COLEGIO DE CIENCIAS E INGENIERÍAS

INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN

Entregable 2 del Proyecto Integrador

Tutor: Daniel Fellig Goldvechmiedt

Autor: Guillermo Nicolás Cisneros Villota

Quito – Ecuador

2023

1. Título del Proyecto:

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.

2. Resumen de actividades realizadas:

1. Revisión de la documentación de Firebase

Firebase podría describirse como la columna vertebral para el desarrollo de la aplicación y durante este periodo de desarrollo del prototipo, se hizo una revisión más exhaustiva de la documentación oficial. ¿Esto que implica? Que se ha estudiado aquellos temas que más relevancia tienen para el proyecto e implementado varias practicas y funciones existentes en la documentación de forma practica en el proyecto. Se hará referencia a esto más adelante, pero como ejemplo podemos decir que los procesos de autenticación ya vistos en el código original, así como los nuevos métodos hechos durante este avance se dieron gracias a esta investigación y estudio de los documentos oficiales de Firebase.

2. Investigación e implementación de nuevas dependencias

Esta actividad realizada se entrecruza con los avances hechos en el entregable 1 ya que durante esa actividad y ahora se buscaron dependencias basadas en librerías como mavenCentral que permita ejecutar las funciones mas importantes que la aplicación debe realizar. Esto se muestra en todo el aspecto de la actividad lectora de códigos QR debido a que se tuvo que buscar, entender e implementar dependencias para camaraX y Firebase-mlkit (explicación mas detallada cuando hablemos de la actividad de QR). Esto se hizo en si para las siguientes etapas del proyecto: proceso de autenticación, proceso lectura de QR, implementación de la cámara del celular y conexión a Firebase Storage.

3. Corrección de errores presentes en el primer entregable

El periodo de desarrollo para el segundo avance sirvió al principio para la corrección de errores presentes en la aplicación que tuvieron que ser ignorados para el entregable 1. Esto hace referencia a que en las 3 primeras actividades (Login, Registro y QRActivity) el flujo de entre estas estaba poco desarrollado lo que implica que cualquier actividad errónea de los usuarios podía crear un problema cuando pasáramos de actividad en actividad. Por ejemplo, una vez que se hacia el proceso de inicio de sesión no se podía regresar a esta actividad debido a que no existía un manejo adecuado de las actividades. Esto fue corregido con la implementación de botones y modificaciones en las distintas etapas del ciclo de vida de la app que permitieron crear un flujo de trabajo enfocado en salvaguardar la protección de datos de los usuarios, así como uno que garantiza la comodidad de ellos al usar la app. Esto ya que se implementó una comunicación de información mas estable como el paso del nombre de usuario y de su cuenta como tal para el escaneo de QR y en el menú principal del restaurante. Además, se añadieron algunas contingencias para proteger la información del usuario como siempre reiniciar las actividades cuando este salga de ellas para que no se pueda hacer un uso negativo de su cuenta y posteriormente de su información mas sensible como lo sería su información financiera.

4. Implementación de autenticación por medio de cuenta de Google

La autenticación por medio de una cuenta de Google fue uno de los primeros cambios impuestos en el código al inicio del desarrollo del entregable 2. Esto se lo hizo para tener un método adicional al inicio de sesión hecho con un correo electrónico y contraseña almacenada por el método de registro. Lo que se hizo fue que se tuvo que hacer un proceso de autenticación en el cual nos conectábamos a Firebase y por medio de este se podían detectar las cuentas de Google por las cuales los usuarios podrían seguir en la aplicación. Mas detalladamente, se sigue el mismo proceso de autenticación hecho originalmente con el método de mail y contraseña, pero se tuvieron que añadir otros recursos como el uso del API de Google para poder hacer sesión. Además de esto, se unifica el objeto de autenticación en el código para que sea uno solo para todos los procesos de inicio de sesión y que, sin importar el método, se tenga un control mas concreto para el usuario. Los aspectos mas esenciales se encuentran en el código donde se hace se confirma desde la app que la solicitud del servicio es legítima, esto permite la conexión que permite a los usuarios usar sus cuentas, autenticar por medio de Firebase que sus datos de cuenta sean los mismos que los de su cuenta de Google y por ende permite el ingreso al resto del programa.

Cabe destacar que esto fue posible en mayor parte que el mismo Firebase presenta de cierta manera un acceso más libre a la información de usuarios, así como el resto de los procesos de conexión entre la plataforma y el código debido a que, si ya usamos este servicio, se ofrece mas confianza a las Apis y recursos oficiales de Google. Una implementación especial que se hizo para el login de Google fue que cada vez que el usuario saliera de la actividad, se tuviera que hacer inicio de sesión de nuevo y por medio de los recursos y API que el servicio nos da acceso. Esto ya que se busca no comprometer a los usuarios o si estos tienen algún problema en todo el flujo de vida de la aplicación. Como se verá a continuación, esto fue más complejo al implementar el inicio de sesión por medio de Facebook.

5. Implementación de autenticación por medio de cuenta de Facebook

La autenticación por medio de Facebook fue un proceso mas complejo en comparación al de Google. Esto se debe y se hace la hipótesis que debido a que Firebase es de propiedad de Google, todo el proceso de autenticación por este medio fue mas sencillo en comparación debido a que hay mayor confianza al usarse Firebase en todo y que muchas funcionalidades estaban presentes en las dependencias principales del servicio. Facebook fue más complicado debido al manejo estricto de la información de los usuarios, además hay que mencionar que requiere de un proceso de sincronización más detallado ya que se debe configurar la información tanto en Firebase Autenticación como en Meta para desarrolladores. Esto incluye que se debe pasar datos únicos de la aplicación hacia meta como lo es el código de la misma, así como el código de autenticación que esta tiene en sus distintos gradles así como el hash SHA1 que debe darse a Meta.

Dejando todos los aspectos técnicos del proyecto como tal, el uso de Facebook requiere como Google, un proceso de autenticación especial y que se realizó en la actividad de Login. En el código como tal, se especifica que el usuario debe dar el acceso a su correo electrónico, así como la información de su perfil (nombre y foto) y que una vez que esto se dé, la misma aplicación de Facebook registra a nuestra aplicación como una de las cuales se da acceso a la info de la cuenta. Puede decirse que fue un proceso mas complicado de lo habitual debido a que Facebook como tal requiere de mucha información especifica de la app y de valores creados en base a esta info como lo sería el token de identificación, el de login y la clave de ingreso. Aparate de esto, se vio un problema en las dependencias usadas en el código ya que solo por el uso de las versiones mas actuales se puedo hacer que este método de ingreso

sea viable ya que varias funciones que se usaban en el código solo sirven desde las versiones mas recientes y en especial con el proceso de comunicación de los tokens.

6. **Creación de la clase QRactivity**

La actividad como tal es la mas importante en si de la aplicación y podría decirse que es el corazón de esta. Esta se ve primero solo accesible si se logro hacer un login exitoso con uno de los tres métodos implementados y hablados anteriormente ya que se hace como extra de intent al nombre del usuario y se añaden instrucciones en la actividad para lo que debe hacer. Como mencionamos antes en la parte de implementación de nuevas dependencias, se tuvo que investigar e implementar librerías que permitan como tales dos cosas: el acceso de la cámara y la lectura o escaneo de códigos QR. Esto se logra con CamaraX y Firebase MLKit que permiten respectivamente usar el hardware el dispositivo y la lectura de tanto códigos QR como de Barras. Todo lo que engloba la actividad de QRactivity fue lo mas complicado de este proceso ya que en un principio se usaron distintas dependencias que al momento de trabajar en conjunto no se podía alcanzar el objetivo como tal. Se tuvo que investigar en los documentos oficiales de Firebase a fondo para saber como usar estos recursos y funciones de forma efectiva para cumplir con su trabajo

En el layout de la actividad como tal, se implementan dos elementos interactivos, una previewView que enlaza a la cámara del dispositivio y un botón de regreso que permite el retroceso a Login y cambiar o usar otro método de inicio de sesión. Lo que pasa aquí es que la cámara como tal debe debe tener permiso del usuario la primera vez que se le activa, esto permite la conexión al hardware de la cámara trasera. Una vez que se tiene acceso se activa el modo de análisis y en la función de esta como tal se debe definir que se desea escanear un código (QR o de Barras) y que una vez realizado esto se debe enviar la información obtenida aquí junto al nombre del usuario hacia la actividad de menú principal del restaurante. Esto permite un paso de información concreto al leer este tipo de códigos lo cual permite que no se pierda ningún bit de información. Esto en retrospectiva fue de las partes mas complicadas debido a que distintas librerías como Zxing habían sido implementadas, pero no podían realizar una lectura correcta de los valores ya que no era compatible con camaraX que en si otorgaba las funciones para este análisis de imágenes.

7. **Desarrollo de un código generador de códigos QR y el papel de Firebase Storage**

Este segmento de código es especial en un sentido, ya que en si en la versión actual del proyecto se encuentra comentado sin embargo tuvo un uso previo esencial para el desarrollo de este. Para esta etapa se volvió a usar zxing como librería ya que si bien no era apta para la lectura de QRs en la actividad correspondiente, esta de forma independiente si posee varias herramientas para la creación o generación de códigos QR. Un elemento principal que se usa en el proyecto es un archivo .txt (Mesa1.txt) que sirve como un manual de instrucciones como tal ya que contiene el numero de mesa en la que se encuentra el usuario, un menú en este momento de 2 platos donde cada uno tiene: nombre, descripción, precio, elementos extra y una imagen referencial. Esto ya que una vez que se lea el contenido del código se lo carga en n objetos llamados platillos que tienen estas características especificadas y luego como tal un objeto MenuPrincipal donde se tiene el numero de mesa, nombre de usuario y una lista con los n platillos.

Inicialmente, el código generador de QR se encargaba de leer el archivo .txt, convertirlo en una imagen JPG y almacenarlo en la memoria externa del dispositivo. Sin embargo, esto presenta problemas logísticos ya que el QR al tener el texto en su totalidad podía presentar problemas a la hora de ser

escaneado por distintos dispositivos y de cierta manera no se presentaba como un sistema mas especializado. Lo de guardar el código en el almacenamiento tampoco es viable debido a que si el usuario no borra el contenido en distintos restaurantes su memoria se puede saturar además de que en si solo se hizo esta acción para comprobar que si se genera el código y en realidad no se tiene un enfoque, así como tal. Lo que se hizo fue otra cosa, en vez de que el QR se genere a partir del texto como tal se hizo de la manera que se genere de un link. Este enlace hace referencia al lugar donde esta almacenado el texto que fue en Firebase Storage, esto permite en si un sistema mas profesional y de mayor acceso para los desarrolladores o encargados ya que solo necesita pasarse el nombre del archivo y usando las funciones de los objetos de FirebaseStorage y StorageReference se puede tener este hipervínculo y descargar los contenidos del texto.

Recapitulando y simplificando las cosas, lo que se tiene es un código que genera QR basados en la dirección del texto dentro de Firebase storage y que siempre se almacena en esta en forma de PNG ya que permite tener los códigos simplificados y asegurados en el almacenamiento del servicio. Lo que se hace es que cuando se lee el código en la QRactivity es que se lee el link hacia el storage donde esta el texto y se lo pasa a la actividad de MenuPrincipal. Es aquí donde usando el enlace se descarga el texto como tal y se abre su contenido. En esta versión preliminar se utiliza el contenido del texto para un TextView todo en fin de comprobar la utilidad y la veracidad de la descarga del archivo. Este es un proceso que puede ser mas complicado de lo que se necesitaría pero se lo hace de esta manera para garantizar la comodidad tanto para el usuario como para la gente encargada de gestionar la app en sus propios restaurantes ya que cada código qr en un mismo restaurante puede ser distinto para diferenciar por el numero de mesas y como todo se almacena de forma ordenada en storage, resulta más fácil tener todos los menus y códigos de un restaurante en el mismo lugar y sin problemas de seguridad.

8. Cambios en los layouts del Proyecto

Ya habiendo explicado los mayores cambios en el proyecto como lo fue la adición de la lectura de QR o de los métodos de inicio de sesión, es hora de hablar de las pequeñas actualizaciones que se hicieron al código como lo seria en el aspecto visual. Esto incluye la adición de un logo que representa al servicio de comida con un tenedor, chuchillo y plato. Esto se añadió tanto como icono de la aplicación en el celular, así como un elemento recurrente en todo el ciclo de vida de esta haciendo su primera aparición en la actividad splash inicial. De igual manera se ha personalizado los botones de inicio de sesión dándole una estética simple pero que ayuda a distinguir cada elemento del otro. Aparte de estos añadidos gráficos, se implementó un estilo de texto llamado “Minecraft” que toma el font del videojuego y lo usa en la app para darle un estilo característico al programa. Si bien esta área del apartado gráfico se tiene en mente cuando se tenga el programa completamente funcional, es una buena idea ya tener implementado una idea de como se desea que se vea la app.

Aparte de todo lo mencionado y considerando la retroalimentación recibida en el entregable 1, se creo un repositorio digital en GitHub que almacena las distintas versiones del código del proyecto integrador para que se entienda los cambios y avances que se han hecho. El link es el siguiente:

https://github.com/GuilleCisneros23/Proyecto-Ragnarok-Proyecto-Integrador-.git

Secciones o capítulos del documento final desarrollados

1. Resumen
2. Abstract
3. Introducción.
4. Importancia.
5. Descripción de la Propuesta/Proyecto.
6. Estado del arte.
7. Desarrollo del prototipo.
8. Análisis del código.
9. Análisis de resultados.
10. Problemas/Dificultades
11. Recomendaciones
12. Conclusiones
13. Bibliografía/Referencias
14. Anexo A: Diagrama de Actividades de Usuario
15. Anexo B: Diagrama de Actividades del Sistema
16. Anexo C: Diagrama de Robustez del Usuario Parte I
17. Anexo D: Diagrama de Robustez del Usuario Parte II

3. Revisión y firma del tutor del proyecto

Yo, **Daniel Fellig Goldvechmiedt**, profesor de la carrera de Ingeniería en Ciencias de la Computación, hago constar que he revisado y, por lo tanto, apruebo las actividades realizadas durante este período de trabajo. Por otra parte, considero que el avance del proyecto integrador es adecuado y se corresponde con el cronograma definido en el documento de planificación.

_____

Fdo: **Daniel Fellig Goldvechmiedt**

Quito, 5 de noviembre de 2023