UA.MASTER MÓVILES

SERVICIOS DE LAS PLATAFORMAS MÓVILES

Credential Manager

Introducción a Credential Manager

Credential Manager: La evolución de la autenticación en Android

Contexto:

- Google Sign-In ha sido descontinuado (soporte hasta 2024).
- Credential Manager es la solución moderna recomendada por Google.

Objetivo:

- Gestionar credenciales de forma segura y unificada:
  - Locales (usuario/contraseña).
  - Federadas (Google, Facebook, etc.).
  - Passkeys (autenticación sin contraseña).

¿Por qué Credential Manager?

¿Por qué migrar de Google Sign-In a Credential Manager?

Ventajas frente a Google Sign-In:

- Soporta estándares modernos (passkeys, FIDO2).
- Unifica todas las formas de autenticación en una API.
- Elimina dependencia de bibliotecas obsoletas.

Comparativa:

Característica	Google Sign-In	Credential Manager
Passkeys	No soportado	Si
Credenciales locales	No	Si
Autenticación federada	Solo Google	Múltiples proveedores

Dependencias y Reglas de ProGuard

Añadir dependencias y reglas de proguard

build.gradle.kts:

```
dependencies {
...
    implementation("androidx.credentials:credentials:x.y.z")

    // For devices running Android 13 (API 33) and below.
    implementation("androidx.credentials:credentials-play-services-auth:x.y.z")
    // Google Id SDK
    implementation("com.google.android.libraries.identity.googleid:googleid:x.y.z")
...
}
```

proguard-rules.pro

```
-if class androidx.credentials.CredentialManager
-keep class androidx.credentials.playservices.** {
  *;
}
```

Implementación sin dominio

Cómo empezar sin un dominio propio

Opción 1: Credenciales locales

```
val credentialManager = CredentialManager.create(this)

// Build the Google sign-in option using Credential Manager.
// GetSignInWithGoogleOption is designed for the Google sign-in button flow.
val googleSignInOption = GetSignInWithGoogleOption
    .Builder("copy url from google-services.json”)
    // You may optionally set additional parameters, such as a nonce for extra security.
    .build()

// Build the credential request with the Google sign-in option.
val credentialRequest = GetCredentialRequest
    .Builder()
    .addCredentialOption(googleSignInOption)
    .build()
```

Implementación sin dominio

Cómo empezar sin un dominio propio

Opción 1: Credenciales locales

```
// Launch an asynchronous operation (using coroutines) to request the credential.
CoroutineScope(Dispatchers.Main).launch {
    var response: GetCredentialResponse? = null
    if (Build.VERSION.SDK_INT >= 34) {
        response = getCredentialResponseAPI34(credentialManager, credentialRequest)
    }
    else {
        response = getCredentialResponseOld(credentialManager, credentialRequest)
    }

    if (response != null) {
        handleGoogleCredential(response)
    }
}
```

Implementación sin dominio

Cómo empezar sin un dominio propio

Opción 1: Credenciales locales

```
private suspend fun getCredentialResponseAPI34(credentialManager: CredentialManager, credentialRequest: GetCredentialRequest):
GetCredentialResponse? {
    var response: GetCredentialResponse? = null
    try {
        response = credentialManager.getCredential(
            context = this@CredentialManagerActivity,
            request = credentialRequest
        )
    } catch (@SuppressLint("NewApi") e: GetCredentialException) {
        // In case no Google credentials are available,
        // you can decide to show a fallback UI to let the user add an account.
        e.printStackTrace()
    }

    return response
}
```

Nota: Para APIs anteriores simplemente capturad **Exception**, ya que **GetCredentialException** no está definido.