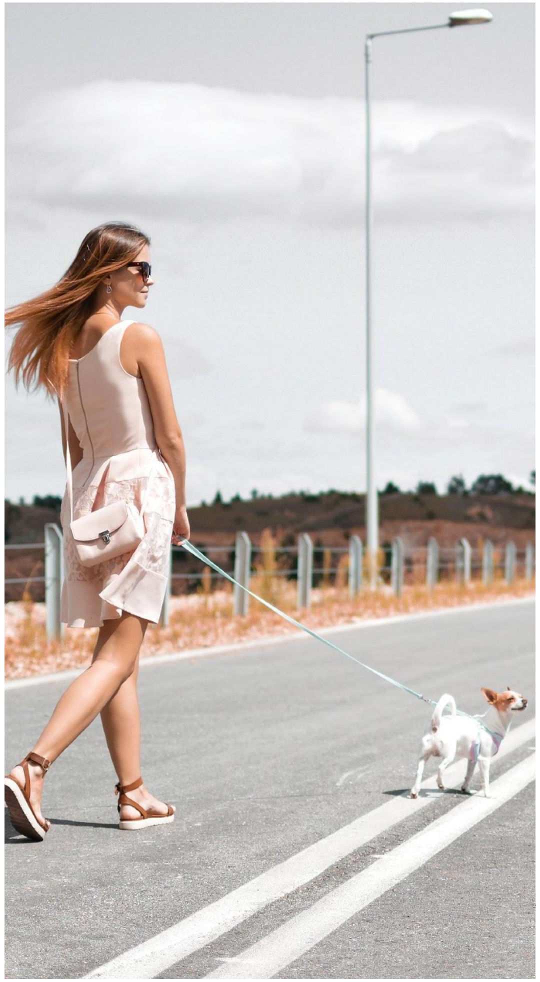
Using an Event Bus

Theory to practice
2018-03-23

vente-privee privalia



Summary

Using an Event Bus

Event Bus Theory

- What is an Event Bus?
- What is an Event?
- How does an event look like?
- Is it mandatory to use a Message Broker?

Arch. Using RabbitMQ

- What is RabbitMQ?
- RabbitMQ components.
- Architecture.

Demo Time

Event Bus Theory

What is an Event Bus?

An Event Bus is a mechanism that allows different microservices to **communicate with each** other **without knowing about each other**.

A microservice can **send an Event to the Event Bus** without knowing who will pick it up or how many others will pick it up.

Microservices can also **listen to Events on an Event Bus**, without knowing who sent the Events.

That way, components can communicate without depending on each other.

What is an Event?

An Event is basically a **message** sent by a microservice to indicate something happened in the domain.

An event has a **name**, a **payload** and usually a **timestamp**.

The name determine the kind of event and should be written with a past-participle verb, such as ***OrderReceived***.

The payload should contains everything that the receiver needs to know in order to process the Event.

How does an event look like?

An Event can be represented in **JSON format**:

```
{
  "name": "UserRegistered",
  "createdAt": "2018-01-17T18:45:58+00:00",
  "payload": {
    "id": 453340053,
    "first_name": "John",
    "last_name": "Smith",
    "email": "john.smith@example.com"
  }
}
```

How does an event look like?

An Event can be represented also as a **Java class**:

```
public class UserRegistered {
    private long id;
    private String firstName;
    private String lastName;
    private String email;
    private Date createdAt;

    public UserRegistered(long id, String firstName, ...) {...}
    public String getEventName() { return "UserRegistered"; }
    public Date getCreatedAt() { return createdAt; }
    public Map<String, String> getPayload {...}
}
```

Is it mandatory to use a Message Broker?

It isn't. But it's **highly recommended**.

A Message Broker is a software component used for passing messages between modules or applications of an organization with provides **security, reliability & fault tolerance**.

Message Broker systems are also called message-oriented middleware and queuing systems.

Examples of Message Brokers are **RabbitMQ** and **Apache Kafka**.

Architecture using RabbitMQ

What is RabbitMQ?

RabbitMQ is an **open source message broker** software which implements the Advanced Message Queuing Protocol (AMQP).

The Advanced Message Queuing Protocol (AMQP) is an open standard application layer protocol for **passing business messages between applications or organizations**.

It it a robust messaging system which provides message orientation, queuing, security, reliability, fault tolerance and routing system, including point-to-point and publish-and-subscribe.

RabbitMQ components

Message - A piece of information. It can be a text (i.e a json) or binary data.

Exchange - Receives messages from producers and it pushes them to queues using the routing key if necessary.

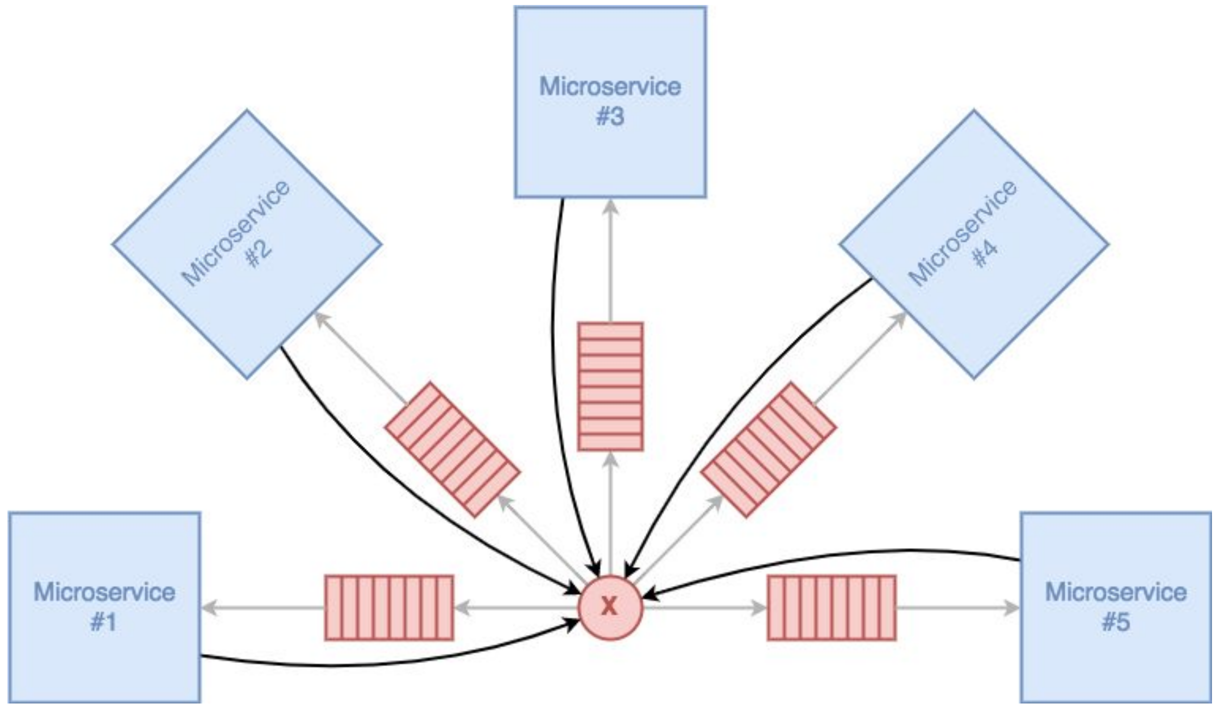
Queue - Essentially an infinite buffer which stores the messages to be consumed.

Producer - A program that sends messages to the exchange or eventually to the queue.

Consumer -A program who receives messages from the queue.

The Event Bus in RabbitMQ

Architecture (i)



The Event Bus in RabbitMQ

Architecture (ii)

When something happens in the domain an event is **sent to the exchange**, which represents the event bus, using the **event name as the key**.

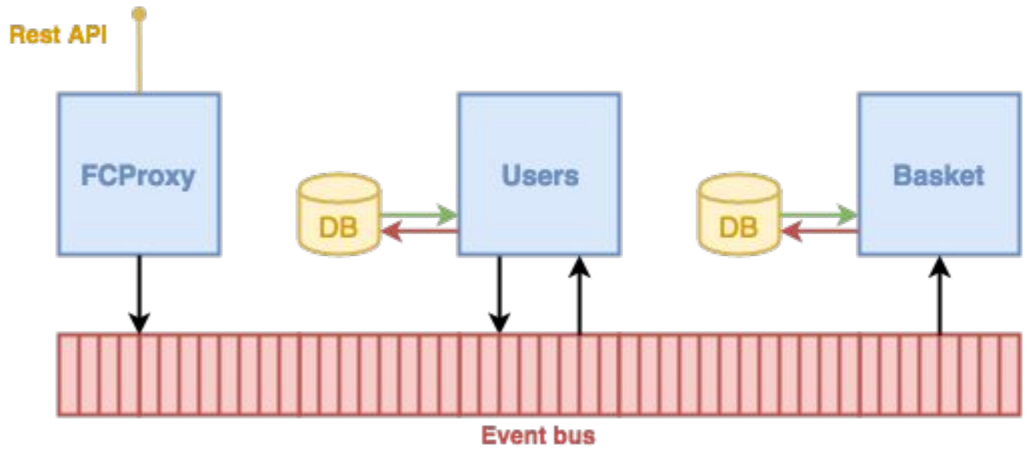
The exchange **enroutes the event** to the microservices who need this event. Not all the microservices receive all the events.

That means that the **routing logic is delegated** to RabbitMQ and the microservices don't have to deal with events they are not processing.

The routing has to be set as binding rules from the exchange to the queues

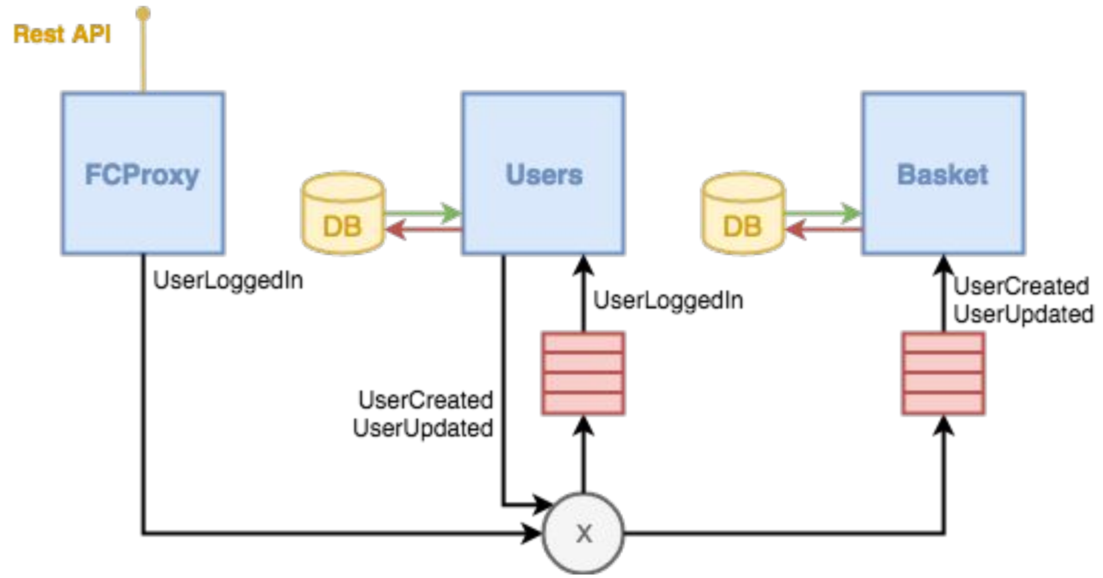
Demo Time

Schema



Using an Event Bus

Schema



vente-privee GROUP