ANEXO EXAMEN DE CERTIFICACIÓN

Plan de Estudio	Desarrollo Aplicaciones Android Trainee
Anexo	Caso "Crypto Invest center"

Caso “Criptomonedas Invest center”

Dos emprendedores del ámbito tecnológico, le han pedido a tu empresa realizar un mínimo producto viable. Una aplicación móvil nativa en Android que permita mostrar las diferentes ofertas de Trading de criptomonedas que luego van a comercializar en sus proyectos futuros.

En una primera instancia, solo se pide mostrar información, en este caso será el catálogo de criptomonedas, los usuarios de la app pueden visualizar los detalles y hacer llegar una petición de envío de información.

Esta app permitirá probar la idea por lo tanto se te ha encargado realizar una app Android nativa, para sentar las bases de este proyecto, con la idea de que pueda escalar en un futuro.

Alcance del Proyecto

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 enviar información actualizada en tiempo real.

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

Esto quiere decir que debe utilizar patrones de arquitectura escalables y buenas prácticas de la industria.

En esta primera versión, la app realizará lo siguiente:

- Debe consumir un servicio **REST**, desde donde provienen los datos a mostrar.
- El servicio REST cuenta con **2 endpoints**, uno que entregará una colección de **Criptomonedas** con datos generales y el segundo que entregará más información de estas monedas.

- Considerando que para esta primera versión se busca tener una gran cobertura de dispositivos manteniendo los costos de mantención bajos, la API mínima debe ser 23 y el target 30.

Se espera que cualquier usuario instale la app sin necesidad de autenticarse, y pueda ver una lista de las monedas disponibles, ver los detalles de la que seleccione y enviar un correo al área de marketing con información predefinida.

- La primera pantalla de la app debe ser una lista de las monedas ofrecidas (Recyclerview).
- La segunda pantalla muestra la información en detalle de la moneda seleccionada.
    - Debe tener un botón para solicitar información (Este deberá ejecutar un intent implícito a una app de correo electrónico, con los datos necesarios).

Código, arquitectura y dependencias

El jefe de proyectos entiende que, aunque el alcance de la aplicación es inicial, la arquitectura que utilice le permitirá seguir expandiendo o modificando el proyecto de acuerdo a los resultados iniciales, es por eso que se requiere cuide un código legible y que use específicamente la siguiente arquitectura en conjunto con estas dependencias:

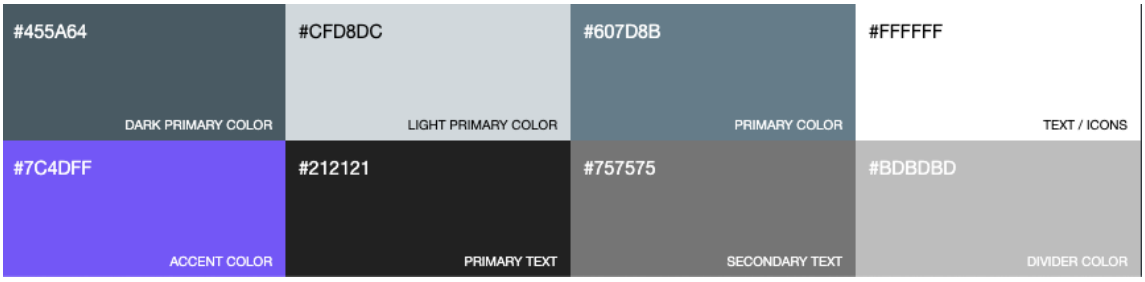
- Respetar las convenciones y estilos de codificación (indentación, reutilización, comentarios, legibilidad, convenciones de nombre, etc.)
- La arquitectura del aplicativo puede ser **MVP-LiveData-ROOM** o **MVVM-LiveData-ROOM**.
    - Debe guardar los datos en la persistencia del teléfono(ROOM) y mostrarlos en las vistas correspondientes.
- Los request HTTP deben ser realizados utilizando la librería **Retrofit**.
- 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 Runnables o Executors para manejar el trabajo asíncrono, según corresponda.
- Utilice las bibliotecas que estime necesario para hacer TEST.

Diseño del Aplicativo

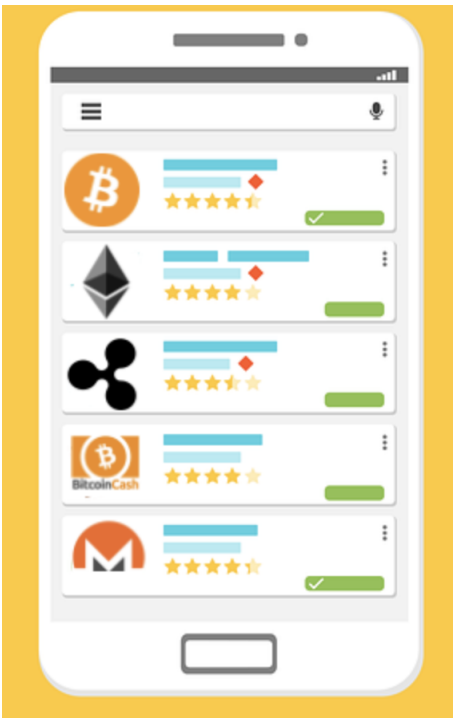
No existe una guía de marca, ni colores corporativos, por lo que se sugieren los siguientes elementos. Como desarrollador puedes tomar decisiones, cambiando algunos elementos siempre buscando que la app sea llamativa para los usuarios.

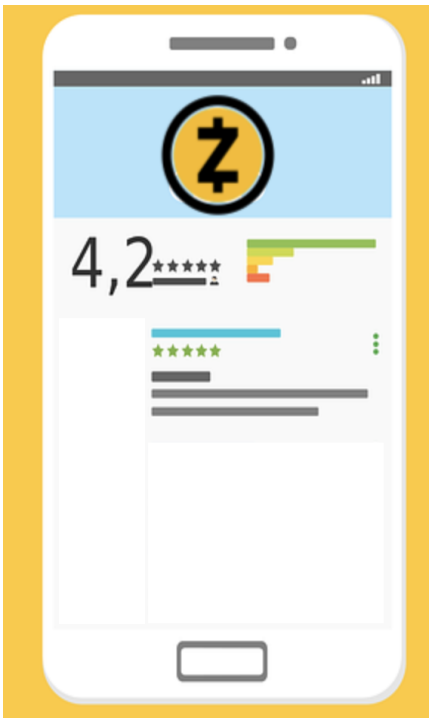
Para evitar confusiones, el jefe de proyecto ha decidido enumerar las especificaciones que no pueden faltar:

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



- Aunque los tamaños pueden variar, la diagramación de los layouts se debe mantener, esto incluye las filas de la lista y la segunda pantalla (Utilizar recomendaciones de Material Design).
- Se recomienda utilizar una actividad y múltiples fragmentos.
- No se especifican qué tipos de layouts debe utilizar, por lo tanto es responsabilidad del desarrollador, debe utilizar su criterio manteniendo como punto importante la correcta visualización de los elementos.
- Se recomienda que la segunda pantalla tenga la posibilidad de hacer scroll para visualizar el contenido de forma correcta y que se ajuste a cualquier tamaño de pantalla.
- Todos los textos que **NO** sean obtenidos a partir de la API REST deben ser traducibles.
- El área de diseño todavía no ha estandarizado la línea gráfica de los íconos, pero debe tratar de utilizar al menos 2 para mejorar el apartado visual de la app. Para los íconos tiene que utilizar vector graphics.
- Se adjuntan algunos ejemplos de pantallas tipo, pero basándonos en el tiempo, debe tomar decisiones que permitan realizar el proyecto y que se ajusten a los datos entregados por el servicio REST.





Listado. Detalle.

Solo referenciales, Los campos no coinciden con los entregados por la API. Son solo una referencia y no es exigencia que sean iguales.

End Points y Modelos de Datos

Ambos end points se deben acceder a través del verbo de request HTTP GET. El primer end point es para obtener una lista de monedas disponibles.

1) **https://fake-server-app-crypto.herokuapp.com/general/**

En esta respuesta se encuentran las criptomonedas agrupadas en una colección donde cada moneda tiene la data reducida:

```
{
    "id": "BTC",
    "currency": "BTC",
    "symbol": "BTC",
    "name": "Bitcoin",
    "logo_url":
"https://s3.us-east-2.amazonaws.com/nomics-api/static/images/currencies/btc.sv
g",
    "status": "active",
    "price": "57908.88281873",
    "price_date": "2021-05-08T00:00:00Z",
    "price_timestamp": "2021-05-08T06:15:00Z",
    "rank": "1"
  }
```

Ilustración 1: Reduced crypto Model

A continuación, puede ver un ejemplo con 2 objetos:

```
[
  {
    "id": "BTC",
    "currency": "BTC",
    "symbol": "BTC",
    "name": "Bitcoin",
    "logo_url":
"https://s3.us-east-2.amazonaws.com/nomics-api/static/images/currencies/btc.sv
g",
    "status": "active",
    "price": "57908.88281873",
    "price_date": "2021-05-08T00:00:00Z",
    "price_timestamp": "2021-05-08T06:15:00Z",
    "rank": "1"
```

```
  },
  {
    "id": "ETH",
    "currency": "ETH",
    "symbol": "ETH",
    "name": "Ethereum",
    "logo_url":
"https://s3.us-east-2.amazonaws.com/nomics-api/static/images/currencies/eth.sv
g",
    "status": "active",
    "price": "3538.44529165",
    "price_date": "2021-05-08T00:00:00Z",
    "price_timestamp": "2021-05-08T06:15:00Z",
    "rank": "2"
  }
]
```


Ilustración 2: Example Crypto Response

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

2) https://fake-server-app-crypto.herokuapp.com/details/ETH

En esta respuesta se encuentra tan solo 1 objeto producto que corresponde al ID **ETH**, la diferencia es que tiene todos los datos, para la vista de detalle.

```
{
  "id": "ETH",
  "currency": "ETH",
  "symbol": "ETH",
  "name": "Ethereum",
  "logo_url":
"https://s3.us-east-2.amazonaws.com/nomics-api/static/images/currencies/eth.sv
g",
  "status": "active",
  "price": "3538.44529165",
  "price_date": "2021-05-08T00:00:00Z",
  "price_timestamp": "2021-05-08T06:15:00Z",
  "circulating_supply": "115785365",
  "market_cap": "409700180507",
  "num_exchanges": "387",
  "num_pairs": "40629",
  "num_pairs_unmapped": "25629",
  "first_candle": "2015-08-07T00:00:00Z",
  "first_trade": "2015-08-07T00:00:00Z",
  "first_order_book": "2018-08-29T00:00:00Z",
  "rank": "2",
  "rank_delta": "0",
  "high": "3545.46612020",
  "high_timestamp": "2021-05-08T00:00:00Z",
  "1d": {
    "volume": "59414294288.88",
    "price_change": "99.40150754",
    "price_change_pct": "0.0289",
    "volume_change": "-3886350195.15",
    "volume_change_pct": "-0.0614",
    "market_cap_change": "11555867487.59",
    "market_cap_change_pct": "0.0290"
  }
}
```


Ilustración 3: Full Cryto Detail Model

Funcionalidad de Envío de Correo

Cuando el usuario está viendo el detalle de una Moneda seleccionada, al hacer click sobre algún botón, tiene que enviarse un correo con la siguiente información pre llenada:

- Destinatario: Info@cryptoinvest.cl

- Asunto: Solicito información sobre esta criptomoneda ID {ID}
- Mensaje:

> “Hola
>
> Quisiera pedir información sobre esta moneda {NAME}, me gustaría que me contactaran a este correo o al siguiente número _____
>
> Quedo atento.”

Recuerde reemplazar lo indicado entre paréntesis cursivos por la data correspondiente a la moneda seleccionada. 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 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.

Uso de Recursos. SVG

Los enlaces de las imágenes que provienen del servidor están en formato **SVG.** Esto hace que no sean cargados directamente por librerías como 'Picasso' o ´Glide´ sin tener que ajustar su configuración. Recomendamos el uso de la librería 'COIL'. https://coil-kt.github.io/coil/

También se proporciona un ejemplo de su uso en el siguiente repositorio. https://github.com/Himuravidal/CoilUseExample