Repetition Questions (dt. Wiederholungsfragen) Lesson 11

Topics and Concepts (Link to Lecture, via Fact Sheets)

The lesson of the lecture today covered the following concepts:

1. Service Contract, WSDL, Swagger
2. Microservices tenets and pattern example (API Gateway)

In the corresponding exercise[1], we worked with these concepts.

Questions

Topic/Concept: Service Contract

1. What is the business motivation for the Service Contract pattern? Name at least one of its design goals.
2. How does service contract design contribute to tight or loose coupling between service consumer and service provider?
3. List at least four topics to be covered in a platform-independent service model (including, but not limited to technology concerns).
4. Name at least three concepts in Open API Specification/Swagger.
5. Which REST maturity level(s) does Swagger correspond to?
6. List at least two differences between WSDL and Swagger.

Topic/Concept: Microservices tenets and pattern example (API Gateway)

1. What are the defining/unique characteristics of microservices?
2. How do the microservices relate to SOA: a) competition? b) refinement and implementation approach? c) same thing?
3. Name at least one microservices pattern and explain how it relates to a DDD, PoEAA or SOA one.
4. Which criteria to find a suited service size and granularity have been mentioned in the lecture?
5. What makes microservices hard to design, implement, test, maintain?
6. What are the defining characteristics of an API Gateway according to C. Richardson?

Answers

Topic/Concept: Service contract

1. *Business goal: be able to assemble applications from independendtly developed parts and release them often; design goals: interoperability of decomposed and possibly distributed services*
2. *The more shared understanding between service provider and service consumer, the tighter their coupling. So it is good to make the shared knowledge (data formats, error codes, etc.) explicit in the contract, but the amount of such shared knowledge should be as small as possible (but also as large as needed).*
3. *Business meaning of call, pre- and postconditions, price, Service Level Agreement (SLA), owner (see lecture slide 15 with service model template for more contract elements).*

[1] ../3-exercises-solutions/ZIO-AppArch-ExerciseWeek11.pdf

4. *Path, schema, operation, parameters (of different types: query", "header", "path" or "cookie"[2]), error codes*
5. *Level 1 and 2 (with some semantic misalignments or unfortunate naming such as `operationId`); basic but insufficient support for level 3 (see sample solution to excercise 11).*
6. *Focus on resource/path (with fixed set of verbs, from HTTP) vs. focus on ports types with flexible number of named operations); usage of XML Schema vs. JSON Schema. See this paper for a detailed (but partialy dated) comparison between RESTful HTTP and SOAP/WSDL Web services from 2008 (pre-Swagger times): http://design.inf.usi.ch/publications/2008/www*

Topic/Concept: Microservices tenets (incl. DevOps) and patterns (API Gateway)

1. *Seven tenets (see lecture slide), core properties: independently deployable and independently scalable (the notion of "micro" and "small" is unfortunate, services can have different sizes depending on consistency needs and other non-functional requirements); particular emphasis on decentral decision making (design and service autonomy) in microservices publications and projects.*
2. *Option b)*
3. *API Gateway is the microservices pattern looked at in the exercise; Remote Facade, Anti Corruption Layer (ACL), ESB are related patterns*
4. *Lines of code, team size, data model and dependencies (and resulting consistency needs)*
5. *Their number and their distributed nature, see "Microservices Tenets"[3] paper and articles by M. Fowler (and many more, of varying quality)*
6. *See sample solution to exercise 11: protocol adaptation, model transformation, routing (incl. fanning out, which is a form of routing); also security policy enforcement point and mediation*

[2] https://swagger.io/specification/#parameterObject
[3] http://rdcu.be/mJPz