Distributed Systems - Assignment 3

Remote Procedure Call

Sensor Monitoring System and Real-Time Notification for a Pill Dispenser

Conceptual architecture of the distributed system:

Client

Server

Aplicatie

Desktop

-RunMedication.java

Aplicatie

SPRING

Backend

Pkg gui

-GUI.java

Pkg config

-AppConfig.java

-MyWebInitializer.java

Pkg medication

-Medication.java

-MedicationBean.java

-MedicationPlan.java

-MedicationService.java

Pkg medication

-Medication.java

-MedicationPlan.java

-MedicationService.java

-MedicationServiceImpl.java

Postgres DB / med.txt

This sistem represents a pill dispenser that can be used by any pacient that gets from his doctor a medication plan in order to keep the evidence that he took or not the corresponding pills, which is important both for the pacient and for the doctor to be able to follow the necessary treatment.

We have a Java application with 2 modules representing the two parts of this distributed system:

-the first one is the spring java app that has the role of thid distributed system’s server, a backend app built using Spring framework that also makes possible the connection with the clients by the remote procedure calls. Here we can find the configuration of HttpInvokerServiceExporter that it’s used to allow the rpc’s between the server and the clients. The server also containts the implementation of the methods that can be called between the 2 apps, the getMedicationPlan(..) method, the notifyTakenMed(..) method and the notifyNOTTakenMed(..), methods that the client calls to get the medication plan for the current day, or to notify the server if some medication has been taken or not that day.

-the second one is the Client app, a simple desktop spring app that connects with the Server app(Backend) and communicates with it by calling methods from the MedicationService interface that is common to both apps. The implementation of those methods is found in the MedicationServiceImpl class on the server-side.

As said, I used the HttpInvoker option to build this distributed system which essentially enables Remote Procedure Calls (RPC) over HTTP. In order to accomplish this, an outbound representation of a method invocation is serialized using standard Java serialization and then passed within an HTTP POST request. After being invoked on the target system, the method's return value is then serialized and written to the HTTP response. There are two main requirements. First, you must be using Spring on both sides since the marshalling to and from HTTP requests and responses is handled by the client-side invoker and server-side exporter. Second, the Objects that you are passing must implement Serializable and be available on both the client and server. While traditional RPC provides physical decoupling, it does not offer nearly the same degree of logical decoupling as a messaging-based system. In other words, both participants in an RPC-based invocation must be aware of a specific interface and specific argument types.

I chosed the already mentioned framework in order to be able to deploy the server-side of the project using Heroku platform, process that run the entire pipeline of CI/CD.

Configure .gitlab-ci.yml

Setting CI/CD stages

GitLab setup repository

Client Desktop app

Postgres Server\*

NGINX server(frontend)

Tomcat server(backend)

browser

Develop&push on gitlab

Deploy stage

Package stage

Codestyle stage

Test stage

DB / med.txt

Aplicatie

Desktop

CLIENT

Client

RMI

Aplicatie

SPRING

Backend

Server

RMI

The component that has to be started first it’s, obviously the server -from heroku, and then the desktop client app it’s turned on - it will call the getMedicationPlan(..) method from the shared interface, the server will return the medication plan for the current day, and from the gui, the pacient can press the Taken button after selecting from the table of medications which it will take which will announce the server which med was taken, or if the pacient does not take some meds in the intake interval prescripted in the plan, the server will also be notified about that.