Final exam
Part 1

[10 minutes]

Suppose you are designing a microservice architecture and in your team you have a discussion if a particular microservice should be split up into 2 microservices or not.

Give 4 **different and valid** reasons when it is a good idea to split up one microservice into 2 separate microservices.

(Be careful, only give valid reasons. If you give one or more reasons that are not valid you will lose some points)

1. If the microservice gets to complex
2. If we need more than 1 agile team to build/maintain the microservice
3. If we have different scaling requirements for the 2 separate microservices
4. If we have different deployment requirements for the 2 separate microservices
5. If we need/want to use different technologies for the 2 separate microservices

[15 minutes]

a. JWT uses a digital signature. Explain clearly why JWT uses a signature instead of encrypting the whole token.

Because the token does not contain any sensitive information, so we don't need to encrypt the token. We only need a signature to be sure the token is valid.

b. If we use JWT, we still need a private key and a public key. Who is the owner of these 2 keys? Explain clearly which service uses the public key and which service uses the private key

The authorization server is the owner of both keys. The authorization server signs the token with its own private key, and the resource server(s) use the public key of the authorization server to check the signature.

[15 minutes]

Fill in the following table:

Challenges of a microservice architecture

Challenge
Complex communication
Performance
Resilience
Security
Transactions
Keep data in sync
keep interfaces in sync
Keep configuration in sync
Monitor health of microservices
Follow/monitor business processes

For the numbers 1 to 10, write down all techniques we studied that helps in managing these challenges.

Challenge	Solution
Complex communication	Feign Registry API gateway
Performance	Event Driven Architecture (EDA)
Resilience	Registry replicas Load balancing between multiple service instances Circuit breaker
Security	Token based security (OAuth2) Digitally signed (JWT) tokens
Transactions	Compensating transactions Eventual consistency
Keep data in sync	Publish-subscribe data change event
Keep interfaces in sync	Spring cloud contract
Keep configuration in sync	Config server
Monitor health of microservices	ELK + beats
Follow/monitor business processes	Zipkin ELK

[10 minutes]

a. When you create a topic in Kafka you need to enter 3 different properties. The first property is the name of the topic. What are the 2 other properties you need to enter?

Number of partitions
Replication factor
b. For both 2 other properties, explain clearly what they mean and why they are important properties for a Kafka topic.

Number of partitions: every topic has 1 or more partitions so that messages can be load balanced over the partitions. Main importance of partitions is load balancing

Replication factor: every partition can be replicated to multiple nodes so when one node fails we don't lose data. Main importance of replication is fault tolerance.

[15 minutes]

a. What are the 3 main characteristics of a circuit breaker in a microservice architecture?

Fail fast: Caller gets an immediate response

Fail gracefully: Caller can fall back to an alternative

Recover seamlessly: Circuit breaker will periodically check if to be called service is back online

b. Does the API gateway make use of a circuit breaker?

Yes. It needs to call other services.

c. Does Zipkin make use of a cuircuit breaker?

No. It does not need to call other services.

d. Consider the following scenario:

1. Microservice A wants to call Microservice B and asks the registry for the URL of Microservice B.

2. Suppose Microservice B is not available, and the registry tells Microservice A that Microservice B is not available

3. Microservice A knows that Microservice B is not available, and will not call Microservice B.

In this scenario we see that if one microservice want to call another microservice, the registry will tell us if that microservice is available or not. Explain clearly why microservices still need a circuit breaker, even if the registry tells us already if a microservice is available or not.

We still need a circuit breaker because the registry does not tell us if a service is slow

[10 minutes]

Give **3 different and valid** reasons when it is a good idea to apply the CQRS pattern. (Be careful, only give valid reasons. If you give one or more reasons that are not valid you will lose some points)

- When queries and commands have different scaling requirements
- When read performance is critical
- When your screens start to look very different then your tables
- When you apply event sourcing

[10 minutes]

a. Explain clearly what a micro front-end is.

We divide the one large front-end application into small UI components belonging to a microservice domain. We still write a front-end application, but it makes use of the small UI components

b. Explain clearly the main reasons to adapt the micro front-end pattern.

1. We don’t have one large (complex) front-end

2. The people who build the microservice also build the corresponding UI. These people are domain experts and know about the details of the UI

3. we can reuse these UI components in different UI applications

[10 minutes]

Explain clearly why you always should apply the CQRS pattern if you choose for event sourcing.

Because with event sourcing it is difficult/time consuming to compute the current state. With CQRS the query part maintains the current state so that we can get the current state very fast.

[15 minutes]

Select all statements that are correct

☐ A. When Microservice A wants to call Microservice B, and we have 3 instances of Microservice B then Microservice A will load balance the calls between these 3 instances of Microservice B using the round robin algorithm.

☐ B. When Microservice A wants to call the Registry, and we have 3 instances of the Registry then Microservice A will load balance the calls between these 3 instances of the Registry using the round robin algorithm.

☐ C. For calls between Microservice A and Microservice B we use client side load balancing.

☐ D. For calls between an external Angular web application and the API gateway we use client side load balancing.

☐ E. Conways law means that from the 3 qualities availability, consistence and partition tolerance we can have on 2 of these qualities for 100 %

☐ F. The 4 grant types for OAuth2 are: - Authentication code - Password - Client credentials - Refresh token **This question is not graded**

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

☐ H. If we want to implement the point-to-point pattern in kafka, we have to give all consumers the same group-id

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

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

In sakai you see that G is also correct, but that is not true.