Push Notifications Opal

Table of contents:

Installation

Instructions iOS

https://firebase.google.com/docs/cloud-messaging/ios/certs

1. Go to the xcode project build.
2. Click on the project
3. Go to the General tab
4. Go down to signing, choose your development team, if not team add your account to xcode, then sign using your account. (Make sure it allows you, might tell you that the bundle id is not unique, in which case, create a unique bundle id. I.e. the bundle id is (com.cordova.helloworld string))
5. Go to the capabilities tab.
6. Scroll to Push Notifications.

7. Turn them on.
8. Go to https://developer.apple.com/
9. Sign with your account.
10. Go to: Account > Certificates, Identifiers & Profiles> App IDs
11. Click on the App ID that matches the bundle Id
12. Click on edit
13. Go to push notifications> Create Certificate
14. Follow Instructions.
15. Once the .cer file is saved in your keychain. Go back the safari development page.
16. Navigate to, Certificates, Identifiers & Profiles>Provisioning Profiles>Development
17. Select correct AppID
18. Select your account.
19. Select all the devices you want to test on. (In this case, your phone and such, anytime you build the app on a device using xcode it will add device to the list)
20. Enter name for provisioning profile, Opal Health, or just Opal.
21. Download and double-click.
22. Restart xcode
23. Open Opal project in Xcode, check capabilities, Push notifications should be on and have to check marks. That indicates that everything is fine. If not, contact me.
24. Get the .cer file and open in Keychain
25. Select file and key.
26. Export it as p12 file to a folder.
27. Navigate to folder
28. Run the following command:
    ```
    openssl pkcs12 -in pushcert.p12 -out pushcert.pem -nodes -clcerts
    ```
29. Copy .pem file to push notifications folder.
30. You are all set up! This was the hardest part :)

Adding iOS Devices

https://developer.apple.com/library/content/documentation/IDEs/Conceptual/AppDistributionGuide/MaintainingProfiles/MaintainingProfiles.html

In case you need to add more devices later:

There are two ways, through xcode and through the Apple page.

Through Xcode:

1. Plugin device
2. Run app on device

Through Apple:

1. Plugin device.
2. Go to iTunes
3. Click on device, then click on Serial Number:XXXXX, until UUID shows up. Right Click, Copy
4. Go to https://developer.apple.com/
5. Go to: Account > Certificates, Identifiers & Profiles> Devices >All
6. Add device, you will need the name and the UUID for the device.

Instructions Android

1. Navigate to https://console.firebase.google.com/
2. Select Opal project
3. In the Top Menu click Gear Icon> Project Settings
4. Click on the Cloud Messaging tab
5. Copy Server API key into the config.php file. Replace, ANDROID_ API_KEY mapping.
6. Copy the SenderID and Proceed to the App Section of the set up.

Instructions Cordova App

1. Remove Push Notifications plugin via:

```
cordova plugin rm phonegap-plugin-push
```

2. Re add via:

```
cordova plugin add phonegap-plugin-push@1.8.1 --variable SENDER_ID="XXXXXXX"
```

NOTE: The SenderID refers to the number in your Project Settings>Cloud Messaging in Firebase.
Use version 1.8.1, the latest version still has problems which they are working on

3. Build project again for both devices. It's all set up!

(Note: For Opal, the push notification management is done at the file mainController.js, this needs to be slightly modified to refresh the patient information IF the device is authenticated and the user is inside the tab view. If not, it's fine, once they login, that information should be loaded.)

There is also the issue of what happens when the device is loading the information and a push notification arrives, technically the request has already gone through before that information is posted. A flag has to be set to refresh that information again once they arrive to the tab view.

Installation Push API libraries

1. Copy .pem file obtained from Key Chain.
2. Go to config.php file, set the following parameters:
    a. API_KEY representing the Server key in the Firebase Project.
    b. CERTIFICATE_FILE representing the absolute, or relative path of .pem
    c. CERTIFICATE_PASSWORD representing the password use to encrypt the .pem file.
    d. Database Credentials.
3. Make sure db.inc.php is correctly connecting to Database.

Testing Installation

1. **Getting RegistrationID:**

    Through the database:
    1. Login as a patient.
    2. Find that patient in the Patient table of the database, Go to the PatientDeviceIdentifier table.
    3. Check the RegistrationId that matches your iPhone and patient. (May need to clear table completely or use the UUID for your device.) Check add more devices section for how to find the UUID.

    Through the device:
    1. Open the inspect device for safari or chrome (depending on whether the build is for iOS or Android).
    2. Refresh the app.
    3. Look for RegistrationId, copy and paste the Id.

2. **Running Script**
    1. Replace the registration Id in the testPushApi.php file.
    2. Run the testPushApi.php
    3. Check your phone!!

OR

    1. Navigator to <attachment-push-api-path>/test-website
    2. Replace URL to match the domain url in listener.js file
    3. Try the different buttons and see response.

Documentation

To test use website to look at the Push Notification Post Interface and results. Make sure the replace the url in the listener.js file.

Post API

Post request API calls the HospitalPushNotification functions. It makes the Push API available to websites and other scripts.

For each call, a **NotificationType** needs to be defined via **$_POST["NotificationType"]**

Calls

Here are the different NotificationTypes:

- **RoomAssignedNotification**
  Post Parameters:
  $_POST["PatientId"],
  $_POST["RoomLocation"],
  $_POST["AppointmentAriaSer"],
  Response:
  JSON Object with the results from HospitalPushNotification::sendCallPatientNotification

- **SendNotification**

Post Parameters:

$_POST["DeviceType"],
$_POST["RegistrationId"],
$_POST["NotificationTitle"],
$_POST["NotificationDescription"].

Response:

JSON Object with the results from HospitalPushNotification::sendNotification

- **SendNotificationMultipleDevices**

Post Parameters:

$_POST["Devices"],
$_POST["NotificationTitle"],
$_POST["NotificationDescription"].

Response:

JSON Object with the results from HospitalPushNotification::sendNotificationToMultipleDevices

- **SendNotificationUsingPatientId**

Post Parameters:

$_POST["PatientId"],
$_POST["NotificationTitle"],
$_POST["NotificationDescription"].

Response:

JSON Object with the results from HospitalPushNotification::sendNotificationUsingPatientId

HospitalPushNotification Class

Main Functions

- **HospitalPushNotification::sendNotification:**

```
/**
 *    (sendSingleNotification($deviceType, $registrationId, $title, $description)
 *    Consumes a $deviceType, a $registrationId, a  $title and a $message
 *    Description: Sends notification with $title and $message to that device
```

```
 *    Requires: $deviceType = 0 or 1, 0 for iOS, 1 for Android.
 *    Returns: Object containing keys of success, failure, error if any.
 **/
public function sendNotification($deviceType, $registrationId, $title, $description)
```

- **HospitalPushNotification::sendNotificationToMultipleDevices**:

```
/**
 *    (sendNotificationToMultipleDevices($devices, $title, $description)
 *    Consumes a $title,  a $description and an array of $devices.
 *    Sends notification with $title and $description to those devices listed
 *    Requires: $devices must have two fields for each array item: DeviceType,
 *    RegistrationId.
 *    Returns: Array of Objects, each object has keys of success, failure,
 *    RegistrationId, DeviceType and error if any.
 **/
public function sendNotificationToMultipleDevices($devices, $title, $description)
```

- **HospitalPushNotification::sendCallPatientNotification:**
  (Called by John's waiting room.)

  Procedure:
  1. Obtain Patient and appointment information from Database
  2. Update appointment room location in database
  3. Insert into notifications table, here it assumes the NotificationControlSerNum = 10;
  4. Obtain NotificationSerNum for the last inserted Id.
  5. Obtain message (title, description) for room assignment
  6. Build message, replace the $roomLocation with the actual room location argument.
  7. Obtain patient device identifiers
  8. Send message to patient devices and record in database

```
/**
 *    sendCallPatientNotification($patientId, $room, $appointmentSerNum):
 *    Consumes a PatientId, a room location, and an AppointmentAriaSer, it
 *    stores notification in database, updates appointment with room location, sends
 *    the notification to the pertinent devices that map to that particular patientId,
 *    and finally records the send status for the push notification.
 *    (sendRoomNotification String, String, String) -> Array
 *    Requires: - PatientId and AppointmentSerNum are real values in the Database.
 *              - NotificationControlSerNum = 10, Corresponds to the AssignedRoom
 *                notification.
 *    Returns: Object containing keys of success, failure,
 *             responseDevices, which is an array containing, (success, failure,
```

```
         *             registrationId, deviceId) for each device, and Message array containing
         *             (title,description),  NotificationSerNum, and error if any.

         **/
public function sendCallPatientNotification($patientId, $room,$appointmentAriaSer)
```

Other Functions:

This function could be useful in case of failure, or in case of a different notification from John's perspective.

- **HospitalPushNotification::sendNotificationUsingPatientId:**

```
/**
*    (sendNotificationUsingPatientId($patientId, $title, $description))
*    Consumes a patientId, a title and a description
*    Description: Sends push notification containing title and description to all the
*                 devices matching that $patientId.
*    NOTE: This function does not log anything into database. Its meant to send other
*    Returns: Object with success, failure, responseDevices
*            (array of response for each device), and the message array sent.
**/
public function sendNotificationUsingPatientId($patientId, $title, $description)
```

- **HospitalPushNotification::sendFailedNotificationUsingNotificationSerNum:**

```
/**
*    (sendFailedNotificationUsingNotificationSerNum($patientId, $title, $description,
*     $notificationSerNum))
*    Consumes a patientId, a title, a description, and a notificationSerNum
*    Description: Sends push notification containing title and description to all the
*                 devices matching that $patientId and records results into OpalDB
*    Returns: Object with success, failure, responseDevices
*            (array of response for each device), and the message array sent.
**/
public function sendFailedNotificationUsingNotificationSerNum($patientId, $title,
$description, $notificationSerNum)
```

Helper Functions:

- **HospitalPushNotification::pushNotificationDatabaseUpdate:**

```
/**
 *    (pushNotificationDatabaseUpdate($deviceSerNum, $notificationSerNum, $sendStatus)
 *    Consumes a PatientDeviceIdentifierSerNum, $deviceSerNum,  a NotificationSerNum,
```

```
     *    $notificationSerNum
     *    and send status, $sendStatus, where send status is a 1 or 0 for whether is was
     *    successfully sent or not.
     *    Inserts a into the PushNotification table or updates SendLog flag.
     *    RegistrationId.
     *    Returns: Returns the send status
     **/
    private function pushNotificationDatabaseUpdate($deviceSerNum, $notificationSerNum,
$sendStatus)
```

- **HospitalPushNotification::getDevicesForPatient:**

```
     /**
     *    (getDevicesForPatient($patientId)
     *    Consumes a PatientId, $patientId
     *    Returns: Returns array with devices that match that particular PatiendId.
     **/
    private function getDevicesForPatient($patientId)
```

- **HospitalPushNotification::buildMessageForRoomNotification:**

```
     /**
     *    (buildMessageForRoomNotification($room, $title, $description)
     *    Consumes a room, a title and a description of message
     *    Description: Builds push notification message for Room Notification
     *    Returns: Returns array with the push notification message to be sent
     **/
    private function buildMessageForRoomNotification($room, $title, $description)
```

PushNotification Class

Properties:

```
    // (Android)API access key from Google API's Console.
    private static $api_key = API_KEY ;
    // (iOS) Private key's passphrase.
    private static $passphrase = CERTIFICATE_PASSWORD;
    //(iOS) Location of certificate file
    private static $certificate_file = CERTIFICATE_FILE;
```

Functions:

- **PushNotification::Android:**

```
    /**
```

```
*      (android($data, $reg_id)) Consumes an array with message
*      to be sent and a registration id.
*      Description: Creates curl request to FCM (Firebase Cloud Messaging) and sends
*                push notification to android device
*      Requires: $data must contain mtitle, and mdesc for the push notification.
**/
public function android($data, $reg_id)
```

- **PushNotification::iOS:**

```
/**
*      (iOS($data, $devicetoken)) Consumes an array with message
*      to be sent and a registration id.
*      Description: Creates a connection to APN (Apple Push Notification
*              socket and sends push notification to android device
*   Requires: $data must contain mtitle, and mdesc for the
*             push notification.
**/
public function iOS($data, $reg_id)
```