Version 1.0

2018-Aug-08



Interview Feedback

Homework

Presenter: Arturo Sanchez

Mobile Apps Company

3301 Windy Ridge Parkway, Suite 400, Atlanta, GA 30339

# **Interview Feedback**

## Introduction

### Elevator

My name is Arturo Sanchez Chavarria.

I have 6 Years of I.T. experience in Software Development, 5 years android development experience and five Android Apps in Play.

I have experience developing mobile applications throughout the entire software development lifecycle (SDLC) right from collecting requirements until implementation in the Play Store. All of that using Agile Scrum and considering the design patter such as MVC, MVP and MVVM depending on the client requirements.

I also have worked with an internal QA team on system, performance, and acceptance testing.

I have experienced integrating apps with web services using Retrofit, OkHTTP.

I have also worked with third-party APIs and web services like Google, Facebook, Twitter and YouTube Player.

### Previous Experience

I am currently working with Delta Airlines as a Senior Android Developer.

I am using Android Studio as the primary IDE to develop, test and deploy the Android application.

My responsibilities are:

* Refactoring code
* Migrate from MVC to MVP
* Transition to RxJava and Retrofit.
* Fix memory leaks.

They were using MVC as the designing patter and now we are migrating it to MVP.

I get to improve performance, security tracking.

I also work on fixing memory leaks using Leak Canary.

For testing, we implemented automated testing with Espresso.

Some of the features implemented are:

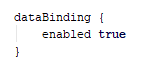
* Usign Zxing for scanning to provide various user features in airport.
* Implemented various third-party technologies to optimize the app functionality including:
  + Firebase Cloud Messaging
  + SQLite
  + Glide
  + Shared Preferences
  + MixPanel
  + EventBus
  + RxJava

We used Bitbucket to manage Git repositories and versioning control.

## Questions Asked

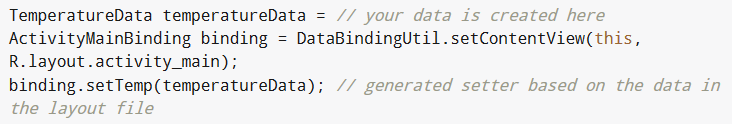
### What is dataBinding? How do we use it?

Data binding allows synchronizing your user interface with your application model and logic.

1. You first need to enable data binding in your Gradle
   1. 
2. To use it we create a <data> tag on our activity layout
   1. Create variables tags in which we specify the name and the type.
   2. The type must be the class we are trying to bind with our activity.
3. Set the properties of your controls (views) with the specified properties
   1. Example.- text=”@{myDataVariableTypeClass.Property}”
4. Build your project
   1. Sometimes you have to restart Android Studio
   2. This creates a Binding object
      1. {ActivityName}Binding
      2. Example MainActivityBinding
5. Set Content View and data

Example.-





### How to update the view reactively with DataBinding?

1. You create an observable property
2. Create a TextWatcher on your Model
3. You override the methos on your TextWatcher
4. You set your observable property to the editable value of one of the methods of your TextWatcher.

### How to pass an object from xml to the ViewModel?

1. You define an event (Example.- click event)
2. Define an object in the event
3. Pass the object as a parameter for the event call

Example.-



“theView” is the object to pass from the xml.

### What are different types of encryption? Elaborate.

**Symmetric** — the oldest and best-known technique. The encryption key and the decryption key are the same. Also it is generally categorized as being either Stream Cipher or Block cipher.

The most common Symmetric AES — the Advanced Encryption Standard (AES) is the algorithm trusted as the standard by the U.S. Government and numerous organizations.

**Asymmetric** — a modern branch of cryptography. Also known as public-key cryptography in which the algorithms employ a pair of keys (a public key and a private key) and use a different component of the pair for different steps of the algorithm.

The most common Asymmetric algorithm is RSA — a public-key encryption algorithm and the standard for encrypting data sent over the internet.

### How to generate keys in Android?

1. You need a KeyStore and a KeyPairGenerator object.
   1. In both objects, you must set them from the getInstance() method.
   2. For the KeyPairGenerator you must specify an algorithm and a provider.
2. You then specify a certificate for the KeyPairGenerator object
3. Initialize the KeyPairGenerator object.
4. Call the generateKeyPair() method from the KeyPairGenerator object.

### How to encrypt and decrypt the data using those keys?

1. Create a Cipher instance
   1. Define a Transformation for the instance.
2. To Encrypt
   1. Call the init() method from the Cipher instance passing the Encrypt Mode and the key as parameters.
   2. Get the bytes from the Cipher calling the doFinal() method passing the plain text as a parameter.
   3. Convert the byte array to string using the Base64 class encodeToString() method to get the encrypted text.
3. To Decrypt
   1. Call the init() method from the Cipher instance passing the Decrypt Mode and the key as parameters.
   2. Call the decode() method from the Base64 class and pass the cipher text as a parameter to get the encrypted bytes.
   3. Call the doFinal() method from the Cipher to get the decrypted bytes.
   4. Create a new String passing the decrypted bytes as a parameter and get the plain text.

### What is CertificatePinning? How does it work?

Pinning is the process of associating a host with their expected X509 certificate or public key. Once a certificate or public key is known or seen for a host, the certificate or public key is associated or 'pinned' to the host. If more than one certificate or public key is acceptable, then the program holds a pinset. In this case, the advertised identity must match one of the elements in the pinset.

Typically certificates are validated by checking the signature hierarchy; MyCert is signed by IntermediateCert which is signed by RootCert, and RootCert is listed in my computer's "certificates to trust" store.

Certificate Pinning is where you ignore that whole thing, and say trust this certificate only or perhaps trust only certificates signed by this certificate.

So for example, if you go to google.com, your browser will trust the certificate if it's signed by Verisign, Digicert, Thawte, or the Hong Kong Post Office (and dozens others). But if you use (on newer versions) Microsoft Windows Update, it will ONLY trust certificates signed by Microsoft. No Verisign, no Digicert, no Hong Kong Post office.

Also, some newer browsers (Chrome, for example) will do a variation of certificate pinning using the HSTS mechanism. They preload a specific set of public key hashes into this the HSTS configuration, which limits the valid certificates to only those which indicate the specified public key.

### How do we test Android apps?

Small tests are unit tests that you can run in isolation from production systems. They typically mock every major component and should run quickly on your machine.

Medium tests are integration tests that sit in between small tests and large tests. They integrate several components, and they run on emulators or real devices.

Large tests are integration and UI tests that run by completing a UI workflow. They ensure that key end-user tasks work as expected on emulators or real devices.

### What is TDD? What are the benefits of TDD?

Test-driven development (TDD), is an evolutionary approach to development which combines test-first development where you write a test before you write just enough production code to fulfill that test and refactoring.

The power of TDD lies in the ability to advance in small steps when necessary. It allows a programmer to focus on the current task and the first objective is usually to pass the test. Initially, exceptional cases and error handling are not considered. These are implemented after the main functionality has been reached. Another advantage is that, when used correctly, it ensures that all written codes are covered by a test. This can give the programmer a higher level of confidence in the code.

### What are @Before and @After annotations?

The @Before annotation indicates that this method must be executed before each test in the class, so as to execute some preconditions necessary for the test.

The @After methods declared in superclasses will be run after those of the current class, unless they are overridden in the current class.

### How to mock objects for testing?

1. Add dependencies to Gradle.
   1. Mockito dependencies already with the latest gradle
2. Add the mock annotation to the object you want to mock
3. Set it before you use it.

### How to verify if a method was invocated in a mock object?

Call the verify method after the call of the object.

verify(mockObject).methodToVerify();

### What are the main components in Espresso?

Espresso has 4 main components which are:

Espresso – It is the entry point for any interaction with views (using onView() and onData()). It also exposes certain APIs that are not specifically linked with any view, like pressBack().

ViewMatchers – They are a collection of objects implementing the Matcher <? super View> interface. One or more matchers can be passed to the onView() method to find a view inside the current view’s hierarchy. In case the view that we are trying to match is not present in the view hierarchy, Espresso throws a NoMatchingViewException.

ViewActions – They are a collection of actions (ViewAction) that can be passed as an argument to the ViewInteraction.perform() method to perform some action on the view, such as click().

ViewAssertions – They are a collection of assertions (ViewAssertion) that can be passed as an argument to the ViewInteraction.check() method to assert some state of the view. Most commonly used ViewAssertion is the matches assertion, which uses a ViewMatcher to assert the state of the currently selected view.

### What does an activity test rule do?

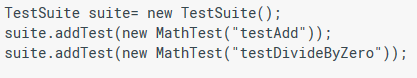
This rule provides functional testing of a single Activity.

To test an activity, you use the ActivityTestRule class provided by the Android Testing Support Library.

This rule provides functional testing of a single activity. The activity under test will be launched before each test annotated with @Test and before any method annotated with @Before. It will be terminated after the test is completed and all methods annotated with @After are finished. The Activity under Test can be accessed during your test by calling ActivityTestRule#getActivity().

### How to run multiple test classes?

A TestSuite is a Composite of Tests. It runs a collection of test cases. Here is an example using the dynamic test definition.



Alternatively, a TestSuite can extract the tests to be run automatically. To do so you pass the class of your TestCase class to the TestSuite constructor.



This constructor creates a suite with all the methods starting with "test" that take no arguments.

A final option is to do the same for a large array of test classes.



### How to create Custom views in Android?

1. Create a class that extends from View
   1. Can extend from an specific view such as Button, Linear Layout.
2. Implement the Constructor methods.
3. Add functionality as needed.

### What is the lifecycle of a view?

1. onAttachedToWindow()
2. measure()
3. onMeasure()
4. layout()
5. onLayout()
6. dispatchDraw()
7. draw()
8. onDraw()

### How to add custom attributes to a view?

To add a built-in View to your user interface, you specify it in an XML element and control its appearance and behavior with element attributes. Well-written custom views can also be added and styled via XML. To enable this behavior in your custom view, you must:

* Define custom attributes for your view in a <declare-styleable> resource element
* Specify values for the attributes in your XML layout
* Retrieve attribute values at runtime
* Apply the retrieved attribute values to your view

### How to use a custom listener for a compound view?

There are four steps to using a custom listener to manage callbacks in your code:

1. Define an interface as an event contract with methods that define events and arguments which are relevant event data.
2. Setup a listener member variable and setter in the child object which can be assigned an implementation of the interface.
3. Owner passes in a listener which implements the interface and handles the events from the child object.
4. Trigger events on the defined listener when the object wants to communicate events to it's owner

### How to analyse user behavior to find trends?

Creating an analytics implementation to track the users behavior and have statistics on what are the trends used in the app.

### How to track duration for events?

1. Create an instance of your analyrics API
2. Call the timeEvent() method from the API in the beginning of the event.
3. Call the timeEvent() method from the API in the end of the event.
4. Compare time with stadistics.