# Case Study for learning Cloud Application Programming Model

## Use case

Develop chat service which will provide to user A send a text message to user B or to several users.

The chat service will consist of two parts:

1. Central service – responsible for onboarding and routing messages from user A to user B. This will have only one deployed instance in SAP Cloud Platform. Central service will not store message for end user but keep message in queue if it cannot be delivered. Central service will have own persistence.
2. User service – client service which will connect to the central service, get the authorization to send message to central service. Service can be deployed as multiple instances on SCP. User service has own persistence and messages for user are stored in own client.



## Central Service – Requirements

## 

## User On/Off Boarding

1. Create new user with following personal data:
   1. Name, Surname, Nickname
   2. Address Data, mail, mobile
   3. Avatar (foto)
   4. MHP Department
   5. Generate unique ID for user. As alternative – use S\* user of MHP User ID (technical)
   6. Flag: active/not active (technical)
   7. Host URL of user service – can be added later.
   8. Client registered: true/false
   9. Client ID

During creation of user check that user ID is unique.

1. Edit user and change all data, except ID
2. Delete user. Technically – deactivate and leave user in DB.

Error handling:

* Validate User ID (if it will be entered manually)
* Define mandatory fields

## Message dispatching and queuing

Message from user client will not be stored permanently on central service but stored in local queue of service. After successful delivery, the message removed from queue and log entry is written to log.

Local queue must contain mandatory attributes like:

* Sender (ID) -client ID
* Recipient (ID) – client ID
* Message ID -
* Message Text
* Date/Time of receiving message in queue
* Number of attempts
* Scheduled message – send message based on time
* Scheduled DateTime

The task of service to go over the queue with status failed and number of attempts is not maximum, find the recipient, connect to recipient and send message.

If message was sent successfully, add to log and delete from queue

If message was not sent successfully and it was last attempt – also delete from queue and put to log.

## Queueing configration

Queue should be configurable.

Attempts: how many attempts it should do before it finally stopped to send.

Intervals: interval for each attempt (sec)

## Logging

Log must contain following attributes:

* Sender (ID)
* Recipient (ID)
* Message ID
* Date/Time of receiving message in queue
* Status (DELIVERED/FAILED)
* Number of attempts
* Date/Time of attempt to send

## API:

The following API has to be provided:

Users:

* Get user list (with filtering) – needed for client to get list of registered users
* Create user
* Update user
* Delete user
* Pair client

Messages:

* Add message to queue
* Delete message from queue
* Get queue (filter)

Log:

* Add log entry
* Get log (with filter)

# Client Service – Requirements

Is separate application deployed to CP. Has own persistence.

In current design each User has own application, e.g. Client-a, client-b, etc.

## Connection to central service

### Pairing with central service.

It should be possible to establish trust between central service and user client. Therefore, at first connection, client must generate client ID (like a token) and give it to main service. Main service check whether Client ID is already given and if not – register client.

Client service must also provide own host URL to the central service.

Security: “Authorization-Code”

### Sending message to another user

User can select list of users from central service and select user or multiple to whom he wants to send message.

After selection of user(s) he can write text message. Text length is restricted by 1000 chars.

After pressing send – message saved to local database with parameters:

* User ID (recipient)
* Message ID
* Text
* Date / Time
* Type of message: Inbound(received) or outbound(sent)

And the users to who message was sent also should be collected in local database that user can get them without connecting to central service.

* User ID
* Name/Surname
* Avatar

### Receiving message from another user

Client service provide API to receive message

* Add message

The message should be saved in the same table as for sent messages.