

PINON Benjamin

Titre professionnel niveau III

Développeur d'application Web et mobile



Mémoire Professionnel

FIRST SELLER

Application de contrôle de fiches produits



Remerciements

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

Je remercie nos professeurs pour leur disponibilité et pour avoir su nous guider tout au long de la formation.

J'aimerais également remercier les différents intervenants tels que Mr Anthony M et Mr Philippe D qui ont su partager leurs connaissances et leur passion avec nous.

Je souhaite également exprimer ma reconnaissance envers mes collègues de formation pour leur enthousiasme, leur esprit de partage et d'entraide qui est 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

ABSTRACT –	p.6
INTRODUCTION –	p.7
Compétences couvertes par le projet –	p.8
I – Le cahier des charges	
1. Présentation de l’entreprise	p.9
1.1) Présentation	
1.2) Interlocuteurs	
2. Besoin du client	p.9
2.1) L’expression du besoin	
2.2) Les objectifs du logiciel	
2.3) La clientèle	
2.4) Périmètre du projet	
3. Exigences techniques	p.10
3.1) Framework	
3.2) Système d’administration	
3.3) Interface	
3.4) Exigences non fonctionnelles	
II – Spécifications fonctionnelles	
1. Planification	p.15
2. Méthodologie et organisation	p.15
III – Spécifications techniques	
1. Maquettage fonctionnel	p.16
2. L’environnement de développement	p.18

IV – Modélisation

- 1. Les diagrammes de conception UML p.20
 - 1.1) Le diagramme de cas d'utilisation
 - 1.2) Le diagramme de séquence
 - 1.3) Le diagramme d'activité
 - 1.4) Le diagramme de classe

VI - Conception et architecture de l'application

- 1. Les modèles physiques des données p.27
 - 1.1) Le modèle conceptuel de données (MCD)
 - 1.2) Le modèle logique des données (MLD)
 - 1.3) Dictionnaire des données
- 2. Conception de la base de données p.33
 - 2.1) Conception avec PhpMyAdmin et MySQL

VI – Réalisation

- 1. Méthodologie p.35
 - 1.1) Structure du projet Symfony
 - 1.2) Le modèle en couche MVC
- 2. La couche « métier » p.37
 - 2.1) Les relations entre entités avec Doctrine
- 3. Le repository p.39
 - 3.1) Les requêtes de Doctrine
 - 3.2) l'EntityManager
- 4. Les formulaires p.44
 - 4.1) L'objet « Form » et la création d'un formulaire
- 5. L'interface graphique p.46
 - 5.1) Gestion de la responsivity avec Bootstrap
 - 5.2) Utilisation de Twig
 - 5.3) Le routage

6. Le système d'alerte avec PHP Mailer	p.49
6.1) Librairie PHPMailer	
6.2) Script PHP	
7. Python et sa librairie Scrapy	p.53
7.1) Introduction à Python	
7.2) Python et les librairies utilisées	
7.3) Les scripts du « Scraper » de Cdiscount	
CONCLUSION -	p.60
ANNEXES -	p.61

Abstract

My client is an E-commerce business based in the Toulouse north area selling products over various E-commerce platforms as Amazon, Cdiscount, Fnac and more...

This project consists to develop a web application which aims to help the company's employees in their daily tasks to check if any sellers were to be present on any of their product's sheets on the Ecommerce website C-Discount.

With this software, the employee would be able to check how many sellers are present over their product's sheet and be notified by email if any of the criteria passed to the mailer/notifier are true, as the number of sellers present or if any seller may be selling the same product cheaper than they are.

This project had to follow a previous one made over the Amazon website, which was made with the PHP language and Symfony 4 framework. As the client also liked the previous design of the Amazon Application, I had to keep the same background color and architecture.

As I needed to learn to develop with the MVC model, I thought this project was the perfect opportunity to develop my PHP skills.

I also had to learn Python's language when I discovered I had to collect a large amount of datas as fast as I could knowing that PHP wasn't the perfect language for this task, and also to automate those tasks.

Introduction

Dans le cadre de ma formation en tant que Développeur d'application numérique web et mobile de niveau III, j'ai eu pour tâche de réaliser un projet pendant ma période de stage d'une durée de 10 semaines avec l'entreprise E-commerce First Seller basée au Nord de Toulouse.

A la demande de mon client (David Bonavia), le but de l'application demandé lors de mon stage au sein de leur entreprise, était de collecter des données sur le site Cdiscount selon une place de marché donnée et de vérifier les données collectées pour améliorer leurs ventes et de suivre l'évolution des prix des produits voulus grâce à un système de notification par Email ainsi que de détecter la présence de vendeurs autre que mon client sur une fiche produit.

Les conditions imposées lors de l'élaboration de cette application étaient que la barre de navigation doive garder la même couleur qu'une application précédemment faite par un autre stagiaire ainsi que le design général des pages du site. Cette application sera donc développée avec le Framework Symfony 4 et PHP 7 pour pouvoir joindre les différents projets ensemble par la suite dans le but d'héberger ces sites en ligne.

Ce fut l'opportunité pour moi d'apprendre à travailler avec un Framework de développement tout en respectant les exigences du modèle MVC (**M**odel **V**iew **C**ontroler). Mon objectif étant d'approfondir mes connaissances et développer une application sur le langage de programmation PHP et sur ce Framework qui mobilise une très forte communauté et étant de plus en plus utilisé en milieu professionnel.

J'ai en adéquation avec le planning établi, pu développer certaines fonctionnalités demandées par le client, telle qu'un « scraper » écrit avec le langage Python et le Framework **Scrapy**, un système de notification avec l'envoi d'email en PHP. Le tout automatisé via les tâches **CRONS** que met à disposition l'hébergeur. 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.

Compétences couvertes par le projet

1. 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.

2. 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.

I – Le cahier des charges

1 - Présentation de l'entreprise

1.1) Présentation

FirstSeller est une entreprise familiale spécialisée dans l'E-commerce (Retailer), fondée en 2016 et dirigée par trois associés :

- David BONAVIA : Président et responsable des Achats
- Sébastien BERTHUY : Directeur général et commercial
- Kevin BONAVIA : Webmaster et administrateur des places de marchés.

A ce jour, la société compte en plus 6 employés répartis comme suit : 2 web designer, 1 chef de projet designer, 1 logisticien, 1 assistant commercial, 1 assistant pour la gestion et la mise en ligne des différents produits.

A ses débuts la société proposait la vente en ligne des produits de différents fournisseurs afin qu'ils puissent étendre leur marché et leur visibilité dans l'e-commerce. Aujourd'hui elle est plus focalisée sur la vente en ligne de leurs propres produits qu'elle vend par le biais de place de marchés tel qu'Amazon, Cdiscount, Fnac et eBay.

L'entreprise vend tout type de produits, de la cosmétique jusqu'à l'œnologie en passant par la téléphonie et l'informatique.

1.2) Interlocuteurs

Mon interlocuteur principal sera Mr Bonavia David, l'application sera principalement axée sur ses besoins.

2 – Besoin du client

2.1) L'expression du besoin

Mon client a exprimé le besoin d'une application permettant la collecte de données sur la plateforme de Cdiscount.

Cette application leur permettrait d'être alertés en cas de présence d'un autre vendeur sur une de leur fiche produit et ainsi de demander le retrait de ce vendeur sur le produit en question, ou de suivre l'évolution des prix sur certains produits donnés.

La mise en place de cette application leur évitera de scruter leurs annonces présentes sur Cdiscount pour vérifier la présence de ces autres vendeurs en question et ainsi d'avoir un gain de temps considérable chaque jour.

2.2) Les objectifs du logiciel

Cette application leur permettra de renseigner une place de marché donnée déjà présente sur le site E-commerce Cdiscount et d'en suivre tous ses produits et ainsi être alerté par emails lors de la présence d'un autre vendeurs sur un des produits de la place de marché en question ou autres critères pouvant être ajoutés au mailer principal.

2.3) La clientèle

L'application sera accessible à n'importe qui présent dans l'entreprise, avec différents niveaux d'administrations. Un stagiaire pourra donc ajouter sa place de marché sur l'application et être alerté par emails, pour un suivi plus rapide des fiches produits sur Cdiscount.

2.4) Périmètre du projet

Le site sera mis en ligne en français et avec l'avancement des technologies mobiles il est donc indispensable de fournir au moins un site responsive adaptable sur tablette.

3 - Exigences techniques

3.1) Framework

Ce projet sera ajouté avec une application précédemment faite par un autre stagiaire, application faite via le Framework Symfony, le maquetage de mon application restera similaire à celle déjà proposé et sera donc développé sur Symfony avec un modèle MVC avec le langage PHP principalement.

3.2) Système d'administration

Le système d'administration restera similaire à l'autre l'application déjà crée précédemment avec un Utilisateur et un Administrateur. L'administrateur pourra ajouter, modifier les droits ou supprimer un autre utilisateur ainsi que la suppression d'une place de marché.

L'utilisateur pourra accéder au service de base de l'application tel que l'ajout d'une place de marché et son suivi, l'ajout d'autres utilisateurs sur sa place de marché et la création de nouveaux utilisateurs sur sa place de marché. Chaque nouvel inscrit aura le rôle d'Utilisateur obligatoirement.

Chaque utilisateur devra se connecter via un formulaire de connexion avant de pouvoir accéder aux services de l'application.

Concernant les droits des places de marchés, nous aurons trois niveaux d'administration :

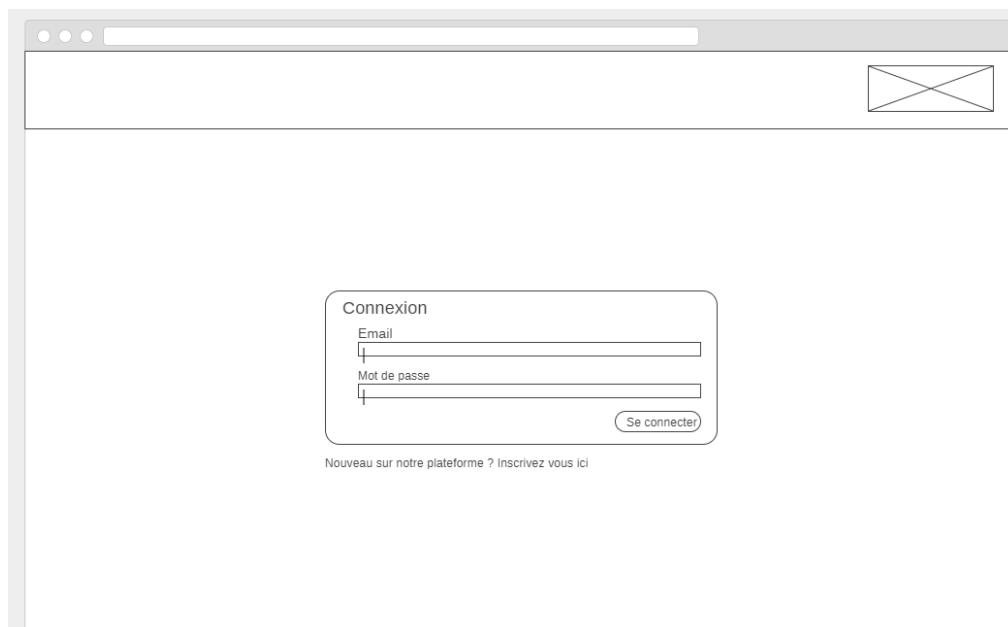
- Le créateur : Créateur de la place de marché, pourra supprimer la place de marché et ses utilisateurs, modifier les droits de chaque utilisateurs, géré les alertes via les préférences générales de la place de marché en question et éditer ses informations en cas de changement de nom de la boutique par exemple.
- L'administrateur : l'administrateur pourra changer les rôles des utilisateurs et les supprimer, il ne pourra pas supprimer un autre administrateur, seul le créateur en aura les droits.
- L'utilisateur : l'utilisateur aura accès à la boutique, pourra suivre les produits et être alerté par emails.

3.3) Interface

Le client a exprimé le désir de garder le maquettage de l'application antérieur comme base pour la nouvelle application.

Le rajout d'une barre latérale de navigation entre les différentes parties de l'application sera donc obligatoire pour rendre l'application plus simple d'utilisation.

Fig 1 : Formulaire de connexion sur ordinateur

La figure présente une maquette d'un formulaire de connexion dans une fenêtre de navigateur. Le formulaire est intitulé "Connexion" et se trouve au centre de la page. Il contient deux champs de saisie : "Email" et "Mot de passe", chacun avec un curseur de texte. À droite du champ "Mot de passe", il y a un bouton "Se connecter". En dessous du formulaire, un lien hypertexte dit "Nouveau sur notre plateforme ? Inscrivez vous ici". Dans le coin supérieur droit de la fenêtre du navigateur, il y a un bouton de fermeture (un rectangle avec une croix).

Connexion

Email

Mot de passe

Se connecter

Nouveau sur notre plateforme ? Inscrivez vous ici

Fig 2 : Formulaire de connexion sur tablette

The image shows a tablet interface with a login form. The form is titled "Connexion" and contains two input fields: "Email" and "Mot de passe". A "Se connecter" button is positioned to the right of the password field. Below the form, there is a link that says "Nouveau sur notre plateforme ? Inscrivez vous ici". The tablet has a home button at the bottom and a status bar at the top.

Connexion

Email

Mot de passe

Se connecter

Nouveau sur notre plateforme ? Inscrivez vous ici

Fig 3 : Dashboard sur ordinateur

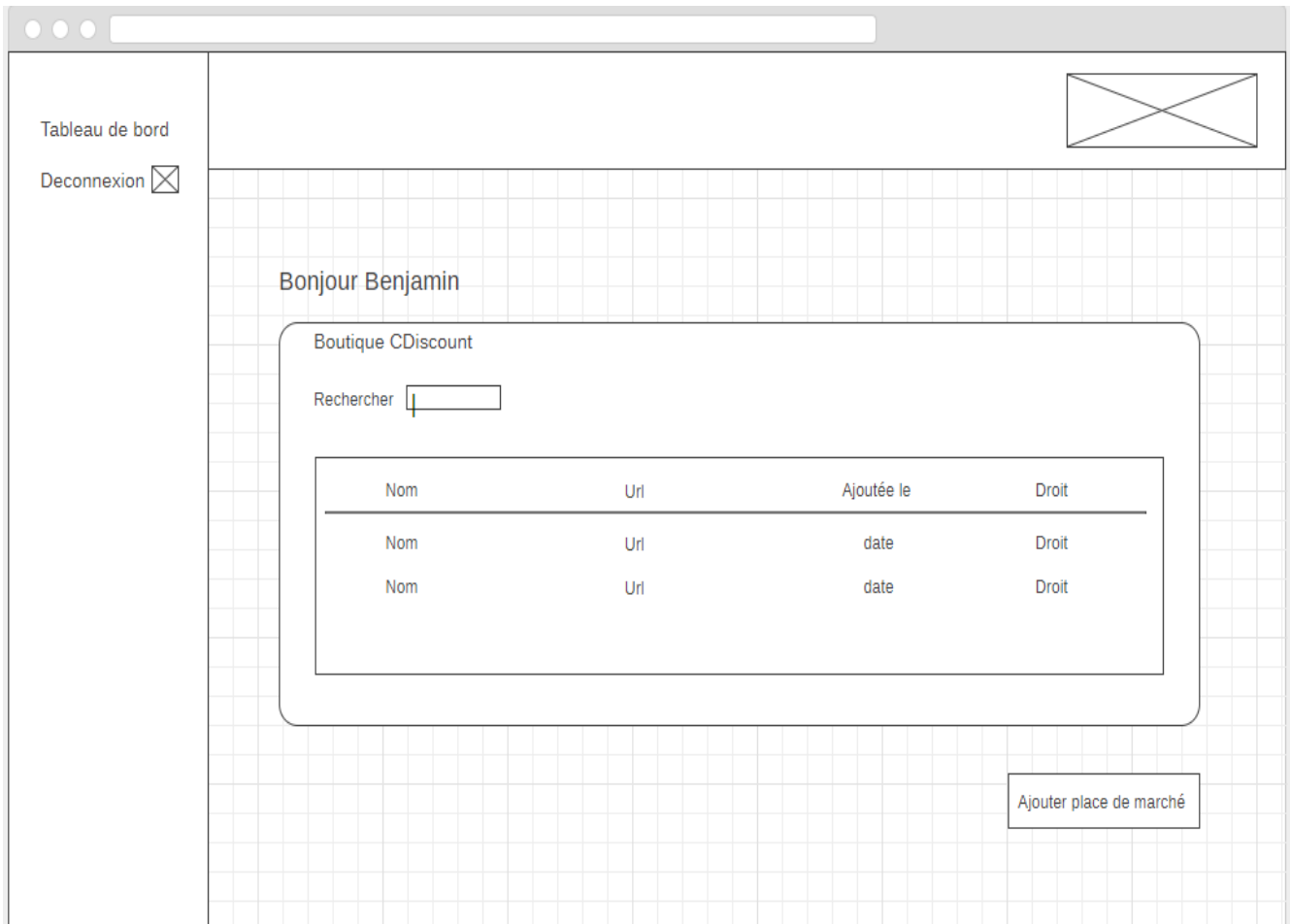
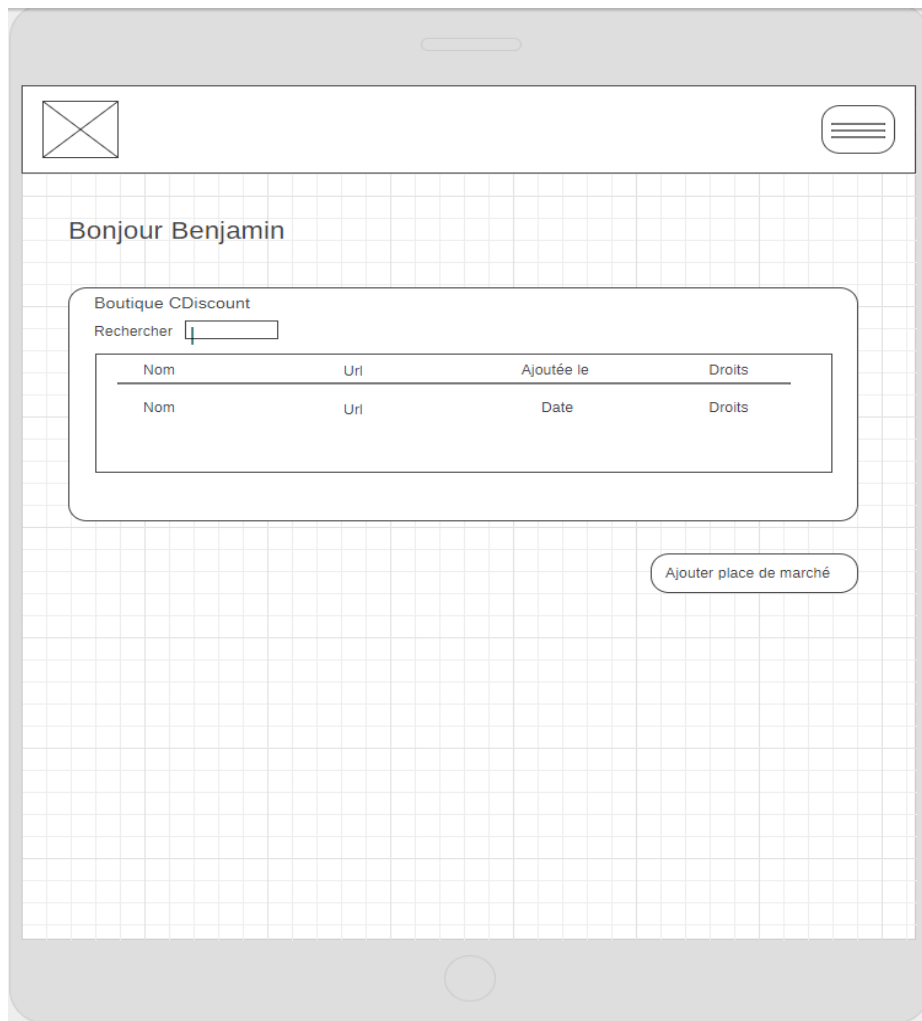
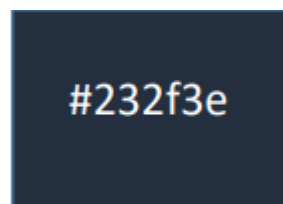


Fig 4 : Dashboard sur tablette



Charte graphique : Pour réaliser le projet nous devons utiliser principalement le bleu foncé #232f3e comme couleur de fond.



3.4) Exigences non fonctionnelles

La phase de développement comprendra divers tests fonctionnels. De plus, en fin de développement, j'effectuerais un test de validation.

Régulièrement, avec mon client, nous avons prévu d'effectuer des réunions afin de vérifier si les fonctionnalités développées correspondaient bien à ses attentes.

II – Spécifications fonctionnelles

1 - Planification

Pour ce projet de développement, aucun planning ne fut réalisé étant donné que le seul participant à ce projet était donc moi-même, la seule condition était d'avoir une version BETA de l'application prête pour début Mars 2019.

2- Méthodologie

L'usage de plus en plus répandu de Symfony 4 peut s'expliquer par une communauté très active et la pléthore de « packages » de développement, les bundles mise à disposition du développeur, en facilitent grandement son travail.

J'ai donc privilégié la création d'un projet avec Symfony pour avoir une arborescence dites MVC avec la création d'un répertoire **App (src)**, dans le cadre de la réalisation de l'arborescence et de l'organisation du projet, ainsi que l'usage de bundles. Bien qu'il ne soit pas obligatoire, la création de ce répertoire **App (src)** est vivement recommandée par les créateurs et la communauté des utilisateurs du Framework Symfony ceci pour plusieurs raisons :

- Afin d'organiser son projet de manière efficiente pendant les différentes phases de développement.
- De mieux gérer les dépendances et ainsi éviter les conflits.
- Afin d'assurer un meilleur développement et maintenabilité de l'application, et par la suite sa mise en production.

Comme pour d'autres Frameworks, l'utilisation de Symfony 4 répond à des conventions et des normes de développement qui régissent les « best-practices » et la création d'un répertoire **App (Src)** comme bundle principal, cela fait partie des approches recommandées par les développeurs de Symfony.

III – Spécifications techniques

1. Maquettage fonctionnel

D’après la définition du cahier des charges, le Mockup (ou dessin filaire) est une étape primordiale du design d’un site web. Le Mockup permet de concevoir une ébauche du design final.

Pour la réalisation du maquettage fonctionnel nous avons utilisé le programme de **Balsamiq Mockup**. Nous avons opté pour ce programme dans la conception du firmware de l’application, son utilisation et sa prise en main rapide ainsi que son efficacité dans l’agencement des divers éléments composant le firmware.

Nous avons présenté quelques maquettes des pages du site au client afin de les faire valider avant de commencer la réalisation de l’application. Voici un exemple du wireframe de la page d’accueil de l’application.

L’application sera aussi disponible sur le format tablette pour gérer son utilisation plus facilement, et ainsi de répondre aux exigences du responsive design, nous lui avons également présenté à cet effet un maquettage de cette même page.

Fig 5 : Mockup de la page d’accueil format ordinateur

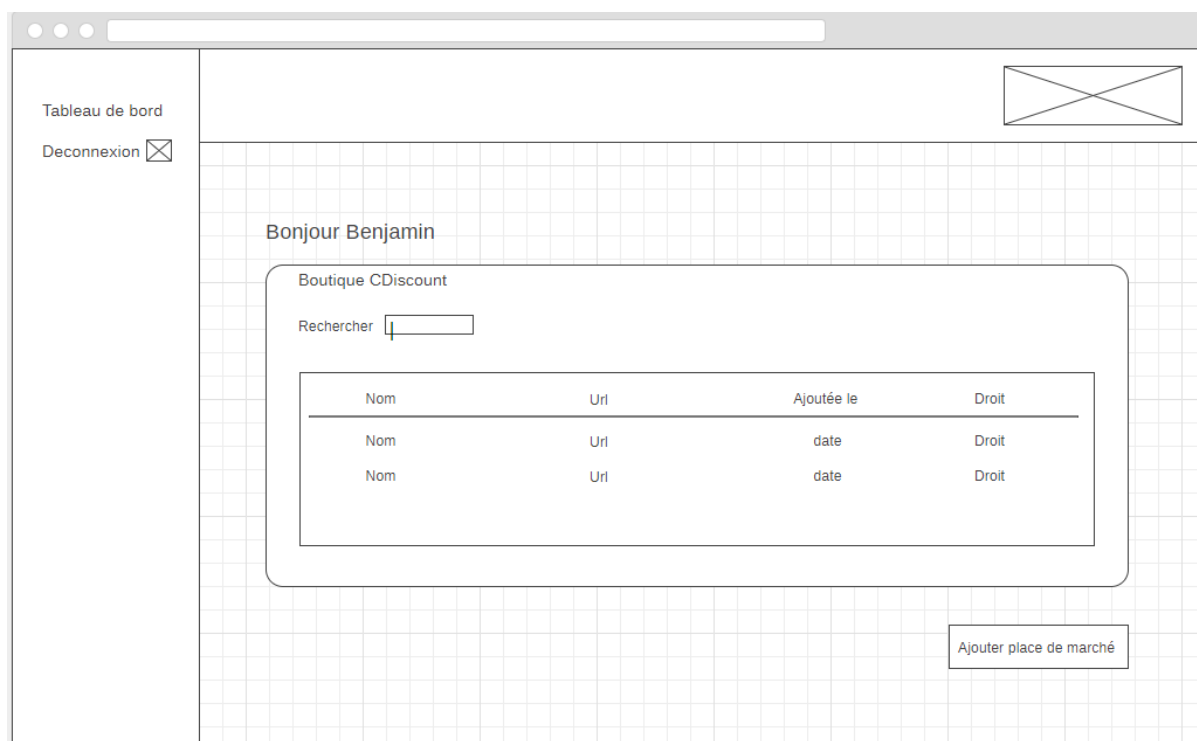
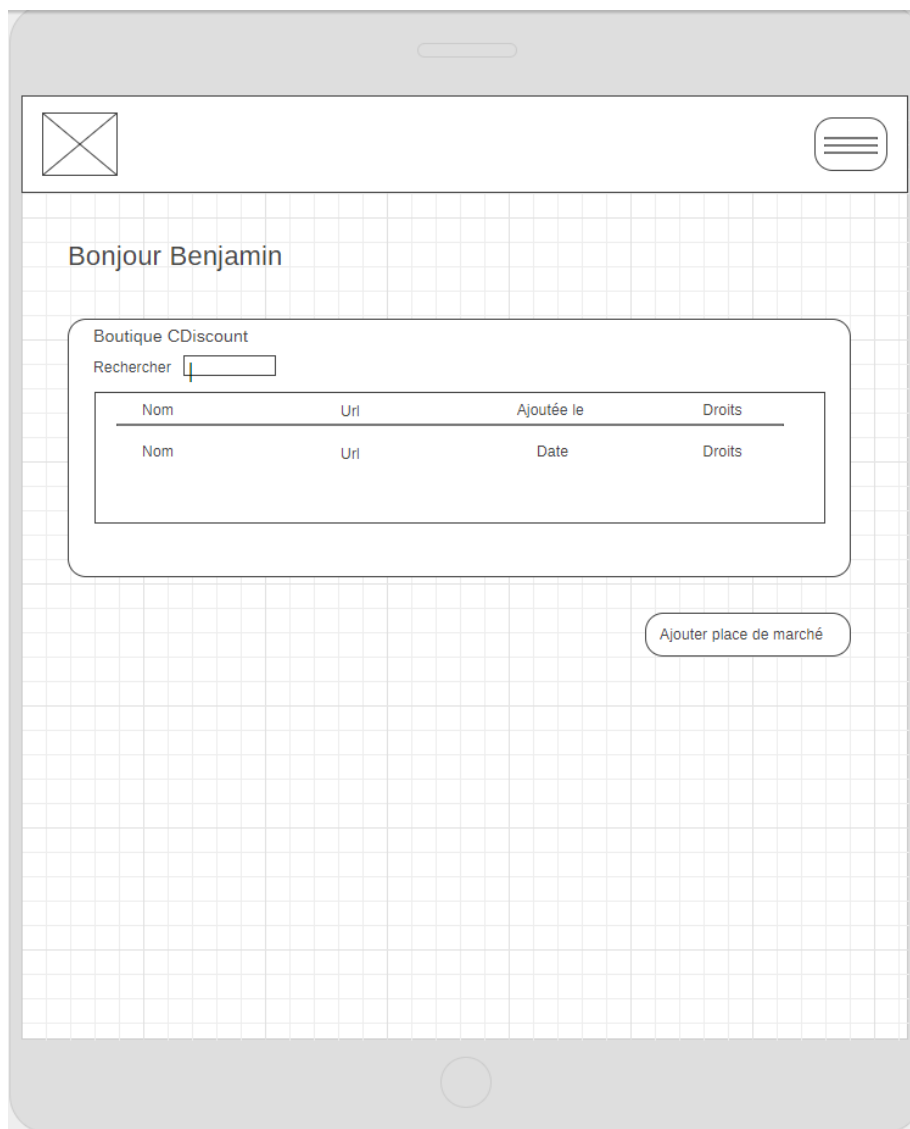


Fig 6 : Mockup de la page d'accueil format tablette responsive web design



La version mobile sera dotée d'un menu (Format dit « Burger ») déroulant pour ne pas encombrer les pages principales de l'application une fois que l'écran sera à 992px ou moins de largeur.

2. L'environnement de développement

- **PhpStorm**

Nous avons choisi cet IDE pour ses nombreux plugins qui ont grandement facilité au quotidien le développement de l'application, tel que la reconnaissance et l'accès rapide aux méthodes utilisées, l'auto-complétion avec PHP mais également avec l'ORM Doctrine

- **Le Framework MVC Symfony**

Pour la réalisation de l'application dans le respect des exigences imposées par le design pattern MVC, nous avons fait le choix de travailler avec le Framework Symfony.

L'usage de plus en plus répandu de Symfony peut s'expliquer par une communauté active et de nombreux « packages » de développement, les bundles, mis à disposition du développeur et qui facilitent grandement son travail.

Nous avons décidé de travailler avec la version 4.0 qui est la dernière version stable connue à ce jour.

- **Bootstrap**

Bootstrap est un framework CSS couramment utilisé pour structurer simplement ses pages et rendre son site web adaptatif (ou *responsive* en anglais, s'adaptant aux différentes tailles d'écrans).

Pour la réalisation du markup, nous avons utilisé la version 4.0 de Bootstrap, par son système de **Grid** qui a facilité l'intégration du **mockup** et le positionnement des éléments des pages.

- **Twig**

Twig est un moteur de template qui permet d'afficher du code dynamiquement directement dans les templates.

- **Composer**

Le gestionnaire de dépendances Composer installe automatiquement les différents composants externes indispensables au bon fonctionnement de notre application, comme par exemple la bibliothèque Symfony 4 ou les bundles que nous avons utilisés.

- **StarUml**

StarUml est un logiciel de modélisation UML.

- **JMerise**

Nous avons utilisé Merise pour la modélisation du modèle conceptuel des données selon les méthodes de conception Merise.

- **Langages :**

- PHP 7.2 : pour la partie serveur
- Python 3.7 : pour la partie scraping/collecte de données
- Javascript : pour la partie FRONT
- HTML 5
- CSS 3

IV – Modélisation

1. Les diagrammes de conception UML

Cette application ne partant d'aucune base de données existante, son élaboration n'en était que plus simple.

Nous avons réalisé les différents diagrammes de conception et pour cela, nous avons utilisé le langage UML (**U**nified **M**odeling **L**anguage) qui permet de modéliser l'intégralité de l'application grâce à un langage normé. Celui-ci permet de visualiser aussi bien l'analyse des besoins client, comme le fonctionnement du système pendant un traitement d'information ou comment notre application sera déployée.

Nous avons ainsi travaillé sur quatre types de diagrammes :

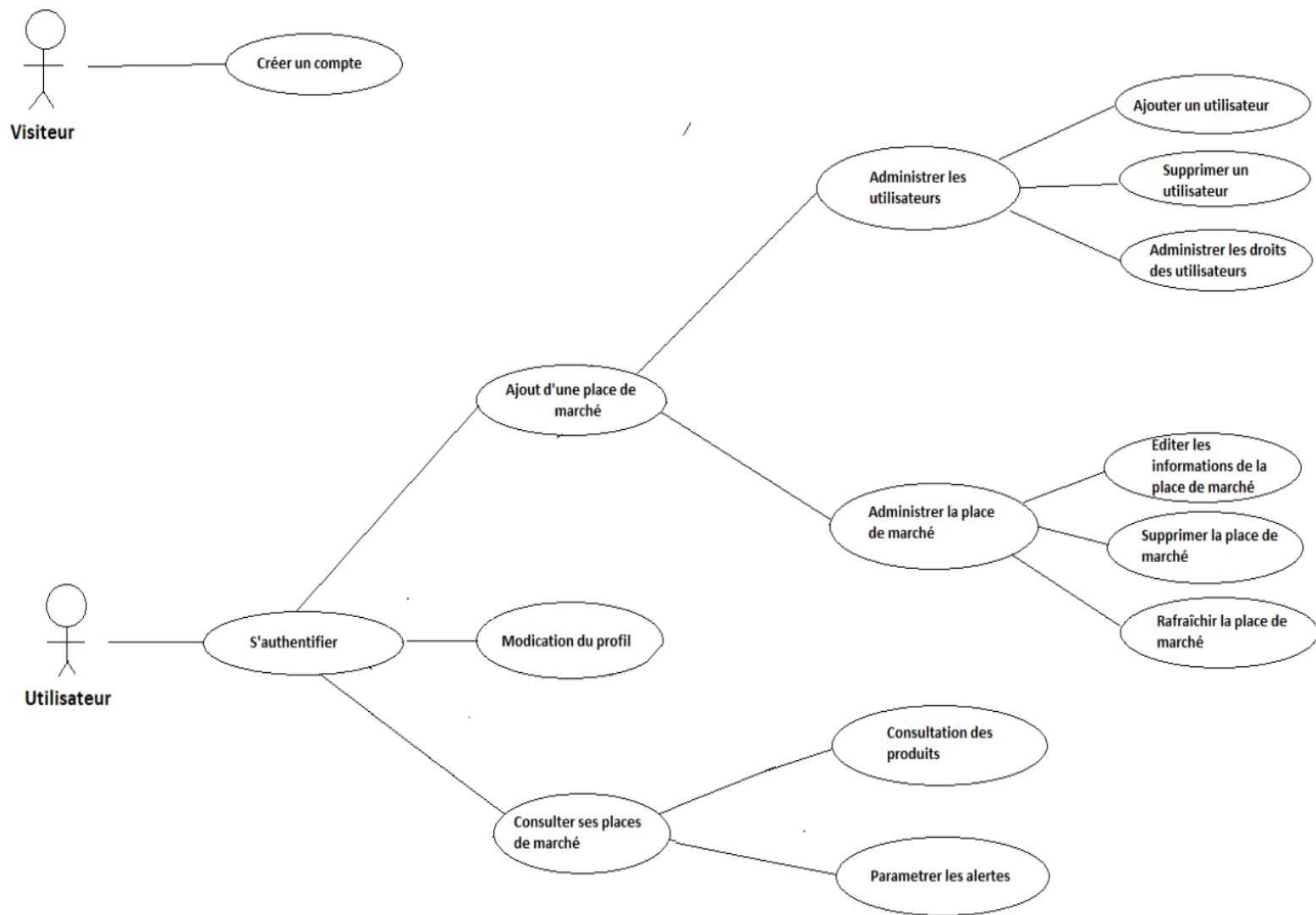
- Le diagramme de cas d'utilisation « use case »
- Le diagramme de séquence
- Le diagramme d'activité
- Le diagramme de classes

1.1) Le diagramme de cas d'utilisation

Le « Use Case » a pour but d'exprimer de façon claire les besoins de notre client. Il permet d'identifier les possibilités d'interaction entre le système et les acteurs (intervenants extérieurs au système), c'est-à-dire, toutes les fonctionnalités que doit fournir le système. Il permet aussi de définir les limites.

Afin d'avoir une bonne visibilité sur ces diagrammes, nous avons découpé ces diagrammes en 4 packages indépendants, tous nécessitant une connexion de la part de notre utilisateur.

Fig 7 : Diagramme de cas d'utilisation

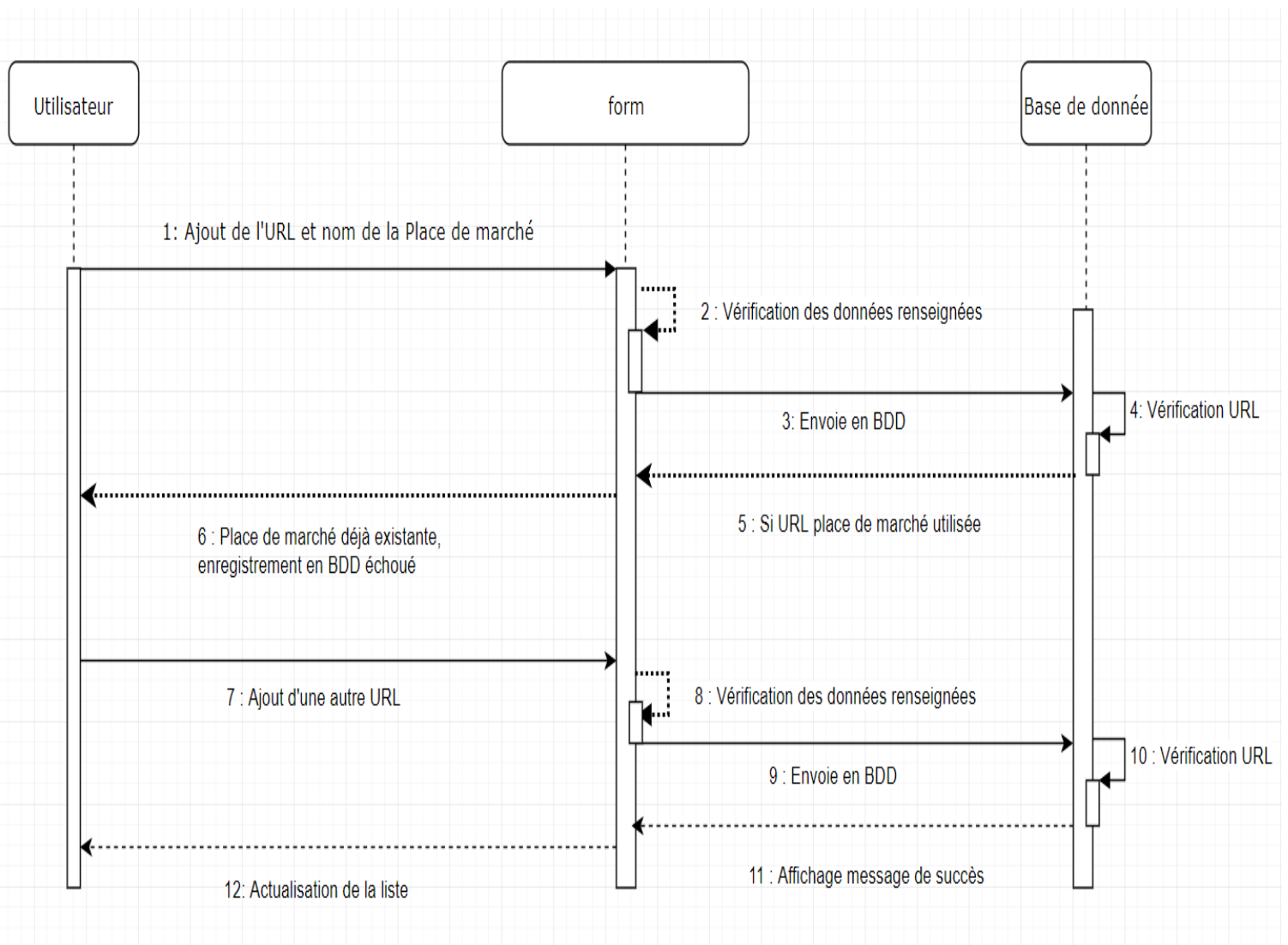


1.2) Le diagramme de séquence

Le diagramme suivant que nous avons réalisé est le diagramme de séquence. Il permet pour une situation donnée, de voir quelles vont être les interactions entre l'utilisateur et le système et ce de façon chronologique appelé aussi ligne de vie.

Les périodes d'activités de notre système correspondent aux rectangles, et chacun des acteurs communiquent par le biais de messages. Dans le cas de cet exemple qui correspond à la création et ajout d'une nouvelle place de marché sur un espace personnel de l'application par un utilisateur préalablement authentifié avec le renvoi d'un message d'erreur si l'URL renseignée est déjà utilisée, il n'y a que des messages simples : d'autres types de messages comme les messages asynchrones n'auraient pas eu d'utilité particulière étant donné que les vérifications doivent être faites avant de pouvoir continuer.

Fig 8 : Diagramme de séquence : Ajout d'une nouvelle place de marché sur l'application



Description du diagramme de séquence d'ajout d'une place de marché :

Avant de pouvoir ajouter une place de marché à son espace personnel, l'utilisateur devra avant tout s'identifier sur la plateforme via son adresse email et son mot de passe.

Une fois l'utilisateur connecté, il rejoindra son espace personnel (Dashboard) et ainsi cliquer sur « Ajouter une nouvelle place de marché » et renseigné le nom de la boutique ainsi que son URL disponible sur le site E-commerce Cdiscount.

Si l'utilisateur en question rentre une URL de boutique déjà enregistré en Base de données, un message d'erreur lui sera transmis et l'insertion en BDD échouera, sinon, l'insertion en BDD sera effectuée et l'utilisateur sera alors redirigé vers sa page d'accueil.

Etape 1 : Insertion du nom de la boutique et de l'URL dans le formulaire

Etape 2 : Vérification avec Regex du format de L'Url

Etape 3 : Envoie en BDD des informations du formulaire

Etape 4 : Vérification en BDD de doublon

Etape 5 : Url déjà existante, insertion échoué

Etape 6 : Envoie d'un message d'erreur

Etape 7 : L'utilisateur s'il le souhaite, renseigne une autre URL d'une autre boutique

Etape 8 : Vérification avec Regex du format de L'Url

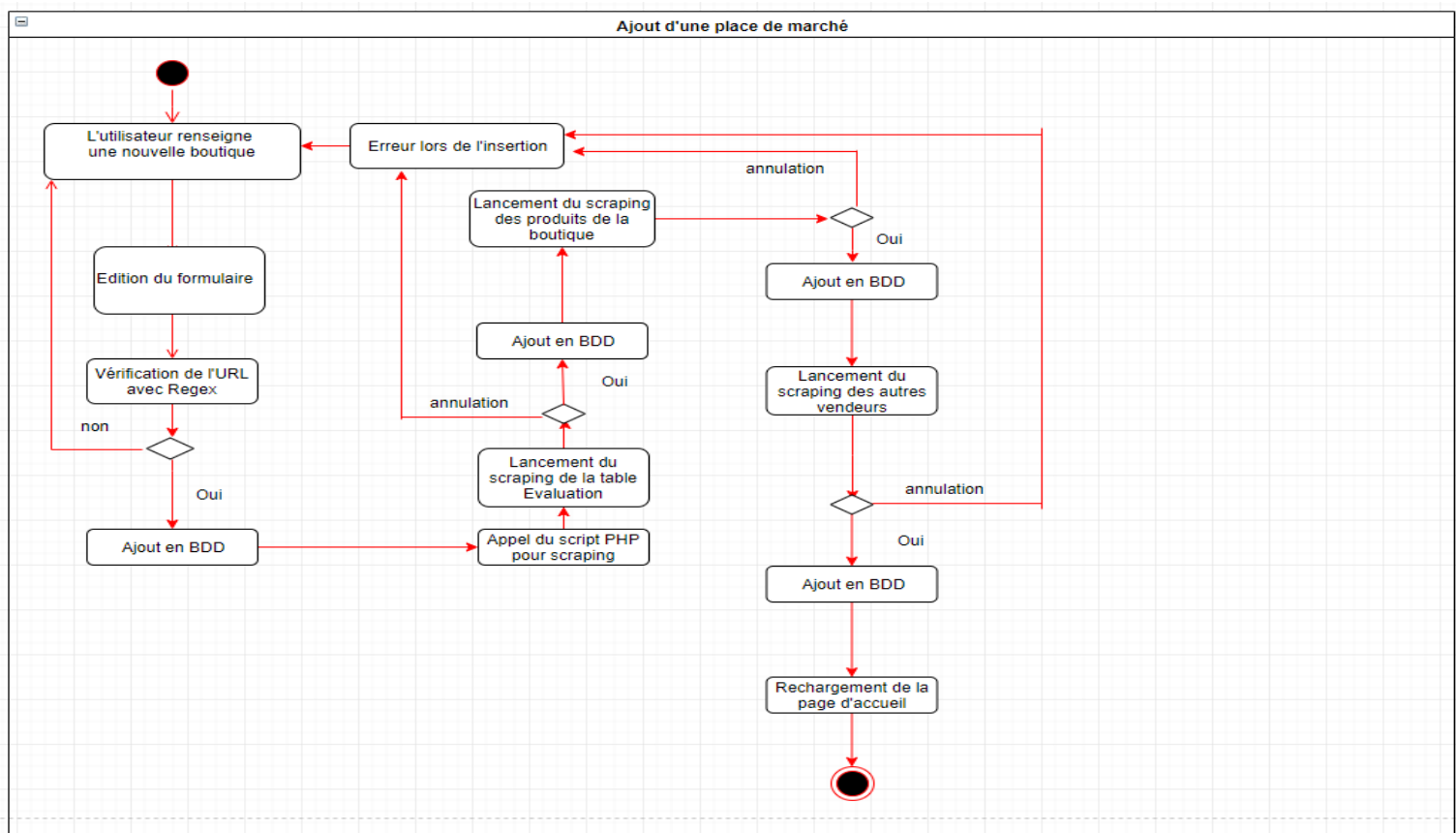
Etape 9 : Envoie en BDD des informations

Etape 10 : Url non existante, Insertion effectuée en BDD

Etape 11 : Affichage d'un message de succès sur le formulaire

Etape 12 : Rechargement de la page et affichage de la nouvelle Boutique sur la page d'accueil.

1.3) Le diagramme d'activité



Le diagramme d'activité permet de représenter le déclenchement d'évènements en fonction des états du système. Ils permettent ainsi de représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation. Il nous a permis de modéliser les différents traitements que le système effectue pour une situation donnée, et de visualiser la logique derrière les choix qu'il peut effectuer.

Description du cas d'utilisation « Ajout d'un nouvel utilisateur dans la boutique »

Afin de décrire les scénarios d'utilisation de certaines fonctionnalités, nous les avons détaillés de manière textuelle, notamment pour qu'elles soient plus aisées à comprendre pour notre client. Ceci nous a aussi permis de réfléchir à comment intégrer une fonctionnalité à l'application de façon à ce qu'elle soit facilement accessible.

- Nom : « Ajout d'un nouvel utilisateur à une boutique ».
- Acteur principal : l'utilisateur (créateur).
- Objectif : permettre l'utilisateur d'ajouter un nouvel utilisateur à sa boutique.
- Description : L'utilisateur ajoute un nouvel utilisateur dans la base de données FirstSeller.
- Préconditions : L'utilisateur doit être authentifié sur le site (en cas d'utilisation « S'authentifier »).
- Scénario nominal :
 1. L'utilisateur sélectionne dans le menu bandeau le lien « Utilisateur ».
 2. Le système renvoie la liste de tous les utilisateurs contenus dans la base de données, non présent sur cette boutique.
 3. L'utilisateur clique sur le bouton « ajouter ».
 4. Le système envoie le formulaire d'ajout d'un utilisateur.
 5. L'utilisateur renseigne les champs du formulaire et le soumet.
 6. Le système vérifie que les champs sont correctement renseignés.
 7. Le système enregistre dans la base de données les informations du formulaire.
 8. Le système redirige vers la page « Utilisateur » mise à jour.
- Nom : « Erreur lors de l'envoi du formulaire ».
- Acteur principal : L'utilisateur authentifié.
- Description : Le scénario commence au point 4 du scénario nominal.

- Scénario d'exception :

4.1. Le système vérifie les champs renseignés.

4.2. Le système retourne une erreur, dans le cas où le champ « Utilisateur » ne remplit pas les conditions d'ajout dans la base de données.

- Retour au scénario nominal :

5.1. Le système redirige sur le formulaire d'ajout d'un utilisateur avec un message d'erreur.

1.5) Le diagramme de classes

Pour réaliser des diagrammes UML corrects et représentatifs de notre application, nous avons dû respecter les contraintes imposées par les composants de Symfony et par Symfony lui-même, surtout en ce qui concerne les entités et le nommage des méthodes, les namespaces et leur règles de nommages et les formulaires et éventuelles surcharges de champs et ou de validateurs.

Tous les diagrammes UML sont utiles pour le développement d'une application, du diagramme des cas au diagramme des paquetages qui peuvent aussi servir à y voir plus clair dans la structuration des bundles.

Je souhaite ici mettre en avant l'architecture MVC du Framework Symfony au travers du diagramme de classe de l'action « Ajout d'un nouvelle utilisateur à la boutique ».

Dans l'exemple ci-dessous, le **Kernel** reçoit la requête du client, il interroge le routage dans le but de savoir quel contrôleur il doit invoquer.

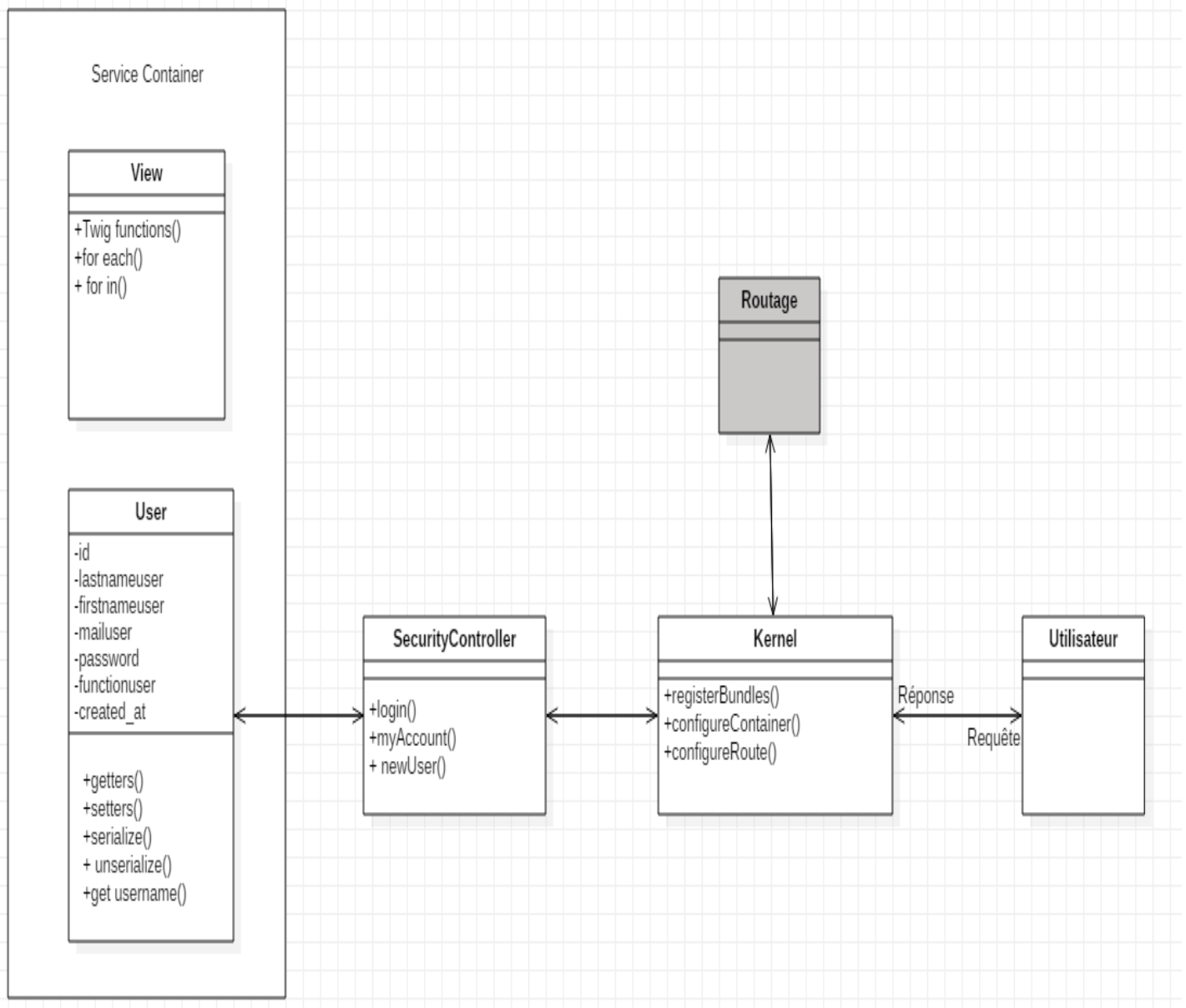
Une fois le contrôleur invoqué, le **Kernel** attend de lui qu'il retourne une réponse HTTP, peu importe la manière dont elle est générée.

De ce fait, le contrôleur est libre d'invoquer les différents services comme le modèle ou la vue, mis à disposition au sein du Service Container. De ces différentes interactions naît une réponse HTTP, qui est renvoyé au **Kernel**. Finalement, le **Kernel** transmet cette réponse au client.

Les services ne sont pas des concepts abstraits : chaque service est un Objet PHP. Le service Container est la seule entité mise à disposition du contrôleur pour l'aider à générer sa requête HTTP, c'est donc un élément central de l'application.

En installant Symfony, nous nous rendrons compte qu'il est préconfiguré avec un certain nombre de services, chacun répondant à un besoin particulier (gérer les templates, communiquer avec des bases de données, etc...).

Fig 9 : Diagramme de classe d'un formulaire de création d'un nouvel utilisateur



V – Conception et architecture de l'application

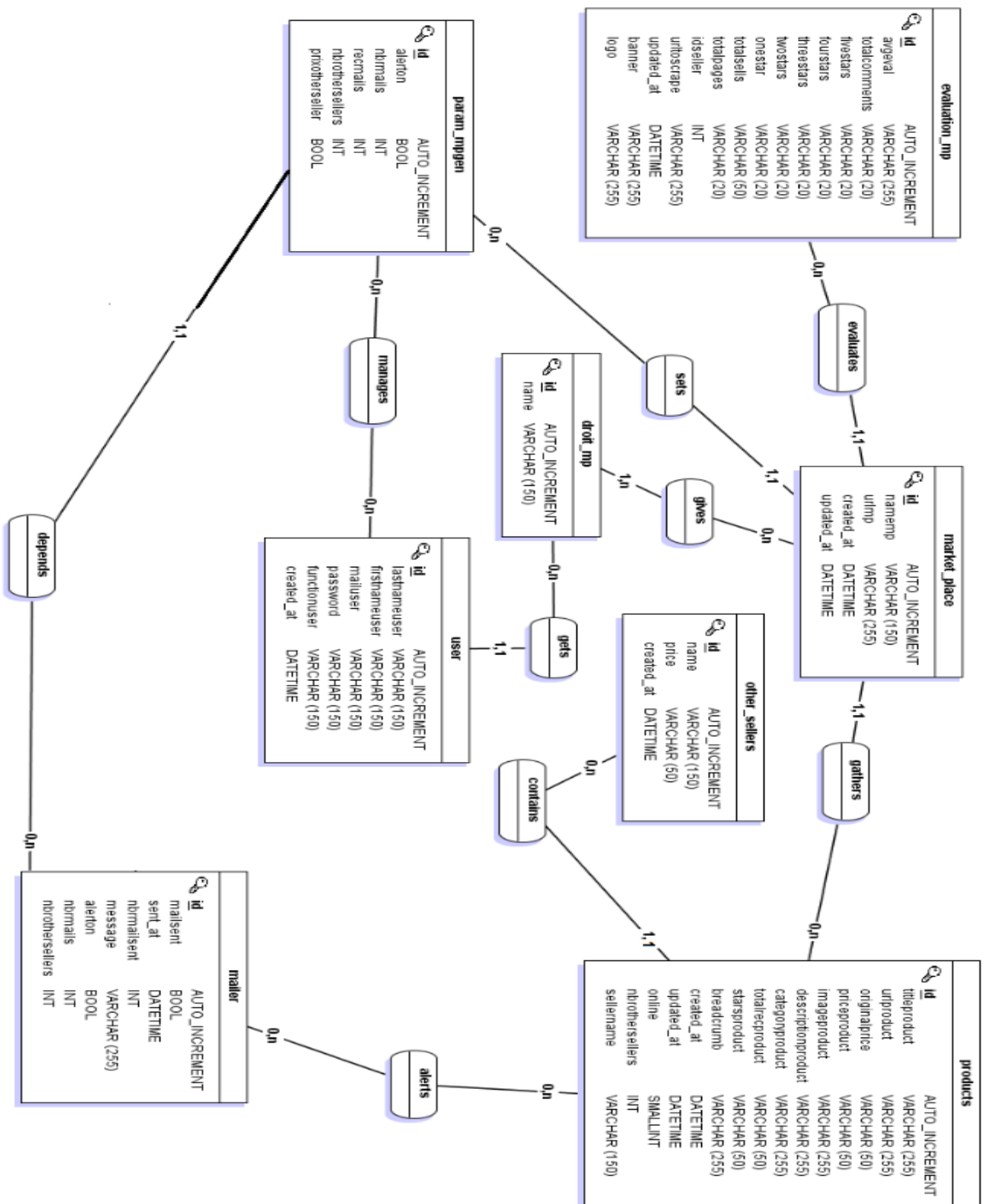
1. Les modèles physiques des données

1.1) Le modèle conceptuel des données (MCD)

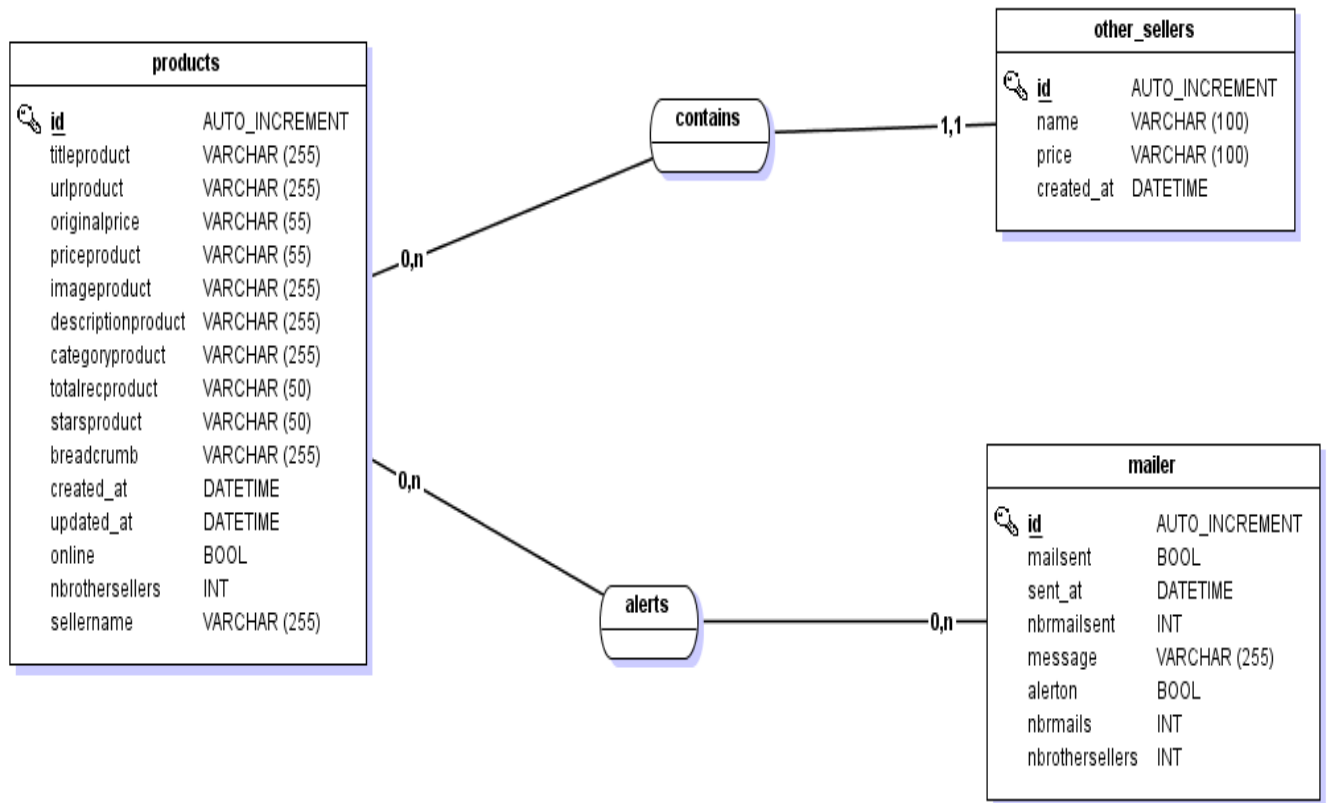
Le modèle conceptuel des données (MCD) a pour but d'écrire de façon formelle les données qui seront utilisées par le système d'information. Il s'agit donc d'une représentation des données, facilement compréhensible, permettant de décrire le système d'information à l'aide d'entités.

Le MCD appartient à la méthodologie Merise (Méthode d'Etude et de Réalisation Informatique pour les Systèmes d'Entreprises), qui permet notamment de concevoir un système d'information d'une façon standardisée et méthodique.

Fig 10 : illustration du MCD



Chaque ensemble d'objet ayant les mêmes caractéristiques sont représenté par une entité, et chaque donnée contenue dedans est appelé propriété. Une association correspond à un lien entre une ou plusieurs entités.



Dans l'exemple ci-dessus, l'association « alerts » joints les tables « products » et « mailer » et se compose par de deux cardinalités (0,N et 0,N) qui se traduisent par :

- Un « produit » peut avoir un ou plusieurs « mailer »
- Un « mailer » peut aussi avoir un ou plusieurs « produits »

Pour le second cas, l'association « contains » joints les tables « products » et « other_sellers » et se compose des cardinalités (1,1 et 0,N) qui se traduisent par :

- Un « produit » peut avoir un ou plusieurs « other_sellers »
- Un « other_sellers » ne peut être associé qu'à un seul « produit »

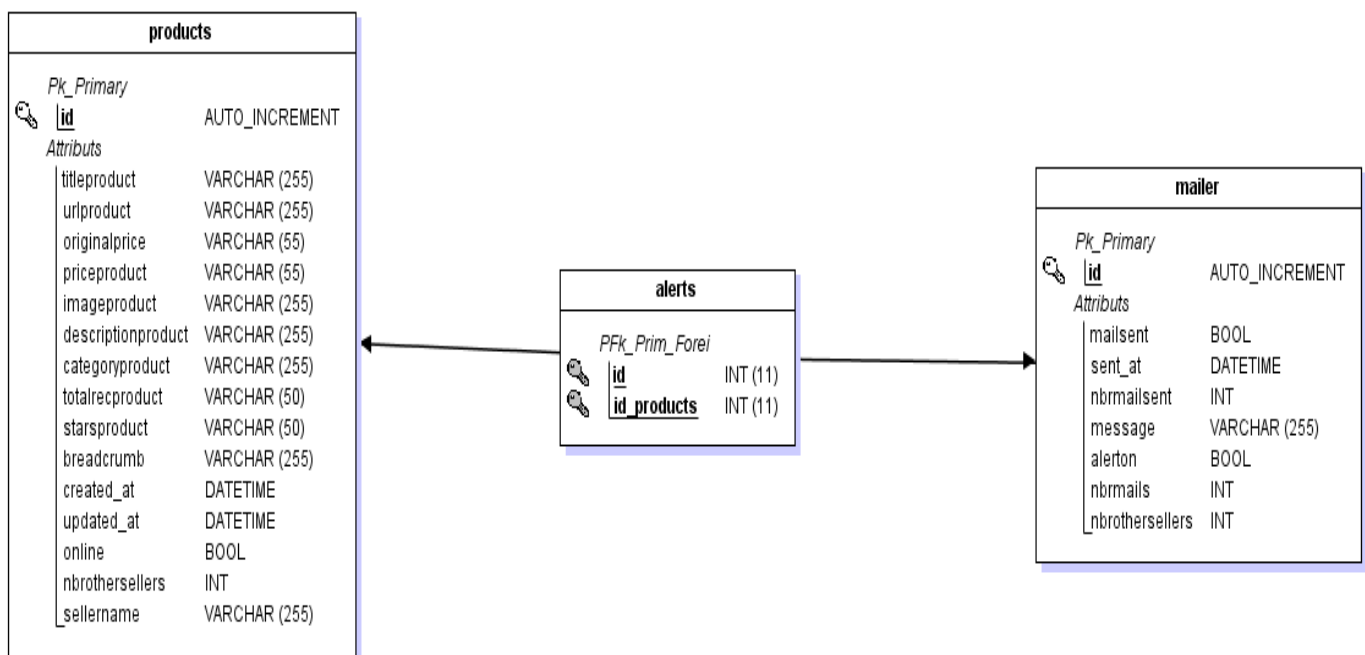
1.2) Le modèle logique des données (MLD)

Afin de modéliser graphiquement comment seront stockées nos données dans la base de données, nous avons créé le MLD afin de créer par la suite nos tables au niveau physique.

Pour passer du MCD au MLD, il y a trois règles à respecter :

1. Toute entité du MCD devient une relation dans laquelle les attributs deviennent des colonnes. L'identifiant de cette entité constitue alors une clé primaire de la relation (qui permet de la caractériser à elle seule).
2. Une association binaire (de cardinalité **1-N**) disparaît au profit d'une clé étrangère dans la relation côté **0,1** ou **1,1** qui référence la clé primaire de l'autre relation. Cette clé étrangère ne peut être nulle dans le cas d'une cardinalité **1,1**.
3. Une association binaire de type **N-N** devient une table supplémentaire d'association dont la clé primaire est constituée des deux clés étrangères.

Ci-dessous, vous trouverez l'exemple d'association des tables « products » et « mailer » avec une relation **N-N** :

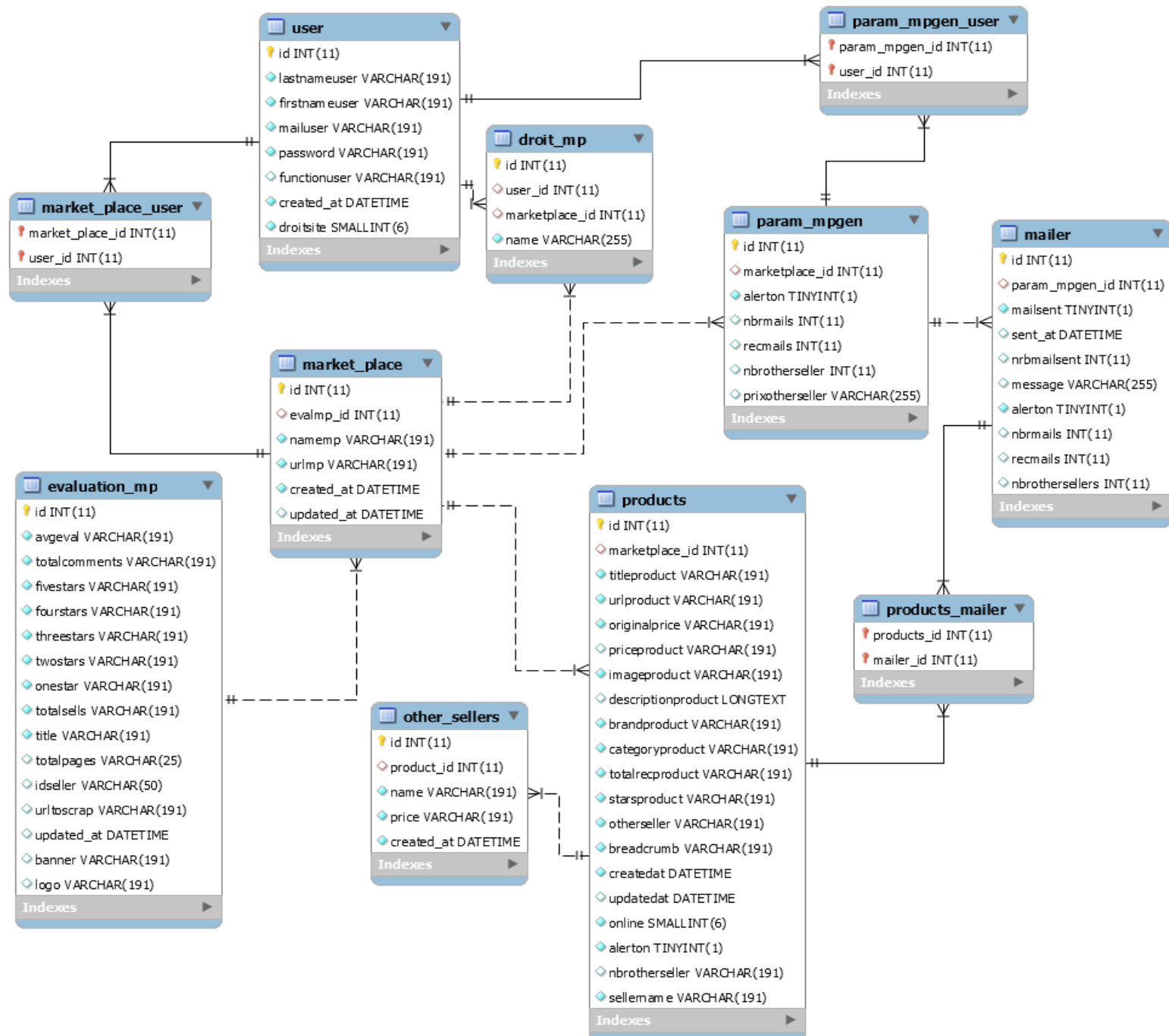


Si la relation est de type « **plusieurs à plusieurs** » ou **N-N**. Dans le MLD, la relation devient alors une **nouvelle table**, elle-même en relation avec les deux tables produites par les deux entités. Une telle table est dite **table de correspondance**, ou encore **table de liaison**, **table de jonction**, **table d'association**, etc.

Elle ne contient pas à proprement parler des données : son rôle est d'organiser les rapports entre les éléments des tables qui, elles, les contiennent. Une table de jonction contiendra uniquement des propriétés correspondant aux clés primaires des deux entités.

La table de jonction va permettre d'associer tout élément de « products » à tout élément de « mailer » autant de fois que souhaité. Un même produit pourra ainsi être mis en rapport avec plusieurs mailer, et un même mailer avec plusieurs produits.

Fig 11 : illustration du MLD



1.3) Le dictionnaire de données

Un dictionnaire des données est une collation des métadonnées ou de données de référence nécessaires à la conception d'une base de données relationnelle.

- Entités : les entités représentent ici le nom des tables contenues dans la base de données.
- Code mnémorique : il s'agit d'un libellé désignant une donnée (exemple « id » qui représente l'identifiant).
- Type de données :
 - ⇒ A ou Alphabétique : désigne la donnée qui est uniquement composé de caractère alphabétique (de 'A' à 'Z' et de 'a' à 'z').
 - ⇒ N ou Numérique : lorsque la donnée est composée uniquement de nombres (entiers ou réels)
 - ⇒ AN ou Alphanumérique : lorsque la donnée peut être composée à la fois de caractère alphabétiques et numériques.
 - ⇒ Date : lorsque la donnée est une date (au format AAAA/MM/JJ HH:MM:SS).
 - ⇒ Logical (boolean) : lorsque la donnée est un type de variable à deux états : vrai ou faux (*true or false*).
- Taille : elle s'exprime en nombre de caractères ou de chiffres. Dans le cas d'une date au format AAAA/MM/JJ HH:MM:SS, on compte le nombre de caractères, soit un total de 18 caractères.
- Désignation : il s'agit d'une description de la donnée utilisée.

Fig 12 : Exemple de dictionnaire de données de la table mailer

Entités	Code mnémorique	Type	Taille	Désignation
Mailer	Id	N	10	Identifiant du mailer
	mailsent	LOGICAL	1	Email envoyé
	sent_at	DATE	18	Date d'envoi
	nbrmailsent	N	10	Nombre d'emails envoyé
	message	AN	191	Message envoyé
	alerton	LOGICAL	1	Etat de l'alerte
	nbrmails	N	1	Nombre d'emails maximum
	recmails	N	1	Recurrence en heure des envoies
	nbrothersellers	N	10	Nombre de vendeur minimum

2. Conception de la base de données

2.1) Conception avec PhpMyAdmin et MySQL

Pour la gestion de la base de données, j'ai utilisé PhpMyAdmin. Ci-dessous, voici une représentation de la table « products » dans le système de gestion de base de données de PhpMyAdmin.

Fig 13 : Exemple de la composition de la table “ evaluationMP ” dans PHPMYAdmin

Serveur: MySQL:3306 » Base de données: firstsellerdiscount » Table: evaluation_mp										
#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action	
<input type="checkbox"/>	1 id	int(11)			Non	Aucun(e)		AUTO_INCREMENT		
<input type="checkbox"/>	2 avgeval	varchar(255)	utf8mb4_unicode_ci		Non	Aucun(e)				
<input type="checkbox"/>	3 totalcomments	varchar(255)	utf8mb4_unicode_ci		Non	Aucun(e)				
<input type="checkbox"/>	4 fivestars	varchar(255)	utf8mb4_unicode_ci		Non	Aucun(e)				
<input type="checkbox"/>	5 fourstars	varchar(255)	utf8mb4_unicode_ci		Non	Aucun(e)				
<input type="checkbox"/>	6 threestars	varchar(255)	utf8mb4_unicode_ci		Non	Aucun(e)				
<input type="checkbox"/>	7 twostars	varchar(255)	utf8mb4_unicode_ci		Non	Aucun(e)				
<input type="checkbox"/>	8 onestar	varchar(255)	utf8mb4_unicode_ci		Non	Aucun(e)				
<input type="checkbox"/>	9 totalsells	varchar(255)	utf8mb4_unicode_ci		Non	Aucun(e)				
<input type="checkbox"/>	10 title	varchar(255)	utf8mb4_unicode_ci		Non	Aucun(e)				
<input type="checkbox"/>	11 totalpages	varchar(25)	utf8mb4_unicode_ci		Oui	NULL				
<input type="checkbox"/>	12 idseller	varchar(50)	utf8mb4_unicode_ci		Oui	NULL				
<input type="checkbox"/>	13 urltoscrap	varchar(255)	utf8mb4_unicode_ci		Oui	NULL				
<input type="checkbox"/>	14 updated_at	datetime			Oui	NULL				
<input type="checkbox"/>	15 banner	varchar(255)	utf8mb4_unicode_ci		Oui	NULL				
<input type="checkbox"/>	16 logo	varchar(255)	utf8mb4_unicode_ci		Oui	NULL				

Fig 14 : Exemple de la table “evaluationMP” dans PHPMYAdmin

Server: MySQL 3306 Base de données: firstsellerdiscount Table: evaluation_mp

Parcourir

Structure

SQL

Rechercher

Insérer

Exporter

Importer

Privileges

Opérations

Déclencheurs

traitement en 0.0000 seconde(s)

Protéger [Éditer en ligne] [Éditer] [Exécuter SQL] [Créer le code source PHP] [Actualiser]

25

Filter les lignes : Chercher dans cette table

id	avgval	totalcomments	firststars	fourstars	threestars	twostars	onestar	totalseils	title	totalpages	idseller	urltoscrap	updated_at	banner	logo
1	4.4	56	34 avis	16 avis	1 avis	4 avis	1 avis	13636 Ventes	Plastimea 4	4	1024	https://www.cdiscount.com/impv-1024-Plastimea.html	2019-03-06 11:20:27	https://2.cdschd.com/sellerzone/Images/OKshop/pub...	https://2.cdschd.com/sel

Éditer

Copier

Supprimer

Exporter

25

Filter les lignes : Chercher dans cette table

Exporter

Afficher le graphique

Créer une vue

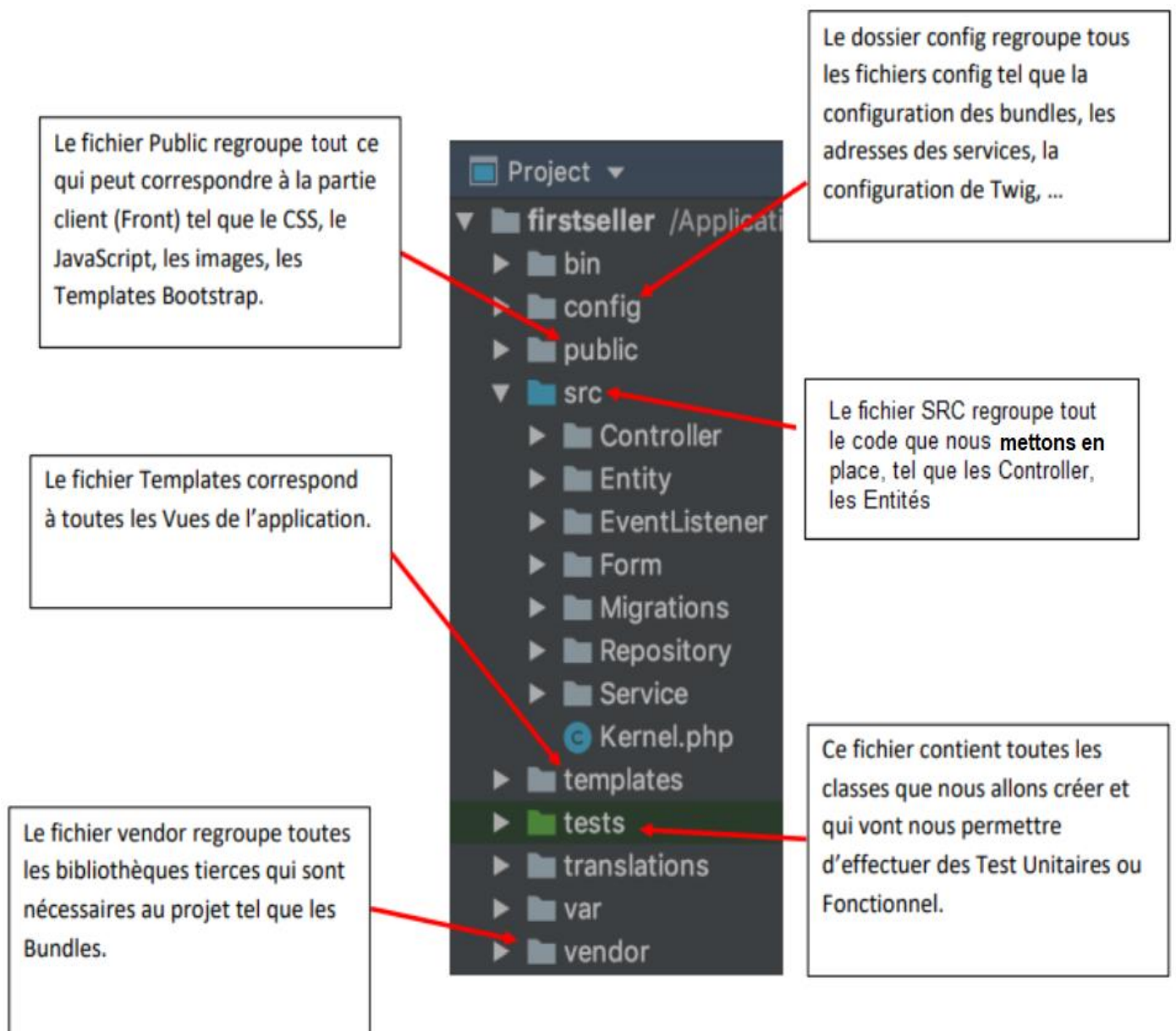
VI – Réalisation

1. Méthodologie

1.1) Structure d'un projet Symfony 4

Lorsque vous créez votre projet Symfony 4, vous obtiendrez l'arborescence suivante:

Fig 15 : Architecture d'un projet Symfony 4



1.2) Le modèle en couche MVC

Le modèle 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** ;

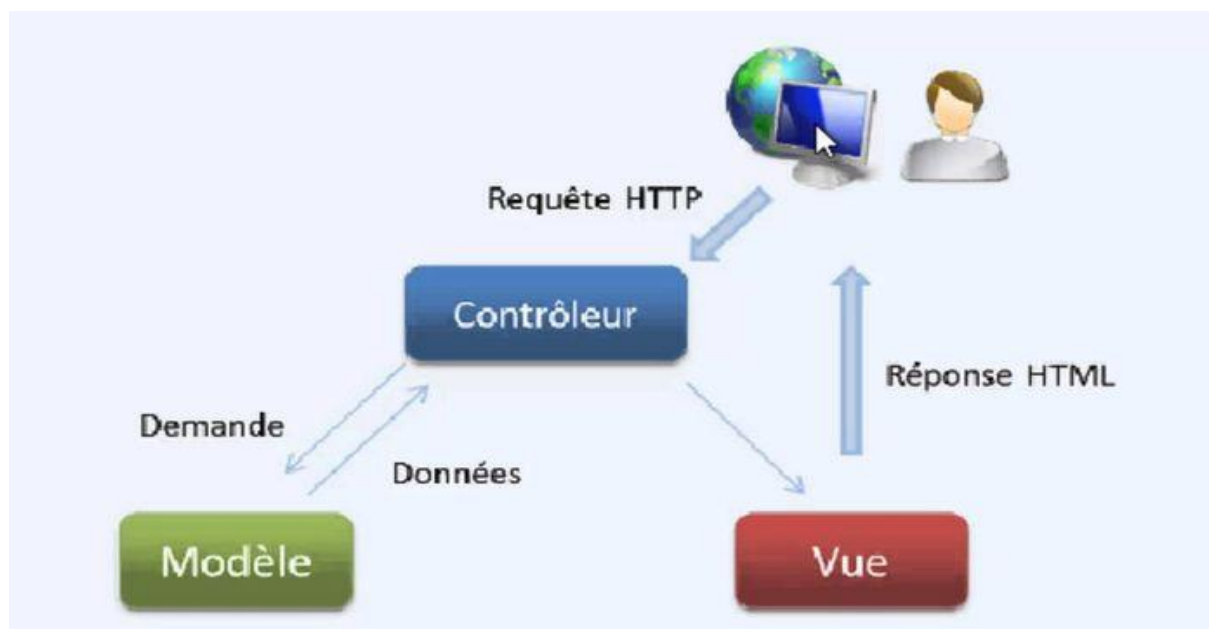
Ce modèle de conception (design pattern) a été imaginé à la fin des années 1970 pour le langage Smalltalk afin de séparer le code de l'interface graphique de la logique applicative. Il est utilisé dans de très nombreux langages : bibliothèques Swing et Model 2 (JSP) de Java, Frameworks PHP, ASP.Net,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

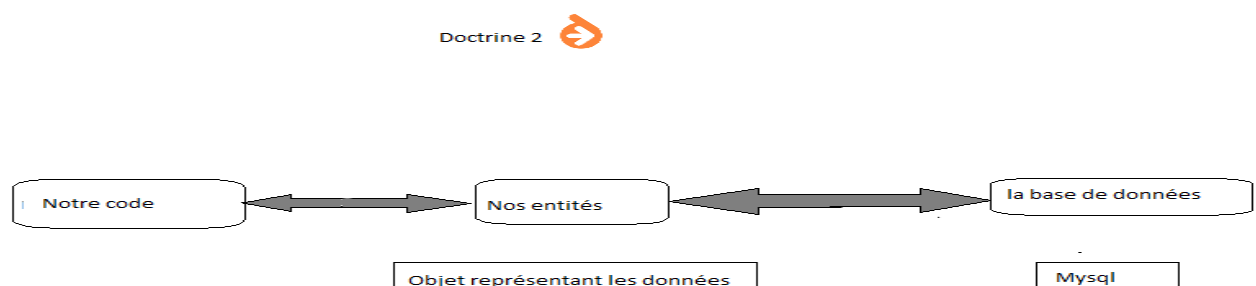
2) La couche métier

2.1) Les relations entre entités avec Doctrine

Une entité, c'est ce qui va nous permettre de décrire à travers une classe l'implémentation de notre logique métier. Cela correspond à la partie Modèle du modèle MVC.

La plupart des logiques métiers nécessitent un stockage en base de données, une persistance. Cette persistance se fait via un ORM. J'ai utilisé l'ORM (Object Relational Mapping) fourni par défaut avec Symfony qui est Doctrine. Doctrine est un data mapper. C'est ce qui nous permet de faire le lien entre notre code, nos entités et notre base de données comme illustré sur la figure ci-dessous :

Fig 16 : Schéma d'illustration Doctrine



Une des spécificités de Symfony avec Doctrine est qu'il ne reconnaît que trois formes de relations :

- One-To-One
- Many-To-One
- Many-To-Many

Dans l'exemple ci-dessous, nous avons la relation entre l'entité « products » et « otherseller » qui est d'ordre inversé **One-To-Many** pour l'entité « products » et **Many-To-One** dans l'entité « otherseller » :

Comme mentionné plus haut, en utilisant Doctrine, nous n'avons pas fait notre base de données à la main. Le MCD nous a permis de réaliser nos tables de données et de déterminer le type de relation entre entités.

Fig 17 : Exemple relation OneToMany de la table « products » et « othersellers »

```
/**
 * @ORM\OneToMany(targetEntity="App\Entity\OtherSellers", mappedBy="product", cascade={"persist", "remove"})
 */
private $otherSellers;
```

Fig 18 : Exemple relation ManyToOne de la table « othersellers » et « products »

```
/**
 * @ORM\ManyToOne(targetEntity="App\Entity\Products", inversedBy="otherSellers")
 */
private $product;
```

Il est nécessaire avec Doctrine de préciser la nature de la relation par l'annotation **Inversed By** et **Mapped By**, en effet, ce sont ces deux annotations qui vont établir la relation de dépendance entre les deux entités et assurer le « mapping » entre celles-ci.

Entre les entités *products* et *othersellers*, l'idée ici est que la table *othersellers* puisse avoir un produit et que chaque produit puisse avoir plusieurs tables *othersellers*. Nous avons donc plusieurs *othersellers* (Many) à lier (To) à une seul produit (One). On a donc une relation Many-To-One ou n..1.

3- Le repository

3.1) Les requêtes avec Doctrine

Le repository est l'endroit où sont regroupées les requêtes relatives à une entité, la majorité étant liées à la récupération des données (**SELECT**) mais peut également contenir des opérations de création, modification ou suppression. Chaque entité possède son propre **repository** matérialisé par une classe définie par le développeur.

Nous avons donc utilisé un **repository** pour récupérer l'entité « *products* » en fonction de la variable passée en paramètre. Ci-dessous les filtres de recherche des produits par boutique dans la page /mymyp.

Fig 19 : Exemple des filtres de recherche des produits

Produits en ligne

Filtres de recherches

Nom du produit recherché

Tous

Prix

Avis

Min

Max

Min

Max

Recommandations

Min

Max

Rechercher

Pour effectuer cette recherche, nous avons passé en paramètre dans le MPController, au niveau de la route un paramètre {mpid} qui est défini au niveau des requirements.

Fig 20 : Récupération du formulaire de recherche dans le Controller

```
95  /**
96   * @Route("/user/mymp/{mpid}",name="user.mymp.index")
97   * @param $idmp
98   * @param PaginatorInterface $paginator
99   * @param Request $request
100  * @return \Symfony\Component\HttpFoundation\Response
101  * @throws \Exception
102  */
103  public function myMP($mpid, Request $request){
104
105
106      $userId = $this->getUser()->getId();
107      //Récupération de la MP choisie pour affichage des produits
108      $myMP = $this->marketPlaceRepository->findMyMp($mpid);
109
110
111      if (count($myMP->getProducts())) {
112
113          $countProd = count($myMP->getProducts());
114
115      }else
116          $countProd = 0;
117
118
119
120      //Création du formulaire de recherche des produits par MP
121      $search = new \App\Entity\ProductSearch();
122      $formsearch = $this->createForm( type: ProductsSearchType::class,$search);
123      $formsearch->handleRequest($request);
124
125      //Récupération de la liste de produits du MP en question avec l'envoi du formulaire de recherche
126      $products = $this->productsRepository->findAllVisible($search,$mpid)->getResult();
127  }
```

Initialisation du formulaire de recherche des produits dans le contrôleur.

Fig 21 : Exemple du ProductsRepository correspondant au formulaire de recherche

```
class ProductsRepository extends ServiceEntityRepository
{
    public function __construct(RegistryInterface $registry)
    {
        parent::__construct($registry, Products::class);
    }

    public function findAllVisible(\App\Entity\ProductSearch $search, $idmp)
    {
        $query = $this->findVisibleQuery($idmp);

        if($search->getTitleproduct()){
            $query = $query
                ->andWhere('products.titleproduct LIKE :searchTitle')
                ->setParameter( key: 'searchTitle', value: "%".$search->getTitleproduct()."%")
                ->orderBy( sort: 'products.id', order: 'DESC');
        }
        if($search->getOtherseller()){
            $query = $query
                ->andWhere('products.nbbrotherseller = :otherseller')
                ->setParameter( key: 'otherseller', $search->getOtherseller())
                ->orderBy( sort: 'products.id', order: 'DESC');
        }
        if($search->getPricemin()){
            $query = $query
                ->andWhere('products.priceproduct >= :pricemin')
                ->setParameter( key: 'pricemin', $search->getPricemin())
                ->orderBy( sort: 'products.id', order: 'DESC');
        }
        if($search->getPricemax()){
            $query = $query
                ->andWhere('products.priceproduct <= :pricemax')
                ->setParameter( key: 'pricemax', $search->getPricemax())
                ->orderBy( sort: 'products.id', order: 'DESC');
        }
        if($search->getRecmin()){...}
        if($search->getRecmax()){...}
        if($search->getStarsmin()){...}
        if($search->getStarsmax()){...}
        return $query->getQuery();
    }
}
```

Pour créer un **repository** personnalisé, il faut étendre la classe de base **Doctrine\ORM\EntityRepository**. Ensuite, il doit être référencé au niveau de l'entité, au travers du paramètre repositoryClass de l'annotation **@ORM\Entity**.

Description :

- La commande **SELECT** permet de spécifier les entités à récupérer. Les entités à sélectionner sont séparées par des virgules et référencées par leur alias, qui est défini juste après la classe de l'entité.
- La commande **FROM** est semblable à celle du langage SQL, à la différence que la classe de l'entité est utilisée à la place du nom de la table, et que l'alias est obligatoire.
- La clause **WHERE** filtre les entités à récupérer grâce à des opérateurs de comparaison.

Vous trouverez ci-dessous un exemple de requête dans la table `MarketPlaceRepository` pour récupérer la liste des boutiques assignés à l'utilisateur connecté :

Fig 21 : Exemple d'une requête avec Doctrine

```
//SELECT * FROM `market_place` INNER JOIN user ON user.id = user.id INNER JOIN droit_mp
// ON droit_mp.user_id = user.id AND droit_mp.marketplace_id= market_place.id
public function findAllMyMps($userCoId)
{
    return $this->createQueryBuilder( alias: 'mp')
        ->innerJoin( join: 'mp.user', alias: 'user')
        ->andWhere('user.id = :id')
        ->setParameter( key: 'id', $userCoId)
        ->innerJoin( join: 'mp.droitMPs', alias: 'droitMPs')
        ->addSelect( select: 'droitMPs.name')
        ->andWhere('droitMPs.user =:user')
        ->setParameter( key: 'user', $userCoId)
        ->getQuery()
        ->getResult();
}
```

3.2) L 'EntityManager

Fig 22 : Formulaire création d'un utilisateur



L'EntityManager (Appelé objectmanager) est l'objet permettant d'effectuer les opérations liées à l'altération des données, à savoir les requêtes de type **INSERT**, **UPDATE** et **DELETE**, pour que Doctrine fasse la synchronisation en base de données, ces opérations doivent être gérées avec l'EntityManager.

Pour insérer une ligne dans une table de données présente en base, nous avons tout d'abord :

- Instancier l'entité correspondant à cette table.
- Ensuite, elles ont été injectées à travers un formulaire.
- Et nous avons ensuite demandé à Doctrine d'envoyer la requête d'insertion vers la base de données.

L'EntityManager (objectManager) est invoqué à deux reprises pour une insertion avec les méthodes **persist()** et **flush()**. La première méthode indique à Doctrine que l'entité passée en argument a subi des modifications, et la seconde lui demande de répercuter ces modifications en base de données.

4- Les formulaires

Les formulaires sont des éléments indispensables aux sites web : c'est le principal moyen par lequel les utilisateurs interagissent avec l'application.

Il est un exemple parfait du fameux patron de conception MVC. Ils sont affichés dans des pages (Couche Vue) et une fois soumis, ils sont généralement utilisés pour modifier des données (couche Modèle), tout ceci étant orchestré par le contrôleur.

Fig 23: Affichage du formulaire d'ajout d'un utilisateur



Nouvel utilisateur

Nom

Prénom

Email

Mot de passe

Fonction

Valider Annuler

4.1) L'objet « Form » et la création d'un formulaire

Afin d'illustrer notre développement, nous nous référerons au code source affiché précédemment (voir fig22). Pour concevoir notre formulaire, nous avons fait appel à l'objet Form, qui sera contenu dans la variable **\$form**. Il représente également l'ensemble des champs du formulaire, sous forme hiérarchique. L'objet représentant la requête HTTP, où l'objet de la couche Modèle y est « injecté ». Les tâches de traitements classiques (telles que la gestion de la soumission, la validation ou la création de la vue du formulaire) se font également à travers l'objet **Form**.

Cet objet **Form** comporte une méthode `handleRequest()`, prenant en paramètre l'objet Request (la requête HTTP courante).

Fig 24 : Illustration du code source du formulaire d'ajout d'un utilisateur

```
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;

class UserType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add( child: 'lastnameuser', type: TextType::class, [
                'attr' => [
                    'class' => "form-control"
                ],
                'label' => false,
                'required' => true
            ])
            ->add( child: 'firstnameuser', type: TextType::class, [
                'attr' => [
                    'class' => "form-control"
                ],
                'label' => false,
                'required' => true
            ])
            ->add( child: 'functionuser', type: TextType::class, [
                'attr' => [
                    'class' => "form-control"
                ],
                'label' => false
            ])
            ->add( child: 'mailuser', type: TextType::class, [
                'attr' => [
                    'class' => "form-control"
                ],
                'label' => false,
                'required' => true
            ])
            ->add( child: 'password', type: PasswordType::class, [
                'attr' => [
                    'class' => "form-control"
                ],
                'label' => false,
                'required' => true
            ])
    ];
    }

    /**
     * @param OptionsResolver $resolver
     */
    public function configureOptions(OptionsResolver $resolver)
    {
        $resolver->setDefaults([
            'data_class' => User::class,
        ]);
    }
}
```

5- L'interface graphique

5.1) Gestion de la responsivité avec Bootstrap

Nous avons implémenté dans le layout de base **Bootstrap** qui intègre un grid qui gère la responsivité. Ce grid est un système de colonnes (12 au total horizontalement) qui vient adapter le contenu d'une page en fonction de sa taille et ainsi gérer la responsivité.

Pour ceux ou celles qui n'auraient pas suivi, **Bootstrap est un framework open-source** HTML, CSS et même JavaScript. À son origine, deux génies de chez Twitter, *Mark Otto* et *Jacob Thornton*, qui en **2010**, ont développé un guide de style pour construire plus facilement leurs produits et optimiser leurs codes. Depuis, **Bootstrap a bien grandi** et aujourd'hui c'est toute une team de développeurs du monde entier qui s'affairent à faire évoluer le projet **utilisé sur 3,4% de l'Internet mondial !**

Plus en détail, Bootstrap est donc **une grosse librairie, qui regroupe principalement une multitude de classes CSS** qui vont te permettre, par exemple, de facilement donner du style au texte, aligner des blocs ou encore de construire rapidement une grille responsive. Mais ce n'est pas tout, aux classes CSS s'ajoutent **bien d'autres fonctionnalités JavaScript** pour créer des sliders, des menus déroulants ou encore des pop-up.

5.2) Utilisation de Twig

Le principe de Twig est d'établir un layout de page qui permet à toutes les pages du site d'hériter du layout de base.

Nous avons utilisé Twig pour la création des vues. Il offre de nombreux avantages, twig est un moteur de template. En voici les principaux avantages :

- Twig est un langage interprété, cela apporte plus de sécurité et évite les requêtes SQL au milieu du template.
- La séparation des responsabilités est plus claire, il n'est pas possible d'écrire en langage PHP dans les fichiers.
- Le début est plus simple, Twig fournit des fonctions qui permettent de trouver les variables et d'identifier les fichiers de template utilisés. Par exemple, la fonction `dump()`.

Un template contient des variables ou des expressions, qui sont remplacées par des valeurs lorsque le modèle est évalué, et des balises, qui contrôlent la logique du modèle.

Outre toutes les structures de contrôle de base permettant de gérer conditions et boucles, Twig offre également de très nombreuses autres possibilités comme l'inclusion ou l'héritage de templates. En effet, il suffit d'étendre le layout de base du fichier **base.html.twig** aux autres pages.

Fig 25 : Exemple de la page Twig de la liste des utilisateurs sur une boutique.

```
{#-----affichage des comptes-----#}
<div style="..." class="py-3">
  <table id="tableUsers" style="...">
    <thead>
      <tr>
        <th>Nom</th>
        <th>Prénom</th>
        <th>Fonction</th>
        <th>E-mail</th>
        <th>Privilege</th>
        <th>Date d'inscription</th>

        {% if (droituser) > 0 %}
          <th>Action</th>
        {% endif %}
      </tr>
    </thead>
    <tbody>
      {% for info in users %}
        <tr>
          <td>{{ info.user.lastnameuser }}</td>
          <td>{{ info.user.firstnameuser }}</td>
          <td>{{ info.user.functionuser }}</td>
          <td>{{ info.user.mailuser }}</td>
          <td>{% if ( info.name == 2 ) %Créateur{% elseif ( info.name == 1 ) %Administrateur{% else %Utilisateur{% endif %}</td>

          <td class="text-center">{{ info.user.createdat | date ('d/m/Y', "Europe/Paris") }}</td>
          <td class="pl-0 text-center pr-0">
            {% if (droituser) > 1 %}
              {% if(info.user.id) != userid %}
                {#-----BTN pour modification de l'user-----#}

                <a class="btn btn-primary btnEditmodal" data-toggle="modal" data-target="#modalEdit" data-information="{{ info.user.id }}"
                  style="..." href="{{ path('/mympp/mpid/users/{userid}', { mpid: mpid,userid: info.user.id}) }}"><i class="fas fa-user-edit fa-lg"></i></a>

                {#-----BTN de suppression de l'user du MP -----#}
                <form method="post" action="{{ path('/[mpid]/users/delete/{userid}', {userid: info.user.id, mpid: mpid}) }}" style="...">
              {% endif %}
            {% endif %}
          </td>
        </tr>
      {% endfor %}
    </tbody>
  </table>
```

Le moteur de Twig est très performant et extensible. Même s'il ne permet pas l'utilisation directe de PHP, il remplace de manière sécuritaire et parfois enrichies les principales structures de contrôle et la gestion des variables. Twig implémente aussi un système de fonctions, appelées « filtres » qui permettent de modifier une valeur avant de l'afficher.

Afin de se distinguer pleinement de PHP et HTML, la syntaxe de Twig se base sur un balisage par des accolades. Un appel de type **inline** se fera entre des doubles accolades et un appel de type block se fera entre une paire de balises délimitées par une accolade et un symbole de pourcentage. A noter que, comme en HTML, un appel de type block peut contenir d'autres appels de type **inline**. Il est aussi possible d'écrire des commentaires en les encadrant d'une accolade et d'un dièse.

Dans l'exemple ci-dessus, une variable \$users contenant les Utilisateurs de la place de marché sélectionnée est envoyé via le UserController au template **marketplace/users/index.html.twig** que nous récupérons ensuite côté **Vue**. Pour ce cas, nous récupérons un Array d'objet User, nous devons donc boucler à l'intérieur de cet Array avec « {% for info in users %} » et ensuite récupérer les différents éléments de cet objet avec « {{ info.user.lastname }} » pour faire apparaître le nom de l'utilisateur sur le template par exemple.

5.3) Le routage

Symfony 4 n'utilise pas la mise en relation entre l'URL et le système de fichiers pour servir les pages web, c'est le routage qui prend le relais. Le routage est un composant dont le rôle est de trouver l'action à exécuter pour une requête donnée et pour cela, il s'appuie sur un ensemble de règles de routage prédéfinies par le développeur.

- **Une requête une action**

Le rôle du routage est donc de sélectionner l'action à invoquer selon un ensemble de règles. Ces règles sont en rapport à la requête HTTP envoyée par le client, requête qui est la seule entité permettant au routage de pouvoir faire son choix.

Exemple de routage avec annotation sur le contrôleur MyMpController permettant de rediriger l'utilisateur vers la boutique choisie

```
/**
 * @Route("/user/mymp/{mpid}", name="user.mymp.index")
 * @param $mpid
 * @param Request $request
 * @return \Symfony\Component\HttpFoundation\Response
 * @throws \Exception
 */
public function myMP($mpid, Request $request) { ... }
```

```
<tr>
<td><a href="{{ path('/user/mymp/{mpid}', {mpid: mp.0.id}) }}">{{ mp.0.name }}</a></td>
<td><a href="{{ mp.0.urlmp }}">{{ mp.0.urlmp }}</a></td>
<td>{{ mp.0.createdAt|date('d/m/Y', 'Europe/Paris')}}</td>
<td>{% if (mp.name == 2) %}Créateur{% elseif (mp.name == 1) %}Administrateur{% else %}Utilisateur</td>
<td class="text-center">{% if (mp.name != 2) %}<a href="{{ path('mymp/{mpid}/quit', {mpid: mp.0.id}) }}">Quitter</a></td>
</tr>
```

Exemple de routage via « path » dans le template index.html.twig

6. Le système d'alerte avec PHP Mailer

6.1) Librairie PHPMailer

Historique

- PHPMailer fût initialement créé en 2001 par Brent R. Matzelle comme un projet sur [SourceForge.net](https://sourceforge.net).
- Marcus Bointon (coolbru sur SourceForge) et Andy Prevost (codeworxtech) prirent en charge le projet en 2004.
- Puis il est devenu un projet de l'incubateur Apache sur Google Code en 2010, géré par Jim Jagielski⁴.
- Marcus a créé son fork sur [GitHub](https://github.com).
- Jim et Marcus décident d'unir leurs forces et d'utiliser GitHub comme dépôt officiel de PHPMailer.

Introduction

PHPMailer est une bibliothèque de classes pour PHP qui fournit une collection de fonctions pour construire et envoyer des messages électroniques. PHPMailer supporte plusieurs façons d'envoyer des courriels : mail(), Sendmail, qmail ou directement aux serveurs **SMTP** (*Simple Mail Transfert Protocol*) tel que Gmail. Vous pouvez utiliser n'importe quelle fonction de courrier électronique **SMTP**, plusieurs destinataires via, CC, BCC, etc. En bref : PHPMailer est un moyen efficace d'envoyer des e-mails en PHP.

Pourquoi utiliser PHPMAILER au lieu de mail() ?

Pour au moins deux bonnes raisons :

- La première est que mail() s'appuie sur le sous-système de messagerie du serveur pour fonctionner. Cela signifie que si vous souhaitez modifier certains paramètres tels que le serveur SMTP ou les paramètres d'authentification, vous devez le faire à l'échelle du système.

C'est généralement une opération assez difficile, et à moins d'avoir un serveur dédié ou d'utiliser un environnement de développement PHP local, vous n'êtes probablement même pas autorisé à le faire.

Cela rend également presque impossible l'utilisation simultanée de configurations différentes, par exemple en utilisant plusieurs comptes SMTP et en basculant entre eux par programmation.

- La deuxième raison est que mail() n'offre aucune fonctionnalité avancée. mail() est parfait pour l'envoi de courriels simples en texte clair, mais il est très restrictif si vous avez besoin de faire autre chose que cela. Ajouter des pièces jointes ou envoyer des courriels HTML, par exemple, est très difficile avec mail(), alors qu'avec PHPMailer c'est juste une question d'une seule ligne de code.

Y a-t-il des alternatives à PHPMailer ?

Oui, il existe d'autres bibliothèques comme Zend Mail, SwiftMailer et Zeta Components Mail, mais PHPMailer est généralement le premier choix en raison de sa popularité.

Bien sûr, si vous êtes déjà familier avec une autre extension de messagerie et qu'elle fonctionne bien pour vous, vous pouvez vous y tenir.

Mais si vous voulez commencer à utiliser l'un d'entre eux et que vous devez choisir lequel choisir, PHPMailer est probablement le meilleur choix car c'est le plus utilisé.

D'autres extensions comme Zend Mail, SwiftMailer ou Zeta Components Mail sont probablement aussi bonnes que PHPMailer, mais regardez les résultats de recherche Google pour "php mail library" :

Fig 27 : Recherche « php mail library » sur google.com au 01/03/2019

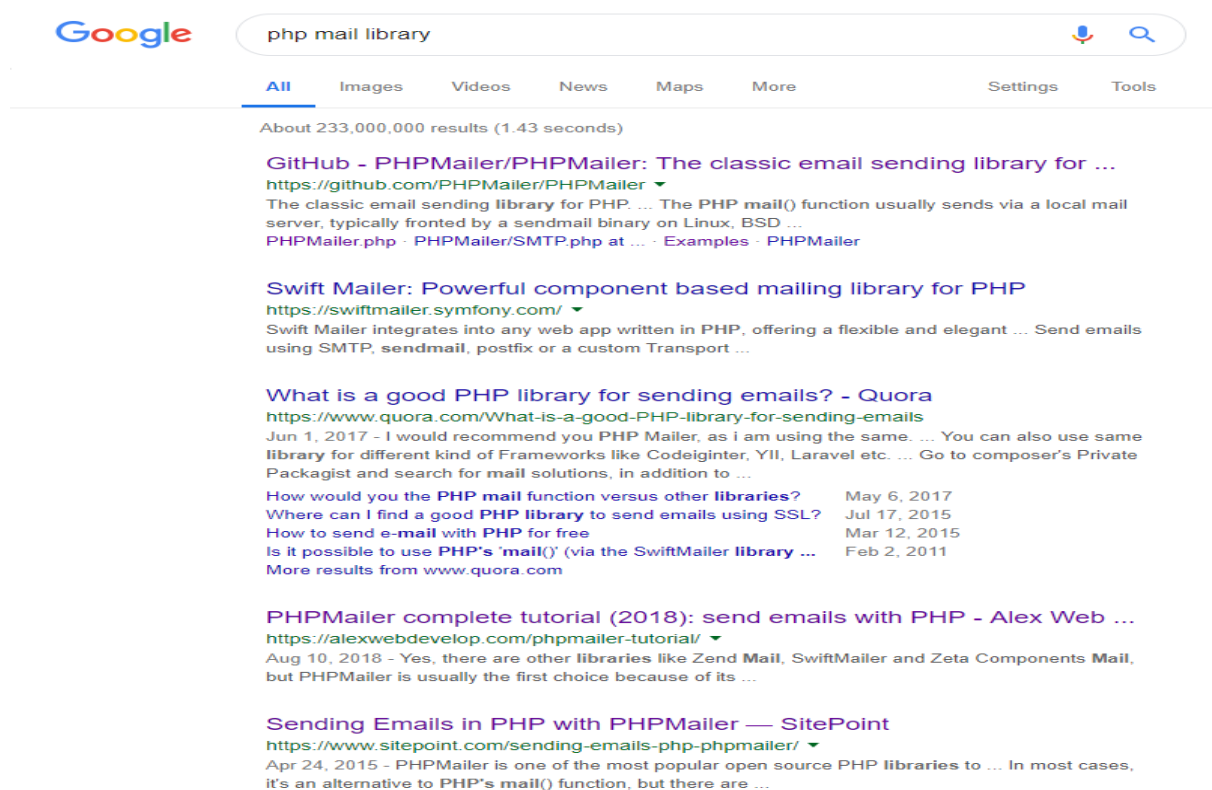
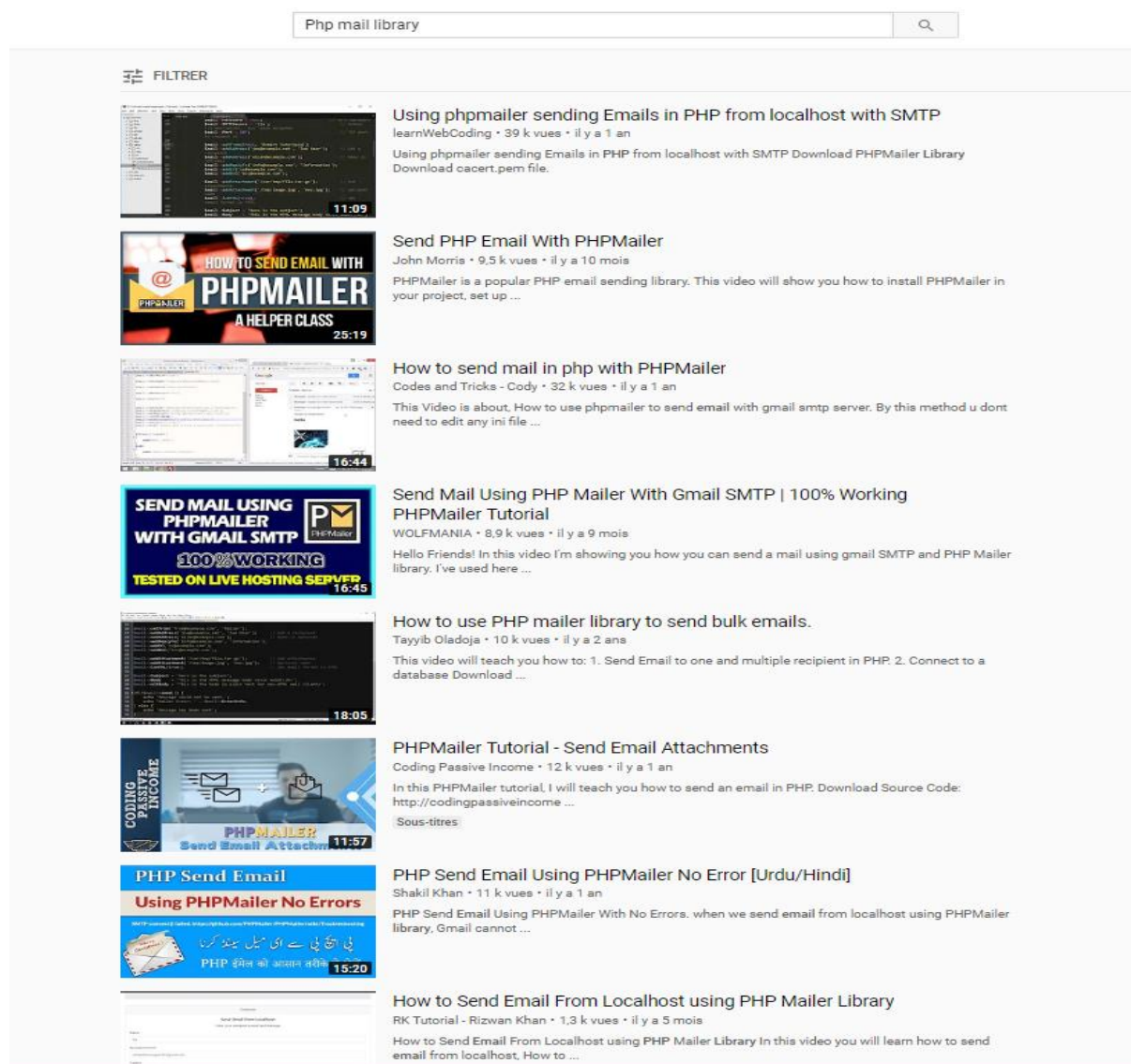


Fig 28 : Recherche « php mail library » sur youtube.com au 01/03/2019



Comme vous pouvez le voir clairement, PHPMailer domine les premiers résultats. Cela ne signifie pas nécessairement que PHPMailer est meilleur que les autres extensions, mais cela signifie que c'est le plus populaire.

La principale raison pour laquelle vous voulez opter pour la bibliothèque la plus utilisée est le support : plus un logiciel est largement utilisé, plus il est facile de trouver de l'aide et des exemples en ligne.

Voir source : <https://alexwebdevelop.com/phpmailer-tutorial/> ,
<https://github.com/PHPMailer/PHPMailer/wiki/Tutorial>,
<https://github.com/PHPMailer/PHPMailer>

6.2) Script PHP du mailer AlerterCdiscount

Fig 29 : Script d'envoi des alertes

```
1  <?php
2
3
4  require 'PHPMailer/src/Exception.php';
5  require 'PHPMailer/src/PHPMailer.php';
6  require 'PHPMailer/src/SMTP.php';
7
8  use PHPMailer\PHPMailer\PHPMailer;
9  use PHPMailer\PHPMailer\Exception;
10
11
12  define('GUSER', 'alerterfirstseller@gmail.com'); // GMail username
13  define('GPWD', 'demo31000'); // GMail password
14
15
16
17  //ON récupère la liste des produits correspondant aux critères d'alertes imposés par l'User
18  $listProd = mailerGenMP();
19
20  //On envoie cette liste à la fonction d'envoi d'email
21  getProductToMail($listProd);
22
23
24
25  //Function pour filtrer les produits avec l'alerte ON pour l'envoi des alertes
26  function mailerGenMP(){...}
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47  // Function qui retournera un tableau des produits avec les alertes pour l'envoi des mails
48
49  function getProductToMail($arrayProd){...}
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87  // Function d'envoi des emails avec PHPMailer
88
89  function smtpmailer($to, $message, $namemp, $producturl, $prodname) {...}
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625

```

Dans l'exemple ci-dessus, vous trouverez le script pour l'envoi d'alertes via PHPMailer.

On commence avec la commande « **require** » pour importer les scripts nécessaires à la configuration et l'envoi des emails via SMTP et un compte Gmail (voir ligne 4,5,6).

On utilise par la suite la déclaration « **use** » pour importer les classes présentes dans ces scripts (voir ligne 8,9) nécessaires à l'envoi de l'email.

Pour la configuration de notre SMTP, on définira les constantes nécessaires à la connexion de notre compte (voir ligne 12,13).

Ensuite, la fonction **mailerGenMP** (voir ligne 26) sera appelée pour récupérer tous les boutiques présentes dans notre Base de données. Une fois toutes les boutiques chargées, la fonction commencera à réunir les données nécessaires pour la vérification des différents critères d'alertes imposés par l'utilisateur en question.

Cette fonction triera tous les produits récupérés dans notre Base de données et correspondant aux critères imposés et retournera un tableau de produits.

La fonction **getProductToMail** (voir ligne 76) récupèrera ensuite ce tableau.

Son rôle est de vérifier les informations du mailer voir table en base de données **Mailer**.

- Envoyer le premier email d'alerte si aucun message n'a été envoyé, la fonction **smtpmailer** (voir ligne 436) sera appelée pour ce cas.
- Envoyer un Rappel via la fonction **rappelsmtpmailer** (voir ligne 506) si le premier email a été envoyé.

Les fonctions **smtpmailer** et **rappelsmtpmailer** récupéreront toutes les deux les paramètres suivant pour l'envoi de l'alerte à l'utilisateur.

- **\$to** : Adresse email de l'utilisateur
- **\$message** : Message à envoyer
- **\$namemp** : Intitulé de la boutique en question
- **\$producturl** : Lien du produit sur le site Ecommerce Cdiscount
- **\$prodname** : Intitulé du produit

Chacune de ces fonctions appelleront pour finir la fonction **cleanMailer** (voir ligne 574), qui s'occupe de nettoyer le mailer du produit en question. Elle prendra en paramètre **\$idmailer** (correspondant à l'identifiant **id** du mailer en Base de données) ainsi que **\$sentAt**, date au format YYYY-MM-DD HH:II:SS.

Cette fonction comparera la date et heure du dernier envoi, si cette valeur est supérieure à 24h, le mailer sera alors réinitialisé en base de données ce qui évitera de **spammer** les utilisateurs.

7. Python et sa librairie Scrapy

7.1) Introduction à Python

Python est un langage de programmation interprété, orienté objet, de haut niveau avec une sémantique dynamique. Ses structures de données intégrées de haut niveau, combinées au typage dynamique et à la liaison dynamique, le rendent très attractif pour le développement rapide d'applications, ainsi que pour l'utilisation comme langage de script ou de collage pour connecter des composants existants ensemble. La syntaxe simple et facile à apprendre de Python met l'accent sur la lisibilité et réduit donc le coût de maintenance du programme. Python supporte les modules et les paquets, ce qui encourage la modularité des programmes et la réutilisation du code. L'interpréteur Python et la vaste bibliothèque standard sont disponibles gratuitement sous forme source ou binaire pour toutes les plates-formes majeures, et peuvent être distribués librement.

Voici quelques-uns de ses avantages :

- Python peut être utilisé pour développer des prototypes, et rapidement parce qu'il est si facile à utiliser et à lire.
- La plupart des plates-formes d'automatisation, d'exploration de données et des grandes plates-formes de données reposent sur Python. En effet, c'est la langue de travail idéale pour les tâches générales.
- Python permet un environnement de codage plus productif que les langages massifs comme C# et Java. Les codeurs expérimentés ont tendance à rester plus organisés et productifs lorsqu'ils travaillent avec Python.
- Python est facile à lire, même si vous n'êtes pas un programmeur expérimenté. N'importe qui peut commencer à travailler avec la langue, il suffit d'un peu de patience et beaucoup de pratique. De plus, cela en fait un candidat idéal pour une utilisation au sein d'équipes de développement multi-programmeurs et de grandes équipes de développement.
- Il a une base de support massive grâce au fait qu'il est open source et développé par la communauté. Des millions de développeurs partageant les mêmes idées travaillent quotidiennement avec ce langage et continuent à améliorer les fonctionnalités de base. La dernière version de Python continue de recevoir des améliorations et des mises à jour au fur et à mesure que le temps passe. C'est un excellent moyen de travailler en réseau avec d'autres développeurs.

Il est utilisé quotidiennement dans les opérations du moteur de recherche Google, du site de partage de vidéos YouTube, de la NASA et du New York Stock Exchange. Ce ne sont là que quelques-uns des endroits où Python joue un rôle important dans le succès des entreprises, du gouvernement et des organisations à but non lucratif ; il en existe de nombreux autres.

Historique

- 1991 : Guido van Rossum travaille aux Pays-Bas sur le projet AMOEBA, un système d'exploitation distribué. Il conçoit Python à partir du langage ABC et publie la version 0.9.0 sur un forum Usenet
- 1996 : sortie de Numerical Python, ancêtre de numpy
- 2001 : naissance de la PSF (Python Software Foundation)
- Les versions se succèdent... Un grand choix de modules est disponible, des colloques annuels sont organisés, Python est enseigné dans plusieurs universités et est utilisé en entreprise...
- 2006 : première sortie de IPython
- Fin 2008 : sorties simultanées de Python 2.6 et de Python 3.0
- 2013 : versions en cours des branches 2 et 3 : v2.7.3 et v3.3.0

Source : <https://www.python.org/doc/essays/blurb/>,
<https://www.pythonforbeginners.com/learn-python/>

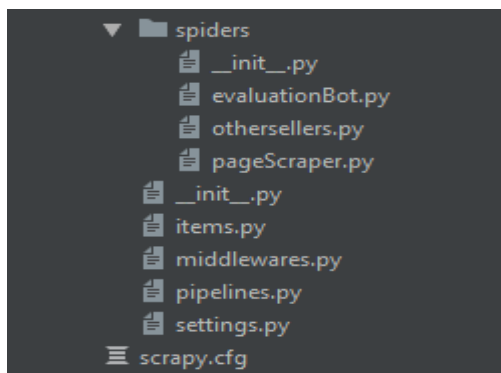
7.2) Python et les librairies utilisées

- **Python 3.7**
- **Pypi (pi-pee-eye)** : Le Python Package Index est un référentiel de logiciels pour le langage de programmation Python. PyPI vous aide à trouver et installer des logiciels développés et partagés par la communauté Python (<https://pypi.org/>).
- **Scrapy 1.6** : Un Framework open source et collaboratif pour extraire les données dont vous avez besoin des sites web. D'une manière rapide, simple et extensible (<https://scrapy.org/>, <https://doc.scrapy.org/en/latest/>).
- **Lxml** : un parser XML et HTML
- **Parsel** : une bibliothèque d'extraction de données HTML/XML écrite sur lxml.

- **w3lib** : une aide polyvalente pour gérer les URL et les encodages de pages Web.
- **Twisted** : Twisted est un moteur réseau événementiel écrit en Python (<https://twistedmatrix.com/trac/>).
- **cryptographie** et **pyOpenSSL**, pour répondre aux différents besoins de sécurité au niveau réseau.
- **PyMySQL** : interface pour se connecter à un serveur de base de données MySQL depuis Python. Il implémente l'API de base de données Python v2.4 et contient une bibliothèque client MySQL en Python pur.

7.3) Les scripts du « Scraper » de Cdiscount

Fig 30 : Structure d'un « spider »



Les "Spiders" sont des classes qui définissent comment un certain site (ou un groupe de sites) sera collecté, y compris comment effectuer le crawl (c'est-à-dire suivre les liens) et comment extraire des données structurées de leurs pages (c'est-à-dire scraper des éléments). En d'autres termes, les "spiders" sont l'endroit où vous définissez le comportement personnalisé pour parcourir et analyser les pages d'un site particulier (ou, dans certains cas, un groupe de sites).

(<https://docs.scrapy.org/en/latest/topics/spiders.html>)

Dans l'exemple ci-dessus, nous possédons 3 différentes araignées (Spiders) se trouvant dans le dossier spiders :

- **evaluationBot.py** : Se chargera de récupérer (scraper) les données présentes sur la page des recommandations d'une boutique sur le site E-Commerce de Cdiscount.
- **pageScraper.py** : Récupère le lien Url de la boutique en question pour collecter tous ses produits présent sur le site Internet. Cette araignée est chargée de « scraper » les différents éléments présents sur la page produit tel que sa description, son intitulé, son prix de vente, ses recommandations, etc...
- **othersellers.py** : Son rôle est de collecter le prix et nom de chaque vendeur présent sur la fiche produit d'une boutique enregistrée en base de données si ce produit en question possède plus d'un vendeur sur sa fiche produit.

- **items.py** : Pour définir un format de données de sortie commun, Scrapy fournit la classe Item. Les objets Item sont de simples conteneurs utilisés pour collecter les données scraper. Ils fournissent une API de type dictionnaire avec une syntaxe pratique pour déclarer leurs champs disponibles.

Dans notre cas, ce fichier contiendra 3 Items :

- ScaperCdiscountItem : Dictionnaire pour le spider pageScaper.py
- EvaluationBotItem : Dictionnaire pour le spider evaluationBot.py
- ScaperOtherSellers : Dictionnaire pour le spider othersellers.py

Fig 31 : Contenu fichier items.py

```
class ScaperCdiscountItem(scrapy.Item):
    # define the fields for your item here like:
    # name = scrapy.Field()
    urlproduct = scrapy.Field()
    titleproduct = scrapy.Field()
    sellername = scrapy.Field()
    originalprice = scrapy.Field()
    priceproduct = scrapy.Field()
    imageproduct = scrapy.Field()
    descriptionproduct = scrapy.Field()
    brandproduct = scrapy.Field()
    categoryproduct = scrapy.Field()
    totalrecproduct = scrapy.Field()
    starsproduct = scrapy.Field()
    otherseller = scrapy.Field()
    breadcrumb = scrapy.Field()
    online = scrapy.Field()
    createdat = scrapy.Field()
    updatedat = scrapy.Field()
    idmp = scrapy.Field()
    receivemails = scrapy.Field()
    nbrotherseller = scrapy.Field()

class EvaluationBotItem(scrapy.Item):
    note = scrapy.Field()
    ventes = scrapy.Field()
    avis = scrapy.Field()
    fiveStars = scrapy.Field()
    fourStars = scrapy.Field()
    threeStars = scrapy.Field()
    twoStars = scrapy.Field()
    oneStar = scrapy.Field()
    title = scrapy.Field()
    totalpages = scrapy.Field()
    updated_at = scrapy.Field()
    banner = scrapy.Field()
    logo = scrapy.Field()

class ScaperOtherSellers(scrapy.Item):
    idprod = scrapy.Field()
    name = scrapy.Field()
    price = scrapy.Field()
    createdat = scrapy.Field()
```

Dictionnaire :

Un **dictionnaire** en **python** est une **liste** mais au lieu d'utiliser des **index**, on utilise des **clefs**, c'est à dire des valeurs autres que numériques.

Par exemple, comme pour un carnet d'adresses où à un nom de famille (key) vous pouvez associer différentes informations (values): prénom, adresse, numéro de téléphone, entreprise, etc.

- **pipelines.py** : Après qu'un élément a été scrappé par une araignée, il est envoyé au pipeline qui le traite à travers plusieurs composants qui sont exécutés séquentiellement.
Chaque composant du pipeline (parfois appelé simplement "Item Pipeline") est une classe Python qui implémente une méthode simple. Ils reçoivent un article et exécutent une action sur lui, décidant également si l'article doit continuer à traverser le pipeline ou s'il doit être abandonné et ne plus être traité.

Les utilisations typiques des pipelines sont :

- nettoyage des données HTML
- la validation des données scrappées (vérification que les postes contiennent certaines zones)
- vérification des doublons (et suppression des doublons)
- stocker l'élément scrappé dans une base de données

Fig 32 : Exemple de script pipelines.py

```
# -*- coding: utf-8 -*-
import logging
import pprint
import pymysql
from pymysql.cursors import DictCursor

from pymysql import OperationalError
from pymysql.constants.CR import CR_SERVER_GONE_ERROR, CR_SERVER_LOST, CR_CONNECTION_ERROR
from twisted.internet import defer
from twisted.enterprise import adbapi

class ScraperdiscountPipeline(object):
    def __init__(self):
        self.connection = pymysql.connect("localhost","root","*****","firstsellerdiscount")
        self.cursor = self.connection.cursor()

    def process_item(self, item, spider):

        if spider.name == 'pageScraper':
            self.cursor.execute("""
            INSERT INTO products
            (titleproduct,urlproduct,originalprice,priceproduct,imageproduct,descriptionproduct,brandproduct,categoryproduct,
            totalrecproduct,starsproduct,otherseller,breadcrumb,online,createdat,updatedat,marketplace_id,alerton,nbrotherseller,sellername)
            VALUES
            (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)
            ON DUPLICATE KEY UPDATE
            titleproduct = VALUES(titleproduct),
            originalprice = VALUES(originalprice),
            priceproduct = VALUES(priceproduct),
            descriptionproduct = VALUES(descriptionproduct),
            brandproduct = VALUES(brandproduct),
            categoryproduct = VALUES(categoryproduct),
            totalrecproduct = VALUES(totalrecproduct),
            starsproduct = VALUES(starsproduct),
            otherseller = VALUES(otherseller),
            breadcrumb = VALUES(breadcrumb),
            updatedat = VALUES(updatedat),
            alerton = VALUES(alerton),
            nbrotherseller = VALUES(nbrotherseller);

            """,
            (item['titleproduct'],item['urlproduct'], item['originalprice'], item['priceproduct'],item['imageproduct'], item['descriptionproduct'],item['brandproduct'],
            item['categoryproduct'], item['totalrecproduct'], item['starsproduct'], item['otherseller'], item['breadcrumb'],item['online'],item['createdat'],
            item['updatedat'],item['idmp'],item['receivemails'], item['nbrotherseller'],item['sellername']))
            self.connection.commit()
            return item
```

- **settings.py** : Les paramètres de Scrapy vous permettent de personnaliser le comportement de tous les composants Scrapy, y compris le noyau, les extensions, les pipelines et les araignées elles-mêmes.

L'infrastructure des paramètres fournit un espace de noms global de mappages de valeurs clés que le code peut utiliser pour extraire les valeurs de configuration. Les paramètres peuvent être complétés par différents mécanismes.

Conclusion

Ce projet est pour moi l'aboutissement d'une année d'apprentissage en tant que jeune Développeur Web.

Ces dix semaines de stages ont été une expérience très épanouissante tant au final professionnelle que sociale. L'expérience de travailler dans une entreprise de taille humaine m'a beaucoup apporté pour le développement de mes compétences, étant seule pour l'élaboration du projet, j'ai appris à rechercher les informations nécessaires au bon fonctionnement de l'application.

J'ai eu l'occasion de rencontrer des professionnelles du milieu du E-commerce comme les Web designers qui m'auront énormément aidé pour le maquettage des interfaces et l'expérience utilisateur(UX), mon chef de projet d'avoir eu des idées innovantes et de m'avoir poussé à apprendre Python pour l'utilisation du Framework Scrapy qui me sera très utile à l'avenir.

Dans l'ensemble, je suis ravi d'avoir délivré une application fonctionnelle sur le temps qui m'était imparti, et de savoir que mon chef de stage ainsi que les différents membres de l'entreprise pourront l'utiliser à l'avenir et de rendre leur entreprise plus productive.

Je continuerai à travailler sur l'application pendant mon temps libre pour la rendre plus ergonomique, et repenser à la Vue de l'application en utilisant un outil plus adapté pour le chargement des données à plus grande échelle.

Annexes



Ligne de commande pour le Framework Symfony :

- **php bin/console doctrine:database:create** Création de la base de données.
- **php bin/console doctrine:schema:update --dump-sql** Permet de faire un visuel de modifications apportées à l'entité avant import vers la base de données.
- **php bin/console doctrine:schema:update --dump-sql --force.** Import des informations vers la base de données.
- **php bin/console make:entity** Création ou modification d'une entité avec la CLI
- **php bin/console make:crud** Création des fichiers utiles à l'entité
- **php bin/console make:form** Création d'un formulaire associé ou non à une entité.
- **php bin/console make:entity --regenerate** Régénération des getters et setter des entités.
- **php bin/console cache:clear** Suppression du cache.

Ligne de commande pour l'utilisation de Scrapy:

- **scrapy startproject myproject [project_dir]** : Création d'un nouveau projet Scrapy.
- **scrapy genspider mydomain mydomain.com** : Création d'un nouveau "spiders".
- **scrapy crawl <nomspider>** : Lancement du spider désiré dans la console.
- **scrapy runspider <nomdefichier>.py** : Même fonction que scrapy crawl.
- **scrapy shell** : Interpréteur de commandes interactif où vous pouvez essayer de déboguer votre code de scraping très rapidement, sans avoir à exécuter le spider.

Page Utilisateur version mobile





Comptes Boutique

▼ Nom	▼ Prenom
Cell 1	Cell 2
Cell 3	Cell 4
Cell 5	Cell 6

Ajouter

Page Boutique version mobile



Boutique

B


Boutique


textarea


▼ Etoiles	▼ Avis
Cell 1	Cell 2
Cell 3	Cell 4
Cell 5	Cell 6

Produits en ligne

Filtres


E


E

E

Page Boutique version ordinateur

Lien
Lien
Lien
Lien





Boutique

text

▼ Etoiles	▼ Avis
5	10
4	10
3	10
2	10
1	10

Produits

Filtres

Pinon Benjamin | session dev 18.2 | année 2018

First Seller 62 | 63

Création d'une tâche CRON sur hébergeur :

Editer la tâche

Étape 1 sur 3

* Les champs suivis d'un astérisque sont obligatoires.

Commande à exécuter *

Langage *

PHP 7.2

Activation

☒

Logs par e-mail i *

Autre

Description

Annuler

Suivant

Editer la tâche

Étape 2 sur 3

Sélectionnez la périodicité pour cette tâche :

Mode simple

Mode expert

Heures

Toutes les N heures

Toutes les

heures

Jours

Tous les jours

Jours de la semaine

De tel jour de la semaine à te

Du

Lundi

au

Vend

Mois

Tous les mois

Annuler

Précédent

Suivant

Editer la tâche

Étape 3 sur 3

Résumé de la tâche

Commande à exécuter

Langage

État

☒ Activé

Logs par e-mail

Description

Heures

Jours

Jours de la semaine

Mois

Crontab

Annuler

Précédent

Valider