## New functionalities to add

### 1.- Search characters by name in the characters list

Antes de comenzar con la tarea número 1, voy a refactorizar un poco el código. (Tarea 4)

activity_home xml > Cambio la etiqueta tools:context que hace referencia al Activity HomeActivity con un nombre de paquete erróneo.

Creo una clase HomeToolbar para tratar de forma independiente al toolbar de la pantalla HomeActivity.

Creo una clase Constants para almacenar constantes globales. Creo una constante global para usar siempre la misma etiqueta cuando muestro un log de error. (Ejemplo en la case HomeToolbar).

En la clase HomeActivity borro todo lo relacionado con el toolbar y desde el onCreate() creo la clase encargada de trabajar con el Toolbar.

Saco los Fragments y sus adapters a sus clases separadas. Las pongo dentro de los paquetes 'characters' y 'houses'.

Saco la clase 'GotCharacterViewHolder' a una clase independiente. Para ello en el constructor, le pasamos el Activity del que viene y creamos getters para acceder a los atributos desde el adaptador.

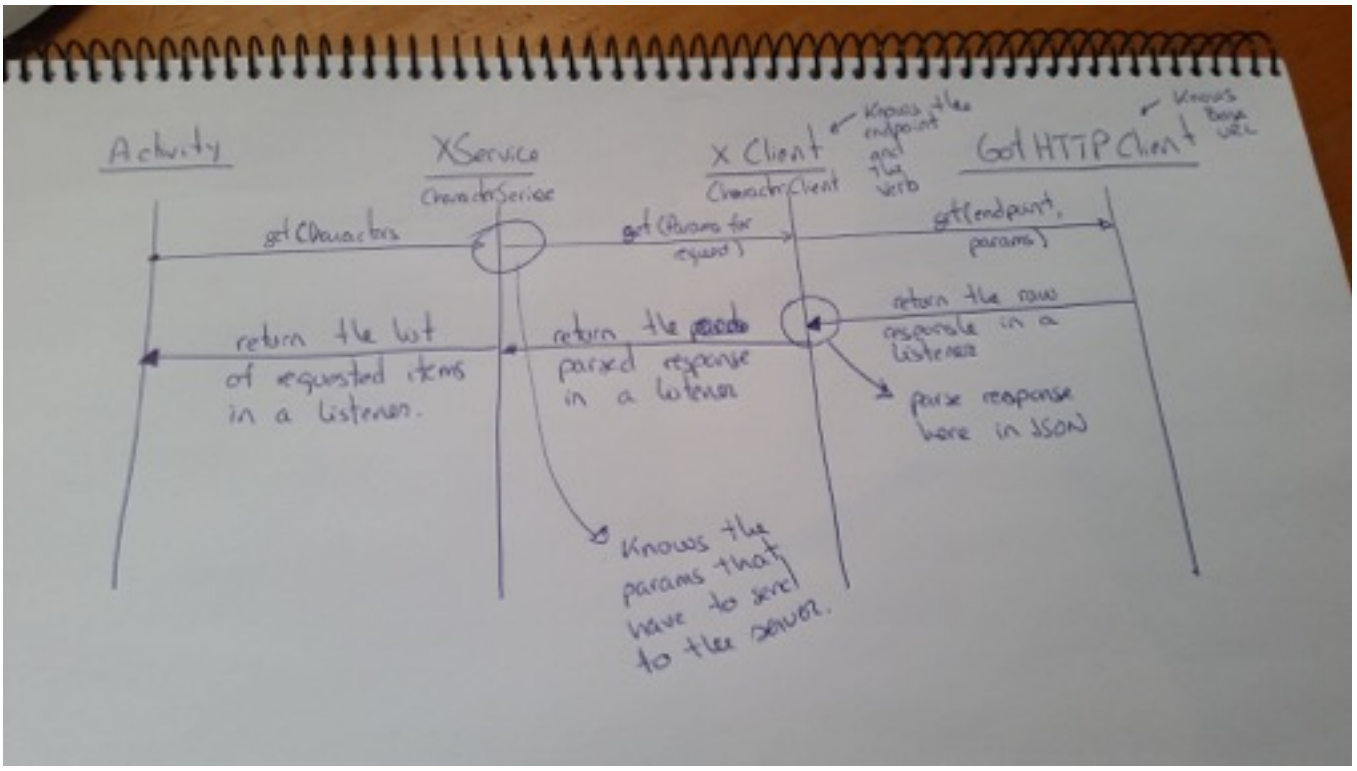
Dentro de la clase GoTHouseAdapter renombro la clase 'GotCharacterViewHolder' a 'GotHouseViewHolder' y la extraigo a una clase independiente. Igual que la anterior, en el constructor, le paso el Activity.

Extraemos también el ViewPager de HomeActivity a una clase independiente.

Creo una clase 'HomeViewPager' que tendrá la responsabilidad de setear el ViewPager que vemos en la pantalla Home.

Refactor de la forma de llamar al web service:

Comienzo con un pequeño diagrama de como va a funcionar el sistema.
(La explicación le he hecho en inglés debido a que los commits acostumbro a hacerlos en ingés)




Desde el activity llamo a un servicio que se encargará de delegar la llamada http a un cliente http y cuando obtenga la respuesta se encargará de setear los datos en el adaptador de listado de la vista.

El cliente se encargará de decidir a quien tiene que consultar la información solicitada. En este caso, ahora mismo se quiere hacer una petición HTTP, con lo cual se delega la responsabilidad de hacer esta petición al cliente http.

El cliente Http realiza la consulta y devuelve los datos al cliente que se lo ha solicitado. El cliente avisa al servicio y el servicio como he comentado anteriormente, avisa a la pantalla que cargará los datos.

Después de este refactor, comienzo a implementar la búsqueda de personajes por nombre en el listado de characters.

En la vista ‘fragment_list’ añado un elemento SearchView.
Después de varios intentos de aplicar el patrón ‘Decorator’ por composición, no consigo nada que funcione bien. He realizado varias pruebas en un proyecto aparte envolviendo un TextView en un ‘WrapperTextView’ al que en el constructor se le pase un elemento TextView para evitar la extensión y de esta manera poder decorar componiendo, pero no he conseguido que funcionase bien del todo, de modo que voy a decorar extendiendo.
Creo una clase que extienda del adaptador ‘GoTAdapter’ y la llamo ‘SearchableGoTAdapter’ que implementa la interfaz Filterable. En el getFilter() devuelvo una clase que extiende de Filterable a la que se le pasa en el constructor este adaptador en el que estamos (‘SearchableGoTAdapter’) para que se encargue de filtrar los elementos de la lista de personajes.
Ya sólo queda que en la clase ‘GoTListFragment’ implementemos la interfaz ‘SearchView.OnQueryTextListener’ para que en el onQueryTextChange() llamemos al getFilter del adaptador para filtrar los elementos de la lista.

Extraigo las cadenas de texto ‘Characters’ y ‘Houses’ al fichero de strings.xml.

**2.- Create a list of a characters by house, accessing to it by clicking a house image in the list of houses**

Me creo una nueva clase('DetailHouseActivity') que extienda de 'AppCompatActivity', creo su layout asociado, declaro el activity en el AndroidManifest.
Creo la clase encargada de trabajar con el Toolbar de este activity 'DetailHouseToolbar'.
En la clase GotHouseAdapter seteo el onclick de la imagen para que ejecute el intent que lanzará la actividad 'DetailHouseActivity' enviando el id de la casa y el nombre de la misma.

He tratado re reutilizar el fragment 'GoTListFragment' para evitar la duplicidad de código, ya que lo único que va a cambiar son los elementos que se van a mostrar. Después de varios intentos de mostrar el fragment dentro de este activity y no conseguirlo, voy a copiar y pegar sabiendo que esto no es lo que se debe hacer en la prueba. Dejará un TODO en el código para tratar de resolverlo antes de enviar el código.

Sé que lo que debería hacer mi código seria que Cuando se construya el Fragment se enviase en el constructor el listado de personajes/casas para que fuese más flexible y poder reusarlo. No obstante como no se que es lo que tengo que hacer para que el Fragmen funcione en la vista del activity lo dejo como está.

Lo que hago es en el servicio de obtener a los personajes, crear un nuevo método que llamaré 'getCharactersByHouse'.

Una vez hecho esto ya tenemos funcionando la pantalla.

**3.- Capability to work offline**

Para afrontar esta tarea, al primer problema al que nos tenemos que enfrentar es a que es lo que vamos a hacer con las imágenes. Po supuesto el resto de información es muy fácil de almacenar. Lo que voy a hacer es crear en los assets un fichero json con la respuesta del servidor. De esta forma, lo único que tengo que hacer es cambiar el 'GotHttpClient' por un 'GotAssetsClient'.

Ahora para afrontar el problema de las imágenes lo primero que voy a hacer es refactorizar la forma de obtener un Bitmap a partir de una url.

Una vez hecho esto empiezo a crear el cliente que leerá los JSON's de la carpeta assets.

Ahora me doy cuenta de que voy a tener que hacer más cambios de la cuenta, con lo cual, voy a volver a refactorizar el código para delegar la instanciación del cliente http en una factoría, de esta forma cuando quiera cambiar el cliente a un cliente que lea de los assets, solo tendré que hacer el cambio en la factoría.
Para ello hago que la clase 'GotHttpClient' implemente una interfaz común, creo una clase 'GotClientFactory' que instancia el cliente y listo.

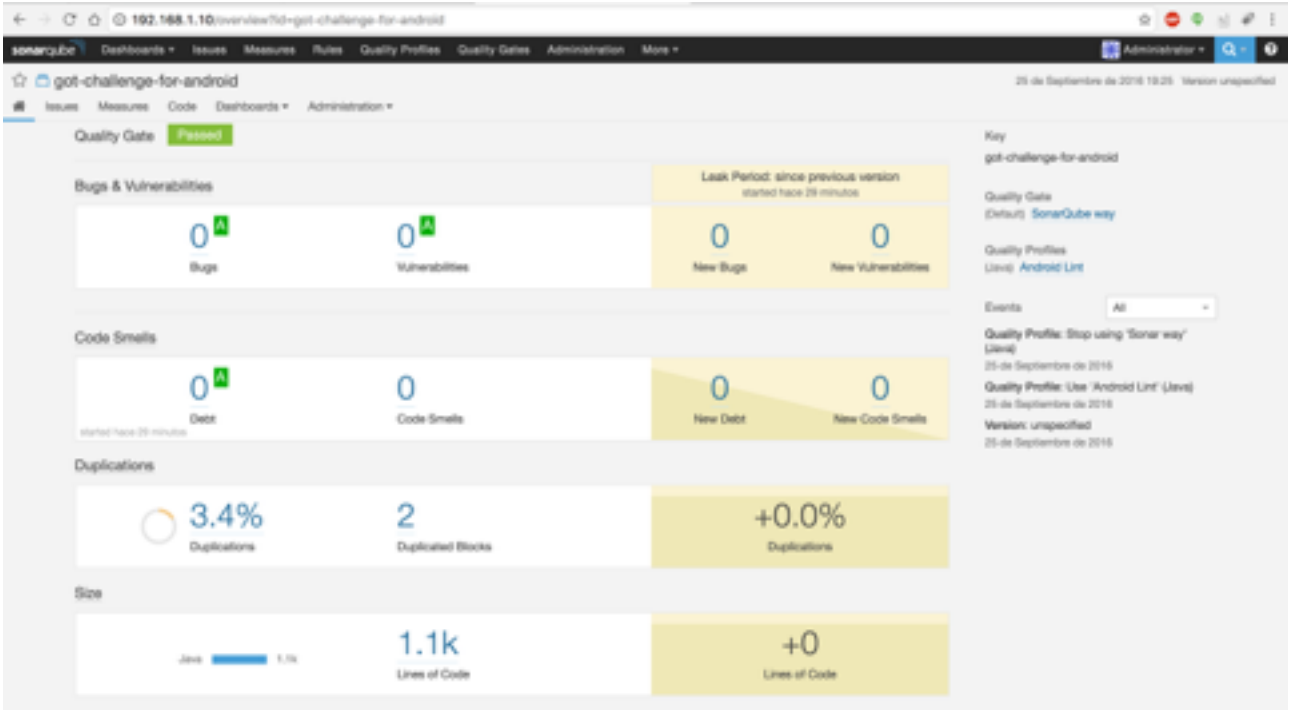
Hago algo parecido con las imágenes.
Creo un cliente que busca la imagen en la carpeta assets. Como no tiene mucho sentido que me descargue todas las imágenes una a una, lo que voy a hacer es poner una imagen que indique que la imagen no está disponible.
Ahora desde el ImageFactory en lugar de llamar al 'GotImageHttpClient' llamaré al 'GotImageAssetsClient' y este buscará la imagen en los assets del movil.

Sé que si se van a guardar imágenes para mostrar en la aplicación, estas deben ir guardadas en las carpetas drawablw-xxxhdpi, drawable-xxhpi, …

Una vez terminada esta tarea, para ver como se puede pasar de trabajar online a trabajar offline ver commits '0b3a4813fc5f6a47c17dcff25da35d194dd7c48e'(working online) y '4b0e3fadd9ca62708307115346ef90fc1e0fb4ba' (working offline) de mi rama master.

Después de realizar algunos cambios, ejecuto SonarQube para ver los posibles code semells que haya pasado por alto.

Reporte de SonarQube usando el perfil 'Android Lint'




## Some optional tasks to do:

**1.- Tests the main logic and a high level flows.**
No sabría como hacer un test de este tipo. He realizado algunas pruebas con Robotium, pero la principal razón por la que me gustaría entrar en idealista es trabajar con más gente de desarrollo android y aprender a hacer TDD en android entre otras cosas.

He realizado un pequeño test para la clase 'HomeSectionsPagerAdapter' usando 'Mockito'.

**2.- Add transitions between list and detail**

Lo he intentado de varias formas, pero no consigo ponerlo a funcionar. En otras ocasiones lo que he hecho ha sido simplemente sobreescribir la transición después de ejecutar el startActivity usando la función 'overridePendingTransition' y pasándole las referencias a las animaciones de entrada y salida.

Estas animaciones deben ir almacenadas en $workspace/app/main/src/res/anim.
Son ficheros xml en el que se indica como se debe comportar la animación.
Por ejemplo:

```
<set xmlns:android="http://schemas.android.com/apk/res/android" >
   <translate
        android:duration="@android:integer/config_mediumAnimTime"
        android:fromXDelta="-100%p"
        android:toXDelta="0" />
</set>
```

En el que se indica que la transición se mueve en el eje X y pasa de no estar visible (-100%) a colocarse en la pantalla al completo (posición 0).

### 3.- Add parallax effect into detail page

Para añadir el efecto parallax, me creo un activity (‘ParallaxDetailActivity’ > Declarándolo en el ‘AndroidManifest’) que extienda de ‘DetailActivity’ y que se encargue de crear el efecto parallax seteando la posición Y de la imagen cuando se hace scroll en la vista.

Una vez hecho esto, basta con cambiar el nombre del Activity al que se llama cuando se quiere abrir el detalle del Activity.

(Ver commit f197c241f41070d842b051876be3fca2685993a9).

Notas:
Soy consciente de que la organización de los ficheros es muy mala, no obstante, he optado por no reorganizar los ficheros a última hora antes de entregar, con el fin de que se pueda ver el histórico de cada uno de ellos.

Adjunto pequeño diagrama de como se relacionan las clases (no están todas las clases, pero sirve para hacerse una idea de donde se encuentra qué).