salesians
SARRIÀ

Programación multimedia y dispositivos móviles.

Introducción a Flutter

Alumno: Damian Altamirano Ontiveros Rivero.

Índice

0. Perspectiva General

Flutter es un framework de desarrollo de aplicaciones creado por Google que permite construir apps nativas para Android, iOS, web y escritorio desde un solo código. Utiliza el lenguaje Dart y se basa en widgets personalizables para crear interfaces modernas y fluidas.

Su motor gráfico propio garantiza alto rendimiento sin depender de componentes nativos. Además, el hot reload acelera el desarrollo al reflejar cambios en tiempo real.

Empresas como Google, BMW y Alibaba lo usan por su eficiencia y escalabilidad. Flutter destaca por su versatilidad, rapidez y capacidad multiplataforma.

1. Instalación de los diferentes entornos de desarrollo (IDEs)

1.0 Instalación de Flutter SDK y configuración.

Es muy importante antes de instalar y configurar nuestros IDEs el que descarguemos y configuremos el SDK de Flutter.

El SDK de Flutter (Software Development Kit) es un conjunto de herramientas que permite desarrollar aplicaciones móviles, web y de escritorio desde una sola base de código, usando el lenguaje Dart. Incluye todo lo necesario para crear interfaces gráficas rápidas y atractivas: compilador, motor de renderizado, librerías de widgets, y utilidades para probar y depurar tus apps. Con Flutter SDK, puedes construir aplicaciones nativas para Android, iOS, Windows, macOS, Linux y la web, sin tener que escribir código diferente para cada plataforma.

1.0.1 Requerimientos Previos

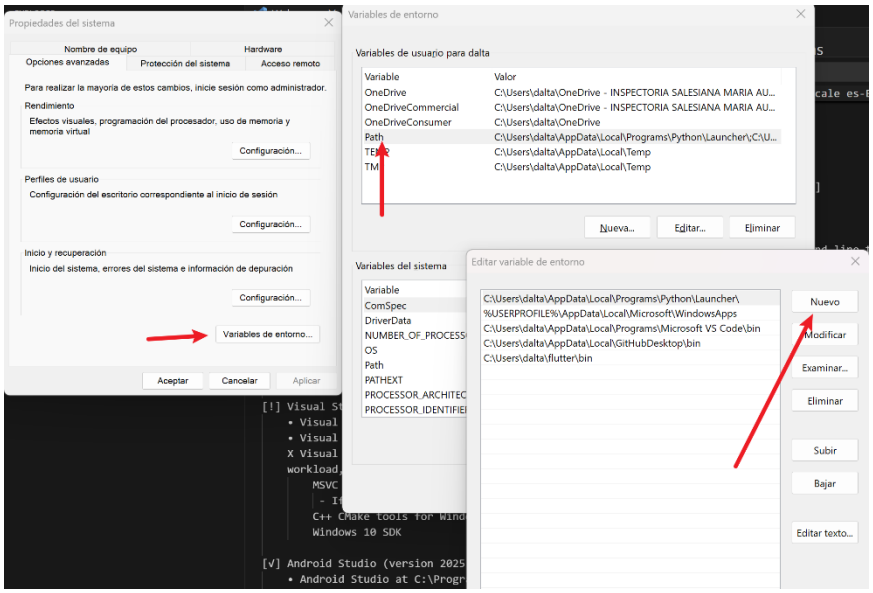
Deberemos de tener instalada la versión más reciente de Git, así como tener los privilegios necesarios para poder editar variables de entorno dentro de nuestro usuario. En nuestro caso, al tener control total editaremos las del sistema

1.0.2 Instalación

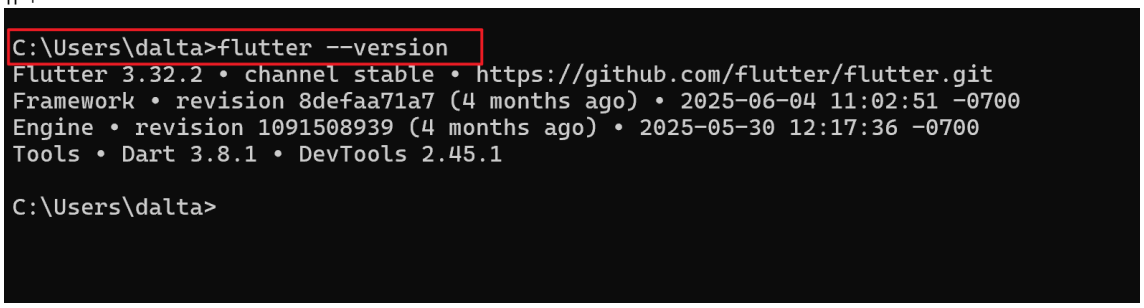
Previamente deberemos de descargar la última versión estable del SDK Flutter:
> Flutter SDK 3.32.2

Una vez terminada la descarga, deberemos de descomprimir el zip en algún directorio en el que tengamos suficientes privilegios (como puede ser nuestro user) y anotamos el PATH.

Luego, haremos Windows + s y buscaremos “Variables de entorno del Sistema/o Usuario” y añadiremos la ruta del bin de nuestro flutter sdk.



Una vez hecho esto podemos verificar desde la línea de comando si se ha agregado correctamente con el comando:
|| $ flutter -- version



Con esto hecho, estaremos listos para comenzar con lo demás.

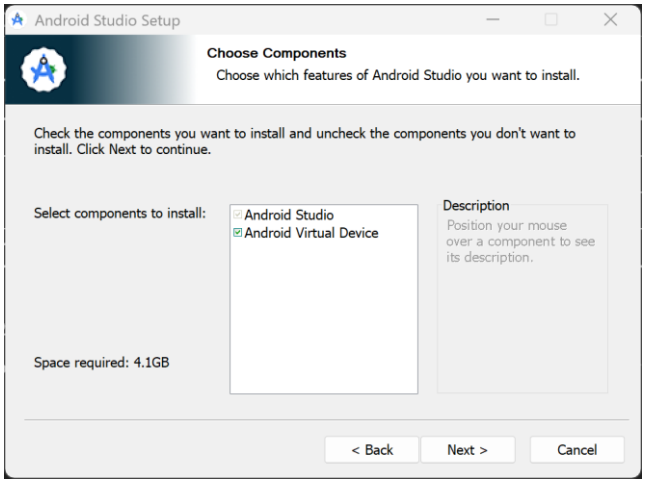
1.1 Android Studio 2025

1.1.1 Instalación

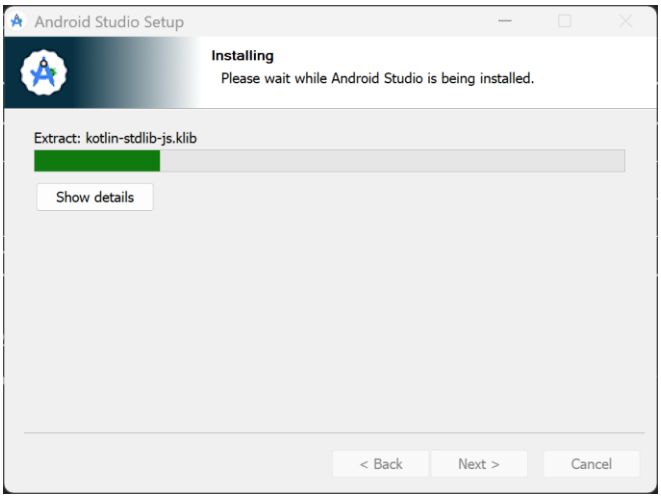
Lo primero es descargar la versión de Android Studio 2025.1.3.7 mediante el siguiente enlace:
> Android Studio 2025.1.3.7

Luego, ejecutaremos el instalador, dándole los permisos que este requiera:

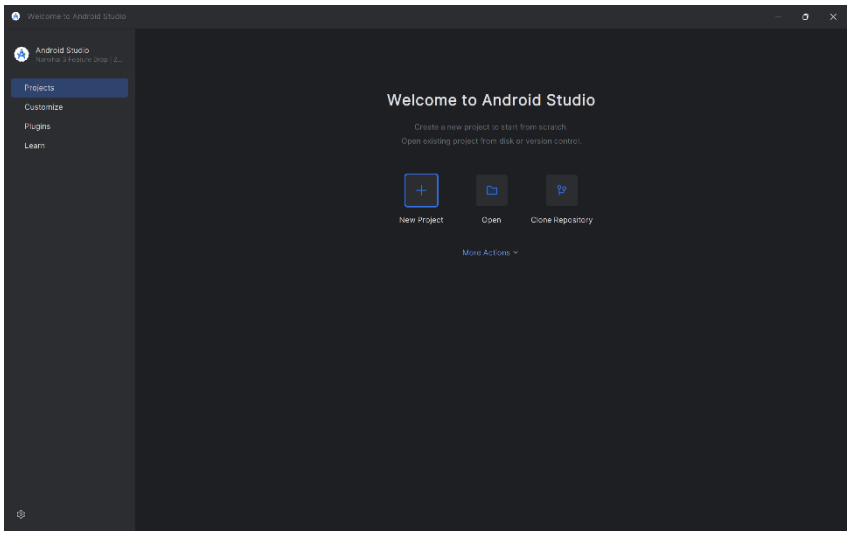
Instalaremos el componente **Android Virtual Device**:



Una vez hecho esto, seleccionamos la ruta de instalación y damos a instalar:



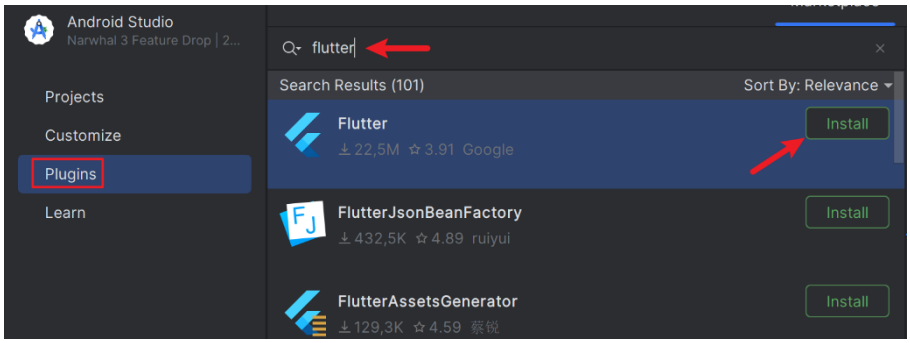
Una vez hecho esto ya tendremos instalado nuestro IDE:



1.1.2 Instalación de plugins

Ahora, instalaremos los plugins necesarios para poder trabajar con el entorno de Flutter y Dart.

Nos vamos a **Plugins > “Flutter” > Install**



Al instalarlo, se nos instalará automáticamente Dart también. Una vez finalizada la instalación reiniciamos nuestro IDE.

1.2 Visual Studio Code

Ahora configuraremos el entorno de Visual Studio Code como alternativa a Androide Studio para los que lo deseen.

1.2.1 Instalación

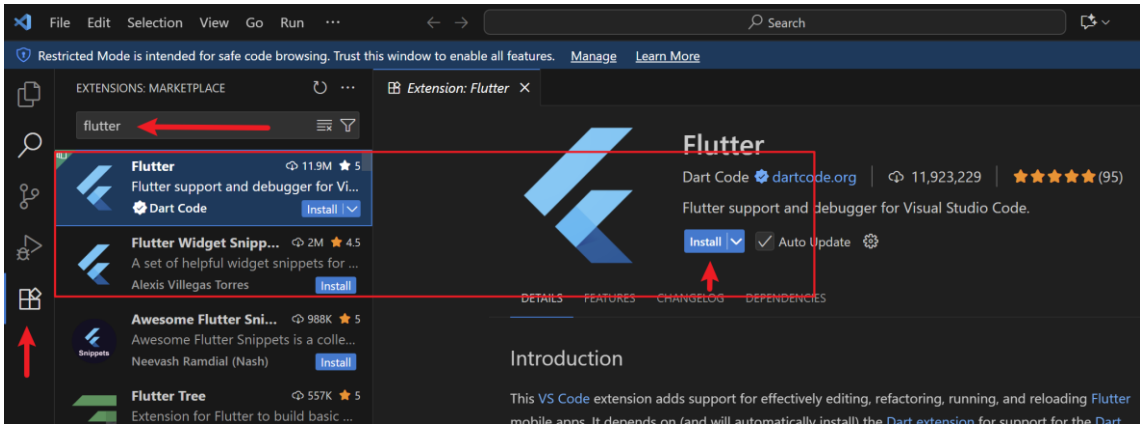
Previamente deberemos de instalar nuestro editor de código mediante el siguiente enlace:

- Visual Studio Code last version

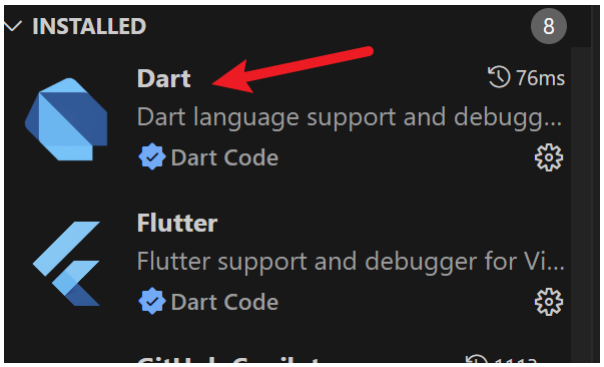
Omitiremos el proceso de instalación puesto que nos centraremos en configurar un entorno ya existente de VS Code para el desarrollo de apps mobiles en Flutter.

1.2.2 Plugins

No iremos a **Extensions > “flutter” > Install**:



Una vez haya terminado la instalación, podemos irnos al gestor de extensiones y verificar que se ha instalado correctamente. Además, veremos que automáticamente se instalará Dart como extensión.



Si deseamos hacer una validación más exhaustiva, podemos iniciar un **doctor process** para verificar su funcionamiento.

Entramos en **View > Command Palette > "Flutter: Run Flutter Doctor"**(Previamente, deberemos tener descargado el SDK en nuestro equipo local y git).

```
C:\Users\dalta>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.32.2, on Microsoft Windows [Versi¢n 10.0.26100.6584], locale es-ES)
[√] Windows Version (Windows 11 or higher, 24H2, 2009)
[√] Android toolchain - develop for Android devices (Android SDK version 36.1.0)
[X] Chrome - develop for the web (Cannot find Chrome executable at .\Google\Chrome\Application\chrome.exe)
    ! Cannot find Chrome. Try setting CHROME_EXECUTABLE to a Chrome executable.
[!] Visual Studio - develop Windows apps (Visual Studio Community 2019 16.11.51)
    X Visual Studio is missing necessary components. Please re-run the Visual Studio installer for the "Desktop
      development with C++" workload, and include these components:
        MSVC v142 - VS 2019 C++ x64/x86 build tools
          - If there are multiple build tool versions available, install the latest
        C++ CMake tools for Windows
        Windows 10 SDK
[√] Android Studio (version 2025.1.3)
[√] VS Code (version 1.104.2)
[√] Connected device (2 available)
[√] Network resources

! Doctor found issues in 2 categories.

C:\Users\dalta>
```

Si hemos hecho bien todos los pasos anteriores, podremos ver como se ejecuta in diagnostico de funcionamiento en la terminal donde nos importa que el apartado del Android este verificado. Con esto, ya estaremos listos para comenzar con nustro primer proyecto de flutter.

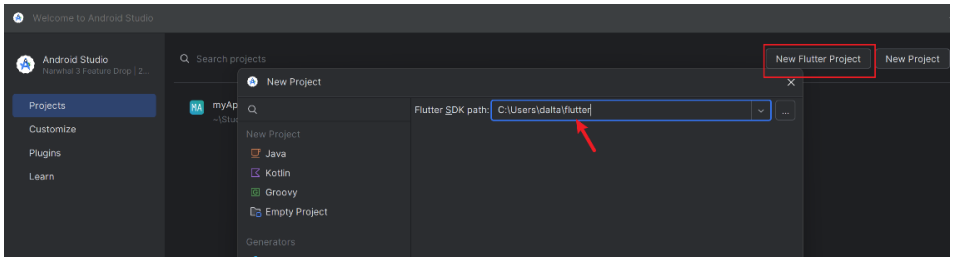
2. Creación de un nuevo proyecto (my_first_app)

Ahora que tenemos los entornos instalados y con sus respectivos plugins configurados, crearemos un primer proyecto muy sencillo (Hello_World) para cada IDE que usemos y veremos las funcionalidades que este nos trae.

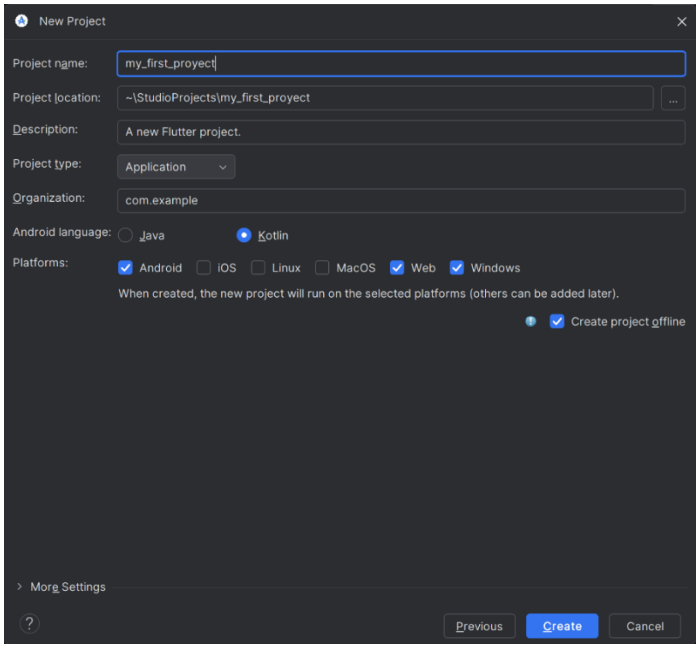
2.1 Android Studio

2.1.1 Creación del Proyecto

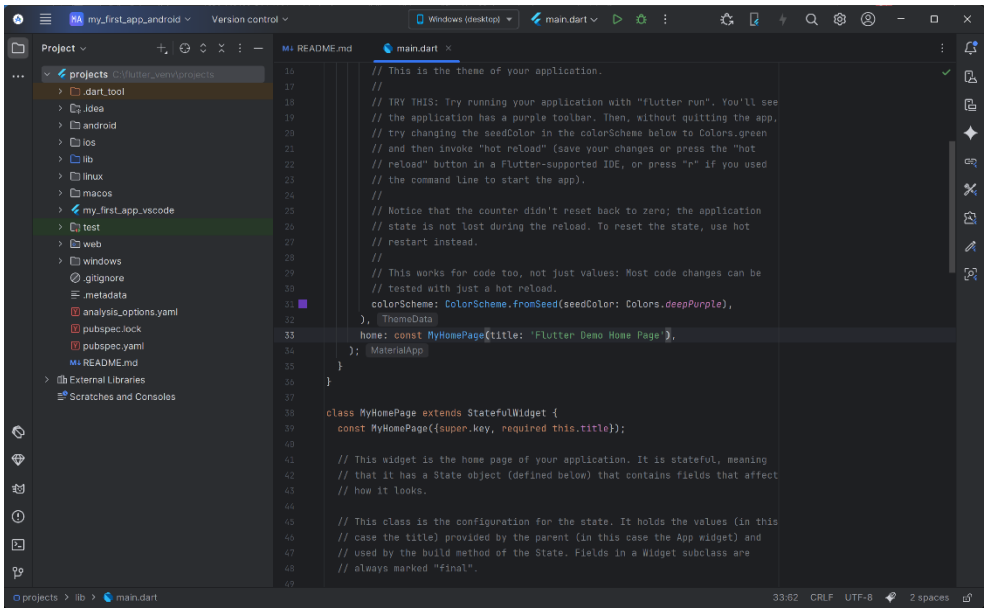
Ejecutamos nuestro IDE y seleccionamos **New Flutter Project** en la interfaz principal. Luego añadiremos el PATH de nuestro SDk



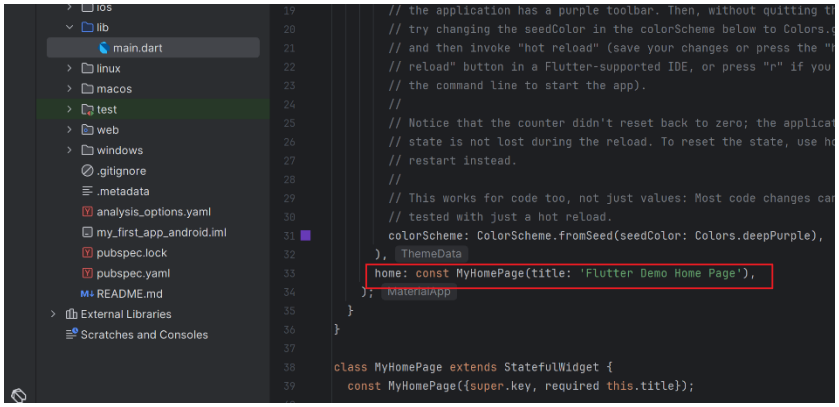
Al darle a next pasaremos a una pantalla en la que pondremos el nombre de nuestro proyecto y seleccionaremos en las plataformas en las que deseamos desarrollarlo.



Esperaremos un poco a que el proyecto se cree y nos aparecerá la siguiente interfaz:



Desde el panel izquierdo podremos ver las rutas de archivos que tendremos del proyecto. Una de ellas que es fundamental es **main.dart**, que será la que editaremos para nuestro proyecto.



Cuando creamos un proyecto nuevo, por predeterminado se nos crea una aplicación de contador, por lo que dejaremos todo tal y como esta y sol modificaremos la constante **MyHomePage**, para poder visualizar nuestros datos personales.

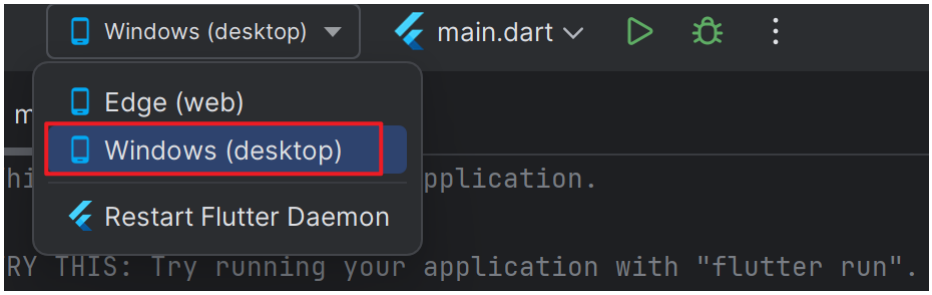
```
29          // This works for code too, not just values: Most code changes can be
30          // tested with just a hot reload.
31          colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
32        ), ThemeData
33        home: const MyHomePage(title: 'Flutter Demo Damian Alt. 2526'),
34      ); MaterialApp
35    }
36  }
```

Cabe recalcar que la linea de codigo que modificamos es la 33 del main.dart en default.

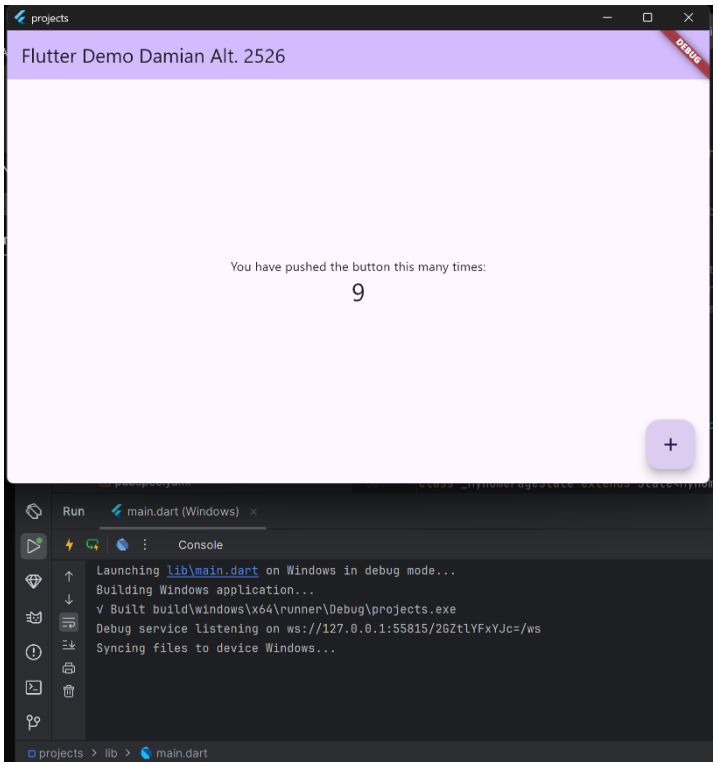
2.1.2 Debugging para Windows

Una vez guardado los cambios, debemos asegurarnos de que nuestro programa se ejecuta de forma correcta en nuestros dispositivos. Para ello, Android Studio cuenta con múltiples herramientas como emuladores y mirrorings que nos ayudarán con esta tarea.

Si deseamos ver cómo se debuggea nuestra app nuestro equipo Windows como app, bastará con irnos al **Flutter device selection**, que se encuentra en la parte superior de la interfaz:

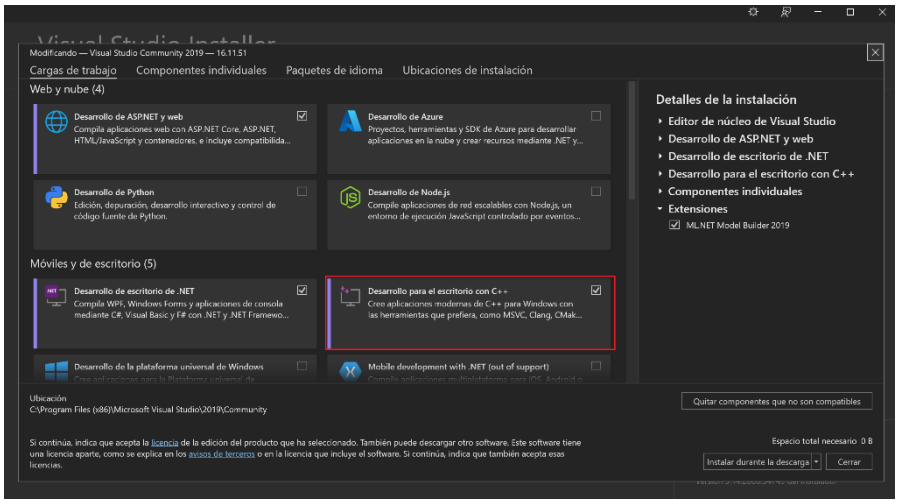


Dentro de este podremos seleccionar multiples entornos que tengamos disponibles. Por el momento, seleccionaremos **Windows (desktop)** y luego **Run**. Tomará unos segundos en que se ejecute nuestro programa, pero una vez haya finalizado se verá tal que así:



Como podemos ver en el encabezado, nuestras modificiaciones se han aplicado exitosamente.

Una nota muy importante es que si queremos hacer uso de la interfaz de windows deberes de tener instalado el Visual Studio con el workload de **Desarrollo para el escritorio ocn C++**.



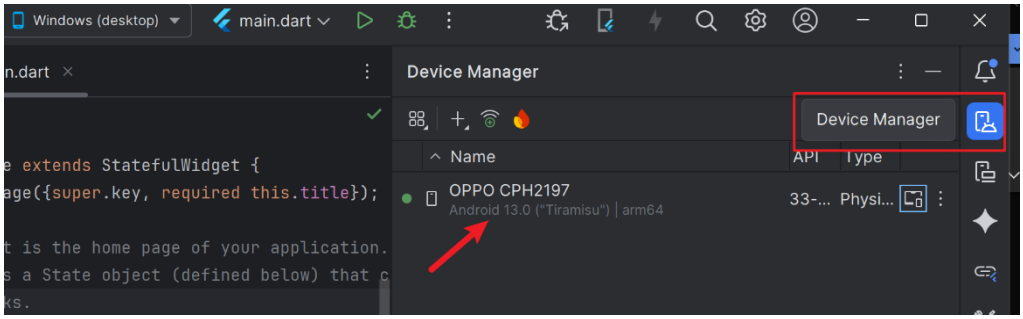
Si deseamos validar que nuestro SDK lea este recibiendo correctamente este entorno, bastara con ejecutar el comando:
|| $flutter doctor

2.1.3 Debugging para Android

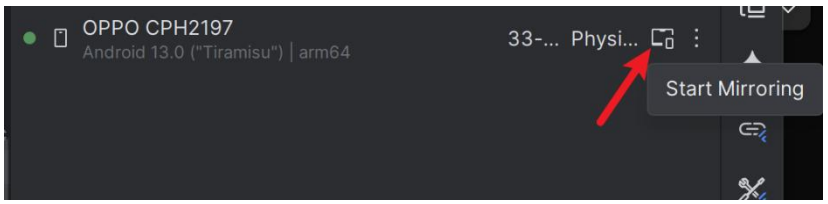
Pero nuestro mayor interes, es poder ver que nuestro proyecto pueda ejecutarse en un dispositivo mobile, por lo que explicaremos paso a paso como hacerlo.

Primero, deberemos habiliatar en nuestro dispositivo las **funciones de desarrollador** para evitar cualquiero problema de permisos y ejecutamiento. Esta habilitación será diferente dependiendo del modelo de mobile que tengamos.

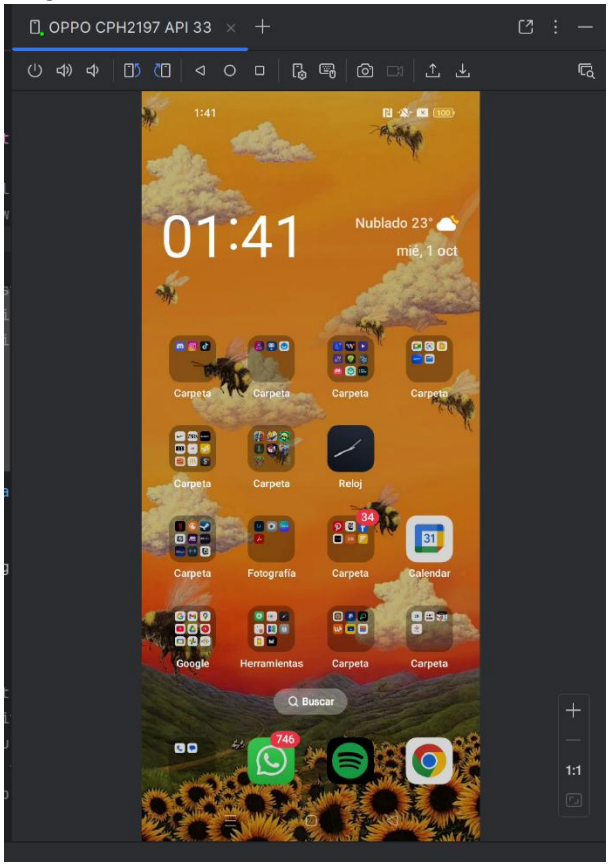
Una vez hecho, conectaremos nuestro mobile a nuestro ordenador mediante cable. Si hemos habilitados las opciones de desarrollador correctamente, podremos ver como nos aparece la imagen de nuestro dispositivo desde **device manager**:



Desde aquí, podremos hacer uso de una herramienta muy útil para monitorear en tiempo real nuestro mobile, que es el **Start Mirroring**.

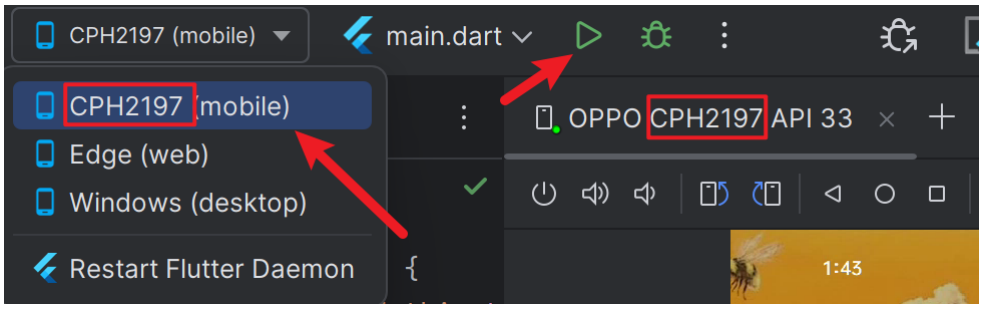


En este caso, haremos uso de ella, por lo que activaremos la opción y esperaremos a que se cargué.

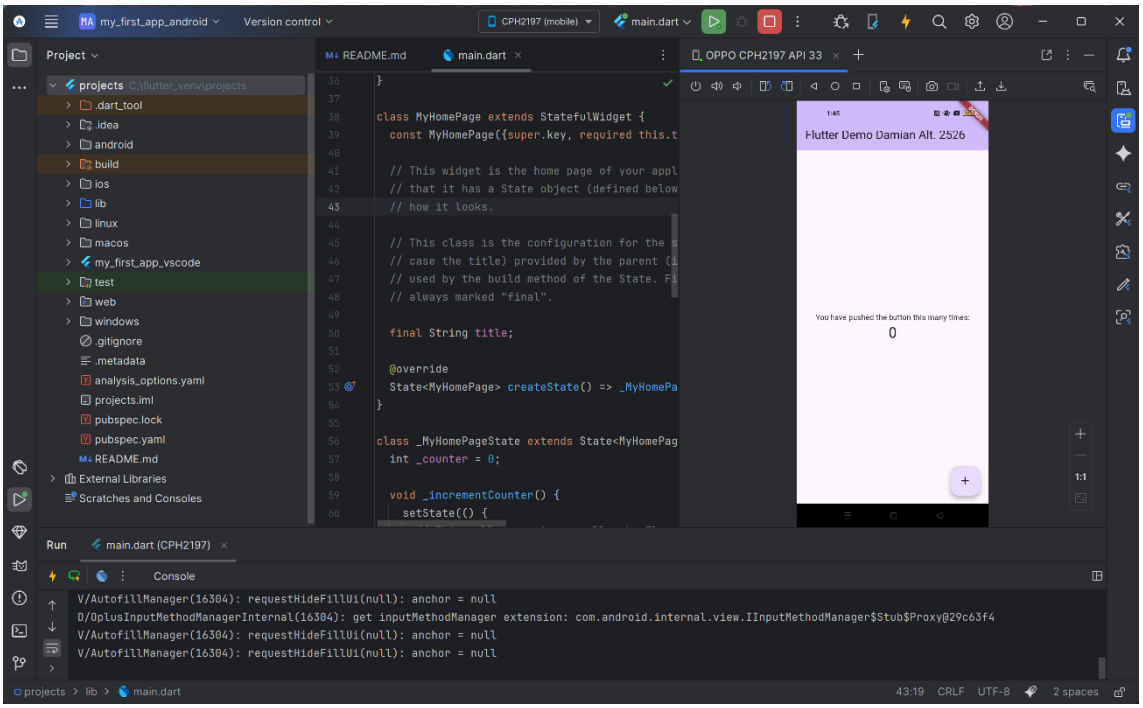


Una vez hecho esto, podremos ver e interactur por nuestro mobile desde el ordenador.

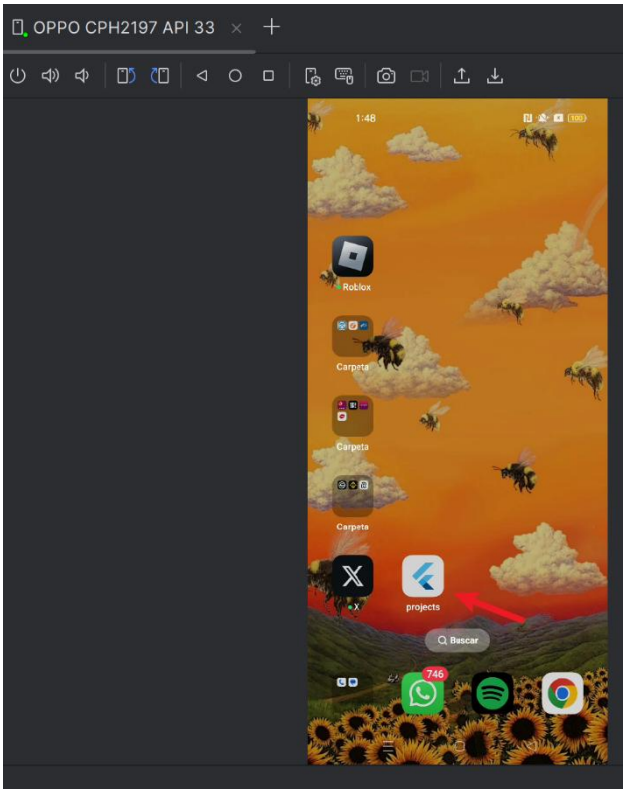
Ahora, volveremos al **Flutter device selection** para ejecutar nuestro programa mientras vemos como se inicia este en nuestro mobile:



Esperado unos segundos, se instalara la app en nuestro mobile y se abrirá la interfaz de inicio de esta.



Como vemos, desde el device mirroring podemos monitorear e interactuar con nuestra app desde el ordenador, como ver lo que hacemos con nuestro mobile por separado. También, tendremos un control y registro constante de lo que hagamos que se irá viendo desde la **console**.

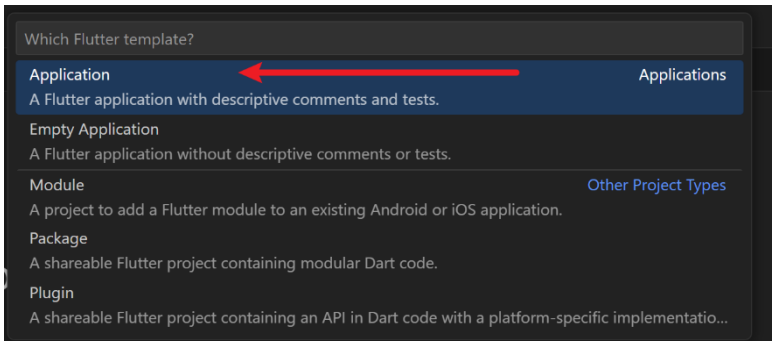


Habiendo hecho esto y sabiendo usar algunas herramientas fundamentales, ya estaremos listos para empezar con futuros proyectos desde nuestro IDE de Android Studio.

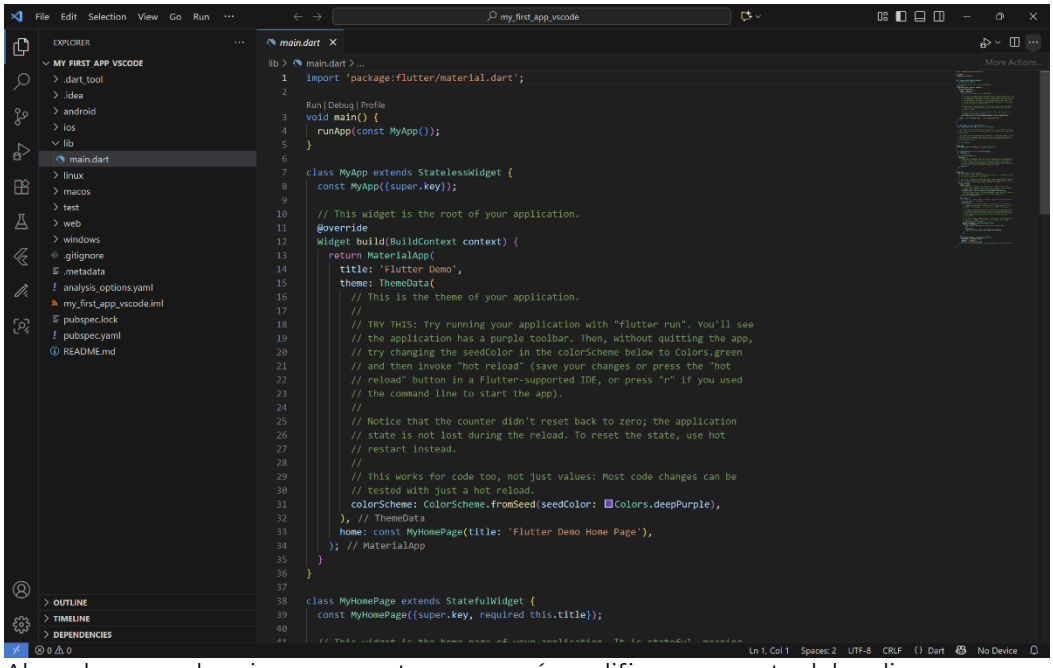
2.2 Visual Studio Code

2.2.1 Creación del Proyecto

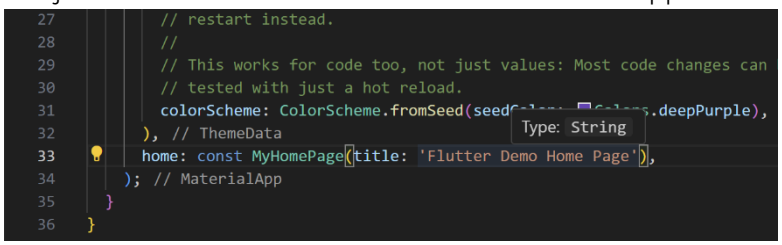
Iniciaremos nuestro editor de código y abriremos el **Command Palette** mediante el comando ctrl + Shift + p. Una vez en el buscador, buscaremos **Flutter: New Project**, y seleccionaremos **Application**.

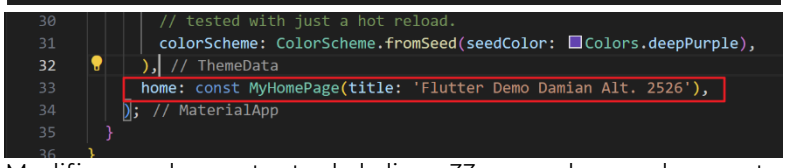


Luego, añadimos la ruta donde se guardara nuestro proyecto y le asignamos un nombre. Esperamos un momento y se nos cargarán todos nuestro archivos necesarios tal y como cuando lo hicimos en el Android Studio.



Ahora haremos lo mismo que antes, que será modificar una parte del codigo para que se reflejen nuestros datos en el encabezado de nuestra app.





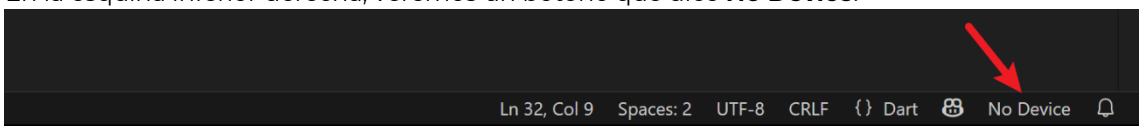
Modificamos la constante de la linea 33 y guardamos el proyecto.

2.2.2 Debugging para Web

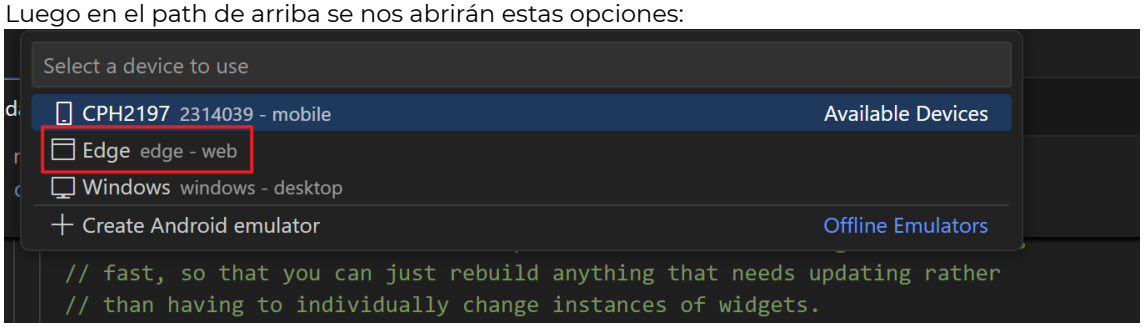
Ahora veremos que visual studio nos ofrecen herramientas similares al Android Studio para debugging, solo que desde un entorno diferente.

Cabe aclarar que el debug para Windows lo podremos hacer para el VS Code, como para el Android Studio, por lo que en este apartado veremos como hacerlo desde nuestro navegador.

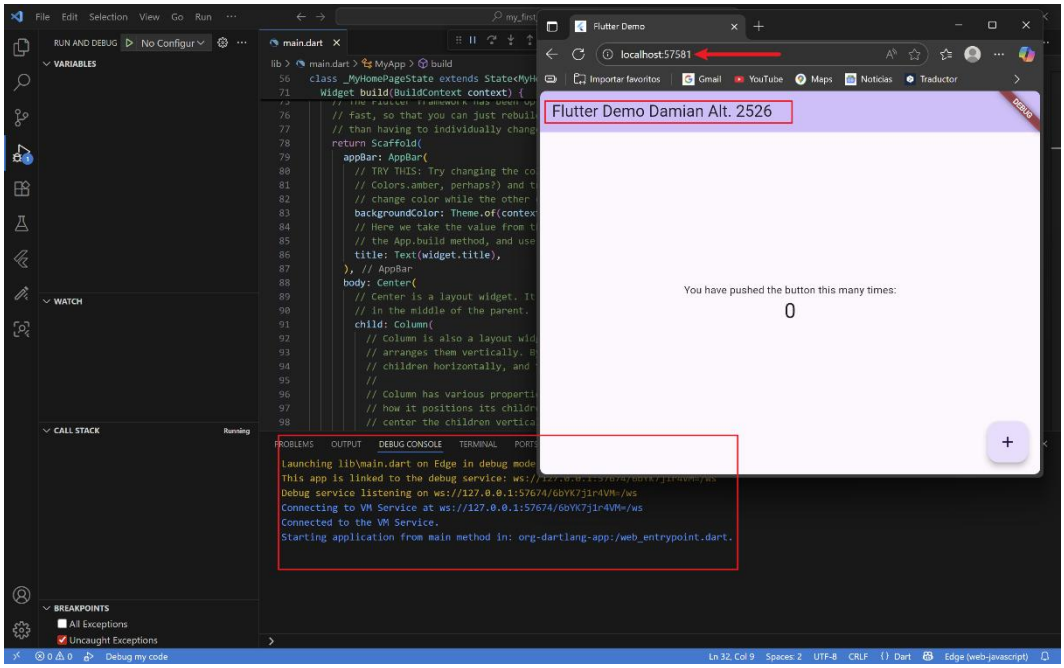
En la esquina inferior derecha, veremos un botono que dice **No Device**:



Luego en el path de arriba se nos abrirán estas opciones:



Damos clic sobre **Edge** y de forma rápida se nos cargará nuestra app en nuestro navegador de forma local.



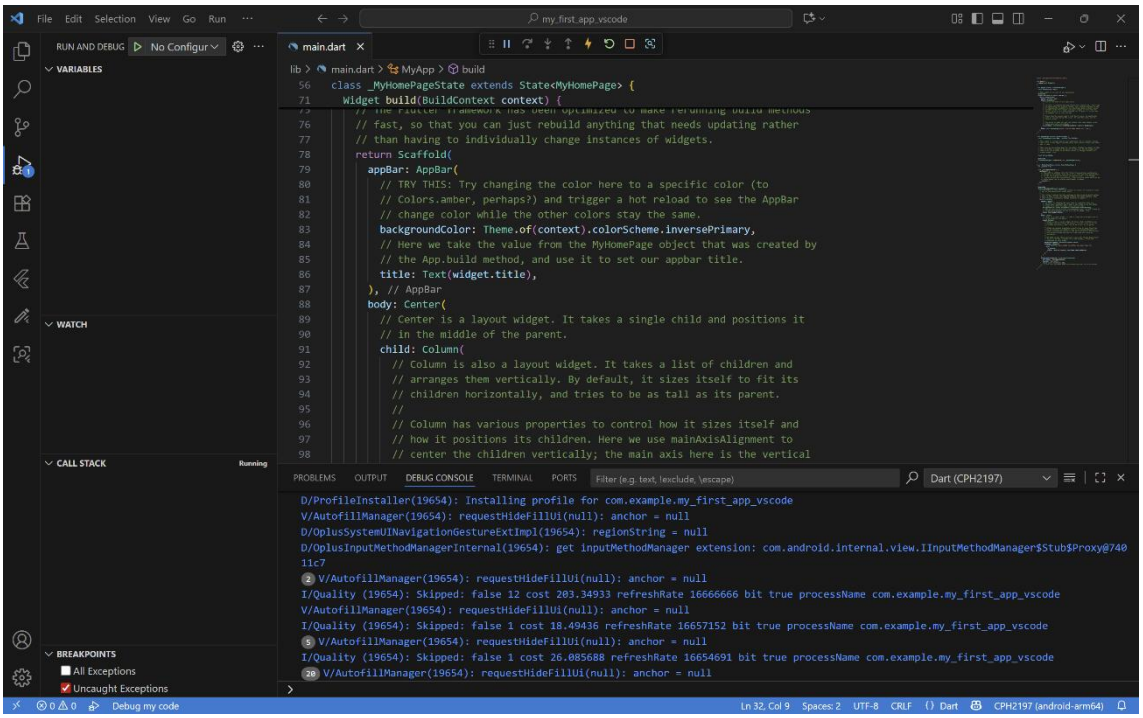
Una ventaja de hacerlo de esta manera es que a comparación de otros dispositivos, el navegador tiende a cargar mucho mas rapido, por lo que será útil para revisiones rapidas.

2.2.3 Debugging para Android

Una limitante de VS Code es que no cuenta con un emulador o herramienta de mirroring nativa para poder visualizar en tiempo real nuestro dispositivo. Sin embargo, podremos seguir debuggearlo y ver desde el mobile como se carga nuestra app.

Nos vamos al selector de dispositivos, y seleccionamos nuestro mobile. Una vez hecho, le damos a **Run**.

Cabe recalcar que nuestro dispositivo deberá de estar en modo desarrollador y conectado a nuestro ordenador mediante cable.



Como vemos, desde el **Debug Console** podremos ver en tiempo real lo que hagamos en nuestro dispositivo.

```
20 V/AutofillManager(19654): requestHideFillUi(null): anchor = null
```

En este caso, llevamos 20 toques hechos.

Y con esto estaremos listos para poder empezar proyectos mas grandes.

3. Repositorio Github

Ahora que tenemos nuestros respectivos proyectos creados y debuggeados, subiremos cada uno de estos a un repositorio de github para poder llevar un mejor control de versiones de estos.

Veremos que dependiendo del entorno en el que estemos trabajando, nos podremos valer de herramientas útiles para poder subir nuestros proyectos a nuestro repositorio.

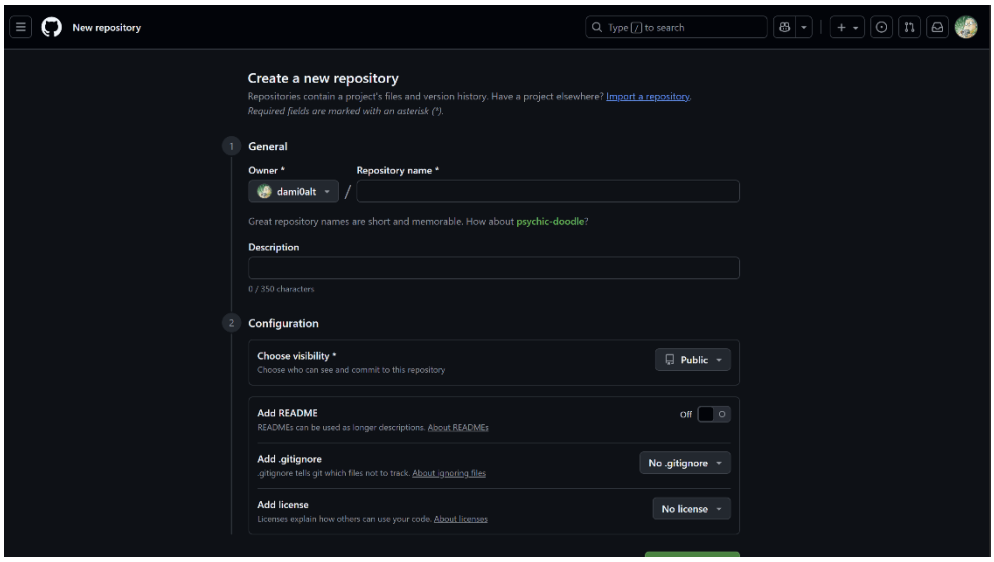
3.1 Creación de un repositorio en Github (web interface)

Antes de comenzar, veremos como crear un repositorio nuevo desde github.com y como configurarlo.

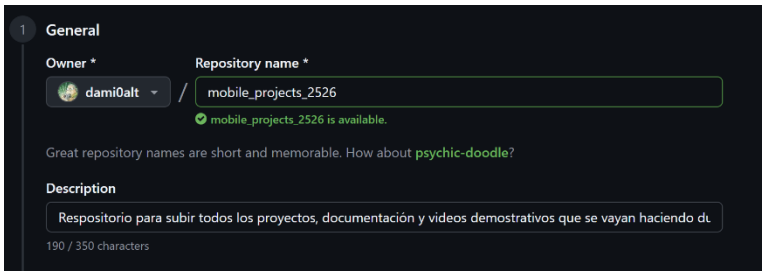
Antes deberemos de crear una nueva cuenta o loguearnos con una que ya tengamos. Una vez estemos dentro, podemos crear un nuevo repositorio desde el siguiente enlace:

>New repository

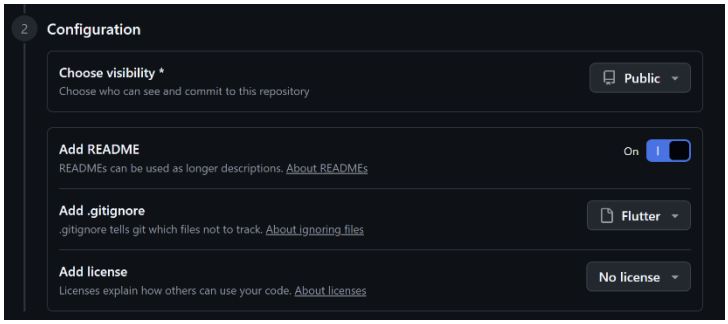
Una vez ingresemos nos aparecerá una interfaz como esta:



En **General** podremos añadir a nuestro equipo (si tuviéramos) y configurar sus roles. Darle un nombre a nuestro repositorio y una descripción si lo deseamos.

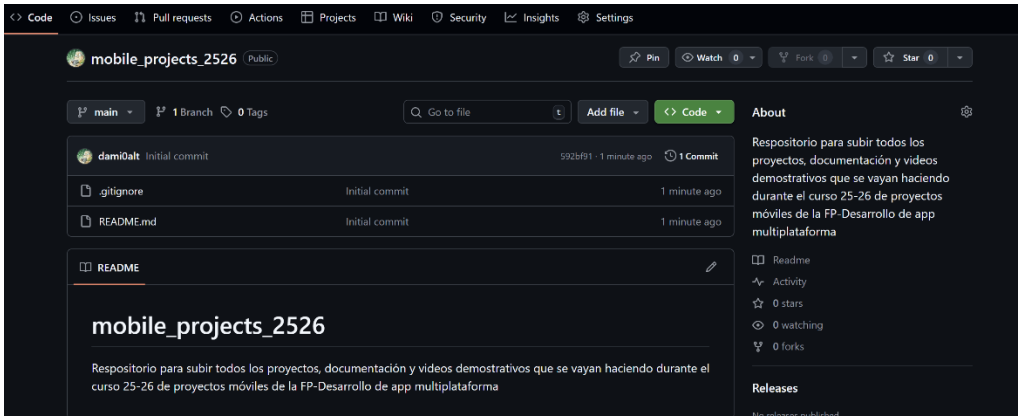


En **Configuration** podremos gestionar temas de visibilidad (pública o privada), Configurar un README, .gitignore o añadir licencias si se requiriesen.

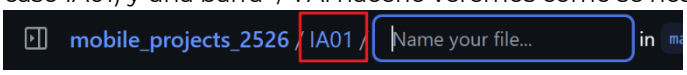


En mi caso dejaremos la visibilidad en **public**, añadiremos un README para explicar el flujo de nuestro repositorio y añadiremos una plantilla .gitignore > Flutter, ya que estaremos trabajando proyectos de este estilo.

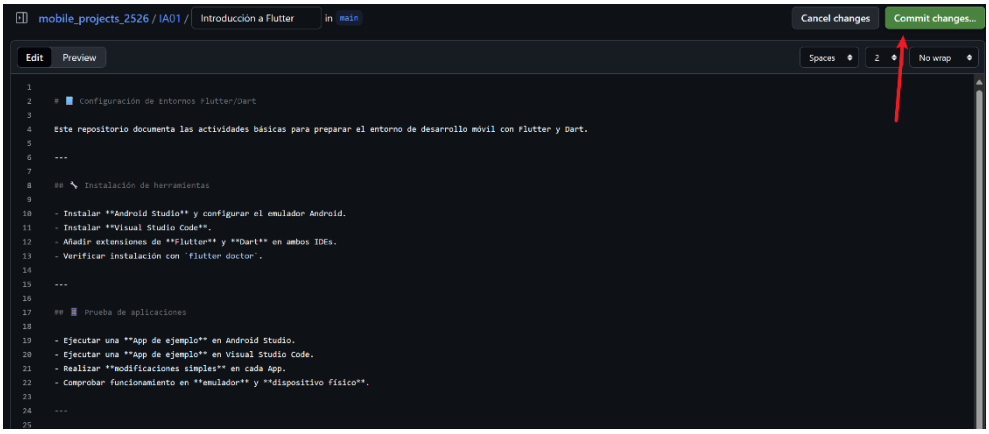
Por último, damos en **Create Repository** y ya tendremos listo nuestro repositorio para poder usarlo.



Ahora, crearemos una nueva carpeta dentro de nuestro repositorio donde pondremos nuestros dos proyectos. Damos clic en **Add File** > **Create new file** y escribiremos en el place holder “Name your file....” el nombre del fichero donde estaran nuestros proyectos (en este caso IA01) y una barra “/”. Al hacerlo veremos como se nos va generando una ruta respectiva.



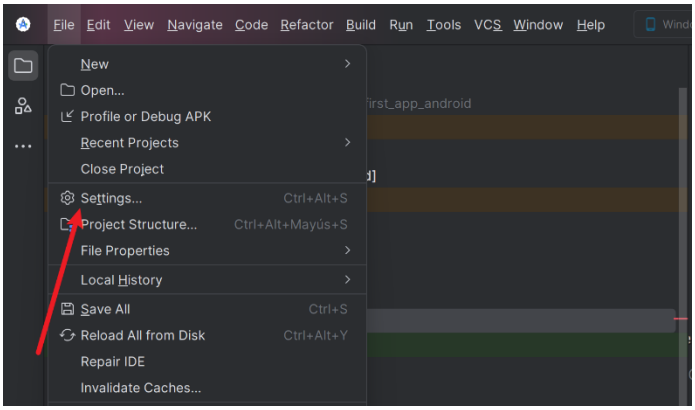
Una vez finalizado, damos a **Commit changes...**



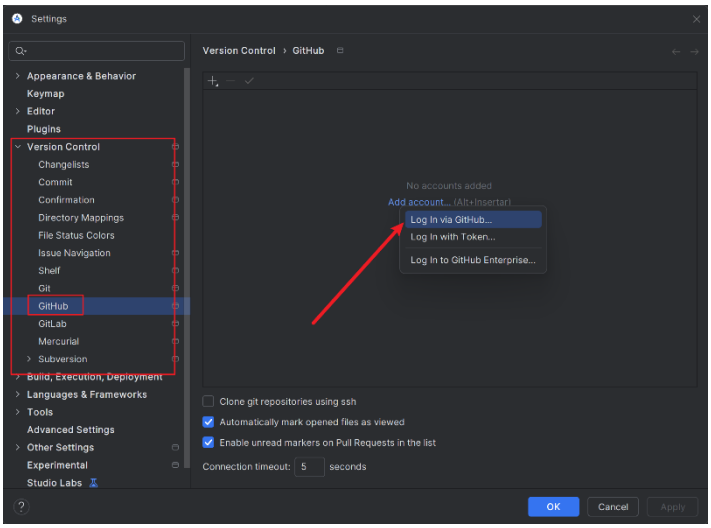
Guardado los cambios, ya estamos listos para subir nuestros proyectos a nuestro repositorio.

3.2 Clonación y primer commit

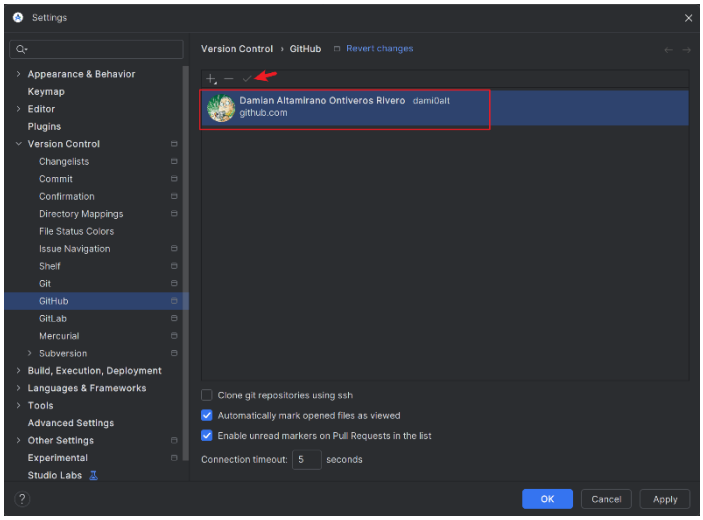
Empezamos abriendo nuestro proyecto ya en el Android Studio y luego nos dirigimos a **View > Settings.**



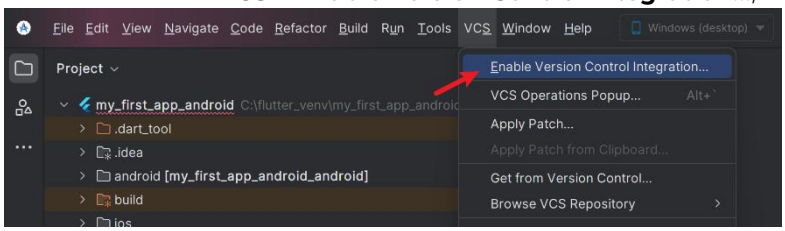
Ahora, nos dirigimos **Version Control > Github > Add account... > Log in via GitHub** y nos logueamos.

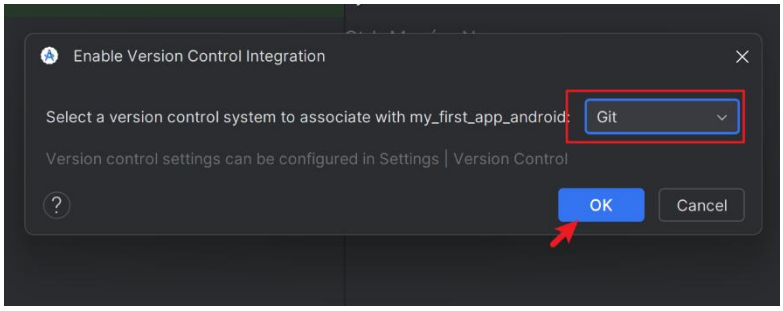


Si hemos hecho todo bien, nos aparecerá nuestra cuenta:



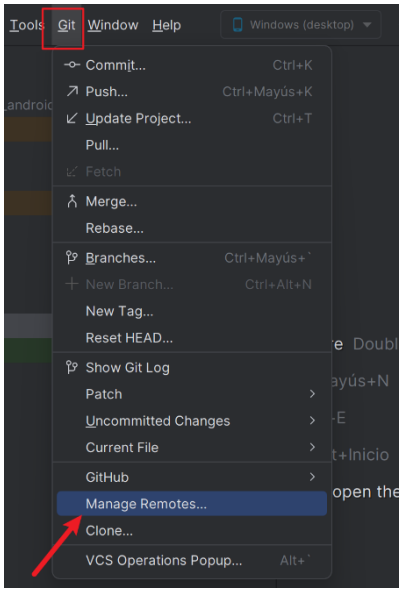
Ahora nos vamos **VCS > Enable Version Control Integration...**, luego seleccionamos Git:

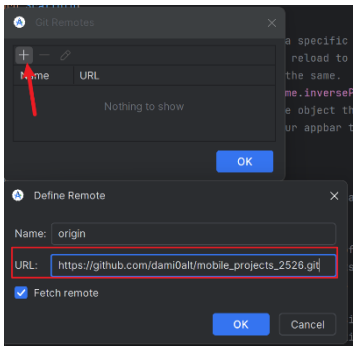




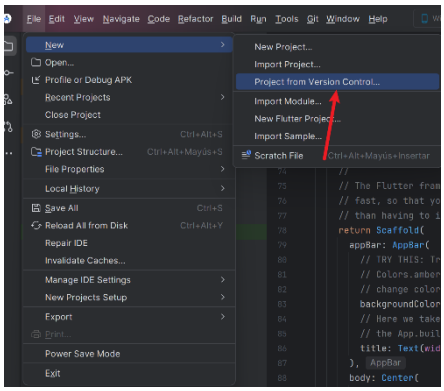
Con esto, Android Stuido empezara a rastrear los archivos del proyecto.

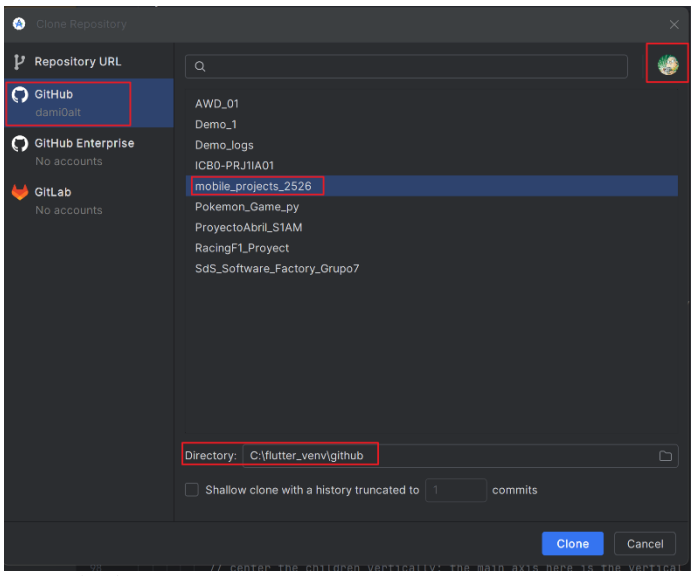
Ahora agregaremos el repositorio que ya hemos creado a nuesto IDE. Nos dirigimos a **Git > Manage Remotes... > +**, y en Name dejaremos la palabra **origin**, y pondremos la URL de nuestro repositorio.



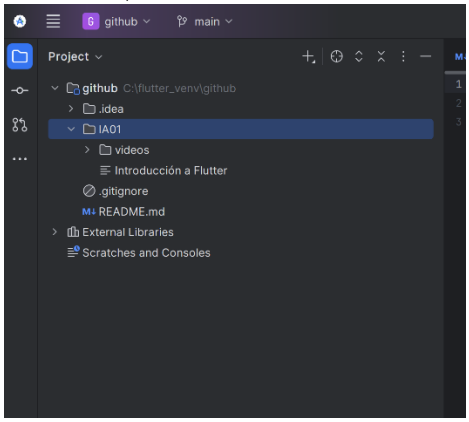


Ahora, buscaremos hacer un **commit** a nuestro repositorio para poder subir nuestro proyecto desde el Android Studio. Para ello clonaremos nuestro repositorio en local y haremos el commit. Nos vamos a **File > Project from Version Control... > GitHub** y una vez aquí, podremos ver nuestra cuenta de github vinculada, junto con nuestros respositorios. Seleccionamos el que nos interesa y luego el directorio donde se clonara:

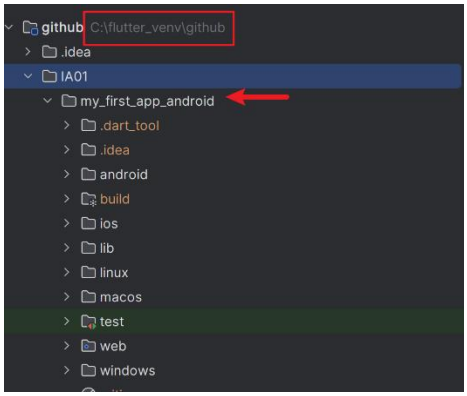




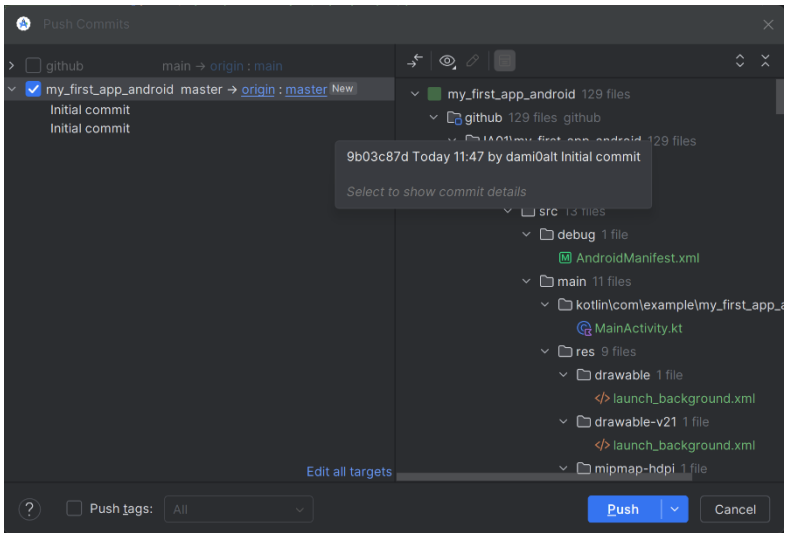
Hecho esto, se nos clonará en local nuestro repositorio:



Ahora bastará con mover todo nuestro projecto dentro de nuestra carpeta en local donde deseemos:



Bastará con ejecutar el comando ctrl + k y dar en **commit and push** si deseamos subir los cambios ahora:



Luego nos saltara la ventana del Push Commits donde podremos verificar los cambios hechos y hacia donde van, que en este caso será hacia el master. Una vez hecho, volvemos al github de web para revisar los cambios. Y con esto podremos ir subiendo nuestros cambios, gestionar versiones y branchs desde el mismo IDE.

Recursos

Videos explicativos de debug para los diferentes entornos de desarrollo (Android Studio y Visual Studio Code)

- Drive