

ATELIER I -12h-

COMPETENCES

Concepts :

- Différentes architectures de conception logicielle
 - MVC
 - SOA
 - Bus de communication
- Avantages/inconvénients des architectures
- Monolithique to SOA
- Découpage logique des projets (Front, Back)
- Interaction Back-Front

Techno :

- ActiveMq
- Spring boot, Maven
- React, Javascript, AJAX

SUJET

1. PHASE A :

Une entreprise disposant uniquement d'un BackEnd d'application de gestion de cartes (monolithique) souhaite faire évoluer son application.

Ayant entendu des approches SOA (Service Oriented Application), cette dernière fait appel à vous afin de transformer son application

- 1.1. Après avoir analysé et réalisé un diagramme d'architecture et un diagramme de classe de l'existant, proposer une nouvelle organisation du BackEnd répondant aux exigences du SOA (organisation et architecture uniquement) mêlant micro service et ESB.

FAIRE VALIDER VOTRE APPROCHE

- 1.2. Réaliser un tableau récapitulatif des bus de communication les plus répandus (avantage/inconvénients)
- 1.3. Transformer le BackEnd existant suivant vos recommandations

FAIRE VALIDER VOTRE APPROCHE

2. PHASE B :

L'entreprise ne dispose pas de FrontEnd et souhaite utiliser un Framework FrontEnd pour le réaliser

- 2.1. Réaliser un tableau comparatif des principaux Framework FrontEnd existants dressant les avantages et inconvénients de chacun.

Votre choix se porte sur le Framework ReactJS. Votre application devra remplir les fonctionnalités suivantes :

- Une connexion basique de l'utilisateur
- Un écran d'accueil
- La possibilité d'acheter des cartes parmi une liste
- La possibilité de vendre les cartes parmi une liste

2.2. En vous inspirant des spécifications proposées par le client, réaliser un découpage du FrontEnd en composant REACTJS

FAIRE VALIDER VOTRE APPROCHE

2.3. Réaliser votre application REACTJS sans interaction avec le BackEnd dans un premier temps

FAIRE VALIDER VOTRE APPROCH

2.4. Finaliser votre application REACTJS en finalisant les interactions avec le BackEnd

FAIRE VALIDER VOTRE APPROCHE

TIPS

- Penser à versionner vos projets (utiliser gitlab)
- Afin de paralléliser le travail, autoriser le **CROSS ORIGIN** sur vos navigateurs web **ET** sur votre BackEnd
- Utiliser une Bus de communication ActiveMq indépendamment de Springboot.

BONUS

- Utiliser les pipelines de Gitlab pour l'Intégration Continuer (CI), le déploiement continu (CD)
- Utiliser les composants Material-UI de REACT (attention customisation des composants plus complexes)

QUESTIONS

Qu'est ce que le CROSS ORIGIN ? En quoi cela est-il dangereux ?

Comment REACTJS fait-il pour afficher rapidement les modifications sur les composants ?

Quelle est la fonction essentielle de REACTJS ?

Quelle est la fonction essentielle de FLUX ?

Qu'est ce que REDUX ?

Qu'est ce que JMS ? Est-ce spécifique à Springboot ?

Quelles sont les différents modes de transmissions de JMS ?

Quel est le mode de transmission activé par défaut dans activeMq ?

Qu'est ce que activeMq ?

Quels avantages proposent un bus de communication vis-à-vis de requêtes http classiques ?

Comment faire pour partager des modules Maven avec un partenaire extérieur ?

Comment faire pour exporter un composant REACTJS ?

Quel est le pré-requis pour Springboot afin de pouvoir convertir automatiquement le message reçu dans un bus de communication en objet ?

Comment est réalisée la traduction des messages reçus (bus de communication ou request http) en objet Java ? Quelles sont les prérequis ? Est-ce toujours possible ?

Quelles sont les fonctionnalités des différentes annotations en Springboot ?:

- @EnableJpaRepositories
- @EntityScan
- @ComponentScan

REFERENCES

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

[Architecture]

- *** <https://www.oreilly.com/library/view/microservices-vs-service-oriented/9781491975657/>

[Entreprise Service Bus (ESB)]

- *** <https://searcharchitecture.techtarget.com/definition/Enterprise-Service-Bus-ESB> (5min)
- *** <https://www.oracle.com/technical-resources/articles/middleware/soa-ind-soa-esb.html> (10 min)
- ** https://fr.wikipedia.org/wiki/Enterprise_service_bus

[ActiveMq]

- *** <https://activemq.apache.org/getting-started>

[JMS and Springboot]

- *** <https://grokonez.com/java-integration/spring-jms-activemq-send-java-object-messages-activemq-server-specially-bi-directional-relationship-java-objects>
- ** <https://www.devglan.com/spring-boot/spring-boot-jms-activemq-example>
- ** <https://spring.io/guides/gs/messaging-jms/>
-

[Json to Java Object]

- ** <https://www.mkyong.com/java/jackson-2-convert-java-object-to-from-json/>

[React]

- *** <https://fr.reactjs.org/tutorial/tutorial.html> (fondamentaux)
- ** <https://fr.reactjs.org/tutorial/tutorial.html>

ELEMENTS DONNES

- IDE configuré
- 1 sketch
- Une application Monolithique
 - Repo : <https://gitlab.com/js-asi2/asi2-backendmarket-monolithic-student>
 - Server : <https://asi2-backend-market.herokuapp.com/>
- 1 exemple simple de création d'utilisateur REACT + Redux + Springboot
- Visuel Html UI Semantic

Add Card

http://localhost/addUser.html

Q

UserForm

Name

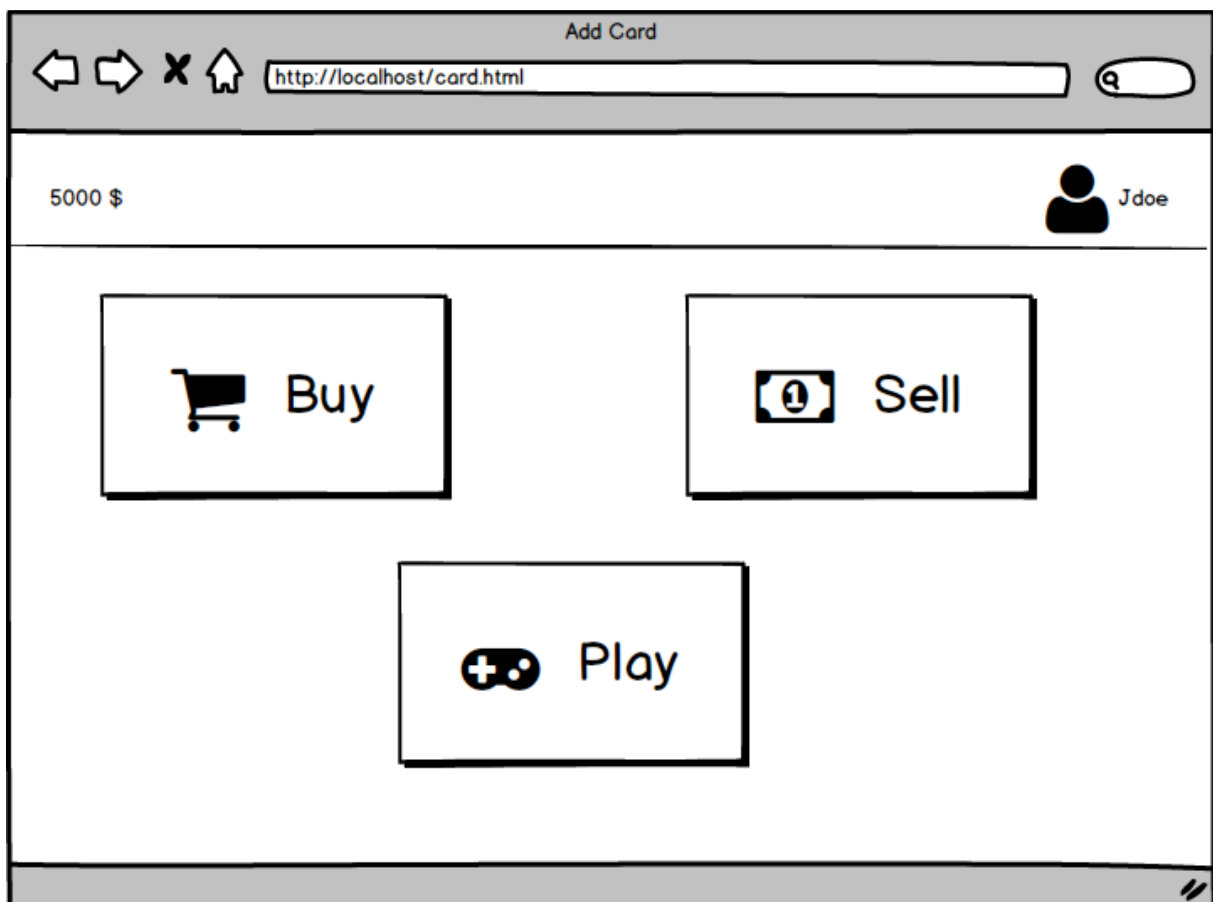
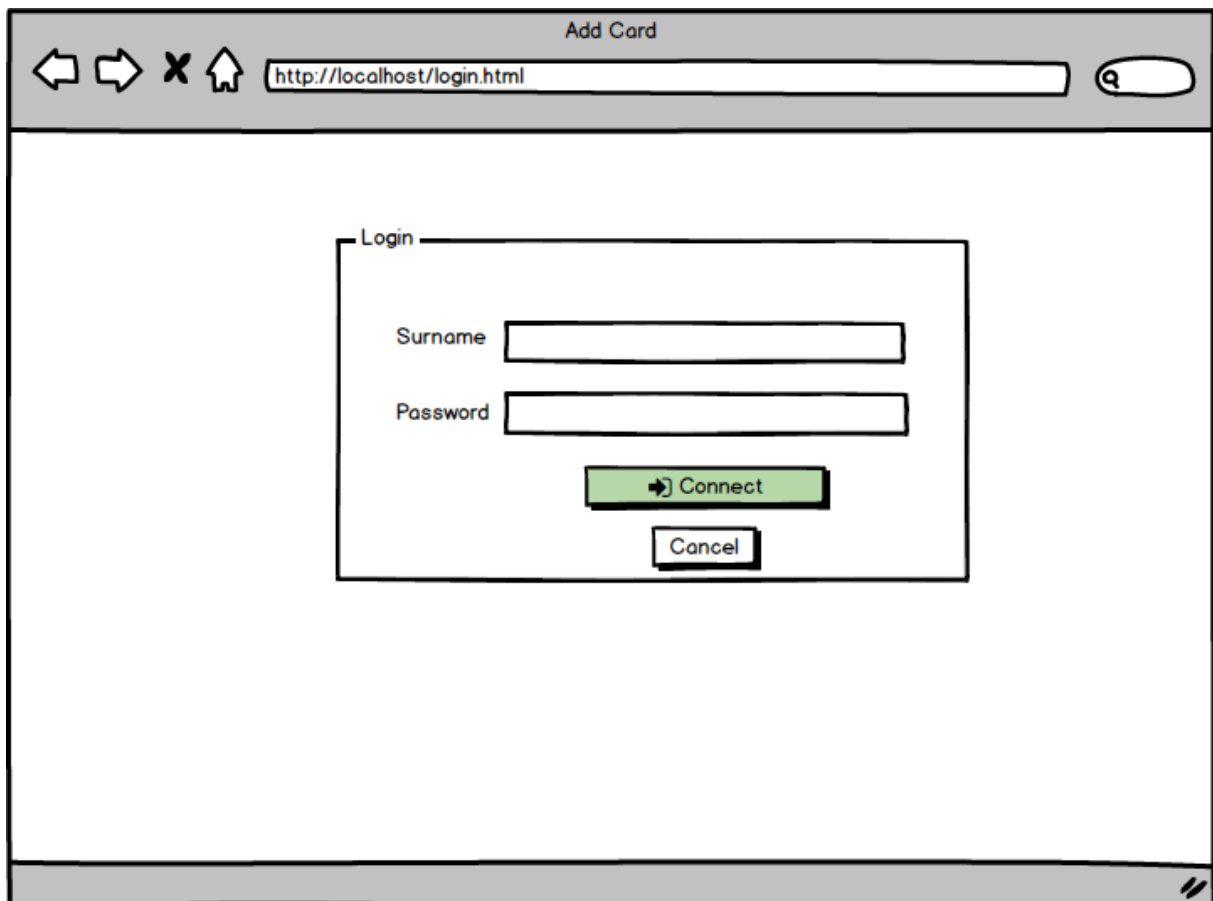
Surname

Password

Re-Password

Cancel

+ OK



Jdoe

← → ✕ 🏠


Add Card

http://localhost/card.html

🔍

5000 \$

SELL


Jdoe

My CARD

Name	Descript	Family	Affinity	Energy	HP	Price	
blabla	blabla blablabla	blabla	blabla	blabla	blabla	blabla	50\$
blabla	blabla blablabla	blabla	blabla	blabla	blabla	blabla	50\$
blabla	blabla blablabla	blabla	blabla	blabla	blabla	blabla	50\$
blabla	blabla blablabla	blabla	blabla	blabla	blabla	blabla	50\$
blabla	blabla blablabla	blabla	blabla	blabla	blabla	blabla	50\$
blabla	blabla blablabla	blabla	blabla	blabla	blabla	blabla	50\$
blabla	blabla blablabla	blabla	blabla	blabla	blabla	blabla	50\$
blabla	blabla blablabla	blabla	blabla	blabla	blabla	blabla	50\$

⚡ 20 Happy Tree Famil 50 🔒

Super John



=====

=====

SELL