

DAM - Práctica 1

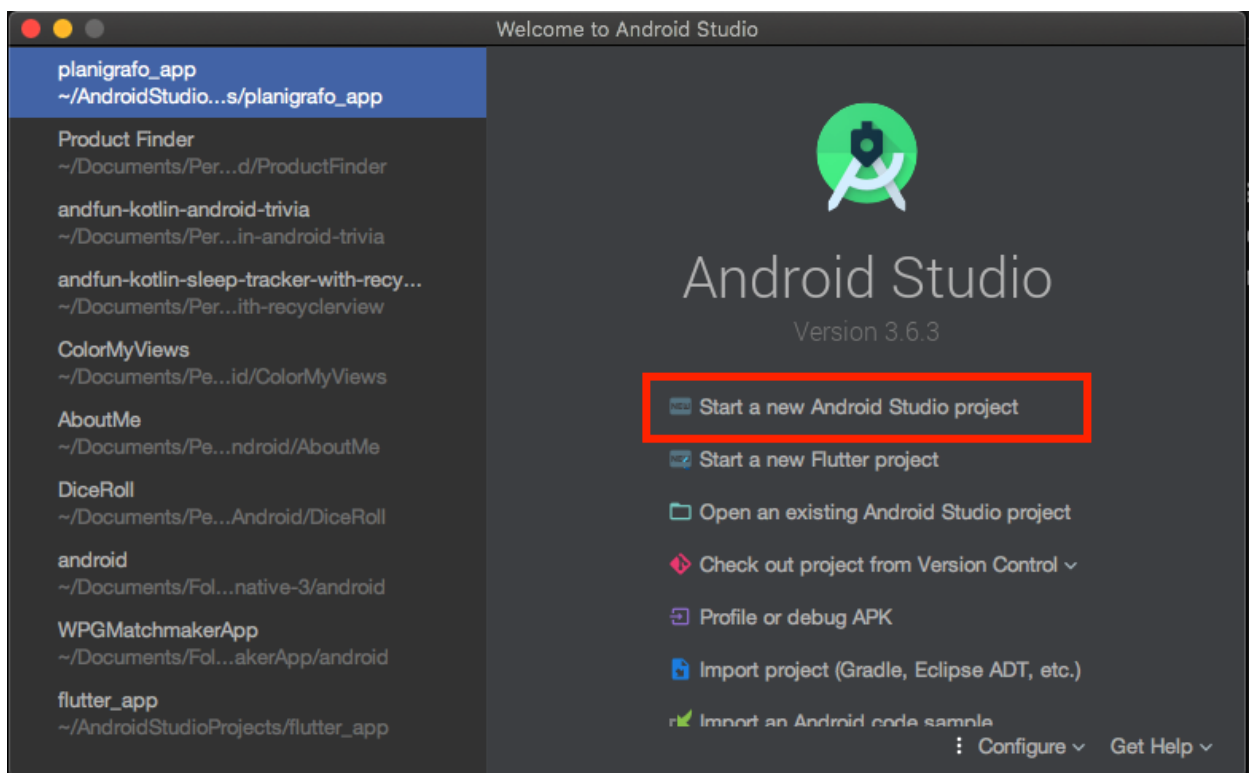
Objetivos

- Familiarizarse con Android Studio
- Aprender a configurar control de versiones con GIT
- Utilizar los distintos componentes gráfico disponibles
- Navegar entre pantallas
- Familiarizarnos con las librerías de compatibilidad de Android X

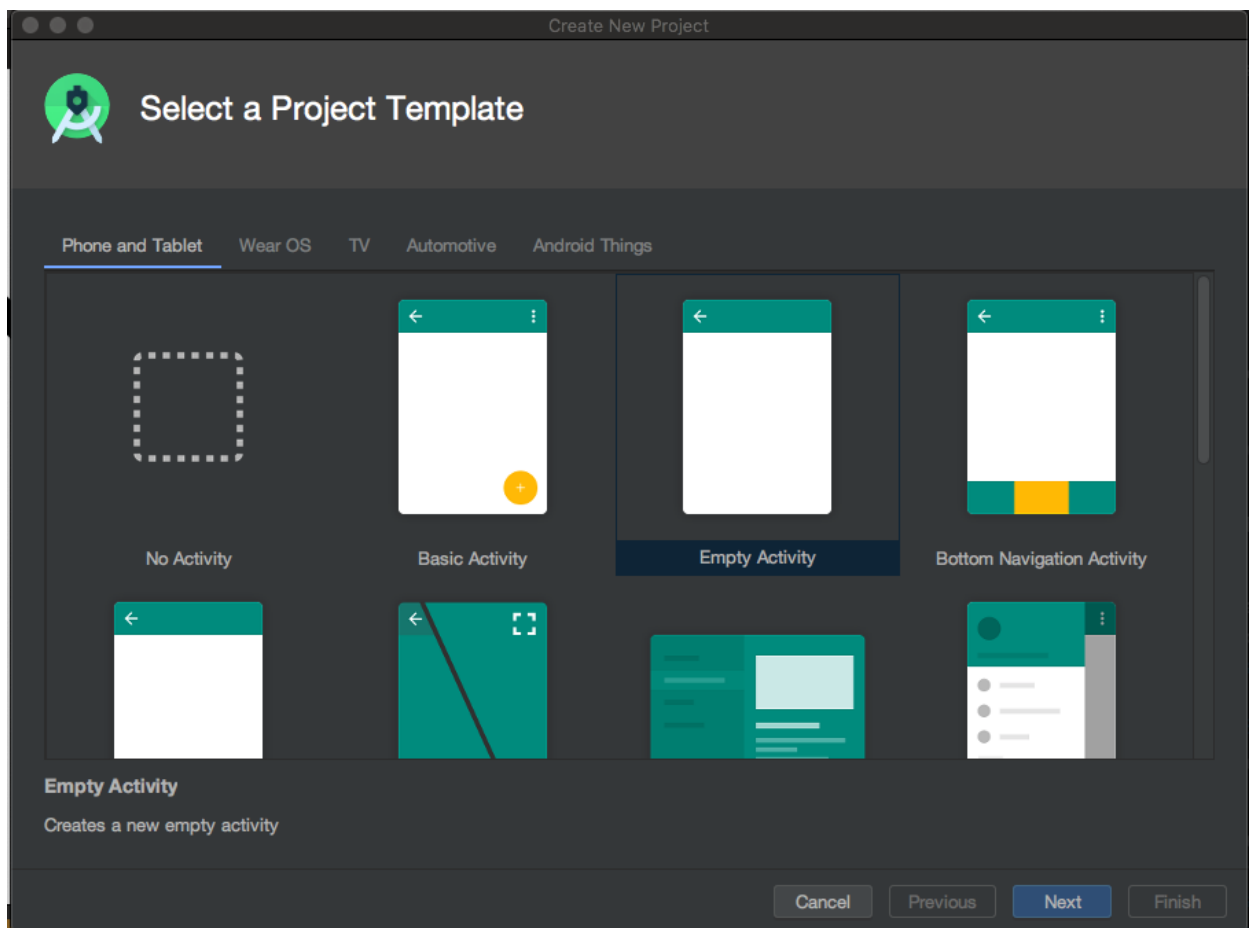
Tareas a desarrollar

1. Crear el proyecto

1. Abrir android studio y seleccionar la opción para comenzar un nuevo proyecto.



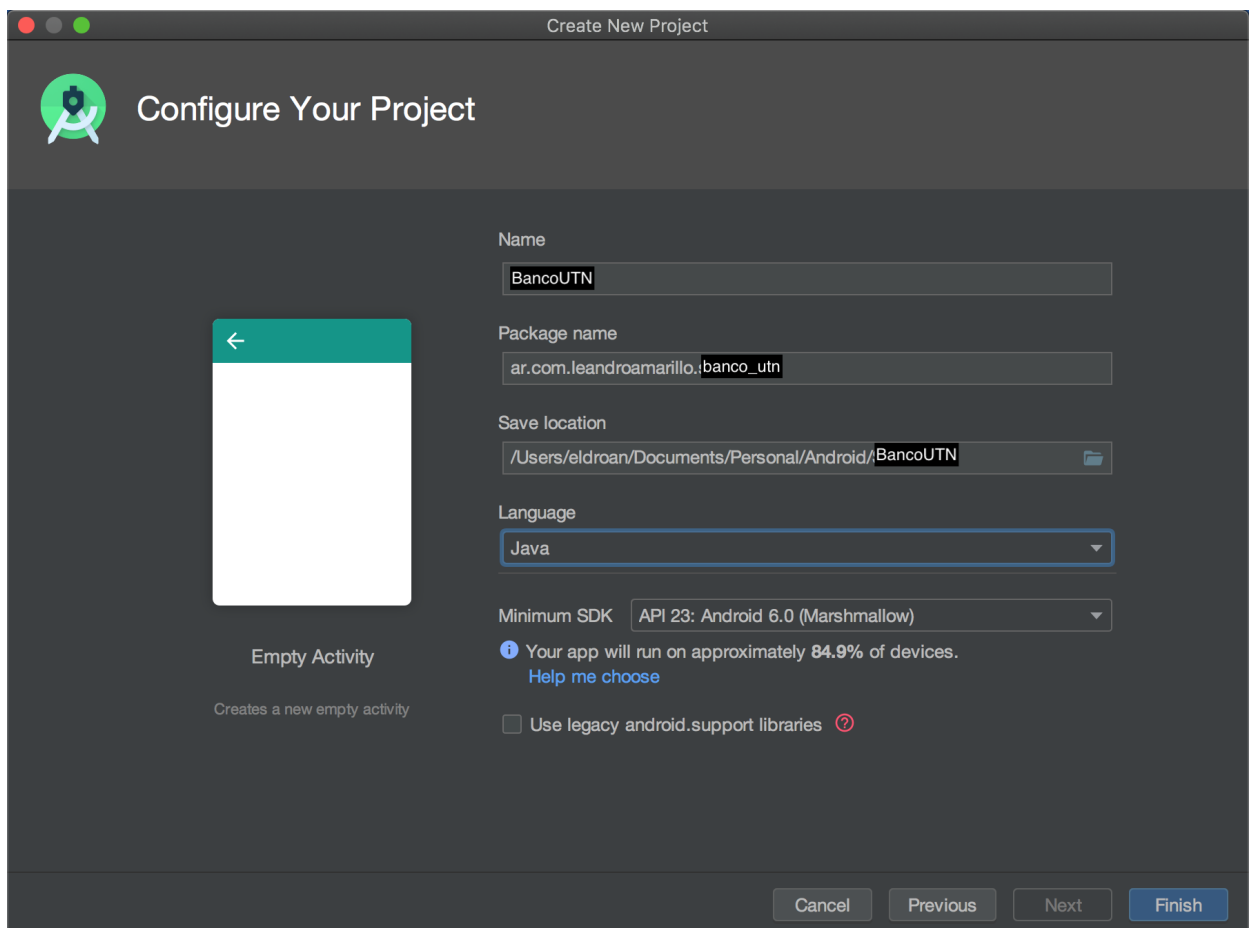
2. Seleccionar **Empty Activity** como template



3. En la sección de configuración de proyecto seleccione.

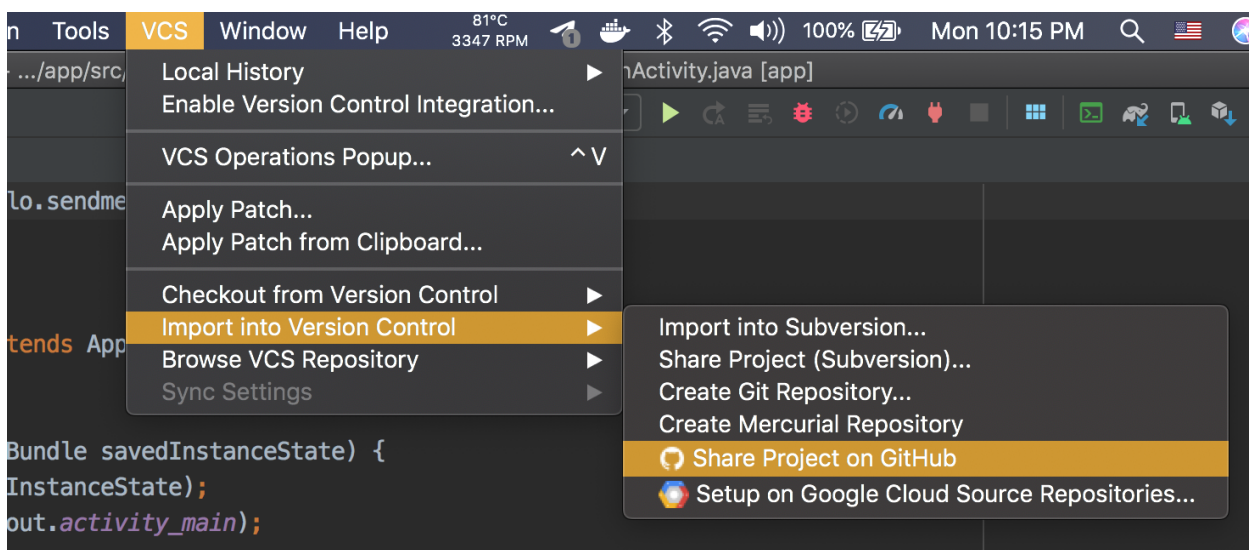
- Seleccione **JAVA** como lenguaje
- Nombre de la aplicación **BancoUTN**
- Nombre del paquete (Puede utilizar su nombre que identifique a su grupo)
- Seleccionar el Minimum SDK (Se recomienda API 23, como mínimo elija el API correspondiente a su dispositivo para poder ejecutar la aplicación)

Asegúrese de haber seleccionado Java como lenguaje ya que Android Studio en sus versiones más recientes pre-selecciona Kotlin.

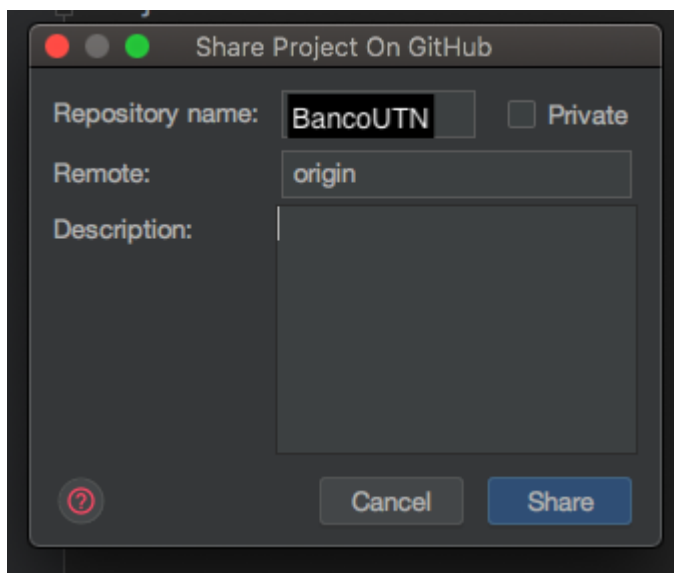


2. Configurar GIT

1. Es necesario tener una cuenta de Github, en caso de no tenerla puede registra una en <https://github.com>
2. Desde Android Studio, en la barra superior de tareas seleccione la opcion VCS → Import into Version Control → Share Project on Github



3. Asignamos el nombre de nuestro repositorio en Github bajo **Repository Name** asegurandonos que la opcion **Private** se encuentre des-marcada y presionamos **Share**.



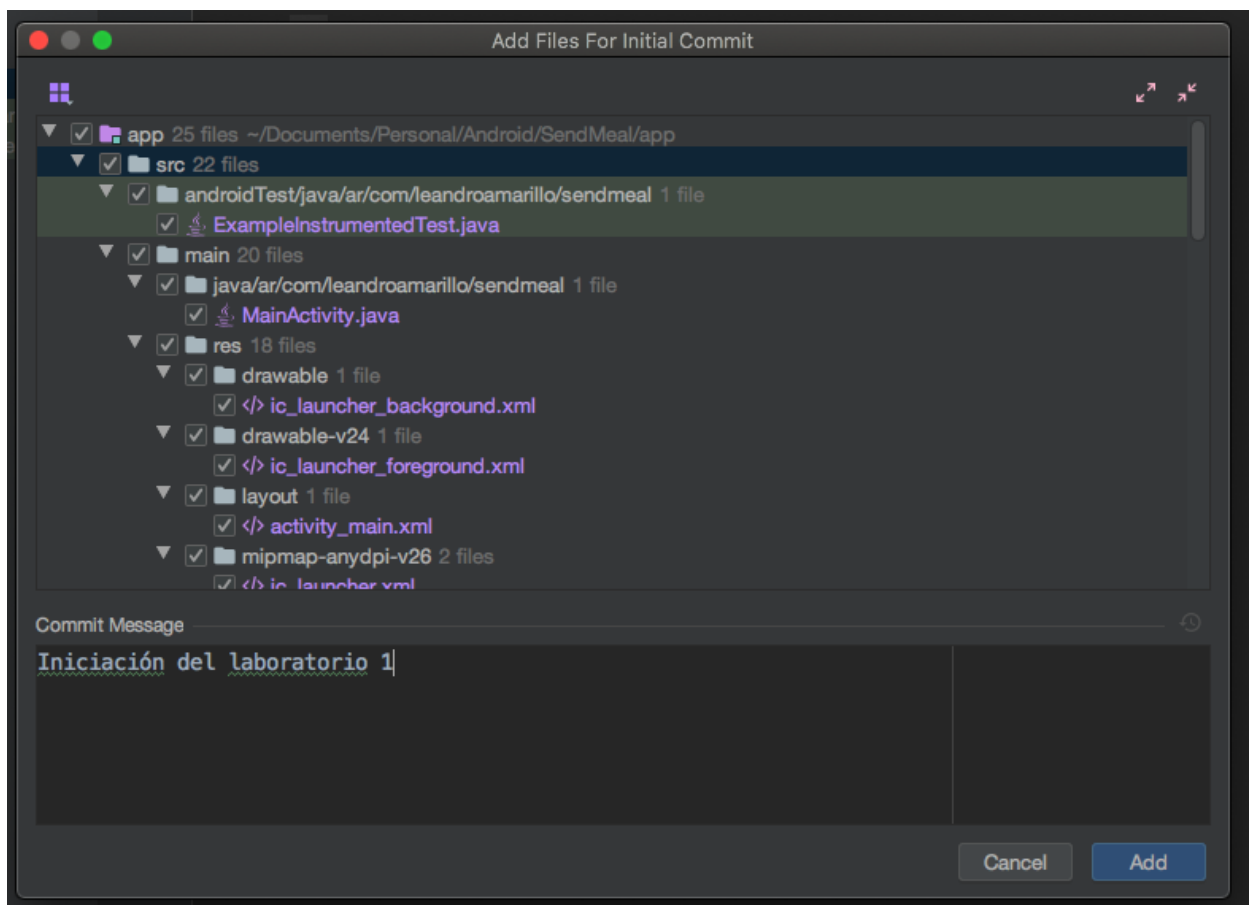
Es preferible que el repositorio sea publico para que podamos revisarlos facilmente desde la cátedra. Para usos personales puede utilizar repositorios privados, los cuales solo pueden ser visualizados con la autorización del creador.

El campo **Remote** es el nombre con el cual git identificará localmente a este nuevo repositorio remoto en Github, es recomendado dejar el nombre **origin** ya que es la convención para los casos con un único repositorio remoto como el nuestro.

4. Para finalizar con la creación de nuestro repositorio debemos inicializarlo con un primer commit, la siguiente ventana de Android Studio nos presenta la opcion de agregar todos los archivos al seguimiento de GIT. Colocamos como mensaje para nuestro primer commit

Iniciación de la practica 1

Y luego presionamos **Add**

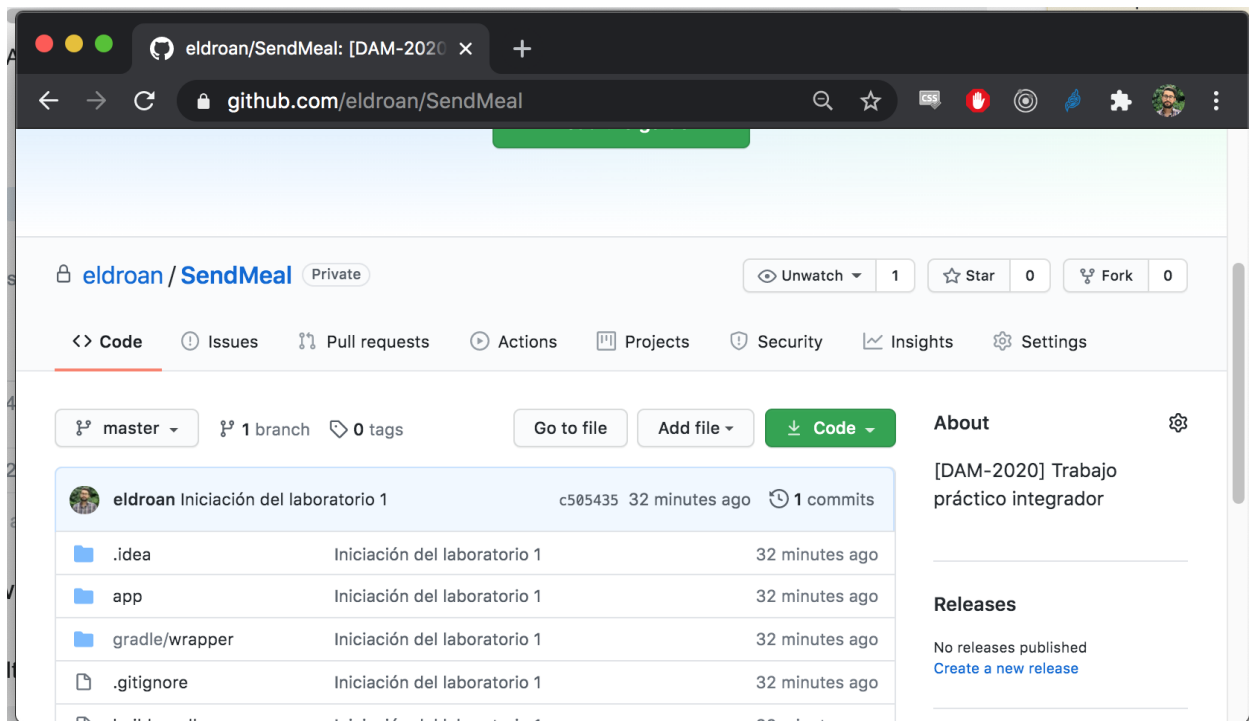


Recordemos que git solo realiza seguimiento de los archivos que le nosotros le indicamos o **tracked**, los que no, aparecerán como **untracked**.

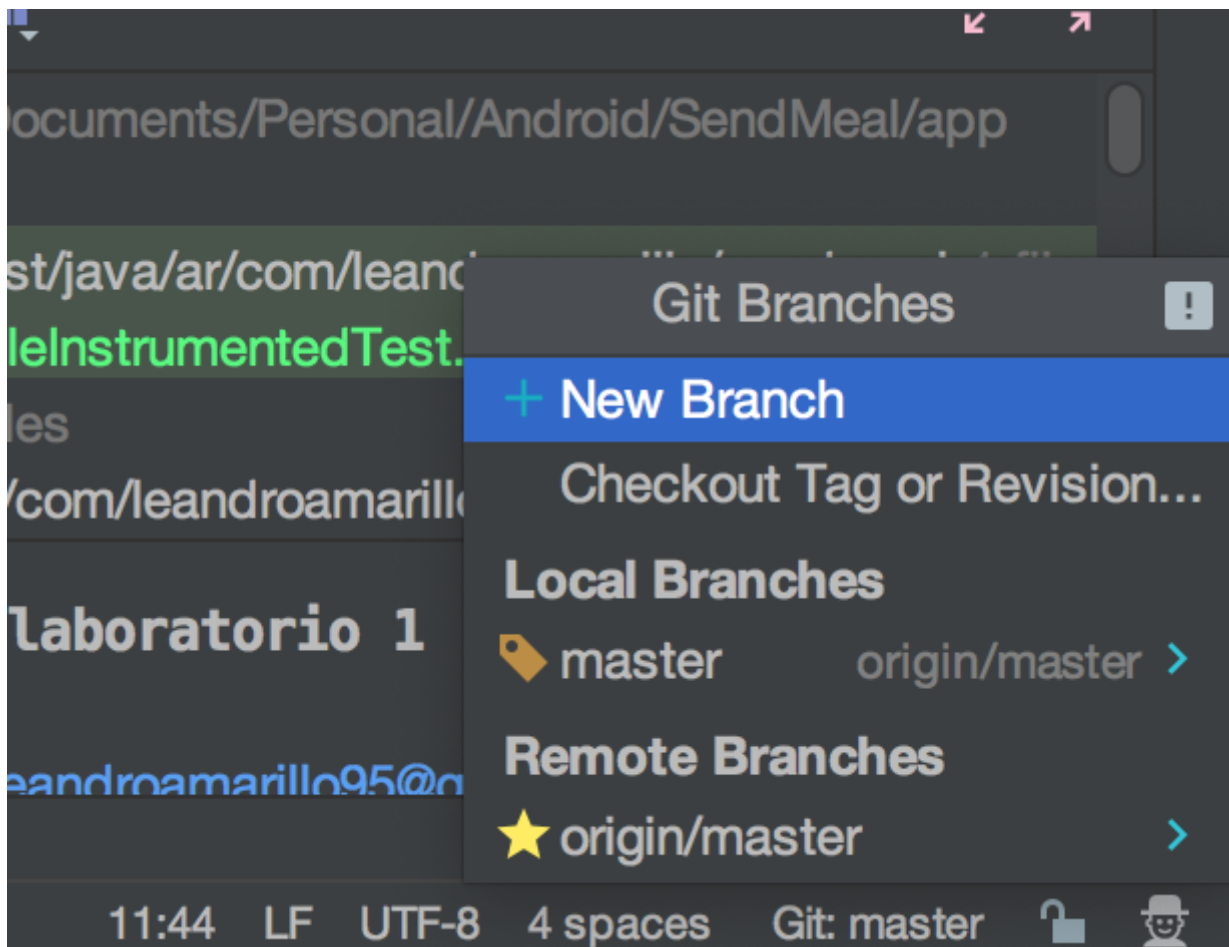
Apretar el botón **Add** en Android Studio para inicializar el repositorio realiza 2 acciones, primero realiza un **commit** a nuestro repositorio local y luego realiza un **push** al repositorio remoto en Github.

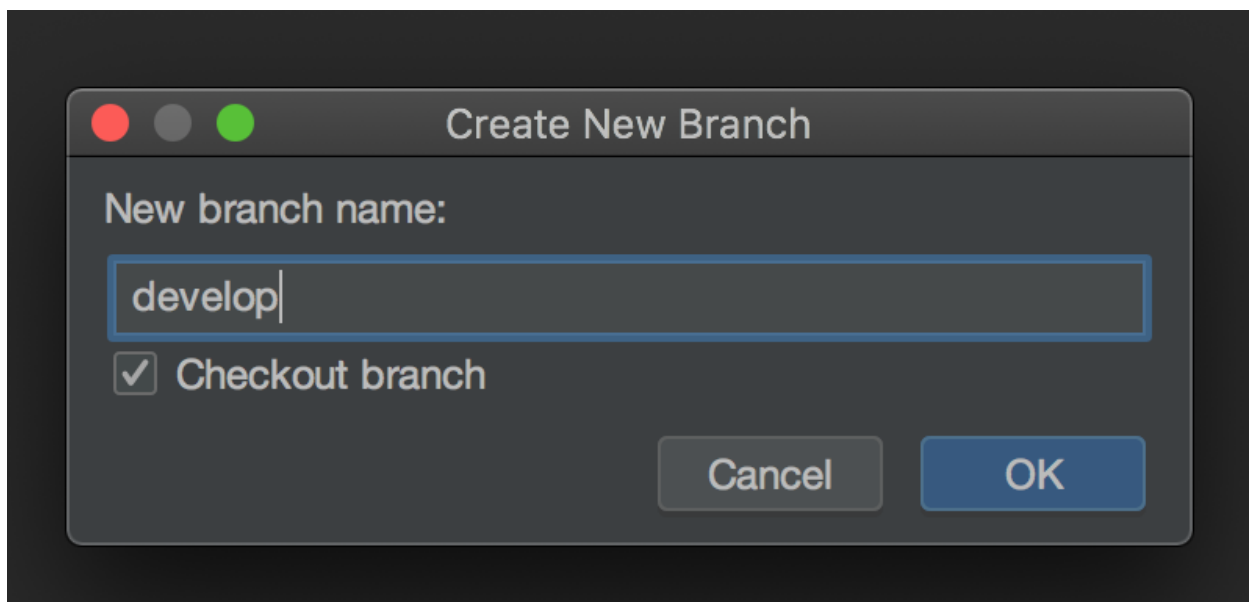
5. Verificar información en Github. La url del repositorio suele seguir la forma

https://github.com/{USUARIO}/{NOMBRE_REPOSITORIO}

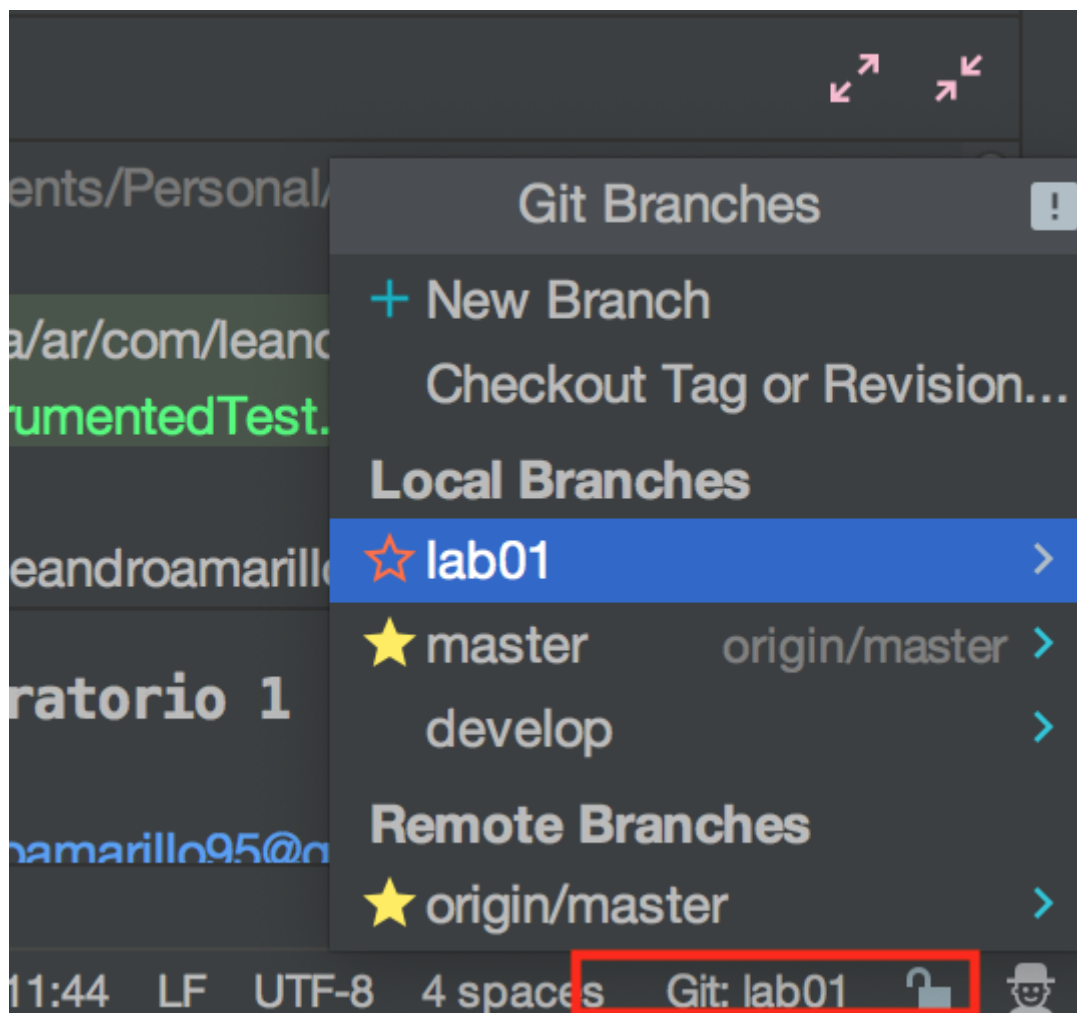


6. Crear la rama/branch `develop`, haciendo click con el IDE (Android Studio) en la barra inferior del margen derecho.





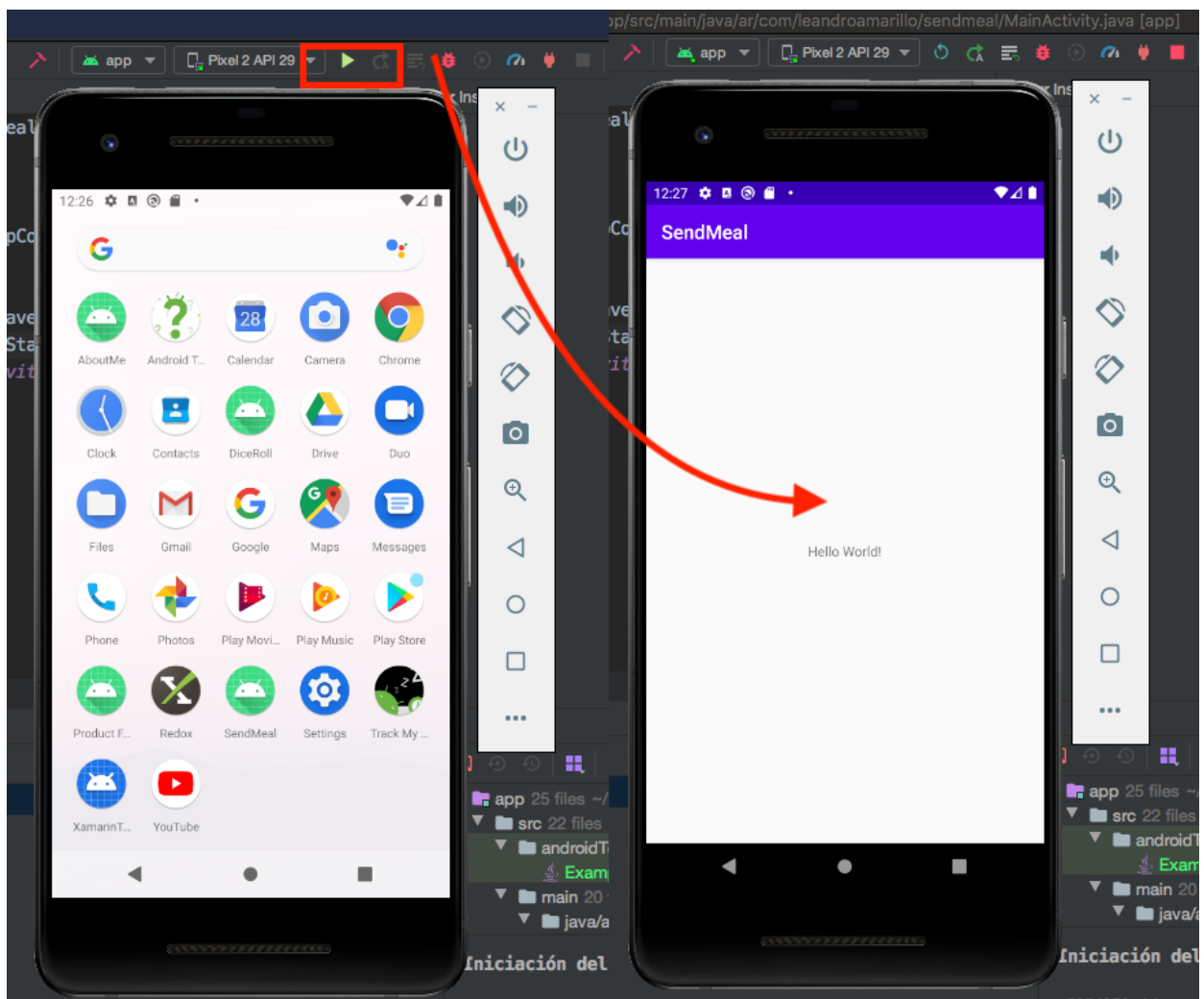
7. Crear una nueva rama con el nombre `lab01`.



Asegurarse de que se encuentre trabajando en la rama lab01 para la realizacion del laboratorio 1.

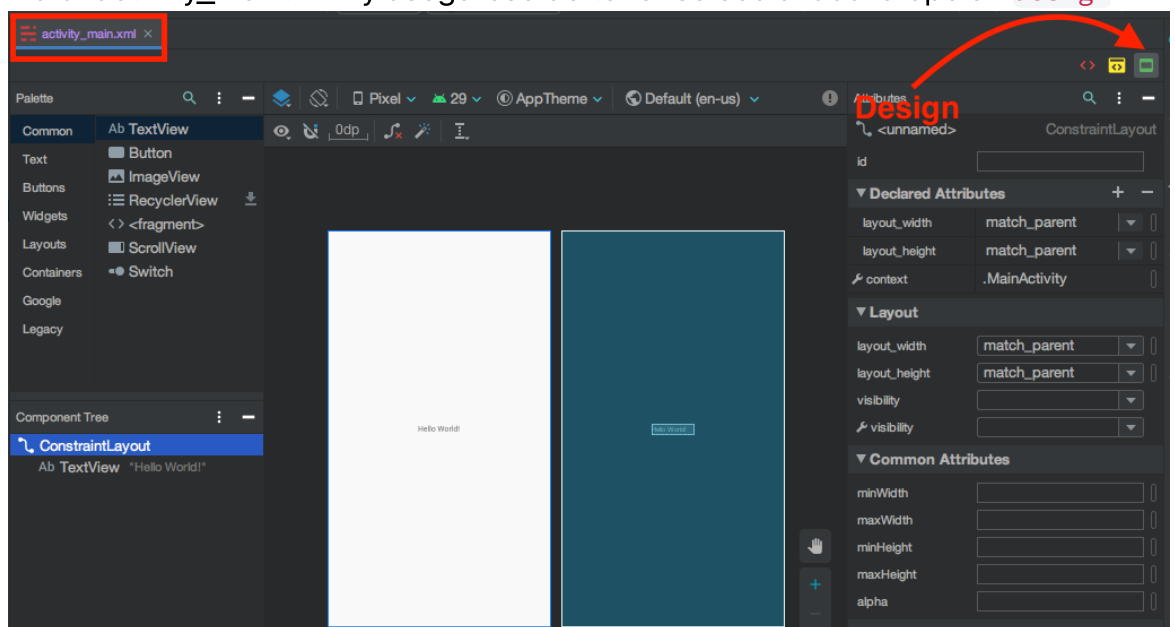
3. Configurar UI - (LinearLayout)

1. Para asegurarse que todo se encuentre funcionando correctamente probemos correr la aplicación generada presionando el boton verde con una flecha. Una vez que cargue deberíamos ver el mensaje 'Hello World' en pantalla

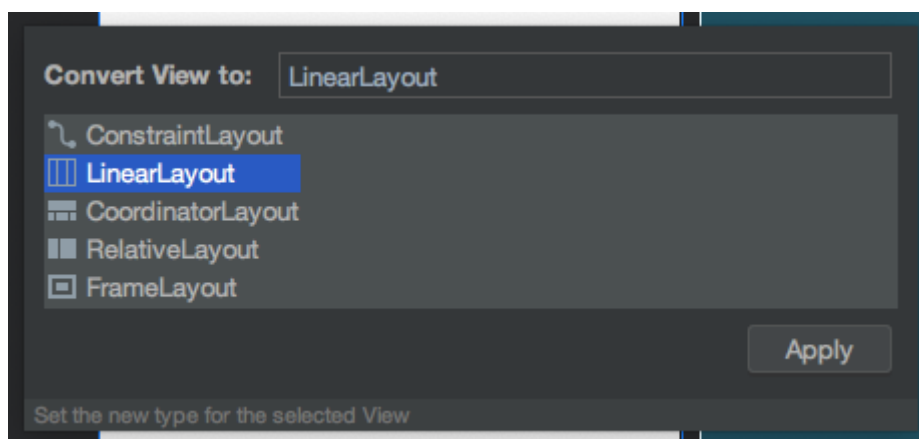
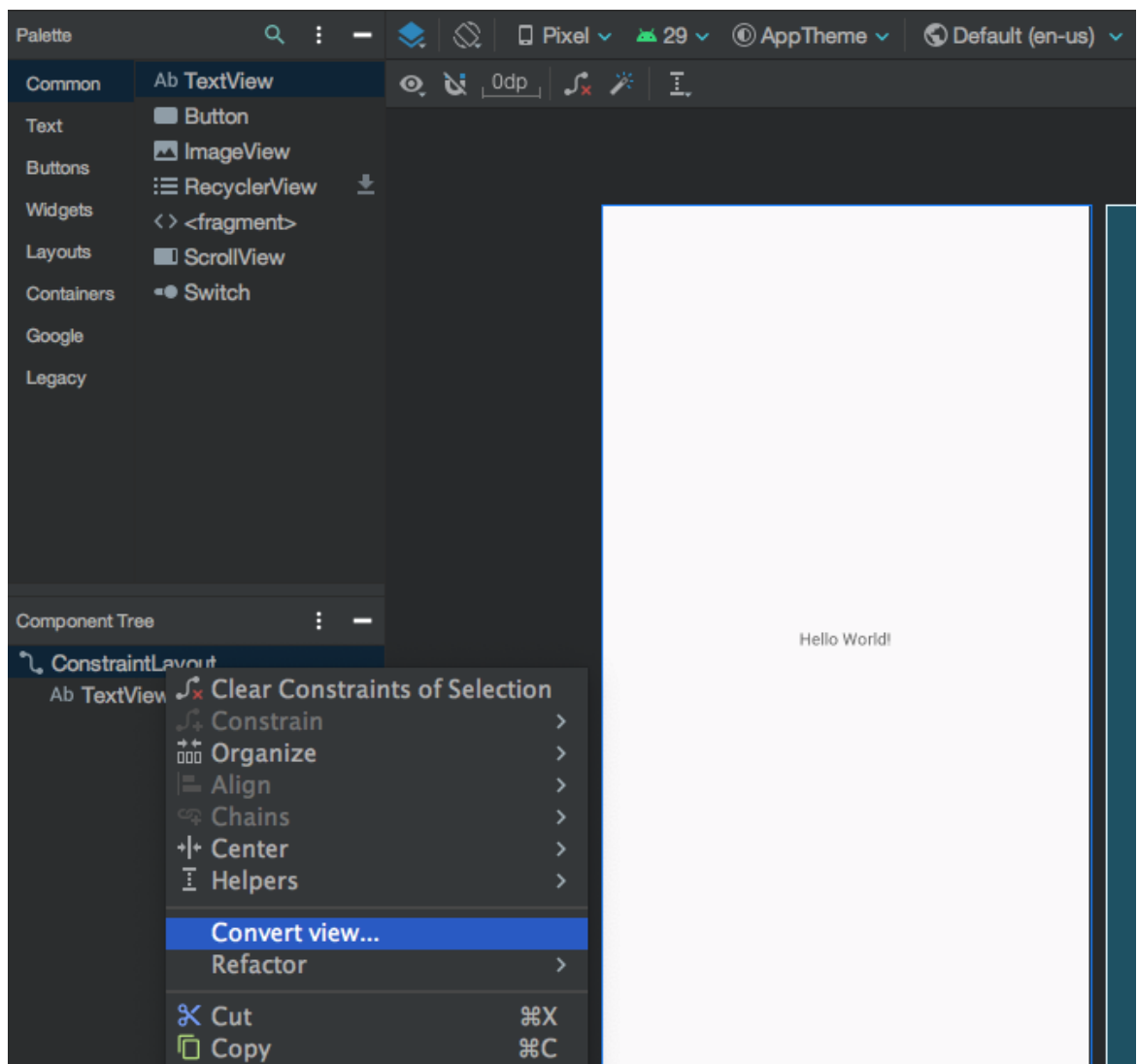


2. El mensaje 'Hello World!' que podemos observar se encuentra en la carpeta 'res' → 'layout' → 'activity_main.xml'. Para la realización de esta práctica utilizaremos un **LinearLayout** de orientación vertical pero el archivo 'activity_main.xml' fue autogenerado utilizando un **ConstraintLayout** por lo que tendremos que cambiarlo

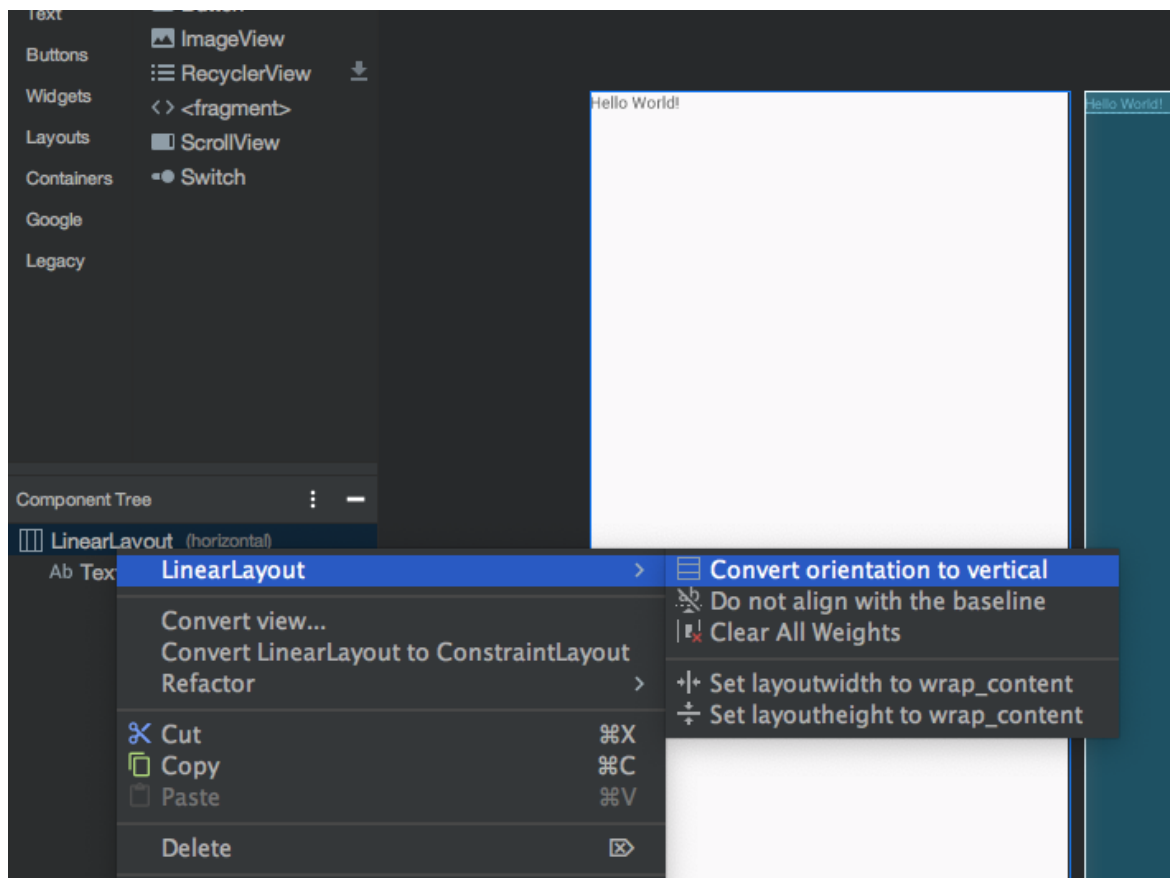
1. Abra 'activity_main.xml' y asegurese de tener seleccionada la opción **design**.



2. Seleccione la **ConstraintLayout** en el arbol de componentes y presione 'Convert view...' y seleccione **LinearLayout**



3. Por default el **LinearLayout** coloca los elementos de forma horizontal. Para cambiar este comportamiento realizamos click → LinearLayout → Convert orientation to vertical



4. UI Propuesta


Como nuestro primer proyecto se plantea desarrollar una especie de home banking llamada BancoUTN de la cual implementaremos dos pantallas para la funcionalidad de constituir plazos fijos

| Pantalla ConstituirPlazoFijo | Pantalla SimularPlazoFijo |
|------------------------------|---------------------------|
| | |

01:10

100%

Constituir plazo fijo



Banco UTN Santa Fe

Para constituir tu plazo fijo necesitamos que nos proveas algunos datos.

Nombre

Leandro

Apellido

Amarillo

Moneda

PESOS

SIMULAR


CONSTITUIR

Confirmación

01:10

100%

Simular plazo fijo



Banco UTN Santa Fe

Tasa Nominal Anual

69.5

Tasa Efectiva Anual

96.33

Capital a Invertir (\$)

100000

30 días

☐

Con renovación automática

Simulador Plazo Fijo en Pesos

Plazo: 30 días

Capital: 100000.0

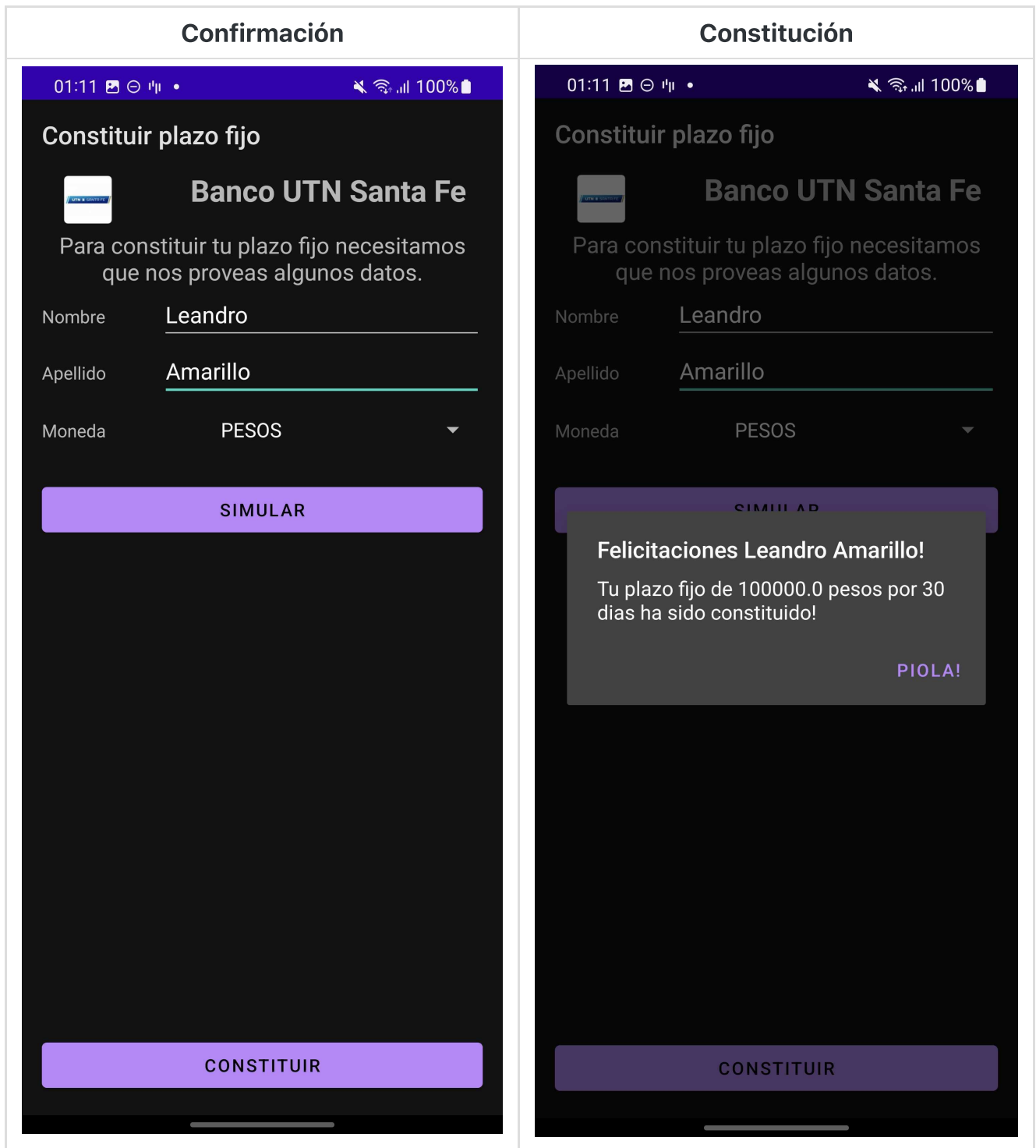
Intereses ganados: \$ 5791.6665

Monto total: \$ 105791.664

Monto total anual: \$ 169,500

CONFIRMAR

Constitución



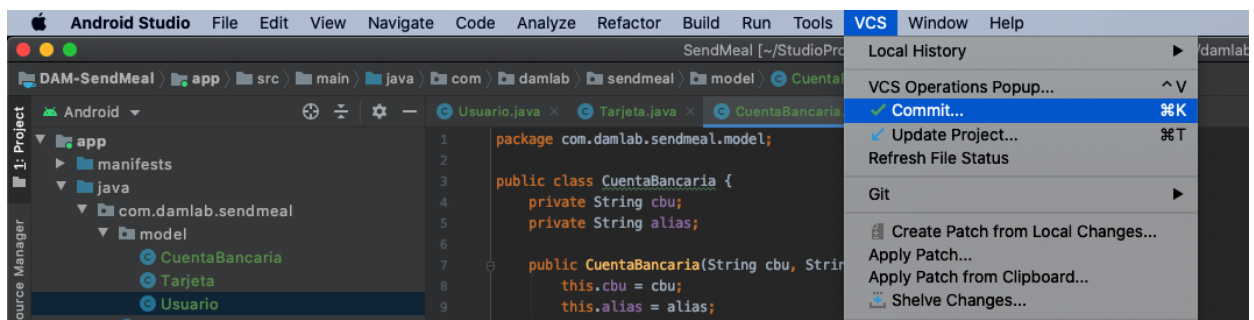
Consideraciones

- Generales:
 - Utilizar view binding en lugar de `findViewById` (revisar ppts de teoría).
 - La app no debe perder su estado al rotar el telefono.
 - Ambas pantallas tendran una imagen que contenga el logo de la facultad (https://upload.wikimedia.org/wikipedia/commons/6/67/UTN_logo.jpg) y el titulo **Banco UTN Santa Fe**
- Pantalla "ConstituirPlazoFijo":

- Agregar un text view con el mensaje "Para constituir tu plazo fijo necesitamos que nos proveas algunos datos."
 - Agregar campos de texto para ingresar un nombre y otro para ingresar un apellido.
 - Agregar un spinner para ingresar el tipo de moneda.
 - Agregar un botón "Simular" que redireccione a la pantalla de "SimularPlazoFijo".
 - Se deberá colocar un botón "Constituir" inicialmente en estado deshabilitado, el cual solo se activará luego de regresar de la pantalla "SimularPlazoFijo" habiendo confirmado el capital y los días del plazo fijo. Al presionar este botón deberá mostrarse un `AlertDialog` con un título "Felicitaciones {nombre} {apellido}!" y un mensaje "Tu plazo fijo de {capital} {moneda} por {dias} ha sido constituido!"
- Pantalla "SimularPlazoFijo":
 - Debe contar con un campo de texto para ingresar la **Tasa Nominal Anual** que solo permita inputs numéricos.
 - Debe contar con un campo de texto para ingresar la **Tasa Efectiva Anual** que solo permita inputs numéricos.
 - Debe contar con un campo de texto para ingresar el capital a invertir que solo permita inputs numéricos.
 - Para la cantidad de meses de inversion se utilizara un seekbar que permita como valor máximo y un label por debajo con la conversion a días (asumir que todos los meses tienen 30 dias).
 - Agregar un botón confirmar que al ser presionado regrese a la pantalla de "ConstituirPlazoFijo" notificandole el `capital` y los `dias` completados.
 - Agregar TextViews que permitan visualizar los siguientes valores:
 - "Plazo: {dias} días"
 - "Capital: {capital}"
 - "Intereses ganados: {interes}"
 - "Monto total: {interes + capital}"
 - "Monto total anual: {interesAnual + capital}"
 - Implementar un metodo `calcular()` que actualice los TextViews mencionados en el punto anterior y que sea llamado automaticamente al modificarse alguno de los valores de la pantalla. En caso de valores invalidos se deberá desabilitar el botón de confirmar.

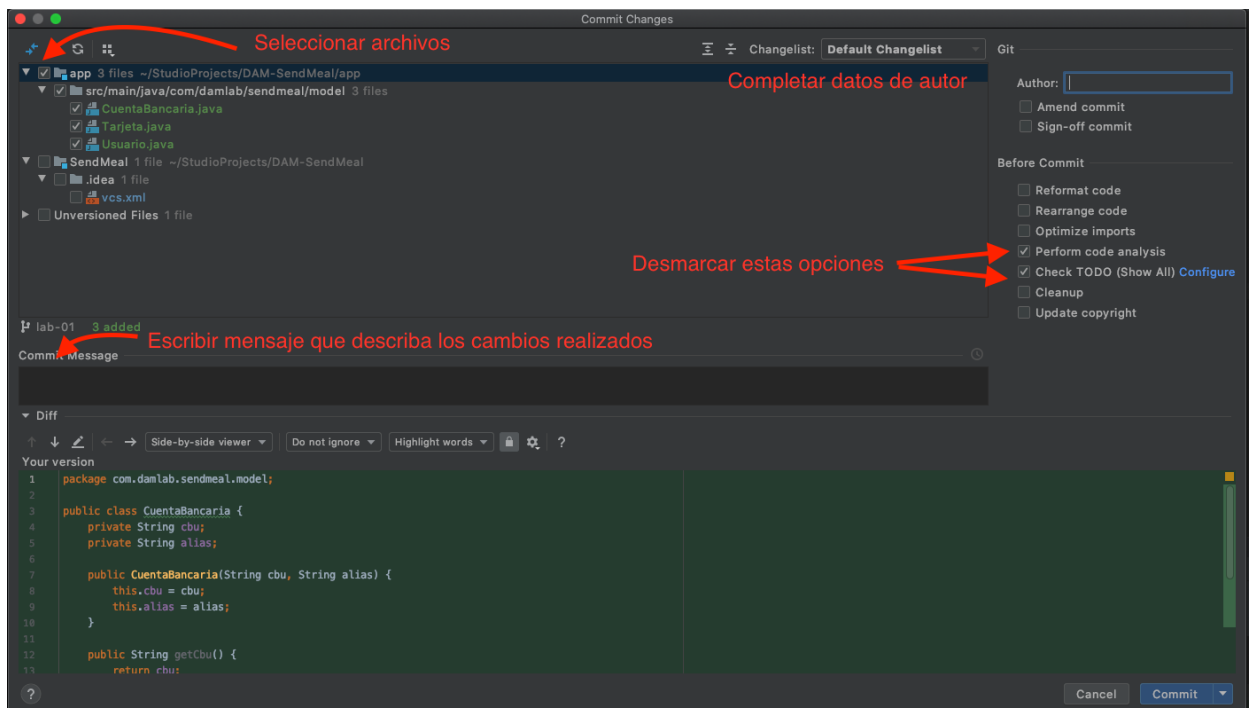
5. Persistir en git y compartir en github

1. Verificar que la app se encuentre funcionando al presionar el boton `Play`
2. Desde android studio presionar `VCS` → `Commit` o el atajo de teclado `CTRL + K` (Win y Linux) o `CMD + K` (Mac)

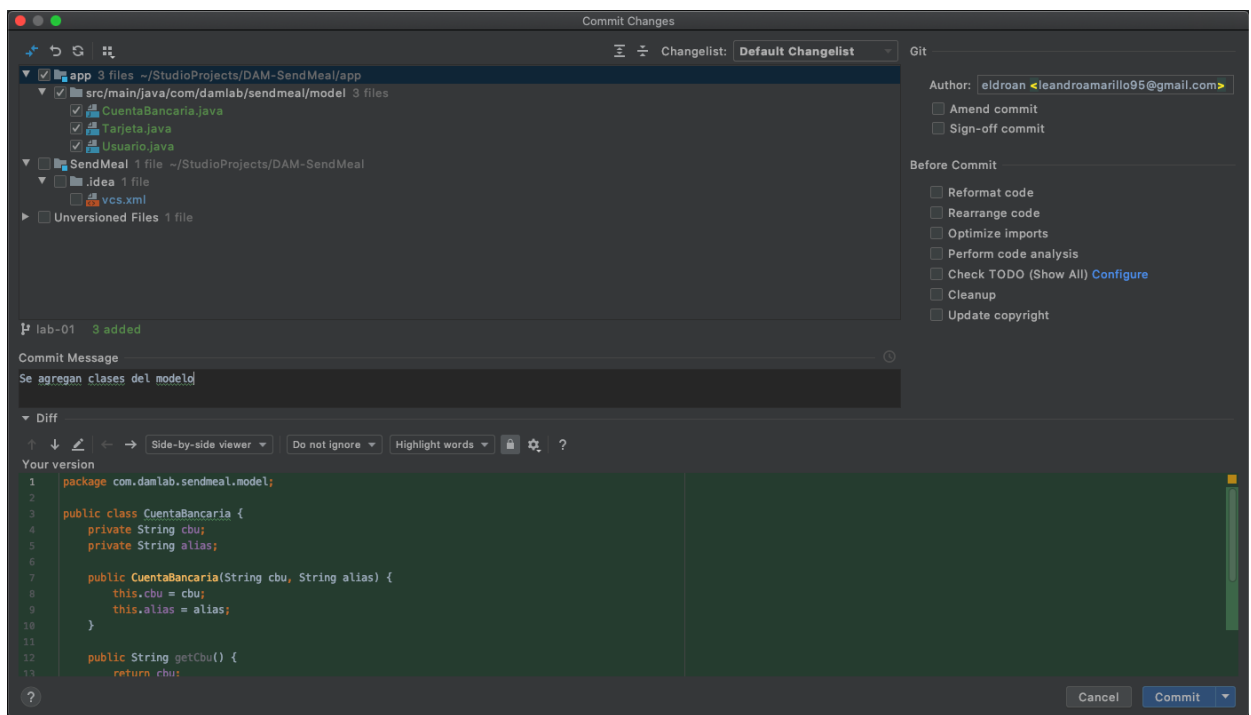


3. En la ventana seleccione todos los archivos que desea persistir en el commit y asegurarse de:

- Escribir un mensaje de commit significativo de los cambios que realizo
- Completar el nombre del autor en formato `userDeGit`
`<correo@electronico.com>`
- Destildar `Perform code analysis` y `Check TODO` (No es estrictamente necesario, si no los destildamos nos mostrara un warning con cosas que el ide considera que podrian mejorarse o mensajes marcados como "TO-DO")

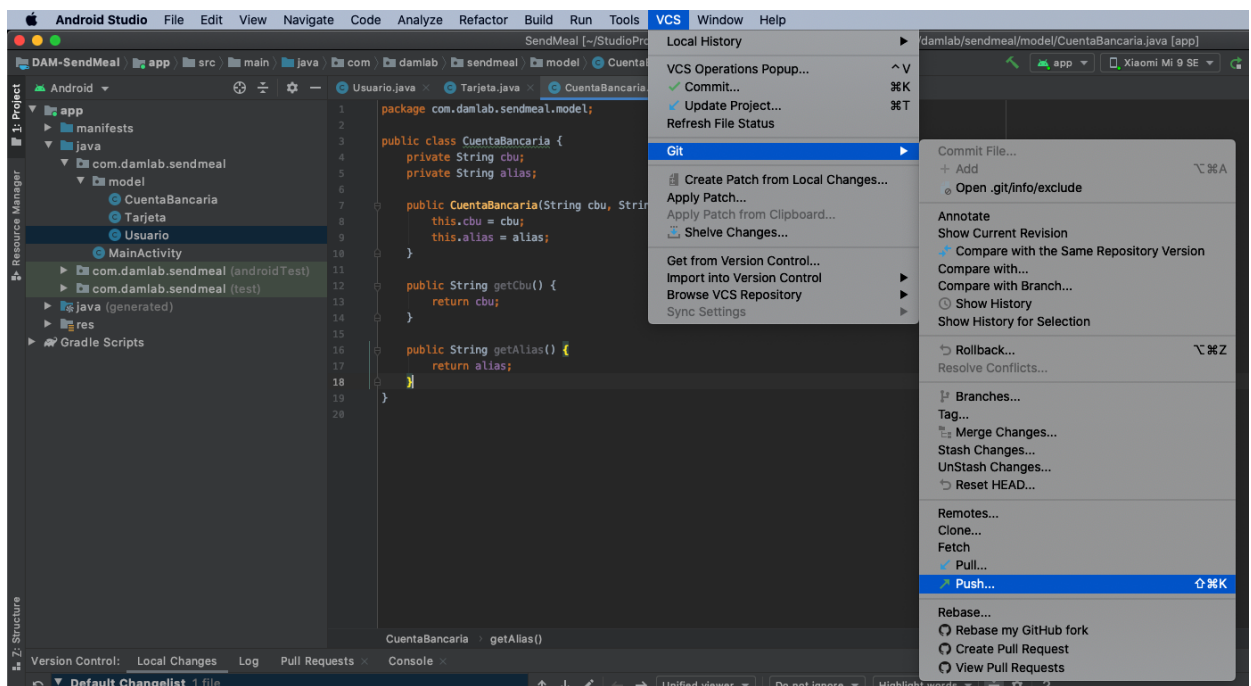


EJEMPLO:

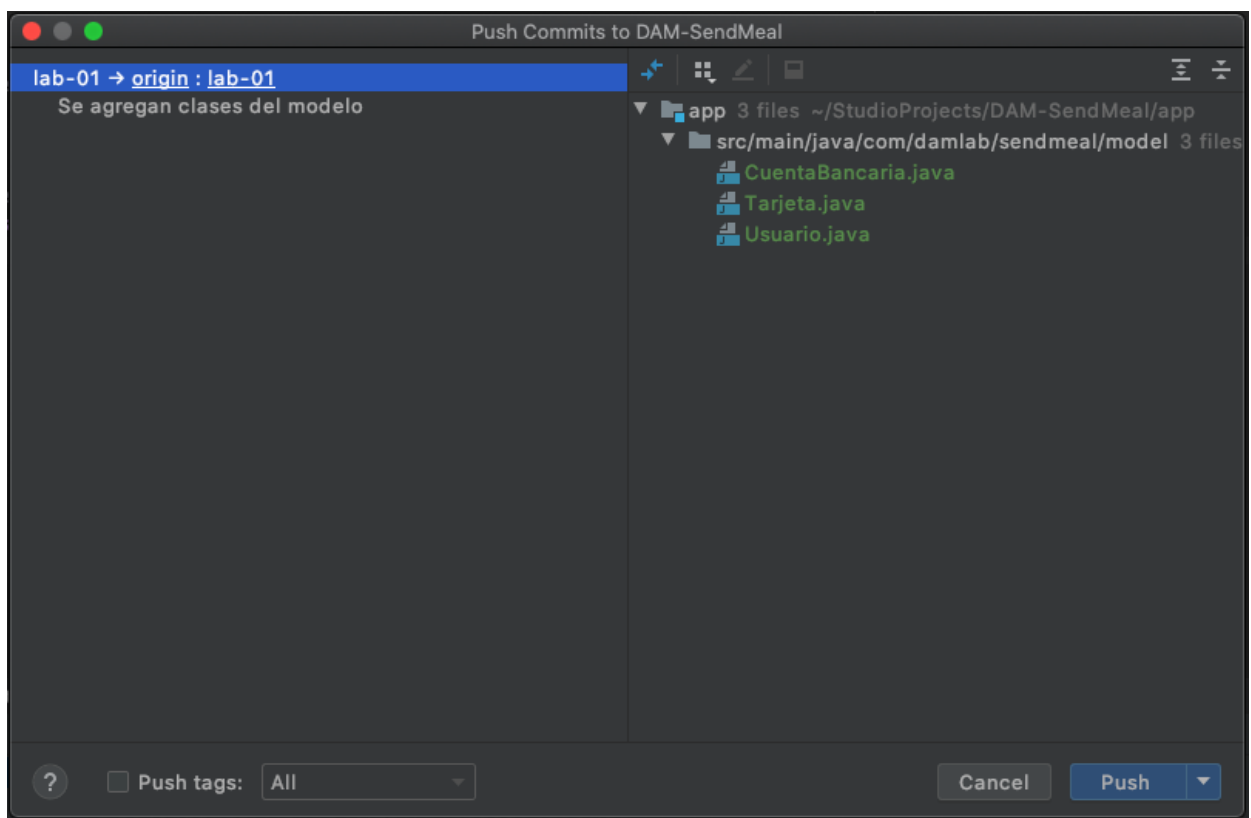


Notese que en este ejemplo solo se estan subiendo las clases del model, esto es porque en este caso ya se habian realizado commits previos con otras partes del lab. Ademas, no se esta subiendo el contenido de la carpeta `.idea` ya que son archivos de configuración del IDE y no es necesario.

- Una vez completados los datos del commit presionar el botón **Commit** para efectuar la acción
- Con nuestros cambios commiteados en nuestro repositorio local es momento de persistirlos en github, para esto necesitamos realizar un **PUSH** a nuestro repositorio remoto (en github), desde el IDE podemos encontrarla presinando **VCS** → **Git** → **Push** o el atajo de teclado **CTRL + SHIFT + K** (Win y Linux) o **CMD + SHIFT + K** (Mac)



6. Luego nos aparecera una ventana de confirmacion con los commits que esten en nuestro repositorio local y no en el remoto, estos son los commits que seran pusheados.



En la imagen se puede ver que nos indica que la rama lab-01 (local) sera pusheado (→) a la rama lab-01 en github (origin, o el nombre que le hayamos asignado al **Remote** en el paso 3 del item 2 - Configurar GIT)

7. Si todo esta ok presionamos **Push** y comenzará a realizar el proceso para pushear nuestros cambios a github (Es posible que nos pida nuestras credenciales de acceso a github si no las habiamos puesto antes).

8. Comprobar en el sitio

- Navegar a <https://github.com/{usuario}/{nombre-de-repo}/>
- Seleccionar la rama lab-1
- Comprobar que se vea reflejado el commit que hemos realizado

