CourtCircuit

HLSE602 – Projet CMI Annuel

B. Rima O. Farajallah W. Soussi

L3 CMI Informatique

16 mai 2018

Sommaire

Contexte du projet

Introduction

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

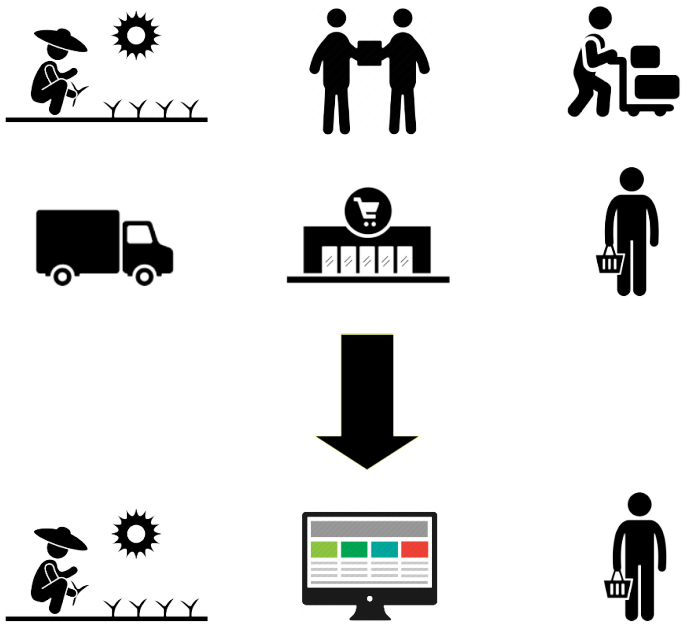
Responsable CMI Informatique : Mme Anne-Elisabeth Baert

Encadrant du projet : M. Eric Bourreau

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

Rappels

Problématique



Consommateurs :

Acheter des produits frais et minimiser les étapes de processing.

Producteurs :

Maîtriser le prix de vente et les débouchés de leurs productions en se libérant des intermédiaires de distribution.

Rappels

Solution proposée : `CourtCircuit`

Site web *e-commerce*

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

Ruche

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

Vision Décentralisée et Autonome

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

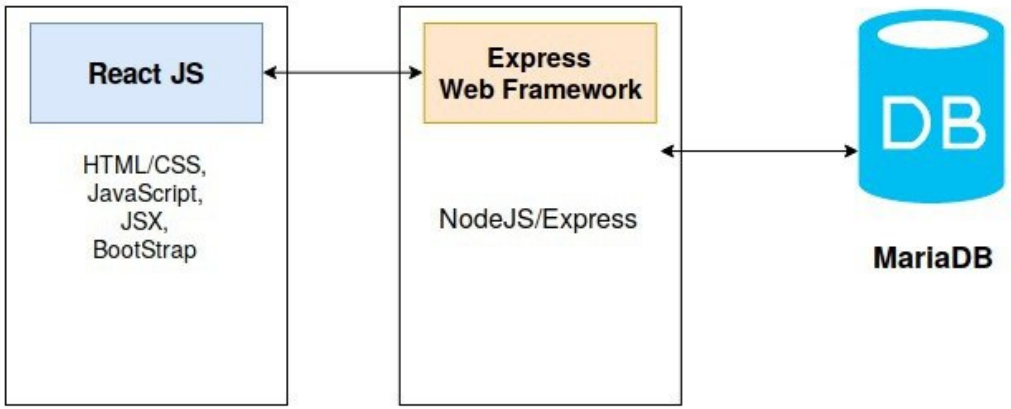
Rappels

Outils de conception

1. *User Stories* (outil de conception agile)
2. Diagrammes de cas d'usage
3. Modèle EA
4. Schéma de base de données
5. *Storyboard*

Outils d'implémentation

Implémentation



Front-end : React.js, JSX, Bootstrap

Back-end : Node.js, Express.js

Base de données : MariaDB

Application web monopage (SPA)

Implémentation

schéma d'une SPA

Application web monopage (SPA)

Implémentation

pros et cons d'une SPA

React

Front-end

Bootstrap et Font-Awesome

Front-end

Charte Graphique

Front-end

Node.js (Introduction)

Back-end

- environnement d'exécution **JavaScript** côté serveur utilisant le moteur **JavaScript V8** de *Google Chrome*
- libre et gratuit
- modélisation événementielle, monothread et non-bloquante
- architecture modulaire
- gestionnaire de paquets **NPM** (*Node Package Manager*) $\rightarrow$ facilité d'usage et d'extensibilité

Node.js (Raisons du choix)

Back-end

- écrire du code **JavaScript** du côté serveur → un seul langage pour les côtés client et serveur
- modélisation événementielle, monothread et non-bloquante → performance fluide et gestion efficace d'un ensemble important de données
- ensemble important de modules utilitaires facilement téléchargeable via **NPM**

Node.js (Utilisation)

Back-end

- création d'une **API** factorisée, non redondante et facilement lisible (*Client*, *Utilisateur*, *Produit*, ...) permettant d'interfacer avec la base de données.
- héberger **Express**.

Express

Back-end

MariaDB

Back-end

Ce qui marche et ce qui ne marche pas

Résultats

Difficultés survenues

Résultats

- Nouveaux concepts et outils d'implémentation nécessitant un temps d'apprentissage considérable
- Temps dédié à l'implémentation insuffisant
- Problèmes liés au serveur d'hébergement

Apports personnels du projet

Conclusion

- Apports personnels = difficultés survenues
- Apprentissage d'outils d'implémentation récents et en pleine évolution
- Appréciation plus profonde du langage JavaScript

Perspectives

Conclusion

- Continuation du projet au niveau personnel
- Récolte de *feedback* des utilisateurs potentiels
- Optimisation de la logistique
- Implémentation de fonctionnalités supplémentaires (paiement en ligne, commandes retardées, portefeuille virtuel, . . .)
- Internationalisation