ZAGHOUANE Fares
CHOURAQUI Maxime
CLEMENT Charles
LUPPI Daniel

Groupe 1
4 ETI
Majeure CLBD

Architecture des Systèmes d'Information

OBJECTIF

Concevoir une architecture logicielle respectant les bonnes pratiques de programmation et les patterns d'architecture micro services.

SUJET

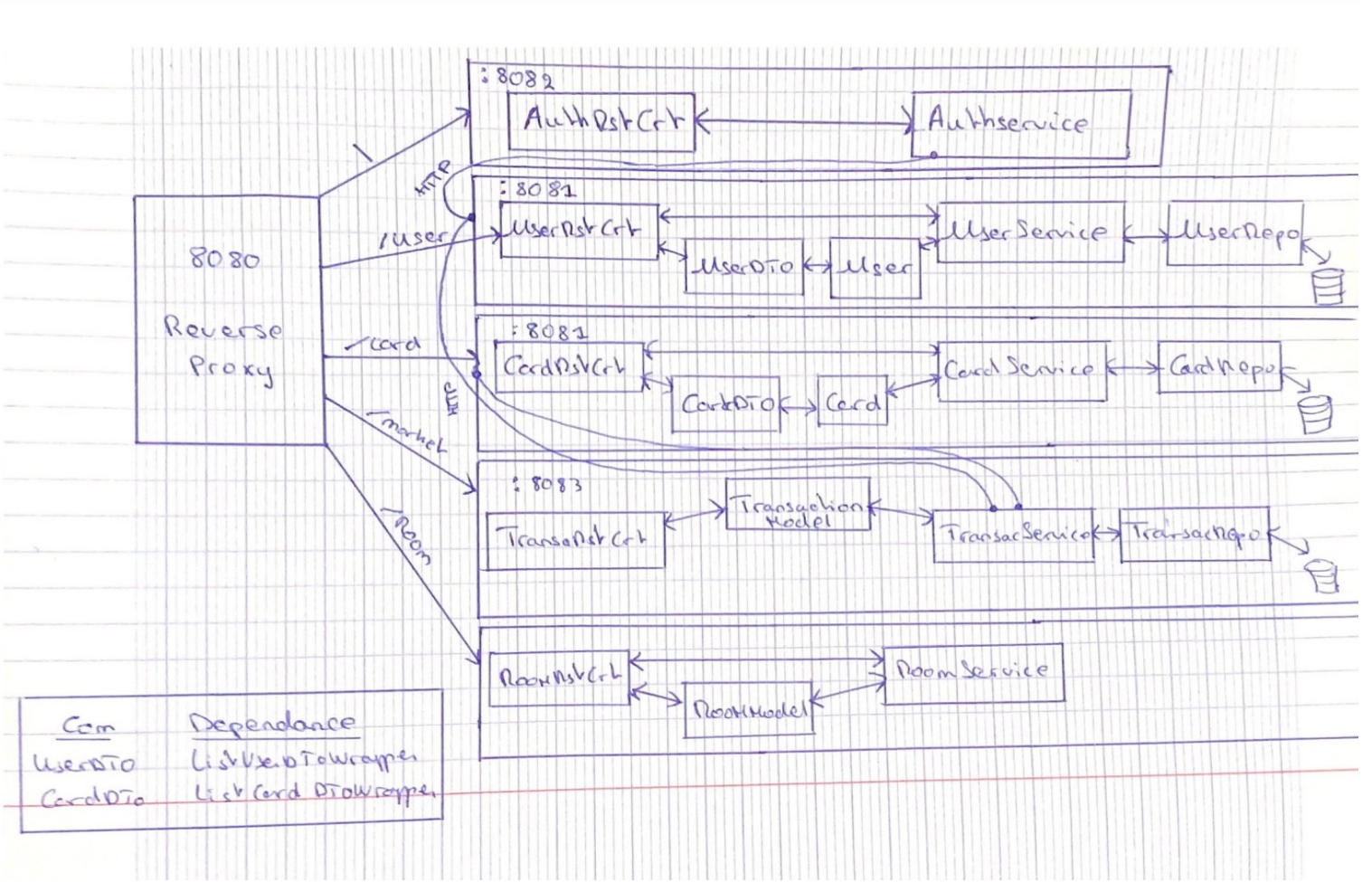
Réalisation d'un système de jeu de cartes incluant les fonctionnalités suivantes :

- Création de cartes
- Achat/vente de cartes
- Gestion d'utilisateurs (ajout/connexion)
- Compétition de cartes

SOMMAIRE

I – Architecture du projet

Voici un schéma de la solution d'architecture retenue pour notre jeu de carte. Grâce à celle-ci nous respectons le cahier des charges, à savoir : la création de carte, la création et la gestion d'utilisateurs ou encore l'achat et la vente de carte. Il s'agit d'une architecture microservice comme suit :



II – Avantages/différences architecture Microservice versus SOA

Rappelons dans un premier temps les différences majeures entre les architecture microservice et les SOA mais aussi les avantages proposés par ces solutions.

Quels intérêts présentent les micros services comparés aux architectures SOA?

- Les SOA impliquent souvent le partage de composants.
- Services plus précis.
- Les SOA impliquent le partage du stockage de données entre les services.
- Les microservices utilisent REST.
- Déploiement moins flexible

Quelles sont les différences entre les micros services et le SOA?
Quel intérêt présente l'usage de docker et des micros services?

Les micros services sont conçus pour les services pouvant fonctionner de façon autonome, tandis que le SOA est conçue pour partager les ressources entre les services. A l'inverse des micros services, le SOA propose des services plus importants et modulables. Enfin, les microservices sont plus adaptés aux applications de petit calibre basées sur le web, le SOA aux intégrations à plus grande échelle.

En ce qui concerne notre projet, nous avons opté pour une architecture microservice car ceux-ci sont plus adaptés aux applications WEB de petit calibre. De plus, cela nous évitais un partage de composant et de stockage de données entre les différents services. Cela nous a permis un développement plus aisé lors de l'ajout de fonctionnalité tels que les 'Rooms'. Cela nous a aussi permis de mieux gérer les accès aux bases de données qui se retrouvent séparées. Ainsi, on évite de les surcharger.

III – Couverture de tests effectués

Pour ce qui est des différents test effectués, nous devions utilisé SONAR un outil de mesure de la qualité d'un code en Java, PHP ou C++. Celui-ci se base sur six axes pour son analyse :
- Identification des duplications
- Mesure du niveau de documentation
- Respect des règles de programmation
- Détection des bugs potentiels
- Evaluation de la couverture de code par les test unitaires
- Analyse de la répartition de la complexité

Cependant, la mise en place de Sonar nous a pris beaucoup de temps à configurer et n'a pas pu être mené a bien. Ainsi, la majorité des test effectués dans le cadre de ce projet sont des test fait manuellement via Postman. Le logiciel nous a permis de réaliser des test unitaires : ceux-ci visent à décomposer un code source en plusieurs parties et de vérifier que chacune d'entre elles fonctionnent. Nous avons ensuite réalisé des tests fonctionnels afin de vérifier le bon fonctionnement des parties de code entre elles.