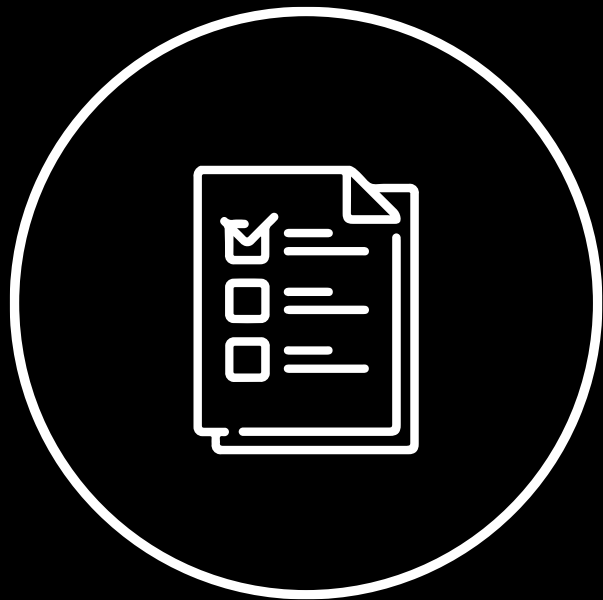


Testing Flutter

Smith Camilo Quevedo Rozo
Edison Fernando Aza Casanova
Grupó Bancolombia





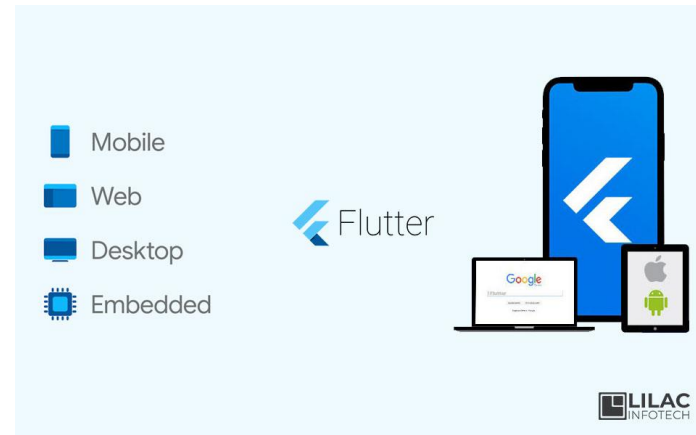
CONTENIDO

1. Introducción a Flutter
2. Instalación de Flutter
3. Configuración del entorno de desarrollo
4. Creación de un nuevo proyecto Flutter
5. Estructura básica de la app de login-Choucair
6. Tipos de pruebas en Flutter
7. Gestión de dependencias
8. Ejecución de pruebas en local
9. Ejercicio practico
10. Preguntas

¿Qué es Flutter?

Flutter es el SDK de aplicaciones móviles de Google para crear interfaces nativas de alta calidad en iOS y Android. Entre sus ventajas se incluyen:

- Desarrollo rápido: Flutter permite ver los cambios en el código de inmediato mediante su característica de "hot reload".
- UI expresiva y flexible: Gracias a su amplio conjunto de widgets personalizables, Flutter facilita la creación de interfaces de usuario atractivas.
- Rendimiento nativo: Flutter compila directamente a código nativo, lo que garantiza un rendimiento óptimo.



¿Qué es un Widget en Flutter?

"En Flutter, casi todo es un Widget"

Un widget es un componente de la interfaz gráfica de usuario que muestra información o permite al usuario interactuar con una aplicación de una manera específica. Son fundamentales en Flutter, ya que definen las vistas que componen la pantalla que ve el usuario.

Los widgets en Flutter se construyen utilizando un framework moderno inspirado en React Native. La idea principal es crear la interfaz de usuario a partir de widgets.

Los widgets describen cómo debería lucir su vista según su configuración y estado actuales. Cuando el estado de un widget cambia, este reconstruye su descripción, la cual es comparada con la que posee el framework. Esto permite determinar los cambios mínimos necesarios en el árbol de renderizado para pasar de un estado a otro.

Todo en Flutter es un widget, desde un texto hasta una pantalla completa.





Configuración de entorno



Visual Studio Code

android
studio





Flutter

Para instalar Flutter en Windows, primero debe descargar el archivo ZIP del SDK de Flutter desde la [página oficial](#) y descomprimirlo en la ubicación deseada, por ejemplo, C:\dev\Flutter. A continuación, debe configurar la variable de entorno PATH para incluir la ruta C:\dev\flutter\bin. Una vez hecho esto, puede verificar la instalación abriendo una terminal de comandos (CMD) y ejecutando flutter doctor, lo que descargará las dependencias adicionales y verificará la instalación.

1 Download then install Flutter

To install Flutter, download the Flutter SDK bundle from its archive, move the bundle to where you want it stored, then extract the SDK.

1. Download the following installation bundle to get the latest stable release of the Flutter SDK.

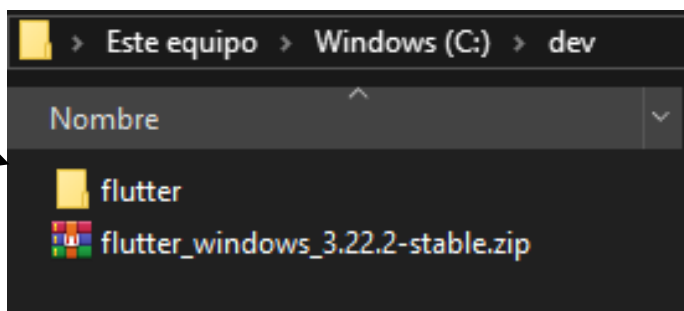
[flutter_windows_3.22.2-stable.zip](#)

For other release channels, and older builds, check out the [SDK archive](#).

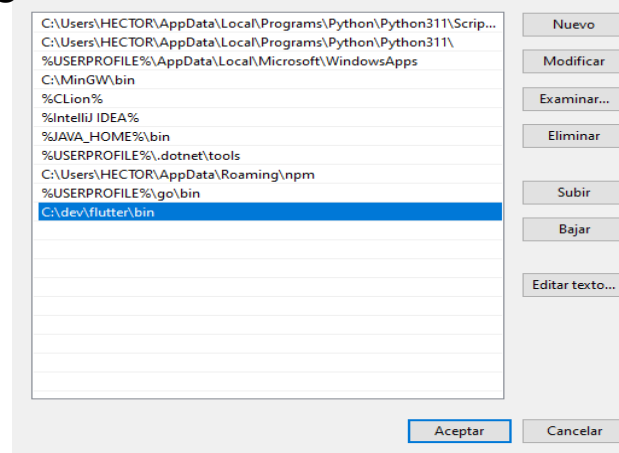
The Flutter SDK should download to the Windows default download directory:
%USERPROFILE%\Downloads.

If you changed the location of the Downloads directory, replace this path with that path. To find your Downloads directory location, check out this [Microsoft Community post](#).

2



3 Editar variable de entorno



4

```
C:\Users\HECTOR>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.22.2, on Microsoft Windows [Versi n 10.0.19045.4598], locale es-CO)
[✓] Windows Version (Installed version of Windows is version 10 or higher)
[✓] Android toolchain - develop for Android devices (Android SDK version 35.0.0)
[✓] Chrome - develop for the web
[X] Visual Studio - develop Windows apps
    X Visual Studio not installed; this is necessary to develop Windows apps.
      Download at https://visualstudio.microsoft.com/downloads/.
      Please install the "Desktop development with C++" workload, including all of its default components
[✓] Android Studio (version 2024.1)
[✓] VS Code, 64-bit edition (version 1.90.2)
[✓] Connected device (3 available)
[✓] Network resources

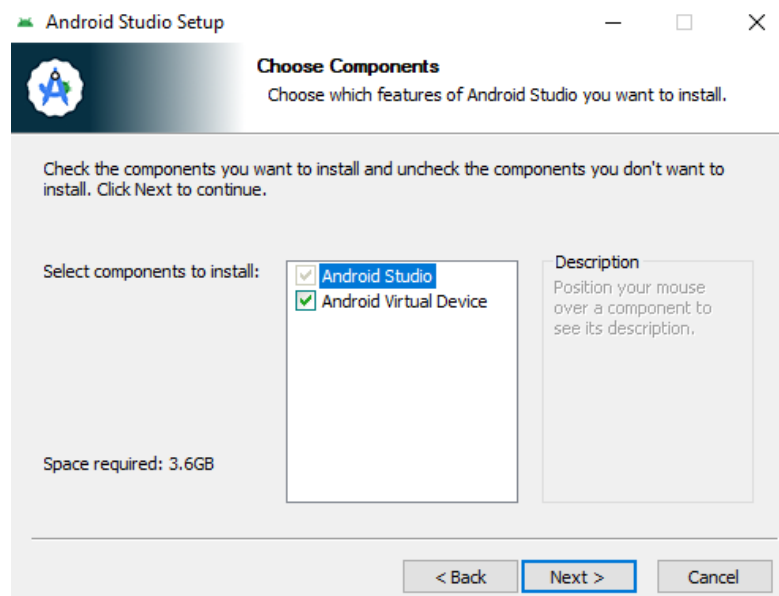
! Doctor found issues in 1 category.
```



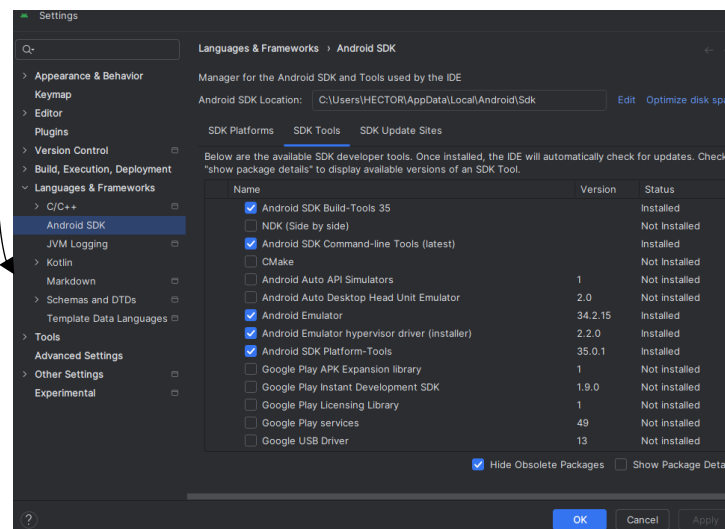
Android Studio

Para configurar Android Studio, descarga e instala desde el [sitio oficial](#), asegurándote de incluir el Android Virtual Device. Abre Android Studio y dirígete a Settings > SDK Manager > SDK Tools y selecciona la opción SDK Command-Line Tools para instalar las herramientas de línea de comandos.

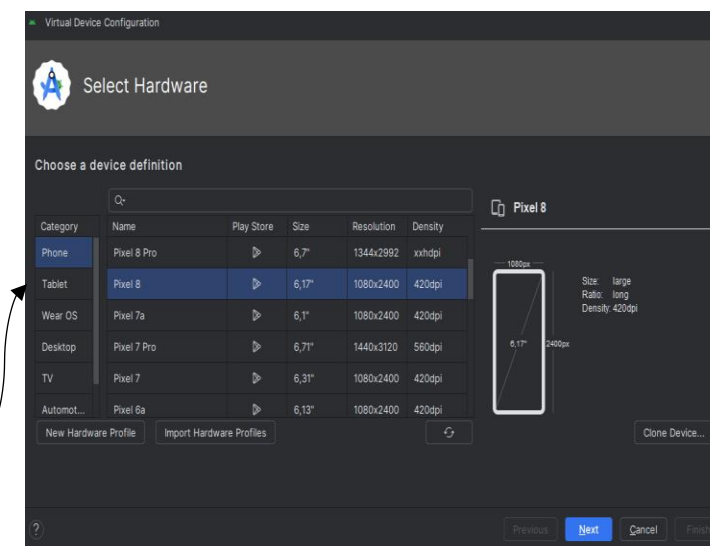
1



2



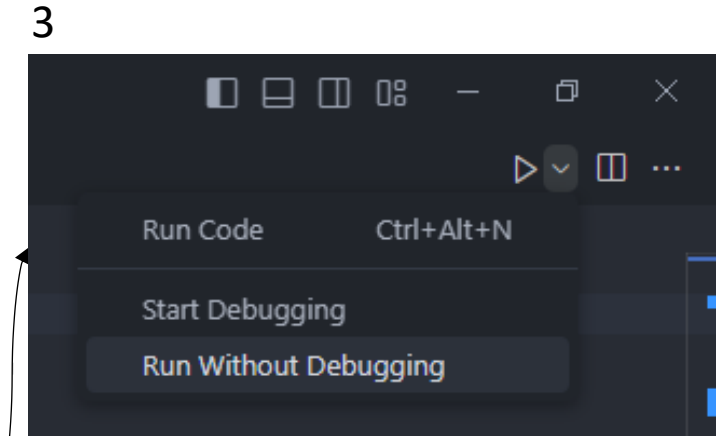
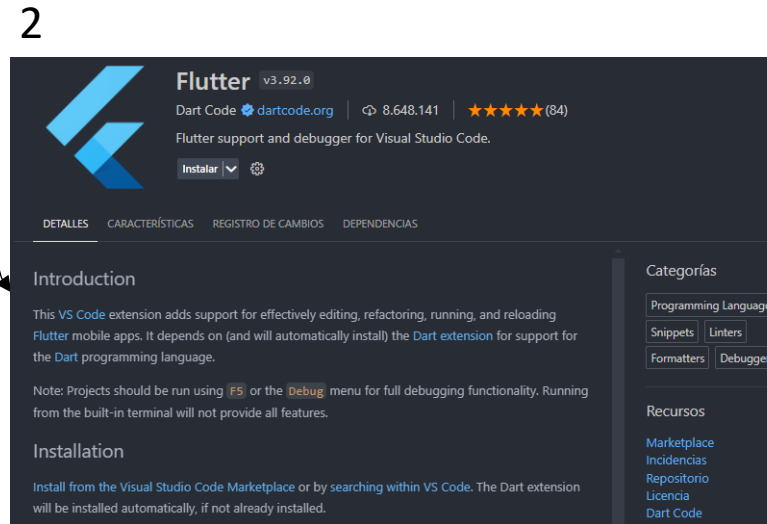
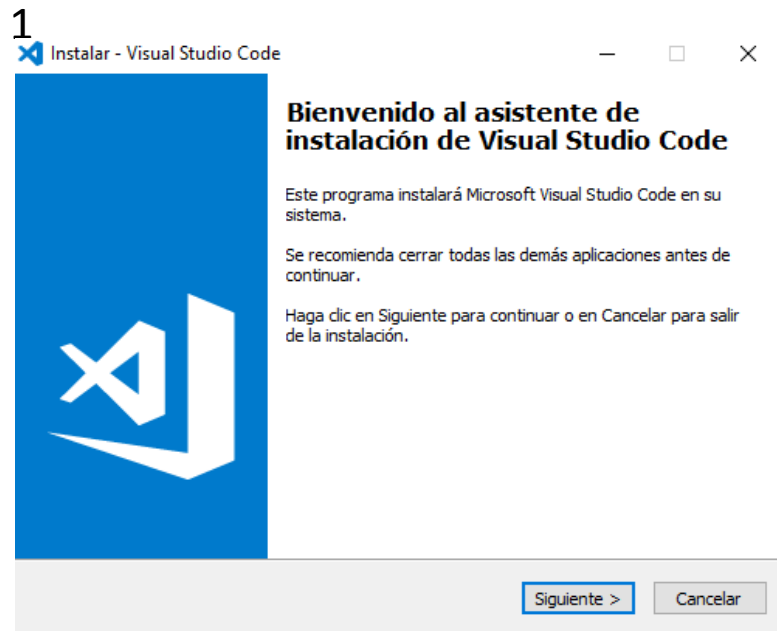
3





Visual Studio Code

Para configurar Visual Studio Code, primero descargue e instale VS Code desde el [sitio oficial](#). Instale la extensión de Flutter en VS Code desde la pestaña de Extensiones. Cree un nuevo proyecto Flutter y configure un emulador en Android Studio o conecte un dispositivo físico. Finalmente, ejecute la aplicación Flutter presionando F5 o seleccionando "Start Debugging" en la pestaña "Run".



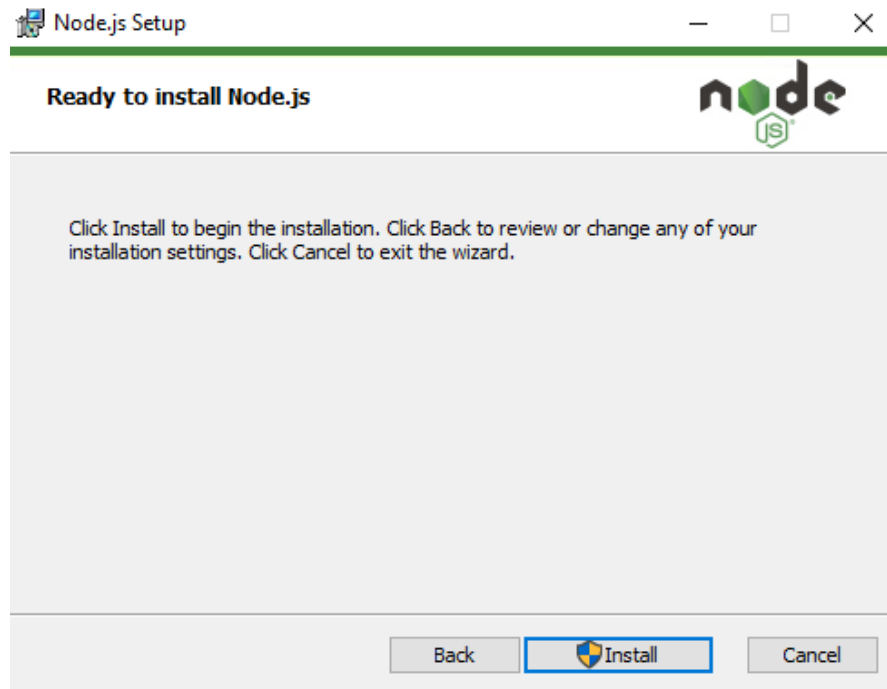


Node js y NPM

BCT

Para instalar Node.js y npm, visita el [sitio oficial](#) de Node js . Al completar la instalación, tanto Node.js como npm se instalarán automáticamente. Puedes verificar la instalación abriendo una terminal y ejecutando los comandos `node -v` y `npm -v`, los cuales deben mostrar las versiones instaladas de Node.js y npm respectivamente.

1



2

```
PS C:\Users\HECTOR\Desktop> node -v  
v20.15.1  
PS C:\Users\HECTOR\Desktop> npm -v  
9.5.1
```

Verificación de versiones

Para ver las versiones instaladas de las diversas herramientas desde la terminal, se puede usar los siguientes comandos. Para Flutter “flutter --version”. Para Visual Studio Code “code --version”. Para npm “npm -v”, y para Node.js “node -v”

. Estos comandos te proporcionarán la versión de cada herramienta instalada en tu sistema.

```
PS C:\Users\HECTOR\Desktop> flutter --version
Flutter 3.22.2 • channel stable • https://github.com/flutter/flutter.git
Framework • revision 761747bfc5 (5 weeks ago) • 2024-06-05 22:15:13 +0200
Engine • revision edd8546116
Tools • Dart 3.4.3 • DevTools 2.34.3
PS C:\Users\HECTOR\Desktop> code --version
1.91.0
ea1445cc7016315d0f5728f8e8b12a45dc0a7286
x64
PS C:\Users\HECTOR\Desktop> npm -v
9.5.1
PS C:\Users\HECTOR\Desktop> node -v
v20.15.1
```



Creación de un nuevo proyecto

Proyecto en Flutter

Para crear un nuevo proyecto en Flutter, abra una terminal y ejecute el comando flutter create my_app, donde "my_app" es el nombre del proyecto. Este comando generará una estructura de carpetas y archivos básica necesaria para empezar a desarrollar en Flutter.

La estructura incluye:

- Las carpetas android y ios que contienen configuraciones específicas para cada plataforma.
- la carpeta lib, que contiene el archivo principal main.dart donde se escribe el código de la aplicación.
- La carpeta test para pruebas unitarias.
- El archivo pubspec.yaml donde se gestionan las dependencias del proyecto.

```
PS C:\Users\HECTOR\Desktop\h> flutter create my_app
Creating project my_app...
Resolving dependencies in `my_app`... (1.7s)
Downloading packages...
Got dependencies in `my_app`.
Wrote 129 files.

All done!
You can find general documentation for Flutter at: https://docs.flutter.dev/
Detailed API documentation is available at: https://api.flutter.dev/
If you prefer video documentation, consider: https://www.youtube.com/c/flutterdev

In order to run your application, type:

$ cd my_app
$ flutter run

Your application code is in my_app\lib\main.dart.
```

```
my_app
├── android
├── ios
├── lib
│   └── main.dart
├── test
└── pubspec.yaml
```





FLUTTER: Como Crear una APP DESDE CERO

<https://www.youtube.com/watch?v=-pWSQYpkkjk&t=5s>

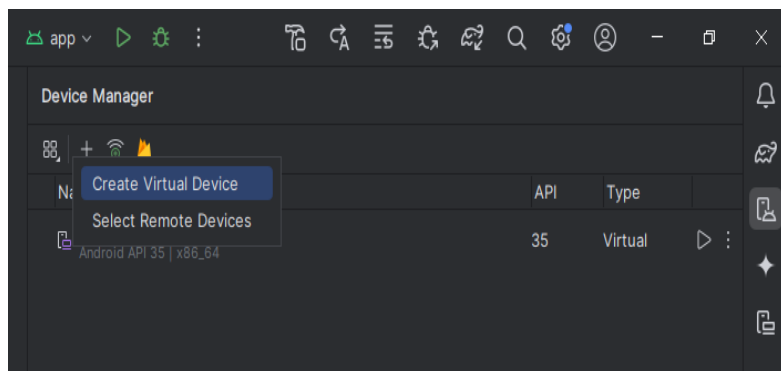


Levantar emulador en Android Studio

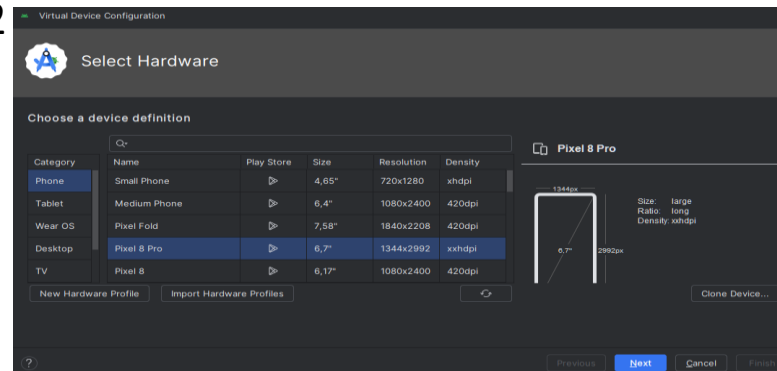
BCT

Para levantar un emulador en Android Studio, primero se debe abrir la aplicación y acceder al Device Manager desde el icono en la barra de herramientas. Crea un emulador nuevo haciendo clic en Create Virtual Device, selecciona un dispositivo, elige la imagen del sistema Android. Luego, en el Device Manager, haz clic en el icono de play junto al dispositivo virtual que deseas iniciar. Espera unos momentos mientras el emulador arranca y verás una ventana que emula el dispositivo Android seleccionado.

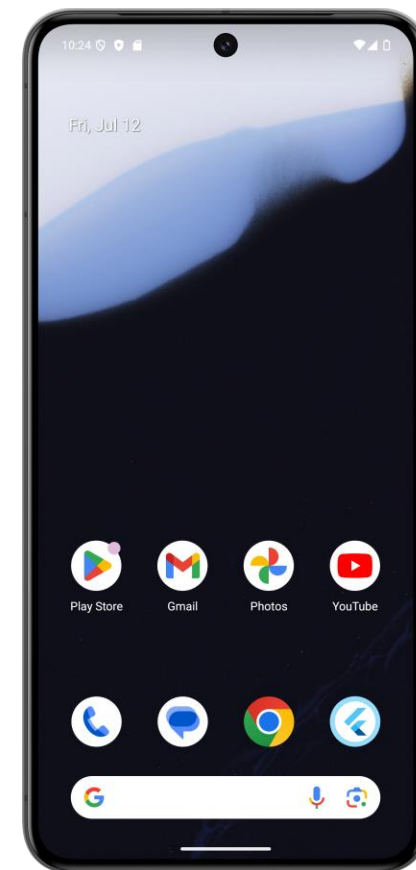
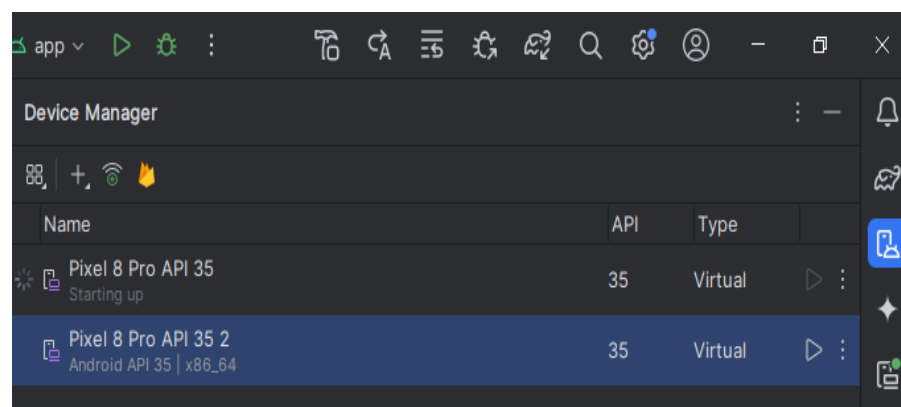
1



2

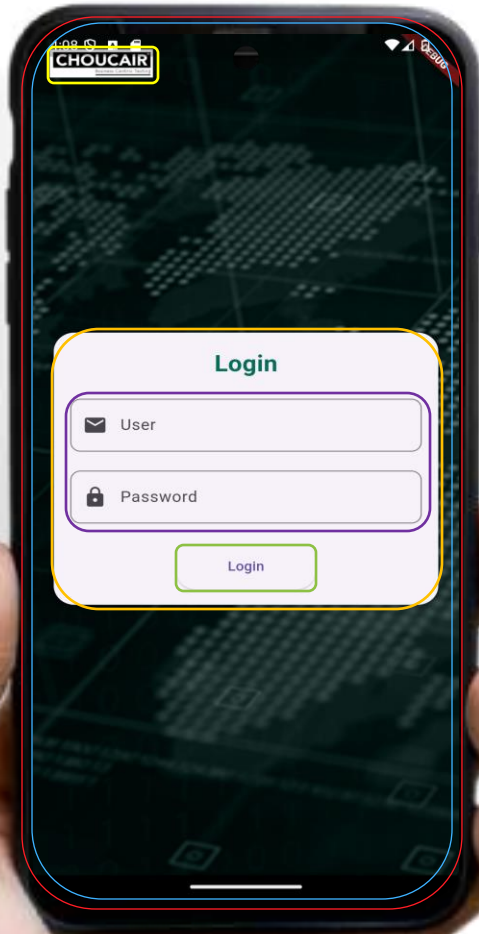


3



Estructura básica de login-Choucair

CONFIANZA



MaterialApp: Envuelve toda la aplicación, configurando el título, el tema y la pantalla principal.

Scaffold: Proporciona una estructura básica para la pantalla, incluyendo la AppBar, el Drawer, y el cuerpo de la pantalla.

Container: Utilizado para establecer la imagen del logo y del fondo de pantalla y aplicar el padding alrededor del contenido.

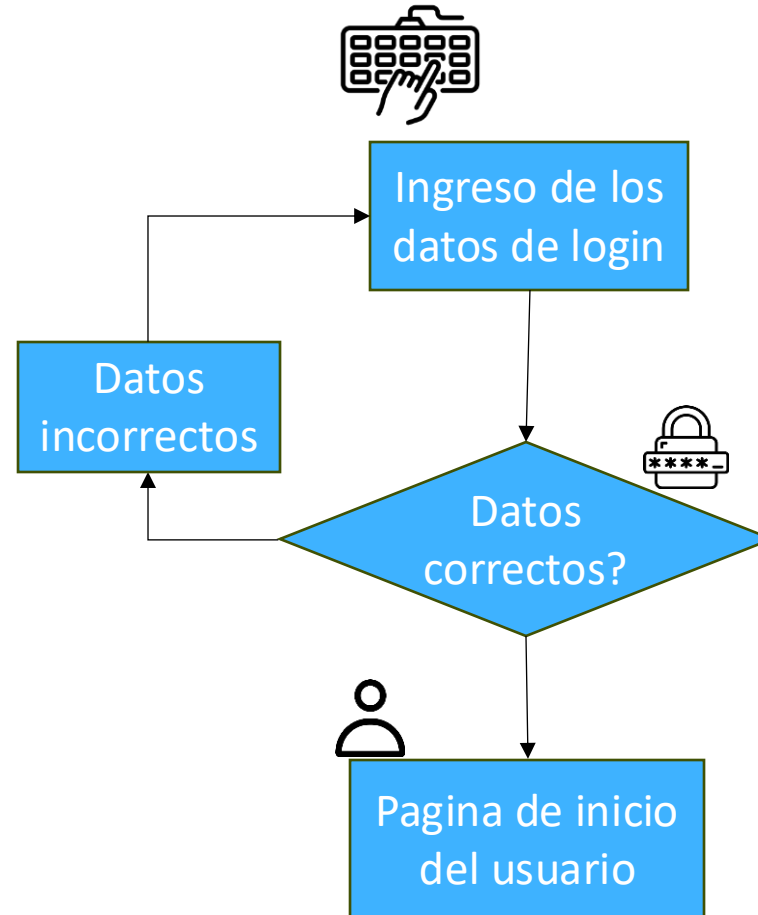
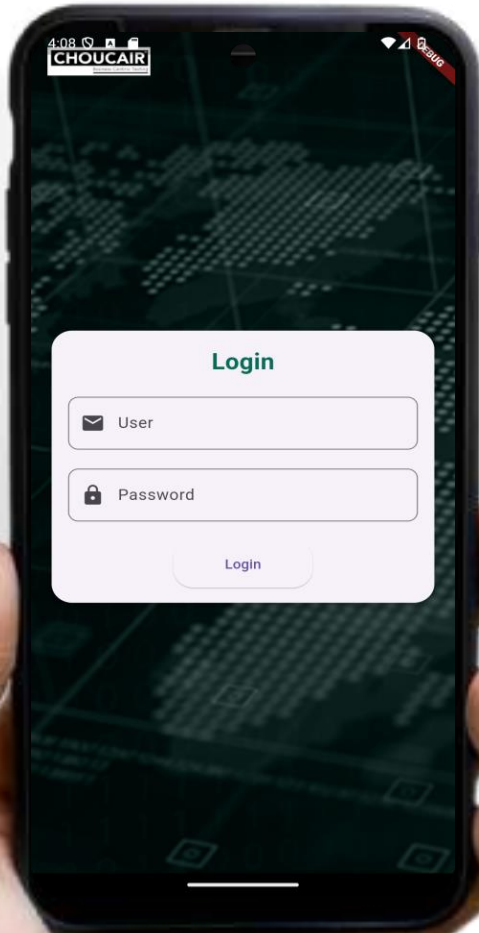
Card: Proporciona un contenedor rectangular con esquinas redondeadas y sombra, utilizado para contener el formulario de inicio de sesión.

Form: Un contenedor para los widgets de entrada de usuario que proporciona validación de formulario.

ElevatedButton: Este es el widget de botón que se utiliza para crear un botón con una sombra que se eleva cuando se presiona.

Diagrama de flujo de la app

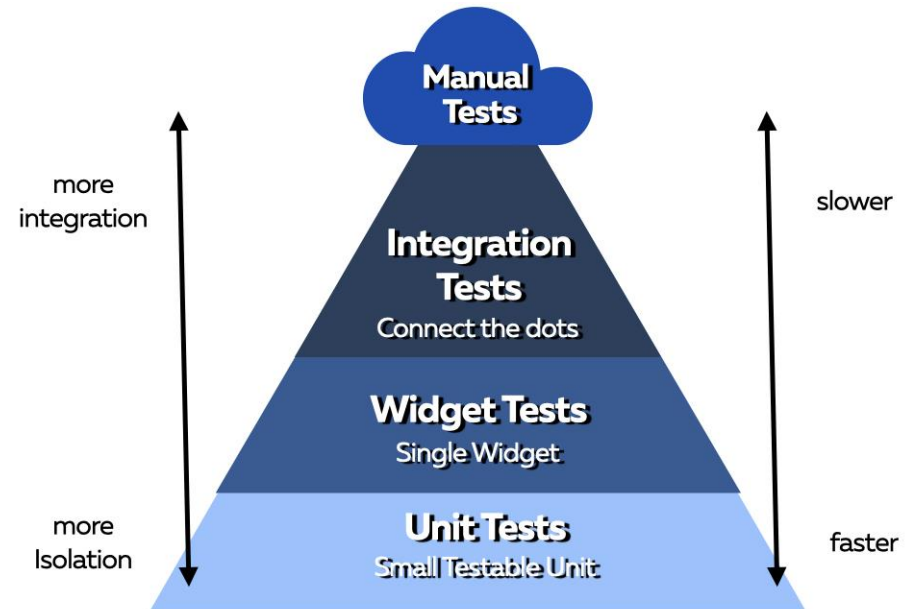
CONFIANZA



Tipos de pruebas en Flutter



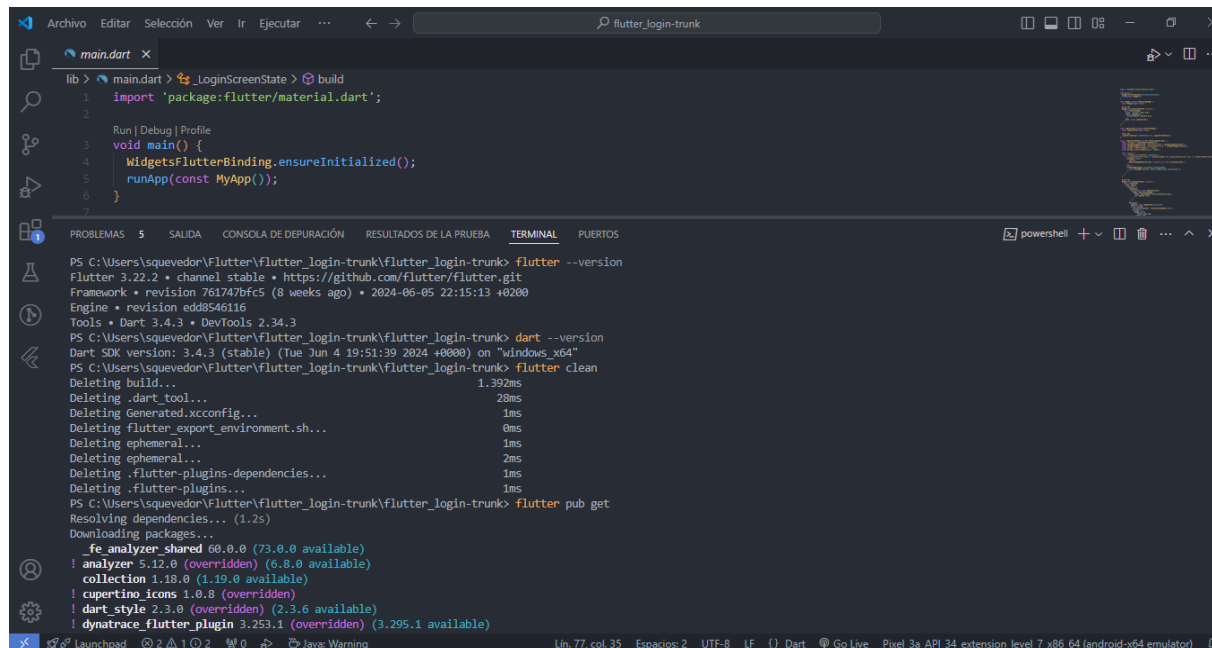
1. Unit Tests: Prueban funciones, métodos y clases individuales.
2. Widget Tests: Prueban la interfaz de usuario, aseguran que los widgets se comporten como se espera.
3. Integration Tests: Prueban una aplicación completa o una gran parte de ella, asegurando que todo funcione bien en conjunto.



Gestión de dependencias

BCT

1. Asegurar tener correctamente la configuración del ambiente de desarrollo, Es importante tener presente las versiones que tenemos instaladas, ya que muchos errores que se nos pueden presentar en el proyecto son por incompatibilidad entre versiones del flutter y el dart sdk.
2. Lanzar el comando "**flutter clean**" para limpiar el proyecto eliminando archivos generados y carpetas de construcción (build). Es especialmente útil cuando se encuentran problemas extraños durante el desarrollo, como errores de compilación inesperados o comportamientos anómalos en la aplicación.
3. Lanzar el comando "**flutter pub get**" para gestionar las dependencias del proyecto. Específicamente, este comando descarga e instala los paquetes listados en el archivo **pubspec.yaml**.



The screenshot shows an IDE window titled 'flutter_login-trunk'. The editor displays a Dart file named 'main.dart' with the following code:

```
lib > main.dart > _LoginScreenState > build
1 import 'package:flutter/material.dart';
2
3 Run | Debug | Profile
4 void main() {
5   WidgetsFlutterBinding.ensureInitialized();
6   runApp(const MyApp());
7 }
```

The terminal output shows the execution of the following commands and their results:

```
PS C:\Users\squevedor\Flutter\flutter_login-trunk\flutter_login-trunk> flutter --version
Flutter 3.22.2 • channel stable • https://github.com/flutter/flutter.git
Framework • revision 761747bfc5 (8 weeks ago) • 2024-06-05 22:15:13 +0200
Engine • revision edd8546116
Tools • Dart 3.4.3 • DevTools 2.34.3

PS C:\Users\squevedor\Flutter\flutter_login-trunk\flutter_login-trunk> dart --version
Dart SDK version: 3.4.3 (stable) (Tue Jun 4 19:51:39 2024 +0000) on "windows_x64"

PS C:\Users\squevedor\Flutter\flutter_login-trunk\flutter_login-trunk> flutter clean
Deleting build... 1.392ms
Deleting .dart_tool... 28ms
Deleting Generated.xcconfig... 1ms
Deleting flutter_export_environment.sh... 0ms
Deleting ephemeral... 1ms
Deleting ephemeral... 2ms
Deleting .flutter-plugins-dependencies... 1ms
Deleting .flutter-plugins... 1ms

PS C:\Users\squevedor\Flutter\flutter_login-trunk\flutter_login-trunk> flutter pub get
Resolving dependencies... (1.2s)
Downloading packages...
! _fe_analyzer_shared 60.0.0 (73.0.0 available)
! analyzer 5.12.0 (overridden) (6.8.0 available)
! collection 1.18.0 (1.19.0 available)
! cupertino_icons 1.0.8 (overridden)
! dart_style 2.3.0 (overridden) (2.3.6 available)
! dynatrace_flutter_plugin 3.253.1 (overridden) (3.295.1 available)
```

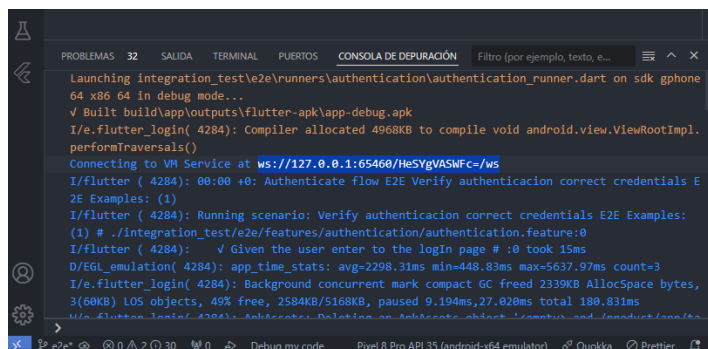


Flutter Inspector en chrome

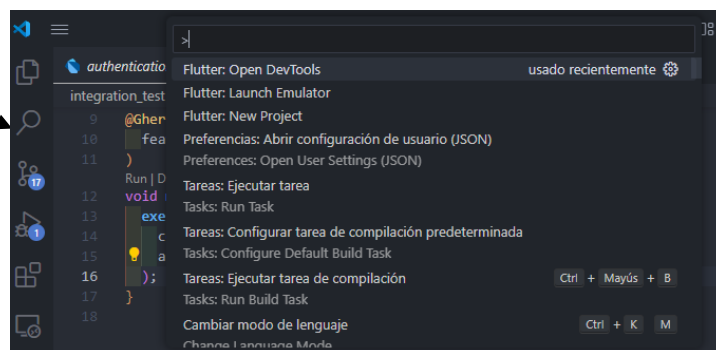
BCT

Para abrir el Flutter Inspector en Visual Studio Code y poder inspeccionar los elementos de tu aplicación Flutter, primero asegúrate de tener tu proyecto abierto en Visual Studio Code. Luego, inicia la aplicación en modo debug. Una vez que la aplicación esté en ejecución, abre la vista de Flutter seleccionando el icono correspondiente en la barra lateral izquierda. En la pestaña de Flutter, haz clic en el botón "Open Flutter DevTools", lo que abrirá una nueva ventana en el navegador con Flutter DevTools. Dentro de Flutter DevTools, selecciona la pestaña "Inspector" y activa el modo de inspección haciendo clic en el icono de la lupa. Ahora puedes hacer clic en cualquier widget de tu aplicación para ver su estructura y propiedades en el Flutter Inspector.

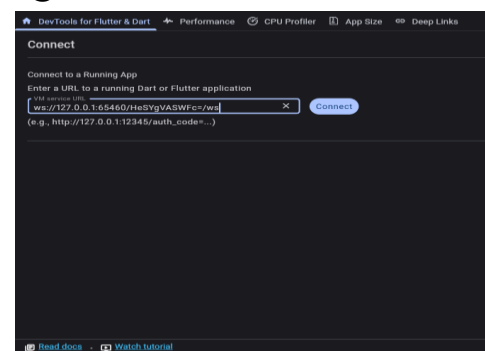
1



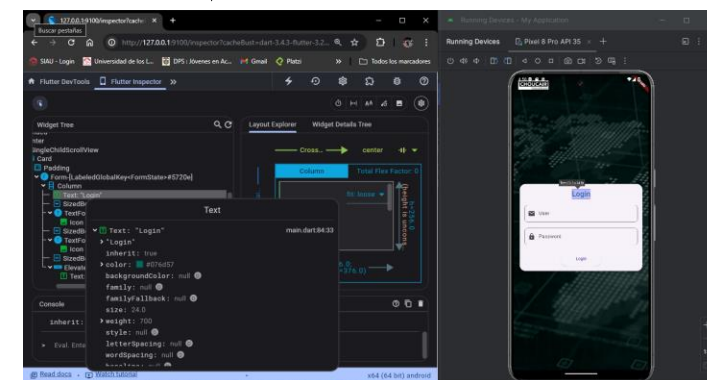
2



3



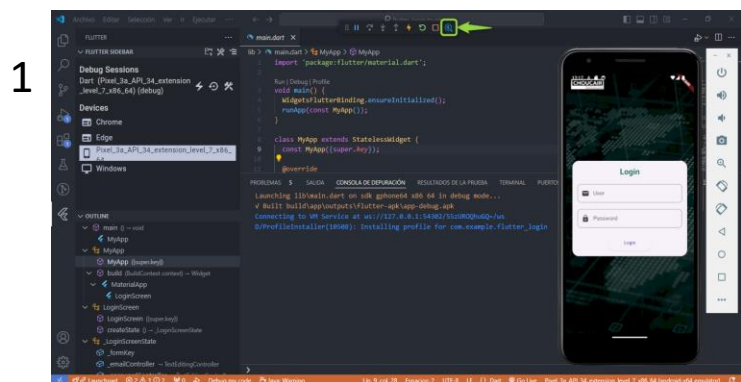
4



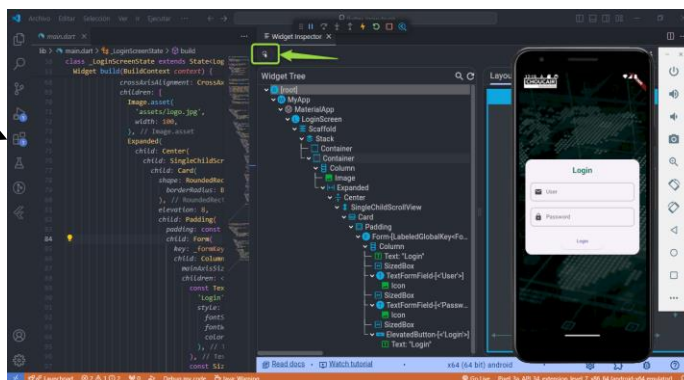


Flutter Inspector en VSC

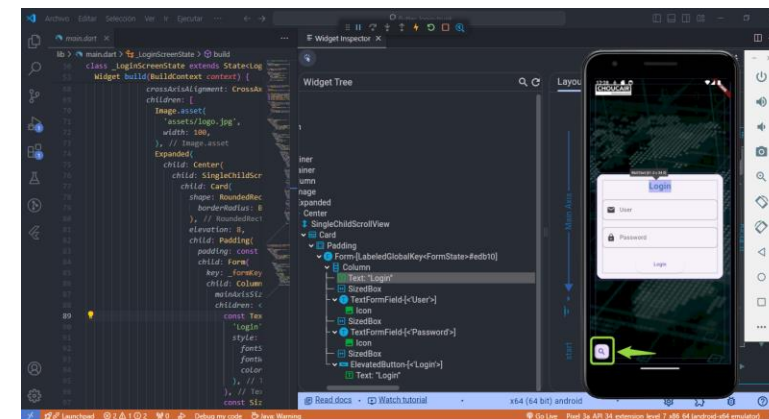
Para abrir el Flutter Inspector en Visual Studio Code y poder inspeccionar los elementos de tu aplicación Flutter, primero asegúrate de tener tu proyecto abierto en Visual Studio Code. Luego, inicia la aplicación en modo debug. Una vez que la aplicación esté en ejecución sigue los siguientes pasos para inspeccionar cada uno de los widgets que tiene la app.



2



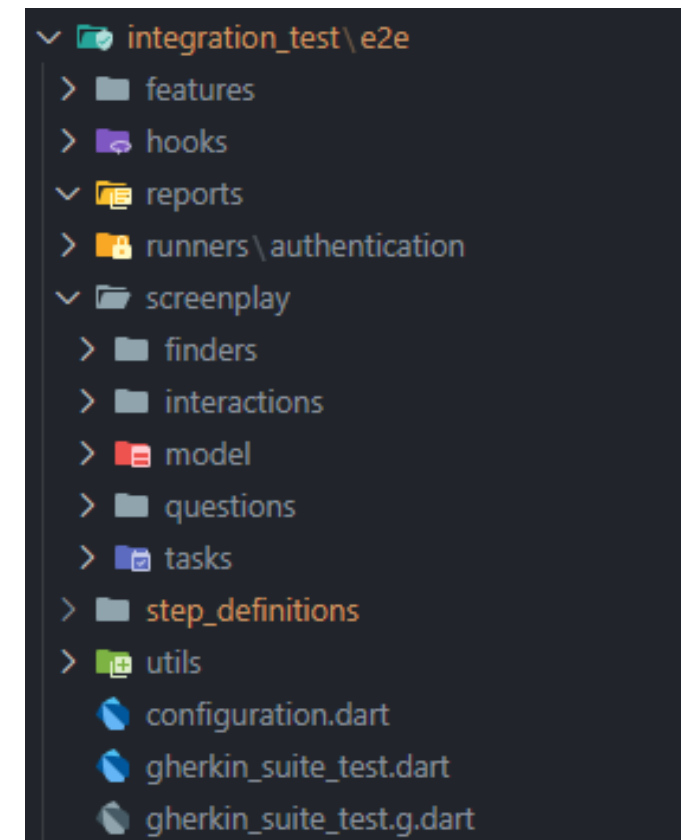
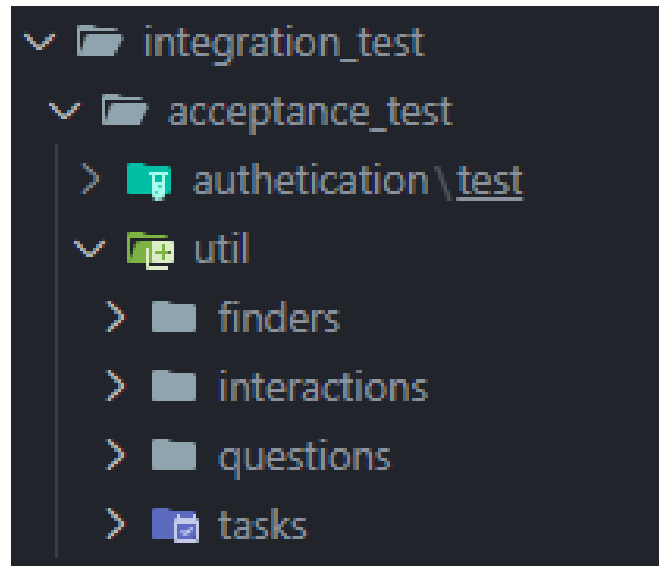
3



Métodos más usados en test

- **pumpWidget**: Construye y renderiza un widget en el árbol de widgets.
`await tester.pumpWidget(MyApp());`
- **find**: Localiza widgets en el árbol de widgets.
`find.text('Hello');`
- **tap**: Simula un gesto de toque en un widget.
`await tester.tap(find.byIcon(Icons.add));`
- **pump**: Avanza el reloj de la aplicación y reconstruye el widget.
`await tester.pump();`
- **pumpAndSettle**: Avanza el reloj de la aplicación y espera a que todos los cambios de animación y cuadros se completen.
`await tester.pumpAndSettle();`
- **expect**: Verifica que un valor cumpla con una condición.
`expect(find.text('Hello'), findsOneWidget);`
- **Drag**: Simula un gesto de arrastre en un widget.
`await tester.drag(find.byType(ListView), Offset(0, -200));`
- **longPress**: Simula un gesto de presión prolongada en un widget.
`await tester.longPress(find.byType(ListTile));`
- **ensureVisible**: Desplaza un widget en un contenedor desplazable para que sea visible.
`await tester.ensureVisible(find.byKey(Key('myKey')));`

Arquetipo para Acceptance Test y E2E Test





Ejecucion de acceptance test

Para ejecutar los test se ejecuta el siguiente comando en la terminal:

flutter test <test_path>

test_path: Esta ruta debe apuntar al archivo main que debe llamar a todos los otros test, por ejemplo: `integration_test\acceptance_test\authetication\test\main_test.dart`.

En el mundo de widgets móvil con las pruebas de aceptación no se busca una corroboración minuciosa de los componentes sino se centran principalmente en verificar que la aplicación cumple con los requisitos y expectativas del negocio y de los usuarios finales.

```
integration_test > acceptance_test > authentication > test > main_test.dart > main
1 import 'login_correct_test.dart' as login_correct_test;
2
3 Run | Debug
4 void main() {
5   login_correct_test.main();
6 }
```

```
PS C:\Users\squevedor\Flutter\flutter_login-trunk\flutter_login-trunk> flutter test integration_test\acceptance_test\authetication\test\main_test.dart
00:00 40: ...or/Flutter/flutter_login-trunk/flutter_login-trunk/integration_test/acceptance_test/authetication/test/main_test.dart R
00:19 40: ...or/Flutter/flutter_login-trunk/flutter_login-trunk/integration_test/acceptance_test/authetication/test/main_test.d 18,9s
✓ Built build\app\outputs\flutter-apk\app-debug.apk
00:20 40: ...or/Flutter/flutter_login-trunk/flutter_login-trunk/integration_test/acceptance_test/authetication/test/main_test.dart I
00:21 40: ...or/Flutter/flutter_login-trunk/flutter_login-trunk/integration_test/acceptance_test/authetication/test/main_test.d 1.387ms
00:37 41: Verify button enabled to user start with zero
```

Ejercicio practico

Implementa un nuevo escenario de acceptance test teniendo en cuenta el arquetipo que se muestra anteriormente y realiza su ejecución en local.



Ejecucion de E2E

Para realizar la ejecución de los test en local es necesario tener en cuenta los siguientes comandos y el momento de su uso en la terminal:

flutter pub run build_runner clean: Es el primer comando a ejecutar, se realiza cuando se abre el proyecto por primera vez o cuando se realizan cambios en los features esto con el fin de limpiar la cache y asegurar que se actualice correctamente

flutter pub run build_runner build --delete-conflicting-outputs: Es el comando que se ejecuta siempre que se hayan realizados cambios en la carpeta feature (Agregar nuevos escenarios, modificar escenarios o agregar nuevos features).

flutter drive --driver=test_driver/integration_test_driver.dart --target=integration_test/e2e/gherkin_suite_test.dart: Con este comando se ejecuta de forma local las pruebas, es necesario tener disponible un dispositivo emulado o fisico para la compilación de las pruebas



Ejecucion de E2E

En el mundo de widgets móvil con las pruebas E2E se busca verificar que todas las partes del sistema funcionan bien juntas en un entorno de producción simulado. Evalúan el flujo completo desde el inicio hasta el fin, incluyendo la integración de todos los componentes del sistema.

The screenshot shows an IDE interface with an Android emulator on the left displaying "Test starting...". The main editor shows a Dart file named `gherkin_suite_test.g.dart` with the following content:

```
1 // GENERATED CODE - DO NOT MODIFY BY HAND
2
3 part of 'gherkin_suite_test.dart';
4
5 // *****
6 // GherkinSuiteTestGenerator
7 // *****
8
9 class _CustomGherkinIntegrationTestRunner extends GherkinIntegrationTestRunner {
10   CustomGherkinIntegrationTestRunner() {}
11 }
```

The bottom panel shows the terminal output:

```
vector_graphics 1.1.10+1 (1.1.11+1 available)
vector_graphics_codec 1.1.10+1 (1.1.11+1 available)
vector_graphics_compiler 1.1.10+1 (1.1.11+1 available)
vm_service 14.2.1 (14.2.4 available)
web 0.5.1 (1.0.0 available)
web_socket_channel 2.4.5 (3.0.1 available)
! webview_flutter 4.4.2 (overridden) (4.8.0 available)
! webview_flutter_wkwebview 3.5.0 (overridden) (3.14.0 available)
! xml 6.5.0 (overridden)
Got dependencies!
35 packages have newer versions incompatible with dependency constraints.
Try 'flutter pub outdated' for more information.
Running Gradle task 'assembleDebug'... 5,8s
✓ Built build/app/outputs/flutter-apk/app-debug.apk
Installing build/app/outputs/flutter-apk/app-debug.apk... 1,154ms
E/OpenGLRenderer(12916): Unable to match the desired swap behavior.
VMServiceFlutterDriver: Connecting to Flutter application at http://127.0.0.1:49748/Gs6xN6QaCXo=/
VMServiceFlutterDriver: Isolate found with number: 8091849762072703
VMServiceFlutterDriver: Isolate is paused at start.
VMServiceFlutterDriver: Attempting to resume isolate
I/flutter (12916): 00:00 +0: Authenticate flow E2E: Verify authentication correct credentials E2E Examples: (1)
VMServiceFlutterDriver: Connected to Flutter application.
```


Ejercicio practico

Implementa un nuevo escenario de E2E teniendo en cuenta el arquetipo que se muestra anteriormente y realiza su ejecución en local.



¡Muchas Gracias!