ATELIER III -12h-

COMPETENCES

Concepts :

- Architecture logicielle MicroService
- Différence entre architecture SOA et Microservice
- Avantages/inconvénients des Microservices
- Bonne pratique de programmation, Test Unitaires, Couverture de tests
- Haute dispo., virtualisation

Techno :

- Maven, Spring boot
- Junit, Mokito, Sonar
- Javascript, AJAX

SUJET

Après avoir analysé plus en détail votre application vous souhaitez basculer vers une architecture MicroService à l'aide de **Spring Boot**. Afin de convaincre votre client du bien fondé de votre décision rédiger un rapport synthétisant les avantages et **différences** des **MicroServices** face aux architectures **SOA.**

Une fois votre production précédente transformée en **MicroService ,** mettre en place une politique de **tests unitaires** permettant de tester le fonctionnement de votre application (services Web, **Mokito**) ainsi que le fonctionnement de vos programmes Java (**Junit**,…).

Réaliser un état des lieux de votre couverture de tests à l'aide d'application telle que **Sonar**.

Une fois votre application mise à jour, ajouter de nouvelles fonctionnalités :

- Création de Room (zone de jeu) ou un joueur peut déterminer une mise et un nom de Room.
- Donner la possibilité aux utilisateurs de participer à ces Rooms après avoir sélectionné 1 carte pour jouer.
- Une fois les deux utilisateurs prêts (une carte sélectionnée par joueur), le jeu commence :
    - chaque carte effectuera n coups (impacts des coups dépend de la puissance de la carte, des règles d'affinité e.g feu vs eau, et d'une partie aléatoire)
    - le jeu prend fin lorsqu' une carte à son HP à 0.
    - Le joueur gagnant la manche remporte la mise.
    - Automatique X pt d'énergie sera retiré de chaque carte. Si un joueur possède une carte d'énergie inférieur <Y alors il ne pourra pas jouer cette carte lors d'un prochain combat.
- Z pts d'énergie sont gagnés par les cartes de tous les joureurs toutes les N secondes.

Procéder au même professionnalisme que précédemment en commentant votre code, réalisant des tests unitaires et des tests de votre application (incluant Web Services)

[Bonus] Dans le cas d'un usage massif de votre application et par souci de maintenabilité, héberger chaque micro-service dans un container **docker**

[Bonus] Afin de répondre à la demande grandissante d'usage de votre application configurer un **serveur de répartition de charge** (e.g **ngnix**) permettant de distribuer les requêtes entrantes.

Présenter l'**architecture technique** de votre application ainsi qu'un **plan de tests** de non régression dans un rapport que vous fournirez aux clients.

QUESTIONS

Quelle est la différence entre un test fonctionnel et un test unitaire ? A quoi sert la couverture de code ?

Qu'est ce qu'un test de non régression ? à quoi sert-t-il ?

Expliquer le principe de développement « test driven » ?

Quels intérêts présentent les micros services comparés aux architecture SOA ?

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

Qu'est-ce que docker ? En quoi diffère-t-il des méthodes de virtualisation dites classiques (vmware, virtualbox) ?

Quelle organisation en équipe permet la mise en œuvre de micro services ?

Que permet de faire l'outil Sonar ?

Qu'est ce que l'intégration continue ? Quels avantages/contraintes présentent cette organisation ?

REFERENCES ****(indispensable) **(important) *(optionnel)*

[Test on Spring boot]

- *** http://www.springboottutorial.com/spring-boot-unit-testing-and-mocking-with-mockito-and-junit
- *** http://www.springboottutorial.com/unit-testing-for-spring-boot-rest-services
- ** https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-testing.htm

[Sonar and Springboot]

- *** https://thepracticaldeveloper.com/2016/02/06/test-coverage-analysis-for-your-spring-boot-app/

[MicroServices]

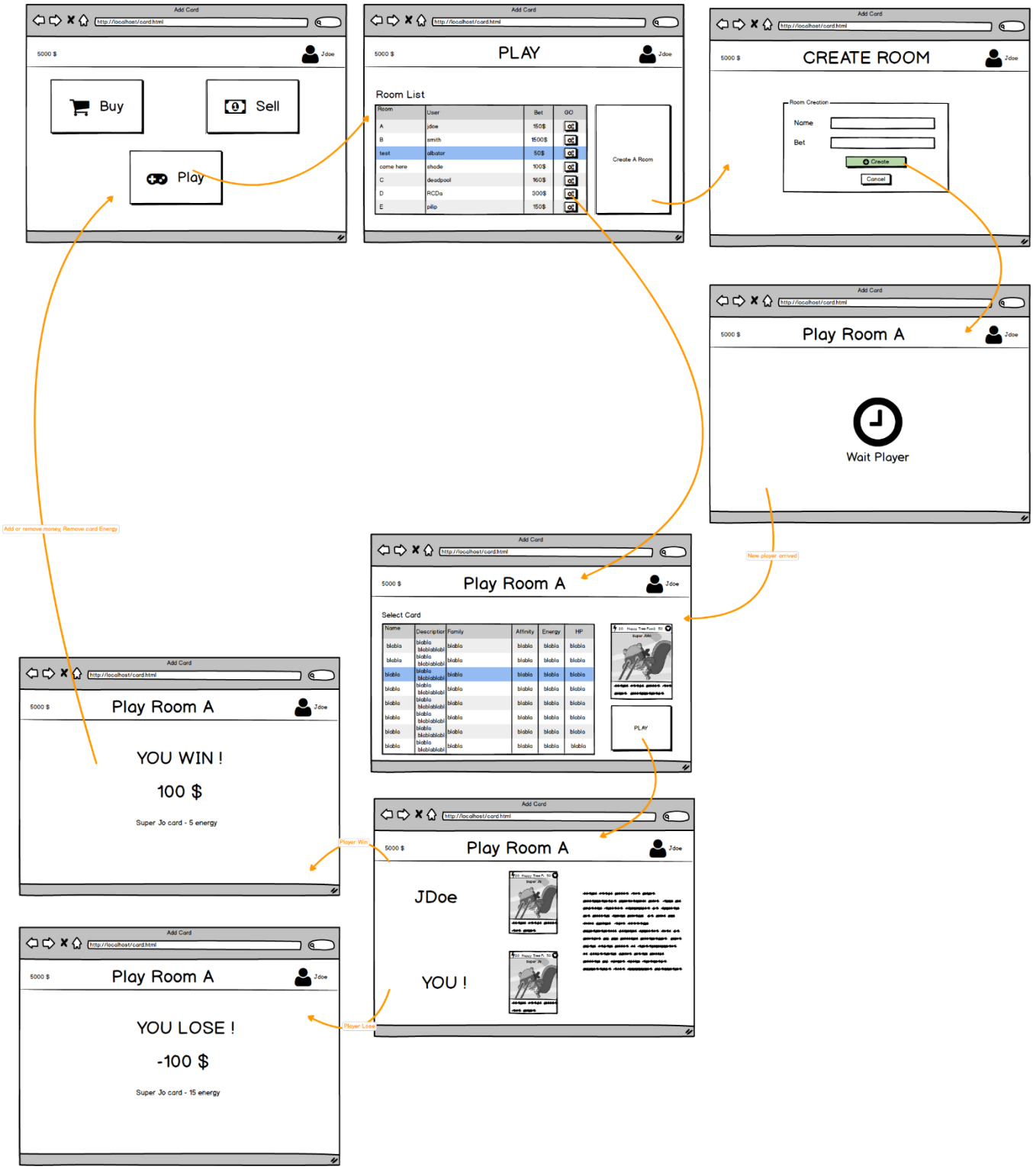
- *** http://blog.xebia.fr/2015/03/02/microservices-les-concepts/
- *** http://blog.xebia.fr/2015/03/09/microservices-des-architectures/
- ** http://blog.xebia.fr/2015/03/16/microservices-des-pieges/
- ** http://www.oreilly.com/programming/free/migrating-cloud-native-application-architectures.csp
- * http://blog.xebia.fr/wp-content/uploads/2016/01/Microservices-Programmez1.pdf

[Commuication MicroServices]

- *** https://o7planning.org/fr/11647/exemple-spring-boot-restful-client-avec-resttemplate

ELEMENTS DONNES

- IDE configuré
- Ensemble des éléments graphiques fournis dans les mokups
- Eléments d'affichage plus javascript simple en semantic UI
- 1 sketch


Add Card
http://localhost/card.html
5000 $
Jdoe
Buy
Sell
Play
PLAY
Room List
Room
User
Bet
GO
A
jdoe
150$
B
asmith
1500$
test
albator
50$
come here
ahade
100$
C
deadpool
160$
D
RCDs
300$
E
pilip
150$
Create A Room
CREATE ROOM
Room Creation
Name
Bet
Create
Cancel
Play Room A
Wait Player
Add or remove money, Remove card Energy
New player arrived
Select Card
Name
Description
Family
Affinity
Energy
HP
blabla
PLAY
YOU WIN !
100 $
Super Jo card - 5 energy
Player Win
JDoe
YOU !
Player Lose
YOU LOSE !
-100 $
Super Jo card - 15 energy