

LaRuche

HLSE602 – Projet CMI Annuel

B. Rima O. Farajallah W. Soussi

L3 CMI Informatique

5 février 2018

Sommaire

Introduction

Problématique et Méthodologie de Résolution

- Problématique

- Solutions

- Méthodes agiles

Conception

- Outils de conception utilisés

- User Stories* : outil de conception agile

- Côté fournisseur

- Côté client

Outils d'implémentation

- Front-end*

- Back-end*

Conclusion

- Écosystème décentralisé et autonome

- Perspectives

Contexte du projet

Introduction

Projet CMI : Module d'un projet annuel pour l'année 2017–2018
dans le cadre du **CMI Informatique**.

Responsable CMI Informatique : Mme Anne-Elisabeth Baert.

Tuteur et Encadrant du projet : M. Eric Bourreau.

Lieux de travail : La **FDS** et le **LIRMM**.



Problématique

Problématique et Méthodologie de Résolution



Consommateurs :

Acheter des produits frais et minimiser les étapes de processing.

Producteurs :

Se libérer des centres d'achat et des intermédiaires de distribution.



Solution possible : La Ruche Qui Dit Oui

Problématique et Méthodologie de Résolution

Site Web

Une interface directe entre **consommateurs** et **fournisseurs**.

Definition (Ruche)

Un regroupement de plusieurs membres **consommateurs** et **fournisseurs** d'une région, guidé par un **responsable de ruche**.

Vision Centralisée

- l'ensemble des ruches obéit à une **Ruche-Mama**.
- les besoins de chaque ruche sont transmis à la **Ruche-Mama** via le responsable de ruche correspondant.
- la **Ruche-Mama** s'occupe de la gestion des ruches : création, réglementations internes, interactions, évolution et extensibilité des services,

Solution proposée : LaRuche

Problématique et Méthodologie de Résolution

Site Web

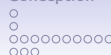
Une interface directe entre **consommateurs** et **fournisseurs**.

Definition (Ruche)

Un regroupement de plusieurs **fournisseurs** d'une région, **sans guide explicite** préfixé par le site.

Vision Décentralisée et Autonome

- l'ensemble des ruches ne répond à **aucune entité centralisée fédéral**.
- chaque ruche s'occupe de ses propres besoins et de leur gestion sans besoin d'un intermédiaire et d'une hiérarchie autoritaire à respecter.



Méthodologie de résolution : méthodes agiles

Problématique et Méthodologie de Résolution

Méthodes Agiles

Une approche de développement logiciel de plus en plus prépondérante basée sur une conception/développement itérative, orientée-test et orientée-client.

Pourquoi ?

- meilleure gestion des ressources
- sortie plus fréquente de versions fonctionnelles et testées du produit
- interaction plus fréquente avec les clients : adaptation et extensibilité du produit selon leurs besoins

Outils de conception utilisés

Conception

Diagrammes de cas d'usage

Des diagrammes dynamiques, souvent utilisés en **UML** pour décrire en haut niveau des fonctionnalités d'un système.

Modèle EA

Un modèle **conceptuel** utilisé pour décrire les entités du projet et les associations décrivant leurs relations et comportements.

Schéma de base de données

Un schéma en modèle **relationnel** traduit à partir du **modèle EA** et servant comme **support** lors de l'implémentation de la **base de données**.

mockup storyboard

Un document de haut niveau (peu de détails sur les fonctionnalités) pour schématiser l'utilisation d'un projet.

User Stories : outil de conception agile

Conception

User Stories

Des requis fournis par les clients, décrivant en langage naturel les fonctionnalités qu'ils souhaitent avoir dans le produit développé.

Intitulé

*En tant que <rôle>, je souhaite <fonctionnalité>,
dans le but de <bénéfice>.*

Côté fournisseur

Conception

Besoins

1. Un profil d'utilisateur **fournisseur** : propriétés et fonctionnalités via des *user-stories*.
2. Une **structure de données** pour décrire le **regroupement des fournisseurs** et leurs **interactions** : ruche, opérateur cellule, voisins, ...

Intitulés des *user-stories* fournisseurs

Côté fournisseur

1. Définition et stockage de produits.
2. Offre de Paniers.
3. Rapports de suivi périodiques.
4. Politique de rupture des stocks.
5. Politique de partage intracellulaire dans une ruche.
6. Validation de commandes.
7. Attribution de factures.

Exemple d'une *user-story* fournisseur

Côté fournisseur

Définition et stockage de produits

En tant que **fournisseur**, je souhaite **définir** ma **sélection de produits** selon des **informations caractéristiques** à fournir dans des **formulaire**s,

dans le but de **maximiser** la transparence de mes produits pour gagner la fidélité de mes clients, tout en **gérant** (création, modification, ajout, suppression) ma sélection à travers le site.

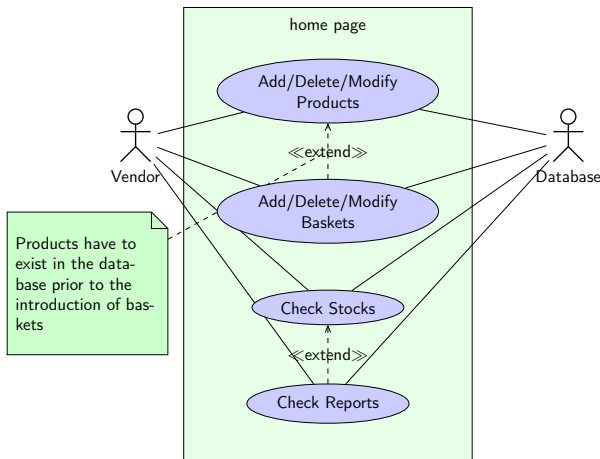
Profil fournisseur

Côté fournisseur

1. Un fournisseur offre des produits/paniers de produits divers.
2. Un fournisseur gère ses produits/paniers dans des stocks.
3. Un fournisseur suit l'évolution de ses ressources via des rapports de suivi périodiques.
4. Un fournisseur interagit avec d'autres fournisseurs pour créer des ruches et organiser des événements de collecte et uniquement avec des clients qui l'ont déjà contacté.
5. Un fournisseur valide les commandes de réservation des produits.
6. Un fournisseur règle les commandes physiquement, en premier temps, et puis via le site dans les versions ultérieures.
7. Un fournisseur imprime les factures, créées par le site lors de la réservation des produits par des clients, et les émet aux clients correspondants lors de la collecte de leurs produits.

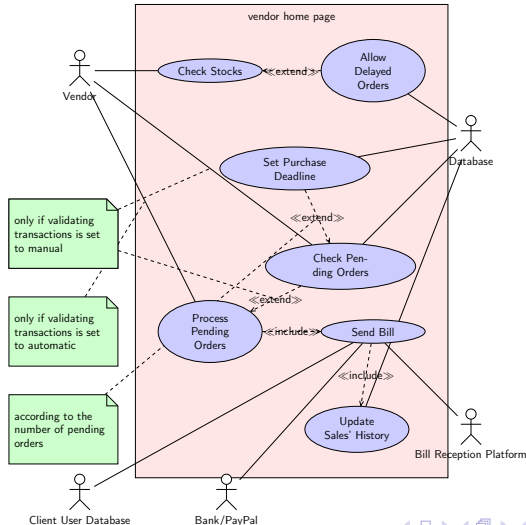
Exemple d'un diagramme de cas d'usage (version de base)

Côté fournisseur



Exemple d'un diagramme de cas d'usage (version ultérieure)

Côté fournisseur



Ruche : Structure de données proposée

Côté fournisseur

Définitions de base

V ensemble des fournisseurs.

C ensemble des clients.

π_v **opérateur** appliqué à $v \in V$ désignant une **cellule**, c.à.d. un **cercle** dont le centre est le point représentant les coordonnées du fournisseur v et dont le rayon est la distance maximale en **km** qu'il souhaite parcourir afin de se rendre à un lieu de collecte.

Ruche : Structure de données proposée(2)

Côté fournisseur

Ruche

Soit $v_1, v_2, \dots, v_n \in V^n$. Une **ruche** $R = (p, V_R)$ est composée de :

V_R ensemble de fournisseurs dont les cellules s'intersectent ;

p un point de collecte obtenu à partir d'une opération¹ sur la zone d'intersection des cellules correspondantes aux vendeurs de V .

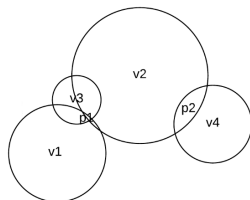
Autrement dit,

$$R = \{p, \{v_1, v_2, \dots, v_n \in V \mid \pi_{v_1} \cap \pi_{v_2} \cap \dots \cap \pi_{v_n} \neq \emptyset\}\}$$

1. le choix du point est relatif aux fournisseurs de la ruche, étant indéterministe en soi

Ruche : Structure de données proposée(3)

Côté fournisseur



Appartenance Simultanée

Un fournisseur peut appartenir à plusieurs ruches simultanément.

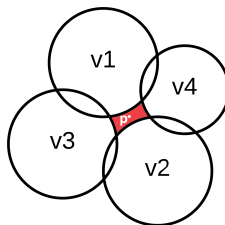
Fournisseurs Voisins

Soit $R = \{p, V_R\}$ une ruche. Deux fournisseurs v_1 et v_2 sont dits **voisins** $\iff v_1 \in V_R$ et $v_2 \in V_R$. On note $\text{Voisins}(v)$

l'ensemble des voisins d'un fournisseur v , contenant **tous les voisins de toutes les ruches auxquelles il appartient.**

Ruche : Structure de données proposée(4)

Côté fournisseur

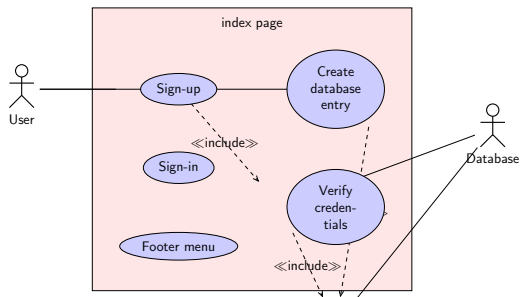


Voisinage Imposé

Soient v_1 et v_2 deux fournisseurs tels que $v_2 \notin \text{Voisins}(v_1)$. S'il existe des fournisseurs v_3 et v_4 tels que $v_3, v_4 \in \text{Voisins}(v_1) \cap \text{Voisins}(v_2)$ et $v_3 \notin \text{Voisins}(v_4)$, alors il existe une ruche plus optimale contenant v_1, v_2, v_3 et v_4 que les ruches séparées les contenant.

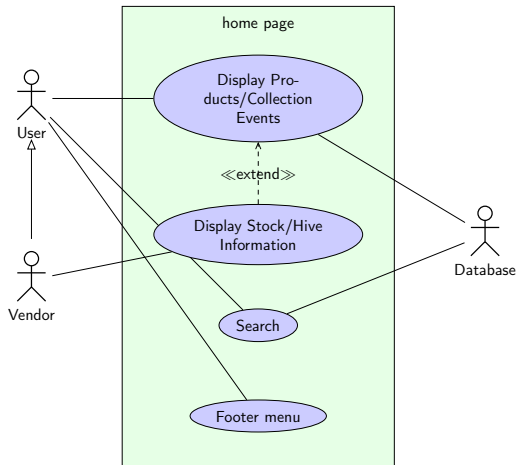
Côté client

Conception



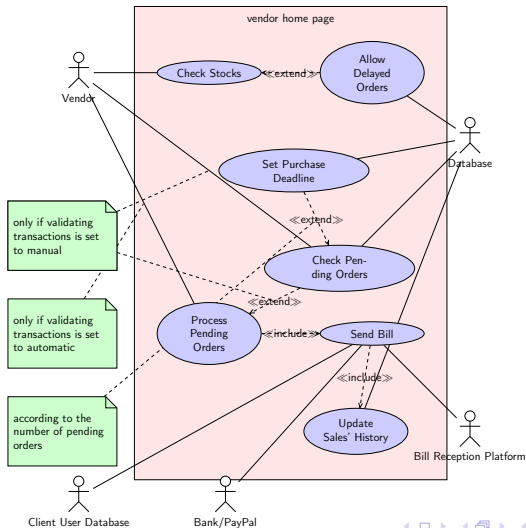
Côté client

Conception



Côté client

Conception





Front-end

Outils d'implémentation

React.js

React est une librairie JavaScript, créée par Facebook, utilisée uniquement pour le côté “vue” dans le paradigme MVC.

Front-end

Outils d'implémentation

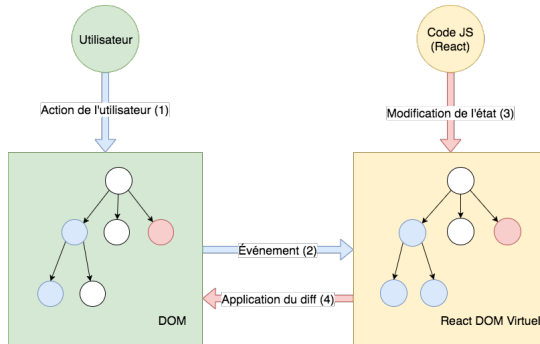


Figure – Schéma explicatif du principe de réconciliation de la librairie React



Front-end

Outils d'implémentation

JSX

JSX est une extension JavaScript dont l'usage est très recommandé lors du développement d'une application React.



```
1 const laRucheElement = <h1>Hello, world!</h1>;  
2  
3 }
```

Back-end

Outils d'implémentation

Language de développement : Serveur codé en PHP et le framework Symfony pour un système modulaire et basé sur le design pattern MVC.

Base de données : Base de données relationnelle mise en place via MySQL, interaction avec l'ORM Doctrine.



Écosystème décentralisé et autonome

Conclusion

Autonomie

Les fournisseurs sont **autonomes** : la formation des ruches et l'organisation des évènements de collecte.

Extensibilité

Le système est versatile et extensible : extension des ruches, augmentation de leur nombre, ajout de fonctionnalités supplémentaires,



Perspectives

Conclusion

- Changement du nom du projet
- Récolte de feedback de la part d'utilisateurs potentiels
- Optimisation de la logistique
- Implémentation de quelques fonctionnalités (inscription, paiement...)
- Internationalisation