Retos Iniciales Android Studio

1º) Cambiar el color del tema

2º) Manda un captura de los plugins.

3º) Plugin asignado por docente. Buscar información del mismo en la red.

Para el plugin adjuntar una captura, descripción y enlace al plugin

4º) Buscar repositorios de Android (al menos dos repositorios y uno de ellos en ***lenguaje Kotlin***)

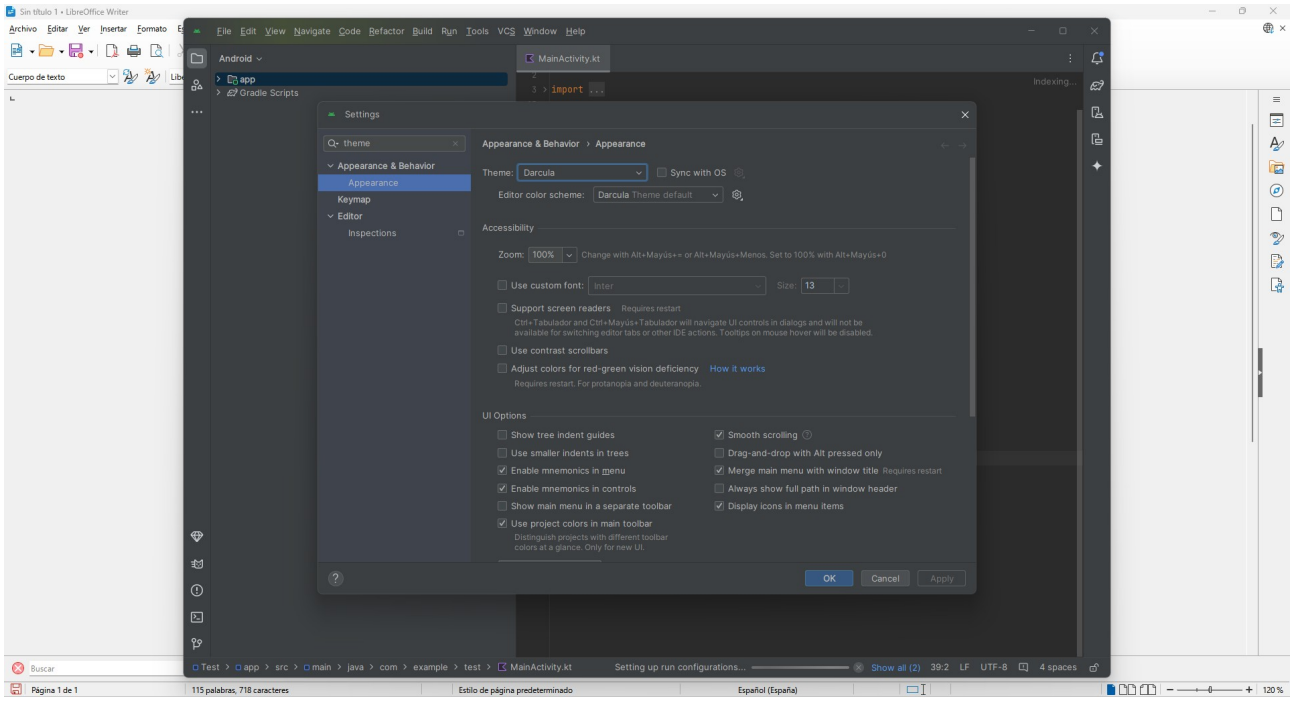
Para cada repositorio adjuntar una captura, descripción y enlace al repositorio correspondiente.

5º) ***New.*** Explica utilizando imágenes y describiendo la acción ¿Cómo se importa un proyecto github en Android Studio?

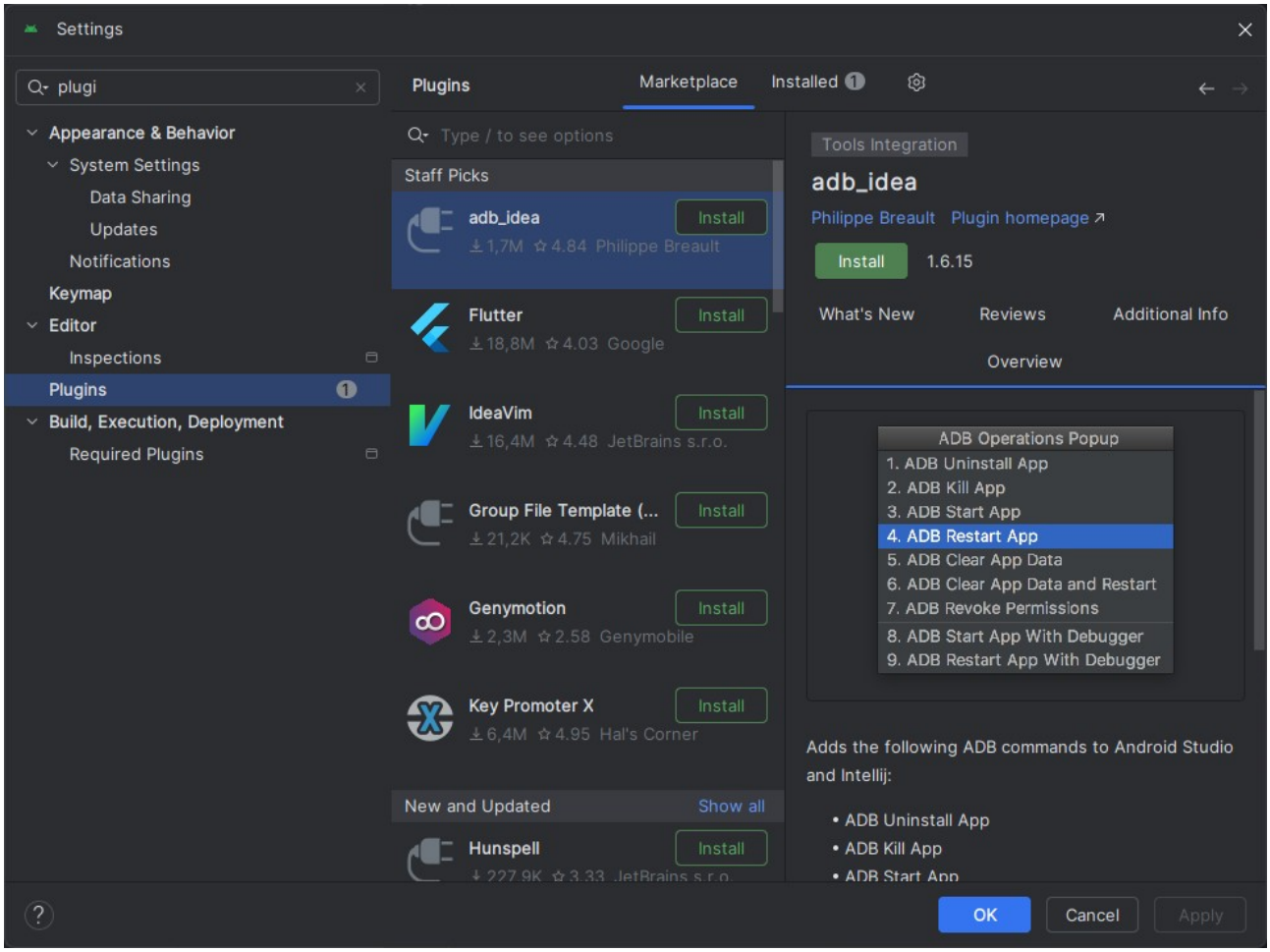
Agrega el contenido del readme en este apartado con un epígrafe propio tipo. **5.2 - Readme del repositorio seleccionado.**

Indica alguna curiosidad que se explica en el readme.

1º) Cambiar el color del tema



2º) Manda un captura de los plugins.



3º) Plugin asignado por docente. Buscar información del mismo en la red.

Para el plugin adjuntar una captura, descripción y enlace al plugin.

Informacion: Este plugin es ampliamente usado por desarrolladores que necesitan convertir estructuras complejas de JSON a clases de Kotlin de manera rápida. Algunas características y ventajas incluyen:

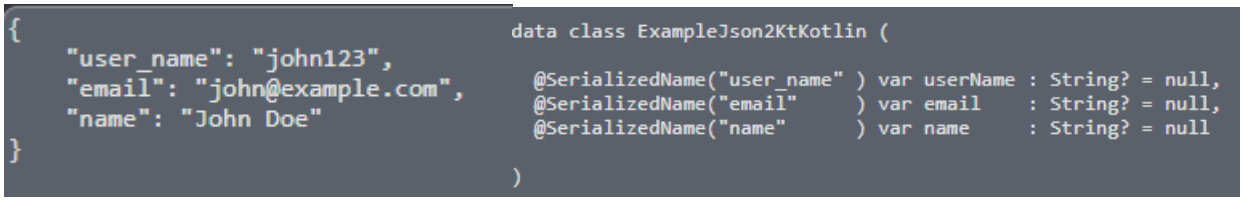
- Genera las clases con propiedades correspondientes a cada campo en el JSON.
- Facilita la compatibilidad con librerías de serialización populares.
- Ahorras tiempo en el desarrollo al no tener que escribir las clases manualmente.

Descripción: El plugin json2kotlin convierte un archivo o string JSON en una estructura de clases de datos en Kotlin con la sintaxis adecuada. Estas clases de datos son fundamentales cuando trabajas con APIs REST, ya que permiten manejar las respuestas de JSON de manera fácil y eficiente dentro del código Kotlin. Además, el plugin genera el código con opciones como:

- Manejo de valores nulos (`null`) en Kotlin.
- Uso de anotaciones de librerías como Gson o Moshi para la serialización/deserialización automática.
- Creación de clases anidadas según la estructura jerárquica del JSON.

Link: https://json2kt.com/

Captura ejemplo:



```
{
    "user_name": "john123",
    "email": "john@example.com",
    "name": "John Doe"
}
```

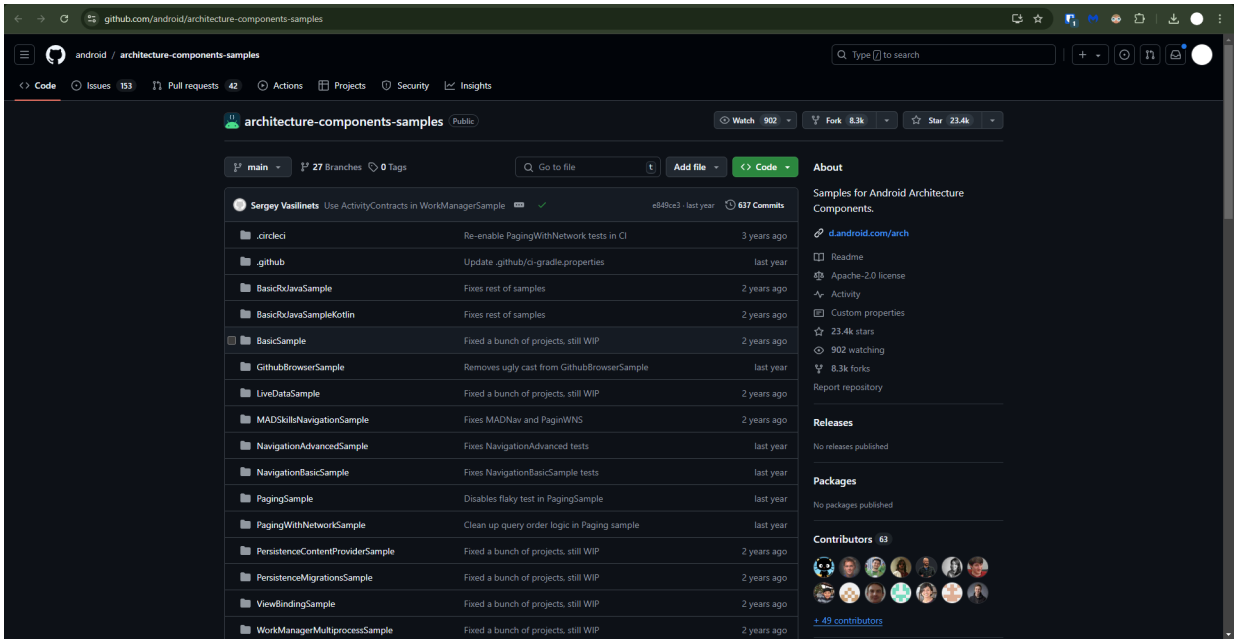
```
data class ExampleJson2KtKotlin (

  @SerializedName("user_name" ) var userName : String? = null,
  @SerializedName("email"     ) var email    : String? = null,
  @SerializedName("name"      ) var name     : String? = null

)
```

4º) Buscar repositorios de Android (al menos dos repositorios y uno de ellos en **lenguaje Kotlin**)

Para cada repositorio adjuntar una captura, descripción y enlace al repositorio correspondiente.

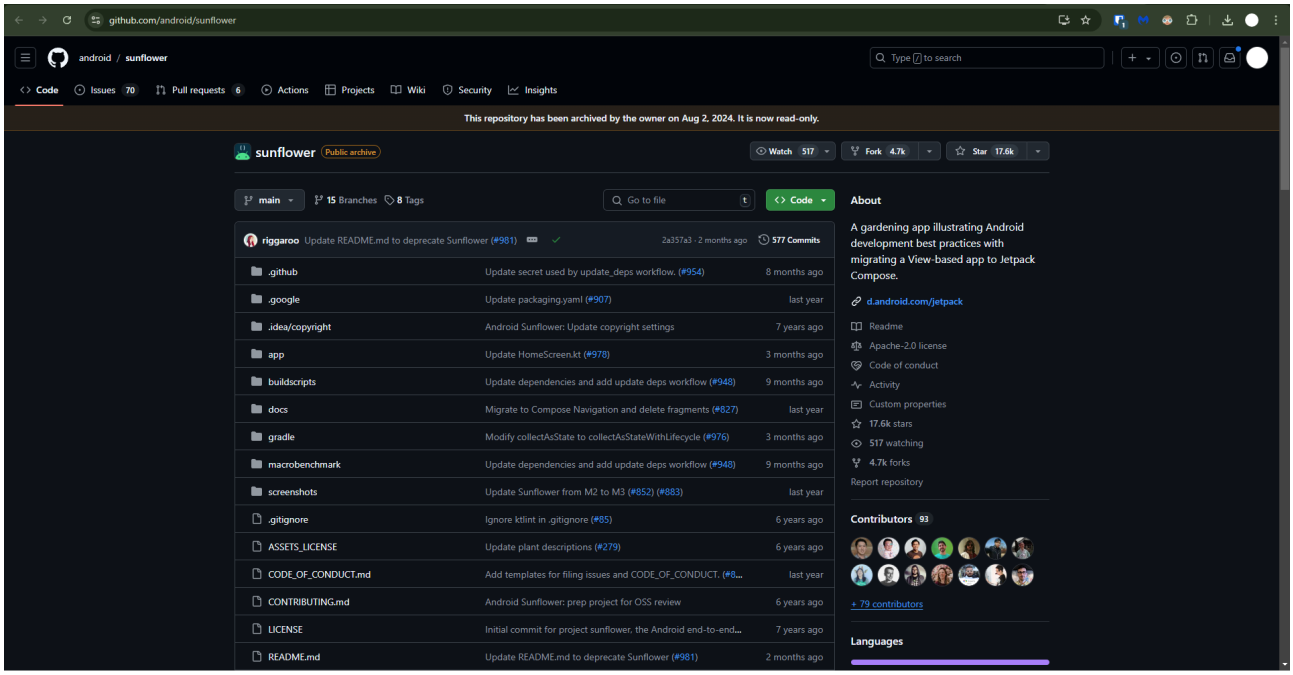
1. Repositorio en Java: AndroidArchitectureComponents

- **Descripción**: Este repositorio, gestionado por Google, proporciona ejemplos del uso de los componentes de arquitectura de Android, como **ViewModel**, **LiveData**, y **Room**. Estos componentes están diseñados para ayudar a los desarrolladores a estructurar su código de manera más eficiente, separando la lógica de la interfaz de usuario y los datos.
- **Enlace al repositorio**: [Android Architecture Components (Java)]



2. Repositorio en Kotlin: Android-Sunflower

- **Descripción**: **Android Sunflower** es un repositorio de muestra desarrollado por Google que utiliza el lenguaje **Kotlin**. Este proyecto demuestra cómo crear una aplicación moderna y de arquitectura limpia en Android utilizando las mejores prácticas. Incluye el uso de tecnologías como **Room** (para la persistencia de datos), **ViewModel**, **LiveData**, y **Navigation Component**, todas integradas con **Kotlin Coroutines** y **Flow**. Es un excelente ejemplo para entender cómo combinar componentes de arquitectura de Android con Kotlin.
- **Enlace al repositorio**: [Android Sunflower (Kotlin)]



5º) **New.** Explica utilizando imágenes y describiendo la acción ¿Cómo se importa un proyecto github en Android Studio?

Agrega el contenido del readme en este apartado con un epígrafe propio tipo. **5.2 - Readme del repositorio seleccionado.**

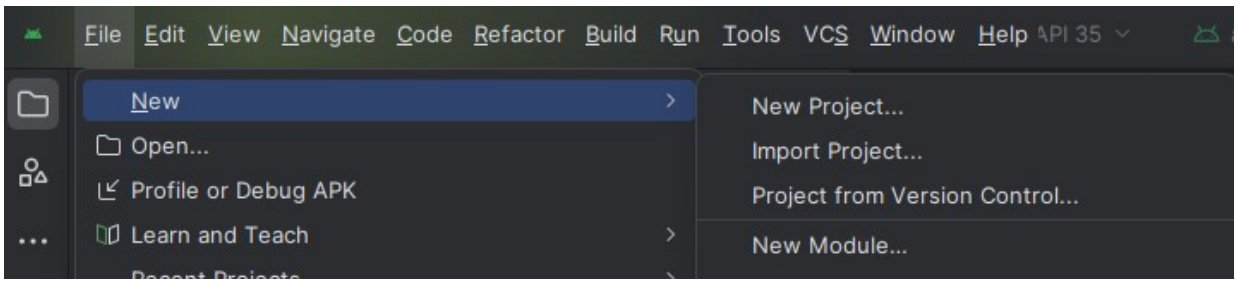
Indica alguna curiosidad que se explica en el readme.

1. Abrir Android Studio

- Abre **Android Studio** en tu ordenador.

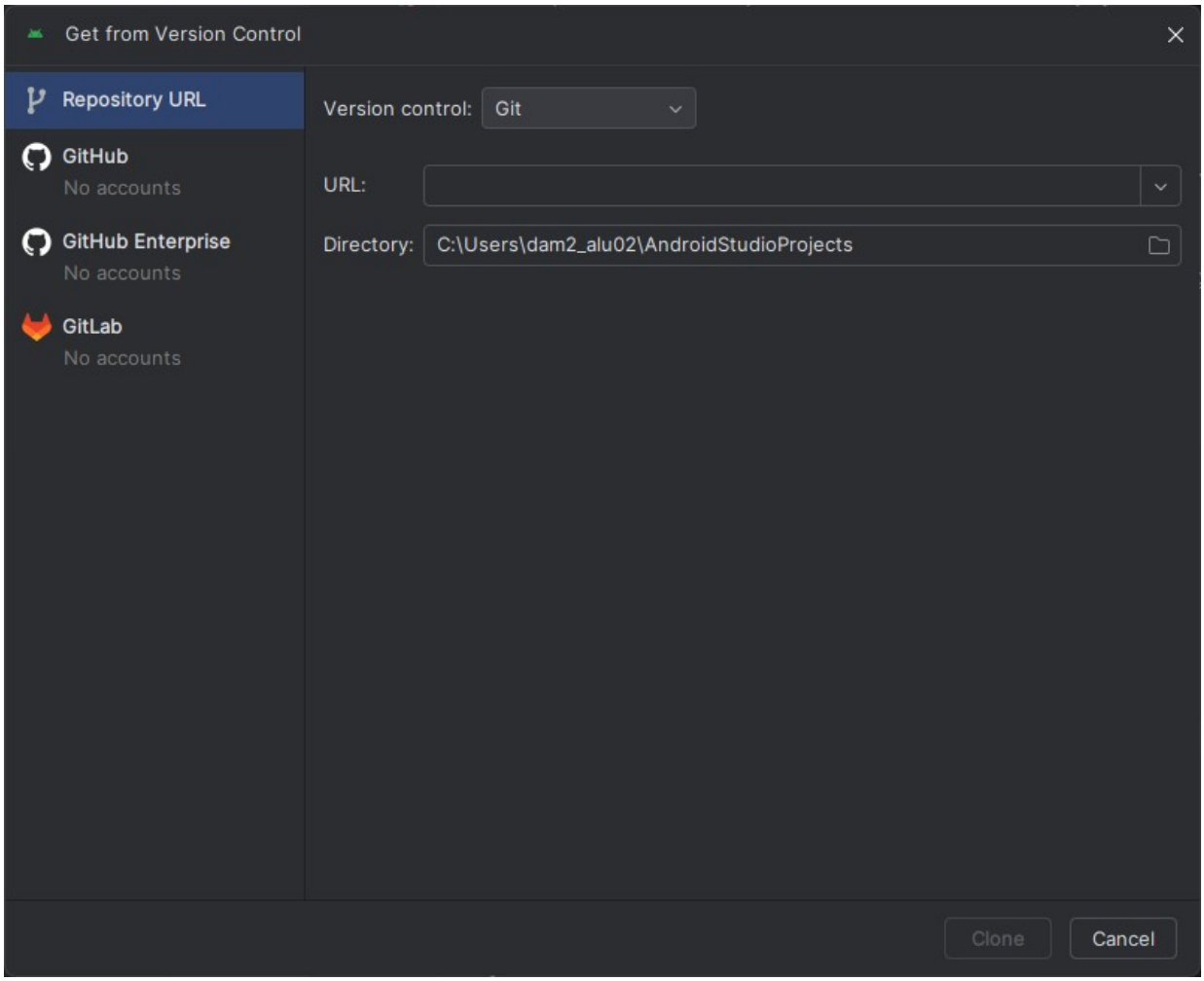
2. Seleccionar "Project from Version Control"

- Ve a la esquina superior izquierda, selecciona el menú **File**.
- En el menú desplegable, selecciona la opción **New** > **Project from Version Control...**.



3. Importar por URL

- En la siguiente pantalla, selecciona Respository URL
- Pega la **URL del repositorio GitHub** que deseas clonar en el campo de texto proporcionado.
- A continuación, selecciona la ubicación donde deseas clonar el proyecto en tu sistema.



4. Clonar el proyecto

- Haz clic en **Clone**.
- Android Studio descargará el proyecto y lo abrirá automáticamente en el entorno de desarrollo.

5. Sincronizar Gradle

- Una vez que el proyecto se haya clonado, Android Studio sincronizará los archivos **Gradle** automáticamente. Es posible que debas actualizar o sincronizar los archivos Gradle en función de las dependencias utilizadas.

Redme de Android Sunflower (Kotlin):

```
# Android Sunflower with Compose

Warning: The Sunflower repository is no longer under maintenance, We are prioritizing https://github.com/android/compose-samples as the up-to-date source of truth for Compose best practises. Please use that repository and sample set to continue learning about Jetpack Compose. If you'd like to continue using Sunflower, we encourage you to maintain your own fork of the sample.

A gardening app illustrating Android development best practices with migrating a View-based app to

Jetpack Compose. To learn about how Sunflower was migrated to Compose, see the

[migration journey](https://github.com/android/sunflower/blob/main/docs/MigrationJourney.md) document.

> [!Note]

> To see the original View implementation of Sunflower, checkout the [`views`](https://github.com/android/sunflower/tree/views) branch.

## Screenshots

<img src="screenshots/SunflowerM3Screenshots.png"/>

## Features

This sample showcases how to migrate an existing View-based app (Material 2) to Compose (Material 3).

See the linked migration journey doc above to learn more.

> [!Note]

> As Compose cannot render HTML code in `Text` yet. The

> `AndroidViewBinding` API is used to embed a `TextView` in Compose. See the

> `PlantDescription` composable in the

> [PlantDetailView file](app/src/main/java/com/google/samples/apps/sunflower/compose/plantdetail/PlantDetailView.kt).

## Requirements

### Unsplash API key

Sunflower uses the [Unsplash API](https://unsplash.com/developers) to load pictures on the gallery

screen. To use the API, you will need to obtain a free developer API key. See the

[Unsplash API Documentation](https://unsplash.com/documentation) for instructions.

Once you have the key, add this line to the `gradle.properties` file, either in your user home

directory (usually `~/.gradle/gradle.properties` on Linux and Mac) or in the project's root folder:

```

unsplash_access_key=<your Unsplash access key>

```
```

The app is still usable without an API key, though you won't be able to navigate to the gallery screen.

Android Studio IDE setup

For development, the latest version of Android Studio is required. The latest version can be

downloaded from [here](https://developer.android.com/studio/).

Sunflower uses [ktlint](https://ktlint.github.io/) to enforce Kotlin coding styles.

Here's how to configure it for use with Android Studio (instructions adapted

from the ktlint [README](https://github.com/shyiko/ktlint/blob/master/README.md)):

- Close Android Studio if it's open

- Download ktlint using these [installation instructions](https://github.com/pinterest/ktlint/blob/master/README.md#installation)

- Apply ktlint settings to Android Studio using these [instructions](https://github.com/pinterest/ktlint/blob/master/README.md#-with-intellij-idea)

- Start Android Studio

Additional resources

Check out these Wiki pages to learn more about Android Sunflower:

- [Notable Community Contributions](https://github.com/android/sunflower/wiki/Notable-Community-Contributions)

- [Publications](https://github.com/android/sunflower/wiki/Sunflower-Publications)

Non-Goals

Previously, this sample app was focused on demonstrating best practices for

multiple Jetpack libraries. However, this is no longer the case and development

will instead be focused on how to adopt Compose in an existing View-based app.

So, there are no plans to implement features outside of this scope. Keep this

in mind when making contributions to this library.

Support

- Stack Overflow:
  - https://stackoverflow.com/questions/tagged/android-jetpack-compose

If you've found an error in this sample, please file an issue:

https://github.com/android/sunflower/issues

Patches are encouraged, and may be submitted by forking this project and submitting a pull request

through GitHub.

Third Party Content

Select text used for describing the plants (in `plants.json`) are used from Wikipedia via CC BY-SA 3.0 US (license in `ASSETS_LICENSE`).

"[seed](https://thenounproject.com/search/?q=seed&i=1585971)" by [Aisyah](https://thenounproject.com/aisyahalmasyira/) is licensed under [CC BY 3.0](https://creativecommons.org/licenses/by/3.0/us/legalcode)

Curiosidad del README:

Una curiosidad interesante del archivo README de Android Sunflower es que no solo utiliza los componentes básicos de arquitectura de Jetpack, sino que también promueve el uso de modularización dentro de la estructura del proyecto. Esto significa que el proyecto está diseñado para ser escalable y mantenible a largo plazo, dividiendo las funcionalidades en módulos que pueden desarrollarse y probarse de forma independiente, una práctica avanzada en el desarrollo de aplicaciones grandes.