News System

# Overview

A set of news stream will be pushed to this system through different news servers. The News System (NS) must be able to process all the coming news and notify its users about these news. Users can subscribe to multiple news channels, and when they login they start receiving news related to the channels they subscribe to. The NS need to process coming news asynchronously.

Extra Functionality: The system will be able to allow users to have multiple device, and the news will be pushed to all devices the use has.

# News System (NS) Components

## Overview

NS system composed of Receiving service (RS), Notification Service (NotS), User management service (US) and Background service (BS). Each one of these service will implement a set of use cases as you will see later on.

The RS will asynchronously receive news from News Servers, it will process the data and notify needed devices, which is described in more details in Receiving Service (RS). Non logged in devices will be notified later on the Background Service (BS).

## Receiving Service (RS)

### Overview

This service will receive news from a set of news servers. RS is a stateless and loosely coupled service, which make it scalable and can be auto scaling. After the RS receives a news it stores it into a database and notify CommingNewsService(CNS) about the new news. RS APIs are asynchronies APIs.

When RS receives a news, it creates it in the news table, find all the active user devices belongs to active users who had subscribed to the same news category and create entities for these into NewsDevices table by relating them to the newly created news. After that the user will send a notification to the asynchronous Notification Service (NotS) to notify the needed devices about the new news.

### Use cases

Data is a constant stream. This means that this service APIs should be asynchronous APIs, which is the case. Every RS service will have a thread pool used to execute coming requests in the background.

System accepts news from multiple news servers. RS is a scalable system because it is stateless, and loosely coupled.

### System Devices

A set of News Servers (Berlin, London) will continuously send news to RS servers which will asynchronously save this information to NS server and notify NS about the received news. Since RS servers are stateless and loosely coupled, the RS system can be expandable.



### Details

When RS receives a news from News Server through its web interface NewsController class, the news controller will return immediately after create a thread task NewsProcessorTask (NPT), which will run in the background. The NPT will process the news data by first get all active users (logged in and non-logged in) and their devices, then it will attaches the news to every device then it will notify only the devices that are logged in about the coming news. Due to the heavy load this RS system will have, and since the RS system will continuously does read and write on the data, then the system seem is required to use read uncommitted isolation transaction. But because, these are news adding to the system and will have block any other operations with their read or writer locks, then the system can use any isolation level. Even though read uncommitted isolation level will cause dirty reads, unrepeatable reads, and phantom reads but this is acceptable due the system nature. An example of this is: a device logged out when a system receives a news and though it is not logged in because of dirty reads and unrepeatable reads problem. So even if the RS current transaction find it needs to notify a logged in device, and another transaction occurs at the same time changing the device state to logged out, then the notification system will not be able to deliver the message and the backend services will find that the device is logged out and will not try to resend it back. This situation could occur when the notification system is sending a news to a device while the device is logging out, similar problem could without transaction isolation. Another example of issues that may occur is: a user search for news but could not get the latest ones because of phantom problems. Even though this situation could occur but the notification system will notify the user about the missing news. The same problem could occur when a user search about some news and after few micro seconds of replying to user the system receives a new news from news server.

### Class diagram



### State Diagram



## Notification Service (NotS)

### Overview

This components will be responsible to notify user devices about news related to their subscribed channels. This system will be able to scale in order to be able to notify any number of users regardless of the news volume.

The NotC components is loosely coupled, it does not depends on other serves except the authentication service. To achieve the loosely couple nature, it is required that the data it receives include all need information for the service to process required request. For example, the component will receive news data, user data and devices data.

### System Devices

This NS composed of a set of NewsQueueNotifiers servers running behind a load balancer. The NotS service is stateless and loosely coupled service, which make it a scalable service. News Notification System figure shows the hardware architecture of NS service and how it can scale.



### Class Diagram



## Background Service (BS)

### Overview

BS will look on the news that has created or notifying status with more than N minutes and resend notifications to the required devices through the Notification Service (NotS). BS will also be notified when a device statues get changed to logged in and notify the device about the news it missed while it was logged out through the NotS. When a user change it news categories, BS will be use to clean DeviceNews table from data that belongs to a user and not in the user categories.

Will not go into the details since it is out of scope.

### Use Cases

Device logged in, so resend news with status created.

User change its news category.

Device did not receive a news, resend the news to the device when it is logged in.

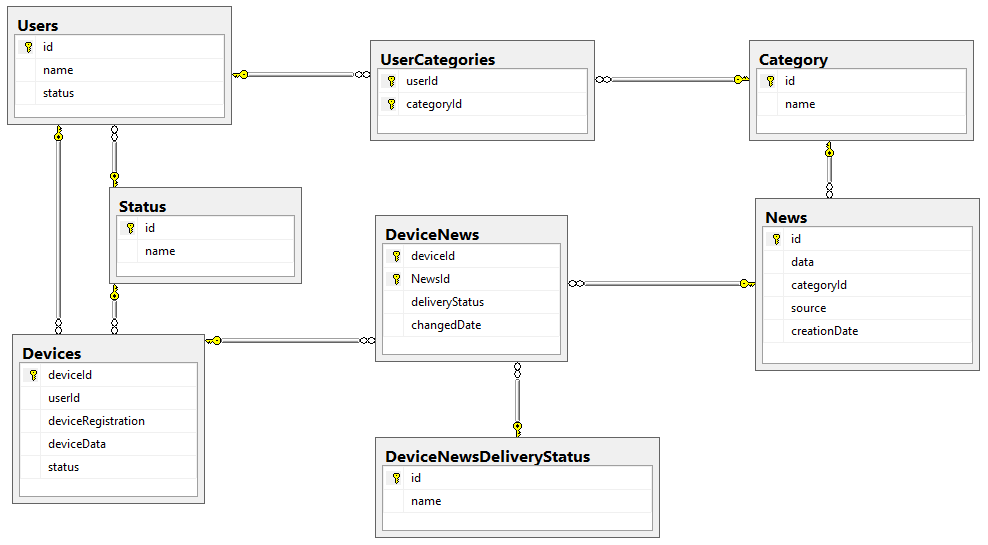
## User Management

This system is needed to manage users, registration, login, logout, category management, user device management.

Will not go into the details since it is out of scope.

# Database

## Diagram



Even though this design mentioned a relational database system, but this system can be implemented also using a nosql document databases like mongodb.

Queries that will be answered:

1. What are a user categories?
2. What are the active devices belongs to active users with specific news category?
3. What are the logged in devices?
4. What are the news that has delivery status created and belongs to a device that is logged in?
5. What are the news that has delivery status notifying and belongs to a device that is logged in and its changed date is greater than N minutes?
6. What are the DeviceNews that belongs to a user and not in a specific category set?

# Security

OAuth2 service to service authentication will be used. This is will not discussed in this document since it is out of the scope.

# Original Requirements

You have a user database and multiple external news servers. Each news server serves a region, e.g. Berlin, Hamburg, Munich.

At login the user gets data from the news channels he subscribed to. Apart from that, channels are not personalized.

Since there is a constant data stream from different sources, the system needs to be asynchronous by nature. Additionally the users should be notified, when new news arrive on a channel they registered on.

1) How would you design such a system. How would the news be stored? How would they be passed to the frontend and how would new news be communicated to the frontend upon arrival? How would you persist users and news?

2) JPA and DB: Given that the information is stored in an SQL database.

For database transactions there are different isolation levels? What is the reasoning behind it. Why would you choose one over the other. What would be a typical use case given the scenario above?

What is an optimistic lock exception in JPA?

What is a stale object, what is a detached object?

How would you deal with them given the asynchronous scenario above?

3) As the business grows, there are many more users. How could you scale such a system? Where would you store the user session data?

4) Since news will not change after a while, how would you handle with news that are older than 1 year to control the capacity of the database? How does this affect your persistence logic?

5) Please provide a sample application that implements the desired system.