real-time frontend updates

TwitterV2

Daniil Blagoev

Contents

[Introduction & Problem Statement 2](#_Toc169513446)

[Research questions & methods 2](#_Toc169513447)

[Answers 2](#_Toc169513448)

[How can TwitterV2’s tweet service communicate with the frontend? 2](#_Toc169513449)

[Which method of communication between TwitterV2’s tweet service and the frontend is most optimal? 3](#_Toc169513450)

[How can the method of communication from the previous sub-question be implemented into TwitterV2? 4](#_Toc169513451)

[Conclusion 6](#_Toc169513452)

[Sources 7](#_Toc169513453)

# Introduction & Problem Statement

The purpose of this research is to solve the following problem: When a user tweets, other users of TwitterV2 (only his followers in an ideal case) need to receive a real-time update to their feed with the new tweet, which is not the case currently. There are multiple different ways to solve this problem, and, in this document, I will evaluate them and see which one would be most fit to solve the problem.

# Research questions & methods

Main research question:

* How can I automatically push a newly created tweet to TwitterV2’s frontend?

Sub-questions:

* How can TwitterV2’s tweet service communicate with the frontend?
  + Literature study
    - Googling to find out what ways of communication between a service and the frontend there are.
* Which method of communication between TwitterV2’s tweet service and the frontend is most optimal?
  + Problem analysis
    - Analyze TwitterV2’s problem in more detail.
  + SWOT analysis
    - Evaluate the pros and cons of the different methods of communication.
* How can the method of communication from the previous sub-question be implemented into TwitterV2?
  + Available Product Analysis
    - Check if there are any examples of implementation on the internet which would be useful in my case.

# Answers

## How can TwitterV2’s tweet service communicate with the frontend?

The different ways the tweet service can communicate with the frontend in real-time while keeping them decoupled are:

* Server-Sent Events (SSE) - Server-Sent Events (SSE) is a server push technology enabling a client to receive automatic updates from a server via an HTTP connection and describes how servers can initiate data transmission towards clients once an initial client connection has been established. They are commonly used to send message updates or continuous data streams to a browser client and designed to enhance native, cross-browser streaming through a JavaScript API called EventSource, through which a client requests a particular URL in order to receive an event stream.[[1]](#footnote-1)
* Message Queues - A message queue is a form of asynchronous service-to-service communication used in serverless and microservices architectures. Messages are stored on the queue until they are processed and deleted. Each message is processed only once, by a single consumer. Message queues can be used to decouple heavyweight processing, to buffer or batch work, and to smooth spiky workloads.[[2]](#footnote-2)
* WebSockets - A WebSocket is a communication protocol that provides full-duplex communication channels over a single TCP connection. It enables real-time, event-driven connection between a client and a server. Unlike traditional HTTP software, which follows a request-response model, WebSockets allow two-way (bi-directional) communication. This means that the client and the server can send data to each other anytime without continuous polling.[[3]](#footnote-3)
* ASP.NET SignalR - A library for ASP.NET developers that simplifies the process of adding real-time web functionality to applications. Real-time web functionality is the ability to have server code push content to connected clients instantly as it becomes available, rather than having the server wait for a client to request new data.[[4]](#footnote-4)

## Which method of communication between TwitterV2’s tweet service and the frontend is most optimal?

When users post tweets, their followers should receive the new tweets in their feed automatically. For this, the tweet service must be able to “talk” to the frontend when a new tweet is created, and it should be done as close to real-time as possible. The characteristics of the solution should be the following:

* Scalable
* Secure
* Efficient in supplying multiple frontends with updates

Now it is time to evaluate each method of communication from the backend to the frontend:

* SSE – SSE would be a good choice if it didn’t have to create a new connection to thousands of different frontends. This limits its scalability quite a lot. Aside from that, it cannot communicate in different topics, which means each frontend which it is connected to will be seeing the messages and it would have to make use of additional filtering to know if the message is for it or not.
* Message Queues – Message queues would be a good choice for scalability and efficiency, but not for security. The frontend would have to be a Kafka consumer as well as an interface and interaction component, which is bad for separation of concerns.
* WebSockets – WebSockets are a good choice, since they are lightweight and scalable forms of communication. The problem with them is that, just like with SSE, additional filtering would have to be done in each frontend to find out of the message is for it or not.
* SignalR – SignalR is almost as lightweight as WebSockets, but the thing which makes it stand out most is its “Groups” functionality. By making use of it, each frontend can subscribe to the corresponding group depending on the logged in user and receive updates only relevant to it. Additionally, in case of a network interruption, it tries to automatically reconnect, which is bonus points for scalability.

Based on this comparison, implementing SignalR would be the best solution to the problem.

## How can the method of communication from the previous sub-question be implemented into TwitterV2?

The implementation of SignalR would roughly involve the following steps:

A computer screen shot of text

Description automatically generated

Figure 1 Creating a SignalR hub in the tweet service

A screenshot of a computer program

Description automatically generated

Figure 2 Configuring the gateway

A screenshot of a computer program

Description automatically generated

Figure 3 Configuring the React frontend

The workflow would be the following:

Prerequisites – user A follows user B and user B is logged in

Workflow – user A logs in -> user A’s frontend gets all users which user A follows -> user A’s frontend joins a different group for each different user in the following list -> user B tweets -> the tweet service processes the tweet -> SignalR uses user B’s name as the group name to send a message to -> user A’s frontend receives the message and updates.

# Conclusion

Pushing a newly created tweet to TwitterV2’s frontend can be done using multiple different technologies, but the one which fits TwitterV2’s situation the most is SignalR. Following [the steps of implementation](#_How_can_the) should provide a working solution to the problem of pushing newly created tweets to the frontend.

# Sources

*Server-sent events*. (2024, May 9). Wikipedia. https://en.wikipedia.org/wiki/Server-sent\_events

*What is a Message Queue?* (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/message-queue/

Bradygaster. (2020, September 9). *Introduction to SignalR*. Microsoft Learn. https://learn.microsoft.com/en-us/aspnet/signalr/overview/getting-started/introduction-to-signalr

*What are WebSockets?* (n.d.). PubNub. https://www.pubnub.com/guides/websockets/

1. *Server-sent events*. (2024, May 9). Wikipedia. https://en.wikipedia.org/wiki/Server-sent\_events [↑](#footnote-ref-1)
2. *What is a Message Queue?* (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/message-queue/ [↑](#footnote-ref-2)
3. *What are WebSockets?* (n.d.). PubNub. https://www.pubnub.com/guides/websockets/ [↑](#footnote-ref-3)
4. Bradygaster. (2020, September 9). *Introduction to SignalR*. Microsoft Learn. https://learn.microsoft.com/en-us/aspnet/signalr/overview/getting-started/introduction-to-signalr [↑](#footnote-ref-4)