

POMMIER Romain



MEMOIRE DE STAGE

Mémoire Professionnel:
Concepteur/Développeur
D'application Web/Mobile

Remerciements

Je souhaiterais remercier les personnes m'ayant accompagné tout au long de ma formation et de ce projet sans qui il n'aurait pas été réalisable.

Je remercie nos formateurs pour leur disponibilité et de nous avoir guidé tout au long de la formation.

Je remercie également les différents intervenants tels que Mr Anthony, Mr SCHIEX Benoit et Mr Philippe D qui ont partagé leurs connaissances et leur passion avec nous.

Je souhaite également exprimer ma reconnaissance envers mes collègues de formation pour leur esprit de partage et d'entraide primordial dans ce domaine ainsi que les personnes travaillant au sein de l'entreprise First Seller durant ma période de stage, plus particulièrement Mr Bonavia David, mon chef de stage.

Table des Matières

Remerciements	2
Table des Matières	3
I – PRESENTATION	3
1 – Abstract	3
2 – Présentation	4
3 – Compétences couvertes par le projet.....	6
II – CAHIER DES CHARGES	7
1 – Présentation du projet	7
2 – Cadre technique de l’application	8
3 – Système d’administration	8
III – ANALYSE FONCTIONNELLE.....	9
1 – Fonctionnalités Utilisateur	9
2 –Analyse des besoins	10
3 – Diagramme de cas d’utilisation	10
4 – Diagramme de classe.....	13
5 – Diagramme de séquence	14
IV – CONCEPTION.....	17
1 – Modèle conceptuel de données (MCD)	17
2 – Modèle logique de données (MLD)	18
3 – Arborescence du site	19
4 – Maquettage	20
V – SPECIFICATIONS TECHNIQUES	27
VI – REALISATION	29
1 – Architecture	29
2 – Modèle en couche MVC	30
3 – Fonctionnement du router.	32
4 – Fonctionnalité d’ajout, et visualisation d’une commande.....	33
3 – Fonctionnalité d’ajout, d’un nouvel utilisateur.	41
VII – CONCLUSION	44

I – PRESENTATION

1 – Abstract

My client is an E-commerce business called FirstSeller based in Castelginest. They sell products on several E-commerce platforms like Amazon, Cdiscount and Fnac.

Every day an employee is in charge of creating a spreadsheet using Excel with products ordered on amazon, and then send it to the corresponding partners.

So my project consisted in saving time to them on the execution of orders. To do this I was responsible for making a WEB software allowing the employees to record different products categorized by partner in a database as well as to add an order using a form in order to group together the client information and the Amazon platform.

Then the user will have the possibility to consult Excel, download or send directly the orders of the different partners in an Excel format.

This project will be designed in PHP POO back-end , I also used the ajax protocol to make requests to optimize the user's comfort of use as well as the JQUERY library to facilitate the use of ajax and javascript for the handling of the DOM.

I had to use the same color code as the other applications previously developed for this company. I also set up the MVC layer system for efficient and organized development.

I am happy to know the software is working well as the client uses it every day. We are still in touch for regular feedbacks.

2 – Présentation

J'ai effectué mon stage de 10 semaines chez FirstSeller dans le cadre de ma formation Développeur d'Application numérique Web/Mobile.

Fondée en 2018, cette entreprise se situe à Castelginest. First Seller est une entreprise familiale de référence d'achat en ligne.

A la demande de mon client David BONOVIÀ, et dans la finalité d'une application WEB demandée, permettant principalement la génération d'un Excel à partir des commandes réalisées par les clients sur le site web Amazon.

J'ai donc réalisé une interface permettant à l'utilisateur d'ajouter des produits, ainsi qu'un formulaire pour regrouper les informations des commandes d'un client.

La seule condition graphique imposée lors de l'élaboration de cette application était de conserver la barre de navigation dans les mêmes couleurs que l'application précédemment faite par un autre stagiaire ainsi que le design général des pages du site.

Ce fut l'opportunité pour moi d'approfondir mes connaissances en POO et en PHP tout en respectant les exigences du modèle MVC (Model View Controller).

Mon objectif étant d'approfondir mes connaissances et développer une application sur le langage de programmation PHP qui mobilise une très forte communauté.

Ce projet m'a permis d'apprendre à penser et concevoir une architecture MVC d'un projet de développement d'une application, de mettre en place cette architecture et ainsi développer mes compétences personnelles pour l'élaboration de cette application.

3 – Compétences couvertes par le projet

« Développer une application client-serveur »

- ☐ Maquetter une application.
- ☐ Concevoir une base de données.
- ☐ Mettre en place une base de données.
- ☐ Développer une interface utilisateur.
- ☐ Développer des composants d'accès aux données.

« Développer une application web »

- ☐ Développer des pages web en lien avec une base de données.
- ☐ Mettre en œuvre une solution de gestion de contenu ou e-commerce.
- ☐ Développer une application simple de mobilité numérique.
- ☐ Utiliser l'anglais dans son activité professionnelle en informatique.

II – CAHIER DES CHARGES

1 – Présentation du projet

L'application s'adresse à l'entreprise FirstSeller axée sur les besoins de MR Bonavia David et de ses employés.

Mr Bonavia David a exprimé le besoin de réduire le temps passé à la réalisation de ses tableaux Excel de commande réalisés tous les jours à la main pour deux partenaires, en particulier (Whynote, Emotional).

Effectivement ces tableaux Excel peuvent être compliqués à remplir. Certains contiennent du Json et demandent plusieurs connexions à des sites différents.

L'application développée sera une solution afin de gagner en temps sur l'élaboration de l'Excel, ainsi que la possibilité de consulter un historique de commandes journalier.

L'application serait accessible à n'importe qui présent dans l'entreprise possédant un accès utilisateur sur la plateforme.

Elle sera mise en ligne en français et ainsi que sur un site responsive afin de pouvoir consulter plus librement les données.

La seule condition graphique imposée lors de l'élaboration de cette application est de conserver la barre de navigation dans les mêmes couleurs que l'application précédemment faite par un autre stagiaire ainsi que le design général des pages du site.

L'application doit permettre :

- L'enregistrement d'un utilisateur.
- L'enregistrer, Supprimer ou Modifier des produits en fonction des partenaires associés à FirstSeller.
- La création ou Supprimer des commandes en fonction des informations de commande enregistrées par le client sur Amazon.
- La création d'un tableau Excel à partir des commandes et d'une date sélectionnée.
- Consulter, télécharger un historique des commandes.
- Envoie direct du tableau Excel aux partenaires concernés.

2 – Cadre technique de l'application



La partie front-end utilise les technologies :

- HTML
- CSS
- JAVASCRIPT / JQUERY
- AJAX
- Bootstrap
- PHP

La partie back-end sera développée en PHP7.

Pour le stockage des données j'utilise le langage SQL.

Le système de requêtes sera entièrement réalisé avec le protocole Ajax, afin d'optimiser le confort de l'utilisateur.

L'application est destinée à la France donc il n'y a pas de stratégie d'internationalisation à prévoir.

Elle sera également réalisée en système de couches du modèles MVC (Model View Controler) afin de permettre une meilleur organisation et maintenance ainsi qu'un développement efficace du produit.

3 – Système d'administration

Le système d'administration sera simple : chaque utilisateur sera administrateur. Ces derniers pourront accéder au service de base de l'application tel que l'ajout d'un produit ou d'une commande. Chaque nouvel inscrit aura le rôle d'administrateur obligatoirement. Chaque utilisateur devra se connecter via un formulaire de connexion avant de pouvoir accéder aux services de l'application.

III – ANALYSE FONCTIONNELLE

1 – Fonctionnalités Utilisateur

L'utilisateur pourra :

▸ Se connecter : à partir d'un formulaire sur la page d'accueil, il s'identifie et sera redirigé vers les fonctionnalités de l'application.

À partir du moment où un utilisateur est connecté, toutes les fonctionnalités se feront à partir du site. Il pourra :

▸ Se déconnecter. Si la déconnexion est réussie il est alors redirigé vers le formulaire de connexion.

▸ Ajouter, supprimer ou modifier un produit.

▸ Consulter la liste des produits enregistrés.

▸ Faire une recherche pour trouver un produit ou organiser son tableau récapitulatif.

▸ Réaliser ou supprimer une commande.

▸ Consulter la liste des commandes enregistrés.

▸ Faire une recherche pour trouver une commande ou organiser son historique de commandes sous forme d'un tableau récapitulatif.

▸ Une fois connecté l'utilisateur pourra créer un nouveau compte à partir de la page d'accueil.

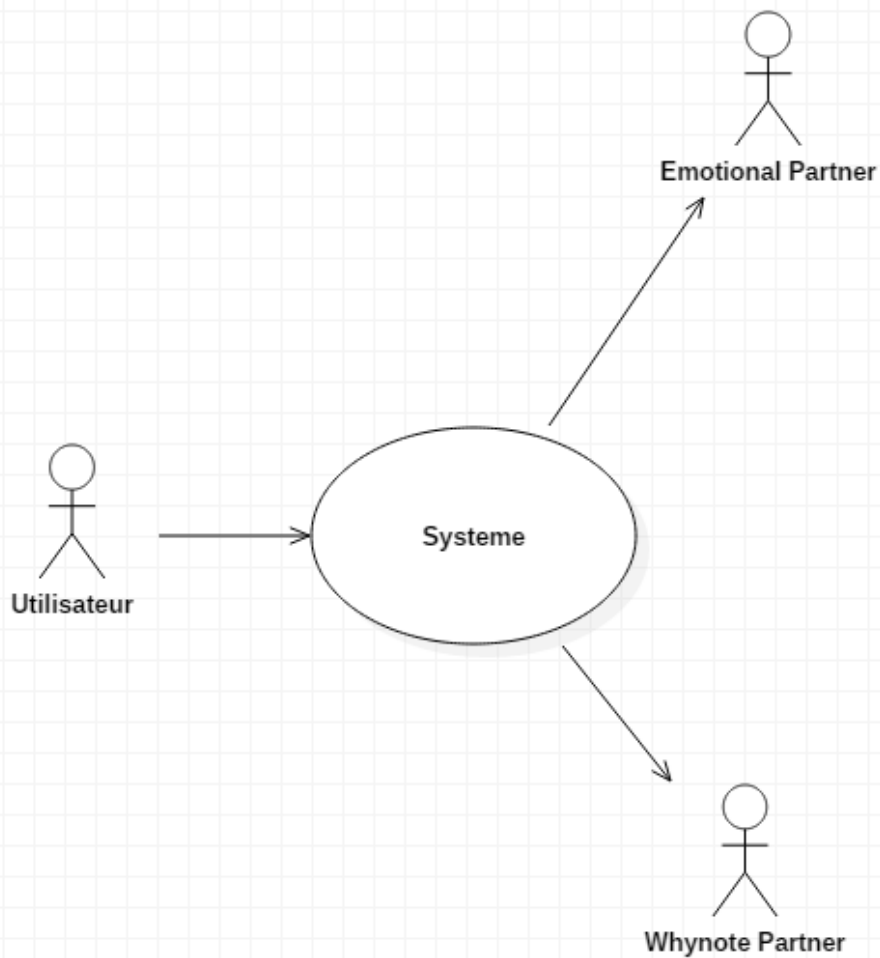
▸ Télécharger des commandes journalières sous format Excel.

▸ Envoyer des commandes journalières sous format Excel aux partenaires concernés.

▸ Ajouter un nouvel utilisateur.

2 –Analyse des besoins

L'application sera composée d'un seul acteur principal et de deux partenaires principaux
Nous pouvons alors déterminer le diagramme de contexte suivant :

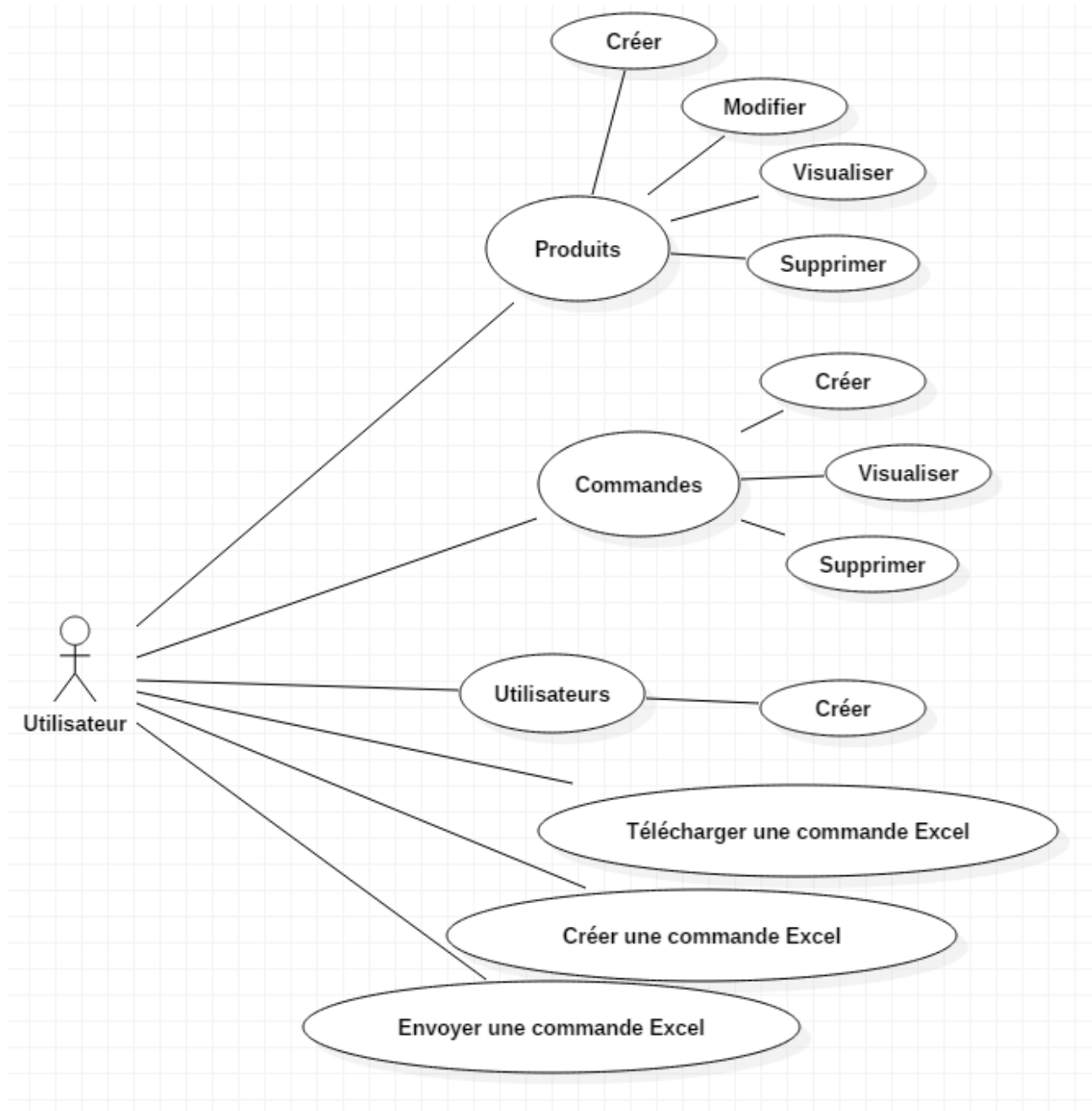


3 – Diagramme de cas d'utilisation

A. Les acteurs :

Les différents acteurs qui vont utiliser l'application seront le logicien et les 3 responsables. Les 4 utilisateurs auront tous le même niveau de rôle, Administrateur.

B. Le Diagramme :



C. Description fonctionnelle détaillée :

Fonctions : Ajouter /Modifier/Visualiser/Supprimer un produit	
Objectif	Permettre d'ajouter et de manipuler des produits enregistrés dans une base de données
Description	L'administrateur pourra créer, modifier, supprimer ou visualiser un produit, permettant par la suite de lier ce produit a une commande client. La personne doit renseigner plusieurs paramètres du produit suivant le partenaire concerné (ex: SKU, Couleur, taille, EAN)
Contraintes	Seuls les administrateurs peuvent utiliser cette fonctionnalité

Fonctions : Ajouter /Visualiser/Supprimer une commande	
Objectif	Permettre d'ajouter et de manipuler des commandes enregistrées dans une base de données
Description	L'administrateur pourra créer, supprimer ou visualiser une commande, permettant par la suite de créer un tableau Excel de commandes journalières. La personne doit renseigner plusieurs paramètres de la commande suivant le partenaire concerné (ex: Adresse, voie postal, NOM, Code postal)
Contraintes	Seuls les administrateurs peuvent utiliser cette fonctionnalité

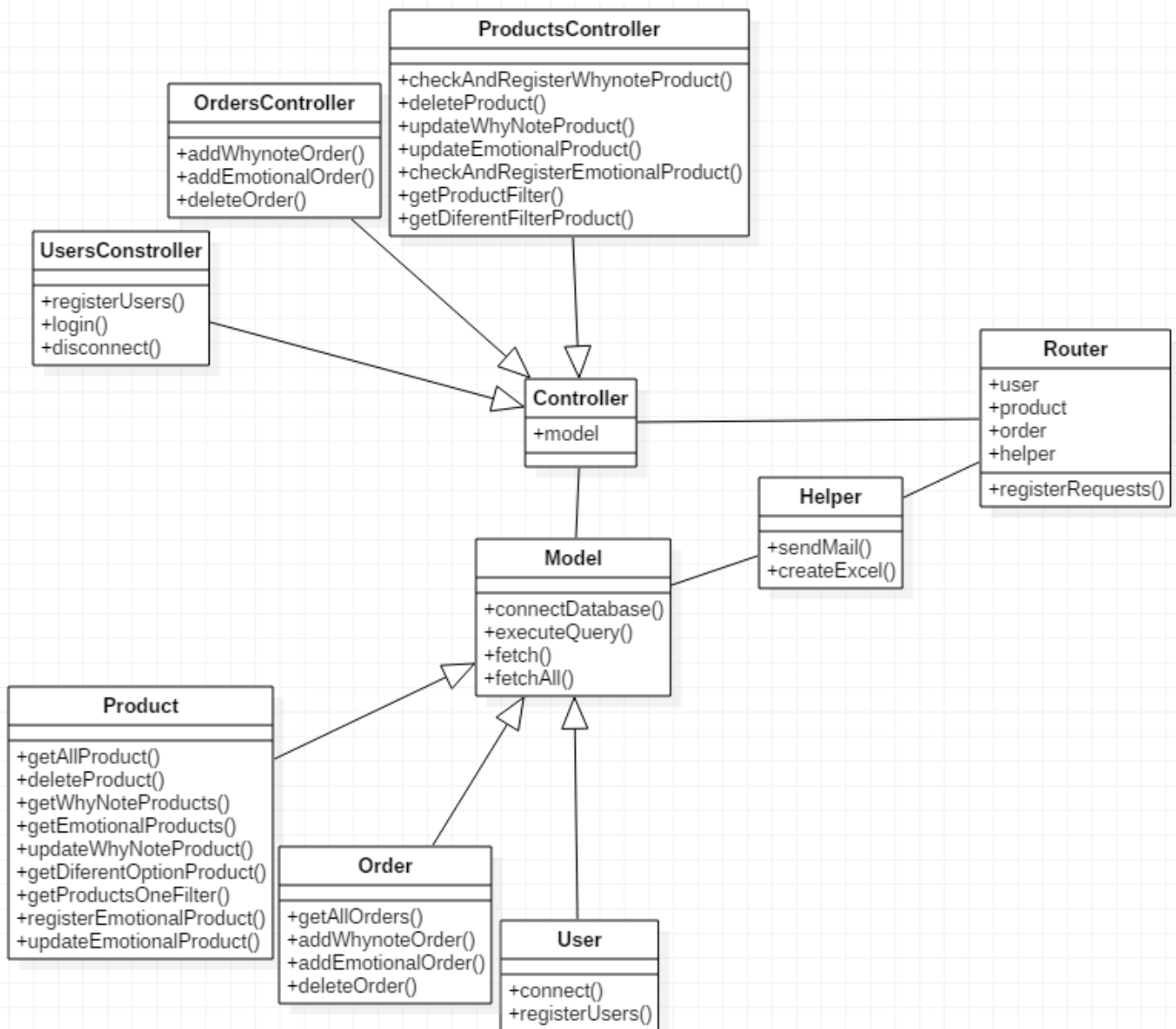
Fonctions : Créer/ Télécharger / Envoyer une commande au format Excel	
Objectif	Permettre de créer et de manipuler des tableaux Excel à partir de données stockées dans la base de données
Description	L'administrateur pourra créer, télécharger ou envoyer un tableau Excel. La personne doit renseigner la date ainsi que le nom du partenaire concerné par la demande
Contraintes	Seuls les administrateurs peuvent utiliser cette fonctionnalité

4 – Diagramme de classe

Voici le diagramme de classe théorique de l'application :

J'ai choisi de réaliser des méthodes **d'insert** pour les différents partenaires afin de faciliter la réalisation et la visibilité des requêtes.

(Exemple : **addWhynoteOrder ()** / **addEmotionalOrder ()**).

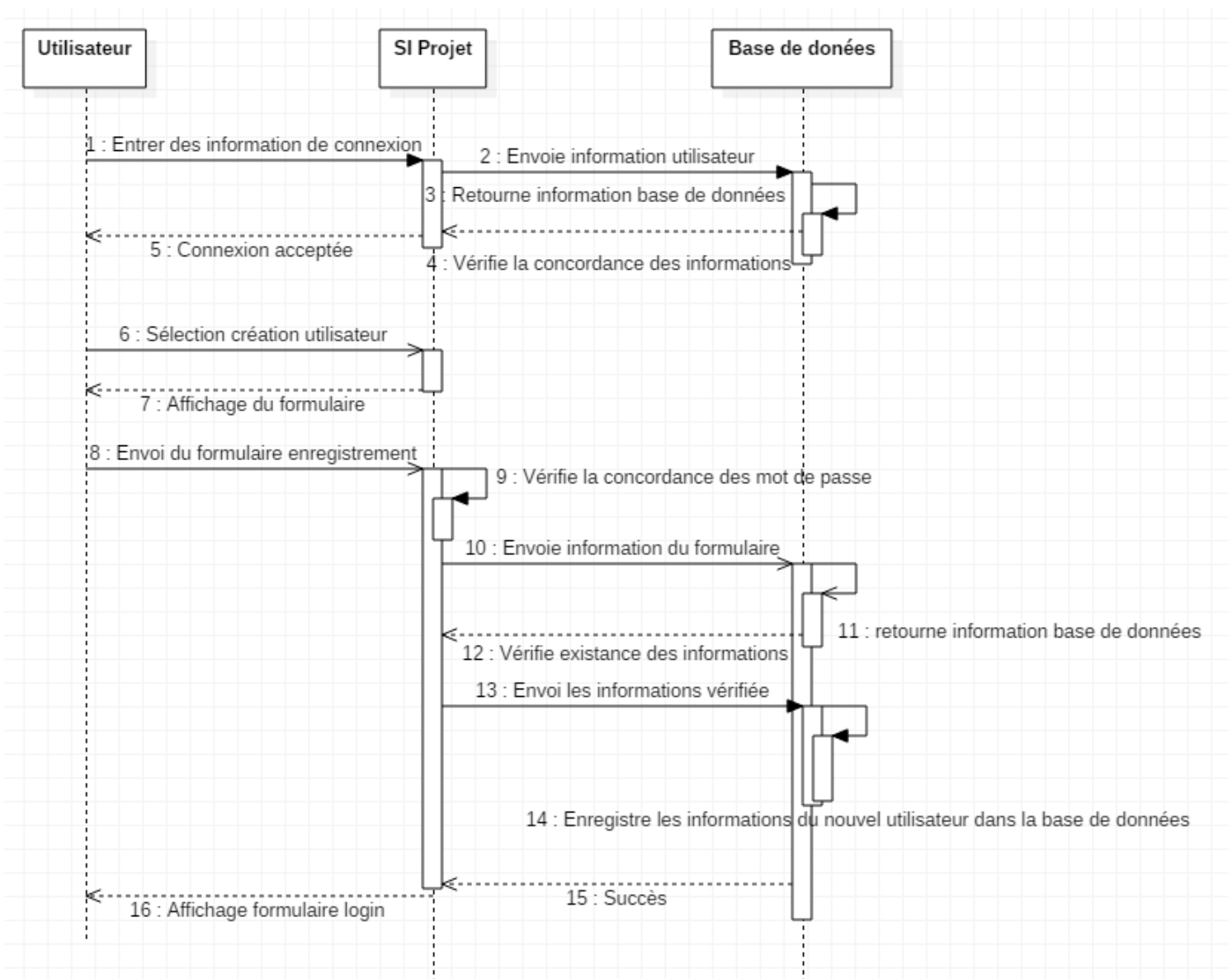


5 – Diagramme de séquence

Diagramme de séquence de l'enregistrement d'un nouvel utilisateur

L'enregistrement d'un nouvel utilisateur demande obligatoirement une connexion sur le site

Séquence nominale :



Scénario Alternatif : (4)

4.1 – Le SI indique à l'internaute que les informations saisies ne sont pas justes.

4.2 – L'internaute saisit à nouveau ses informations.

Scénario Alternatif : (9)

9.1 – le SI indique que les mots de passe ne correspondent pas.

9.2 – L'internaute saisit à nouveau ses mots de passe.

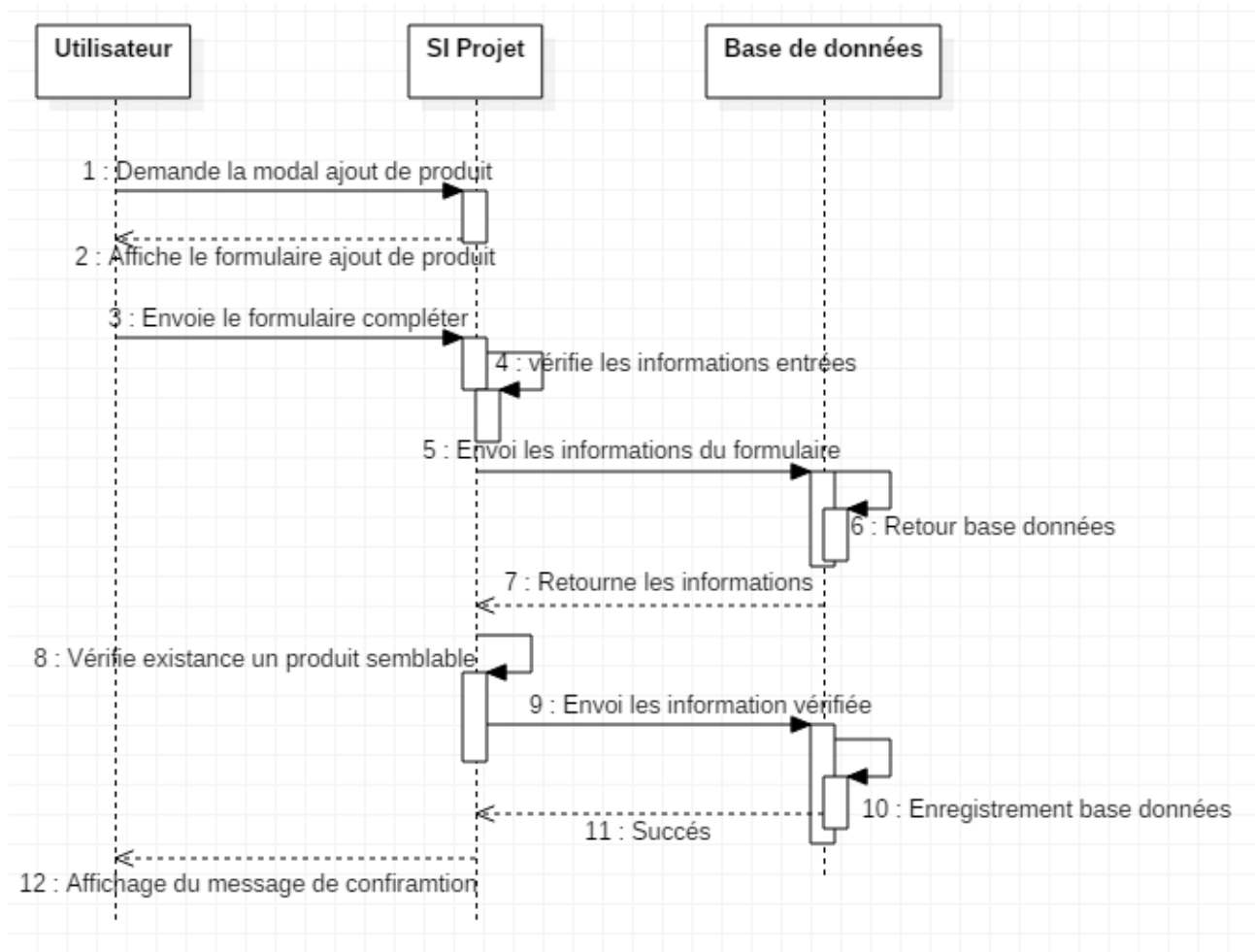
Scénario Alternatif : (12)

12.1 – La Base de données retourne un nom d'utilisateur déjà connu.

12.2 – L'internaute saisit une nouvelle fois le formulaire.

Diagramme de séquence enregistrement d'un nouveau produit :

Séquence nominale :



Scénario Alternatif : (4)

4.1 – Le Si indique à l'internaute que les informations saisies ne sont pas justes

4.2 – L'internaute saisit à nouveau ses informations

Scénario Alternatif : (8)

8.1 – le SI indique qu'un produit semblable existe déjà

8.2 – L'internaute saisit un nouveau produit

IV – CONCEPTION

1 – Modèle conceptuel de données (MCD)

Le MCD est une représentation graphique qui permet facilement et simplement de représenter les données utilisées par le système. Il permet de comprendre les différents éléments qui sont liés entre eux.

Ici nous sommes sur le système Merise.

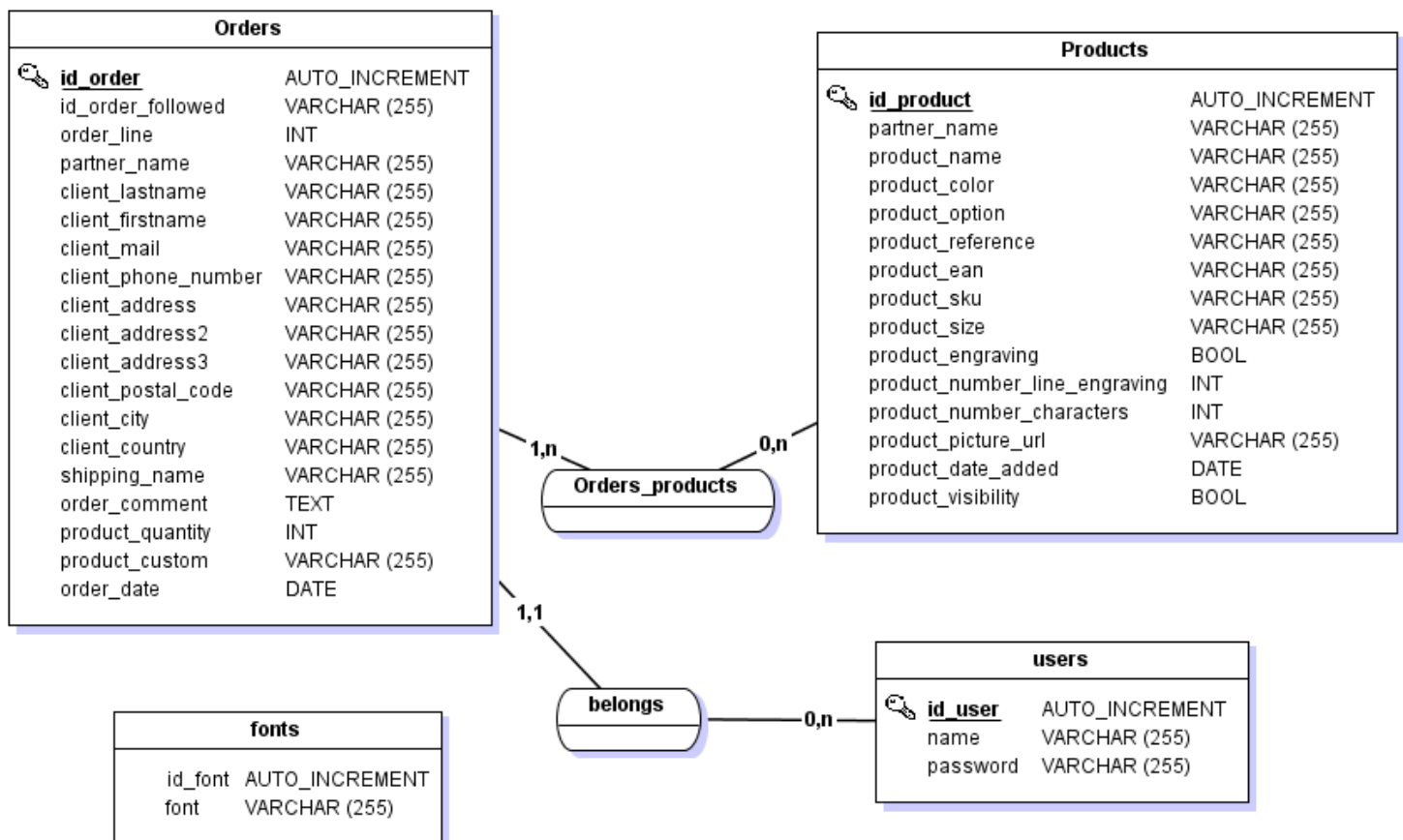
Nous constatons qu'un produit peut avoir plusieurs commandes et qu'une commande peut avoir un ou plusieurs produits.

Nous constatons également qu'un utilisateur peut avoir plusieurs commandes mais qu'en revanche une commande appartient qu'à un seul utilisateur.

Nous avons donc deux types de relations différentes à savoir :

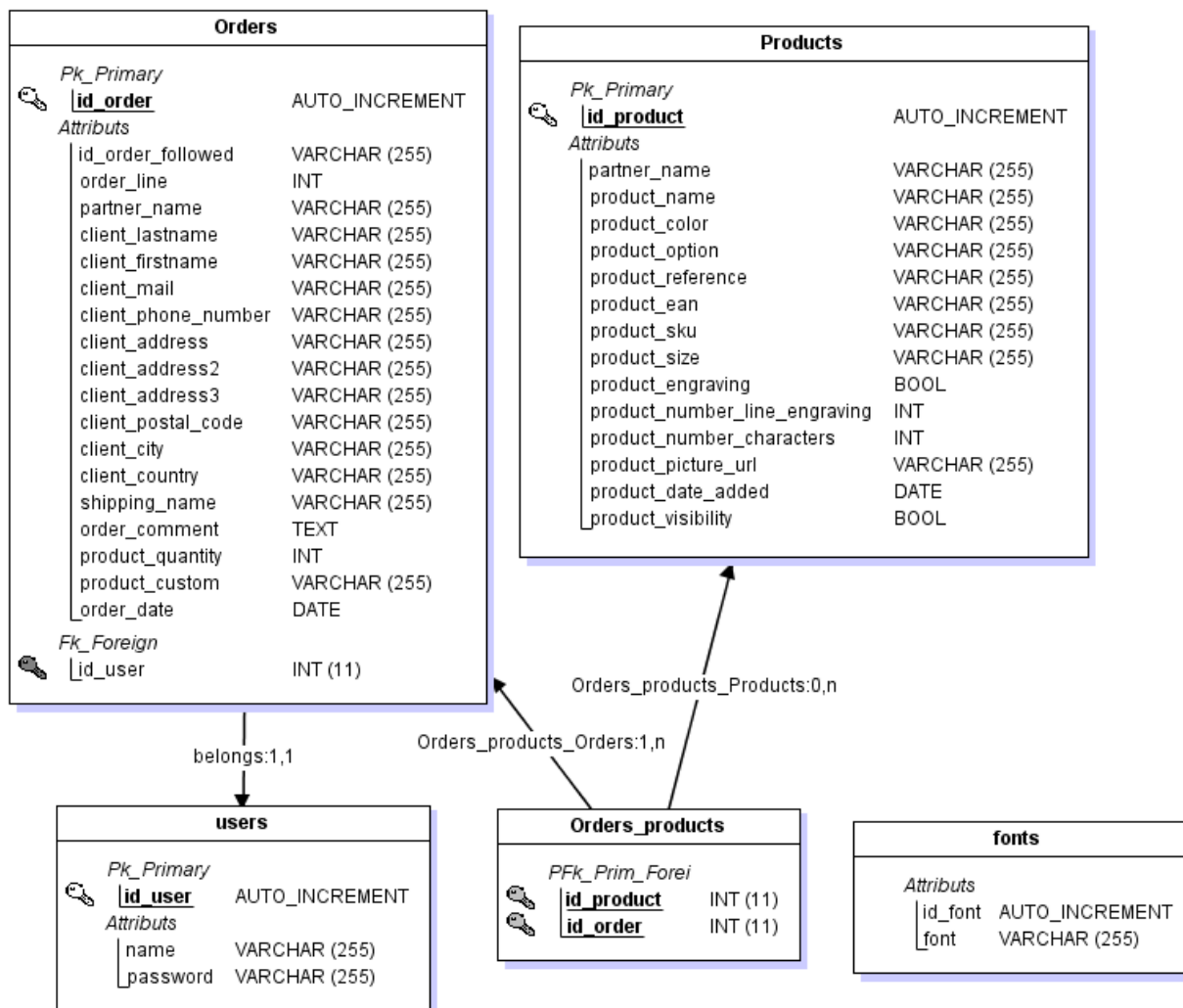
Orders « N, N » Products

Users « 1, N » Orders



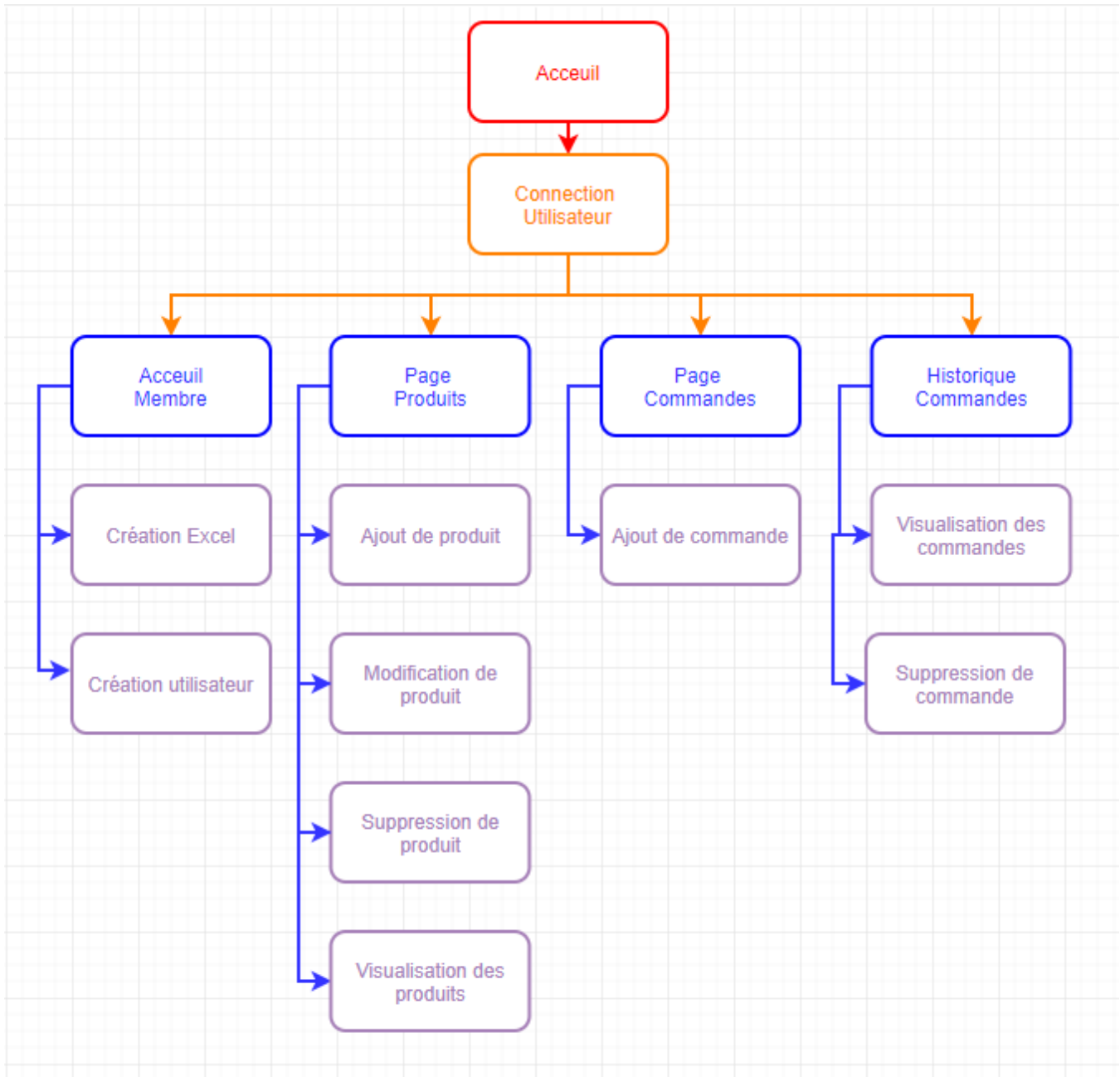
2 – Modèle logique de données (MLD)

Le Modèle Logique de Données (MLD) est une représentation des données en formalisme logique adapté au Système de Gestion de Base de Données envisagé, ici MySQL. C'est l'intermédiaire entre le modèle entité-association et le modèle physique des données.



3 – Arborescence du site

Arborescence réalisée avec draw.io.



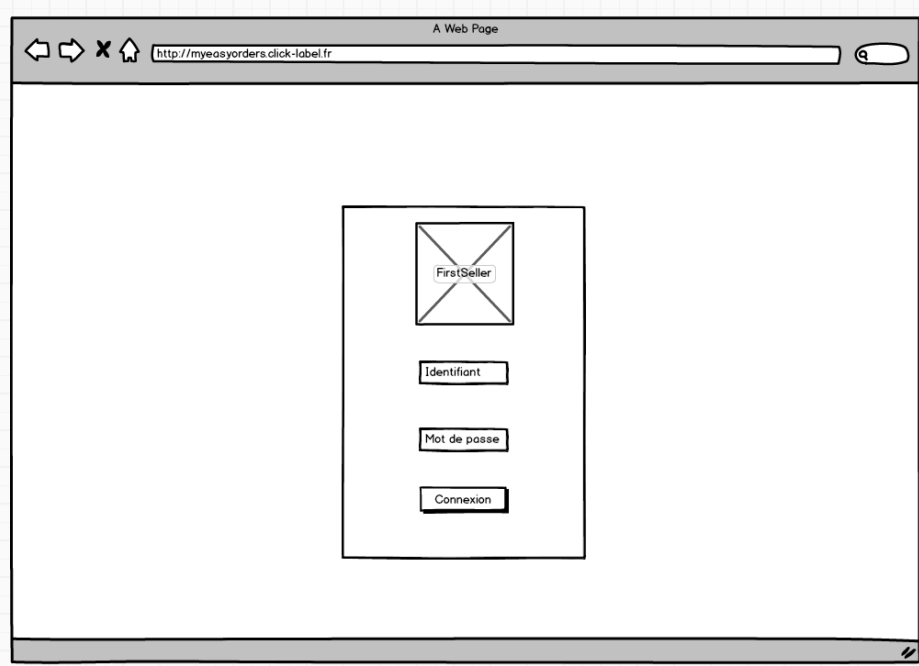
4 – Maquettage

J'ai choisi de vous présenter une maquette réalisée à l'aide de « balsamiq », représentant mes différentes pages.

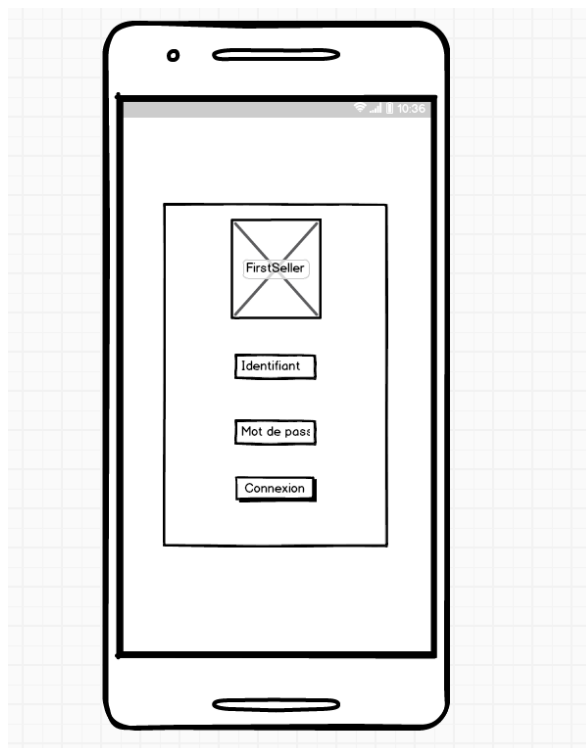
A) Connexion de l'utilisateur

Cette page permet à l'utilisateur de s'identifier afin de profiter des fonctionnalités de l'application.

Voici le visuel sur ordinateur :



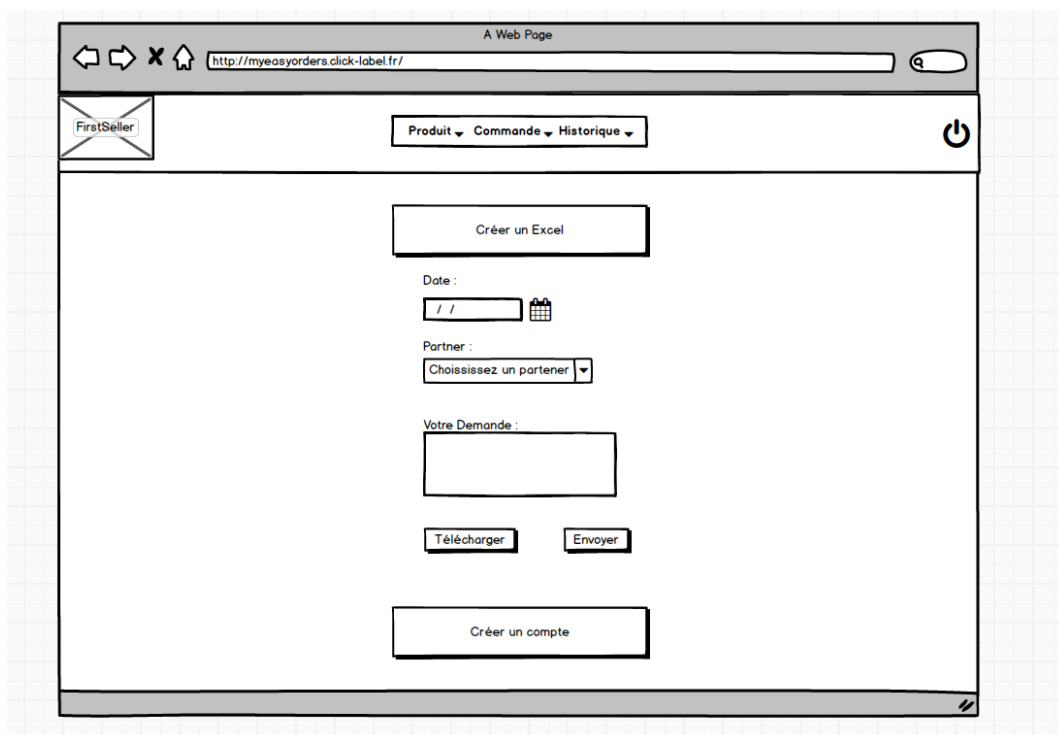
Voici le visuel sur mobile :



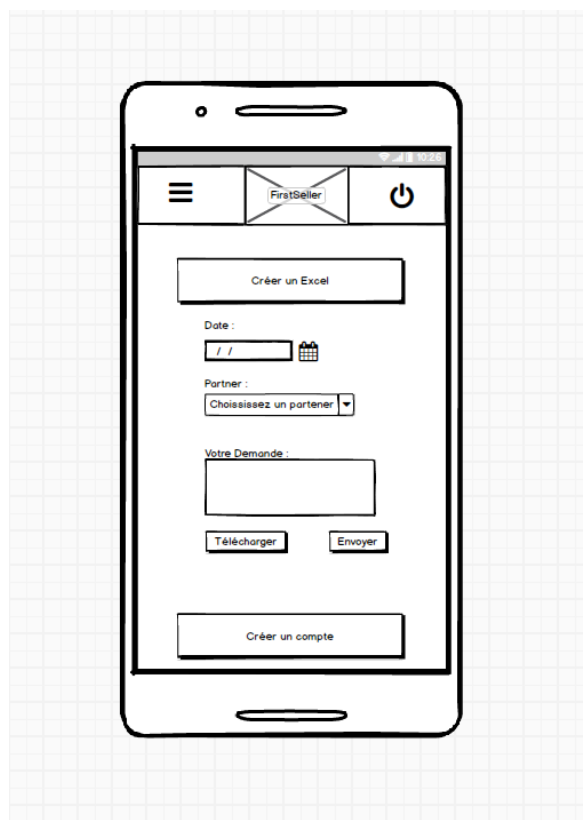
B) Page d'accueil (onglet création d'Excel ouvert)

Cette page représente visuellement l'interface de la fonctionnalité de création d'un tableau Excel avec comme filtre la date des commandes et le nom du partenaire.

Voici la représentation visuelle sur ordinateur :



Voici la représentation visuelle sur mobile :



C) Page Commande

Cette page représente visuellement l'interface d'ajout de commandes. Cette interface est semblable pour les deux partenaires pour seule différence, les informations demandées.

Voici la représentation visuelle sur ordinateur :

A Web Page

http://myeasorders.click-label.fr

FirstSeller

Produit Commande Historique

Formulaire de Commande WhyNote

Information Client :

Prénom : Nom :

Adresse : Complément Adresse :

Pays : Ville : Code Postal : Téléphone :

Produit Whynote :

Nom du produit : Couleur :

Quantité :

Voici la représentation visuelle sur mobile :

The image shows a mobile app interface for a 'Formulaire de Commande WhyNote'. At the top, there is a navigation bar with a hamburger menu icon, a 'FirstSeller' label, and a power icon. Below the navigation bar, the title 'Formulaire de Commande WhyNote' is displayed. The form is divided into two main sections: 'Information Client' and 'Produit Whynote'. The 'Information Client' section contains input fields for 'Prénom', 'Nom', 'Adresse', 'Complément Adresse', 'Pays', 'Ville', 'Code Postal', and 'Téléphone'. The 'Produit Whynote' section contains an input field for 'Nom du produit'.

Lorsqu'un utilisateur choisit un produit, une étiquette apparaît contenant les informations du produit afin de visualiser pleinement le produit sélectionné et s'assurer sur la sélection du produit.

Produit Whynote :

Nom du produit :

Book A5

Couleur :

Rechercher

The image shows a product selection card. At the top, there is a placeholder for the product image labeled 'Image Produit'. Below the image, the text reads: 'Nom du produit: Book A5', 'Couleur: Girafe', and 'Option: Lignée'.

The image shows a product selection card. At the top, there is a placeholder for the product image labeled 'Image Produit'. Below the image, the text reads: 'Nom du produit: Book A5', 'Couleur: Blanc', and 'Option: Lignée'.

The image shows a product selection card. At the top, there is a placeholder for the product image labeled 'Image Produit'. Below the image, the text reads: 'Nom du produit: Book A5', 'Couleur: Rouge', and 'Option: Lignée'.

Quantité :

Confirmer

D) Page Historique de commande (WhyNote)

Cette page représente l'historique des commandes sous format d'un tableau. Il peut être classé par colonnes, ou en effectuer une recherche grâce à l'input « search ».

Voici la représentation visuelle sur ordinateur :

Nom/Prén	Adresse	ville/Pays	Code Pos	Tel	Nom du Prod	Optic	Couleur	Date d'ajc	Modification
Romain DUPONT	40 rue blab	TOULOUS France	31400	06 66 66 66 66	Stylo effaçable		Rouge	10/05/2019	
Bob ALBERT	9 rue blab	TOULOUS France	31400	06 66 66 66 66	Stylo effaçable		Rouge	10/05/2019	
Maxime MARTIN	14 rue blab	TOULOUS France	31400	06 66 66 66 66	Stylo effaçable		Rouge	10/05/2019	
Julio ROBERT	25 rue blab	TOULOUS France	31400	06 66 66 66 66	Stylo effaçable		Rouge	10/05/2019	
Sébastien DATAGUEL	69 rue blab	TOULOUS France	31400	06 66 66 66 66	Stylo effaçable		Rouge	10/05/2019	

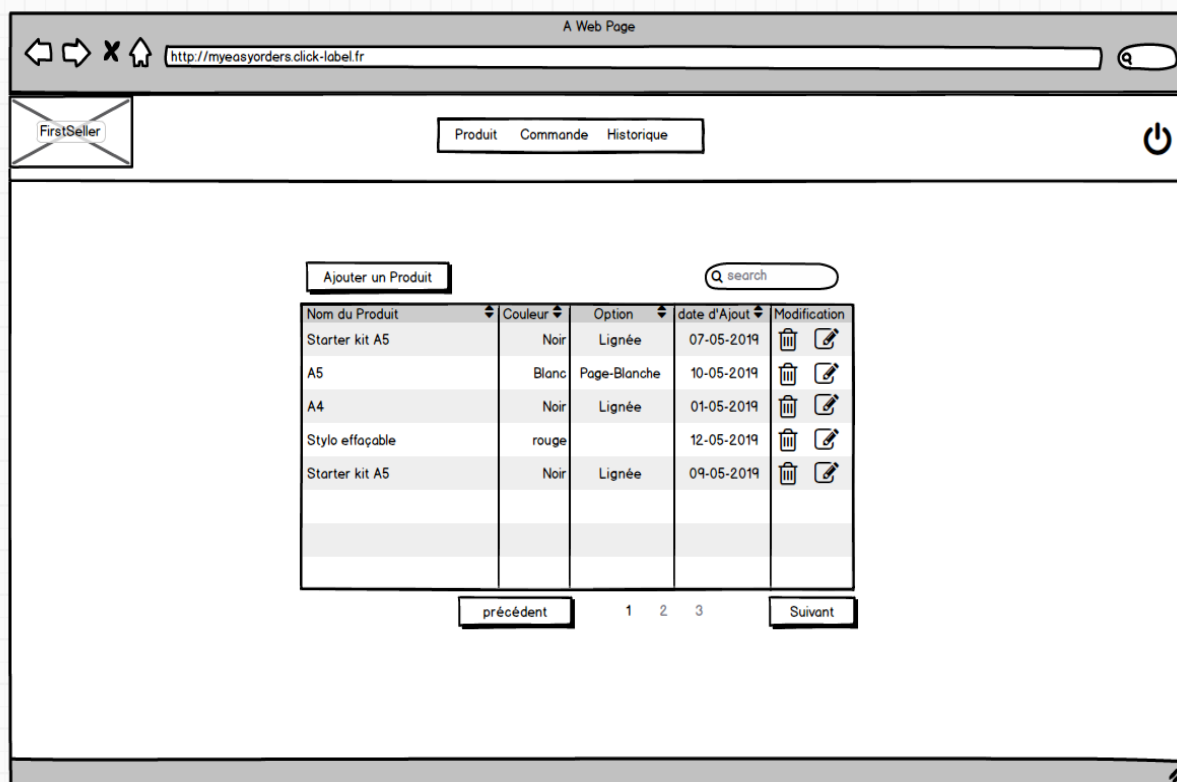
Voici la représentation visuelle sur mobile :

Nom/P	Adre	ville	Code	Tel
Romain DUPONT	40 rue	TOULC France	31400	06 66 66 66
Bob ALBERT	9 rue bl	TOULC France	31400	06 66 66 66
Maxime MARTIN	14 rue l	TOULC France	31400	06 66 66 66
Julio ROBERT	25 rue l	TOULC France	31400	06 66 66 66
Sébastien DATAGUEL	69 rue l	TOULC France	31400	06 66 66 66
Francois CALOT	23 rue	TOULC France	31400	06 66 66 66
Salim HENRIT	17 rue l	TOULC France	31400	06 66 66 66

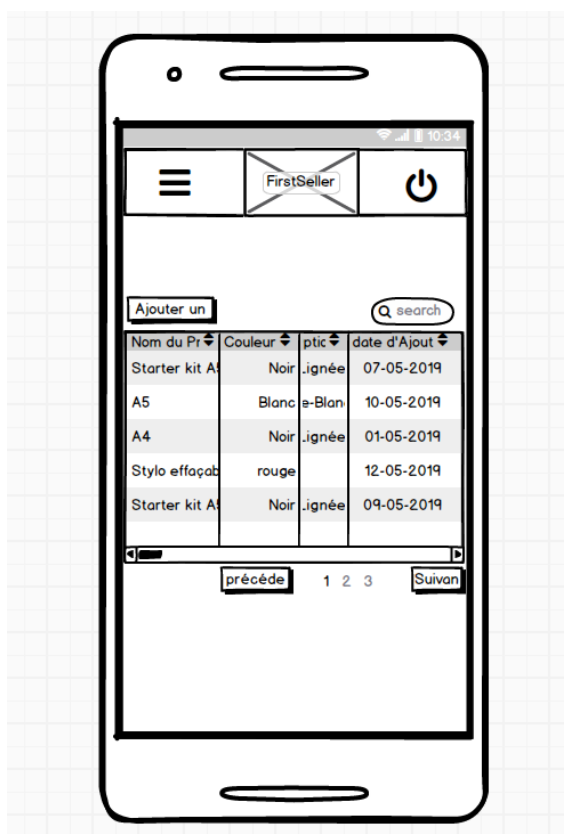
E) Page produit (WhyNote)

Cette page représente visuellement l'interface d'ajout de produit, cette interface est semblable pour les deux partenaires pour seule différence les informations demandées. Elle comporte un tableau montrant les produits déjà entrés en base de données, ainsi qu'un bouton permettant d'en ajouter.

Voici la représentation visuelle sur ordinateur :

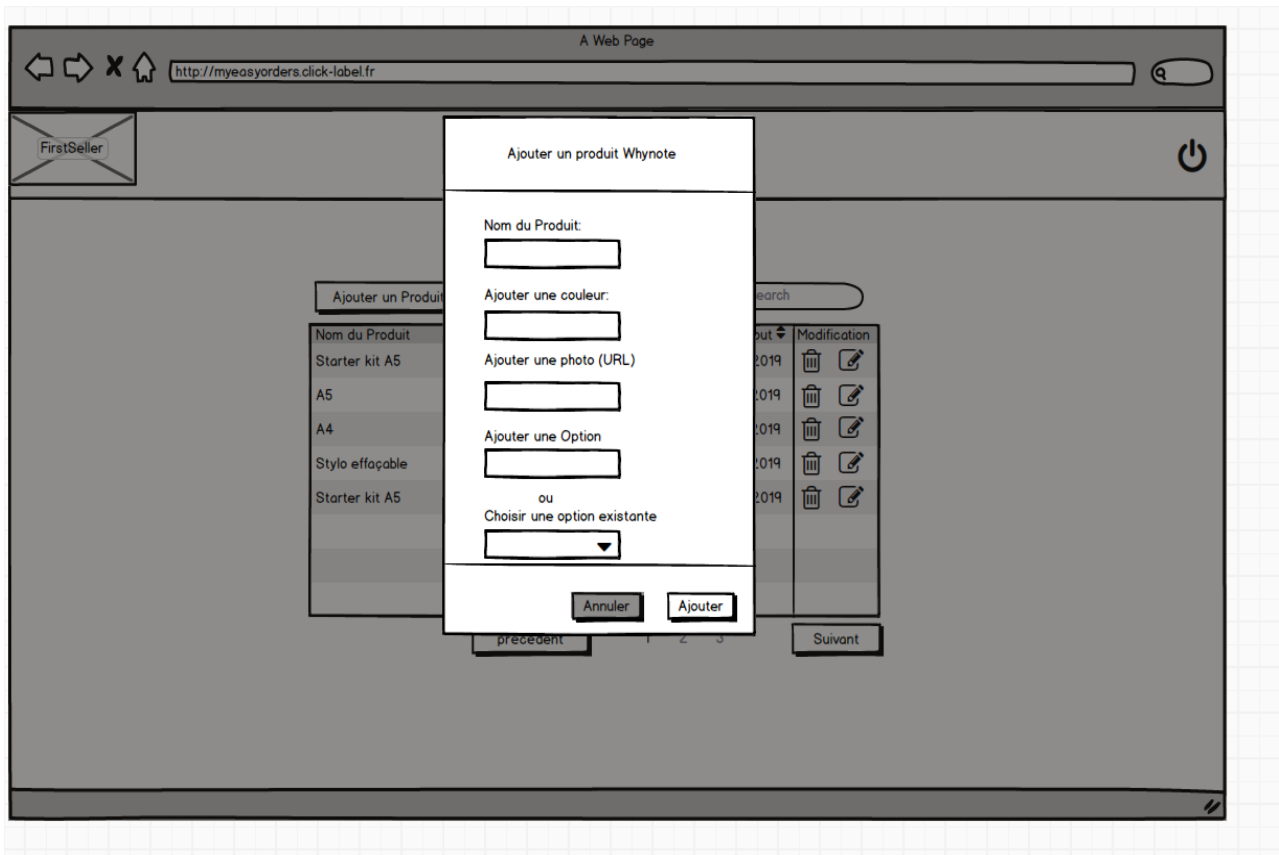


Voici la représentation visuelle sur Mobile :

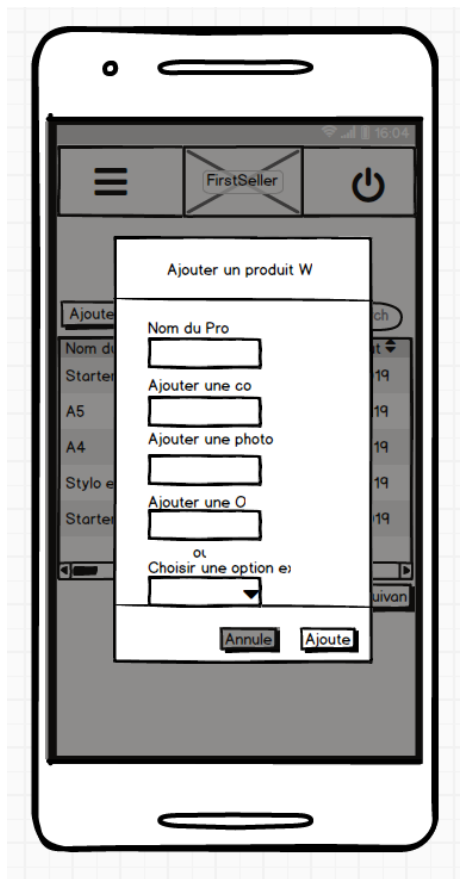


Lorsque nous appuyons sur le bouton d'Ajout de produit une modale apparaît nous permettant d'ajouter les options et la caractéristique du produit, ces modales sont également semblable pour les Partenaires pour seule différence les informations demandées

Voici la représentation visuelle sur ordinateur :



Voici la représentation visuelle sur Mobile :



V – SPECIFICATIONS TECHNIQUES

- SQL –Wamp –PhpMyAdmin -le système de gestion de base de données est MYSQL pour le développement en local. Le langage SQL me sert à exploiter notre base de données PhpMyAdmin est une application Web qui me permet plus facilement de contrôler l'état de ma base de données, de faire des requêtes SQL sans passer par des lignes de commande.



- Visual Code studio – est un éditeur de code open-source, gratuit et multiplateforme
J'ai utilisé cet IDE pour ces multiples extensions ainsi que ma connaissance sur ce support



- Ajax- Cette architecture informatique permet de construire des applications Web et des sites web dynamiques interactifs.
Je l'ai utilisé afin d'optimiser le confort d'utilisation des utilisateurs



- JQuery- jQuery est une bibliothèque JavaScript libre et multiplateforme créée pour faciliter l'écriture de scripts côté client dans le code HTML des pages web.

Elle m'a été utile afin de me faciliter l'écriture de mes requêtes Ajax ainsi que pour la manipulation du Dom.



- Composer- C'est un logiciel gestionnaire de dépendances libre écrit en PHP. Il permet à ses utilisateurs de déclarer et d'installer les bibliothèques dont le projet principal a besoin.



- **PhpSpreadsheet** – C'est une bibliothèque écrite en PHP pur fournissant un ensemble de classes permettant de lire et d'écrire dans différents formats de fichier de feuille de calcul, comme Excel et LibreOffice Calc.

PhpSpreadsheet

- **PHPMailer** –C'est une bibliothèque logicielle d'envoi d'e-mails en PHP.
Cette bibliothèque m'a facilité l'envoi de mes fichiers Excel via ma fonctionnalité d'envoi au Partenaire



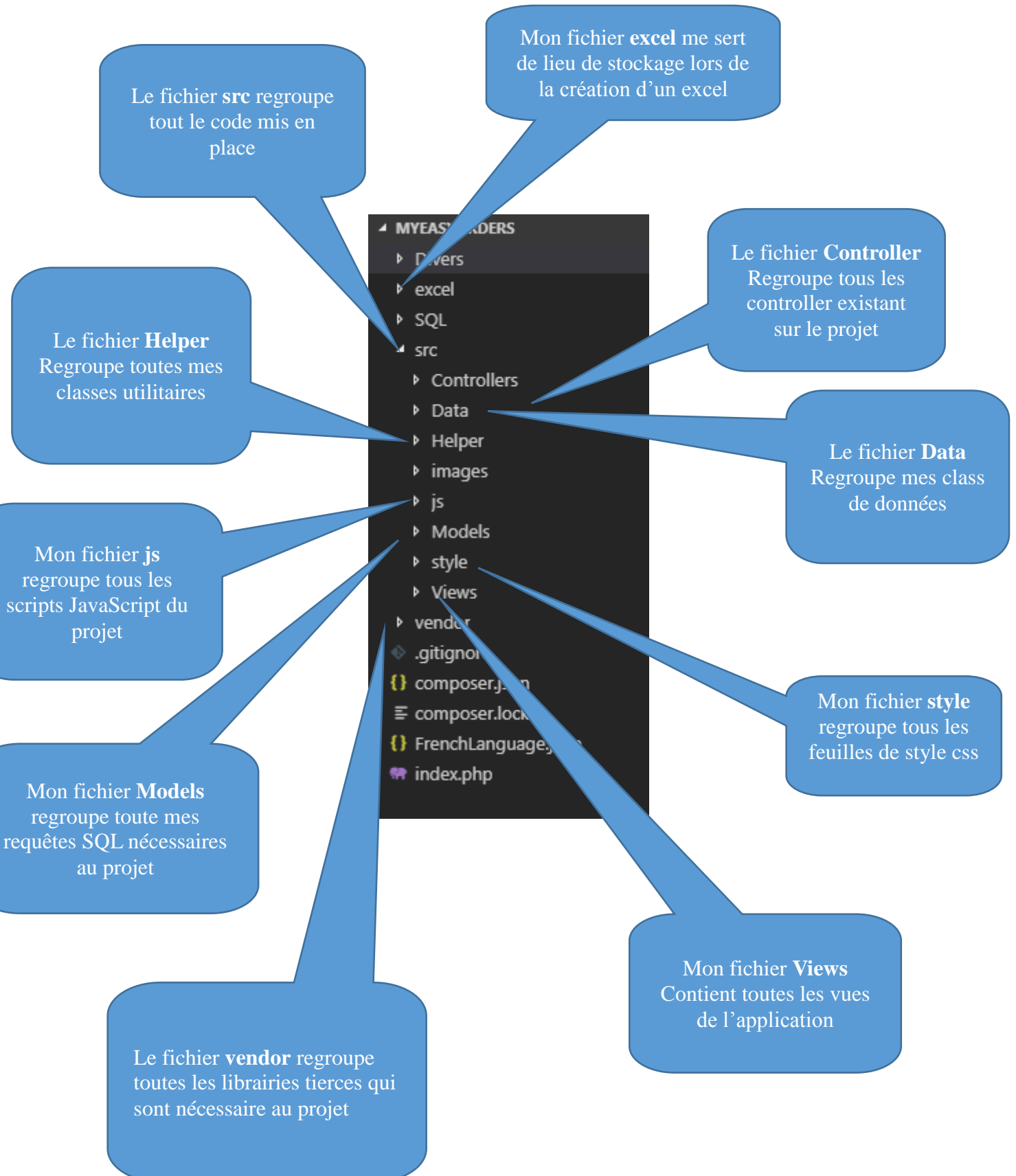
- **GitHub** – M'a permis de versionner mon application tout au long de son développement et de partager en parallèle l'avancé de mon travail sur mes différents postes de travail



VI – REALISATION

1 – Architecture

Voici l'arborescence adoptée pour ce projet.



2 – Modèle en couche MVC

Le modèle en couche MVC décrit une manière d'architecturer une application informatique en la décomposant en trois sous-parties :

- ☐ La partie Modèle ;
- ☐ La partie Vue ;
- ☐ La partie Contrôleur ;

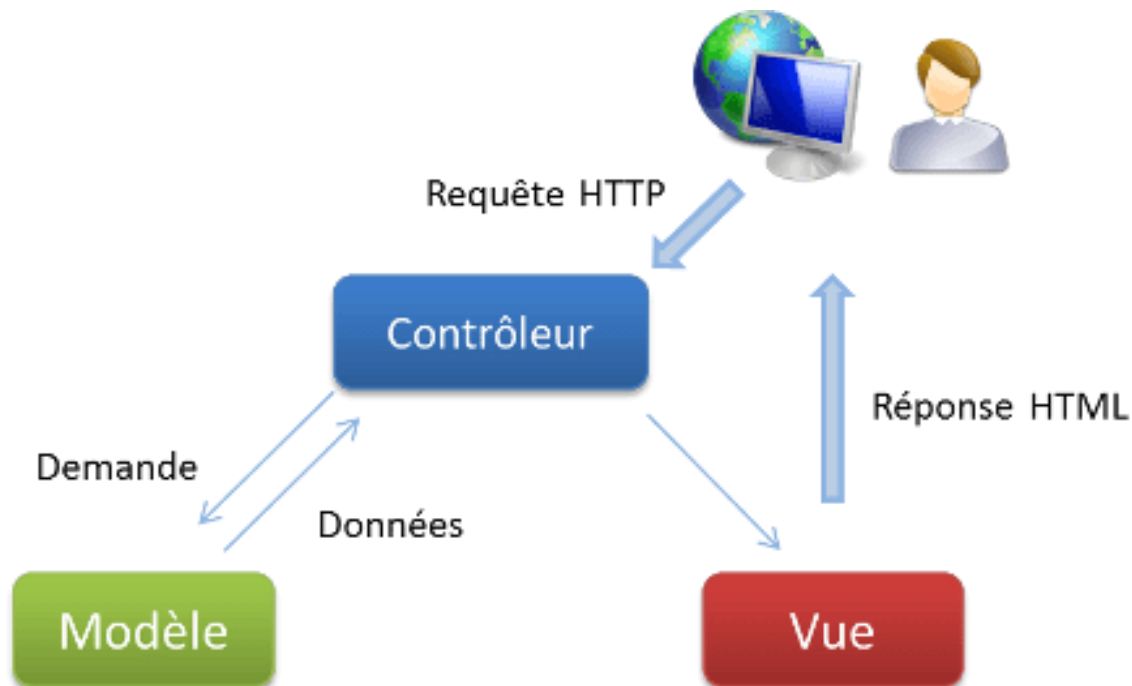
Ce modèle de conception (design pattern) a été imaginé à la fin des années 1970 pour le langage Smalltalk afin de séparer le code de l'interface graphique de la logique applicative. Il est utilisé dans de très nombreux langages : bibliothèque Swing et Model 2 (JSP) de java, Frameworks PHP, ASP.Net, MVC, etc.

La partie Modèle d'une architecture MVC encapsule la logique métier (business logic) ainsi que l'accès aux données. Il peut s'agir d'un ensemble de fonctions (Modèle procédural) ou de classes (Modèle orienté objet).

La partie Vue s'occupe des interactions avec l'utilisateur : présentation, saisie et validation des données.

La partie Contrôleur gère la dynamique de l'application. Elle fait le lien entre l'utilisateur et le reste de l'application.

Le diagramme ci-dessous résume les relations entre les composants d'une architecture MVC.



La demande de l'utilisateur (exemple : requête HTTP) est reçue et interprétée par le Contrôleur. Celui-ci utilise les services du Modèle afin de préparer les données à afficher. Ensuite, le Contrôleur fournit ces données à la Vue, qui les présente à l'utilisateur (par exemple sous la forme d'une page HTML).

On peut trouver des variantes moins « pures » de cette architecture dans lesquelles la Vue interagit directement avec le Modèle afin de récupérer les données dont elle a besoin.

□ Les avantages et inconvénients :

Le modèle MVC offre une séparation claire des responsabilités au sein d'une application en conformité avec les principes de conception déjà étudiés : responsabilité unique, couplage faible et cohésion forte. Le prix à payer est une augmentation de la complexité de l'architecture.

Dans le cas d'une application Web, l'utilisation du modèle MVC permet aux pages HTML (qui constituent la partie Vue) de contenir le moins possible de code serveur, étant donné que le scripting est regroupé dans les deux autres parties de l'application.

3 – Fonctionnement du router

Je vais tout d'abord vous expliquer mon système de lecture de requête.

Ma class **Router** sert à stocker les différentes requêtes possibles de mon application. Elle contient deux fonctions principales :

- La première méthode s'appelle **registerRequests ()**, elle me permet d'enregistrer toutes mes requêtes connues.

Mes requêtes sont stockées sous forme d'un tableau associatif avec pour clef « names » qui prendra la clé de ma requête ainsi qu'une deuxième clé « action » qui prendra en valeur la méthode qui s'exécutera lors de la réception de cette requête :

```
Router::$requests[] = [  
    'names' => ['displayListFilterEmotional'],  
    'action' => function($request){  
        echo json_encode(Router::$product->getDiferentFilterProduct('emotional'));  
        die();  
    }  
];
```

Après l'enregistrement des requêtes, la classe Router instancie toutes les classes nécessaires (Contrôleurs, Helpers) :

```
Router::$user = new UsersController();
Router::$product = new ProductsController();
Router::$order = new OrdersController();
Router::$font = new FontsController();
Router::$helper = new Helper();
```

- La deuxième fonctionnalité est **ReadRequest ()**.

Elle me permet de vérifier si la requête reçue existe dans les requêtes précédemment enregistrées, dans le cas où elle existe, elle est exécutée :

```
public static function readRequest($request)
{
    foreach (Router::$requests as $reqDatas)
    {
        $isRequest = true;

        foreach ($reqDatas['names'] as $name)
        {
            if (!isset($request[$name])) {
                $isRequest = false;
                break;
            }
        }

        if ($isRequest) {
            $reqDatas['action']($request);
            break;
        }
    }
    require 'src/Views/loginform.php';
}
```


4 – Fonctionnalité d'ajout, et visualisation d'une commande

Nous allons voir en détail le processus d'ajout d'une commande de suppression et de visualisation.

Lorsque nous voulons ajouter une commande nous arrivons sur un formulaire.

Formulaire Commandes Emotional :

Informations Clients :

ID commande :	Nom :	
<input type="text"/>	<input type="text"/>	
Prénom :	Téléphone :	
<input type="text"/>	<input type="text"/>	
Adresse :	Complément d'adresse :	
<input type="text"/>	<input type="text"/>	
Code Postal :	Ville :	Pays :
<input type="text"/>	<input type="text"/>	<input type="text"/>
Email :	Commentaire :	
<input type="text"/>	<input type="text"/>	

Chercher un produit Emotional:

SKU :	Taille :	<input type="button" value="Rechercher"/>
<input type="text"/>	<input type="text"/>	
Quantité :	<input type="button" value="Confirmer"/>	
<input type="text"/>		

Les informations demandées sont des informations que retourne la Platform Amazon suite à une commande.

Après avoir remplis les informations client L'utilisateur doit choisir un produit.

Une liste déroulante apparait donc pour les SKU (une unité de gestion des stocks ou SKU (de l'anglais Stock Keeping Unit)) et les différentes tailles déjà inscrites en base de données.

Adresse : Complément d'adresse :

Code Postal : Ville : Pays :

Email : Commentaire :

R048FJ824
J8564PJKD
S924GEFZE
P3178EFK

Unit Emotional:

Taille :

Quantite :

Email : Commentaire :

Chercher un produit

SKU :

Quantite :

Les volets déroulants sont générés comme suis :

```
function displayDataList(data, target) {
    for (i = 0; i < data.length; i++) {
        target.append("<option value='" + data[i][0] + "'>" + data[i][0] + "</option>");
    }
}

function displayListFilter() {
    $.post("index.php", 'displayListFilterEmotional',
        function(data) {
            dataResult = JSON.parse(data);
            displayDataList(dataResult[0], $('#dropDownOrderProductSku'));
            displayDataList(dataResult[1], $('#dropDownOrderProductSize'));
        })
}

displayListFilter();
```

Lorsque l'utilisateur accède à la vue command, la méthode **displayListFilter()** s'exécute générant la liste déroulante.

Cette fonctionnalité envoie une requête par protocole AJAX au serveur, celui-ci traite les requêtes depuis une classe statique **Router** expliquée précédemment :

```
1 <?php
2     require_once 'src/Data/Router.php';
3
4     session_start();
5
6
7     Router::registerRequests();
8     Router::readRequest($_POST);
9
```

Ici lorsque la requête reçue aura pour clef « **displayListFilterEmotional** », la méthode présente en tant que valeur de la clef « action » sera exécutée.

```
Router::$requests[] = [
    'names' => ['displayListFilterEmotional'],
    'action' => function($request){
        echo json_encode(Router::$product->getDiferentFilterProduct('emotional'));
        die();
    }
];
```

Dans ce cas la clé « action » contient la méthode **getDiferentFilterProduct('emotional')** contenu dans mon Controller.

```
public function getDiferentFilterProduct($partner){
    if($partner == 'whynote'){
        $filter=["product_name","product_color"];
        $products=[$this->model->getDiferentOptionProduct($filter[0],'whynote'),
        $this->model->getDiferentOptionProduct($filter[1],'whynote')];
        return $products;
    }
    else{
        $filter=["product_sku","product_size"];
        $products=[$this->model->getDiferentOptionProduct($filter[0],'emotional'),
        $this->model->getDiferentOptionProduct($filter[1],'emotional')];
        return $products;
    }
}
```

Cette méthode associe dans un tableau « \$filter » les noms des filtres en fonction du paramètre entrées.

Puis renvoie un tableau « \$product » contenant le résultat des requêtes SQL appeler par la méthode (**getDiferentoptionProduct ()**).

```
function getDiferentOptionProduct($filter,$partner){
    $products = $this->fetchAll([
        'query' => "SELECT DISTINCT :filter FROM Products where
        product_visibility = true and partner_name = :partner_name;",
        'definitions' => [ ':filter' => $filter, ':partner_name' => $partner]);
    return $products;
}
```

Elle contient une surcharge de la méthode **fetchAll ()** ;

Cette méthode retourne un tableau contenant tout le résultat de ma requête.

```
public function fetchAll($data)
{
    $request = $this->executeQuery($data);
    return $request->fetchAll();
}
```

exectuteQuery () est une méthode me permettant de me connecter à ma base de données et de pouvoir réaliser une requête à partir d'un tableau associatif ayant pour première clé « query » et pour deuxième « definition ».

```
public function executeQuery($data)
{
    $request = $this->connectDatabase()->prepare($data['query']);
    if (!isset($data['definitions']) || $data['definitions'] == null || $data['definitions'] == []) {
        $request->execute();
    }
    else {
        $request->execute($data['definitions']);
    }
    return $request;
}
```

Le résultat de cette requête est donc transformé en tableau Json avec la méthode **Json_encode ()**.

Une fois le SKU choisit ou la taille choisi, L'utilisateur voit apparaitre des propositions de produit ans une balise « figure » avec la description de celle-ci.

Ces informations sont récupérées de la même façon que l'exemple illustré ci-dessous puis

stockées dans des data-attributs :

```
for (i = 0; i < dataEmotionalProduct.length; i++) {
  $("#choicesProduct").append("<div class='divProduct'><figure class='figureProduct'>"+
    "<img src='" + dataEmotionalProduct[i]["product_picture_url"] + "' class='pictureProduct'>"+
    "<figcaption class='description'>description:<br>skuProduct:<strong>" + dataEmotionalProduct[i]["product_sku"] +
    "</strong><br>sizeProduct:<strong>" + dataEmotionalProduct[i]["product_size"] + "</strong></figcaption></figure>"+
    "<input type='radio' data-id_product='" + dataEmotionalProduct[i]["id_product"] +
    "' data-product_name='" + dataEmotionalProduct[i]["product_name"] +
    "' data-product_reference='" + dataEmotionalProduct[i]["product_reference"] +
    "' data-product_ean='" + dataEmotionalProduct[i]["product_ean"] +
    "' data-product_size='" + dataEmotionalProduct[i]["product_size"] +
    "' data-product_engraving='" + dataEmotionalProduct[i]["product_engraving"] +
    "' data-product_number_line_engraving='" + dataEmotionalProduct[i]["product_number_line_engraving"]
    + "' name='inputCheckProduct'></div>")
}
```

Chercher un produit Emotional:

SKU :

Taille :

A8-11U7-GJ1H

Rechercher

Produits Trouver :



L'utilisateur peut donc sélectionner le produit en question.
Il ne lui reste plus qu'à indiquer la quantité ainsi qu'à valider le formulaire.

Une fois le formulaire rempli avec les champs obligatoires, la commande s'enregistre de la façon suivante :

```
emotionalOrderData = new Object();
$("#EmotionalOrderForm input").each(function() {
  emotionalOrderData[$(this).attr("name")] = $(this).val();
});
```

On récupère les valeurs stockées dans les inputs dans un objet dont la clé sera le nom de l'input et la valeur la valeur de l'input.

Ensuite on envoie cet objet dans une requête par protocole AJAX au serveur.

```
$.post("index.php", [{
    "dataEmotionalClient": emotionalOrderData,
}], function(data) {
    alert(data);
});
```

Je récupère donc cette requête dans ma class Router afin de vérifier l'existence d'une requête du même nom.

Cette requête appelle donc ma méthode **addEmotionalOrder ()** contenu dans ma class **ordersController**.

```
Router::$requests[] = [
    'names' => ['dataEmotionalClient'],
    'action' => function($request){
        Router::$order->addEmotionalOrder($request['dataEmotionalClient']);
        die();
    }
];
```

Cette méthode sert à appeler les requêtes contenues dans la méthode **addEmotionalOrder ()**, ainsi que d'ajouter un message de succès.

```
public function addEmotionalOrder($dataClient){
    $this->model->addEmotionalOrder($dataClient);
    echo 'Commande Ajouté';
    die();
}
```

Comme expliquer antérieurement ma méthode **executeQuery ()** me permet d'exécuter ma requête avec un tableau a deux clés l'une contenant une requête et la deuxième contenant la valeur des clés.

Je répète donc cette opération pour ma table « Orders » puis pour ma table associative

```
function addEmotionalOrder($dataClient){
    $this->executeQuery([
        'query' => 'INSERT INTO Orders VALUES (
            NULL,:id_order_followed,NULL,:partner_name,:client_lastname,
            :client_firstname,:client_mail,:client_phone_number,:client_address,
            :client_address2,NULL,:client_postal_code,:client_city,:client_country,
            :shipping_name,:order_comment,:product_quantity,:product_custom,
            NOW(),(SELECT id_user FROM users WHERE name = "'.$_SESSION['name'].'" ));',
        'definitions' => [
            ':partner_name' => 'emotional',
            ':id_order_followed' => $dataClient['id_order_followed'],
            ':client_lastname' => $dataClient['client_lastname'],
            ':client_firstname' => $dataClient['client_firstname'],
            ':client_mail' => $dataClient['client_mail'],
            ':client_phone_number' => $dataClient['client_phone_number'],
            ':client_address' => $dataClient['client_address'],
            ':client_address2' => $dataClient['client_address2'],
            ':client_postal_code' => $dataClient['client_postal_code'],
            ':client_city' => $dataClient['client_city'],
            ':client_country' => $dataClient['client_country'],
            ':shipping_name' => $dataClient['shipping_name'],
            ':order_comment' => $dataClient['order_comment'],
            ':product_quantity' => intval($dataClient['product_quantity'], 0),
            ':product_custom' => $dataClient['product_custom'],
        ]
    ]);
    $this->executeQuery([
        'query' => 'INSERT INTO Orders_products VALUES(
            :id_product,
            (SELECT id_order FROM Orders ORDER BY id_order DESC LIMIT 1)
        );',
        'definitions' => [':id_product' => $dataClient['id_product']]
    ]);
    return true;
}
```

Lorsque l'utilisateur voudra consulter l'historique des commandes il devra se rendre sur l'onglet Historique.
Un tableau récapitulatif s'affichera alors sous la forme ci-dessous :

ments

Rechercher :

Numero de la commande:	Nom/Prenom:	Téléphone:	Adresse:	Complément d'adresse:	Code Postal:	Ville/Pays:	Ean du Produit:	Référence du Produit:	Nom du Produit	Quantité:	date D'ajout:
55	Géraud / Karine	0607603558	48 Résidence de Seignan		09200	Montjoie en Couserans / France (Métropole)	3663457065687	3663457065687	Bague femme Médaille Charm's	1	2019-07-09
89	HAMDANI / Johan	0635479199	Hameau du Saouze	Villa 42	84120	PERTUIS / France	3663457065687	3663457065687	Bague femme Médaille Charm's	1	2019-07-15
104	Gros / Gwenaëlle	0625025977	13 Rue de Château		08190	St Germainmont / France (Métropole)	3663457065687	3663457065687	Bague femme Médaille Charm's	1	2019-07-17
106	Cruz / Audrey	0609868822	2 impasse des vergers		69630	Chaponost / France (Métropole)	3663457065687	3663457065687	Bague femme Médaille Charm's	1	2019-07-17
48	Bernat / Josiane	0786492928	17 rue Salvador Dali		31860	Labarthe Sur Leze / France (Métropole)	3663457072753	3663457072753	Bague femme Médaille Charm's Star	1	2019-07-08
101	Malveau / Allisson	0608682331	86 Avenue Marcel Lacassagne		82240	Septfonds / France (Métropole)	3663457065687	3663457065687	Bague femme Médaille Charm's	1	2019-07-16

Ce Tableau s'affiche grâce à une requête par protocole AJAX au serveur lors de l'appel de la page.

```
$.post("index.php", { "contentTableHistoricEmotional"
    function(Json) {
        data = JSON.parse(Json);
```

Je récupère donc cette requête dans ma class Router afin de vérifier l'existence d'une requête du même nom.

```
Router::$requests[] = [
    'names' => ['contentTableHistoricEmotional'],
    'action' => function($request){
        echo json_encode(Router::$order->model->getAllOrders('emotional'));
        die();
    }
];
```

Cette requête appelle donc ma méthode **getAllOrders()** contenu cette fois ci dans ma class **orders** qui correspond à mon model order, afin d'exécuter la requête ci-dessous :

```
function getAllOrders($partner){
    $orders=$this->fetchAll(['query' => 'SELECT * FROM Products , Orders, Orders_products
    where Products.partner_name= :partner_name
    and orders.id_order = Orders_products.id_order
    and Products.id_product = Orders_products.id_product;',
    'definitions' => [':partner_name' => $partner]
    ]);

    return $orders;
}
```

Le résultat est donc retourné dans mon callback lors de l'envoi de ma requête puis traiter par une méthode **appendTableEmotionalHistoric ()**.

```
function appendTableEmotionalHistoric(dataArrayBody, targetEmotional) {
    countElementHeaderTable = $("#headerTableForEmotionalProduct").children().length;
    for (i = 0; i < dataArrayBody.length; i++) {
        tr = document.createElement("tr");
        for (y = 0; y < countElementHeaderTable; y++) {
            td = document.createElement("td");
            if (y == 0) {
                td.append(dataArrayBody[i]["id_order_followed"]);
                tr.append(td);
            } else if (y == 1) {
                td.append(dataArrayBody[i]["id_order"]);
            } else if (y == 2) {
                td.append(dataArrayBody[i]["client_lastname"] + " / " + dataArrayBody[i]["client_firstname"]);
            } else if (y == 3) {
                td.append(dataArrayBody[i]["client_phone_number"]);
            } else if (y == 4) {
                td.append(dataArrayBody[i]["client_address"]);
            } else if (y == 5) {
                td.append(dataArrayBody[i]["client_address2"]);
            } else if (y == 6) {
                td.append(dataArrayBody[i]["client_postal_code"]);
            } else if (y == 7) {
                td.append(dataArrayBody[i]["client_city"] + " / " + dataArrayBody[i]["client_country"]);
            } else if (y == 8) {
                td.append(dataArrayBody[i]["product_ean"]);
            } else if (y == 9) {
                td.append(dataArrayBody[i]["product_reference"]);
            } else if (y == 10) {
                td.append(dataArrayBody[i]["product_name"]);
            } else if (y == 11) {
                td.append(dataArrayBody[i]["product_quantity"]);
            } else if (y == 12) {
                td.append(dataArrayBody[i]["order_date"]);
            } else if (y == 13) {
                $("", { class: 'deleteBoutonForEmotionalProduct', type: 'button',
                    "id": dataArrayBody[i]["id_order"] }).appendTo(td);
            }
            td.setAttribute("scope", "row");
            tr.append(td);
        }
        targetEmotional.append(tr);
    }
    $("", { class: 'fas fa-trash-alt' }).appendTo($(".deleteBoutonForEmotionalProduct"));
}
```


Cette méthode me permet de créer une balise « td » à l'intérieur de chaque « tr » avec la valeur de l'index correspondant.

3 – Fonctionnalité d'ajout, d'un nouvel utilisateur.

Nous allons voir en détail le processus d'ajout d'un nouvel utilisateur

Lorsque nous voulons ajouter un utilisateur nous arrivons sur ce formulaire.
L'utilisateur doit rentrer les informations dans le formulaire ci-dessous

Créer un compte



INSCRIPTION:

Nom d'utilisateur

Mot de passe

Confirmer mot de passe

VALIDER

Lorsque l'utilisateur envoie les données du formulaire,

La requête est récupérée par ma class **route** comme ci-dessous :

```
Router::$request[] = [  
    'names' => ['loginregister'],  
    'action' => function($request){  
        Router::$user->registerUsers($request['loginregister'],$request['passregister'],$request['pass_confirmregister']);  
    }  
];
```

Cette requête appelle donc ma méthode **registerUsers ()** contenu dans mon Controller **users**.

```
public function registerUsers($loginName,$loginPass,$loginPass_confirm){  
    if(count($this->model->getAllUser($loginName))==0){  
        if ($loginPass == $loginPass_confirm ) {  
            $this->model->registerUsers($loginName,$loginPass);  
            echo ("compte créer </br>");  
        }  
        else {  
            echo("<p>mot de passe différent</p>");  
            require 'src/view/membre.php';  
        }  
    }  
    else{  
        echo('Identifiant déjà utiliser </br>');  
        require 'src/view/membre.php';  
    }  
}
```

Cette méthode appelle plusieurs requêtes contenues dans mon model user.

La première s'appelle **getAllUser ()**, elle me permet de récupérer la totaliser des utilisateurs ayant le même nom que le nom choisit par l'utilisateur sous forme d'un tableau afin de vérifier l'existence d'un utilisateur portant le même nom.

```
public function getAllUser($login){  
    $this->executeQuery([  
        'query' => 'SELECT * FROM users WHERE name=:name',  
        'definitions' => [':name' => $login]  
    ]);  
}
```

La deuxième s'appelle **registerUsers ()**,
Cette méthode est appelée uniquement si le tableau de **getAllUser ()** est égal à zéro,
Elle me permet d'exécuter un insert dans ma table **users**.

```
public function registerUsers(){  
    $this->executeQuery([  
        'query' => 'INSERT INTO users VALUES (null ,:name,:password);',  
        'definition' => [':name' => $loginName,  
            ':password' => password_hash($loginPass, PASSWORD_DEFAULT)]  
    ]);  
}
```

VII – CONCLUSION

Ce stage m'a permis de m'épanouir dans le milieu du développement, et de confirmer mon envie d'évoluer dans ce métier.

Lors de mon stage j'ai voulu en priorité satisfaire mon client. Pour cela, mon optique était de réaliser une application fonctionnelle avec une phase test dans les temps.

Avec plus de temps j'aurais souhaité mettre en place toutes les fonctionnalités souhaitées comme un scraper afin de récupérer directement les produits sur les différents sites ou une meilleure gestion des utilisateurs avec un profil visiteur.

Ces fonctionnalités sont optionnelles mais restent des améliorations possibles de l'application.

Je n'ai pas vraiment eu l'occasion de travailler en équipe au cours de mon stage ce qui, d'une certaine façon, m'a manqué.

Cependant cette expérience m'a amené à analyser petit à petit mes erreurs et j'en retire un grand profit.

À ce jour l'application comporte toutes les fonctionnalités principales demandées par le client, une phase de tests d'utilisation a été réalisée et l'application est utilisée tous les jours. J'ai gardé contact avec l'utilisateur principal afin d'être averti d'éventuels bugs ou d'amélioration.

Pouvoir consulter mon application et savoir qu'elle est utilisée tous les jours reste pour moi la meilleure des récompenses.