Dear Students,
as anticipated in class, this year you have **two options** for the final exam:

1) **carry out the same type of work done for the midterm test (tutorial/implementation/migration), but now in the domain of microservices.**

   Two sub-options:
   a. migrate (and possibly extend, based on the number of group members) the “Spring Boot Openjob system” presented by Doctor Filippone during the last lecture (this require to fork https://github.com/gianlucafilippone/springboot-microservices-development). This requires to extend the .pptx of the Filippone’s slides.
   b. students are free to propose other examples (In this case, the approval of the professor is required).

2) **engineering and implementing a (micro-)SOA application. In this case, follow the instructions below (please, read them carefully).**

IMPORTANT:
- **Students are required to inform the professor of the chosen option at their earliest convenience.**
- **I am assuming that the groups are the same as those from the midterm test. Please, send the professor an email if anything has changed.**
- **Moreover, the group must notify the professor (via email/Teams) when they intend to take the exam and must submit all the required material at least two days before the exam date.**

Option 2) instructions
For engineering and implementing a (micro-)SOA application, students must:
- follow the SOA engineering principles,
- apply the SOA best practices,
- adopt the technologies explained in the classroom, the way they have been taught (please avoid downloading and adapting examples from the Internet: instead, develop an original project on your own ;).

Although you are free to choose the application domain and the specific purpose of the application, please adhere to the following constraints.

1. The application must include REST and SOAP services and microservices (developed using Apache CXF, Jakarta, Spring).

2. The application must be composed of (at least) three service providers, (at least) two service prosumers, and (at least) one client application (e.g., java application, web application, mobile app). As for the microservices, the interaction all client-to-service interactions must pass through an API gateway.

3. The client(s) must interact with (at least) the prosumer(s), and the prosumer(s) must interact with (at least) two providers.

4. Develop at least one asynchronous service (polling and/or callback approaches). At least two prosumers must execute their job “in parallel” and then synchronize/coordinate each other before responding to the client(s), e.g., run in parallel to collect data, then synchronize or coordinate to process and aggregate the data before responding to the client(s)... Importantly, the need to use asynchrony must be clearly motivated/justified, not in general, but by considering your system’s logic/interactions/...

5. Concerning the client(s), the goal is to show at least three different interactions with the prosumer(s). In addition to the interactions with the prosumer(s), the client(s) can also interact with the providers directly, if needed.

6. From an engineering point of view, you must create an architectural diagram using a service/component diagram or the classical blocks (for services/components) and lines (for connections) diagram. The interaction scenarios must be clearly shown and thoroughly documented. For each interaction scenario, the corresponding sequence diagrams must also be produced.

7. The application, its goals, expected outputs, how it works, etc., must be introduced with a clear textual description (a .docx or .pptx format is acceptable).

8. The code must be documented with clear and verbose comments within the source code, WSDL files, etc. Use Open API and Swagger for all REST services.

9. A README.txt file must be provided to clearly and unambiguously explain all the steps to be followed to set up the application.

10. Use Spring Boot, and Docker for microservices.

11. The project(s) must use Maven for project management (Please do not use other project management tools)

12. [Optionally] Maven archetype(s) can be realized for the project(s) structure.

13. Microservices must support deployment in multiple instances, and and must include load-balanced interactions. Note that load balancers, API gateways, and discovery services are “additional services” with respect to the business services to be realized as described above. Importantly, the need for scaling must be motivated/justified, not in general, but by considering your system's logic/load/...

During the project defense, each member of the group must clearly demonstrate his/her contribution to the project realization.

HOMEWORK ASSESSMENT METHOD

- Completeness of contents (i.e., quantity in terms of having all the necessary or appropriate parts)
- Quality of contents (e.g., completeness and soundness)

- Quality of presentation (e.g., clarity)
- Contents organization (e.g., gradualness)
- Presentation format (e.g., slides arrangement -> no hasty preparation)
- Originality of the project with respect to existing examples from the Internet.
- Ability to build on and elaborate on what has been explained in the classroom.