

# **Bootcamp Desarrollo Movil con Flutter**

Diego Saavedra

Nov 20, 2024

# Table of contents

<b>1</b>	<b>Bienvenido</b>	<b>4</b>
1.1	¿De qué trata este Bootcamp? . . . . .	4
1.2	¿Para quién es este bootcamp? . . . . .	4
1.3	¿Qué aprenderás? . . . . .	4
1.4	¿Cómo contribuir? . . . . .	4
<b>I</b>	<b>Unidad 0: Introducción a Git y GitHub</b>	<b>6</b>
<b>2</b>	<b>Git y GitHub</b>	<b>7</b>
2.1	¿Qué es Git y GitHub? . . . . .	7
2.2	¿Quiénes utilizan Git? . . . . .	8
2.3	¿Cómo se utiliza Git? . . . . .	8
2.4	¿Para qué sirve Git? . . . . .	9
2.5	¿Por qué utilizar Git? . . . . .	10
2.6	¿Dónde puedo utilizar Git? . . . . .	11
2.7	Pasos Básicos . . . . .	11
2.8	Instalación de Visual Studio Code . . . . .	12
2.8.1	Descarga e Instalación de Git . . . . .	13
2.8.2	Configuración . . . . .	14
2.8.3	Creación de un Repositorio “helloWorld” en Flutter . . . . .	14
2.8.4	Comandos Básicos de Git . . . . .	17
2.8.5	Estados en Git . . . . .	17
2.8.6	Modificar Archivos en el Proyecto . . . . .	18
2.8.7	Creación y Fusión de Ramas . . . . .	18
2.8.8	Implementar Funcionalidades en la Rama: . . . . .	18
2.8.9	Fusionar Ramas con la Rama Principal: . . . . .	18
2.8.10	Revertir Cambios en un Archivo . . . . .	19
<b>3</b>	<b>Conclusión</b>	<b>20</b>
<b>4</b>	<b>Asignación: “Hello World!” en Flutter</b>	<b>21</b>
4.1	La Asignación . . . . .	21
4.2	Probar la Aplicación . . . . .	21
<b>5</b>	<b>GitHub Classroom</b>	<b>22</b>
5.1	¿Qué es GitHub Classroom? . . . . .	22
5.1.1	Funcionalidades Principales . . . . .	22
5.2	Ejemplo Práctico . . . . .	23
5.2.1	Creación de una Asignación en GitHub Classroom . . . . .	23
5.3	Trabajo de los Estudiantes . . . . .	25

<b>II</b>	<b>Unidad 1: Introducción e Instalaciones Necesarias</b>	<b>31</b>
<b>6</b>	<b>Instalaciones Necesarias.</b>	<b>32</b>
<b>7</b>	<b>Introducción a Dart</b>	<b>33</b>
7.1	¿Qué es Dart? . . . . .	33
7.1.1	Características. . . . .	33
7.2	¿Qué es Flutter? . . . . .	33
<b>8</b>	<b>Hola Mundo en Dart.</b>	<b>34</b>

# 1 Bienvenido

¡Bienvenido al Bootcamp de Desarrollo Movil con Flutter

En este bootcamp, exploraremos todo, desde los fundamentos hasta las aplicaciones prácticas.

## 1.1 ¿De qué trata este Bootcamp?

Este bootcamp está diseñado para enseñarle a desarrollar aplicaciones movil modernas con Dart y Flutter.

## 1.2 ¿Para quién es este bootcamp?

Este bootcamp es para cualquier persona interesada en aprender a desarrollar aplicaciones movil modernas.

## 1.3 ¿Qué aprenderás?

Aprenderás a desarrollar con el lenguaje de programación Dart y el Framework de Flutter.

## 1.4 ¿Cómo contribuir?

Valoramos su contribución a este bootcamp. Si encuentra algún error, desea sugerir mejoras o agregar contenido adicional, me encantaría saber de usted.

Puede contribuir a través del repositorio en línea, donde puede compartir sus comentarios y sugerencias.

Juntos, podemos mejorar continuamente este recurso educativo para beneficiar a la comunidad de estudiantes y entusiastas de la programación.

Este ebook ha sido creado con el objetivo de proporcionar acceso gratuito y universal al conocimiento.

Estará disponible en línea para cualquier persona, sin importar su ubicación o circunstancias, para acceder y aprender a su propio ritmo.

Puede descargarlo en formato PDF, Epub o verlo en línea en cualquier momento y lugar.

¡Esperamos que disfrute este emocionante viaje de aprendizaje y descubrimiento en el mundo del desarrollo móvil con Dart y Flutter!

## **Part I**

# **Unidad 0: Introducción a Git y GitHub**

## 2 Git y GitHub



Figure 2.1: Git and Github

### 2.1 ¿Qué es Git y GitHub?

- Git y GitHub son herramientas ampliamente utilizadas en el desarrollo de software para el control de versiones y la colaboración en proyectos.
- Git es un sistema de control de versiones distribuido que permite realizar un seguimiento de los cambios en el código fuente durante el desarrollo de software. Fue creado por Linus Torvalds en 2005 y se utiliza mediante la línea de comandos o a través de interfaces gráficas de usuario.
- GitHub, por otro lado, es una plataforma de alojamiento de repositorios Git en la nube. Proporciona un entorno colaborativo donde los desarrolladores pueden compartir y trabajar en proyectos de software de forma conjunta. Además, ofrece características adicionales como seguimiento de problemas, solicitudes de extracción y despliegue continuo.

En este tutorial, aprenderás los conceptos básicos de Git y GitHub, así como su uso en un proyecto de software real.

## 2.2 ¿Quiénes utilizan Git?



Figure 2.2: Git

Es ampliamente utilizado por desarrolladores de software en todo el mundo, desde estudiantes hasta grandes empresas tecnológicas. Es una herramienta fundamental para el desarrollo colaborativo y la gestión de proyectos de software.

## 2.3 ¿Cómo se utiliza Git?

```
commit e072c20b5577c37af7c4fb274b6b53d15dd336ae
Author: Julio Xavier <julioxavierr@live.com>
Date:   Fri Aug 19 16:17:10 2016 -0300

    Commit with error

commit a497c0c03657549e7d4c5ba1b23ffce5faaf46b8
Author: Julio Xavier <julioxavierr@live.com>
Date:   Mon Jan 11 10:51:42 2016 -0200

    Adding common html code in a form

commit 9fa7605ad1837aa44dfb9c711dc8bd60cab7c5d
Author: Julio Xavier <julioxavierr@live.com>
Date:   Sun Jan 10 22:29:52 2016 -0200

    Pages to show 'details' + Editing Clients
```

Figure 2.3: Git en Terminal



Se utiliza mediante la **línea de comandos** o a través de **interfaces gráficas** de usuario. Proporciona comandos para realizar operaciones como:

1. Inicializar un repositorio,
2. Realizar cambios,
3. Revisar historial,
4. Fusionar ramas,
5. Entre otros.

## 2.4 ¿Para qué sirve Git?



Figure 2.4: Seguimiento de Cambios con Git

Sirve para realizar un seguimiento de los cambios en el código fuente, coordinar el trabajo entre varios desarrolladores, revertir cambios no deseados y mantener un historial completo de todas las modificaciones realizadas en un proyecto.

## 2.5 ¿Por qué utilizar Git?

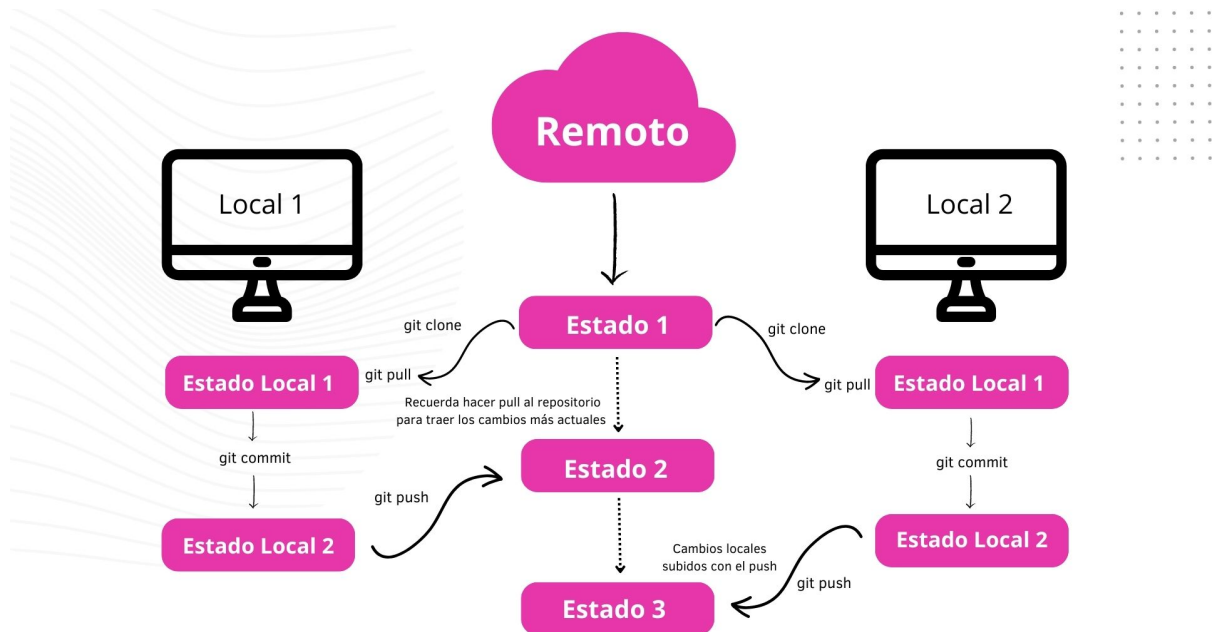


Figure 2.5: Ventajas de Git

Ofrece varias ventajas, como:

- La capacidad de trabajar de forma distribuida
- La gestión eficiente de ramas para desarrollar nuevas funcionalidades
- Corregir errores sin afectar la rama principal
- La posibilidad de colaborar de forma efectiva con otros desarrolladores.

## 2.6 ¿Dónde puedo utilizar Git?

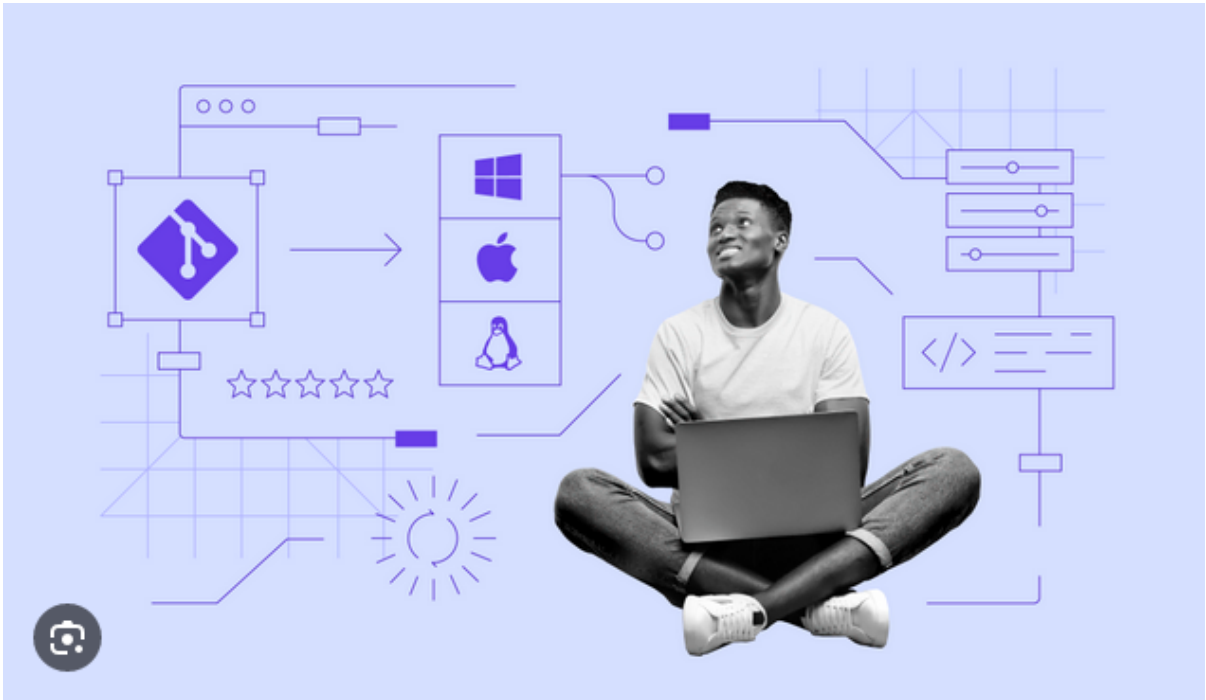


Figure 2.6: Git en Diferentes Sistemas Operativos

Puede ser utilizado en cualquier sistema operativo, incluyendo Windows, macOS y Linux. Además, es compatible con una amplia variedad de plataformas de alojamiento de repositorios, siendo GitHub una de las más populares.

## 2.7 Pasos Básicos

### 💡 Tip

Es recomendable tomar en cuenta una herramienta para la edición de código, como Visual Studio Code, Sublime Text o Atom, para trabajar con Git y GitHub de manera eficiente.

## 2.8 Instalación de Visual Studio Code

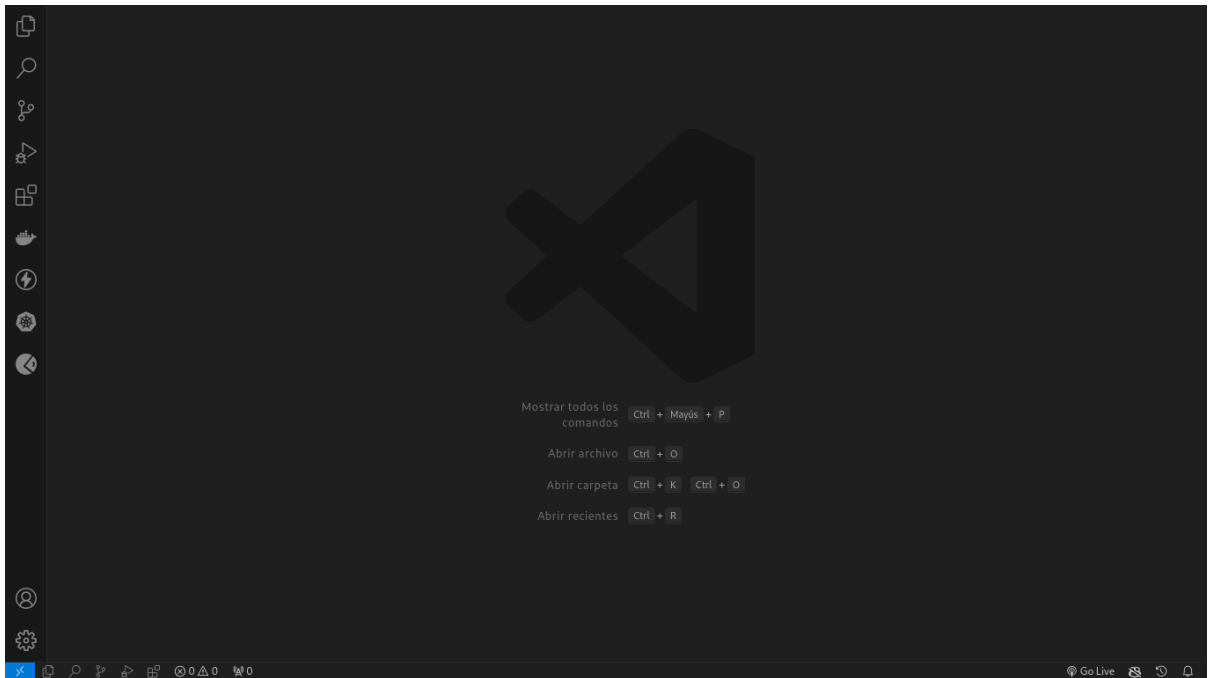


Figure 2.7: Visual Studio Code

Si aún no tienes Visual Studio Code instalado, puedes descargarlo desde <https://code.visualstudio.com/download>. Es una herramienta gratuita y de código abierto que proporciona una interfaz amigable para trabajar con Git y GitHub.

A continuación se presentan los pasos básicos para utilizar Git y GitHub en un proyecto de software.

## 2.8.1 Descarga e Instalación de Git

The image shows the Git website homepage. At the top left is the Git logo with the tagline "--distributed-even-if-your-workflow-isnt". A search bar is located at the top right. The main content area features a description of Git as a free and open source distributed version control system, highlighting its speed and efficiency. It also mentions that Git is easy to learn and has a tiny footprint with lightning fast performance, outperforming tools like Subversion, CVS, Perforce, and ClearCase. A diagram of a distributed version control system is shown to the right. Below this, there are four main sections: 'About' (advantages compared to other systems), 'Documentation' (command reference, Pro Git book, videos), 'Downloads' (GUI clients and binary releases), and 'Community' (bug reporting, mailing list, chat). A central monitor displays the latest source release '2.44.0' and a 'Download for Linux' button. At the bottom, there are links for 'Linux GUIs', 'Mac Build', 'Tarballs', and 'Source Code'. A small book cover for 'Pro Git' is also visible.

Figure 2.8: Git

1. Visita el sitio web oficial de Git en <https://git-scm.com/downloads>.
2. Descarga el instalador adecuado para tu sistema operativo y sigue las instrucciones de instalación.

## 2.8.2 Configuración



Figure 2.9: Configuración de Git

Una vez instalado Git, es necesario configurar tu nombre de usuario y dirección de correo electrónico. Esto se puede hacer mediante los siguientes comandos:

```
git config --global user.name "Tu Nombre"  
git config --global user.email "tu@email.com"
```

## 2.8.3 Creación de un Repositorio “helloWorld” en Flutter

**Paso 1:** Crear un Nuevo Proyecto en Flutter

Abre una terminal y ejecuta el siguiente comando para crear un nuevo proyecto Flutter:

```
flutter create hello_world
```

```
statick@fedora ~/workspaces/practicas/flutter/hello_world flutter create hello_world
Creating project hello_world...
Resolving dependencies in 'hello_world'...
Downloading packages...
Got dependencies in 'hello_world'.
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 hello_world
$ flutter run

Your application code is in hello_world/lib/main.dart.

The configured version of Java detected may conflict with the Gradle version in your new Flutter app.

[RECOMMENDED] If so, to keep the default Gradle version 8.3, make
sure to download a compatible Java version
(Java 17 ≤ compatible Java version < Java 21).
You may configure this compatible Java version by running:
`flutter config --jdk-dir=<JDK_DIRECTORY>`
Note that this is a global configuration for Flutter.

Alternatively, to continue using your configured Java version, update the Gradle
version specified in the following file to a compatible Gradle version (compatible Gradle version range: 8.4 - 8.7):
/home/statick/workspaces/practicas/flutter/hello_world/hello_world/android/gradle/wrapper/gradle-wrapper.properties

You may also update the Gradle version used by running
`./gradlew wrapper --gradle-version=<COMPATIBLE_GRADLE_VERSION>`.

See
https://docs.gradle.org/current/userguide/compatibility.html#java for details
on compatible Java/Gradle versions, and see
https://docs.gradle.org/current/userguide/gradle_wrapper.html#sec:upgrading_wrapper
for more details on using the Gradle Wrapper command to update the Gradle version
used.
```

Navega a la carpeta del proyecto:

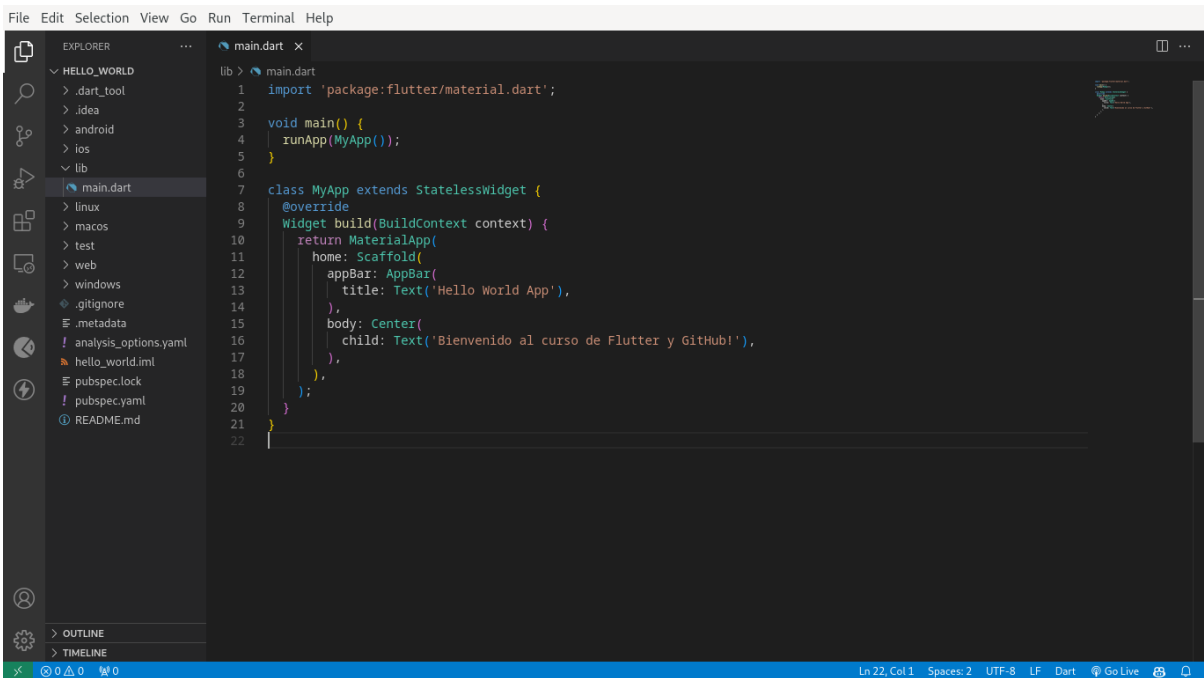
```
cd hello_world
```

Abre el proyecto en Visual Studio Code:

```
code .
```

## Paso 2: Modificar el Código de la Aplicación

En el archivo **lib/main.dart**, modifica el contenido para incluir el siguiente código:



```
File Edit Selection View Go Run Terminal Help
EXPLORER
HELLO_WORLD
  .dart_tool
  .idea
  android
  ios
  lib
    main.dart
  linux
  macos
  test
  web
  windows
  .gitignore
  .metadata
  ! analysis_options.yaml
  hello_world.iml
  pubspec.lock
  ! pubspec.yaml
  README.md
  OUTLINE
  TIMELINE
0 0 0 W0 Ln 22, Col 1 Spaces: 2 UTF-8 LF Dart Go Live
```

```
lib > main.dart
1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(MyApp());
5 }
6
7 class MyApp extends StatelessWidget {
8   @override
9   Widget build(BuildContext context) {
10    return MaterialApp(
11      home: Scaffold(
12        appBar: AppBar(
13          title: Text('Hello World App'),
14        ),
15        body: Center(
16          child: Text('Bienvenido al curso de Flutter y GitHub!'),
17        ),
18      ),
19    );
20 }
21
22
```

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false, // Oculta el banner de depuración
      home: Scaffold(
        appBar: AppBar(
          title: Text('Hello World App'),
        ),
        body: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.min,
            children: [
              Text('Bienvenido al curso de Flutter y GitHub!'),
              SizedBox(height: 20), // Espacio entre el texto y el botón
              ElevatedButton(
                onPressed: () {
                  print('¡Gracias por usar la app!');
                },
                child: Text('Presiona aquí'),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```



```
    ),  
  ),  
),  
);  
}  
}
```

Guarda los cambios.

### Paso 3: Inicializar un Repositorio Git

Inicializa un repositorio Git en la carpeta del proyecto:

```
git init
```

Añade los archivos al área de preparación:

```
git add .
```

Realiza un commit de los cambios con un mensaje descriptivo:

```
git commit -m "Inicializar proyecto hello_world en Flutter"
```

## 2.8.4 Comandos Básicos de Git

- **git init:** Inicializa un nuevo repositorio Git.
- **git add:** Añade archivos al área de preparación.
- **git commit -m ““:** Realiza un commit de los cambios con un mensaje descriptivo.
- **git push:** Sube los cambios al repositorio remoto.
- **git pull:** Descarga cambios del repositorio remoto.
- **git branch:** Lista las ramas disponibles.
- **git checkout:** Cambia a una rama específica.
- **git merge:** Fusiona una rama con la rama actual.
- **git reset:** Descarta los cambios en un archivo.
- **git diff:** Muestra las diferencias entre versiones.

## 2.8.5 Estados en Git

- **Local:** Cambios realizados en tu máquina antes de hacer un commit.
- **Staging:** Cambios añadidos al área de preparación con git add.
- **Commit:** Cambios confirmados en tu repositorio local con git commit.
- **Server:** Cambios subidos al repositorio remoto con git push.
- **Tutorial:** Moviendo Cambios entre Estados en Git

## 2.8.6 Modificar Archivos en el Proyecto

Abre `lib/main.dart` en Visual Studio Code.

Cambia el texto en el `Text widget` de “Bienvenido” a “¡Bienvenido a Flutter y GitHub!”.

Guarda los cambios y luego mueve los cambios al área de preparación:

```
git add lib/main.dart
```

Realiza un commit de los cambios:

```
git commit -m "Actualizar mensaje de bienvenida"
```

## 2.8.7 Creación y Fusión de Ramas

Crear una Nueva Rama:

```
git branch feature
```

## 2.8.8 Implementar Funcionalidades en la Rama:

Cambia a la nueva rama:

```
git checkout feature
```

Añade un botón interactivo en el archivo `lib/main.dart`:

```
ElevatedButton(  
  onPressed: () {  
    print('¡Gracias por usar la app!');  
  },  
  child: Text('Presiona aquí'),  
),
```

Guarda los cambios y realiza un commit:

```
git add lib/main.dart  
git commit -m "Añadir botón interactivo"
```

## 2.8.9 Fusionar Ramas con la Rama Principal:

```
git checkout main  
git merge feature
```

### 2.8.10 Revertir Cambios en un Archivo

Si necesitas revertir cambios, puedes usar:

```
git reset lib/main.dart
```

## 3 Conclusión

Este tutorial te ha mostrado cómo usar Git con un proyecto Flutter, administrar cambios, y trabajar con ramas.

¡Ahora estás listo para colaborar y desarrollar de manera eficiente!

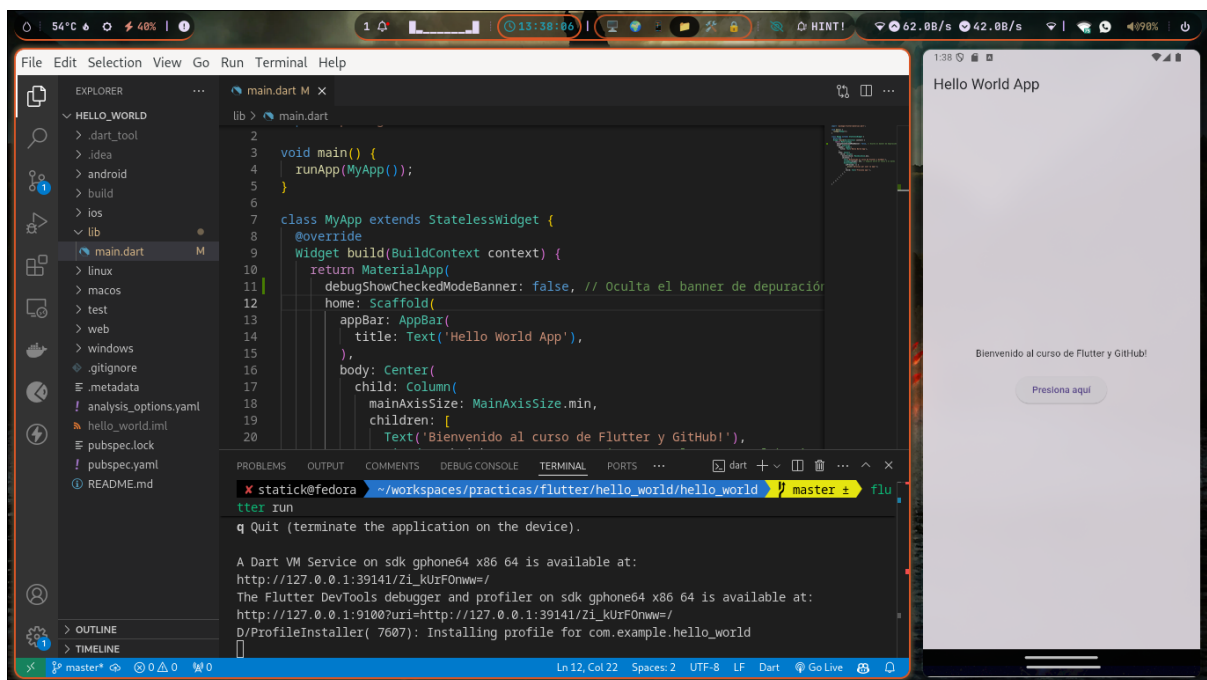
## 4 Asignación: “Hello World!” en Flutter

### 4.1 La Asignación

Modifica el archivo `main.dart` para que la app muestre un botón que, al presionarlo, cambie el mensaje en pantalla a “¡Hola Mundo!”.

### 4.2 Probar la Aplicación

Ejecuta tu proyecto con:



```
flutter run
```

¡Mucha suerte!

## 5 GitHub Classroom



Figure 5.1: Github Classroom

GitHub Classroom es una herramienta poderosa que facilita la gestión de tareas y asignaciones en GitHub, especialmente diseñada para entornos educativos.

### 5.1 ¿Qué es GitHub Classroom?



Figure 5.2: Github Classroom Windows

GitHub Classroom es una extensión de GitHub que permite a los profesores crear y gestionar asignaciones utilizando repositorios de GitHub. Proporciona una forma organizada y eficiente de distribuir tareas a los estudiantes, recopilar y revisar su trabajo, y proporcionar retroalimentación.

#### 5.1.1 Funcionalidades Principales

**Creación de Asignaciones:** Los profesores pueden crear tareas y asignaciones directamente desde GitHub Classroom, proporcionando instrucciones detalladas y estableciendo

criterios de evaluación.

**Distribución Automatizada:** Una vez que se crea una asignación, GitHub Classroom genera automáticamente repositorios privados para cada estudiante o equipo, basándose en una plantilla predefinida.

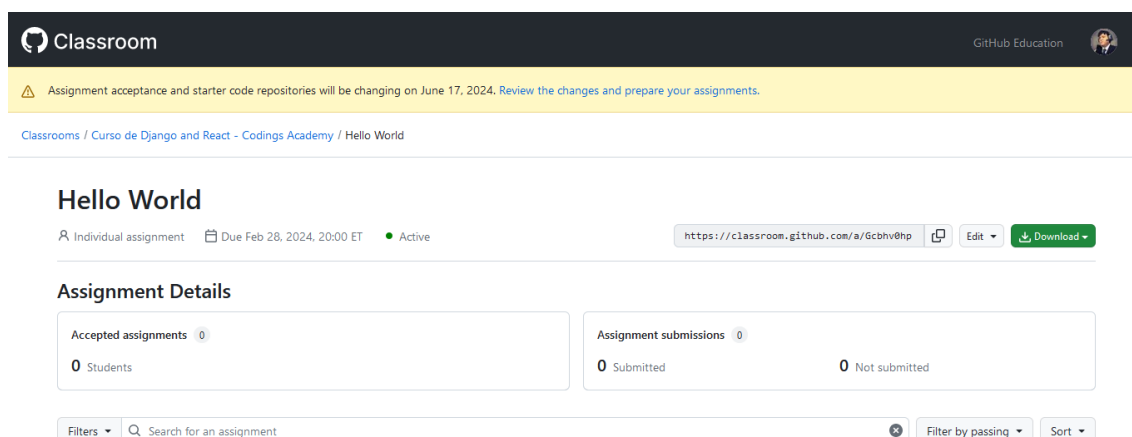
**Seguimiento de Progreso:** Los profesores pueden realizar un seguimiento del progreso de los estudiantes y revisar sus contribuciones a través de solicitudes de extracción (pull requests) y comentarios en el código.

**Revisión y Retroalimentación:** Los estudiantes envían sus trabajos a través de solicitudes de extracción, lo que permite a los profesores revisar y proporcionar retroalimentación específica sobre su código.

## 5.2 Ejemplo Práctico

### 5.2.1 Creación de una Asignación en GitHub Classroom

**Iniciar Sesión:** Ingresa a GitHub Classroom con tu cuenta de GitHub y selecciona la opción para crear una nueva asignación.



**Definir la Tarea:** Proporciona instrucciones claras y detalladas sobre la tarea, incluyendo cualquier código base o recursos necesarios. Establece los criterios de evaluación para guiar a los estudiantes.

- Assignment basics
- Starter code and environment
- Grading and feedback

### Assignment basics

Assignment title \*  
Hello World

Student assignment repositories will have the prefix: `hello-world`

Deadline  
02 / 28 / 2024, 08:00 PM  
(Optional) If left blank, there will be no deadline. Date format: YYYY-MM-DD HH:MM a

This is a cutoff date  
If selected, the student will lose write access to their repository after the date is reached.

Assignment status: Active | Individual or group assignment: Individual assignment  
Assignment type cannot be changed after assignment creation.

Repository visibility

**Configurar la Plantilla:** Selecciona una plantilla de repositorio existente o crea una nueva plantilla que servirá como base para los repositorios de los estudiantes.

**Add a template repository to give students starter code**  
Your assignment will be created with empty student repositories if you don't add starter code. Changes to starter code after students have accepted the assignment will not retroactively change existing student repositories.  
Note: All starter code must use a [template repository](#). Your starter code repository must be either in the same organization as this classroom or a public repository if elsewhere. [Learn about transferring your repositories](#).

Find a GitHub repository  
  
education/autograding-example-python  
GitHub Classroom autograding example repo with Python and Pytest

**GitHub Codespaces**  
Your organization is eligible for GitHub Codespaces. Enable Codespaces in students' repositories to give them a one-click experience for getting started coding, running, and collaborating on their code. [Enable it in Classroom settings](#).

**Supported editor**  
Changing the online IDE after an assignment has been created is not possible.  
✓ Don't use an online IDE

**Grading and feedback**

**Add autograding tests**  
Autograding tests help provide feedback for students immediately upon submission using [GitHub Actions](#). Add a test to enable autograding.

Hello world test

+ Add test

**Distribuir la Asignación:** Una vez configurada la asignación, comparte el enlace generado con tus estudiantes para que puedan acceder a sus repositorios privados.



Curso de Django and React - Codings Academy

## Accept the assignment — Hello World

Once you accept this assignment, you will be granted access to the `hello-world-statick88` repository in the [Coding-Academy-ec](#) organization on GitHub.

Accept this assignment

### 5.3 Trabajo de los Estudiantes

**Aceptar la Asignación:** Los estudiantes reciben el enlace de la asignación y aceptan la tarea, lo que les permite crear un repositorio privado basado en la plantilla proporcionada.



You accepted the assignment, **Hello World**. We're configuring your repository now. This may take a few minutes to complete. Refresh this page to see updates.

Your assignment is due by **Feb 28, 2024, 20:00 ET**

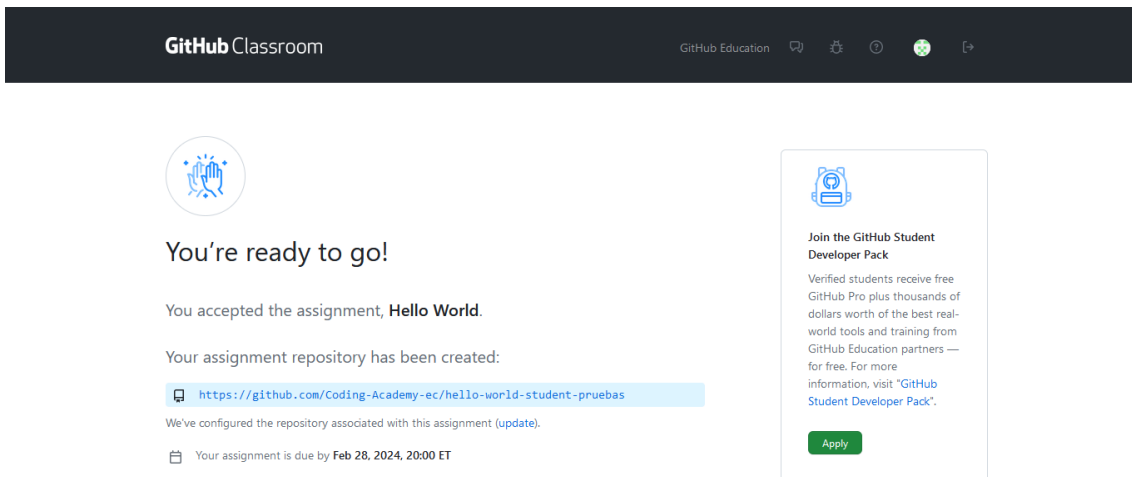


#### Join the GitHub Student Developer Pack

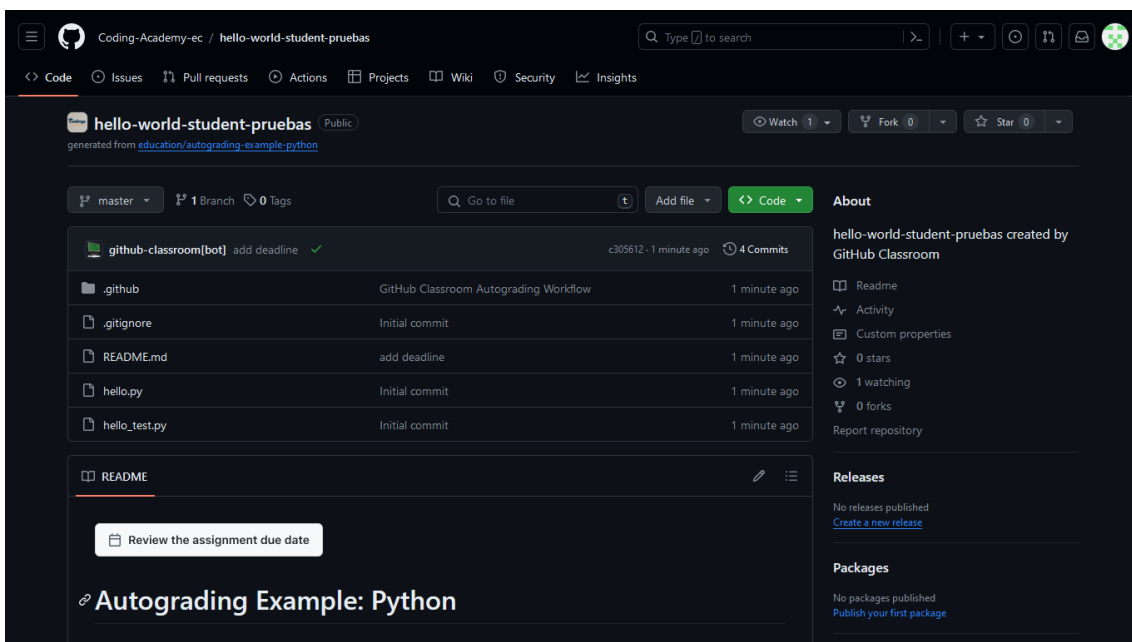
Verified students receive free GitHub Pro plus thousands of dollars worth of the best real-world tools and training from GitHub Education partners — for free. For more information, visit ["GitHub Student Developer Pack"](#).

Apply

**Actualizar el Navegador:** Los estudiantes actualizan su navegador para ver el nuevo repositorio creado en su cuenta de GitHub.



**Clonar el Repositorio:** Los estudiantes clonan el repositorio asignado en su computadora local utilizando el enlace proporcionado.



Utilizar el comando git clone: Aplique el comando git clone para clonar el repositorio en su computadora local.

```
git clone <enlace-del-repositorio>
```

```
Desktop :: pwsh
~\Desktop git clone https://github.com/Coding-Academy-ec/hello-world-student-pruebas.git
Cloning into 'hello-world-student-pruebas' ...
remote: Enumerating objects: 19, done.
remote: Counting objects: 100% (19/19), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 19 (delta 4), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (19/19), 4.69 KiB | 1.17 MiB/s, done.
Resolving deltas: 100% (4/4), done.
```

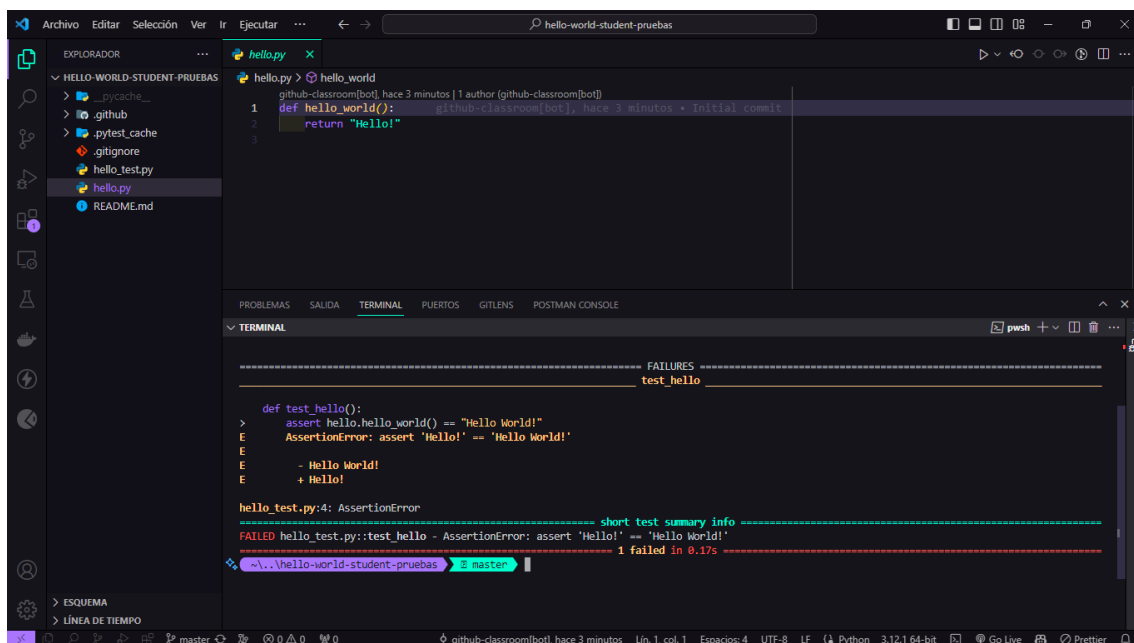
**Desarrollar la Tarea:** Los estudiantes trabajan en la tarea, realizando los cambios necesarios y realizando commits de manera regular para mantener un historial de su trabajo.

### 💡 Tip

Puedes probar el test incorporado con el comando `pytest` en la terminal, para verificar que el código cumple con los requerimientos

## pytest

Una vez desarrollado el código de acuerdo a la asignación en local deberían pasar el o los test



```
hello.py
1 def hello_world():
2     return "Hello!"
3

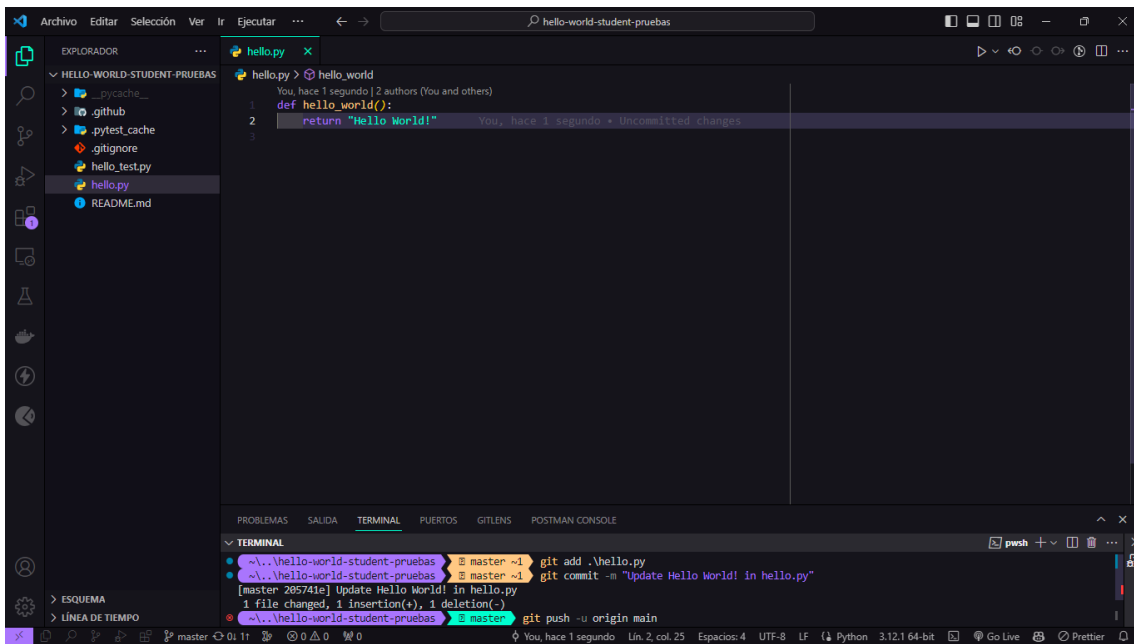
Terminal
----- FAILURES -----
test_hello

def test_hello():
> assert hello.hello_world() == "Hello World!"
E       AssertionError: assert 'Hello!' == 'Hello World!'
E
E       - Hello World!
E       + Hello!

hello_test.py:4: AssertionError
----- short test summary info -----
FAILED hello_test.py::test_hello - AssertionError: assert 'Hello!' == 'Hello World!'
1 failed in 0.17s

~\..\hello-world-student-pruebas X master
```

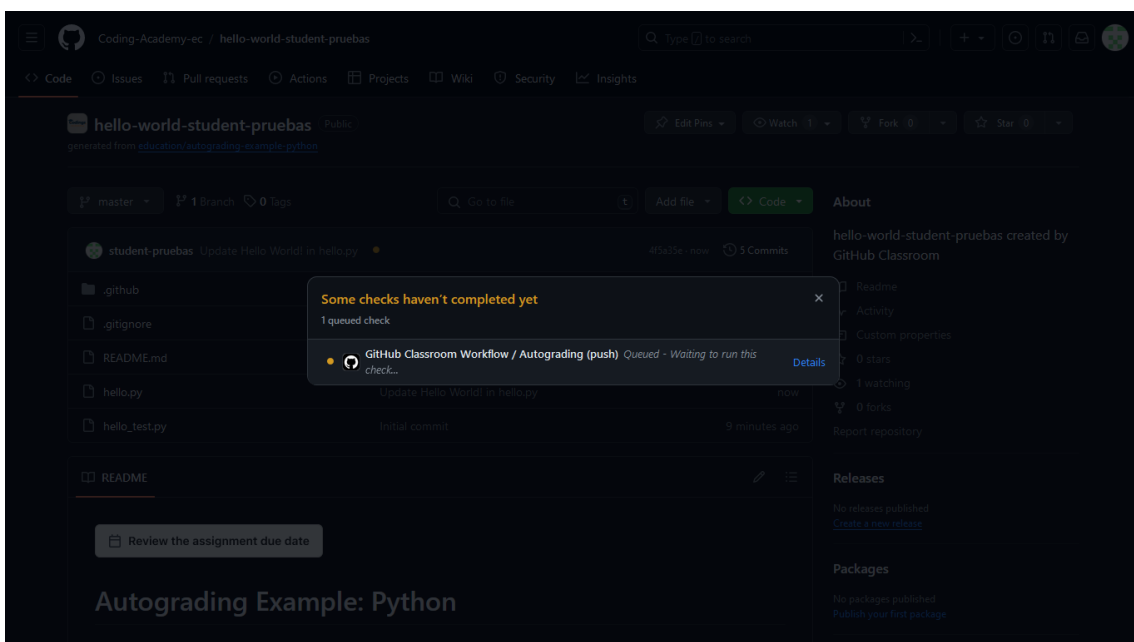
**Enviar la Solicitud de Extracción:** Una vez completada la tarea, los estudiantes envían una solicitud de extracción desde su rama hacia la rama principal del repositorio, solicitando la revisión del profesor.



Una vez realizado el push se envía al repositorio principal y se ejecutan los test en Github

### Tip

Se recomienda hacer las pruebas en local antes de enviar los cambios al repositorio en Github



Este Action lo que hace es evaluar los cambios realizados

```

Autograding
succeeded now in 6s
Beta Give Feedback Search logs

Run education/autograding@v1 2s
13 Requirement already satisfied: tomli>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from pytest) (2.0.1)
14 Collecting iniconfig
15   Downloading iniconfig-2.0.0-py3-none-any.whl (5.9 kB)
16 Collecting exceptiongroup>=1.0.0rc8
17   Downloading exceptiongroup-1.2.0-py3-none-any.whl (16 kB)
18 Collecting pluggy<2.0,>=1.3.0
19   Downloading pluggy-1.4.0-py3-none-any.whl (20 kB)
20 Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from pytest) (23.2)
21 Installing collected packages: pluggy, iniconfig, exceptiongroup, pytest
22 Successfully installed exceptiongroup-1.2.0 iniconfig-2.0.0 pluggy-1.4.0 pytest-8.0.2
23 WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package
  manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
24
25 ===== test session starts =====
26 platform linux -- Python 3.10.12, pytest-8.0.2, pluggy-1.4.0
27 rootdir: /home/runner/work/hello-world-student-pruebas/hello-world-student-pruebas
28 collected 1 item
29
30 hello_test.py . [100%]
31
32 ===== 1 passed in 0.01s =====
33
34  Hello world test
35
36
37 ::***::
38
39 All tests passed
40
41 🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟
42
43 Points 100/100
  
```

**Tip**

Se recomienda hacer las pruebas en local antes de enviar los cambios al repositorio en Github

**Revisión y Retroalimentación:** Los profesores revisan las solicitudes de extracción, proporcionan comentarios sobre el código y evalúan el trabajo de los estudiantes según los criterios establecidos.

**Hello World**

Individual assignment Due Feb 28, 2024, 20:00 ET Active <https://classroom.github.com/a/6cbvh0hp> Edit Download

**Assignment Details**

- Accepted assignments 1: 1 Students
- Assignment submissions 1: 1 Submitted, 0 Not submitted
- Passed students 1: 1/1 Passed

Filters Search for an assignment Filter by passing Sort

**Total students**

	student-pruebas Submitted	latest commit 2 minutes ago ✓	1 commit	100/100	Repository
--	---------------------------	-------------------------------	----------	---------	------------

© 2024 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

 Tip

**GitHub Classroom** ofrece una manera eficiente y organizada de administrar tareas y asignaciones en entornos educativos, fomentando la colaboración, el aprendizaje y la retroalimentación efectiva entre profesores y estudiantes.

## **Part II**

# **Unidad 1: Introducción e Instalaciones Necesarias**

## **6 Instalaciones Necesarias.**



# 7 Introducción a Dart

## 7.1 ¿Qué es Dart?

**Dart** es un **lenguaje de programación** creado por Google, según su concepción es un lenguaje del lado del cliente optimizado para aplicaciones.

### 7.1.1 Características.

- Optimizado para UI.
- Permite **Hot Reload**.
- Rápido en todas las arquitecturas ARM & x64.
- Futures, Async-Await, código no-blocking, Streams al abrirlo de la caja.
- Toda aplicación de Dart ejecuta una función inicial llamada **main()**
- Su sintaxis es muy similar a la de **C#**, **Java**, **TypeScript**.
- Su curva de aprendizaje es muy baja.

## 7.2 ¿Qué es Flutter?

Flutter es el Framework del lenguaje de programación Dart. Esta pensado en el desarrollo multiplataforma, actualmente este framework puede exportar a movil, web, aplicaciones de escritorio y envevidos (en pantallas de vehiculos).

## 8 Hola Mundo en Dart.

Un punto de entrada en cualquier lenguaje de programación es la creación de un “**Hola Mundo**” que permite conocer la complejidad del lenguaje que se está aprendiendo.

Ejemplo:

```
void main(){
  print('Hello World');
}
```

Muy bien ahora veremos algunas cosas más llamativas

```
void main(){
  var myName = 'Diego Saavedra';
  print('Hello $myName');
}
```

Como podemos observar en el código anterior es necesario con la palabra reservada **var** declara un nombre a una variable, en este caso llamada **myName**, y a la misma es asignada con el valor **Diego Saavedra**, al final de toda sentencia en Dart es necesario terminar con un **;**.

sin embargo se sugiere declarar el tipo de variable que se utiliza, esto con el objetivo de normar de alguna forma el desarrollo de código.

```
void main(){
  String myName = 'Diego Saavedra';
  print('Hello $myName');
```