7

ARCHIVOS DE PROYECTO FLUTTER

Objetivos

1. Comprender la estructura de un proyecto Flutter.
2. Conocer el archivo lib.
3. Conocer el archivo pubspec.yaml.
4. Conocer los archivos para iOS.
5. Conocer los archivos para Android.
6. Aprender a gestionar paquetes y dependencias.
7. Utilizar recursos como imágenes e iconos en una aplicación.

ⓘ Tip

Cuando en una carpeta se ve un > significa que contiene más archivos o subcarpetas. Tómate tu tiempo para que abras cada carpeta solo para inspección general.

Introducción

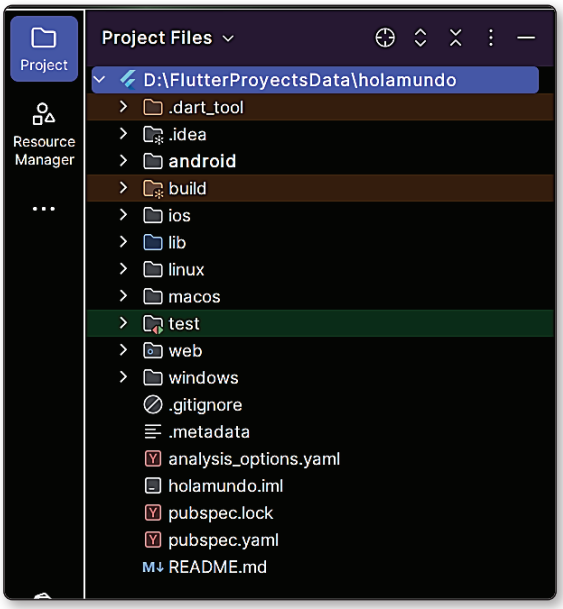
En este capítulo, exploraremos la estructura de un proyecto Flutter y aprenderemos a gestionar paquetes, librerías y recursos como imágenes e iconos.

Tomaremos siempre como ejemplo el Demo que Flutter genera cada vez que se crea un proyecto nuevo.

Al crear un proyecto Flutter genera varios archivos los cuales se encuentran en la carpeta Project files. En la parte superior izquierda del IDE.

En la siguiente imagen puedes ver las carpetas y archivos para el proyecto DEMO predeterminado de Flutter.

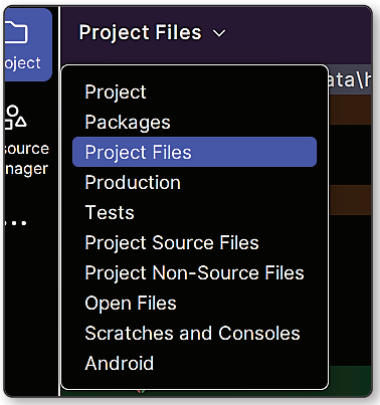
Si te parecen muchos, te sorprenderás más por la cantidad de subcarpetas, archivos, imágenes y recursos que se encuentran ahora ocultos bajo cada uno de ellos.



Estructura de un proyecto Flutter

Al crear un nuevo proyecto Flutter, se genera una estructura de carpetas y archivos que contienen el código fuente, las dependencias, los recursos y la configuración de la aplicación.

Para ver los archivos del proyecto, búscalos en la parte superior izquierda del IDE, selecciona el nombre del proyecto, da clic y luego haz lo mismo en Proyect Files para que se abra el directorio de archivos.



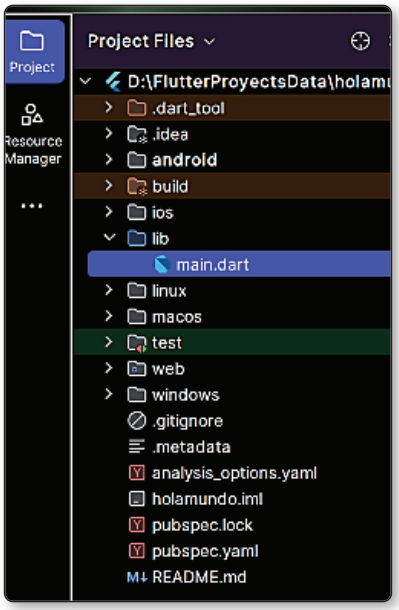
Carpeta lib

Tip:

La carpeta lib contiene el archivo main.dart que aloja el código que se muestra en el editor cuando se crea un proyecto.

- **lib**: contiene el código fuente principal de la aplicación. El archivo main.dart es el punto de entrada de la aplicación y alberga el código fuente de la pantalla de inicio.

  Observa en la imagen la carpeta lib conteniendo el archivo main.dart.



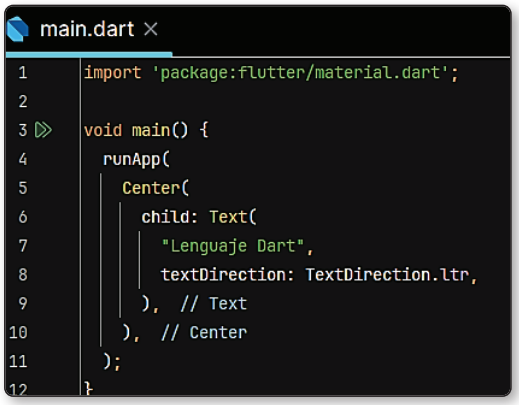
En la carpeta lib está el archivo main.dart que contiene el código fuente de la pantalla principal o de inicio de la app no solo esto, main.dart puede con todo el código de la interfaz, no obstante es recomendable no saturarlo de código.

main.dart

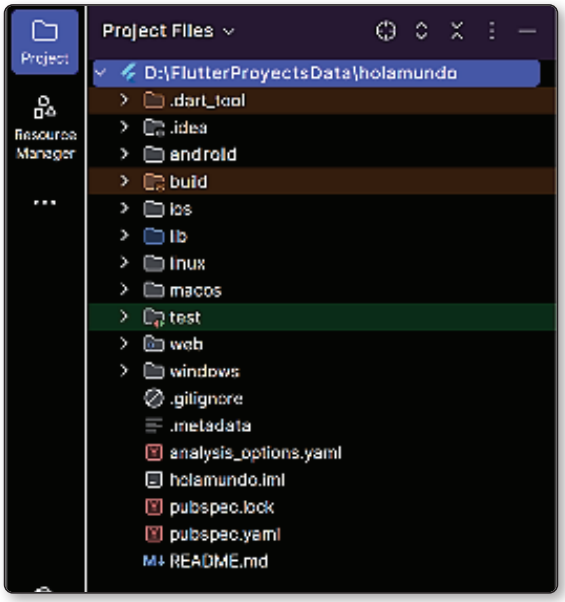
Es un archivo principal que debes conocer su ubicación porque allí se escribe el código fuente de la pantalla de inicio de toda app en Flutter.

Se encuentra en la carpeta **lib** de los archivos del proyecto.

En la siguiente imagen muestro el contenido de ejemplo del archivo main.dart.



- **test:** contiene las pruebas unitarias para la aplicación.



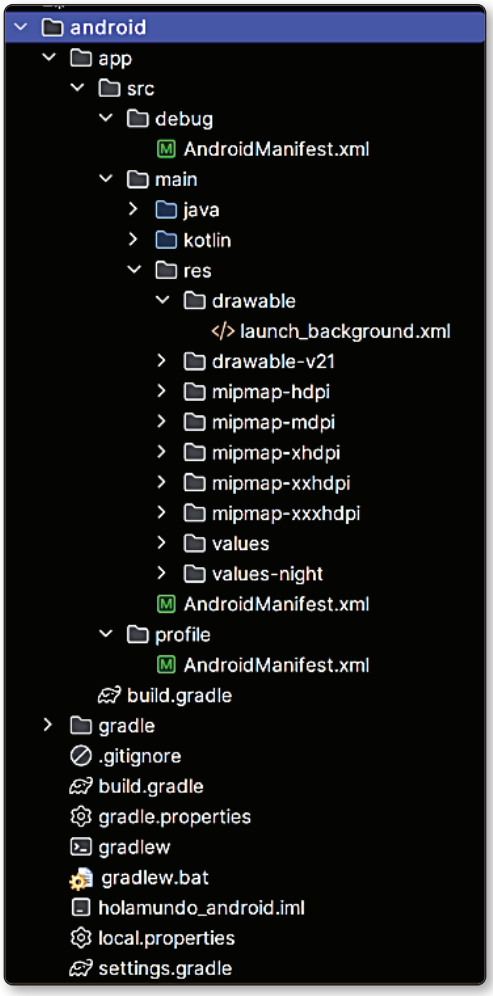
Las pruebas unitarias sirven para probar unidades individuales de código.

Se recomienda borrar esta carpeta antes de hacer producción para publicación de app o en tiempo de depuración, si se bloquea que el proyecto se muestre como se espera.

- **Android:** contiene el código y los recursos específicos para la plataforma Android.

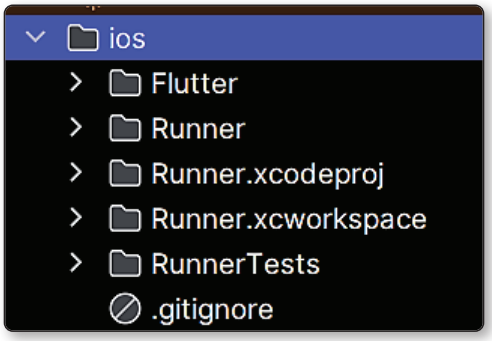
  Esta carpeta es muy importante porque contiene los principales archivos y recursos del proyecto como por ejemplo las imágenes de los iconos.

  Observa los archivos alojados en la carpeta **Android.**



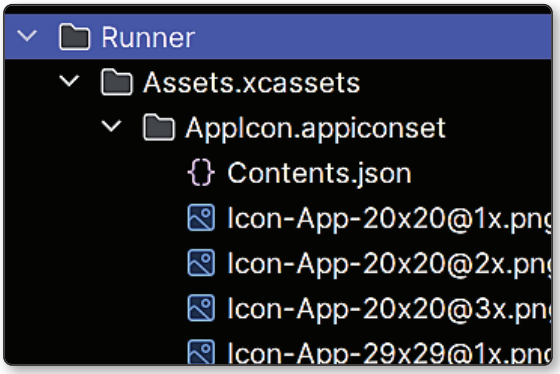
- **ios**: contiene el código y los recursos específicos para la plataforma iOS.

- Allí se encuentran las imágenes de los iconos de app para iPhone.



Despliega cada carpeta para que veas su contenido.

Por ejemplo si abres la carpeta Runner verás que contiene la subcarpeta Assets.xcasset que contiene otra subcarpeta Appicon.appiconset en donde están alojadas las imágenes en diferente resolución para el icono de lanzamiento de la app.



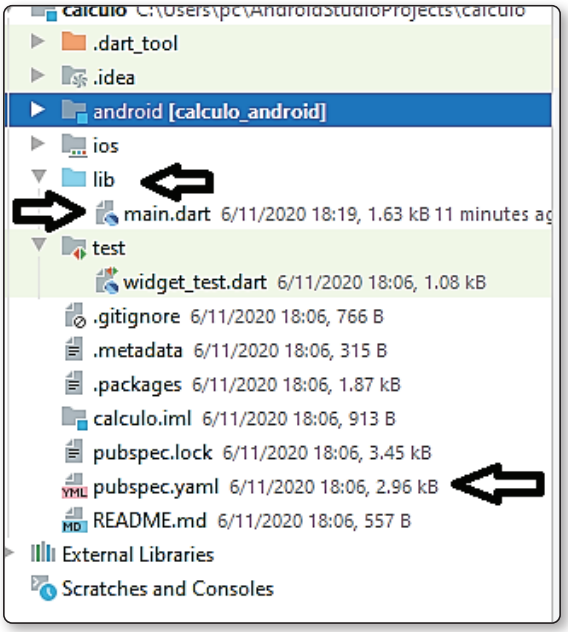
Pubspec.yaml

Es el archivo de configuración del proyecto. Aquí se definen las dependencias, los recursos y la metadata de la aplicación.

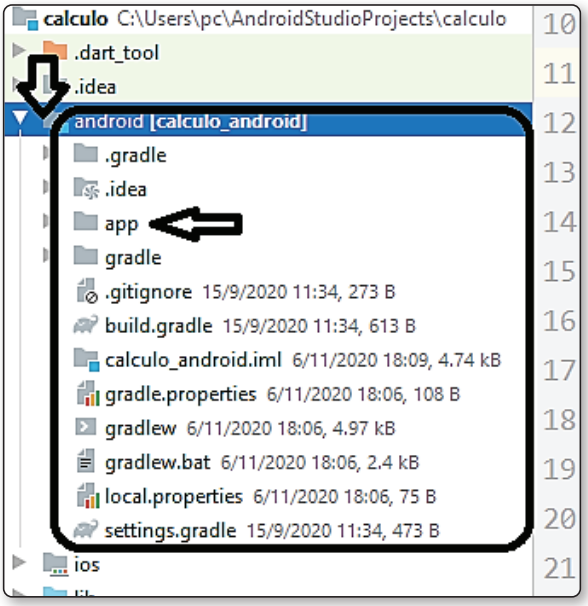
- Por ejemplo si vas a usar imágenes en el proyecto, debes guardarlas en una carpeta creada en el directorio del proyecto. Además debes hacer la referencia a esas imágenes en el archivo pubspec.yaml.
- Si vas a usar un paquete externo debes instalarlo en el archivo **pubspec.yaml.**

```
environment:
  sdk: ^3.6.1

# Dependencies specify other packages that
# To automatically upgrade your package dep
# consider running `flutter pub upgrade --
# dependencies can be manually updated by
# the latest version available on pub.dev.
# versions available, run `flutter pub out
dependencies:
  flutter:
    sdk: flutter

  # The following adds the Cupertino Icons
  # Use with the CupertinoIcons class for
  cupertino_icons: ^1.0.8

dev_dependencies:
  flutter_test:
    sdk: flutter
```

En la siguiente imagen puedes observar señalados por flechas el archivo main.dart y pubspect.yaml.

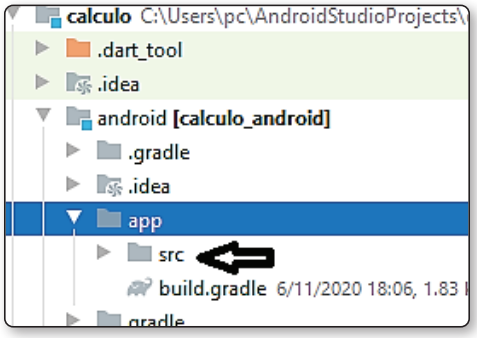


Ahora expande la carpeta Android que contiene los recursos para desarrollo de apps para Android.



La carpeta **app** se encuentra allí.

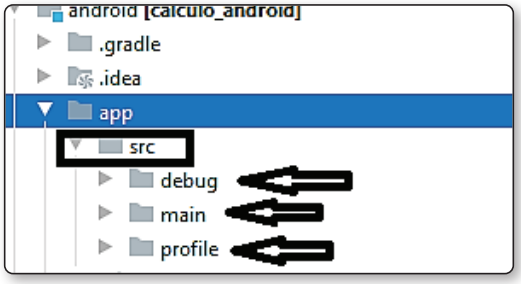
Abre la carpeta **app** y allí está la subcarpeta **src.**



Carpeta src

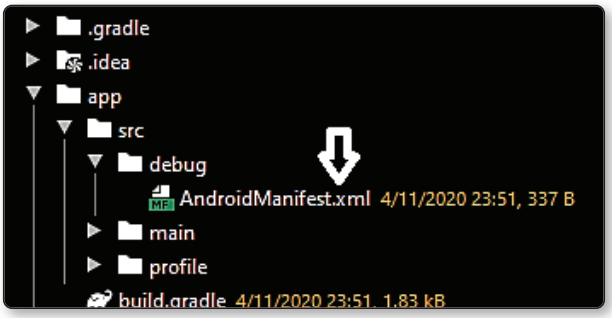
Abre src y allí veras su contenido.

Contiene las subcarpetas debug, main y profile.



Abre **debug** y allí encontraras el **AndroidManifest.xml.**

AndroidManifest.xml



Es donde se configuran los permisos que tienes que pedir al usuario antes que instale tu app.

Fíjate en el siguiente código ubicado en AndroidManifest.xml.

<uses-permission android:name=”android-permision.INTERNET”/>.

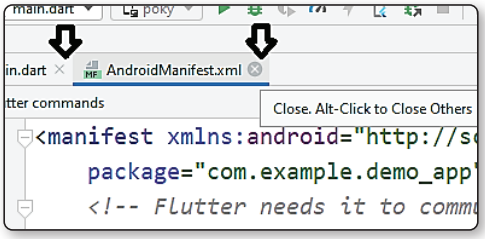
Se está pidiendo permiso al usuario para usar su internet en la app.

También aquí puedes cambiar el nombre a la app.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.calculo">
    <!-- Flutter needs it to communicate with the running application
         to allow setting breakpoints, to provide hot reload, etc.
    -->
    <uses-permission android:name="android.permission.INTERNET"/>
</manifest>
```

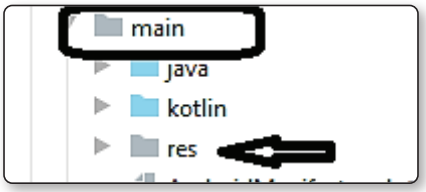
Ahora cierra el archivo AndroidManifest.xml.

Cualquier ventana que abras en el editor puedes cerrarla dando clic en x arriba del editor.



Carpeta main.

Abre la carpeta main y verás varias subcarpetas, enfócate en la subcarpeta res.



Abre res.

La carpeta **drawable** está allí, junto a varias carpetas con imágenes en varias resoluciones.

La carpeta res tiene los recursos de imágenes para el icono que mostrará la app y tienes que visitar esta carpeta res, si deseas cambiar el icono.

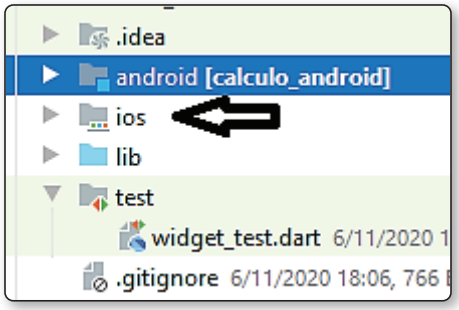
Si vas a agregar imágenes para el icono de tu proyecto, es aquí donde debes colocarlas.

Da clic en mipmap-hdpi te mostrará el archivo que contiene, es un archivo de imagen.

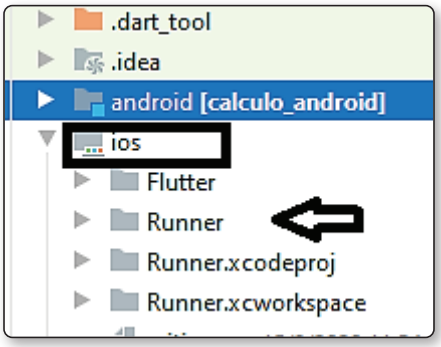
Ábrelo…

Es el icono lanzador. Es decir el que se muestra en la pantalla.

Ahora pasemos a la carpeta iOS.

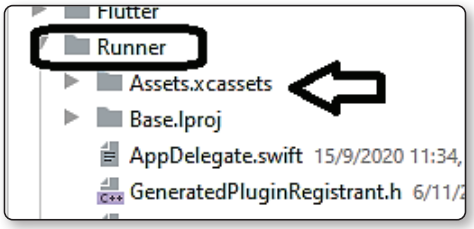


Ábrela.

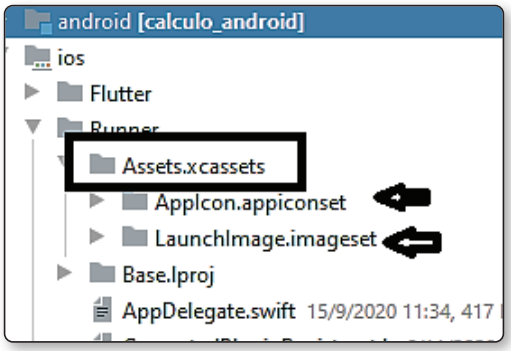


La carpeta iOS tiene varias subcarpetas.

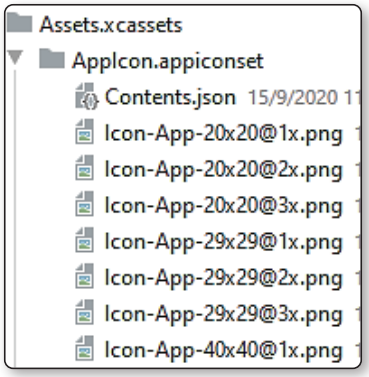
Runner es la subcarpeta en la cual colocaremos los archivos de imágenes para icon, que vienen contenidos en el paquete Assets.xcassets.



Si abres **Assets.xcassets** verás lo siguiente…



Son archivos de imágenes.



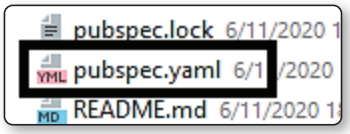
Abre uno…

Son imágenes predefinidas de Flutter para la creación del icono.

Antes de pasar a otro tema no olvides que en esta área está el archivo pubspec.yaml.

pubspec.yaml

Archivo en el cual se configuran las dependencias y archivos de imágenes assets y los paquetes procedentes de pubs.dev.



pubspec.yaml se visita con frecuencia para colocar allí los paquetes que descargues de internet, específicamente de pub.dev que es una página web que provee paquetes para flutter y otros entornos de desarrollo.

También debes visitar este archivo cuando utilices imágenes en tus proyectos, a menos que vengan de internet o de un archivo local en tu pc.

```
dependencies:
  flutter:
    sdk: flutter


  # The following adds
  # Use with the Cupert
  cupertino_icons: ^0.1

dev_dependencies:
  flutter_test:
    sdk: flutter

# For information on th
# following page: https

# The following section
flutter:
```

Gestionando paquetes y dependencias en pubspec.yaml

Los paquetes son colecciones de código reutilizable que te permiten añadir funcionalidades a tu aplicación sin tener que escribir todo el código desde cero.

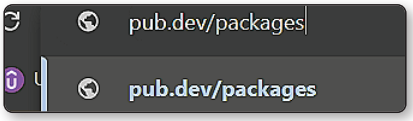
Para añadir un paquete a tu proyecto, debes seguir estos pasos:

1. Busca el paquete que necesitas en *pub.dev***.**
2. Añade la dependencia en el archivo pubspec.yaml.
3. Ejecuta flutter pub get en la terminal para descargar el paquete.
4. Importa el paquete en tu código Dart.

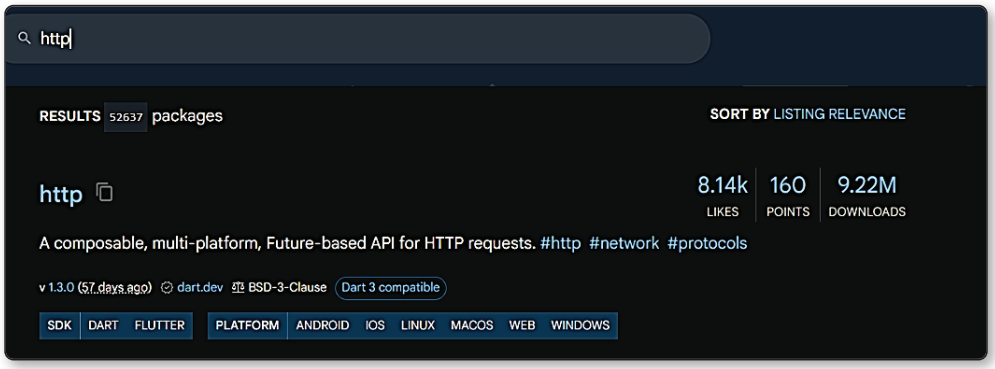
Ejemplo

Para añadir el paquete http que permite realizar peticiones HTTP, debes hacer lo siguiente:

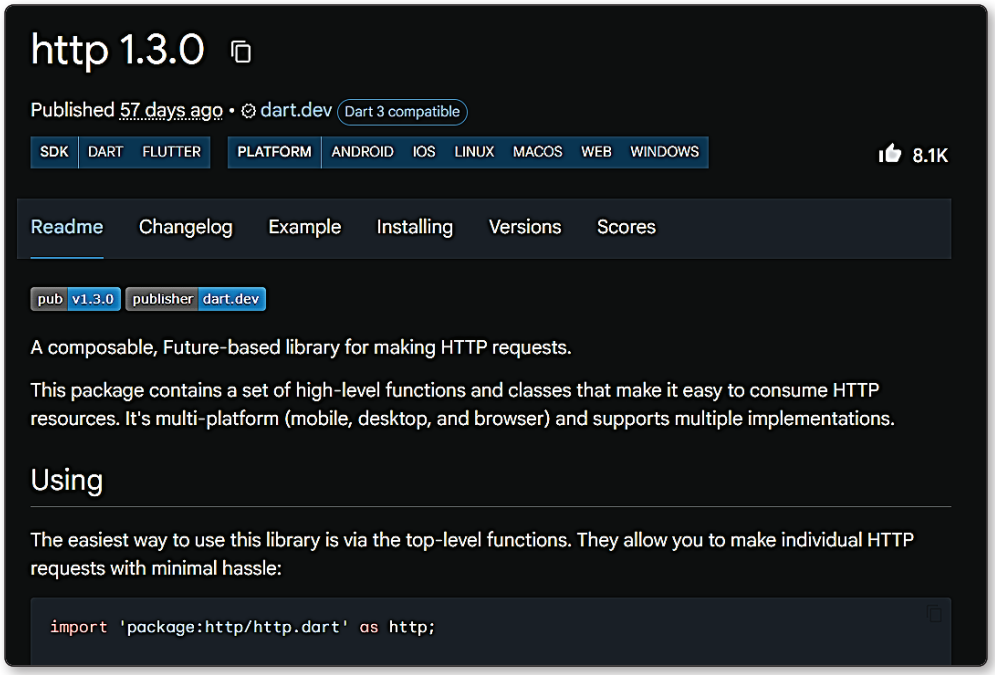
1. Busca en la página web pub.dev el paquete que necesitas.



2. Escribe el paquete que buscas en la casilla del buscador de pub.dev.

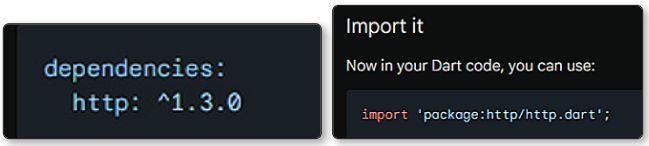


3. Te mostrará el paquete disponible.

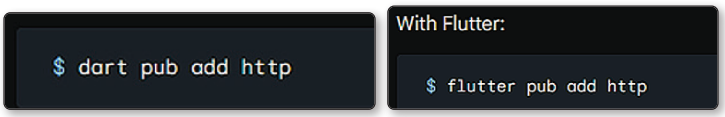


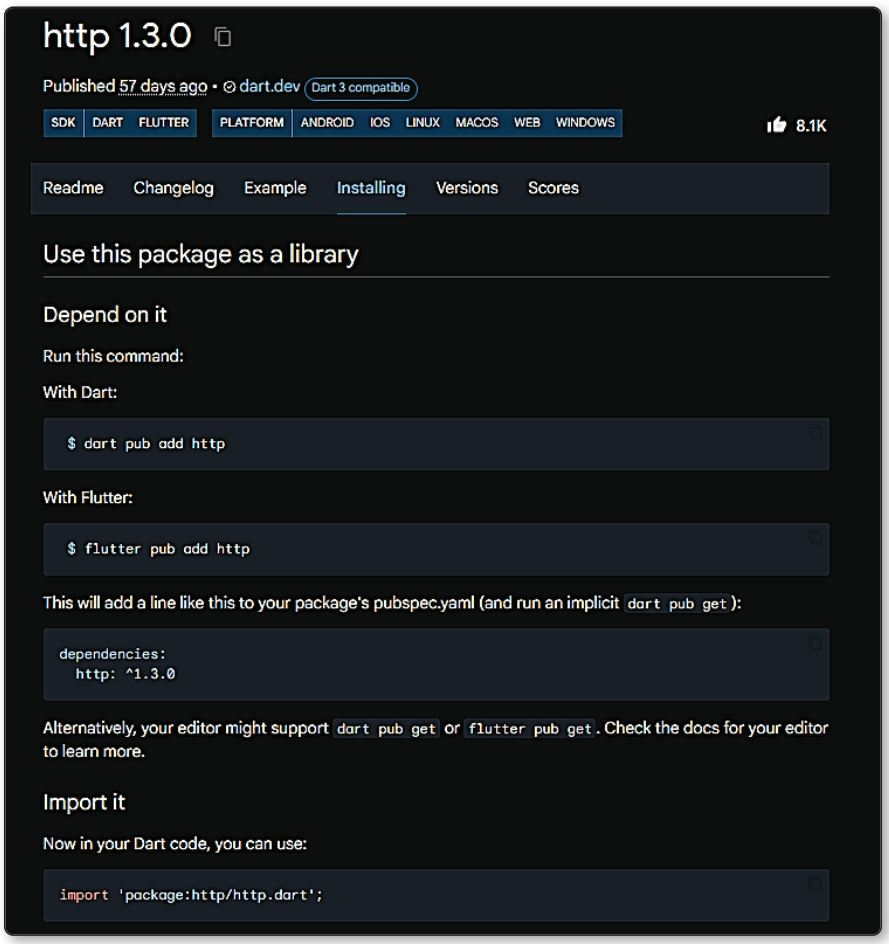
4. Clic en Instalación (Installing) y te llevará a la ventana donde te explica con detalle el paquete que instalaras en dependencias del pubspec.yaml.

También te da el código para hacer la importación del paquete en main.dart.



También puedes hacer la instalación del paquete usando comandos de Dart y Flutter.





Dependencias en pubspec.yaml

dependencies:

```
  http: ^1.3.0.
```

Luego, ejecuta flutter pub get y podrás usar el paquete en tu código:

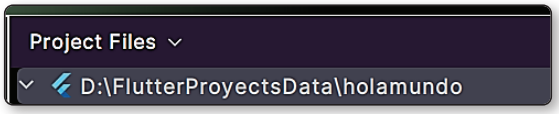
```
import 'package:http/http.dart';
```

Utilizando recursos

Los recursos son archivos que se incluyen en la aplicación, como imágenes, iconos, audios, fuentes, etc.

Crea la carpeta assets para guardar imágenes en el proyecto.

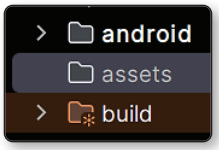
Clic derecho en el nombre del proyecto.



Del nombre del proyecto navegas New y de aquí a Directory.

Al dar clic derecho sobre Directory abre una casilla para que escribas el nombre de esa nueva carpeta que se llamará assets.

Dale Enter y observa la carpeta assets en el árbol de archivos y paquetes.



Para utilizar una imagen en tu aplicación, debes seguir estos pasos

1. Añade la imagen en la carpeta assets de tu proyecto.
2. Declara la imagen en el archivo pubspec.yaml.
3. Utiliza el Widget Image.asset() para mostrar la imagen en tu código.

Ejemplo

```
 assets:
    - assets/vademecum.jpg.
Image.asset('assets/vademecum.jpg').
```

El archive pubspec.yaml es bien sensible a la indentacion lo que significa que la primera letra, carácter o fuente de los paquetes debe tener espacios predefinidos previamente.

En secciones posteriores de este libro habrá ejemplos para que este tema de agregar recursos a pubspec.yaml quede bien claro.

El archivo **pubspec.yaml** es uno de los más visitados por el desarrollador en Flutter. Escrito en formato YAML (un lenguaje de serialización de datos legible para humanos), es donde se definen todas las reglas, dependencias y recursos que la aplicación necesita para funcionar y compilarse correctamente.

A continuación un resumen de sus secciones

1. Identificación del Proyecto

- **name**: el nombre de su aplicación (debe ser en minúsculas y usar guiones bajos).
- **description**: una breve explicación de lo que hace el proyecto.
- **version**: define la versión de su app (ej. 1.0.0+1). El número después del + es el número de compilación, vital para subir actualizaciones a la Play Store o App Store.

2. Entorno (Environment)

Esta sección es crucial para asegurar la compatibilidad con el SDK de Flutter que descargó manualmente.

- **sdk**: especifica el rango de versiones de Dart compatibles. Si está usando **Flutter 3.38.7**, este archivo se asegura de que el código no se ejecute en versiones antiguas que podrían generar errores de compilación.

3. Dependencias (dependencies y dev_dependencies)

- **dependencies**: aquí se listan los paquetes o librerías externas que su app necesita para ejecutarse, como http para internet o provider para el estado.

  Hay muchos paquetes que deben instalarse aquí, según lo requiera el proyecto.

- **dev_dependencies**: paquetes que solo se usan durante el desarrollo (como herramientas de prueba o generadores de código), pero que no se incluyen en la versión final que el usuario descarga.

4. Recursos y Activos (Flutter)

Bajo la etiqueta flutter, se configuran los elementos visuales que no son código:

- **uses-material-design: true**: habilita los iconos y componentes de Material Design.
- **assets**: es la lista de imágenes, vídeos o archivos JSON que desea incluir en su app.
- **fonts**: donde se declaran las tipografías personalizadas que darán estilo a sus textos.

Resumen

En este capítulo, hemos aprendido sobre la estructura de un proyecto Flutter, cómo gestionar paquetes y dependencias, y cómo utilizar recursos como imágenes e iconos. Se conoció la importancia de conocer la ubicación de la carpeta lib y su principal contenido el archivo main.dart.

Los archivos y carpetas más visitados mientras se desarrollan apps con Flutter son la carpeta lib y su archivo main.dart, el archivo pubspec.yaml en donde se agregan las dependencias. La página pub.dev es la que proporciona valiosos paquetes externos o creados por terceros y que ahorran trabajo al proveer datos y funcionalidades empaquetadas.

Preguntas

1. ¿Qué es el archivo pubspec.yaml y para qué sirve?
2. ¿Cómo se añade un paquete a un proyecto Flutter?
3. ¿Dónde se almacenan las imágenes en un proyecto Flutter?
4. ¿Cómo se declara una imagen en el archivo pubspec.yaml?
5. ¿Qué Widget se utiliza para mostrar una imagen en Flutter?
6. Donde se encuentra el archivo main.dart.
7. Defina la importancia de la carpeta lib.
8. En qué carpeta se encuentran las imágenes para el icono de iOS.

Ejercicios

1. Añade el paquete url_launcher a tu proyecto.
2. Crea una aplicación que muestre una imagen desde la carpeta **assets.**
3. Investiga cómo utilizar iconos en Flutter.
4. Dedica 15 minutos a abrir carpetas y subcarpetas en el directorio de archivo del proyecto.

8

MÁS SOBRE EL IDE

Objetivos

- Conocer el IDE Android Studio.
- Identificar las herramientas de desarrollo del IDE.

Introducción

Conocer las herramientas, facilidades y ayuda que el entorno de desarrollo (IDE) Android Studio ofrece para programar con Flutter es crucial para aumentar la productividad en el desarrollo.

En este capítulo damos detalles de cómo encontrar y utilizar cada elemento de Android Studio.

No profundizamos en Android Studio porque para eso hay un libro de la serie "Desarrollo practico de Apps con Dart, Flutter y Android Studio" dedicado completamente a Android Studio para flutter.

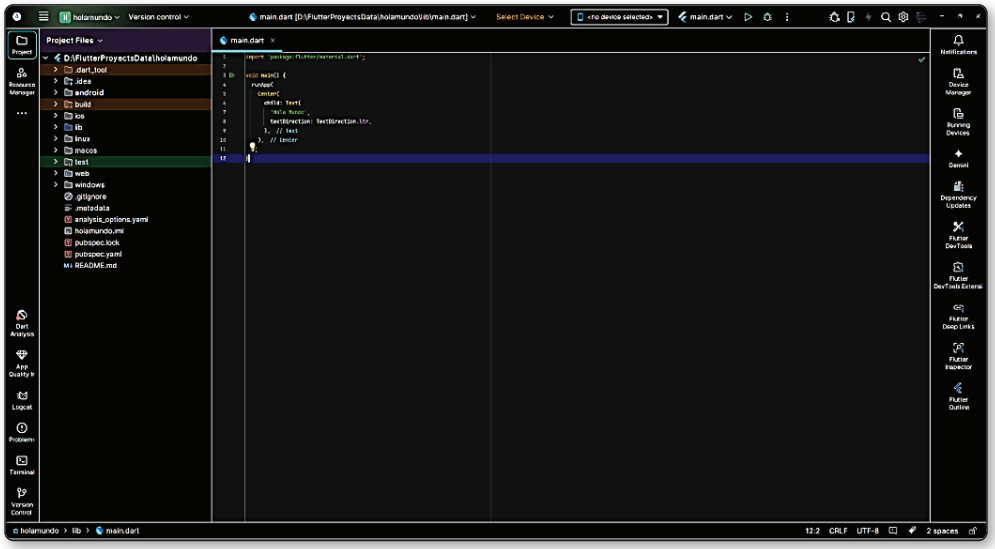
Sí, explico lo elemental para manejar el IDE para el desarrollo de tus proyectos con Flutter.

Android Studio

Es un entorno de desarrollo presentado en mayo del 2013 en la conferencia I/O de Google. La primera versión estable se publicó en diciembre del 2014.

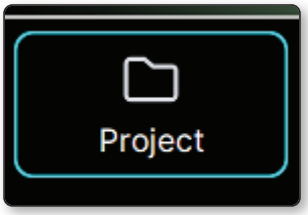
Tiene editor, emulador y muchas características como sugerencia para completar código. Y ahora tiene incorporado a Gemini la IA de Google.

Abre Android Studio presionando el icono correspondiente. A estas alturas ya lo debes tener instalado por lo que solo busca si hay versiones más recientes para que las instales.



En la imagen anterior puedes ver el aspecto general que tiene Android Studio.

A continuación se muestran secciones individuales del IDE.



Contiene los archivos del proyecto.

Si los archivos no son visibles, abre esta carpeta y se desplegará el directorio de archivos, paquetes y recursos como se ve a continuación.