hacer recortes para llegar a tiempo con la entrega.

La interpretación de la letra también ayudó a que esta versión no nos halla dejado tan conformes como buscábamos, a medida que avanzábamos surgían dudas y estas dudas producían muchos cambios y que fueron atrasandonos, haciendo que tuvieramos que ordenar por prioridad las tareas y para abarcar lo más que pudieramos.

Vernos en esta situación nos obligó a detener la producción y tener una charla para plantear cuales eran los aspectos más importantes que debíamos tener listos para la entrega, la idea era cumplir de la forma más básica con cada uno y a medida que avanzábamos irlos perfeccionando de a poco.

Lo siguiente que haremos será planear los pasos a seguir y plantear las prio-

ridades desde el comienzo y hacer un Refactoring nosotros mismos, otra vez, es decir volver al código para hacer mejoras que no cambien las funcionalidades, sino que hagan un código más profesional, claro y ameno para continuar con el desarrollo.

Bibliografía