

# Distributed Computing: Documentation

Zhong-Xi Lu, Angela Mizero, Thomas Van Bogaert

## 1 Microservice Oriented Architecture

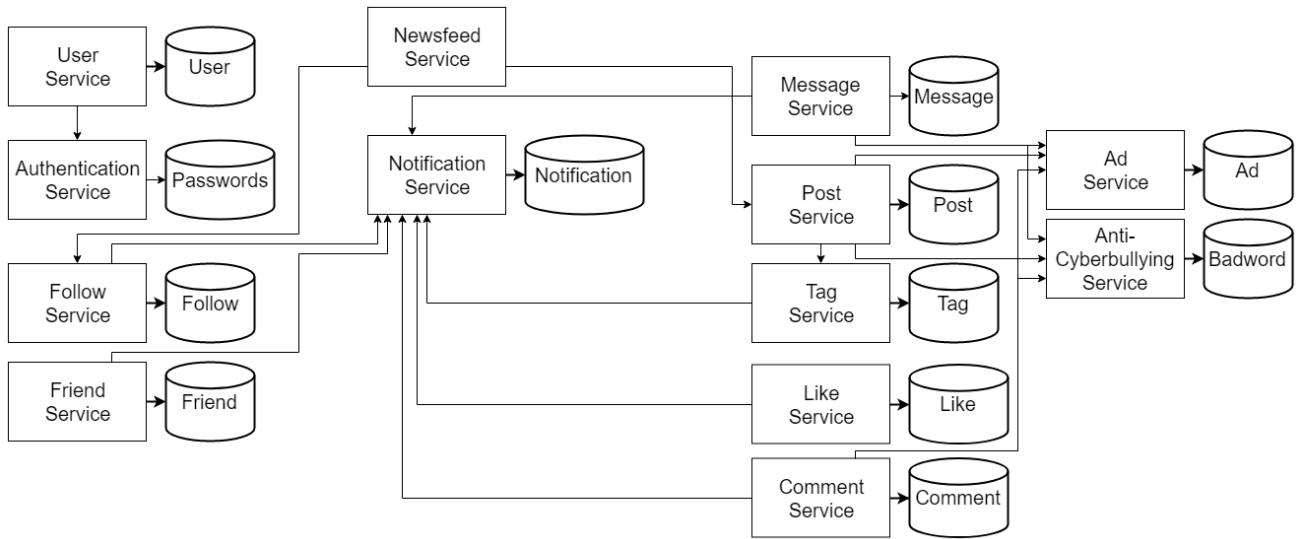


Figure 1: Architecture of core microservices with their interaction

The core microservices that form the backend of our system can be found in figure 1. The interactions, i.e. dependencies, between the different microservices are shown as well. Most of these services also have a corresponding database which is in fact a microservice on its own. More on these services in section 2.

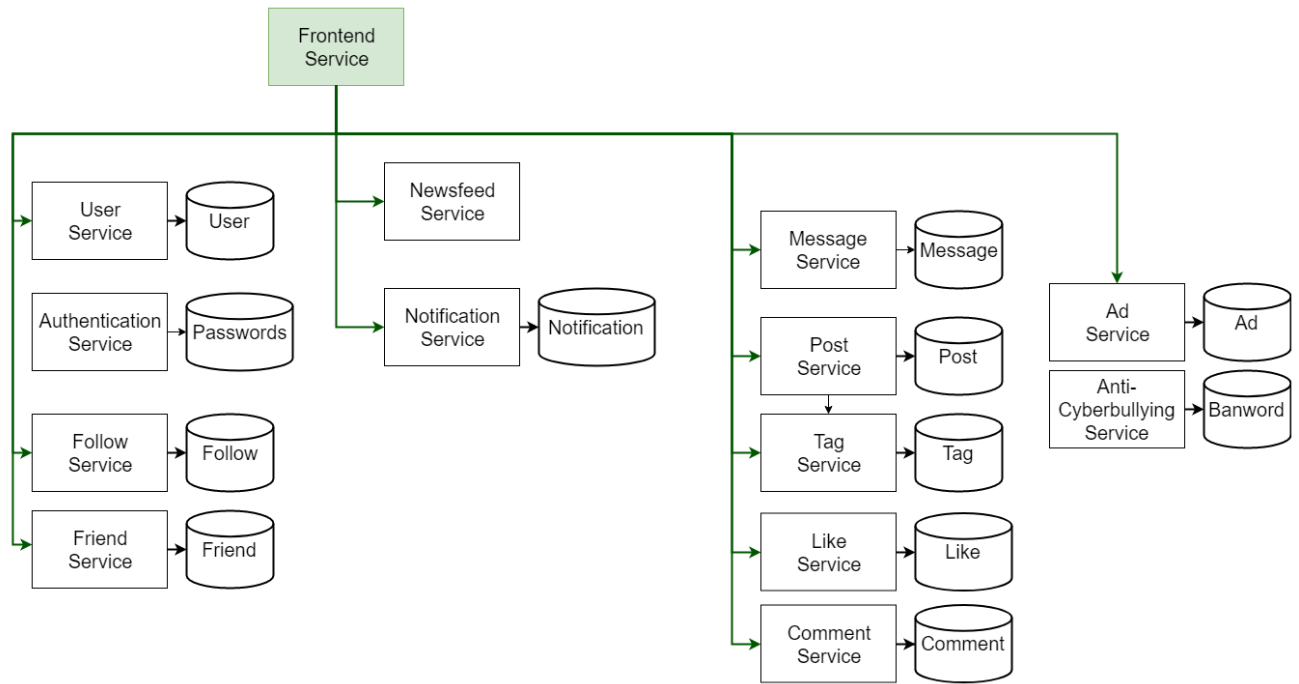


Figure 2: Architecture of core microservices plus frontend with their interaction

On top of all these components, we defined a specific service for the frontend as well (see figure 2). This service will of course communicate with our backend and make use of the provided api. As a side note, each request that is sent from the frontend implicitly goes through the authentication service to verify the user credentials that was stored in a token.

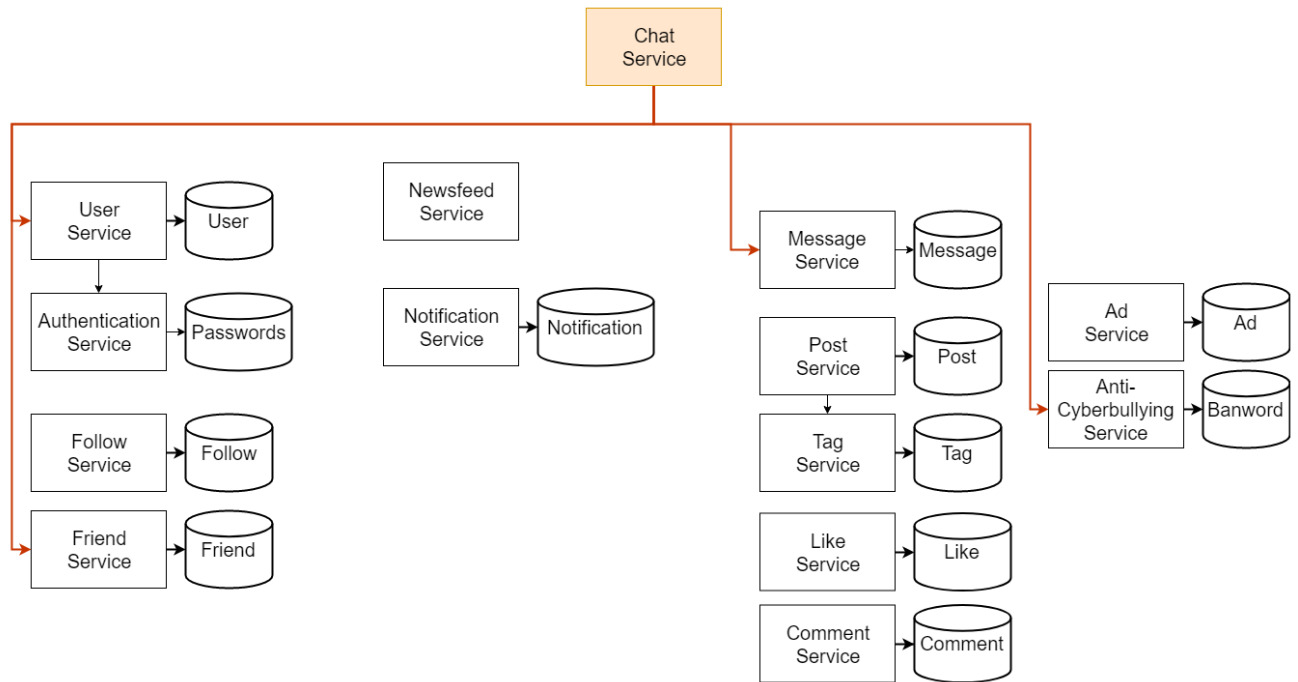


Figure 3: Architecture of core microservices plus chat service with their interaction

Similarly to the frontend service, there exists a standalone chat application as well. This application is as minimalistic as possible while still providing the basic functionality to login/register and sending messages to your friends.

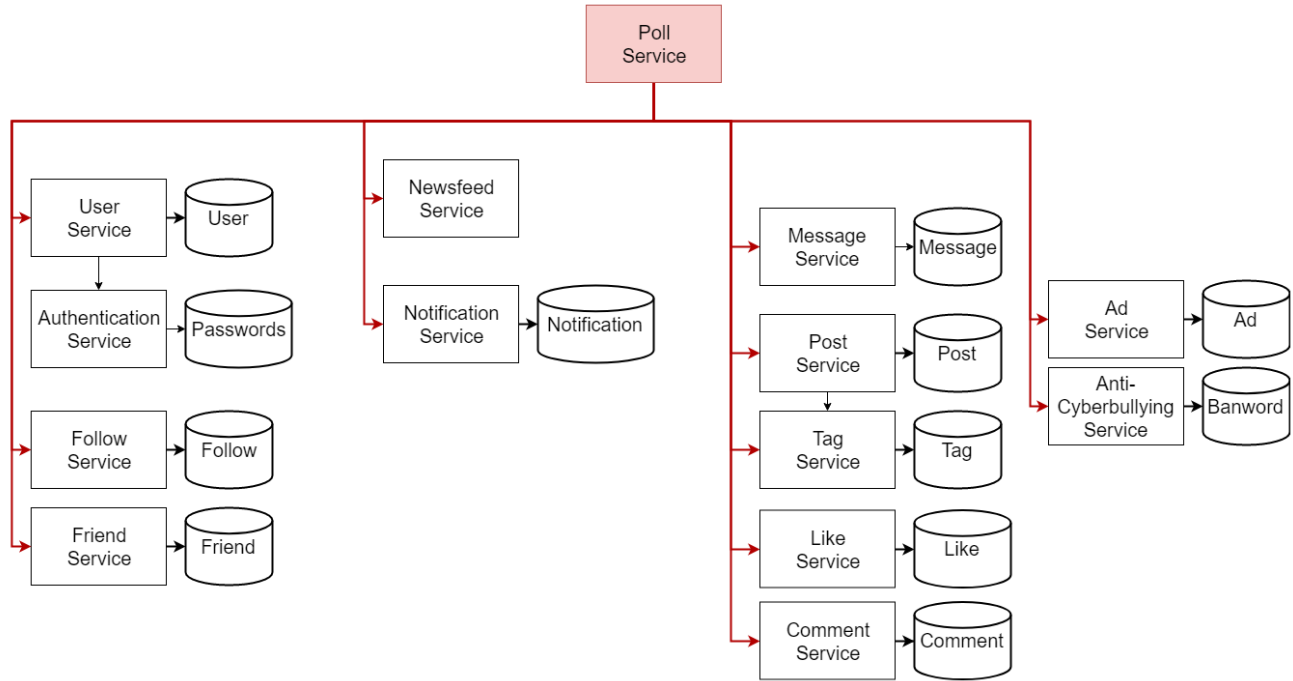


Figure 4: Architecture of core microservices plus poll service with their interaction

Lastly, we added a separate poll service as well that has the responsibility to poll (send ping request) every service (except user, authentication and notification) and if it detects that a certain service is down for some reason, it will then create a notification for all users respectively.

## 2 API Services

This section will go more into these core microservices by precisely defining their REST API. Note that most of these services also have a ping url.

### User service:

**In:** POST users (email, username, password)

**Out:** /

**Logic:** Registers or creates a user

**In:** POST users/login (username, password)

**Out:** Response indicating whether the login was successful or not (and return some token if it was successful)

**Logic:** Tries to login a user

**In:** GET users

**Out:** users (user\_id, email, username)

**Logic:** Gets all the users

**In:** GET users/user/username

**Out:** users (user\_id, email, username)

**Logic:** Gets a user by username

**In:** GET users/user\_id

**Out:** user (user\_id, email, username)

**Logic:** Get a particular user

**In:** DELETE users/user\_id

**Out:** /

**Logic:** Deletes a user

**In:** PUT users/user\_id/block

**Out:** /

**Logic:** Blocks a user

**In:** PUT users/user\_id/unblock

**Out:** /

**Logic:** Unblocks a user

#### **Authentication service:**

**In:** POST passwords (user\_id, password, is\_admin)

**Out:** /

**Logic:** Adds user and password pair to the database to verify tokens in the future

**In:** PUT passwords (password)  
**Out:** /  
**Logic:** Updates a user password

**In:** DELETE passwords  
**Out:** /  
**Logic:** Remove password from the database

**In:** PUT passwords is\_admin  
**Out:** Flag indicating whether the user is an admin or not  
**Logic:** Check if the user is an admin

**In:** GET verify\_credentials (token)  
**Out:** Flag indicating whether the authentication was successful  
**Logic:** Tries to authenticate a user by checking their credentials

**Message service:**

**In:** POST message (contents, sender\_id, receiver\_id)  
**Out:** /  
**Logic:** Add a new message to DB

**In:** GET messages/receiver\_id/sender\_id/<amount>  
**Out:** Messages (content, timestamp, id)  
**Logic:** Get messages in conversation between receiver and sender sorted by timestamp

**In:** GET messages/user\_id/unread  
**Out:** Messages (content, timestamp, id)  
**Logic:** Get all unread messages where the user is the receiver

**Notification service:**

**In:** POST notifications (content, recipients)  
**Out:** /  
**Logic:** Add notification to DB

**In:** GET notifications/notification\_id  
**Out:** Notification (ID, timestamp, content)  
**Logic:** Get a specific notification

**In:** PUT notifications/notification\_id (user\_id)  
**Out:** /  
**Logic:** Mark notification as read

**In:** GET notifications/user/user\_id  
**Out:** Notifications (ID, timestamp, content)  
**Logic:** Get all unread notifications from a specific user

**Ad service:**

**In:** POST ads (image, categories)  
**Out:** /  
**Logic:** Add advertisement image to DB (note that the images are actually saved locally instead of in the database; only the image path is stored)

**In:** GET ads  
**Out:** /  
**Logic:** Homepage for the ads service (only accessible for admins)

**In:** GET ads/filename  
**Out:** Advertisement (image link, category)  
**Logic:** Get a specific advertisement based on the filename

**In:** POST ads/user/user\_id  
**Out:** /  
**Logic:** Update (if necessary) the categories of a user (personalized ads)

**In:** GET ads/users/user\_id  
**Out:** Advertisements (image link, category)  
**Logic:** Get ads specifically targeted to a user if any

**Anti-Cyberbullying service:**

**In:** POST anti\_cyberbullying (words)

**Out:** /

**Logic:** Add new "bad words" to DB

**In:** GET anti\_cyberbullying

**Out:** /

**Logic:** Homepage for the anti\_cyberbullying service (only accessible for admins)

**In:** GET anti\_cyberbullying/bad\_words

**Out:** All the bad words

**Logic:** Get all the bad words from the database

**In:** POST anti\_cyberbullying/contains\_bad\_word (sentence)

**Out:** True if the provided sentence contains one or more bad words, false otherwise

**Logic:** Check if there's a bad word in the sentence

**Follow service:**

**In:** POST follow (follower\_id, followee\_id)

**Out:** /

**Logic:** User follows another user

**In:** DELETE follow/follower\_id/followee\_id

**Out:** /

**Logic:** Unfollow a user

**In:** GET followees/follower\_id

**Out:** Follows (follower\_id, followee\_id)

**Logic:** Get all users the follower follows

**In:** GET followers/followee\_id

**Out:** Follows (follower\_id, followee\_id)

**Logic:** Get all followers of the user

**Friend service:**

**In:** POST friend/request (friend\_initiator\_id , friend\_acceptor\_id)



**Out:** /  
**Logic:** Create a friend request

**In:** PUT friend/accept (friend\_initiator\_id , friend\_acceptor\_id)  
**Out:** /  
**Logic:** Accept a friend request

**In:** DELETE friend/friend1\_id/friend2\_id  
**Out:** /  
**Logic:** Unfriend a user

**In:** GET friend/user\_id/requests  
**Out:** Requests (friend\_id)  
**Logic:** Get all open friends requests sent to the user

**In:** GET friend/user\_id  
**Out:** Friends (friend\_id)  
**Logic:** Gets the friends of a user

**Newsfeed service:**

**In:** GET newsfeed/user\_id  
**Out:** Posts  
**Logic:** Get the newsfeed of a user (posts of the people the user is following  
+ own posts)

**Post service:**

**In:** POST posts (creator, content, tags (usernames))  
**Out:** /  
**Logic:** Create a new post

**In:** GET posts/user/user\_id  
**Out:** Posts (post\_id, creator, timestamp, content)  
**Logic:** Get all the posts from a user

**In:** GET posts/post\_id  
**Out:** Post (post\_id, creator, timestamp, content)  
**Logic:** Get a specific post

**In:** DELETE posts/post\_id  
**Out:** /  
**Logic:** Delete a specific post

**In:** GET posts/stats  
**Out:** Number of posts on each day  
**Logic:** Get the post statistics, i.e. number of posts on each day

**Like service:**

**In:** POST likes (post\_id, user\_id)  
**Out:** /  
**Logic:** User likes a specific post

**In:** GET likes/posts/post\_id  
**Out:** Likes (post\_id, user\_id)  
**Logic:** Get all the likes of a post

**In:** DELETE likes/posts/post\_id (user\_id)  
**Out:** /  
**Logic:** Undo a like

**Tag service:**

**In:** POST tags (post\_id, user\_id's)  
**Out:** /  
**Logic:** Create user tags on a specific post

**In:** GET tags/posts/post\_id  
**Out:** Tags (post\_id, user\_id)  
**Logic:** Get all the tags on a post

**Comment service:**

**In:** POST comments (post\_id, creator, content)

**Out:** /

**Logic:** Create a comment on a specific post

**In:** GET comments/comment\_id

**Out:** Comment (comment\_id, creator, timestamp, content)

**Logic:** Get a specific comment

**In:** GET comments/posts/post\_id

**Out:** Comments (comment\_id, creator, timestamp, content)

**Logic:** Get all the comments on a post

**In:** DELETE comments/comment\_id

**Out:** /

**Logic:** Delete a comment