Documentación del Inicio de Sesión en JetpackCompose y Firebase Authentication

Descripción:

Se ha implementado un inicio de sesión de usuarios utilizando Jetpack Compose para la interfaz de usuario y Firebase Authentication para la autenticación de las credenciales.

Estructura del Código:

```
@Composable
fun PantallaDeLogin(navController: NavController, users: List<User>) {
    LoginForm(navController, users)
}

@Composable
fun LoginForm(navController: NavController, users: List<User>) {
    val correo = remember { mutableStateOf("") }
    val contrasena = remember { mutableStateOf("") }
    val context = LocalContext.current
    val userAuthService = UserAuthService()

    fun validarCredenciales(
        correo: String,
        contrasena: String,
        context: Context,
        onValidacionFallida: (String) -> Unit
    ): Boolean {
        if (correo.isBlank()) {
            onValidacionFallida("El campo de correo está vacío")
            return false
        }
        if (contrasena.isBlank()) {
            onValidacionFallida("El campo de contraseña está vacío")
            return false
        }
        if (!Validaciones.isValidCorreo(correo)) {
            onValidacionFallida("Correo inválido")
            return false
        }
        return true
    }
    Button(
        onClick = {
            if (validarCredenciales(correo.value, contrasena.value)) {
                userAuthService.loginUser(correo.value, contrasena.value) { success, uid ->
                    if (success) {
                        navController.navigate("pantalla_principal/$uid")
                    } else {
                        showErrorDialog()
                    }
                }
            } else {
                showErrorDialog()
            }
        }
    ) {
        Text("Iniciar Sesión")
    }
}
```

- PantallaDeLogin representa la pantalla principal de inicio de sesión, que contiene el formulario de autenticación. Se utiliza un controlador de navegación (navController) para redirigir al usuario tras una autenticación exitosa.

Componentes de la Interfaz

- **CorreoField y ContrasenaField:** Campos de entrada de texto para que el usuario ingrese su correo y contraseña respectivamente. Se utiliza PasswordVisualTransformation en ContrasenaField para enmascarar el texto de la contraseña.

- **Button:** Botón que, al ser presionado, llama a la función de autenticación de Firebase.

Flujo del Inicio de Sesión:

1. **Entrada de datos por el usuario**
    - El usuario ingresa su correo y contraseña en los campos correspondientes.
2. **Validación previa**
    - La función validarCredenciales verifica que los campos no estén vacíos y que el correo sea válido antes de intentar la autenticación.
3. **Autenticación con Firebase**
    - Si la validación es exitosa, userAuthService.loginUser envía las credenciales a Firebase. Firebase verifica las credenciales:
        - **Éxito:** El usuario es redirigido a la pantalla principal.
        - **Error:** Se muestra un diálogo de error indicando que las credenciales son incorrectas.

Errores Conocidos y Soluciones:

1. **Error:** Campos Vacíos
   Solución: Validar los campos de entrada antes de enviar las credenciales. El sistema muestra un mensaje si faltan datos.

2. **Error:** Credenciales Incorrectas
   Solución: Si la autenticación en Firebase falla, el sistema muestra un diálogo de error para indicar que las credenciales son incorrectas y permitir reintentar.

3. **Error:** Conexión a Firebase
   Solución: Si hay problemas de red o de conexión con Firebase, el sistema muestra un mensaje indicando un error en el servidor o conexión. Se sugiere verificar el estado de red y configurar un manejo de errores que retrase la autenticación para reintentar más adelante o que informe adecuadamente al usuario.

Comentarios Adicionales:

- Futuras mejoras podrían incluir una opción de recuperación de contraseña y autenticación multifactorial para mejorar la seguridad.
- Se recomienda probar el flujo de inicio de sesión en diferentes condiciones de red y dispositivos para asegurar la robustez del sistema.

Documentación de la Pantalla del Perfil en JetpackCompose y Firestore

Descripción:

Se ha implementado la pantalla de perfil que permite a los usuarios ver y editar su información personal almacenada en Firestore, como nombre, teléfono, y CUI. También incluye opciones para cerrar sesión y eliminar la cuenta. La estructura de esta pantalla utiliza Jetpack Compose y un ViewModel para la carga y gestión de datos del usuario.

Estructura del Código:

```
@Composable
fun PantallaDePerfil(navController: NavController, uid: String,
authService: UserAuthService) {
    val userProfileViewModel: UserProfileViewModel = viewModel(
        factory = UserProfileViewModelFactory(UserFireStoreService(),
ValidarCUI())
    )

    // Cargar los datos del usuario al iniciar la pantalla
    LaunchedEffect(uid) {
        userProfileViewModel.loadUser(uid)
    }

    contenidoPantallaDePerfil(
        navController = navController,
        userProfileViewModel = userProfileViewModel,
        uid = uid,
        authService = authService
    )
}
```

Esta función inicializa la pantalla de perfil y carga los datos del usuario a través del ViewModel, que interactúa con Firestore.

Componentes de Funcionalidad Clave:

1. Visualización y Edición del Perfil:
    - La pantalla muestra la información del usuario en campos editables. La funcionalidad de edición se activa mediante un botón que permite modificar campos específicos (nombres, apellidos, teléfono, y CUI). Al guardar, se valida el CUI y se actualizan los datos en Firestore.
2. Validación de Campos:
    - La función validarCampos verifica que los datos ingresados en los campos (como nombre y teléfono) tengan el formato correcto antes de guardar los cambios. El CUI se valida específicamente para asegurar que corresponda con los valores esperados.
3. Opciones de Gestión de Cuenta:
    - Cerrar Sesión: Permite al usuario cerrar su sesión y redirigir a la pantalla inicial.
    - Eliminar Cuenta: El usuario puede eliminar su cuenta tras un temporizador de confirmación de 10 segundos. Esta función requiere que el usuario confirme su decisión, y elimina la cuenta desde Firestore al final del temporizador.

Código de Componentes:

```
@Composable
fun confirmarEliminacionDialog(
    timerText: String,
    isDeleteButtonEnabled: Boolean,
    onCancel: () -> Unit,
    onDelete: () -> Unit
) {
    AlertDialog(
        onDismissRequest = onCancel,
        title = { Text(text = "Confirmar eliminación de cuenta") },
        text = { Text(text = "¿Estás seguro de que deseas eliminar tu
cuenta? Esta acción no se puede deshacer.") },
        confirmButton = {
            Button(
                onClick = onDelete,
                enabled = isDeleteButtonEnabled,
                colors = ButtonDefaults.buttonColors(containerColor =
Color.Red)
            ) {
                Text(text = timerText, color = Color.White)
            }
        },
        dismissButton = {
            Button(onClick = onCancel) {
                Text(text = "Cancelar")
            }
        }
    )
}
```

Flujo de la Pantalla de Perfil:

1. Carga de Datos

- Al abrir la pantalla, se cargan los datos del usuario desde Firestore usando userProfileViewModel.

2. Edición de Perfil

- Los datos se muestran en campos de texto. Al presionar el botón de editar, los campos se desbloquean para permitir modificaciones.

3. Guardar Cambios

- Cuando el usuario guarda, se validan los datos y se actualizan en Firestore.

4. Cierre de Sesión y Eliminación de la Cuenta

- **Cierre de Sesión:** El usuario puede cerrar sesión, lo cual redirige a la pantalla inicial.
- **Eliminación de Cuenta:** Incluye un temporizador que permite al usuario cancelar antes de confirmar la eliminación.

Errores Conocidos y Soluciones:

1. Error: Validar los campos
Solución: validarCampos muestra mensajes de error específicos para campos inválidos (como nombres o CUI).

2. Error: Eliminación de cuenta
Solución: Si la eliminación de la cuenta falla, se muestra un mensaje indicando el problema. Se recomienda verificar la conexión a Firestore.

3. Error: Carga de datos en Perfil
Solución: Implementar mensajes de error en caso de fallo al cargar los datos desde Firestore para mejorar la experiencia del usuario.

Comentarios Adicionales:

- Agregar indicadores de carga mientras se actualizan o eliminan datos para mejorar la usabilidad.
- Integrar listeners en Firestore para actualizar automáticamente la pantalla del perfil si hay cambios en la base de datos.
- Al guardar cambios, incluir un diálogo de confirmación para confirmar que los datos fueron actualizados correctamente.

Autor:

Astrid Mileidy Peña Polanco