Travail pratique 2 Partie 2

Exploration et Comparaison des Approches de Décomposition d'Applications Monolithiques en Microservices

Introduction

Objectif du TP :

L'objectif de ce TP est de vous permettre d'explorer le domaine de la décomposition d'applications monolithiques en microservices. Vous aurez l'occasion de mettre en pratique trois approches automatiques bien établies, en plus de la décomposition manuelle. Ce TP offre une opportunité d'appliquer différentes méthodes de décomposition, de comprendre leur fonctionnement et de comparer leurs résultats. Vous pourrez ainsi acquérir une expérience concrète des défis liés à la transformation d'applications monolithiques en microservices,

La migration d'une monolithique vers une architecture de microservices est un processus long qui comprend plusieurs étapes telles que la définition de l'objectif, la décomposition du monolithe et la transformation du code. Dans ce TP, nous ne considérerons que l'étape de décomposition. Plus précisément, nous nous concentrerons sur le partitionnement des classes au sein de ces applications monolithiques en microservices. Ainsi, pour chaque classe dans ces systèmes, nous devons identifier à quel microservice elle appartiendra. La définition du nombre de ces microservices et de leurs noms fait partie de la tâche.

Ce travail doit être fait en équipe de 2 ou 3.

Applications monolithiques

Les applications monolithiques qui sont considérées sont :

- **Spring PetClinic** : Une application monolithique de gestion de clinique vétérinaire en utilisant le Framework Spring.

- **Plants** : Plants sert de démonstration pour les technologies et bonnes pratiques de Java en implémentant une plateforme d'achat de plantes.

Sources fournies

Tous les outils et les données nécessaires sont disponibles dans le répertoire suivant : https://github.com/khalsel/glo-4008-7008-TP2

Le répertoire comprend également des guides pour utiliser les approches de décomposition et compiler les applications monolithiques. Pour chaque projet monolithique, vous aurez accès à un fichier appelé "{nomduprojet}_monolith.json" qui inclut toutes les classes que vous devrez prendre en compte lors de la décomposition. Voici un exemple de ce fichier :

```
[
    {
        "name": "monolithe",
        "classes": [
            "org.example.classExemple1",
            "org.example.classExemple2",
            "org.example.classExemple3"
        ]
    }
]
```

Définitions :

Nous définissons une décomposition comme une liste de microservices. Un microservice est défini comme une structure avec 2 paires clé/valeur. Il est composé d'un nom et d'une liste de noms de classes définis respectivement dans les clés "name" et "classes".
Chaque décomposition doit être définie dans un fichier JSON avec le format suivant : `List[Object[["name":string, “classes”:List[string]]]`.
Chaque classe doit être dans un seul microservice.
Une décomposition potentielle de l'exemple de monolithe peut être définie comme suit :

```
[
    {
        "name": "microservice1",
        "classes": [
            "org.example.classExemple1"
        ]
    },
    {
        "name": "microservice2",
        "classes": [
            "org.example.classExemple2",
            "org.example.classExemple3"
        ]
    }
]
```

Travail demandé

1. Décomposition manuelle : (20 points)

La phase de décomposition est généralement réalisée manuellement par les développeurs et les experts du domaine. Il existe plusieurs processus que les développeurs peuvent suivre

pour accomplir cette tâche. Votre objectif dans cette question est d'utiliser un processus manuel pour générer une décomposition pour l'application PetClinic.

Bien que vous soyez libre de choisir le processus que vous estimez le mieux adapté à cette tâche, nous recommandons les étapes suivantes :

- Analyser et comprendre le monolithe : Cela peut être réalisé en lisant la documentation de l'application, son code source et ses documents de conception (tels que les graphiques de dépendance et les diagrammes de cas d'utilisation).

- Extraire les contextes délimités (« bounded context » de la méthodologie Domain Driven Design) de l'application : Chaque contexte délimité représentera un microservice. Par exemple, lors de la décomposition d'une application de shopping, le "paiement" est un contexte délimité potentiel.

- Définir les dépendances entre les contextes délimités : Établir les relations de dépendance entre les différents contextes délimités.

- Associer chaque classe à un contexte délimité et, par conséquent, à un microservice : Certaines classes peuvent être associées à plusieurs contextes délimités et, en tant que tel, une décision doit être prise quant à savoir à quels microservices elles devraient appartenir.

a- Quelle que soit votre approche, décrivez les détails de chaque étape. Toutes les classes du fichier "petclinic_monolith.json" doivent être incluses dans la décomposition. Chaque classe manquante entraînera une pénalité. Enregistrez votre décomposition générée dans un fichier appelé "manual_petclinic.json". (L'utilisation d'approches de décomposition automatisées est interdite dans cette question)

2. Mono2micro : (20 points)

Au cours de votre formation, vous avez appris l'existence d'un outil appelé Mono2micro qui tente d'automatiser le processus de décomposition. Pour cette raison, vous décidez de l'appliquer sur PetClinic et Plants dans l'espoir d'obtenir de meilleurs résultats.

a- En utilisant vos connaissances de Mono2micro issues du cours et des laboratoires, appliquez-le sur les deux projets monolithiques et enregistrez les résultats dans les fichiers "m2m_petclinic.json" et "m2m_plants.json".

b- Le développeur peut-il influencer la qualité de la sortie de Mono2micro par le biais de ses entrées ? Si oui, décrivez ces entrées et comment vous les avez modifiées.

3. MSExtractor : (20 points)

Afin d'avoir plus d'options et de perspectives différentes, vous avez décidé d'appliquer une autre approche automatisée de décomposition sur ces applications monolithiques. Pour cette raison, vous avez choisi MSExtractor. Contrairement à Mono2micro, MSExtractor est basé sur l'analyse statique du code source de l'application. Les résultats de l'analyse statique sont ensuite analysés et, à travers un algorithme évolutif, l'approche génère une décomposition.

a- En utilisant les images Docker des outils d'analyse statique fournies sur le code source de PetClinic et Plants, générez les données requises pour MSExtractor. Afficher les fichiers générés à la fin de cette étape.

b- Appliquez MSExtractor sur PetClinic et Plants en utilisant les résultats de l'analyse statique pour générer leurs décompositions. Enregistrez les résultats dans les fichiers "mse_petclinic.json" et "mse_plants.json" respectivement.
c- Le développeur peut-il influencer la qualité de la sortie de MSExtractor par le biais de ses entrées ? Si oui, décrivez ces entrées et comment vous les avez modifiées.

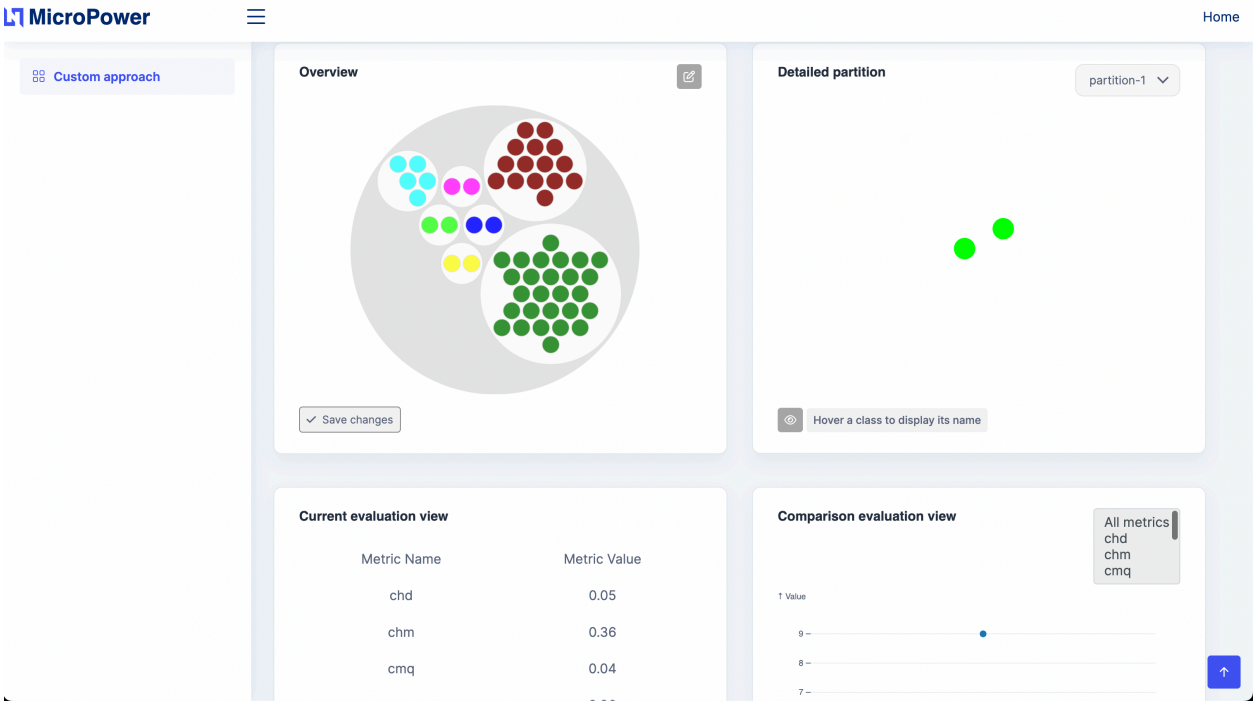
4. TopicDecomp : (20 points)

En approfondissant la recherche sur les approches automatisées de décomposition, vous découvrez une nouvelle approche qui combine l'analyse statique avec la modélisation de sujets (topic modeling) [1]. La modélisation de sujets est une technique d'apprentissage automatique non supervisée qui consiste à découvrir et à détecter des motifs sémantiques dans un ensemble de documents. Dans notre cas, les documents sont les classes et leur code source. Donc, l'approche analyse le contenu textuel associé à chaque classe, identifiant des thèmes ou des contextes spécifiques pour proposer une décomposition en microservices plus contextuelle.

a- Appliquez TopicDecomp sur PetClinic et Plants. Enregistrez les résultats dans les fichiers "topic_petclinic.json" et "topic_plants.json" respectivement.
b- Le développeur peut-il influencer la qualité de la sortie de TopicDecomp par le biais de ses entrées ? Si oui, décrivez ces entrées et comment vous les avez modifiées.

5. Évaluation : (30 points)

À ce stade, vous avez plusieurs décompositions pour PetClinic et Plants. Cependant, vous devez choisir une seule décomposition pour chacun. Heureusement, vous avez trouvé une plateforme qui, étant donné une décomposition en entrée, permet sa visualisation et fournit des métriques d'évaluation. Voici une capture d'écran de la plateforme :



Les métriques considérées sont définies comme suit :

- CMQ : Quantifie la qualité conceptuelle de la décomposition. Les composants de cohésion et de couplage de cette métrique sont basés sur les termes textuels communs entre les classes.
- SMQ : Quantifie la qualité structurelle de la décomposition. Les composants de cohésion et de couplage de cette métrique sont basés sur appels et les interactions entre les classes dans le code source.
- ICP : Quantifie les dépendances entre chacun des microservices et peut ainsi représenter le critère de couplage.
- IFN : Évalue le nombre d'interfaces dans un microservice. Des valeurs plus basses de l'IFN indiquent des recommandations de meilleure qualité.
- NED : Peut quantifier l'aspect de granularité des microservices suggérés. Il pénalise les microservices extrêmement petits ou extrêmement grands.
- COV : Mesure le ratio des classes utilisées par l'approche de décomposition par rapport à l'ensemble des classes du système monolithique.
- MSN : Le nombre de microservices.

a- Lancez la plateforme et chargez une décomposition. Quels sont les services qui composent cette plateforme ?

b- Les métriques d'évaluation définies représentent différents aspects de la décomposition. À votre avis, quelle métrique devrait être utilisée lors de la sélection de la décomposition ?

c- La plateforme offre quelques panneaux utiles pour évaluer une décomposition :
   - Le panneau "vue d'ensemble" visualise la décomposition. De plus, il est possible de modifier la décomposition via ce panneau en sélectionnant une classe et en changeant le microservice auquel elle appartient.
   - La "vue principale" montre les dépendances entre les microservices ainsi que les dépendances entre les classes au sein de chaque microservice.
   - La "vue d'évaluation actuelle" montre les résultats d'évaluation de la décomposition actuelle tandis que la "vue d'évaluation comparative" garde trace de leurs changements à travers les modifications de la décomposition.

   En utilisant cette plateforme, examinez et comparez les décompositions que vous avez générées précédemment. Il est autorisé de générer une nouvelle décomposition avec les approches précédentes en utilisant les métriques d'évaluation et la plateforme comme guide. Il est également possible de modifier la décomposition. Fournissez votre décomposition finale dans un fichier appelé "final_petclinic.json" pour PetClinic et "final_plants.json" pour Plants.

d- Sur quelle approche de décomposition parmi manuelle, Mono2micro, MSExtractor et TopicDecomp avez-vous basé vos résultats ? Pourquoi avez-vous sélectionné cette décomposition de base par rapport aux autres ?

e- Avez-vous modifié la décomposition pendant la phase d'évaluation ? Si oui, quelles modifications avez-vous apportées et pourquoi ? Sinon, pourquoi la décomposition de base était-elle suffisante ?

Questionnaire de Clôture (10 points)

Le TP se conclura par un questionnaire, joint à cet énoncé, où vous justifierez vos choix de décomposition finale, d'approche préférée et de modifications apportées. Vous répondrez également à des questions évaluant votre expérience avec les approches.

Format de la Soumission

Votre remise devrait contenir les fichiers suivants :

- Un fichier « rapport.pdf » contenant votre rapport. Ce fichier doit obligatoirement être basé sur le gabarit de rapport fourni avec l'énoncé.
- Un dossier nommé "decompositions" contenant toutes les décompositions générées pour chacune des questions.
- Un fichier « questionnaire.pdf » contenant vos réponses du questionnaire. Il doit obligatoirement être basé sur le fichier du questionnaire.
- Aucun autre fichier.