

# SM2\_Examen\_CICD

---

**Nombre:** Ximena Ortiz

**Curso:** Soluciones Móviles II



---

## Repositorio

**URL del Repositorio:** [https://github.com/ximena-ortiz/SM2\\_Examen\\_CICD](https://github.com/ximena-ortiz/SM2_Examen_CICD)

---

## Parte 1: Lógica y Pruebas Unitarias

### Funciones Implementadas

Se creó la clase **Validator** con las siguientes funciones utilitarias:

#	Función	Descripción	Criterio de Prueba
1	<code>isValidEmail()</code>	Valida formato de email	Retorna true si contiene "@" y "."
2	<code>isStrongPassword()</code>	Valida seguridad de contraseña	Retorna true si tiene más de 6 caracteres
3	<code>calcularDescuento()</code>	Calcula precio con descuento	Retorna el precio final después de aplicar el porcentaje
4	<code>isRangoValido()</code>	Valida rango numérico	Retorna true si el número está entre 1 y 10 (inclusive)
5	<code>toUpperText()</code>	Convierte texto a mayúsculas	Retorna el texto completamente capitalizado

### Código de la Clase Validator

```
Frontend > lib > models > validator.dart
1  class Validator {
2    static bool isValidEmail(String email) {
3      return email.contains('@') && email.contains('.');
4    }
5
6    static bool isStrongPassword(String password) {
7      return password.length > 6;
8    }
9
10   static double calcularDescuento(double precio, double porcentaje) {
11     final descuento = precio * (porcentaje / 100);
12     return precio - descuento;
13   }
14
15   static bool isRangoValido(int numero) {
16     return numero >= 1 && numero <= 10;
17   }
18
19   static String toUpperText(String texto) {
20     return texto.toUpperCase();
21   }
22 }
23 |
```

## Tests Unitarios

```
Frontend > test > validator_test.dart
1 import 'package:flutter_test/flutter_test.dart';
2 import 'package:english_app/models/validator.dart';
3 void main() {
4     test('Validar Email', () {
5         expect(Validator.isValidEmail('correo@ejemplo.com'), true);
6         expect(Validator.isValidEmail('correo_malo'), false);
7     });
8
9     test('Seguridad Contraseña', () {
10        expect(Validator.isStrongPassword('1234567'), true);
11        expect(Validator.isStrongPassword('123'), false);
12    });
13
14     test('Calculadora Descuento', () {
15        final resultado = Validator.calcularDescuento(100.0, 10.0);
16        expect(resultado, 90.0);
17    });
18
19     test('Rango Válido', () {
20        expect(Validator.isRangoValido(5), true);
21        expect(Validator.isRangoValido(0), false);
22        expect(Validator.isRangoValido(11), false);
23    });
24
25     test('Texto a Mayúsculas', () {
26        expect(Validator.toUpperCaseText('hola mundo'), 'HOLA MUNDO');
27    });
28 }
29
```

## Evidencia de Tests Locales

Los tests se ejecutaron localmente con el comando flutter test y todos pasaron exitosamente:

```
PS C:\Users\HP\Desktop\movilesFinal> cd .\Frontend\
① PS C:\Users\HP\Desktop\movilesFinal\Frontend> flutter test
● 00:01 +5: All tests passed!
② PS C:\Users\HP\Desktop\movilesFinal\Frontend>
```

## Audite la calidad del código (Linting).

```
● PS C:\Users\HP\Desktop\movilesFinal\Frontend> flutter analyze
Analyzing Frontend...
No issues found! (ran in 15.4s)
② PS C:\Users\HP\Desktop\movilesFinal\Frontend>
```

## Parte 2: Configuración del Pipeline CI/CD

### Workflow de GitHub Actions

Se configuró el archivo `.github/workflows/ci-pipeline.yml` con el siguiente contenido:

```
.github > workflows > ! ci-pipeline.yml
1   name: Mobile CI/CD Pipeline
2
3   on:
4     push:
5       branches: [master]
6     pull_request:
7       branches: [master]
8     workflow_dispatch:
9
10  jobs:
11    build-and-deploy:
12      runs-on: ubuntu-latest
13
14    steps:
15      - name: Checkout Code
16        uses: actions/checkout@v3
17
18      - name: Setup Flutter
19        uses: subosito/flutter-action@v2
20        with:
21          flutter-version: '3.35.5'
22
23      - name: Install Dependencies
24        working-directory: Frontend
25        run: flutter pub get
26
27      - name: Code Quality Check
28        working-directory: Frontend
29        run: flutter analyze
30
31      - name: Run Unit Tests
32        working-directory: Frontend
33        run: flutter test
34
35      - name: Build Application
36        working-directory: Frontend
37        run: flutter build apk --release
38
39      - name: Upload Artifact
40        uses: actions/upload-artifact@v4
41        with:
42          name: app-release
43          path: Frontend/build/app/outputs/flutter-apk/*.apk
44
```

---

## Parte 3: Evidencias de Ejecución

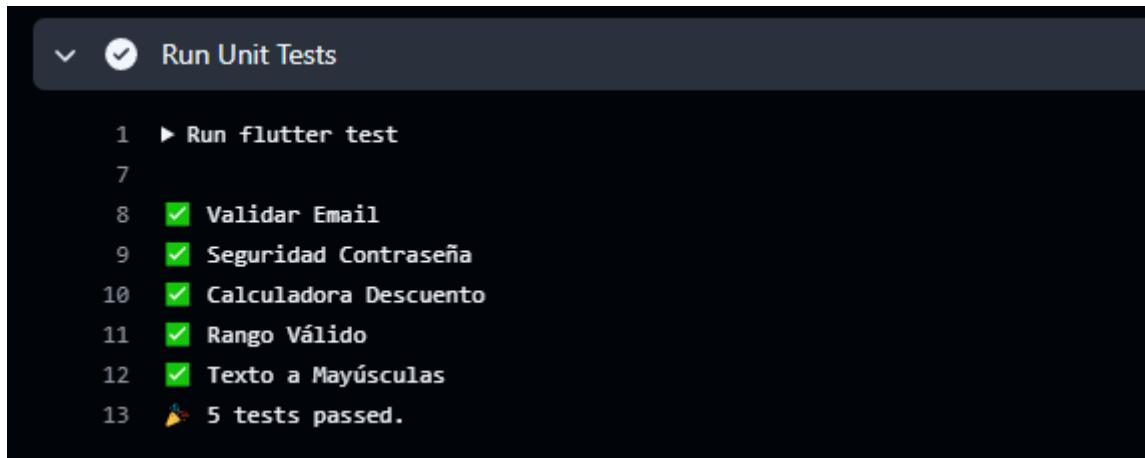
Badge de Estado del Pipeline



**Estado:**  **Passing** - El pipeline se ejecuta correctamente en cada push.

## Evidencia de Tests Unitarios

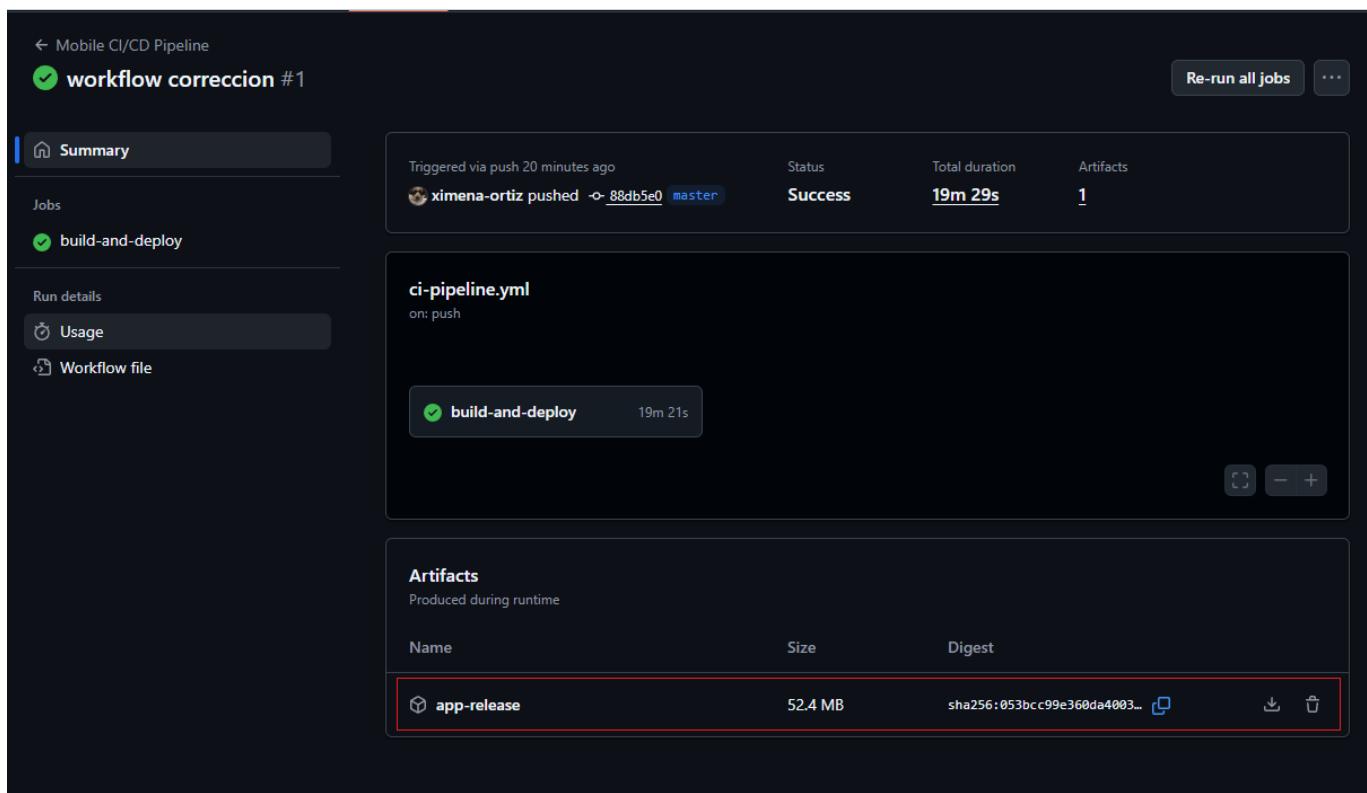
El log de GitHub Actions muestra que los 5 tests unitarios pasaron exitosamente:



The screenshot shows a GitHub Actions test log titled "Run Unit Tests". It lists 13 steps, numbered 1 to 13. Steps 1 through 12 each have a green checkmark and a descriptive name: "Run flutter test", "Validar Email", "Seguridad Contraseña", "Calculadora Descuento", "Rango Válido", and "Texto a Mayúsculas". Step 13 is a summary: "5 tests passed.".

## Evidencia de Construcción y Artefacto

El APK se generó exitosamente y está disponible para descarga en la sección de Artifacts:



The screenshot shows the "Mobile CI/CD Pipeline" workflow run details. The workflow is named "workflow correccion #1". The "Summary" card shows the run was triggered via push 20 minutes ago by "ximena-ortiz" on branch "master", with a status of "Success" and a total duration of "19m 29s". The "build-and-deploy" job is listed under "Jobs". The "Usage" card shows the "ci-pipeline.yml" file with a single step: "build-and-deploy" (status: success, duration: 19m 21s). The "Workflow file" card shows the YAML code for the workflow. The "Artifacts" section shows a single artifact named "app-release" produced during runtime, with a size of 52.4 MB and a digest of "sha256:053bcc99e360da4003...".

### Detalles del Artefacto:

- **Nombre:** app-release
- **Formato:** APK
- **Tamaño:** 52.4 MB

## Etapas del Pipeline

1. **Checkout Code:** Descarga el código fuente del repositorio
  2. **Setup Flutter:** Configura el entorno Flutter 3.35.5
  3. **Install Dependencies:** Ejecuta flutter pub get
  4. **Code Quality Check:** Analiza el código con flutter analyze
  5. **Run Unit Tests:** Ejecuta las pruebas unitarias con flutter test
  6. **Build Application:** Compila el APK con flutter build apk
  7. **Upload Artifact:** Sube el APK generado como artefacto descargable
-