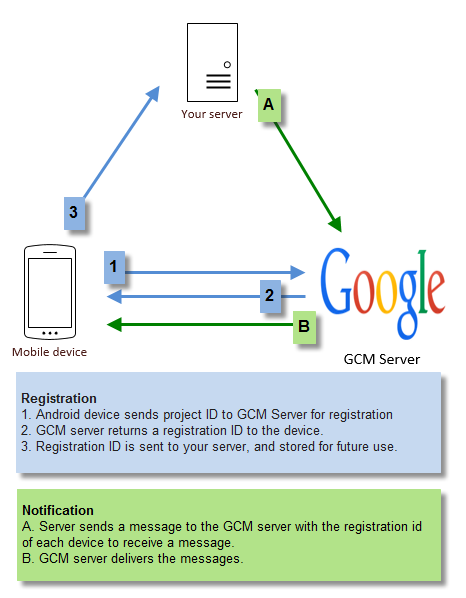
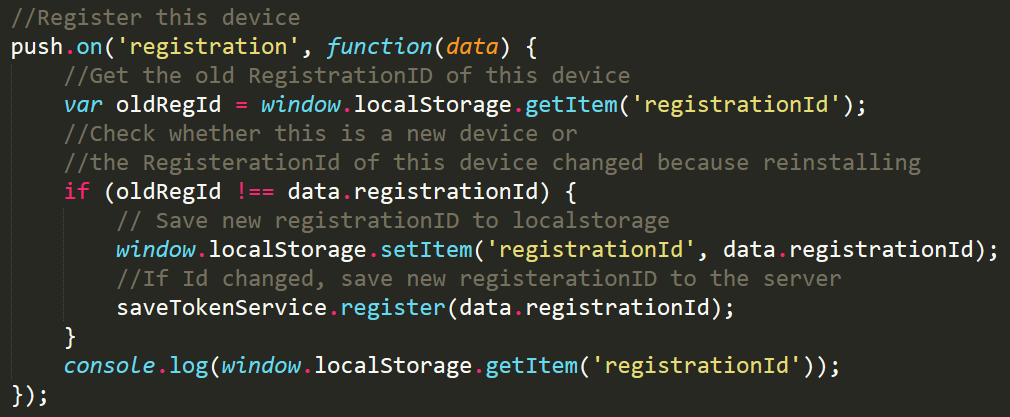
Push Notification to Android device through GCM



On Client side:

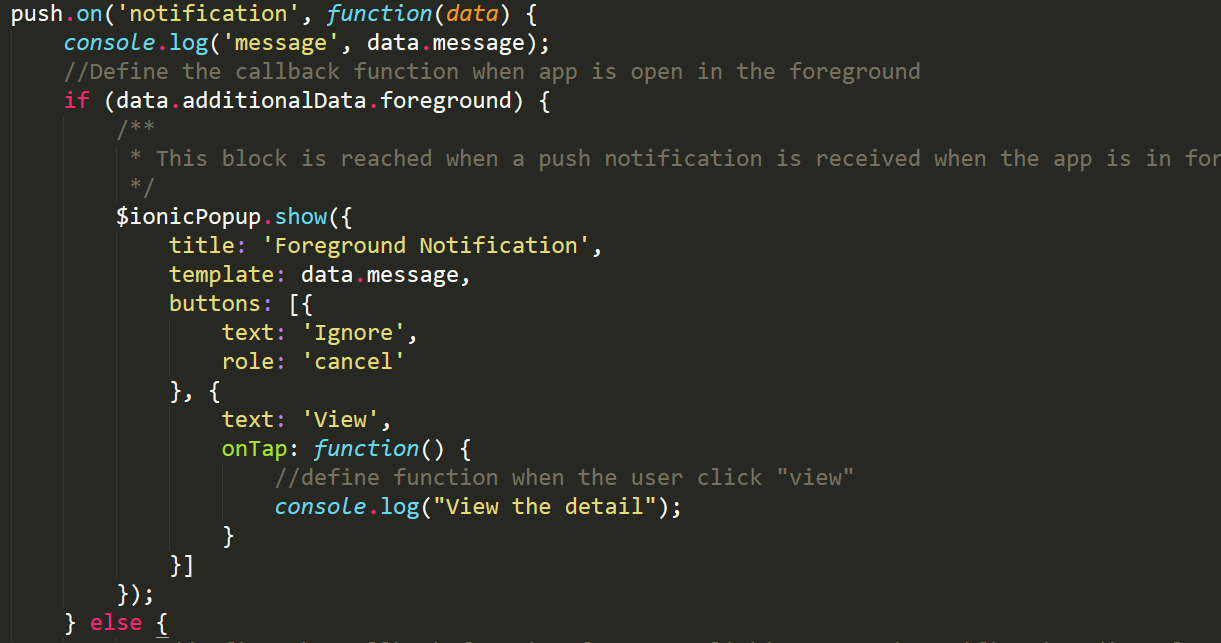
1. In our project, we first need to initialize the PushNotification service:

2. Then register this device on GCM server to get the registration Id and store this registration id to our server:

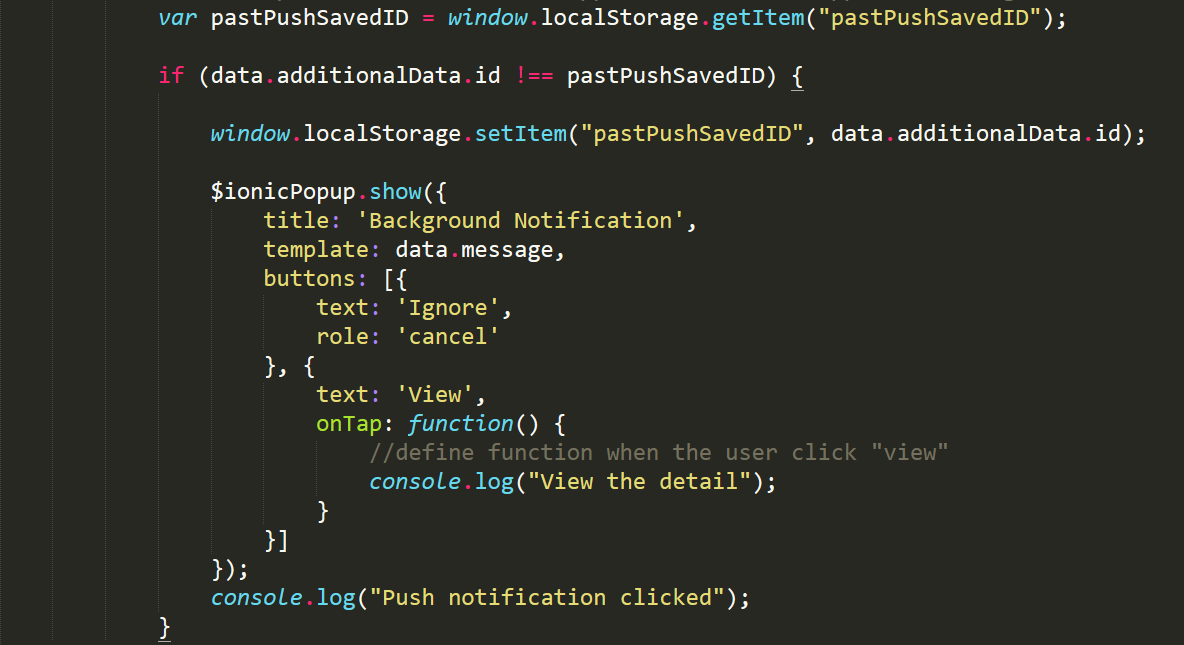


Check the localStorage whether there is a registrationId stored to judge whether this is a new device or whether the registrationId has changed because of reinstall. If yes, set the ‘registrationId’ in locaLStorage and store the ‘registrationId’ in our server. I build a saveTokenService to send the registrationId to server.

3. Define the callback function for user clicking on push notification: I divided the situations into two cases. First when the notification comes, the app is open is the foreground:



Second, when a push notification is tapped on and the app is in background:

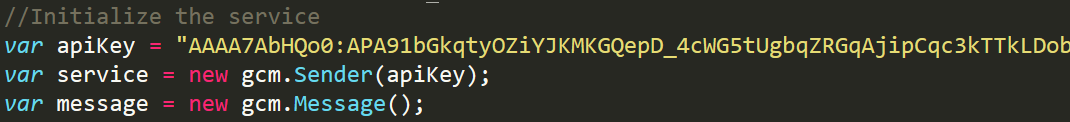


In every message sent from the server, I added a unique id to sign them. I use the if statement to determine whether this notification has been handled to avoid calling the function twice.

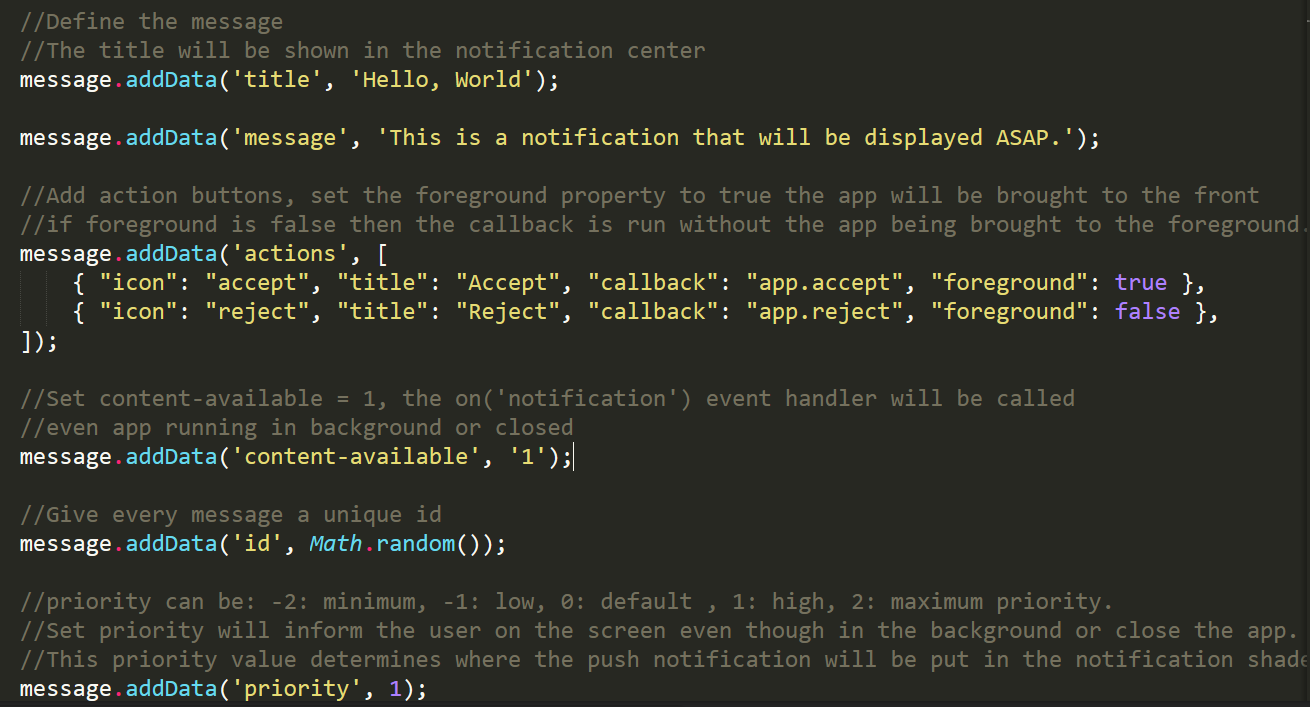
To display the notification, I used the Ionic Popup service.

On Server side:

I built the server with express.js and used node-gcm module. To send a message from server, first need to initialize GCM service with the apiKey of your project:



Then define the message payload:



Important message payload:

1. Title and message:

Without title and message, the notification will not appear in the notification center.

2. Content-available:

Set content-available = 1, the on('notification') event handler will be called even app running in background or closed when the user clicked one the app icon.

3. Priority:

Priority can be: -2: minimum, -1: low, 0: default, 1: high, 2: maximum priority.

Set priority will inform the user on the screen even though in the background or close the app. This priority value determines where the push notification will be put in the notification shade.

4. Stacking:

By default when using this plugin on Android each notification that your app receives will replace the previous notification in the shade. If you want to see multiple notifications in the shade you will need to provide a notification ID:

message.addData('notId', 1);

5. Inbox Stacking:

We can also stack our notifications with the inbox style to have up to 8 lines of notification text in a single notification.

Notification 1:

message.addData('style', 'inbox');

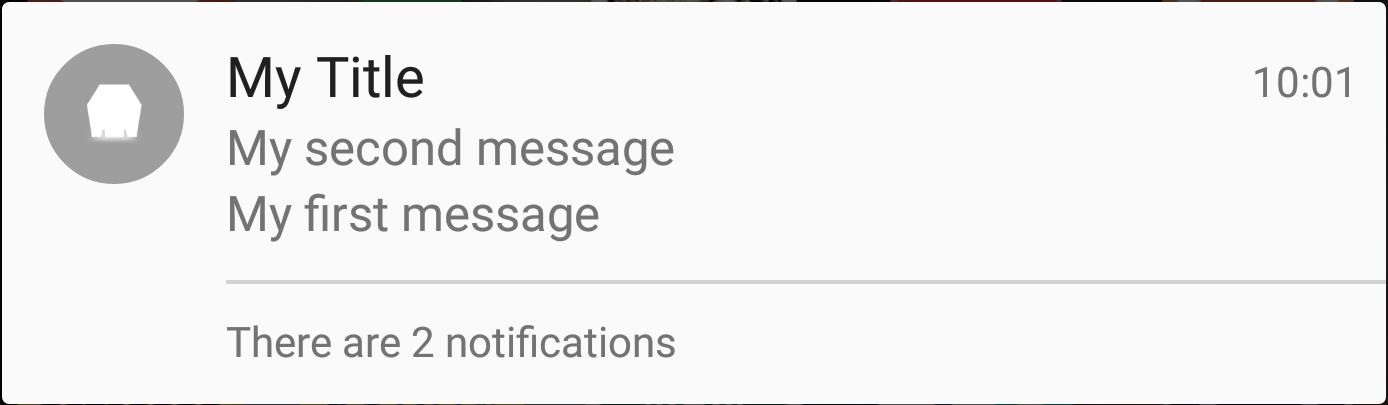
message.addData('summaryText', 'There are %n% notifications');

Notification 2:

message.addData('style', 'inbox');

message.addData('summaryText', 'There are %n% notifications');

The notifications will look like:



5. Action Buttons:

message.addData('actions', [

{ "icon": "emailGuests", "title": "EMAIL GUESTS", "callback": "app.emailGuests", "foreground": false, "inline": true},

{ "icon": "snooze", "title": "SNOOZE", "callback": "app.snooze", "foreground": false},

]);

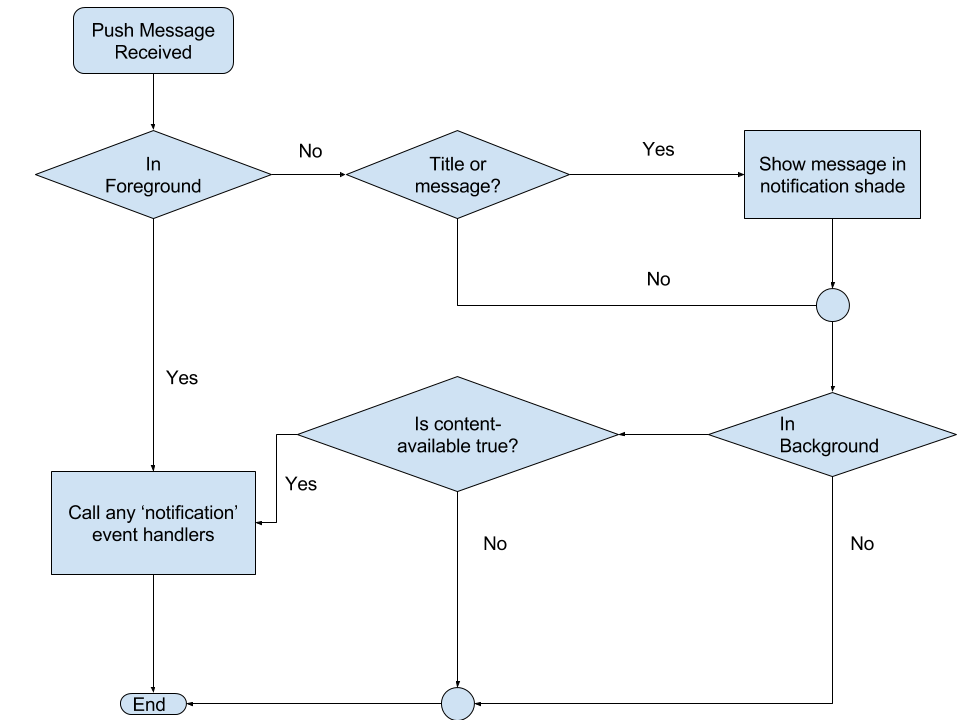
Our notification can include a maximum of three action buttons. If user clicks on the main body of the notification our app will be opened. If they click on either of the action buttons the app will call the callback function, and if there isn't an event will be emitted with the callback name.

If we set the foreground property to true the app will be brought to the front, if foreground is false then the callback is run without the app being brought to the foreground.

At last, set the retry times and send notification to all the devices which have registered:



This diagram shows what happens when a push message arrives on the device:



Push Notification to IOS device

In the client side, push notifications to IOS devices is similar to Android. Just change the initialization of PushNotification:



In the server side, we can send a notification through APNS or set up APNS SSL certificate on GCM and push notifications through GCM.