Final exam
Part 1

[10 minutes]

Circle all statements that are true

☐

Suppose we need to apply load balancing for the API gateway. Client side load balancing is then the best option.

☐ B. Suppose we need to apply load balancing for the API gateway. Server side load balancing is then the best option.

☐ C. Suppose we have a microservice architecture and we need to apply load balancing for one of the microserices. Client side load balancing is then the best option.

☐ D. Suppose we have a microservice architecture and we need to apply load balancing for one of the microserices. Server side load balancing is then the best option.

☐ E. One reason why kafka uses event sourcing is that it allows producers to load balance the messages send to kafka

☐ F. A microservice architecture is always the best architectural style to use

☐ G. With JWT tokes, the user role is included in the token

☐ H. In a stream based architecture you always should use CQRS.

☐ I. We can implement the blackboard architectural style using kafka

☐ J. With the saga pattern we can implement strict consistency between microservices

[15 minutes]

A microservice architecture has many advantages, but also disadvantages. Suppose you are the architect of a large application build with many microservices

a. One problem of a microservice architecture is to make the whole system secure. What technique(s) would you use to solve this problem?

OAuth2 and JWT

b. One problem of a microservice architecture is to make the whole system resilient to failure.
What technique(s) would you use to solve this problem?

Hystrix, Registry with heartbeat, client side load balancing

c. One problem of a microservice architecture is that it is difficult to get a good overview of the status of the business processes that are executed in the different microservices.
What technique(s) would you use to solve this problem?

ELK stack (Zipkin alone is not enough to monitor the status of your buss. process.)

d. One problem of a microservice architecture is that it is difficult to make a collection of actions that execute in different microservices in a transactional way. So I want to update data in different microservices within a transaction.
What technique(s) would you use to solve this problem?

SAGA, compensating transaction, eventual consistency

[15 minutes]

Suppose you are responsible for designing and building Microservice A and Microservice B.
In System A you have to call system B in an **asynchronous** way.

a. We learned **2 completely different techniques/protocols** to implement an **asynchronous** call between Microservice A and Microservice B. Give the **name** of both **techniques/protocols**.

Messaging and webflux

b. Clearly explain the architectural relevant **advantages and disadvantages** of both techniques/protocols given in part a. Explain clearly when you would use which techniques/protocols.

Webflux:

Adv: non blocking threads and performance

Disadv: complex, hard to debug, whole stack needs to be reactive

When to use: When you cannot make use of a messaging middleware (like an angular webclient calling a microservice). When you need to use REST.

Messaging

Adv: loose coupling, you have a buffer

Disadvantage: you need messaging middleware

When to use:When you have messaging middleware, prefer messaging over webflux because it is much simpler and much more powerful.

When you want to broadcast data

[20 minutes]

We have an existing hotel booking system that offers generic hotel booking functionality like :

- Search hotels on keyword(s) like city, price, etc.
- View hotel details
- Book a hotel room
- Manage hotels (add new hotel, update hotel details, remove hotel, etc.)
- Etc.

a. You are the responsible architect for this system, and one of your colleagues advices to modify the existing system and apply the **CQRS** pattern for this system. Give **2 valid reasons** when it is a good idea to apply the CQRS pattern for this system. (**Be careful, only give valid reasons. If you give one or more reasons that are not valid you will lose some points**)

1. You want to scale queries different from commands
2. You need very fast read performance on data (from different services)
3. Your UI look different than your tables

b. In the existing hotel booking system we have many different domain classes. If we apply the CQRS pattern for this system, explain clearly what would change in the existing domain classes. In other words, how will the new domain classes differ from the old domain classes?

The domain classes in the command side have domain logic while the domain classes on the query side are simple DTO without domain logic and optimized for fast view performance

[15 minutes]

Suppose you have an existing application that contains about 10 components. You need to implement the logic of one business process that executes logic in 6 different components.
An example business process would be that a customer purchases a flight at an airline. In the customer component we have to add the customer. In the flight component we have to reserve the seat for the particular flight. In the meals component we have to add the meal for this customer. In the rewards component we have to add the rewards points for this flight.

You have 2 options for implementing the logic of this business process:
Option 1: You create a separate component that contains the overall business process itself, and this separate component will call the other 6 components to perform the necessary logic.

Option 2: You divide the overall business process in smaller parts, and implement these smaller parts in the existing 6 different components.

a. What are the advantages and disadvantages of option 1 and option 2?

Option 1: orchestration

Adv: good overview of the process

Disadvantage: does not work well when things get large and/or complex

Option 2: choreography

Adv: does work well when things get large and/or complex

Disadvantage: not a good overview of the process, we have to do extra stuff to get that overview (monitoring with ELK stack)

b. Your boss asks you for your advice. Explain clearly when you would choose option 1 and when you would choose option 2.

Option 1: when things are simple and/or the scope is small
Option 2: when things are complex and/or the scope is large

[10 minutes]

Suppose we have a system that contains many microservices. Now we need to write a client web application in Angular that shows data that comes from these different microservices.
Give **3 valid reasons** why it is not a good idea to have this client web application talk directly to the different microservices.
(Be careful, only give valid reasons. If you give one or more reasons that are not valid you will lose some points)

1. Tight coupling between client and microservice
2. Difficult to add crosscutting concerns (security, logging, etc)
3. Microservices need to support REST
4. Difficult to implement load balancing for every microservice, you need service side LB for all microservices.

[10 minutes]

In this course we learned that the registry uses a heartbeat to check if a service is still up and running. Suppose serviceA instance1 is down and serviceA instance2 is up and running, and serviceB wants to call serviceA. ServiceB will call the registry, and gets the address of serviceA instance2 because the registry knows that serviceA instance1 is down.

If we also add load balancing to these calls between serviceA and serviceB we basically have failover. If a service fails then the registry knows about it, and that service will not be used till it gets up and running again. If the registry and load balancing give us already failover, why do we need hystrix? In other words, what does hystrix provide that we don't get already with the registry and load balancing?

Hystrix adds support to deal with slow microservices.

[15 minutes]

a. Explain clearly why in a microservice architecture typically does not make use of an ESB.

Because an ESB implements orchestration and that does not works well in a complex system with a large scope

b. Explain clearly why every microservice uses its own database instance? Why don't they share the same database?

Because it creates tight coupling between the microservices.

[10 minutes]

Describe how we can relate a **stream based architecture** to one or more principles of SCI. Your answer should be about 2 to 3 paragraphs. The number of points you get for this questions depends how well you explain the relationship between a **stream based architecture** and one or more principles of SCI.