Messaging

Obsah prezentácie

Definícia

Priamy messaging

Asynchrónny messaging

Stavba správy

Výmena správ

Ukážka na príklade

Implementácie

Záver + diskusia

Definícia

Architektonický vzor, ktorý popisuje, ako sa (nie len) dve rôzne časti aplikácie alebo rôzne systémy spájajú a komunikujú

Priamy (synchrónny) messaging

REST, SOAP, ...

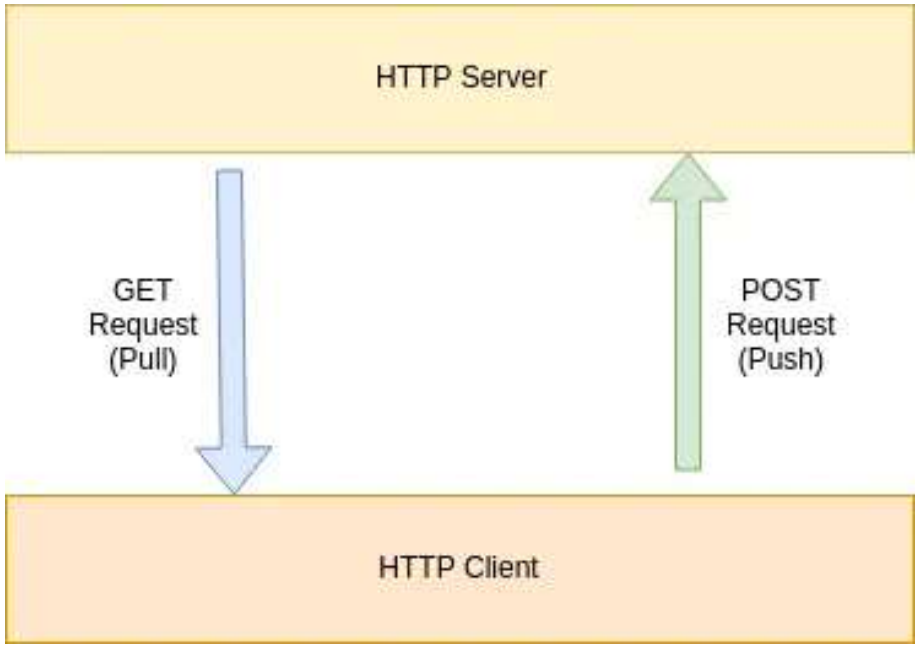
Priame odoslanie/vyžiadanie dát

s očakávaním na odpoveď

- Tight Coupling

- Blocking

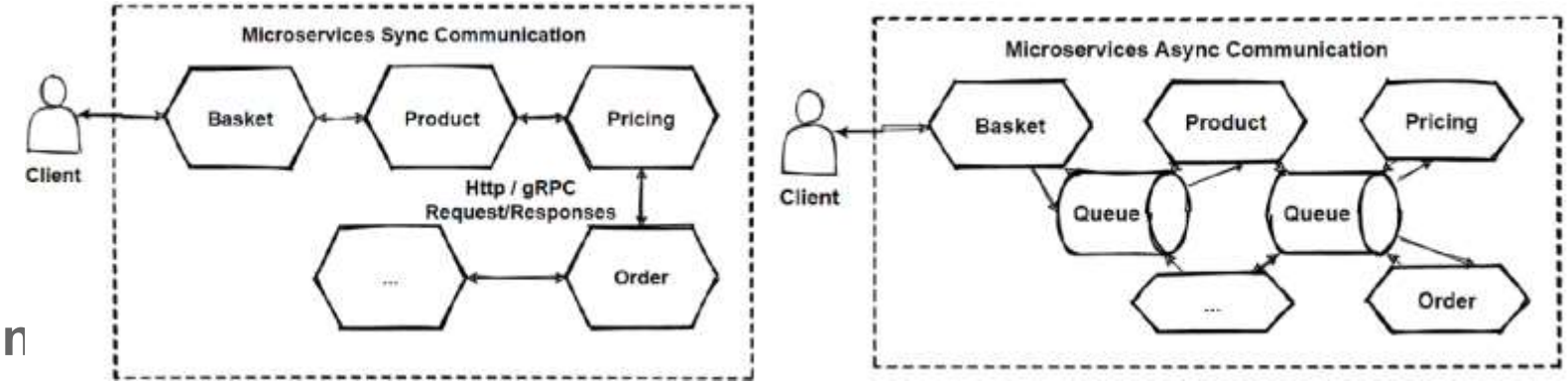
- Error Handling



Asynchrónny messaging

Komunikácia častí systému pomocou správ (udalostí) cez message borker

Nie je nutnosť čakať na odpoveď



+ Loose Coupling

+ Non-Blocking

+ Jednoduché na škálovan

+ Jednodnoduchý error handling

Správa

Údaje vymieňané medzi časťami aplikácie

SOAP Message, Message, ...

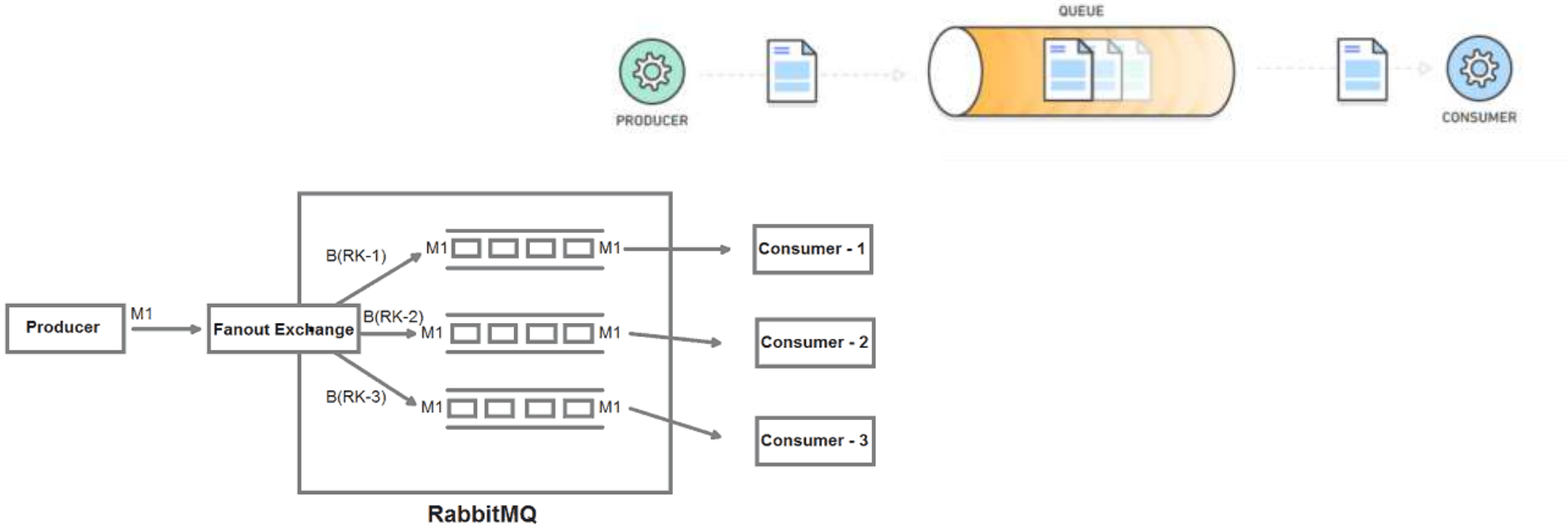
Základné dve časti:

- **Hlavička –** informácie pre messaging systém
- **Telo –** prenášané dáta

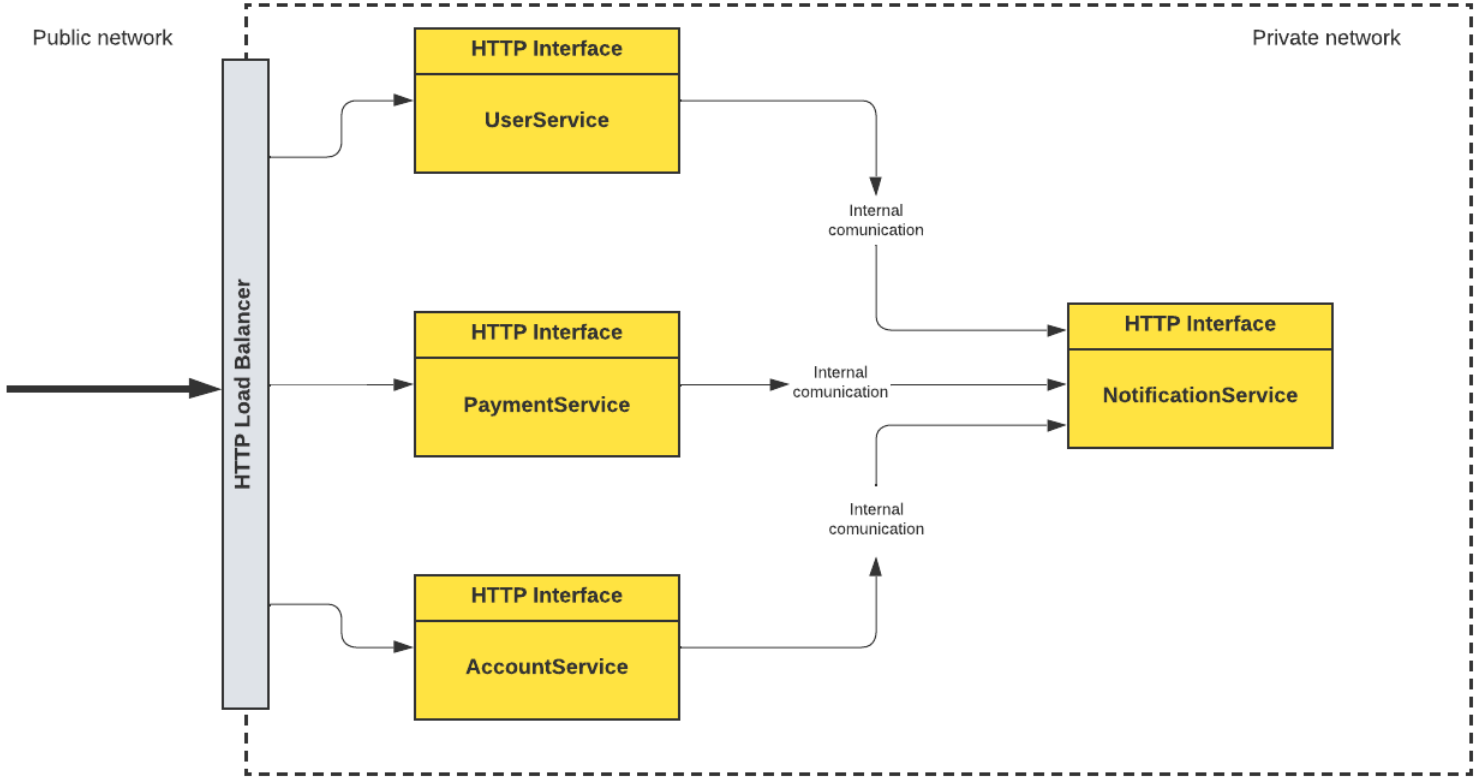
Výmena správ

Point-to-point (queues)

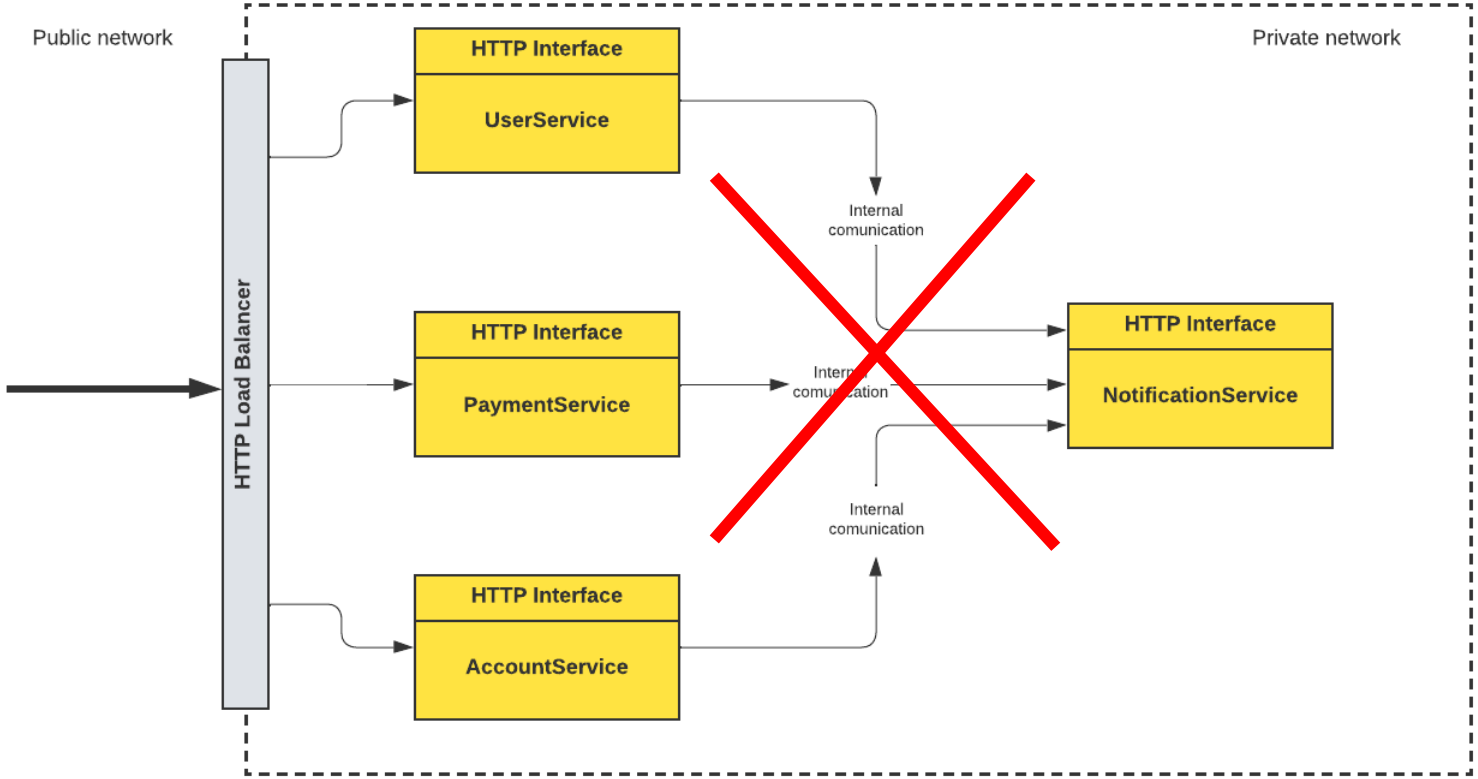
Publish-subscribe (exchanges)



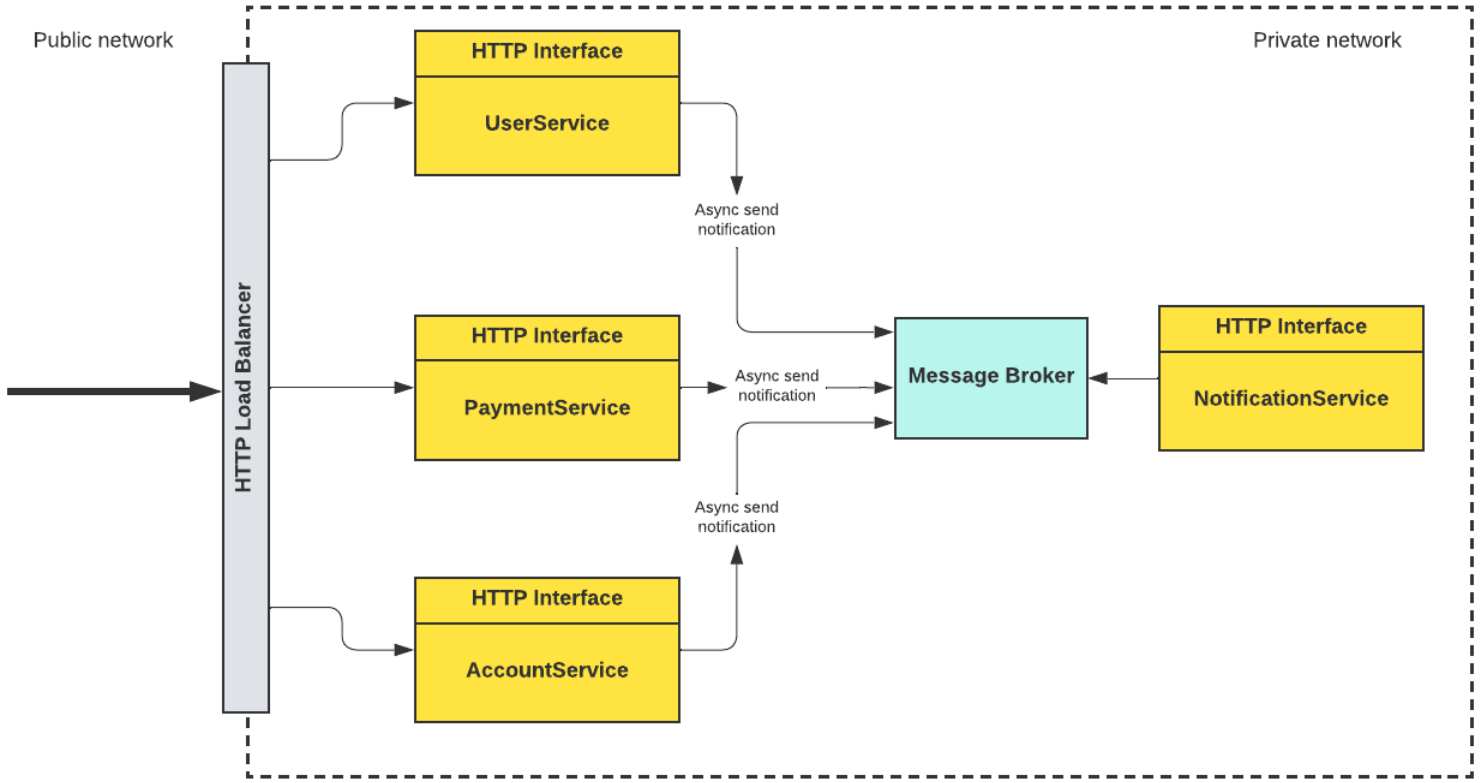
Jednoduchý príklad



Jednoduchý príklad



Jednoduchý príklad



```java
@Service
public class NotificationSender {

    private static final String IDENTIFIER = "identifier";
    private static final String NOTIFICATION_QUEUE = "example_notification_queue";
    private final AmqpTemplate rabbitmqTemplate;

    public NotificationSender(AmqpTemplate rabbitmqTemplate) { this.rabbitmqTemplate = rabbitmqTemplate; }

    public void sendNotification() {

        // Example notification text
        String notification = "Example notification";
        // Random UUID
        String personId = "4fa5317a-cb19-11ec-9d64-0242ac120002";

        rabbitmqTemplate.convertAndSend(
                NOTIFICATION_QUEUE,     // Set queue name
                notification,           //Message (String),
                message -> {
                    message.getMessageProperties().getHeaders().put(IDENTIFIER, personId);  // Recipient's identifier as header
                    return message;
                });
    }
}
```

```java
@Component
public class NotificationSenderListener {
    private static final Logger LOGGER = LoggerFactory.getLogger(NotificationSenderListener.class);
    private static final String IDENTIFIER = "identifier";
    private static final String NOTIFICATION_QUEUE = "example_notification_queue";

    @Autowired
    private INotificationService notificationService;

    // Set listener to the notification queue
    @RabbitListener(queues = NOTIFICATION_QUEUE)
    public void processMessage(Message message) {

        // Get notification text from Message body
        String notification = new String(message.getBody());
        // Get identifier of person from specific header
        UUID identifier = UUID.fromString(message.getMessageProperties().getHeader(IDENTIFIER));

        // Some validation and other logic here ...

        try {
            // Process the message
            notificationService.sendNotification(identifier, notification);
        } catch (Exception e) {
            LOGGER.error(e.getMessage());
        }
    }


}
```

Implementácie

RabbitMQ

Apache Kafka

Jakarta Messaging (JMS)

Apache ActiveMQ

...