

EXAMEN PRÁCTICO – UNIDAD III

Curso: Desarrollo de Aplicaciones Móviles

Fecha: 18 de noviembre de 2025

Estudiante: Jaime Elias Flores Quispe

Repositorio: https://github.com/jf2021070309/SM2_ExamenUnidad3

Entrega: README.md convertido a PDF con evidencia y documentación

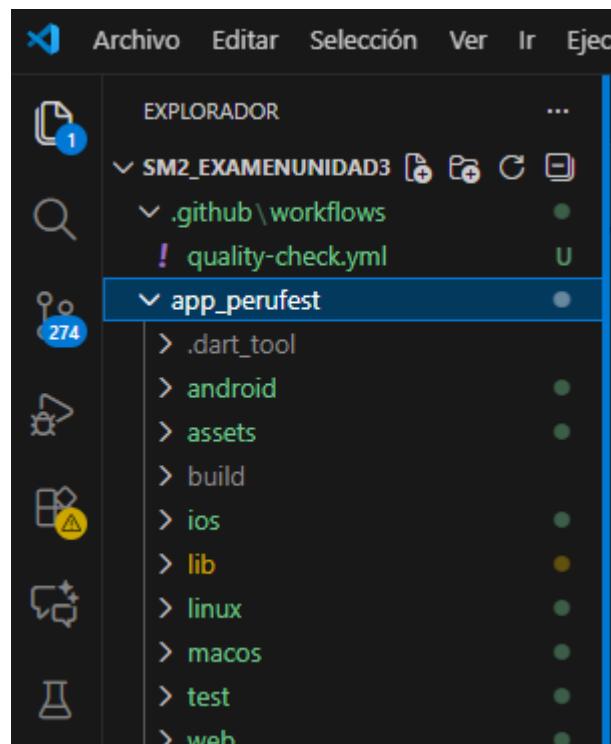
Objetivo: Implementar un flujo de trabajo (workflow) automatizado en GitHub Actions para realizar análisis de calidad sobre el proyecto móvil e integrar prácticas de DevOps.

Actividades realizadas

- **Crear repositorio:** Se creó el repositorio público con nombre exacto [SM2_ExamenUnidad3](#).
- **Copiar proyecto:** Se copió todo el contenido del proyecto móvil al repositorio.
- **Crear carpetas:** Se añadieron las carpetas obligatorias: [/.github/workflows/](#) y [/test/](#).
- **Crear archivos:** Se creó [quality-check.yml](#) en [/.github/workflows/](#) y [main_test.dart](#) en [/test/](#) con al menos 3 pruebas unitarias.

Estructura relevante (evidencia)

Captura de la estructura de carpetas mostrando [.github/workflows/](#):



Captura de [test/main_test.dart](#)

Captura que muestre el archivo [test/main_test.dart](#) con las pruebas unitarias:

Contenido de main_test.dart

```
> linux
> macos
└ test
  └ main_test.dart
> web
> windows
```

Mock de una servicio simulado:

```
app_perufest / test / main_test.dart | 🔍 main_test.dart | 🔍 mock_firestore_service | 🔍 login_usuario
1 import 'package:flutter_test/flutter_test.dart';
2 import 'package:app_perufest/models/usuario.dart';
3
4 // MOCK SIMULADO DEL SERVICE
5 class MockFirestoreService {
6   static bool retornarUsuarioCorrecto = true;
7
8   static Future<Usuario?> loginUsuario(String correo, String contrasena) async {
9     if (!retornarUsuarioCorrecto) return null;
10
11    if (correo == "jaime@gmail.com" && contrasena == "123456") {
12      // Simula un registro de usuario válido
13      return Usuario(
14        id: "1",
15        nombre: "Administrador",
16        username: "admin",
17        correo: correo,
18        telefono: "924655655",
19        rol: "administrador",
20        contrasena: "hashX",
21      );
22    }
23
24    return null;
25  }
26
27  static Future<bool> correoExiste(String correo) async => false;
28  static Future<void> registrarUsuario(Usuario usuario) async {}
29  static Future<Usuario?> obtenerUsuarioPorId(String id) async => null;
30
31 }
```

ViewModel de una clase para login:

```
31  
32 // VIEWMODEL DE PRUEBA UNITARIA  
33 enum AuthState { idle, loading, success, error }  
34  
35 class AuthViewModelMock {  
36     AuthState state = AuthState.idle;  
37     Usuario? currentUser;  
38     String errorMessage = '';  
39  
40     Future<void> login(String correo, String contrasena) async {  
41         if (correo.isEmpty || contrasena.isEmpty) {  
42             state = AuthState.error;  
43             errorMessage = 'Los campos no pueden estar vacíos';  
44             return;  
45         }  
46  
47         state = AuthState.loading;  
48  
49         final usuario = await MockFirestoreService.loginUsuario(correo, contrasena);  
50  
51         if (usuario != null) {  
52             currentUser = usuario;  
53             state = AuthState.success;  
54         } else {  
55             state = AuthState.error;  
56             errorMessage = 'Credenciales incorrectas';  
57         }  
58     }  
59 }  
60 }
```

Funcion de las 3 pruebas unitarias:

```
61 void main() {
62   Run | Debug
63   group('Pruebas unitarias del AuthViewModel', () {
64     late AuthViewModelMock viewModel;
65
66     setUp(() {
67       viewModel = AuthViewModelMock();
68     });
69
70     // LOGIN EXITOSO
71     Run | Debug
72     test('Login con credenciales correctas', () async {
73       await viewModel.login("jaime@gmail.com", "123456");
74
75       expect(viewModel.state, AuthState.success);
76       expect(viewModel.currentUser?.nombre, "Administrador");
77     });
78
79     // LOGIN FALLIDO
80     Run | Debug
81     test('Login con credenciales incorrectas', () async {
82       await viewModel.login("jaimeelias@gmail.com", "123");
83
84       expect(viewModel.state, AuthState.error);
85       expect(viewModel.errorMessage, "Credenciales incorrectas");
86     });
87
88     // CAMPOS VACÍOS
89     Run | Debug
90     test('Validar que no se puede iniciar sesión con campos vacíos', () async {
91       await viewModel.login("", "");
92
93       expect(viewModel.state, AuthState.error);
94       expect(viewModel.errorMessage.isNotEmpty, true);
95     });
96   });
97 }
```

Contenido de `quality-check.yml`

Captura de pantalla del contenido del archivo `quality-check.yml`:

The screenshot shows a code editor with three tabs at the top: 'quality-check.yml' (marked with a warning icon), 'README.md' (marked with an info icon), and 'main_test.dart' (marked with a question icon). The main content area displays the 'quality-check.yml' file, which defines a workflow named 'Quality Check' that runs on pushes to the 'main' branch or pull requests. It uses an 'ubuntu-latest' runner and includes steps for checkout, Java setup (JDK 11, Temurin distribution), Flutter setup (version 3.29.2, stable channel, cache enabled), dependency installation (working directory ./app_perufest, flutter pub get), code analysis (working directory ./app_perufest, flutter analyze --no-fatal-warnings --no-fatal-infos), and test execution (working directory ./app_perufest, flutter test). All steps are marked as completed.

```
! quality-check.yml U × ⓘ README.md M ⓘ main_test.dart U

.github > workflows > ! quality-check.yml
  1   name: Quality Check
  2
  3   on:
  4     push:
  5       branches: [main]
  6     pull_request:
  7       branches: [main]
  8
  9   jobs:
10     analyze-and-test:
11       runs-on: ubuntu-latest
12
13     steps:
14       - name: Checkout code
15         uses: actions/checkout@v4
16
17       - name: Set up JDK 11
18         uses: actions/setup-java@v4
19         with:
20           java-version: '11'
21           distribution: 'temurin'
22
23       - name: Set up Flutter
24         uses: subosito/flutter-action@v2
25         with:
26           flutter-version: '3.29.2'
27           channel: 'stable'
28           cache: true
29
30       - name: Install dependencies
31         working-directory: ./app_perufest
32         run: flutter pub get
33
34       - name: Analyze code
35         working-directory: ./app_perufest
36         run: flutter analyze --no-fatal-warnings --no-fatal-infos
37
38       - name: Run tests
39         working-directory: ./app_perufest
40         run: flutter test
```

Ejecución del workflow (evidencia)

Captura de la pestaña **Actions** mostrando la ejecución del workflow y el resultado **100% passed**:

The screenshot shows a GitHub Actions workflow named "Update Flutter version in CI workflow #3". The main page has a "Quality Check" link at the top left. On the right, there's a summary card for the workflow. Below it, under "Jobs", the "analyze-and-test" job is selected and expanded. The job status is "succeeded 2 hours ago in 2m 34s". The steps listed are: Set up job, Checkout code, Set up JDK 11, Set up Flutter, Install dependencies, Analyze code, and Run tests. The "Run tests" step is expanded, showing the following log output:

```
1 ► Run flutter test
9
10  ✓ Pruebas unitarias del AuthViewModel Login con credenciales correctas
11  ✓ Pruebas unitarias del AuthViewModel Login con credenciales incorrectas
12  ✓ Pruebas unitarias del AuthViewModel Validar que no se puede iniciar sesión con campos vacíos
13
14  ▶ 3 tests passed.
```

Explicación de lo realizado

- **¿Qué hace el workflow?**: Al hacer `git push` a `main` o crear un `pull_request` hacia `main`, GitHub Actions ejecuta los pasos: instalar Flutter, obtener dependencias, ejecutar `flutter analyze` y ejecutar `flutter test`. Esto permite detectar errores estáticos y regressiones en pruebas.
- **`flutter analyze`**: Verifica estilo, convenciones y errores estáticos.
- **`flutter test`**: Ejecuta las pruebas unitarias definidas en `test/`.