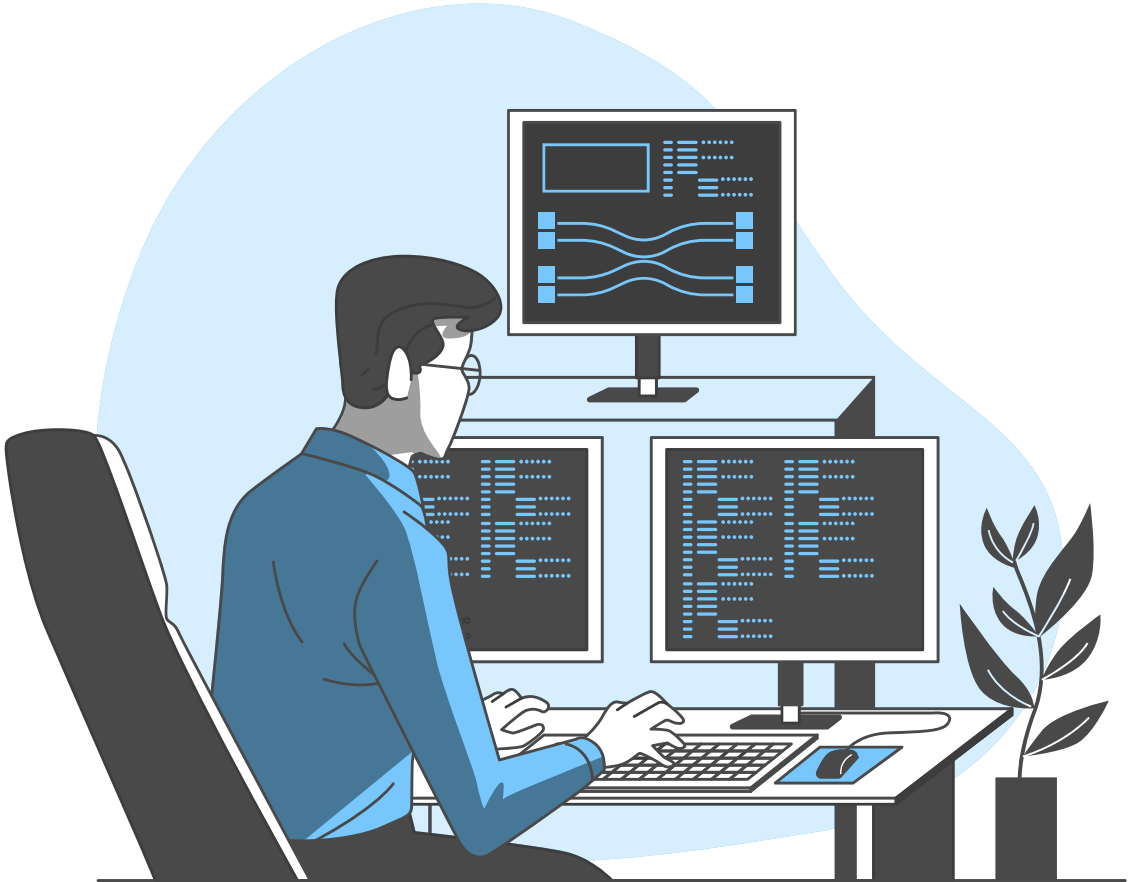
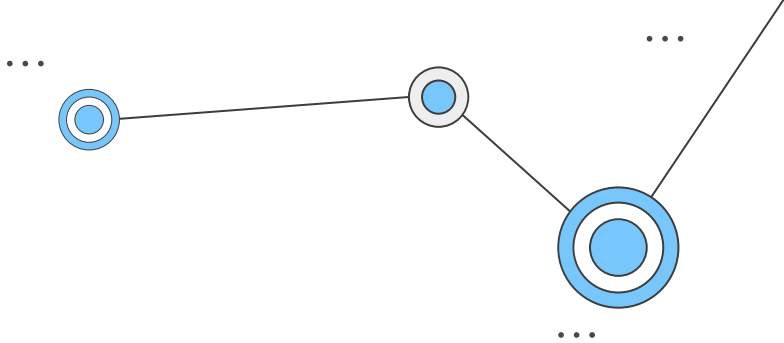
xideral®



Final Project: Back-End Infrastructure for a Microblogging App

By Kenneth Rodríguez



Project Overview

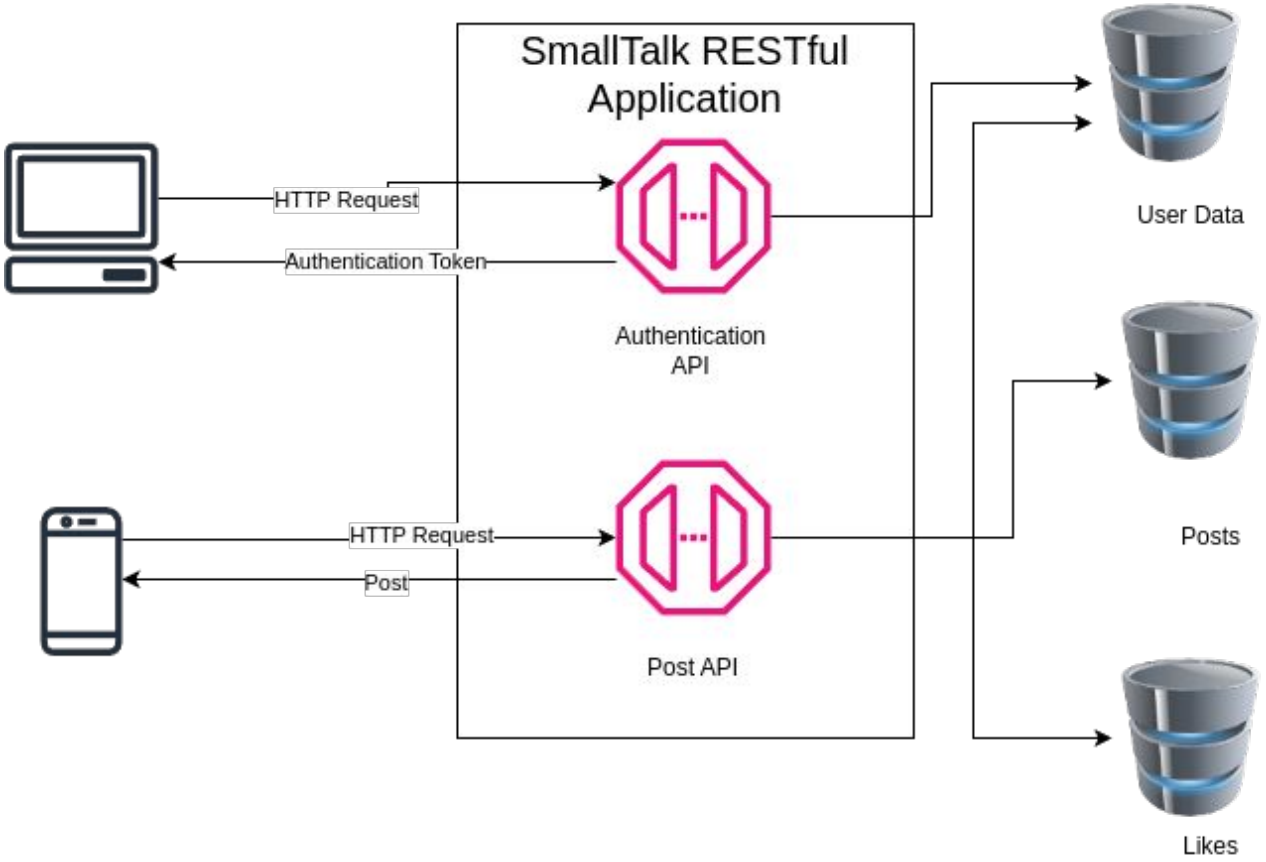
The main objective of this project is to develop the basic back end infrastructure for a microblogging social media platform called “**SmallTalk**”. This platform will enable users to create and interact with short-form text content.

Scope

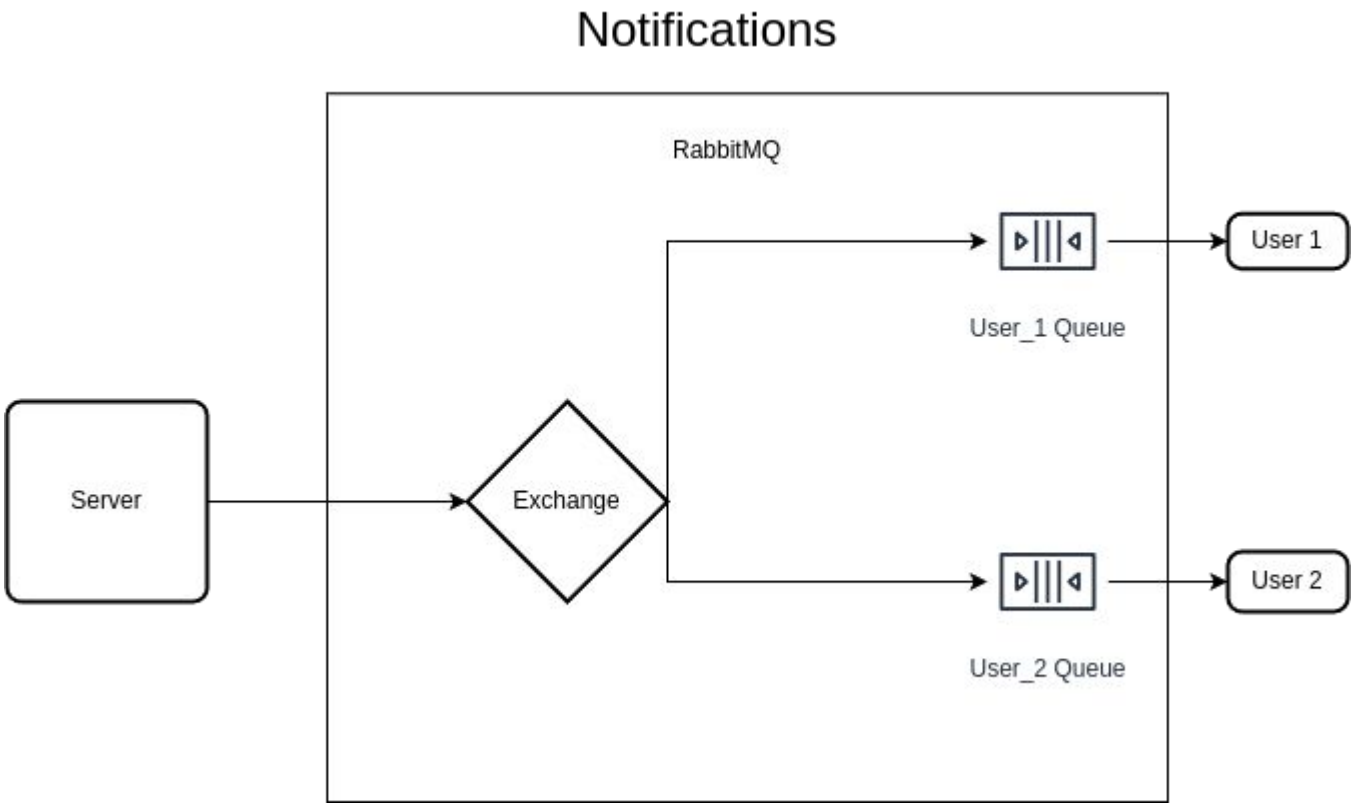
The project will consist of a few key components to provide the necessary infrastructure for a minimum viable product.

- **User registration and authentication:** users should be able to create accounts, log in, and log out securely.
- **Post creation:** Users will be able to publish short text-based posts to the public feed.
- **Post feed:** All users will see the most recent posts published by other users in the platform.
- **Post liking:** Users can like other users' posts, as well as their own, and the liked content should be persistent.
- **Real time updates:** Users will receive live updates when their content has been liked by other users.
- **Backup job:** A batch job to backup all of the system's database should run on a scheduled basis.

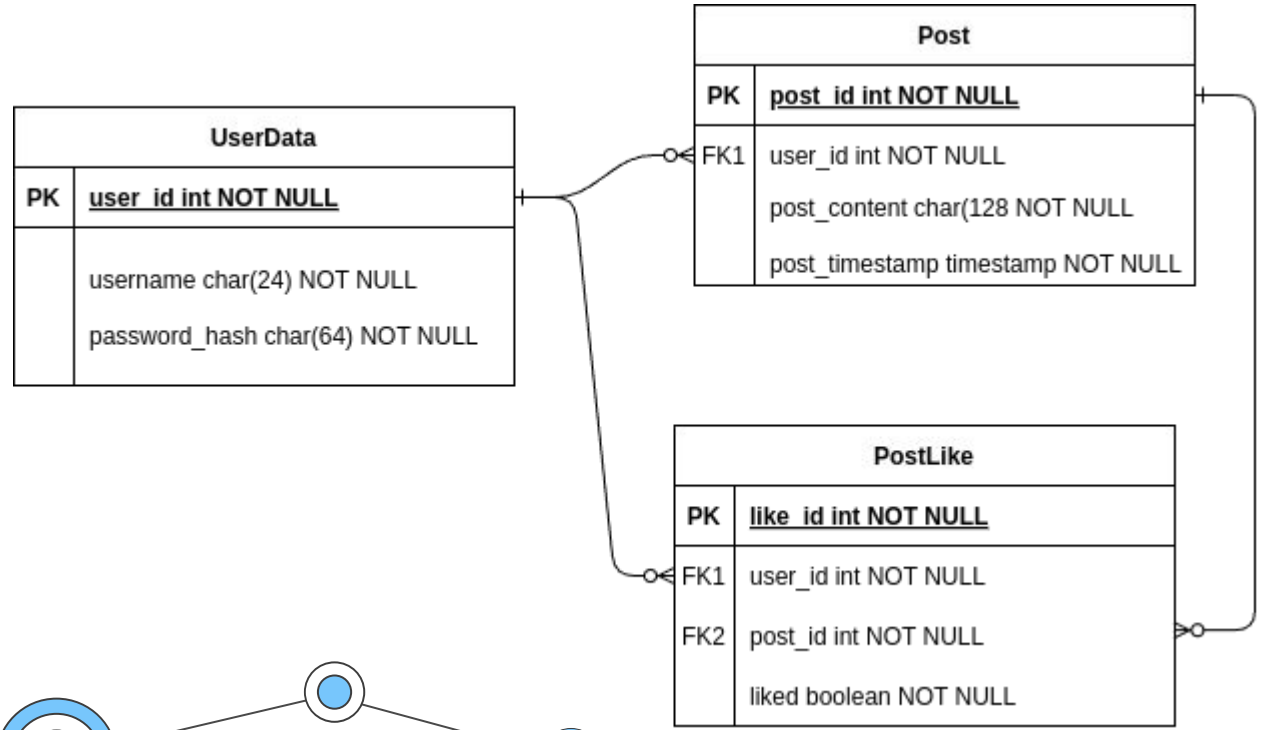
System Architecture

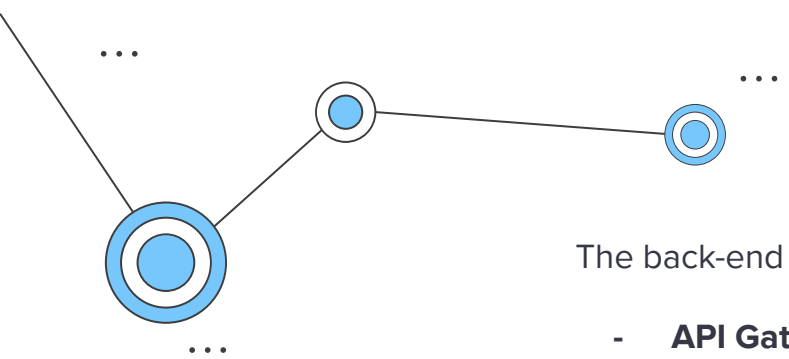


System Architecture



Database

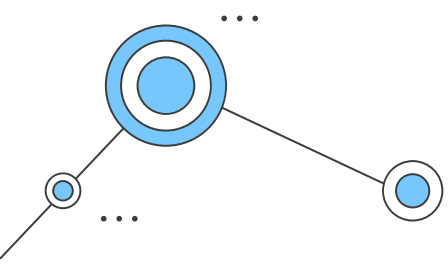

UserData
PK
user_id int NOT NULL
username char(24) NOT NULL
password_hash char(64) NOT NULL
Post
PK
post_id int NOT NULL
FK1
user_id int NOT NULL
post_content char(128 NOT NULL
post_timestamp timestamp NOT NULL
PostLike
PK
like_id int NOT NULL
FK1
user_id int NOT NULL
FK2
post_id int NOT NULL
liked boolean NOT NULL



Back-End Services

The back-end infrastructure of the project is the basis of the entire service.

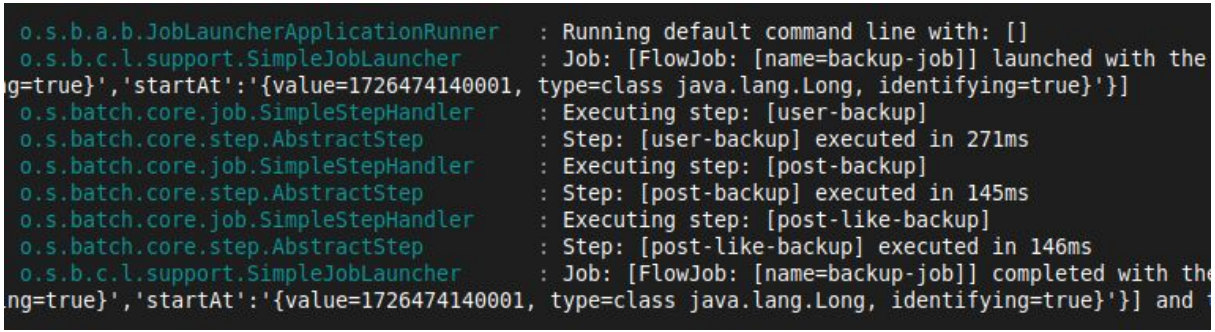
- **API Gateway**: The server provides a REST API entrypoint, which handles all incoming requests and routes them to the appropriate service.
- **User Service:** The user service handles requests for user registration and login.
- **Post Service:** The post service publishes and retrieves posts.
- **Like Service:** The like service updates the likes of a user created post, and triggers the notification service to notify the relevant users.
- **Notification Service:** The notification service sends a notification to a message broker which will route the message to its destination on the client side.
- **Back Up Job:** The backup job creates a daily backup of the existing database, to prevent data loss.



Batch Backup Job

The Backup Job is a scheduled job which runs every day at midnight. When triggered, the service will use the Spring Batch component to backup all of the available databases to .csv files, providing a way of rolling back the database in case of failure or data corruption.

The backup job consists of three steps, where each step is responsible for backing up one of the three different tables in the database.



Message Broker – RabbitMQ

To provide users with real time notifications, it is necessary to use a message-brokering system, which will allow users to receive updates without directly receiving the message from the application.

The RabbitMQ instance will receive messages from the server in an **Exchange** and send them to the appropriate **Queue**, where messages wait to be consumed by a client.

The **Notification service** sends updates to the Exchange named “**smalltalk.direct**”. This Exchange sends a copy of the message only to the queues where the routing key is an exact match.

Exchanges

All exchanges (9)

Pagination

Page 1 of 1 - Filter: Regex ?

Name	Type	Features	Message rate in	Message rate out	+/-
(AMQP default)	direct	D			
amq.direct	direct	D			
amq.fanout	fanout	D			
amq.headers	headers	D			
amq.match	headers	D			
amq.rabbitmq.trace	topic	D I			
amq.topic	topic	D			
amqp.fanout	topic	D			
smalltalk.direct	direct	D	0.00/s	0.00/s	

Queues

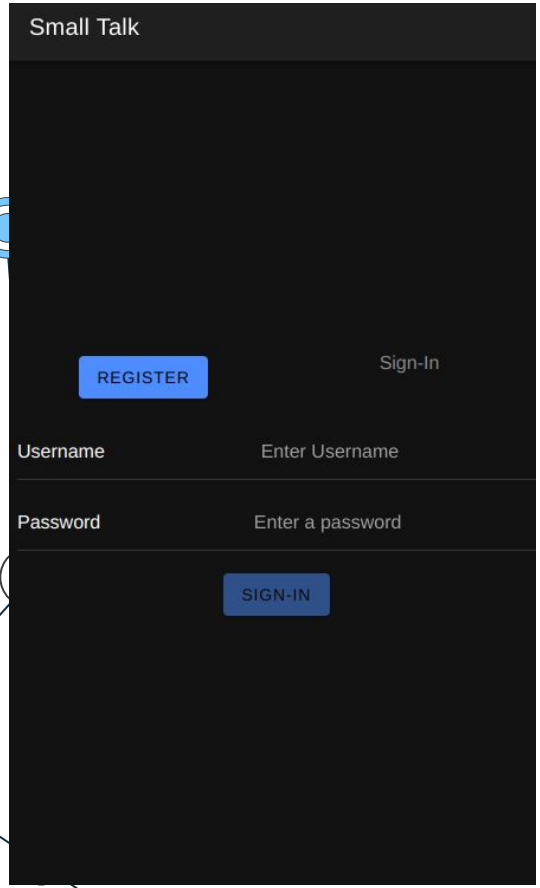
All queues (4)

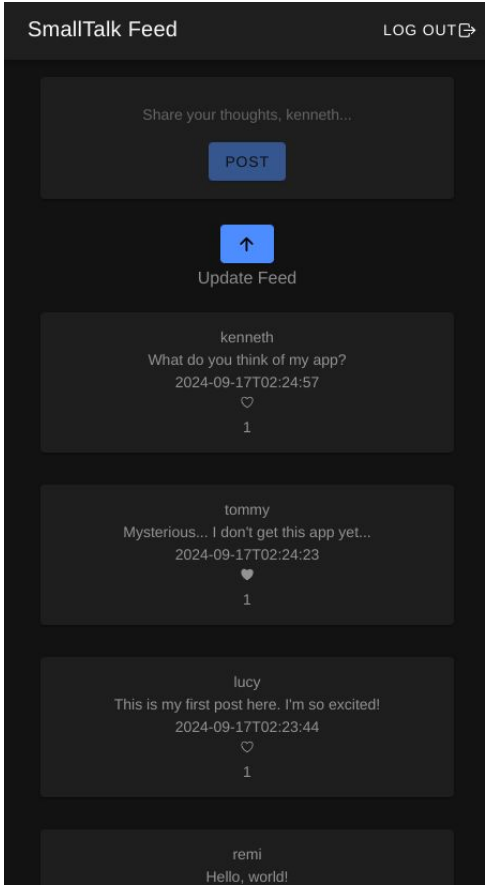
Pagination

Page 1 of 1 - Filter: Regex ?

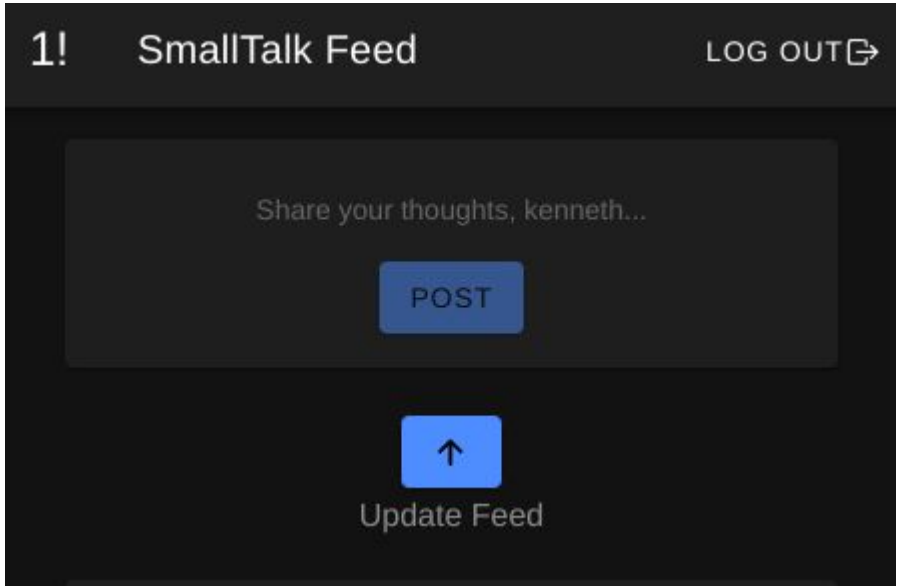
Overview				Messages			Message rates			+/-
Name	**Type**	Features	**State**	**Ready**	**Unacked**	**Total**	**incoming**	**deliver / get**	**ack**	
kenneth	classic	D	idle	0	0	0	0.00/s	0.00/s	0.00/s	
lucy	classic	D	idle	0	0	0	0.00/s	0.00/s	0.00/s	
remi	classic	D	idle	0	0	0	0.00/s	0.00/s	0.00/s	
tommy	classic	D	idle	0	0	0	0.00/s	0.00/s	0.00/s	

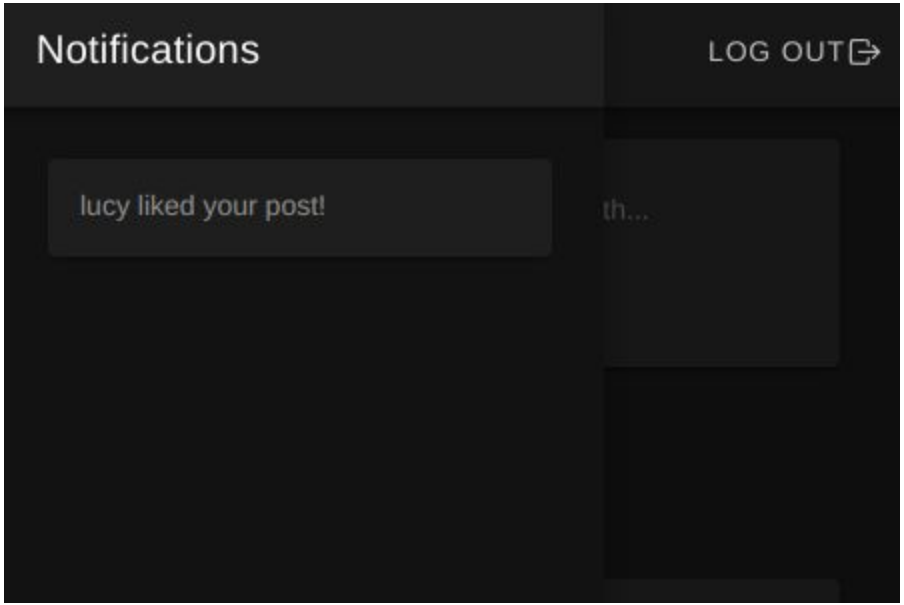
Front-End


Small Talk
REGISTER
Sign-In
Username
Enter Username
Password
Enter a password
SIGN-IN


SmallTalk Feed
LOG OUT
Share your thoughts, kenneth...
POST
Update Feed
kenneth
What do you think of my app?
2024-09-17T02:24:57
1
tommy
Mysterious... I don't get this app yet...
2024-09-17T02:24:23
1
lucy
This is my first post here. I'm so excited!
2024-09-17T02:23:44
1
remi
Hello, world!

Front-End (Notifications)





Testing

To ensure that everything works properly, it is necessary to set up a proper testing process that validates the correct execution of the application. To achieve this, Spring Boot provides the tools necessary to test the program.

We can leverage the JUnit 5 and Mockito frameworks to set up appropriate unit tests for the project. These dependencies will allow us to design specific tests that compare the expected behavior of the application against its actual execution.

```
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 4.371 s -- in com.kennethrdzg.smalltalk.AppTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  6.662 s
[INFO] Finished at: 2024-09-17T02:38:40-06:00
[INFO] ------------------------------------------------------------------------
```

Deployment

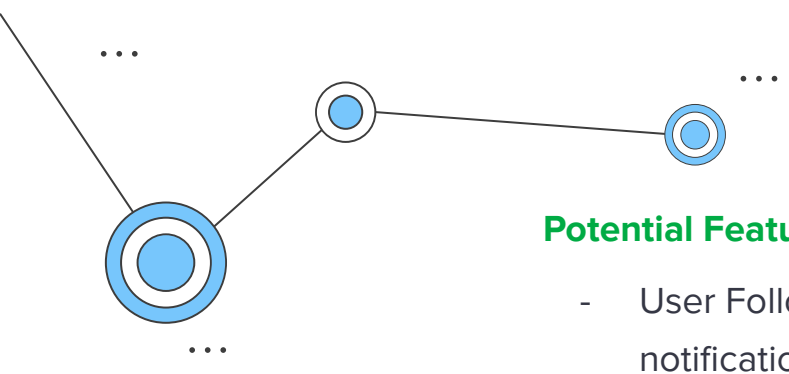
Deployment Environment

As a Java application, the web server can run on virtually any machine that runs the JVM with Java 17.

The environment variables, such as the login credentials for MySQL and RabbitMQ should be in a shell script named “**env_variables.sh**”.

Deployment Process

The “**run.sh**” script, found in the project’s files, should be executed in a shell from within the server’s directory. The script will call the environment variables script, and perform the cleaning, compiling, testing and execution process using Maven.



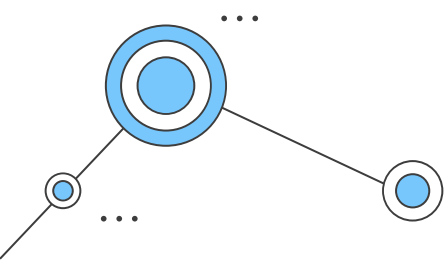
Future Considerations

Potential Features

- User Following: Users could potentially follow other users and receive notifications when they post new content.
- Multimedia User Content: The platform could allow for content upload other than text, such as images, audio, or video.
- Password Salting: To increase security, the password salting technique could be implemented on top of the current password hashing.

Technology Considerations

- Containerization: By using tools like Docker and Kubernetes, multiple instances of the application could be easily deployed for improved scalability.
- Cloud Technologies: The project could eventually be moved from a local server to a virtual machine in a cloud provider.



The End