SM2_Examen_CICD

Nombre: Ximena Ortiz
Curso: Soluciones Moviles II

Repositorio

URL del Repositorio: 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	`isValidEmail()`	Valida formato de email	Retorna true si contiene "@" y "."
2	`isStrongPassword()`	Valida seguridad de contraseña	Retorna true si tiene más de 6 caracteres
3	`calcularDescuento()`	Calcula precio con descuento	Retorna el precio final después de aplicar el porcentaje
4	`isRangoValido()`	Valida rango numérico	Retorna true si el número está entre 1 y 10 (inclusive)
5	`toUpperText()`	Convierte texto a mayúsculas	Retorna el texto completamente capitalizado

Código de la Clase Validator

Frontend > lib > models > validator.dart

```dart
class Validator {
  static bool isValidEmail(String email) {
    return email.contains('@') && email.contains('.');
  }

  static bool isStrongPassword(String password) {
    return password.length > 6;
  }

  static double calcularDescuento(double precio, double porcentaje) {
    final descuento = precio * (porcentaje / 100);
    return precio - descuento;
  }

  static bool isRangoValido(int numero) {
    return numero >= 1 && numero <= 10;
  }

  static String toUpperText(String texto) {
    return texto.toUpperCase();
  }
}

```

Tests Unitarios

Frontend > test > validator_test.dart

```dart
import 'package:flutter_test/flutter_test.dart';
import 'package:english_app/models/validator.dart';
void main() {
  test('Validar Email', () {
    expect(Validator.isValidEmail('correo@ejemplo.com'), true);
    expect(Validator.isValidEmail('correo_malo'), false);
  });

  test('Seguridad Contraseña', () {
    expect(Validator.isStrongPassword('1234567'), true);
    expect(Validator.isStrongPassword('123'), false);
  });

  test('Calculadora Descuento', () {
    final resultado = Validator.calcularDescuento(100.0, 10.0);
    expect(resultado, 90.0);
  });

  test('Rango Válido', () {
    expect(Validator.isRangoValido(5), true);
    expect(Validator.isRangoValido(0), false);
    expect(Validator.isRangoValido(11), false);
  });

  test('Texto a Mayúsculas', () {
    expect(Validator.toUpperText('hola mundo'), 'HOLA MUNDO');
  });
}

```

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\
PS C:\Users\HP\Desktop\movilesFinal\Frontend> flutter test
00:01 +5: All tests passed!
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)
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

```yaml
name: Mobile CI/CD Pipeline

on:
  push:
    branches: [master]
  pull_request:
    branches: [master]
  workflow_dispatch:

jobs:
  build-and-deploy:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout Code
        uses: actions/checkout@v3

      - name: Setup Flutter
        uses: subosito/flutter-action@v2
        with:
          flutter-version: '3.35.5'

      - name: Install Dependencies
        working-directory: Frontend
        run: flutter pub get

      - name: Code Quality Check
        working-directory: Frontend
        run: flutter analyze

      - name: Run Unit Tests
        working-directory: Frontend
        run: flutter test

      - name: Build Application
        working-directory: Frontend
        run: flutter build apk --release

      - name: Upload Artifact
        uses: actions/upload-artifact@v4
        with:
          name: app-release
          path: Frontend/build/app/outputs/flutter-apk/*.apk

```

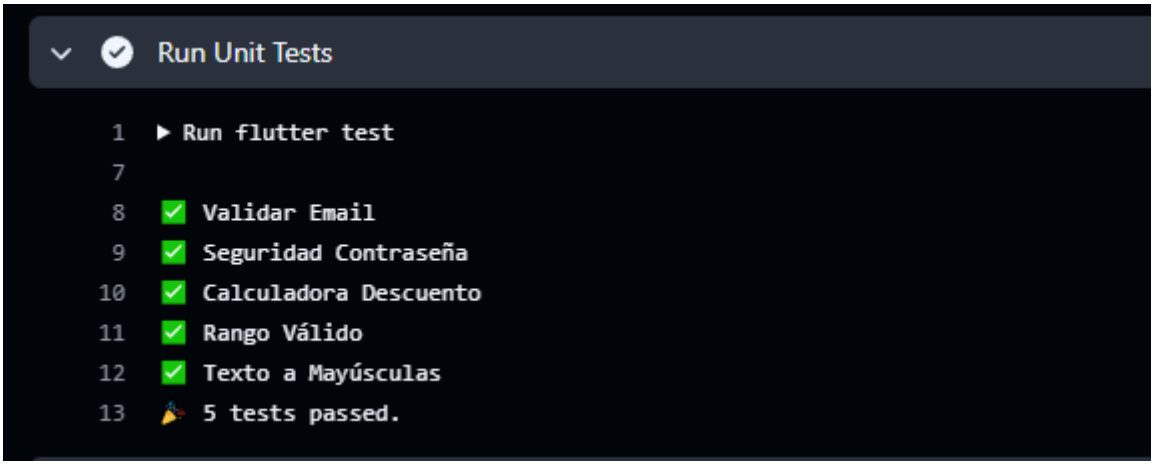
Parte 3: Evidencias de Ejecución

Badge de Estado del Pipeline

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

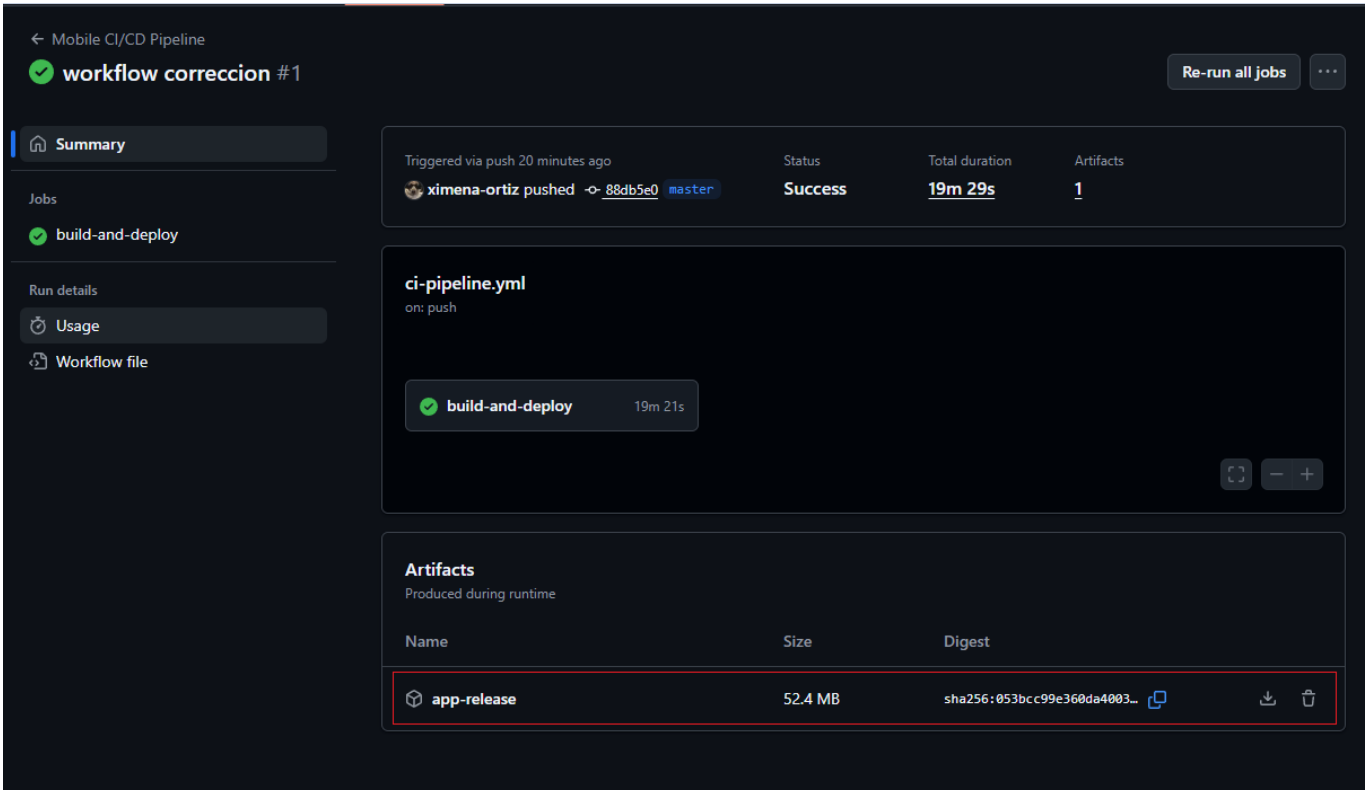
Evidencia de Tests Unitarios

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



Evidencia de Construcción y Artefacto

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



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

---