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

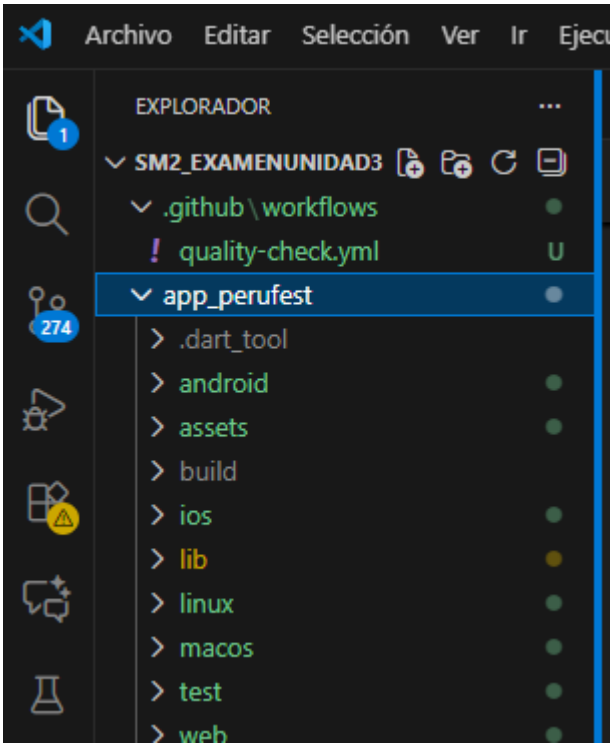
Objetivo: Implementar un flujo de trabajo (workflow) automatizado en GitHub Actions para realizar análisis de calidad sobre el proyecto móvil e integrar prácticas de DevOps.

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)

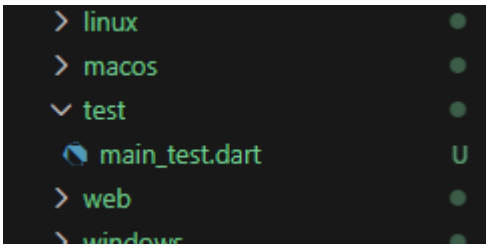
Captura de la estructura de carpetas mostrando `.github/workflows/`:



Captura de `test/main_test.dart`

Captura que muestre el archivo `test/main_test.dart` con las pruebas unitarias:

Contenido de `main_test.dart`



Mock de una servicio simulado:

```dart
import 'package:flutter_test/flutter_test.dart';
import 'package:app_perufest/models/usuario.dart';

// MOCK SIMULADO DEL SERVICE
class MockFirestoreService {
  static bool retornarUsuarioCorrecto = true;

  static Future<Usuario?> loginUsuario(String correo, String contrasena) async {
    if (!retornarUsuarioCorrecto) return null;

    if (correo == "jaime@gmail.com" && contrasena == "123456") {
      // Simula un registro de usuario válido
      return Usuario(
        id: "1",
        nombre: "Administrador",
        username: "admin",
        correo: correo,
        telefono: "924655655",
        rol: "administrador",
        contrasena: "hashX",
      );
    }

    return null;
  }

  static Future<bool> correoExiste(String correo) async => false;
  static Future<void> registrarUsuario(Usuario usuario) async {}
  static Future<Usuario?> obtenerUsuarioPorId(String id) async => null;
}
```

ViewModel de una clase para login:

```dart
// VIEWMODEL DE PRUEBA UNITARIA
enum AuthState { idle, loading, success, error }

class AuthViewModelMock {
  AuthState state = AuthState.idle;
  Usuario? currentUser;
  String errorMessage = '';

  Future<void> login(String correo, String contrasena) async {
    if (correo.isEmpty || contrasena.isEmpty) {
      state = AuthState.error;
      errorMessage = 'Los campos no pueden estar vacíos';
      return;
    }

    state = AuthState.loading;

    final usuario = await MockFirestoreService.loginUsuario(correo, contrasena);

    if (usuario != null) {
      currentUser = usuario;
      state = AuthState.success;
    } else {
      state = AuthState.error;
      errorMessage = 'Credenciales incorrectas';
    }
  }
}
```

Funcion de las 3 pruebas unitarias:

```dart
void main() {
  Run | Debug
  group('Pruebas unitarias del AuthViewModel', () {
    late AuthViewModelMock viewModel;

    setUp(() {
      viewModel = AuthViewModelMock();
    });

    // LOGIN EXITOSO
    Run | Debug
    test('Login con credenciales correctas', () async {
      await viewModel.login("jaime@gmail.com", "123456");

      expect(viewModel.state, AuthState.success);
      expect(viewModel.currentUser?.nombre, "Administrador");
    });

    // LOGIN FALLIDO
    Run | Debug
    test('Login con credenciales incorrectas', () async {
      await viewModel.login("jaimeelias@gmail.com", "123");

      expect(viewModel.state, AuthState.error);
      expect(viewModel.errorMessage, "Credenciales incorrectas");
    });

    // CAMPOS VACÍOS
    Run | Debug
    test('Validar que no se puede iniciar sesión con campos vacíos', () async {
      await viewModel.login("", "");

      expect(viewModel.state, AuthState.error);
      expect(viewModel.errorMessage.isNotEmpty, true);
    });
  });
}

```

Contenido de `quality-check.yml`

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

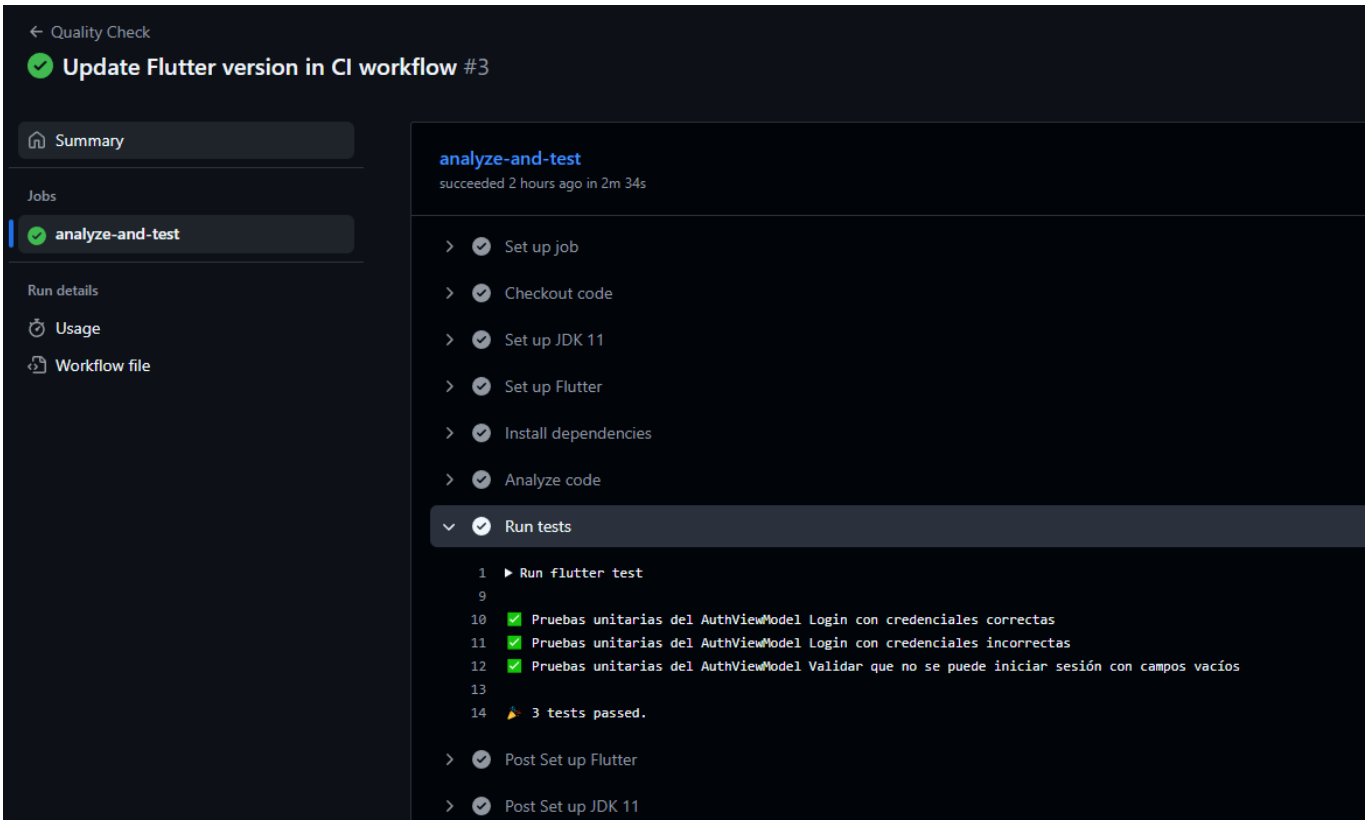
quality-check.yml U | README.md M | main_test.dart U

.github > workflows > quality-check.yml

```yaml
name: Quality Check

on:
  push:
    branches: [main]
  pull_request:
    branches: [main]

jobs:
  analyze-and-test:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Set up JDK 11
        uses: actions/setup-java@v4
        with:
          java-version: '11'
          distribution: 'temurin'

      - name: Set up Flutter
        uses: subosito/flutter-action@v2
        with:
          flutter-version: '3.29.2'
          channel: 'stable'
          cache: true

      - name: Install dependencies
        working-directory: ./app_perufest
        run: flutter pub get

      - name: Analyze code
        working-directory: ./app_perufest
        run: flutter analyze --no-fatal-warnings --no-fatal-infos

      - name: Run tests
        working-directory: ./app_perufest
        run: flutter test
```

Ejecución del workflow (evidencia)

Captura de la pestaña `Actions` mostrando la ejecución del workflow y el resultado `100% 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 regresiones en pruebas.
- `flutter analyze`: Verifica estilo, convenciones y errores estáticos.
- `flutter test`: Ejecuta las pruebas unitarias definidas en `test/`.