ANEXO EXAMEN DE CERTIFICACIÓN

Plan de Estudio	Desarrollo Aplicaciones Android Trainee v.2
Tiempo	6 horas
Anexo	Caso PlantApp - Cuidado verde

Caso: “PlantApp - Cuidado verde”

PlantApp es un emprendimiento familiar que actualmente, está en proceso de masificación de sus ventas.
Actualmente sus canales de venta son por medio de redes sociales. Con estas aplicaciones, captan posibles compradores, repartiendo a domicilio.
Debido a los múltiples mensajes que reciben, no todos son respondidos, ya que sus dueños, sólo a ratos libres, se dedican a revisar sus redes sociales. Por lo que se han comunicado con un equipo de desarrollo que ha tomado los requerimientos del caso. Se han determinado los siguientes puntos para poder automatizar por medio de una aplicación la venta de productos de “PlantApp”.

Alcance

Para comenzar con el desarrollo se va a desarrollar un mínimo producto viable, donde los posibles clientes puedan observar las plantas y productos que estos emprendedores tienen en venta y poder seleccionar alguna, además de hacer llegar esta información al departamento de ventas.

La API desde donde provienen los datos es limitada y cuenta solo con la información mínima para poder mostrarla en la app Android. Por este mismo motivo, aún no se termina de crear ni implementar una forma de recopilar datos de los usuarios, o procesar las ventas.

En primera instancia, se solicita generar una aplicación Android que consuma los servicios proporcionados por la API. Se solicita que este mínimo producto viable permita con el tiempo poder escalar la aplicación sin mayores inconvenientes.

Características generales:

- Considerar que para esta primera versión se busca tener una gran cobertura de dispositivos manteniendo los costos de mantención bajos, la API mínima es 27 y el target 33.
- La aplicación no cuenta con autenticación de usuario.

- Consumir un servicio **REST** con **2 endpoints**:
  - El primero entrega una colección de **plantas**
  - El segundo entrega la información de un elemento dado un identificador.
- Se necesita que funcione de forma offline, por lo que se debe guardar en una base de datos local del dispositivo.
- Primera pantalla: Listado de los productos disponibles
- Segunda pantalla: Detalle del producto. Se navega a esta pantalla al seleccionar un producto del listado.
  - Además del detalle debe tener un botón para enviar un correo de contacto (Este deberá ejecutar un *Intent* implícito a una app de correo electrónico).

Código, arquitectura y dependencias

Dadas las características de este proyecto, la arquitectura debe permitir que la aplicación funcione sin conexión a internet y que se puedan agregar funcionalidades de forma ordenada, es por eso que se requiere cuide un código legible y que use específicamente la siguiente arquitectura:

- La arquitectura puede ser **MVP-LiveData-ROOM** o **MVVM-LiveData-ROOM**.
- Debe guardar los datos en la persistencia del teléfono (ROOM) y mostrarlos en las vistas correspondientes. (debe guardar el listado de productos y también aquellos que selecciona para ver su detalle).
- Puede utilizar los lenguajes de programación **Kotlin** o **Java**.
  - Si utiliza Kotlin, debe usar Coroutines para manejar el trabajo asíncrono, según corresponda.
  - Si utiliza Java, debe usar alguna alternativa como executors para manejar el trabajo asíncrono, según corresponda.
- Utilizar paquetes representativos, respetar convenciones y estilos de codificación (indentación, reutilización, legibilidad, convenciones de nombre, comentarios cuando sea necesario)
- Se requiere utilizar una actividad y múltiples fragmentos.
- Se recomienda utilizar la biblioteca navigation para navegar entre fragmentos.
- Todos los textos que no sean obtenidos a partir de la API REST deben ser traducibles.

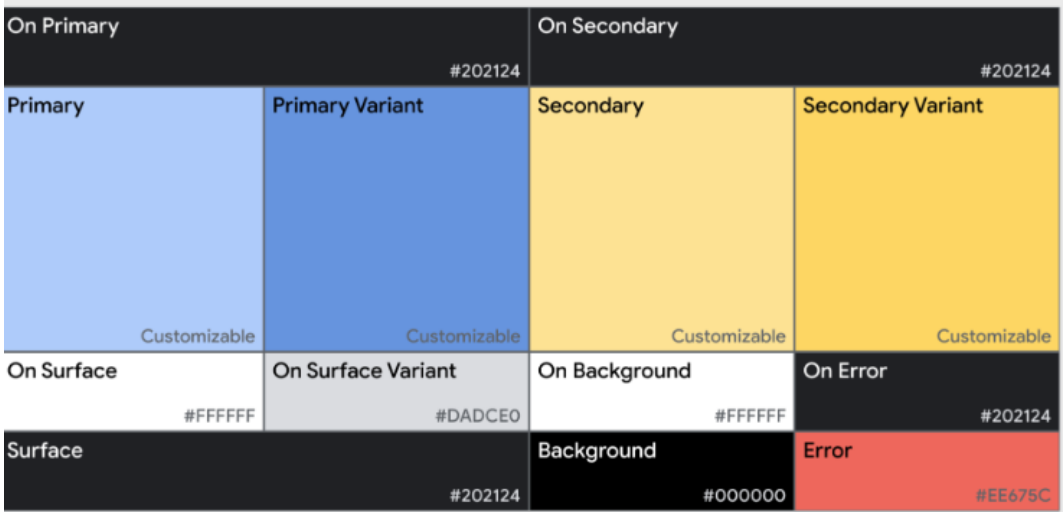
Sobre las dependencias:

- Los request HTTP deben ser realizados utilizando la librería **Retrofit**.
- Para mostrar imágenes puede utilizar cualquier biblioteca, por ejemplo: Glide, Picasso, Coil.

Diseño del Aplicativo

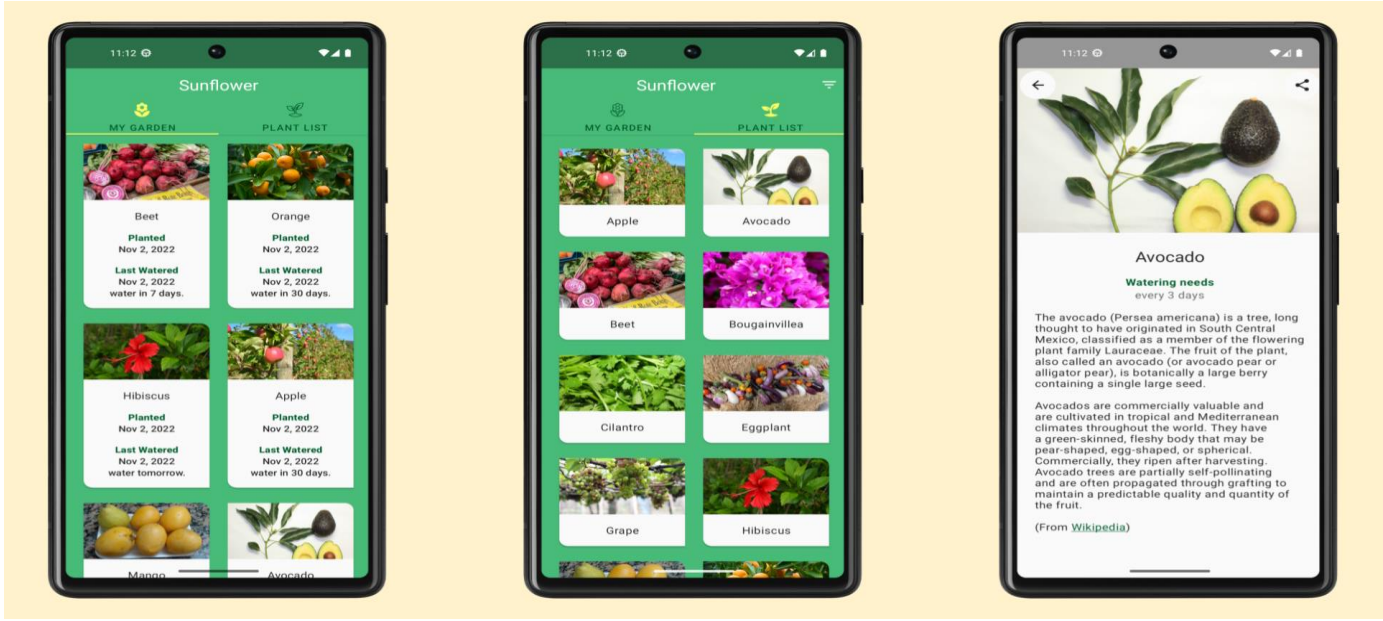
El diseñador es nuevo en el equipo, por lo que para la primera versión de la aplicación no ha podido traspasar el diseño a un programa que le entregue las medidas en formato Android. Por lo que los tamaños y distancias pueden no ser exactos en esta primera versión. Para evitar confusiones, el jefe de proyecto ha decidido enumerar las especificaciones que no pueden faltar:

- Use la siguiente paleta para los colores de la aplicación:



- No se especifican los tipos de layouts a utilizar, por lo tanto, es responsabilidad del desarrollador. Debe utilizar su criterio manteniendo una correcta visualización de los elementos.
- Se recomienda que la segunda pantalla tenga la posibilidad de hacer scroll para visualizar el contenido de forma correcta.
- El área de diseño todavía no ha estandarizado la línea gráfica de los íconos, pero debe al menos agregar un ícono usando vectores.
- Para los layouts se recomienda utilizar Material Design.

A continuación se presentan algunas ejemplos de pantallas que cumplen con lo requerido:



Tipos de listado utilizando distintos CardViews **Detalle**

Solo referenciales, puede cambiar el diseño según estime conveniente en base al aplicativo

https://github.com/android/sunflower

Endpoints y Modelos de Datos

Ambos endpoints se deben acceder a través del verbo HTTP de request GET.

El primer endpoint es para obtener una lista de libros.

1) **`https://my-json-server.typicode.com/mauricioponce/TDApi/plantas`**

Este endpoint retorna con un listado de productos.
A continuación un ejemplo con 2 elementos en el listado:

```
[
	{
		"id": 1,
		"nombre": "Rosa",
		"tipo": "Flor",
		"imagen": "https://upload.wikimedia.org/wikipedia/commons/thumb/f/f7/Rosa_%27Matador%27_%28actm%29.jpg/640px-Rosa_%27Matador%27_%28actm%29.jpg",
		"descripcion": "La rosa es una hermosa flor conocida por su amplia variedad de colores y aromas cautivadores. Es símbolo de amor y belleza en muchas culturas."
	},
	{
		"id": 2,
		"nombre": "Lavanda",
		"tipo": "Hierba",
		"imagen": "https://upload.wikimedia.org/wikipedia/commons/thumb/8/8f/Fiori_di_Lavanda.jpg/640px-Fiori_di_Lavanda.jpg",
		"descripcion": "La lavanda es una hierba aromática famosa por su fragancia relajante y propiedades medicinales. Sus flores de color violeta la hacen una planta encantadora."
	}
]
```

Ilustración 2: Example Response

El segundo endpoint corresponde al detalle y se accede indicando el ID específico:

2) https://my-json-server.typicode.com/mauricioponce/TDApi/plantas/{ID}

En esta respuesta se encuentra tan solo 1 objeto producto que corresponde al ID. A continuación el resultado cuando se indica el id número 1

https://my-json-server.typicode.com/mauricioponce/TDApi/plantas/1


```
{
        "id": 1,
        "nombre": "Rosa",
        "tipo": "Flor",
        "imagen":
    "https://upload.wikimedia.org/wikipedia/commons/thumb/f/f7/Rosa_%27Matador%27_%28actm%29.jpg/640px-Rosa_%27Matador%27_%28actm%29.jpg",
        "descripcion": "La rosa es una hermosa flor conocida por su amplia variedad de colores y aromas cautivadores. Es símbolo de amor y belleza en muchas culturas."
}
```

Funcionalidad de contacto

Cuando el usuario está viendo el detalle de un equipo al hacer click en un botón del detalle, tiene que tener la posibilidad de envíar un correo con la siguiente información prellenada:

- Destinario: luci@plantapp.cl
- Asunto: Consulta por Producto {TITLE}
- Mensaje:

> "Hola
>
> Vi el producto {nombre} y me gustaría que me contactaran a este correo o al siguiente número ___
>
> Quedo atento."

Recuerde reemplazar lo indicado entre paréntesis cursivos por la información correspondiente al producto seleccionado. No incluya los paréntesis cursivos en correo.

No es necesario que el usuario llene el número de ante mano, conserve los guiones bajos u otro símbolo que le indique al usuario que ahí tiene que escribir su número cuando esté viendo el mensaje en la aplicación de correos.

Testing

Preocupados por la calidad del código y estabilidad de la app, se ha definido un mínimo de test a llevar a cabo.

- Al menos un test unitario
- Al menos un test de instrumental.

Sugerencias:

- Test unitario que verifique la respuesta de los endpoints usando un servidor de pruebas como mockwebserver.
- Test instrumental que compruebe la persistencia de datos con ROOM.
- Test unitarios sobre cualquier función.