```
using System.Collections;  // Espacio de nombres para trabajar con colecciones, como listas y diccionarios.

using System.Collections.Generic;  // Espacio de nombres para colecciones genéricas.

using UnityEngine;  // Espacio de nombres para Unity.

using UnityEngine.UI;  // Espacio de nombres para usar UI de Unity.

using Firebase;  // Espacio de nombres de Firebase.

using Firebase.Auth;  // Espacio de nombres para autenticación con Firebase.

using Firebase.Extensions;  // Espacio de nombres para las extensiones de Firebase (como para trabajar en el hilo principal).

using TMPro;  // Espacio de nombres para trabajar con texto en UI usando TextMesh Pro.



public class AuthHandler : MonoBehaviour  // Clase que maneja la autenticación de usuarios con Firebase.

{

    private FirebaseAuth auth;  // Instancia de la autenticación de Firebase.

    public static FirebaseUser CurrentUser;  // Variable estática que almacena al usuario autenticado globalmente.



    // Referencias a la interfaz (UI) para interactuar con los campos de entrada y botones.
```

```
    public TMP_InputField emailInput;  // Campo de texto para que el usuario ingrese su correo.

    public TMP_InputField passwordInput;  // Campo de texto para que el usuario ingrese su contraseña.

    public TextMeshProUGUI feedbackText;  // Texto para mostrar mensajes de retroalimentación al usuario.

    public Button loginButton;  // Botón para iniciar sesión.



    // Referencias a las pantallas de la interfaz principal y del perfil del paciente.

    public GameObject Iniciodesesion;  // Pantalla con las opciones de iniciar sesión o revisar datos guardados.

    public GameObject Registrado;  // Pantalla que muestra el perfil del paciente después de iniciar sesión.



    // Credenciales predeterminadas para pruebas.

    private string defaultEmail = "montesdiazdayana@gmail.com";  // Correo de prueba.

    private string defaultPassword = "Usuario";  // Contraseña de prueba.



    void Start()

    {

        // Inicializar Firebase cuando la aplicación empieza.
```

```
FirebaseApp.CheckAndFixDependenciesAsync().ContinueWithOnMainThread(task
=>

    {

        if (task.Result == DependencyStatus.Available)  // Si las dependencias de
Firebase están disponibles.

        {

            auth = FirebaseAuth.DefaultInstance;     // Asignar la instancia de
FirebaseAuth.

            Debug.Log("Firebase listo para autenticación.");  // Mensaje en consola
indicando que Firebase está listo.

        }

        else

        {

            // Si no se pueden resolver las dependencias de Firebase, mostrar error.

            Debug.LogError($"No se pudieron resolver las dependencias de Firebase:
{task.Result}");

            feedbackText.text = "Error al inicializar Firebase. Contacte soporte.";  //
Mostrar mensaje de error en la interfaz.

        }

    });
```

```
        // Verifica las referencias de UI (esta parte está comentada por ahora).
        //CheckUIReferences();
    }

    /*
    void CheckUIReferences()
    {
        // Verificar si las referencias de UI están asignadas en el Inspector.
        if (emailInput == null)
            Debug.LogError("Campo de correo (emailInput) no asignado en el Inspector.");
        if (passwordInput == null)
            Debug.LogError("Campo de contraseña (passwordInput) no asignado en el Inspector.");
        if (feedbackText == null)
            Debug.LogError("Texto de mensajes (feedbackText) no asignado en el Inspector.");
        if (loginButton == null)
            Debug.LogError("Botón de inicio de sesión (loginButton) no asignado en el Inspector.");
    }
```

```
*/

public void LoginUser()  // Método que se llama cuando el usuario intenta iniciar sesión.
{
    string email = emailInput.text;  // Obtener el correo ingresado por el usuario.
    string password = passwordInput.text;  // Obtener la contraseña ingresada por el usuario.

    // Verificar si los campos están vacíos.
    if (string.IsNullOrEmpty(email) || string.IsNullOrEmpty(password))
    {
        feedbackText.text = "Por favor, completa todos los campos.";  // Mostrar mensaje de advertencia.
        return;  // Salir del método si los campos están vacíos.
    }

    // Verificar el formato básico del correo (simple validación).
    if (!email.Contains("@"))
    {
```

```
            feedbackText.text = "Correo inválido. Por favor revisa.";  // Mensaje de error si el correo es inválido.

            return;  // Salir del método si el correo no es válido.

        }

        // Verificar si el correo y la contraseña coinciden con las credenciales predeterminadas.

        if (email != defaultEmail || password != defaultPassword)

        {

            feedbackText.text = "Gmail o contraseña incorrectos, por favor volver a ingresar datos.";  // Mensaje si las credenciales son incorrectas.

            return;  // Salir del método si las credenciales no coinciden.

        }

        // Desactivar el botón para evitar múltiples clics mientras se procesa la solicitud.

        loginButton.interactable = false;

        // Intentar iniciar sesión con las credenciales ingresadas.

        auth.SignInWithEmailAndPasswordAsync(email, password).ContinueWithOnMainThread(task =>

        {
```

```
        if (task.IsCompleted && !task.IsFaulted)  // Si la tarea se completó exitosamente.
        {
            FirebaseUser user = task.Result.User;  // Obtener el usuario autenticado.
            CurrentUser = user;  // Guardar el usuario autenticado globalmente.
            // Desactivar la pantalla principal y activar la pantalla de perfil del paciente.
            Registrado.SetActive(true);
            Debug.Log($"Usuario autenticado: {user.Email}");  // Mostrar en consola el correo del usuario autenticado.
        }
        else
        {
            // Manejo de errores si la tarea no se completó correctamente.
            if (task.Exception != null)
            {
                FirebaseException firebaseEx = task.Exception.GetBaseException() as FirebaseException;  // Obtener la excepción de Firebase.
                AuthError errorCode = (firebaseEx != null) ? (AuthError)firebaseEx.ErrorCode : AuthError.None;  // Obtener el código de error de la excepción.
```

```
            // Verificar el tipo de error y mostrar el mensaje correspondiente.

            switch (errorCode)

            {

                case AuthError.InvalidEmail:

                    feedbackText.text = "Correo inválido. Inténtalo de nuevo.";  // Mensaje si el correo es inválido.

                    break;

                case AuthError.WrongPassword:

                    feedbackText.text = "Contraseña incorrecta. Inténtalo de nuevo.";  // Mensaje si la contraseña es incorrecta.

                    break;

                case AuthError.UserNotFound:

                    feedbackText.text = "Usuario no encontrado. Revisa el correo.";  // Mensaje si el usuario no existe.

                    break;

                default:

                    feedbackText.text = "Error al iniciar sesión. Intenta nuevamente.";  // Mensaje por defecto en caso de error desconocido.

                    break;

            }
```

```
                // Registrar el error en la consola.

                Debug.LogError($"Error al iniciar sesión: {task.Exception.Flatten().Message}");

            }

            else

            {

                feedbackText.text = "Error inesperado. Intenta nuevamente.";  // Mensaje para errores inesperados.

            }

        }



        // Reactivar el botón para permitir otro intento de inicio de sesión.

        loginButton.interactable = true;

    });

  }

}
```