Architecture de service

Résumé de cours/TD et projet 5-ISS SOA

Claire Gaigeot

Sommaire

I. Architecture N-tiers

Pour expliquer le principe d'architecture N-tiers je vais présenter les différents types d'architectures. Nous avons:
-L'application logicielle (locale): l'application est exécutée sur une machine.

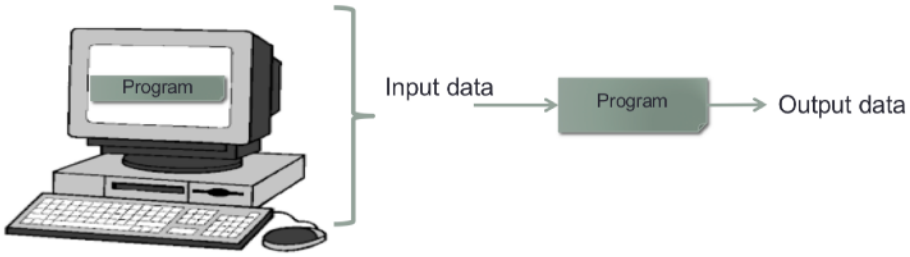

Fig 1 - Application logicielle (Support de cours Moodle: N-tiers architecture)

-L'architecture 2-tiers: le programme (application logique) et la présentation sont séparés sur deux machines différentes.

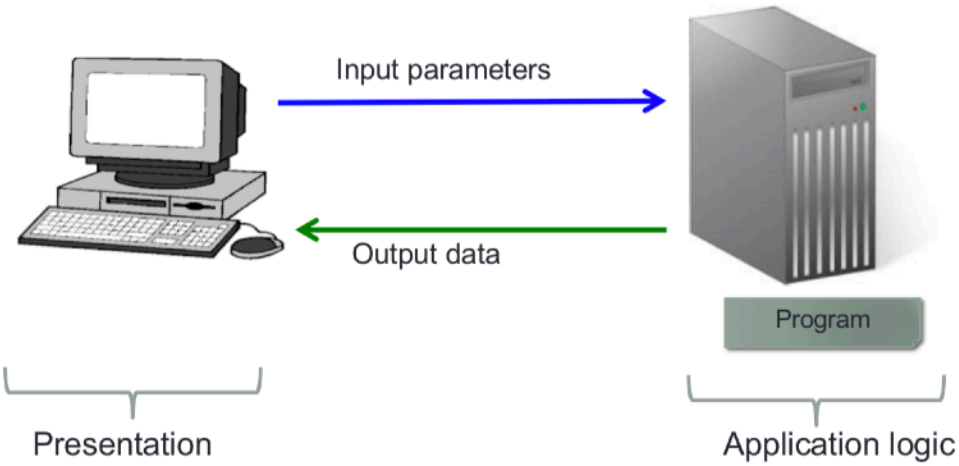

Fig 2 - Architecture 2-tiers (Support de cours Moodle: N-tiers architecture)

-L'architecture 3-tiers: le programme (processing), la base de données (data) et la présentation sont séparés sur trois machines différentes.

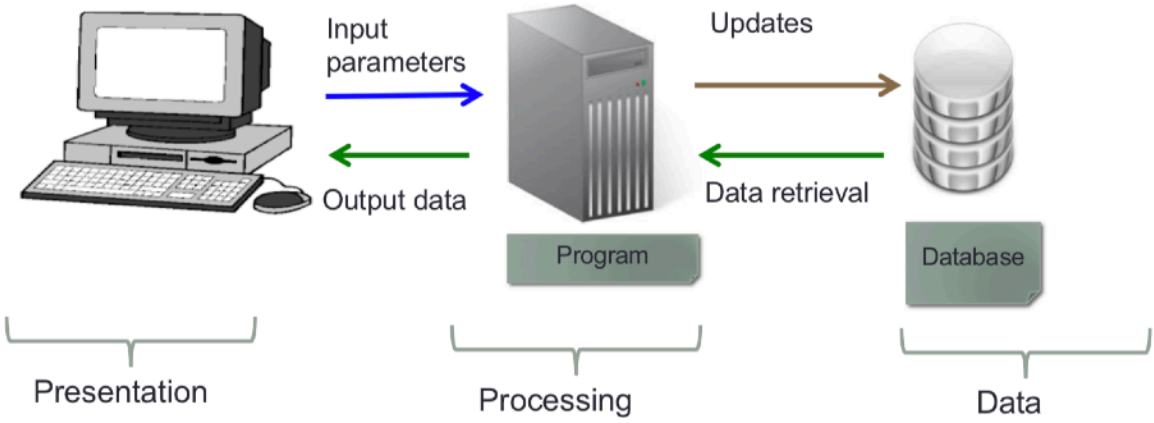

Fig 3 - Architecture 3-tiers (Support de cours Moodle: N-tiers architecture)

-L'architecture N-tiers: repose sur les 3 couches de l'architecture 3-tiers mais en plus, elle divise et distribue l'application logique en différents services. Par ailleurs, elle est basée sur des « business components », qui représentent les objets du domaine. Par exemple, un business component pourra être « customer », « invoice », « purchase order »…

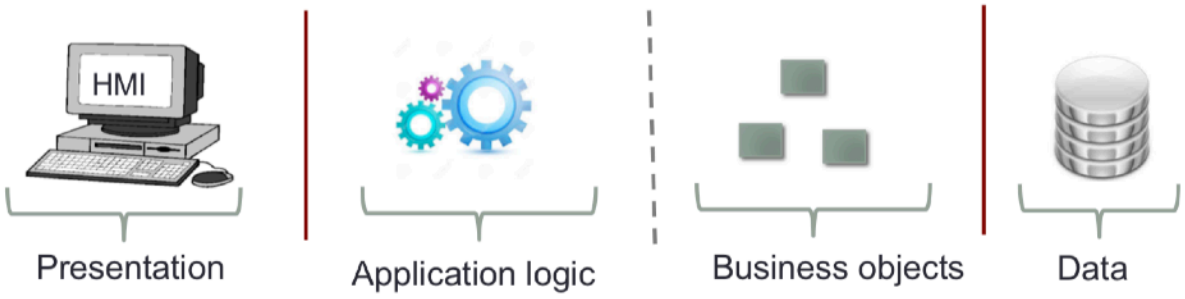

Fig 4 - Architecture N-tiers (Support de cours Moodle: N-tiers architecture)

II. Architectures orientées services – SOA

Face aux problèmes d'intégration d'applications, l'architecture orientée services vient apporter des solutions. Afin d'exposer les différentes fonctionnalités requises sous forme d'un ou plusieurs services, la SOA va encapsuler les applications sous forme de briques logicielles que l'on appelle services.

1. Service Web

Un service Web est un protocole d'interface informatique permettant la communication et l'échange de données et accessible via le Web. Actuellement, le protocole de transport est essentiellement HTTP. Pour utiliser un service Web, seule la description de son interface est nécessaire. Les détails liés à son implémentation ne sont ni exposés ni requis.

2. WSDL

Le WSDL (Web service description language) est un langage de description basé sur XML. Ce langage a pour but de faire une description détaillée de l'interface d'un service Web indépendamment de son implémentation. Cette description fondée sur XML fournit des informations nécessaires à l'appel du service, comme les méthodes que le client peut invoquer, le format de messages requis pour communiquer avec ce service, les protocoles

de communication utilisés et la localisation du service. La description WSDL des services Web peut être publiée dans des annuaires respectant le standard UDDI (Universal Description Discovery and Integration).

3. SOAP

SOAP (Simple Object Access Protocol) est un protocol d'échange d'information structurée dans l'implémentation de services web basé sur XML . En effet, les services Web communiquent via l'échange de messages qui respectent un certain format. Et le transfert de messages SOAP se fait le plus souvent à l'aide du protocole HTTP, mais on peut utiliser d'autres protocoles comme SMTP. En raison du nombre d'informations qu'impose le format XML et la nécessité d'ajout d'une surcouche supplémentaire pour la communication, SOAP peut alourdir significativement les échanges par rapport à d'autres solutions. En effet, la verbosité de XML peut causer des lourdeurs lors de l'analyse syntaxique.

III.Orchestration de services - BPEL

Un processus business (business process) est basé sur différents éléments d'une entreprise: des activités, des resources, des évènements, des résultats… Il existe aujourd'hui plusieurs languages de spécification de processus business: UML, XPDL, BPMN… Néanmoins, ces technologies ne donne pas de détails sur le comportement et le rôle des services lors d'une collaboration complexe. Afin de détailler et construire un processus business il existe: la chorégraphie de services et l'orchestration de services.

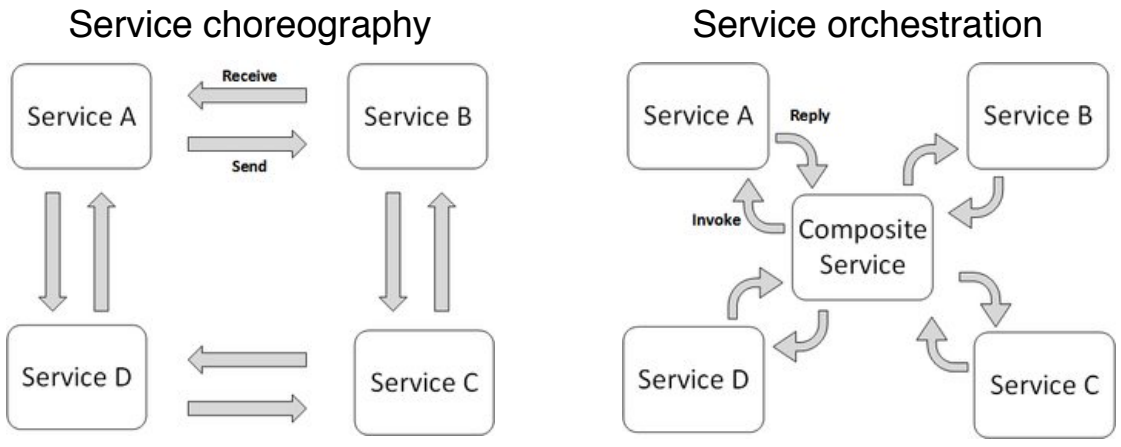

Fig 5 - Service choreography vs. service orchestration (source stackoverflow)

Contrairement à la choreography qui a une vue globale des interactions, l'orchestration aura une vue plus locale. Par ailleurs, dans une orchestration de services, un processus centralisé coordonnera l'exécution des services et on utilisera le language

BPEL. En effet, BPEL est un language de programmation destiné à l'exécution de procédures d'entreprises. Il est issu des languages WSFL et XLANG, et est dérivé du XML. Le fichier BPEL définit le processus. Quant à l'enchaînement et la logique des actions, ils seront exécutés par le moteur d'orchestration.

IV.Architectures orientées ressources – ROA

En informatique une ressource est tout ce qui a une identité: information, objet, application. Par exemple, une page, un sous-ensemble de page ou une liste de documents sont des ressources. Les architectures orientées ressources nécessitent d'extraire des instances d'une ressource particulière comme par exemple l'extraction d'une page HTML par une requête HTTP GET. Une ressource est accessible via un URI (Uniform Resource Identifier) qui identifie une ressource par localisation, par nom ou les deux. Un URI est en faite une généralisation des URNs (Uniforme Resource Names) et des URLs (Uniforme Resource Locators).

1. Architecture REST

Ces dernières années, une nouvelles architecture est apparue comme une alternative à l'architecture SOA: Rest (REpresentational State Transfer). Rest n'est pas un protocol, ni un standard, ni un format comme SOAP, c'est un style architectural pour le développement d'applications. Rest utilise des spécifications HTTP et ses opérations n'ont pas de statut. Basé sur la notion de ressources, REST présente un ensemble de conventions et de bonnes pratiques pour le développement de services Web. Un service REST-compliant s'appelle un service RESTfull.

2. Services web REST

Les services Web Rest sont principalement basés sur le protocole HTTP qui fournit les opérations nécessaires pour manipuler des ressources. Peu importe le langage dans lequel le service est développé et le type de plateforme d'exécution, un service Web Rest offre obligatoirement des interfaces basées sur les verbes HTTP qui sont GET, POST, PUT, ou DELETE.

3. Format des données échangées

L'invocation des services Web Rest permet d'envoyer et de recevoir des données via des requêtes HTTP. Ces données peuvent être sous différents formats tels que XML, JSON, ou XHTML.

V. Architecture micro-services

Une application monolithique est une application basée sur un code de base, simple à réaliser, efficace pour les petites applications et la base de donnée est facile à prendre en main. Mais cela a des limites, en effet, le déploiement est long, la structure est peu scalable et est très liée à la base de donnée.

La SOA ou REST ont l'avantage de la modularité, de la liberté de technologie… mais peuvent parfois être complex à implementer, notamment pour la SOA, et on peut parfois leur reprocher une granularité de services.

Une application basée sur des micro-services va implémenter un lot de fonctionnalités à l'aide de petit services indépendants. Les principes des micro-services sont les suivants: diviser une application en un lot de petits services, chaque micro-service a une responsabilité, chaque micro-service a sa propre base de données (on va parler de base de données décentralisée), les micro-services sont indépendants les uns des autres, un micro-service peut avoir son propre environnement d'execution (Machine virtuel, container), l'organisation est basée sur la couche business. Au final une application basée sur des micro-services sera plus facile à maintenir, sera flexible, autonome et scalable.

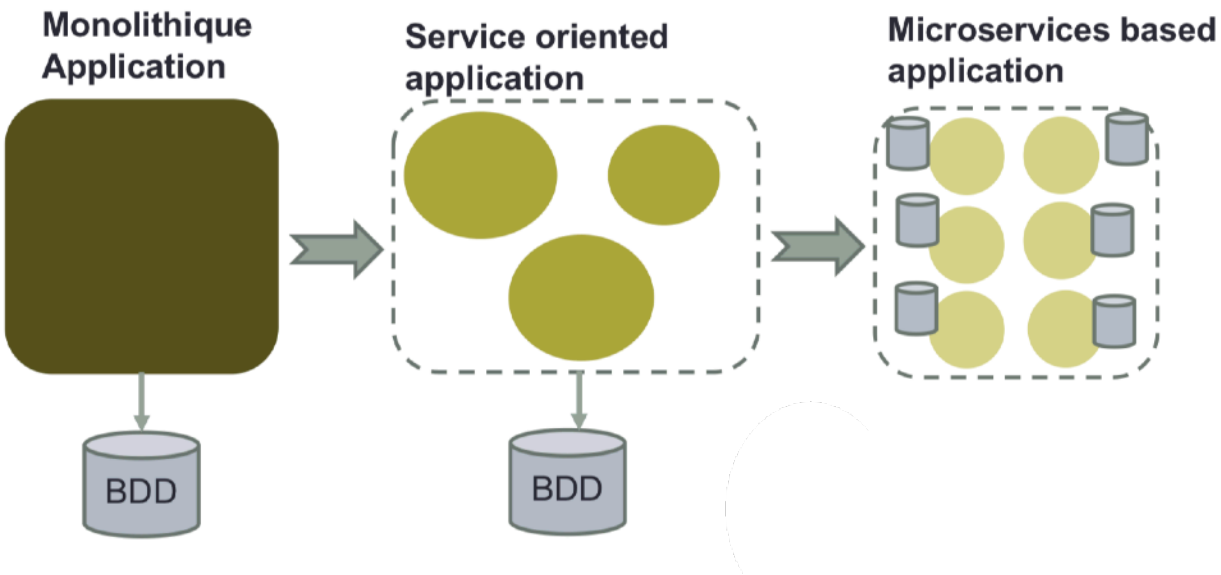

Fig 6 - Les types d'architectures applicatives (Support de cours Moodle: Microservice architecture)

VI. Travaux pratiques

1. Résumé de TDs

Lors des TDs j'ai pu mettre en pratique les différentes notions vues en cours.

- [TD.1.2.3] Création de services web SOAP, à partir d'une classe java, en suivant l'approche bottom-up et top-down. Et réalisation d'un client pour le service web SOAP.

Bottom-up: *création d'un service web SOAP et génération du document WSDL à partir de ce service web.*

Top-down: *création et configuration d'un document WSDL puis création d'un service web SOAP en fonction de ce document WSDL*

- [TD.4.5] Création, déploiement et test d'une procédure BPEL orchestrant trois services web.
- [TD.6.7.8.9] Création d'un service web Rest et d'un client de ce même service. Et ensuite, création d'un service web Rest et d'un client de ce même service mais consommant et produisant des données au format JSON ou XML.
- [TD.10.11] Utilisation du framework Spring Boot pour créer trois micro-services communicants et collaborants ensembles.

2. Projet d'architecture micro-services

Avec Arnaud Guiller, nous avons réalisé en binôme une architecture micro-services et avons implémenté deux micro-services individuellement. Nous avons utilisé l'outil de gestion Maven et le framework Spring Boot pour créer nos différents micro-services et nous avons testé notre implémentation avec Postman. Nous avons réalisé ensemble le micro-service « Runningcompetion ». J'ai réalisé « Namecompetitor » et « Winnercompetion ». Et Arnaud a réalisé « Timecompetitor » et « Averagetime ».

Notre proposition d'architecture représente la gestion d'une compétition de course à pied. Il y a un total de 5 micro-services. On peut imaginer que lors d'une course, les informations des compétiteurs sont stockées à un endroit et qu'à la ligne d'arrivée le temps des compétiteurs associé à leur dossard est stocké à un autre endroit. Le micro-service « Runningcompetition » permet donc de rassembler les informations d'un compétiteur (nom, prénom) et son temps de course. Quant aux micro-services « Winnercompetion » et « Averagetime » ils permettent de connaître respectivement le numéro de dossard du vainqueur et le temps moyen de la course.

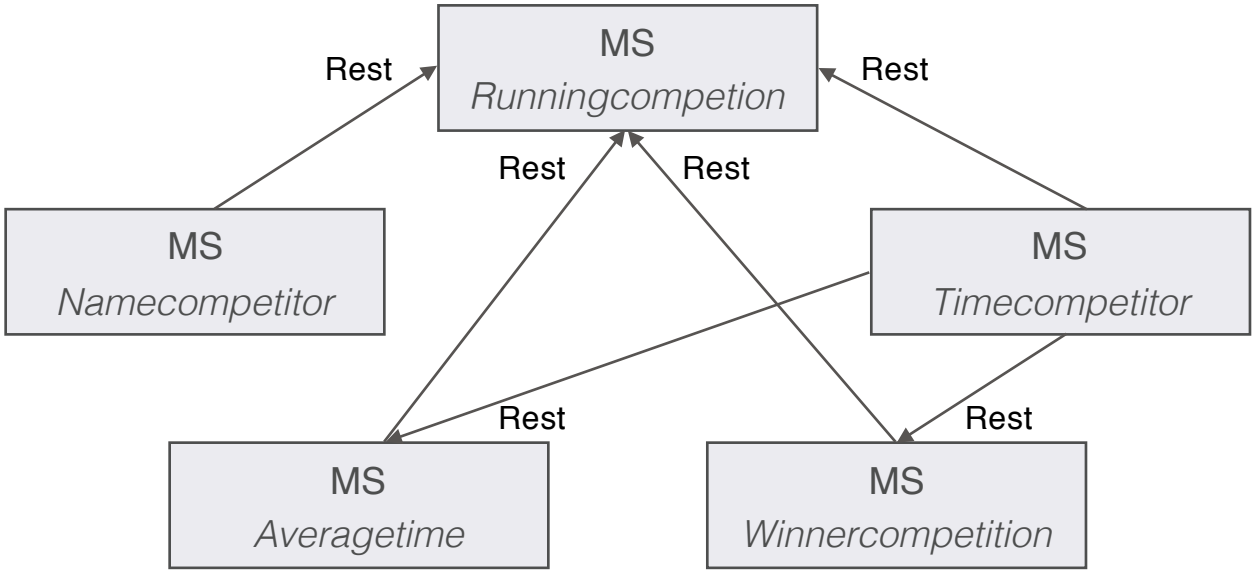

Fig 7 - Architecture micro-services pour une compétition de course à pied

- Le micro-service « Namecompetitor » possède une liste d'informations (nom, prénom) de compétiteurs et leur numéro de dossard (id) associé. Dans ce micro-service les méthodes GET et PUT sont implémentées.
  - GET *http://localhost:8083/competitor/firstname/{id}* nous permet d'avoir le prénom associé au compétiteur en fonction du numéro de dossard (id)
  - GET *http://localhost:8083/competitor/lastname/{id}* nous permet d'avoir le nom de famille associé au compétiteur en fonction du numéro de dossard (id)
  - PUT *http://localhost:8083/competitor/{id}/{firstname}/{lastname}* nous permet d'ajouter les informations (nom, prénom) d'un compétiteur, en fonction de son numéro de dossard (id)
- Le micro-service « Timecompetitor » possède une liste de temps de compétiteurs et leur numéro de dossard (id) associé. Dans ce micro-service les méthodes GET et PUT sont implémentées.
  - GET *http://localhost:8081/time/{id}* nous permet d'avoir le temps (en secondes) d'un compétiteur en fonction de son numéro de dossard (id)
  - PUT *http://localhost:8081/time/{id}/{seconds}* nous permet d'ajouter le temps d'un compétiteur en fonction de son numéro de dossard (id)
- Le micro-service « Winnercompetition » détermine le vainqueur de la course.
  - GET *http://localhost:8082/winner/* nous permet d'avoir le numéro de dossard (id) du vainqueur. Pour se faire, ce micro-service va interroger le micro-service « Timecompetitor » et nous retourner le numéro de dossard du compétiteur au temps de course le plus faible.
- Le micro-service « Averagetime » calcul le temps de course moyen.
  - GET *http://localhost:8084/averagetime/* nous permet d'avoir le temps moyen de la course. Pour se faire, le micro-service va interroger le micro-service « Timecompetitor » et nous retourner la moyenne des temps de course.
- Le micro-service « Runningcompetion »
  - GET *http://localhost:8080/finisher/{id}* → ***« Lors de la course, {lastName} {firstName} est arrivé en {seconds} secondes. »*** Cela nous permet

d'avoir les informations (nom, prénom) et le résultat d'un compétiteur en fonction de son dossard (id). Pour se faire, le micro-service va interroger les micro-services « Timecompetitor » et « Namecompetitor ».

- GET *http://localhost:8080/winner/* → ***« And the winner is ... {lastName} {firstName} avec un temps de {seconds} secondes. »*** Cela nous permet d'avoir les informations (nom, prénom) du vainqueur et son temps de course. Pour se faire, le micro-service va interroger les micro-services « Winnercompetition », « Timecompetitor » et « Namecompetitor ».
- GET *http://localhost:8080/averagetime/* → ***« Le temps moyen de la course est de {averagetime} secondes. »*** Cela nous permet d'avoir le temps moyen de la course. Pour se faire, le micro-service va interroger le micro-service « Averagetime ».

VII. Bibliographie

- Open Classrooms - Mettez en place une architecture pour objets connectés avec le standard oneM2M - [en ligne] : https://openclassrooms.com/fr/courses/5079046-mettez-en-place-une-architecture-pour-objets-connectes-avec-le-standard-onem2m
- Moodle - I5ISSIL11/I5ISSIF11 - Architecture de service/Ingénierie Logicielle - N-tiers architecture - [en ligne] : https://moodle.insa-toulouse.fr/pluginfile.php/102474/mod_resource/content/4/N-tiers-JEE.pdf
- Moodle - I5ISSIL11/I5ISSIF11 - Architecture de service/Ingénierie Logicielle - Service Oriented Architecture (S0A) - [en ligne] : https://moodle.insa-toulouse.fr/pluginfile.php/102530/mod_resource/content/6/SOA-En.pdf
- Moodle - I5ISSIL11/I5ISSIF11 - Architecture de service/Ingénierie Logicielle - Service Orchestration (BPEL) - [en ligne] : https://moodle.insa-toulouse.fr/pluginfile.php/102610/mod_resource/content/2/BPEL-en.pdf
- Moodle - I5ISSIL11/I5ISSIF11 - Architecture de service/Ingénierie Logicielle - Resource Oriented Architecture (Restfull) - [en ligne] : https://moodle.insa-toulouse.fr/pluginfile.php/102621/mod_resource/content/5/ROA.pdf
- Moodle - I5ISSIL11/I5ISSIF11 - Architecture de service/Ingénierie Logicielle - Microservice architecture - [en ligne] : https://moodle.insa-toulouse.fr/pluginfile.php/127888/mod_resource/content/6/Microservice.pdf
- Architectures orientées services SOA - Jean-Paul FIGER - [en ligne] : https://books.google.fr/books?id=FShsogMmzIwC&printsec=frontcover&hl=fr&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false