UNIVERSIDAD SAN FRANCISCO DE QUITO
USFQ

Colegio de Ciencias e Ingenierías

La aplicación "Seva": Desarrollo y digitalización de la industria de comida en el Ecuador para la optimización de los negocios y facilitación de la interactividad entre el restaurante y el cliente.

.

Guillermo Nicolás Cisneros Villota
Ingeniería en Ciencias de la Computación

Trabajo de fin de carrera presentado como requisito
para la obtención del título de
Ingeniero en Ciencias de la Computación

Quito, 19 de diciembre de 2023

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

HOJA DE CALIFICACIÓN
DE TRABAJO DE FIN DE CARRERA

La aplicación "Seva": Desarrollo y digitalización de la industria de comida en el Ecuador para la optimización de los negocios y facilitación de la interactividad entre el restaurante y el cliente.

.

Guillermo Nicolás Cisneros Villota

Nombre del profesor, Título académico Daniel Fellig Goldvechmiedt , MS

Quito, 19 de diciembre de 2023

© DERECHOS DE AUTOR

Nombres y apellidos: Guillermo Nicolás Cisneros Villota

Código: 00206286

Cédula de identidad: 1723253371

Lugar y fecha: Quito, 19 de diciembre de 2023

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

RESUMEN

El proyecto se basa en el desarrollo de una aplicación móvil que por medio del uso de distintas tecnologías de software y hardware facilita y optimiza la interacción entre cliente y establecimientos o restaurantes. La utilidad máxima de este proyecto y el enfoque de su desarrollo está en maximizar la relevancia actual de las aplicaciones móviles para facilitar el flujo de trabajo dentro de la industria de los restaurantes en Ecuador. Esta es una aplicación móvil que puede desplegar un menú digital proporcionado por un restaurante al leer un código QR que representa una mesa dentro del establecimiento . Lo que sigue una vez que los usuarios hayan pasado por un proceso de autenticación hecho por la actividad de registro o "login" dentro de la aplicación, es la lectura de un código QR asignado a su mesa lo cual permite cargar el "menú" que ofrece el establecimiento. Lo siguiente en el flujo de la orden, los clientes escogen los ítems en el menú que desean, la cantidad de estos y si necesitan añadir alguna instrucción especial, se incluye en la orden y eso pasa para ser procesado por el restaurante. El objetivo principal del proyecto es que tenga un diseño amigable, una estructura simple y que sea entendible. En este trabajo se utilizó conceptos esenciales aprendidos en los cursos principales en la carrera de ciencias de la computación. Esta aplicación al ser desarrollada en Android Studio, se la hizo manteniendo en mente el poder ser utilizada en la mayor cantidad de dispositivos posibles, sin embargo, no se descarta una posible implementación en sistemas operativos IOS ya que haría bien en su propósito de ser lo más accesible para el público en general.

Palabras clave: Aplicaciones, Restaurantes, Ecuador, Menú, Clientes, QR

ABSTRACT

Las aplicaciones móviles han gozado de una gran popularidad estos últimos 20 años ya que han logrado simplificar en base a la automatización brindada por la tecnología. El avance en términos de ideas para las aplicaciones prácticas de tecnología móvil sigue expandiéndose cada vez más. Una parte importante de esta expansión en aplicaciones prácticas es en la industria alimenticia. Estas aplicaciones han traído muchos beneficios tanto a clientes como a propietarios de restaurantes. El trabajo del proyecto integrador se basa en el desarrollo de una aplicación que automatiza el flujo de pedido de órdenes para restaurantes que por medio de la lectura de QRs, puede cargar un menú de platillos principales, platos extras y bebidas. Esto le permite al usuario la habilidad de crear un orden con la cantidad e instrucciones que este quiera para cada uno de sus platos ya que se pasara a cocina como un resumen de lo que la mesa o cliente necesite. Se desarrolla actualmente para el sistema operativo Android, aprovechando los beneficios del ambiente de Android Studio y la facilidad que este otorga para proyectos de este tipo. De igual manera, se tiene como base del proyecto al servicio de Google conocido como Firebase y busca beneficiar tanto a usuarios como a propietarios de los restaurantes.

Palabras clave: Aplicaciones, Restaurantes, Ecuador, Menú, Clientes, QR

TABLA DE CONTENIDO

INDICE DE ANEXOS

1. INTRODUCCIÓN

Ecuador, desde los días de su fundación en 1822 hasta la Postpandemia de 2020, siempre ha sido una nación que en general ha sido abierta a las nuevas tendencias y cambios. Esto no quita que haya también actitudes conservadoras por parte de su población. A pesar de esto, el país en general ha podido adaptarse a los cambios de las épocas y particularmente a lo que ha implicado la creciente globalización. En este contexto, nada puede ejemplificar esta adaptabilidad y crecimiento cultural como la acogida que se le ha dado a la tecnología y a la digitalización de varias industrias, desde finales del Siglo XX hasta los tempranos 2020s. Ninguna nueva tecnología o cambios que esta proporcione viene sin críticas y en Ecuador este hecho se vuelve muchas veces muy evidente. Desde la desconfianza hacia las maquinas hasta el rechazo de cualquier medio digital, muchos sectores de la población han tenido que aceptar que la tecnología vino para quedarse. El constante cambio no es una cuestión de posibilidad sino de tiempo. La tecnología no es inherentemente buena o mala, sino que depende del uso que se le dé y el uso responsable de esta puede traer más beneficios que problemas a la sociedad. Esta es la razón por la cual muchas áreas e industrias han podido modernizarse en cuanto a su estructura de trabajo, usando al máximo las nuevas tecnologías que disponen. Una de estas nuevas tecnologías es la computación, simplificando muchos procesos y mejorando la eficiencia tanto en el trabajo como en la vida diaria de las personas.

La implementación total de la computación en nuestras actividades cotidianas representa un reto para todo desarrollador de programas y aplicaciones. No solo se debe investigar, sino que se debe entender la actividad cotidiana que se busca trabajar. Una vez que se entiende, debemos usar la computación para convertir a esta acción en algo más tangible y concreto. Esto permite la formación de un programa que encapsula el concepto deseado, lo representa de manera

sencilla y permite accesibilidad para que los usuarios y otros agentes que vayan a utilizarlo. El análisis del sistema para poder satisfacer plenamente los requisitos de los usuarios es uno de los mayores retos que un desarrollador puede tener al trabajar en nuevos proyectos. Debido a esto, se planteó un proyecto integrador que lograra adaptar nuevas tecnologías como las aplicaciones móviles con la cultura gastronómica del Ecuador. Para que esto pudiera suceder, se debe entender como los clientes actúan antes, durante y después de consumir los alimentos. También es necesario conocer la interacción de los usuarios con el personal del restaurante. Esto permite tener una aplicación móvil con una estructura sencilla, dinámica y que se acopla a las necesidades de tanto usuarios como establecimientos.

Para entender mejor de que trata el proyecto integrador, se lo puede ver más en detalle en la sección de "Descripción del Proyecto (Pag.10)". Ahí se profundiza sobre la estructura de este, sus objetivos y alcance. En resumen, es una aplicación móvil que permite a los usuarios interactuar de manera más eficiente con los establecimientos al momento de ordenar y consumir los platillos de estos. La aplicación se divide en 4 secciones: autenticación del usuario, lectura de códigos QRs, menú principal y checkout. Da la libertad del cómo se accederá, de que se consumirá y de qué manera se pagará. Esta es una aplicación centrada en el sistema operativo Android, usando como ambiente de desarrollo a Android Studio. Se hace eso para obtener una mayor compatibilidad con dispositivos que usen este sistema operativo. Esto ya que se busca manejar posibles problemas de implementación como lo sería que los usuarios no lleven su propio celular. El restaurante puede prevenir esta situación e instalar dispositivos que estarían usando la aplicación sin mayor inconveniente. La principal motivación del proyecto es la "universalidad", el permitir que cualquier persona pueda utilizar el mayor número de tipos de dispositivos posibles y beneficiarse de este programa. Este es un trabajo que busca el beneficio de los clientes y negocios mejorando un esquema de trabajo ya

existente. La aplicación lleva el nombre de “Seva”, lo cual en sanscrito significa “servicio incondicional o desinteresado”. Se escogió este nombre ya que se acopla bien con el objetivo de esta y es de fácil memorización. Actualmente, se encuentra desarrollado en su etapa Beta, es decir, la versión más funcional que podría mostrarse tanto a los clientes como a los establecimientos interesados.

2. IMPORTANCIA.

La falta de disponibilidad de una aplicación como esta en el territorio ecuatoriano fue una de las principales razones por las que se escogió este tema como proyecto integrador. No es la primera aplicación enfocada en modernizar el flujo de trabajo de un establecimiento ya que en varios países se ha implementado este tipo de modernización y nuevos usos de tecnología en los restaurantes. Como ejemplos se tiene la creación de menús digitales en pdf o aplicaciones web que leen códigos QR y generan menús virtuales. La modernización y automatización en la industria de restaurantes promete mejoras en términos de atención al cliente y del desempeño laboral de los empleados de los establecimientos. En cuanto al proyecto, lo que lo hace destacar es que unifica varios conceptos ya implementados por otras aplicaciones para dar un producto nuevo, más estructurado y enfocado al mercado ecuatoriano. Específicamente, es una aplicación que permite la autenticación de los usuarios, la lectura de códigos QR, la generación de un menú interactivo y un proceso de chekout y de comunicación con el counter del restaurante.

Aparte de las motivaciones socioculturales que tiene la aplicación, se tiene como objetivo demostrar los conocimientos adquiridos durante la carrera de ingeniería en Ciencias de la Computación en la universidad. El desarrollo de esta representa una utilización muy práctica de los conceptos aprendidos como la programación orientada a objetos (POO), el uso de interfaces y la implementación de servicios como Firebase. El potencial de la aplicación en cuanto a su funcionamiento y estructura permite pensar en un futuro lanzamiento comercial de la misma hacia plataformas como Play Store en Android. Una importante aclaración es que, si bien se busca mejorar la interacción entre los clientes y restaurantes, también se busca alivianar la carga de trabajo para el personal de los locales comerciales. Los empleados ya no tienen que tomar directamente las órdenes y esto ayuda a reducir los altercados con clientes, evitar problemas de falta de comunicación, lo cual resulta en un mejor rendimiento del personal, tanto meseros como cocineros.

3. DESCRIPCIÓN DEL PROYECTO

Describiendo ya con detalle la aplicación, fue desarrollada en Android Studio con versiones de MinSDK 24 y TargetSDK 34. Esto se configuró para tener una compatibilidad del 95% de los dispositivos Android. La aplicación se divide en 4 secciones importantes; Login/Registro, Lectura de QRs, Menú Principal y Checkout. Cada sección representa una o más actividades dentro de la aplicación ya que cada sección es la agrupación de distintos componentes visuales. Por ejemplo, el menú principal este compuesto de varios nodos que representan cada uno de los platillos disponibles. Si se hace click a uno de estos ítems, se muestra la actividad de “detallesPlatillo” que tiene una imagen del plato, su nombre/precio, descripción, instrucciones, editor de cantidades y precio total. Volviendo al comienzo del programa, la actividad “Registro” permite la creación de una cuenta de usuario basada en el método personalizado de

autenticación. La actividad “Login” permite la autenticación de los clientes por medio de tres métodos de inicio de sesión: Método personalizado (ingreso de mail y contraseña configurada en “Registro”), Google y Facebook por medio de las cuentas ya vinculadas en el dispositivo. La sección de Checkout muestra la orden del cliente/s compuesta por los distintos platos con un precio a pagar y se le concede la libertad de actualizar la cantidad de platos, escoger los métodos de pago y si se necesita factura. También se desarrolla un modelo de comunicación entre usuario-recepción ya que permite él envió de las ordenes por medio de mensajes SMS hacia el número registrado por el establecimiento.

Para realizar el desarrollo de la aplicación, se tuvo que crear diagramas de actividad de tanto usuarios como del sistema, así como un diagrama de clases de todo el proyecto. Esto se realizó para diseñar y concretar el flujo de trabajo de la aplicación y todas las decisiones lógicas que los actores principales y el sistema deben realizar en cooperación para que el programa funcione correctamente. Los diagramas representan un ciclo de vida de la aplicación en sus primeras etapas, si bien hay cambios con el producto final sirven como una guía para entender el programa. Estos pueden verse en el índice del proyecto y están organizadas de la siguiente manera:

1. Diagrama de Secuencia de usuario (“Anexo 1: Diagrama de Actividades de Usuario”)
2. Diagrama de Secuencia del sistema: (“Anexo 2: Diagrama de Actividades del Sistema”)
3. Diagrama de Robustez del usuario: (“Anexo 3: Diagrama de Robustez del Usuario Parte I”)
4. Diagrama de Robustez del usuario: (“Anexo 4: Diagrama de Robustez del Usuario Parte II”)
5. Diagrama UML del Proyecto: (“Anexo 5: Diagrama UML del Proyecto”)

Debido a que se busca obtener la versión Beta de la aplicación (La versión más funcional del programa), se decidió mantener el enfoque por ahora en el sistema de Android. Una vez que se tenga una versión más completa de la app, se considera hacer una adaptación de esta hacia los sistemas operativos de IOS. Esta acción se realiza para enfocar todos los esfuerzos en conseguir una versión estable y funcional. Para el desarrollo de la aplicación, se ve un uso de diferentes implementaciones en el "gradle" (archivo único de cada proyecto en Android Studio que permite el uso de implementaciones y herramientas externas al ambiente de desarrollo). Esto permite agregar al código distintos objetos y funciones especiales que cumplen con los requisitos de las 4 partes del programa. Esto puede decirse de las dependencias otorgadas por Firebase como "com.google.firebase:firebase-auth:22.1.2" para el proceso de autenticación, "com.google.firebase:firebase-storage" para el uso y descarga de elementos como PNGs e incluso de otras funciones como "com.google.mlkit:barcode-scanning:17.2.0" que permite a la cámara poder analizar e identificar códigos de barra y en lo que este proyecto enfoca, el uso de QRs. Estos siendo ejemplos de los distintos elementos que el programa usa como herramientas para habilitar la funcionalidad deseada. Para usar los recursos mencionados, se tuvo que configurar cuentas en distintos servicios para garantizar la implementación de las APIs de los mismos en la aplicación. Por ejemplo, en "Firebase" hay que conectar al proyecto de Android Studio con un proyecto de ese servicio. Esta acción permite una comunicación entre los dos lados, el proyecto local pueda acceder a los recursos de autenticación y storage de Firebase mientras que el proyecto remoto puede almacenar los datos que se den desde la aplicación. "Meta for Developers" sigue la lógica previamente descrita, este es un servicio proporcionado por Meta para que los desarrolladores puedan acceder a los datos de los usuarios de Facebook e implementarlos de diversas maneras en sus proyectos. Meta otorga los datos de los usuarios para el proceso de autenticación proporcionado por Firebase. Los clientes pueden hacer un proceso de inicio de sesión por medio de sus cuentas de Facebook al tener esta conexión entre

Meta-Firebase-Android. Como último ejemplo, “PayPal for Developers” es otro servicio que permite a las aplicaciones usar los recursos de la API de la empresa homónima como una pasarela de pagos válida. Para el proyecto se establece a PayPal en modo Sandbox, este modo emula transacciones reales con cuentas falsas para simular los procesos de pago.

4. ESTADO DEL ARTE

Como ya se menciona en las anteriores secciones, la aplicación está destinada a usarse en restaurantes y negocios de manera presencial. Para cumplir esto, se debe usar las herramientas adecuadas para un óptimo desempeño del programa en tiempo real. Dichas herramientas se proveen al desarrollar la aplicación en el ambiente de trabajo de Android Studio. Este es reconocido como una plataforma sólida y confiable para la creación de aplicaciones móviles en el sistema Android. Esta nos ofrece un conjunto de herramientas altamente robustas que agilizan todo el proceso de desarrollo. Su interfaz de usuario intuitiva y sus características de depuración integradas, facilitan la creación y prueba de la aplicación de manera eficiente. Como software, fue de los primeros IDE que abarcaron de forma directa el problema de crear aplicaciones en Android. Desde ese entonces, ha ido cambiando y evolucionando con el tiempo. Su amplia variedad de plantillas y componentes predefinidos permite acelerar el desarrollo de aplicaciones y garantizar la consistencia en la interfaz de usuario. Debido a que ya es un ambiente de desarrollo maduro y reconocido, podemos acceder a un vasto repositorio de recursos, tutoriales y soluciones a problemas comunes. Usar este ambiente habilita a los desarrolladores de implementar recursos gráficos y lógicos que son innovadores y eficientes en comparación con los otorgados por distintos ambientes de trabajo. Esto se debe a que Android Studio nos permite crear una aplicación robusta, altamente receptiva y de alta calidad que se ejecuta sin problemas en una amplia gama de dispositivos Android. Los usuarios

disfrutarán de una aplicación que es fácil de usar, rápida y confiable, lo que contribuye a una experiencia positiva en la plataforma que, a su vez, beneficiará a los restaurantes al aumentar la satisfacción del cliente y fomentar la lealtad a la marca. (Chaubey & Sharma, 2023)

Para lograr una gestión eficiente de menús, órdenes de clientes, transacciones financieras y recibos, se ha optado por utilizar bases de datos relacionales como MySQL y Firebase. Este último siendo la base del proyecto, ya que su alcance llega a las 4 secciones principales de la aplicación. Como se explicó previamente, Firebase es un servicio desarrollado por Google y que permite a los desarrolladores una amplia gama de herramientas y recursos. Esto con el fin de hacer a las aplicaciones más organizadas, eficientes y autosostenibles. En el caso de Android Studio, se permite la implementación de este servicio por medio de un archivo .json que conecta el proyecto del ambiente con un proyecto creado en el servicio. Este por ejemplo proporciona distintas ventajas como lo sería los métodos de autenticación de usuarios, estadísticas, base de datos, así como el de método de pagos. En un principio, se decidió usar MySQL, pero se cambió a Firebase por el mayor número de recursos que esta podría proporcionar más allá de una base de datos. El servicio de Google se volvió la columna vertebral del código que se desarrolló ya que su sinergia con Android Studio es más eficaz al completar los aspectos que el proyecto se plantea. Además de esto, debido a su extensa documentación oficial, nos permite tener una mayor confianza en sus elementos, objetos y funciones que usamos y que las dependencias de dónde vienen estos elementos están actualizados y funcionales.

Ya que mencionamos los aspectos importantes del código, uno de los más destacados es la implementación de códigos QR. En sí, no es algo nuevo ya que como lo establece (Luna et al., 2021), ya se ha implementado este sistema de QR para recibir las ordenes de una tienda de alitas “Las Che Alitas” en Tizayuca, México. Esto aplica una lógica similar a la del proyecto

para agilizar el trabajo de los meseros en cuanto a la recepción de ordenes de clientes. Esto ya que se da un uso de códigos QR similar al planteado en el proyecto integrador. En este restaurante, se usan los códigos QR para representar cada platillo del menú y el usuario los añade de manera similar a su orden para luego enviársela a la cocina. Aquí se reciben las ordenes en un software basado en NetBeans que procesa cada pedido. Si bien es similar en cuanto al concepto, el proyecto integrador es diferente en comparación al trabajo realizado en México. Principalmente porque usamos los códigos QR para representar un menú asignado a una mesa en específico y no a cada platillo. No usamos un programa adicional para recibir las ordenes de los clientes, sino que se envían por SMS hacia la cocina. Se busca implementar en Ecuador, basarse en la cultura del consumo nacional y poseer funcionalidades extra en la aplicación. Estas funcionalidades siendo el uso de distintos métodos de registro para los usuarios, una lectura de QRs que usan el número de mesa como diferenciador principal, un menú interactivo de fácil manejo y una sección de checkout que puede usar distintos métodos de pago para procesar las órdenes. En el inicio de sesión de usuarios, se busca ofrecer múltiples opciones para satisfacer las preferencias de los clientes. Los usuarios pueden iniciar sesión con su cuenta de Google, Facebook o incluso optar por un registro manual, lo que brinda una flexibilidad considerable y se ajusta a las preferencias individuales de cada cliente. Todo esto se logra por medio de Firebase Authentication, ya que como lo establece en la página de Firebase (Google, 2023) esta nos permite una autenticación segura y sin problemas, una base de datos de repuesto, Storage o almacenamiento para los distintos elementos que necesitemos. y de distintas maneras de conectarnos al servicio por medio de la autenticación de los tokens de la aplicación y del usuario.

Además de la información técnica previamente mencionada, se ha logrado investigar artículos científicos relacionados con la gestión de restaurantes y la digitalización de procesos para entender los avances que se han hecho en estas áreas de estudio y como el proyecto puede colaborar. Uno de los artículos investigados, titulado "User Requirements and Design Guidelines for Digital Restaurant Menús" nos proporciona información valiosa sobre las expectativas de los clientes y las mejores prácticas de diseño en la implementación de menús digitales. Esto resulta relevante ya que podemos aprovechar los hallazgos encontrados aquí y de cierta manera aplicar esta guía en nuestro proyecto para determinar su validez de una forma más práctica. Si escuchamos a Lessel y colaboradores (2012), la adopción de tecnologías móviles en la industria de restaurantes es un fenómeno que está en constante crecimiento. La digitalización de procesos, la gestión eficiente de menús y órdenes, y la mejora de la experiencia del cliente son objetivos comunes en este campo. Por lo tanto, ahora existen ciertos parámetros y reglas que deben seguirse en el desarrollo e implementación en un proyecto como este. Si se busca aplicar y contribuir en esta área de estudio, debemos aplicar las mejores prácticas identificadas en la literatura y considerar los resultados que ya se han dado en aplicaciones similares a esta para entender su confiabilidad. Debemos aprovechar la capacidad de bases de datos como Firebase por su fortaleza para administrar datos críticos en tiempo real y el uso de las tecnologías inherentes y recursos en el ámbito de aplicaciones móviles para que tener un proyecto estable. Esto en si cobra relevancia al tener en cuenta que en nuestro país si bien existen aplicaciones que ya usan la digitalización de procesos en la industria de la comida, todavía no ha sido adoptado del todo estos cambios y algo como lo que propone el proyecto aún tiene que verse aplicado aquí.

Como una de las piezas de información más relevantes para este trabajo, tenemos al artículo "Restaurant Information System (RIS) with QR Code to improve service operations of Casual Fine Dining Restaurant" que ofrece de cierta manera una aplicación real de un programa como el que el proyecto propone ya que se basa en el uso de códigos QR para mejorar la experiencia de usuarios en restaurantes. Su estudio se enfoca en mejorar la calidad del servicio en la industria alimentaria, abordando específicamente problemas en la toma de pedidos, emisión de facturas y procesos de pago. Este enfoque es altamente relevante para el proyecto “Seva” ya que se busca agilizar estos procesos, mejorar la experiencia del cliente en restaurantes locales y evitar que los errores comunes en la industria ocurran. El artículo propone la implementación de un sistema basado en web que utiliza códigos QR para simplificar el proceso de pedidos y pago. Este enfoque se relaciona directamente con la visión de Seva de aprovechar la tecnología móvil para facilitar la gestión de restaurantes y mejorar la experiencia del cliente. Como en el mismo documento se destaca, uno de los mayores problemas para los restaurantes es que a los clientes se les sirva platos que no fueron escogidos por ellos ya que en la encuesta realizada en la investigación a 1003 participantes, el 62% de quejas era de que su comida no era lo que ellos pedían (Intal et al., 2020). Esto es un factor que se busca reducir en el trabajo planteado al ser un problema de cierta manera universal, sin embargo, en este mismo se busca una diferente implementación de la tecnología planteado en el artículo. Esto ya que, si bien nos basamos en los QR como fundamento del proyecto, este no es el único elemento importante de la aplicación ya que se busca mezclar diferentes tipos de tecnologías vistas en otras aplicaciones como en Uber Eats o Rappi para tener una aplicación más fuerte y que de más libertad a los usuarios con respecto a sus órdenes. Además de esto, hay que considerar que se busca adaptar a una población diferente ya que en el artículo se destaca la implementación de este sistema en restaurantes más lujosos de Filipinas mientras que el proyecto busca activamente ser aplicado en pequeños y grandes negocios para volverlo más universal. Esto sin contar que tiene como

objetivo ser más inclusivo hacia el público con una opción de múltiples idiomas y formas de interacción para personas que cuenten con algún problema de salud o discapacitada lo cual no es el enfoque de la investigación citada.

5. ANALISIS GENERAL DE LA APLICACIÓN.

El desarrollo del proyecto presenta distintas etapas que se deben seguir en orden para garantizar la obtención de una aplicación lo más completa posible. Como ya se ha mencionado, se ha dividido el proyecto en las siguientes etapas basado en las acciones más importantes que se deben realizar en el ciclo de vida del código: Autenticación, Lectura de QRs, Menú Principal y Checkout. Cada una de estas partes representan a las respectivas actividades presentes en el código que son como puntos de control debido a que en cada uno de estos se hace un manejo de la mayor cantidad de información que se le da y envían un similar número de datos hacia las otras actividades. Gráficamente, esta aplicación implementa un estilo personalizado para los diseños con colores escogidos de tal manera que emule “chalkboards” o pizarrones de tiza de restaurantes donde normalmente se ponen los platillos disponibles de estos negocios. En el código, implementamos al lenguaje de programación de Java para todo el proyecto. Java representa un mayor grado de cercanía en cuanto a lenguajes aprendidos por su extenso estudio en las materias cursadas de la carrera y además que el ambiente de trabajo nos ofrece más comodidad y beneficios al usar este. Firebase es la piedra fundacional de todo el proyecto debido a que abarca todo el proceso de desarrollo al darle a nuestra aplicación, por medio de las dependencias necesarias, los elementos para tener lo más cercano a una aplicación como se pueda tener. Sin embargo, también se ha visto el uso de distintas librerías y recursos como lo ha sido hasta el momento:

- CamaraX (Permite la utilización del hardware de la cámara).
- ZXing (Permite la creación y utilización de códigos de barra y QR)
- Facebook (Permite el uso de herramientas de Meta para la autenticación de usuarios)

Volviendo al uso de Firebase, de este se usa la sección de autenticación ya que involucra todo el proceso de login y registro. Usamos por otro lado la sección de storage en la sección del menú principal y las actividades adyacentes de esta. Storage funciona como un repositorio donde podemos llamar y descargar las imágenes de los platos del menú y lo más importante que sería el archivo de texto. Nombrado como “Mesa1.txt”, es un archivo donde se establece el nombre del restaurante, número de mesa, numero de platos y cada uno de estos con un nombre, descripción, precio, extras y una imagen referencial. Este está parametrizado con distintos signos como ”/”, “%” o “()”. Estos sirven para dividir el texto y que cuando se procese se puedan sacar los valores correspondientes como las características de cada plato, cada uno de estos u otros elementos como el nombre del restaurante. Es importante aclarar ya que el QR que generamos se basa en un enlace hacia este repositorio donde está el texto, una vez que se lee en la actividad correspondiente lo descargamos e implementamos el contenido leído del mismo hacia nuestro esquema en la actividad del Menú Principal. Esta posee 3 categorías de platos que los restaurantes pueden implementar: Platillos principales, Platos Extras y Bebidas. Esto ya que son las 3 categorías más comunes que poseen la mayoría de los establecimientos.

Sin embargo, debido a la estructura planteada para el código , si es necesario solo se debe modificar el archivo fuente de donde se obtienen los ítems del menú y arreglar ciertos aspectos en la lectura de este para poder ser usado por más restaurantes. Volviendo a la estructura de la aplicación, cada platillo presenta su valor o índice dentro de la lista, así como su nombre y el

precio. Cuando se toca alguno de los ítems, se redirige a una actividad que muestra la imagen del plato con mejor escala, el nombre, precio y descripción del plato. El usuario puede introducir un texto en el área de instrucciones si este quiere que algún alimento se incluya o no en el plato, así como alguna posible alergia. Cada actividad de detalles de los platos cuenta con un LinearLayout que almacena 3 elementos: botón "-", contador de platos y el botón "+": Esto permite a los clientes poder definir la cantidad de platos que quiera, actualizando el precio en el botón de agregar para mostrar la cantidad actual y su respectivo precio. Esta actividad envía una señal (Mas detalles en la sección Actividad de Detalles de Plato) que actualiza el diseño de la actividad donde se encuentra el menú con los ítems agregados al carrito, el precio total de la orden y un identificador visual en formato de numero para diferenciar la cantidad de platos que se han seleccionado. El plato se lo añade a un objeto llamado "Orden" el cual se va actualizando cada vez que los usuarios añaden un platillo, sumando un valor en el textField de cantidad y precio totales.

Una vez que los usuarios deciden ir a pagar deben hacer click al botón de "Checkout" donde el contenido del objeto se muestra en una lista con la cantidad seleccionada para cada plato, instrucciones si se añadieron y el precio total individual de cada platillo como el precio general de todo el plato. El usuario aquí ya posee una vista previa de los platillos y la habilidad de añadir más o reducir la cantidad de platos, cambiando el precio del subtotal y total de su orden. Lo que se hace es que se le ofrece métodos de pago para completar su orden, el selecciona el que quiera y la orden se envía hacia la cocina en formato de Texto/Impresión con toda la info de la orden, número de mesa de donde viene la orden, nombre del cliente y la cantidad total debitada de la orden. Esta sección por el momento requiere de mayor atención puesto que se sigue investigando una manera efectiva de "lanzar" la orden hacia la cocina y en si se estima que eso será un aspecto que vera más desarrollo después del proyecto. En conclusión, se puede

decir que el proyecto en esencia es la interacción entre estas 4 actividades ya que buscan guiar a los usuarios de manera efectiva y concreta para que puedan hacer su orden en los restaurantes y que tengan la mayor cantidad de información con respecto a sus órdenes. A continuación, se hablará más en detalle de las actividades más importantes usadas en el proyecto.

Se hizo uso de diversas clases como el de “Platillo”, “Menú Principal” y sus respectivos descendientes “OrderItem” y “Order” que en si permiten la recolección de la información dada del código QR cuando se lo lee, así como de las elecciones de los clientes. De igual manera, se estableció una estructura sencilla y modular que permita la fácil edición y corrección de errores ya que por ejemplo si se necesita editar algún elemento del QR, no se tiene que usar un código que genere uno nuevo ya que al basarse en la dirección del archivo fuente (Para las pruebas el de Mesa1.txt), solo se debe cambiar este siguiendo el formato especificado para que el código pueda leer el contenido y clasificar correctamente el contenido del archivo. Son estas acciones que en si son simples pero que garantizan una facilidad de utilización tanto por los usuarios como del staff del restaurante.

6. ACTIVIDAD DE LOGIN

Como se menciona en el análisis general del código, La actividad entrada hacia la aplicación es la de Login. En si todo inicia con una actividad splash para mejorar la entrada de los usuarios y que no se aparezca inmediatamente la pantalla de inicio de sesión. En si se utiliza un estilo personalizado de la aplicación de modo que busca asemejarse a los “chalkboards” que varios restaurantes y negocios poseen para mostrar los platos disponibles o diarios. se tienen 3 métodos de inicio de sesión: Inicio por medio de Email y contraseña, verificación por Google y verificación por Facebook. Para que todos estos métodos tengan efecto al tocar sus respectivos botones, se debe usar el objeto de FirebaseAuth “mAuth” ya que es lo que permite

entregar al usuario actual para que se lo pase por los métodos de verificación de tanto el método manual, Google y Facebook. En si cada usuario manda un Token especifico hacia Firebase lo cual permite saber que es un cliente tratando de acceder a la aplicación. Aunque los tres métodos sean en cierta manera distintos, deben seguir esta regla de verificarse con el objeto de mAuth ya que esto es lo que conecta con el servicio de autenticación, una vez que el login es exitoso se actualiza el nombre, ID y perfil del usuario y se lo pone por medio de un HashMap en la base de datos de Firebase como usuarios que usan la aplicación. Google realiza el adicional de que se invoca al elemento visual de su API para escoger la cuenta que hará login, es decir se busca en el dispositivo las cuentas vinculadas y se permite el acceso al autenticar con esa cuenta y enviando a FirebaseDatabase esa información. Facebook es un poco más complejo debido al hecho que al no ser tan familiar con respecto a Google, Facebook o meta necesita que se declaren permisos específicos en la página de Meta para desarrolladores para poder usar sus recursos y en especial por usar la información de sus usuarios. Este asunto fue tan complejo que se podría decir que de cierta manera demoro al desarrollo del proyecto en sus etapas más tempranas ya que se tuvo que investigar de manera amplia con FirebaseAuthentication y Meta para usar todas las implementaciones necesarias, definir en el código tokens de usuarios específicos que se envían a Facebook y usar al mAuth para otorgar el usuario a Firebase y otorgarle el acceso al resto del programa. Cabe mencionar que este método funciona ya que, al ser una aplicación móvil, los propios recursos de Facebook enlazan al cliente con la aplicación de Facebook para que este de acceso de su información personal para poder ser utilizada en la aplicación. Si bien lo máximo que se hace con la información de los usuarios es identificarlos por su nombre en el resto de la aplicación, esto es importante permitir el acceso ya que, si se rechaza, Meta es muy restrictivo con la info de sus usuarios.

En cuanto al método de personalizado, se hizo de una manera sencilla pidiendo un mail y contraseña que son verificados por mAuth y dan acceso al programa. Lo importante aquí es como se define o crea un usuario y para hacerlo se lo debe realizar por medio de la actividad Registro de la cual en la actividad actual se ofrece un botón sin fondo para acceder a esa porción del proyecto. Esta actividad en cuestión de forma es simple ya que da tres áreas para que los usuarios llenen y que puedan registrarse siendo estas: Email, contraseña y nombre de usuario. Esto último se hace para tener una manera más personalizada de tratar al cliente . Este es un proceso parecido al de FirebaseDatabase ya que una vez que se registra, se debe hacer conexión con el servicio, comprobar que la señal enviada de registro tiene sentido en cuanto a quien la envía y se guarda en la base de datos al nuevo usuario. Una vez que pasa esto se lleva al cliente de nuevo a la actividad login ya que para tener un nivel más de seguridad, se pide que ingrese su mail y contraseña nuevamente para comprobar con el nuevo registro en la base de datos y dar acceso si es necesario. En sí, lo anteriormente mencionado es lo más importante que hace la aplicación ya que es la puerta principal al resto del programa y guarda la info del usuario en el sistema de Base de Datos de Firebase. Cabe mencionar que para el uso del nombre del usuario tanto en el Menú Principal como en la creación de los objetos ordenes, se pasa la info como un extra en el intent que manda esta actividad hacia “QRActivity”. Finalmente, se implementan “Failsafes” en caso de que ocurra un error al momento de hacer Login y que la aplicación no se cierre de manera aleatoria, es la definición de un tema importante como lo es la captura de errores que los mismos métodos inherentes en las implementaciones de Firebase, PlayServices y Facebook.

7. ACTIVIDAD DE QR

La lectura de los códigos QR puede decirse que es el proceso más importante que se hace en el código ya que si no se efectúa bien o se lo hace con errores, simplemente no se podrá acceder al programa. Para hacer esta tarea se debe usar al menos tres elementos: CamaraX, MLKit y ZXing. CamaraX ya que permite a la aplicación acceso a los elementos del hardware de la cámara, es decir permite usar la cámara posterior del dispositivo. Para poder acceder a ella, el usuario en su primer uso debe otorgarle permiso a la aplicación del uso de la cámara. En el código, se tiene que setear correctamente al objeto ProccessCameraProvider para que habilite el uso de la cámara posterior y que se inicie a los objetos de ImageAnalisis para que empiece a buscar el código QR. Esto es otorgado por el uso de MLkit, una implementación de Google que ayuda al escaneo no solo de códigos QR, sino que también de códigos de barras. Esta actividad inicia como se indicó cuando el usuario le da acceso a la cámara, se obtiene la info o conexión con la cámara trasera y se inicia la actividad de analiza del ImageAnalisis donde la cámara buscara los patrones definidos por los códigos QR. Una vez que logre leer el contenido del QR, se lo guarda en un String “URL”.

Antes de explicar cómo se hace específicamente la lectura del QR, debemos saber cómo se crea este para la mesa. Aquí es donde usamos lo que nos da la implementación de ZXing para la creación del QR.jpg. En si este no tiene como contenido al texto de Mesa1.txt sino que se definió que almacene el URL o enlace hacia Firebase que guarda este archivo. el código se usó una vez para crear el URL de la mesa, pero el código se encuentra comentado en el código si se busca crear más QRs con distintas mesas. ZXing permite tomar este String enlace, codificarlo en un QR y guardarlo como una jpg que por medio de del servicio de Storage de Firebase se lo puede guardar ahí directamente. Volviendo a la lectura del QR, lo que se tiene al escanearse el código es un enlace hacia el archivo texto que se usara para setear al menú

principal. Se debe usar un Storage Reference otorgado por Firebase para conectar al Storage del servicio y se debe pasar a este texto al método “getReferenceFromURL(URL)” para poder descargar el contenido dentro del texto. Si esto no ocurre, ya sea porque los servicios de Firebase o Storage no se encuentren disponibles, el programa enviara un Toast que mostrar un mensaje de Error y se regresara al usuario de vuelta a la actividad de Login. Si bien esto es una contingencia no muy elaborada, existe para limitar el acceso de los usuarios y que estos no tengan problemas hasta que se restablezcan los servicios en Firebase. Si la descarga es exitosa, el contenido del texto se almacena en un String que se pasar junto al nombre del usuario como un extra en el intent hacia la actividad de Menú Restaurante ya que en esta actividad se puede cargar adecuadamente el contenido encontrado en la línea de caracteres.

8. ACTIVIDAD DE MENU RESTAURANTE

El “Menú Restaurante” es la actividad principal por así decirlo de la aplicación, para que todo tenga sentido debemos explicar todo lo que se hace desde que se recibe el intent de QRActivity hasta lo que los usuarios deciden mandar su orden. Todo empieza con la captura del extra-QR del intent, el cual hasta el momento es una cadena de caracteres que contiene todo el contenido encontrado en el archivo de texto almacenado en el Storage de Firebase. Este texto se formateo de manera que todo el contenido se encuentra en todo el texto que posea un “[ ]”. Si el contenido no se encuentra almacenado entre esos elementos, el programa no leerá el contenido y varios elementos pueden verse omitidos por error. Existen tres elementos importantes que el carácter “%” divide, el nombre del restaurante, Numero de Mesa, Lista de platillos, Lista de Extras y Lista de Bebidas. Lo que se hace es que se crean varios elementos String que guardan estas partes del texto. Las listas de platos se almacenan en un proceso similar al anterior donde todos los platillos se encuentran entre los caracteres “< >” y donde cada elemento o platillo se separa

con un “-“ . Aquí, se debe realizar un for donde cada parte del plato, dividido por un “/” como su: Índice, Nombre, Descripción, Precio e Imagen se guardan en String temporales para luego añadirse a un objeto tipo “Platillo” que permite la creación de este objeto y su adición a una lista de tipo “List<Platillo> platillos”. Un proceso similar ocurre con las listas de Extras y Bebidas ya que su única diferencia con respecto a los platillos es que se usa “{ }” y “( )” para delimitar hasta donde cada elemento pertenece. Estos se guardan en Listas de objetos “Platillo” y una vez que se obtienen todos los platos encontrados en el texto, se puede dar paso a la creación del Objeto MenuPrincipal que contiene los siguientes atributos: Nombre del restaurante, Numero de mesa, List<Platillo> platillos, List<Platillo>extras y List<Platillo>bebidas. Este objeto es lo que representa al menú principal y de lo cual el usuario tendrá acceso por medio de los recyclerViews que componente el diseño de la Actividad. Antes de explicar todo el funcionamiento de esos elementos, es meritorio explicar los componentes del diseño implementado. Se está usando un ConstrainLayout donde se puede ver el Nombre del Restaurante, Numero de Mesa y Nombre del Cliente. Se creo un diseño que le sigue que contiene un ScrollView (View que permite hacer un scroll para poder ver todo el contenido de la actividad) especializado para almacenar RecyclerViews ya que si se usó uno normal cuando los usuarios hacen alguna acción para ver el contenido, el programa presentaba fallos en términos de clipping o como se volvía más complicado desplazarse ya que si no se realiza desde el lugar indicado la actividad se trababa. El ScrollView especializado en RecicleViews permite un movimiento más sencillo y fluido para los usuarios y por eso se decidió su uso. En el diseño se definieron tres LinearLayouts para las tres listas de objetos y cada uno de estos contienen una sección donde se especifica el nombre de la lista y el recyclerView correspondiente.

En la estructura del diseño, se añadió tres elementos importantes en el último LinearLayout de la actividad: PrecioTotal, Platos y el Botón para realizar el checkout. Volviendo al contenido del código y las acciones que este realiza, una vez que ya se tiene el objeto de MenuPrincipal, se debe llenar el contenido de los recyclerViews para mostrarlos visualmente. Para esto usamos la clase PlatilloAdaptador al cual le pasamos la lista de elementos del objeto Menú. Esto se lo hace en la actividad de UIMenu donde se deben inicializar tres adaptadores para cada una de las listas. En cada adaptador debemos mandar el LinearLayout de cada lista, un LinearLayoutManager, el recyclerView apropiado y la lista del objeto MenuPrincipal. Esto permite al adaptador el acceso especifico a la lista definida, obtener su Nombre, Precio e Imagen, añadirlos a cada fragmento correspondiente e inflando el recycler como se debe. Este proceso que se realiza en el adaptador es un poco más complejo de lo que parece ya que se debe definir ciertos aspectos como lo sería la descarga de Imágenes ya que los objetos Platillo no tienen un atributo Bitmap que guarda la imagen ya que además de no permitir el paso del objeto por medio de los intents, el proceso serio más complejo al definirse esto en el adaptador. Lo que se hace es que cogemos el valor de Imagen que es un nombre, lo que se hace es que en esta sección del código armamos un nuevo String con el siguiente formato “ ‘Comida/’ + nombre de la imagen + ‘.jpg’” ya que debemos usar de nuevo un StorageReference, pasarle esta referencia al Storage de Firebase para que sepa en que carpeta y que nombre del archivo se deba seleccionar para descargar. Si se obtiene esta referencia se inicia un proceso de descarga desde el servicio lo cual almacena la imagen en un bitmap el cual es seteado como la imagen principal en el ImageView del Fragmento del platillo.

Lo anterior describe como se carga cada elemento de los fragmentos basado en los atributos de los objetos Platillo, estas acciones en el adaptador permiten tener nuestros recyclerViews llenos de los ítems del restaurante y como se usan tres adaptadores en la actividad de Menú Restaurante, esto nos permite llenar los recyclerViews de las tres listas que componen al objeto de MenuPrincipal. Algo que se hace en la clase del adaptador es que cada vez que se haga click en un ítem del recycler, se inicia la actividad de "DetallesPlato" donde se pasa como extra el objeto Platillo que por ser serializable no presenta problemas en ser enviados con un intent. Sin embargo, ya habrá tiempo en la sección de DetallesPlato para hablar a fondo de lo que ocurre en esa actividad, lo importante es saber lo que esta actividad envía de regreso a la actividad del Menú Restaurante.

Cabe mencionar que como una variable local se declararon dos códigos de activación , esto se hace con el objetivo de tener un onActivityResult y sería conveniente hablar un poco más de esto. se crean dos códigos: "CODIGO_PARA_ORDEN" y "CODIGO_PARA_SUBTOTAL" Esto se hace con el objetivo de obtener algo de las actividades "DetallesPlatillos" y "CheckOut". Como se mencionó antes, cuando volvemos de regreso de la actividad de DetallesPlatillo, volvemos con un objeto OrderItem y con un mensaje o Código parecido a los establecidos en la actividad principal conocido como "RESULT_OK" esto se hace para que en la función de onActivityResult se intercepte esta señal si los códigos de envió y recibo son los adecuados. Lo que se hace aquí es que ese objeto de OrderItem se carga en una variable local, se añade a la lista de objetos OrderItem, se saca el valor de ese platillo, así como la cantidad especificada en la actividad y la lista se usa en el constructor del objeto Order, si ya existe este objeto, simplemente se actualiza su lista por medio de la función "setOrdenes" y el precio total por medio de "setPrecioTotal" del objeto. Cada vez que se regresa de la actividad de DetallesPlatos, se realiza esta actualización en el contenido de la orden y visualmente se puede

ver en el diseño ya que los textField se actualizan con los nuevos valores de cantidad y precios. De igual manera , se invoca por medio de los adaptadores el método de “Actualizar” lo cual manda al adaptador la cantidad actual que se ordenó del platillo que se añadió para que los usuarios puedan ver de manera más visual que elementos nomas están en su orden, así como su cantidad. Una vez que el usuario decide que es suficiente para su orden, debe hacer click en el botón derecho de su pantalla ya que permite que la orden se pase como un extra en el intent hacia la actividad del Checkout. Es aquí donde igualmente se incluye el código de “CODIGO_PARA_SUBTOTAL” definido anteriormente ya que cuando se llegue al checkOutActivity, el usuario puede editar la cantidad de platos en su orden, esto debe verse reflejado si el usuario decide volver a Menú Restaurante y añadir otros platos más. Lo que se hace es que checkOutActivity mandara un “ResultOK” en su intent por lo que en OnActivityResult si se capta el código de envío y recibo adecuados, se mandara igualmente una señal por medio del Código de Actualizar del adaptador para que los platos afectados muestren los cambios realizados en la actividad de checkout y no tengan confusión o asuman que su plato no fue actualizado con respectó a sus propias acciones. De igual manera, si el usuario vaciara por alguna razón el contenido de su orden en checkout, este le regresa inmediatamente a la actividad del Menú y todos los ítems de las tres listas se ven actualizadas correctamente y el usuario no puede realizar un envió de orden ya que se pondrá al objeto como nulo y se necesitara añadir un nuevo plato para crear una nueva orden. eso sería en grandes rasgos todo lo que se hace en el Código de la actividad de Menú Restaurante, por así decirlo es el centro de la aplicación ya que es la que permite él envió de información hacia el checkout y es a donde siempre se redirigirá si ocurren cambios tanto en DetallesPlato como en Checkout. Además, es lo que visualmente muestra el contenido almacenado en el archivo de texto único para su mesa y permite tener un medio para que el restaurante pueda identificar la fuente de la orden que reciban ellos.

9. ACTIVIDAD DE DETALLES PLATO

Los detalles de los platillos son fundamentales para el programa ya que es la manera principal que tiene el usuario de interactuar con el programa y de crear al objeto de la orden. Como se describió en la sección de Menú Restaurante, uno puede acceder a esta actividad y a su respectivo diseño por medio del adaptador de cada recyclerView, esto ya que se envía un intent con un Anexo 4e objeto Platillo. La actividad presenta 4 secciones: Imagen del platillo, Nombre del platillo junto al precio de este, Descripción del platillo y un área de escritura donde se almacena todo lo que el usuario escriba. Esta existe con la intención de que el usuario pueda hacer requerimientos, alerte al restaurante de una alergia a los ingredientes de la comida o solo indicarle que añada o quite algo de su orden. En esta actividad hay un contador que representa el número de platos que el cliente desea, esta esta seteada para iniciar en 1 y que se pueda aumentar o reducir la cantidad del mismo platillo. Sin embargo, se hizo un cambio en la lógica para que el mínimo valor siempre sea uno y que no se pueda pedir 0 o valores negativos. Esta acción de modificar la cantidad modifica al precio que se muestra en el botón de "añadir n platos" a la orden y permite a los usuarios saber la cantidad y precio que pagaran por ese plato en especial.

En términos de lo que se hace a nivel del Código, una vez que se configuren los valores, es decir, definir la cantidad y las instrucciones para cada plato, se debe hacer una conversión del objeto Platillo hacia el objeto OrderItem ya que las principales diferencias es que esta tendrá el valor de cantidad definida y las instrucciones. Esto es importante ya que como se especificó en la parte de Menú Restaurante, esta actividad debe enviar un objeto OrderItem con un código RESULT_OK devuelta a la otra actividad para actualizar las cantidades en los elementos de los recycleViews y añadir un objeto OrderItem al objeto Order . Es una actividad sencilla, lo

más complicado podría decirse que es lo de la implementación de los cambios al añadirse o eliminarse la cantidad de ese ítem que se desea, pero tanto en código como en concepto, se realizan actividades sencillas y dentro de todo el ámbito del programa, es de las partes más sencillas de entender y manipular.

10. ACTIVIDAD DE CHECKOUT

La aplicación se formuló con la idea de que se divida en 4 secciones: Autenticación, Lectura de QRs, MenuPrincipal y Checkout. Esta podría definirse como la última actividad sin contar con las extras como lo seria detallesPlatillo o Registro ya que podría decirse que esas actividades están autocontenidas en las secciones mencionadas. Checkout tiene la función de mostrar visualmente los elementos de la Orden hecha por el usuario ya que se implementa como en Menú Restaurante un RecyclerView que cargar los elementos del objeto por medio de su atributo de List<OrderItem>. Esta sección existe por medio de un adaptador conocido como CheckoutAdapter que toma al objeto Order, obtiene la lista de este de OrderItems y los define en fragmentos donde se puede ver: La cantidad especificada por el cliente, el nombre del platillo, si existe alguna Instrucción especial y el precio . Para cada fragmento existe un listener o listeners ya que como en DetallesPlato, se usa un contador para definir la cantidad de comida que se quiera añadir o restar y los listeners y se encargan de captar cualquier cambio y hacer un notify a los elementos correspondientes en la lista de la orden. Estos cambios afectan los valores de cantidad de cada platillo, así como el precio de cada ítem aumentando o disminuyendo según las acciones del usuario. Esto era importante también enviar una señal de vuelta a Menú Restaurante si el usuario regresaba ya que es importante mostrarle los cambios realizados a su orden . Si se aumenta o disminuye se envía por medio de un listener el objeto Order actualizado y este se envía como extra de un intent hacia la actividad Menú Restaurante donde se actualizan los campos de paltos totales, precio total de toda la orden y las cantidades

individuales de los platillos que se encuentran en este objeto Orden. Si se elimina un ítem de la lista de Order, se notifica al objeto, se manda de nuevo como intent y se actualizarán los valores correspondientes. Si la lista entera está vacía, representa que se han borrado todos los ítems de la orden por lo que se asume que la orden se borra. Esto manda al objeto Order como extra con el código "Vacío" y cuando se recepte en la actividad principal o el Menú Principal, se hace por así decirlo un reinicio o todos los platillos vuelven a su estado inicial, la orden estará en nulo y las cantidades totales, así como el precio total estarán en 0.

En términos de pagos, se procesa el subtotal de la orden el cual sería el pago total presente en el objeto Order y que se configuró al momento de aumentar o disminuir el número de ítems de cada platillo o incluso si se eliminó de la orden. Este es recogido en el código para sacar su valor del 0.12 o 12% (valor del IVA en Ecuador) y el de servicio (0.10 o 10% del subtotal de la orden) ya que con esto podemos configurar correctamente el precio total al sumar estos 3 valores. Cabe aclarar que para que los cambios en estos valores se reflejen cuando se aumenta o se resta las cantidades, el usuario debe hacer click al botón de "Confirmar Orden" ya que esto permite que todos los valores se reinicien a 0, se recoja los nuevos valores subtotales del aumento o resta de platillos y se calcule un nuevo valor total basado en los cambios que el usuario haga en la orden. Esto en si es una alternativa que se tuvo que optar ya que el objetivo original era que los cambios por medio de un listener actualizaran los valores totales en tiempo real, pero existieron muchos errores con los valores obtenido ahí. En esta sección también se tiene que utilizar dos adaptadores en total: del recyclerView que muestra los objetos de la orden y uno especializado para un spinner que debe tener una imagen y un nombre. Esto ya que el adaptador de spinners se utiliza para uno que almacena los métodos de pago disponibles en el programa, así como otro que les da la opción a los clientes de incluir una factura o no en su orden.

Ya que lo mencionamos, tenemos que explicar la sección de métodos de pago utilizados en la aplicación y que se ven implementados en la parte del Checkout. , estamos usando dos métodos de pago para que los clientes en cuestión puedan procesar sus órdenes: método de efectivo y PayPal. En el efectivo es muy directo ya que la idea es que el restaurante procese el pago directamente ahí. Lo que se hace es que se manda a una nueva actividad llamada "Resumen Orden" un StringBuilder que contiene un resumen de la orden, el nombre del restaurante, así como el nombre del cliente y la mesa a la que este pertenece. La siguiente función se aplica en ambos métodos de pago, pero se lo hace directamente si se escoge el efectivo, lo que ocurre es que este mismo StringBuilder que estamos mandando se lo adapta en un arreglo de strings para poder ser mandado por un número telefónico en formato de mensaje. Estos en Android tienen un límite de caracteres de 160, por eso se debe implementar el uso de un arreglo de strings para que un mensaje, sin importar la cantidad de platos que tenga, pueda ser enviado a ese número telefónico. En este caso de prueba se usa mi número de celular, pero la idea en si es que sea el número telefónico correspondiente a la recepción o counter del establecimiento ya que ellos al recibir esta información pueden procesar la orden por el número de mesa y entregar los platos solicitados, así como tener un registro de cuantas ordenes pidió la mesa 1 y los valores a cobrar. Aún falta un método que permita guardar las ordenes generadas por los usuarios en Firebase, pero como está estructurado el código, no se descarta esta acción en futuras versiones.

Como se menciona antes, este proceso de envió de mensajes se tiene que implementar directamente al usar el método de efectivo en la aplicación, sin embargo, esto es un poco diferente en la actividad alternativa conocida como "Resumen Orden Pay". Esta actividad sigue el mismo sistema de recibir un StringBuilder de la orden del Checkout como se lo hace con el método de efectivo, lo que diferencia a esto es que no se procesa ni envía mensajes hasta que el usuario haya procesado el pago de la orden por medio de PayPal. Esto se lo hace con un

botón otorgado por la misma implementación del SDK de PayPal. Como se trata en si de pruebas, se tiene que usar a PayPal en modo Sandbox. Esto significa que se procesa dinero falso con el solo objetivo de probar la funcionalidad del servicio del SDK, igual para esto se debe setear en la página oficial de desarrolladores de PayPal algunas cuentas en modo Sandbox para darles un valor que pueda ser manipulado en las pruebas al igual que falsas tarjetas de crédito. Una vez que el usuario da click al botón, s ele pide iniciar sesión en su cuenta de Sandbox y se le permite pagar por la orden. Ya que se procesa el pago y se hace la transacción monetaria, la pantalla de la actividad vuelve a cambiar para ocultar el botón de PayPal y mostrar al igual que el método de efectivo un resumen de la orden y enviar en un arreglo de strings la orden en formato de mensaje. Por eso se debía explicar esta diferencia entre el método de efectivo y PayPal ya que en esencia se hace lo mismo, pero en la última primero se debe confirmar el pago por medio del API del servicio para proceder. Ambas opciones de pago tienen dos botones, uno para regresar al checkout para acceder a la edición de sus órdenes y otro que los lleva directamente al Menú Principal donde todo se reinicia a 0 y el usuario en cuestión es capaz de realizar nuevas órdenes hacia el restaurante. En si pudiera decirse que en lo que concierne a la sección, aquí se acaba la parte del Checkout y de la aplicación ya que todo se reinicia hacia el Menú Principal para procesar más pagos. Sin embargo, aún se deben explicar los problemas encontrados en el desarrollo de la aplicación , así como las modificaciones y cambios que se podrían hacer en nuevas implementaciones.

11.ANÁLISIS DE RESULTADOS.

Con la aplicación ya finalizada, podemos mencionar los resultados que proporciona. En si pudiera decirse que esta versión (7ma versión de todo el programa) cuenta como la versión beta de la aplicación ya que, si bien faltan muchos cambios y arreglos, así como un port hacia el sistema IOS, es lo suficientemente funcional para ser usada ya que las 4 secciones principales cumplen con su propósito, se toman las ordenes e interacciones de los usuarios y se establece un puente de comunicación entre el usuario y la cocina o counter que recibe las ordenes de los clientes. Se considera que la aplicación está bien estructurada en el sentido que tiene un claro ciclo de vida y que ayuda o trata de mantener a los usuarios con el mayor control posible de sus órdenes, así como mantenerlos en un camino entendible y conciso entre las distintas actividades del programa. El código en términos generales está bien estructurado y hace un uso coherente de los distintos objetos proporcionados por las clases creadas, así como elementos provenientes de las implementaciones que se tiene como lo sería de la CamaraX (hardware de la cámara) así como el de los servicios integrados como Facebook, Google y PayPal.

Puede describirse a la aplicación como un buen producto de pruebas que, si bien aún debe cumplir muchas implementaciones, reglas y condiciones para ser considerado ser comerciable, como prototipo es un muy buen proyecto y que permite como una guía solidad para futuros cambios y corrección de errores. El estilo visual, así como los colores y fonts de texto usados son adecuados al considerar que es una aplicación destinada al uso en restaurantes o establecimientos de comida. Para ser considerado un proyecto integrador que pone a prueba todos los conocimientos que se han recibido durante la carrera, es una muy buen proyecto ya que estamos usando varios conceptos vistos desde el inicio hasta el final. Algunos ejemplos pueden ser la programación orientada a objetos que fue crucial en el programa ya que con esto

se pueden conceptualizar y concretar a las órdenes y platillos, así como el diseño de layouts y programación en el lenguaje de Java que se vio en uno de los últimos cursos de la malla. En lo posible, trata de ser un proyecto que honra a los conocimientos adquiridos, así como posiciona nuevos conceptos y sus implementaciones respectivas.

12. PROBLEMAS/DIFICULTADES

El desarrollo de la aplicación no vino sin dificultades ya que la versión más sólida que se tuvo nació de prueba y error durante un periodo de análisis y desarrollo extenso. En un principio, podríamos decir que fue complicado la implementación de nuevos recursos otorgados por los distintos SDKs ya que muchos necesitaban ser configurados tanto en el código como en las páginas de desarrollo como lo sería Meta, PayPal, etc. En el primer caso fue uno de los problemas más consistentes ya que se tenía que declarar la obtención de metadatos del usuario para poder hacer login, así como varios tokens que permiten esta interacción con la app de Facebook. En un principio incluso se usó viejas implementaciones del SDK que por ya estar deprecadas no permitían el acceso a su respectivo recurso y como se tenía que investigar que versión era la más actual, se tenía que hacer un trabajo de investigación, así como usar las nuevas funciones que en las propias páginas de los recursos se declararon. Esto provoco en ciertas ocasiones un retraso según lo planificación establecida ya que tenía que usarse correctamente estos recursos para permitir el posterior progreso en el desarrollo. De igual manera, nacieron varios problemas con la visualización de los precios ya que en un principio cuando los usuarios escogían los ítems de los recyclerViews, estos valores no se iban añadiendo en la actividad principal, así como la cantidad escogida. Esto tuvo que ver con el OnActivityResult ya que al principio no se usaban códigos como los descritos en la sección que explica la actividad del Menú Principal además de que se tuvo que ver la manera de

diferenciar cuando se regresaba de “DetallesPlatillos” y del “Checkout” ya que en ambos casos se debían usar distintos métodos para que el usuario no fuera a confundirse con respecto a su orden. La sincronización fue un problema que estuvo envuelto aquí ya que cuando el usuario regresaba de checkout por medio del botón de regreso propio de los dispositivos, los cambios hechos en la orden no se veían procesados correctamente ya que solo se consideraban los valores originales que se enviaron hacía en un principio. Todo esto tiene que ver en la manera en la que se recibían los datos del intent para los resultados ya que se necesitaba por medio de una función de actualización hacer los cambios respectivos en cada uno de los ítems de los recyclerViews que fueron alterados en chekout, pero la manera de hacer esto fue complejo por decir lo mucho ya que se tenía que volver un objeto OrderItem (ítems de la orden) hacia un objeto Platillo que es lo que está presente en las 3 secciones del Menú Principal.

Otro tipo de problemas fue la manera en la que los mensajes eran enviados hacia el counter ya que al principio no se tenía conocimiento de que había un límite de caracteres de cada mensaje y cuando se enviaba la orden muchas veces no se recibía nada por la longitud de esta o no venían ítems que estaban presentes en la orden. Lo que se tuvo que hacer fue que para enviar el mensaje se tienen que establecer como un arreglo de strings para que el mensaje procese hasta los 160 caracteres y que cuando se haga click en estos, se muestre la orden completa.

La sincronización de valores, el envío de mensajes y las implementaciones de los SDKs fueron los temas más complejos y que presentaron dificultades. También lo fue el uso de las interfaces en Checkout que ayudaban a alterar los valores de la orden. Sin embargo, para cada uno de estos problemas y dificultades se logró encontrar una solución alterna que ayudo a que la aplicación pudiera seguir siendo desarrollada y cumplir con los requisitos impuestos en las entregas . A pesar de estos arreglos, hubo otros errores que surgieron durante el desarrollo de

la aplicación que no pudieron ser resueltos en la última versión del código ya que, por cuestiones de tiempo, no se pudo trabajar en estos asuntos.

13. ERRORES COMETIDOS

Si bien el trabajo del desarrollo de la aplicación dio como resultado un programa estable , esta no está sin errores en la misma que necesitaran de soluciones futuras o “patches”. Uno de los más evidentes es el del uso de las imágenes de la aplicación. ¿Qué significa esto? Bueno cuando iniciamos los elementos o ítems de los recyclerViews estamos descargando directamente de Firebase las imágenes de cada ítem lo cual en si representa un consumo de mbs de nuestro proyecto. La cosa está en que cada vez que se requiere de imágenes se descargan de nuevo estas del servicio de Storage, lo cual representa un consumo innecesario de datos o recursos de esta parte y si consideramos que muchas personas deben tener acceso a esta aplicación en distintos establecimientos, el consumo por la descarga innecesaria será un problema en cuestión de tiempo. Este exceso de descargas es un problema en el plan actual que se tiene en Firebase ya que un consumo masivo puede resultar en que el proyecto incremente de precio y por ende limita a que varios negocios quieran usar este programa por el costo que representaría. Una posible solución es la creación de cache local y temporal en la aplicación una vez que se descargue por primera vez las imágenes ya que se puede ahorrar de manera inmensa el consumo de datos por usuario y por ende representar un costo más estable para los negocios que estén interesados en usar los servicios de la aplicación.

Otro error que se encuentra presente en la aplicación es que no se manejan adecuadamente algunas excepciones. Esto puede explicarse por cuestiones de tiempo, sin embargo, su manejo debe ser implementado para que en cualquier problema los usuarios puedan tener al menos una explicación de estos. Esto es más evidente en la actividad de lectura de QRs ya que si se trata

de leer un código que no posea la dirección del texto “Mesa1.txt”, simplemente la aplicación colapsa y saca a los usuarios de esta. Este es un problema claro y la corrección de este error en retrospectiva y comparándolo al código en general es de una resolución más simple que los otros problemas. Ya se ha hablado de errores presentes en el código , pero falta de las cosas que no están presentes. Uno de estos puede ser que cada vez que los usuarios realizan una compra de ítems, no se mantiene un counter de cuantas ordenes ha hecho el usuario así como no se almacenan estas en Firebase cada vez que ocurren lo cual en retrospectiva y considerando que a cada negocio le gustaría tener disponible esta información, se puede entender porque sería necesario un manejo más adecuado de esto y de involucrar a Firebase de una manera más especial al guardar la información correspondiente de sus actividades. Aparte de los temas discutidos, la mayor cantidad de errores pueden encontrarse más en el diseño y aspectos visuales del código . Sin embargo, estos asuntos pueden arreglarse de manera más fácil en comparación con los problemas previamente descritos ya que solo se necesita tiempo en realidad y no de soluciones más complejas como lo serían las previamente descritas.

14. TRABAJO FUTURO

SEVA como aplicación sigue una estructura sólida y coherente y ya se han mencionado algunos de los errores que se encuentran presentes, así como los conceptos que no se implementan aquí. Esto demuestra que, si bien ya se tiene una versión solida del programa y aplicación, aún existen muchos cambios, implementaciones y posibles añadidos al código que podrían ayudar tanto a los usuarios como a los negocios. Una opción que se presento fue que en los archivos de texto que representan Mesas se ponga más de un restaurante para que los usuarios puedan tener una mayor variedad de establecimientos que escoger. Este enfoque sería posible si consideramos que está pensado para centros comerciales que tienen un área de comidas con varios establecimientos. Esto podría servir para los usuarios ya que solo tendría que hacer su

orden en la mesa e ir a recoger la comida cuando el negocio les notifique de alguna manera. Debido a la forma en la que está construida la aplicación y como en realidad solo se necesita alterar el archivo de texto, esta es una posible situación que no se descarta ya que puede representar una expansión del concepto original y que permitiría a la gente un mayor acceso a este recurso siendo esto el objetivo principal de la aplicación. Se puede ver en un futuro un port o adaptación del código para los sistemas operativos de Apple como lo sería IOS ya que en si se consiguió una versión funcional y con algunos pequeños arreglos se podría pasar para IPhone y productos de Mac para que el mercado objetivo sea aún más grande de lo que ya sería. Esto en fin de incluir la mayor cantidad de personas posibles para que prueben el programa.

De igual manera, se prevé un mayor uso de recursos de Firebase como funciones especializadas que se pueden crear en la plataforma, así como un mayor uso de la base de datos y base de datos en tiempo real. Los propios negocios estarán encantados de tener información como los clientes recurrentes, la cantidad de ordenes que han hecho junto a una fecha ya que pueden establecer dinámicas con su grupo objetivo e incluso requerir de funciones adicionales que la misma aplicación podría darles. Firebase como un servicio es como un iceberg al usarse solo la punta con respecto a todo el contenido que representa y que ofrece a los desarrolladores de aplicaciones. Nuevas funciones como un counter de ordenes o botones que sirvan como indicadores, guías o puntos de información para los usuarios serían de gran utilidad. Esto ya que, si bien el ciclo de vida de la app está pensado para guiar fácilmente a la gente para realizar su orden, a veces es un poco complicado entender el funcionamiento del programa si uno no es muy diestro con las aplicaciones móviles. Igualmente, añadir ciertos elementos como un spinner que permite traducir el contenido de la aplicación en distintos idiomas serpia perfecto si la persona en cuestión no habla español, ayudando a los negocios y restaurantes para que sus

clientes puedan usar la app y que con esto se tenga ya un modo regularizado de hacer pedidos hacia restaurantes. En si las posibilidades son infinitas ya que el código fue hecho con modularidad en mente lo cual permite añadir como piezas nuevas funciones que mejoran sobre lo ya hecho y que ayudan a expandir el programa.

15. CONCLUSIONES

El proceso de desarrollo de la aplicación fue extenso, implicando el minucioso diseño de las partes visuales y la formulación de lógica en cada parte del ciclo de vida del programa. Se realizó un trabajo extenso de investigación con las diferentes implementaciones y SDKs usados. Como resultado, se obtuvo una aplicación lo suficientemente compleja y que podía considerarse como un prototipo completamente funcional de la idea original. La versión obtenida de esta aplicación no representa el fin de esta, debido a que hay muchas posibilidades de mejorarla y corregir errores que de hecho se presentan.

La aplicación fue planificada como una forma de modernizar el flujo de trabajo en establecimientos de comida y restaurantes, tratando un problema de la vida real y que puede resolverse por medio del uso de la tecnología. En Ecuador, falta un largo recorrido hasta que se haga uso de la tecnología en los establecimientos de comida como en otros países como por ejemplo Estados Unidos, Reino Unido y otros países del primer mundo.

Este tipo de aplicaciones puede servir como inspiración para que más programadores empiecen a desarrollar e implementar programas similares que pueden enfocarse de igual manera en restaurantes, bares y varios sitios de distracción, así como en innumerables áreas de servicios donde se pueda utilizar estos procesos de automatización que benefician tanto a los clientes como a los proveedores de servicios.

El trabajo ha involucrado distintas actividades, desde la planificación del ciclo de vida de la aplicación en los diagramas iniciales hasta la implementación de los cambios visuales en la última versión del programa. El desarrollo del código dio como resultado siete versiones del programa, cada una representando distintas etapas y que entre sí poseen diferencias notables. En términos de dificultad, esta aplicación tuvo un nivel moderado debido al desarrollado de la misma en un tiempo relativamente pequeño en comparación al tiempo que este tipo de aplicaciones suele tener en el mundo laboral.

Como ya se mencionó, varios conceptos y temas estudiados durante el transcurso de la carrera fueron abordados por la aplicación. Como ejemplos, tenemos el uso del lenguaje Java, la programación orientada a objetos, desarrollo de aplicaciones y el uso de servicios de red. Su uso en la aplicación demuestra un entendimiento de estos temas y el manejo de estos de una manera entendible y eficaz. “Seva” es una aplicación con una arquitectura orientada a la modularidad, permitiendo un mayor control de cada segmento de esta. Esto permite que la corrección de errores y la implementación de nuevas características no comprometan la integridad de todo el programa. También se refleja esto cuando la app lee códigos QR y la manera en la que estos están almacenados en Firebase. Si se buscara añadir, editar o eliminar un platillo dentro del menú, solo se tiene que editar el .txt que esta guardado en Firebase. El código QR representa la dirección de este archivo en el servicio y no el texto en si por lo que

no es necesario hacer cambios a nivel de código. Estos aspectos son mencionados para resaltar el grado de planificación y diseño por el cual pasó la aplicación durante su estado de desarrollo. Muchos problemas surgieron por este tipo decisiones, pero considerando la actual estructura del código, se puede justificar estas acciones para "afinar" la aplicación y acercándola a la versión deseada.

En conclusión, el desarrollo del código dio como resultado una aplicación robusta, bien estructurada y con un gran nivel de complejidad. Si bien se puede considerar como un "prototipo" de un producto más grande, la aplicación cumple con los objetivos propuestos en la planificación del proyecto y funciona como solución al problema planteado originalmente. En términos de tiempo de entrega, se cumplió con el cronograma establecido en el archivo de planificación. También proporciona resultados visibles y una aplicación totalmente funcional que sigue los parámetros impuestos para este tipo de trabajo. El tutor y el estudiante se desempeñaron de manera excepcional, organizada y con un alto nivel de profesionalismo. Se resolvieron todos los problemas que se presentaron antes, durante y después del desarrollo de la aplicación. Finalmente, se puede decir que todo el proceso del desarrollo del proyecto integrador ha sido un éxito por parte de todos los agentes implicados y que estoy satisfecho de cómo se desenvolvió todo.

16. REFERENCIAS BIBLIOGRÁFICAS

- R. R. Luna, A. S. D. F. Roldán, J. A. M. Arvi, I. A. Galván, and A. E. S. Galindo, "Sistema de Pedidos Por Medio de Códigos QR para minimizar el tiempo al servicio al cliente en el restaurante 'las che alitas' De Tizayuca, 2020 – 2021.," Boletín Científico INVESTIGIUM de la Escuela Superior de Tizayuca, https://repository.uaeh.edu.mx/revistas/index.php/investigium/article/view/7169.

- Android Developers. (2023, January 25). Camerax : Desarrolladores de Android : android developers. https://developer.android.com/jetpack/androidx/releases/camera?hl=es-419

- Cisneros, G. (2023, October 18). Guillecisneros23/Proyecto-Ragnarok-Proyecto-integrador-: Este es un repositorio con el solo fin de Almacenar y documentar El Progreso que se ha hecho para el desarrollo de mi proyecto integrador. GitHub. https://github.com/GuilleCisneros23/Proyecto-Ragnarok-Proyecto-Integrador-.git

- Firebase. (2023a, September 21). Autentica con google en android | firebase authentication. Google. https://firebase.google.com/docs/auth/android/google-signin?hl=es-419

- Firebase. (2023b, September 21). Autenticación de Android mediante El Uso de Inicio de Sesión en facebook | firebase authentication. Google. https://firebase.google.com/docs/auth/android/facebook-login?hl=es

- Firebase. (2023c, September 22). Escaneo de Códigos de Barras | ML kit for Firebase. Google. https://firebase.google.com/docs/ml-kit/read-barcodes?hl=es-419

- Kunneth, T. (2021, November 14). Integrating Zxing Android embedded in a compose app. DEV Community. https://dev.to/tkuenneth/integrating-zxing-android-embedded-in-a-compose-app-5ela

- "¿Por Dónde Empiezo con firebase authentication? | firebase authentication," Google, https://firebase.google.com/docs/auth/where-to-start?hl=es-419.

- C. Chaubey and A. Sharma, “The Integrated Development Environment (IDE) for application ...,” Research Gate, https://www.researchgate.net/publication/368846341_The_integrated_development_environment_IDE_for_applicati on_development_Android_studio_and_its_tools.

- P. Lessel, M. Böhmer , A. Kröner , and A. Krüger, “User Requirements and Design Guidelines for ... - ACM Digital Library,” ACM Digital Library, https://dl.acm.org/doi/10.1145/2399016.2399096.

- S. Umap, S. Surode, P. Kshirsagar, M. Binekar, and Prof. N. Nagpal, “Smart menu ordering system in Restaurant,” International Journal of Scientific Research in Science and Technology, https://ijsrst.com/IJSRST1845447.

- G. L. Intal, J. D. Payas, L. M. Fernandez, and B. M. Domingo, “Restaurant Information System (RIS) with QR code to improve service ...,” Research Gate, https://www.researchgate.net/publication/341691351_Restaurant_Information_System_RIS_with_QR_Code_to_Improve_Service_Operations_of_Casual_Fine_Dining_Restaurant.

- “Add Payment Checkout to an App with PayPal Mobile Checkout SDK.” Developer.Paypal.Com, PayPal Developer, 6 Sept. 2023, developer.paypal.com/limited-release/paypal-mobile-checkout/.

- Developer Dashboard, PayPal Developer, developer.paypal.com/dashboard/accounts.

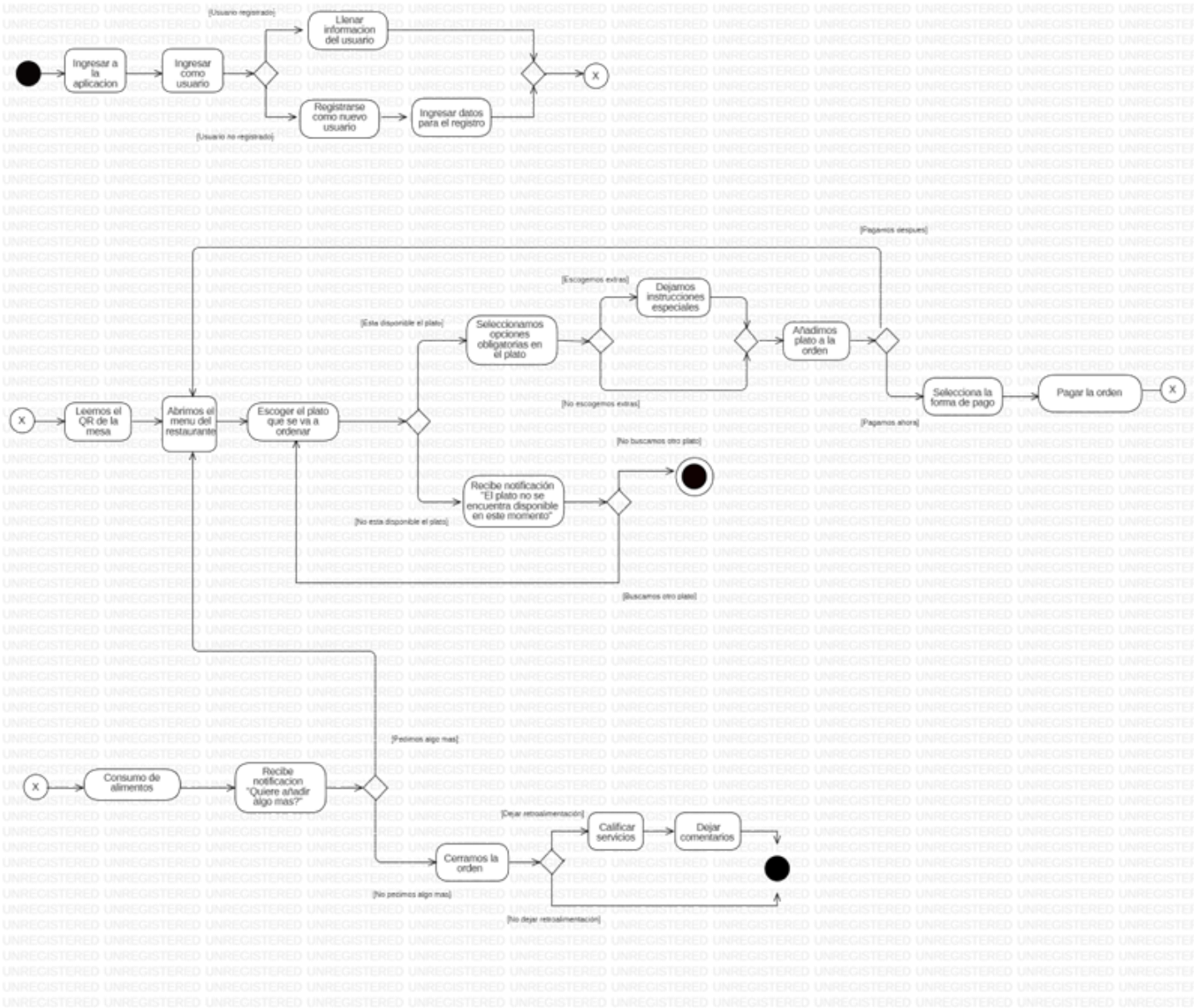
- “Enviar, Recibir y Ver El Historial de Sms Con Android.” *Programación.Net*, Programación.net,programacion.net/articulo/enviar_recibir_y_ver_el_historial_de_sms_con_android_1089.

- “Controles de Números : Desarrolladores de Android : Android Developers.” *Android Developers*, developer.android.com/guide/topics/ui/controls/spinner?hl=es-419.

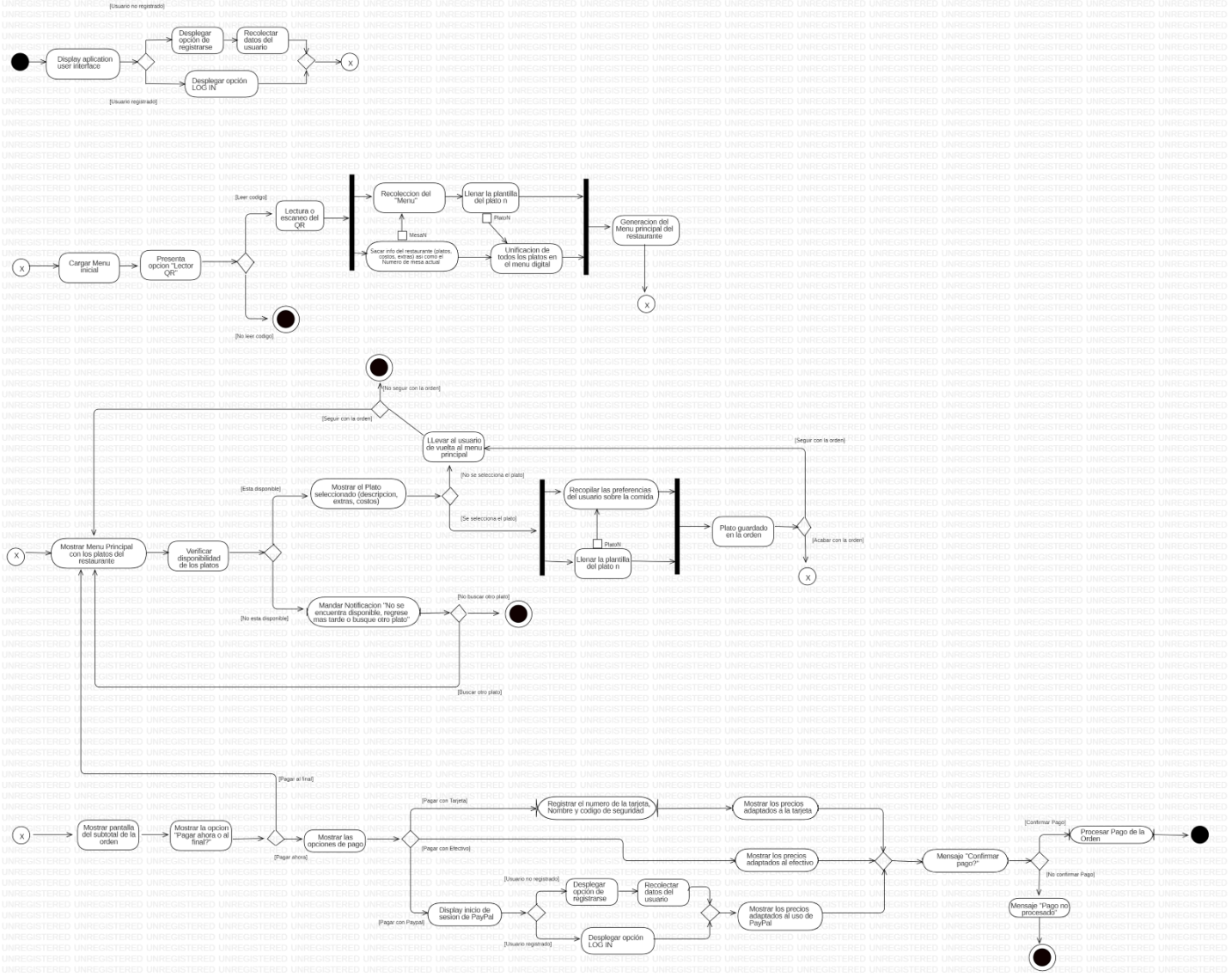
- “Cómo Crear Listas Dinámicas Con Recyclerview : Desarrolladores de Android : Android Developers.” *Android Developers*, developer.android.com/guide/topics/ui/layout/recyclerview?hl=es-419.

- Avery, Helen. “Seva: The Art of Serving from Our Hearts.” *Wanderlust*, 14 Nov. 2017, wanderlust.com/es/journal/seva/.

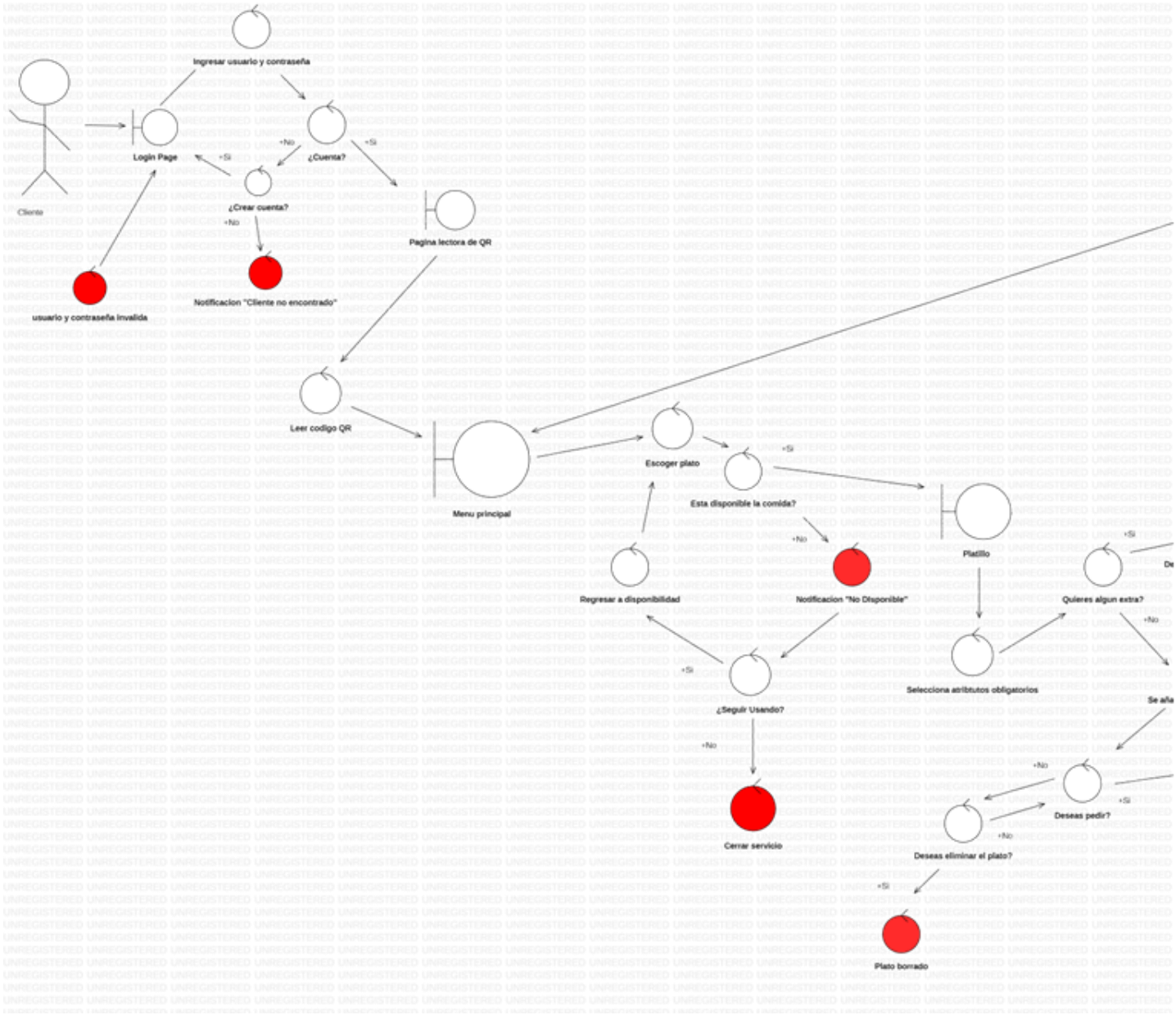
ANEXO 1: DIAGRAMA DE SECUENCIA DE USUARIO:



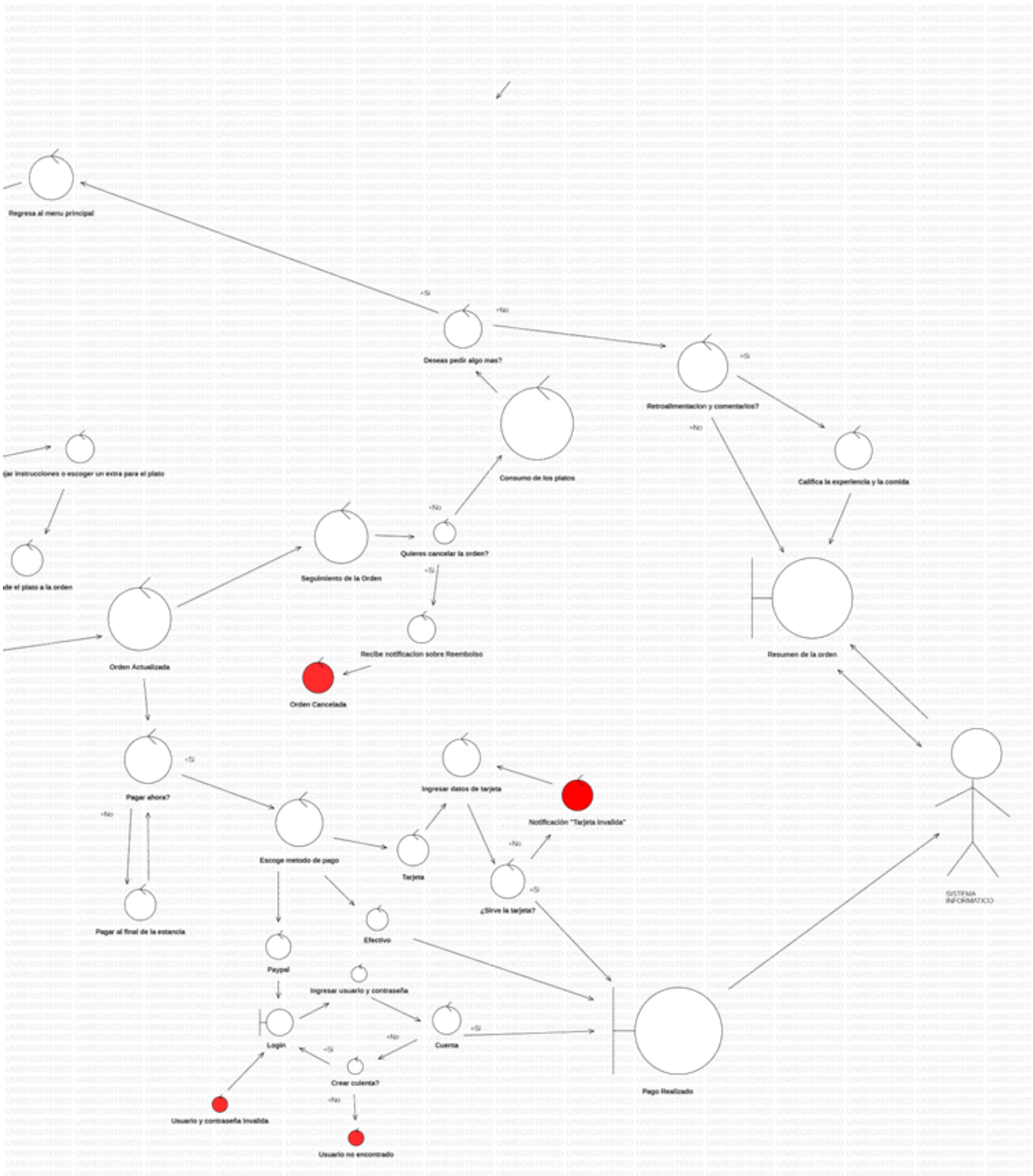
ANEXO 2: DIAGRAMA DE SECUENCIA DEL SISTEMA:



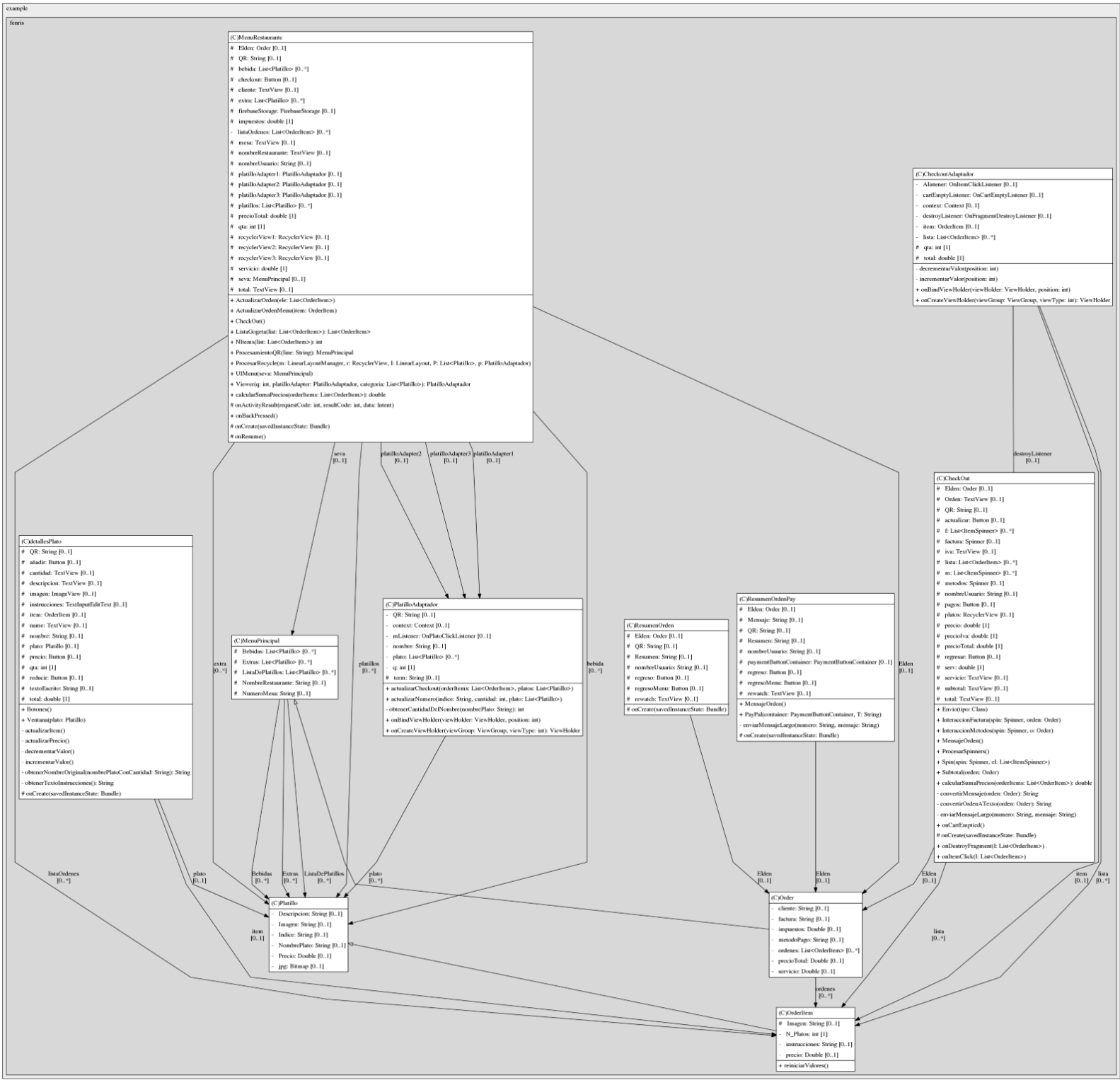
ANEXO 3: DIAGRAMA DE ROBUSTEZ DEL USUARIO PARTE I:



ANEXO 4: DIAGRAMA DE ROBUSTEZ DEL USUARIO PARTE II:

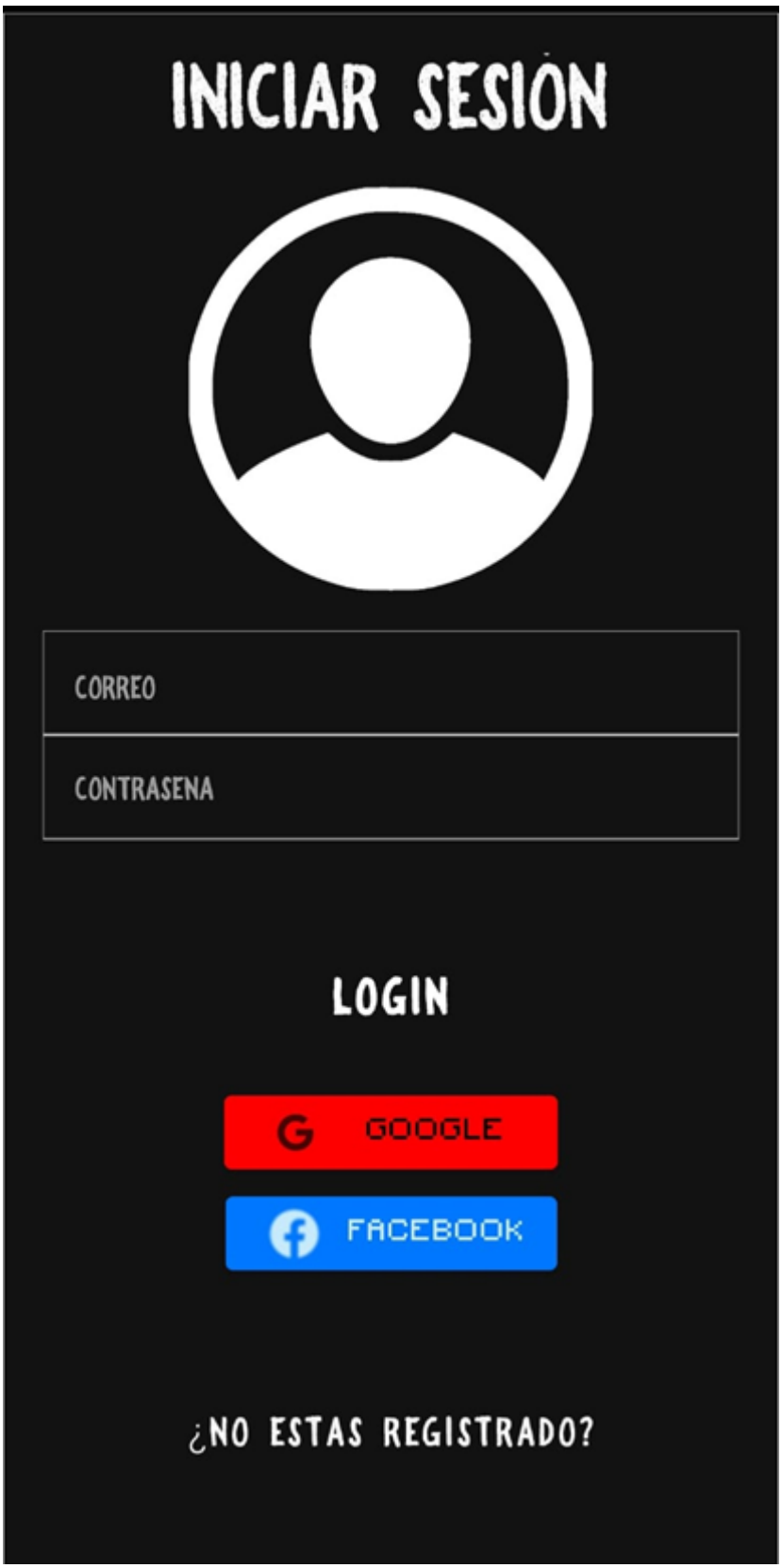


ANEXO 5: DIAGRAMA UML DEL PROYECTO :



ANEXO 6: DISEÑO DEL SPLASH:

ANEXO 7: DISEÑO DE LA ACTIVIDAD "LOGIN":



ANEXO 8: DISEÑO DE LA ACTIVIDAD “REGISTRO”:

ANEXO 9: DISEÑO DE LA ACTIVIDAD “QR”:

ANEXO 10: DISEÑO DE LA ACTIVIDAD "MENÚ PRINCIPAL":

ANEXO 11: DISEÑO DE LA ACTIVIDAD "DETALLESPLATILLO":

ANEXO 12: DISEÑO DE LA ACTIVIDAD "CHECKOUT":

ANEXO 13: DISEÑO DE LA ACTIVIDAD "RESUMEN ORDEN":

ANEXO 14: DISEÑO DE LA ACTIVIDAD "RESUMEN ORDEN PAY":