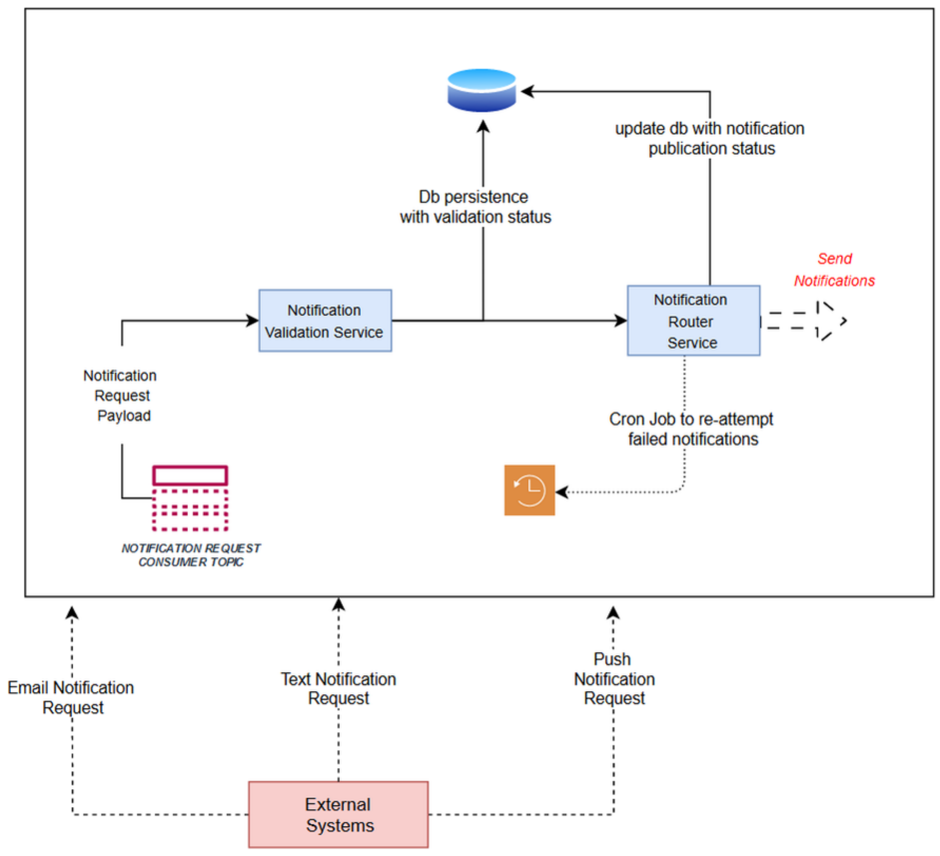
Multichannel Notification System Design

Overview : There will be a Microservice that has a Kafka consumer (or similar messaging system) listening to a specific topic/queue, External systems will send `NotificationRequest` to this topic/queue. All the information required for a successful publishing the notification will be available in `NotificationRequest` message. Incoming message will also contain a header containing information like the priority or notification type which allows us for routing the Notification in different channel. We will be persisting the notification data with all the metadata such as type, priority and status info such as validity, publication status etc.

Running in the background is a scheduled job meant for handling failed notifications. The system persistently retries, potentially using strategies such as exponential back-off, and updates the notification status on success publish. This configuration ensures that notifications, even if initially unsuccessful, get additional chances for successful delivery.


update db with notification publication status
Db persistence with validation status
Send Notifications
Notification Validation Service
Notification Router Service
Notification Request Payload
Cron Job to re-attempt failed notifications
NOTIFICATION REQUEST CONSUMER TOPIC
Email Notification Request
Text Notification Request
Push Notification Request
External Systems

Notification Validation Service:

- **Purpose:** Ensures notifications adhere to predefined rules and formats before being sent.
- **Responsibilities:**
  - Validate basic notification fields like `content`, `sender`, and recipient information.
  - Check for format consistency based on the notification channel (e.g., character limits for SMS, email address validity).
  - Apply channel-specific validation rules (e.g., push notification payload formatting).
  - Implement security measures to prevent spam, phishing, or malicious content.
  - Handle error scenarios gracefully and provide informative feedback to the sender.

Notification Router Service:

- **Purpose:** Directs notifications to the appropriate communication channels based on type, priority, or other criteria.
- **Responsibilities:**
  - Maintain a mapping of users to their preferred notification channels and preferences.

- Determine the most suitable channel based on notification urgency, content type, and user preferences.
- Integrate with various channel-specific APIs to send notifications (e.g., push notification providers, email gateways, SMS services).

Data Models

Assumptions is that the NotificationRequest will come along with the sender as well as receiver information. On the other hand we can also have a table for holding user information from where the sender information will be fetched depending the user information coming from the external systems in the NotificationRequest

BaseNotification:

```
JSON{
  "content": "This is a test notification",
  "sender": "My App",
  "timestamp": "2024-02-18T04:53:17.000Z"
}
```

PushNotification:

```
JSON{
  "content": "Your order has been shipped!",
  "sender": "E-commerce App",
  "title": "Order Update",
  "body": "Your order #12345 has been shipped and will be delivered on February 20th.",
  "badge": 1,
  "additionalData": {
    "orderId": 12345
  }
}
```

EmailNotification:

```
JSON{
  "content": "Welcome to our platform!",
  "sender": "My Platform",
  "subject": "Welcome aboard!",
  "fromAddress": "noreply@myplatform.com",
  "toAddress": "user@example.com",
  "ccAddress": ["manager@example.com"],
  "body": "Hi John, thank you for signing up for our platform!",
  "htmlBody": "<p>Hi John,<br>Thank you for signing up for our platform! We're excited to have you on board.</p>
}

```

SmsNotification:

```
JSON{
  "content": "Meeting reminder: 10:00 AM today",
  "sender": "Calendar App",
  "recipientNumber": "+1234567890",
  "countryCode": "US"
}
```