

ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UN PROTOTIPO DE APLICACIÓN MÓVIL PARA LA TOMA DE PEDIDOS DE COMIDA A DOMICILIO PARA EL SISTEMA OPERATIVO ANDROID

**TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE
INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN**

MARCO ANTONIO GUTIÉRREZ NARANJO

DIRECTOR: ING. XAVIER ALEXANDER CALDERÓN HINOJOSA, M.Sc.

Quito, enero 2021

AVAL

Certifico que el presente trabajo fue desarrollado por Marco Antonio Gutiérrez Naranjo, bajo mi supervisión.

ING. XAVIER ALEXANDER CALDERÓN HINOJOSA, M.Sc.
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Marco Antonio Gutiérrez Naranjo, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

MARCO ANTONIO GUTIÉRREZ NARANJO

DEDICATORIA

Dedico este trabajo a mi familia, en especial a mi padre quien me apoyo de manera incondicional, sacrificándose día tras día para que pueda terminar mis estudios y alcanzar mi meta de ser un profesional.

A mis hermanas que supieron ayudarme en los momentos difíciles, fueron un pilar que me ayudó a levantarme y seguir adelante en los momentos en los que creí que ya no tenía fuerzas para continuar.

A mis sobrinos a quienes espero poder ser un ejemplo de que los sueños pueden cumplirse.

AGRADECIMIENTOS

Agradezco a mi padre por darme su confianza y apoyo a lo largo de mi vida, por nunca perder su fe en mí y guiarme por el camino correcto.

A mis hermanas Verónica, Cristina y Paola, ya que siempre estuvieron a mi lado dándome ánimo y ayudándome en las pequeñas cosas que me llevaron a construir mi camino hasta llegar a mi objetivo.

A mis sobrinos Ricardo, Daniel y Esteban, gracias por todos esos momentos alegres y divertidos que hemos pasado a lo largo de los años, sin ustedes nada sería igual.

A mis compañeros a los que conocí en estos años de estudio.

Un agradecimiento especial al Ingeniero Xavier Calderón por todo el tiempo y esfuerzo invertido en este Proyecto de Titulación sin su ayuda no lo habría logrado.

Finalmente, un agradecimiento a todos los profesores que tuve en mi carrera universitaria, ellos me enseñaron a esforzarme, a ser responsable y de una u otra forma ayudaron a forjar mi carácter, les estaré agradecido toda la vida.

Marco Gutiérrez

ÍNDICE DE CONTENIDO

AVAL.....	I
DECLARACIÓN DE AUTORÍA	II
DEDICATORIA.....	III
AGRADECIMIENTOS	IV
ÍNDICE DE CONTENIDO.....	V
ÍNDICE DE FIGURAS	VIII
ÍNDICE DE TABLAS.....	XII
ÍNDICE DE CÓDIGOS	XIII
RESUMEN.....	XV
ABSTRACT.....	XVI
1. INTRODUCCIÓN	1
1.1. OBJETIVOS.....	1
1.2. ALCANCE	2
1.3. MARCO TEÓRICO	3
1.3.1. TECNOLOGÍAS.....	4
1.3.1.1. React Native.....	4
1.3.1.1.1. <i>Arquitectura de React Native</i>	4
1.3.1.2. React	5
1.3.1.3. Flux.....	5
1.3.1.4. Expo.....	6
1.3.1.5. Bases de datos NO-SQL.....	6
1.3.1.6. Firebase.....	7
1.3.2. HERRAMIENTAS DE DESARROLLO.....	9
1.3.2.1. Node.js	10
1.3.2.2. Yarn	10
1.3.2.3. Expo CLI.....	10
1.3.2.4. Visual Studio Code.....	11
1.3.2.5. Google Maps (API).....	11
1.3.2.6. Marvel App	12
1.3.3. METODOLOGÍA KANBAN.....	12
2. METODOLOGÍA.....	16
2.1. TABLERO KANBAN	16

2.2.	RECOLECCIÓN DE REQUERIMIENTOS	18
2.2.1.	ESTUDIO DE APLICACIONES DE ENTREGA DE COMIDA A DOMICILIO	18
2.2.2.	ENTREVISTAS.....	20
2.3.	HISTORIAS DE USUARIO	20
2.4.	LISTA DE REQUERIMIENTOS.....	21
2.5.	ARQUITECTURA DEL PROTOTIPO	23
2.6.	MÓDULOS.....	24
2.6.1.	MÓDULO ADMINISTRADOR	24
2.6.2.	MÓDULO EMPLEADO.....	25
2.6.3.	MÓDULO CLIENTE.....	25
2.7.	ACTUALIZACIÓN DEL TABLERO KANBAN.....	26
2.8.	DISEÑO	27
2.8.1.	DISEÑO DE LA INTERFAZ DE LA APLICACIÓN.....	27
2.8.2.	DIAGRAMA DE CASOS DE USO.....	27
2.8.3.	DIAGRAMA DE ACTIVIDADES	31
2.8.4.	DIAGRAMA DE ESTRUCTURA	34
2.8.5.	ESTRUCTURA DEL PROYECTO	35
2.8.6.	DISEÑO DE LA BASE DE DATOS	35
2.9.	IMPLEMENTACIÓN.....	38
2.9.1.	INSTALACIÓN DE HERRAMIENTAS	39
2.9.2.	BASE DE DATOS.....	42
2.9.3.	APLICACIÓN ANDROID	43
2.9.3.1.	Módulo Administrador	46
2.9.3.2.	Módulo Cliente	51
2.9.3.3.	Módulo empleado.....	56
2.9.3.4.	Implementación navegación entre pantallas	60
2.9.3.5.	Instalación de Firebase.....	62
2.9.3.6.	Codificación de métodos.....	62
3.	RESULTADOS Y DISCUSIÓN	81
3.1.	PRUEBAS DE FUNCIONAMIENTO.....	82
3.1.1.	PRUEBAS DE FUNCIONAMIENTO MÓDULO ADMINISTRADOR ..	82
3.1.1.1.	Inicio de sesión	83
3.1.1.2.	Agregar un nuevo plato o promoción al menú	84
3.1.1.3.	Editar y eliminar un plato o promoción del menú	86

3.1.1.4.	Agregar nuevo empleado al sistema	88
3.1.1.5.	Editar y eliminar información del empleado	89
3.1.1.6.	Actualizar perfil.....	91
3.1.1.7.	Reporte de pedidos	91
3.1.2.	PRUEBAS DE FUNCIONAMIENTO MÓDULO CLIENTE	91
3.1.2.1.	Registrar usuario	91
3.1.2.2.	Inicio de sesión	94
3.1.2.3.	Visualizar menú.....	95
3.1.2.4.	Actualizar perfil.....	95
3.1.2.5.	Agregar dirección de entrega	96
3.1.2.6.	Información del restaurante.....	97
3.1.3.	PRUEBAS DE FUNCIONAMIENTO MÓDULO EMPLEADO	98
3.1.3.1.	Inicio de sesión	98
3.1.3.2.	Actualizar perfil.....	98
3.1.3.3.	Pedidos pendientes.....	99
3.1.3.4.	Pedidos en ruta.....	99
3.1.3.5.	Pedidos entregados	99
3.1.4.	PRUEBAS DE INTEGRACIÓN	100
3.1.4.1.	Notificaciones.....	101
3.1.4.2.	Chat	101
3.1.5.	RECUPERAR CONTRASEÑA	103
3.2.	PRUEBAS DE VALIDACIÓN	104
3.3.	CORRECCIÓN DE ERRORES.....	108
3.4.	ACTUALIZACIÓN TABLERO KANBAN	110
4.	CONCLUSIONES Y RECOMENDACIONES	112
4.1.	CONCLUSIONES	112
4.2.	RECOMENDACIONES	113
5.	REFERENCIA BIBLIOGRÁFICA	114
	ANEXOS.....	116

ÍNDICE DE FIGURAS

Figura 1.1. Arquitectura del prototipo	3
Figura 1.2. Arquitectura React Native	4
Figura 1.3. Arquitectura Flux y flujo de datos [6]	6
Figura 1.4. Captura de pantalla consola desarrollo de Firebase.....	8
Figura 1.5. Captura de pantalla consola Cloud Messaging de Firebase	8
Figura 1.6. Servicios de Firebase.....	9
Figura 1.7. Captura de pantalla instalación de Yarn	10
Figura 1.8. Captura de pantalla Visual Studio Code	11
Figura 1.9. Captura de pantalla de Marvel App	12
Figura 1.10. Tablero Kanban.....	14
Figura 2.1. Ingreso de datos, (a) Uber Eats, (b) Glovo	18
Figura 2.2. Ingreso de datos, Rappi	19
Figura 2.3. Arquitectura del prototipo	23
Figura 2.4. Arquitectura de notificaciones usando Expo [24]	24
Figura 2.5. Diseño de inicio de sesión y recuperación de contraseña	28
Figura 2.6. Diseño de página principal de (a) Administrador, (b) Cliente, (c) Empleado.....	28
Figura 2.7. Caso de uso: Módulo administrador	30
Figura 2.8. Caso de uso: Módulo cliente	30
Figura 2.9. Caso de uso: Módulo empleado	31
Figura 2.10. Diagrama de actividades crear plato	32
Figura 2.11. Diagrama de actividades ingresar nuevo empleado	33
Figura 2.12. Diagrama de actividades realizar un pedido.....	34
Figura 2.13. Diagrama de estructura.....	35
Figura 2.14. Estructura del proyecto Asadero de Lali en React Native.....	36
Figura 2.15. Base de datos de Firebase.....	36
Figura 2.16. Estructura base de datos en Firebase.....	37
Figura 2.17. Descarga de Node.js.....	39
Figura 2.18. Instalación Expo.CLI	40
Figura 2.19. Instalación Visual Studio Code	40

Figura 2.20. Visual Studio Code.....	41
Figura 2.21. Extensiones VS Code	41
Figura 2.22. Descarga de Expo client	42
Figura 2.23. Base de datos: Usuarios	43
Figura 2.24. Fragmento de la base de datos de platos del menú	43
Figura 2.25. Interfaz de Visual Studio Code	45
Figura 2.26. Inicio de sesión	45
Figura 2.27. Recuperación de contraseña.....	46
Figura 2.28. Página principal rol administrador	47
Figura 2.29. UI de (a) Platos en el menú, (b) Actualización de datos del menú, (c) Ingreso de un nuevo plato en el menú.	48
Figura 2.30. UI de las promociones del restaurante	50
Figura 2.31. Registro de nuevo empleado.....	50
Figura 2.32. (a) Menú desplegable Administrador, (b) Perfil de usuario, (c) Pantalla información del restaurante.....	51
Figura 2.33. Formulario de registro del cliente e ingreso de dirección	52
Figura 2.34. Pantalla principal rol cliente.....	53
Figura 2.35. (a) Carrito de compras, (b) Datos del usuario y valor de la compra ..	53
Figura 2.36. Vista de perfil del cliente	54
Figura 2.37. Actualización de contraseña.....	54
Figura 2.38. Menú desplegable del usuario cliente	55
Figura 2.39. Ingreso de nueva dirección	55
Figura 2.40. Página principal Empleado.....	56
Figura 2.41. Pedidos pendientes.....	57
Figura 2.42. (a) Pedidos en camino, (b) Dirección de entrega.....	57
Figura 2.43. Pedidos entregados	58
Figura 2.44. Menú desplegable del empleado.....	58
Figura 2.45. Pantalla platos del menú	68
Figura 2.46. Carrito de compras.....	71
Figura 2.47. Datos del administrador actualizados	75
Figura 2.48. Registrar la aplicación móvil en Firebase	75
Figura 2.49. Descarga archivo de configuración google-services.json	76
Figura 2.50. Ruta del archivo google-services.json	76

Figura 2.51. Obtención de clave del servidor de Firebase.....	77
Figura 2.52. Configuración correo de recuperación de contraseña	79
Figura 2.53. Recuperación de contraseña.....	79
Figura 2.54. Reporte de pedidos.....	80
Figura 3.1. Inicio de sesión administrador	83
Figura 3.2. Mensajes de error con datos incorrectos.....	83
Figura 3.3. Pantalla principal administrador	84
Figura 3.4. Agregar nuevo plato.....	84
Figura 3.5. Ingreso de un nuevo plato	85
Figura 3.6. Mensajes de error nuevo plato	85
Figura 3.7. Nueva promoción	86
Figura 3.8. Ingreso nueva promoción.....	86
Figura 3.9. Menú módulo administrador	87
Figura 3.10. Editar plato del menú	87
Figura 3.11. Mensaje de eliminación del plato.....	88
Figura 3.12. Ingreso de un nuevo empleado	88
Figura 3.13. Registro de nuevo empleado.....	89
Figura 3.14. Ingreso de nuevo empleado con información incorrecta.....	89
Figura 3.15. Actualización de nombre y número de teléfono de empleado.....	90
Figura 3.16. Eliminación de cuentas de clientes.....	90
Figura 3.17. Pantalla actualización de perfil	91
Figura 3.18. Reporte de pedidos.....	92
Figura 3.19. Pantalla registro de nuevo cliente.....	92
Figura 3.20. Mapa de ubicación	93
Figura 3.21. Registro de información nuevo cliente.....	93
Figura 3.22. Mensajes de alerta	94
Figura 3.23. Inicio de sesión cliente	94
Figura 3.24. Platos del menú	95
Figura 3.25. Carrito de compras.....	96
Figura 3.26. Actualizar datos del cliente.....	96
Figura 3.27. Ingreso de una nueva dirección de entrega.....	97
Figura 3.28. Pantalla de información	97

Figura 3.29. Inicio de sesión empleado	98
Figura 3.30. Actualización datos empleado	98
Figura 3.31. (a) Lista de pedidos pendientes, (b) detalle del pedido.....	99
Figura 3.32. (a) Lista de pedidos en ruta, (b) Mapa de lugar de entrega	100
Figura 3.33. Pedidos entregados	100
Figura 3.34. (a) Confirmación del pedido, (b) Notificación de nuevo pedido	101
Figura 3.35. (a) Pedidos pendientes, (b) Notificación de pedido en camino	102
Figura 3.36. Chat	102
Figura 3.37. Datos del Chat	103
Figura 3.38. Recuperar contraseña	103
Figura 3.39. Cambio de contraseña	104
Figura 3.40. Resultados entrevistas clientes	105
Figura 3.41. Resultados entrevistas empleados.....	107

ÍNDICE DE TABLAS

Tabla 1.1. Características metodología Kanban.....	13
Tabla 1.2. Metodología Kanban ventajas	14
Tabla 1.2. Metodología Kanban ventajas	15
Tabla 2.1. Lista de tareas Tablero Kanban.....	17
Tabla 2.1. Lista de tareas Tablero Kanban.....	18
Tabla 2.2. Resumen de funcionalidades	19
Tabla 2.3. Historias de usuario.....	20
Tabla 2.4. Módulos del prototipo	25
Tabla 2.5. Actualización del Tablero Kanban	26
Tabla 2.6. Tablero Kanban. Implementación.....	38
Tabla 3.1. Tablero Kanban actualización	81
Tabla 3.2. Resultados entrevistas clientes	104
Tabla 3.3. Resultados entrevista administrador.....	106
Tabla 3.4. Resultados entrevista empleados.....	107
Tabla 3.5. Correcciones módulo administrador	108
Tabla 3.6. Correcciones módulo cliente	109
Tabla 3.7. Correcciones módulo empleado	110
Tabla 3.8. Correcciones generales.....	110
Tabla 3.9. Tablero Kanban actualización final	110

ÍNDICE DE CÓDIGOS

Código 1.1. Ejemplo de objeto JSON [10].....	7
Código 2.1. Notificaciones con el API de Expo [25].....	24
Código 2.2. Importación de librerías.....	44
Código 2.3. Fragmento de código para pantalla de Inicio de sesión.....	46
Código 2.4. Fragmento de código pantalla de inicio rol administrador.....	47
Código 2.5. Importación de librerías para pantalla de inicio rol administrador.....	49
Código 2.6. Fragmento de código creación de un nuevo plato del menú.....	49
Código 2.7. Fragmento de código registro nuevo cliente.....	52
Código 2.8. Fragmento de código para el ingreso de datos del nuevo cliente.....	55
Código 2.9. Fragmento de código para UI de nuevo cliente.....	56
Código 2.10. Implementación de la UI pedidos pendientes.....	59
Código 2.11. Función que carga datos de la base de datos en la aplicación.....	59
Código 2.13. Stack de navegación para el módulo administrador.....	60
Código 2.14. Stack de navegación para el módulo cliente.....	61
Código 2.15. Método navigation.....	61
Código 2.16. Instalación SDK de Firebase.....	62
Código 2.17. Inicialización de Firebase.....	62
Código 2.18. Función nuevoCliente.....	63
Código 2.19. Función login.....	64
Código 2.20. Función verUsuario.....	64
Código 2.21. Fragmento de código para crear nuevo plato del menú.....	65
Código 2.22. Permisos para abrir la galería en Android con React Native.....	66
Código 2.23. Fragmento de código que carga el menú en la aplicación móvil.....	67
Código 2.24. Fragmento de la función renderItemMenu.....	67
Código 2.25. Fragmento de código importación de librerías.....	68
Código 2.26. Fragmento del código ubicación de entrega.....	69
Código 2.27. Permiso para la ubicación del usuario.....	70
Código 2.28. Fragmento de código carrito de compras.....	71
Código 2.29. Función actualizar.....	72

Código 2.30. Función useEffect.....	72
Código 2.31. Importación librería expo-notifications	73
Código 2.32. Fragmento de código archivo app.json	73
Código 2.33. Función registerForPushNotificationsAsync	74
Código 2.34. Función updateToken.....	74
Código 2.35. Función sendPushNotification	77
Código 2.36. Fragmento de código recuperación de contraseña	78
Código 2.37. Función onSubmit	78
Código 2.38. Fragmento de código filtro por fecha	80

RESUMEN

En el presente trabajo de titulación se desarrolla un prototipo de aplicación móvil, para la toma de pedidos de comida a domicilio. El desarrollo del presente trabajo abarca las necesidades y recomendaciones del restaurante El Asadero de Lali, para ofrecer a sus clientes habituales, así como a nuevos usuarios una forma de acceder a sus productos.

El prototipo es desarrollado con React Native el cual, es un Framework JavaScript para el desarrollo de aplicaciones nativas, además de la plataforma Firebase de Google en la cual se almacenarán y recuperarán los datos para el prototipo.

La aplicación móvil permite el registro de usuarios, tanto de los clientes como de los empleados, estos últimos serán registrados por el administrador del restaurante el cual estará previamente registrado en la base de datos. Se empezará con el diseño del prototipo de la aplicación mediante la metodología de desarrollo ágil Kanban.

En el primer capítulo se expone el marco teórico sobre las herramientas usadas en el desarrollo del prototipo, las tecnologías implementadas y la metodología de desarrollo Kanban.

En el segundo capítulo se realizan las entrevistas a los clientes, se recogen los requerimientos del prototipo y se presenta el diseño del mismo.

En el tercer capítulo se muestran los resultados de las entrevistas de validación, así como de las pruebas realizadas.

En el cuarto capítulo son presentadas las conclusiones y recomendaciones que se obtuvieron al desarrollar este Trabajo de Titulación.

Finalmente, en los anexos se adjuntan: el modelo de entrevista, manual de usuario y el código implementado en la aplicación.

PALABRA CLAVE: Android, React, React Native, toma de pedidos, Firebase, aplicación móvil.

ABSTRACT

In this capstone project, a mobile application prototype is developed for taking food orders at home. The development of this work covers the needs and recommendations of the restaurant El Asadero de Lali, to offer its regular customers, as well as new users, a way to access its products.

The prototype is developed with React Native which is a JavaScript Framework for the development of native applications, in addition to Google's Firebase platform in which the data for the prototype will be stored and retrieved.

The mobile application allows the registration of users, both customers and employees, the latter will be registered by the restaurant administrator who will be previously registered in the database. It will begin with the design of the application prototype using the agile Kanban development methodology.

In the first chapter the theoretical framework on the tools used in the development of the prototype, the implemented technologies, and the Kanban development methodology is exposed.

In the second chapter, interviews with clients are carried out, the requirements of the prototype are collected, and its design is presented.

The third chapter shows the results of the validation interviews, as well as the tests carried out.

The fourth chapter presents the conclusions and recommendations that were obtained when developing this Degree Work.

Finally, the annexes include: the interview model, user manual, and the code implemented in the application.

KEYWORDS: Android , React, React Native, taking orders, Firebase, mobile app.

1. INTRODUCCIÓN

Actualmente vivimos en una época de rápido y constante cambio en la cual el uso de la tecnología ha cambiado la forma en la que se realizan las actividades del día a día, un ejemplo en el cual se nota claramente este avance en la tecnología son los dispositivos móviles como los teléfonos inteligentes o smartphone. El instrumento principal de los smartphone son las aplicaciones móviles, es así que existe una aplicación móvil para casi cualquier actividad imaginable, ya sea para trabajo, diversión o salud, y los negocios de comida no son la excepción.

Las empresas de comida aprovechan la versatilidad de las aplicaciones móviles para promocionar sus productos y llegar a un mayor número de posibles clientes.

El presente trabajo de titulación utiliza la metodología ágil Kanban para el desarrollo de un prototipo de aplicación móvil utilizando el framework React Native, para la toma de pedidos de comida a domicilio para los usuarios del restaurante El asadero de Lali, el prototipo permite, visualizar los diferentes menús y promociones con los que el restaurante cuenta, permite actualizar y editar dicha información al usuario administrador el cual será el administrador del restaurante.

El prototipo utilizará información suministrada por los clientes para determinar el sitio en el cual debe ser entregado el pedido.

1.1. OBJETIVOS

El objetivo general de este Proyecto Técnico es:

Desarrollar un prototipo de aplicación móvil que permita la toma de pedidos de comida a domicilio para el sistema operativo Android.

Los objetivos específicos de este Proyecto Técnico son:

- Definir los fundamentos teóricos necesarios para el desarrollo del proyecto.
- Diseñar los módulos del prototipo de la aplicación móvil.
- Implementar los módulos del prototipo de aplicación móvil en función de los módulos diseñados.
- Analizar los resultados obtenidos en las pruebas de funcionamiento del prototipo.

1.2. ALCANCE

En el presente trabajo de titulación se detalla el desarrollo de un prototipo de aplicación móvil para el envío de comida a domicilio para el restaurante El Asadero de Lali. La aplicación permitirá a los clientes crear una cuenta de usuario, por medio de la cual, el cliente podrá acceder al menú del restaurante, así como también a las promociones que el restaurante disponga y en el caso de así desearlo realizar un pedido a domicilio.

La aplicación tendrá un usuario administrador el cual estará precargado en la base de datos, este usuario será el dueño del restaurante El Asadero de Lali, el administrador será el único usuario que puede registrar en el sistema a nuevos empleados, posteriormente los empleados podrán actualizar sus datos, así como su contraseña en la aplicación móvil, además, el usuario administrador es el encargado de editar, actualizar, borrar y crear nuevos menús o promociones de acuerdo a su conveniencia, también podrá eliminar a los empleados del sistema.

La aplicación estará basada en la metodología Kanban, se utilizará React Native para el desarrollo de la aplicación, además los datos serán guardados en Firebase, el cual provee la base de datos para el prototipo. Además, se usará herramientas adicionales para facilitar el desarrollo del prototipo, herramientas como el API de Google maps, Expo el cual es un grupo de herramientas y librerías que permite el desarrollo de aplicaciones nativas en un entorno JavaScript [1].

La aplicación móvil contará con tres roles de usuario: Administrador, Empleado y Cliente.

El usuario con el rol Cliente, que al ingresar por primera vez a la aplicación deberá registrarse para crear una cuenta ingresando sus datos y una contraseña; el usuario con el rol Administrador que estará precargado en la base de datos; y, el usuario con el rol Empleado que será ingresado a la base de datos por el Administrador.

Para iniciar sesión e ingresar a la aplicación móvil los tres tipos de usuario deberán ingresar con sus credenciales: usuario y contraseña, para ser validados.

Cada rol realizará las siguientes actividades:

- **Cliente:** Es el usuario que desea realizar un pedido de comida a domicilio, para lo cual realiza el proceso de registro en la aplicación, podrá visualizar el menú del restaurante, así como la información de cada uno de los platos ofertados, su precio, y ofertas del restaurante. Realiza el pedido de comida y antes de

confirmar el pedido, se visualizarán los productos, así como el precio total del pedido.

- **Empleado:** Es el encargado de visualizar el pedido hecho por el cliente, así como encargarse de enviar o entregar el pedido al lugar especificado en la orden de compra. Notificará al cliente que su pedido está en camino de forma automática al momento de entregar el pedido al motorizado, el cual es un empleado del restaurante que realiza las funciones de entregar los pedidos cuando así se lo necesite, también confirma la entrega del pedido al cliente.
- **Administrador:** Es el encargado de realizar los procesos de creación, edición y eliminación de los datos correspondientes a cada menú presente en la carta de platos, entre estos los precios, actualizar la información de precios, agregar un nuevo aviso sobre ofertas del día. Tendrá la responsabilidad de registrar a nuevos empleados dentro de la aplicación para que hagan uso de ella. Podrá visualizar los pedidos de comida realizados por los clientes.

Cada usuario en los tres roles indicados tendrá una interfaz diferente.

La arquitectura del prototipo se muestra en la Figura 1.1.

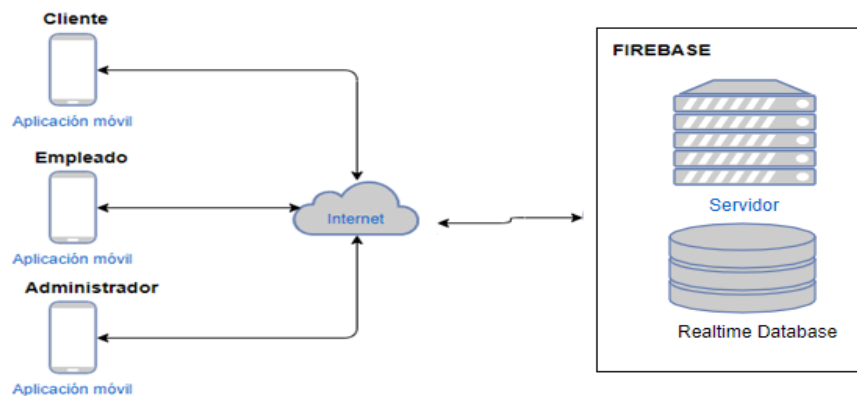


Figura 1.1. Arquitectura del prototipo

1.3. MARCO TEÓRICO

En este apartado se redacta conceptos importantes, así como también, se especifican las tecnologías y herramientas de desarrollo usadas a lo largo del diseño y desarrollo de este prototipo de aplicación móvil.

1.3.1. TECNOLOGÍAS

1.3.1.1. React Native

React Native es un framework de código abierto para el desarrollo nativo de aplicaciones móviles, que, a su vez, está basado en React. Utiliza varias tecnologías estándar de la web tales como JavaScript (JSX), HTML y CSS, pero a pesar de esto sus aplicaciones son nativas completamente, esto quiere decir que las aplicaciones desarrolladas en React Native son fluidas y veloces por lo que son equiparables a cualquier aplicación desarrollada con la tradicional tecnología de Android como es JAVA o Kotlin.

1.3.1.1.1. Arquitectura de React Native

El framework React Native al usar JavaScript como la base para su programación no cuenta con un método para compilar de manera directa el código sobre las plataformas móviles sean estas Android o iOS. De esta manera, React Native emplea compiladores de código para cada plataforma, en otras palabras, hay un compilador para Java o Kotlin, los cuales producen aplicaciones que se pueden instalar en Android y de esta misma forma, hay un compilador para Objective-C o Swift que producen aplicaciones que se instalan en iOS.

React Native crea un puente hacia los compiladores y crea la aplicación ya sea para Android o iOS. La Figura 1.2 representa la arquitectura de React Native, así como el salto de código elaborado en JavaScript a artefactos multiplataforma [2].

El puente posibilita a React Native relacionarse con las librerías nativas ya sea de iOS o Android mediante Java o C++, de esta forma el código JavaScript permanece ejecutándose dentro del JavaScript Core, similar a JVM en Java.

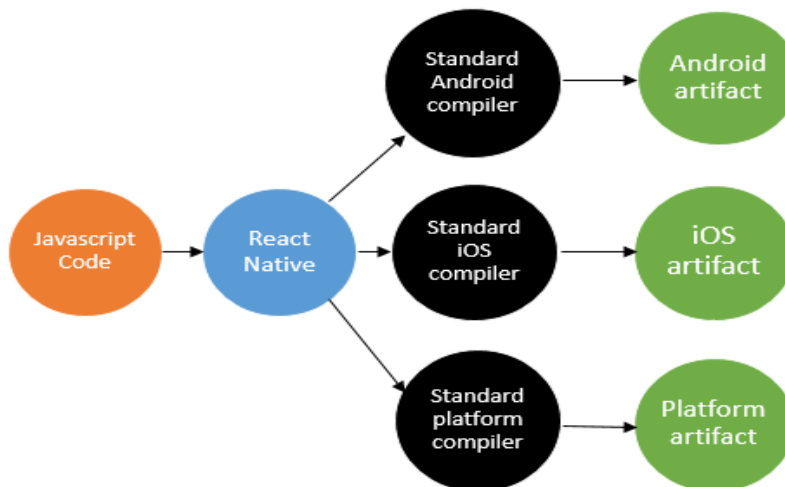


Figura 1.2. Arquitectura React Native [2]

React Native dispone de algunas librerías que añaden funcionalidad. Algunas de las más relevantes son presentadas a continuación:

- *Native base*: es una librería de código abierto que proporciona componentes de UI (Interfaz de usuario) para ser usados en React Native.
- *React Navigation*: librería que proporciona una forma fácil para navegar entre pantallas en una aplicación React Native, también posee alternativas para personalizar cada pantalla, así como las rutas entre las mismas.
- *React Native Elements*: proporciona un kit de interfaz de usuario para aplicaciones creadas con React Native.

1.3.1.2. React

React es una librería JavaScript, la cual fue desarrollada por Facebook para el diseño de interfaces de usuario basándose en componentes. Es flexible, rápida y de un alto rendimiento. Los componentes son la unidad más básica en la implementación de aplicaciones React; en palabras simples son la unidad fundamental de React. Estos componentes establecen como los elementos del DOM (Document Object Model) son creados y como es la interacción entre estos y los usuarios.

Los componentes están encapsulados, esta característica los hace fácil de probar, así como también facilitar su reutilización [3]. Los componentes son capaces de administrar su estado de manera que pueden reaccionar de manera inmediata a los cambios que se dan en los datos. Posibilita que las vistas se enlacen con los datos, de este modo si los datos sufren algún cambio, también se modifican las vistas [4]. Esto da como resultado que, ante cualquier cambio en los datos, automáticamente el componente vuelve a ser renderizado en la pantalla.

Además del lenguaje JavaScript, React hace uso de JSX, la cual es una sintaxis similar a HTML, la cual fue creada explícitamente para que el código sea legible en el caso de que se necesite implementar de manera directa secciones de la interfaz de usuario [5].

1.3.1.3. Flux

Flux es una arquitectura desarrollada por Facebook, para conducir el flujo de los datos en una aplicación. La razón detrás de esto es el de reemplazar el flujo bidireccional de los datos por un flujo unidireccional, de esta manera, el control y manipulación de los datos se facilita, y labores como la depuración se vuelven más fáciles.

Flux establece cuatro capas en su arquitectura,

- *Actions*: que establece y despacha acciones.
- *Store*: el cual abarca al estado y lo modifica.
- *Dispatchert*: acepta acciones y se comporta como una sola devolución de requerimiento al store.
- *Views*: representa el estado de los datos de forma en la interfaz gráfica.

Flux establece una arquitectura en la que el flujo de datos va en una sola dirección es decir un flujo unidireccional. Los datos se trasladan desde el *View* a través de los *Actions* y arriban a un *Store* en el cual la vista se actualiza [6]. En la Figura 1.3 se observa el camino de los datos entre cada uno de los elementos de la arquitectura Flux.

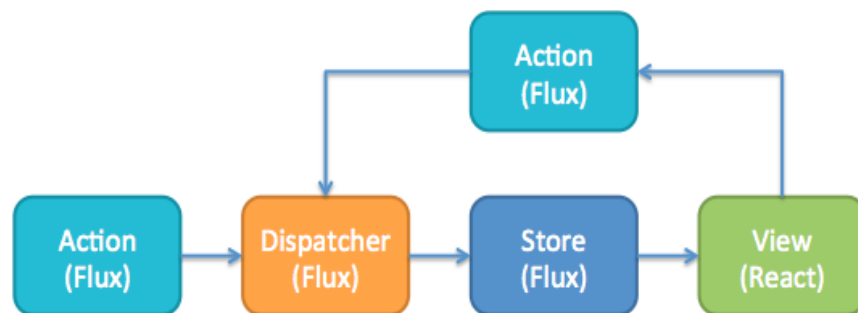


Figura 1.3. Arquitectura Flux y flujo de datos [6]

1.3.1.4. Expo

Expo es un framework y una plataforma para aplicaciones universales React. Es un conglomerado de herramientas y servicios implementados alrededor de React Native y plataformas nativas que apoyan en la implementación, construcción, desarrollo rápido en iOS, Android y aplicaciones web con base de código JavaScript/TypeScript. [7].

Expo usa Expo SDK, el cual es una librería nativa y de JavaScript que permite acceso a las funcionalidades de los dispositivos tales como la cámara, contactos, galería, entre otros, sin la necesidad de manipular código nativo, con la ventaja de que el código se vuelve portable, esto debido a que la aplicación puede correr en cualquier dispositivo que tenga instalado la aplicación de Expo [7].

1.3.1.5. Bases de datos NO-SQL

Las bases de datos no relacionales, son las que no poseen un identificador que relacione un segmento de datos con otro en contraparte a las relacionales [8]. Un gran número de bases de datos NO-SQL (Not Only SQL) comparten varias características entre las que sobresale que no es indispensable definir un esquema que represente la estructura de los

datos, en virtud de que guardan datos semiestructurados. Esta falta de esquema justifica la superior flexibilidad de los sistemas NO-SQL frente a los sistemas relacionales. Una de las ventajas que sobresale es la posibilidad de contar con datos no uniformes, es decir que algunos elementos de una misma entidad pueden ser de diferentes tipos o ser opcionales y la evolución de los datos es sencilla en vista de que no es necesario cambiar ningún esquema [9].

Existen algunos tipos de bases de datos NO-SQL:

- *Clave-valor*: Admiten escalado horizontal a escalas que las demás bases de datos no pueden.
- *Documentos*: En el código de aplicación, los datos se muestran como un objeto o un documento tipo JSON. En el Código 1.1 se indica un ejemplo de un objeto JSON.
- *Gráficos*: El objetivo principal de estas bases de datos es el de facilitar la implementación y ejecución de aplicaciones con datos extremadamente conectados.

```
{
  "users": {
    "alovelace": {
      "name": "Ada Lovelace",
      "contacts": { "ghopper": true },
    },
    "ghopper": { ... },
    "eclarke": { ... }
  }
}
```

Código 1.1. Ejemplo de objeto JSON [10]

1.3.1.6. Firebase

Es una plataforma suministrada por Google la cual funciona como un BaaS (Backend-As-A-Service) la cual proporciona herramientas para el desarrollo web y de aplicaciones móviles y de esta manera facilitar el trabajo de los programadores al no tener que preocuparse en administrar un servidor.

Firebase cuenta con las siguientes características [11]:

- *Desarrollo*: Permite minimizar el tiempo de desarrollo al implementar diferentes funciones entre las que se puede resaltar, la detección de errores, guardar todo en la nube y tener una estadística de los datos consumidos por la aplicación como muestra la Figura 1.4.

- **Análítica:** Se cuenta con un control del rendimiento de la aplicación a través de métricas todo esto de forma gratuita. Con estos datos analíticos se toma decisiones con base en datos reales.

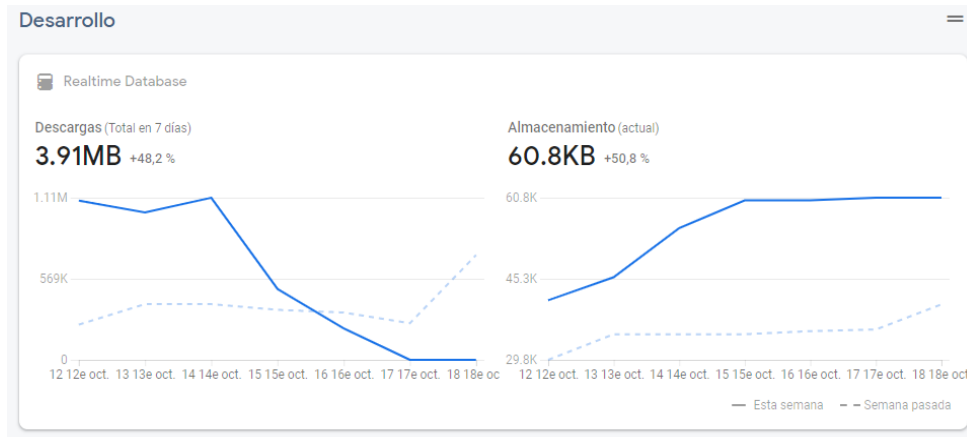


Figura 1.4. Captura de pantalla consola desarrollo de Firebase

- **Poder de crecimiento:** Nos permite gestionar de forma ágil a los usuarios de las aplicaciones, con un plus el cual es que atraer nuevos usuarios al enviar invitaciones o mediante notificaciones y un registro de la actividad como se muestra en la Figura 1.5.

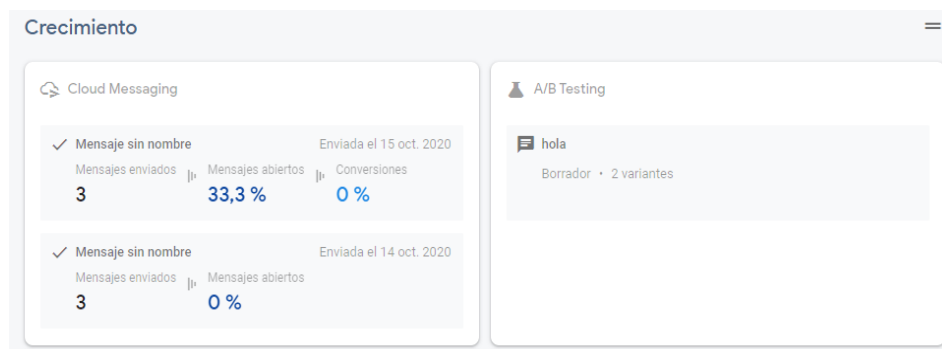


Figura 1.5. Captura de pantalla consola Cloud Messaging de Firebase

- **Rapidez:** Poner en marcha Firebase es fácil y rápido, ya que provee un SDK (Software Development Kit) a los desarrolladores para de esta manera íntegra los servicios que ofrece.

Firebase cuenta con los siguientes servicios:

- **Realtime Database:** Es una base de datos NoSQL almacenada en la nube. Estos datos son guardados en formato JSON y sincronizados en tiempo real con los clientes activos. Todos los clientes de la aplicación comparten una instancia de la

base de datos, lo cual permite que reciban de forma automática actualizaciones de los datos.

- *Cloud Storage*: Es un servicio que permite almacenar archivos, en el cual se puede guardar archivos de audio, imágenes, videos u otro tipo de archivos que genere el usuario.
- *Authentication*: Firebase Authentication provee de servicios de backend, SDK sencillos de usar y bibliotecas UI ya elaboradas, con el fin de autenticar usuarios en la aplicación. Permite autenticación mediante contraseñas, proveedores de identidad como Facebook, Google y Twitter, número de teléfono, y muchos más. El servicio de autenticación está ligado a otros servicios proporcionados por Firebase.
- *Cloud Functions*: es un framework sin servidores el cual nos permite ejecutar código de backend como respuesta a solicitudes HTTPS. El código se guarda en la nube de Google en un ambiente administrado por Firebase, por lo que en consecuencia no se requiere administrar ni escalar servidores propios.
- *Hosting*: es un servicio que proporciona Firebase para el alojamiento seguro de contenido web, este contenido puede ser dinámico o estático.

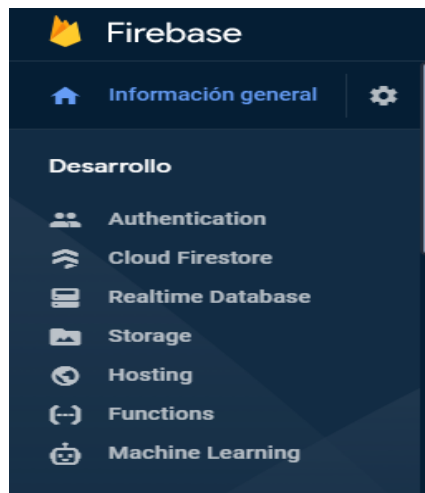


Figura 1.6. Servicios de Firebase

Firebase entrega un SDK a los desarrolladores para enlazar sus servicios de autenticación, almacenamiento, etc. en sus aplicaciones. Por otra parte, el SDK añade seguridad a las transacciones de carga y descarga de los datos.

1.3.2. HERRAMIENTAS DE DESARROLLO

A continuación, se detallan las herramientas de desarrollo usadas en el proyecto.

1.3.2.1. Node.js

Node.js es un ambiente de ejecución JavaScript multiplataforma y de código abierto; ofrece una forma de crear programas de forma rápida y escalable. Ejecuta el motor JavaScript V8 basado en Chrome [12]. Para React Native ejecuta el paquete Metro el cual es un paquete JavaScript que compila varios archivos JavaScript en un solo archivo.

La descarga de nodejs se la realiza desde su página oficial <https://nodejs.org/es/> y se lo puede instalar en cualquier sistema operativo. Al instalar nodejs, vendrá incluido con un gestor de paquetes por defecto llamado npm (Node Package Manager) el cual ayuda a distribuir paquetes, además de agregar dependencias a un proyecto.

1.3.2.2. Yarn

Yarn es un gestor de dependencias JavaScript, orientado hacia la velocidad y seguridad. Yarn guarda en cache todos los paquetes descargados por lo que no es obligatorio volver a descargarlos.

Se descarga de su página oficial <https://yarnpkg.com/> es multiplataforma y de código abierto. Para su instalación se debe elegir el sistema operativo y seguir las instrucciones. Una captura de pantalla del proceso de instalación de Yarn es mostrado en la Figura 1.7.

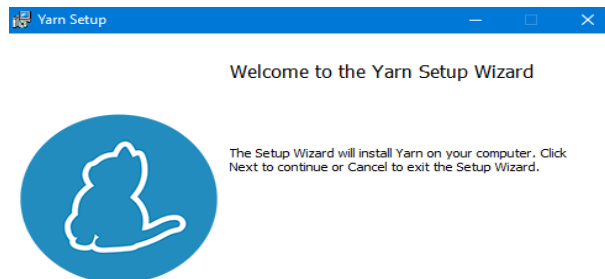


Figura 1.7. Captura de pantalla instalación de Yarn

Para instalar dependencias mediante yarn se usa el comando `yarn add nombre_del_paquete`, de la misma manera para desinstalar una dependencia se usa el comando `yarn remove nombre_del_paquete` [13].

1.3.2.3. Expo CLI

Expo CLI es una utilidad de línea de comandos que funciona como interfaz entre el desarrollador y Expo, el cual sirve para crear nuevos proyectos React Native, publicar la

aplicación JavaScript, crear los archivos binarios apk y ipa entre otras opciones [14]. Para su instalación con el gestor yarn se ejecuta el comando: `yarn global add install expo-cli`.

Expo CLI se la puede usar desde le terminal de comandos o en una interfaz basada en web. El comando: `expo init nombre_del_proyecto` permite crear un proyecto nuevo.

1.3.2.4. Visual Studio Code

Es un editor de código fuente, el cual está disponible para los sistemas operativo Windows, macOS y Linux, el cual se ejecuta en el escritorio. VS Code viene con soporte integrado para JavaScript, TypeScript y Node.js además posee un amplio ecosistema de extensiones para distintos lenguajes de programación [15]. Viene integrado con un depurador, una terminal y opciones de personalización. Dispone de una tienda de extensiones desde la cual se descargan complementos, paquetes de lenguajes y otras dependencias. Una captura de pantalla de VS Code en la que se presenta la tienda de extensiones junto con el editor es mostrada en la Figura 1.8.

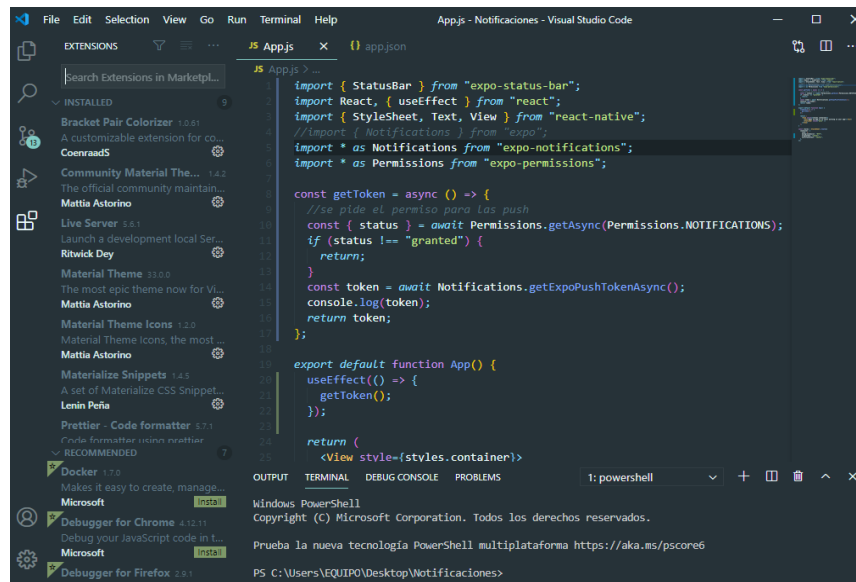


Figura 1.8. Captura de pantalla Visual Studio Code

1.3.2.5. Google Maps (API)

Google Maps es un servicio proporcionado por Google, el cual es gratuito y provee información geográfica. Google Maps posee una API (Interfaz de programación de aplicación) de JavaScript la cual admite personalización para mostrar los mapas en páginas web, dispositivos móviles y adecuar a las exigencias del sistema [16].

Para utilizar la API de Google Maps, en el prototipo se descarga el paquete `react-native-maps` mediante el comando: `expo install react-native-maps`.

El prototipo utiliza el API de Google Maps para encontrar y guardar la ubicación a la cual el cliente desea se envíe el pedido de comida, también se lo utiliza para que el encargado de entregar el pedido sepa la dirección exacta a donde debe dirigirse.

1.3.2.6. Marvel App

Marvel app es una herramienta que permite crear prototipos para aplicaciones móviles, así como para sitios web, desde el navegador [17].

El prototipo de la aplicación móvil se lo desarrolla en Marvel App para lo cual, se crea diseños sin código y así validar la experiencia del usuario para obtener una retroalimentación del diseño. En la Figura 1.9 se observa una captura de la herramienta Marvel App.

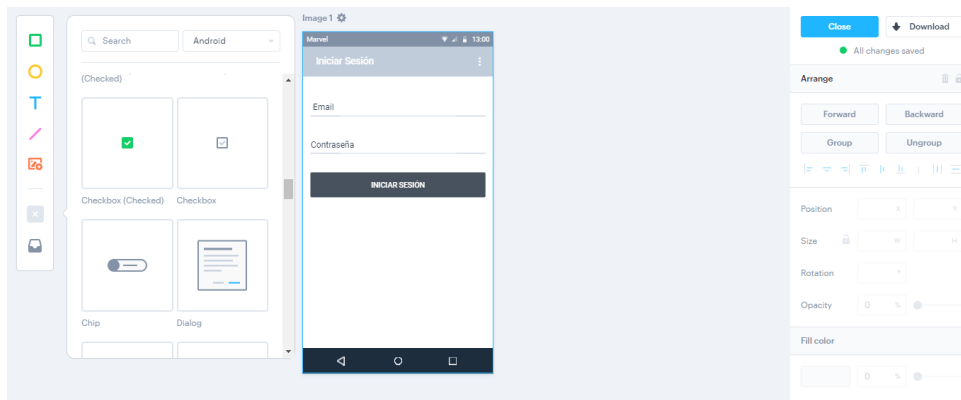


Figura 1.9. Captura de pantalla de Marvel App

1.3.3. METODOLOGÍA KANBAN

Kanban es un método para administrar el trabajo que emergió en Toyota Production System (TPS).

La palabra Kanban proviene del idioma japonés y en su traducción literal quiere decir “tarjeta con signos” o “señal visual”; es una herramienta de desarrollo la cual procede de la filosofía Lean, de tipo “pull” esto se traduce en que los recursos toman la decisión de cuándo y cuánto trabajo se van a realizar [18].

Esta metodología está fundamentada en la optimización de procesos y resaltar la respuesta al cambio por encima de seguir un plan, lo cual permite conseguir una respuesta mucho

más veloz sobre otras metodologías de desarrollo como Scrum. Por otro lado, Kanban es una metodología que se acomoda a los requerimientos variables del cliente, de esta manera los progresos del producto son entregados regularmente y de manera incremental [18].

Kanban usa la idea de crear un backlog (bloque de tareas) del producto a través de un conjunto de tareas priorizadas, Kanban observa el flujo de trabajo (*workflow*), en otras palabras, secciona el trabajo en tareas y hace un control de las tareas en progreso (*work in progress*) así de esta forma, al existir poco trabajo en progreso se agrega la tarea con prioridad más alta del backlog.

Algunas características de Kanban se representan en la Tabla 1.1.

Tabla 1.1. Características metodología Kanban

Característica	Descripción
Observa el Workflow	Se escribe cada tarea en una tarjeta y se la ubica en el tablero. Se usa columnas con la descripción para identificar en qué etapa del proceso está ubicada la tarea.
Trabajo en curso	Se establece límites en relación a cuantas tareas pueden ser evaluadas a la vez en cada flujo de trabajo.
Mide el "Lead Time"	Se mide el tiempo para completar una tarea o en su defecto el tiempo que la tarea se demora en atravesar las columnas del tablero.

Para que Kanban sea empleado en el desarrollo de software, es requerido que el trabajo se divida en partes. Sin embargo, Kanban no establece la estructura ni magnitud de los partes del trabajo o tareas. Las tareas deben ser mostradas en el tablero Kanban, como se

muestra en la Figura 1.10. Conforme el trabajo vaya avanzando en el ciclo de vida del desarrollo, las tareas se trasladan de un estado a otro hasta que llegan a la última columna.

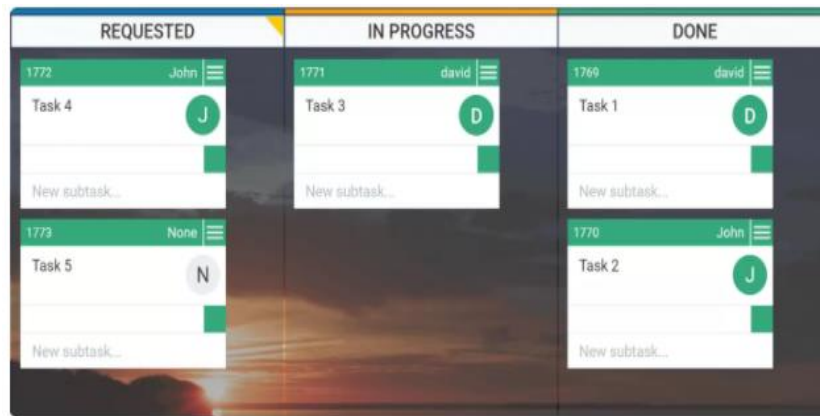


Figura 1.10. Tablero Kanban

Cada columna simboliza un determinado estado al interior del *workflow*, el número de columnas son establecidas en relación a la experiencia del desarrollador y a los requerimientos del proyecto, no están establecidos el número de columnas ni tampoco estados dentro de la metodología Kanban [19].

En Kanban es necesario establecer prioridades en las tareas y se debe delimitar el umbral de trabajo en curso, en otras palabras, es necesario conocer la cantidad de tareas que se pueden llevar a cabo en cada fase ya que hay una cantidad de trabajo ideal que se realiza sin sacrificar eficiencia, adicionalmente para iniciar una tarea nueva se debe terminar una tarea previa.

En la Tabla 1.2 se muestra algunas de las ventajas al usar Kanban.

Tabla 1.2. Metodología Kanban ventajas

Ventajas al usar Kanban
Se optimizan los tiempos de entrega.
Los “cuellos de botella” son visibles en tiempo real, a través de esta característica se puede usar de forma efectiva los recursos para determinada tarea. En el tablero Kanban se puede observar el “cuello de botella” cuando las tareas se acumulan en la columna A mientras que la columna B está vacía.

Tabla 1.2. Metodología Kanban ventajas

Ventajas al usar Kanban
Es relativamente sencillo ver el avance del proyecto con solo observar el tablero.
Posibilita detectar si alguna tarea es defectuosa.

2. METODOLOGÍA

En este capítulo se expone el diseño e implementación del prototipo, para lo cual, como punto inicial se procede a realizar, elaborar el tablero Kanban con la lista de actividades a desarrollarse, posteriormente se realiza un estudio de las aplicaciones de entrega de comida a domicilio presentes en el mercado ecuatoriano más concretamente aplicaciones como Uber Eats [20], Glovo [21] y Rappi [22], este estudio se lo realiza con la finalidad de tener un punto de partida en cuanto a funcionalidad de la aplicación, enfocada en el cliente. Adicionalmente, se procederá a elaborar una entrevista para el Administrador del restaurante, esto con la finalidad de determinar los requerimientos necesarios desde el punto de vista del Administrador del restaurante el Asadero de Lali, también se hará una entrevista a diez clientes del establecimiento para establecer cuáles son los requerimientos del prototipo desde la perspectiva del cliente, estas actividades permiten obtener los requerimientos funcionales y no funcionales del prototipo.

Para obtener los requerimientos desde el punto de vista del usuario se hará uso de historias de usuario. Se procederá a establecer la arquitectura del prototipo y se definirán los módulos del mismo.

Para la implementación del prototipo se hará uso de la metodología Kanban, la cual es una metodología de desarrollo ágil. Se usará el tablero Kanban para crear y observar las tareas en cada sección de desarrollo. La metodología Kanban, no especifica el número de columnas o etapas con las que debe contar el tablero Kanban, por lo que se contará con las columnas: Tareas, Tareas en proceso y Tareas realizadas. Se desarrollarán diagramas entre los cuales estarán: diagrama de casos de uso, diagrama de actividades, y diagrama de estructura. Se realizarán los bosquejos para las interfaces de usuario.

2.1. TABLERO KANBAN

Para la realización del prototipo se usará el tablero Kanban, el cual muestra las distintas tareas a realizar, estará dividido en tres columnas las cuales son: Tareas, Tareas en proceso y Tareas realizadas. La Tabla 2.1 describe tres columnas, la primera columna, *Tareas*, describe todas las tareas que van a ser desarrolladas a lo largo del proyecto; la columna *Tareas en proceso*, que son aquellas tareas que se encuentran en proceso de desarrollo y, por último, la columna *Tareas realizadas*, que son aquellas tareas que se han completado.

Tabla 2.1. Lista de tareas Tablero Kanban

Tareas	Tareas en proceso	Tareas realizadas
Diseño de las vistas de la aplicación.	Toma de Requerimientos.	
Realización de diagramas de casos de uso.	Realización de entrevistas al administrador y a diez clientes del restaurante.	
Realización de diagrama de actividades.	Identificación de historias de usuario.	
Diagrama de estructura.	Identificación de requerimientos funcionales.	
Diseño de la estructura del proyecto.	Identificación de requerimientos no funcionales.	
Diseño base de datos.	Determinación de los módulos del prototipo.	
Instalación de Node.js		
Instalación de Yarn.		
Instalación Expo CLI.		
Instalación de Visual Studio Code.		
Instalación cliente Expo.		
Crear la base de datos en Firebase.		
Codificación pantallas del módulo administrador.		
Codificación pantallas del módulo Cliente.		
Codificación pantallas del módulo empleado.		
Codificación de navegación entre pantallas.		
Instalación de Firebase en la aplicación móvil.		

Tabla 2.1. Lista de tareas Tablero Kanban

Tareas	Tareas en proceso	Tareas realizadas
Codificación de funcionalidades del prototipo.		
Pruebas de funcionamiento módulo Administrador.		
Pruebas de funcionamiento módulo Cliente.		
Pruebas de funcionamiento módulo Empleado.		
Pruebas de funcionamiento con todos los módulos.		
Resultados.		

2.2. RECOLECCIÓN DE REQUERIMIENTOS

2.2.1. ESTUDIO DE APLICACIONES DE ENTREGA DE COMIDA A DOMICILIO

Se realizará un estudio del proceso que realizan las aplicaciones móviles de entrega de comida a domicilio Uber Eats, Rappi y Glovo. La primera vez que se usa estas aplicaciones solicitan que el usuario se registre, para lo cual, se ingresan ciertos datos entre los más generales están: el nombre, un correo electrónico, número celular y la dirección de entrega. Las Figuras 2.1 y 2.2 muestran las pantallas de ingreso de datos de estas aplicaciones.

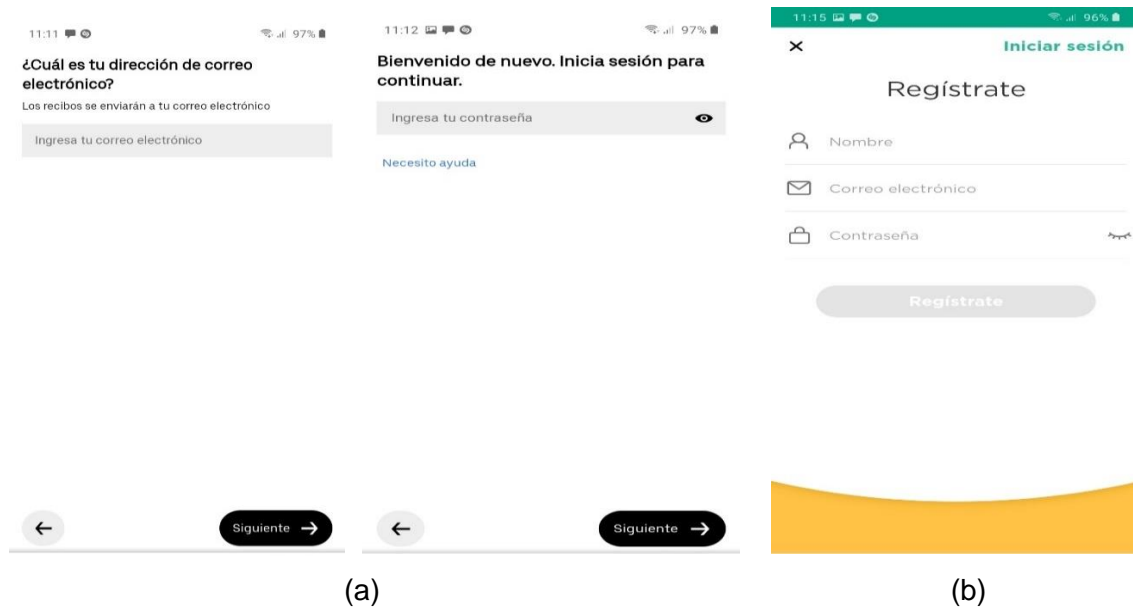


Figura 2.1. Ingreso de datos, (a) Uber Eats, (b) Glovo



Figura 2.2. Ingreso de datos, Rappi

De manera general, el proceso que el usuario sigue para realizar un pedido de comida a domicilio en estas aplicaciones es: una vez ingresados los datos se procede a iniciar sesión dentro de la aplicación, se visualiza las distintas opciones de comida con las que cuenta la aplicación. Se selecciona el producto, este producto se guarda en un carrito de compras hasta que el usuario termine de elegir otros productos en el caso de así necesitarlo, en el carrito estarán guardados los productos hasta que el cliente decida realizar el pedido, al realizar el pedido, se muestran en pantalla los datos personales del cliente, el valor total del pedido y la dirección a la cual deben ser enviados.

La Tabla 2.2 presenta un resumen general de las funcionalidades que presentan estas aplicaciones y que servirán como referencia para el prototipo.

Tabla 2.2. Resumen de funcionalidades

Funcionalidad
Permiten el registro de un nuevo usuario ingresando datos como nombre, dirección número de teléfono.
Permiten el inicio de sesión mediante un usuario y contraseña.
Permiten visualizar los menús de los que disponen.
Permiten guardar en un carrito de compras los productos.
Permiten editar el perfil de usuario entre los cuales esta poder actualizar la contraseña.
Permiten agregar más de una dirección de entrega.
Antes de realizar la compra se visualiza los productos y el valor a cancelar.

2.2.2. ENTREVISTAS

Las entrevistas se las realiza al administrador del restaurante, así como a los clientes del mismo, esto tiene como finalidad determinar historias de usuario y a partir de estas historias los requerimientos funcionales y no funcionales. Los requerimientos desde la perspectiva del administrador son expuestos en las entrevistas, en las mismas se manifiesta las necesidades que se pretende resolver con el desarrollo del prototipo, estos requerimientos se los representa en historias de usuario.

De la misma forma se entrevista a diez clientes del restaurante. El Anexo A presenta el modelo de entrevista y los resultados.

2.3. HISTORIAS DE USUARIO

Las historias de usuario presentan los requerimientos desde el punto de vista del usuario, tanto para el usuario administrador como para el usuario cliente.

Estas historias fueron definidas en base al estudio de las aplicaciones existentes en el mercado, así como también en base a las entrevistas realizadas al administrador del restaurante y a diez clientes del mismo. La Tabla 2.3 contiene estas historias de usuario.

Tabla 2.3. Historias de usuario

Id	Título	Descripción
HU01	Registrar usuario	El usuario cliente para utilizar la aplicación debe registrarse, ingresando datos como su nombre, número de celular, correo electrónico y contraseña, estos dos últimos ítems son los valores con los que el usuario podrá iniciar sesión en la aplicación.
HU02	Recuperar contraseña	Para recuperar la contraseña, el usuario necesita ingresar el correo electrónico que ingreso a la aplicación al momento de registrarse.
HU03	Iniciar sesión	Para iniciar sesión el usuario necesita ingresar el correo electrónico junto con la contraseña con las cuales realizó el registro.

Tabla 2.3. Historias de usuario

Id	Título	Descripción
HU04	Ingresar a la página principal	El usuario, después de iniciar sesión, se redirigirá a la página principal de la aplicación, esta página principal será diferente, para cada uno de los tres usuarios.
HU05	Visualizar los menús	El usuario cliente necesita ver los menús que oferta el restaurante.
HU06	Administrar el menú	El usuario administrador necesita crear, editar o eliminar los platos del menú.
HU07	Ingresar dirección de entrega	El usuario cliente necesita ingresar la dirección de entrega de su pedido en el mapa.
HU08	Visualizar dirección de entrega	El usuario empleado necesita visualizar en el mapa la dirección a la cual debe entregar el pedido.
HU09	Visualizar los pedidos	El usuario empleado, necesita ver los pedidos realizados para enviarlos al lugar correspondiente.
HU10	Enviar notificación	El usuario empleado necesita que se envíe una notificación cada que haya un pedido nuevo. El usuario cliente necesita que se envíe una notificación cuando su pedido este en camino.
HU11	Registrar nuevos empleados	El usuario administrador necesita ingresar los datos de nuevos empleados al sistema.

2.4. LISTA DE REQUERIMIENTOS

Tomando como base las historias de usuario y el análisis de las aplicaciones de envío de comida se determinan los requerimientos del prototipo.

Requerimientos funcionales

- Cada usuario que desee usar la aplicación móvil necesita tener una cuenta de usuario en el sistema.

- Las cuentas de usuario estarán asociadas a un rol de usuario.
- La aplicación cuenta con tres roles de usuario: administrador, cliente y empleado.
- El usuario administrador estará registrado con anterioridad en el sistema.
- El usuario administrador será el encargado de registrar nuevos empleados en el sistema.
- El sistema posibilita registrarse; con este fin se debe ingresar datos personales como: nombre, apellido, dirección, correo electrónico, número de teléfono, y contraseña.
- El sistema admite inicio de sesión, con este fin es necesario proporcionar la contraseña y el correo electrónico ingresados al momento del registro. En caso de que los datos ingresados sean válidos, se permite el ingreso, caso contrario, se comunicará que los datos son incorrectos.
- El sistema admite recuperación de contraseña, para lo cual, se debe ingresar el correo electrónico, al cual se envía los pasos para recuperar la contraseña.
- El sistema admite cerrar sesión.
- La página principal para cada uno de los tres roles será diferente.
- El sistema admite la actualización del perfil de usuario.
- El sistema admite al usuario administrador crear, editar y eliminar la información de los platos del menú, así como la información de las promociones del restaurante.
- El sistema admite que el usuario cliente visualice el menú y lo seleccione para agregarlo al carrito de compras para realizar el pedido.
- El sistema permite al usuario empleado ver los productos dentro del pedido realizado por el cliente para despachar la orden hacia la dirección respectiva.
- El sistema enviará una notificación a los empleados cuando se realice un nuevo pedido.
- El sistema enviará una notificación al cliente en el momento que su pedido este en camino.

Requerimientos no funcionales

- Desarrollar un prototipo de aplicación móvil con React Native para el sistema operativo Android.
- La base de datos se almacenará en Firebase.
- Para que los usuarios puedan acceder a la aplicación deben estar conectados a internet.

2.5. ARQUITECTURA DEL PROTOTIPO

La Figura 2.3 muestra la arquitectura del prototipo, y hace uso de Google Firebase para ser implementada.

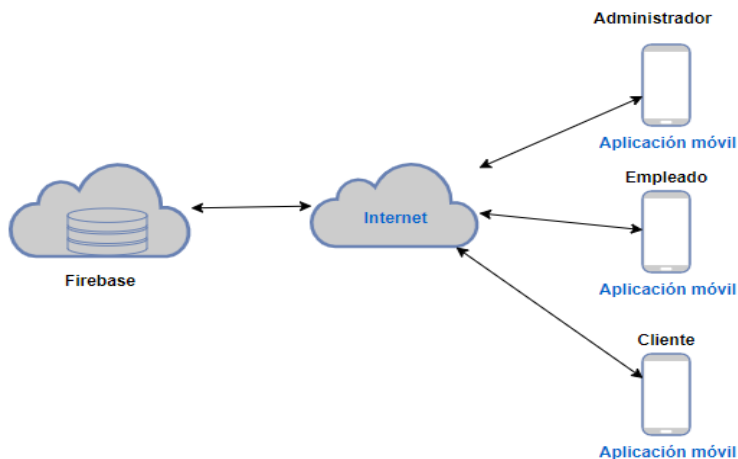


Figura 2.3. Arquitectura del prototipo

El prototipo está compuesto en el lado del cliente por una aplicación Android, la misma que es usada por clientes, empleados y el administrador. Por otra parte, el prototipo hace uso de los servicios de Firebase para almacenar la base de datos, así como también el servicio de autenticación. La aplicación se comunica con el BaaS (Backend as a Service) de Firebase para obtener datos tanto del usuario como del menú del restaurante, y también para realizar la autenticación, esto a través del SDK provisto por Firebase a través de internet.

De la misma forma, para el envío de las notificaciones se hará uso de los servicios provistos por la plataforma Expo, esto debido a que Expo provee una API gratuita que se enlaza con los servicios de Firebase Cloud Messaging (FCM) la cual es una solución multiplataforma de mensajería segura y gratuita para enviar mensajes o notificaciones [23].

En la Figura 2.4 se observa la arquitectura de las notificaciones usando el API de Expo.

El API de notificaciones de Expo es multiplataforma, al recibir el mensaje en su servidor este de forma automática lo redirección hacia Firebase Cloud Messaging en el caso de aplicaciones Android o hacia APNS en el caso de aplicaciones iOS. Esta notificación será enviada a los empleados en el momento que se realice un pedido, también se enviará una notificación al cliente cuando su pedido este en camino.

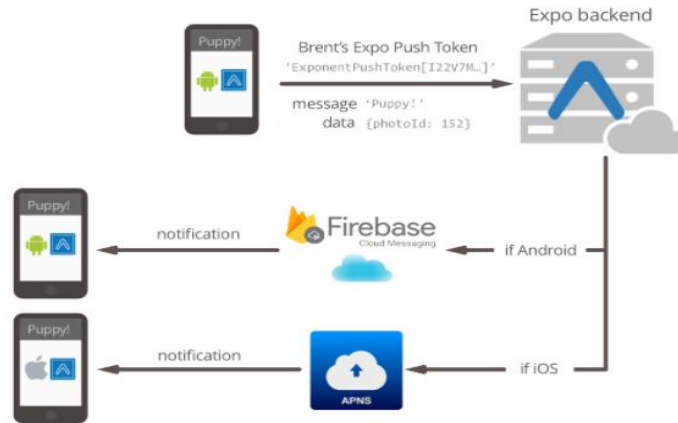


Figura 2.4. Arquitectura de notificaciones usando Expo [24]

En el Código 2.1 se indica un ejemplo de comunicación con el API de notificaciones de Expo.

```

async function sendPushNotification(expoPushToken) {
  const message = {
    to: expoPushToken,
    sound: 'default',
    title: 'Original Title',
    body: 'And here is the body!',
    data: { data: 'goes here' },
  };

  await fetch('https://exp.host/--/api/v2/push/send', {
    method: 'POST',
    headers: {
      Accept: 'application/json',
      'Accept-encoding': 'gzip, deflate',
      'Content-Type': 'application/json',
    },
    body: JSON.stringify(message),
  });
}

```

Código 2.1. Notificaciones con el API de Expo [25]

2.6. MÓDULOS

El prototipo de aplicación móvil contará con los siguientes módulos:

2.6.1. MÓDULO ADMINISTRADOR

En este módulo el administrador del restaurante, podrá leer, actualizar y eliminar los datos del menú y de las promociones. Además, será el encargado de registrar a los nuevos empleados en el sistema, para lo cual ingresará los datos de los nuevos empleados junto con una contraseña provisional la cual podrá ser cambiada por el empleado en el caso de así necesitarlo, podrá visualizar los pedidos realizados por los clientes, tanto los pedidos

que aún se encuentran en proceso de ser atendidos como aquellos que ya están en camino de entrega y los pedidos entregados.

Podrá ver un listado de los empleados con los que cuenta el restaurante y eliminarlos del sistema cuando sea necesario, así como también, ver un listado de los clientes que se encuentran registrados en el sistema, tendrá la posibilidad de actualizar sus datos y actualizar su contraseña, además en el caso de olvidar su contraseña de ingreso, podrá recuperarla mediante un enlace que será enviado al correo con el cual se registró.

2.6.2. MÓDULO EMPLEADO

El empleado después de ser registrado en el sistema por el administrador, ingresará al sistema con su correo y la contraseña provista por el administrador, podrá acceder a sus datos y editarlos, así también, podrá actualizar su contraseña en el caso de requerirlo, podrá visualizar los pedidos que han sido hechos por los clientes, así como también, el detalle de los productos presentes en el pedido para entregarlos a la persona que se encargará de enviarlos a la dirección de entrega, podrá ver en el mapa la ubicación en la cual debe ser entregado el pedido, así también, podrá observar los pedidos que ya han sido entregados a su destino.

2.6.3. MÓDULO CLIENTE

El cliente deberá crear una cuenta con los datos requeridos, así como también, una contraseña, en el proceso de registro se le pedirá que ingrese una dirección de entrega, para lo cual se hará uso del API de Google Maps, para tener la ubicación exacta del cliente.

El cliente podrá ver los diferentes platos con los que cuenta el restaurante, así como también las promociones y el menú del día, podrá realizar un pedido de los productos que desee, además podrá actualizar su información y su contraseña en el caso de necesitarlo. La Tabla 2.4 presenta los módulos con los identificadores de historias de usuario respectivas.

Tabla 2.4. Módulos del prototipo

Módulo	Id
Administrador	HU02, HU03, HU04, HU06, HU11
Cliente	HU01, HU03, HU04, HU05, HU07, HU10
Empleado	HU02, HU03, HU08, HU09, HU10

2.7. ACTUALIZACIÓN DEL TABLERO KANBAN

La Tabla 2.5 presenta la actualización del tablero Kanban, con las actividades que ya se han completado, así como también, presenta las actividades que están en proceso y aquellas que están en cola de iniciar el proceso de desarrollo.

Tabla 2.5. Actualización del Tablero Kanban

Tareas	Tareas en proceso	Tareas realizadas
Instalación de Node.js	Diseño de las vistas de la aplicación.	Toma de Requerimientos.
Instalación de Yarn.	Realización de diagramas de casos de uso.	Realización de entrevistas al administrador y a diez clientes del restaurante.
Instalación Expo CLI.	Realización de diagrama de actividades.	Identificación de historias de usuario.
Instalación de Visual Studio Code.	Diagrama de estructura.	Identificación de requerimientos funcionales.
Instalación cliente Expo.	Diseño de la estructura del proyecto.	Identificación de requerimientos no funcionales.
Crear la base de datos en Firebase.	Diseño base de datos.	Determinación de los módulos del prototipo.
Codificación pantallas del módulo administrador.		
Codificación pantallas del módulo Cliente.		
Codificación pantallas del módulo empleado.		
Codificación de navegación entre pantallas.		
Instalación de Firebase en la aplicación móvil.		
Codificación de funcionalidades del prototipo.		

Tabla 2.5. Actualización del Tablero Kanban

Tareas	Tareas en proceso	Tareas realizadas
Pruebas de funcionamiento módulo Administrador.		
Pruebas de funcionamiento módulo Cliente.		
Pruebas de funcionamiento módulo Empleado.		
Pruebas de funcionamiento con todos los módulos.		
Resultados.		

2.8. DISEÑO

En esta sección, se desarrollará el diseño de las pantallas de la aplicación móvil, así como también, los diagramas de casos de uso de cada uno de los módulos, diagrama de actividades y diagrama de estructura. Se diseñará la estructura del proyecto, así como también se hará el diseño de la base de datos.

2.8.1. DISEÑO DE LA INTERFAZ DE LA APLICACIÓN

Se diseñará las UI de la aplicación, no contienen colores, así como tampoco gráficos ni mucho menos estilos, esto se lo hace con el fin de dar prioridad al contenido, así como también para distribuir el espacio y obtener un primer plano de la aplicación que se desea desarrollar.

En la Figura 2.5 se muestra la pantalla de inicio de sesión, así como la de recuperación de contraseña las cuales serán comunes para los tres roles de usuario.

En la Figura 2.6 se muestra las pantallas principales de cada uno de los tres roles de usuario. El diseño completo se encuentra en el ANEXO B: Diseño de la interfaz El Asadero de Lali.

2.8.2. DIAGRAMA DE CASOS DE USO

Los diagramas de casos de uso (CU) se utilizan para explicar la funcionalidad que tendrá el sistema. En estos diagramas se emplean actores, que son las entidades que utilizan el sistema, es decir son aquellos que van a interactuar con el sistema.

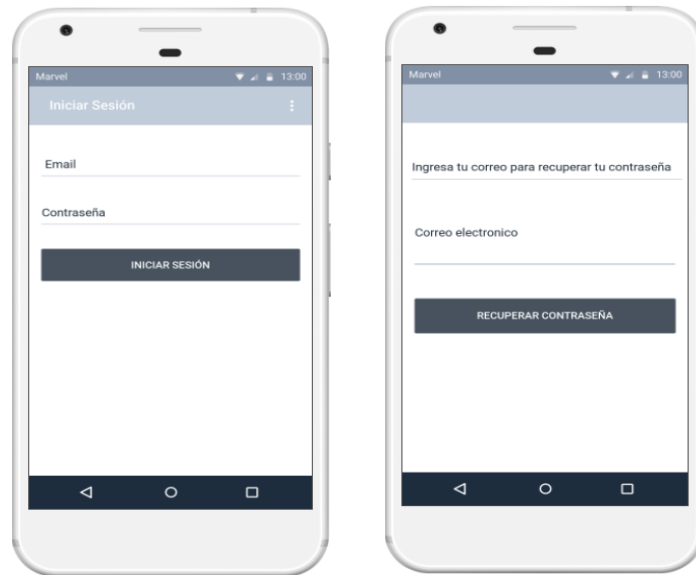


Figura 2.5. Diseño de inicio de sesión y recuperación de contraseña



(a)

(b)

(c)

Figura 2.6. Diseño de página principal de (a) Administrador, (b) Cliente, (c) Empleado

Los actores con los que cuenta el prototipo son:

- Usuario Administrador
- Usuario Cliente
- Usuario Empleado

Cada uno de estos actores cumple una función el cual se describe a continuación:

- *Administrador.* El usuario con el rol administrador, a través del prototipo administrará los datos de los diferentes platos con los que cuenta el restaurante,

así como también, las promociones que él crea necesarias, esta administración implica la actualización, lectura, creación y eliminación de los datos de los platos presentes en el menú, también la administración de las promociones, así como del menú del día.

Por otra parte, también podrá administrar los datos de su perfil de usuario, entre estos datos están su nombre, su número de teléfono, así como también, la posibilidad de subir una foto en su perfil, podrá ingresar al sistema a nuevos empleados, podrá visualizar los datos de los empleados presentes en el sistema y eliminarlos de ser necesario, así como a los clientes registrados en el sistema, podrá ver los pedidos realizados por los clientes y de igual forma los pedidos que estén en camino de entrega y los pedidos entregados, también podrá recuperar su contraseña en el caso de ser necesario. El diagrama de casos de uso se muestra en la Figura 2.7.

- *Cliente*: El usuario con el rol cliente, antes de poder hacer uso del prototipo debe registrarse en el sistema, para lo cual ingresará los datos que la aplicación le indica, de igual forma también ingresará una dirección de entrega, una vez registrado en el sistema, podrá visualizar los distintos platos del menú del restaurante, podrá elegir los platos y la cantidad que desee para que sean enviados a su domicilio. Podrá administrar los datos de su perfil de usuario, antes de confirmar el pedido, podrá observar el valor total de su compra, y de igual forma sus datos, podrá realizar una compra, podrá ingresar una o más direcciones de entrega en el caso de que lo necesite. Recibirá una notificación cuando su pedido este en camino, adicionalmente podrá recuperar su contraseña en el caso de que no la recuerde. El diagrama de casos de uso se muestra en la Figura 2.8.
- *Empleado*: El usuario con el rol empleado, una vez registrado en el sistema por el administrador, podrá observar los nuevos pedidos hechos por los clientes, con la finalidad de armar el pedido con los productos que el cliente haya solicitado, podrá observar los pedidos que están en camino de entrega, también podrá ver en el mapa la dirección de entrega del pedido y confirmar la entrega del pedido, de la misma forma podrá recuperar su contraseña en el caso de ser necesario. El diagrama de casos de uso se lo muestra en la Figura 2.9.

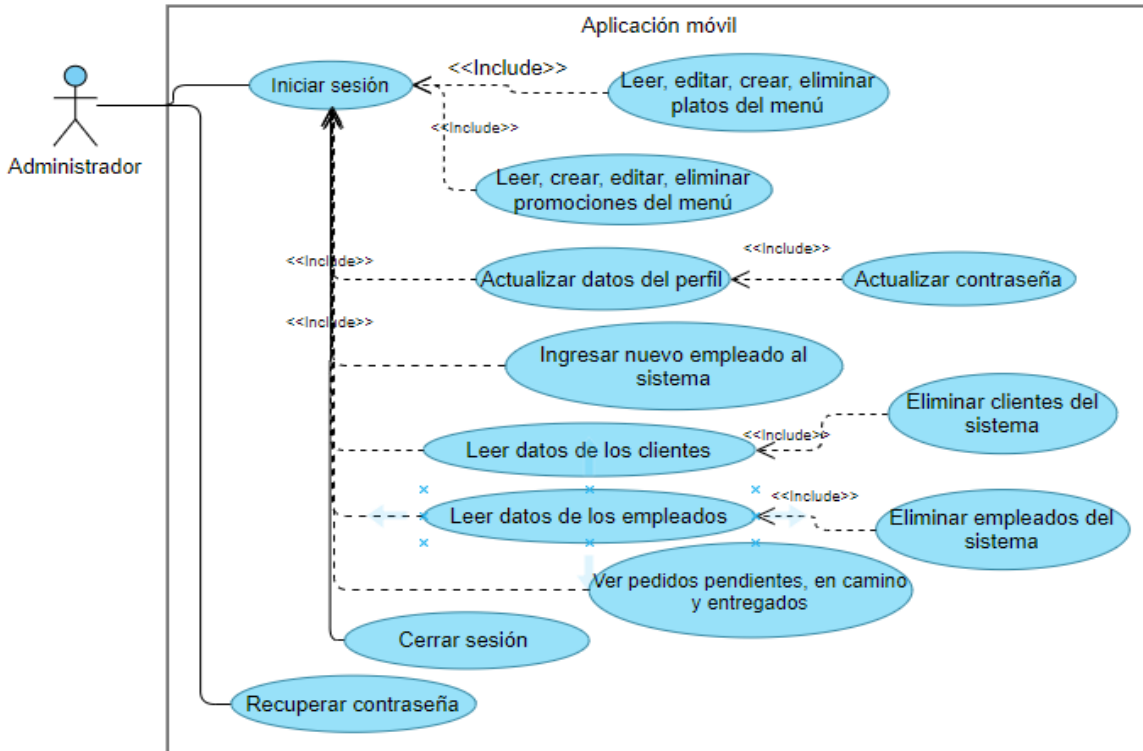


Figura 2.7. Caso de uso: Módulo administrador

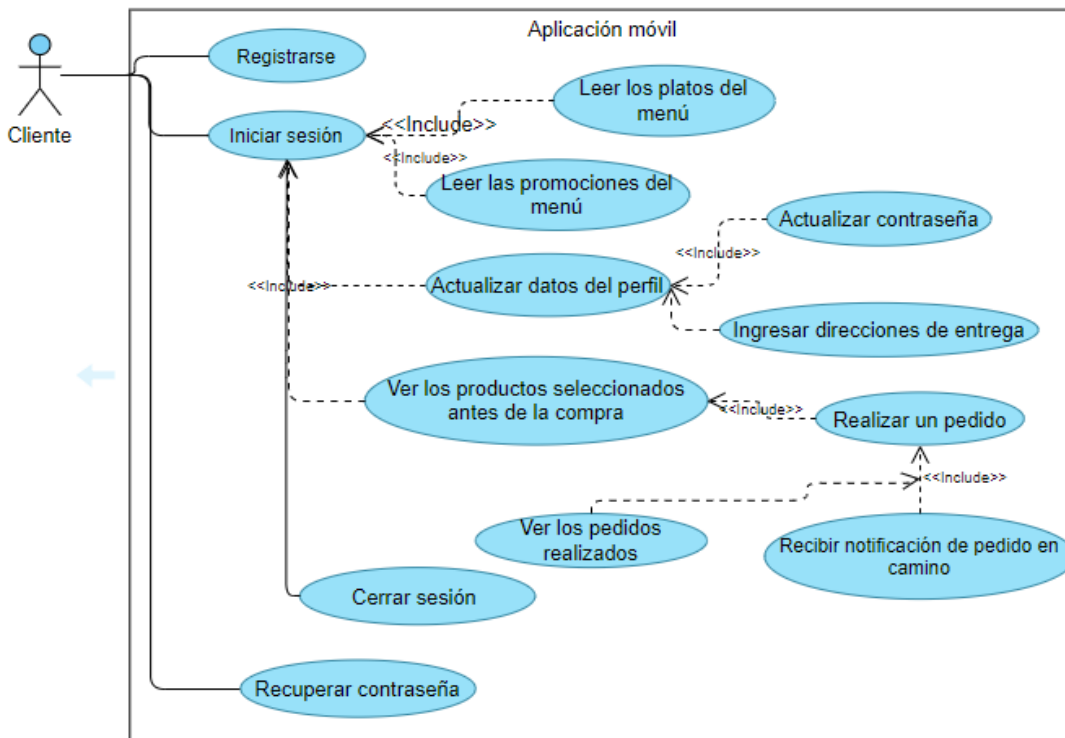


Figura 2.8. Caso de uso: Módulo cliente

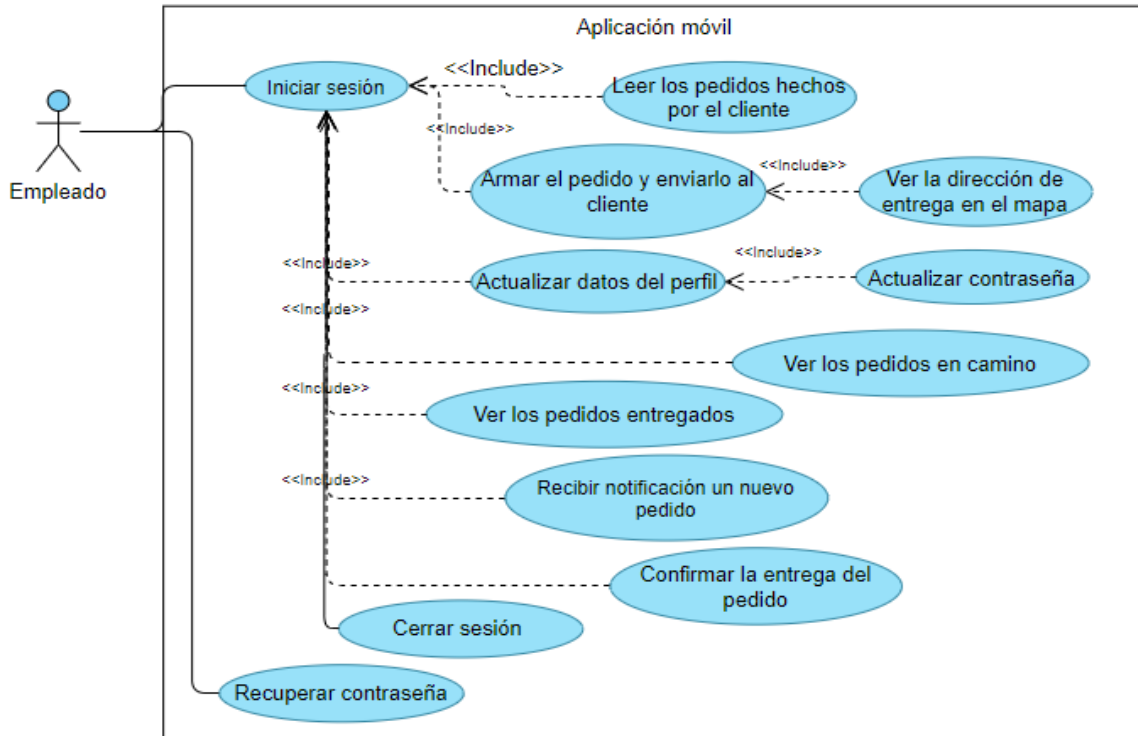


Figura 2.9. Caso de uso: Módulo empleado

2.8.3. DIAGRAMA DE ACTIVIDADES

En las figuras siguientes se expone los diagramas de actividades de los casos de uso más importantes. La Figura 2.10 expone el diagrama de actividades del caso de uso del módulo administrador para el ingreso de un nuevo plato al menú. El ingreso de un nuevo plato al menú del restaurante se da una vez el administrador ha ingresado a la aplicación, ingresando su correo y contraseña, en el caso de que aún no haya ingresado a la aplicación.

Al ingresar al sistema para agregar un nuevo plato al menú, el administrador debe ingresar los datos del plato tales como: nombre del plato, una descripción de mismo, el precio y una imagen del plato, estos datos no pueden quedar vacíos por lo que el sistema se encargará de validarlos, una vez se haya ingresado toda la información requerida el sistema registrará el nuevo plato en la base de datos y el administrador podrá observar un mensaje en el cual se le informa la creación exitosa del nuevo plato y se dará por finalizado el proceso.

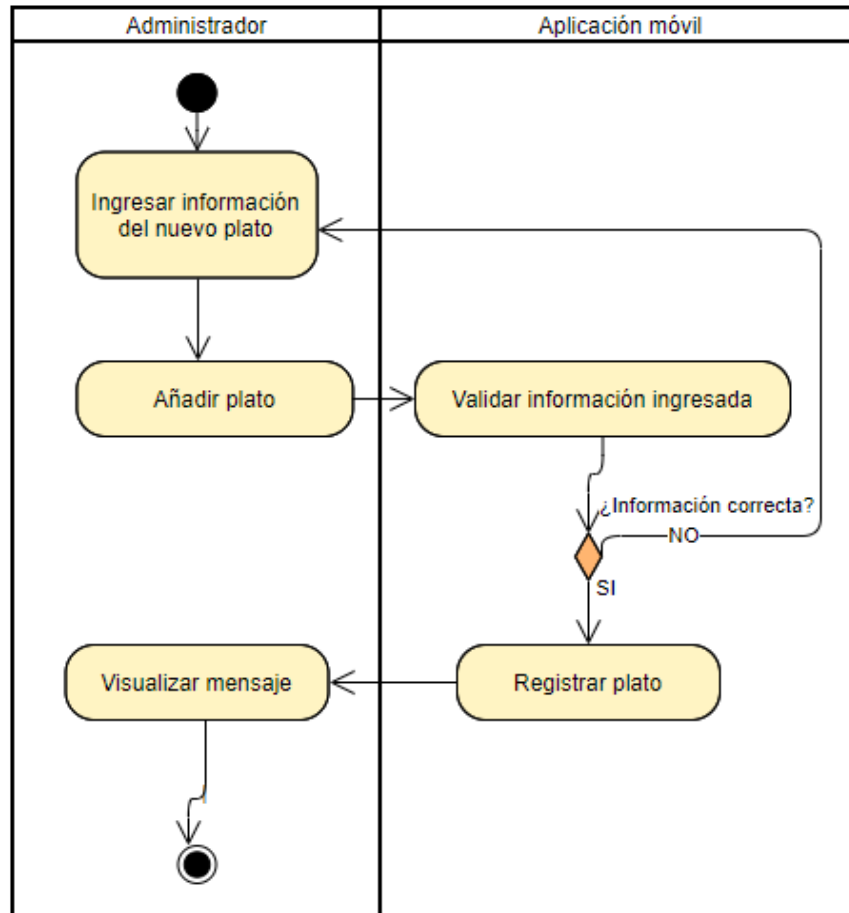


Figura 2.10. Diagrama de actividades crear plato

La Figura 2.11 muestra el diagrama de actividades correspondientes al caso de uso ingresar nuevos empleados al sistema. El proceso da inicio cuando el administrador dentro de la aplicación se dirige a la opción de ingresar nuevo empleado, en esta nueva pantalla el administrador debe ingresar los datos del nuevo empleado, así como una contraseña la cual será provisional y podrá ser actualizada por el empleado, el sistema validará que los datos que se solicitan no queden en blanco, así como que el correo ingresado tenga una sintaxis válida y que los caracteres ingresados en la contraseña tengan un tamaño mayor a seis, una vez validados estos datos el sistema procederá a ingresar al nuevo empleado en la base de datos, el administrador observará un mensaje en el cual se le indica la creación exitosa del nuevo empleado en el sistema, con lo cual se da por finalizado el proceso.

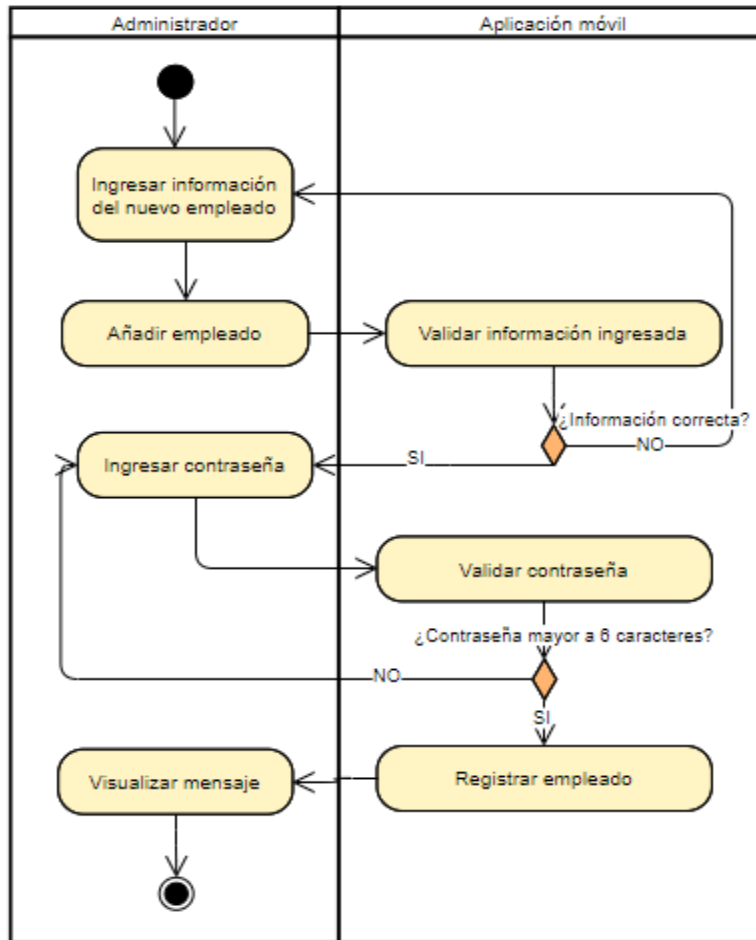


Figura 2.11. Diagrama de actividades ingresar nuevo empleado

La Figura 2.12 muestra el diagrama de actividades del caso de uso realizar una compra del módulo cliente. El proceso da inicio cuando el usuario cliente una vez dentro de la aplicación desea realizar un pedido de comida a domicilio. Al ingresar a la aplicación para realizar una compra, el cliente podrá visualizar tanto las promociones como los platos del menú, elegirá los que más le gusten y se almacenarán en el carrito de compras, una vez haya elegido los productos se dirigirá al carrito de compras en el cual estarán los productos que seleccionó junto el valor total del pedido, con la posibilidad de quitar de la lista alguno de los productos enlistados, una vez confirmados los productos se dirige a la pantalla de pedido en la cual estarán sus datos así como también deberá elegir la dirección hacia la cual debe ser enviada el pedido, una vez confirmada la orden el cliente observará un mensaje en el cual se le indica que su pedido fue exitoso, con lo cual se da por finalizado el proceso.

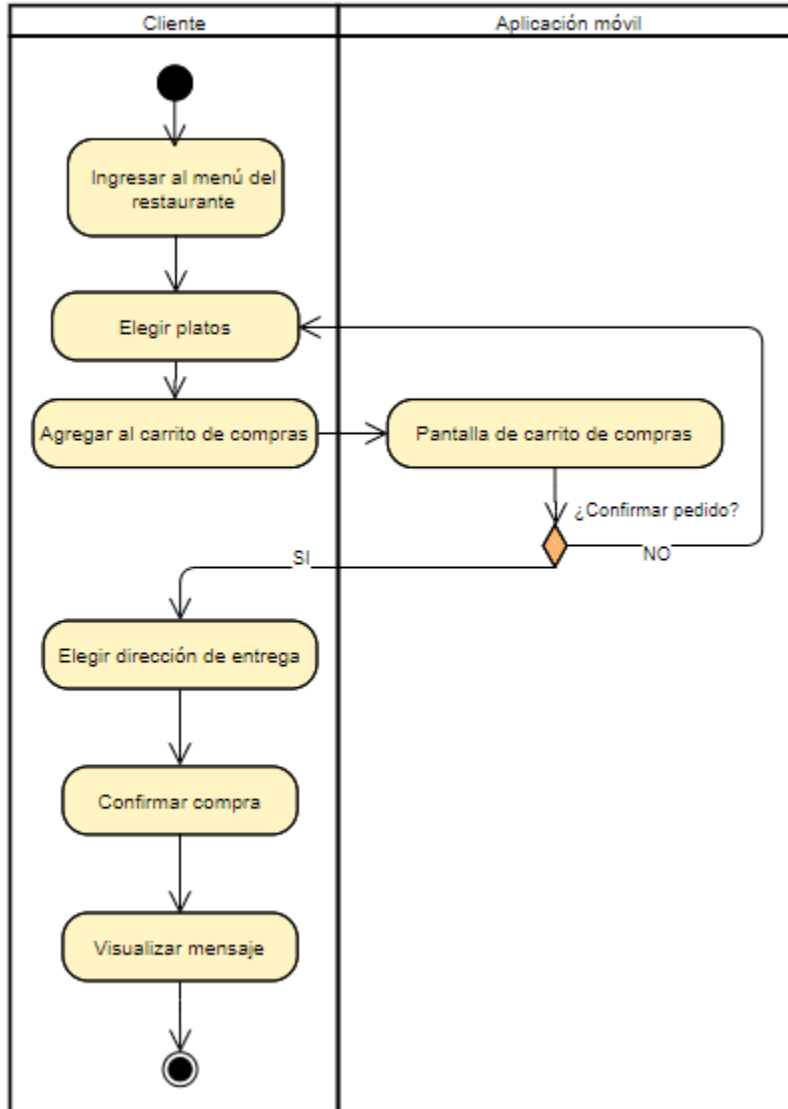


Figura 2.12. Diagrama de actividades realizar un pedido

2.8.4. DIAGRAMA DE ESTRUCTURA

En la Figura 2.13 se presenta el diagrama de estructura compuesta del módulo administrador, el proceso da inicio cuando el usuario administrador inicia sesión en la aplicación, el administrador puede elegir entre varios de los elementos, entre ellos: editar perfil, agregar nuevo plato, ingresar nuevo empleado entre otros, una vez elegida la instancia se ingresan los datos y se validan, y posterior a su validación son almacenados en la base de datos.

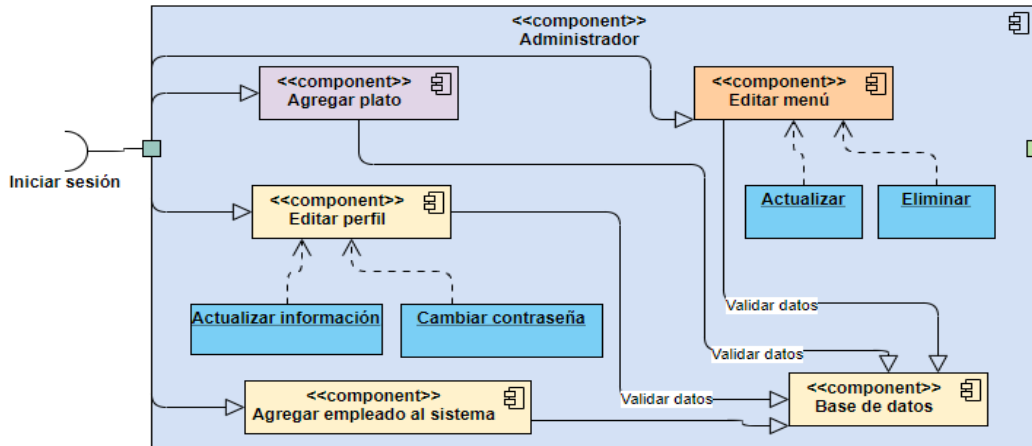


Figura 2.13. Diagrama de estructura

2.8.5. ESTRUCTURA DEL PROYECTO

Al momento de crear el proyecto con React Native, el cual llevará el nombre “El-Asadero-de-Lali”, se crea una carpeta dentro de la cual se estructuran tanto los archivos como los directorios que usará el proyecto. El directorio screen será aquel que contenga los directorios de los tres módulos definidos en la aplicación: Administrador, Cliente y Empleado, a su vez estos directorios contienen los archivos que generan el contenido de la aplicación, además se crea directorios para contener archivos que permiten la conexión con la base de datos almacenada en Firebase, más concretamente el directorio conexión, y otros directorios que serán necesarios a lo largo del desarrollo del proyecto.

En la parte izquierda de la Figura 2.14 se presenta la estructura del proyecto en su totalidad, tal como se muestra en VS Code, en tanto que en la parte derecha de la figura se muestra las carpetas, así como algunos archivos creados para la aplicación.

2.8.6. DISEÑO DE LA BASE DE DATOS

Los datos que utiliza la aplicación “El asadero de Lali” son almacenados y recuperados desde Firebase Realtime Database, que guarda y sincroniza los datos con la base de datos NO-SQL de Google la cual está almacenada en la nube. Esta información se sincroniza con los usuarios de la aplicación en tiempo real.

Una característica de las bases de datos NO-SQL es que los diagramas entidad-relación no suelen ser comunes, esto en vista de que en vez de usar tablas y propiedades utilizan colecciones y documentos respectivamente.

La Figura 2.15 muestra una captura de pantalla de las colecciones usadas para la aplicación capturada desde Firebase.

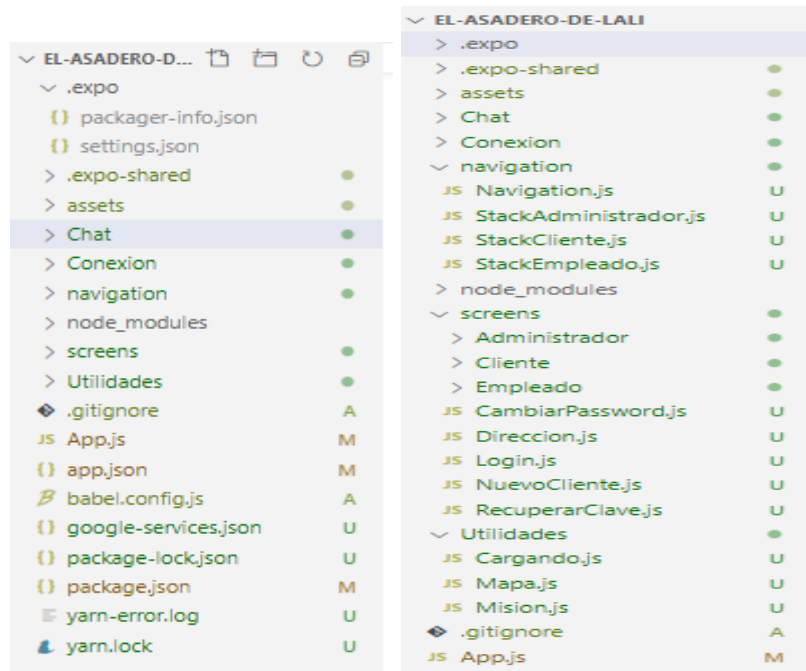


Figura 2.14. Estructura del proyecto Asadero de Lali en React Native

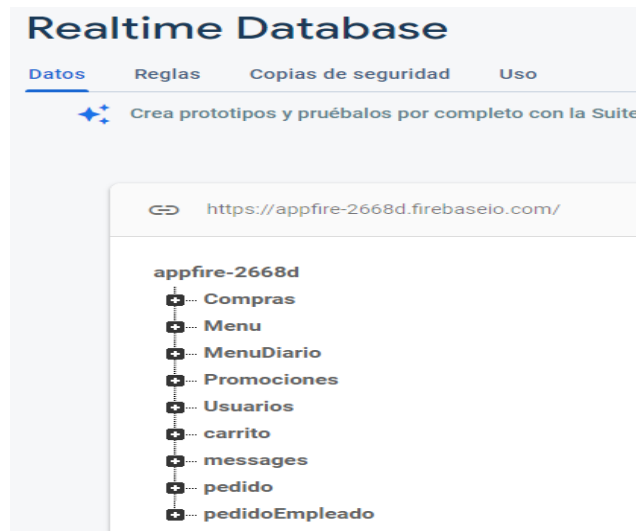


Figura 2.15. Base de datos de Firebase

Este proyecto elabora las colecciones siguientes para guardar los datos que serán usados y generados por la aplicación.

- **Compras**, es una colección que guarda las compras que realiza el usuario cliente, con las cantidades, así como el nombre del producto que adquirió.

- **Promociones**, es una colección que guarda las promociones que genera el administrador del restaurante.
- **Usuarios**, esta colección contiene a las colecciones Administrador, Clientes y Empleados, que a su vez contienen los identificadores de cada uno de los usuarios, así como la información de los mismos entre los que constan nombre, dirección, teléfono entre otros.
- **Carrito**, esta colección contiene los productos que el usuario cliente selecciona antes de realizar la compra definitiva.
- **Menu**, esta colección contiene la información del menú del restaurante, contiene los datos como nombre, precio y descripción de cada uno de los platos del menú, clasificados en sopas, plato fuerte, postres y bebidas.
- **Pedido**, esta colección contiene los datos de los pedidos de los clientes una vez han sido enviados a su dirección de destino, así como también contiene la información de los productos que han sido entregados al cliente.
- **pedidoEmpleado**, esta colección contiene la información de los platos que han sido solicitados por los clientes, para que los empleados visualicen que productos son los que deben ser preparados para su envío.

Para poder implementar la base de datos con Firebase se debe tener una cuenta de Gmail. La Figura 2.16 indica parte de la estructura de la base de datos.



Figura 2.16. Estructura base de datos en Firebase

2.9. IMPLEMENTACIÓN

En esta parte se presenta la implementación de los componentes de la aplicación móvil, así como también, la forma como estos interactúan entre sí. Como primer paso se presenta la instalación del software que será necesario para el desarrollo del prototipo, se presenta la actualización del tablero Kanban mostrado en la Tabla 2.6, se presenta la codificación del prototipo y de sus diferentes módulos, de forma simultánea, se presenta la implementación de la base de datos en Firebase para el almacenamiento de los datos y por último la implementación de las notificaciones push.

Tabla 2.6. Tablero Kanban. Implementación

Tareas	Tareas en proceso	Tareas realizadas
Pruebas de funcionamiento módulo Administrador.	Instalación de Node.js.	Toma de Requerimientos.
Pruebas de funcionamiento módulo Cliente.	Instalación de Yarn.	Realización de entrevistas al administrador y a diez clientes del restaurante.
Pruebas de funcionamiento módulo Empleado.	Instalación Expo CLI.	Identificación de historias de usuario.
Pruebas de funcionamiento con todos los módulos.	Instalación de Visual Studio Code.	Identificación de requerimientos funcionales.
Corrección de errores.	Instalación cliente Expo.	Identificación de requerimientos no funcionales.
	Crear la base de datos en Firebase.	Determinación de los módulos del prototipo.
	Codificación pantallas del módulo administrador.	Diseño de las vistas de la aplicación.
	Codificación pantallas del módulo Cliente.	Realización de diagramas de casos de uso.

Tabla 2.6. Tablero Kanban. Implementación

Tareas	Tareas en proceso	Tareas realizadas
	Codificación pantallas del módulo empleado.	Realización de diagrama de actividades.
	Codificación de navegación entre pantallas.	Diagrama de estructura.
	Instalación de Firebase en la aplicación móvil.	Diseño de la estructura del proyecto.
	Codificación de funcionalidades del prototipo.	Diseño base de datos.

2.9.1. INSTALACIÓN DE HERRAMIENTAS

Node.js

Como primer paso se debe instalar el entorno de ejecución Node.js el cual se lo descarga desde su página oficial, <https://nodejs.org/es/>, una vez descargado el archivo lo ejecutamos y seguimos las instrucciones, para el desarrollo de este proyecto se utiliza la versión 14 de Node como muestra en la Figura 2.17. Al instalar Node.js este vendrá por defecto con un gestor de paquetes JavaScript denominado npm.



Figura 2.17. Descarga de Node.js

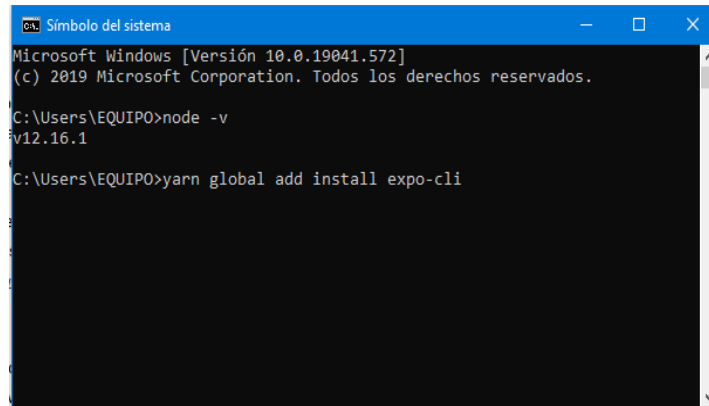
Yarn

Yarn al igual que npm es un gestor que sirve para gestionar paquetes JavaScript, es necesario instalarlo para instalar la herramienta Expo.cli, se lo descarga desde su página

oficial, <https://classic.yarnpkg.com/es-ES/docs/install#windows-stable> y al igual que con Node.js se ejecuta el archivo descargado y se sigue las instrucciones de instalación.

Expo CLI

Para instalar expo CLI introducimos el comando `yarn global add install expo-cli` en la terminal de Windows (CMD), este comando instala de forma global Expo.cli, la Figura 2.18 presenta la instalacion de Expo.



```
Símbolo del sistema
Microsoft Windows [Versión 10.0.19041.572]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\EQUIPO>node -v
v12.16.1

C:\Users\EQUIPO>yarn global add install expo-cli
```

Figura 2.18. Instalación Expo.CLI

Instalación Visual Studio Code

Visual Studio Code (VSCode) es un editor gratuito y de código abierto, que permite compilar varios lenguajes de programación entre ellos JavaScript y Node.js, por lo tanto, permite la ejecución del React Native; se lo descarga de su página oficial, <https://code.visualstudio.com/download>, la Figura 2.19 Muestra parte del proceso de instalación de VSCode.

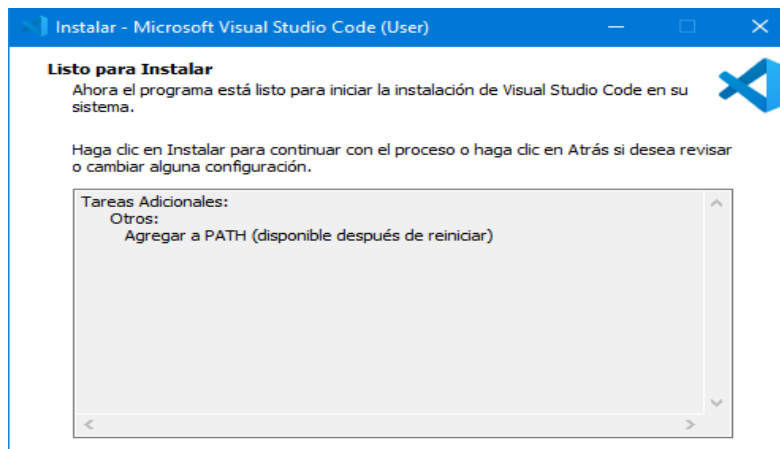


Figura 2.19. Instalación Visual Studio Code

Una vez se haya ejecutado el instalador, se muestra una interfaz tal y como lo muestra la Figura 2.20, para facilitar el trabajo instalamos extensiones las cuales ayudan a tener un código más ordenado, entre las extensiones a instalar están *Prettier*, que nos permite anidar el código para obtener una mejor visualización del mismo, también se instala *Bracket Pair Colorized*, el cual nos permite dar colores a las diferentes secciones del código, todo esto para un mejor entendimiento de donde inicia y termina determinado código, tal como la muestra la Figura 2.21.

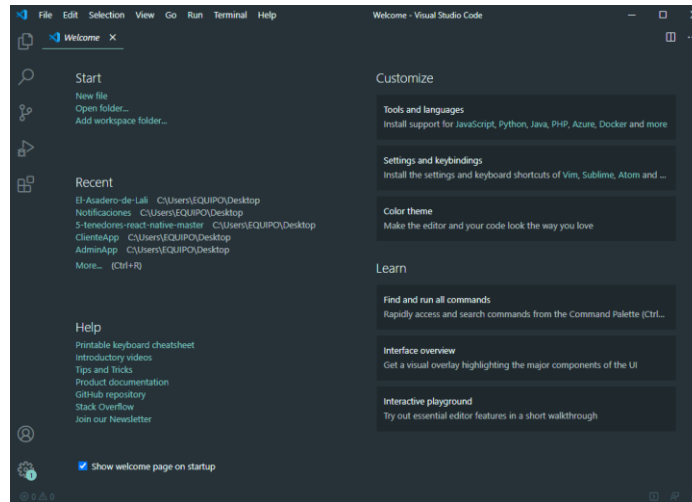


Figura 2.20. Visual Studio Code

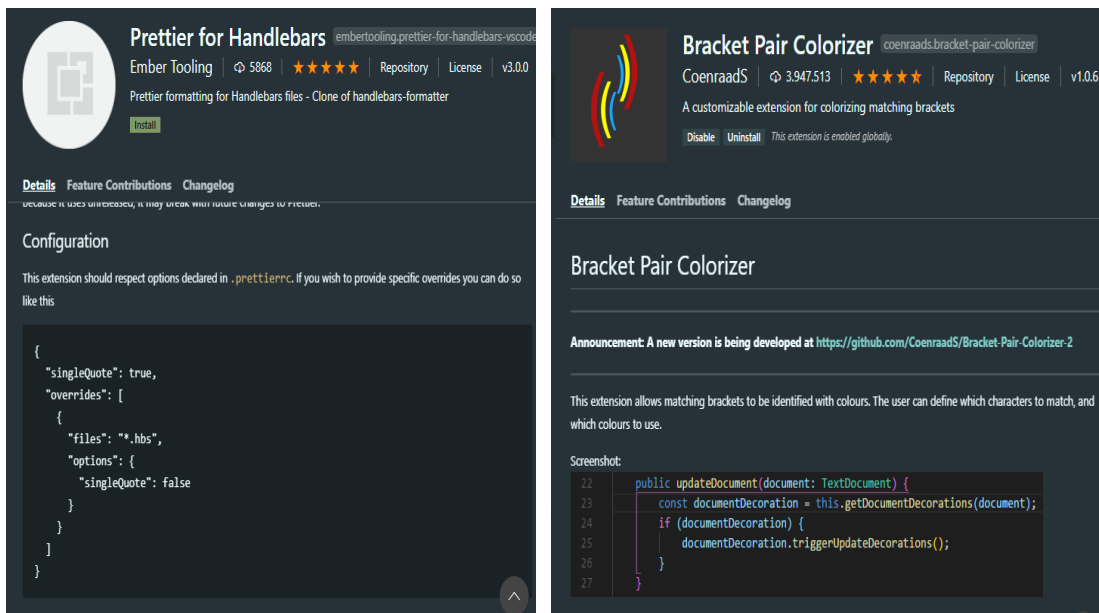


Figura 2.21. Extensiones VS Code

Cliente Expo

Para ejecutar la aplicación desarrollada con la herramienta Expo, se descarga el cliente expo el cual está disponible en la Google Play de forma gratuita, Expo client permite ejecutar la aplicación en un teléfono Android al leer el código QR que genera al momento de compilar la aplicación con Expo.CLI. La Figura 2.22 muestra la descarga del cliente Expo de la tienda de Google Play.

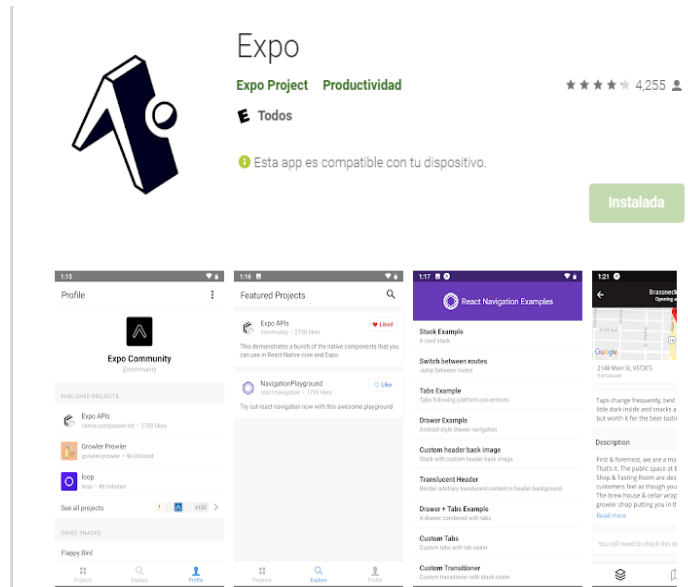


Figura 2.22. Descarga de Expo client

2.9.2. BASE DE DATOS

Para la implementación de la base de datos se utilizó la consola de Firebase, para el ingreso de los datos de inicio, tales como los datos del administrador, así como los datos preliminares del menú tal como se describe en el apartado de diseño de la base de datos. La Figura 2.23 muestra un fragmento de la implementación de la base de datos correspondiente a los usuarios y una parte de la información del administrador, así como la información de un cliente, los datos de los empleados son similares a los datos mostrados para el administrador.

En los datos presentados, el ID provisto por Firebase es el nodo principal y contiene los datos como nombre, apellido, teléfono, etc.



Figura 2.23. Base de datos: Usuarios

La Figura 2.24 muestra los nodos correspondientes a los platos del menú, así como un fragmento del tipo de plato Bebidas, el menú contará con cuatro tipos de platos entre los que están: Sopas, Fuerte, Postre y Bebidas. Los datos para Fuerte, Postre y Sopas son similares a los mostrados en la Figura 2.24, cambiando en el ítem Tipo.



Figura 2.24. Fragmento de la base de datos de platos del menú

Además, se implementó nodos adicionales que ayudarán en el desarrollo de la aplicación móvil, tales como un nodo denominado carrito de compras, este nodo contiene los productos que el usuario haya elegido pero que aún no se confirme su compra; un nodo denominado compras, en el cual contará con los productos que hayan sido adquiridos por el usuario cliente; un nodo denominado pedido el cual contiene la información del cliente, así como el valor total de su compra y la fecha de la misma.

2.9.3. APLICACIÓN ANDROID

La aplicación móvil fue desarrollada usando VS Code y Expo client mediante el Framework React Native.

El desarrollo de la aplicación es importante, pero no lo es menos la parte de la interfaz de usuario (UI) de la misma ya que, de la experiencia que el usuario tenga al momento de usar

la aplicación dependerá hasta cierto punto el éxito o fracaso de la misma, si la aplicación no presenta una UI agradable o su uso no es sencillo e intuitivo, los usuarios no se sentirán atraídos hacia la misma.

La UI del prototipo de aplicación móvil El Asadero de Lali, se lo hizo en base a los requerimientos del Administrador del restaurante, a los diseños presentados anteriormente, así como a los requerimientos de los clientes obtenidos en las entrevistas, la elección de los colores, así como el logo del proyecto fueron hechos en colaboración con el Administrador del restaurante.

El desarrollo de las diferentes pantallas de la aplicación se la realizó con los componentes que provee React Native, así como con React Native Elements. El Código 2.2 muestra la forma de importar algunos módulos para desarrollar la UI.

```
1 import React, { useState } from "react";
2 import {
3   StyleSheet,
4   Text,
5   View,
6   TextInput,
7   TouchableOpacity,
8   Dimensions,
9 } from "react-native";
```

Código 2.2. Importación de librerías

En la primera línea se importa React desde la librería react, el cual es el componente principal en todos los archivos con React Native, así como también otras funcionalidades como useState y useEffect que ayudan a manejar los estados en el código. Desde la línea número dos hasta la línea 9, se importan componentes que ayudan a construir la interfaz gráfica del proyecto tales como botones (TouchableOpacity) entradas de texto (TextInput) entre otros.

Para crear las vistas en Visual Studio Code, se lo hace a través de código, las utilidades como botones y cuadros de texto también se los agrega mediante código. La Figura 2.25 presenta la interfaz gráfica de Visual Studio Code para la implementación de la interfaz gráfica del prototipo. En parte derecha se muestra la sección donde se edita el código, en la parte izquierda se muestran los archivos que forman parte del prototipo, y en la parte inferior se encuentra la sección que funciona como consola de VS Code.

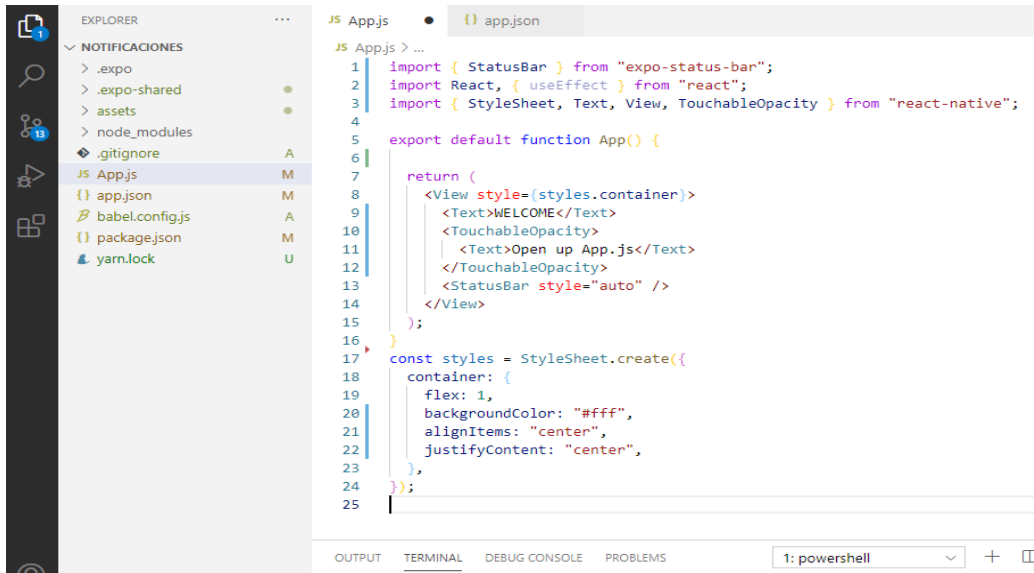


Figura 2.25. Interfaz de Visual Studio Code

A continuación, se presentan las UI del inicio de sesión, así como el de recuperación de contraseña ya que estas UI son comunes a todos los roles de usuario. Las interfaces fueron desarrolladas con el IDE Visual Studio Code con el framework React Native. La Figura 2.26 muestra el inicio de sesión, en la Figura 2.27 se muestra la UI para recuperar la contraseña.

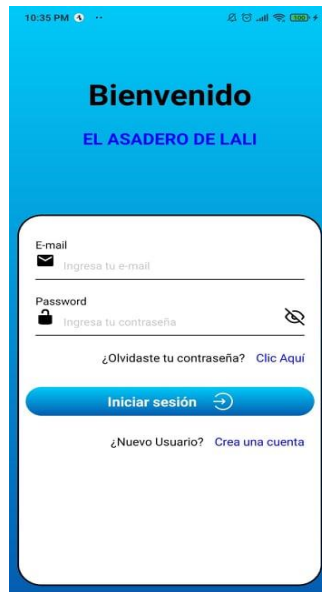


Figura 2.26. Inicio de sesión

El Código 2.3 presenta un fragmento del código usado para la implementación de la pantalla mostrada en la Figura 2.26.

```

144     return (
145       <KeyboardAwareScrollView keyboardShouldPersistTaps="always">
146         <View style={styles.container}>
147           <LinearGradient
148             colors={[ "#01caf9", "#055db1" ]}
149             style={{ padding: 10, flex: 1 }}
150           >
151             <StatusBar style="auto" barStyle="light-content" />
152             <View style={styles.header}>
153               <Text style={{ fontSize: 40, fontWeight: "bold" }}>Bienvenido</Text>
154               <View style={{ height: 20 }} />
155               <Text style={{ fontSize: 22, color: "blue", fontWeight: "bold" }}>
156                 EL ASADERO DE LALI
157               </Text>
158             </View>
159             <Animatable.View animation="fadeInUpBig" style={styles.styleText}>
160               <Text style={{ styles.title, { marginTop: 20 } }}>E-mail</Text>
161               <View style={styles.action}>
162                 <MaterialIcons
163                 >
164                   styles={{ ...
165                 />
166               </View>
167               <View style={{ width: 10 }} />
168               <TextInput
169               >
170                 placeholder="Ingresa tu e-mail" ...
171                 value={email}
172               />
173             </Animatable.View>
174           </View>
175         </KeyboardAwareScrollView>
176       )
177     )

```

Código 2.3. Fragmento de código para pantalla de Inicio de sesión



Figura 2.27. Recuperación de contraseña

A continuación, se presentan algunas UI para cada módulo de usuario.

2.9.3.1. Módulo Administrador

En este módulo, el usuario con rol administrador, realiza varias acciones, entre ellas están: el ingreso de nuevos empleados al sistema, la edición o eliminación de los menús del restaurante, crear un nuevo plato para el menú, podrá visualizar una lista de los empleados, así como también, una lista de los clientes registrados en la aplicación, podrá actualizar su contraseña, así como los datos de su perfil entre ellos, sus datos personales, actualizar o agregar una foto de perfil. El detalle se presenta a continuación.

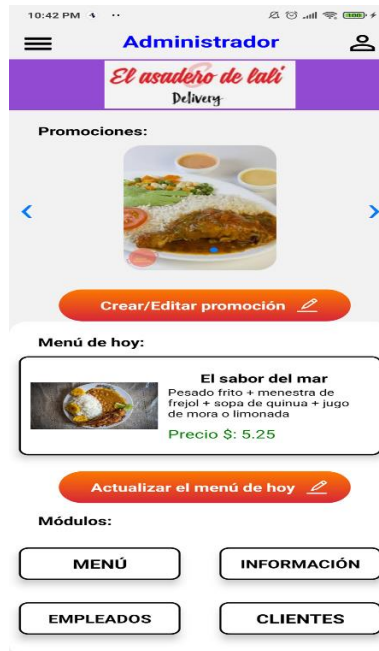


Figura 2.28. Página principal rol administrador

Una vez que el usuario haya ingresado sus credenciales, correo electrónico y contraseña de forma correcta, se lo redirige a la página principal del administrador, la Figura 2.28 presenta la implementación de la página principal del Administrador. El Código 2.4 muestra un fragmento de código para implementar la pantalla de la Figura 2.28.

```

202 | <ScrollView>
203 |   <View style={{ width: width, alignItems: "center" }}>
204 |     <View style={styles.viewFoto}>
205 |       <Image
206 |         source={require("../assets/imagenes/Logo-principal.jpg")}
207 |         style={{
208 |           width: width / 2,
209 |           height: 80,
210 |           flex: 1,
211 |         }}
212 |         resizeMode="contain"
213 |       />
214 |     </View>
215 |     <Text
216 |       style={{
217 |         fontSize: 18,
218 |         fontWeight: "bold",
219 |         paddingLeft: 30,
220 |         textAlign: "left",
221 |         alignSelf: "flex-start",
222 |       }}
223 |     />
224 |     Promociones:
225 |   </Text>
226 |   <View style={{ height: 10 }} />
227 |   <Swiper ...
228 | />
229 | </View>
230 | <View style={{ height: 10 }} />

```

Código 2.4. Fragmento de código pantalla de inicio rol administrador

A continuación, en la Figura 2.29 se muestra la forma en la que el administrador puede realizar el CRUD (creación, lectura, actualización, eliminación) de los platos que posee el restaurante, la Figura 2.29 (a) muestra como el administrador lee los datos de los platos presentes en el menú, la Figura 2.29 (b) muestra el formulario que permite al administrador la actualización de los datos de los platos del menú, la Figura 2.29 (c) permite visualizar la manera en la que el administrador puede realizar el ingreso de un nuevo plato en el menú.

Cabe recalcar que el menú contará con cuatro opciones las cuales son Sopas, Plato Fuerte, Postres y Bebidas. En el Código 2.6 se muestra parte del código que permite el ingreso de los datos de un nuevo plato al menú por parte del administrador.

En la Figura 2.30 se indica la pantalla para las promociones, estas promociones al igual que el menú pueden ser leídas, creadas, actualizadas y eliminadas. En la Figura 2.31 el administrador podrá ingresar la información de un nuevo empleado para ser registrado en el sistema, esta información incluye sus datos personales, así como el cargo que tiene en el restaurante.

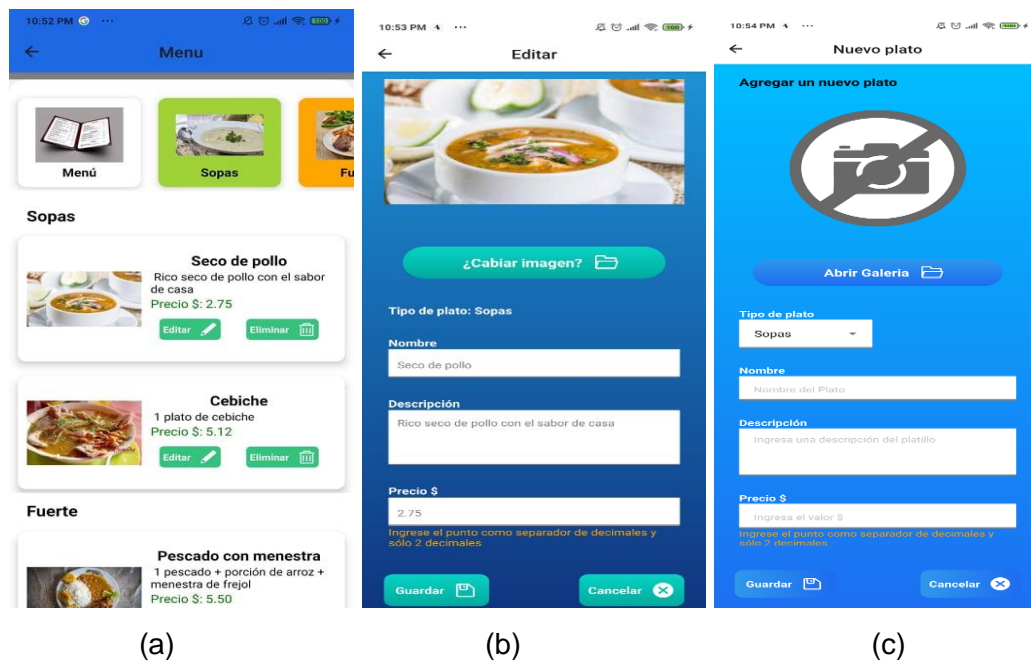


Figura 2.29. UI de (a) Platos en el menú, (b) Actualización de datos del menú, (c) Ingreso de un nuevo plato en el menú.

```

1 import { StatusBar } from "expo-status-bar";
2 import React, { Component } from "react";
3 import {
4   StyleSheet,
5   Text,
6   View,
7   Image,
8   Dimensions,
9   ScrollView,
10  TouchableOpacity,
11  Modal,
12  Platform,
13 } from "react-native";
14 import Swiper from "react-native-swiper";
15 import { LinearGradient } from "expo-linear-gradient";
16 import {
17   AntDesign,
18   Entypo,
19   MaterialIcons,
20   Ionicons,
21   Octicons,
22 } from "@expo/vector-icons";
23 import * as Notifications from "expo-notifications";
24 import Constants from "expo-constants";

```

Código 2.5. Importación de librerías para pantalla de inicio rol administrador

```

157 <TouchableOpacity
158   style={{ marginTop: 20 }}
159   onPress={() => openImagePickerAsync()}
160 >
161   <LinearGradient
162     colors={['#2f98eb', '#1e6cf0']}
163     style={styles.gradient}
164   >
165     <Text style={styles.textGalery}> Abrir Galeria </Text>
166     <AntDesign
167       styles={{ paddingHorizontal: 50 }}
168       name="folderopen"
169       size={26}
170       color="#fff"
171     />
172   </LinearGradient>
173 </TouchableOpacity>
174 <View style={{ height: 10 }} />
175 <Text style={styles.title}>Tipo de plato</Text>
176 <View style={styles.TextUser}>
177   <View style={{ width: 10 }} />
178   <Picker
179     selectedValue={tipoPlato}
180     style={{ height: 50, width: 150 }}
181     onChange={(itemValue, itemIndex) => setTipoPlato(itemValue)}
182     mode="dropdown"
183   >
184     <Picker.Item label="Sopas" value="Sopas" />
185     <Picker.Item label="Fuerte" value="Fuerte" />
186     <Picker.Item label="Postre" value="Postre" />
187     <Picker.Item label="Bebidas" value="Bebidas" />
188     {console.log(tipoPlato)}
189   </Picker>
190 </View>

```

Código 2.6. Fragmento de código creación de un nuevo plato del menú

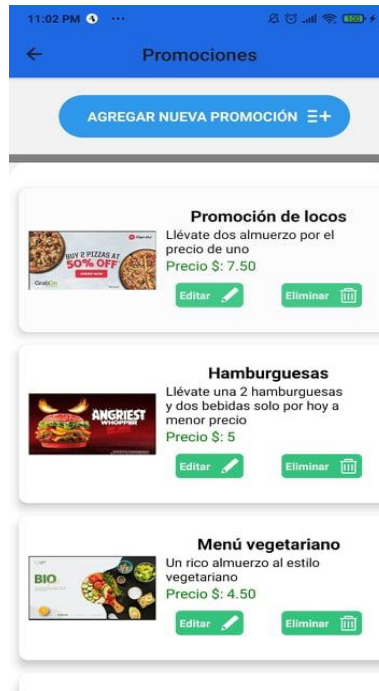


Figura 2.30. UI de las promociones del restaurante

En la Figura 2.32 se despliegan el menú por el cual el administrador podrá navegar hacia las distintas opciones presentes en el prototipo, entre las que se encuentran el perfil de usuario, la opción de cerrar sesión entre otros.

Figura 2.31. Registro de nuevo empleado

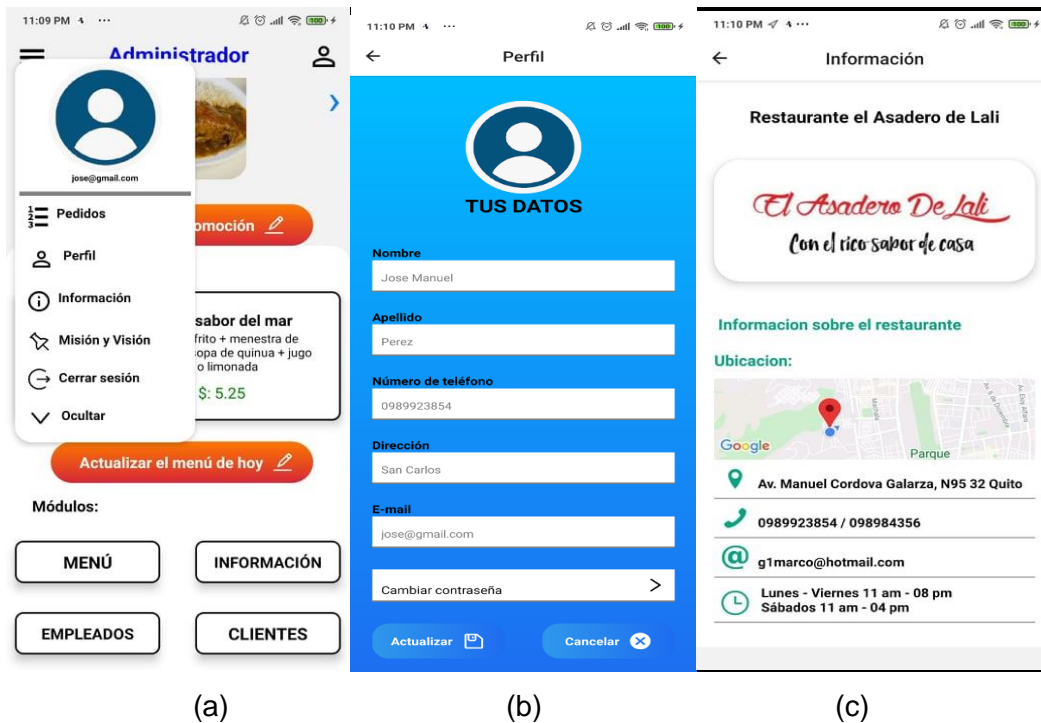


Figura 2.32. (a) Menú desplegable Administrador, (b) Perfil de usuario, (c) Pantalla información del restaurante

2.9.3.2. Módulo Cliente

En este módulo el usuario con el rol cliente podrá realizar las acciones que se muestran a continuación. El cliente antes de poder ingresar a la aplicación deberá registrar sus datos en el sistema, para lo cual debe ingresar sus datos tal como lo muestra la Figura 2.33, una vez ingresados sus datos, se le pedirá que ingrese su dirección para lo cual se usa la API de Google Maps y de esta manera el cliente ingresa su ubicación con una mayor precisión, una vez registrado en el sistema se abrirá la pantalla de inicio de sesión donde deberá ingresar su usuario y su contraseña para ser validados, si los datos son correctos ingresa a la aplicación como se muestra en la Figura 2.34.

A partir de esta pantalla el cliente podrá navegar a las distintas opciones que le ofrece la aplicación entre ellas la de elegir un plato del menú, el menú del día o alguna promoción que esté disponible en ese momento. El Código 2.7 muestra una parte del código usado para el registro de un nuevo cliente.

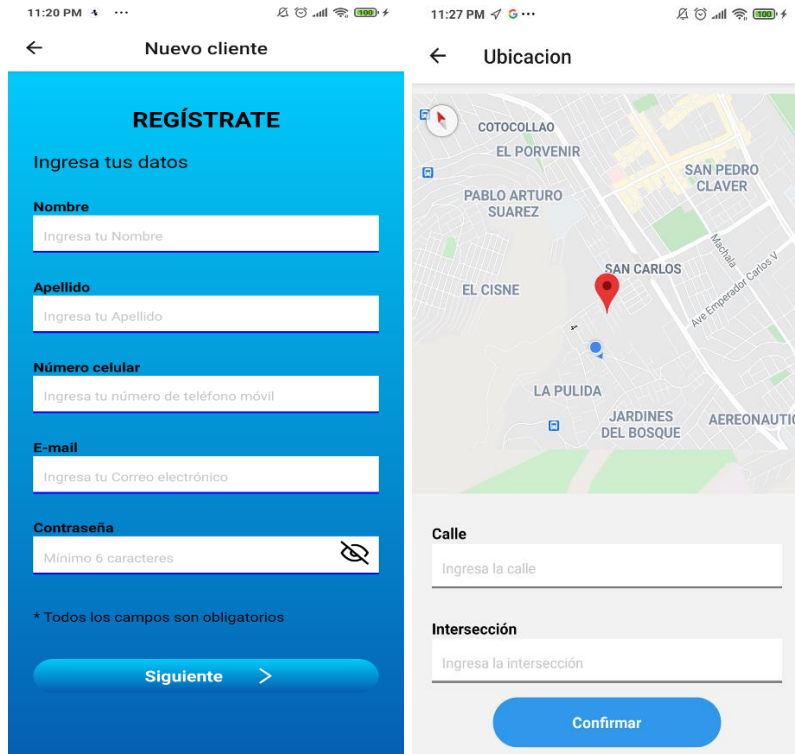


Figura 2.33. Formulario de registro del cliente e ingreso de dirección

```

163 <View style={styles.container}>
164   <StatusBar style="auto" barStyle="light-content" />
165   <Text style={styles.TitleHeader}>REGÍSTRATE</Text>
166   <View style={{ height: 20 }} />
167   <Text style={{ fontSize: 20, color: "black" }}>
168     Ingres tus datos
169   </Text>
170   <Text style={styles.title}>Nombre</Text>
171   <View style={styles.TextUser}>
172     <View style={{ width: 10 }} />
173     <TextInput
174       placeholder="Ingres tu nombre"
175       style={styles.textInput}
176       onChangeText={setNombre}
177       value={nombre}
178     />
179   </View>
180   <Text style={styles.title}>Apellido</Text>
181   <View style={styles.TextUser}>
182     <View style={{ width: 10 }} />
183     <TextInput
184       placeholder="Ingres tu apellido"
185       style={styles.textInput}
186       onChangeText={setApellido}
187       value={apellido}
188     />
189   </View>
190   <Text style={styles.title}>Número celular</Text>

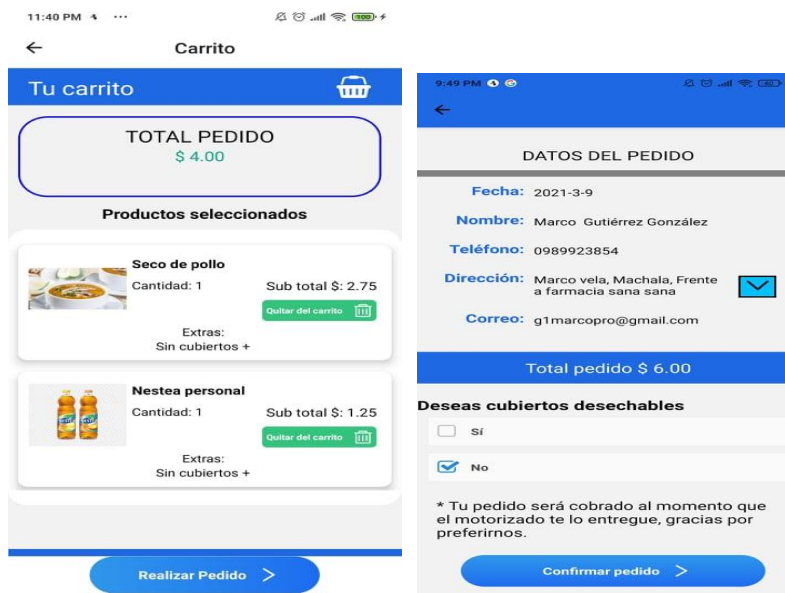
```

Código 2.7. Fragmento de código registro nuevo cliente



Figura 2.34. Pantalla principal rol cliente

Los platos que el usuario elija se los almacenará en un carrito de compras, este carrito de compras guardará los platos elegidos por el cliente hasta que decida realizar la compra, en el carrito de compra estarán los platos que el cliente desea comprar, a su vez también tiene la posibilidad de quitar algún plato del carrito antes de realizar la compra, tal y como se muestra en la Figura 2.35 (a), antes de confirmar la compra el cliente observará el valor total de su pedido, así como también sus datos y deberá elegir la dirección a la cual le será enviado su pedido tal como se indica en la Figura 2.35 (b).



(a)

(b)

Figura 2.35. (a) Carrito de compras, (b) Datos del usuario y valor de la compra

El usuario cliente podrá visualizar su perfil de usuario, esto con la finalidad de actualizar sus datos en el caso de así requerirlo, también cuenta con la posibilidad de actualizar su contraseña tal como se observa en la Figura 2.36 y 2.37.

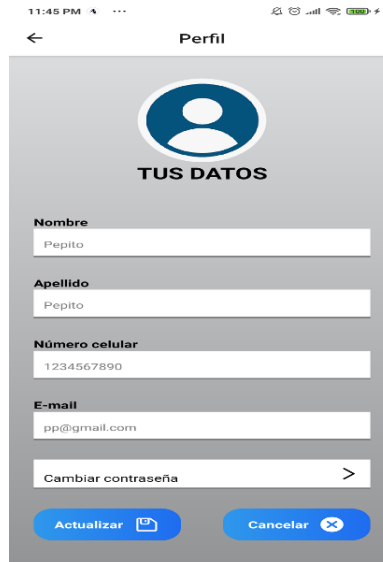


Figura 2.36. Vista de perfil del cliente

Para la actualización de la contraseña, el cliente debe proporcionar la contraseña anterior, esto como una medida de seguridad.



Figura 2.37. Actualización de contraseña

En la Figura 2.38 se muestra el menú con las opciones con las que cuenta el usuario cliente, la opción dirección permite que el cliente ingrese una dirección de entrega adicional a la que ya ingresó al momento del registro de sus datos, tal como se muestra en la Figura 2.39.

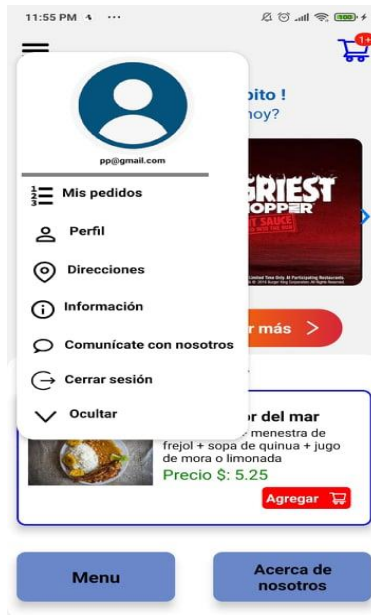


Figura 2.38. Menú desplegable del usuario cliente

La implementación de las interfaces de usuario del módulo cliente, se lo hace de la misma forma que el modulo administrador, es decir las interfaces se las desarrolla mediante código en VS Code, el Código 2.8 muestra una parte del código para el ingreso de datos de un nuevo cliente. Así también, el Código 2.9 muestra una parte de la implementación de la UI del registro de nuevo cliente.



Figura 2.39. Ingreso de nueva dirección

```

42 export default function NuevoCliente({ navigation }) {
43   const [nombre, setNombre] = useState("");
44   const [apellido, setApellido] = useState("");
45   const [email, setEmail] = useState("");
46   const [password, setPassword] = useState("");
47   const [telefono, setTelefono] = useState("");
48   const [expoPushToken, setExpoPushToken] = useState("");
49   const [loading, setLoading] = useState(false);
50   const item = "Login";

```

Código 2.8. Fragmento de código para el ingreso de datos del nuevo cliente


```

156     return (
157       <KeyboardAwareScrollView keyboardShouldPersistTaps="always">
158         <View style={{ flex: 1, height: height }}>
159           <LinearGradient
160             colors={['#01caf9', '#055db1']}
161             style={{ padding: 5, flex: 1 }}
162           >
163             <View style={styles.container}>
164               <StatusBar style="auto" barStyle="light-content" />
165               <Text style={styles.TitleHeader}>REGÍSTRATE</Text>
166               <View style={{ height: 20 }} />
167               <Text style={{ fontSize: 20, color: "black" }}>
168                 | Ingresar tus datos
169               </Text>
170               <Text style={styles.title}>Nombre</Text>
171               <View style={styles.TextUser}>
172                 <View style={{ width: 10 }} />
173                 <TextInput
174                   placeholder="Ingresar tu nombre"
175                   style={styles.textInput}
176                   onChangeText={setNombre}
177                   value={nombre}
178                 />
179               </View>
180               <Text style={styles.title}>Apellido</Text>
181               <View style={styles.TextUser}>
182                 <View style={{ width: 10 }} />
183                 <TextInput
184                   placeholder="Ingresar tu apellido"
185                   style={styles.textInput}
186                   onChangeText={setApellido}
187                   value={apellido}

```

Código 2.9. Fragmento de código para UI de nuevo cliente

2.9.3.3. Módulo empleado

En este módulo el usuario con el rol Empleado podrá realizar las acciones que se muestran a continuación. El empleado deberá ser ingresado al sistema por el usuario administrador, de lo contrario no podrá ingresar a la aplicación; el empleado deberá iniciar sesión con sus credenciales de correo electrónico y contraseña, si estos son correctos, se mostrará la pantalla principal del usuario empleado, tal y como se muestra en la Figura 2.40.



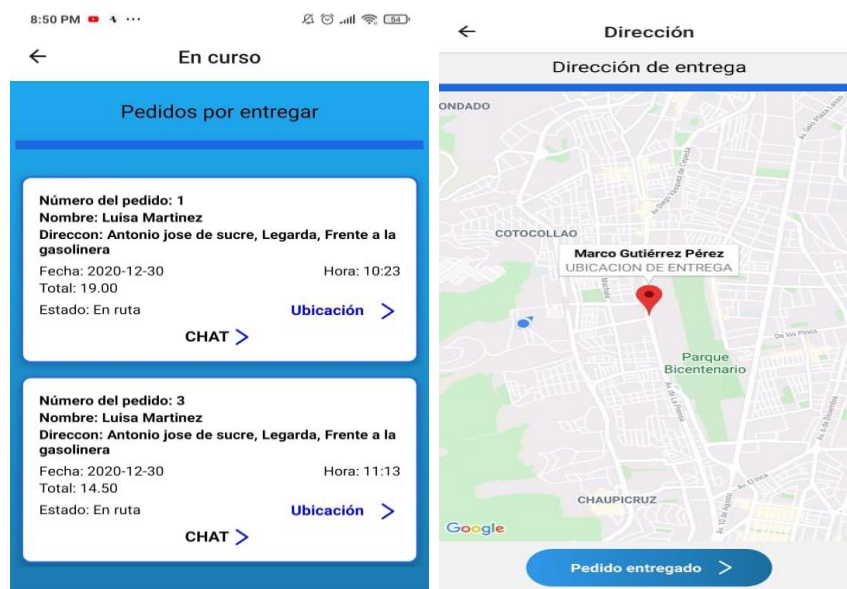
Figura 2.40. Página principal Empleado

A partir de esta pantalla el empleado podrá acceder a las opciones que ofrece la aplicación móvil. Las opciones mostradas en la pantalla principal son: Pedidos pendientes, pedidos en curso y pedidos entregados. La Figura 2.41 muestra la pantalla de pedidos pendientes, en esta pantalla se mostrará los pedidos hechos por los clientes, para que los empleados puedan observar cuales y cuantos son los productos que deben ser enviados.



Figura 2.41. Pedidos pendientes

La Figura 2.42 muestra la pantalla de pedidos en camino, en esta pantalla se mostrarán los pedidos que están en curso de entrega, es decir aquellos pedidos que fueron entregados al motorizado para su envío al cliente; así también se mostrará en el mapa la ubicación a la cual deben ser entregados los productos.



(a)

(b)

Figura 2.42. (a) Pedidos en camino, (b) Dirección de entrega

La Figura 2.43 muestra la pantalla de pedidos entregados, en esta pantalla se presentarán todos los pedidos que ya han sido entregados a los clientes.



Figura 2.43. Pedidos entregados

En la Figura 2.44 se presenta el menú desplegable al cual tiene acceso el usuario empleado, podrá ingresar a su perfil para actualizar sus datos, así como también podrá actualizar su contraseña en caso de requerirlo.

El Código 2.10 muestra una parte del código que permite la implementación de la UI de los pedidos pendientes, el Código 2.11 muestra la función que permite cargar en la aplicación móvil los productos del pedido para ser agrupados para su envío.



Figura 2.44. Menú desplegable del empleado

```

81     return (
82         <TouchableOpacity
83     > style={styles.Pedido} ...
86     </TouchableOpacity>
87     <View
88     > style={{ ...
94     </View>
95     <Text style={styles.styleText}>Pedido: {item.Id}</Text>
96     <Text style={styles.styleText}>
97     | Nombre: {item.Nombre} {item.Apellido}
98     </Text>
99     <Text style={styles.styleText}>Dirección: {item.Direccion}</Text>
100    <View style={styles.styleView}>
101    | <Text>Fecha: {item.Fecha}</Text>
102    | <Text>Total: {item.Total}</Text>
103    </View>
104    <View style={styles.styleView}>
105    | <Text>Estado: {item.Estado}</Text>
106    | <TouchableOpacity
107    | style={{
108    |   flexDirection: "row",
109    |   borderRadius: 5,
110    |   alignSelf: "flex-end",
111    | }}
112    | onPress={() => {
113    |   navigation.navigate("DetalleEmpleado", { item });
114    | }}
115    | <Text style={{ fontSize: 15, color: "blue", fontWeight: "bold" }}>
116    | | Detalle
117    | </Text>
118    </View>
119    </View>

```

Código 2.10. Implementación de la UI pedidos pendientes

```

28    useEffect(() => {
29        let unmounted = true;
30        var key = firebase.auth().currentUser.uid;
31        var vector = [];
32        try {
33            ref
34                .ref("pedidoEmpleado/")
35                .orderByChild("Estado")
36                .equalTo("Pendiente")
37                .on("value", (snapshot) => {
38                    var datos = snapshot.val();
39                    if (unmounted) {
40                        for (var key in datos) {
41                            const obj = {
42                                Id: datos[key].Id,
43                                Nombre: datos[key].Nombre,
44                                Apellido: datos[key].Apellido,
45                                Estado: datos[key].Estado,
46                                Fecha: datos[key].Fecha,
47                                Telefono: datos[key].Telefono,
48                                Total: datos[key].Total,
49                                Email: datos[key].email,
50                                UserId: datos[key].UserId,
51                                Direccion: datos[key].Direccion,
52                                Token: datos[key].Token,
53                            };
54                            vector.push(obj);
55                        }
56                        setDatosPedido(vector);
57                    }
58                    vector = [];
59                });
60        } catch (error) { ...
62    }

```

Código 2.11. Función que carga datos de la base de datos en la aplicación

2.9.3.4. Implementación navegación entre pantallas

Para desarrollar la navegación entre pantallas se usa la librería React Navigation, en el prototipo se define la navegación por stack, el cual es una pila de pantallas por medio de las cuales se navega entre pantallas.

```
1  import React, { useState, useRef } from "react";
2  import { createStackNavigator } from "@react-navigation/stack";
3  import { StyleSheet } from "react-native";
4  import MainAdministrador from "../screens/Administrador/MainAdministrador";
5  import Menu from "../screens/Administrador/MenuAdmin";
6  import ListaPromo from "../screens/Administrador/ListaPromociones";
7  import Informacion from "../screens/Administrador/Informacion";
8  import NuevaPromocion from "../screens/Administrador/NuevaPromocion";
9  import EditPromo from "../screens/Administrador/Edicion/EditarPromocion";
10 import NuevoPlato from "../screens/Administrador/NuevoPlato";
11 import EditMenu from "../screens/Administrador/Edicion/EditarMenu";
12 import NuevoEmpleado from "../screens/Administrador/NuevoEmpleado";
13 import Empleados from "../screens/Administrador/ListaEmpleados";
14 import EditarEmpleados from "../screens/Administrador/Edicion/EditarEmpleado";
15 import PerfilAdministrador from "../screens/Administrador/PerfilAdministrador";
16 import CambiarPassword from "../screens/CambiarPassword";
17 import ControlPedido from "../screens/Administrador/ControlPedidos";
18 import DetalleAdmin from "../screens/Administrador/DetalleAdministrador";
19 import Clientes from "../screens/Administrador/ListaClientes";
20 import MenuDiario from "../screens/Administrador/MenuDiario";
21 import Nosotros from "../Utilidades/Mision";
22 import Chat from "../Chat/Chat";
```

Código 2.12. Importación de rutas de las pantallas de navegación

El Código 2.12 muestra la importación de las pantallas en el stackAdministrador, de la línea 4 a la línea 22 se importan las pantallas que conforman el modulo administrador, en el Código 2.13 se muestra parte del código para la implementación de la navegación entre pantallas para el modulo Administrador.

```
26  export default function StackAdministrador({ navigation }) {
27    return (
28      <Stack.Navigator>
29        <Stack.Screen
30          name="Main"
31          component={MainAdministrador}
32          options={() => ({
33            title: "",
34            headerTitleAlign: "center",
35            headerShown: false,
36          })}
37        />
38        <Stack.Screen
39          name="Menu"
40          component={Menu}
41          options={() => ({
42            title: "Menu",
43            headerTitleAlign: "center",
44            headerStyle: { backgroundColor: "#1e69e3" },
45          })}
46        />
47        <Stack.Screen
48          name="listaPromo"
49          component={ListaPromo}
50          options={() => ({
51            title: "Promociones",
52            headerTitleAlign: "center",
53            headerStyle: { backgroundColor: "#1e69e3" },
54          })}
55        />
56      )
57    }
58  }
```

Código 2.13. Stack de navegación para el módulo administrador

Un `Stack.Navigator` (línea 28) agrupa a los elementos que son parte de la pila de navegación, en otras palabras, agrupa a las pantallas (`Stack.Screen`) línea 29 a 54, que forman parte del módulo administrador, dentro de los `Stack.Screen` se define varias opciones entre ellas el nombre con el cual será invocada la pantalla (`title`), el color de la cabecera entre otros.

De la misma forma se implementan los Stack de navegación para los módulos Cliente y Empleado; estos módulos no tendrán relación entre ellos es por eso que sus Stacks se los implementa de forma separada. El Código 2.14 muestra un fragmento de código para la implementación del Stack del módulo cliente.

```
23 export default function StackCliente({ navigation }) {
24   return (
25     <Stack.Navigator headerMode="screen">
26       <Stack.Screen
27         name="MainCliente"
28         component={MainCliente}
29         options={{
30           title: "",
31           headerShown: false,
32         }}
33       />
34       <Stack.Screen
35         name="MenuCliente"
36         component={MenuCliente}
37         options={() => ({
38           title: "Menu",
39           headerTitleAlign: "center",
40         })}
41       />
42       <Stack.Screen
43         name="Promociones"
44         component={Promociones}
45         options={() => ({
46           title: "Promociones",
47           headerTitleAlign: "center",
48           headerStyle: { backgroundColor: "#1e69e3" },
49         })}
50     />
51   )
52 }
```

Código 2.14. Stack de navegación para el módulo cliente

La navegación se lo realiza a través del método `navigate`, el cual está definido en `react-navigation`. Este método admite el nombre de la pantalla hacia la cual se va a navegar y también admite valores en el caso de enviar datos de una pantalla a otra, el Código 2.15 muestra el método `navigation`.

```
.navigation.navigate("Menu");
```

Código 2.15. Método `navigation`

2.9.3.5. Instalación de Firebase

Para realizar la conexión entre la aplicación y la base de datos ofrecida por Firebase hay que agregar la librería firebase, la cual esta provista con los servicios que provee Firebase, tales como bases de datos, autenticación y almacenamiento. El Código 2.16 muestra el código para instalar el SDK de Firebase en la aplicación móvil.

```
expo install firebase
```

Código 2.16. Instalación SDK de Firebase

Una vez instalado el SDK de Firebase en la aplicación React Native se debe agregar la clave de acceso que Firebase entrega al momento de crear un nuevo proyecto, el Código 2.17 presenta la inicialización de Firebase con las claves de acceso del proyecto.

```
1 import * as firebase from "firebase";
2
3 var firebaseConfig = {
4   apiKey: "AIzaSyCwp507Qu1AF0vN5_FHWG59ovj4vFOMUeI",
5   authDomain: "appfire-2668d.firebaseio.com",
6   databaseURL: "https://appfire-2668d.firebaseio.com",
7   projectId: "appfire-2668d",
8   storageBucket: "appfire-2668d.appspot.com",
9   messagingSenderId: "271629350062",
10  appId: "1:271629350062:web:03951974b94ad190cca93f",
11 };
12
13 try {
14   firebase.initializeApp(firebaseConfig);
15 } catch (e) { ...
16 }
17
18 export default firebase;
```

Código 2.17. Inicialización de Firebase

En la línea 1 se importa firebase, entre las líneas 3 y 11 se presenta las claves de acceso del proyecto de Firebase las cuales están asignadas a una variable firebaseConfig, en la línea 14 se procede a realizar la inicialización de la conexión con Firebase. Y como último paso se exportan las instancias para que puedan ser usadas en el prototipo.

2.9.3.6. Codificación de métodos

En esta sección se presentarán las funcionalidades más sobresalientes con las que cuenta el prototipo, entre ellas: crear una cuenta de usuario, editar los platos del menú, agregar una dirección de entrega, entre otras.

El Código 2.18 muestra parte del código que permite la creación de un nuevo cliente en el sistema, en la línea 90 se crea la función nuevoCliente, la cual verifica que los datos solicitados en la Figura 2.34 no estén vacíos, así como verificar que la contraseña contenga un número mínimo de 6 caracteres, caso contrario muestra un mensaje de advertencia en pantalla, posteriormente, en la línea 109 se invoca a la función createUserWithEmailAndPassword(), la cual es una función que recibe como parámetros un email y una contraseña, esto permite crear un usuario en el servicio de autenticación de Firebase. De forma secuencial se guarda los datos del usuario tales como nombre, apellido, email y teléfono en la base de datos de Firebase en la ruta Usuarios/Clientes+key, donde key, es el identificador de usuario (uid) que Firebase asigna al usuario.

El Código 2.19 muestra parte del código implementado para la autenticación de los usuarios. La línea 67 crea la función login la cual, verifica que tanto el correo como la contraseña sean valores no vacíos, tal como se muestra en la Figura 2.27, una vez validados estos datos, esta función devuelve el método signInWithEmailAndPassword la cual requiere un email y contraseña para proceder a realizar la autenticación, si los datos son correctos se invoca a la función verUsuarios caso contrario se muestra en pantalla un mensaje de error.

```
90  const nuevoCliente = () => {
91    if (
92      nombre === "" ||
93      apellido === "" ||
94      email === "" ||
95      password === "" ||
96      telefono === ""
97    ) {alert("No pueden quedar campos vacios");
98  > } else if (!validacion(email)) {...
100 > } else if (password.length < 6) {...
102 > } else if (telefono.length < 10) {...
104 } else {
105   setLoading(true);
106   firebase
107     .auth()
108     .createUserWithEmailAndPassword(email, password)
109     .then(() => { var key = firebase.auth().currentUser.uid;
110       ref
111         .ref("Usuarios/Clientes/" + key)
112         .set({
113           Nombre: nombre,
114           Apellido: apellido,
115           TipoUsuario: "Cliente",
116           Telefono: telefono,
117           email: email,
118           Estado: "Habilitado",
119           UserId: firebase.auth().currentUser.uid,
120           Token: expoPushToken,
121         })
122 >     .then((data) => {...
125     })
126     .catch((error) => {
127       alert("Error al crear la cuenta, inténtelo más tarde");
128     });
```

Código 2.18. Función nuevoCliente


```

67     const login = () => {
68       if (email === "" || password === "") {
69         alert("No pueden quedar campos vacíos");
70       } else if (!validacion(email)) {
71         alert("El email no es correcto");
72       } else {
73         setLoading(true);
74         firebase
75           .auth()
76           .signInWithEmailAndPassword(email, password)
77           .then(() => {
78             setEmail("");
79             setPassword("");
80             verUsuario();
81           })
82           .catch((error) => {
83             setLoading(false);
84             if (
85               error ==
86               "Error: There is no user record corresponding to this identifier.
87             ) {
88               alert("Usuario no encontrado");
89             } else {
90               alert("Email o contraseña incorrecta");
91             }
92             console.log(error);
93           });
94       }
95     };

```

Código 2.19. Función login

El Código 2.20 presenta la función verUsuario, en la línea 105 se carga la información de la base de datos con la ruta Usuarios/, de la línea 107 a la línea 109, se carga la información de los nodos hijos del nodo Usuarios, se verifica si esta información es nula o no lo es, y también se verifica que el usuario este habilitado en el sistema, dependiendo del resultado se navega a la pantalla principal del módulo correspondiente.

```

97     const verUsuario = () => {
98       let mounted = true;
99       var datos3;
100      var datos1;
101      var datos2;
102      var key = firebase.auth().currentUser.uid;
103      firebase
104        .database()
105        .ref("Usuarios/")
106        .on("value", (snapshot) => {
107          datos1 = snapshot.child("Clientes/" + key).val();
108          datos2 = snapshot.child("Administrador/" + key).val();
109          datos3 = snapshot.child("Empleados/" + key).val();
110          if (datos1 != null) {
111            if (datos1.Estado === "Habilitado") {
112              setLoading(false);
113              navigation.navigate("StackCliente");
114            } else {
115              setLoading(false);
116              alert("Usuario no encontrado");
117            }
118          } else if (datos2 != null) {
119            setLoading(false);
120            navigation.navigate("StackAdministrador");
121          } else if (datos3 != null) {
122            if (datos3.Estado === "Habilitado") {
123              setLoading(false);
124              navigation.navigate("StackEmpleado");
125            } else {
126              setLoading(false);
127              alert("Usuario no encontrado");
128            }
129          }
130        });
131     };

```

Código 2.20. Función verUsuario

El Código 2.21 presenta un fragmento de código para el ingreso de un nuevo plato al menú, la línea 68 guarda en el Storage de Firebase la imagen del plato en la ruta Menu/NombrePlato, una vez guardada la imagen del plato se invoca la función `snapshot.ref.getDownloadURL` (líneas 74 y 75), función que devuelve como respuesta la URL de la imagen, la misma que será guardada en la base de datos junto con la información del plato en la ruta `Menu/tipoPlato/key` (líneas 80 a línea 96), estos datos previamente se validaron para que no sean nulos, el ingreso de un nuevo plato al menú requiere de una imagen como se lo vio anteriormente, para lo cual se debe pedir los permisos necesarios para que la aplicación tome las imágenes de la galería del dispositivo, el Código 2.22 muestra el código para solicitar los permisos al usuario. Se debe agregar previamente la siguiente librería:

- Expo-image-picker, la cual permite tomar imágenes que se encuentren en la galería del dispositivo.

```

68     var uploadTask = refStorage.child("Menu/" + NombrePlato).put(resolve);
69     uploadTask.on(
70         "state_changed",
71         function (snapshot) {},
72         function (error) {},
73         function () {
74             uploadTask.snapshot.ref
75                 .getDownloadURL()
76                 .then(function (downloadURL) {
77                     console.log("File available at", downloadURL);
78                     console.log(typeof downloadURL);
79                     uploadURI = downloadURL;
80                     var key = firebase
81                         .database()
82                         .ref("Menu/" + tipoPlato + "/" +
83                             ".push({}).key");
84                     ref
85                         .ref("Menu/" + tipoPlato + "/" + key)
86                         .update({
87                             Id: key,
88                             Nombre: NombrePlato,
89                             URL: downloadURL,
90                             Descripcion: Descripcion,
91                             Precio: Precio,
92                             Tipo: tipoPlato,
93                         })
94                         .then((data) => {
95                             setLoading(false);
96                             navigation.navigate("Menu");
97                         })
98                         .catch((error) => {
99                             setLoading(false);
100                             alert("Error al subir el plato, inténtelo más tarde");
101                         });
102                 });

```

Código 2.21. Fragmento de código para crear nuevo plato del menú

La línea 31 crea la función `openImagePickerAsync`, la línea 33 verifica que los permisos hayan sido concedidos mostrando en pantalla un mensaje en el caso de que no se haya permitido el ingreso a la galería del dispositivo, la línea 38, una vez que se hayan dado los permisos, verifica que se haya seleccionado una imagen de la galería, caso contrario muestra un mensaje en pantalla, la línea 60 se guarda el URI de la imagen seleccionada dentro de la variable `seletedImage`.

```
31 | let openImagePickerAsync = async () => {
32 |   let permissionResult = await ImagePicker.requestCameraRollPermissionsAsync();
33 |   if (permissionResult.granted === false) {
34 |     alert("Los permisos de la cámara son requeridos!");
35 |     return;
36 |   }
37 |   let pickerResult = await ImagePicker.launchImageLibraryAsync();
38 |   if (pickerResult.cancelled === true) {
39 |     alert("Has cerrado la galería sin seleccionar ninguna imagen");
40 |     return;
41 |   }
42 |   setSelectedImage({ selectedImage: pickerResult.uri });
43 | };
```

Código 2.22. Permisos para abrir la galería en Android con React Native

El Código 2.23 presenta un fragmento de código que permite cargar los datos del menú alojados en Firebase en la aplicación móvil, la línea 35 invoca la función `componentDidMount` la cual se ejecuta de forma automática después la primera vez que se renderizan los componentes, de la línea 36 a la línea 40 se definen variables auxiliares, la línea 42 carga los datos almacenados en la base de datos de Firebase que se encuentran ubicados en la ruta `Menu/`, esta ruta contiene los datos del menú del restaurante El asadero de Lali, de la línea 43 a la línea 46 se almacena los datos cargados de acuerdo al tipo de plato, de la línea 48 a la línea 54 se almacenan los datos de cada tipo de plato en un objeto el cual a su vez también será almacenado en un vector para posteriormente ser agregados a una variable de estado (línea 58), esto se lo hace para los cuatro tipos de platos del menú.

El Código 2.24 muestra una parte del código que permite mostrar en pantalla los datos e imágenes de los platos del menú almacenados en la base de datos de Firebase, en la línea 143 se crea la función `renderItemMenu`, la cual acepta como parámetro de entrada un objeto, el cual es uno de los datos que fueron cargados desde la base de datos presentados en el Código 2.23, en la línea 145 se agrega el componente `TouchableOpacity`, el cual es un botón que servirá de contenedor para los demás elementos, en la línea 146 se agrega el componente `Image`, el cual permite cargar la imagen del plato almacenada en Firebase, en el caso de que no exista la imagen, mostrará una imagen por defecto la cual está

almacenada en el directorio con la dirección ../../assets/imagenes/no-image.png, la línea 155 implementa el componente View, el cual sirve como contenedor para otros componentes como Text (línea 163 a línea 181) los cuales mostrarán en pantalla el nombre, precio y descripción del plato del menú.

```

35     componentDidMount() {
36         this._isMounted = true;
37         var vector = [];
38         var vector2 = [];
39         var vector3 = [];
40         var vector4 = [];
41         try {
42             ref.ref("Menu/").on("value", (snapshot) => {
43                 var datos = snapshot.child("Sopas").val();
44                 var datos2 = snapshot.child("Fuerte").val();
45                 var datos3 = snapshot.child("Postre").val();
46                 var datos4 = snapshot.child("Bebidas").val();
47                 for (var key in datos) {
48                     let obj = {
49                         Id: datos[key].Id,
50                         Nombre: datos[key].Nombre,
51                         URL: datos[key].URL,
52                         Descripcion: datos[key].Descripcion,
53                         Precio: datos[key].Precio,
54                         Tipo: datos[key].Tipo,
55                     };
56                     vector.push(obj);
57                 }
58                 this.setState({ datosSopas: vector });
59                 vector = [];

```

Código 2.23. Fragmento de código que carga el menú en la aplicación móvil

```

143     renderItemMenu(item) {
144         return (
145             <TouchableOpacity style={styles.Menu}>
146                 <Image
147                     style={{ width: width / 3, height: width / 3 }}
148                     resizeMode="contain"
149                     source={{
150                         item.URL
151                         ? { uri: item.URL }
152                         : require("../../assets/imagenes/no-image.png")
153                     }}
154                 />
155                 <View
156 > style={{...
161                 }}
162                 >
163                     <Text
164 > style={{...
168                     }}
169                     >
170                         {item.Nombre}
171                     </Text>
172                     <Text> {item.Descripcion}</Text>
173                     <Text
174 > style={{...
178                     }}
179                     >
180                         Precio $: {item.Precio}
181                     </Text>

```

Código 2.24. Fragmento de la función renderItemMenu

La Figura 2.45 muestra el resultado de la implementación del Código 2.24.

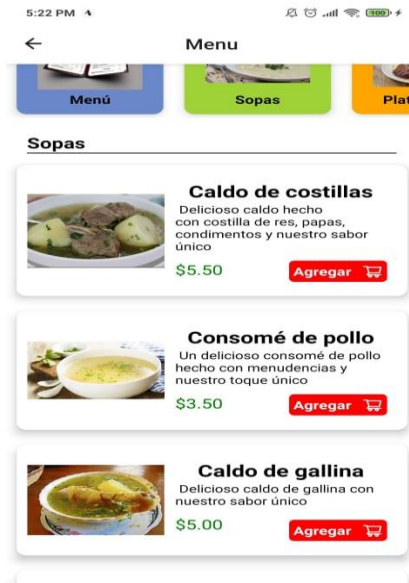


Figura 2.45. Pantalla platos del menú

Para mostrar en pantalla el mapa y obtener la ubicación del dispositivo hay que agregar las siguientes librerías:

- `react-native-maps`: la cual contiene componente que usan los servicios de Google Maps.

Para tomar la posición del usuario hay que pedir los permisos respectivos, así como también, leer los datos de ubicación del dispositivo móvil, para la cual se agrega las siguientes librerías:

- `expo-permissions`: permite solicitar permisos en el dispositivo móvil.
- `expo-location`: posibilita leer la información de geolocalización del dispositivo móvil.

El Código 2.25 muestra un fragmento del código que importa los módulos para presentar en pantalla la ubicación en el mapa (línea 16), la librería para solicitar permisos en el dispositivo móvil (línea 14), y la librería para obtener la ubicación del dispositivo (línea 15).

```
14 import * as Permissions from "expo-permissions";
15 import * as Location from "expo-location";
16 import MapView from "react-native-maps";
```

Código 2.25. Fragmento de código importación de librerías

El Código 2.26 presenta parte del código para la implementación en pantalla del mapa para agregar una dirección de entrega, la línea 133 muestra la implementación del componente View, el cual sirve como contenedor para la visualización del mapa en pantalla, de la línea 135 a la línea 151 se presenta la configuración del mapa, para lo cual, se usa las propiedades `initialRegion` (línea 137), en esta propiedad se configura los valores de longitud y latitud iniciales del mapa, se configura también propiedades como `scrollEnabled` (línea 150), `zoomEnabled` (línea 151), las cuales permiten una mejor visualización del mapa.

```

133 | <View>
134 |   {location && (
135 |     <MapView
136 |       style={styles.mapStyle}
137 |       initialRegion={
138 |         location
139 |         ? location
140 |         : {
141 |           latitude: -0.133772,
142 |           longitude: -78.498105,
143 |           latitudeDelta: 0.01649,
144 |           longitudeDelta: 0.042,
145 |         }
146 |       }
147 |       showsUserLocation={true}
148 |       onRegionChange={(region) => setLocation(region)}
149 |       followsUserLocation={true}
150 |       scrollEnabled={true}
151 |       zoomEnabled={true}
152 |     >
153 |       <MapView.Marker
154 |         coordinate={{
155 |           latitude: location.latitude,
156 |           longitude: location.longitude,
157 |         }}
158 |         draggable
159 |       />
160 |     </MapView>
161 |   )}
162 | </View>

```

Código 2.26. Fragmento del código ubicación de entrega

En la línea 153 se configura el componente `MapView.Marker`, la línea 154 se configura la propiedad `coordinate`, la cual toma los valores de longitud y latitud de la posición del usuario, en la línea 158 se activa la propiedad `draggable`, la cual permite visualizar en pantalla un marcador para determinar la posición del usuario.

El Código 2.27 muestra el código que permite solicitar los permisos de ubicación en la aplicación móvil. En la línea 63 se crea la función `permisos`, la cual es la función encargada de pedir al usuario los permisos para acceder a la ubicación del dispositivo.

La línea 65 crea una constante de tipo `Permissions` la cual almacenará el resultado de haber pedido permisos al usuario, la línea 68 crea una constante la cual almacena el estado

del permiso dado a la solicitud de la localización del dispositivo, la línea 69 verifica que los permisos hayan sido aceptados, de lo contrario muestra un mensaje en pantalla indicando al usuario que los permisos son necesarios, una vez concedidos los permisos, se toma los valores de ubicación del dispositivo (línea 73), y se los guarda en la variable de estado Location (línea 74 a línea 79).

```
63   const permisos = () => {
64     async () => {
65       const resultPermissions = await Permissions.askAsync(
66         Permissions.LOCATION
67       );
68       const statusPermissions = resultPermissions.permissions.location.status;
69       if (statusPermissions !== "granted") {
70         alert("Tienes que aceptar los permisos de localización", 3000);
71       } else {
72         try {
73           const loc = await Location.getCurrentPositionAsync({});
74           setLocation({
75             latitude: loc.coords.latitude,
76             longitude: loc.coords.longitude,
77             latitudeDelta: 0.01,
78             longitudeDelta: 0.01,
79           });
80         } catch (error) {
81           alert(
82             "No podemos encontrar tu posición. Asegurate de tener encendida tu ubicación"
83           );
84         }
85       }
86     };
87   };
```

Código 2.27. Permiso para la ubicación del usuario

De acuerdo a los requerimientos funcionales el cliente debe poder hacer un pedido al restaurante, para lo cual se debe crear un carrito de compras, el Código 2.28 muestra un fragmento de código que permite visualizar en pantalla los ítems que contenga el carrito de compras.

De línea 103 a la línea 107 se presenta el código que permite mostrar en pantalla la imagen del producto seleccionado, en la línea 108 se implementa el componente View, el cual contiene otros componentes como el componente Text que muestra en pantalla el nombre del plato seleccionado (línea 123), así como la cantidad seleccionada (línea 128) y el subtotal (línea 135), el resultado de este código es mostrado en la Figura 2.46.

```

103 <Image
104   style={{ width: width / 4, height: width / 4 }}
105   resizeMode="contain"
106   source={{ uri: item.URL }}
107 >/>
108 <View
109 > style={{ ...
114   }}
115 >
116   <Text
117 > style={{ ...
121   }}
122   >
123     {item.Nombre}
124   </Text>
125   <View
126   style={{ flexDirection: "row", justifyContent: "space-between" }}
127   >
128     <Text>Cantidad: {item.Cantidad}</Text>
129     <Text
130 > style={{ ...
133   }}
134   >
135     Sub total $: {item.SubTotal}
136   </Text>
137 </View>

```

Código 2.28. Fragmento de código carrito de compras

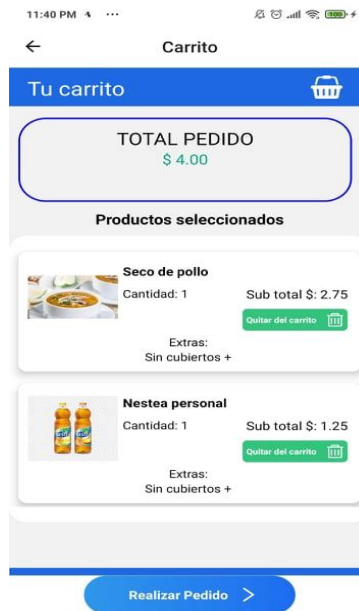


Figura 2.46. Carrito de compras

El Código 2.29 muestra el código que permite remover del carrito de compras un producto que ya no se desee comprarlo.

La línea 89 crea la función actualizar, la cual recibe como parámetro de entrada el objeto que se desea eliminar, para lo cual se adquiere el ID del usuario actual (línea 90), y con

este dato se arma la ruta en la base de datos almacenada en Firebase (línea 91) y se procede a borrar dicho plato del carrito de compras (línea 92 a línea 95).

```
89 actualizar(item) {
90   const UserId = firebase.auth().currentUser.uid;
91   db.ref("carrito/" + UserId + "/" + item.Id)
92     .remove()
93     .then(() => {})
94     .catch((err) => {
95       console.log(err);
96     });
97 }
```

Código 2.29. Función actualizar

El Código 2.30 muestra el código que permite al usuario empleado observar el pedido hecho por el cliente.

```
28 useEffect(() => {
29   let unmounted = true;
30   var key = firebase.auth().currentUser.uid;
31   var vector = [];
32   try {
33     ref
34       .ref("pedidoEmpleado/")
35       .orderByChild("Estado")
36       .equalTo("Pendiente")
37       .on("value", (snapshot) => {
38         var datos = snapshot.val();
39         if (unmounted) {
40           for (var key in datos) {
41             const obj = {
42               Id: datos[key].Id,
43               Nombre: datos[key].Nombre,
44               Apellido: datos[key].Apellido,
45               Estado: datos[key].Estado,
46               Fecha: datos[key].Fecha,
47               Telefono: datos[key].Telefono,
48               Total: datos[key].Total,
49               Email: datos[key].email,
50               UserId: datos[key].UserId,
51               Direccion: datos[key].Direccion,
52               Token: datos[key].Token,
53             };
54             vector.push(obj);
55           }
56           setDatosPedido(vector);
57         }
58         vector = [];
59       });
60   } catch (error) { ...
61   }
```

Código 2.30. Función useEffect

La línea 28 invoca a la función `useEffect` la cual se ejecuta inmediatamente después de que la pantalla se haya renderizado, en las líneas 29 a la línea 31 se crean variables auxiliares, la línea 34 carga los datos de la base de datos almacenados en la ruta `pedidoEmpleado/`, pero solo cargará los pedidos que se encuentren con la designación “Pendiente”, la línea 41 a la línea 52 extrae los datos del pedido siempre y cuando los componentes se encuentren montados y los almacena en la variable auxiliar (línea 54), y posteriormente se los guarda en una variable de estado (línea 56).

De acuerdo a los requerimientos funcionales, el usuario cliente como el usuario empleado necesitan recibir una notificación en su dispositivo móvil cuando su pedido este en camino de entrega, así como cuando un nuevo pedido llegue al restaurante respectivamente.

Para esto, se agrega la librería `expo-notifications`, para permitir el envío de notificaciones push, el Código 2.31 muestra la importación de la librería.

```
import * as Notifications from "expo-notifications";
```

Código 2.31. Importación librería `expo-notifications`

En el archivo `app.json` se agrega el código mostrado en el Código 2.32.

```
"android": {  
  "package": "com.AsaderoDeLali.AsaderoDeLali",  
  "useNextNotificationsApi": true,  
  "versionCode": 1,  
}
```

Código 2.32. Fragmento de código archivo `app.json`

El Código 2.33 presenta la función `registerForPushNotificationsAsync`, la cual regresa como resultado un token (identificador único del dispositivo móvil), de la línea 75 a la línea 78, se verifica el status del dispositivo, en las líneas 80 a la línea 89 se verifica que los permisos para las notificaciones hayan sido permitidos, caso contrario se muestra en pantalla un mensaje de advertencia (línea 87 y línea 93), cuando los permisos han sido aceptados se genera el token, de la línea 96 a la línea 102 se configura la notificación para un dispositivo Android y para finalizar se retorna como resultado el token (línea 104).

```

73   const registerForPushNotificationsAsync = async () => {
74     var token;
75     if (Constants.isDevice) {
76       const { status: existingStatus } = await Permissions.getAsync(
77         Permissions.NOTIFICATIONS
78       );
79       let finalStatus = existingStatus;
80       if (existingStatus !== "granted") {
81         const { status } = await Permissions.askAsync(
82           Permissions.NOTIFICATIONS
83         );
84         finalStatus = status;
85       }
86       if (finalStatus !== "granted") {
87         alert("No se pudo obtener el token de inserción para la notificación");
88         return;
89       }
90       token = (await Notifications.getExpoPushTokenAsync()).data;
91       setExpoPushToken(token);
92     } else {
93       alert("Debe ser un dispositivo físico");
94     }
95
96     if (Platform.OS === "android") {
97       Notifications.setNotificationChannelAsync("default", {
98         name: "default",
99         importance: Notifications.AndroidImportance.MAX,
100        vibrationPattern: [0, 250, 250, 250],
101        lightColor: "#FF231F7C",
102      });
103    }
104    return token;
105  };

```

Código 2.33. Función registerForPushNotificationsAsync

El Código 2.34 presenta la función updateToken, la cual guarda en la base de datos el token obtenido con la función mostrada en el Código 2.33, la línea 105 muestra la creación de la función updateToken, la cual recibe como parámetro de entrada el token del dispositivo móvil, obtiene el identificador del usuario activo, key (línea 109) y almacena en la base de datos en la ruta Usuarios/Administrador/key (línea 108 a línea 111), cabe recalcar que un procedimiento similar se lo realiza para los módulos Cliente y Empleado, con la ruta correspondiente a cada módulo así como con el Id de cada usuario.

```

105  updateToken = (token) => {
106    var key = firebase.auth().currentUser.uid;
107    try {
108      ref
109        .ref("Usuarios/Administrador/" + key)
110        .update({
111          Token: token,
112        })
113        .then((data) => {
114          console.log(data);
115        })
116        .catch((error) => {
117          console.log(error);
118        });
119    } catch (error) {
120      console.log(error);
121    }
122  };

```

Código 2.34. Función updateToken

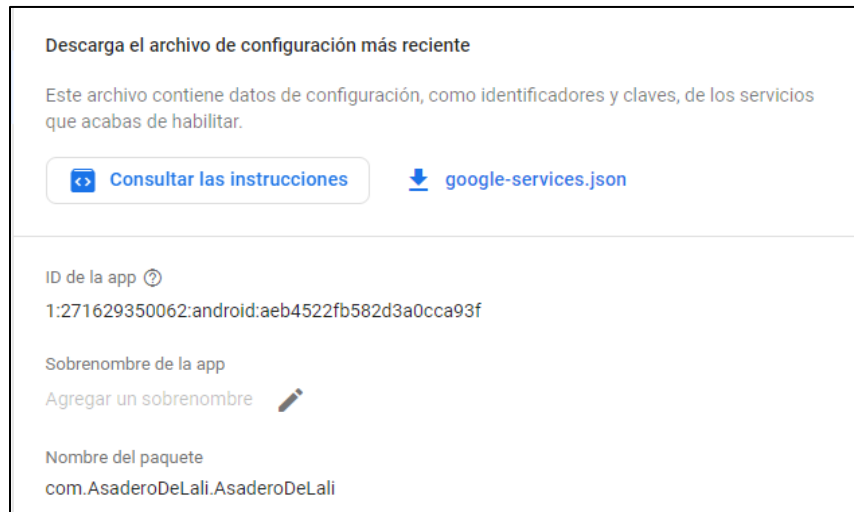


Figura 2.49. Descarga archivo de configuración google-services.json

Este archivo se lo agrega al directorio raíz del prototipo, tal y como muestra la Figura 2.50 en su parte izquierda, de la misma manera en el archivo app.json de la aplicación móvil se agrega la ruta relativa al archivo google-services.json tal y como muestra la Figura 2.51 en su parte derecha, en esta parte también se muestra el nombre con el cual la aplicación móvil fue registrada en Firebase.

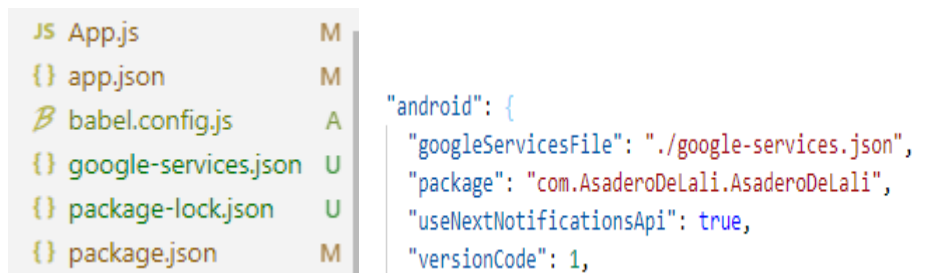


Figura 2.50. Ruta del archivo google-services.json

Para que una aplicación React Native generada con Expo pueda enviar notificaciones push, se debe cargar en los servidores de Expo la clave del servidor, en este caso la clave del servidor Firebase, la cual es presentada en la Figura 2.51. Esta clave del servidor se la almacena en los servidores de Expo mediante la ejecución del código:

```
expo push:android:upload --api-key <your-token-here> reemplazando <your token here> por la clave mostrada en la Figura 2.51.
```

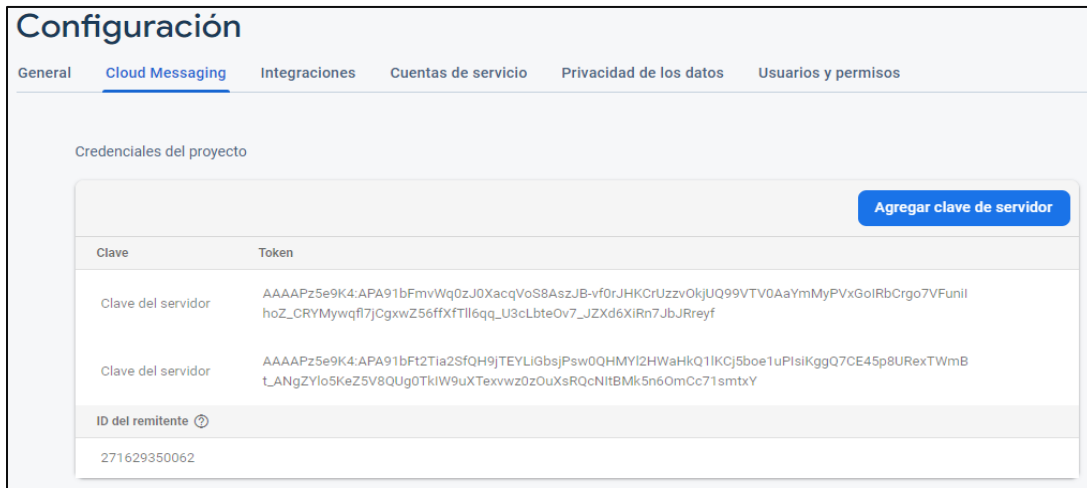


Figura 2.51. Obtención de clave del servidor de Firebase

El Código 2.35 presenta el método `sendPushNotifications` la cual permite enviar una notificación push al usuario, para lo cual recibe como parámetro de entrada el token, el cual será el identificador del dispositivo móvil del usuario al cual va dirigido la notificación, entre las líneas 204 y 208 se implementa el cuerpo de la notificación, la línea 211 hace una petición a la API de Expo, entre las líneas 212 y 218 se configura el cuerpo de esta petición.

El Código 2.36 muestra un fragmento del código para la implementación de la pantalla de recuperación de contraseña. Para lo cual se ingresa el correo con el cual el usuario se registró en la aplicación.

```

202  sendPushNotification = async (Token) => {
203    const message = {
204      to: Token,
205      sound: "default",
206      title: "El Asadero de Lali",
207      body: "Nuevo pedido realizado !",
208      data: { data: "goes here" },
209    };
210
211    await fetch("https://exp.host/--/api/v2/push/send", {
212      method: "POST",
213      headers: {
214        Accept: "application/json",
215        "Accept-encoding": "gzip, deflate",
216        "Content-Type": "application/json",
217      },
218      body: JSON.stringify(message),
219    });
220  };

```

Código 2.35. Función `sendPushNotification`

```

45 <View style={styles.container}>
46   <KeyboardAwareScrollView keyboardShouldPersistTaps="always">
47     <Text style={[styles.title, { fontSize: 25 }]}>
48       ¿Olvidaste tu contraseña?
49     </Text>
50     <Text style={[styles.title, { fontWeight: "normal", fontSize: 18 }]}>
51       Ingresa el correo electrónico con el que te registraste para recuperar
52       tu contraseña. Es posible que debas revisar en tu bandeja de correo no
53       deseado.
54     </Text>
55     <View style={{ height: 50 }} />
56     <View style={styles.TextUser}>
57       <MaterialCommunityIcons
58         style={{
59           marginTop: 10,
60           paddingBottom: 10,
61         }}
62         name="email-outline"
63         size={28}
64         color="black"
65       />
66     <View style={{ width: 10 }} />
67     <TextInput
68       placeholder="Tú correo electrónico"
69       style={styles.textInput}
70       onChangeText={setCorreo}
71       value={correo}
72     />

```

Código 2.36. Fragmento de código recuperación de contraseña

El Código 2.37 presenta la función `onSubmit` la cual, verifica que la casilla correspondiente al correo no este vacía (línea 47), posteriormente, invoca a la función `sendPasswordresetEmail` (línea 26), la cual recibe como parámetro de entrada el correo del usuario, y retorna un mensaje al usuario en el caso de que la acción haya sido completada de forma exitosa (línea 34).

```

25   const onSubmit = async () => {
26     if (correo === "") {
27       alert("El correo no puede estar vacío");
28     } else {
29       setLoading(true);
30       await firebase
31         .auth()
32         .sendPasswordResetEmail(correo)
33         .then(function (user) {
34           alert("Revisa tu correo electrónico...");
35           setCorreo("");
36           setLoading(false);
37         })
38         .catch(function (e) {
39           console.log(e);
40         });
41     }
42   };

```

Código 2.37. Función `onSubmit`

La Figura 2.52 presenta la configuración en Firebase de los datos e indicaciones que serán enviados al correo del usuario una vez haya solicitado la recuperación de su contraseña.

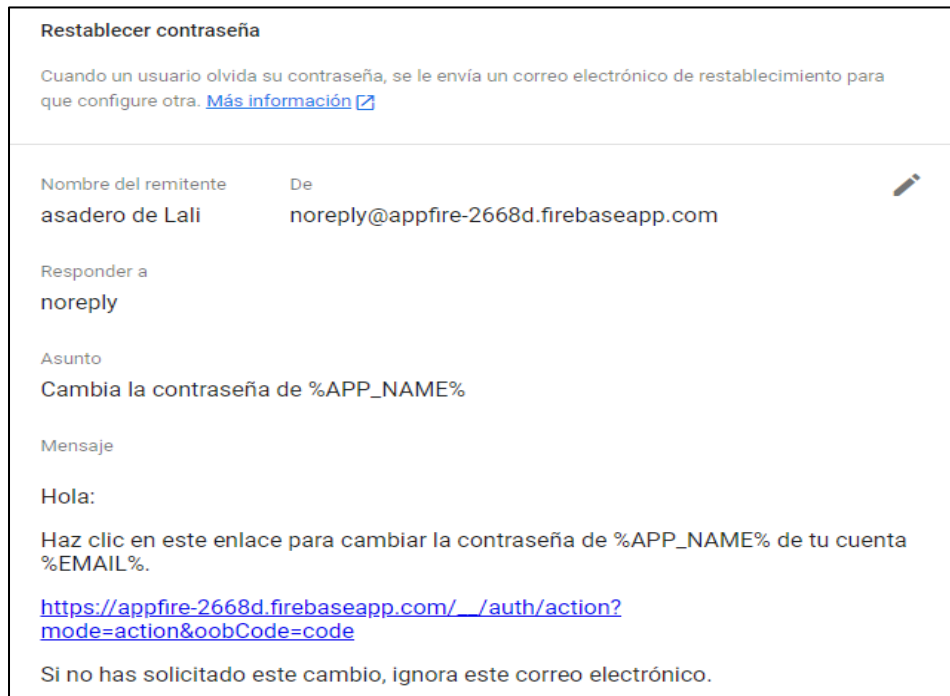


Figura 2.52. Configuración correo de recuperación de contraseña

La Figura 2.53 muestra la pantalla de recuperación de contraseña.



Figura 2.53. Recuperación de contraseña

Adicionalmente el usuario administrador podrá visualizar un reporte de los pedidos realizados por los clientes, estos reportes podrán ser filtrados por fechas y constará de la cantidad de ordenes emitidas en la fecha elegida, así como también, el valor total de las ventas. El Código 2.38 muestra el fragmento de código para la implementación de la función que permite realizar el filtrado de los datos por fecha.


```

139 const buscar = () => {
140   let unmounted = true;
141   var entregado = [];
142   var TotalPedidos = 0;
143   try {
144     ref
145     .ref("pedidoEmpleado/")
146     .orderByChild("Fecha")
147     .equalTo(aux.toString())
148     .limitToLast(30)
149     .on("value", (snapshot) => {
150       var datosEntregado = snapshot.val();
151       if (unmounted) {
152         for (var key in datosEntregado) {
153           {
154             TotalPedidos = TotalPedidos + Number(datosEntregado[key].Total);
155           }
156         }
157         const obj = {
158           Id: datosEntregado[key].Id,
159           Nombre: datosEntregado[key].Nombre,
160           Apellido: datosEntregado[key].Apellido,
161           Estado: datosEntregado[key].Estado,
162           Fecha: datosEntregado[key].Fecha,
163           Telefono: datosEntregado[key].Telefono,
164           Total: datosEntregado[key].Total,
165           Email: datosEntregado[key].email,
166           UserId: datosEntregado[key].UserId,
167           Direccion: datosEntregado[key].Direccion,
168           Token: datosEntregado[key].Token,
169         };
170         entregado.push(obj);
171       }
172       setEntregado(entregado);
173       setTotalPedidos(TotalPedidos.toFixed(2));

```

Código 2.38. Fragmento de código filtro por fecha

La Figura 2.54 muestra la pantalla con los datos de los pedidos hechos por los clientes, así como también, el número de ventas realizadas.



Figura 2.54. Reporte de pedidos

3. RESULTADOS Y DISCUSIÓN

En este apartado, se expone los resultados de las pruebas de funcionamiento a los cuales fue sometido el prototipo para su validación. Estas pruebas de funcionamiento se las realiza para comprobar que se hayan cumplido las historias de usuario. Como primer paso se muestra la actualización del tablero Kanban, expuesto en la Tabla 3.1, luego, se procede a realizar las pruebas de funcionamiento a cada uno de los módulos con los que cuenta el prototipo.

Finalmente, se llevó a cabo una entrevista a 10 clientes del restaurante, así como al administrador y empleados del mismo, a los cuales se les entregó una versión instalable de la aplicación con la finalidad de que el usuario final verifique la operatividad del prototipo, recoger recomendaciones y corregir errores en el caso que los haya y que no hayan sido detectados en las pruebas previas.

Tabla 3.1. Tablero Kanban actualización

Tareas	Tareas en proceso	Tareas realizadas
	Pruebas de funcionamiento módulo Administrador.	Toma de Requerimientos.
	Pruebas de funcionamiento módulo Cliente.	Realización de entrevistas al administrador y a diez clientes del restaurante.
	Pruebas de funcionamiento módulo Empleado.	Identificación de historias de usuario.
	Pruebas de funcionamiento con todos los módulos.	Identificación de requerimientos funcionales.
	Corrección de errores.	Identificación de requerimientos no funcionales.
		Determinación de los módulos del prototipo.
		Diseño de las vistas de la aplicación.
		Realización de diagramas de casos de uso.
		Realización de diagrama de actividades.

Tabla 3.1. Tablero Kanban actualización

Tareas	Tareas en proceso	Tareas realizadas
		Diagrama de estructura.
		Diseño de la estructura del proyecto.
		Diseño base de datos.
		Instalación de Node.js.
		Instalación de Yarn.
		Instalación Expo CLI.
		Instalación de Visual Studio Code.
		Instalación cliente Expo.
		Crear la base de datos en Firebase.
		Codificación pantallas del módulo administrador.
		Codificación pantallas del módulo Cliente.
		Codificación pantallas del módulo empleado.
		Codificación de navegación entre pantallas.
		Instalación de Firebase en la aplicación móvil.
		Codificación de funcionalidades del prototipo.

3.1. PRUEBAS DE FUNCIONAMIENTO

Las pruebas de funcionamiento son realizadas a cada uno de los módulos con los que cuenta el prototipo, y se empieza con el módulo administrador tal y como es mostrado en el tablero Kanban.

3.1.1. PRUEBAS DE FUNCIONAMIENTO MÓDULO ADMINISTRADOR

Para validar que el módulo administrador funcione de forma correcta, se realizan las siguientes pruebas.

3.1.1.1. Inicio de sesión

Para validar la funcionalidad de inicio de sesión en el módulo administrador, el usuario debe ingresar su correo electrónico, así como una contraseña, tal como lo muestra la Figura 3.1.

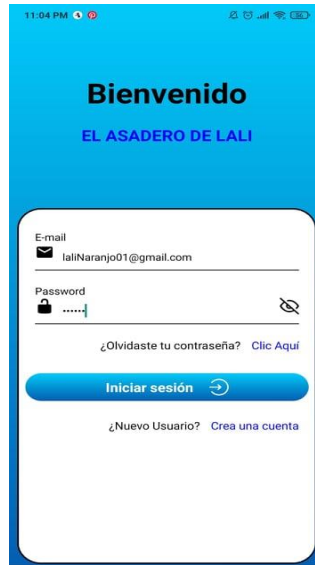


Figura 3.1. Inicio de sesión administrador

Posteriormente se realizan pruebas en las cuales los datos son incorrectos, ingresando una contraseña incorrecta, ingresado el correo incompleto o también dejando los valores en blanco, estos mensajes de error son mostrados en la Figura 3.2.

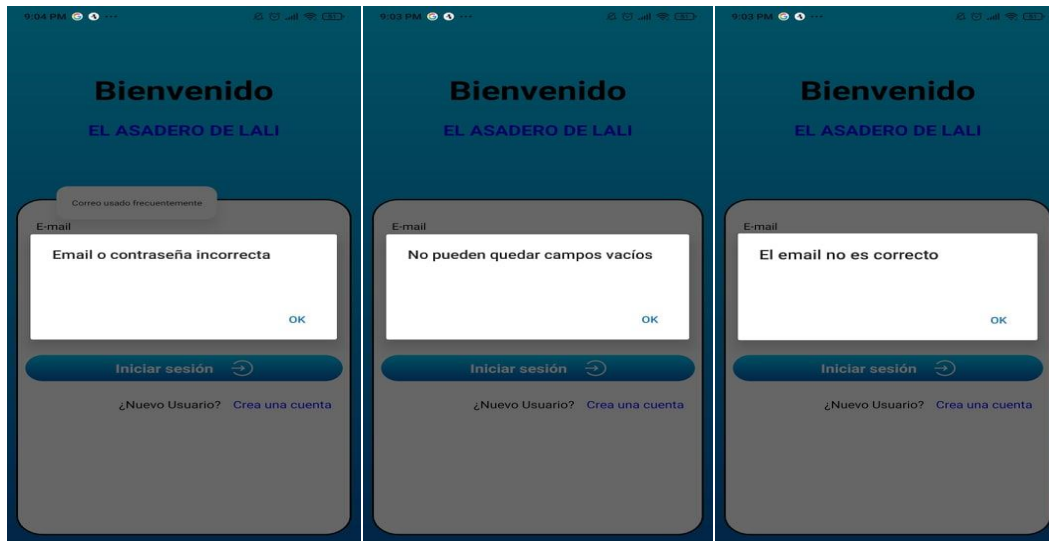


Figura 3.2. Mensajes de error con datos incorrectos

Si los datos son correctos entonces se mostrará la pantalla principal del usuario administrador la cual es presentada en la Figura 3.3.



Figura 3.3. Pantalla principal administrador

3.1.1.2. Agregar un nuevo plato o promoción al menú

El usuario administrador es el encargado de ingresar un nuevo plato al menú del restaurante, así como también es el encargado de ingresar nuevas promociones; para esta prueba se agrega un nuevo plato al menú del restaurante, como se muestra en la Figura 3.4. Durante este proceso la aplicación solicitará los permisos para acceder a la información de la galería del dispositivo móvil.



Figura 3.4. Agregar nuevo plato

Al guardar los datos del nuevo plato se verifica en la base de datos, así como en storage de Firebase que los datos se hayan guardado, estos datos se guardan en el nodo Menu, la Figura 3.5 presenta los datos del nuevo plato que han sido guardados en Firebase.



Figura 3.5. Ingreso de un nuevo plato

Si se ingresan datos erróneos al registrar un nuevo plato como dejar campos vacíos, ingresar la coma como separador de decimales y también si se ingresa más de dos decimales. La Figura 3.6 expone los mensajes de error que son mostrados en pantalla al ingresar datos erróneos.

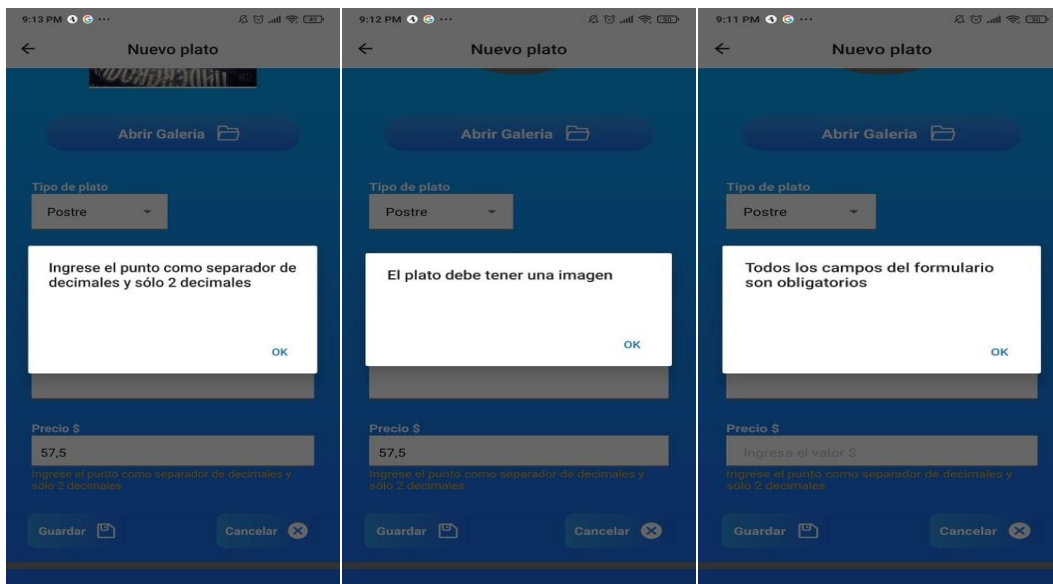


Figura 3.6. Mensajes de error nuevo plato

Para agregar una nueva promoción al sistema, se sigue el mismo procedimiento que al agregar un nuevo plato, la Figura 3.7 muestra el ingreso de una nueva promoción.



Figura 3.7. Nueva promoción

Esta información se guarda en el nodo promociones de la base de datos, tal como lo muestra la Figura 3.8.



Figura 3.8. Ingreso nueva promoción

3.1.1.3. Editar y eliminar un plato o promoción del menú

El usuario administrador es el encargado de editar o eliminar platos del menú y también promociones. Para esta prueba, el prototipo muestra los platos presentes en el menú, junto con la opción de editar o eliminar (Figura 3.9), el administrador elige el plato que desea editar, se despliega otra pantalla en la cual, se observa la información del plato, y se procede a cambiar la imagen y la descripción del plato elegido, el proceso es mostrado en

la Figura 3.10, una vez los datos se actualizan, se muestra estos datos actualizados en la pantalla del menú del administrador.

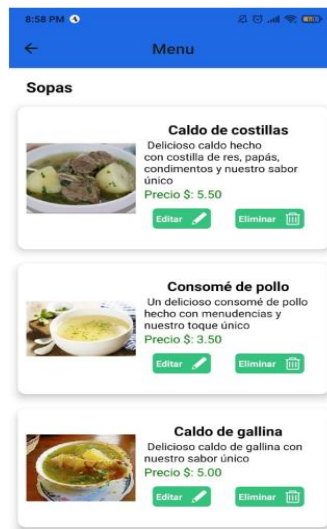


Figura 3.9. Menú módulo administrador

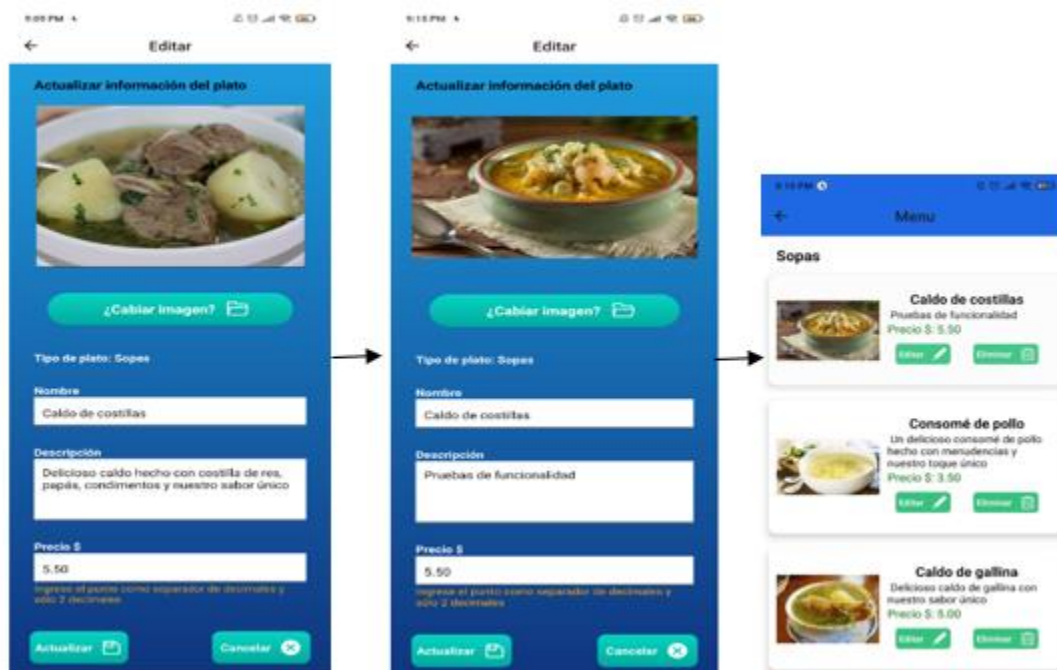


Figura 3.10. Editar plato del menú

Para eliminar un plato del menú, se presenta un mensaje en el cual se pide confirmación para eliminar el plato, la Figura 3.11 muestra dicho mensaje, al confirmar la eliminación del plato se regresa a la pantalla del menú.



Figura 3.11. Mensaje de eliminación del plato

Un proceso similar al mostrado anteriormente se le lleva acabo para editar o eliminar los datos de las promociones del menú.

3.1.1.4. Agregar nuevo empleado al sistema

El usuario administrador es el encargado de ingresar al sistema a nuevos empleados, por lo que, en esta prueba se ingresa la información de un nuevo empleado, esta información se almacena en la base de datos en el nodo empleados, la Figura 3.12 y 3.12 muestran los datos a ingresar, así como, los datos ingresados en la base de datos.

A screenshot of a mobile application interface for registering a new employee. The screen has a blue header with the title 'Empleado' and a back arrow. Below the header, the main content area has a blue background with the title 'REGISTRAR EMPLEADO' and the subtitle 'Ingresar información'. The form contains several input fields: 'Nombre' with the value 'Marcelo', 'Apellido' with 'Garzón', 'Número de teléfono' with '0989923855', 'Cargo' with a dropdown menu showing 'Mesero', 'Dirección' with 'condado, intersección Av. Prensa y Av. Occiden', 'E-mail' with 'marcelo@gmail.com', and 'Contraseña' with a masked password '.....'. At the bottom, there is a blue button labeled 'REGISTRAR EMPLEADO' with a person icon. A small note at the bottom left says '* Todos los campos son obligatorios'.

Figura 3.12. Ingreso de un nuevo empleado

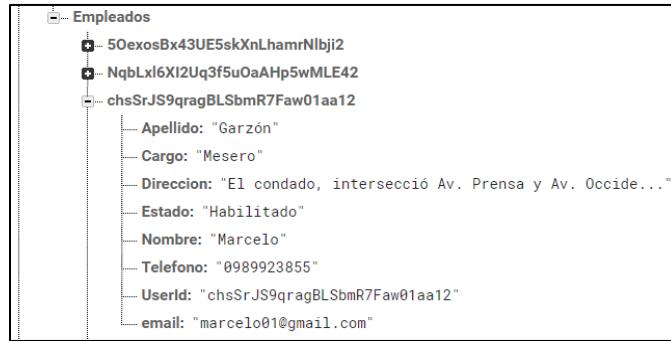


Figura 3.13. Registro de nuevo empleado

En esta prueba se ingresa información errónea, así como también se dejan algunos campos en blanco, esto con la finalidad de evaluar algunos mensajes de error que la aplicación muestra en pantalla en este apartado del prototipo. La Figura 3.14 muestra algunos mensajes de error. La parte (a) de la Figura 3.14 muestra el mensaje de error al ingresar un correo ya existente en el sistema, la parte (b) indica el mensaje de error al dejar campos vacíos y la parte (c) indica el mensaje de error al ingresar una contraseña con un número de caracteres menor a 6.

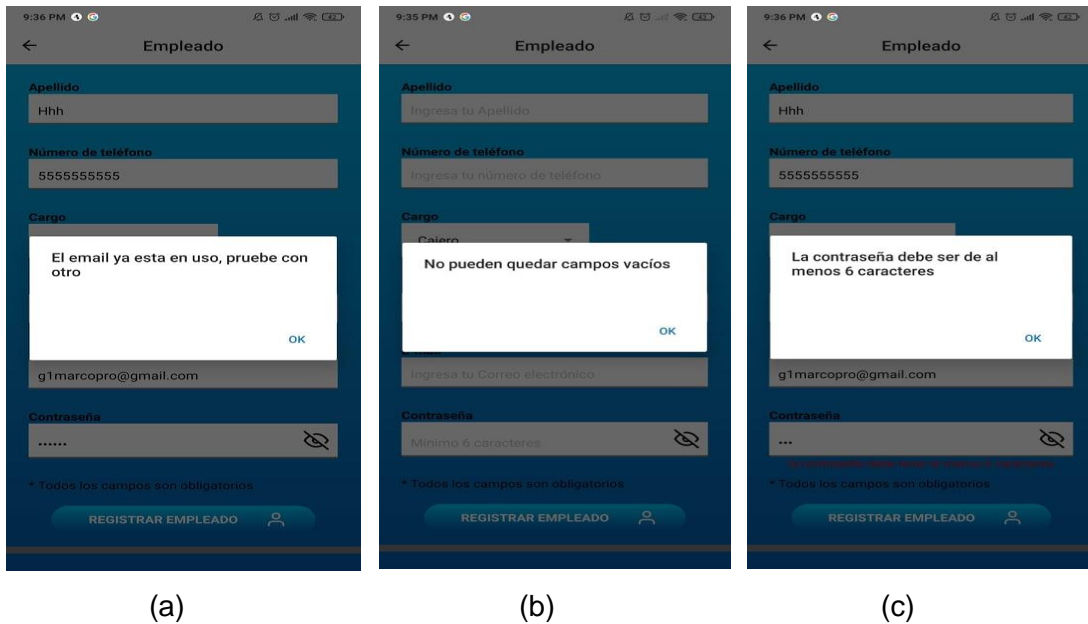


Figura 3.14. Ingreso de nuevo empleado con información incorrecta

3.1.1.5. Editar y eliminar información del empleado

El usuario administrador también puede actualizar los datos de los empleados, así como también, eliminarlo del sistema, también tiene la potestad de eliminar del sistema al usuario cliente, en esta prueba se edita los datos del empleado ingresado anteriormente y se

procede a eliminar a un cliente de prueba. La Figura 3.15 muestra la actualización del nombre y número de teléfono del empleado en la aplicación, y la actualización en la base de datos.

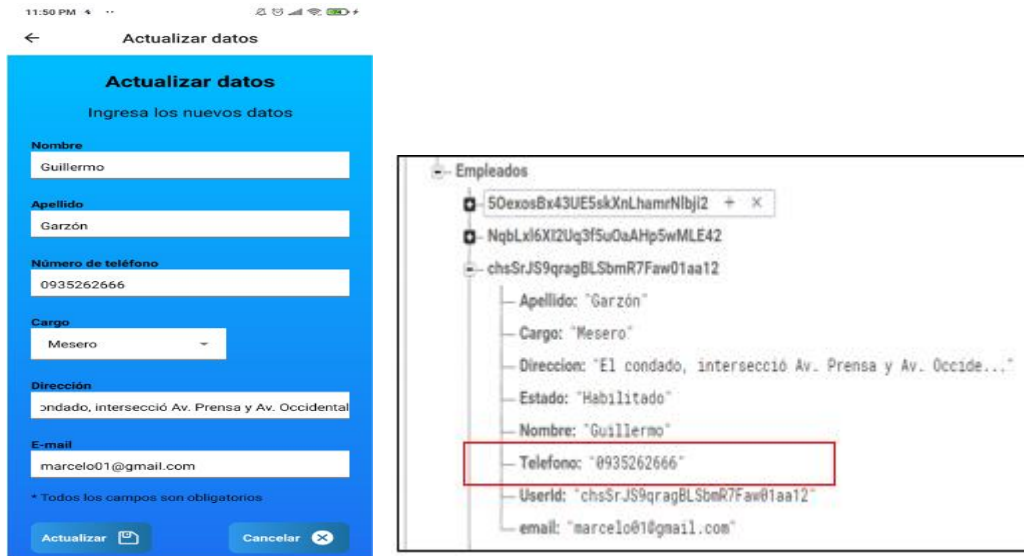


Figura 3.15. Actualización de nombre y número de teléfono de empleado

De la misma manera se realiza la prueba para eliminar al usuario cliente del sistema, la Figura 3.16 presenta la pantalla que permite visualizar los datos de los clientes y eliminarlos del sistema.

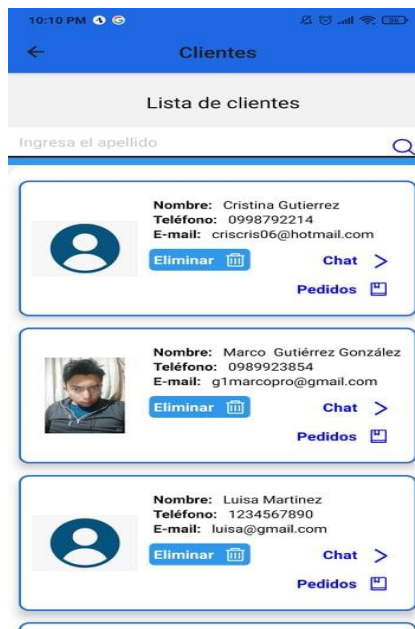


Figura 3.16. Eliminación de cuentas de clientes

3.1.1.6. Actualizar perfil

El usuario administrador puede actualizar sus datos ingresados en el sistema. En esta sección el usuario puede agregar o actualizar su foto de perfil, así como sus datos y cambiar su contraseña. La Figura 3.17 muestra la pantalla que permite actualizar los datos del usuario administrador.



Figura 3.17. Pantalla actualización de perfil

3.1.1.7. Reporte de pedidos

El usuario administrador puede ver un reporte general de los pedidos hechos por los clientes, dentro de este reporte se visualiza el número de pedidos hechos en la fecha seleccionada, y también se visualiza el valor total de los pedidos hechos en esa fecha. La Figura 3.18 muestra los pedidos, así como la elección de la fecha.

3.1.2. PRUEBAS DE FUNCIONAMIENTO MÓDULO CLIENTE

3.1.2.1. Registrar usuario

El usuario cliente debe registrarse en el sistema para acceder al mismo, por lo que en esta prueba se registra un nuevo usuario en el sistema, con el objetivo de constatar el correcto funcionamiento de esta opción. La Figura 3.19 muestra la pantalla para el registro de un nuevo cliente.

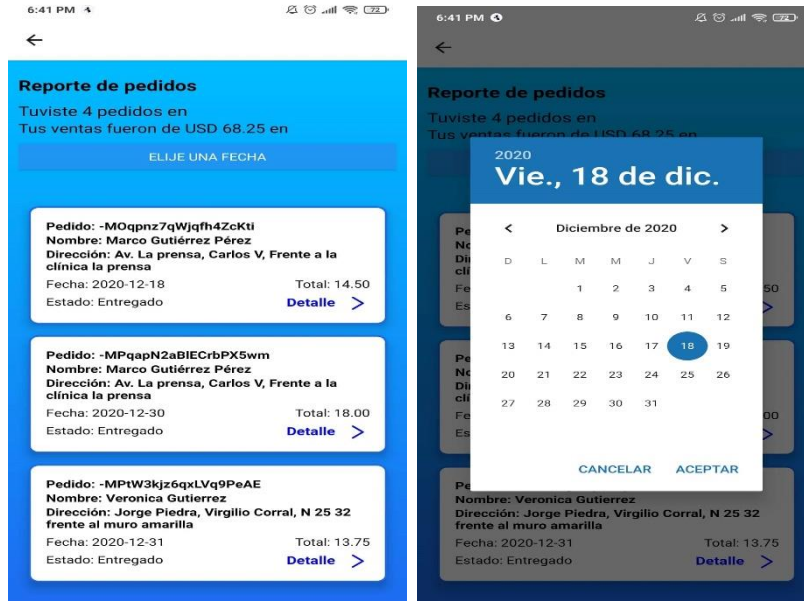


Figura 3.18. Reporte de pedidos

Una vez ingresada la información el sistema pide que se ingrese una dirección, esta dirección será hacia la cual serán enviados los pedidos, el sistema también permite tener más de una dirección de entrega. La Figura 3.20 indica la pantalla en la cual el cliente ingresa la información correspondiente a la dirección de entrega, cabe recalcar que la aplicación pedirá los permisos de ubicación del dispositivo para funcionar de forma correcta.



Figura 3.19. Pantalla registro de nuevo cliente



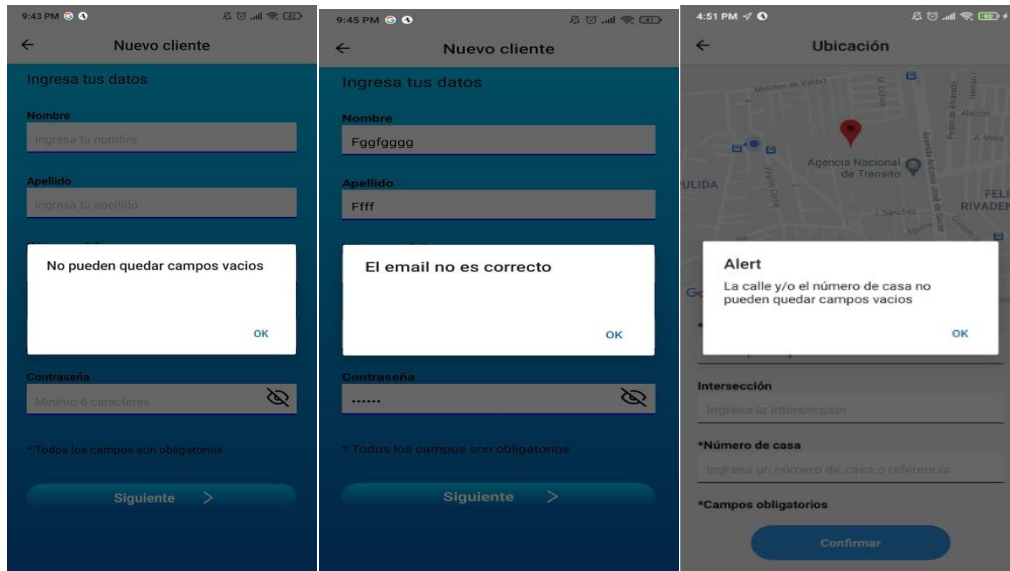
Figura 3.20. Mapa de ubicación

La Figura 3.21 presenta la información del cliente ingresado en la base de datos.



Figura 3.21. Registro de información nuevo cliente

En esta prueba se ingresan valores erróneos con la finalidad de indicar los posibles mensajes de error que el prototipo despliega. La Figura 3.22 (a) muestra el mensaje al dejar campos en blanco, la parte (b) presenta el mensaje de error al ingresar un correo no válido, y la parte (c) indica el mensaje de error al no ingresar la dirección de entrega.

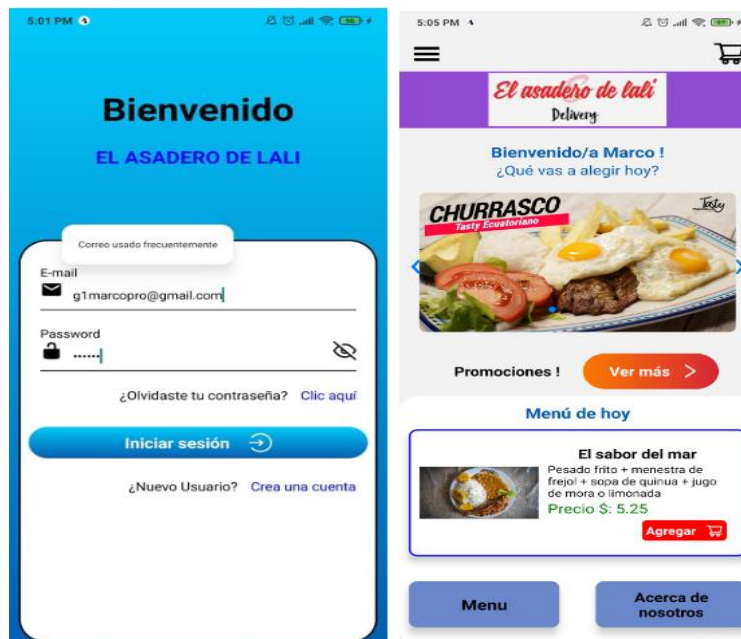


(a) (b) (c)

Figura 3.22. Mensajes de alerta

3.1.2.2. Inicio de sesión

Para esta prueba se usa al usuario de prueba registrado en el paso anterior, la Figura 3.23 (a) muestra la pantalla de inicio de sesión con los datos de correo y contraseña del usuario creado en el paso anterior, una vez el inicio de sesión es exitoso se muestra la pantalla principal del módulo cliente, Figura 3.23 (b).



(a) (b)

Figura 3.23. Inicio de sesión cliente

Los mensajes de advertencia al ingresar ya sea un correo incorrecto o una contraseña incorrecta son similares a los presentados en la Figura 3.2.

3.1.2.3. Visualizar menú

En esta prueba el usuario cliente visualiza los distintos platos con los que cuenta el menú del restaurante, además de las promociones presentes en la aplicación, se eligen como prueba algunos productos los cuales se guardan en el carrito de compras hasta la confirmación del pedido. La Figura 3.24 muestra algunos de los platos presentes en la aplicación, así como las promociones.

Se procede a elegir algunos platos del menú, los cuales son guardados en el carrito de compras hasta que se confirme el envío del pedido, la Figura 3.25 muestra el carrito de compras con los productos seleccionados, el prototipo permite eliminar del carrito los productos, se muestra también los productos del carrito almacenados en la base de datos.

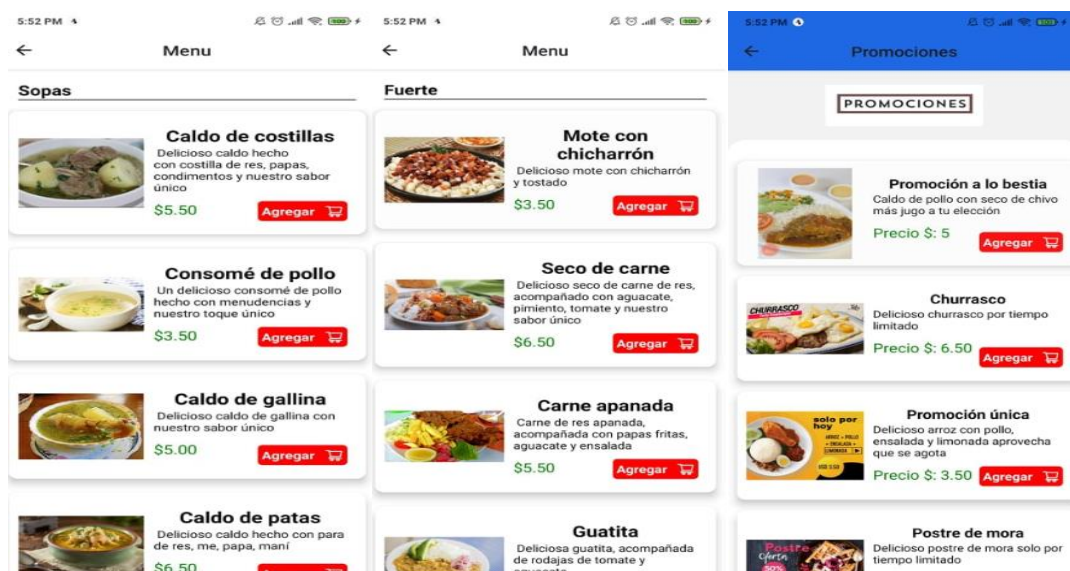


Figura 3.24. Platos del menú

3.1.2.4. Actualizar perfil

Para esta prueba el usuario cliente actualiza sus datos, se cambiará el nombre y se agregará una foto de perfil, la Figura 2.26 presenta la pantalla de actualización de la información del cliente, se presenta también la actualización de los datos en la base de datos.

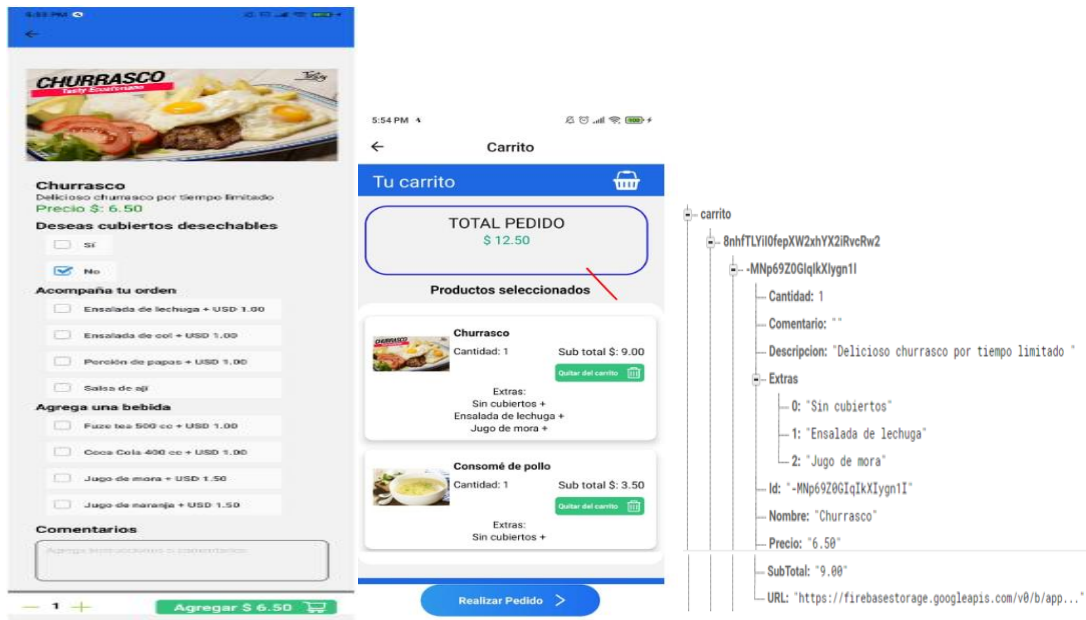


Figura 3.25. Carrito de compras

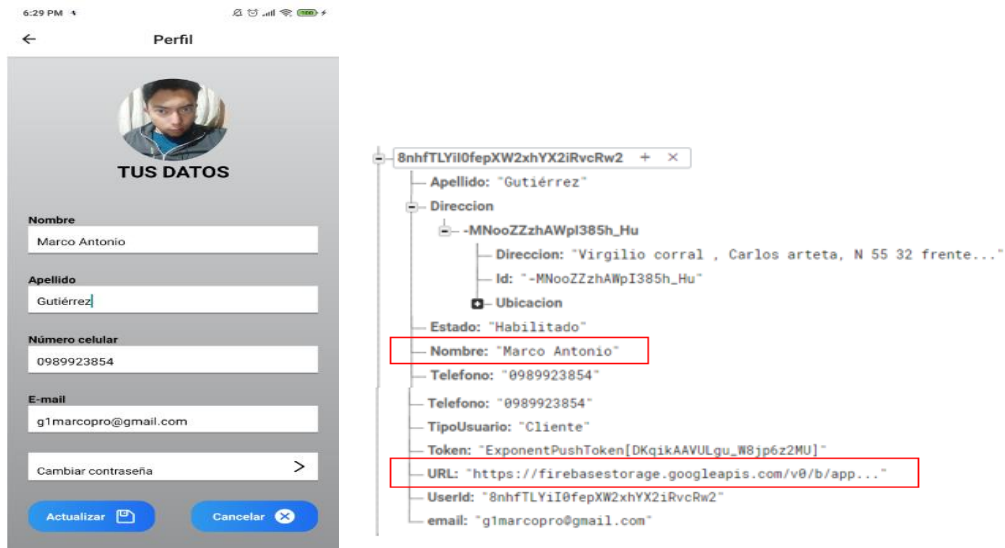
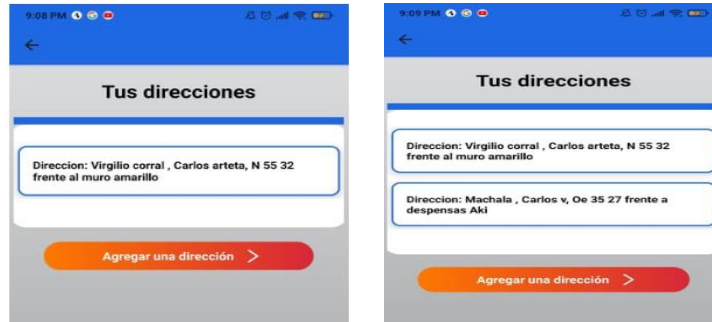


Figura 3.26. Actualizar datos del cliente

3.1.2.5. Agregar dirección de entrega

El prototipo admite el ingreso de varias direcciones de entrega, por lo que para esta prueba se ingresa una nueva dirección de entrega, para agregar una nueva dirección se usa una pantalla similar a la usada en la Figura 3.20, la Figura 3.27 (a) muestra la dirección que el cliente de prueba ingreso al momento del registro, la Figura 3.27 (b) muestra una segunda dirección de entrega ingresada al sistema.



(a)

(b)

```

Direccion
├── -MNooZZzhAWpI385h_Hu
│   ├── Dirección: "Virgilio corral , Carlos arteta, N 55 32 frente..."
│   ├── Id: "-MNooZZzhAWpI385h_Hu"
│   └── Ubicacion
│       ├── latitude: -0.1365262
│       ├── latitudeDelta: 0.01
│       ├── longitude: -78.5062064
│       └── longitudeDelta: 0.01
└── -MNpnwe2Ml_u2t0J2fod
    ├── Dirección: "Machala , Carlos v, Oe 35 27 frente a despensas..."
    ├── Id: "-MNpnwe2Ml_u2t0J2fod"
    └── Ubicacion

```

(c)

Figura 3.27. Ingreso de una nueva dirección de entrega

El cliente podrá elegir la dirección de entrega del pedido al momento de confirma el envío.

3.1.2.6. Información del restaurante

En esta prueba se observa información básica acerca del restaurante, como son sus números de contactos, su ubicación y su correo electrónico. La Figura 3.28 muestra la pantalla de información del restaurante El Asadero de Lali.



Figura 3.28. Pantalla de información

3.1.3. PRUEBAS DE FUNCIONAMIENTO MÓDULO EMPLEADO

3.1.3.1. Inicio de sesión

Para realizar esta prueba se ingresa el correo y la contraseña de un empleado ya ingresado al sistema, la Figura 3.29 muestra los datos de correo y contraseña ingresados para el inicio de sesión, así como también, la pantalla principal del módulo empleado.



Figura 3.29. Inicio de sesión empleado

Los mensajes de advertencia al ingresar ya sea un correo incorrecto o una contraseña incorrecta son similares a los presentados en la Figura 3.2.

3.1.3.2. Actualizar perfil

Para esta prueba se actualiza el nombre del empleado, así como su foto de perfil, la Figura 3.30 presenta la pantalla de actualización de datos, así como los datos actualizados en la base de datos.

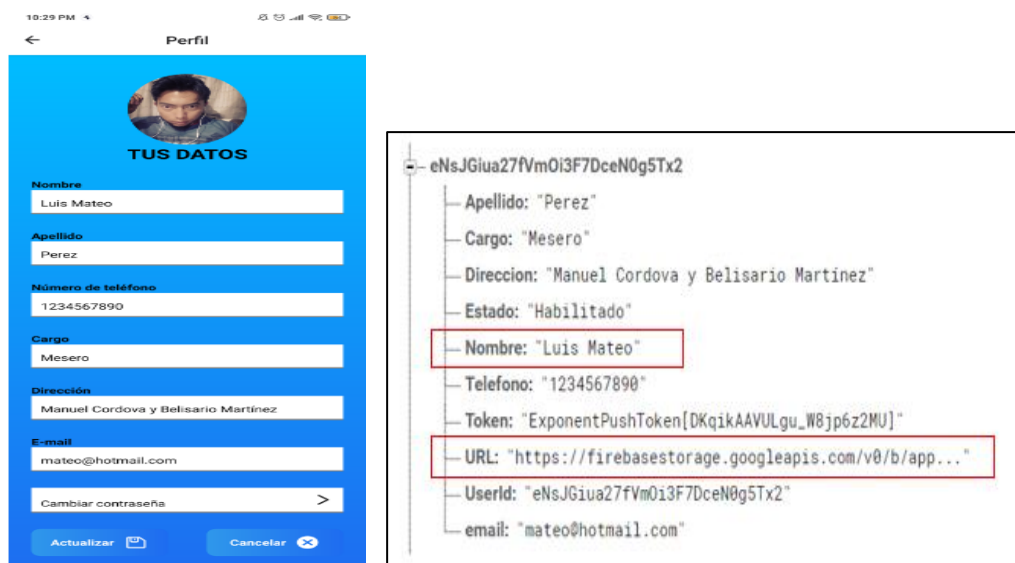


Figura 3.30. Actualización datos empleado

3.1.3.3. Pedidos pendientes

En la sección de pedidos pendientes el empleado encuentra los pedidos nuevos que han llegado al restaurante, en esta prueba se muestran los pedidos, así como también el detalle de los mismos, la Figura 3.31 muestra la lista de pedidos pendientes y el detalle de un pedido.

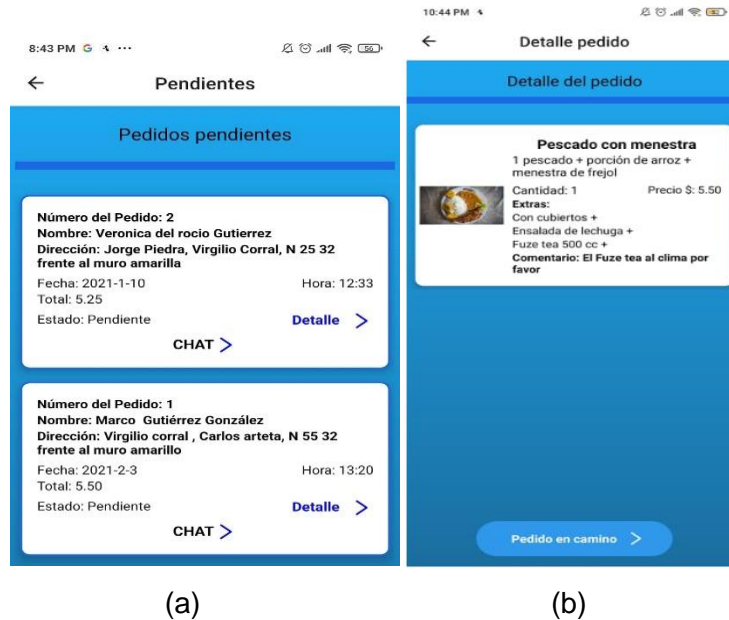


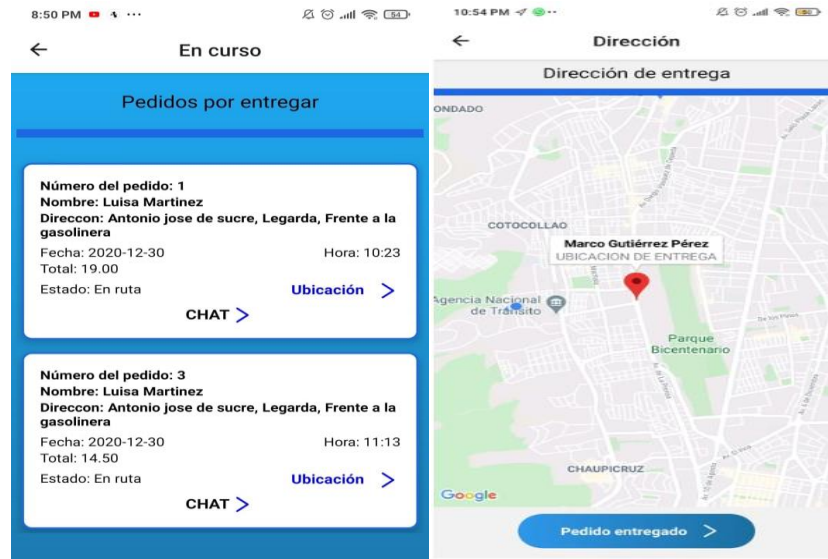
Figura 3.31. (a) Lista de pedidos pendientes, (b) detalle del pedido

3.1.3.4. Pedidos en ruta

En la sección pedidos en ruta el empleado encuentra los pedidos que han sido enviados al cliente, así como la dirección a la cual deben ser entregados, en esta prueba se observa la lista de pedidos en ruta, así como el mapa de la dirección de entrega del pedido. La Figura 3.32 se muestra la lista de pedidos en camino y el mapa de la dirección de entrega.

3.1.3.5. Pedidos entregados

En la sección de pedidos entregados el empleado puede ver una lista de los pedidos que ha sido entregados en su destino, en esta prueba se observa en la Figura 3.33 una lista de los pedidos entregados.



(a)

(b)

Figura 3.32. (a) Lista de pedidos en ruta, (b) Mapa de lugar de entrega



Figura 3.33. Pedidos entregados

3.1.4. PRUEBAS DE INTEGRACIÓN

En este apartado se presentan las pruebas que permiten enlazar a los tres módulos, entre ellos se encuentran las notificaciones y la comunicación del cliente con el restaurante, es decir el chat.

3.1.4.1. Notificaciones

Tanto el cliente como los empleados del restaurante necesitan una notificación que les permita conocer si su pedido está en camino o saber de la llegada de un nuevo pedido respectivamente, por lo que en esta prueba se realiza un pedido de parte del cliente, la Figura 3.34 muestra la confirmación del pedido realizado en la sección 3.1.2.3 y la notificación que es enviada a los empleados registrados en el sistema.

El módulo empleado, en la sección pedidos pendientes se observa el nuevo pedido y su detalle, una vez armado el pedido se cambia su estado de pendiente a en ruta y de forma automática el usuario cliente recibe una notificación en la cual es informado de que su pedido está en camino, la Figura 3.35 muestra el pedido en la lista de pedidos pendientes y la notificación que es enviada al usuario cliente.

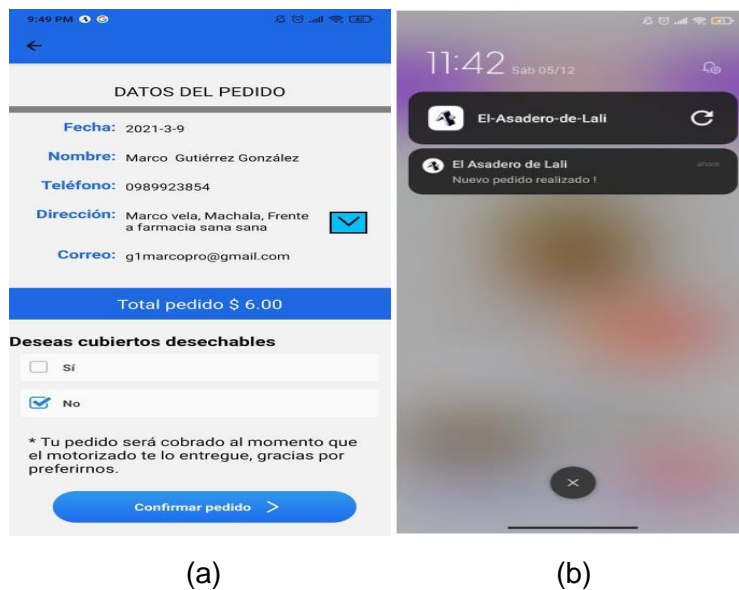


Figura 3.34. (a) Confirmación del pedido, (b) Notificación de nuevo pedido

3.1.4.2.Chat

Esta prueba tiene la finalidad de verificar que el cliente pueda comunicarse con el restaurante a través de un chat. Para esta prueba se envía un mensaje desde el cliente, este mensaje es respondido por un empleado, con lo que la comunicación entre el cliente y el restaurante esta verificada. La Figura 3.36 (a) muestra el mensaje enviado desde el usuario cliente y la Figura 3.36 (b) muestra la respuesta enviada desde el usuario empleado.

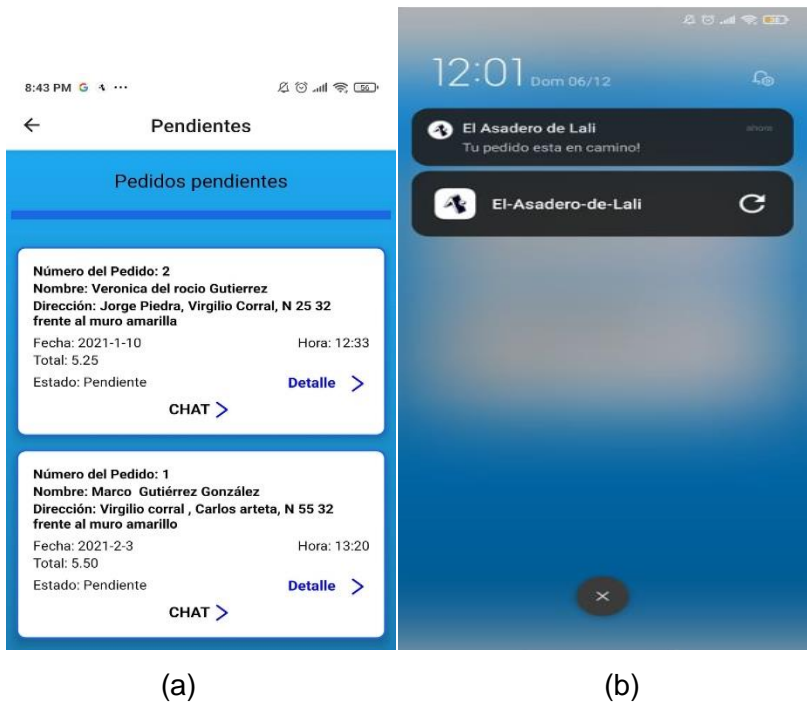


Figura 3.35. (a) Pedidos pendientes, (b) Notificación de pedido en camino

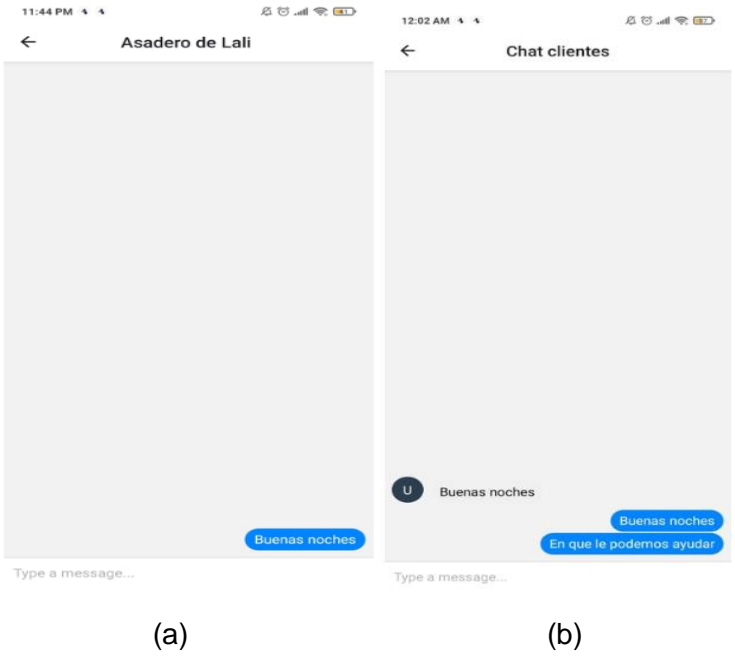


Figura 3.36. Chat

La Figura 3.37 muestra los datos del chat almacenados en la base de datos.



Figura 3.37. Datos del Chat

3.1.5. RECUPERAR CONTRASEÑA

La sección de recuperar contraseña es común para los tres módulos, por lo que en esta prueba se usa los datos de un usuario cliente para recuperar la contraseña, la Figura 3.38 presenta la pantalla de recuperación de contraseña junto con el mensaje que se presenta al momento de enviar la solicitud de recuperar contraseña.

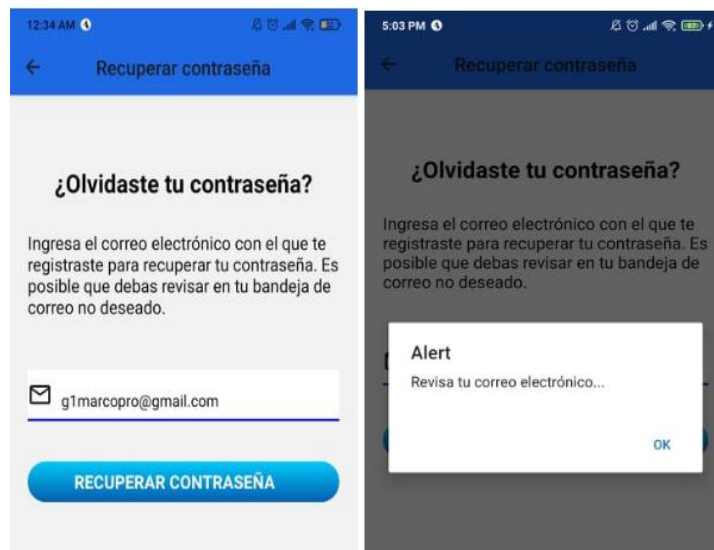


Figura 3.38. Recuperar contraseña

Una vez solicitada la recuperación de contraseña, se envía al correo electrónico un enlace en el cual se debe introducir la nueva contraseña, la Figura 3.39 indica el proceso para recuperar la contraseña.

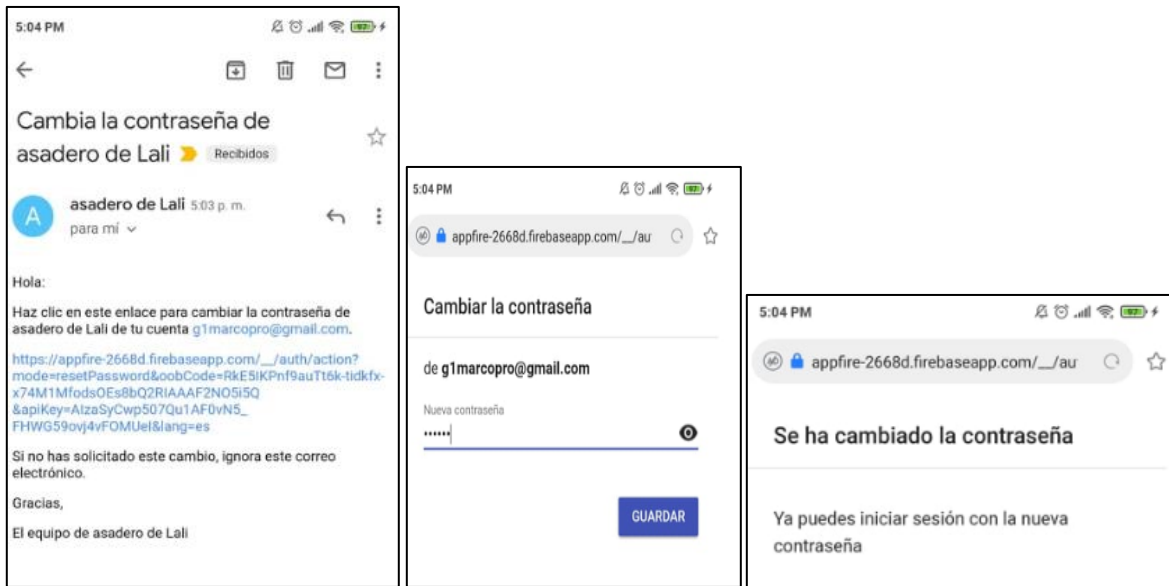


Figura 3.39. Cambio de contraseña

3.2. PRUEBAS DE VALIDACIÓN

Para llevar a cabo las pruebas de validación, se hizo una entrevista a diez clientes del restaurante, así como al administrador del restaurante El asadero de Lali y a cuatro de sus empleados.

Para presentar el resultado de las entrevistas, se elabora una serie de preguntas a las cuales los entrevistados contestan sí o no, la Tabla 3.2 presenta los resultados de las entrevistas a los diez clientes. El ANEXO C presenta el modelo de entrevista de validación, así como los resultados obtenidos.

Tabla 3.2. Resultados entrevistas clientes

No	Pregunta	Respuesta	
		Si	No
1	¿Pudo crear su usuario en la aplicación?	10	0
2	¿Pudo iniciar sesión?	10	0
3	¿La aplicación te permitió editar tus datos de perfil?	10	0
4	¿La aplicación en algún momento te hizo una petición de acceder a la galería de tu teléfono?	8	2

Tabla 3.2. Resultados entrevistas clientes

5	¿La aplicación te permitió ingresar tu lugar de entrega a través de un mapa?	10	0
6	¿Pudiste observar los platos del menú en la aplicación?	10	0
7	¿La aplicación fue sencilla y clara manejar?	8	2
8	¿Recibiste una notificación al momento de realizar tu pedido?	7	3
9	¿Te pudiste comunicar con los empleados del restaurante a través de la aplicación?	8	2
10	¿Recomendarías la aplicación?	9	1

Como se aprecia en los resultados de la Tabla 3.2 la mayoría de los clientes responden de manera afirmativa a las preguntas. La Figura 3.40 presenta el resultado de la entrevista de validación para los clientes.

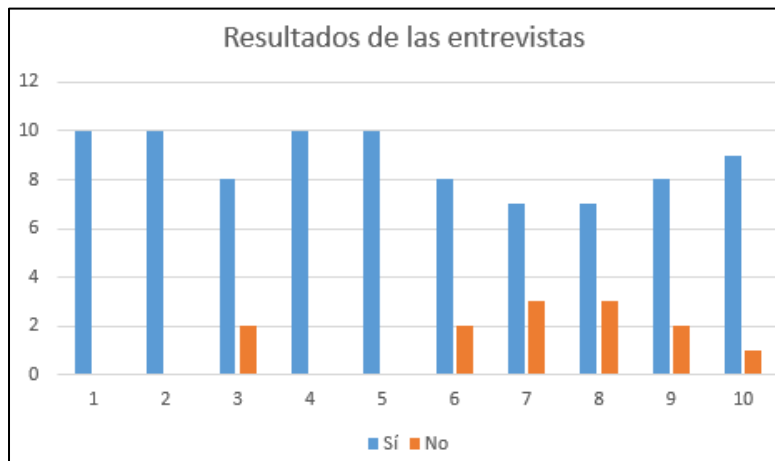


Figura 3.40. Resultados entrevistas clientes

La Tabla 3.3 presenta los resultados de la entrevista al administrador del restaurante

Tabla 3.3. Resultados entrevista administrador

No	Pregunta	Respuesta	
		Si	No
1	¿Pudo iniciar sesión?	X	
2	¿Pudo ingresar nuevos empleados a la aplicación?	X	
3	¿La aplicación te permitió editar tus datos de perfil?	X	
4	¿La aplicación en algún momento te hizo una petición de acceder a la galería de tu teléfono?	X	
5	¿La aplicación te permitió eliminar del sistema tanto a clientes como empleados?	X	
6	¿La aplicación te permitió ingresar un nuevo plato en el menú del restaurante?	X	
7	¿La aplicación fue sencilla y clara al manejar?	X	
8	¿La aplicación te permitió editar y eliminar los datos de los platos del menú?	X	
8	¿Pudo ver una lista tanto de los clientes como empleados registrados en el sistema?	X	
9	¿Pudo ver una lista de los platos que están pendientes de envío, así como aquellos que están en ruta y aquellos ya entregados?	X	
10	¿Recomendarías la aplicación?	X	

Como indican los resultados en la Tabla 3.3 las respuestas afirmativas del administrador son del 100%.

La Tabla 3.4 presenta los resultados de las entrevistas a cuatro empleados del restaurante.

Tabla 3.4. Resultados entrevista empleados

No	Pregunta	Respuesta	
		Si	No
1	¿Pudo iniciar sesión?	4	0
2	¿La aplicación te permitió editar tus datos de perfil?	4	0
3	¿La aplicación te permitió ver los pedido hechos por los clientes?	4	0
4	¿Recibiste una notificación de que hay un nuevo pedido?	3	1
5	¿La aplicación en algún momento te hizo una petición de acceder a la galería de tu teléfono?	3	2
6	¿La aplicación te permitió ver el detalle de los productos solicitados por el cliente?	4	0
7	¿La aplicación te permitió ver el lugar de entrega en el mapa?	3	1
8	¿La aplicación fue sencilla y clara al manejar?	3	1
9	¿Recomendarías la aplicación?	4	0

Como se observa en los resultados de la Tabla 3.4 los empleados respondieron de manera afirmativa a las preguntas teniendo sólo pocos empleados con respuestas negativas. La Figura 3.41 presenta el resultado de la entrevista de validación para los empleados.



Figura 3.41. Resultados entrevistas empleados

De manera general la aplicación es sencilla de usar y de acuerdo a los resultados realiza el trabajo para el cual fue desarrollada.

3.3. CORRECCIÓN DE ERRORES

En esta sección se presenta un resumen de los errores encontrados en la aplicación, tanto al momento del desarrollo como en las pruebas de validación.

La Tabla 3.5 se presentan los errores encontrados y corregidos en el módulo administrador.

Tabla 3.5. Correcciones módulo administrador

Error	Corrección
No se puede iniciar sesión.	La contraseña debe ser mayor o igual a seis caracteres, es un requerimiento de Firebase, se agrega un número mínimo de caracteres a la contraseña.
Los datos de perfil no se actualizan.	El usuario debe dar los permisos de acceso a la galería del dispositivo, la referencia en la base de datos no correspondía al usuario actual por lo que se usa la función <code>firebase.auth.currentUser</code> para identificar el <code>UserId</code> del usuario.
La aplicación se bloquea al ingresa un correo electrónico.	El problema sucedía con dispositivos con Android 10 y dispositivos Xiaomi por lo que se cambia el tipo de texto de entrada de email a text.
No se muestran los pedidos pendientes.	Se aplica un filtro en el cual solo se muestran los pedidos con el estado "Pendiente", de la misma manera para pedidos en ruta y entregados.
No es posible eliminar los datos de los empleados o clientes.	Se implementa un atributo "Estado" a los datos de los clientes y empleados que cambia de "habilitado" a "deshabilitado" cuando se elimina un usuario.
Los precios no se visualizan en pantalla.	Los precios de los platos se muestran como NaN, esto se debe a que JavaScript toma el punto como separador decimal, por lo que se implementa una función que verifica dicha cualidad.

La Tabla 3.6 presenta los errores encontrados y corregidos en el módulo cliente.

Tabla 3.6. Correcciones módulo cliente

Error	Corrección
No se presenta el mapa para elección del lugar de entrega.	Se indica al usuario que debe permitir el uso de la localización del dispositivo en todo momento para el correcto funcionamiento de la aplicación.
Las notificaciones no llegan.	El error sucedió al ingresar a los servidores de Expo una clave del servidor incorrecta por lo que se envía la clave correcta.
La suma de precios en el carrito excede en más de dos en los números decimales.	Se agrega una función propia de JavaScript para solucionar este error, se usa función <code>Number.toFixed()</code> .
La aplicación no mostraba las direcciones de entrega.	Al estar estos datos anidados en la base de datos se crea una referencia que apunta directamente a la localización de la dirección en cada usuario.
El chat no funciona de la manera prevista.	El chat de la aplicación muestra los mensajes de todos los usuarios, por lo que se crea una referencia en la base de datos con el identificador de usuario del cliente que se comunica a través del chat.
No se suman los productos extras agregados a la orden.	Los extras agregados a la orden no estaban contemplados en la base de datos por lo que se crea una función que agrega estos extras antes de guardarlos en el carrito de compras en la base de datos.

La Tabla 3.7 presenta los errores encontrados y corregidos en el módulo empleado.

Tabla 3.7. Correcciones módulo empleado

Error	Corrección
No se presenta el detalle de los productos en el pedido.	Se crea un nuevo nodo en la base de datos que contiene el detalle del pedido hecho por el cliente. independiente del identificador de usuario del cliente.
La dirección de entrega del pedido no aparece en pantalla.	Se pide que se acepten los permisos de ubicación del dispositivo y se habilita el pin de ubicación en las opciones de react-native-maps.

La Tabla 3.8 muestra los errores comunes a los tres módulos y sus correcciones.

Tabla 3.8. Correcciones generales

Error	Corrección
No se puede recuperar la contraseña.	El correo ingresado debe estar activo.
No se puede cambiar la contraseña.	Para realizar el cambio de contraseñase se debe ingresar correctamente la contraseña anterior, por lo que sin este requisito no es posible actualizar la contraseña.

3.4. ACTUALIZACIÓN TABLERO KANBAN

Como parte final de este capítulo se presenta la actualización del tablero Kanban, el cual, presenta las tareas en la última columna, por lo que se da por terminado cada una de las tareas planificadas a lo largo del desarrollo del Proyecto de Titulación. La Tabla 3.9 presenta la actualización del tablero Kanban.

Tabla 3.9. Tablero Kanban actualización final

Tareas	Tareas en proceso	Tareas realizadas
		Toma de Requerimientos.
		Realización de entrevistas al administrador y a diez clientes del restaurante.

Tabla 3.9. Tablero Kanban actualización final

Tareas	Tareas en proceso	Tareas realizadas
		Identificación de historias de usuario.
		Identificación de requerimientos funcionales.
		Identificación de requerimientos no funcionales.
		Determinación de los módulos del prototipo.
		Diseño de las vistas de la aplicación.
		Realización de diagramas de casos de uso.
		Realización de diagrama de actividades.
		Diagrama de estructura.
		Diseño de la estructura del proyecto.
		Diseño base de datos.
		Instalación de Node.js.
		Instalación de Yarn.
		Instalación Expo CLI.
		Instalación de Visual Studio Code.
		Instalación cliente Expo.
		Crear la base de datos en Firebase.
		Codificación pantallas del módulo administrador.
		Codificación pantallas del módulo Cliente.
		Codificación pantallas del módulo empleado.
		Codificación de navegación entre pantallas.
		Instalación de Firebase en la aplicación móvil.
		Codificación de funcionalidades del prototipo.
		Pruebas de funcionamiento módulo Administrador.
		Pruebas de funcionamiento módulo Cliente.
		Pruebas de funcionamiento módulo Empleado.
		Pruebas de funcionamiento con todos los módulos.
		Corrección de errores.

4. CONCLUSIONES Y RECOMENDACIONES

4.1. CONCLUSIONES

- Actualmente los locales de venta de comida ya sean estos pertenecientes a grandes franquicias o a pequeños restaurantes, necesitan una manera de promocionar y vender sus productos de una forma diferente a como se ha venido desarrollando tradicionalmente, es por eso, que se han desarrollado aplicaciones móviles que promocionan y permiten realizar la compra de los productos y además de que estos productos sean entregados en la comodidad del hogar o en el lugar que el cliente lo requiera, es así, que el prototipo desarrollado en este trabajo de titulación pretende ser una herramienta que permita al restaurante “El Asadero de Lali” promocionar sus productos y dotar a sus clientes de una aplicación que les permita realizar pedidos de comida sin salir de su domicilio, con lo que se ha cumplido el objetivo general de este proyecto de titulación.
- Se escogió React Native en este proyecto de titulación ya que permite integrar JavaScript con código nativo de Android el cual es Java. Además de que React Native permite utilizar la herramienta Expo, la cual facilita la ejecución del código en dispositivos Android, además de que tiene la opción de ejecutar el código en un explorador web en el caso de que no se disponga de un dispositivo Android.
- Se eligió los servicios que ofrece Firebase para el backend, ya que permite una integración sencilla con la aplicación móvil y provee de la autenticación, así como de la base de datos y almacenamiento en la nube que utiliza la aplicación para funcionar de forma correcta.
- Las entrevistas permitieron observar que tanto los clientes como los empleados del restaurante encontraron agradable el uso de la aplicación, así como también intuitiva y fácil de usar.
- La metodología Kanban al ser flexible permitió definir 3 etapas de trabajo las cuales fueron desarrolladas a lo largo de este trabajo de titulación, lo que permitió llevar un orden en el desarrollo del prototipo, tanto en la fase de diseño, así como también en la fase de implementación y pruebas.
- Al analizar los resultados obtenidos en las pruebas de validación, se observó que hay algunos errores que no se los noto en tiempo de desarrollo, por lo que es importante que el prototipo sea aprobado por los usuarios finales en este caso los

clientes, el administrador del restaurante y los empleados, con la finalidad de entregar un producto lo más depurado y cercano a lo que el usuario necesita.

4.2. RECOMENDACIONES

- React Native es parte de una amplia gama de framework para desarrollar aplicaciones móviles, por lo que, se recomienda la implementación de este proyecto en un framework diferente, tal como Flutter, Vue o Angular, y así comparar si hay diferencia en cuanto al aspecto visual o el rendimiento que presenta el prototipo al momento de su ejecución.
- Se recomienda el uso y difusión de los servicios de Firebase ya que permite tener un backend sin la necesidad de administrar servidores.
- Se recomienda la expansión del prototipo a mas restaurantes, es decir convertir a la aplicación en una que aglomere a varios restaurantes que necesiten promocionarse y enviar sus pedidos a sus clientes.
- Se recomienda que la aplicación se integre con redes sociales, es decir que el cliente pueda hacer comentarios del restaurante en la aplicación y que estos comentarios se reflejen en la página oficial del restaurante, esto para expandir la funcionalidad de la aplicación, así como también para maximizar el alcance de la misma.
- Como parte de la experiencia al desarrollar el prototipo se notó que hay algunas limitaciones en el lenguaje React Native, en especial con teléfonos de la marca Xiaomi con Android 10, por lo que se recomienda especial atención al momento de ejecutar cualquier aplicación desarrollada en React Native con estos dispositivos.
- El prototipo está desarrollado para el sistema operativo Android, por lo que se recomienda el desarrollo del prototipo para el sistema operativo iOS.
- Se recomienda que la aplicación pueda realizar el inicio de sesión mediante cuentas de redes sociales como Facebook y poder cambiar entre cuentas una vez se realiza la validación del usuario.

5. REFERENCIA BIBLIOGRÁFICA

- [1] Medium, “¿Por qué desarrollar apps de React Native en Expo?”, 2017. [En línea]. Disponible: <https://medium.com/leopark-lab/por-qu%C3%A9-desarrollar-apps-de-react-native-en-expo-2e1b83e4d00a>. [Último Acceso: 18 octubre 2020].
- [2] Computer Traininig, “React Native: Una nueva arquitectura para la generación de Apps”, 2018. [En línea]. Disponible: <https://www.bit.es/knowledge-center/react-native-una-nueva-arquitectura-para-la-generacion-de-apps-moviles/>. [Último acceso: 18 octubre 2020].
- [3] P. Akshat; N. Abhisek, “React Native for Mobile Development”, 2nd. Ed. New York: Apress, 2019, pp. 2.
- [4] Desarrollo web, “Qué es React”, 2019. [En línea]. Disponible: <https://desarrolloweb.com/articulos/que-es-react-motivos-uso.html>. [Último acceso: 20 octubre 2020].
- [5] Facebook Open Source. “Documentación oficial de react”. [En línea]. Disponible: <https://reactjs.org>. [Último acceso: 17 octubre 2020].
- [6] JavaTpoint, “React Flux Concept”. [En línea]. Disponible: <https://www.javatpoint.com/react-flux-concept>. [Último acceso: 18 octubre 2020].
- [7] Expo, “Introduction to Expo”. [En línea]. Disponible: <https://docs.expo.io/>. [Último acceso: 20 Octubre 2020].
- [8] Amazon, “¿Qué es NoSQL?”. [En línea]. Disponible: <https://aws.amazon.com/es/nosql/>. [Último acceso: 19 septiembre 2020].
- [9] P. Sadalage, M. Fowler, “NoSQL Distilled. A Brief Guide to the Emerging World of Polyglot Persistence”. Addison-Wesley, 2012.
- [10] Firebase, “Estructura datos”. [En línea]. Disponible: <https://firebase.google.com/docs/database/web/structure-data>. [Último acceso: 10 octubre 2020].
- [11] IEBS, “¿Qué es Firebase?”. [En línea]. Disponible: <https://www.iebschool.com/blog/firebase-que-es-para-que-sirve-la-plataforma-desarrolladores-google-seo-sem/>. [Último acceso: 12 octubre 2020].
- [12] Node JS, “Introduction to Node.js”. [En línea]. Disponible: <https://nodejs.dev/learn>. [Último acceso: 13 octubre 2020].
- [13] Yarn, “Uso”. [En línea]. Disponible: <https://classic.yarnpkg.com/es-ES/docs/usage>. [Último acceso: 13 septiembre 2020].

- [14] Expo, "Expo CLI". [En línea]. Disponible: <https://docs.expo.io/workflow/expo-cli/>. [Último acceso: 05 octubre 2020].
- [15] V. S. Code, "Getting Started". [En línea]. Disponible: <https://code.visualstudio.com/docs>. [Último acceso: 15 septiembre 2020].
- [16] Google Cloud, "Google Maps Platform". [En línea]. Disponible: <https://cloud.google.com/maps-platform?hl=es>. [Último acceso: 15 octubre 2020].
- [17] Marvel. [En línea]. Disponible: <https://marvelapp.com/>. [Último acceso: 20 octubre 2020].
- [18] Kanmanize, "Qué es Kanban: Definición, Características y Ventajas". [En línea]. Disponible: <https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban>. [Último acceso: 20 octubre 2020].
- [19] D. L. Arias, "Metodologías de desarrollo de software". [En línea]. Disponible: <http://www.alfarosolis.com/content/PDFs/IF7100/semana8/Metodologias.pdf>. [Último acceso: 18 septiembre 2020].
- [20] Uber Eats. [En línea]. Disponible: <https://www.ubereats.com/ec>. [Último acceso: 16 octubre 2020].
- [21] Glovo. [En línea]. Disponible: <https://glovers.glovoapp.com/ec/>. [Último acceso: 16 octubre 2020].
- [22] Rappi. [En línea]. Disponible: <https://www.rappi.com.ec/>. [Último acceso: 17 octubre 2020].
- [23] Firebase, "Firebase Cloud Messaging". [En línea]. Disponible: <https://firebase.google.com/docs/cloud-messaging>. [Último acceso: 18 octubre 2020].
- [24] Expo, "Sending Notifications with Expo's Push API" [En línea]. Disponible: <https://docs.expo.io/push-notifications/sending-notifications/>. [Último acceso: 20 octubre 2020].
- [25] Expo, "Push Notifications Overview". [En línea]. Disponible: <https://docs.expo.io/push-notifications/overview/>. [Último acceso: 26 octubre 2020].

ANEXOS

ANEXO A. Modelo de entrevista inicial y resultados.

ANEXO B. Diseño de la interfaz El Asadero de Lali.

ANEXO C. Modelo de entrevista de validación y resultados.

ANEXO D. Manual de usuario.

ANEXO E. Código del prototipo.

ORDEN DE EMPASTADO