Master 1 informatique – Université Claude Bernard Lyon 1 – jeudi 14 mars 2019

Numéro de copie d'examen :

TIW1 – Intergiciels et Services – Examen

Durée : 1 h 30 – Documents autorisés (4 pages max) – Ordinateurs, calculatrices, tablettes, téléphones portables… interdits

Questions de cours (barème : 12 points)

Répondre sur l'énoncé.

Maximum : 1 phrase ET 3 lignes (en caractères lisibles). Il est inutile de recopier les slides du cours.

1. Pourquoi PicoContainer ne fait-il pas d'injection de dépendances par interfaces, et par quoi cela est-il remplacé ?
   Cela nécessiterait de pré-définir toutes les interfaces -> annotation de méthodes (+ reflection).
2. Pourquoi faut-il toujours « caster » un objet obtenu par référence depuis un annuaire ?
   Pour être générique, un contexte / annuaire ne stocke que des Object -> cast pour pouvoir utiliser leurs méthodes.
3. Donnez un cas d'utilisation « classique » qui justifie l'utilisation d'un handler logique.
   ... Les CU qui ne nécessitent pas les en-têtes (exemple : validation du payload) ou les CU qui utilisent des informations des en-têtes qui ont été traités par d'autres handlers auparavant (exemple : autorisation).
4. En Java EE, les EJB étaient spécialisés en trois types : session, entité et message. Donnez une raison pour laquelle cette classification n'a-t-elle pas été conservée dans Spring (suite question suivante)...
   Trop restrictive / pas extensible / ne correspond plus aux besoins actuels (ex : plus besoin d'EJB entités depuis JPA).
5. …(Suite de la question précédente) Par quoi cette classification a-t-elle été remplacée dans Spring ?
   Des composants annotés, les annotations pouvant être fournies par des sous-projets de Spring liés à diverses préoccupations (Data, Security…).
6. Pourquoi les services Angular ne possèdent-ils pas de fichiers .html ou .css comme les composants ?
   Les services n'ont pas d'interface -> pas d'arbre DOM -> pas de mise en forme.
7. Indiquez deux fonctionnalités du framework OSGI qui ne pourraient pas être obtenues sans le principe de *Design by Contract*.
   Chargement des modules à distance & màj « à chaud » (no reboot).
8. Pourquoi le langage IDL utilisé en CORBA ne l'est-il pas en RMI ?
   Pas besoin d'un autre langage pour décrire les interfaces : tout est en Java.
9. Indiquez, pour SOAP, un autre intérêt du *Design by Contract*.
   Cacher l'implémentation au client (secret industriel).
10. En quoi l'utilisation d'un outil implémentant des Enterprise Integration Patterns permet-elle de rajouter de la souplesse dans une démarche de conception ?
    On développe les services / handlers séparément et on les assemble ensuite.
11. Pourquoi est-ce une mauvaise pratique d'utiliser un bus de services dans une architecture microservices ?
    « Smart endpoint, dumb pipe ».
12. Indiquez en quoi un outil de mesure des performances d'un SI (Application Performance Management, APM) comme Apache Bench n'est pas suffisant pour mesurer la performance globale d'une application.
    La performance (globale) se mesure côté client ! Par ailleurs, apache bench ne permet pas de mesurer la performance des interactions entre les composants internes au SI, ni la performance des opérations asynchrones

Exercice (barème : 10 points)

Répondre sur la copie d'examen.

On considère un système (simplifié) de gestion de l'entretien d'une flotte de trottinettes électriques proposées à la location par l'entreprise "Trot'bleues". Dans le cadre de cet exercice, on se concentrera sur le changement de batteries des trottinettes.

On considère les informations métier suivantes:

- une trottinette envoie un message lorsque le niveau de la batterie est faible
- chaque trottinette envoie à intervalle régulier sa position au système d'information de "Trot'bleues"
- "Trot'bleues" emploie des coursiers à vélo pour changer les batteries. Chaque coursier dispose d'un terminal qui permet de le géolocaliser et qui contient une application lui indiquant la liste et l'emplacement des trottinettes dont ils doivent changer la batterie.
- "Trot'bleues" a codé un algorithme qui lui permet d'optimiser l'allocation d'une tâche de remplacement de batterie à un coursier (en fonction de la position de la trottinette, de celle des coursiers, des tâches qui leur sont déjà attribué, du profil de chaque coursier, etc).

On considère dans un premier temps que le système d'information de "Trot'bleues" est installé sur une machine unique et qu'un seul serveur répond aux différentes requêtes correspondant à des interactions avec les trottinettes et les coursiers. Il vous est demandé de :

13. Représenter, via un diagramme, les différents messages échangés entre les acteurs du système d'information. Afin de ne pas surcharger le diagramme, on numérotera les messages et on donnera une brève description de chaque message en listant les données contenues dans le message. On laissera de côté les aspects liés à la sécurité.

    Messages à lister à minima : envoi de position (trottinette, coursier), batterie faible, ajout d'une tâche de changement de batterie pour un coursier. Le diagramme doit indiquer qui envoie et qui reçoit un message

14. Proposer une architecture de type microservices afin de faire évoluer le système précédent. Pour cela, représenter un diagramme dans lequel on fera apparaître chaque microservice et ses liaisons avec les autres microservices.

    Décomposer la fonctionnalité du serveur en micro-services, à minima: algorithme d'allocation, gestion des données des coursiers, gestion des données des trottinettes, gestion des tâches de remplacement de batterie, reverse-proxy, bases de données

15. À l'aide d'un diagramme de séquence, décrire les différents échanges de messages entre composants de votre architecture déclenchés par l'envoi d'un message de batterie faible par une trottinette.

    Ici on devrait voir apparaître les échanges entre tous les composants précédemment cités

16. Indiquer quels sont, *a priori*, les composants de votre architecture qui risquent de devenir des goulots d'étranglement en termes de performances. Indiquer comment modifier au besoin votre architecture pour pouvoir effectuer un passage à l'échelle, en indiquant si celui-ci se fera par distribution des données, par décomposition de fonctionnalité et/ou par réplication des services.

    Deux exemples : on peut imaginer que l'algorithme d'allocation est couteux : il faut prévoir une réplication des services pour l'algorithme d'allocation. On peut également imaginer que la mise à jour des informations de géolocalisation peut devenir problématique si on est en présence de trop nombreuses trottinettes et coursiers : il faut prévoir de distribuer les données (chaque trottinette / coursier étant indépendants des autres). À noter que si le découpage n'est pas assez fin à la question 14, il peut être nécessaire de décomposer des fonctionnalités.