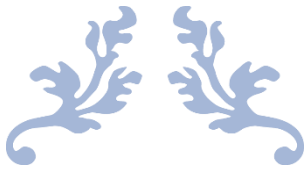


MI PRIMERA APLICACIÓN

Ciclo: CFGS Desarrollo de Aplicaciones Multiplataforma

Autor: Javier Botella Muñoz

Título: MyMarketList

Málaga, Enero 2023

Contenido

1.- Sobre este proyecto

1.1.- Control de versiones

Para poder gestionar y controlar las distintas versiones que se realicen de la app usare la herramienta de control de versiones Git en la plataforma GitHub

Enlace al repositorio: https://github.com/JaviBot00/MyMarketList.git

1.2.- Licencia de uso

Sera una app con licencia de software de privativo. Un uso ajeno al mío, a no ser que sea colaborador, requerirá de una autorización o pago.

2.- Análisis del problema

2.1.- Introducción al problema

En mi empresa hay muchos trabajadores y están en busca de nuevas maneras para mejorar la eficiencia de sus operaciones y satisfacer las necesidades de la empresa.

Uno de los problemas que se han identificado en la empresa, es que muchos de sus trabajadores pasan demasiado tiempo planificando sus comidas y comprando alimentos durante sus horas de trabajo, lo que afecta negativamente a su productividad.

Fue entonces cuando se les ocurrió la idea de crear una aplicación de lista de compras para alimentos. Esta, facilitaría a los trabajadores planificar sus comidas para la semana de forma más eficiente y permitiría agregar los productos necesarios en una lista de compras que se sincronizara automáticamente con la tienda en línea de la empresa.

2.2.- Antecedentes

Esta app no solo ahorraría tiempo a los trabajadores, sino que también permitiría a la empresa comprar alimentos en grandes cantidades, lo que resultaría en una reducción de costes.

Pero no solo se trataría de ahorrar en tiempo y dinero, sino que, esta aplicación podría ayudar a los trabajadores a planificar comidas saludables y equilibradas, lo que a su vez mejoraría su salud y bienestar en general, meta que persigue la empresa hacia sus empleados.

2.3.- Objetivos

Al usar esta app, los trabajadores podrían planificar sus comidas de manera más eficiente y reducir el tiempo que pasan comprando alimentos durante sus horas de trabajo. Además,

La aplicación podría aumentar la satisfacción del empleado. Los trabajadores se sentirían valorados y cuidados, lo que podría verse reflejado en su desempeño y productividad en el trabajo.

En resumen, la creación de una simple aplicación de lista de compras podría tener un impacto positivo en la empresa y sus empleados, demostrando que a veces, las soluciones más simples pueden ser las más efectivas.

2.4.- Requisitos

2.4.1.- Funcionales

Se busca que la app permita a los empleados:

- Crear y editar listas de compras de alimentos.
- Compartir sus listas de compras con otros miembros del equipo o departamentos.
- Añadir, eliminar o modificar los productos en la lista de compras.
- Proporcionar una interfaz fácil de usar e intuitiva para que los empleados puedan realizar sus tareas de manera eficiente.

2.4.2.- No funcionales

Algunos requisitos no funcionales podrían ser:

- Segura y proteger los datos confidenciales de los empleados y de la empresa.
- Escalable y capaz de manejar grandes volúmenes de datos y usuarios.
- Compatible con diferentes dispositivos y sistemas operativos.
- De fácil instalación y actualización.
- De bajo costo y tener una buena relación calidad-precio.

2.5.- Recursos

2.5.1.- Software

En cuanto a software se refiere, se usará:

- Windows 10 Pro | 22H2 build-19045.2846
- Android Studio Flamingo | 2022.2.1
- VMware® Workstation 16 Pro | 16.2.5 build-20904516
- Ubuntu Server 22.04.2 LTS
- Apache 2.4.52
- PHP 8.1
- MySQL Server 8.0.32
- MySQL Workbench 8.0.33
- Sublime Text 4 | build 4143

2.5.2.- Hardware

En cuanto a hardware se refiere, se usará:

- Portatil Acer:
    - Intel Core i5-8250U 1.6GHz
    - Intel Core UHD Graphics
    - 12 GB DDR4
- Elephone P9000
- Samsung Galaxy Tab A 10.1 | SM-T580
- Xiaomi Redmi 7
- Samsung Galaxy A20e
- Xiaomi Redmi Note 11 Pro+ 5G
- Android 7, 8, 10, 11, 13 (respectivamente)

3.- Diseño de la solución software

3.1.- Casos de uso

Mi app permite a los usuarios crear y administrar listas de productos alimenticios que necesitan comprar. Los casos de uso de esta aplicación pueden incluir:

- Registro de usuarios: Los usuarios pueden crear una cuenta en la aplicación proporcionando su nombre, dirección de correo electrónico y contraseña para acceder a las funcionalidades de la aplicación.
- Creación de lista de compras: Los usuarios pueden crear nuevas listas de compras y asignarles un nombre o categoría para organizar sus productos.
- Sugerir nuevos productos: Los usuarios pueden sugerir la adición de nuevos productos a la base de datos de la aplicación. Pueden enviar una solicitud para incluir un producto específico que no se encuentre actualmente en la base de datos. Esta función permite a los usuarios colaborar en la expansión de la base de datos de productos de la aplicación y asegurarse de que esté actualizada y completa.
- Compartir la lista: Los usuarios pueden compartir su lista de compras con otros usuarios, como miembros de la familia o compañeros de piso, para que puedan colaborar y agregar productos adicionales.
- Notificaciones: La aplicación puede enviar recordatorios o notificaciones a los usuarios para recordarles que revisen o actualicen su lista de compras antes de ir de compras.
- Sincronización en múltiples dispositivos: La aplicación puede permitir que los usuarios sincronicen su lista de compras en diferentes dispositivos, como teléfonos móviles y tabletas, para acceder y actualizar la lista desde cualquier lugar.
- Visualización de gráfico de compras tiempo/dinero: Los usuarios pueden ver un gráfico que representa la relación entre el tiempo y el dinero gastado en compras de alimentos a lo largo del tiempo. Este gráfico proporciona una representación visual de los patrones de gasto y ayuda a los usuarios a controlar su presupuesto.

Estos casos de uso proporcionan una visión general de cómo los usuarios interactuarían con la app, facilitando la gestión y organización de sus compras diarias o semanales.

3.2.- Base de datos

3.2.1.- Diseño Conceptual (ER)

El diseño conceptual del Modelo Entidad-Relación (ER) propuesto para el sistema de lista de compras consta de 2 Base de Datos separadas para poder manejar de forma más optima:

Base de datos: BDProductos

- Tipos:
    - Esta entidad representa las distintas categorías de productos que hay con un identificador único (ID_tipo) para poder referenciarlos con los productos
- Productos:
    - La entidad principal en esta base de datos es "Producto", que representa a los diferentes alimentos disponibles para agregar a las listas de compra.
    - Cada producto tiene una identificación única (ID_producto) y un identificador del tipo de producto que es.

Base de datos: BDCliente

- Usuario:
    - La entidad principal en esta base de datos es "Usuario", que representa a los usuarios de la aplicación de listas de la compra de alimentos.
    - Cada usuario tiene una identificación única (ID_usuario), un nombre de usuario, una contraseña, una dirección de correo electrónico y una foto de perfil.
- Listas:
    - Esta entidad representa las distintas listas creadas con un identificador único (ID_lista) para poder referenciarlos con los productos
    - Cada lista tiene una identificación única (ID_ lista), el nombre de lista, la fecha de cuando se creó la lista, la fecha de cuando se realizó la compra y el precio de la compra.
- Productos:
    - Esta entidad representa los distintos productos que se han usado para crear las listas con un identificador único (ID_producto) para poder referenciarlos con las listas
    - Cada producto tiene una identificación única (ID_producto) y un identificador de la lista a la que pertenece.

3.2.2.- Diseño lógico (tablas normalizadas)

A continuación, se presenta el diseño lógico con las tablas normalizadas basado en el diseño conceptual ER previamente mencionado:

Base de datos: BBDD_Alimentos

Tabla: Tipo

- Atributos: ID_tipo_producto (PK), nombre_tipo_producto

Tabla: Producto

- Atributos: ID_producto (PK), nombre_producto

Base de datos: BBDD_Cliente

Tabla: Usuario

- Atributos: ID_usuario (PK), nombre_usuario, contraseña, correo_electronico, foto_perfil

Tabla: Lista

- Atributos: ID_lista (PK), nombre_lista, fecha_creacion, fecha_realizacion, precio_total

Tabla: Producto

- Atributos: ID_producto (PK), nombre_producto, ID_lista (FK)

Ahora, cada base de datos cumple su función específica. La base de datos "BDCliente" almacena la información relacionada con los usuarios, las listas de compra y los productos asociados a cada usuario. La base de datos "BDProductos" almacena la información de los diferentes tipos de productos disponibles.

Este diseño permite una gestión eficiente de los usuarios, las listas de compra y los productos en sus respectivas bases de datos separadas. Cada tabla está normalizada para evitar redundancias y mantener la integridad de los datos. Este diseño lógico con tablas normalizadas cumple con los principios de la normalización de bases de datos y ayuda a evitar redundancias y anomalías en los datos.

3.3.- Prototipado gráfico

3.3.1.- Tablets / Smartphones

El proceso de creación de un prototipo gráfico para una aplicación móvil en Android Studio implica seguir una serie de pasos para asegurar un diseño coherente y una experiencia de usuario intuitiva.

Una vez que hemos definido claramente los requisitos de la aplicación, procedemos al diseño de la interfaz de usuario (UI). En Android Studio, utilizamos el editor de diseño gráfico y los widgets predefinidos para crear la interfaz. Es fundamental asegurarnos de que los elementos de la interfaz sean claros, intuitivos y sigan un estilo coherente en cuanto a su apariencia y disposición.

Además de la apariencia, debemos tener en cuenta la navegación entre pantallas. Para ello, creamos una estructura de navegación lógica y fácil de entender. Android Studio nos proporciona componentes de navegación, como la barra de navegación inferior (Bottom Navigation) o el cajón de navegación (Navigation Drawer), que facilitan la transición entre pantallas y brindan una experiencia de usuario fluida.

Es recomendable seguir las pautas de Material Design, el lenguaje de diseño de Google para aplicaciones Android. Al adherirnos a Material Design, utilizamos colores, tipografía, sombras y animaciones que siguen las directrices establecidas, lo cual nos permite ofrecer una experiencia visual atractiva y familiar para los usuarios de Android.

Una vez completado el diseño, es crucial realizar pruebas exhaustivas de usabilidad y rendimiento en diferentes dispositivos Android. Utilizamos las herramientas de depuración y perfilado proporcionadas por Android Studio para optimizar el rendimiento de la aplicación y asegurarnos de que funcione correctamente en diversas situaciones.

Durante el proceso de prototipado gráfico, hemos tenido en cuenta factores como la usabilidad, la coherencia visual, la navegación intuitiva y el cumplimiento de las directrices de diseño establecidas. Además, hemos realizado pruebas rigurosas para garantizar que la aplicación se desempeñe de manera óptima en una amplia gama de dispositivos Android.

4.- Implementación

4.1.- Codificación

4.1.1.- Backend

El backend utilizado para esta aplicación móvil de gestión de listas de compras ha sido diseñado y creado por mi (Javier Botella Muñoz) sin utilizar servicios de terceros. Este backend se ejecuta en un servidor Linux con LAMP (Linux, Apache, MySQL y PHP) y proporciona funcionalidades para almacenar y manipular listas de compras y productos en una base de datos.

Conexión a la base de datos:

- Para establecer la conexión con la base de datos, se utilizan los siguientes parámetros de conexión: servidor (localhost), puerto (3306), nombre de usuario y contraseña del administrador de la base de datos. Estos datos de conexión se encuentran protegidos y se mantienen confidenciales. La base de datos utilizada se denomina BDMyMarketList y almacena tanto los productos como las categorías de los productos. Cada usuario tiene su propia base de datos, con un nombre correspondiente al nombre de usuario, donde se guardan los datos del usuario, las listas creadas y los productos asociados a esas listas.

Interacción con la aplicación móvil:

- El backend es utilizado por una aplicación móvil dedicada a la gestión de listas de compras. La aplicación móvil se comunica con el servidor a través de solicitudes HTTP utilizando el método POST. La aplicación envía los datos necesarios para realizar las operaciones de inserción, actualización y eliminación de listas de compras y productos al backend, y recibe las respuestas correspondientes.

Inserción de listas de compras y productos:

- Cuando la aplicación móvil envía una solicitud de inserción de listas de compras o productos al backend, se reciben los datos en formato de cadena a través del método POST. Estos datos se desglosan y se insertan en la base de datos del usuario correspondiente, en las tablas "tList" y "tProducts" respectivamente. Cada lista de compras contiene información como el nombre de la lista, la fecha de creación, la fecha de realización y el precio. Cada producto contiene información como el nombre del producto y el tipo de producto. Se realiza una validación de los datos antes de la inserción y se manejan los errores en caso de fallos.

Actualización de datos desde la aplicación móvil:

- Además de las inserciones, la aplicación móvil puede enviar solicitudes de actualización de datos al backend. Por ejemplo, cuando un usuario borra los datos de la aplicación móvil o cuando se crea una lista desde otro dispositivo móvil, pero con la misma cuenta de usuario. En estos casos, la aplicación móvil envía los datos actualizados al backend, y realiza las consultas necesarias para actualizar los datos de productos y listas en la base de datos del usuario correspondiente.

Gestión de errores y transacciones:

- El backend implementa un manejo de errores robusto para garantizar la integridad de los datos. En caso de que ocurra algún error durante las operaciones de inserción o actualización, se realiza un rollback para deshacer cualquier cambio realizado en la base de datos. Esto asegura que los datos se mantengan consistentes y evita posibles inconsistencias en caso de fallos.

Commit y cierre de conexión:

- Una vez que todas las operaciones son exitosas, se realiza un commit para confirmar los cambios en la base de datos. Luego, se cierra la conexión con la base de datos de manera adecuada.

En resumen, el backend proporciona la funcionalidad necesaria para la gestión de listas de compras desde una aplicación móvil. Permite la inserción, actualización y eliminación de listas y productos, así como la gestión de errores y la garantía de la integridad de los datos. A través de una conexión segura entre la aplicación móvil y el servidor, se mantienen actualizados los datos de productos y listas en la base de datos central, incluso en situaciones donde se requiera sincronización de datos desde diferentes dispositivos móviles.

4.1.2.- Frontend

En el "Frontend" de la aplicación, que se refiere a la interfaz de usuario y las interacciones visuales, se encuentran los archivos XML que definen la estructura y apariencia de las pantallas. Por ejemplo, los archivos activity_catalogue.xml, activity_edit.xml, activity_list.xml y activity_stats.xml. Estos archivos describen los elementos visuales y su organización en cada pantalla.

Además de los archivos XML, se utilizan clases en Kotlin para gestionar las acciones de los componentes visuales y mostrar los datos en la interfaz. Estas clases están relacionadas con las actividades de la aplicación, como CatalogueActivity, EditActivity, ListActivity y StatsActivity, que heredan de AppCompatActivity. Su función es manejar el ciclo de vida de las pantallas y configurar los componentes visuales.

Dentro de las actividades, se utilizan fragmentos (clases que heredan de Fragment) para construir interfaces reutilizables. Por ejemplo, los fragmentos CatalogueFragment y Nav1ListsFragment se utilizan para mostrar listas de productos y menús de navegación respectivamente.

En términos de conectividad, se utilizan clases específicas para manejar las peticiones HTTP y la conexión a Internet. Por ejemplo, la clase MyRequest se encarga de realizar peticiones HTTP y verificar la conectividad de red en MainController. Estas clases son responsables de establecer la comunicación con un servidor remoto y obtener los datos necesarios para mostrar en la interfaz de usuario.

Para el manejo de datos, se utilizan clases que interactúan con la base de datos SQLite. Un ejemplo de ello es la clase ClientSQLite, que se encarga de realizar operaciones de lectura y escritura en la base de datos local. Esta funcionalidad permite almacenar y recuperar información relevante para la aplicación, como los elementos de una lista de compras o los datos de un usuario registrado.

En resumen, el "Frontend" de la aplicación está compuesto por archivos XML que definen la apariencia de las pantallas y clases en Kotlin que gestionan la lógica de la interfaz de usuario. Además, se utilizan fragmentos para interfaces reutilizables, adaptadores de RecyclerView para mostrar listas de elementos y clases para interactuar con la base de datos SQLite. También se emplean clases específicas para la conectividad, como la clase MyRequest para las peticiones HTTP y la verificación de la conectividad de red.

5.- Documentación

5.1.- Empaquetado / Distribución

El empaquetado y distribución de la aplicación se realizará en formato APK (Android Package) y estará destinado exclusivamente al personal interno de la empresa para su prueba y uso. La aplicación se desarrolla en la plataforma Android, lo que implica que solo será compatible con dispositivos Android.

En esta etapa del proyecto, la aplicación se encuentra en desarrollo y se espera que se agreguen funcionalidades más completas en el futuro. Por lo tanto, la distribución se limitará a los miembros del equipo de desarrollo y pruebas, quienes podrán instalarla en sus dispositivos Android para evaluar su rendimiento y proporcionar retroalimentación.

Cabe destacar que el código de la aplicación es de carácter privativo, lo que significa que su distribución no está permitida para uso público o comercial en este momento. Su uso estará restringido al ámbito interno de la empresa hasta que se completen las funcionalidades principales y se logre un nivel de estabilidad y calidad apropiado.

A medida que el desarrollo avance y la aplicación alcance un estado más completo, se evaluará la posibilidad de ampliar la distribución y considerar opciones adicionales, como su inclusión en tiendas de aplicaciones o su disponibilidad para un público más amplio.

Es importante destacar que, durante todo el proceso de empaquetado y distribución interna, se dará prioridad a la seguridad y confidencialidad de la aplicación y los datos asociados, asegurando la protección de la información de la empresa y sus empleados.

5.2.- Instalación

Para instalar la aplicación en dispositivos Android, se utilizará el archivo APK (Android Package) proporcionado por el equipo de desarrollo. Dado que la aplicación no está disponible en la tienda Play Store de Android, la instalación se realizará de forma manual utilizando el archivo APK.

A continuación, se detallan los pasos para instalar la aplicación:

1. Obtén el archivo APK proporcionado por el equipo de desarrollo de la aplicación.
2. Ve a la configuración de seguridad de tu dispositivo Android y habilita la opción "Fuentes desconocidas" o "Permitir instalación de aplicaciones de origen desconocido". Esto permitirá la instalación de aplicaciones desde fuentes externas a la tienda Play Store.
3. Transfiere el archivo APK a tu dispositivo Android a través de métodos como el correo electrónico, almacenamiento en la nube o transferencia directa desde una computadora.
4. En tu dispositivo Android, abre el administrador de archivos o una aplicación de explorador de archivos y busca el archivo APK transferido.
5. Toca el archivo APK para iniciar el proceso de instalación.
6. Si se muestra un mensaje de seguridad, confirma que deseas instalar la aplicación seleccionando "Instalar".
7. Espera a que se complete el proceso de instalación. Una vez finalizado, la aplicación estará lista para ser utilizada en tu dispositivo Android.

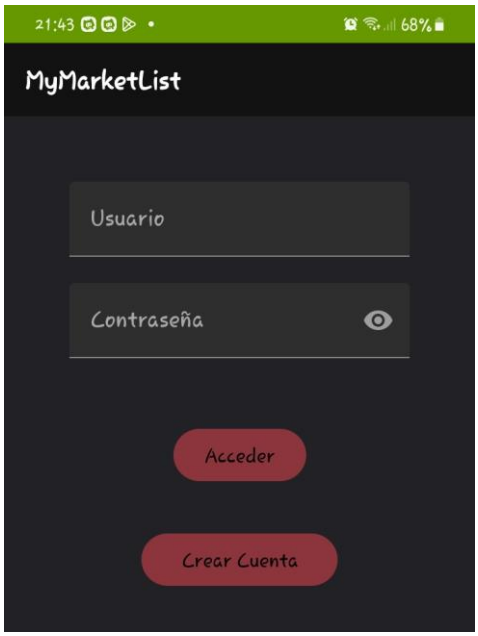
Es importante tener en cuenta que, al instalar la aplicación desde un archivo APK, es responsabilidad del usuario asegurarse de que el archivo provenga de una fuente confiable y de confianza. También se recomienda deshabilitar la opción "Fuentes desconocidas" una vez finalizada la instalación para mantener la seguridad de tu dispositivo.

Este proceso de instalación es específico para la distribución interna de la aplicación y puede diferir del proceso estándar de instalación de aplicaciones desde la tienda Play Store.

5.3.- Manual de Usuario / Referencia

El siguiente manual se elabora con el fin de poder brindar al usuario final un manejo y conocimiento adecuado de la aplicación, facilitando la navegación dentro de sus funciones con capturas de pantalla intuitivas para una mejor comprensión.

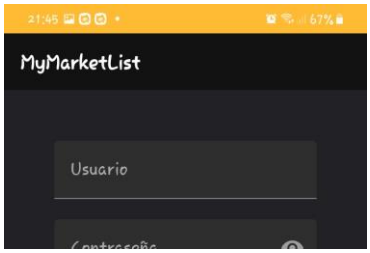
La aplicación se identifica con el icono que se muestra a la derecha, en el cual debe estar previamente instalado en el dispositivo móvil como se ha descrito en punto “5.2.- Instalación”.

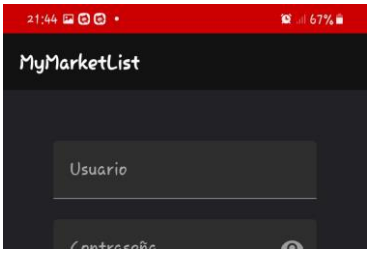


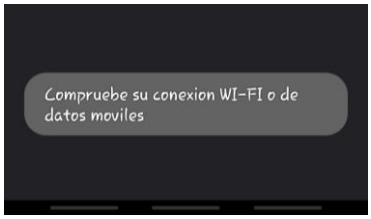
Al acceder a la app nos encontraremos con el inicio de sesión en el cual podremos ingresar si ya disponemos de una cuenta previamente creada o crearemos una nueva en la opción de “Crear Cuenta”.

Como podemos apreciar, la barra de estado está de color verde, lo cual indica que el servidor está operativo y conectado a él para poder hacer uso de las funcionalidades de la app.

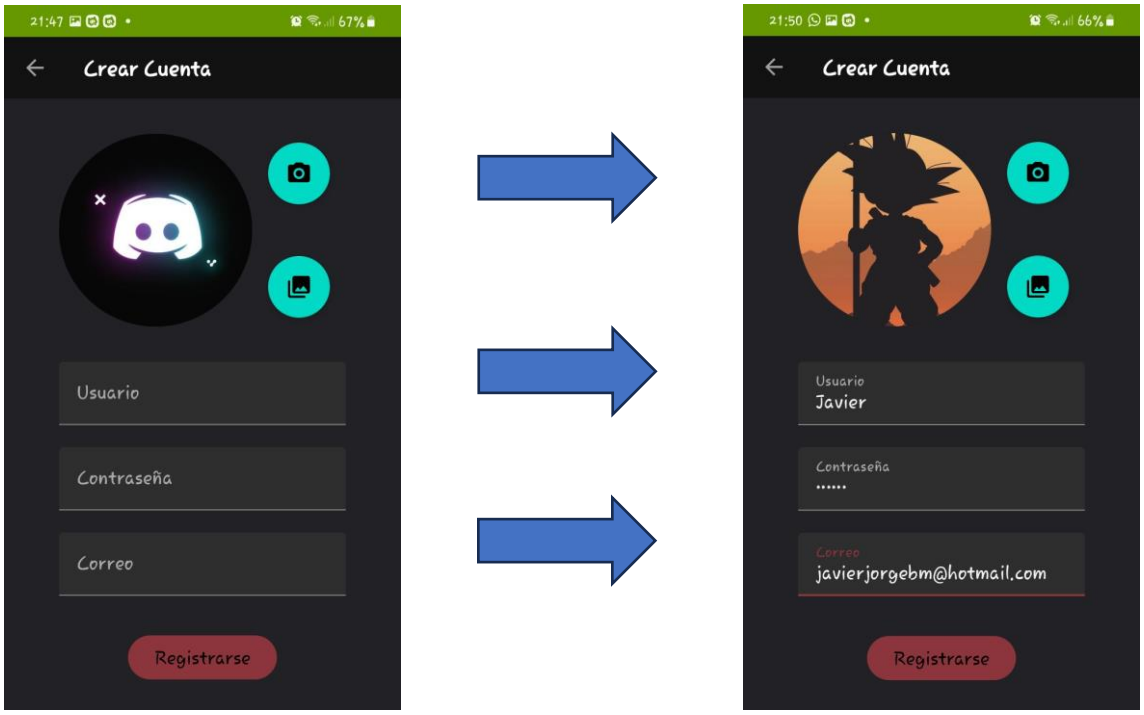
También dispone de otros estados para comprobar de manera visual la conexión con el servidor, esta el amarillo que significa que está tratando de conectarse y el rojo que no tiene conexión con el o que el dispositivo no tiene conexión a Internet.







Si tuviesemos una cuenta creada, solo haria falta introducir los datos en los campos pertinentes e iniciar sesion pero como aun no la tenemos, proceremos a crear una. Para ello le damos a “Crear Cuenta” y rellenamos los campo.



Una vez le demos a “Registrar” se creará la cuenta y accederemos a la vista principal de la app en la que tenemos distintas opciones:

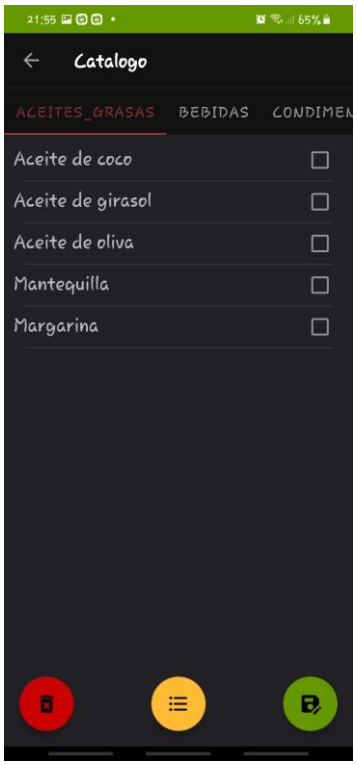
Catalogo: aquí podremos seleccionar los productos que queramos para crear la lista y asignarle un nombre a la misma.

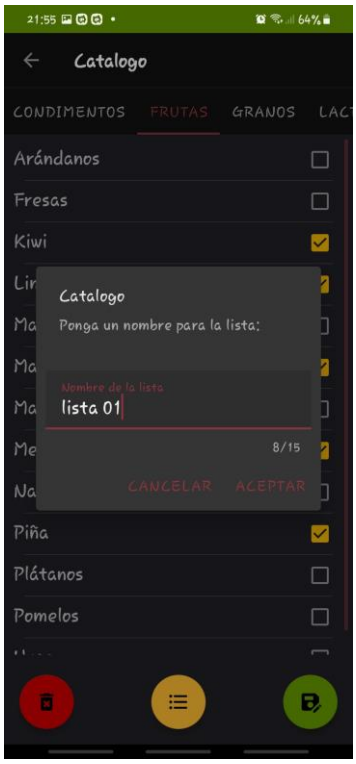
Mis listas: se visualizan todas listas creadas y tambien se los podrán asignar el precio total de la compra y la fecha de misma.

Estadisticas: se elije año y mes sobre los cuales ver la grafica de los gastos de las compras realizadas en esas fechas.

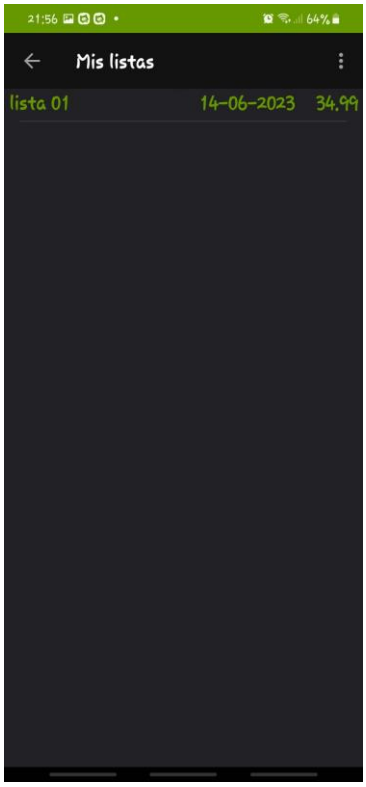
Sugerir: nos podemos poner en contacto con el administrador para sugerir o aconsejar cambios en la app como añadir más productos

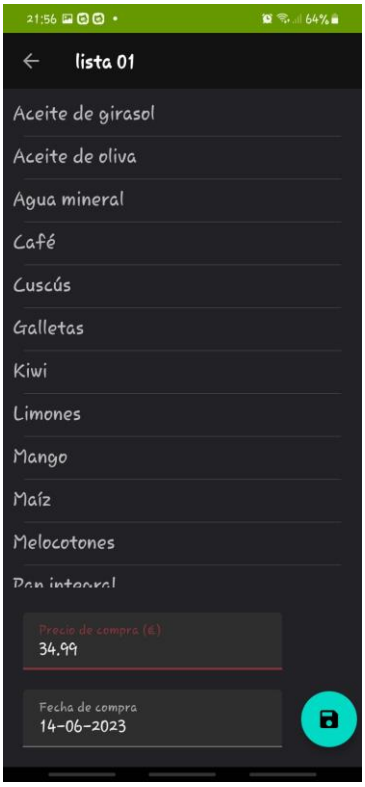
Catalogo



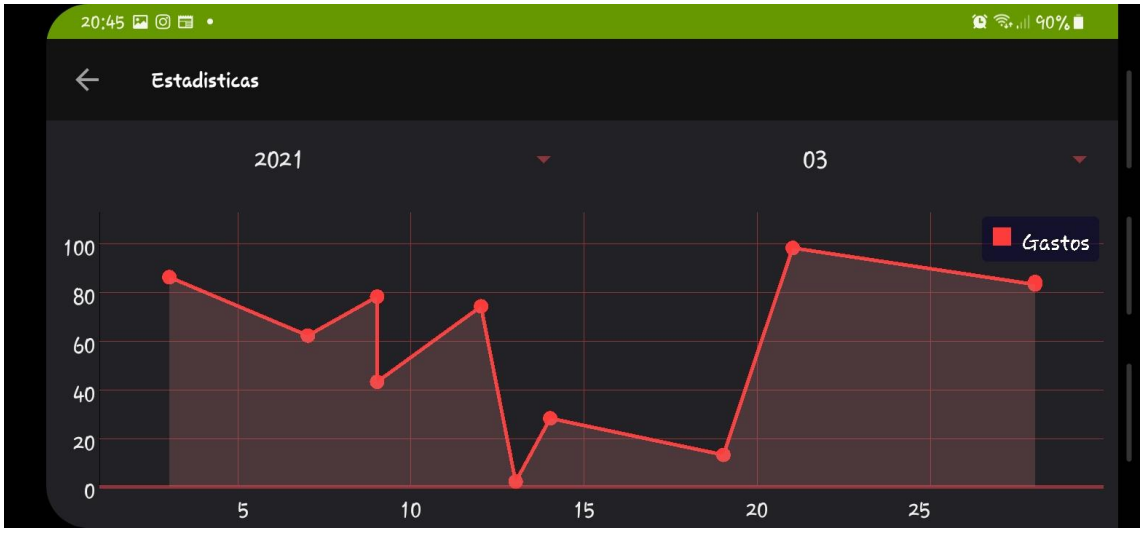


Mis listas

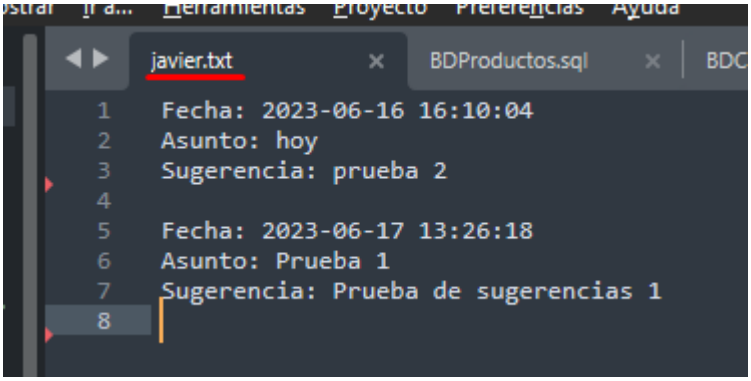




Estadísticas



Sugerencias



También disponemos de un menú “Más” con distintas opciones como datos del usuario registrado, información acerca de la app y cerrar sesión

6.- Conclusiones

En resumen, este proyecto tiene como objetivo abordar el problema de la falta de eficiencia en la planificación de comidas y la compra de alimentos durante las horas de trabajo en nuestra empresa. Para solucionarlo, se propone el desarrollo de una aplicación de lista de compras de alimentos que permitirá a los trabajadores planificar sus comidas de manera más eficiente y sincronizar automáticamente la lista de compras con la tienda en línea de la empresa.

La app buscará ahorrar tiempo a los empleados, reducir los costos de compra de alimentos y fomentar una alimentación saludable y equilibrada. Además, se espera mejorar el bienestar y el rendimiento de los trabajadores en el trabajo.

Esta app permitirá crear y editar listas de compras, compartirlas con otros miembros del equipo y ofrecer una interfaz fácil de usar. Se enfocará en garantizar la seguridad de los datos, ser compatible con diferentes dispositivos y sistemas operativos, y tener un costo accesible.

En resumen, este proyecto busca facilitar la planificación de comidas y la compra de alimentos en el entorno laboral, promoviendo la eficiencia, la salud y el bienestar de los trabajadores.

7.- Bibliografía

https://es.stackoverflow.com/questions/119184/como-hacer-un-switch-en-kotlin

https://stackoverflow.com/questions/39866869/how-to-ask-permission-to-access-gallery-on-android-m

https://www.digitalocean.com/community/tutorials/android-capture-image-camera-gallery

https://medium.com/@hissain.khan/parsing-with-google-gson-library-in-android-kotlin-7920e26f5520

https://www.digitalocean.com/community/tutorials/okhttp-android-example-tutorial

https://www.digitalocean.com/community/tutorials/android-asynctask-example-tutorial

https://stackoverflow.com/questions/2004344/how-do-i-handle-imeoptions-done-button-click

https://m3.material.io/components

https://github.com/material-components/material-components-android/blob/master/docs/components/Tabs.md#scrollable-tabs

https://developer.android.com/codelabs/android-training-create-recycler-view?index=..%2F..%2Fandroid-training#4

https://stackoverflow.com/questions/12134957/android-how-to-save-the-tab-state-when-switching-between-tabs

https://stackoverflow.com/questions/18747975/what-is-the-difference-between-fragmentpageradapter-and-fragmentstatepageradapte

https://pspdfkit.com/blog/2019/android-persistent-tabs/

https://stackoverflow.com/questions/15313598/how-to-correctly-save-instance-state-of-fragments-in-back-stack

https://www.tutorialspoint.com/android/android_php_mysql.htm#

https://developer.android.com/develop/ui/views/notifications/build-notification

https://www.javatpoint.com/android-tablayout-with-framelayout

https://developer.android.com/guide/navigation/navigation-swipe-view?hl=es-419

https://androidapps-development-blogs.medium.com/android-tab-layout-using-fragments-and-viewpager-android-studio-java-cbad951754ec

https://www.digitalocean.com/community/tutorials/android-tablayout-viewpager

https://itnext.io/add-an-animated-indicator-to-your-bottomnavigationview-db04c6aa738f

https://www.geeksforgeeks.org/bottom-navigation-bar-in-android-using-kotlin/

https://es.stackoverflow.com/questions/73038/como-dejar-un-servicio-funcionando-en-segundo-plano-en-android-cerrando-la-aplic

https://developer.android.com/training/run-background-service/create-service?hl=es-419

https://www.albertcoronado.com/2014/09/23/desarrollo-android-hacer-que-una-tarea-se-ejecute-repetidamente/

https://stackoverflow.com/questions/10108774/how-to-implement-the-android-actionbar-back-button

https://color.adobe.com/es/create/color-wheel

https://medium.com/@alvareztech/pull-to-refresh-en-tu-lista-recyclerview-en-android-9f2ce5657b30

https://developer.android.com/develop/ui/views/touch-and-input/swipe/add-swipe-interface#kts

https://developer.android.com/develop/ui/views/launch/splash-screen