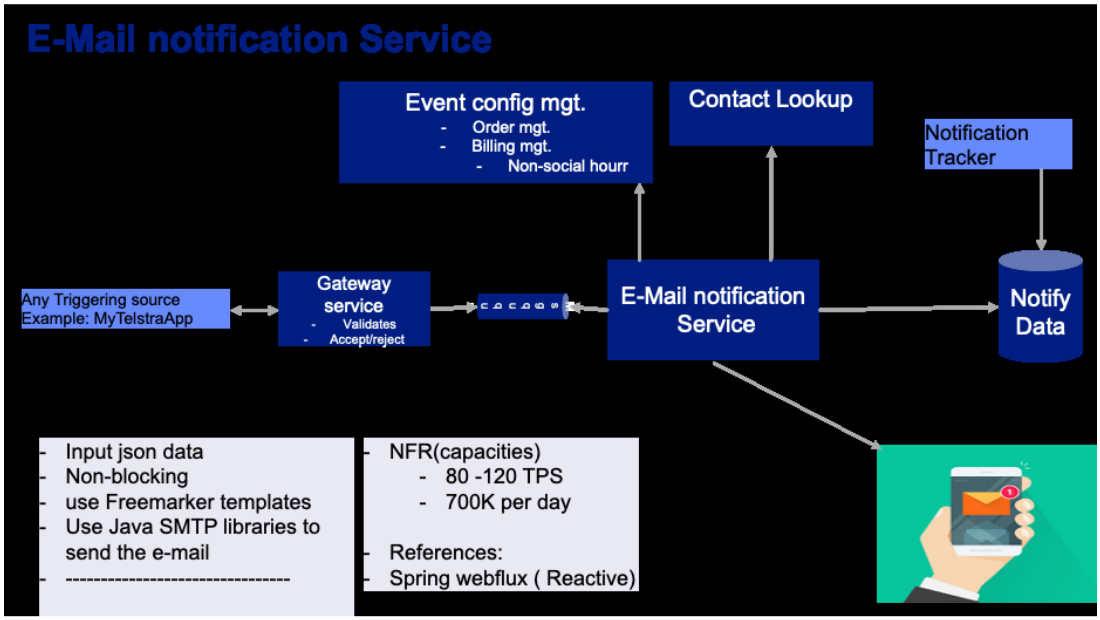
# Resilient E-Mail Notification Service

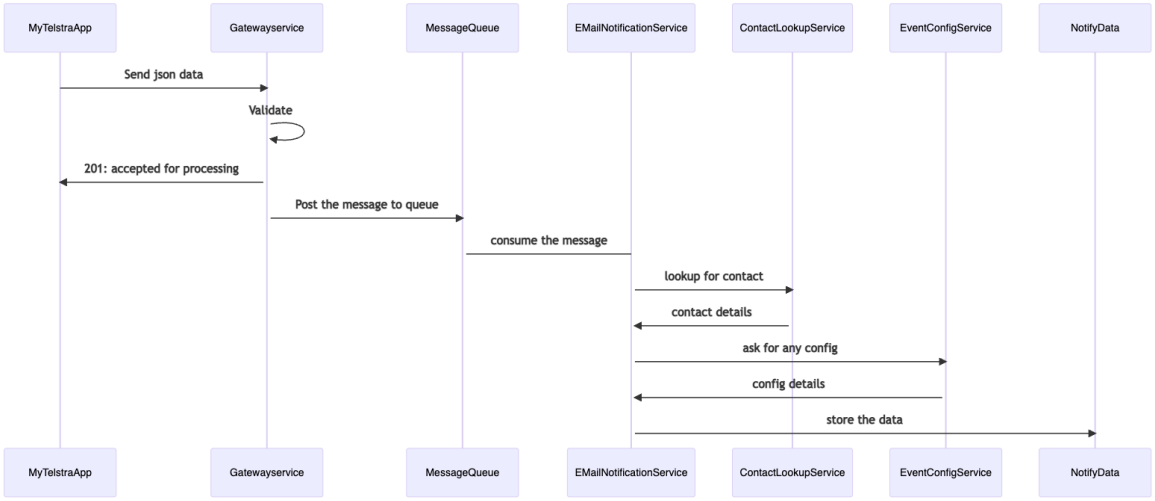
## Purpose

To build a resilient non-blocking event driven e-mail notification service using reactive Spring Webflux framework.
Resilient we mean to support 80-120 TPS and 700K notifications per day.

## Description

In the era of mobile and web Internet, each of us will receive countless notifications every day. How can the product make the messages reach users efficiently and accurately? The message notification can promptly reach the user about the message status and the content so that the user can make follow-up judgments based on the corresponding notification information. Let us discuss the notification system architecture behind the product.



**High Level components**
1/ Gateway Service
2/ Notification Service
- Bulk Notification Handler
3/Notification Handler & Preferences
4/ Rate Limitter
5/ Notification verification, validation
6/ Notification Dispatcher
7/ User contact Lookup
8/ Notify DB
9/ Notification Tracker

**User Stories / Functional Requirements**

1. Send notifications
2. Prioritize notifications/turn on or off the application notifications, categories the message from the settings
3. Adjust notification sequence based on the user's saved preferences (User preferences)
4. Single/simple and a bulk notification message
5. Provide read-only or read and operation feature to notification messages
6. Any notification can be cancelled at any time because the user removes the user ID from the client-side application.
7. No same notification twice
8. Log every notification dispatched, delivered, opened, seen, unsuccessful, canceled

**Non-Functional Requirements**

- Low latency
- Highly available (Handling exceptions when any component failure occurs)
- High performance (real-time)
- Support as many devices as possible
- Durable — Message should not be lost, no duplication
- Scalable — Support a large number of publishers, subscribers, topics, and users
- Decoupled — follow the SOLID principle, and merge with another notification system
- Pluggable — Provide API integration with the other client applications.

**The conditions of a good messaging notification system**
**Comprehensive:** The message notification can provide a better-updated content of the message completely and comprehensively

**Timely:** The way to reach the user with the message should be timely and effective

**Efficient:** To avoid excessive message intrusion to users by allowing to group messages based on the senders or group member senders so that
users can process messages without confusion.