Universidad Mariano Gálvez

Curso: PROGRAMACIÓN II

Docente: CARLOS RENE HERNANDEZ LARIOS

Documentación Técnica sobre la conexión de Firebase, Firestore y Storage

Grupo No.1

Integrantes

Timothy Gerald Palma Pérez 0907-20-6162

Josseline Emérita Galeano Hernández 0907-23-9861

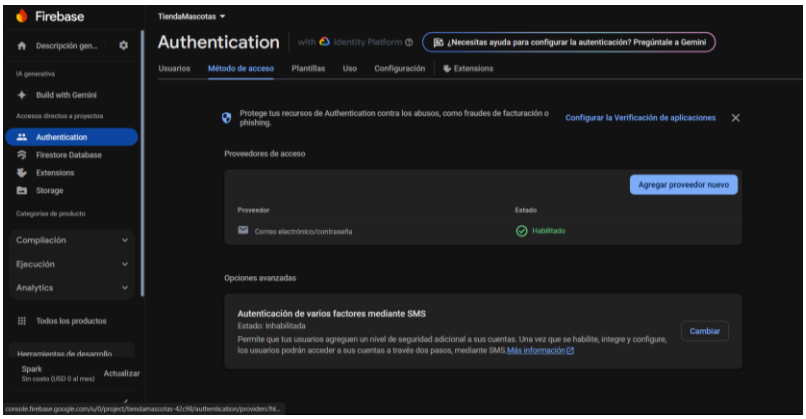
Josué Emanuel Ramírez Aquino 0907-23-1989

Jonathan Estuardo Dionicio Orellana 0907-13-9191

Antes de abordar la implementación de Firebase, es importante explicar cómo está estructurada la aplicación. Al inicio, los usuarios encontrarán una pantalla principal que solicitará su correo electrónico y contraseña. Si no tienen una cuenta, serán redirigidos a una segunda pantalla donde podrán registrarse y completar su información. Una vez que finalicen el registro, serán llevados a la pantalla principal, conocida como **Home**, que les permitirá navegar entre diferentes secciones a través de una barra de navegación.

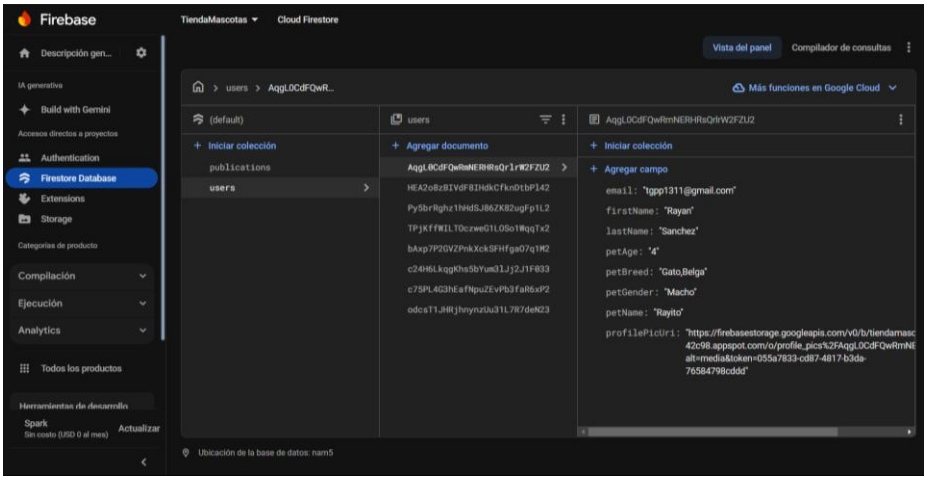
Nuestra aplicación está diseñada para funcionar como una comunidad para mascotas, donde los usuarios podrán publicar información sobre diversas temáticas, discutir la adopción de mascotas, explorar distintos servicios, e informar sobre mascotas desaparecidas para que otros puedan ayudar. Durante la primera semana, nos enfocamos en la creación de las pantallas y sus diseños, y en la segunda semana comenzamos a establecer las conexiones necesarias, las cuales se explicarán en detalle a continuación.

La integración de Firebase en nuestro código no presentó mayores dificultades, ya que seguimos correctamente los pasos recomendados para su conexión. Esta integración fue utilizada principalmente en las pantallas de **registro** y **login**, que son esenciales para el siguiente paso: la conexión con **Firestore** (la base de datos). El primer paso fue habilitar la autenticación por correo electrónico, lo que permite enviar un código de recuperación en caso de que el usuario olvide su contraseña y también sirve como método para verificar si el usuario ya está registrado en el sistema

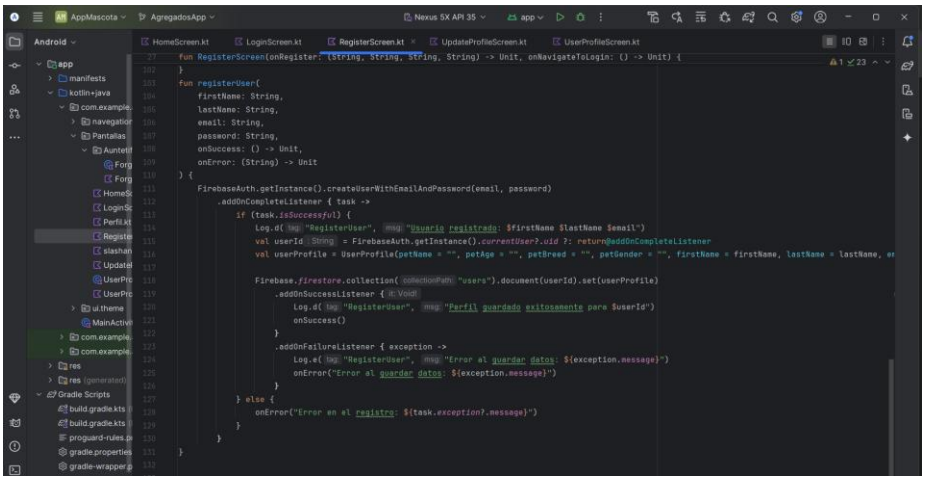


El segundo paso consistió en habilitar la base de datos de **Firestore** para crear las colecciones necesarias, donde se almacenarán los datos de los usuarios, como los correos electrónicos. Estos datos se podrán utilizar posteriormente en diversas funciones, tales como la creación de publicaciones, actualización de información personal, y la recuperación de los datos registrados a través del correo electrónico. Este proceso asegura que la información del usuario esté

correctamente organizada y accesible para futuras operaciones dentro de la aplicación.



Una vez creada la primera colección, se implementó la función **registerUser**, encargada de almacenar los datos en dicha colección. Esta función es fundamental para guardar los correos electrónicos junto con la información asociada de los usuarios. Gracias a que estos datos se encuentran en la base de datos, es posible acceder a ellos para las distintas pantallas de la aplicación, facilitando así operaciones como el registro de usuarios y la gestión de sus datos.



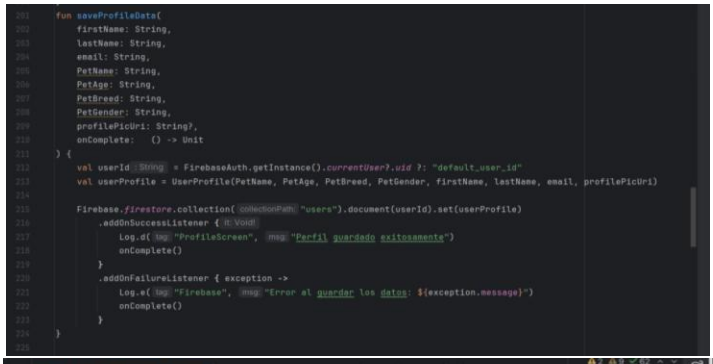
En la pantalla de **Perfil**, se completan los datos que no se ingresaron en la primera pantalla de registro. Mientras que el registro inicial almacena información básica como nombre, apellido, correo electrónico y contraseña, en la pantalla de perfil se añaden otros detalles relacionados con la mascota y la imagen de perfil. Para este último aspecto, habilitamos **Storage**, que se encarga de almacenar todas las imágenes asociadas a los perfiles.

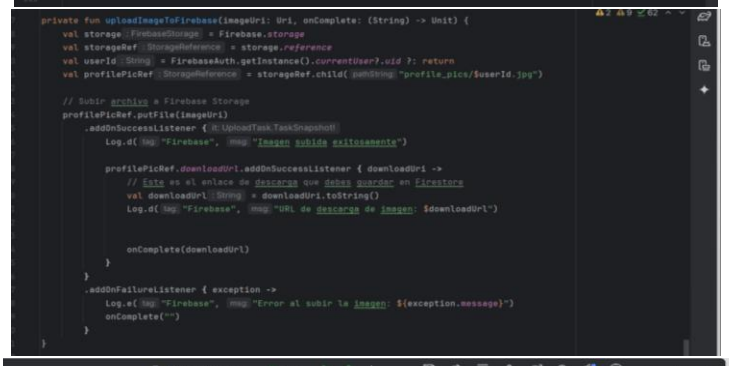
Existen varias funciones que manejan estos procesos:

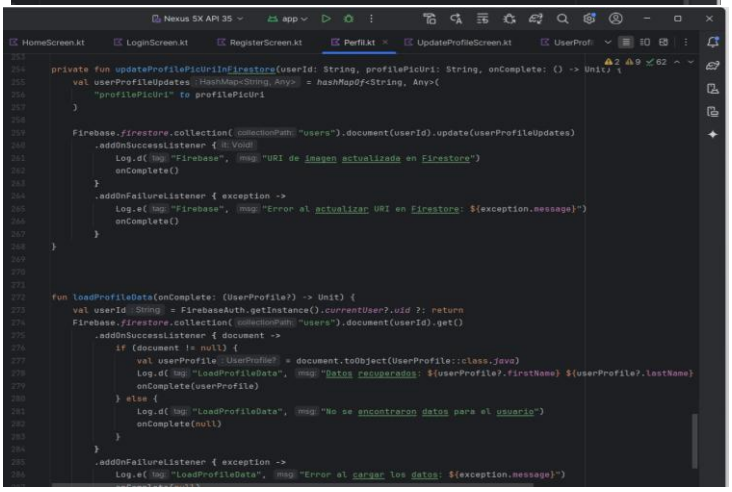
1. La primera función se encarga de guardar los datos básicos del usuario.
2. La segunda permite subir la imagen de perfil al almacenamiento.

3. La tercera actualiza la imagen de perfil correspondiente al **userId** en la colección, asegurando que se guarde en la ubicación correcta.
4. La última función se encarga de actualizar los datos del perfil, como nombre, apellido y correo electrónico, garantizando que toda la información esté actualizada y disponible para el usuario.

Este flujo de trabajo garantiza una experiencia de usuario fluida y eficiente al gestionar su perfil y la información asociada.



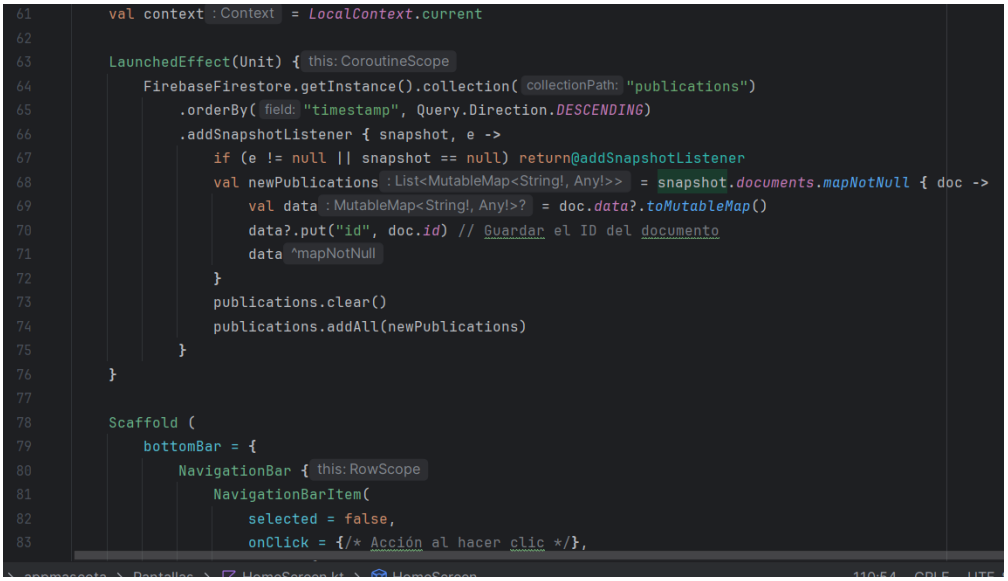




Con la base de datos establecida, el proyecto se encontraba al 50% de su finalización. La base de datos permite realizar llamadas para acceder a la información, lo que resulta fundamental al momento de hacer cambios en el perfil, como la modificación de nombre,

apellido, y los datos relacionados con la mascota, como su raza o nombre. Una vez que se realizan los cambios, estos se guardan y se reflejan en la aplicación.

Además, la base de datos facilita la creación de publicaciones, las cuales incluyen tanto los nombres de los usuarios que publican como el contenido de sus publicaciones. La primera colección se creó directamente desde Firebase, lo que permitió establecer la lógica necesaria para gestionar la información. Por ejemplo, se puede implementar un código que permita crear publicaciones de manera dinámica, asegurando así que la información se mantenga organizada y accesible para los usuarios



Esta parte del código crea una colección destinada a almacenar los datos de las publicaciones. Mediante otras funciones, se realizan llamadas a diferentes colecciones para obtener información como el nombre, apellido y correo electrónico, los cuales se gestionan en colecciones separadas. Esta estructura evita problemas relacionados con la acumulación de datos en una sola colección y previene conflictos de datos compartidos. Al mantener las colecciones organizadas y diferenciadas, se asegura que los parámetros no se sobrepongan y se minimizan los riesgos de errores en la gestión de la información

Pantalla HomeScreen:

1. Funciones principales del código

- **HomeScreen(navController: NavController)**: Esta es la función principal que define la pantalla de inicio de la aplicación. Recibe un controlador de navegación (navController) que permite la navegación entre diferentes pantallas de la aplicación.
- **LaunchedEffect(Unit)**: Utilizado para realizar operaciones asíncronas cuando se lanza la composición. Aquí se usa para escuchar cambios en la colección publications de Firestore y actualizar la lista de publicaciones en tiempo real.
- **Scaffold**: Proporciona una estructura básica para la pantalla, que incluye una barra de navegación en la parte inferior (bottomBar) y un espacio para el contenido principal.
- **LazyColumn**: Se utiliza para mostrar una lista de publicaciones de manera eficiente. Permite el desplazamiento y la carga de elementos de forma dinámica.
- **AlertDialog**: Un cuadro de diálogo que se muestra cuando el usuario quiere crear una nueva publicación. Contiene campos de texto para el título y el contenido de la publicación.

2. Componentes clave

- **Variables de estado**:
    - showDialog: Controla si el cuadro de diálogo para crear una nueva publicación está visible.
    - title y content: Almacenan los valores del título y el contenido de la publicación que el usuario está creando.
    - publications: Una lista mutable que almacena las publicaciones recuperadas de Firestore.
- **Navegación**: Se utilizan NavigationBarItem para permitir que los usuarios naveguen entre diferentes pantallas de la aplicación, como el perfil del usuario, las solicitudes de adopción, servicios para mascotas, y publicaciones de mascotas perdidas.
- **Interacción con Firebase**:
    - **Lectura de publicaciones**: Almacena las publicaciones en tiempo real desde Firestore utilizando addSnapshotListener, lo que permite que la lista se

actualice automáticamente cuando se agregan, eliminan o modifican publicaciones.

- **Eliminar publicaciones**: La función deletePublication permite al creador de la publicación eliminarla de la base de datos.
- **Guardar nuevas publicaciones**: La función savePublicationToFirestore almacena las publicaciones en Firestore junto con información adicional como el ID del usuario y su nombre.

3. Rol en la aplicación

La HomeScreen sirve como el punto de entrada principal para los usuarios de la aplicación. Proporciona la funcionalidad de ver y crear publicaciones, así como navegar a otras partes de la aplicación. El uso de Firebase Firestore permite que los datos sean dinámicos y actualizados en tiempo real, lo que mejora la experiencia del usuario al interactuar con las publicaciones.

4. Interacción del usuario

- **Crear una publicación**: El usuario puede presionar un botón que abre un cuadro de diálogo para ingresar el título y contenido de una nueva publicación. Al confirmar, la publicación se guarda en Firestore.
- **Eliminar publicaciones**: Solo el usuario que creó la publicación puede eliminarla, asegurando que la administración de contenido sea segura y controlada.

Pantalla SplashScreen:

1. Función SplashScreen

- **@Composable fun SplashScreen(navController: NavController)**: Esta es una función composable que representa la pantalla de presentación. Acepta un navController, que es responsable de manejar la navegación entre las diferentes pantallas de la aplicación.
- **LaunchedEffect(Unit) { ... }**: Este bloque se ejecuta una vez cuando se lanza el composable. Utiliza delay(3500) para pausar la ejecución durante 3.5 segundos (3500 milisegundos). Durante este tiempo, la pantalla de presentación se muestra.
- **Navegación**: Después de la pausa, se llama a navController.popBackStack() para eliminar la pantalla de presentación de la pila de navegación y luego se navega a la pantalla de inicio de sesión mediante navController.navigate(AppScreens.LoginScreen.route).
- **Splash()**: Se llama a la función Splash, que contiene la interfaz de usuario de la pantalla de presentación.

2. Función Splash

- **@Composable fun Splash()**: Esta función composable define el diseño de la pantalla de presentación.
- **Box { ... }**: El Box es un contenedor que permite apilar varios elementos. Dentro de este Box, se coloca una imagen de fondo.
    - **Image(painter = painterResource(id = R.drawable.huellas), ...)**: Muestra una imagen (probablemente una huella de perro) que llena toda la pantalla.
- **Column { ... }**: Se utiliza una Column para alinear los elementos verticalmente. En este caso, se centra horizontal y verticalmente.
    - **modifier = Modifier.fillMaxSize()**: La columna ocupa todo el espacio disponible.
    - **horizontalAlignment = Alignment.CenterHorizontally**: Centra los elementos de la columna horizontalmente.
    - **verticalArrangement = Arrangement.Center**: Alinea los elementos de la columna verticalmente en el centro.
- **Image(painter = painterResource(R.drawable.sin_fondo), ...)**: Esta imagen se muestra en el centro de la pantalla. Se ajusta su tamaño a la mitad de la pantalla (50% de fillMaxSize(0.5f)) y se establece una relación de aspecto de 1:1. Además, se aplica un desplazamiento hacia arriba con offset(y = (-80).dp) para que se vea más centrada en la pantalla.

3. Función SplashScreenPreview

- **@Preview(showBackground = true) @Composable fun SplashScreenPreview()**: Esta función se utiliza para mostrar una vista previa de la pantalla de presentación en el entorno de diseño de Android Studio.
- **Splash()**: Llama a la función Splash para renderizar la interfaz de usuario de la pantalla de presentación en la vista previa.

El código implementa una pantalla de presentación (SplashScreen) que se muestra durante 3.5 segundos al iniciar la aplicación. Durante este tiempo.

Pantalla UserProfileScreen:

Función UserProfileScreen

Descripción y Rol: UserProfileScreen es una función composable que representa la interfaz de usuario para mostrar el perfil de un usuario. Su propósito es recuperar y mostrar los datos del usuario autenticado, como su nombre, apellido, y la información de su mascota. También incluye opciones para actualizar los datos y cerrar sesión.

Componentes y Funciones:

1. **Parámetros:**
    - onNavigateToUpdate: Función que se ejecuta al hacer clic en el botón "Actualizar datos". Esta función permite navegar a una pantalla donde el usuario puede modificar su información.
    - navController: Controlador de navegación que se utiliza para navegar entre diferentes pantallas de la aplicación.
2. **Autenticación del Usuario:**
    - val currentUser = FirebaseAuth.getInstance().currentUser: Obtiene el usuario actual autenticado mediante Firebase Authentication.
    - val email = currentUser?.email ?: "": Recupera el correo electrónico del usuario, o un string vacío si no hay usuario.
3. **Estado del Perfil del Usuario:**
    - var userProfile by remember { mutableStateOf<UserProfile?>(null) }: Utiliza un estado para almacenar los datos del perfil del usuario que se recuperarán de Firebase.
    - var imageUri by remember { mutableStateOf<Uri?>(null) }: Almacena la URI de la imagen de perfil.
4. **Carga de Datos:**
    - LaunchedEffect(email): Un efecto que se activa cuando cambia el correo electrónico del usuario. Aquí se llama a loadUserData para cargar los datos del perfil desde Firestore.
    - loadUserData { profile -> userProfile = profile }: Recupera el perfil del usuario y lo asigna a userProfile.

5. **Scaffold:**
    - El Scaffold es un componente de diseño que proporciona una estructura básica de la interfaz de usuario, como la barra de navegación.
    - **bottomBar**: Define una barra de navegación en la parte inferior de la pantalla que incluye varios elementos de navegación (Inicio, Perfil, Adopción, Servicios, Mascotas Perdidas).
6. **Mostrar Datos del Perfil:**
    - if (userProfile != null): Verifica si se han cargado los datos del perfil.
    - Box: Un contenedor que permite superponer elementos, utilizado aquí para mostrar una imagen de fondo y el contenido del perfil.
    - Image: Se utiliza para mostrar la imagen de perfil del usuario, cargada desde la URI obtenida de Firestore.
    - Column: Organiza los elementos de la interfaz (Text, Buttons) en una columna vertical.
7. **Textos Informativos:**
    - Se muestran varios textos que indican el nombre del usuario, apellido, nombre de la mascota, edad, raza y sexo.
8. **Botones de Acción:**
    - **Actualizar datos**: Un botón que, al hacer clic, llama a onNavigateToUpdate() para navegar a la pantalla de actualización de datos.
    - **Salir**: Un botón que cierra la sesión del usuario y navega a la pantalla de inicio de sesión. Se muestra un mensaje de Toast al hacer clic.
9. **Carga de Datos del Usuario:**
    - loadUserData: Esta función se encarga de recuperar los datos del usuario de la colección "users" en Firestore. Si la carga es exitosa, convierte el documento a un objeto UserProfile y lo pasa al callback onComplete. Maneja errores mediante onFailureListener.

Pantalla ProfileScreen

1. Estructura General

El código representa una pantalla de perfil de usuario donde se pueden mostrar y editar los detalles del perfil, incluida la información sobre la mascota del usuario y su foto de perfil. La función principal es ProfileScreen, que se define como un componente composable.

2. Componentes Clave

ProfileScreen(onNavigateToHome: () -> Unit)

- **Rol:** Componente principal de la pantalla que gestiona la interfaz de usuario para mostrar y editar la información del perfil del usuario.
- **Funciones:**
    - Recuperar datos del usuario autenticado desde Firebase (como el correo electrónico).
    - Inicializar variables de estado para almacenar los datos del perfil y la imagen de la mascota.
    - Usar un lanzador para seleccionar una imagen de perfil.
    - Cargar datos de perfil de Firebase en el momento de la inicialización.
    - Renderizar la interfaz de usuario, que incluye campos de entrada para la información de la mascota y botones para seleccionar la foto y guardar los cambios.

rememberSaveable y remember

- **Rol:** Mantienen el estado de las variables a lo largo de la composición.
- **Funciones:**
    - rememberSaveable: Almacena el estado de las variables en caso de que la pantalla se recomposite (por ejemplo, al rotar el dispositivo).
    - remember: Almacena el URI de la imagen seleccionada.

LaunchedEffect

- **Rol:** Permite ejecutar efectos secundarios en un componente composable.
- **Funciones:** En este caso, se usa para cargar los datos del perfil una vez que se ha determinado el usuario autenticado. Si los datos existen, se actualizan las variables de estado correspondientes.

Box y Column

- **Rol:** Son contenedores que organizan otros componentes dentro de la interfaz de usuario.
- **Funciones:**
    - Box: Proporciona un espacio que se puede llenar con otros componentes, útil para superponer elementos.
    - Column: Organiza los elementos en una lista vertical, facilitando la alineación y el espaciado entre ellos.

Image y TextField

- **Rol:** Componentes de UI que permiten mostrar imágenes y campos de texto.
- **Funciones:**
    - Image: Muestra la imagen de perfil (ya sea la seleccionada por el usuario o una predeterminada).
    - TextField: Permite la entrada de texto para varios campos (nombre de la mascota, edad, raza, sexo).

Button

- **Rol:** Permite al usuario interactuar con la aplicación.
- **Funciones:**
    - Botón para seleccionar una imagen de perfil.
    - Botón para guardar los datos del perfil. Este último invoca funciones para guardar la información en Firestore y subir la imagen a Firebase Storage.

3. Funciones Auxiliares

Estas funciones manejan la lógica para interactuar con Firebase:

- **saveProfileData**
    - **Rol:** Guarda la información del perfil del usuario en Firestore.
    - **Funciones:** Crea un objeto UserProfile con los datos proporcionados y los guarda en la colección de usuarios.
- **uploadImageToFirebase**
    - **Rol:** Sube la imagen seleccionada a Firebase Storage.
    - **Funciones:** Maneja la subida de la imagen y retorna la URL de descarga.

- **updateProfilePicUriInFirestore**
    - **Rol:** Actualiza la URI de la imagen de perfil en Firestore.
    - **Funciones:** Modifica el documento del usuario en Firestore para incluir la nueva URI de la imagen.
- **loadProfileData**
    - **Rol:** Carga los datos del perfil del usuario desde Firestore.
    - **Funciones:** Recupera el documento del usuario y lo convierte en un objeto UserProfile.

Pantalla UpdateProfileScreen:

Descripción General

La función UpdateProfileScreen es una pantalla compuesta de Jetpack Compose que permite a los usuarios actualizar su perfil, incluyendo sus datos personales y la imagen de su mascota. Esta función interactúa con Firebase para recuperar y almacenar información del usuario.

Componentes y Funciones

1. **@Composable fun UpdateProfileScreen(onNavigateBack: () -> Unit)**
    - **Rol**: Esta es la función principal que construye la interfaz de usuario para la pantalla de actualización del perfil. Utiliza el sistema de composición de Jetpack para crear componentes visuales.
    - **Parámetros**: onNavigateBack es una lambda que se llama cuando el usuario navega de vuelta a la pantalla anterior.
2. **Estado del Usuario**:
    - **val currentUser = FirebaseAuth.getInstance().currentUser**: Obtiene el usuario actual autenticado.
    - **var firstName, lastName, petName, petAge, petBreed, petGender, imageUri**: Variables de estado que se inicializan para almacenar los datos del usuario y la imagen del perfil. Se utilizan rememberSaveable para preservar su estado durante recomposiciones.

3. **Carga de Datos del Usuario**:
    - **LaunchedEffect(email)**: Se utiliza para cargar los datos del perfil del usuario desde Firebase al momento de que se monta la pantalla. Cuando se carga la información, se actualizan las variables de estado con los datos del usuario.
4. **Interfaz de Usuario**:
    - **Box**: Un contenedor que permite apilar varios elementos. En este caso, se usa para colocar una imagen de fondo y los campos de entrada.
    - **Image**: Muestra la imagen del perfil del usuario. Si no hay imagen seleccionada, se muestra un ícono predeterminado.
    - **TextField**: Campos de entrada donde los usuarios pueden escribir su nombre, apellido, nombre de la mascota, edad, raza y sexo. Los cambios en los campos se almacenan en las variables de estado correspondientes.
5. **Botones de Acción**:
    - **Botón "Guardar datos del perfil"**: Al hacer clic en este botón, se llama a updateProfileData para guardar los datos del perfil en Firestore. Si la actualización es exitosa, se muestra un mensaje de éxito y se navega de vuelta.
    - **Botón "Seleccionar nueva foto"**: Abre el selector de imágenes para que el usuario elija una nueva imagen de perfil.
    - **Botón "Guardar imagen"**: Este botón llama a saveProfileImage para subir la nueva imagen seleccionada a Firebase Storage.

Funciones Auxiliares

1. **fun updateProfileData(...)**:
    - **Rol**: Actualiza los datos del perfil del usuario en Firestore.
    - **Parámetros**: Recibe los datos del perfil que se desean actualizar, así como dos lambdas para manejar el éxito o el error.
    - **Funcionalidad**:
        - Obtiene el ID del usuario actual y la referencia al documento de usuario en Firestore.
        - Verifica si el documento del usuario existe. Si es así, mantiene la URL de la imagen de perfil existente y crea un nuevo perfil con los datos actualizados.

        - Guarda el perfil actualizado y llama a la función correspondiente dependiendo del resultado.

2. **fun saveProfileImage(...)**:
    - **Rol**: Maneja la carga de la nueva imagen de perfil en Firebase Storage y actualiza la URL en Firestore.
    - **Parámetros**: Toma la URI de la imagen, junto con lambdas para manejar el éxito y el error.
    - **Funcionalidad**:
        - Verifica si se ha seleccionado una imagen. Si es así, sube la imagen a Firebase Storage.
        - Una vez que se ha subido la imagen, obtiene la URL de descarga y la actualiza en Firestore. Si hay un error en cualquiera de los pasos, se llama a la función de error correspondiente.

Pantalla ServiciosParaMascotas:

1. Estructura General

La función principal se llama ServiciosParaMascotas, que es una **composición** (Composable) que se encarga de mostrar una lista de publicaciones de servicios para mascotas y gestionar la interacción del usuario con estas publicaciones.

2. Componentes Clave

- **Variables de Estado**:
    - showDialog, title, content, showCommentsDialog, selectedPublicationId, reviewText, showReviewDialog, price: Son variables que utilizan el estado para controlar la UI, como mostrar diálogos, manejar los textos de entrada y almacenar identificadores seleccionados.
- **Lista de Publicaciones**:
    - publications: Utiliza mutableStateListOf para almacenar y observar cambios en las publicaciones recuperadas de Firestore.
- **Recuperación de Datos**:
    - LaunchedEffect: Este efecto se ejecuta cuando la composición se inicializa y configura un addSnapshotListener para escuchar los cambios en la colección "Servicios" de Firestore, actualizando la lista de publicaciones en tiempo real.

3. UI (Interfaz de Usuario)

- **Scaffold**: Utiliza Scaffold para establecer la estructura básica de la UI, que incluye una barra de navegación en la parte inferior y un contenido principal. El bottomBar define los elementos de navegación.
- **Imagen de Fondo y Título**:
    - Se utiliza una Image para mostrar una imagen de fondo y un Text para el título de la pantalla ("Publicacion de Servicios").
- **Lista de Publicaciones**:
    - Utiliza LazyColumn para mostrar una lista de publicaciones de manera eficiente, donde cada publicación se presenta dentro de un Box con borde y sombra.

4. Funciones de Interacción

- **Calificación**:
    - Los usuarios pueden calificar las publicaciones con "Me gusta" o "No me gusta" mediante botones que actualizan los contadores de likes y dislikes.
- **Agregar Reseñas**:
    - Un botón de reseña permite a los usuarios dejar comentarios sobre una publicación. Al presionar este botón, se muestra un AlertDialog donde el usuario puede escribir una reseña. Si se envía una reseña vacía, se muestra un mensaje de error.
- **Borrar Publicaciones**:
    - Solo el usuario que creó la publicación puede ver el botón para borrar la publicación, lo que implica un control de acceso básico.

5. Funciones Auxiliares

- **DeletePublication**: Se encarga de eliminar una publicación específica de Firestore, mostrando mensajes de éxito o error en el log.
- **getUserData**: Recupera los datos del usuario que creó la publicación (nombre y apellido) y usa un callback para devolver estos valores.
- **savePublication**: Guarda una nueva publicación en Firestore, asegurándose de incluir información del usuario que la creó (como el ID del usuario, nombre y apellido).

- **ratePublication**: Actualiza los contadores de likes y dislikes en la base de datos, asegurándose de que un usuario no califique la misma publicación más de una vez.

Pantalla MenuInicial:

La función MenuInicial es una **composición de interfaz de usuario** en Jetpack Compose que gestiona la pantalla principal donde se muestran las publicaciones de adopción de mascotas.

Funciones y roles:

1. **Visualización de publicaciones**: Carga y muestra en tiempo real las publicaciones de adopción desde Firestore, organizadas por fecha de creación (más recientes primero).
2. **Interacción del usuario**:
    - Permite a los usuarios enviar solicitudes de adopción.
    - Ofrece la opción de eliminar publicaciones si el usuario es el creador.
    - Muestra un diálogo para que los usuarios puedan crear nuevas publicaciones de adopción.
3. **Navegación**: Incluye una barra de navegación en la parte inferior que permite a los usuarios navegar a otras pantallas, como "Solicitudes" y "Mis Solicitudes".
4. **Gestión de estado**: Utiliza el estado para manejar el nombre de la mascota, el historial médico, la descripción, la imagen seleccionada y si se debe mostrar el diálogo para crear una nueva publicación.
5. **Carga de imágenes**: Permite seleccionar imágenes para las publicaciones de adopción y las carga a Firebase Storage, guardando su URL en Firestore.

Componentes:

- **Scaffold**: Proporciona una estructura básica para la interfaz, incluyendo la barra de navegación inferior.
- **LazyColumn**: Utiliza una lista perezosa para mostrar las publicaciones de adopción de manera eficiente.
- **AlertDialog**: Muestra un cuadro de diálogo para que el usuario ingrese detalles sobre la nueva publicación de adopción, como el nombre de la mascota y su historial médico.
- **TextField**: Permite a los usuarios ingresar texto para los detalles de la publicación.

- **SelectImage**: Componente que permite al usuario seleccionar una imagen desde su dispositivo.
- **Buttons**: Botones para acciones como "Eliminar" y "Mandar Solicitud", que realizan acciones específicas dependiendo de si el usuario es el creador de la publicación o no.
- **FirebaseFirestore**: Utiliza Firestore para almacenar y recuperar publicaciones de adopción, así como para enviar solicitudes de adopción.

Pantalla PetsLost:

1. Estructura General

La función PetsLost es una pantalla de la aplicación que permite a los usuarios ver y gestionar publicaciones sobre mascotas perdidas. Aquí los usuarios pueden reportar nuevas pérdidas, editar publicaciones existentes, o enviar solicitudes para adoptar mascotas perdidas.

Funciones

1. **Visualización de Publicaciones**: La pantalla permite a los usuarios ver una lista de publicaciones de mascotas perdidas almacenadas en Firestore. Utiliza un LazyColumn para mostrar cada publicación de forma eficiente.
2. **Crear Publicaciones**: Los usuarios pueden crear una nueva publicación al pulsar el botón "Nueva mascota perdida". Esto abre un diálogo donde pueden ingresar detalles sobre la mascota, como el nombre, la fecha de desaparición y una descripción, así como subir una imagen.
3. **Editar Publicaciones**: Los usuarios pueden editar sus propias publicaciones a través de un cuadro de diálogo que se activa al pulsar el botón "Editar". Se prellenan los campos con la información existente para que el usuario pueda hacer cambios.
4. **Eliminar Publicaciones**: Cada publicación incluye un botón "Eliminar" que permite a los usuarios eliminar sus propias publicaciones de la base de datos.
5. **Enviar Solicitudes de Adopción**: Los usuarios que no son los creadores de la publicación pueden enviar solicitudes de adopción para las mascotas perdidas a través de un botón "Mandar Solicitud".

Componentes

1. **Variables de Estado**:
    - showDialog, showEditDialog: Controlan la visibilidad de los diálogos para crear y editar publicaciones.

- petName, fechaDesapariciony, description: Almacenan los datos de la mascota que se están ingresando o editando.
- imageUri: Guarda la URI de la imagen seleccionada.
- selectedPostId: Almacena el ID de la publicación que se está editando.
- PetsLostPosts: Lista que almacena las publicaciones de mascotas perdidas recuperadas de Firestore.

2. **LaunchedEffect**: Se usa para recuperar las publicaciones de Firestore y actualiza PetsLostPosts cada vez que hay un cambio en la colección.
3. **Scaffold**: Proporciona la estructura básica de la pantalla, que incluye una barra de navegación en la parte inferior.
4. **LazyColumn**: Utilizada para mostrar la lista de publicaciones de manera eficiente, cargando solo las que se ven en la pantalla.
5. **AlertDialog**: Utilizado para mostrar diálogos donde los usuarios pueden ingresar detalles sobre nuevas publicaciones o editar publicaciones existentes.
6. **SelectImageL Composable**: Un botón que permite a los usuarios seleccionar una imagen de su dispositivo, utilizando ActivityResultContracts.GetContent() para abrir el selector de imágenes.

Funciones Externas

- **deleteLostPost**: Elimina una publicación específica de Firestore.
- **sendLostRequest**: Envía una solicitud de adopción asociando el ID de la publicación con el ID del usuario actual.
- **saveLostPostFirestore**: Guarda una nueva publicación en Firestore, primero subiendo la imagen a Firebase Storage y obteniendo su URL.
- **saveImageLostPToFirebaseStorage**: Maneja la carga de imágenes a Firebase Storage.
- **editLostPost**: Actualiza los campos de una publicación existente en Firestore sin cambiar la imagen.

Pantalla ForgotPasswordScreen:

1. Estructura General

El ForgotPasswordScreen es una pantalla de la interfaz de usuario (UI) en una aplicación que permite a los usuarios restablecer su contraseña. Este componente es esencial para mejorar la experiencia del usuario al ofrecer una forma de recuperar el acceso a su cuenta en caso de que olviden su contraseña.

Funciones

1. **Manejo de Estado:**
    - Utiliza variables de estado (email y errorMessage) para gestionar la entrada del usuario y mensajes de error.
    - email: Almacena la dirección de correo electrónico ingresada por el usuario.
    - errorMessage: Almacena mensajes de error para notificar al usuario si la entrada no es válida.
2. **Interactividad:**
    - La función onResetPassword es un callback que se invoca cuando el usuario hace clic en el botón para enviar el correo de restablecimiento. Se le pasa el correo electrónico ingresado como argumento.
    - La función onNavigateBack permite al usuario regresar a la pantalla anterior, mejorando la navegación en la aplicación.
3. **Validación:**
    - Verifica si el campo de correo electrónico está vacío antes de intentar restablecer la contraseña. Si está vacío, se establece un mensaje de error.
4. **Interfaz de Usuario:**
    - Proporciona un diseño visual que incluye un campo de texto para el correo electrónico, un botón para enviar el correo de restablecimiento y otro botón para regresar. También muestra mensajes de error en rojo si es necesario.

Componentes

1. **@Composable:**
    - La función está marcada con la anotación @Composable, lo que significa que está diseñada para construir interfaces de usuario declarativas utilizando Jetpack Compose.

2. **Variables de estado:**
    - remember: Permite conservar el estado a través de recomposiciones. Aquí se utiliza para recordar el correo electrónico ingresado y el mensaje de error.
3. **Box:**
    - Un contenedor que permite superponer varios elementos. En este caso, se utiliza para establecer una imagen de fondo que ocupa toda la pantalla.
4. **Image:**
    - Muestra una imagen de fondo (R.drawable.mascota) en la pantalla, escalada para llenar toda la vista.
5. **Column:**
    - Un contenedor que organiza sus elementos secundarios en una columna vertical. Aquí se utilizan para apilar el texto, el campo de entrada y los botones.
6. **Text:**
    - Muestra texto en la pantalla. Se utiliza para mostrar el título de la pantalla ("Restablecer Contraseña") y el mensaje de error.
7. **TextField:**
    - Un campo de entrada donde el usuario puede escribir su dirección de correo electrónico. Su valor se actualiza en tiempo real a medida que el usuario escribe.
8. **Button:**
    - Dos botones que permiten la interactividad:
        - Uno para enviar el correo de restablecimiento.
        - Otro para regresar a la pantalla anterior.