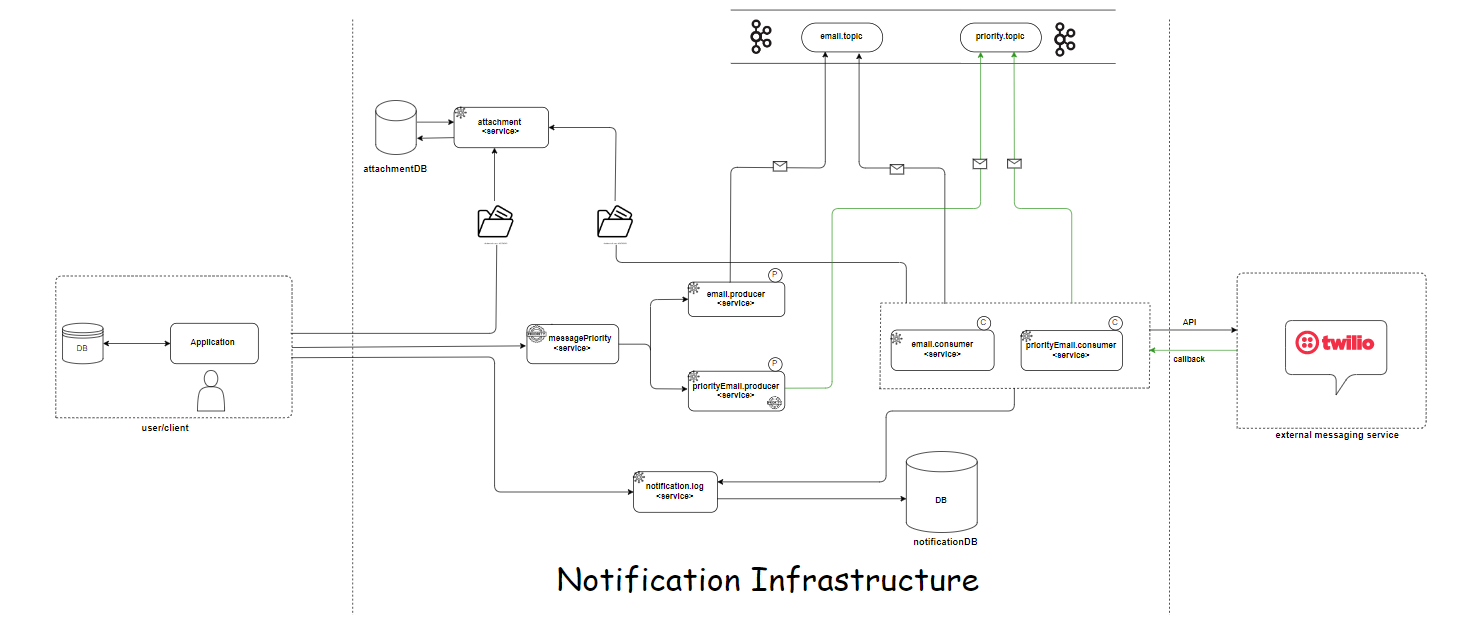
Documentation For Appointment Notification Infrastructure

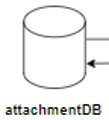
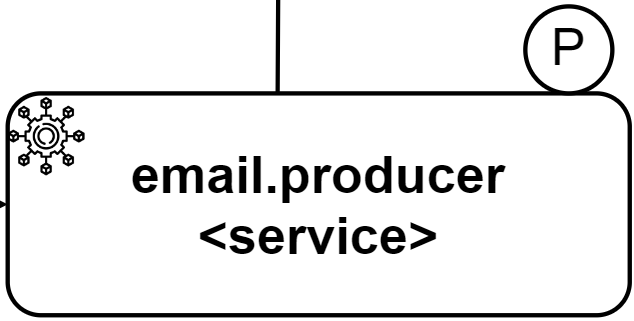
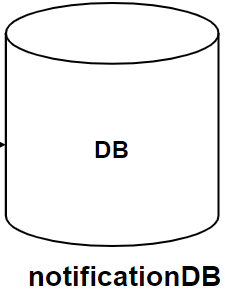
Purpose :- This is a proposed Infrastructure for sending notification to the patient regarding their appointment scheduling with the provider.

Scope :- This system notifies the patient as well as the practice about the patient’s appointment confirmation.

Proposed Approach :-



**List of contents :-**

1. External application - 
2. Attachment service - 
3. Attachment’s database - 
4. Message priority service - 
5. Email producer service – 
6. Topic (email\_Topic) – 
7. Email consumer service – 
8. Twilio - 
9. Notification log service - 
10. Email notification database- 

**Flow/Explanation for notification infrastructure :-**

1. **External application -** There is external application having its own database sending their users data in JSON form first to the attachment service for handling the attachment file it returns back the response with attachment\_ID. After getting response as success and Id this attachment\_ID will be taken from application to send with another JSON data to message priority service.
2. **Attachment** **service** :– As users may also add Attachment’s data with their JSON making the TOPIC heavy as it later gets produced/consumed for this we would make a dedicated service to handle Attachments to keep our application lighter and optimized. Also, it will store attachment data in binary form into its dedicated attachment database and return attachment’s ID to the application again.
3. **Message Priority service :-** After getting attachment ID from attachment service again a JSON post will be done in continuation with earlier attachment JSON including the attachment\_ID and message priority High/Low. This service will ­check for message priority depending upon users request this will decide to which email producer it will send the user JSON data.
4. **Email producer service :-** The low priority email producer will get the JSON data from message priority service and send into the email.Topic(Low priority).
5. **Email Topic :-** This topic (Low priority) will store the JSON data sent by email producer service temporarily later to be consumed by email consumer service(Low priority).
6. **Email consumer service :-** This consumer service will consume the low priority message from email.Topic and later will sent to Twilio.
7. **Twilio** – After the JSON data is consumed by the Email consumer service the same consumer will contain TWILIO message sending business Logic and will send to TWILIO external Application to send email as notification.

Steps to configure Twilio into the consumer.

1. A host. The host for Web API v3 requests is always https://api.sendgrid.com/v3/
2. An [Authorization header](https://sendgrid.api-docs.io/v3.0/how-to-use-the-sendgrid-v3-api/api-authentication#authorization-header). An [API Key](https://app.sendgrid.com/settings/api_keys) must be included in the Authorization header.
3. A request. When submitting data to a resource via POST or PUT, you must submit your payload in JSON.

After the email is delivered/failed/pending its response will be sent back to the consumer. And stored into Database.

1. **Notification log service** :- The response(delivered/failed/pending) from the Twilio will be sent to email consumer service and later to the notification log service from the email consumer with REST post call.
2. **Notification Database** :- The response will be stored into this Database and will be later Application will get the reponse from notification Database with Rest Get call.

1. **Database** - The response/status of the email to be delivered/failed/pending is to be stored into this database for later to be consumed.

**1.1 Contract for application to attachment service :-**

Application :-

|  |  |  |
| --- | --- | --- |
| Method | URL | Action |
| POST | /uploadfile | Upload file |
|  |  |  |

**POST** :-

{

“sourceId” : “S1”

"attachment" : " Blob ",

}

**Returns** response

“attachmentId” : “001”

**1.2 Contract from application to messagePriority service** :-

|  |  |  |
| --- | --- | --- |
| Method | URL | Action |
| POST | /sendemail | Posting data to priority service |

{

*“sourceId” : “S1”,*

*“messageId” : “01ac0cd8-3d69-45ad-943c-35e811d4647a ”,*

*“messagePriority” : “High/Low”,*

*“from” : “xyz@gmail.com” ,*

*“to” : “abc@gmail.com”,*

*“subject” : “This is subject”,*

*“body” : “This is message body”,*

*“replyTo” : “*[*pqr@gmail.com*](mailto:pqr@gmail.com)*”,*

*"attachmentId" : "001"*

*}*

**URL:**  /sendemail

**Method: Post**

**In the above url, The application will be sending email data to messagePriority service to decide the message priority based on application requirement, later will be sent to producer again depending upon the required priority (High/Low)**

**1.3 Contract for messagePriority service to email.producer service(when priority is low )** :-

|  |  |  |
| --- | --- | --- |
| Method | URL | Action |
| POST | /send/lowpriority | Sending low priority data to low priority email producer |

**URL:**  localhost:8090/send/lowpriority

**Method: Post**

**In the above url, The messagePriority service will be sending email data to low priority email producer after being decided the message priority based on application requirement, later will be sent to low priority email producer again depending upon the requirement i.e Low.**

**1.4 Contract for email.producer service to email.topic** :-

**In the above url, The low emailProducer service will be sending email data to low priority email topic to be later consumed by low priority email consumer service.**

**1.5 Contract for email.consumer service to email.topic ( when priority is low )** :-

To be consumed topic = email.Topic

Consumer = email.consumer service

email.consumer service will consume data from email.Topic.

**1.6 Contract for email.consumer service to attachment service :-** email.consumer service will make a rest Get call to attachment service after matching the attachment\_ID got from Topic this will get attachement file for the particular ID.

**1.7 Contract for email.consumer service to external messaging service i.e TWILIO ( when priority is low )** :-

After the JSON data is consumed by the Low priority Email consumer the same consumer will contain TWILIO message sending business Logic and will send to TWILIO external Application to send email to the client as notification. Steps to configure Twilio into the consumer.

1. A host. The host for Web API v3 requests is always https://api.sendgrid.com/v3/
2. An [Authorization header](https://sendgrid.api-docs.io/v3.0/how-to-use-the-sendgrid-v3-api/api-authentication#authorization-header). An [API Key](https://app.sendgrid.com/settings/api_keys) must be included in the Authorization header.
3. A request. When submitting data to a resource via POST or PUT, you must submit your payload in JSON.

After the email is delivered/failed/pending its response will be sent back to the consumer. And stored into Database.

**1. Contract for email.consumer service to notification.log service** :-

|  |  |  |
| --- | --- | --- |
| Method | URL | Action |
| POST | /send/callback | sending the email callback response to notification.log service (success/pending/failed) |

**URL:**  localhost:8090/send/callback

**Method: Post**

**In the above url, The callback response received from Twilio to email.consumer will then send to notification.log .**

|  |  |  |  |
| --- | --- | --- | --- |
| Source\_ID | Message\_ID | Status | Response\_ID |
| S1 | 01ac0cd8-3d69-45ad-943c-35e811d4647a | Pending | RS-S1-1001 |
| S2 | a44e8239-e497-4acd-a195-98e2296713f0 | Success | RS-S2-2001 |
| S3 | 51533dd4-19ae-4880-a125-57e27dd6bb4f | Failed | RS-S3-3001 |

1.10 **Contract for notification.log service to notificationDB** :-

|  |  |  |  |
| --- | --- | --- | --- |
| sourceid | messageid | status | responseid |
| S1 | 01ac0cd8-3d69-45ad-943c-35e811d4647a | Pending | RS-S1-1001 |
| S2 | a44e8239-e497-4acd-a195-98e2296713f0 | Success | RS-S2-2001 |
| S3 | 51533dd4-19ae-4880-a125-57e27dd6bb4f | Failed | RS-S3-3001 |

Once the notification.log service receives the response from email.consumer it will store the data to notificationDB by using post method.

**1.11 Contract for application to notification.log service** :-

|  |  |  |
| --- | --- | --- |
| Method | URL | Action |
| GET | /{messageId}/notificationstatus | Getting the callback response from notificationDB |

**URL:**  localhost:8090/{messageId}/notificationstatus

**Method: Get**

**In the above url, The application will get callback from notification.log service.**