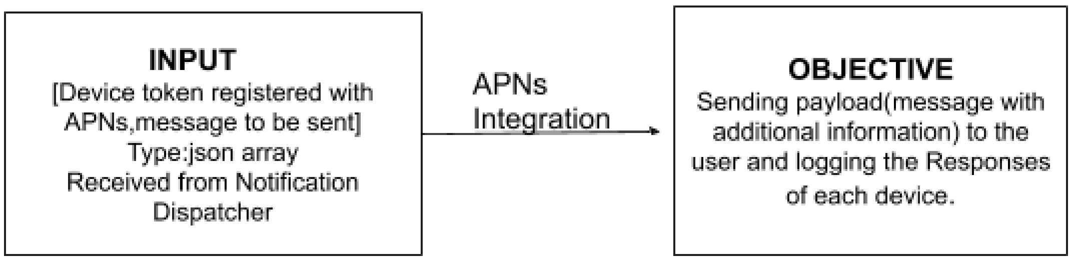
ABSTRACT NOTIFICATION SERVICE

Project Description

The Project aims at developing an abstract notification service that is capable of notifying the user on as many devices on which the user has registered with the application.

Problem Statement

Sending Device token and payload to APNs which will send the notification to respective devices and logging the APNs responses.


INPUT
[Device token registered with APNs,message to be sent]
Type:json array
Received from Notification Dispatcher
APNs Integration
OBJECTIVE
Sending payload(message with additional information) to the user and logging the Responses of each device.

Glossary

1. **Apple Push Notification Service:**
   APNS is a cloud service that allows approved third-party apps installed on Apple devices to send push notifications from remote servers to users over a secure connection.The service delivers notification through an application programming interface that is included in all iOS and MacOS devices.

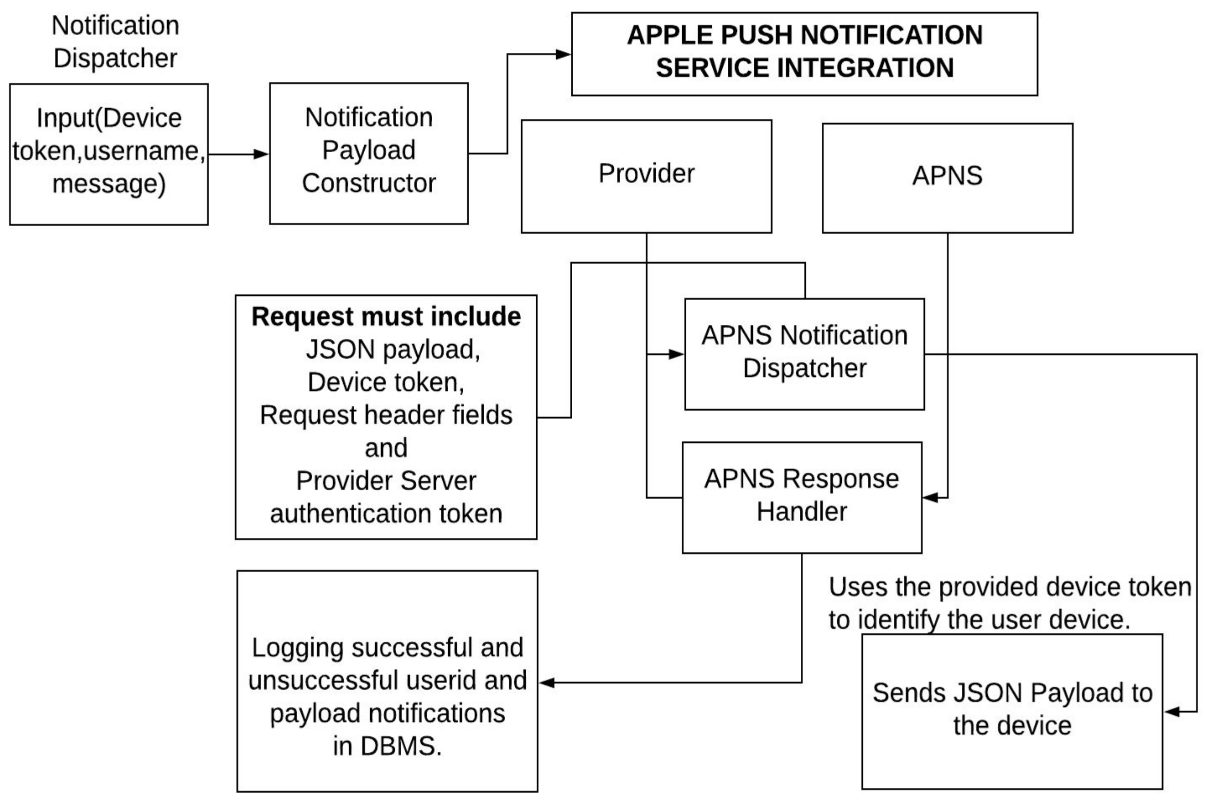
2. **Device Token:**
   A unique key for the app-device combination which is issued by the Apple or Google push notification gateways.

3. **Log:**
   A record group of events executed by Database Management System.

4. **DBMS:**
   Database Management System (DBMS) is a software for storing and retrieving users' data while considering appropriate security measures. It consists of a group of programs which manipulate the database.

5. **Payload:**
   The payload specifies the types of user interactions (alert, sound, or badge) that we want to perform, and includes any custom data the application needs to respond to the notification.

Block Diagram


Notification Dispatcher
Input(Device token,username, message)
Notification Payload Constructor
APPLE PUSH NOTIFICATION SERVICE INTEGRATION
Provider
APNS
Request must include JSON payload, Device token, Request header fields and Provider Server authentication token
APNS Notification Dispatcher
APNS Response Handler
Uses the provided device token to identify the user device.
Logging successful and unsuccessful userid and payload notifications in DBMS.
Sends JSON Payload to the device

Component Description

1. Notification Request Constructor

Some header fields are required for delivering the notification while some are optional.

<u>Required-Header fields</u>:

1. method-POST
2. Path-path to the device token
3. apns push-type(alert, badge,sound etc)
4. apns expiration-type(date at which notification is no longer valid)
5. authentication(encrypted-token if we are using token-based authentication)
6. apns-topic-Bundle id of the application
7. apns-collapse id-used to coalesce multiple notifications into a single notification for the user.

Optional-Header fields:

1. apns id-a unique notification id (default value set by APNs)
2. apns-priority-setting the priority of the notification (default value-10)

Payload is constructed by combining message(data) with the type of user interactions(alert,badge or sound) that to be performed.

2. Connection Establisher

For establishing connection between provider and APNs-configuration is required in the online developer account which can be either token-based authentication or certificate-based authentication.Here in this project token-based authentication is used.

3. APNs Notification Dispatcher

Requests must include payload,Request header field comprising Provider server Authentication token,device token and other header fields which APNs forward it to respective devices.Language used is Javascript with Node js runtime environment.

4. APNS Response Handler

APNs provide a response to each POST request the server transmits. Each response contains a header with fields indicating the status of the response. If the request was successful, the body of the response is empty. If an error occurred, the body contains additional information about the error.Responses are stored in the DBMS along with the status if successfully received and Invalid device tokens are produced in RabbitMQ asynchronous queue.

Logging
DBMS or File System?

Problem

Servers generate a large number of events (i.e. logging,) that contain useful information about their operation including errors, warnings, and user behavior. By default, most servers store these data in plain text log files on their local file systems.While plain-text logs are accessible and human-readable, they are difficult to use, reference, and analyze without holistic systems for aggregating and storing these data.File system does not allow sharing of data or sharing is too complex.There may be cases when some constraints need to be applied on the data before inserting it in the database.The file system does not provide any procedure to check these constraints automatically

Solution
The solution here is to use DBMS.Furthermore, query rate for the logging data in DBMS is substantially lower than common for logging applications with a high-bandwidth event stream in DBMS.Data can be shared easily due to a centralized system in DBMS. DBMS maintains data integrity by enforcing user defined constraints on data by itself.DBMS used here is MongoDB.Along with that RabbitMQ is used.RabbitMQ is an asynchronous messaging-queue.

FAQs

1. **How to prevent sending the same notifications to a particular user more than once?**
   In the header fields of payload, header apns-collapse-id is introduced which enables multiple notifications with the same collapse identifier to display to the user as a single notification i.e. notifications are being overridden.

2. **What should be the life of a log?**
   It depends on the contract of time-duration with the client.According to that we will set the TIME TO LIVE feature in MongoDB which after the specified time will automatically delete the corresponding record.Although without the permission of the client, data unders its log should never be deleted.Even if any device-token is expired,the user should only be marked disabled by Notification-dispatcher.

3. **How does APNS map the Received device token with the user device?**
   Trust between APNs and each device is established automatically.During Device Registration,with a TLS connection established between APNs and the device, apps on the device can register with APNs to receive their app-specific device tokens for push notifications.

4. **What will be the value of apns-priority and apns-expiration fields?**
   Values are set to default value apns-priority 10 and apns-expiration 0.

LLD

INPUT:details Array: type json
[message: type string,device token: type string]
Received from Notification Dispatcher

```
function sendPostRequestApns(details_arr)
{
     for details in details_arr:
           auth_token=tokenGenerator(developerAccountfields)
           collapse_id=getCollapseId(device_token,message)
           header=buildHeader(details[device_token],auth_token
           ,collapse_id)
           payload=buildPayload(details[message])
           sendPostRequest(header,payload)
            response=isSuccessful(response)
           print(response)
            if (response)
           {
             db=createMongooseSchema()
             createLogEntry(db,device_token,message,apns_id)
            }
            if(invalid device_token)
            {
             sendRabbitMessage(device_token,error_reason)
            }
}
```

Function Signatures

```
string tokenGenerator(developerAccountfields)
```
This function uses npm json web token library to generate encrypted authorization web token using the online developer account details.

```
string getCollapseId(string device_token,string message)
```
This function uses SHA256 hash logic to create a unique collapse id corresponding to device token and message.

```
string buildHeader(string device_token,string auth_token,string
collapse_id)
```
This function builds all the request header fields specifying how to deliver the notification for post request-method,apns-id,apns-push-type,apns-collapse-id,

apns-expiration,apns-priority and authorization-token.

```
string buildPayload(string message)
```
This function builds the payload by combining message(data) with the type of user interactions(alert,badge or sound) that we want to perform.

```
void send_PostRequest(string header,string payload)
```
Axios is a very convenient Javascript library to send http requests to apns.Using axios,this function sends the payload and header to the Apple push Notification Service which forwards it to respective devices.

```
bool isSuccessful(string Response)
```
Checks the status of response from APNS.
Each response contains a header with fields indicating the status of the response. If the request was successful, the body of the response is empty. If an error occurred, the body contains a JSON dictionary with additional information about the error.

Logs Schema

```
logs_details
{
  device_token: {type:string}
  status_Code: {type:number}
  apns_id:{type:string}
  message:{type:string}
  date:{type:string, default:Date.now().toString()}
  expires:{type:seconds} //Time to live index
}
```

```
void createLogEntry(string db,string device_token,string
message,string apns_id)
```
This functions logs the information of successful responses in MongoDB database in the db schema.Every time it creates a new document in the collection accounting device_token,message,apns_id,status_code and date.Time to live index is also maintained.

```
void sendRabbitMessage(string device_token,string error)
```
If the sent device token is unregistered or inactive,then it is produced in RabbitMQ queue along with the error reason in order to be rectified by a Notification dispatcher.

Provider Authentication to communicate with APNs

Why Token based Authentication better?

Token-based authentication offers a stateless way to communicate with APNs.Stateless communication is faster than certificate-based communication because it does not require APNs to look up the certificate, or other information, related to the provider server.
There are other advantages of using token-based authentication:

1. We can use the same token from multiple provider servers.
2. We can use one token to distribute notifications for all of your company's apps.

Procedure of Token-based Authentication

Encrypt the resulting data using the authentication token signing key obtained from the developer account.Provider server must include the resulting encrypted data with all notification requests

Reasons to use Node js

1. It's a light, scalable, and cross-platform way to execute code.
2. The ever-growing NPM (Node Package Manager) gives multiple tools and modules to use, thus further boosting their productivity.
3. Node js is very well supported by MongoDB and RabbitMQ as well.
4. Node makes use of callback and promises thus providing an easy interface to allow our code to handle large concurrent connections.

Reasons to use MongoDB for logging the responses

Using NoSQL(MongoDB) over MySQL

1. Since with mostly Non-relational Database in the logs,it is easy to handle large unstructured data.It is magically faster than MySQl Database.
2. In MongoDB, Structure of a single object is clear.There is no fixed schema which ensures flexibility of the data-type.
3. There are no complex joins.
4. MongoDB offers better performance for mass read and write operations than MySQL.

5. MongoDB can be a disadvantage when there are a lot of write queries in the relational database but with our logs working on a Non-Relational database,it is not going to lower the performance of logging.
6. With the Time to live(TTL) feature in MongoDB,it is easier to implement the log life feature in MongoDB than in MySQL.

Mongoose is an Object Modelling library for mongodb and Node js.It used mongodb library to connect the MongoDB database with Node js.Mongoose is used here to provide a straight-forward schema based solution to model our data.

Reasons to use RabbitMQ

RabbitMQ is an asynchronous Messaging queue to store the undelivered notification.The basic architecture of a message queue is simple - there are client applications called producers that create messages and deliver them to the message queue. Another application, called the consumer, connects to the queue and gets the messages to be processed.
RabbitMQ has a high community-support.It has very good documentation.There is no powerful alternative for RabbitMQ to create an asynchronous messaging queue.
.