ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE SISTEMAS

DESARROLLO DE UN SISTEMA PARA LA GESTIÓN DE RELACIÓN CON EL CLIENTE APOYADO EN UNA APLICACIÓN MÓVIL: RESTAURANTE TAKOMAMA

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

ANDRÉS SEBASTIÁN PANTOJA PINO

andres.pantoja@epn.edu.ec

Director: Msc. Carlos Eduardo Anchundia Valencia

carlos.anchundia@epn.edu.ec

Quito, diciembre 2022

APROBACIÓN DEL DIRECTOR

Como director del trabajo de titulación **DESARROLLO DE UN SISTEMA PARA LA GESTIÓN DE RELACIÓN CON EL CLIENTE APOYADO EN UNA APLICACIÓN MÓVIL: RESTAURANTE TAKOMAMA** desarrollado por **ANDRÉS SEBASTIÁN PANTOJA PINO**, estudiante de la Carrera de Ingeniería en Sistemas Informáticos y de Computación, habiendo supervisado la realización de este trabajo y realizado las correcciones correspondientes, doy por aprobada la redacción final del documento escrito para que prosiga con los trámites correspondientes a la sustentación de la Defensa oral.

Msc. Carlos Eduardo Anchundia Valencia

DIRECTOR

DECLARACIÓN DE AUTORÍA

Yo, **ANDRÉS SEBASTIÁN PANTOJA PINO**, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Andrés Sebastián Pantoja Pino

DEDICATORIA

El presente trabajo está dedicado a mi padre, William Pantoja, quien desde el cielo mira como su hijo cumple un objetivo más. Nada de esto habría sido posible sin todo tu amor y esfuerzo.

También dedico este trabajo a mi abuelita Fabiola.

AGRADECIMIENTO

A mi madre, Deyci Eugenia, por su amor, apoyo, paciencia y comprensión a lo largo de este proceso.

A mis hermanos, Daniel y Analía, que me motivan cada día.

A mis profesores de la Escuela Politécnica Nacional, especialmente a mi tutor Carlos Anchundia por su amistad, confianza, enseñanzas y motivación a lo largo del desarrollo del proyecto.

A mis hermanos de otra madre: Adrián, Mate, David, Sebas, Santi, Meli y Cami. Gracias por su amistad, sus palabras de apoyo y los momentos de alegría que han traído a mi vida.

A mis compañeros de la facultad: Juan, Juanjo, Patrick, Ali, Andre y Nico por haber hecho que el camino en la carrera sea más agradable.

A mi familia y todos los que fueron parte del desarrollo de este proyecto.

ÍNDICE DE CONTENIDO

1. INTRODUCCIÓN 1
1.1. Antecedentes 1
1.2. Problemática 3
1.3. Objetivos 5
1.3.1. Objetivo General 5
1.3.2. Objetivos Específicos 5
2. MARCO TEÓRICO 6
2.1. Behavior Driven Development 6
2.2. Feature 8
2.3. Escenarios 8
2.3.1. Gherkin 9
2.4. System Usability Scale 9
3. DESARROLLO DE LA APLICACIÓN 11
3.1. Planeamiento Inicial 11
3.2. Iteraciones 13
3.2.1. Feature 1 Digitalización del menú 13
3.2.2. Feature 2 Generación de pedidos 18
3.2.3. Feature 3 Creación de programa de cliente frecuente 23
3.2.4. Feature 4 Registro de consumos 26
3.2.5. Feature 5 Atención de pedidos 29
3.2.6. Feature 6 Recomendación de amigos 34

3.2.7. Feature 7 Publicación de promociones 37
3.2.8. Feature 8 Volver a pedir 40
3.2.9. Feature 9 Retroalimentación del cliente 42
4. RESULTADOS 44
4.1. Pruebas de aceptación 44
4.2. Pruebas de usabilidad 45
4. CONCLUSIONES Y RECOMENDACIONES 47
4.1. Conclusiones 47
4.2. Recomendaciones 48
BIBLIOGRAFÍA 50
ANEXOS 52

LISTA DE TABLAS

Tabla 1 Palabras claves de Gherkin 9

Tabla 2 Digitalización del menú 14

Tabla 3 Generación de pedidos 18

Tabla 4 Creación de programa de cliente frecuente 23

Tabla 5 Registro de consumos 26

Tabla 6 Atención de pedidos 30

Tabla 7 Recomendación de amigos 35

Tabla 8 Publicación de promociones 37

Tabla 9 Volver a pedir 41

Tabla 10 Retroalimentación del cliente 42

Tabla 11 Preguntas de la encuesta SUS 46

LISTA DE FIGURAS

Figura 1 Diagrama del modelo de alineamiento basado en propósito 7

Figura 2 Diagrama con los features a implementar 12

Figura 3 Issues en GitLab 13

Figura 4 Pruebas - Digitalización del menú 15

Figura 5 Archivos creados automáticamente 15

Figura 6 Excepciones en los archivos creados 16

Figura 7 Configuración previa a la ejecución de pruebas 17

Figura 8 Pasos de las pruebas implementados 17

Figura 9 Interfaz gráfica - Digitalización del menú 18

Figura 10 Pruebas - Generación de pedidos 21

Figura 11 Opción agregar/eliminar elementos del carrito de compras 21

Figura 12 Opción para modificación de ingredientes 22

Figura 13 Resumen de elementos del carrito de compras 22

Figura 14 Pruebas - Creación de programa de cliente frecuente 24

Figura 15 Plantilla de registro de clientes 25

Figura 16 Saldo inicial del cliente 25

Figura 17 Pruebas - Registro de consumos 28

Figura 18 Uso de cashback 29

Figura 19 Movimientos 29

Figura 20 Pruebas - Atención de pedidos 31

Figura 21 Pedidos del usuario 32

Figura 22 Gestión de pedidos 32
Figura 23 Visualización de comanda..... 33
Figura 24 Información de pedido..... 34
Figura 25 Pruebas - Recomendación de amigos 35
Figura 26 Interfaz recomendación de amigos 36
Figura 27 Ingreso exitoso de recomendación de amigos 36
Figura 28 Pruebas - Publicación de promociones 37
Figura 29 Promociones del negocio 38
Figura 30 Promoción aplicada en el menú 39
Figura 31 Notificación de promoción 40
Figura 32 Pruebas - Volver a pedir..... 41
Figura 33 Interfaz para volver a pedir..... 42
Figura 34 Pruebas - Retroalimentación del cliente..... 43
Figura 35 Interfaz para retroalimentación del cliente..... 43
Figura 36 Resultados de Flutter test 44
Figura 37 Promedio de respuestas SUS 45

LISTA DE ANEXOS

Anexo I Mapa de impacto 52

Anexo II Archivos feature 53

Anexo III Resultados de las pruebas por feature 54

Anexo IV Resultado del comando flutter test 55

RESUMEN

En la actualidad los establecimientos de comida deben mantener un modelo de negocio que esté acorde a la realidad social y tecnológica, utilizando a su favor las aplicaciones que permitan mejorar sus ventas y afianzar a sus clientes. Takomama es un restaurante de comida mexicana que inició sus actividades en la ciudad de Ambato en el año 2020. Al igual que muchos negocios pequeños de comida, no contaba con una aplicación que permita crear y mantener una interacción continua con sus clientes.

Es por eso que con el presente trabajo de titulación se busca desarrollar una aplicación móvil que permitiera receptar pedidos, informar las promociones y gestionar un programa de cliente frecuente basado en *cashback*.

Se crea una aplicación por la necesidad de los administradores de contar con un medio que evite que sus clientes se acerquen al local para realizar sus compras, como valor agregado del negocio. Para esta aplicación se utilizó el Desarrollo Guiado por Comportamiento o BDD como proceso metodológico de desarrollo del software. Adicionalmente, se utilizó el lenguaje de programación Dart y el framework Flutter para desarrollar la aplicación móvil. Para los datos se utilizó Firebase como base de datos no relacional, logrando crear una aplicación que contó con el 75.46 de puntaje usabilidad en una encuesta (SUS) realizada a los clientes.

Palabras clave: Aplicación móvil, BDD, fidelización del cliente, Flutter.

ABSTRACT

Nowadays, restaurants must maintain a business model adapted to society and technology, using mobile apps that allow business to improve sales and customer loyalty. Takomama is a Mexican food restaurant that started in Ambato in 2020. Like many small businesses, it did not have a mobile app that allows it to create and maintain a continuous interaction with its customers.
That is why the following project aims to develop a mobile application that allows receiving orders, show promotions, and manage a frequent customer program.

An application was developed due to the need of the administrators to have a means that prevents their clients from coming to the premises to make their purchases, as an added value of the business. For this application, Behavior Driven Development was used as the software development methodology. Plus, the Dart programming language and the Flutter framework were used to develop the mobile application. Also, Firebase was used as a non-relational database, managing to create an application that had a usability score of 75.46 in a survey (SUS) conducted with clients.

Keywords: BDD, customer loyalty, Flutter, mobile app.

1. INTRODUCCIÓN

1.1. Antecedentes

En la actualidad la tecnología ha avanzado en gran escala, llegando a automatizar varios procesos, como es el caso de las aplicaciones móviles, que permiten realizar la compra de comida mediante el uso de aplicaciones, logrando así nuevas estrategias de contacto entre negocios y clientes. Por otra parte, la emergencia sanitaria por el COVID-19 ha permitido el auge de estas aplicaciones, las mismas brindan seguridad al cliente al momento de realizar sus compras. Esto permite establecer lazos mucho más fuertes entre los vendedores y sus clientes.

Así también, estas aplicaciones permiten optimizar el tiempo de los clientes, ya que a través de un celular se puede llegar a obtener información actualizada acerca de los negocios. Dentro de esta información se encuentran: el menú, las promociones, la información de locales, los métodos de pago y los pedidos realizados, entre otros. Esto permite que los clientes obtengan información sin visitar los locales físicos.

Adicionalmente, los negocios también se benefician de la acogida de las aplicaciones. Mediante el uso de datos los negocios pueden obtener más información respecto a sus clientes y con dicha información se puede brindar un trato personalizado y centrado en y para el cliente. Esto permite tener una mejor relación con el cliente, aumentando así la confianza entre el cliente y el negocio.

La empresa Takomama es un negocio dedicado a la venta de comida mexicana. Uno de sus objetivos principales es brindar un servicio de excelencia a todos sus clientes. Esto lo hace cumpliendo con protocolos de calidad y brindando un trato amable en

su servicio, tomando en cuenta las necesidades de los clientes. Es por esto por lo que el negocio pretende establecer y mantener valiosas relaciones con los mismos.

El presente proyecto titulado: “Desarrollo de un Sistema para la gestión de relación con el cliente apoyado en una Aplicación Móvil: Restaurante Takomama”, trata de un tema que actualmente ofrece grandes beneficios en la relación que establece el negocio con sus clientes. Al actualizar los métodos de venta de sus productos, el negocio genera comodidad y bienestar a sus clientes, quienes se volverán sus compradores fieles. Así incentivando a recomendar sus servicios a otras personas y reflejando el desarrollo y crecimiento del negocio.

Este proyecto muestra el desarrollo de una aplicación móvil que permita divulgar las promociones, mostrar la disponibilidad de productos y realizar la compra de estos mediante la utilización de la tecnología a través de un smartphone. Con esto se pretende satisfacer las necesidades de los clientes y resguardar su seguridad física mientras se aprovecha la información obtenida para beneficiar al negocio.

El contenido de la investigación se encuentra detallada en cuatro capítulos, descritos a continuación:

- En el Capítulo I se detalla todo lo referente a la problemática que presenta el negocio y se detallan los objetivos del proyecto.
- En el Capítulo II se profundiza en la metodología a aplicar para el desarrollo del proyecto y se recopila la información teórica necesaria para cada una de las fases de desarrollo, de manera que el avance de este cumpla con las características de la metodología escogida.
- En el Capítulo III se encuentra el proceso evaluativo del proyecto, a través de distintas pruebas a realizarse con la finalidad de comprobar la correcta funcionalidad y usabilidad de la aplicación.
- En el Capítulo IV se detallan las conclusiones y recomendaciones que se logran determinar una vez concluido el presente proyecto.

1.2. Problemática

Alrededor del mundo las grandes empresas han apostado por sistemas informáticos y herramientas tecnológicas para la gestión de relaciones con el cliente y procesos de fidelización, así como aumentar las ventas mediante servicios innovadores, por los beneficios que representan para su negocio y las facilidades para sus clientes. En este aspecto, las aplicaciones móviles han jugado un rol importante permitiendo llegar a los clientes a través de estrategias como sistemas de cupones y beneficios. Por ejemplo, la multinacional MacDonal´s tiene en su página web oficial la opción directa de descarga de la aplicación que resume lo antes mencionado [1]. Se ofrecen también programas de recompensas, como es el caso de la cafetería Sweet & Coffee [2] o el restaurante Vaco y Vaca [3] que realizan la devolución temporal de dinero como *cashback*. Esto ha logrado que las aplicaciones a nivel nacional se encuentren en el top de descargas [4], demostrando así la funcionalidad de las aplicaciones en las estrategias de negocio.

En Ecuador las empresas han tenido que adaptarse a la transformación tecnológica que se ha convertido en una corriente mundial de la modernización y mercado [5] [6]. Esto se ha evidenciado aún más en el periodo surgido por la crisis sanitaria causada por el COVID-19 ya que se tuvieron que utilizar medios tecnológicos para solventar el aislamiento obligatorio. En este sentido las micro, pequeñas y medianas empresas (MiPymes) representan el 99.51% del total de empresas [7], de las cuales el 78% son restaurantes y fuentes de soda, este segmento se compone de negocios franquiciados (180 internacionales y 40 nacionales) y negocios no franquiciados [8]. Al revisar el top 20 de aplicaciones para las plataformas Android y iOS (en el segmento señalado), se observa que 19 corresponden a empresas franquiciadas (10 en el top para iOS y 9 en top para Android) [4]. Este tipo de empresas generalmente cuentan con departamentos o áreas enfocadas a la tecnología que tienen como

objetivo asesorar en la transformación digital empresarial, aspecto que por los costos de inversión se vuelve poco accesible para empresas pequeñas.

Como resultado de esta desventaja las MiPymes se enfrentan al desconocimiento de las herramientas, lo que ocasiona que exista resistencia al cambio y miedo/inseguridad al crecimiento en lo que a tecnología se refiere [9] [10]. Este miedo al crecimiento como empresa, implementación de nuevas tecnologías y adopción de nuevas responsabilidades se conoce como síndrome de Peter Pan [11]. La pandemia provocada por el COVID-19 ha creado un cambio significativo en este panorama y ha obligado a estas empresas a aceptar la transformación digital.

El mercado local puede aprovechar el uso de los smartphones, cuyas cifras en relación a su uso a nivel nacional alcanzan el 76.8%, lo que representa a la población que cuenta con un celular activado [12], y el uso del “cloud computing” cuyas características permiten la facilidad de su uso, conveniencia y reducción de costos en infraestructura tecnológica [13] lo que permite, al negocio, generar aplicaciones móviles que potencien sus estrategias de negocio y mejoren su competitividad. Además, a esto se suma la necesidad de las empresas para reinventarse debido al COVID-19. Esta necesidad incentiva a explorar estrategias de marketing personalizadas para cada cliente con el apoyo de la tecnología.

En virtud de estos antecedentes, se plantea desarrollar un sistema enfocado en la relación con el cliente, apoyado en una aplicación móvil híbrida. El sistema permite mostrar la información del negocio, implementar programas de fidelización de clientes a través de beneficios y recolectar los datos de las interacciones de los clientes con el negocio para poder descubrir estrategias de marketing personalizadas para cada cliente.

1.3. Objetivos

1.3.1. Objetivo General

Desarrollar un sistema para la gestión de relación con el cliente, apoyado en una aplicación móvil para el restaurante Takomama.

1.3.2. Objetivos Específicos

- Utilizar el modelo de desarrollo iterativo incremental usando BDD para entregar el proyecto dentro del cronograma establecido.
- Diseñar la aplicación con un enfoque en el cliente y las estrategias de fidelización.
- Asegurar la usabilidad como factor de éxito de la aplicación móvil utilizando System Usability Scale.
- Alcanzar una puntuación de usabilidad de rango “bueno” o superior por parte del personal y clientes seleccionados por el administrador del establecimiento de comida Takomama de la ciudad de Ambato.

2. MARCO TEÓRICO

2.1. Behavior Driven Development

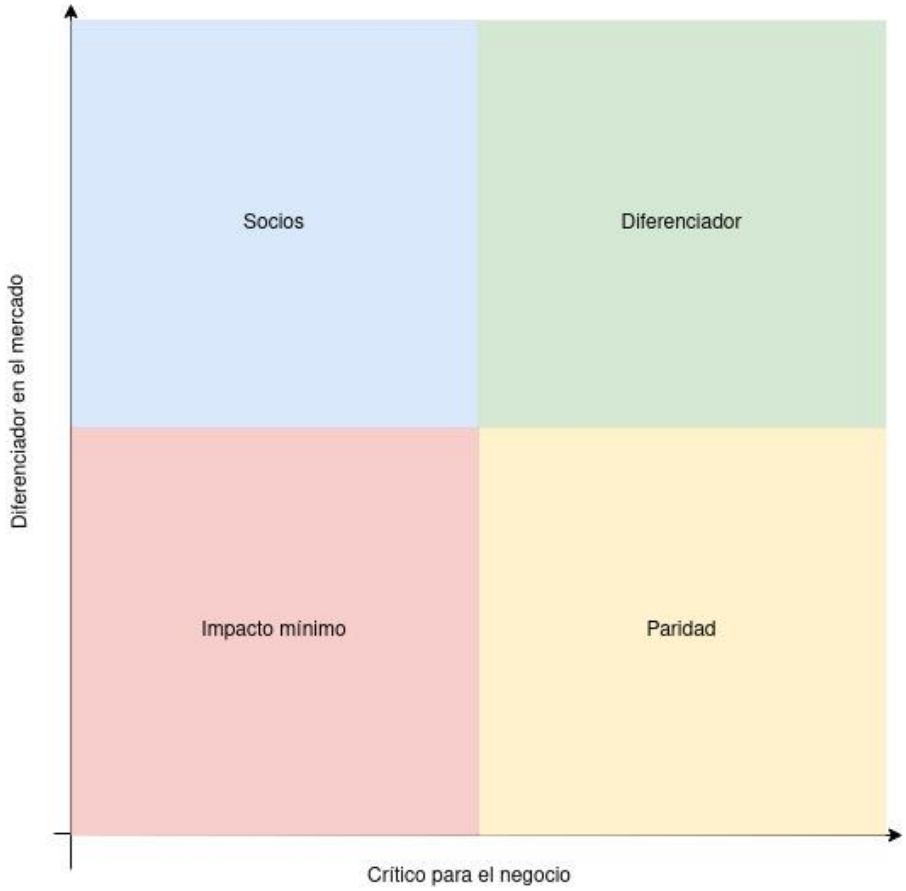
Behavior Driven Development (BDD), conocido en español como Desarrollo Dirigido por el Comportamiento, es un proceso de desarrollo ágil introducido por Dan North. BDD fue utilizado en este proyecto pues permite el uso un lenguaje ubicuo y está relacionado al dominio del problema, permitiendo una mejor comunicación entre los involucrados en el proyecto. Además, se enfoca en crear software que genere valor al negocio a través de la identificación de características o *features*, que indican lo que el sistema debe hacer para cumplir los objetivos del negocio. Por último, BDD se basa en el uso de pruebas de aceptación automatizadas, permitiendo verificar el comportamiento del sistema [14].

Para apoyar al BDD en el objetivo de construir un software de beneficio para el negocio, es óptimo que se busque el apoyo en otras técnicas. La primera es la Inyección de características o *Feature injection.* Esta técnica permite describir la forma en la que el software ayudará al negocio. Mediante conversaciones, se pretende identificar los objetivos del negocio, rescatar las características del software que permiten lograr esos objetivos y obtener ejemplos que describan las características encontradas [15]. La segunda técnica es el Mapeo de impacto o *Impact mapping:* esta técnica ayuda a generar un mapa en donde se identifican las características del software a través de responder las preguntas ¿por qué?, ¿quién?, ¿cómo? y ¿qué? [16].

.

Una vez obtenidas las características del software a implementar, estas deben ser priorizadas. Para esta tarea, BDD se apoya con el modelo de alineamiento basado en propósito o *Purpose based alignment model*. Este modelo es un método usado para evaluar, planear e implementar proyectos. El modelo permite alinear los procesos a implementar con el propósito de una organización. Este propósito está

enfocado a diferenciar a la organización respecto a la competencia en el mercado y a implementar primero los procesos críticos para la organización. Con este modelo, se puede crear un diagrama de cuatro cuadrantes que identifican el propósito de los procesos a implementar. Este diagrama se muestra en la Figura 1.



Fuente: adaptado de Kent J. McDonald, Purpose Based Alignment Model [17]

Figura 1 Diagrama del modelo de alineamiento basado en propósito

La alineación con el propósito ha sido utilizada para definir planes estratégicos, alinear los procesos de TI con las prioridades del negocio y reducir los gastos al enfocar los esfuerzos en las actividades importantes. En resumen, el modelo de alineamiento basado en propósito permite determinar las actividades en las que un equipo de trabajo se debe enfocar y el orden en el que las debe realizar [17].

2.2. Feature

Features o características son las funcionalidades que se implementan para cumplir con los objetivos definidos por el negocio. Estas características indican cómo el software ayudará solventar las necesidades de los usuarios. Dentro de BDD, un *feature* puede estar compuesto de una o más historias de usuario. Así, los *features* del sistema se definen con anterioridad y las historias de usuario pueden ser creadas cuando se acerca el momento de implementar el *feature*.

Las historias de usuario son una herramienta para describir lo que se debe entregar en cada *feature* de manera más detallada. Las historias de usuario responden a las preguntas ¿Qué objetivo del negocio se busca atender?, ¿Quién es el principal beneficiario? y ¿Qué se desea hacer? Adicionalmente, para ilustrar claramente cada *feature* se deben utilizar ejemplos.

Los ejemplos, son una parte esencial de BDD. A través de conversaciones y el uso de ejemplos se mejora el entendimiento de las características que se deben implementar. Además, se debe usar un lenguaje que todos los involucrados en el proyecto entiendan al crear ejemplos. Finalmente, los ejemplos son usados para crear escenarios, los cuales guían el desarrollo y las pruebas a lo largo del proyecto [18].

2.3. Escenarios

Los escenarios son un conjunto de pasos que muestran el comportamiento del sistema al ser ejecutados, para facilitar la escritura de los escenarios se puede utilizar la sintaxis de Gherkin, que define una notación específica para cada paso. Estos pasos indican las acciones que el sistema debe realizar para cumplir con el escenario.

En esta etapa BDD se beneficia de TDD para crear pruebas unitarias que permiten escribir el código de aplicación que verifica los resultados del escenario. Al automatizar la ejecución de los pasos, el escenario se transforma en una prueba de aceptación automatizada. Al automatizar y probar todos los escenarios de un *feature*

este se considera terminado, pues significa que todos los criterios de aceptación han sido cumplidos [19].

2.3.1. Gherkin

Gherkin permite estructurar las especificaciones con el uso de un conjunto de palabras claves, estas palabras han sido traducidas a múltiples idiomas para facilitar su uso, un resumen de las principales palabras clave en inglés y español se encuentra detallado en la Tabla 1, en donde se incluye una breve descripción de cada término.

Tabla 1 Palabras claves de Gherkin

Palabra clave en inglés	Palabra clave en español	Descripción
Feature	Característica	Provee una descripción de la característica a desarrollar y agrupa los escenarios
Scenario	Escenario	Consiste en un conjunto de pasos que conforman una prueba de las reglas del negocio
Given	Dado, Dada, Dados, Dadas	Es el paso que describe el contexto del escenario
When	Cuando	Es el paso que describe el evento del escenario
Then	Entonces	Es el paso que describe el resultado del escenario
And, But	Y, Pero	Son pasos que remplazan el uso sucesivo de los pasos anteriores
Scenario Outline	Esquema del escenario	Permite ejecutar un escenario varias veces con diferentes entradas

Fuente: Cucumber, Gherkin Reference [20]

2.4. System Usability Scale

System Usability Scale (SUS) es una escala que brinda una visión global de la usabilidad de un sistema. La misma está formada por 10 preguntas respecto a la usabilidad. Se decidió usar SUS por la facilidad y simplicidad de su aplicación lo que contribuye a que los clientes contesten con veracidad en base a su experiencia con la aplicación. Además, SUS ha sido utilizada en la evaluación de usabilidad en varias

aplicaciones móviles con resultados óptimos [21]. Para su aplicación se decidió utilizar el modelo de encuesta y cálculos desarrollados por Digital Equipment Corporation [22].

3. DESARROLLO DE LA APLICACIÓN

3.1. Planeamiento Inicial

A través del planeamiento inicial se desea obtener todos los *features* del sistema. Para esto, las técnicas de inyección de características y mapeo de impacto fueron utilizadas. Inicialmente, mediante conversaciones con los administradores del negocio se determinaron los objetivos del negocio. Este acercamiento dio paso a conocer las características del software que pueden ayudar a cumplir los objetivos y brindar ejemplos de cómo se realizaría.

Con esta información, se creó un mapa de impacto que resume la información obtenida con la conversación, sistematizándola de manera técnica. El mapa creado se muestra en el Anexo I. A través del mapa de impacto creado se rescatan las principales características de la aplicación, estas se convierten en las características que serán implementadas.

A continuación, se asignaron prioridades a cada uno de los *features* con la ayuda de los administradores. Esto se logró tras indicar el valor de la alineación del propósito con los administradores del negocio y colocar los *features* dentro de los cuadrantes definidos en el modelo. El diagrama realizado en base al modelo con los *features* del sistema se encuentra descrito en la Figura 2. Además, para obtener el orden de implementación de las características del sistema se tuvo en cuenta los *features* que permitirían tener mayor conocimiento del dominio del negocio.

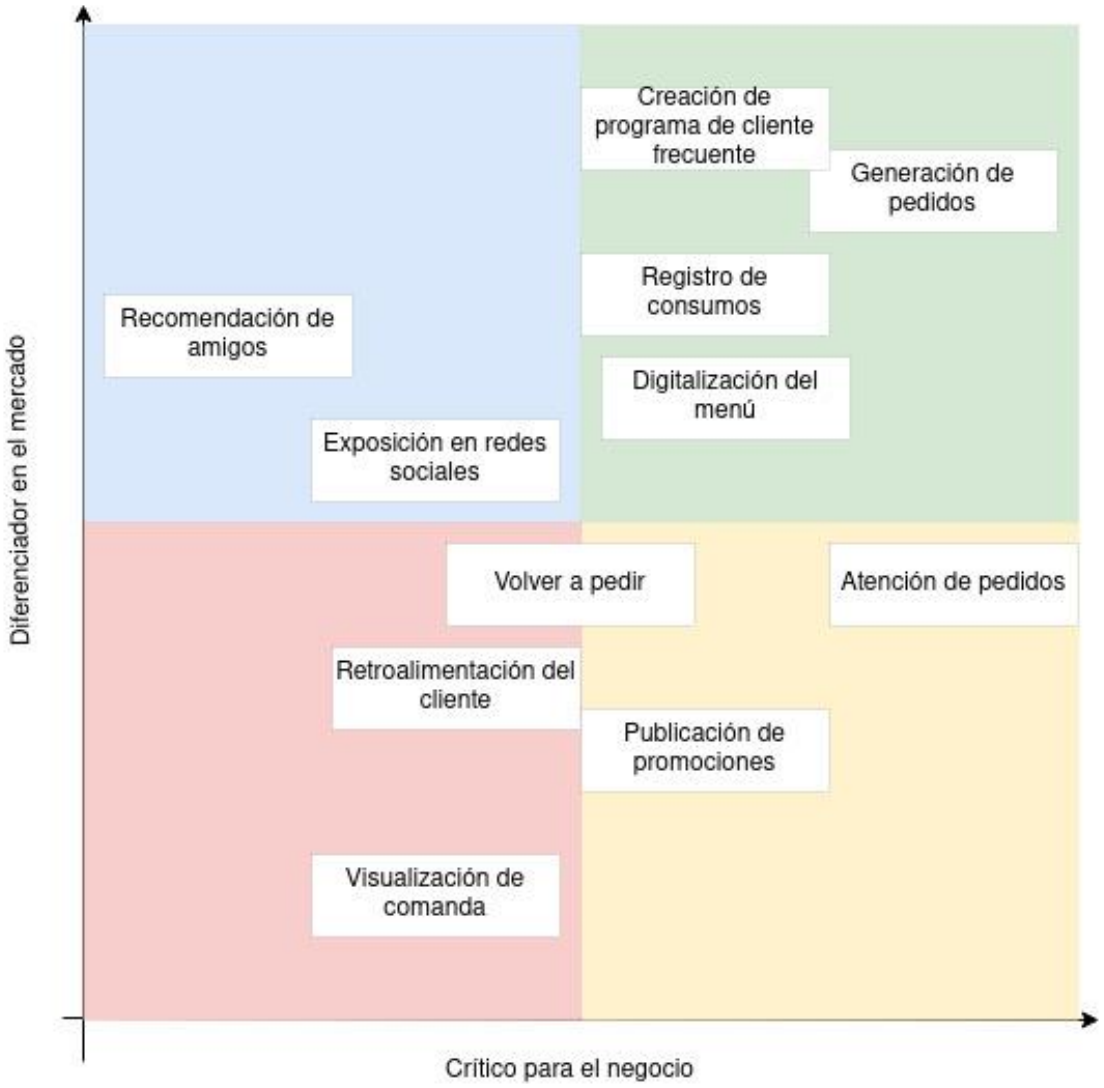


Figura 2 Diagrama con los features a implementar

Para facilitar la gestión de los *features* se usó la herramienta GitLab, esta herramienta ofrece un tablero Kanban para gestionar *issues*. Dentro del proyecto, cada *issue* corresponde a un *feature* por implementar. Las *issues* de GitLab ya priorizadas se muestran en la Figura 3.

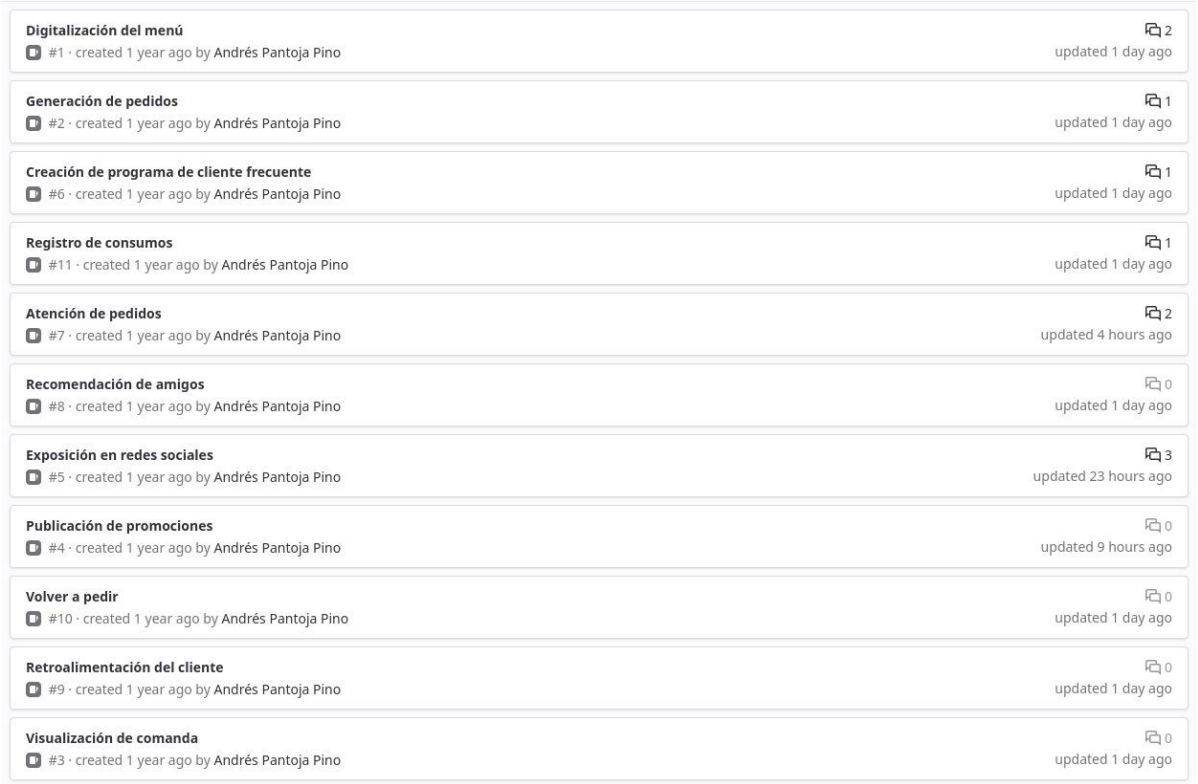


Figura 3 Issues en GitLab

Previo al desarrollo de la aplicación móvil, se definieron las herramientas a utilizar en el proceso de desarrollo. Se optó por el uso del lenguaje de programación Dart y el framework Flutter para desarrollar la aplicación. Además, se utilizó Firebase para la base de datos no relacional, almacenamiento de archivos y autenticación. Para el desarrollo de la aplicación con BDD se utilizó la librería llamada *bdd_widget_test*.

3.2. Iteraciones

3.2.1. Feature 1 Digitalización del menú

El proceso de desarrollo empieza con la definición de la historia de usuario y los escenarios que guían la implementación. Luego de las conversaciones con los administradores del restaurante se definió la historia de usuario y los escenarios descritos en la Tabla 2.

Tabla 2 Digitalización del menú

Feature 1: Digitalización del menú
Historia de usuario: Para pasar menos tiempo interactuando con canales de comunicación, como administrador, quiero poner a disposición del cliente un menú digital.
Escenarios:
El usuario no está registrado o es invitado Dado que el usuario no está registrado en la aplicación Entonces, observa el texto "Ingresa a tu cuenta para ganar beneficios" Y observa 32 productos
El usuario está registrado y logeado Dado que el usuario está logeado en la aplicación Entonces, observa el texto "Vuelve a pedir tus últimos pedidos" Y observa 32 productos

Para empezar con el proceso de desarrollo se utilizó el paquete *bdd_widget_test* que permite usar Gherkin para generar automáticamente los archivos para la realización de las pruebas. Para esto, se definió el archivo .feature relacionado (Anexo II) y con la ayuda del comando *flutter packages pub run build_runner watch --delete-conflicting-outputs* se crearon los archivos .dart que contienen la ejecución de las pruebas y cada uno de los pasos que deben ser implementados. El paquete seleccionado no permite generar los archivos necesarios al usar el idioma español, es por eso que se decidió traducir los escenarios para mantener consistencia en el estilo de escritura. El archivo creado se muestra en la Figura 4 y los pasos generados por comando se detallan en la Figura 5.

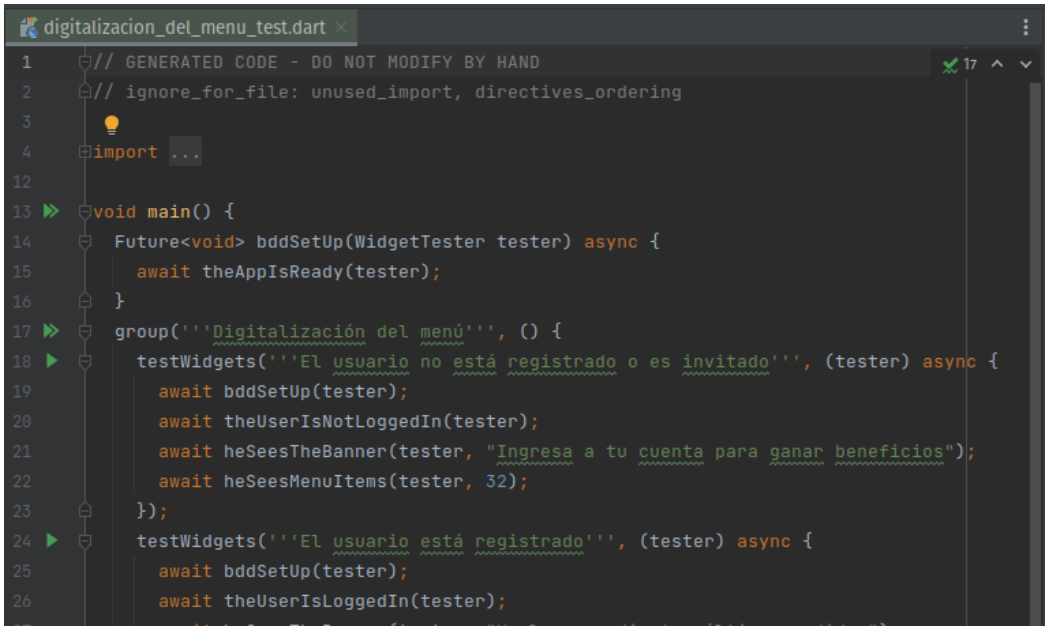


Figura 4 Pruebas - Digitalización del menú

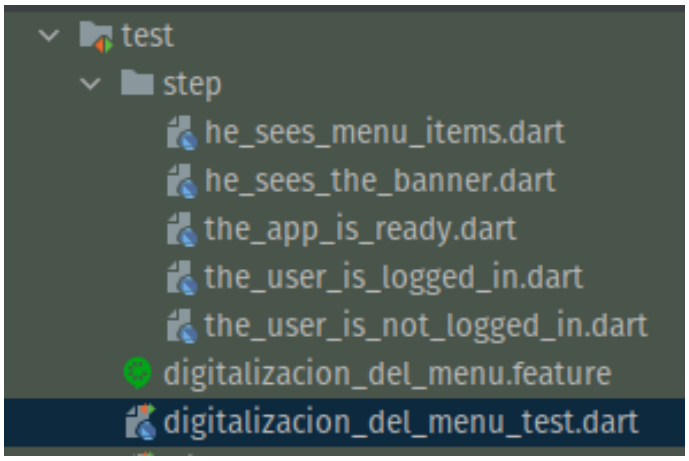


Figura 5 Archivos creados automáticamente

Inicialmente, cada paso contiene una excepción de no implementado. Se debe desarrollar el código necesario para que las pruebas pasen. Es decir, se debe implementar el proceso de TDD. Las excepciones creadas por defecto en los archivos creados automáticamente se muestran en la Figura 6.

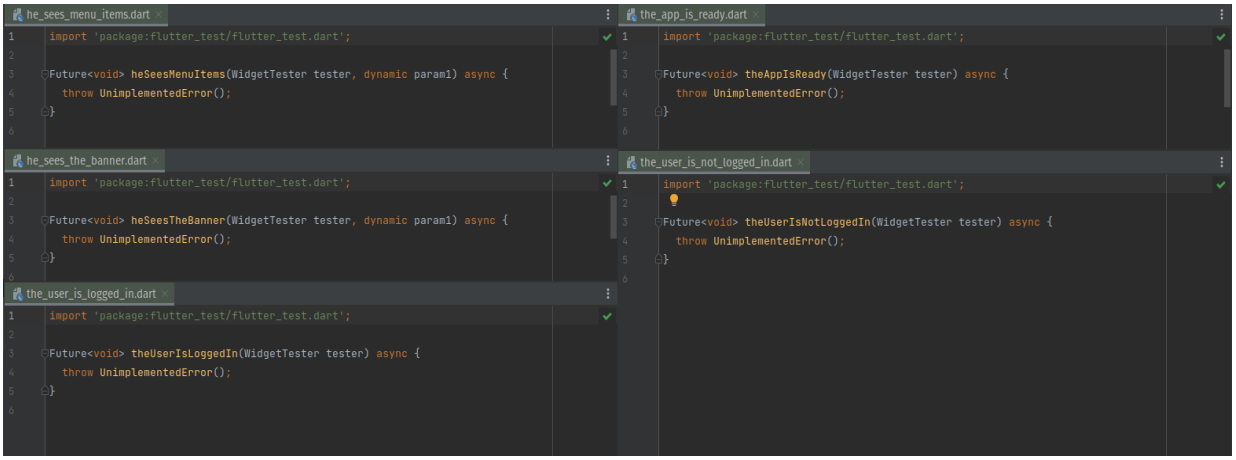


Figura 6 Excepciones en los archivos creados

Para la implementación de la aplicación se utilizó el patrón MVVM con la ayuda de los paquetes *provider,* recomendado por Flutter para la gestión de estado y *get_it* para inyección de dependencias. Con esto se pueden probar las funciones implementadas en cada controlador. Dentro de cada *viewmodel* existen interacciones con diferentes servicios como bases de datos, almacenamiento o notificaciones.

Estas interacciones no se pueden realizar dentro de las pruebas unitarias, para esto se utilizan paquetes que permiten simular la interacción con estos servicios, entre ellos, se utilizaron los paquetes *fake_cloud_firestore, firebase_auth_mocks y firebase_storage_mocks.*

Adicionalmente, previo a la ejecución de los escenarios se inicializan los controladores de la aplicación, para esto, se crea un paso adicional con la palabra clave *Background* de Gherkin. Con esto se pueden probar los controladores con los paquetes que simulan la interacción con los servicios. Esta configuración se muestra en la Figura 7. Finalmente, parar facilitar la lectura del código de las pruebas se utilizó el paquete *shouldly.*

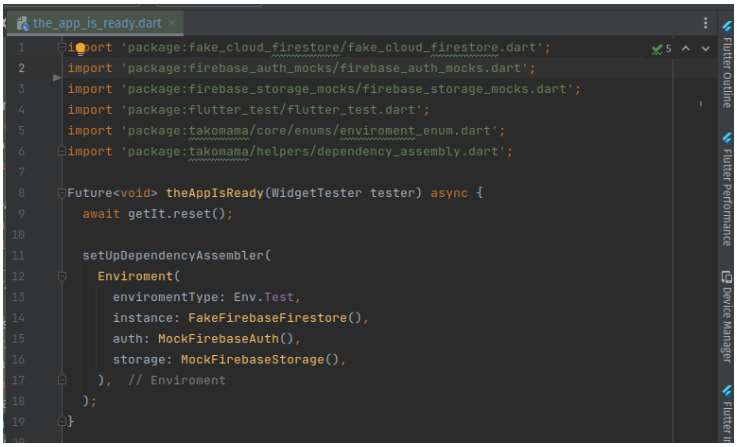


Figura 7 Configuración previa a la ejecución de pruebas

Con esto, se implementan los pasos de cada escenario. Inicialmente, estos escenarios fallan. A continuación, se implementan las funciones que permiten que los escenarios funcionen como se muestra en la Figura 8, y se ejecutan los escenarios. Luego, de ser necesario, se procede a refactorar el código implementado. Finalmente se verifica y se ejecuta todo el *feature* para comprobar su funcionamiento.

Los resultados de la ejecución se encuentran en el Anexo III, el mismo que presenta el resultado de las pruebas corridas en un archivo HTML para que se pueda corroborar desde un navegador de internet. Este archivo fue generado automáticamente por el IDE Android Studio luego de las pruebas respectivas.

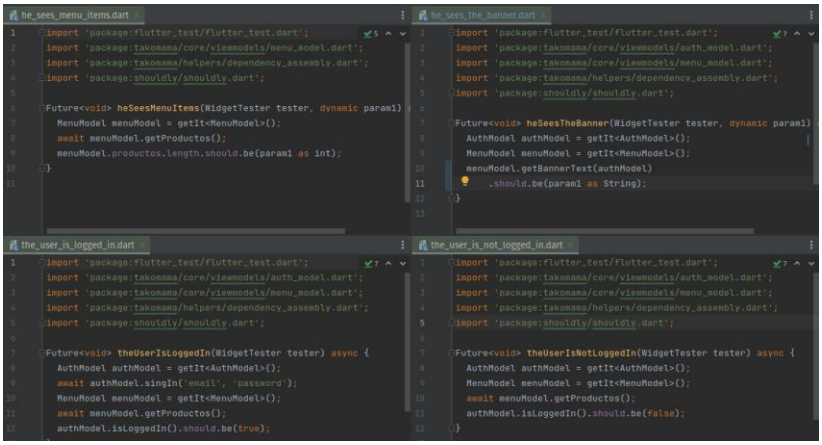


Figura 8 Pasos de las pruebas implementados

Luego, con las funciones de los controladores implementadas, se implementa la interfaz gráfica del *feature*. La interfaz gráfica generada para este *feature* se encuentra detallada en la Figura 9, en la cual se puede verificar el cambio del texto del banner luego de las pruebas.

Figura 9 Interfaz gráfica - Digitalización del menú

3.2.2. Feature 2 Generación de pedidos

El siguiente *feature* que se implementó es la generación de pedidos. Tras las conversaciones con los administradores del negocio se definió la historia de usuario que se detalla en la Tabla 3, de igual manera, se diseñaron los siguientes escenarios descritos.

Tabla 3 Generación de pedidos

Feature 2: Generación de pedidos

Historia de usuario: Para que los clientes sean atendidos más rápidamente, como administrador, deseo generar un pedido automáticamente a partir de los datos ingresados por el cliente en un dispositivo móvil.

Escenarios:

Agregar ítem al carrito

Dado que un taco de pollo cuesta $1.50
Cuando el usuario agrega un taco de pollo al carrito
Entonces el total del carrito es $1.50

Eliminar producto del carrito

Dado que el cliente elige

Cantidad	Nombre	Precio
5	Abc	1.50
2	Def	3.00
3	Ghi	1.00

Pero el cliente elimina def
Entonces el total del carrito es $10.50

Modificar Carrito

Dado que el cliente elige

Cantidad	Nombre	Precio
5	Abc	1.50
2	Def	3.00
3	Ghi	1.00

Pero el cliente elimina

Cantidad	Nombre
2	Abc
1	Def
2	Ghi

Entonces el total del carrito es $8.50

Modificación de Ingredientes

Dado que un taco "Takomama" cuesta $2.00
Y el ingrediente "Queso" cuesta $0.50
Cuando el usuario agrega un taco Takomama al carrito
Y el usuario agrega el ingrediente extra "Queso"
Entonces el total del carrito es $2.50

Creación de Orden

Dado que el usuario está logeado
Y el costo de su carrito es $10.00
Cuando crea su orden
Entonces su pedido es creado

En este *feature* se aprovechó el uso de tablas de datos de Gherkin. Los ejemplos permiten, a través de tablas, indicar las entradas del mismo paso que debe ser ejecutado varias veces. El paso se ejecuta tantas veces como filas en la tabla existan.

Con el uso del paquete *bdd_widget_test* los pasos son repetidos con diferentes argumentos, según la tabla.

El archivo del *feature* creado se encuentra en el Anexo II. La implementación automática de los pasos de este *feature* se muestra en la Figura 10. Adicionalmente, el resultado de ejecutar los escenarios tras implementar los pasos se encuentra en el Anexo III.

```dart
void main() {
  Future<void> bddSetUp(WidgetTester tester) async {
    await theAppIsReady(tester);
  }
  group('''Generación de pedidos''', () {
    testWidgets('''Agregar items al carrito''', (tester) async {
      await bddSetUp(tester);
      await aTacoCosts(tester, 'Takomama', 2.00);
      await theUserAddsATacoToTheCart(tester, 'Takomama');
      await theCartsTotalIs(tester, 2.00);
    });
    testWidgets('''Delete product from cart''', (tester) async {
      await bddSetUp(tester);
      await theClientAddsProductsThatCost(tester, 5, 'abc', 1.50);
      await theClientAddsProductsThatCost(tester, 2, 'def', 3.00);
      await theClientAddsProductsThatCost(tester, 3, 'ghi', 1.00);
      await theClientDeletes(tester, 'def');
      await theCartsTotalIs(tester, 10.50);
    });
    testWidgets('''Modify product from cart''', (tester) async {
      await bddSetUp(tester);
      await theClientAddsProductsThatCost(tester, 5, 'abc', 1.50);
      await theClientAddsProductsThatCost(tester, 2, 'def', 3.00);
      await theClientAddsProductsThatCost(tester, 3, 'ghi', 1.00);
      await theClientDeletesProducts(tester, 2, 'abc');
      await theClientDeletesProducts(tester, 1, 'def');
      await theClientDeletesProducts(tester, 2, 'ghi');
      await theCartsTotalIs(tester, 8.50);
    });
```

```
  testWidgets('''Ingredient Modification''', (tester) async {
    await bddSetUp(tester);
    await aTacoCosts(tester, 'Takomama', 2.00);
    await theIngridientCosts(tester, 'Queso', 0.5);
    await theUserAddsATacoToTheCart(tester, 'Takomama');
    await theUserAddsTheExtraIngredient(tester, 'Queso');
    await theCartsTotalIs(tester, 2.50);
  });
  testWidgets('''Order Creation''', (tester) async {
    await bddSetUp(tester);
    await theUserIsLoggedIn(tester);
    await theCartsCostIs(tester, 10.00);
    await hePlacesHisOrder(tester);
    await theOrderIsCreated(tester);
  });
});
}
```

Figura 10 Pruebas - Generación de pedidos

La implementación de este *feature* produjo varias interfaces de usuario que completan cada uno de los escenarios, es el caso de botones para agregar y reducir elementos del carrito (Figura 11), una pantalla de resumen del carrito (Figura 12) y una pantalla para la modificación de ingredientes (Figura 13). Las interfaces creadas para este *feature* se muestran a continuación.

Figura 11 Opción agregar/eliminar elementos del carrito de compras

Figura 12 Opción para modificación de ingredientes

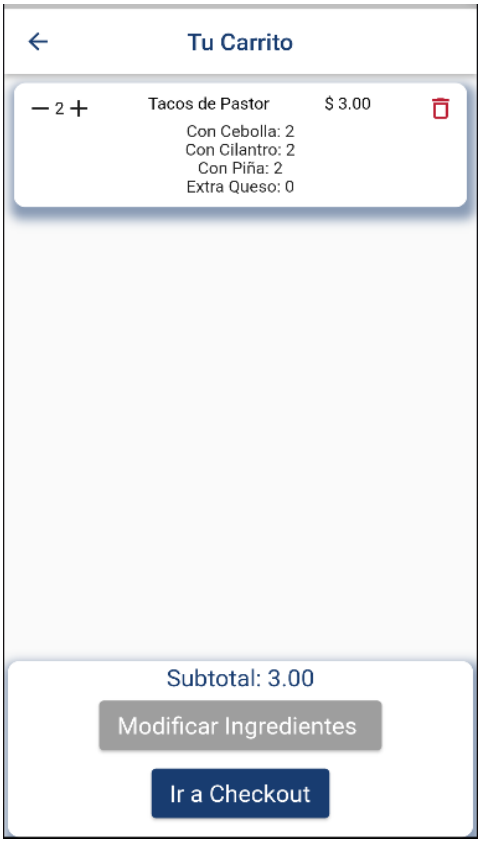


Figura 13 Resumen de elementos del carrito de compras

3.2.3. Feature 3 Creación de programa de cliente frecuente

Los administradores del negocio deseaban crear un programa de cliente frecuente. Luego de que se consideraran varias ideas se consideraron los beneficios de un programa de *cashback*. Tras las conversaciones con los administradores, se decidió crear un programa de *cashback* por cada compra realizada, los administradores decidieron brindar un *cashback* del 2% por cada compra a sus clientes. La historia de usuario y los escenarios de este *feature* se muestran en la Tabla 4.

Tabla 4 Creación de programa de cliente frecuente

<table>
<tr><td colspan="2">Feature 3: Creación de programa de cliente frecuente</td></tr>
<tr><td colspan="2">Historia de usuario: Para gestionar un programa de cliente frecuente, como administrador quiero configurar un programa de cashback del 2% por cada consumo</td></tr>
<tr><td colspan="2">Escenarios:</td></tr>
<tr><td colspan="2">• Calcular cashback

Dado que el costo del carrito es $<precio>

Entonces el cashback calculado es $<cashback>

Ejemplos:
<table>
<tr><th>precio</th><th>cashback</th></tr>
<tr><td>10.00</td><td>0.20</td></tr>
<tr><td>6.25</td><td>0.13</td></tr>
<tr><td>19.88</td><td>0.40</td></tr>
</table></td></tr>
<tr><td colspan="2">• Saldo inicial

Dado que un cliente crea una cuenta

Entonces su nuevo saldo es $0.00</td></tr>
</table>

En este *feature* se utilizó una nueva manera de ejecutar escenarios mediante *Scenario Outline* provisto por Gherkin. Esta característica de Gherkin permitió ejecutar un mismo escenario varias veces según los ejemplos definidos. Esto ayuda a probar el mismo escenario con diferentes datos para tener más casos de prueba.

El archivo *.feature* utilizado se encuentra en el Anexo II, con el uso del paquete de BDD, se generaron las pruebas que se muestran en la Figura 14.

```
import ...

void main() {
  Future<void> bddSetUp(WidgetTester tester) async {
    await theAppIsReady(tester);
  }
  group('''Creación de programa de cliente frecuente''', () {
    testWidgets('''Outline: Calcualte cashback (10.00, 0.20)''', (tester) async {
      await bddSetUp(tester);
      await theCartsCostIs(tester, 10.00);
      await theCashbackCalculatedIs(tester, 0.20);
    });
    testWidgets('''Outline: Calcualte cashback (6.25, 0.13)''', (tester) async {
      await bddSetUp(tester);
      await theCartsCostIs(tester, 6.25);
      await theCashbackCalculatedIs(tester, 0.13);
    });
    testWidgets('''Outline: Calcualte cashback (19.88, 0.40)''', (tester) async {
      await bddSetUp(tester);
      await theCartsCostIs(tester, 19.88);
      await theCashbackCalculatedIs(tester, 0.40);
    });
    testWidgets('''Initial Balance''', (tester) async {
      await bddSetUp(tester);
      await aUserCreatesAnAccount(tester);
      await theNewBalanceIs(tester, 0.00);
    });
  });
}
```

Figura 14 Pruebas - Creación de programa de cliente frecuente

Luego de implementar los pasos de los escenarios de la característica y completar la ejecución de los escenarios, se obtuvo el resultado que se describe en el Anexo III.

La implementación de este *feature* permitió crear varias interfaces de usuario. Entre ellas se encuentra la pantalla de registro, la cual fue tomada del formulario de registro de clientes VIP con el que el negocio cuenta. La pantalla de registro implementada

se muestra en la Figura 15. En la Figura 16 se muestra la interfaz gráfica que se obtiene luego del registro en donde muestra el saldo con el que se inicia la cuenta.

Figura 15 Plantilla de registro de clientes

Figura 16 Saldo inicial del cliente

3.2.4. Feature 4 Registro de consumos

En este *feature* se implementó el comportamiento de la aplicación cuando un cliente registra un consumo. Tras las conversaciones con los administradores del negocio, se obtuvieron la historia de usuario y escenarios de la Tabla 5.

Tabla 5 Registro de consumos

Feature 4: Registro de consumos

Historia de usuario: Para aumentar la cantidad de consumos, como administrador deseo que los clientes ganen puntos en el programa de cliente frecuente al registrar consumos.

Escenarios:

Ganar cashback

Dado que un cliente tiene un saldo de $<inicial>

Y el costo del carrito es $<precio>

Cuando realiza su pedido

Entonces su nuevo saldo es $<final>

Ejemplos:

inicial	precio	final
0.00	15.00	0.30
5.50	3.50	5.57
3.21	12.25	3.46

Utilizar cashback

Dado que un cliente tiene un saldo de $<inicial>

Y el costo del carrito es $<precio>

Cuando usa $<usado> de su saldo para pagar

Y realiza su pedido

Entonces el precio de su pedido es $<pedido>

Y su nuevo saldo es $<final>

Ejemplos:

inicial	precio	usado	pedido	final
1.00	15.00	0.50	14.50	0.80
5.50	3.50	2.50	1.00	3.07
3.21	12.25	3.21	9.04	0.24

Registrar depósito

Dado que el usuario está logeado

Y su carrito cuesta $10.00

Cuando realiza su pedido

Entonces 1 transacción es creada

Y la transacción tiene tipo Depósito y valor de $0.20

Registrar retiro

Dado que el cliente tiene un saldo de $2.00

Y su carrito cuesta $10.00

Cuando usa $0.50 de su saldo

Y realiza su pedido

Entonces 2 transacciones son creadas

Y la transacción tiene tipo Depósito y valor de $0.20

Y la transacción tiene tipo Retiro y valor de $0.50

Y el nuevo saldo es $1.70

Dentro del desarrollo de este *feature*, se aprovechó otra característica de BDD. Los pasos desarrollados en otros *features* se reutilizaron dentro del *feature* actual. El archivo *.feature* utilizado se encuentra en el Anexo II y las pruebas creadas se muestran en la Figura 17.

```dart
void main() {
  Future<void> bddSetUp(WidgetTester tester) async {
    await theAppIsReady(tester);
  }
  group('''Registro de consumos''', () {
    testWidgets('''Outline: Earn cashback (0.00, 15.00, 0.30)''', (tester) async {
      await bddSetUp(tester);
      await aClientHasABalanceOf(tester, 0.00);
      await theCartsCostIs(tester, 15.00);
      await hePlacesHisOrder(tester);
      await theNewBalanceIs(tester, 0.30);
    });
    testWidgets('''Outline: Earn cashback (5.50, 3.50, 5.57)''', (tester) async {
      await bddSetUp(tester);
      await aClientHasABalanceOf(tester, 5.50);
      await theCartsCostIs(tester, 3.50);
      await hePlacesHisOrder(tester);
      await theNewBalanceIs(tester, 5.57);
    });
    testWidgets('''Outline: Earn cashback (3.21, 12.25, 3.45)''', (tester) async {
      await bddSetUp(tester);
      await aClientHasABalanceOf(tester, 3.21);
      await theCartsCostIs(tester, 12.25);
      await hePlacesHisOrder(tester);
      await theNewBalanceIs(tester, 3.45);
    });
    testWidgets('''Outline: Use Cashback (1.00, 15.00, 0.50, 14.50, 0.80)''', (tester) async {
      await bddSetUp(tester);
      await aClientHasABalanceOf(tester, 1.00);
      await theCartsCostIs(tester, 15.00);
      await heUsesOfHisBalance(tester, 0.50);
      await hePlacesHisOrder(tester);
      await thePriceOfHisOrderIs(tester, 14.50);
      await theNewBalanceIs(tester, 0.80);
    });
```

```dart
    testWidgets('''Outline: Use Cashback (5.50, 3.50, 2.50, 1.00, 3.07)''', (tester) async {
      await bddSetUp(tester);
      await aClientHasABalanceOf(tester, 5.50);
      await theCartsCostIs(tester, 3.50);
      await heUsesOfHisBalance(tester, 2.50);
      await hePlacesHisOrder(tester);
      await thePriceOfHisOrderIs(tester, 1.00);
      await theNewBalanceIs(tester, 3.07);
    });
    testWidgets('''Outline: Use Cashback (3.21, 12.25, 3.21, 9.04, 0.24)''', (tester) async {
      await bddSetUp(tester);
      await aClientHasABalanceOf(tester, 3.21);
      await theCartsCostIs(tester, 12.25);
      await heUsesOfHisBalance(tester, 3.21);
      await hePlacesHisOrder(tester);
      await thePriceOfHisOrderIs(tester, 9.04);
      await theNewBalanceIs(tester, 0.24);
    });
    testWidgets('''Register deposit transaction''', (tester) async {
      await bddSetUp(tester);
      await theUserIsLoggedIn(tester);
      await theCartsCostIs(tester, 10.00);
      await hePlacesHisOrder(tester);
      await transactionIsCreated(tester, 1);
      await theTransactionHasTypeAndValue(tester, "Deposito", 0.20);
    });
    testWidgets('''Register withdrawal transaction''', (tester) async {
      await bddSetUp(tester);
      await aClientHasABalanceOf(tester, 2.00);
      await theCartsCostIs(tester, 10.00);
      await heUsesOfHisBalance(tester, 0.50);
      await hePlacesHisOrder(tester);
      await transactionIsCreated(tester, 2);
      await theTransactionHasTypeAndValue(tester, "Deposito", 0.20);
      await theTransactionHasTypeAndValue(tester, "Retiro", 0.50);
      await theNewBalanceIs(tester, 1.70);
    });
  });
}
```

Figura 17 Pruebas - Registro de consumos

Tras completar y correr las pruebas se obtuvo el resultado que se muestra en el Anexo III.

Dentro de este *feature* se crearon nuevas interfaces de usuario. La primera es la encargada de registrar el *cashback* que se va a utilizar en el pedido, esta interfaz se

muestra en la Figura 18. Adicionalmente, en la pantalla del usuario se agregó la visualización de los movimientos creados cuando se realiza un pedido, esto se muestra en la Figura 19.

Figura 18 Uso de cashback

Figura 19 Movimientos

3.2.5. Feature 5 Atención de pedidos

En este *feature* se identificaron dos historias de usuario, cada una contiene sus propios escenarios. Las historias y escenarios identificados se muestran en la Tabla 6.

Tabla 6 Atención de pedidos

Feature 5: Atención de pedidos

Historia de usuario: Para mejorar la atención de pedidos, como administrador deseo proveer varias formas de entrega de pedido.

Escenarios:

- **Tipo de envío**

Dado el carrito cuesta $10.0

Cuando el tipo de envío es a <delivery>

Entonces el costo de envío es $<costo>

Y el precio del pedido es $<total>

Ejemplos:

delivery	costo	total
A domicilio	2.50	12.50
Retirar del Local	0.00	10.00

Historia de usuario: Para mejorar la atención de pedidos, como administrador deseo atender pedidos desde la aplicación móvil.

Escenarios:

- **Estado inicial del pedido**

Dado que el usuario está logeado

Y su carrito cuesta $10.00

Cuando realiza su pedido

Entonces el estado del pedido es "Enviado"

- **Cambio de estado de pedido**

Dado que un cliente ha realizado un pedido

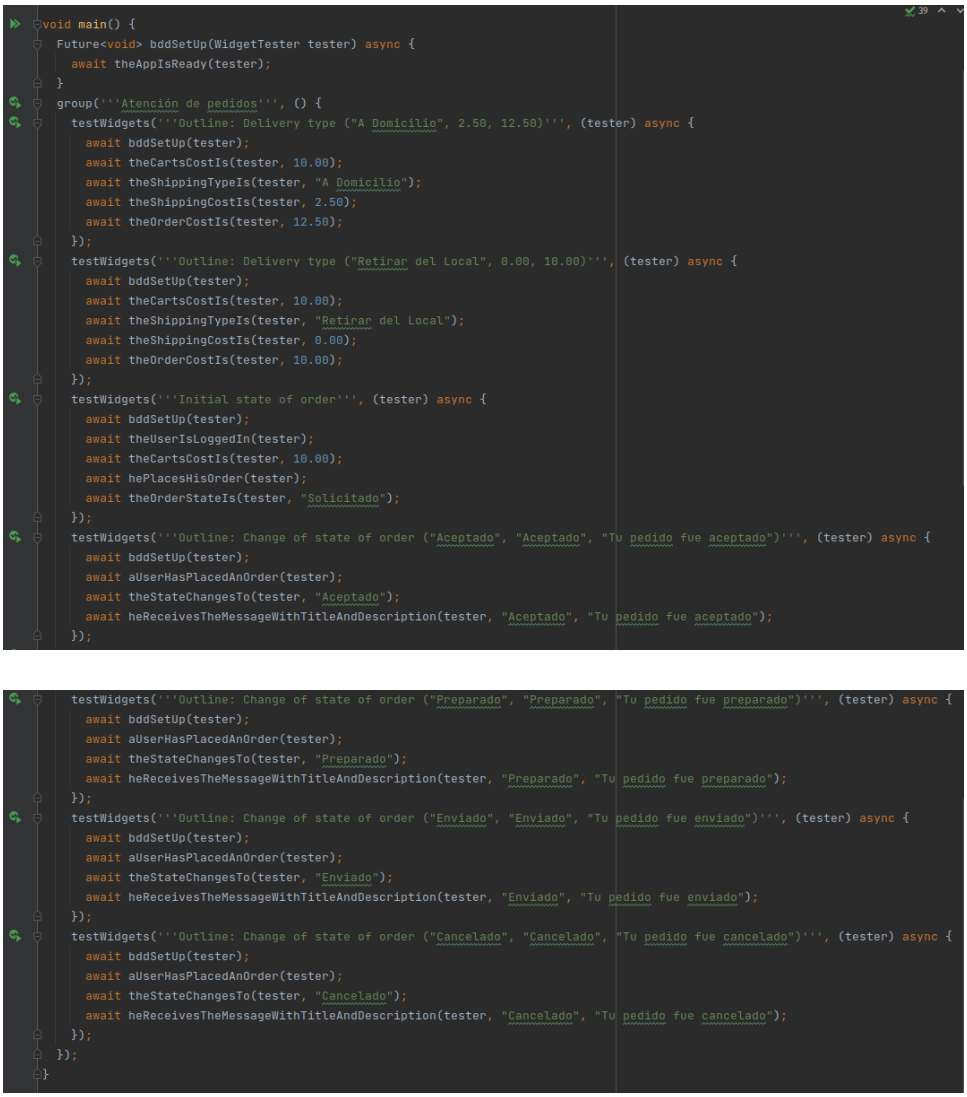
Cuando el estado cambia a <estado>

Entonces recibe un mensaje con el título <titulo> y descripción <descripción>

Ejemplos:

estado	titulo	descripción
Aceptado	Aceptado	Tu pedido fue aceptado
Preparado	Preparado	Tu pedido fue preparado
Enviado	Enviado	Tu pedido fue enviado
Cancelado	Cancelado	Tu pedido fue cancelado

El archivo *.feature* desarrollado para la creación automatizada de las pruebas se encuentra en el Anexo II. Las pruebas creadas se muestran en la Figura 20.



```
void main() {
  Future<void> bddSetUp(WidgetTester tester) async {
    await theAppIsReady(tester);
  }
  group('''Atención de pedidos''', () {
    testWidgets('''Outline: Delivery type ("A Domicilio", 2.50, 12.50)''', (tester) async {
      await bddSetUp(tester);
      await theCartsCostIs(tester, 10.00);
      await theShippingTypeIs(tester, "A Domicilio");
      await theShippingCostIs(tester, 2.50);
      await theOrderCostIs(tester, 12.50);
    });
    testWidgets('''Outline: Delivery type ("Retirar del Local", 0.00, 10.00)''', (tester) async {
      await bddSetUp(tester);
      await theCartsCostIs(tester, 10.00);
      await theShippingTypeIs(tester, "Retirar del Local");
      await theShippingCostIs(tester, 0.00);
      await theOrderCostIs(tester, 10.00);
    });
    testWidgets('''Initial state of order''', (tester) async {
      await bddSetUp(tester);
      await theUserIsLoggedIn(tester);
      await theCartsCostIs(tester, 10.00);
      await hePlacesHisOrder(tester);
      await theOrderStateIs(tester, "Solicitado");
    });
    testWidgets('''Outline: Change of state of order ("Aceptado", "Aceptado", "Tu pedido fue aceptado")''', (tester) async {
      await bddSetUp(tester);
      await aUserHasPlacedAnOrder(tester);
      await theStateChangesTo(tester, "Aceptado");
      await heReceivesTheMessageWithTitleAndDescription(tester, "Aceptado", "Tu pedido fue aceptado");
    });
    testWidgets('''Outline: Change of state of order ("Preparado", "Preparado", "Tu pedido fue preparado")''', (tester) async {
      await bddSetUp(tester);
      await aUserHasPlacedAnOrder(tester);
      await theStateChangesTo(tester, "Preparado");
      await heReceivesTheMessageWithTitleAndDescription(tester, "Preparado", "Tu pedido fue preparado");
    });
    testWidgets('''Outline: Change of state of order ("Enviado", "Enviado", "Tu pedido fue enviado")''', (tester) async {
      await bddSetUp(tester);
      await aUserHasPlacedAnOrder(tester);
      await theStateChangesTo(tester, "Enviado");
      await heReceivesTheMessageWithTitleAndDescription(tester, "Enviado", "Tu pedido fue enviado");
    });
    testWidgets('''Outline: Change of state of order ("Cancelado", "Cancelado", "Tu pedido fue cancelado")''', (tester) async {
      await bddSetUp(tester);
      await aUserHasPlacedAnOrder(tester);
      await theStateChangesTo(tester, "Cancelado");
      await heReceivesTheMessageWithTitleAndDescription(tester, "Cancelado", "Tu pedido fue cancelado");
    });
  });
}
```

Figura 20 Pruebas - Atención de pedidos

Tras completar la ejecución de las pruebas, el resultado de la ejecución se encuentra en el Anexo III

Para este *feature*, se implementaron varias interfaces de usuario, la primera interfaz de usuario es la interfaz de pedidos generados por el usuario. Esta interfaz identifica los pedidos creados y el pedido actual. La mencionada interfaz se muestra en la Figura 21.

Figura 21 Pedidos del usuario

La siguiente interfaz permite a los administradores gestionar los pedidos que han llegado. Esta interfaz se muestra a continuación en la Figura 22.

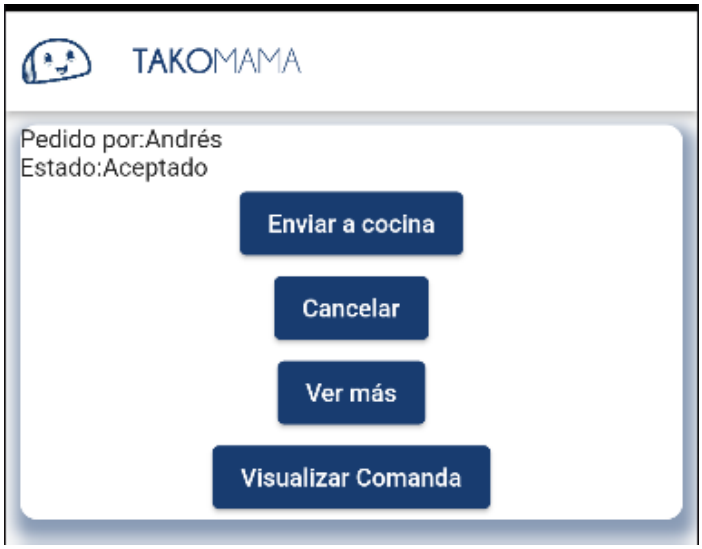


Figura 22 Gestión de pedidos

Al realizar estas interfaces de usuario, también se notó que, al momento de realizar la atención de pedidos, los administradores deben poder visualizar la comanda del

pedido, esta funcionalidad es la misma que el último *feature* definido en el planteamiento inicial, sin embargo, este *feature* no cuenta con ningún comportamiento dentro de la aplicación, sino simplemente puede ser solventada creando una interfaz de usuario con la información necesaria para la comanda. Se decidió solamente crear la interfaz de usuario en este *feature* y eliminar el *feature* de Visualización de comanda de las actividades del proyecto. La interfaz generada para visualizar la comanda se muestra a continuación en la Figura 23.

Figura 23 Visualización de comanda

Del mismo modo, se creó una interfaz con la información del pedido para que tanto usuarios como administradores puedan visualizar los detalles de los pedidos realizados, esto se muestra en la Figura 24.

Figura 24 Información de pedido

3.2.6. Feature 6 Recomendación de amigos

Para este *feature* se obtuvieron la siguiente historia de usuario y los escenarios detallados en la Tabla 7.

Tabla 7 Recomendación de amigos

Feature 6: Recomendación de amigos
Historia de usuario: Para aumentar la cantidad de clientes, como administrador deseo que los clientes ganen puntos en el programa de cliente frecuente por recomendar a nuevos clientes a través de un código de usuario.
Escenarios:
• **Código no encontrado** Dado que un usuario tiene un saldo de $0.00 Cuando usa el código "0000000000" Entonces su saldo es $0.00
• **Código válido** Dado que un usuario tiene un saldo de $0.00 Cuando usa el código "0983495568" Entonces su nuevo saldo es $1.00 Y el usuario ya no puede ingresar otro código

El archivo *.feature* utilizado para crear las pruebas se encuentra en el Anexo II, y las pruebas creadas se muestran en la Figura 25, presentada a continuación.

```dart
void main() {
  Future<void> bddSetUp(WidgetTester tester) async {
    await theAppIsReady(tester);
  }
  group('''Recomendacion de amigos''', () {
    testWidgets('''Code not found''', (tester) async {
      await bddSetUp(tester);
      await aClientHasABalanceOf(tester, 0.00);
      await heUsesTheCode(tester, '0000000000');
      await theNewBalanceIs(tester, 0.00);
    });
    testWidgets('''Valid code''', (tester) async {
      await bddSetUp(tester);
      await aClientHasABalanceOf(tester, 0.00);
      await heUsesTheCode(tester, '0983495568');
      await theNewBalanceIs(tester, 1.00);
      await theUserCantUseACodeToGetABenefit(tester);
    });
  });
}
```

Figura 25 Pruebas - Recomendación de amigos

El resultado de la ejecución de las pruebas se encuentra en el Anexo III.

Para este *feature*, se creó la interfaz de usuario para ingresar un código de la recomendación de amigo. La interfaz diseñada para el ingreso del código se muestra en la Figura 26. Además, cuando el código ya ha sido ingresado exitosamente, no se puede ingresar nuevamente. Este comportamiento se muestra la pantalla de la Figura 27.

Figura 26 Interfaz recomendación de amigos

Figura 27 Ingreso exitoso de recomendación de amigos

3.2.7. Feature 7 Publicación de promociones

La historia de usuario y escenario perteneciente a este *feature* se muestra en la Tabla 8.

Tabla 8 Publicación de promociones

Feature 7: Publicación de promociones
Historia de usuario: Para aumentar la cantidad de pedidos, como cliente, deseo volver a pedir un pedido realizado anteriormente.
Escenarios:
• **Visualización de promociones** Dado que el usuario se encuentra en la aplicación Entonces observa 2 promociones • **Martes de tacos a $1** Dado que el usuario se encuentra en la aplicación Entonces observa 2 promociones

El archivo *.feature* desarrollado para la creación automatizada de las pruebas se encuentra en el Anexo II. Las pruebas creadas se muestran en la Figura 28.

```
void main() {
  Future<void> bddSetUp(WidgetTester tester) async {
    await theAppIsReady(tester);
  }
  group('''Generación de pedidos''', () {
    testWidgets('''Visualización de promociones''', (tester) async {
      await bddSetUp(tester);
      await aUserIsUsingTheApp(tester);
      await heSeesPromotions(tester, 2);
    });
    testWidgets('''Martes de tacos a \$1''', (tester) async {
      await bddSetUp(tester);
      await theCurrentDayIsTuesday(tester);
      await productsOfTypeCost(tester, "Tacos y costras", 1.00);
    });
  });
}
```

Figura 28 Pruebas - Publicación de promociones

Luego de completar la ejecución de las pruebas, el resultado de la ejecución se encuentra en el Anexo III.

Para este *feature* se implementó la interfaz de usuario que muestra las promociones que tiene el negocio. Esta interfaz se muestra en la Figura 29 Promociones del negocio. Adicionalmente, se muestra la interfaz de usuario del menú modificado cuando la promoción de los martes se aplica. Esta funcionalidad se encuentra en la Figura 30.

Figura 29 Promociones del negocio

Figura 30 Promoción aplicada en el menú

Por último, los administradores del negocio solicitaron que se muestre una notificación como recordatorio de las promociones actuales, esto se pudo realizar con la ayuda del paquete *flutter_local_notifications*. El paquete permite crear notificaciones como recordatorios recurrentes, lastimosamente, por ser un paquete que interactúa con medios externos a la aplicación, no pudo ser probado con pruebas unitarias usando BDD. Sin embargo, para constancia de la implementación de la

funcionalidad se muestra la creación de la notificación en el teléfono móvil en la Figura 31.

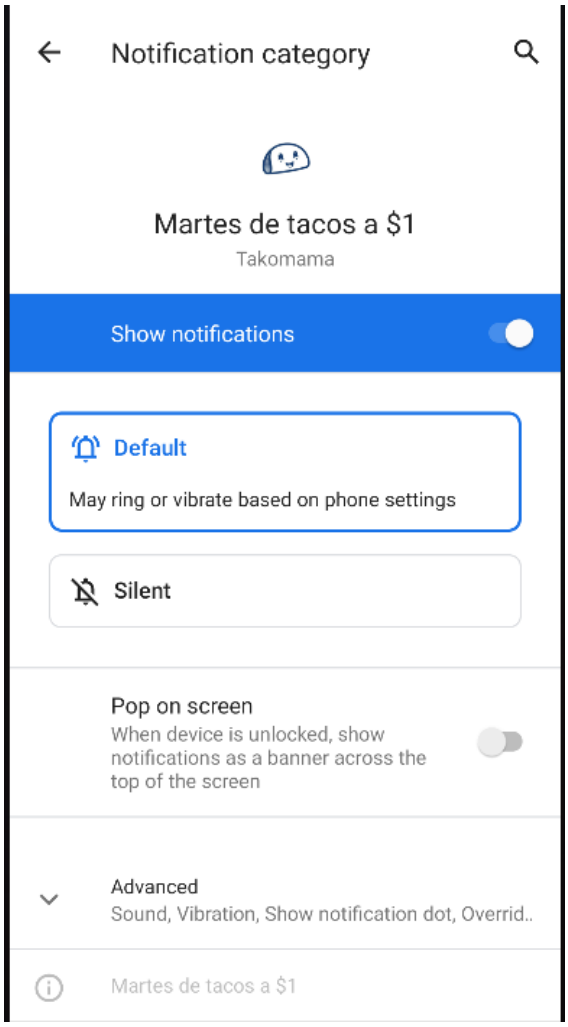


Figura 31 Notificación de promoción

3.2.8. Feature 8 Volver a pedir

La historia de usuario y escenario pertenecientes a este *feature* se muestra en la Tabla 9.

Tabla 9 Volver a pedir

Feature 8: Volver a pedir
Historia de usuario: Para aumentar la cantidad de pedidos, como cliente, deseo volver a pedir un pedido realizado anteriormente.
Escenarios:
• **Volver a pedir** Dado que un usuario tiene un pedido completado Cuando lo vuelve a pedir Entonces los items pedidos se cargan en el carrito

El archivo *.feature* creado para la instauración automatizada de las pruebas y se encuentra en el Anexo II. Las pruebas creadas se muestran en la Figura 32.

```
void main() {
  Future<void> bddSetUp(WidgetTester tester) async {
    await theAppIsReady(tester);
  }
  group('''Volver a pedir''', () {
    testWidgets('''Volver a pedir''', (tester) async {
      await bddSetUp(tester);
      await aUserHasACompletedOrder(tester);
      await heReorders(tester);
      await theOrderedItemsAreLoadedInTheCart(tester);
    });
  });
}
```

Figura 32 Pruebas - Volver a pedir

Tras completar la ejecución de las pruebas, el resultado de la ejecución se encuentra en el Anexo III.

Para este *feature*, se realizó la modificación de la interfaz de usuario de los pedidos realizado por el usuario, agregando la funcionalidad de volver a pedir. La pantalla modificada se muestra en la Figura 33.

Figura 33 Interfaz para volver a pedir

3.2.9. Feature 9 Retroalimentación del cliente

Con este *feature* los administradores del negocio desean conocer la opinión de los clientes respecto a los productos y a el servicio brindado, además, se ofrecería algún tipo de beneficio para el cliente por brindar su retroalimentación. La historia de usuario y los escenarios de este *feature* se obtuvieron tras conversaciones con los administradores del negocio, los resultados se muestran en la Tabla 10.

Tabla 10 Retroalimentación del cliente

Feature 9: Retroalimentación del cliente
Historia de usuario: Para conocer la calidad del servicio brindado, como administrador quiero obtener una calificación del servicio recibido por el cliente.
Escenarios:
Obtener calificación Dado que un pedido ha sido enviado Cuando el cliente califica el pedido Entonces se obtiene una calificación del servicio y de los productos Y el estado de la orden es "Calificado"
Ganar saldo Dado que el cliente tiene un saldo de $1.00 Y que un pedido ha sido enviado Cuando el cliente califica el pedido Entonces el nuevo saldo es $1.10

El archivo *.feature* creado para la creación automatizada de las pruebas se encuentra en el Anexo II y las pruebas que se realizaron se muestran en la Figura 34.

```dart
void main() {
  Future<void> bddSetUp(WidgetTester tester) async {
    await theAppIsReady(tester);
  }
  group('''Retroalimentación del cliente''', () {
    testWidgets('''Obtener calificación''', (tester) async {
      await bddSetUp(tester);
      await anOrderHasBeenCompleted(tester);
      await theClientRatesHisOrder(tester);
      await theStateOfTheOrderIs(tester, "Calificado");
    });
    testWidgets('''Ganar saldo por calificar''', (tester) async {
      await bddSetUp(tester);
      await aClientHasABalanceOf(tester, 1.00);
      await anOrderHasBeenCompleted(tester);
      await theClientRatesHisOrder(tester);
      await theNewBalanceIs(tester, 1.10);
    });
  });
}
```

Figura 34 Pruebas - Retroalimentación del cliente

Tras completar la ejecución de las pruebas, el resultado de la ejecución se encuentra descrita en el Anexo III. La pantalla realizada para proveer retroalimentación por parte del cliente se muestra en la Figura 35.

Figura 35 Interfaz para retroalimentación del cliente

4. RESULTADOS

4.1. Pruebas de aceptación

Según BDD, los escenarios se transforman en criterios de aceptación, para determinar si los criterios de aceptación han sido aprobados, basta con ejecutar todos los escenarios definidos dentro de las características del sistema. En Flutter, para ejecutar todas las pruebas se utiliza el comando *flutter test.* Al correr el comando se obtiene el resultado que se muestra en la Figura 36. Adicionalmente, los resultados del comando con todas las pruebas que han sido ejecutadas se encuentran en el Anexo IV.

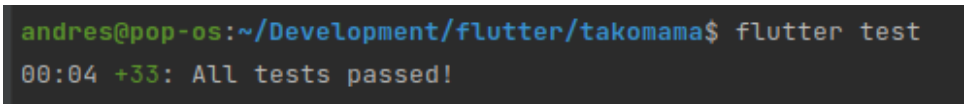


Figura 36 Resultados de Flutter test

Se observa que todas las pruebas han sido superadas con éxito, lo que indica que los criterios de aceptación creados con los escenarios de cada *feature* han sido implementados correctamente.

A lo largo del proceso de desarrollo se implementaron 9 de los 11 *features* definidos en el planteamiento inicial del proyecto. El *feature* Visualización de comanda fue adapatado y desarrollado dentro de la característica Atención de pedidos, mediante la implementación de una pantalla que permite visualizar la comanda de los pedidos. El siguiente *feature* que no se implementó fue el de Exposición en redes sociales. Con esta característica se pretendía brindar un beneficio al usuario por compartir imágenes en redes sociales. Finalmente, se decidió no implementar esta característica ya que los clientes pueden eliminar las publicaciones tan pronto como fueron creadas y sin que el negocio sea notificado de esta eliminación, lo que no permitiría garantizar la entrega del beneficio adecuadamente.

4.2. Pruebas de usabilidad

Para validar la usabilidad de la aplicación se emplea un cuestionario físico, aplicado a los clientes del restaurante luego de haber usado la aplicación en varios dispositivos móviles provistos por los administradores del negocio. Esto se realizó sin entrenamiento previo a los clientes, permitiéndoles explorar y navegar en la aplicación para probar sus funcionalidades. La prueba contiene diez preguntas planteadas en el System Usability Scale adaptadas del trabajo " Spanish Version of the System Usability Scale for the Assessment of Electronic Tools: Development and Validation" [23].

El índice de usabilidad presenta un total de 38 respuestas obtenidas en el periodo de una semana. Se utilizó una escala del 1 al 5 en donde la opción 1 representa "totalmente en desacuerdo" y la opción 5 "totalmente de acuerdo". Las preguntas que se plantearon están detalladas en la Tabla 11 así como el promedio total del cálculo del puntaje de usabilidad. En la Figura 37, en cambio, se presentan los promedios de cada una de las preguntas de la encuesta realizada a los clientes para poder contextualizar el puntaje de usabilidad que se obtuvo.

Figura 37 Promedio de respuestas SUS

Tabla 11 Preguntas de la encuesta SUS

N°	Pregunta/ítem	Promedio total de SUS
1	Creo que me gustaría utilizar este sistema con frecuencia	75,46
2	Encontré el sistema innecesariamente complejo	
3	Pensé que el sistema era fácil de usar	
4	Creo que necesitaría el apoyo de un técnico para poder utilizar este sistema	
5	Encontré que las diversas funciones de este sistema estaban bien integradas	
6	Pensé que había demasiada inconsistencia en este sistema	
7	Me imagino que la mayoría de la gente aprendería a utilizar este sistema muy rápidamente	
8	Encontré el sistema muy complicado de usar	
9	Me sentí muy seguro usando el sistema	
10	Necesitaba aprender muchas cosas antes de empezar con este sistema	

El resultado obtenido del cálculo del System Usability Scale fue 75.46. El valor obtenido se puede calificar como "Bueno" bajo el sistema de adjetivos determinado por los autores Bangor, Kortum y Miller [24].

4. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

- Se desarrolló la aplicación móvil para el restaurante Takomama utilizando BDD como metodología de desarrollo. La aplicación móvil fue desarrollada utilizando el lenguaje de programación Dart y el framework Flutter. Adicionalmente se utilizó Firebase para el almacenamiento de datos, archivos, imágenes y autenticación. Cada una de las características del sistema planteadas fue probada y se realizó una prueba de usabilidad con System Usability Scale.

- La aplicación fue desarrollada utilizando BDD, que se adaptó al modelo de desarrollo iterativo incremental, realizando un *feature* en cada iteración y aumentando la funcionalidad del software con cada iteración. BDD se adaptó al proyecto ya que los administradores del negocio conocen bien sus procesos y fueron capaces de explicar mediante ejemplos dichos procesos que guiaron el comportamiento de la aplicación.

- La aplicación fue desarrollada con un enfoque en el cliente. La aplicación permite que el cliente interactúe con el negocio a través de la interacción con el menú, la generación de pedidos y la visualización de promociones. Estas estrategias fueron implementadas en base a los objetivos del negocio. Adicionalmente, se implementó un sistema de *cashback* y recomendación de amigos como estrategia de fidelización personalizada para el negocio. La aplicación permite ganar saldo por cada compra y utilizarlo en compras futuras. Esto hace que el cliente aumente su confianza hacia el negocio.

- Para asegurar la usabilidad de la aplicación se utilizó System Usability Scale, a través de un formulario. Este se aplicó después de que los clientes tuviesen

la oportunidad de utilizar la aplicación desarrollada en modo depuración. Debido a que los clientes del negocio pertenecen a una población que ha usado previamente aplicaciones móviles, entre ellas de *delivery*. Esto permitió que puedan utilizar la aplicación sin dificultades y la hayan considerado dentro del rango “bueno” bajo el sistema de adjetivos determinado por los autores Bangor, Kortum y Miller.

4.2. Recomendaciones

- En la aplicación desarrollada se utilizó el paquete Provider, sin embargo, este no contará con mejoras ni cambios futuros, por lo que se recomienda la migración de la aplicación al paquete Riverpod.

- Dentro de la aplicación existen componentes de la interfaz gráfica como textos, estilos, botones, entre otros, con características similares. Se recomienda refactorizar estos componentes y extraerlos a otras clases para mejorar la mantenibilidad de la aplicación. De igual manera, las cadenas de caracteres pueden ser extraídas a variables para su fácil modificación en el futuro.

- Al utilizar BDD como metodología de trabajo se debe dar importancia a las conversaciones que se realizan con los involucrados en el proyecto. Es esencial realizar acercamientos con un lenguaje enfocado en el dominio del negocio para poder determinar los objetivos de este, las características de la aplicación que ayudarán a estos objetivos y los ejemplos que indican el comportamiento de la aplicación.

- Tener conocimiento previo de Test Driven Development (TDD) facilita el proceso de desarrollo al utilizar Behavoir Driven Development (BDD).

- En el proyecto, el proceso de desarrollo se realizó mediante pruebas unitarias. Es recomendable realizar también pruebas de *widgets* y pruebas de

integración para aumentar la calidad del software, en este sentido el paquete utilizado para BDD en el proyecto apoyaría en la realización de *widgets test*; pero dado que no es útil para realizar pruebas de integración, se puede complementar con el paquete *flutter_gherkin*.

BIBLIOGRAFÍA

[1] McDonalds, «McDonalds app,» s/f. [En línea]. Available: https://www.mcdonalds.com.ec/apps.

[2] Spoonity, «Sweet & Coffee,» s/f. [En línea]. Available: https://play.google.com/store/apps/details?id=com.spoonity.sweetandcoffee.

[3] Spoonity, «Vaco y Vaca,» s/f. [En línea]. Available: https://play.google.com/store/apps/details?id=com.spoonity.vacoyvaca2.

[4] J. P. Del Alcazar, «Estado Digital Ecuador 2021 – Estadísticas Digitales Actualizadas,» Mentinno, 2021.

[5] La Hora, «El 60% de restaurantes utilizan ´app´ para mejorar sus ventas,» La Hora, 06 Marzo 2019.

[6] Redacción Elcomercio.com, «Los restaurantes se adaptan a la tecnología,» El Comercio , 03 Febrero 2018.

[7] INEC, Directorio de Empresas y Establecimientos 2020, Quito-Ecuador, 2021.

[8] A. M. Sánchez, T. Vayas, F. Mayorga y C. Frerie, «Sector Turístico Ecuador: Alojamiento y Servicios de Comida,» Observatorio Económico y Social de Tungurahua, 2020.

[9] L. A. Lino Sánchez y L. A. Lino Sánchez, Las Tecnologías de Información y Comunicación (TIC) y su influencia en la administración de las pequeñas empresas del Ecuador 2017-2018, Guayaquil, Guayas: Universidad de Guayaquil Facultad de Ciencias Administrativas, 2019.

[10] F. Pérez y M. Ocampo Taco, El progreso tecnológico de las pequeñas y medianas empresas en el cantón Pedro Vicente Maldonado, provincia de Pichincha, Universidad Nacional de Chimborazo, 2019.

[11] A. Ortega, «IDB Invest,» IDB, 20 Octubre 2015. [En línea]. Available: https://idbinvest.org/en/blog/why-do-some-smes-suffer-peter-pan-syndrome#:~:text=In%201983%2C%20Dan%20Kiley%20revolutionized,syndrome%3A%20businesses%20do%20as%20well..

[12] INEC, Tecnologías de la Información y Comunicación-TIC 2019, 2019.

[13] G. P., Raj J. y S. A., «The us a geandad option of cloud computing by small and medium businesses,» Journal of Information Management, vol. 33, nº 5, pp. 861-874, 2013.

[14] C. M. Gerliane y S. J. Iranildo, «The Behavior Driven Development Applied to the Software Quality Test: A Case study Applied to the Promotion of Sports Financing in Brazil,» 14th Iberian Conference on Information Systems and Technologies (CISTI), pp. 19-22, 2019.

[15] Y. Elena, How BDD and Feature Injection can help you be a better..., 2011.

[16] A. Gojko, Impact Mapping: Making a big impact with software products and projects, 2012.

[17] K. J. McDonald, Purpose Based Alignment, s/f.

[18] J. Ferguson Smart, BDD in Action, Behavior-Driven Development for the whole software lifecycle, Manning Publications Co., 2015.

[19] F. de Amorim, L. Costa, R. Adamshuk Silva y F. Rodolfo, «Parallel Testing in Behavior Driven Development,» XII Computer on the Beach, 2021.

[20] Cucumber, «Gherkin Reference,» s/f. [En línea]. Available: https://cucumber.io/docs/gherkin/reference/.

[21] F. Calisir y E. Cevikcan, Industrial Engineering in the Big Data Era, C. A. Hatice, Ed., Springer, 2018.

[22] J. Brooke, «SUS: A quick and dirty usability scale,» Usability Eval. Ind., p. 189, 1995.

[23] M. D. R. Sevilla Gonzalez, L. Moreno Loaeza, L. S. Lazaro Carrera, B. Bourguet Ramirez, A. Vázquez Rodríguez, M. L. Peralta Pedrero y P. Almeda Valdes, «Spanish Version of the System Usability Scale for the Assessment of Electronic Tools: Development and Validation,» JMIR Hum Factors, vol. 7, nº 4, 2020.

[24] A. Bangor, P. Kortum y J. Miller, «Determining What Individual SUS,» Journal of Usability Studies, vol. 4, nº 4, pp. 114-123, Mayo 2009 .

[25] GrupoKFC, «KFC APP - Ecuador, Colombia, Chile y Argentina,» GrupoKFC, [En línea]. Available: https://play.google.com/store/apps/details?id=com.kfcecuador.kfc&hl=es_EC&gl=US. [Último acceso: 25 Febrero 2021].

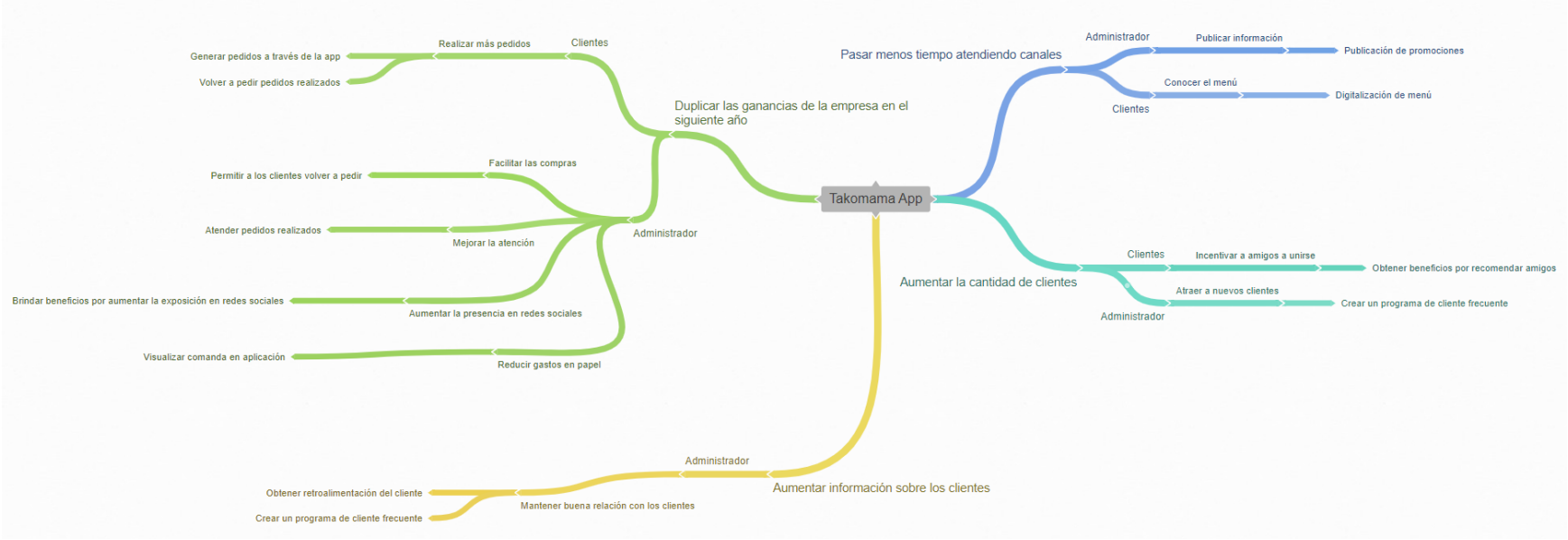
[26] R. K., O. O., Q. A. y P. M., «El e-commercey las Mipymes en tiempos de Covid-19,» Espacios, vol. 41, nº 42, pp. 100-118, 2020

ANEXOS

Anexo I Mapa de impacto

El mapa de impacto de manera digital se puede encontrar siguiendo la siguiente dirección electrónica: https://coggle.it/diagram/Y3-kb2qS3BdYAokA/t/takomama-app/b6aa9685a8a1c314d56fbd23a21d2ac85362dc00c4ce4644c4b26aa430b5cc24

Sin embargo, a continuación, se muestra el diagrama del mapa elaborado:



Anexo II Archivos feature

Los archivos .*feature* usados para crear las pruebas se encuentran en el link:

https://drive.google.com/drive/folders/1U5j5Q4pL5y5aYcxKexoRtq-I95l1YfoC?usp=share_link

Anexo III Resultados de las pruebas por feature

Los resultados de las pruebas de cada *feature* se encuentran en el link:

https://drive.google.com/drive/folders/1UTzQhL3xlIVfDpbuLTr12_khiqASzYNr?usp=share_link

Anexo IV Resultado del comando flutter test

El archivo de texto con el resultado del comando flutter test se encuentra en el siguiente link:

https://drive.google.com/file/d/120AGsgwyTGvJm6cg93XxUK58Y3MeYWah/view?usp=share_link