消息通知系统的架构设计

原创 小技术君 小技术君 2023-07-20 09:06 上海

目标：

设计企业级系统架构，支持使用API集成的电子邮件、短信、聊天和其他公共社交应用程序：

- 电子邮件
- 短信/一次性密码
- 推送通知（移动设备和Web浏览器）
- 聊天 - Whatsapp/Telegram

这是一种通用的功能，适用于所有现代分布式应用程序，无论使用任何编程语言和技术。

我试图简化这个设计概念，以满足高可用性、高性能和分析服务的常见用例需求。这是通过微服务架构实现并在Kubernetes容器上部署，使其成为完全云原生的现代系统。让我们开始吧！

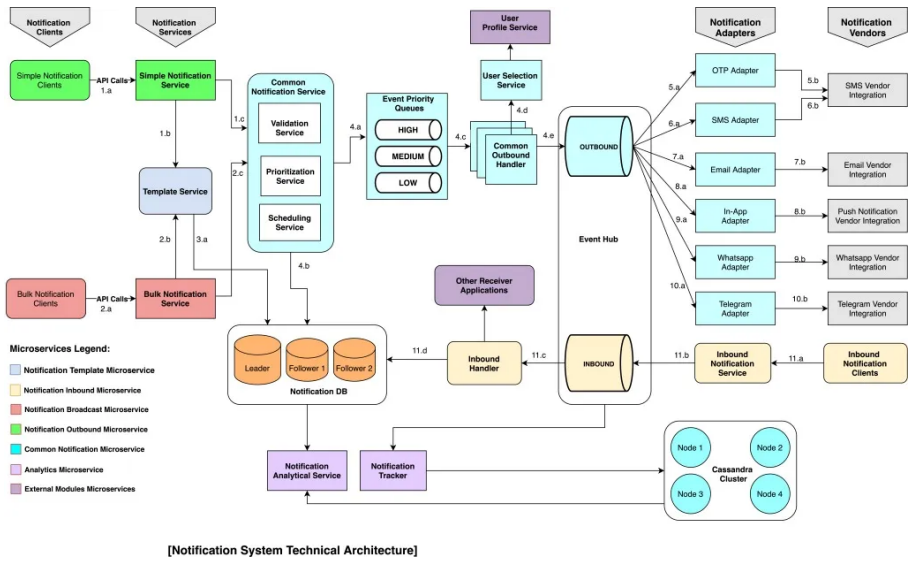
功能要求：

- 发送通知
- 对通知进行优先级排序
- 根据客户的保存偏好发送通知
- 支持单个/简单的通知消息和批量通知消息
- 各种通知的分析用例
- 通知消息的报告

非功能性需求（NFR）：

- 高性能
- 高可用性（HA）
- 低延迟
- 可扩展/可插拔的设计，以添加更多的客户端、适配器和供应商
- 支持Android/iOS移动设备和桌面/笔记本电脑的Web浏览器
- 与所有通知模块的API集成和与客户端和服务提供商/供应商的外部集成
- 可在本地（VMware Tanzu）和AWS、GCP或Azure等公共云服务上扩展负载

系统设计架构：



[Notification System Technical Architecture]

注意：请点击图像以查看清晰的视图！

这些是解决方案设计的考虑因素和组件：

1. 通知客户端：

这些客户端将使用API调用请求单个和批量消息。这些客户端将向简单和批量通知服务发送通知消息。

- 批量通知客户端：这些客户端发送批量通知。
- 简单通知客户端：这些客户端发送单个通知。

2. 通知服务：

这些服务是入口服务，将通过暴露REST API与客户端交互。它们负责通过消耗"模板服务"来构建通知消息。这些消息将使用"验证服务"进行验证。

- 简单通知服务：该服务将暴露API，以将客户端与后端服务集成。它是主要的服务，用于处理简单通知请求。
- 批量通知服务：该服务将暴露API，以将客户端与后端服务集成。它是主要的服务，用于处理批量通知请求。

此服务还将管理通知消息。它将将发送的消息持久化到数据库并维护活动日志。可以使用这些服务的API重新发送同一条消息。它将提供添加/更新/删除和查看旧消息和新消息的API。它还将提供Web仪表板，该仪表板应具有筛选选项，以根据不同的条件（如日期范围、优先级、模块用户、用户组等）筛选消息。

3. 模板服务：

该服务管理所有可用的OTP、短信、电子邮件、聊天和其他推送通知消息的模板。它还提供REST API来创建、更新、删除和管理模板。它还将提供一个UI仪表板页面，以便从Web控制台检查和管理消息模板。

4. 用户选择服务：

该服务将提供选择目标用户和各种应用程序模块的服务。可能存在将批量消息发送到特定用户组或不同应用程序模块的用例。可能是AD/IAM/eDirectory/用户数据库/用户组，根据客户的偏好。在内部，它将使用"用户配置文件服务"API来消耗和检查客户的通知偏好。

5. 用户配置文件服务：

该服务将提供各种功能，包括管理用户配置文件和其偏好设置。它还将提供取消订阅通知以及通知接收频率等功能。"通知服务"将依赖于此服务。

6. 通用通知服务

- 定时服务：

该服务将提供API来安排立即或指定时间的通知。可以是以下任何一种：

- 秒
- 分钟
- 每小时
- 每天
- 每周
- 每月
- 每年
- 自定义频率等。

还可能有其他自动触发的服务，基于预定时间进行消息触发。

- 验证服务：

该服务完全负责根据业务规则和期望的格式对通知消息进行验证。批量消息应由授权的系统管理员批准。

- 优先级服务：

它还将根据高、中和低优先级对通知进行优先级排序。OTP通知消息具有较高的优先级和有时间限制的过期时间，它们将始终以较高优先级发送。"通用出站处理程序"将消耗消息并根据同样的优先级从高、中和低三个不同的队列中发送和处理。批量消息的另一个用例是在非工作时间使用低优先级发送。在交易过程中的应用程序通知可以发送到中优先级，如电子邮件

等。业务将根据通知的重要性决定优先级。

7. 事件优先级队列（事件中心）：

它将提供事件中心服务，从通知服务中消耗高、中和低优先级的消息。它将根据业务优先级发送和接收消息。

它将具有以下三个主题，用于根据业务优先级消耗/发送消息：

- 高优先级
- 中优先级
- 低优先级

8. 通用出站处理程序：

该服务将通过轮询事件优先级队列来消耗事件中心中的通知消息，根据其优先级进行处理。高优先级将优先给予"高"队列，依此类推。最后，它将通过事件中心将通知消息发送到特定的适配器。

该服务还将从用户选择服务中获取目标用户/应用程序。

9. 通知数据库：

该数据库将持久化所有通知消息，包括其发送时间、状态等。它将具有一个数据库集群，其中一个领导者将用于执行所有写操作，读取将在读取副本/跟随者上进行。它应该是一个非关系型数据库。

10. 出站事件中心：

它最终将消息传输到各种支持的适配器。这些适配器将基于不同的设备（桌面/移动）和通知类型（短信/OTP/电子邮件/聊天/推送通知）进行转换。

11. 通知适配器：

这些适配器将从事件中心（Kafka）接收传入消息并根据其所支持的格式发送给外部供应商。以下是一些适配器，根据需求可以添加更多：

- OTP适配器服务
- 短信适配器服务
- 电子邮件适配器服务
- 应用内通知适配器服务
- WhatsApp聊天通知适配器服务
- Telegram通知适配器服务

12. 通知供应商：

这些是外部的SAAS（云上/本地）供应商，使用它们的基础设施和技术提供实际的通知传输。它们可能是像AWS SNS、MailChimp等的付费企业服务。

- 短信供应商集成服务
- 电子邮件供应商集成服务
- 应用推送通知供应商集成服务

- WhatsApp供应商集成服务
- Telegram供应商集成服务

13. 通知分析服务

该服务将执行所有的分析工作，并识别通知使用情况、趋势以及进行报告。它将从分析数据库（Cassandra）和通知数据库中提取所有最终的通知消息，用于分析和报告目的。

以下是一些用例：

- 每天/每秒的总通知数。
- 哪个通知系统使用最频繁。
- 消息的平均大小和频率。
- 基于优先级过滤消息等等...

14. 通知跟踪器

该服务将持续读取事件中心队列并跟踪所有发送的通知。它捕获通知的元数据，如传输时间、传送状态、通信渠道、消息类型等。

15. Cassandra数据库集群

该数据库集群将持久化所有通知，用于分析和报告。它基于“写入更多，读取更少”的概念。

它将提供良好的性能和低延迟，适应大量的通知，因为它内部管理大量的写操作，并与其他数据库节点同步，并保留高可用性和可靠性的冗余数据/消息。在任何节点崩溃的情况下，消息将始终可用。

小技术君
技术闲谈 ✓ 系统设计
254篇原创内容

公众号

应用程序 52 电子邮件 2 系统架构 38 设计 34

应用程序 · 目录

上一篇
系统设计蓝图 / 备忘单

下一篇
现代系统设计风格概览

喜欢此内容的人还喜欢

微服务架构中的挑战及应对方式：Outbox 模式
小技术君

系统设计 —— 随用户扩展
小技术君