Messaging - How to Apply?

Designing an Enterprise Application, using the following steps:

1. Analyzing the context of your system.
2. Defining all non-functional requirements that are important for the project.
3. Think of several examples of communication between (micro)services that could be implemented using messaging.
4. Think about how messaging can help to meet the defined non-functional requirements. Make use of the (65!) available messaging integration patterns Links to an external site.
5. Defining a clear selection criterion for the technology based on the functional and non-functional requirements.
6. Choosing a technology based on the project's criteria and motivate the choices.

Name: Victoria C. A. Fong

1 Analyzing the context of your system.

The system is a medicine tracker application. Due to the large amount of people that take medicine each day and some even multiple, is it hard to keep in track taken the medicine. The goal of the system is to make this task easier for people to keep track of taking their medicine. The benefits of the project for the users are that they would be able to track the medicine they take, get reminders to take their medicine, be able to leave personal feedback and for caregivers of the user to also help them keep track of taking their medicine if they wish to do so.

2 Defining all non-functional requirements that are important for the project.

2.1 Non-Functional Requirements

I have documented the Non-Functional requirements based on the article: "Masters, M. (2020, September 20). *What are Non Functional Requirements?* Requirements.com. Retrieved October 16, 2022, from https://requirements.com/Content/What-is/what-are-non-functional-requirements"

I have defined the non-functional requirements into what type the requirement can fall under. There are two types I have defined in this document which are Process and Data. This has been used to avoid any confusion to what the non-functional requirement is based on.

Non-functional requirement category	**Typically applies to Non-functional type**	**Example**
Accessibility requirements	Process	The project is a web application available for different users and not as a mobile where only certain devices are supported.
Availability requirement	Process	The application is available 24/7 due to each patient taking their medicine at different times. Could be morning, noon or night
Backup and Recovery requirement	Both	**Process**: when a server has sent a message to another server is the data not lost and will the message be stored and send to the server when its back up and running. **Data**: Data of a user is held for one month after deleting the user. This is because a user might go off their medicine and might not think they need their data but after a few weeks will start back. With the previous data can doctor keep track of their feedback for best meds for the patient.
Extensibility requirement	Both	**Process and Data**: To be able to append additional element and features to its existing structure
Legal and Regulatory requirement	Both	**Process**: Users accept that the application stores personal information of them for their own use, but do they have full control of the data to an extent.

		Data: All data of the user will be held for one month after account deletion.
Maintainability requirement	Both	The ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment.
Reliability Requirement	Both	**Process**: the audited tracking of the medicine of the user is accurate to what the patient or designated caregiver has been assigned to the patient **Data**: The medicine that can be chosen in the application must be updated to the latest version of open-source medicine database for reliability.
Security requirement	Both	**Process**: only patients can see their own personal data unless they allow a caregiver to do so by request. **Data**: patient users can give access to their caregivers to view or update/add information. Caregivers can view their patients' data but cannot edit the data until given permission from their patient.
Stress requirement	Process	Up to 10 caregivers can be assigned to a patient to avoid too much unneeded connections.
Supportability requirement	Process	Reminders are given to a patient to take their medicine. If the patient has indicated that they have taken their medicine, the caregiver will get a notification of this.
Testability	Both	**Process and Data**: the system, unit parts and its data communication are tested to be able to show/prove the quality of the application.

3 Examples of communication between (micro)services

Some examples that can be seen of communication between microservices in the system are:

- Retrieving needed information of other services
- Creating new information and informing other services that need to know this information such as if a new user is created would another service want to know their user id.
- Update info: when information is updated is this information shared across other microservices that are interested in this information.
- Delete info: when information is deleted is this information shared across other microservices that are interested in this information.
- Sending a request to a server which is down.
- Two services sending a request to the same service.

4 Messaging integration Patterns for meeting the non-functional requirements

Some Integration Patterns I came across that can be implemented to meet the set non-functional requirements are:

4.1.1 Canonical Data Model

"If a new application is added to the integration solution only transformation between the Canonical Data Model has to created. This makes added integration solutions independent from the current integrated features of the application."- *Enterprise Integration Patterns - Canonical Data Model*. (n.d.). Retrieved October 16, 2022, from https://www.enterpriseintegrationpatterns.com/patterns/messaging/CanonicalDataModel.html

4.1.2 Encapsulate the messaging code

"An application should not be aware that it is using Messaging to integrate with other applications. The application's code should be written without messaging in mind. At the sections where the application integrates with others, there should be a small layer of code that performs the application's part of the integration. When the integration is implemented with messaging, that small layer of code that attaches the application to the messaging system is a Messaging Gateway." - *Enterprise Integration Patterns - Canonical Data Model*. (n.d.). Retrieved October 16, 2022, from https://www.enterpriseintegrationpatterns.com/patterns/messaging/CanonicalDataModel.html

5 Defining a clear selection criterion for the technology based on the functional and non-functional requirements.

I have researched mainly 3 different types of message brokers to handle the messaging throughout the microservices which are: Kafka, RabbitMQ, Redis.

The main qualities that are sought for the project of a message broker are:

- Broker Scale: the number of messages sent per second
  - Project: Does not need a message broker that can handle a large number of messages per second.
- Consumer Efficiency: whether the broker is efficient in managing one-to-one and/or one to many consumers.
  - Project: the message broker is able to handle both one-to-one and one-to-many consumers.
- Data Persistency: the power to recover messages
  - Project: the system will be handling caching and if data is lost is a patient taken not taking their meds their on responsibility and not the system.

5.1.1 Sources:

Kafka vs RabbitMQ: What Are the Biggest Differences and Which Should You Learn? (2022, July 25). Simplilearn.com. Retrieved October 16, 2022, from https://www.simplilearn.com/kafka-vs-rabbitmq-article

Levy, E. (2022, August 29). *Kafka vs. RabbitMQ: Architecture, Performance & Use Cases*. Upsolver. Retrieved October 16, 2022, from https://www.upsolver.com/blog/kafka-versus-rabbitmq-architecture-performance-use-case

RabbitMQ vs. Apache Kafka®: Key Differences and Use Cases. (2022, October 7). Instaclustr. Retrieved October 16, 2022, from https://www.instaclustr.com/blog/rabbitmq-vs-kafka/

Message broker:

1.Broker Scale — the number of messages sent per second within the system.

Don't need a broker that sends a large number per second:

- new users are created each day but not for the entire system to send data

- new medicine is not created each day to be set into the medicine tracker service

- tracks a feedback and intake but not large amounts of data

2.Consumer efficiency — whether the broker is efficient in managing one-to-one and/or one to many consumers.

currently (one to one)

- one to one: if a user is added is the medicine tracker available, when new medicine is added in open source database for us to know about

- one to many (future):

- by adding a new functionality that can for example track activities and movements in its own service where each service must be aware of new users added.

3.Data Persistency — the power to recover messages

not really vital:

- the platform does track if a user has taken their medicine but with the functionality of the application sending reminders to caregivers for instance if data is lost can help as an addition for users taking their meds.,

6 CHOOSING A TECHNOLOGY BASED ON THE PROJECT'S CRITERIA AND MOTIVATE THE CHOICES.

The message broker I have chosen to handle messaging between microservices is RabbitMQ.

Rabbit mq:

https://www.simplilearn.com/kafka-vs-rabbitmq-article

Developers use RabbitMQ to process high-throughput and reliable background jobs, plus integration and intercommunication between and within applications. Programmers also use RabbitMQ to perform complex routing to consumers and integrate multiple applications and services with non-trivial routing logic.

RabbitMQ is perfect for web servers that need rapid request-response. It also shares loads between workers under high load (20K+ messages/second). RabbitMQ can also handle background jobs or long-running tasks like PDF conversion, file scanning, or image scaling.

Summing it up, use RabbitMQ with long-running tasks, reliably running background jobs, and communication/integration between and within applications.