

Examen Unidad 3 - DevOps Móvil: Automatización y CI/CD

Curso: Soluciones Móviles II

Fecha: 25/11/2025

Estudiante: Alvaro Javier Contreras Lipa

Repositorio: https://github.com/AlvaroContreras13/-SM2_Examen_CICD

Descripción del Proyecto

Este proyecto implementa un sistema completo de **DevOps para aplicaciones móviles** usando GitHub Actions, que incluye:

1. **Automatización de Calidad** (Examen 1)
 2. **Pipeline CI/CD Completo** (Examen 2)
 3. **Generación automática de APK**
-

Objetivos Alcanzados

Examen 1: Automatización de Calidad

- Creación de repositorio público en GitHub
- Implementación de 19 pruebas unitarias del proyecto
- Configuración de workflow `quality-check.yml`
- Análisis automático de código con `flutter analyze`
- Ejecución automática de tests con `flutter test`

Examen 2: Pipeline CI/CD Completo

- Creación de 5 validadores basados en AuthService
 - Implementación de 5 pruebas unitarias adicionales
 - Configuración de workflow `ci-pipeline.yml`
 - Construcción automática del APK de release
 - Generación de artifacts descargables
 - **Total: 24 pruebas unitarias (100% passed)**
-

Pruebas Unitarias Implementadas

Examen 1: Pruebas del Proyecto (19 tests)

1. **TripStatus Tests (5 pruebas)**

- Validación de estados válidos e inválidos
- Verificación de textos descriptivos

- Validación de colores hexadecimales

2. AddressResolver Tests (5 pruebas)

- Manejo de datos nulos
- Resolución de direcciones desde coordenadas
- Formateo de ubicaciones

3. RatingService Tests (9 pruebas)

- Validación de rangos de calificación
- Prevención de auto-calificación
- Gestión de calificaciones de usuarios

Examen 2: Validators Tests (5 tests)

Se creó el archivo `lib/utils/validators.dart` con validaciones basadas en las reglas de negocio de AuthService:

#	Función	Descripción	Uso en la App
1	<code>isValidInstitutionalEmail</code>	Valida emails @virtual.upt.pe	Registro de usuarios
2	<code>isSecurePassword</code>	Valida contraseñas ≥ 6 caracteres	Firebase Authentication
3	<code>isValidDNI</code>	Valida DNI peruano (8 dígitos)	Identificación de usuarios
4	<code>isValidPeruvianPhone</code>	Valida teléfonos (9 dígitos, inicia con 9)	Contacto de usuarios
5	<code>isValidLicensePlate</code>	Valida placas vehiculares (ABC-123 o ABC-1234)	Registro de conductores

Estructura del Proyecto

```

SM2_Examen_CICD/
├── .github/
│   └── workflows/
│       ├── quality-check.yml      # Examen 1: Análisis + Tests
│       └── ci-pipeline.yml        # Examen 2: CI/CD + APK Build
├── test/
│   └── main_test.dart            # 24 pruebas unitarias (19 + 5)
└── lib/
    ├── constants/
    │   └── trip_status.dart
    ├── utils/
    │   ├── address_resolver.dart
    │   └── validators.dart         # Examen 2
    └── services/
        └── rating_service.dart

```

```
└── auth_service.dart  
└── ...  
└── README.md
```

⌚ Workflows Configurados

1. Quality Check (Examen 1)

Archivo: `.github/workflows/quality-check.yml`

Pasos:

1. Checkout del código
2. Instalación de Flutter 3.29.0
3. Instalación de dependencias (`flutter pub get`)
4. Ejecución de tests (`flutter test`)

Se ejecuta en: Cada push o pull request a `main`

2. Mobile CI/CD Pipeline (Examen 2)

Archivo: `.github/workflows/ci-pipeline.yml`

Pasos:

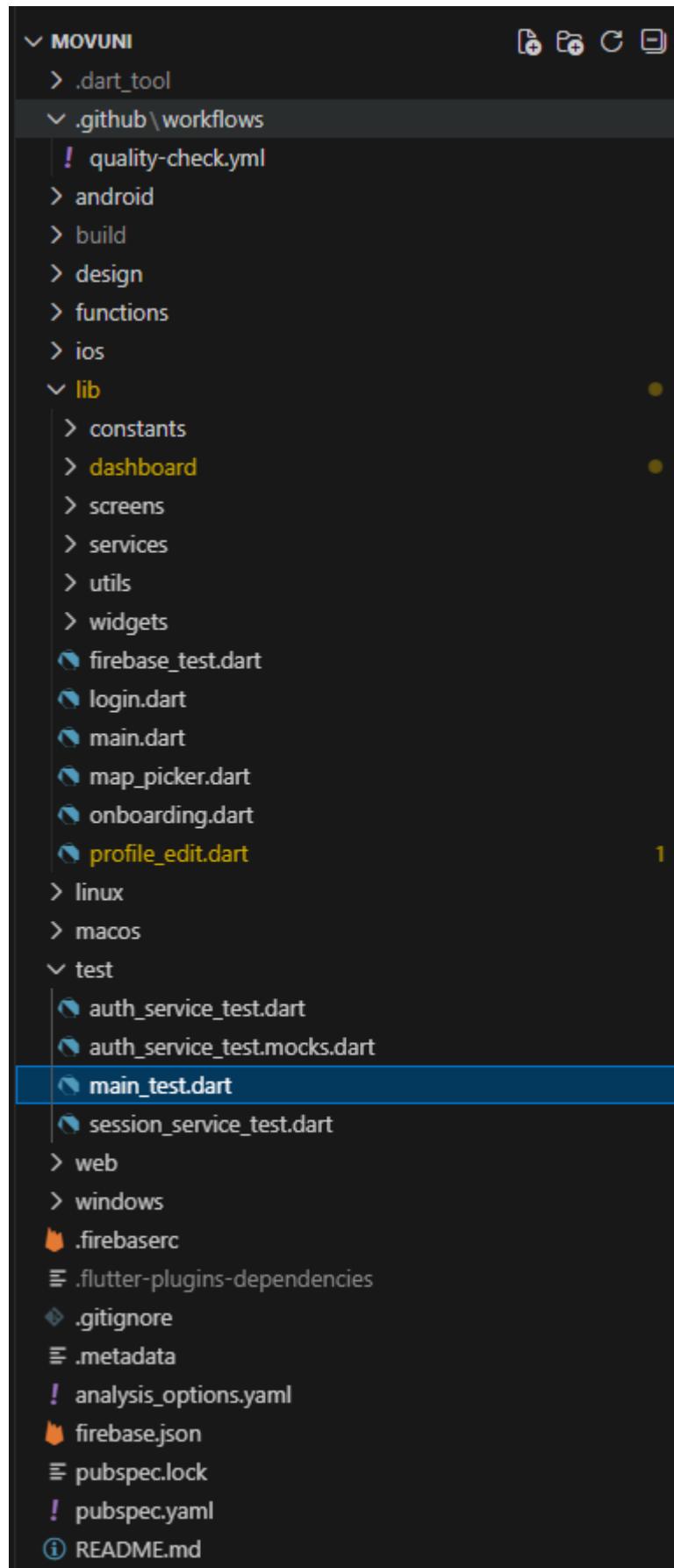
1. Checkout del código
2. Instalación de Flutter 3.29.0
3. Instalación de dependencias (`flutter pub get`)
4. Análisis de calidad (`flutter analyze`)
5. Ejecución de pruebas unitarias (`flutter test`)
6. **Construcción del APK** (`flutter build apk --release`)
7. **Upload del APK como Artifact** (descargable)

Se ejecuta en: Cada push o pull request a `main`

📸 Evidencias

Examen 1: Automatización de Calidad

1. Estructura de carpetas `.github/workflows/`



2. Contenido del archivo quality-check.yml

```
.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:
11       runs-on: ubuntu-latest
12
13     steps:
14       - name: Checkout code
15         uses: actions/checkout@v3
16
17       - name: Set up Flutter
18         uses: subosito/flutter-action@v2
19         with:
20           flutter-version: '3.19.0'
21           channel: 'stable'
22
23       - name: Install dependencies
24         run: flutter pub get
25
26       - name: Analyze code
27         run: flutter analyze
28
29       - name: Run tests
30         run: flutter test
```

3. Ejecución exitosa del workflow

The screenshot shows the GitHub Actions interface for the 'quality-check.yml' workflow. The 'analyze' job is currently running, indicated by a green progress bar. The 'Checkout code' step has completed successfully. The 'Set up job' step is in progress. The 'Flutter' step has failed, as indicated by a red error icon. The 'Install dependencies' and 'Analyze code' steps have not yet started. The 'Run tests' step has not yet started. The 'Logs' tab is selected, showing the terminal output of the workflow steps. The logs show the checkout process, the setup of the Flutter environment, and the initial stages of the flutter pub get command. A significant portion of the logs is highlighted in red, indicating errors related to failed git fetch operations.

```
analyze
Started 5m 20s ago
  1 ► Run actions/checkout@v3
  2 Syncing repository: AlvaroContreras13/SM2_ExamenUnidad3
  3 ► Getting Git version info
  4 Temporarily overriding HOME='/home/runner/work/_temp/cc216507-2eda-4759-9804-556f0107769c' before making global git config changes
  5 Adding repository directory to the temporary git global config as a safe directory
  6 /usr/bin/git config --global --add safe.directory /home/runner/work/SM2_ExamenUnidad3/SM2_ExamenUnidad3
  7 Deleting the contents of '/home/runner/work/SM2_ExamenUnidad3/SM2_ExamenUnidad3'
  8 ► Initializing the repository
  9 ► Disabling automatic garbage collection
 10 ► Setting up auth
 11 ► Fetching the repository
 12 /usr/bin/git -c protocol.version=2 fetch --no-tags --prune --progress --no-recurse-submodules --depth=1 origin +aa2814b87b57d103a645347991a07dff3819e06a:refs/remotes/origin/main
 13 remote: Internal Server Error
 14 Error: fatal: unable to access 'https://github.com/AlvaroContreras13/SM2_ExamenUnidad3/': The requested URL returned error: 500
 15 The process '/usr/bin/git' failed with exit code 128
 16 Waiting 11 seconds before trying again
 17 /usr/bin/git -c protocol.version=2 fetch --no-tags --prune --progress --no-recurse-submodules --depth=1 origin +aa2814b87b57d103a645347991a07dff3819e06a:refs/remotes/origin/main
 18 Error: fatal: unable to access 'https://github.com/AlvaroContreras13/SM2_ExamenUnidad3/': The requested URL returned error: 500
 19 The process '/usr/bin/git' failed with exit code 128
 20 Waiting 10 seconds before trying again
 21 /usr/bin/git -c protocol.version=2 fetch --no-tags --prune --progress --no-recurse-submodules --depth=1 origin +aa2814b87b57d103a645347991a07dff3819e06a:refs/remotes/origin/main
```

4. Detalle de la ejecución

Reporte de Pruebas Unitarias
main_test.dart - Proyecto MovUni
Generado el 16/11/2025

TESTS EXITOSOS **20** 100% de éxito

TOTAL DE TESTS **20** 4 grupos

Detalles de las Pruebas

- TripStatus Tests** (5 tests)
 - isValid debe retornar true para estados válidos
 - isValid debe retornar false para estados inválidos
 - getDisplayText debe retornar el texto correcto para cada estado
 - getColors debe retornar colores válidos en formato hexadecimal
 - allStatus debe contener exactamente 4 estados
- AddressResolver Tests** (5 tests)
 - resolveAddressFromData debe retornar defaultName cuando location es null
 - resolveAddressFromData debe retornar el nombre si ya existe y no es coordinada
 - resolveAddressFromData debe retornar coordenadas formateadas cuando no hay nombre válido
 - resolveAddressFromData debe retornar defaultName cuando no hay coordenadas válidas
 - resolveAddressFromData debe manejar coordenadas como int o double
- RatingService Tests** (9 tests)
 - createRating debe lanzar excepción si rating está fuera del rango 1-5
 - createRating debe lanzar excepción si el usuario intenta calificarse a sí mismo
 - createRating debe crear una calificación válida correctamente
 - getUserAverageRating debe retornar 5.0 cuando el usuario no existe
 - getUserAverageRating debe retornar el rating del usuario si existe
 - getUserTotalRatings debe retornar 0 cuando el usuario no tiene calificaciones
 - getUserTotalRatings debe retornar el total correcto de calificaciones
 - getUserTotalRatings debe retornar 0 cuando el usuario no tiene calificaciones
 - getUserTotalRatings debe retornar el total correcto de calificaciones

Examen 2: Pipeline CI/CD Completo

1. Pruebas Unitarias Exitosas (24/24 passed)

Summary

build-and-deploy Started 5m 49s ago

Jobs

build-and-deploy

Run details

Usage

Workflow file

build-and-deploy

Started 5m 49s ago

1 > **Install Dependencies** 15s

> **Code Quality Check** 16s

> **Run Unit Tests** 11s

1 ► Run flutter test test/main_test.dart

7

8 ✘ TripStatus Tests isValid debe retornar true para estados válidos

9 ✘ TripStatus Tests isValid debe retornar false para estados inválidos

10 ✘ TripStatus Tests getDisplayText debe retornar el texto correcto para cada estado

11 ✘ TripStatus Tests getColors debe retornar colores válidos en formato hexadecimal

12 ✘ TripStatus Tests allStatus debe contener exactamente 4 estados

13 ✘ AddressResolver Tests resolveAddressFromData debe retornar defaultName cuando location es null

14 ✘ AddressResolver Tests resolveAddressFromData debe retornar el nombre si ya existe y no es coordinada

15 ✘ AddressResolver Tests resolveAddressFromData debe retornar coordenadas formateadas cuando no hay nombre válido

16 ✘ AddressResolver Tests resolveAddressFromData debe retornar defaultName cuando no hay coordenadas válidas

17 ✘ AddressResolver Tests resolveAddressFromData debe manejar coordenadas como int o double

18 ✘ RatingService Tests createRating debe lanzar excepción si rating está fuera del rango 1-5

19 ✘ RatingService Tests createRating debe lanzar excepción si el usuario intenta calificarse a sí mismo

20 ✘ RatingService Tests createRating debe crear una calificación válida correctamente

21 ✘ RatingService Tests getUserAverageRating debe retornar 5.0 cuando el usuario no existe

22 ✘ RatingService Tests getUserAverageRating debe retornar el rating del usuario si existe

23 ✘ RatingService Tests getUserTotalRatings debe retornar 0 cuando el usuario no tiene calificaciones

24 ✘ RatingService Tests getUserTotalRatings debe retornar el total correcto de calificaciones

25 ✘ RatingService Tests createRatings debe retornar true cuando no existe calificación previa

26 ✘ RatingService Tests createRatings debe retornar false cuando ya existe una calificación

27 ✘ Validators Tests - Examen CI/CD 1. Validar Email Institucional (.UPT)

28 ✘ Validators Tests - Examen CI/CD 2. Validar Contraseña Segura (mínimo 6 caracteres)

29 ✘ Validators Tests - Examen CI/CD 3. Validar DNI Peruano (8 dígitos)

30 ✘ Validators Tests - Examen CI/CD 4. Validar Teléfono Peruano (9 dígitos, inicia con 9)

31 ✘ Validators Tests - Examen CI/CD 5. Validar Placa Vehicular Peruana (formato ABC-123 o ABC-1234)

32

33 ▲ 24 tests passed.

Resultado: ✓ **24 tests passed** (100%)

- 19 tests del proyecto original
- 5 tests nuevos de Validators

2. Construcción del APK Exitosa

```

Build Application
14 "Install NDK (Side by side) 27.0.12077973 > 27.0.12077973" complete.
15 "Install NDK (Side by side) 27.0.12077973 > 27.0.12077973" finished.
16 Checking the license for package Android SDK Platform 31 in /usr/local/lib/android/sdk/licenses
17 License for package Android SDK Platform 31 accepted.
18 Preparing "Install Android SDK Platform 31 (revision 1)".
19 "Install Android SDK Platform 31 (revision 1)" ready.
20 Installing Android SDK Platform 31 in /usr/local/lib/android/sdk/platforms/android-31
21 "Install Android SDK Platform 31 (revision 1)" complete.
22 "Install Android SDK Platform 31 (revision 1)" finished.
23 Checking the license for package Android SDK Platform 33 in /usr/local/lib/android/sdk/licenses
24 License for package Android SDK Platform 33 accepted.
25 Preparing "Install Android SDK Platform 33 (revision 3)".
26 "Install Android SDK Platform 33 (revision 3)" ready.
27 Installing Android SDK Platform 33 in /usr/local/lib/android/sdk/platforms/android-33
28 "Install Android SDK Platform 33 (revision 3)" complete.
29 "Install Android SDK Platform 33 (revision 3)" finished.
30 Font asset "MaterialIcons-Regular.otf" was tree-shaken, reducing it from 1645184 to 11212 bytes (99.3% reduction). Tree-shaking can be disabled by providing the --no-tree-shake-icons flag when building your app.
31 Checking the license for package CMake 3.22.1 in /usr/local/lib/android/sdk/licenses
32 License for package CMake 3.22.1 accepted.
33 Preparing "Install CMake 3.22.1 v.3.22.1".
34 "Install CMake 3.22.1 v.3.22.1" ready.
35 Installing CMake 3.22.1 in /usr/local/lib/android/sdk/cmake/3.22.1
36 "Install CMake 3.22.1 v.3.22.1" complete.
37 "Install CMake 3.22.1 v.3.22.1" finished.
38 Note: Some input files use or override a deprecated API.
39 Note: Recompile with -Xlint:deprecation for details.
40 Caught exception: Already watching path: /home/runner/work/-SM2_Examen_CICD/-SM2_Examen_CICD/android
41 Running Gradle task 'assembleRelease'...
402.0s
42 ✓ Built build/app/outputs/flutter-apk/app-release.apk (55.3MB)

```

```

Upload Artifact
1 ► Run actions/upload-artifact@v4
12
12 With the provided path, there will be 1 file uploaded
13 Artifact name is valid!
14 Root directory input is valid!
15 Beginning upload of artifact content to blob storage
16 Uploaded bytes 8388608
17 Uploaded bytes 16777216

```

Resultado: APK generado en [build/app/outputs/flutter-apk/app-release.apk](#)

3. Artifact Descargable

The screenshot shows the GitHub Actions run summary for the workflow `ci-pipeline.yml`. The job `build-and-deploy` is listed as successful, completed 11 minutes ago. It produced one artifact named `app-release`, which is a 252 MB file with a SHA256 digest of `sha256:7928ba03a7bfacccb01acf42b22cf22a21c3b5db61f2560495e5ebb8e02b85`. There are download and delete icons next to the artifact link.

Ubicación: GitHub Actions → Artifacts → [app-release.apk](#)

Disponible para descarga directa desde GitHub

Resultados Finales

Examen 1

- **Tests ejecutados:** 19/19
- **Tests pasados:** 19 (100%)
- **Tests fallidos:** 0
- **Análisis de código:** Completado

Examen 2

- **Tests ejecutados:** 24/24
 - **Tests pasados:** 24 (100%)
 - **Tests fallidos:** 0
 - **Análisis de código:** Completado
 - **APK generado:** Sí
 - **Artifact disponible:** Sí
-

🚀 Ejecución Local

Instalar dependencias

```
flutter pub get
```

Ejecutar todas las pruebas

```
flutter test test/main_test.dart
```

Construir APK localmente

```
flutter build apk --release
```

📦 Dependencias de Testing

```
environment:  
  sdk: '>=3.7.0 <4.0.0'  
  flutter: ">=3.29.0"  
  
dev_dependencies:  
  flutter_test:  
    sdk: flutter  
  flutter_lints: ^5.0.0  
  build_runner: ^2.8.0  
  fake_cloud_firestore: ^3.0.3  
  firebase_auth_mocks: 0.14.2  
  mockito: ^5.5.1
```

🔧 Comandos Ejecutados en CI/CD

```
# 1. Instalación  
flutter pub get  
  
# 2. Pruebas unitarias  
flutter test test/main_test.dart  
  
# 3. Construcción del APK  
flutter build apk --release  
  
# 4. Upload del artifact  
# (automático via GitHub Actions)
```

Conclusiones

Se implementó exitosamente un **pipeline completo de DevOps** para aplicaciones móviles que:

Examen 1: Automatización de Calidad

- Valida la calidad del código automáticamente
- Ejecuta pruebas unitarias en cada commit
- Garantiza estándares de código
- Previene integración de código con errores

Examen 2: CI/CD Completo

- Todo lo anterior +
- **Construye el APK automáticamente**
- **Genera artifacts descargables**
- **Listo para distribución inmediata**
- **Proceso 100% automatizado**