CourtCircuit

HLSE602 – Projet CMI Annuel

B. Rima O. Farajallah W. Soussi

L3 CMI Informatique

17 mai 2018

Sommaire

Introduction
Rappels
- Problématique
- Solution proposée : `CourtCircuit`
- Outils de conception

Implémentation
- Application web monopage
- Outils d'implémentation
- *Front-end*
- *Back-end*

Résultats
- Bilan
- Difficultés survenues

Conclusion
- Apports personnels du projet
- Perspectives

Contexte du projet

Introduction

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

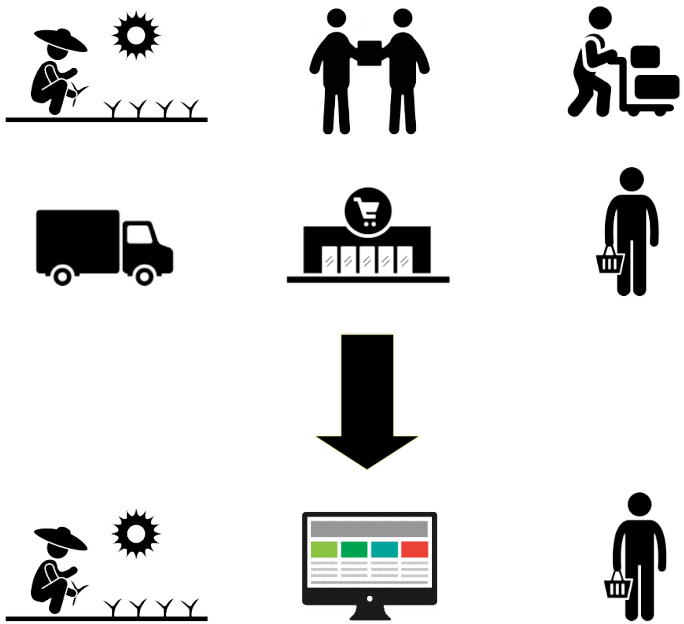
Responsable CMI Informatique : Mme Anne-Elisabeth Baert

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.

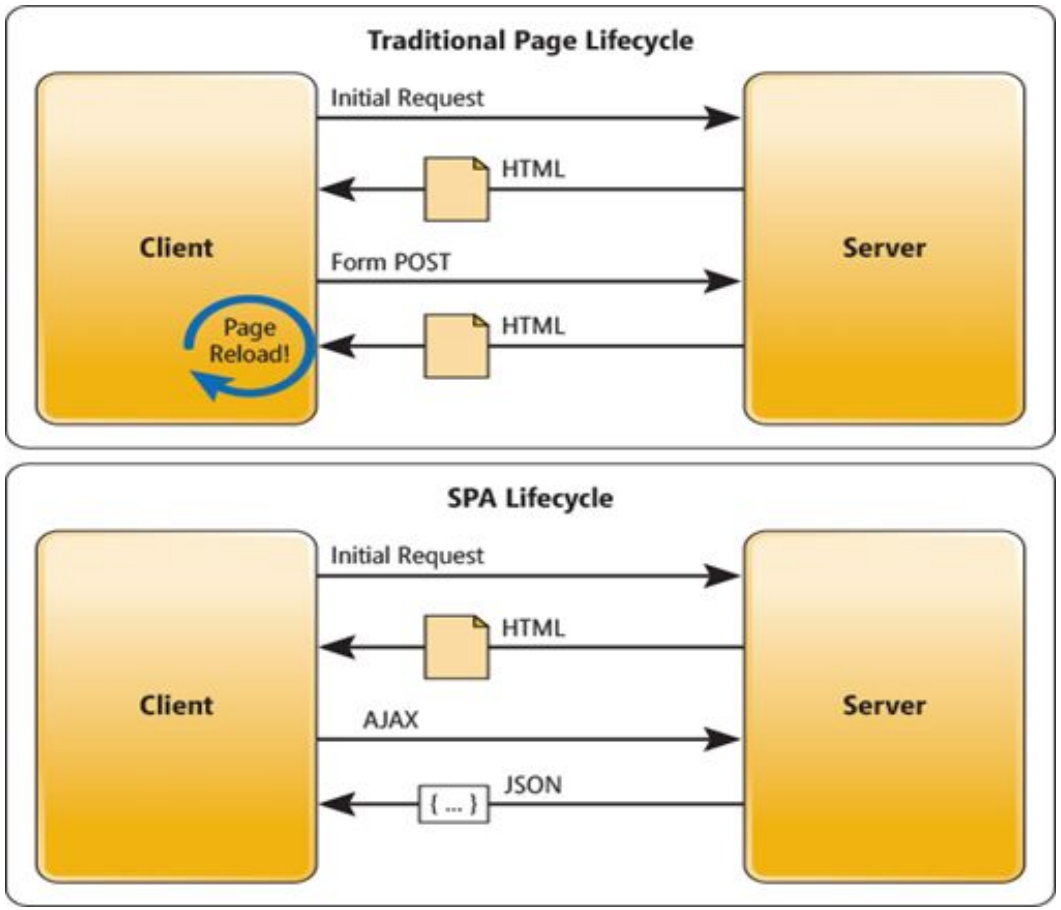
Rappels

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*

Application web monopage (Introduction)

Implémentation



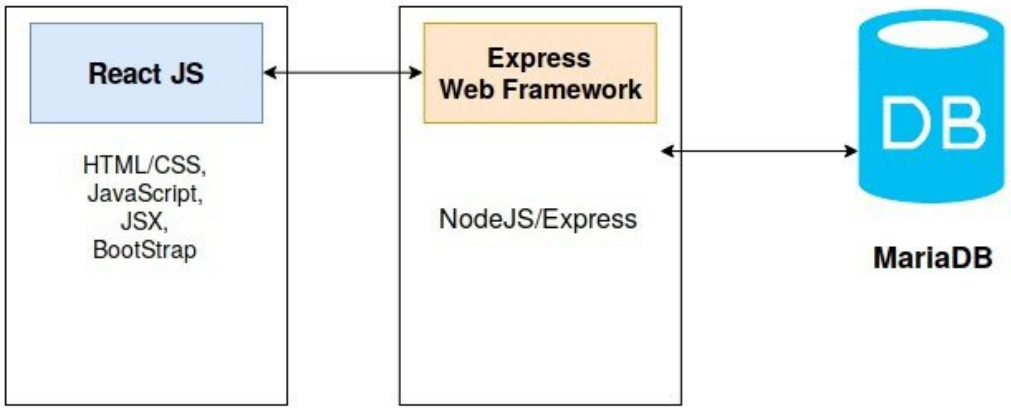
Application web monopage (Avantages)

Implémentation

- chargement plus rapide de la page.
- séparation des tâches de contrôle et de vue $\rightarrow$ meilleure implémentation des contraintes du *design pattern* **MVC**.
- facilité de déploiement.

Outils d'implémentation

Implémentation



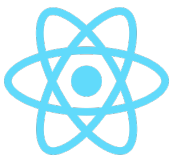
Front-end : React.js, JSX, Bootstrap

Back-end : Node.js, Express.js

Base de données : MariaDB

React (Introduction)

Front-end



- Une bibliothèque JavaScript libre développée par Facebook en 2013.
- Elle permet de créer des interfaces hautement personnalisables et interactives pour des applications monopages.

React (Raisons du choix)

Front-end

- la performance.
- la modularité.

React (Raisons du choix)

Performance

Performance :

Les applications développées en React peuvent gérer des mises à jour complexes tout en restant rapides et réactives.

React (Raisons du choix)

Performance

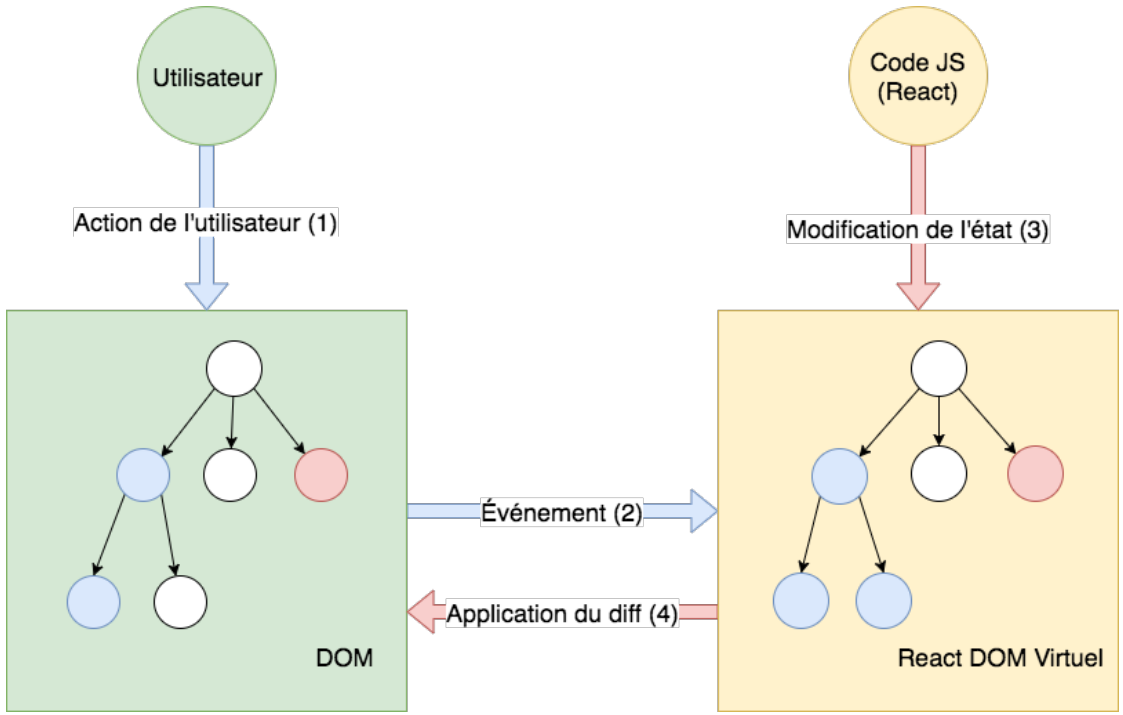


Figure – Schéma représentant le principe de réconciliation de la librairie `React`

React (Raisons du choix)

Modularité

Modularité

Une application React peut être représentée par un ensemble de composants imbriqués les uns dans les autres.

- la réutilisabilité des différents composants
- meilleure lisibilité du code
- facilité de maintenance

React (Organisation choisie)

Modularité

- **Components :** des éléments indépendants qui s'occupent uniquement de présenter les informations au DOM.
- **Containers :** des éléments de haut niveau responsables de gérer les changements de state en correspondant chaque état avec un comportement particulier.

React (Organisation choisie)

Exemple de la page d'accueil

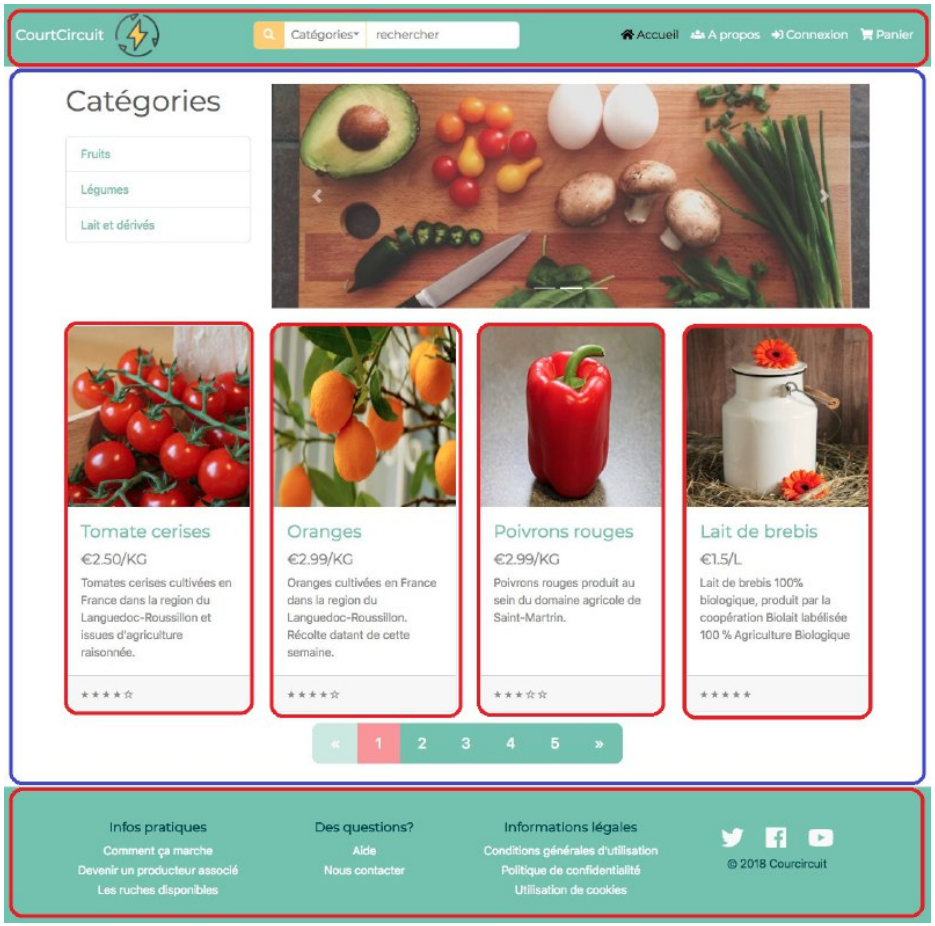


Figure – Schéma représentant le principe de réconciliation de la librairie React

React

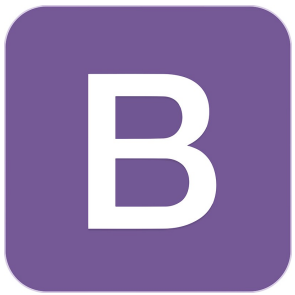
JSX

JSX est une extension syntaxique du JavaScript.

- Réduire la verbosité
- Lecture plus souple du code

Bootstrap et Font-Awesome

Front-end



- réalisation de pages **responsives**.
- personnalisation de la mise en forme des pages à travers des **classes CSS** prédéfinies.
- icônes modifiables dynamiquement.

Figure – Entête du site faite avec la classe *Navbar* de **Bootstrap** avec des icônes **Font-awesome**

Node.js (Introduction)

Back-end

- environnement d'exécution **JavaScript** côté serveur utilisant le moteur **JavaScript V8** de *Google Chrome*.
- gratuit et *open-source*.
- 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*, *Fournisseur*, *Produit*, ...) permettant d'interfacer avec la base de données.
- héberger **Express**.

Express (Introduction)

Back-end

express

- framework web minimaliste pour **Node.js**.
- gratuit et *open-source*.
- utilisation de *middleware*.
- gestion des routes **REST** (*Representational State Transfer*) et des formulaires en s'appuyant sur des concepts du *design pattern* **MVC**.
- moteurs de templates (*EJS (Embedded JavaScript)*, *Pug*, *Handlebars*, ...).

Express (Raisons du choix et utilisation)

Back-end

- framework web *de-facto* pour **Node.js**.
- réduire la verbosité du code **Node.js** natif pour la création du serveur **HTTP**.
- utilisation de *middleware* pour le traitement des requêtes clients.
- gestion des routes **REST** pour les opérations **CRUD**.

MariaDB (Introduction)

Back-end

- **SGBD** relationnel.
- gratuit et *open-source*.
- assure l'interopérabilité avec **MySQL**.
- introduit par les créateurs de **MySQL** suite à l'achat de ce dernier par **Oracle**.

MariaDB (Raisons du choix et utilisation)

Back-end

- fork communautaire de **MySQL** mis à jour plus souvent que ce dernier.
- **modèle EA** déjà traduit en **modèle relationnel** depuis la phase de conception du projet → mise en œuvre directe.
- mise en place d'un système de validation des données dans le cadre d'une politique de sécurité en trois couches (*client*, *serveur*, *base de données*) → assurer l'intégrité des données stockées.
- le serveur de la faculté qui nous a été attribué pour le déploiement hébérge bien **MySQL** → possibilité d'utiliser **MariaDB**.

Bilan

Résultats

Base de données conçue et testée	✓
Composants de base de l'interface graphique	✓
Connexion du client au serveur	✓
Connexion du serveur à la base de données	✓
Gestion des routes entre le serveur et le client	✗
Test du comportement dynamique des composants	✗
Déploiement en ligne	✗

Table – Bilan des résultats

Difficultés survenues

Résultats

- nouveaux concepts et outils d'implémentation nécessitant un temps d'apprentissage considérable.
- temps d'apprentissage considérable $\rightarrow$ adoption d'une méthodologie agile de développement de plus en plus compliqué.
- temps dédié à l'implémentation insuffisant.
- problèmes liés au serveur d'hébergement.

Apports personnels du projet

Conclusion

- Apports personnels = difficultés survenues.
- Apprentissage d'outils *front-end* et *back-end* 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.
- Mise en place et optimisation de la logistique.
- Implémentation de fonctionnalités supplémentaires (paiement en ligne, commandes retardées, portefeuille virtuel, . . .).
- Internationalisation.