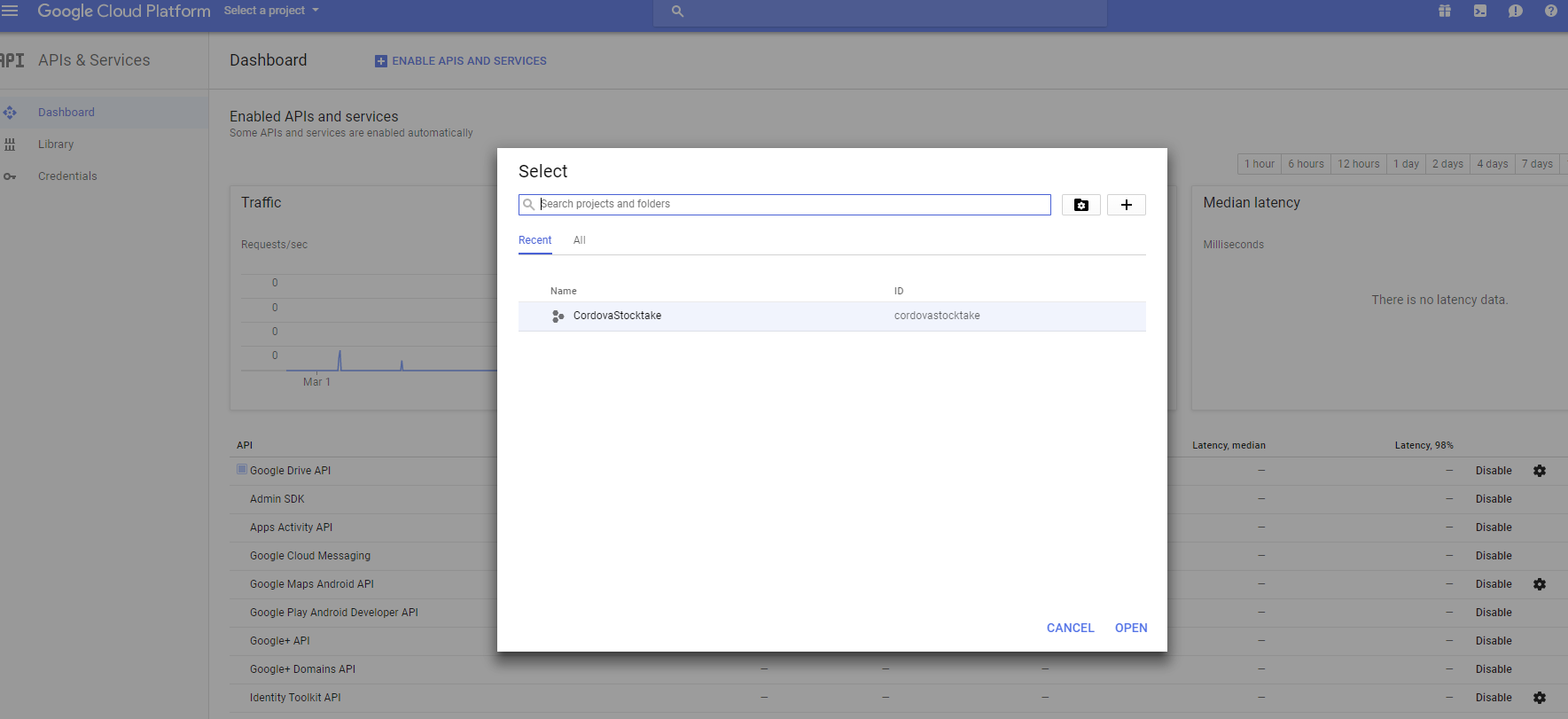
Google API for cordova/phone app

So, this process is lengthy and may only be applicable and may leave out some parts that could be needed but this is a rough breakdown of the process I have completed to get signin working on an app basis (this is roughly the same for the coural app and other google signin projects but I needed an extra step to allow all the API usage for google drive syncing).

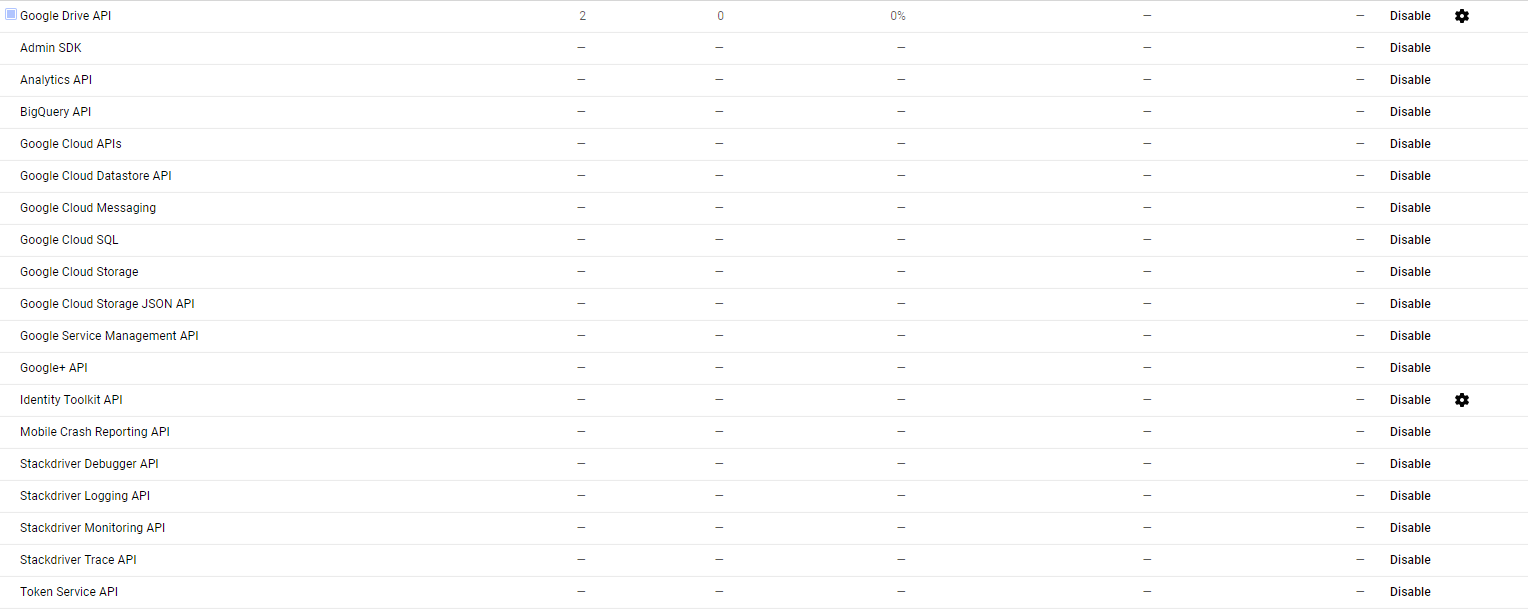
So, first step is an account. Make any google account – my original one was [StocktakeTestSolution@gmail.com](mailto:StocktakeTestSolution@gmail.com) with the password: Solution  
This is simply a thing to monitor, it won’t need to be signed into as the signin just uses this acc as the intermediary from what I can gather.

So, once an account is setup you will need to go to “console.cloud.google.com” which and sign in with your new account created prior.

Once signed in, it is important to set up a project on the console (see below - + button on the right):

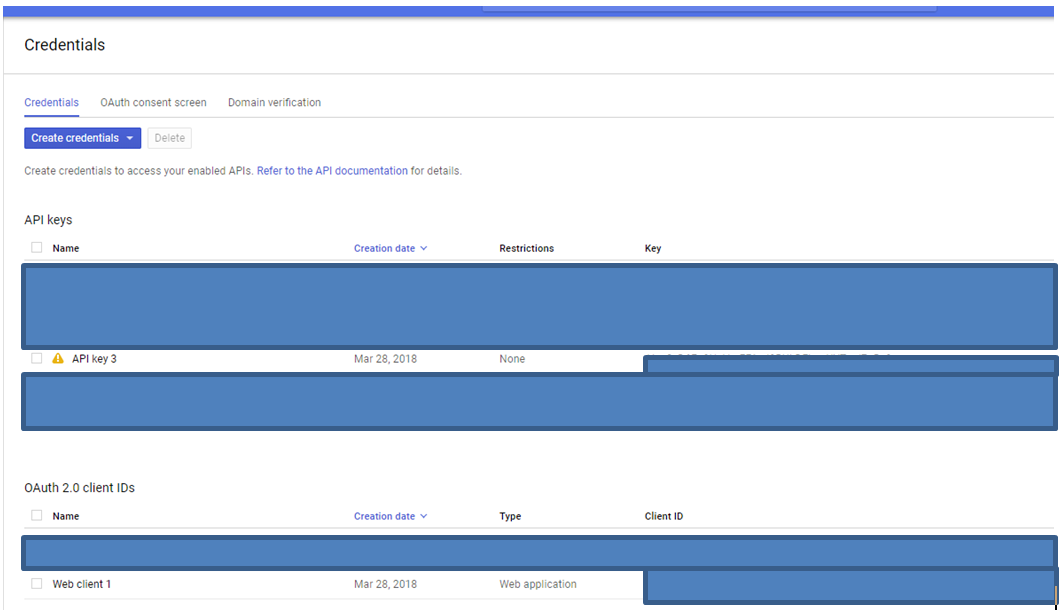
  
Once one is created, onto the important things:

Now, THE MOST IMPORTANT THING THAT WILL SAVE YOU A LOT OF TIME: Do not make credentials till AFTER you add API’s you are going to use (primarily the google sign in one but others too). If you make the credentials then add API’s, the credentials you created will not have use of the API’s you selected to use making them useless and will give you random errors that are utterly useless.

Now, after adding and enabling all the API’s you want – I added the following to my list, the red ones, and the others are default (this allowed signin and drive access).

If you don’t have all the ones above just add them, no harm done by having extras really as the apps won’t use it if it’s not needed anyway but should only need the 4 highlighted (maybe google+ API if not there by default).

Now the API’s are added, you need some credentials to access your newly setup API’s.   
So, first you need an OAuth key and an API key (for testing mainly – see the code at the end of this doc for some test code to try out the setup – the API key is not used in the app as you sign in manually but still, its testing stage).  
Go to the Credentials screen and click the OAuth consent tab and set the name.  
You also need to create 2 keys, an API key and an OAuth key (for websites NOT ANDROID – ANDROID IS FOR NATIVE APPS NOT THE CORDOVA STUFF WE NORMALLY USE AND IS USED CURRENTLY). Create credentials button then API key, and OAuth key (FOR WEB APPLICATION). The secret is ignorable, that is not needed but the two keys are important (I hid them below just for safety sake).



Using the code at the end of the file, you can test that you set up so far correctly (it will not work in the app till the next step but baby steps not baby leaps).

Now that we have gotten through the nice simple stuff online, time to hit the cmd because who needs this functionality included in easy to find places, lets do it all in command line using stuff you have to google. On google, its not in the google docs from what I found, but google will still save you since stackoverflow will get you the answer (see link on the right for how to set up the second part of the keystore as example of the glory of 1000’s of minds).

So, now you have tested the thing and it works (I assume, these instructions are ok but if its not working get the previous file running first else you will have problems and have no idea where to start – 1 change then test etc..) it is time to get it working in a Cordova app or similar.

First, you need navigate to the jdk you have installed (or install, please have a jdk installed – link below):

C:\Program Files (x86)\Java\jdk1.8.0\_161\bin   
as an example  
  
Again, cmd prompt (administrator mode needed), in the bin folder of the jdk. There is a little thing called the keytool in this location which is key (pun not intended) to this process. So, in the cmd in this directory, enter the following (replacing [ALIAS] with the name you want for your keystore (name of app works ez)):  
  
**keytool -genkey -alias Stocktake -keyalg RSA -keystore keystore.jks -deststoretype pkcs12**

Go through and add a password (can just be the name or proper, just document it if it isn’t the name. Its encrypted in the keystore so its gone if you forget it and its annoying to repair). Then just skip all the naming except the person name could be set to the app name if you want for consistency.

Once done, say yes to it and keystore.jks should be created. Copy this into the project you are making at same level as the config.xml etc…   
  
Now you have it signed (nearly), you just need it to be able to signin from an app in a browser which currently it will not let you do because it thinks you are using android to do it because technically you will be but will need the web client OAuth to login. Very annoying but works if you follow the following :

<https://developers.google.com/mobile/add?platform=android&cntapi=signin&cnturl=https:%2F%2Fdevelopers.google.com%2Fidentity%2Fsign-in%2Fandroid%2Fsign-in%3Fconfigured%3Dtrue&cntlbl=Continue%20Adding%20Sign-In>

(if this link works, it explains more of the process really <https://stackoverflow.com/questions/34367870/where-do-i-get-a-google-services-json>)

The main link will take you to where you will set up your app for google signin. First, you need to copy in what your package name is (e.g. com.solutionm.myappnamehere). Then you will need to add google signin into it, and the analytics (it works using it, had some issues when not using analytics so safety first). This will require the SHA1 key which comes from the keystore which can be extracted from the keystore by doing the following:

So, go to where you put your keystore.jks and open a cmd. Then, run this:  
**keytool -exportcert -list -v -alias Stocktake -keystore keystore.jks**

Now you have the SHA1 key in there that you can Ctrl+M, Enter, the thing out and use. Now you have set up the keystore and the google signin side for the app.  
  
If you did it right, you should now be able to run your app with the Web Client ID set right.

If there are issues, you can repeat the key creation again, or repeat the setup of things as much as you want, or even start from scratch, there is no issue with it unless its in production. Don’t reset the keys of anything in use, it will kill it entirely and stop it being able to run anything until entire updated with new keys. I would also recommend not changing any settings post creation – just safer that way. Make sure you add any API’s you may want before creating keys (can recreate here if you want but not in production) just to make it easier later.

Code for the html is on the next page.

# Code Time – Javascript testing and python SimpleHTTPServer

To run this you will need python installed and have probably admin powers so you can run the commands set below.

First, copy the code below and save into a HTML file, name is not important but I called it “GoogleDriveTest.html” and dump it in a folder on your desktop called “GoogleDrive”. Next, set the OAuth and API key in the file to the ones you have set up in your credentials on the console on web. Now, shift right click in the GoogleDrive folder and create command window or open cmd and navigate to that folder. In the cmd window, enter “python -m SimpleHTTPServer 8000” which will create a sudo server in the folder you run it allowing you to run the html/javascript you have in that GoogleDrive folder.

So, you can go to localhost:8000 in your browser (Chrome please) and test the thing by selecting the name of the file you created from the list. Can inspect to check errors etc…. If it doesn’t work at this stage, godspeed, goodluck.

To open the thing below, just double click and will open it fully (it only shows the first chunk since word is meh). The HTML file is also just in the folder with this.

