ISA-DEVOPS – Contrôle écrit

- Date : Avril 2023
- Durée : 3 heures
- Aucun document autorisé ; barème donné à titre indicatif.

Lisez tout le sujet avant de commencer à répondre aux questions ; les questions sont identifiées en gras dans le texte ; les différentes parties sont indépendantes.

En cas de doute ou d'inconnues, posez vos hypothèses en tête de votre réponse

Toute fraude identifiée sera systématiquement transmise au conseil de discipline de l'Université

ISA Partie #1 : Principes /4

Répondez à chaque question de manière synthétique (~150 mots, 200 max par réponse) :

1. Que signifie *stateless* dans le cadre des contrôleurs REST Spring ?

2. A quoi sert le *cascading* dans la gestion de la persistance ? Donnez un exemple.

ISA Partie #2 : Étude de cas /8

Le système multi-fidélités de votre projet est déployé et ce sont de nouvelles fonctionnalités qu'il faut déjà implémenter ! Voici le briefing d'une fonctionnalité à étudier dans cette question :

Plusieurs fois par an, les commerçants d'une zone de carte multi-fidélités peuvent organiser le « Fusion Game » :

- *Plusieurs personnes peuvent fusionner des points obtenus sur chacune de leur carte sur une seule carte choisie au début de la fusion. Chacun détermine combien de points, ils souhaitent donner et les points « fusionnés » se retrouvent sur la carte « fusion » dans un compte de points spécifiques (le porteur de la carte qui va devenir « fusion » peut lui aussi reverser des points classiques vers les points « fusion »).*
- *Pour simplifier, les points fusionnés ne peuvent pas être repris, ils restent sur la carte « fusion » et peuvent être utilisés dans un autre « fusion game » s'ils ne sont pas tous utilisés. Toute carte peut devenir « fusion », il n'y a rien de spécial à faire.*
- *Seuls des cadeaux spécifiques « fusion » sont accessibles durant la période avec les points « fusion » (ce sont normalement des cadeaux importants, mais nécessitant un grand nombre de points). Ces cadeaux sont disponibles auprès des différents magasins, comme des cadeaux classiques.*
- *Un partenariat est aussi effectué avec une association caritative, potentiellement différente à chaque « fusion game ». Des points « fusion » peuvent être donnés à l'association et le système devra permettre de sortir un décompte complet des points donnés à la fin de chaque « fusion game » pour que le syndicat des commerçants verse directement la somme correspondante à l'association caritative (le montant versé doit pouvoir être récupéré par le front du site web pour l'affichage de chaque don complet à la fin de chaque fusion game).*

Points à aborder

Sur la base des informations contenues dans ce briefing, **et en posant toutes les hypothèses qui vous semblent cohérentes**, vous devez proposer une extension de l'architecture 3-tiers du

système Multi-fidélités, implémentable en Spring selon les principes vus en cours ISA (notamment avec persistance par ORM sur une BD relationnelle).

Pour chaque question qui suit, vous pourrez récupérer des composants, interfaces et objets métier de votre projet actuel en expliquant comment vous les réutilisez et/ou étendez. Vous ne présentez bien évidemment que ceux qui sont utiles à l'extension demandée.

Les points suivants vous permettront de cadrer votre étude de cas :

1. Identifiez les différents composants (composants et contrôleurs REST) à mettre en jeu dans votre architecture ;
   - **Décrivez l'assemblage** sous la forme d'un diagramme de composants ;
   - **Décrivez les interfaces** de chaque composant (signatures typées des interfaces en pseudo-code).
   - **Décrivez les routes REST** servis par les contrôleurs (donnez juste la route avec ses paramètres, pas d'annotation Spring ou d'autres définitions lourdes).
2. Identifiez les objets métiers au sein de votre architecture ;
   - Décrivez ces objets sous la forme d'un diagramme de classes ;
   - Expliquez le *mapping* Objet-Relationnel quand c'est pertinent.
3. **Justifiez vos choix de conception** pour cette architecture : il s'agit d'argumenter pourquoi vous avez découpé les composants de telle ou telle manière (qu'est-ce que cela permet en termes de découpage des traitements, des responsabilités, etc.), en quoi les interfaces permettent de bien gérer le métier et les responsabilités, en quoi les objets métiers que vous proposez permettent de capturer et structurer les données importantes du métier dont le comportement est géré par les composants.
4. Décrivez les deux scénarii suivants en expliquant quels objets sont utilisés et quels composants/interfaces sont traversés par plusieurs requêtes pour traiter le scénario :
   a. Le don de points d'une carte vers une carte qui devient « fusion » ;
   b. Le calcul final des points donnés à l'association à la fin d'un « fusion game ».

N'oubliez pas que les justifications ont une très grande importance, des propositions bien justifiées rapporteront des points même si elles sont incomplètes.

Devops Partie #1 : Principes /3

Répondez à chaque question de manière synthétique (~150 mots, 200 max par réponse) :

1. Artifactory est décrit comme étant un « gestionnaire universel de dépôts d'artefacts». Expliquez le but de ce type de dépôts et des cas d'utilisations.

2. Votre collègue a déployé sur sa machine un artifactory et un jenkins au moyen de deux docker-compose. Celui-ci n'arrive pas à faire communiquer artifactory et jenkins. La stacktrace dans les logs Jenkins indique « Host not found : my_container_artifactory ». Quelle est votre démarchage de résolution de ce problème ? Expliquez quels types d'erreurs sont possibles ?

Devops Partie #2 : Étude de cas /5

Préambule (source : https://fr.wikipedia.org/wiki/RISC-V)

RISC-V (prononcé en anglais « RISC five ») est une architecture de jeu d'instructions [...] RISC ouverte et libre, disponible en versions 32, 64 et 128 bits. Ses spécifications sont ouvertes et peuvent être utilisées librement par l'enseignement, la recherche et l'industrie.

Ce projet, [...], avait d'abord une visée d'étude et de recherche, mais est devenu de facto un standard d'architecture ouvert dans l'industrie. Le but de ce projet est de faire un standard ouvert de jeu d'instructions de microprocesseur, à l'image du standard TCP/IP pour les réseaux ou d'UNIX pour les systèmes d'exploitations [...].

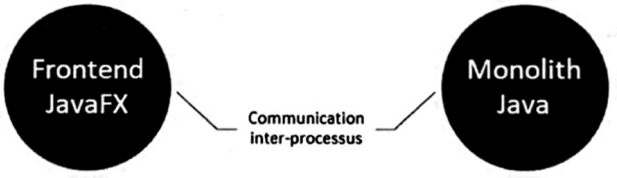
[...]

En décembre 2021, RISC-V ratifie 15 nouvelles spécifications jusqu'alors à l'état de brouillon (draft), ajoutant en tout 40 nouvelles extensions optionnelles, dont les extensions vectorielles ou de chiffrement standard.

Clap !

Fort de sa croissance exponentielle depuis plusieurs années, vous décidez de rejoindre une jeune start-up nommée *HighFive!* qui développe un logiciel monolithique Java qui aide à la conception de puces électroniques (System on a chip) basées sur RISC-V. Ce logiciel est un client "lourd", c'est à dire qu'il nécessite l'usage d'une interface graphique propre au programme (en opposition avec les clients "légers" qui reposent sur un navigateur web).

Le logiciel est composé d'un monolithe Java, qui affiche une interface graphique basée sur JavaFX.



Topologie entre l'interface graphique et le backend métier sous forme de monolithe

Jusqu'à présent, la maintenance et les évolutions du code de l'application se faisaient sur les postes de travails et par échange d'archives "ZIP" par email entre les collaborateurs.

D'autre part, le nombre de contributeurs sur le projet est passé d'un développeur (le fondateur) à une dizaine en l'espace de quelques mois, afin de répondre aux attentes des clients et aux évolutions permanentes de l'architecture (extensions).

Votre mission

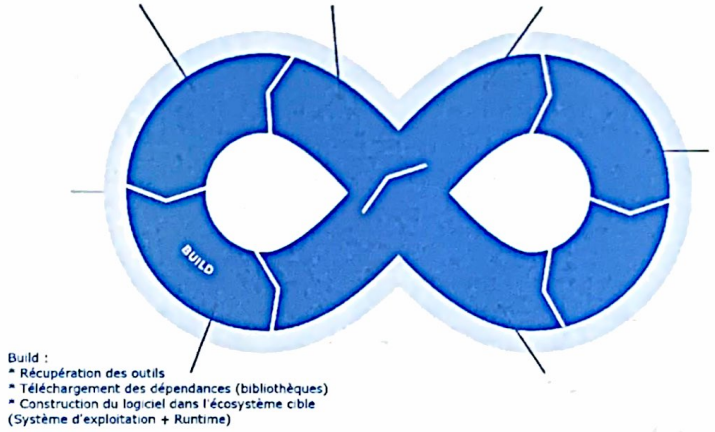
Répondez aux points suivants, en appliquant les pratiques DevOps :

- [A] Une boucle de développement qui permet de construire et distribuer en toute confiance un produit de qualité
  - Proposez et justifiez un schéma avec les étapes de transition (annotées) entre les différentes phases. Exemple : **Annexe 1.1**
- [B] Un schéma de travail collaboratif afin de livrer les nouvelles fonctionnalités de manière continue avec confiance
  - Proposez et justifiez un schéma des stratégies sur les étapes de Création / Fusion / Intégration d'une branche de développement. Exemple : **Annexe 1.2**
- [C] Les technologies et services requis pour construire une chaine d'intégration et de déploiement continue
  - Listez les briques techniques requises pour implémenter la forge logicielle décrite ci-dessus, avec une justification.

- [D] Une méthodologie de test afin garantir que l'interface graphique a le comportement attendu par rapports aux réactions du backend applicatif
    - Expliquez comment qualifier que l'interface graphique du produit est conforme tout au long du cycle de vie du produit. Proposez des types de tests, justifiez leur intérêt et quand les exécuter dans un pipeline d'intégration continue.
- [E] Une méthode de déploiement pour distribuer le logiciel auprès des utilisateurs
    - Proposez et justifiez une méthode de distribution *pertinente* du logiciel, à destination d'un utilisateur final.

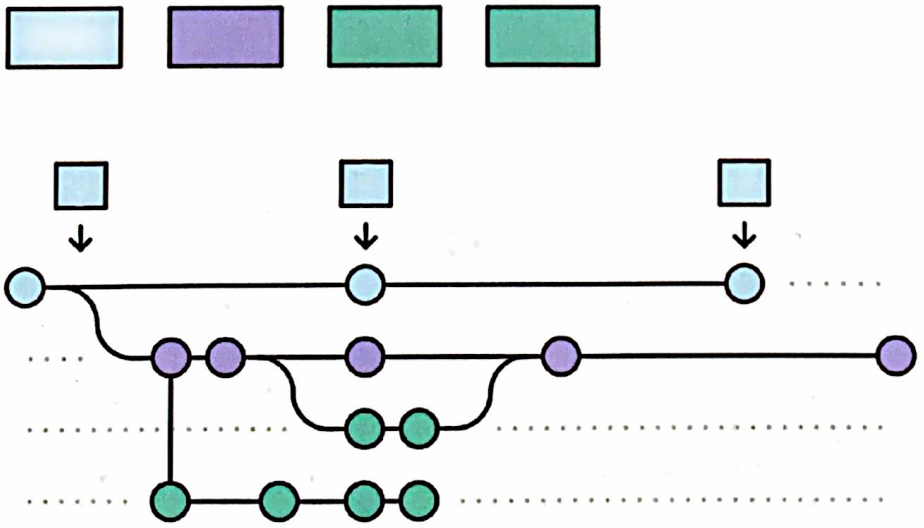
Annexes

Annexe 1.1



Exemple de boucle de développement, avec emplacements des annotations.

Annexe 1.2



Exemple de squelette pour un schéma explicitant la stratégie de branching. Remarque : il manque ici des explications sur l'usage et les conditions de création / fusion d'une branche !