P.PORTO

Methods and Techniques for Software Development

Ricardo Santos | 2019/2020
rjs@estg.ipp.pt

P.PORTO

Design Patterns for Microservices

Methods and Techniques for Software Development

Aula 7

2019/2020

P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Application architecture patterns

Retirado de:
https://microservices.io/patterns/

- Which architecture should you choose for an application?
  - **Monolithic architecture** - architect an application as a single deployable unit
  - **Microservice architecture** - architect an application as a collection of loosely coupled, services

P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Decomposition

- How to decompose an application into services?
  - **Decompose by business capability** - define services corresponding to business capabilities
  - **Decompose by subdomain** - define services corresponding to DDD subdomains
  - **Self-contained Service** - design services to handle synchronous requests without waiting for other services to respond
  - **Service per team**

Data management

- How to maintain data consistency and implement queries?
  - **Database per Service** - each service has its own private database
  - **Shared database** - services share a database
  - **Saga** - use sagas, which a sequences of local transactions, to maintain data consistency across services

- **API Composition** - implement queries by invoking the services that own the data and performing an in-memory join
- **CQRS** - implement queries by maintaining one or more materialized views that can be efficiently queried
- **Domain event** - publish an event whenever data changes
- **Event sourcing** - persist aggregates as a sequence of events

P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Transactional messaging

- How to publish messages as part of a database transaction?
  - Transactional outbox
  - Transaction log tailing
  - Polling publisher

P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Testing

- How to make testing easier?

  - **Consumer-driven contract test** - a test suite for a service that is written by the developers of another service that consumes it

  - **Consumer-side contract test** - a test suite for a service client (e.g. another service) that verifies that it can communicate with the service

  - **Service component test** - a test suite that tests a service in isolation using test doubles for any services that it invokes

Deployment patterns

- How to deploy an application's services?
  - **Multiple service instances per host** - deploy multiple service instances on a single host
  - **Service instance per host** - deploy each service instance in its own host
  - **Service instance per VM** - deploy each service instance in its VM

- **Service instance per Container** - deploy each service instance in its container
- **Serverless deployment** - deploy a service using serverless deployment platform
- **Service deployment platform** - deploy services using a highly automated deployment platform that provides a service abstraction

P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Cross cutting concerns

- How to handle cross cutting concerns?
  - **Microservice chassis** - a framework that handles cross-cutting concerns and simplifies the development of services
  - **Externalized configuration** - externalize all configuration such as database location and credentials

P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Communication patterns

- *Style*
- *External API*
- *Service discovery*
- *Reliability*

P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Style

- Which communication mechanisms do services use to communicate with each other and their external clients?

  - **Remote Procedure Invocation** - use an RPI-based protocol for inter-service communication
  - **Messaging** - use asynchronous messaging for inter-service communication
  - **Domain-specific protocol** - use a domain-specific protocol

External API

- How do external clients communicate with the services?

  - **API gateway** - a service that provides each client with unified interface to services

  - **Backend for front-end** - a separate API gateway for each kind of client

P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Service discovery

- How does the client of an RPI-based service discover the network location of a service instance?

  - **Client-side discovery** - client queries a service registry to discover the locations of service instances

  - **Server-side discovery** - router queries a service registry to discover the locations of service instances

- **Service registry** - a database of service instance locations
- **Self registration** - service instance registers itself with the service registry
- **3rd party registration** - a 3rd party registers a service instance with the service registry

P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Reliability

- How to prevent a network or service failure from cascading to other services?
  - **Circuit Breaker** - invoke a remote service via a proxy that fails immediately when the failure rate of the remote call exceeds a threshold

Security

- How to communicate the identity of the requestor to the services that handle the request?
  - **Access Token** - a token that securely stores information about user that is exchanged between services

P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Observability

- How to understand the behavior of an application and troubleshoot problems?

  - **Log aggregation** - aggregate application logs
  - **Application metrics** - instrument a service's code to gather statistics about operations
  - **Audit logging** - record user activity in a database
  - **Distributed tracing** - instrument services with code that assigns each external request an unique identifier that is passed between services

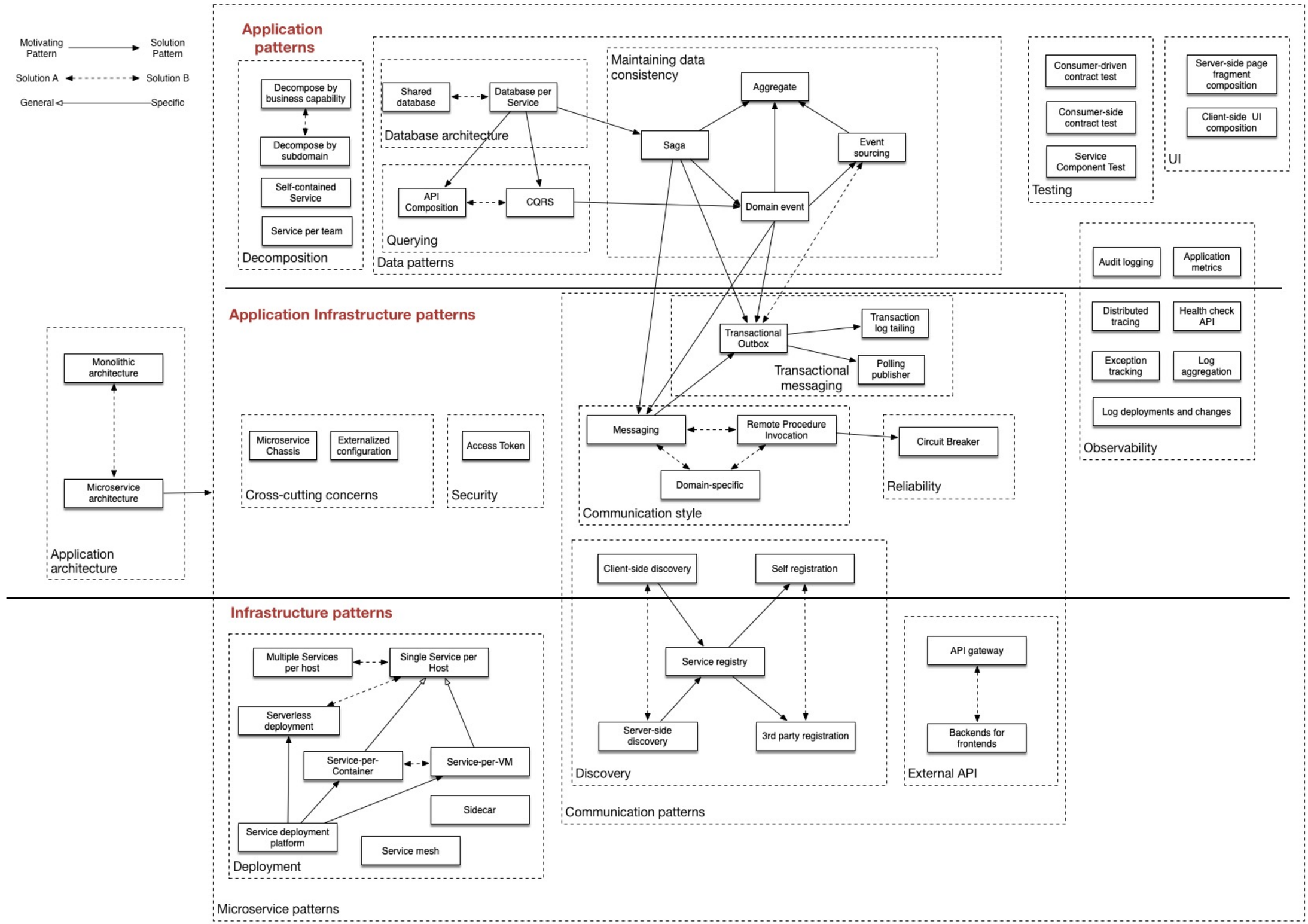
- **Exception tracking** - report all exceptions to a centralized exception tracking service that aggregates and tracks exceptions and notifies developers.
- **Health check API** - service API (e.g. HTTP endpoint) that returns the health of the service and can be pinged, for example, by a monitoring service
- **Log deployments and changes**

P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

UI patterns

- How to implement a UI screen or page that displays data from multiple services?
  - **Server-side page fragment composition** - build a webpage on the server by composing HTML fragments generated by multiple, business capability/ subdomain-specific web applications
  - **Client-side UI composition** - Build a UI on the client by composing UI fragments rendered by multiple, business capability/subdomain-specific UI components



P.PORTO

Polytechnic of
Porto

ESTG - School of
Management and Technology