Muun Coding Challenge
Android
Nicolás Dubiansky

Documento de diseño:

1. **Decisiones tomadas**

Respecto de los puntos de evaluación, decidí enfocarme en la solidez, en la experiencia del usuario y el rendimiento (lo cual no significa que he descuidado el resto de los puntos). Tomé esta decisión ya que creo que es un combo interesante para que el usuario pueda disfrutar de utilizar la aplicación.

Contemplé escenarios donde no hay conexión a internet, así como también desarrollé la aplicación tanto en modo “portrait” como “landscape”. Sabía que dicha decisión implicaba redoblar el esfuerzo y por lo tanto un mayor desafío al que pedía el “coding challenge”, pero aún así era algo que me motivaba para hacer una aplicación más robusta.

En cuanto a la UX&UI traté de seguir strandards de Material Design (ya sea desde combinacion de colores, como tamaños de elementos visuales). También he utilizado el recurso de “Dialogs” - “Toasts” - “SnackBars” para mantener al usuario informado acerca de lo que esta ocurriendo en la aplicación, así como también para guiarlo con la funcionalidad de cada pantalla. Considero que es un elemento importante ya que permite al usuario sentirse seguro y confiado en la navegación.

En cuanto a la estructura de diseño organice las pantallas con una AppBar y por lo general FloatingActionButtons con iconos intuitivos para el usuario. Traté de mostrar la información de una manera simple y concisa para no ensuciar la visual. He decidido seguir la lógica de “una actividad por pantalla”, en contraposición de utilizar “single activity + fragments/views”. Por supuesto que esto trae como todo ventajas y desventajas, y no hay una opción que sea la correcta. Al ser una app con pocas pantallas y con poca comunicación entre las mismas me pareció que una activity por pantalla podía solucionar el problema. No quiero ahondar mucho en el tema de Fragments v.s Activities porque ya fue hablado en la entrevista anterior, simplemente decir que en este caso tampoco había mucha reutilización visual, salvo en el caso de la “Text Address & QR Address” que se repetían en las pantallas de Address Generation y en la de Wallet Balance. Lo cual fue logrado a través de una Custom View.

Respecto a la arquitectura combiné MVVM (model view viewModel), con el patrón Observer, y utilice Retrofit+Event Bus (lo cual será detallado en la próxima sección). No seguí la misma arquitectura en todas las pantallas ya que tomé como motivación extra el poder hacer una app con distintas arquitecturas para evaluar performance. Finalmente fue casi imperceptible el resultado obtenido con ambas arquitecturas. Me hubiese gustado (con un poco más de tiempo) migrar toda la app a la estructura MVVM por cuestiones de prolijidad, mantenibilidad.

También utilice el patrón Singleton para mantener en memoria algunas entidades como “User” que contaba con información acerca de la Address, o del balance del usuario

por ejemplo, así como también para la creación de la clase encargada de manejar las Api´requests.

2. **Librerías y Frameworks:**
Utilicé Retrofit 2.0 para el manejo de Api Requests ya que permite tener un código prolijo, legible, brinda facilidad para realizar operaciones asincrónicas (threading) y cuenta con mucha documentación en internet.
Por otra parte utilice Event Bus, (combinado con Retrofit) para lograr la comunicación entre los componentes visuales (Activities por ej) y la respuesta de los request realizados a BlockCypher. Esta combinación permite desacoplar de las actividades la responsabilidad de hacer peticiones (como podría hacerse con un AsyncTask dentro de la misma) lo cual otorga flexibilidad a la hora de que el usuario pueda seguir usando la app mientras ocurre un pedido en background, así como también lograr una app “Portrait & Landscape”.

Para parsear respuestas, datos, y demás utilice Gson. La cual permite de una manera simple serealizar y deserealizar objetos. Utilice RecyclerView para mostrar el historial de transacciones en vez de utilizar ListView, por cuestiones de memoria/performance (ViewHolder pattern, LayoutManager, etc). No quiero entrar demasiado en detalle de RecyclerView, solo mencionar que es importante tener en cuenta que la implementación del “clickListener” la he hecho en el onCreateViewHolder y no en el onBindViewHolder ya que este método es llamado cada vez que scrolleamos el recycler (lo cual demandaría montones de creaciones innecesarias de dicho listener).

Además utilice Butterknife para evitar Boilerplate de “FindViewByID” y tener un código más simple a través de las annotations que brinda dicha librería.
Para realizar algunas animaciones utilicé la librería AndroidViewAnimations (YoYo), lo cual le da un poco mas de vida y movimiento a los componentes visuales. En cuanto a la generación del código QR para la address utilicé la librería Zxing.
Para la arquitectura MVVM utilicé componentes como “ViewModel” “Livedata”, que son brindados por la librería de “android.arch.lifecycle”.
Después de un análisis previo, decidí no utilizar una base de datos local para la persistencia de datos, ya que los volúmenes de datos que maneja esta aplicación pueden lograrse a través de persistir en las “Shared Preferences”. Noté que la performance era buena y no tuve la necesidad de utilizar alguna librería para el manejo de ORM.

3. **Dificultades surgidas:**
Me tope con la dificultad de generar la imagen de código QR, cosa que nunca había hecho antes, así como también con la utilización de “@Url” de Retrofit para resolver pedidos a url distintas (la de send bitcoins era distinta al resto) en vez de hacerlo desde un Interceptor.
Para el parseo de fechas me pasó que en algunos casos el formato que obtenía como respuesta al historial de transacciones era “`yyyy-MM-dd'T'HH:mm:ss.SSS'Z'" o "yyyy-MM-dd'T'HH:mm:ss'Z'"`

lo cual trajo algunos inconvenientes a la hora del parseo.

En cuanto a potenciales mejoras me quedó en el tintero poder mantener la misma arquitectura de MVVM en toda la app. Poder hacer alguna pantalla más para mostrar

información más detallada de cada transacción a la hora de que el usuario seleccione alguna en el recyclerView. Poder desacoplar un poco más la clase encargada de los requests de Retrofit para evitar una futura GodClass.

4. **Inconvenientes**
Me surgió el inconveniente de no poder enviar bitcoins cuando me conectaba a los faucets que Google me brindaba, lo cual me obligó a investigar y a interiorizarme un poco sobre terminologías, páginas, documentaciones relacionadas a este mundo de los Bitcoins que no conocía. Me parece que estaría bueno poner como “Puntos/Consignas Bonus” para motivar a los programadores, acerca de “hacer la app landscape / portrait”, “Estimación de horas para la app”, “Alguna consigna/pantalla adicional”. De todos modos, el no incluir esto, deja en manos del programador qué decisiones extras tomar y eso también es algo provechoso para la experiencia de cada uno.

5. **Comentarios**
Disfruté de hacer la aplicación y de empezar a conocer un terreno nuevo que hasta el momento no conocía. Se puede ver que hay mucho campo por explorar y que es uno de los grandes desafíos de la actualidad. Me gustaría seguir investigando y conociendo esto aún más!