Silvera survey

Silvera is a tool for accelerating the development of microservice architectures, and it is developed as a part of PhD research.

With this survey, we aim to collect feedback from users who did not participate in developing the tool to achieve an objective assessment of Silvera's quality. The results of this survey will be published in a research paper, so please carefully follow the given instructions.

We understand that the survey is time-consuming, and we would like to thank you in advance for your participation!

Silvera represents a nice introduction to the world of microservice architectures for the interested students. By doing this survey, you are eligible to pass the course Advance Programming Techniques at the Faculty of Technical with the highest grade.

SUBMISSION DEADLINE: Will be announced during lectures.

Instructions

To complete the survey, you will need to do follow the steps given in the text that follows.

Step 1: Familiarization with microservices and Silvera

Get familar with microservices. The following links provide great introduction in the world of microservices:

- Fowler and Lewis: Microservices
- Chris Richardson: Introduction to Microservices
- Microservice design patterns
    - Deployment: Multiple service instances per host, Service instance per container
    - Communication: Remote Procedure Invocation, Messaging
    - External API: API gateway
    - Service discovery: Service registry
    - Reliability: Circuit breaker

Get familiar with Silvera. Please check the video tutorials, examples or documentation.

The best way to get ready for the task is to try to type along as you watch the tutorials. This way, you will learn the process of building the applications with Silvera, which will allow you to solve the task more efficiently.

Step 2: The task implementation

Implement the task described in the next section. The task must be implemented both in Silvera and manually (in programming language of your choice). Also, we need you to measure the time spent during both implementations and be as precise as possible.

In case of any questions, use this email: alienized91@gmail.com. You can also contact your TA Vladimir Inđić.

How to perform time measurement correctly: Start the stopwatch when you start working on the task. Pause the stopwatch only when you are taking a break or when you are waiting for our help. This means that the stopwatch runs not only when you are typing but also when you are reading the documentation, etc. **Note**: Initial familiarization with Silvera (watching tutorials, readings examples and documentation) is not part of time measurement. The stopwatch is stopped only after you finish the task. Please be as precise as possible, because this information is of great value to us.

Important Note: You should do time measuring separately for manual implementation of the task and implementation done by using the Silvera tool. You will be asked to fill in those two times in the questionnaire explained in the following section. *The grade for your assignment will be based solely on the quality of the implementations and the answers you provide during the project defense. The time spent for the implementation will have no effect whatsoever, so please try to be as precise as possible.*

Step 3: The questionnaire

Once you finish the task implementation, please fill in the questionnaire that can be found here.

At the end of the questionnaire you will be asked to upload the task implementations.

Step 4: Project Defense as part of the Advanced Programming Techniques Course

After you finish the task and filled out the questionnaire, you should contact your TA Vladimir Inđić. He will check if all steps are done correctly, and let you know whether you are eligible to give a presentation as part of the project defense.

Prerequisites

You will also need the following:

- Python 3.7.4, or newer
- Java 17
- MongoDB
- Apache Kafka

In case you encounter a bug...

Silvera is still in early development, so in case of bugs, please report here.

The task

Implement a distributed system based on microservices for publishing scientific papers. The system contains the following microservices: **User**, **SciPaper**, and **Library**.

User microservice allows users to register and log in. When registering, a user provides a *username* (ID), *password* (mandatory), *first name* (mandatory), *last name* (mandatory), and *email* (optional). In order to log in, the user must provide a username and password.

SciPaper microservice allows logged-in users to write new scientific papers. Each paper has an *author*, a *title*, and an arbitrary number of *sections*. Each section has its *name* and *content*. This microservice has a special method, *publish*, which, for a given paper ID, publishes a PUBLISH_PAPER message to a message broker. The message contains the paper's ID, paper's title, and author.

Library microservice has a method that listens to the PUBLISH_PAPER message and keeps data provided via message in the database. This microservice also has a public method that lists all data from the database.

The user is not aware of the application's architecture and uses the API gateway as an entry point. All services are registered in the service registry (omitted in the picture for brevity).

IMPORTANT

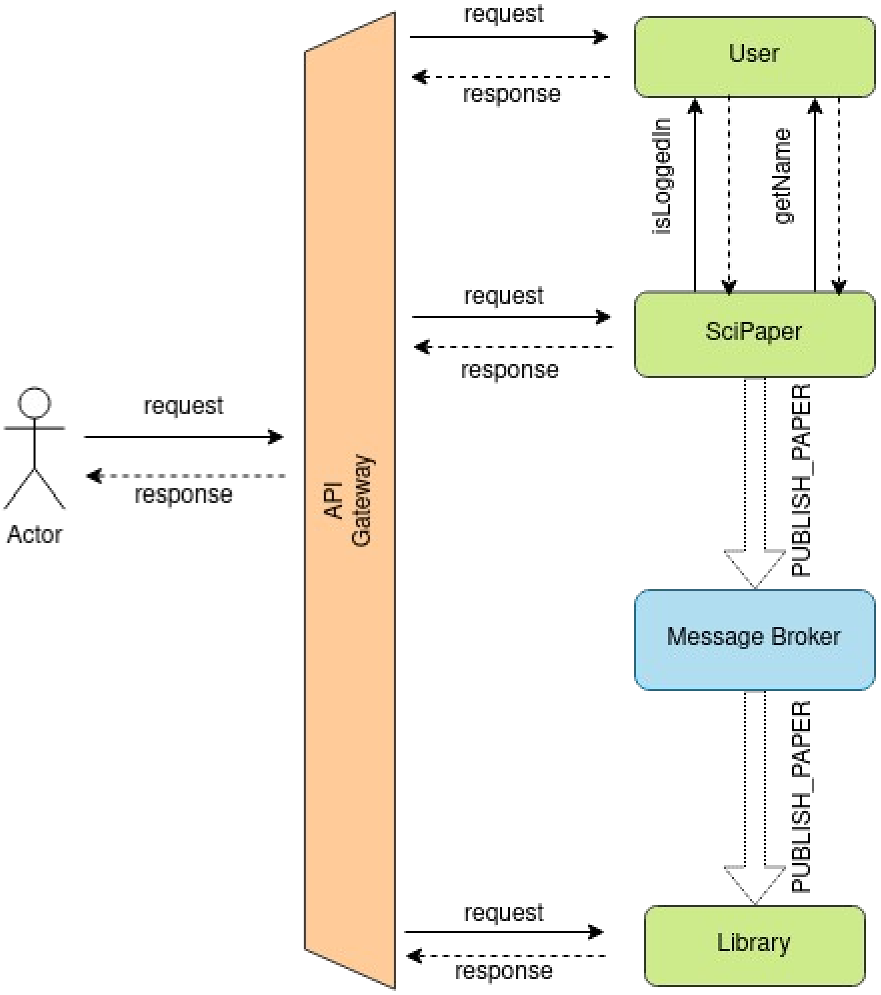
The task *MUST* be implemented both in Silvera and Java, and in the following order:

1. Implementation in *a programming language of your choice.*, then
2. Implementation in Silvera.

Also, we need you to measure the time spent during both implementations and be as precise as possible (for more info about time measuring, please refer to the *Step 2: The task implementation* section).

In case of any questions, use this email: alienized91@gmail.com. You can also contact Vladimir.

The architecture of the application


request
response
User
isLoggedIn
getName
request
response
SciPaper
PUBLISH_PAPER
request
response
Actor
API Gateway
Message Broker
PUBLISH_PAPER
request
response
Library