Experimenting microFreshener
for detecting architectural smells and corresponding refactorings

Microservice-based applications

The applications that will be considered are:

Sock Shop

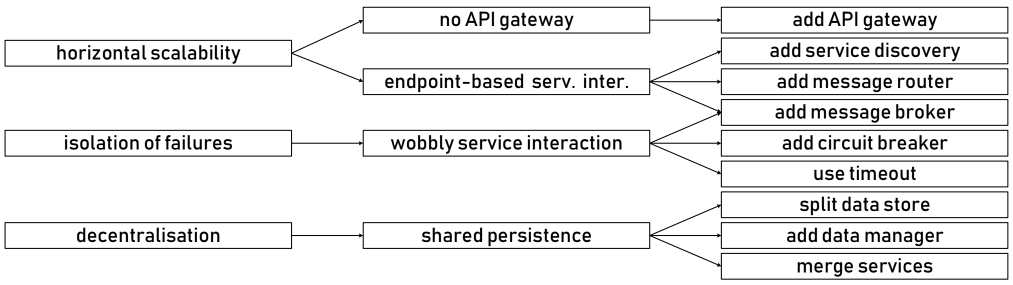
- microTOSCA spec: sockshop.yml
- sources: https://github.com/microservices-demo/microservices-demo

FTGO

- microTOSCA spec: FTGO.yml
- sources: https://github.com/microservices-patterns/ftgo-application

Design patterns, architectural smells and architectural refactorings

Consider the following taxonomy of design principles, architectural smells and refactorings:



which is an excerpt of that identified in the paper

D. Neri et al. "Design principles, architectural smells and refactorings for microservices: A multivocal review", SICS Software-Intensive Cyber-Physical Systems, 2019"

Please also find below the conditions determining the occurrence of each architectural smell (as per their discussion in the above mentioned paper):

- The **no API gateway smell** occurs whenever the external clients of a microservice-based application directly invoke functionalities offered by some internal service(s).
- The **endpoint-based service interaction smell** occurs in an application when one or more of its microservices invoke a specific instance of another microservice (e.g., because its location is hardcoded in the source code of the microservices invoking it, or because no load balancer is used).
- The **wobbly service interaction smell** occurs when a microservice M1 is interacting with another microservice M2 and a failure in M2 can result in triggering a failure also in M1. This typically happens when M1 is not provided with any solution for handling the possibility of M2 to fail and be unresponsive (e.g., timeouts or circuit breakers).
- The **shared persistence smell** occurs whenever two (or more) microservices are accessing and managing the same database, possibly violating the decentralisation design principles.

Activity

You are asked to perform two steps:

(i) discover all the architectural smells affecting the two given applications, and

(ii) list a sequence of architectural refactorings to be applied to each application so as to resolve all its architectural smells.

The experiment will consist of two rounds:

(1) in the first round you will have access only to the application spec and its sources, and

(2) in the second round you will have the possibility of exploiting the microFreshener analyser.

You will be divided in two groups:

	Group A	**Group B**
*round 1: working **without** microFreshener on*	SockShop	FTGO
*round 2: working **with** microFreshener on*	FTGO	SockShop

Group [A] [B] – Round [1] [2] – Application [SockShop] [FTGO]

Step (i) – Discovering architectural smells

List all the architectural smells affecting the given application, by filling the table below:

No API Gateway	***Endpoint-based serv. inter.***	***Wobbly serv. inter.***	***Shared persistence***	***Involved service relation(s)***

Example

X				*(extUser,S1) (extUser,S2)*
	X			*(S2,S4) (S3,S4) (S5,S4)*
		X		*(S4,MR1) (S4,S7)*
			X	*(S1,DB1) (S5,DB1) (S7,DB1)*

Note: You can use shortened versions of node names, provided that nodes can still be uniquely identified.

Group [A] [B] – Round [1] [2] – Application [SockShop] [FTGO]

Step (ii) – Resolving architectural smells via refactoring
List a sequence of architectural refactorings to be applied to the given application, so as to resolve all architectural smells affecting the application (and making it ultimately “smell-free”).

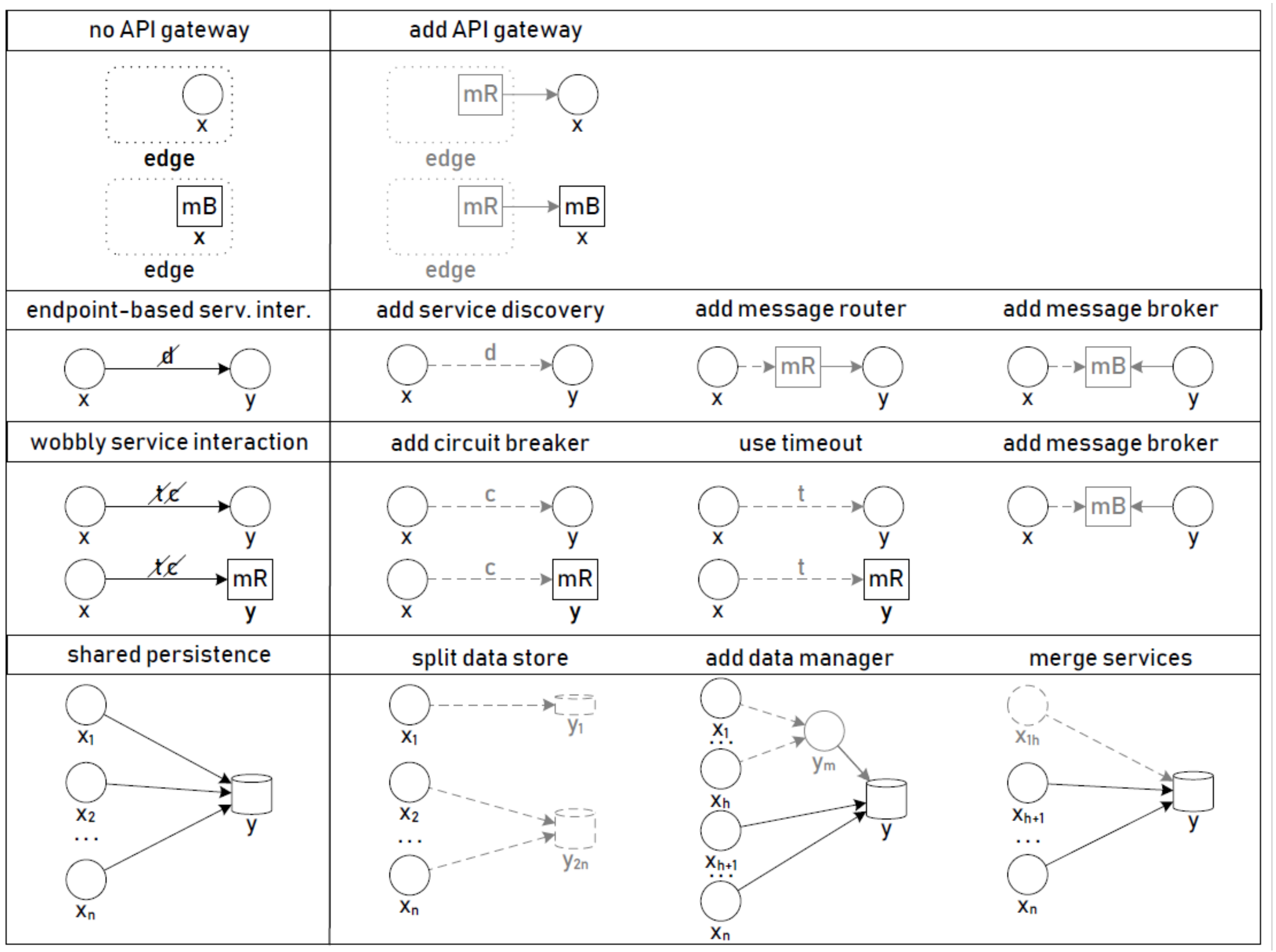
Refactored service relation(s)	Refactoring name	Introduced/reused node(s)

Example:

(S2,S1), (S3,S1)	***Add message router***	***MR1***
(S4,S1)	***Add message router***	***MR1***
(S1,S2)	***Add timeout***	***-***

Note: You can use shortened versions of node/refactoring names, provided that nodes/refactorings can still be uniquely identified.

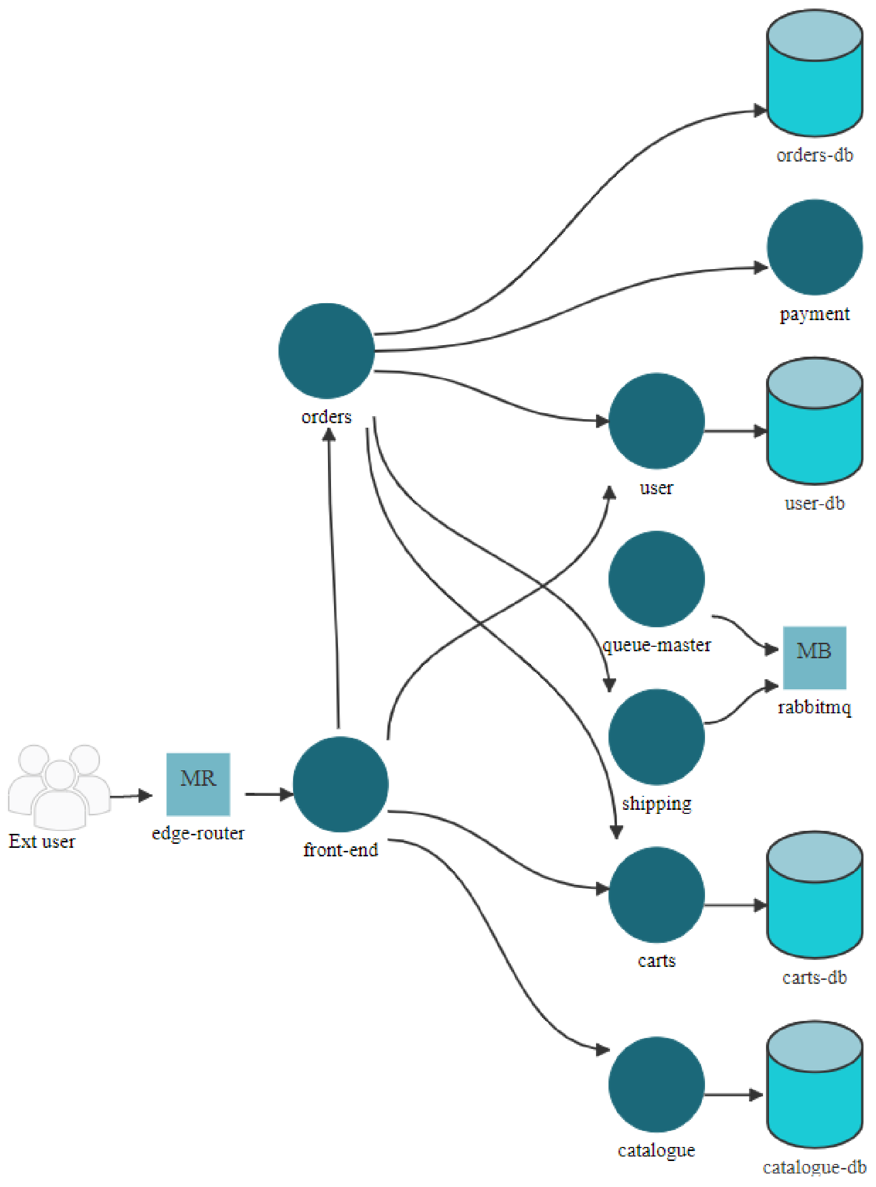
Smells and refactorings



Experimenting microFreshener
for detecting architectural smells and corresponding refactorings
(Application Specifications)

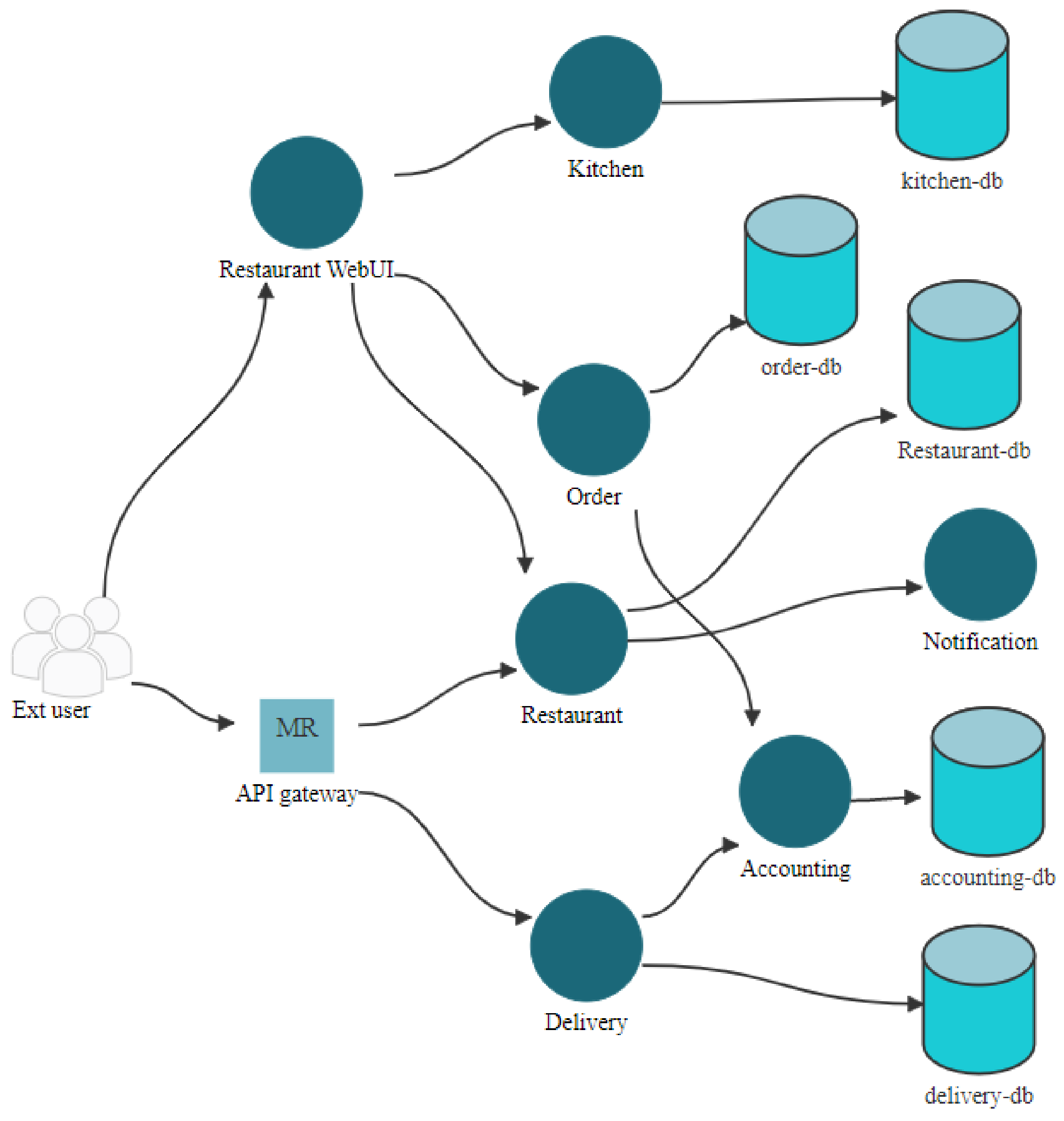
GROUP A - Sockshop

- microTOSCA spec: https://github.com/di-unipi-socc/microFreshener/blob/master/server/media/examples/sockshop.yml
- sources: https://github.com/microservices-demo/microservices-demo



GROUP B - FTGO

- microTOSCA spec: github.com/di-unipi-socc/microFreshener/blob/master/server/media/examples/FTGO.yml
- sources: https://github.com/microservices-patterns/ftgo-application



Kitchen
kitchen-db
Restaurant WebUI
order-db
Restaurant-db
Order
Notification
Ext user
MR
API gateway
Restaurant
Accounting
accounting-db
Delivery
delivery-db