N° d'ordre : 24 / M2 / TCO Année Universitaire : 2013 / 2014

UNIVERSITE D'ANTANANARIVO

ECOLE SUPERIEURE POLYTECHNIQUE

DEPARTEMENT TELECOMMUNICATION

MEMOIRE DE FIN D'ETUDES

en vue de l'obtention

du DIPLOME de Master

Titre : Ingénieur

Mention : Télécommunication

Parcours : Télécommunication et Multimédia

par : **RAKOTONIRINA Lantoniaina Mampionona**

CONCEPTION ET REALISATION D'UN FRAMEWORK PHP MVC ET D'UN SITE INTERNET DE VENTE EN LIGNE MULTI-BOUTIQUES

Soutenu le 26 Mars 2015 devant la Commission d'Examen composée de :

Président :

M. RAKOTOMALALA Mamy Alain

Examinateurs :

M. RATSIMBAZAFY Andriamanga

M. RAVONIMANANTSOA Ndaohialy Manda-Vy

M. RAKOTONDRAINA Tahina Ezéchiel

Directeur de mémoire : M. RANDRIARIJAONA Lucien Elino

REMERCIEMENTS

Je remercie en premier lieu Dieu tout puissant qui a toujours su me montrer la bonne voie et m'accompagner dans tout ce que j'entreprends. Sans lui, aucune démarche n'aurait été possible dans la réalisation de ce mémoire.

Je tiens également à adresser mes vifs remerciements aux personnes suivantes qui ont contribué de près ou de loin à la concrétisation de ce projet :

- Monsieur ANDRIANARY Philippe, Directeur de l'Ecole Supérieure Polytechnique d'Antananarivo (ESPA) ;
- Monsieur RAKOTOMALALA Mamy Alain, Maître de conférences, Chef du département Télécommunication à l'ESPA, qui a accepté de m'intégrer au sein de son département, et qui me fait le grand honneur de présider le jury de soutenance de ce mémoire ;
- Monsieur RANDRIARIJAONA Lucien Elino, Assistant, enseignant chercheur au sein de l'ESPA, Directeur de ce mémoire, qui n'a jamais ménagé ses efforts dans mon encadrement malgré ses lourdes responsabilités. Je lui suis énormément reconnaissant. Il m'a davantage poussé dans le désir de multiplier mes efforts en vue de réaliser un travail meilleur, toute ma gratitude lui est cordialement adressée ;

Je remercie aussi les personnes suivantes qui ont bien voulu consacrer une partie de leur temps pour juger ce travail, malgré leurs occupations. Je leur suis entièrement reconnaissant pour l'intérêt qu'ils ont porté à l'égard de ce mémoire :

- Monsieur RATSIMBAZAFY Andriamanga, Maître de conférences, enseignant chercheur au sein du département Télécommunication ;
- Monsieur RAVONIMANANTSOA Ndaohialy Manda-Vy, Maître de conférences, enseignant chercheur au sein du département Télécommunication ;
- Monsieur RAKOTONDRAINA Tahina Ezéchiel, Maître de conférences, enseignant chercheur au sein du département Télécommunication.

Je réitère également mes sincères remerciements à tous les enseignants et au personnel de l'ESPA, spécialement ceux du Département Télécommunication, qui nous ont fourni l'enseignement et l'éducation nécessaire pour qu'on puisse se forger un meilleur avenir.

A ma famille et mes proches, je suis infiniment touché par leur soutien infaillible, m'ayant permis d'avancer continuellement. Il serait plus qu'ingrat d'oublier mes parents pour leurs sacrifices durant toutes ces années dans l'unique but de me donner la possibilité d'étudier dans les meilleures conditions possibles.

TABLE DES MATIERES

ABREVIATIONS

AJAX	Asynchronous Javascript and XML
API	Application Programming Interface
ASP	Active Server Pages
BLL	Business Logic Layer
BO	Business Object
CGI	Common Gateway Interface
COBOL	Common Business Oriented Language
CRM	Customer Relationship Management
CRUD	Create Read Update Delete
CSS	Cascading Style Sheets
DAL	Data Access Layer
DNS	Domain Name Server
DOM	Document Object Model
EAI	Enterprise Application Integration
EDI	Enterprise Data Interchange
ERP	Enterprise Resource Planning
GPL	General Public Licence
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IHM	Interface Homme Machine
IIS	Internet Information Services
IPN	Instant Paiement Notification
IT	Information Technology
JSP	Java Server Pages
KM	Knowledge Management
LCD	Langage de Contrôle de Données
LDD	Langage de Définition de Données
LMD	Langage de Manipulation de Données
MVC	Model View Controller

MVVM	Modèle View View Model
NTIC	Nouvelles Technologies de l'information et de la Communication
NUI	Network User Interface
PGI	Progiciel de gestion intégré
PHP	Hypertext Preprocessor
POO	Programmation Orienté Objet
SCM	Supply Chain Management
SEO	Search Engine Optimisation
SGBD	Système de Gestion de Base de Données
SGBDR	Système de Gestion de Base de Données Relationnelles
SQL	Structured Query Language
TPE	Très Petites Entreprises
URL	Uniform Resource Locator
VOD	Video On Demand
WAMP	Windows Apache MySQL PHP
WIMP	Windows, Icons, Mouse, and Pointer
WWW	World Wide Web

INTRODUCTION GENERALE

Microordinateurs, smartphones, tablettes, et même smartwatches, toutes sortes d'appareils viennent coloniser nos foyers. Ils nous accompagnent partout, allant jusqu'à se glisser au fond de nos poches. A cela s'ajoute Internet, immense portail qui permet désormais d'avoir accès à plusieurs formes d'informations électroniques, partout et n'importe quand. Les Technologies de l'Information et de la Communication n'ont jamais cessé et ne cessent de se développer depuis l'avènement d'Internet. De nombreux domaines se sont vus développer en rendant les médias accessibles au grand public.

Joindre l'utile à l'agréable, c'est dans cette lignée que nous allons concevoir un projet de vente en ligne multi-boutiques. Ainsi, l'utilisateur n'aura plus à se déplacer, tout en ayant la possibilité de consulter une multitude de produits. Et cela n'importe où, n'importe quand, à condition seulement d'avoir un terminal adéquat. C'est une réelle occasion de profiter de cette omniprésence d'internet pour un projet dont l'objectif est de concevoir et réaliser un site internet e-commerce. Nous allons ainsi permettre à tout un chacun de faire leurs courses sans peine grâce à leurs terminaux via le web.
La conception d'un site internet est un travail de longue haleine. Cependant, la maîtrise du code est impérative. Afin de permettre une maintenabilité optimale - le site étant amené à plusieurs évolutions dans le temps – nous allons aussi concevoir un framework PHP avec une certaine architecture pour permettre d'organiser tout notre code source. Cela est en plus une occasion de fournir aux autres développeurs finaux un outil avec lequel développer avec rapidité, efficience et en groupe.

De ce fait, pour avoir de plus amples lumières sur le développement du thème de ce mémoire, le plan est divisé en cinq (5) chapitres. Le premier chapitre est consacré aux notions sur le web. Ensuite, on parlera de quelques généralités sur la vente en ligne dans le second chapitre. Le troisième chapitre développera les modèles de programmation qui sont de bonnes pratiques à adopter lors du développement d'un projet informatique. Les grandes lignes de notre framework constitueront le quatrième chapitre. Enfin, c'est dans le dernier chapitre que l'on mettra en avant la réalisation du site web.

CHAPITRE 1
LE WEB

1.1. Introduction

1.1.1. Présentation

Le World Wide Web, appelé aussi communément Web, ou la toile, ou la toile d'araignée mondiale, est un des principaux services offerts par l'internet. C'est un système hypertexte public car il permet de consulter une page mise en ligne dans un site web vers une autre par un simple pointage ou clic à partir d'un navigateur. C'est à partir de ces liens hypertextes qui lient les pages web entre elles que vient l'image de la toile d'araignée.

C'est aussi un système d'information multimédia parce qu'il offre des informations sous plusieurs formats de données, et hypermédia car il s'organise pour naviguer d'un document à un autre par un simple pointage, c'est ce qu'apporte l'hypertexte.

Le protocole de communication utilisé pour transférer les multiples ressources du web est le protocole HTTP (HyperText Transfer Protocol). Cependant, Le web est multiprotocole dans la mesure où, à travers le protocole HTTP, on peut utiliser d'autres protocoles comme FTP (File Transfert Protocol) et SMTP (Simple Mail Transport Protocol) et les services qui en découlent pour différents usages : télécharger des données, consulter un forum, envoyer un email, diffuser de la musique, de la vidéo, etc.

1.1.2. Schéma architectural du web

Le web suit une architecture 3-tiers. Au premier niveau, le client est celui qui demande les ressources au serveur à l'aide de l'URL saisi dans son navigateur. C'est le navigateur qui se charge de la présentation (affichage, contrôles de saisie, mise en forme des données, etc.). Au deuxième niveau, on a le serveur d'application qui est chargé de fournir les ressources, mais aussi de faire appel à un autre serveur du niveau 3. Ce dernier est généralement un serveur de base de données. Il gère les données de façon centralisée et fournit ses services au serveur d'application. [1] [2]

Voici la figure correspondante :

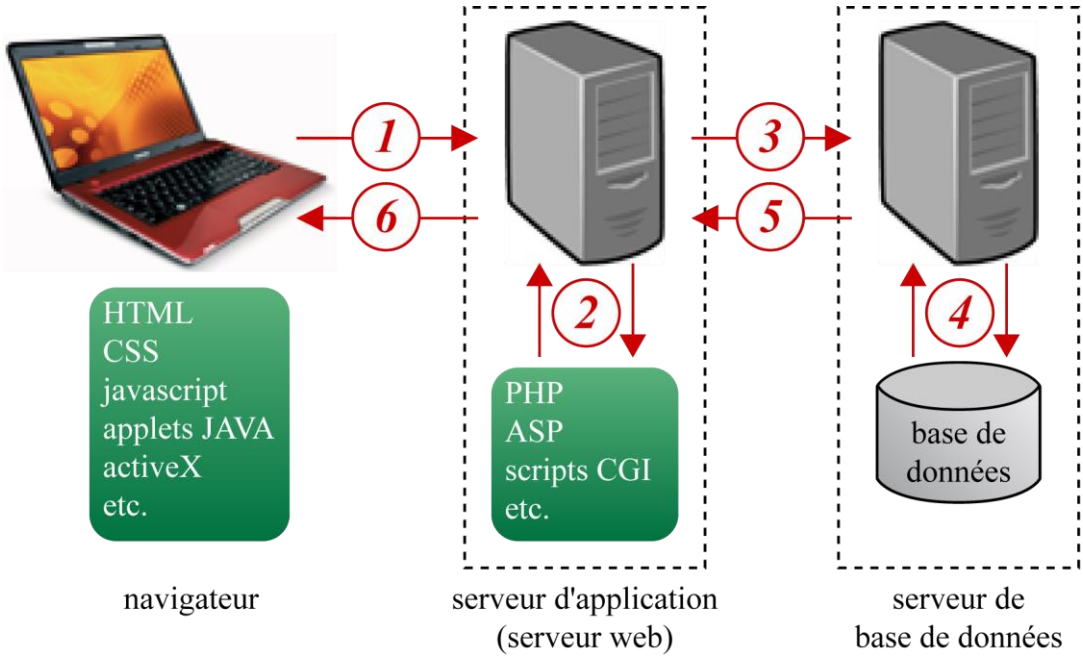


Figure 1.01 : Schéma architectural du web

1. Le navigateur est l'application qui fonctionne sur le poste client permettant d'accéder au Web, et de naviguer grâce aux liens hypertextes. L'utilisateur saisit l'adresse dans son navigateur internet pour localiser les ressources à partir des URLs (Uniform Resource Locator) ;
2. Dès réception de la requête HTTP, le serveur web exécute le code contenu dans la page ;
3. Le serveur web interroge le serveur de base de données au cas où on y aurait recours ;
4. Le serveur SQL traite la demande par l'extraction des données requises ;
5. Le serveur SQL renvoie les enregistrements correspondants à la requête du serveur web ;
6. Le serveur web renvoie une page HTML (HyperText Markup Language) contenant les données. Le navigateur du client traduit le code HTML en page Web. [3]

Quel que soit le navigateur utilisé, le traitement se fera puisqu'il est centralisé sur le serveur. On parle alors de client universel.

La sécurité et la confidentialité des données sont assurées par le serveur de bases de données.

1.1.3. Différents types de sites web

Internet fourmille de sites internet de types différents. En voici les plus connus :

- **Site institutionnel :** site informatif sur les activités et services d'une entreprise ;
- **Site marchand :** site de commerce sur le net qui propose un catalogue de produits que l'on peut commander ou payer en ligne ;
- **Journal en ligne :** journal sur internet ;
- **Portail d'entreprise :** peut prendre plusieurs formes selon la cible :
    - Pour les employés : intranet qui offre en interne des fonctionnalités de travail collaboratif ;
    - Pour les clients : site internet de diffusion d'informations relatives à leur compte ;
    - Pour les partenaires ou fournisseurs : extranet pour le suivi des activités d'échanges, commandes, stocks, etc.
- **Portail généraliste :** offre une multitude de services au grand public (ex : Yahoo!, MSN, etc.) ;
- **Portail thématique :** espace qui propose de traiter un domaine spécifique sujet à un thème et qui offre tous les services qui s'y réfèrent ;
- **Site communautaire ou communauté virtuelle :** espace de communication autour d'une activité commune. Sa différence par rapport au portail thématique est la communauté qui forme un groupe unique et identifié : ses membres ;
- **Réseaux sociaux :** site web ayant pour but de relier des individus différents ;
- **Blog :** carnet de bord mis à jour par son auteur permettant généralement aux lecteurs de poster des commentaires.

1.2. Conception IHM (Interface Homme Machine) [4] [5] [6]

1.2.1. Définitions

L'Interface Homme Machine constitue l'ensemble des dispositifs matériels et logiciels permettant à un utilisateur d'interagir avec un système interactif. L'Interaction Homme Machine est l'ensemble des aspects d'interaction avec un système (pousser un bouton, bouger la souris, etc.).

L'IHM doit permettre à un utilisateur d'accomplir ses tâches efficacement, avec une bonne productivité, en toute sécurité, en y prenant plaisir et en apprenant rapidement et facilement le système à utiliser.

C'est l'ergonomie qui permet de faciliter l'intercompréhension entre utilisateurs et concepteurs. Elle vise à adapter les postes de travail aux caractéristiques physiologiques et psychologiques de l'utilisateur.

1.2.2. Historique

Il est important de saisir l'importance de l'IHM (Interface Homme Machine). L'histoire nous l'a montré, il n'y a pas de solution d'IHM triviale. Une solution que nous considérons être la plus idéale aujourd'hui pourrait ne plus l'être demain.

Prenons exemple sur l'évolution de l'ordinateur même. Les ordinateurs de 1re génération (1945 à 1956) étaient uniquement au niveau matériel avec des langages dépendants de la machine. L'Interaction Homme Machine était quasi-inexistante et l'utilisation de ces machines était réservée à des experts. De plus, leur taille atteignait des mètres.

Ensuite, pour les ordinateurs de 2ème et 3ème générations (1956), la séparation matériel et logiciel fit ses débuts, avec l'utilisation de systèmes d'exploitations et l'évolution du langage machine aux langages de haut niveau comme Cobol. L'interaction est cependant restreinte car elle se fait avec des dispositifs d'entrée-sortie limités (lecteurs ou perforateurs de cartes, tableaux de bord voyants, imprimantes) avec absence de clavier. Les langages de commande sont difficiles à apprendre et sont réservés à des experts.

Les ordinateurs modernes (à partir de 1971) sont caractérisés par l'apparition des « mini-ordinateurs » avec clavier, écran alphanumérique mais toujours un langage de commande réservé aux experts. Depuis, les nouveaux usages d'applications et nouvelles formes vont émerger progressivement comme les écrans cathodiques pour l'affichage graphique.

A partir des années 1980, les ordinateurs commencent à se tourner vers le grand public à travers les GUI (Graphical User Interface) ou interfaces graphiques : fenêtres, pointeur (souris), menu, images, etc. On assistera au paradigme d'interaction naturel adapté finalement aux non spécialistes : action directe sur les objets, feedback immédiat sur les actions, pas de syntaxe d'où erreurs limitées, actions réversibles, etc. L'interface est transparente grâce à la métaphore du bureau.

De nos jours, l'IHM ne cesse d'évoluer vers un paradigme toujours idéal. On a assisté depuis peu à l'émergence de nouveaux dispositifs interactifs et enrichis : GUI tactile, périphériques sans fils, vision 3D, imprimantes 3D, etc. Tout cela pour de nouveaux usages et vers une informatique pervasive et ubiquitaire.

1.2.3. Facteurs humains

Il faut connaître les principaux facteurs humains qui influencent l'ergonomie des IHM. On parle de « design for all » ou « conception pour tous ». Si un caractère humain est affecté et que le critère ergonomique associé est non respecté, il y a un problème lors de la conception de l'IHM.

L'utilisabilité d'un logiciel doit satisfaire trois (3) critères :

- **Apprenabilité :** Elle concerne la facilité avec laquelle l'utilisateur peut prendre en main le logiciel et en découvrir les fonctionnalités. Aussi, il s'agit de la facilité offerte à l'utilisateur de vérifier les effets de ses actions (par exemple à l'aide d'info-bulles, icônes d'attente, messages, désactivation de boutons, etc) ;
- **Flexibilité :** C'est la capacité du système à offrir des modes d'interactions multiples pour répondre aux besoins, aux préférences et à l'expérience de l'utilisation. Mais aussi à s'adapter au contexte et à la diversité des utilisateurs (sexe, culture, âge, niveau d'expertise, etc.). Les utilisateurs ont des attentes et besoins différents selon leurs types ;
- **Robustesse :** Il s'agit du niveau de satisfaction dans la réalisation des tâches permises par le système. En cas de manipulation indésirable (par exemple le respect du format d'email lors d'une saisie sur un champ de texte), les messages doivent être précis et spécifiques, ne pas affoler l'utilisateur, et placés à des endroits adéquats. L'apparition fréquente des erreurs d'utilisation influence l'acceptation du logiciel, il faut donc les prévenir au maximum, cela par une conception sérieuse d'un point de vue ergonomique.

Facteurs humains (perçus par l'utilisateur)	**Principes ergonomiques (causes)**
Apprenabilité	Observabilité, guidage, cohérence, familiarité, généricité
Flexibilité	Adaptation, styles d'utilisation
Robustesse	Prévention, gestion des erreurs

Tableau 1.02 : Facteurs humains et critères ergonomiques associés

1.2.4. Ergonomie des interfaces web

1.2.4.1. Introduction

Une interface web se nomme aussi NUI pour Network User Interface. Pour comprendre l'importance de l'ergonomie des interfaces web, des études ont montré que 62% des acheteurs en ligne abandonnent au moins une fois en cours de la transaction (Davis Z. 1999) et 40 % des

consommateurs ne reviennent pas sur un site où leur première visite s'est soldée par un échec (Nogier J. F. 2005).

1.2.4.2. Spécificités face aux logiciels interactifs

GUI (Logiciels interactifs)	NUI
Gestion événementielle complexe	Peu de gestion événementielle
Manipulation directe importante et potentiellement très riche	Manipulation directe limitée : toucher, clic et saisie clavier
Aucune notion de parcours sur la navigation sauf dans les menus	Navigation centrale avec les liens, signets, menus, onglets, retour avant/arrière, etc.
Effort d'apprentissage nécessaire	Pas ou peu d'apprentissage
Utilisateurs motivés	Utilisateurs pressés et paresseux

Tableau 1.03 : GUI et NUI

1.2.4.3. Démarche de conception

La démarche de conception d'une NUI se déroule en trois (3) étapes :

- **Ciblage :** Il s'agit de définir les objectifs et le contenu du site en fonction des utilisateurs visés, de la définition des services proposés et de l'analyse des besoins ;
- **Structuration :** Il faut organiser les différentes rubriques et les protocoles de navigation pour que l'information soit facilement identifiable et accessible. D'ici découle l'organisation des pages du site en regroupant les « cartes » par similarité, avec possibilité de faire des sous-rubriques ;
- **Finalisation :** Elle définit la charte graphique et l'implémentation des interfaces en tenant compte des recommandations ergonomiques.

Les tests d'utilisabilité doivent être faits tout au long de la conception en prenant compte des utilisateurs novices correspondant à la cible, avec les tests de perception sur les visuels d'interface.

1.2.4.4. Règles d'or

- La facilité d'utilisation (apprenabilité) est plus au centre des attentes de l'utilisateur donc il faut rester standard, sobre et respecter la page et son contexte ;
- Dans la navigation, l'utilisateur ne doit jamais se sentir perdu. Il doit savoir se repérer dans la page en cours de navigation et pouvoir retourner directement à l'accueil du site ;

- Concevoir un rendu variable suivant la taille du terminal (ordinateurs, tablettes, smartphones, etc.) ;
- Utiliser une charte graphique homogène sur tout le site. On peut cependant faire distinguer légèrement la page d'accueil pour faciliter son identification ;
- Définir la largeur des pages en une taille inférieure à la définition horizontale de la majorité des visiteurs pour éviter l'apparition de la barre de défilement horizontal ;
- Tester l'affichage sur les navigateurs les plus usuels pour éviter les incompatibilités.

1.3. Langages de développement

1.3.1. Introduction

Il existe deux (2) formes au développement web :

- **Le développement statique :** Il s'agit de la création de simples pages dites statiques c'est-à-dire avec un contenu figé durant toute la navigation d'un utilisateur ;
- **Le développement dynamique :** Contrairement au développement statique, on a à faire à des pages dynamiques c'est-à-dire avec un contenu qui change. C'est dans ce type de développement que l'on trouve éventuellement une connexion à une base de données dans laquelle on extrait les informations à afficher, en fonction des besoins et des actions de l'utilisateur.

On distingue aussi, pour le développement web, deux (2) types de langages :

- Les langages **côté client** ou langages de script côté client : Ils ne nécessitent pas de compilation mais juste d'interpréteurs qui sont les navigateurs web. On dit qu'ils sont réalisés du côté client du fait que ce sont les navigateurs qui interprètent le code pour générer la page reçue ;
- Les langages **côté serveur** ou langages de script côté serveur : Ces langages interagissent avec un serveur HTTP pour créer une page web qui sera renvoyée au client. Le serveur génère donc un résultat en langage HTML qui est renvoyé au navigateur qui, à son tour, va afficher la page et exécuter les scripts côté client générés. Exemples : PHP, ASP, CGI et JSP. L'utilisation d'un langage serveur est nécessaire pour utiliser une base de données.

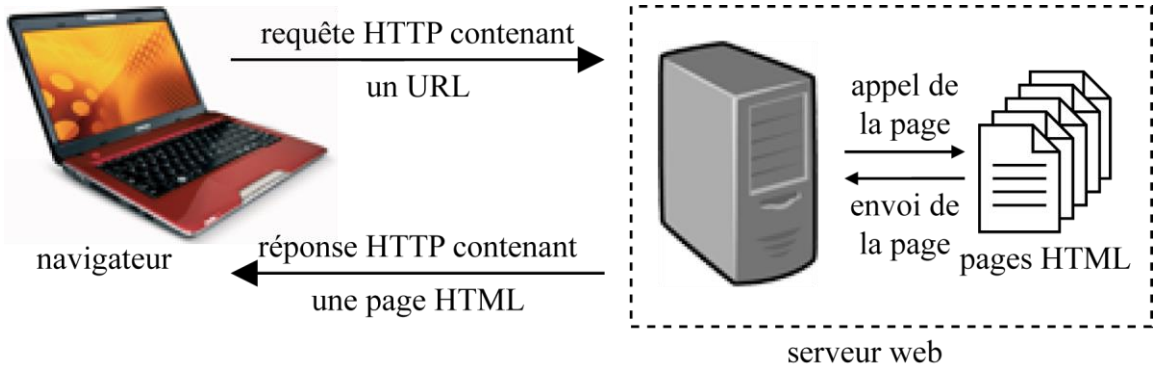


Figure 1.02 : Fonctionnement des langages côté client

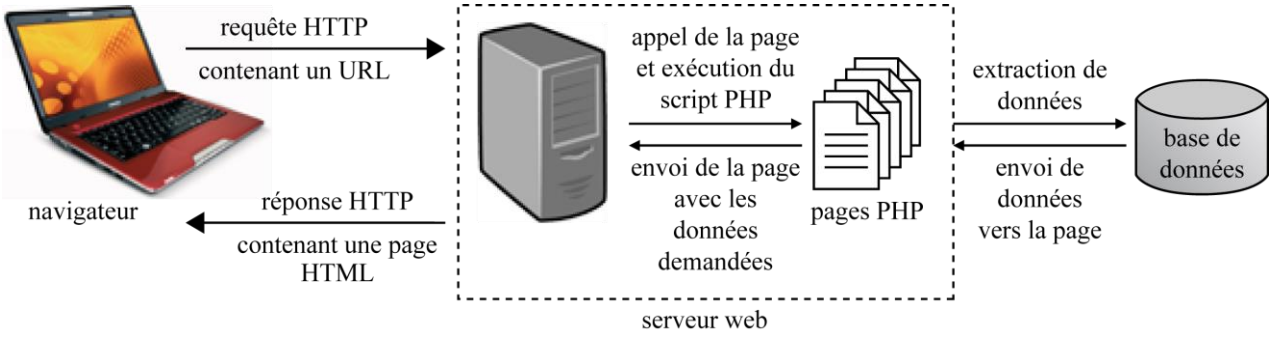


Figure 1.03 : Fonctionnement des langages côté serveur

1.3.2. HTML

1.3.2.1. Définition

Le HTML est un langage côté client né dans les années 1990. C'est le langage universel de base qui est compris dans toute page web. Il est décrit comme étant un langage de balisage car il définit essentiellement la structure de la page web (titres, tableaux, paragraphes, listes, etc.) et une mise en forme basique du document (styles de caractères, choix de police, etc.). Cela se fait à l'aide de « balises HTML » qui sont des marques placées par l'auteur pour ordonner l'aspect du document en y incluant des informations variées telles que du texte, des images, etc.

Le code source d'une page HTML est donc enrichi par du texte enveloppé ou non par différentes sortes de balises ayant chacun son nom et sa fonctionnalité. Chaque balise contenant éventuellement ou obligatoirement un ou plusieurs attributs dont les valeurs sont écrites entre guillemets. Ce sont ces attributs qui permettent de paramétrer une balise. [7]

1.3.2.2. Caractéristiques

Le langage HTML présente les caractéristiques suivantes :

- Langage côté client ;
- Langage de balisage pour définir la structure et la mise en page du document ;
- Relativement facile à aborder ;
- Insensible à la casse ;
- Possibilité d'écrire des liens hypertextes liant plusieurs documents accessibles grâce à leur URL ;
- Multiplateforme c'est-à-dire reconnu par tous les navigateurs web, à l'exception de quelques balises qui sont propres à certains navigateurs (comme la balise <marquee> pour Internet Explorer et Google Chrome) ;
- Actuellement à sa version 5 qui présente des problèmes de compatibilité pour les versions anciennes de navigateurs ;
- Compatibilité avec des langages de scripts tels que PHP, CSS et Javascript. [7] [8] [9]

1.3.2.3. HTML5

Le HTML5 est l'actuelle version du HTML qui introduit une toute autre série de nouveaux éléments pour définir une page web, sans rien casser au support utilisé pour les versions précédentes.

La sémantique est améliorée en définissant des balises plus précises pour décrire la signification du contenu. Il devient beaucoup plus simple de structurer les pages qui sont désormais représentées en plusieurs sections en utilisant des éléments spécifiques de séparation :

- Balise <header> **:** l'en-tête ;
- Balise <footer> : le bas de page ;
- Balise <nav> : section de navigation ;
- Balise <aside> : essentiellement utile pour les barres latérales ;
- Balise section : une section générique d'un document ou d'une application, comme un chapitre par exemple ;
- Balise <article> : une section indépendante d'un document, d'une page ou d'un site, qui convient pour du contenu comme des nouvelles, des articles, des messages de forum ou des commentaires individuels ;
- Balise <hgroup> : en-tête d'une section, utilisé pour regrouper des niveaux de titres.

En outre, combiné avec des langages de scripts tels Javascript, le HTML5 devient un outil puissant. Premièrement, il permet l'inclusion de balises multimédias qui intègrent directement un lecteur dans la page. Elles ciblent donc les éléments audio et vidéo. Ensuite, le HTML5 apporte la balise la plus prometteuse qui est la balise <canvas>, balise permettant l'insertion, la génération et l'animation de formes, images ou textes. Cette zone de traçage permet en somme de créer d'impressionnantes animations, voire même des jeux au sein même du navigateur.
D'autres fonctionnalités plus secondaires sont fournies par le HTML5. On peut en citer l'amélioration des balises de formulaires et la présentation de vocabulaires pour les microdonnées.

1.3.2.4. Eléments de présentation

Voici les quelques éléments de présentation proposés par le HTML pour définir la mise en forme d'un document : contenu textuel et modification typographique, niveaux de titre, hyperliens et courrier électronique, paragraphes, listes et glossaires, traits ou lignes de séparation, formulaires, insertion d'images, tableaux, cadres ou frames, division de page par les balises génériques .

1.3.2.5. Limites du HTML

Une page web intégralement construite en HTML est totalement statique et n'offre qu'une faible possibilité d'interaction avec le visiteur. Chaque page doit être créée au préalable et doit être manuellement mise à jour. Néanmoins, quelques fonctionnalités des CSS permettaient de créer un peu de dynamisme dans la page, par exemple un changement de couleur lors d'un survol du cuseur sur un élément de la page.

1.3.3. CSS [7]

1.3.3.1. Définition

Le CSS ou Cascading Stylesheets ou encore feuilles de style en cascade est un langage de script côté client qui, associé aux documents HTML, permet une meilleure mise en forme de la page web. Les styles définissent des règles plus poussées de mise en forme en redéfinissant les règles appliquée aux éléments de la page : couleurs, polices, rendu, organisation des blocs, et d'autres caractéristiques liées à la présentation d'un document.
Cependant, le CSS, même s'ils sont standardisés par le W3C, ne sont pas toujours reconnus de la même manière par les différents navigateurs web.

1.3.3.2. Objectifs

L'objectif du CSS est de séparer la structure d'un document écrite en HTML et la présentation qui est écrite en CSS. Certes, avec du HTML, on peut définir à la fois la structure et la présentation mais cela pose quelques problèmes. Avec le couple HTML/CSS, on peut créer des pages web où la structure du document se trouve dans le fichier HTML tandis que la présentation se situe dans un fichier CSS. De plus, avec CSS on peut définir un ensemble de règles stylistiques communes à toutes les pages d'un site internet. Cela facilite ainsi la modification de la présentation d'un site entier et facilite également la lisibilité du HTML car les styles sont définis à part.

1.3.3.3. Infrastructure et éléments

CSS est un langage non sensible à la casse et qui possède sa propre syntaxe. Un jeu de règles de styles appliqué à un élément d'une page HTML est composé de sélecteurs et de déclarations de style :

- Le sélecteur indique l'élément auquel les styles s'appliquent. Une multitude de sélecteurs peuvent être écrites pour une seule déclaration de style. Séparés par un espace, cela indique l'arborescence dans le document tandis que, séparés par une virgule, on indique une déclaration de style commune pour les éléments énumérés ;
- Une déclaration de style est formée d'un ou de plusieurs couples propriété/valeur entre accolades. Elle représente les différents styles appliqués à l'élément HTML. Il est possible de mettre entre accolades autant de déclarations que nécessaire en les séparant par des points-virgules.

La syntaxe générale est la suivante :

Figure 1.04 : Syntaxe CSS

1.3.3.4. CSS3

Le CSS3 est la dernière version du CSS qui propose un très enrichissant ensemble de nouveautés sur les feuilles de style.

En premier lieu, il offre la mise à disposition de nouveaux effets visuels impressionnants à appliquer sur les éléments HTML : coins arrondis, ombre de bloc et de texte, dégradé, opacité, transformation, transition et transformation 3D. Les contraintes sur les polices de texte se voient

aussi être levées car on peut désormais utiliser des polices personnalisées de texte indépendamment de celles présentes sur le système du navigateur utilisé.

En second lieu, on assiste à une arrivée de toute une vague de nouveaux sélecteurs. Il est alors maintenant possible d'être plus précis sur les critères de sélection d'éléments HTML (par le numéro position, les valeurs d'attributs, l'exclusion, etc.).

D'autre part, les « *média queries* » permettent de définir différentes versions de styles selon le type de média de l'utilisateur (ordinateur, smartphones, tablettes, etc.) afin de s'adapter par exemple à la largeur de l'écran.

A tout cela s'ajoutent d'autres atouts tout aussi révolutionnaires tels :

- une nouvelle manière de redéfinir les couleurs : en RGB qui accueille une composante Alpha permettant de définir l'opacité ;
- la détection des caractéristiques de l'appareil de l'utilisateur ;
- les calculs ;
- la possibilité d'utiliser des SVG (Scalable Vector Graphics) en arrière plan ;
- la division d'un bloc en colonnes.

Néanmoins, il existe quelques problèmes d'incompatibilités pour les versions nouvelles de navigateurs. Ces derniers ne connaissent pas toujours les dernières fonctionnalités de HTML et CSS, à moins d'effectuer des mises à jour. Il faudra donc vérifier lors du développement comment le site s'affiche sur les différentes versions de navigateurs.

1.3.4. Javascript et ses bibliothèques

1.3.4.1. Définition

Javascript est un langage de script incorporé dans des pages HTML pour les rendre interactives avec le client. Le client obtient des réponses aux évènements qu'il déclenche de la page (ex : clic de la souris, envoi de formulaire, etc.) et l'exécution des commandes se fait au niveau du navigateur-même sans nécessité de transmission vers le serveur. Par conséquent, Javascript est un langage côté client.

Déporter les traitements sur le client allège la charge du serveur ; les opérations sont exécutées du côté du client et sont donc très rapides.

JavaScript ne doit pas être confondu avec Java, qui est un langage beaucoup plus complexe permettant de créer des applications autonomes. [10]

1.3.4.2. Structure de JavaScript

C'est un langage orienté objet à prototype. Les bases du langage et ses principales interfaces sont fournies par des objets qui ne sont pas des instances de classes, mais qui sont chacun équipés de constructeurs permettant de générer leurs propriétés, et notamment une propriété de prototypage qui permet d'en générer des objets héritiers personnalisés. Il donne à disposition des objets JavaScript natifs ou prédéfinis tels que String (chaînes de caractères), Number (nombres), Array (tableaux), RegExp (expressions régulières) ou Image (images). Il existe également d'autres objets comme les informations sur le navigateur.

En outre, tout élément à l'intérieur d'une page HTML peut être représenté par un objet par l'intermédiaire de la structure DOM ou Document Object Model. Le DOM est une interface de programmation entre un document HTML et un script JavaScript ; le DOM est donc une API permet d'accéder au code HTML, de naviguer dedans, de modifier sa structure et son style, tout cela via Javascript. On parle désormais d'éléments pour qualifier les balises HTML puisque chaque balise est vue comme un objet. La spécification du W3C définit le DOM comme un document HTML schématisé hiérarchiquement sous la forme d'un arbre. Depuis, on parle d'éléments parents et enfants quand il y a des éléments qui s'imbriquent les uns des autres. Chaque élément est appelé nœud ou node qui contient ou non un contenu textuel.

JavaScript est aussi un langage de script événementiel. Un évènement permet de déclencher des fonctions selon l'état d'une action. Autrement dit, un évènement est une perturbation de l'environnement courant qui permet de déclencher un code Javascript lors d'une action due à l'utilisateur ou au navigateur (clic, survol du curseur, chargement d'une image, etc.). [10]

1.3.4.3. Ajax

Dans une application web classique, un client effectue une requête HTTP et un serveur web renvoie la page. Pour communiquer une nouvelle fois avec le serveur, il faudra recharger la page. C'est pour contourner ce système qu'intervient Ajax (Asynchronous Javascript and XML). Son principe de base est que Javascript émet des requêtes vers le serveur qui répond avec uniquement les informations demandées. Ensuite, des scripts vont modifier le contenu de la page en fonction de la réponse reçue. Toutes ces étapes se font sans aucun chargement de la page. On utilise Ajax par exemple pour l'autocomplétion de recherches, la pagination d'éléments par sélection à partir d'une listbox, la discussion instantanée, etc.

La méthode Ajax permet de réaliser des applications Internet riches, offrant une maniabilité et un confort supérieur ; c'est un des sujets phares du mouvement Web 2.0.

Ajax se base sur l'objet Javascript nommé « XmlHttpRequest ». En somme, Ajax est donc une combinaison de langages du web, à savoir Javascript, DOM, HTML et XML grâce à l'objet XmlHttpRequest. [11]

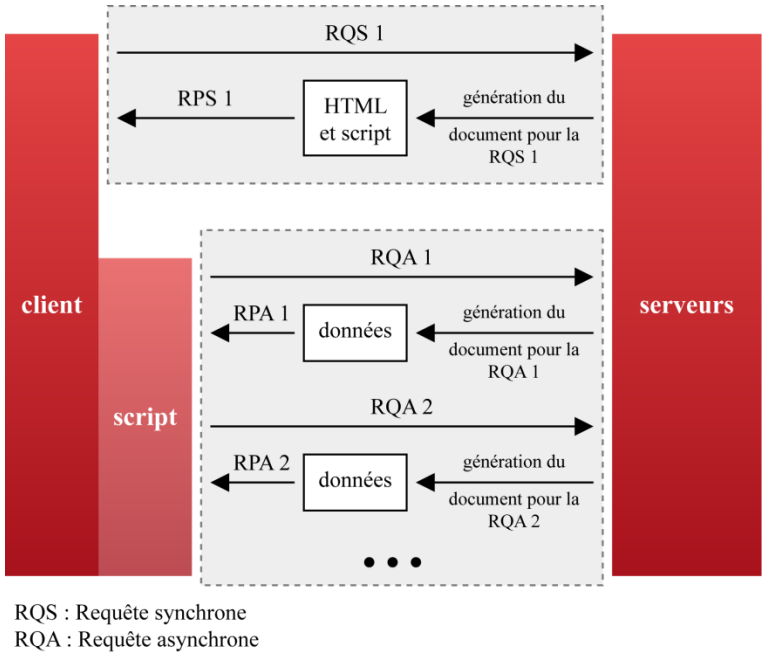


Figure 1.05 : Principe d'Ajax

Certes, les avantages fournis par Ajax sont considérables dans le domaine du développement web, mais il présente aussi des inconvénients de taille :

- L'option Javascript doit être activée ;
- L'objet « XmlHttpRequest » n'est pas supporté par certains vieux navigateurs ;
- Les données qui sont chargées ne font pas en réalité partie de la page ;
- Les requêtes Ajax asynchrones passent par internet, il n'y a donc aucune garantie que les paquets qui y circulent arrivent dans l'ordre ;
- Le bouton « page précédente » ne marche pas donc il est très difficile d'annuler des modifications faites par Javascript.

Il est préférable d'éviter les appels multiples, mieux vaut faire une seule requête lourde au début au lieu de plusieurs petites requêtes. [11]

1.3.4.4. JSON [13]

JSON, pour JavaScript Object Notation, est un format utilisant la notation des objets JavaScript pour transmettre de l'information structurée, d'une façon plus compacte et plus proche des langages de programmation comme XML. JSON reste le moyen le plus simple d'accéder à des données, puisque chaque flux JSON n'est rien d'autre qu'un objet JavaScript sérialisé. De plus, JSON est un format de données structurées, et peut être utilisé facilement par tous les langages de programmation. En effet, JSON est un format texte complètement indépendant de tout langage, mais avec des conventions universelles.

Les éléments de JSON sont:

- Un objet : représenté par une accolade contenant d'autres objets ou des variables ;
- Une variable scalaire : Number, String, Boolean ;
- Un tableau : représenté par un crochet qui contient une liste d'éléments séparés par des virgules ;
- Des valeurs littérales : null, true, false, chaîne de caractères, et des valeurs numériques.

1.3.4.5. Bibliothèque JQuery [12]

Javascript offre une syntaxe assez longue et répétitive d'écriture. C'est pour cette raison que l'on a créé les différentes bibliothèques Javascript.

Une bibliothèque Javascript est composée d'un ou de plusieurs fichiers Javascript qui définissent une multitude de fonctions permettant de simplifier son utilisation, de synthétiser certaines tâches répétitives, d'augmenter la performance du langage et surtout de sauver un temps précieux en limitant le temps de production du développeur.

JQuery est une bibliothèque JS gratuite parmi tant d'autres tels que Mootools, Scriptaculous, Prototype, Yahoo UI, Mochikit, ExtJS, Dojo, etc. Il a été créé par John Resig et les premières expérimentations ont eu lieu en 2005.

En plus d'utiliser une syntaxe beaucoup plus courte, de nombreuses fonctionnalités ont été ajoutées afin de gérer les événements et les attributs, de manipuler le DOM avec emploi intelligent de sélecteurs basés sur CSS3, de créer des effets, d'utiliser AJAX, de lever les problèmes d'incompatibilités entre navigateurs, etc. Depuis l'arrivée de jQuery le web a fait un bond de géant au niveau de l'interactivité. Plus de 50 fonctionnalités sont maintenant possibles : des menus, onglets, accordéons, carrousels d'images, lightbox, validation de formulaires, tableaux, etc. Selon une statistique d'Elie Bursztein de Google, plus de 45% des 100 000 sites webs les plus fréquentés

utilisent jQuery. Il permet une multitude d'actions et remplace certaines autres technologies comme Flash.
Au final, jQuery permet de simplifier l'utilisation de Javascript en offrant de puissantes fonctions de manipulation d'objets du DOM avec une syntaxe astucieuse, d'où son slogan « Write Less, Do More » en anglais, qui signifie « Ecrire moins pour faire plus ».

1.3.4.6. JavaScript coté serveur

Actuellement, JavaScript peut aussi être utilisé comme langage de programmation côté serveur. Le projet CommonJS travaille dans le but de spécifier un écosystème pour JavaScript en dehors du navigateur donc sur le serveur ou pour les applications de bureau natives. Le projet a été lancé par Kevin Dangoor en janvier 2009.
Il existe par ailleurs des projets indépendants et Open Source d'implémentation de serveurs en JavaScript. Parmi eux, on pourra distinguer Node.JS, une plateforme polyvalente de développement d'applications réseau se basant sur le moteur JavaScript V8 et les spécifications CommonJS.

1.3.5. PHP

1.3.5.1. Définition

PHP ou Hypertext Preprocessor est un langage qui fonctionne côté serveur.

L'avantage d'un langage serveur est qu'il permet d'adapter le site à l'utilisateur d'après ses besoins, ses autorisations ou d'autres informations provenant d'une base de données, tout cela sans nécessiter le support d'une technologie supplémentaire par le client.

PHP étant un langage de script orienté serveur, il faut installer un serveur HTTP (le plus répandu étant Apache) et un interpréteur PHP local. C'est ce dernier qui va lire les scripts et les exécuter afin par exemple de retourner une page HTML. L'installation de PHP peut se faire via des kits d'installation complets qui contiennent Apache et PHP mais également un serveur de base de données MySQL et quelques utilitaires. [14]

1.3.5.2. Architecture web avec PHP

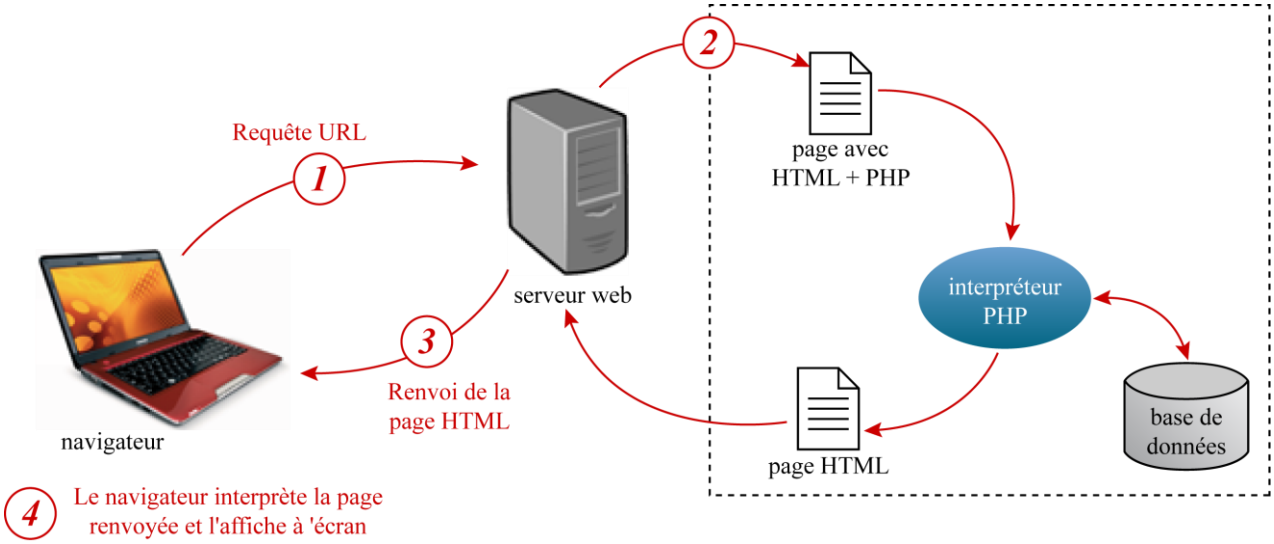


Figure 1.06 : Architecture web avec PHP

La figure ci-dessus présente l'architecture lorsque le serveur web est couplé avec un interpréteur PHP. Il existe un intermédiaire entre le serveur web qui réceptionne la requête HTTP et le renvoi de la page correspondante. En effet, le serveur web qui stocke des pages PHP (avec du code PHP imbriqué dans les balises HTML) doit les envoyer à l'interpréteur PHP pour exécuter le code qui y est contenu, en interrogeant éventuellement la base de données. L'interpréteur PHP restitue alors une page HTML (sans le PHP qui a été exécuté) et dont les résultats sont visibles dans le code HTML sous forme de texte et/ou balises HTML qui seront interprétés par le navigateur web.

C'est l'extension du fichier (php, php3, php5, etc.) qui indique au serveur web que l'interpréteur PHP peut rencontrer du code PHP et donc l'interpréter. Il n'y a pas de compilation du code source pour le rendre exécutable, seulement une interprétation ligne par ligne du fichier PHP.
En somme, PHP permet de générer du code HTML en fonction des instructions données par le développeur et visibles uniquement dans le code source du fichier et non dans la page HTML retournée. [14][15]

1.3.6. Avantages de PHP

Plusieurs caractéristiques font de PHP un langage populaire et facile d'accès pour le web :

- Il est interprété, par conséquent il ne nécessite pas d'être compilé pour obtenir un objet ou un exécutable avant d'être utilisable. C'est pourquoi c'est un langage de scripts ;
- Il est supporté en tant que module, par le serveur web Apache qui est le plus répandu dans le monde. Mais il fonctionne aussi avec d'autres serveurs web comme IPlanet, IIS, etc. ;
- Il permet d'exploiter facilement de très nombreuses bases de données comme Oracle, MySQL, dBase, Sybase, PostgreSQL, MySQL, etc. ;
- Il peut se connecter avec des systèmes de paiement en ligne : PayPal, Verisign, Cybercash, Crédit Mutuel, etc. ;
- Il est gratuit et performant ;
- Le duo PHP et MySQL (tout aussi gratuit et performant) est particulièrement aisé à mettre en place et proposé à des prix modestes chez les hébergeurs ;
- Il est très largement documenté car il est devenu de plus en plus répandu notamment sur les sites professionnels ;
- PHP est multi plateformes : Windows, UNIX, LINUX, IBM iSeries (AS/400), SGI IRIX, Novell Netware, RISC OS, AmigaOS, MAC OS, etc. ;
- PHP, à partir de la version 4, s'exécute rapidement avec une stabilité à toute épreuve ;
- Il est développé actuellement par des milliers d'ingénieurs regroupés au sein de la fondation Apache ;
- Le code PHP est fortement inspiré de C et de Perl, ce qui en facilite l'apprentissage ;
- Un des gros avantages de PHP sur d'autres langages comme Perl est l'intégration dans la même page du code HTML « brut » et du code PHP. Le code PHP s'imbrique dans le code HTML en étant délimité comme tel. Ainsi, les scripts PHP ne nécessitent pas de répertoires spéciaux comme le dossier « /cgi-bin » pour les CGI ;

- Grâce à de nombreuses extensions dynamiques, PHP peut générer des fichiers PDF, s'interfacer avec des serveurs de messagerie, des serveurs LDAP, générer de la cartographie, des images et graphiques, générer des animations flash, pour ne citer que quelques-unes des fonctionnalités les plus impressionnantes ;
- Il réunit environ 4,5 millions de développeurs dans le monde et fait partie des langages les plus utilisés ;
- Depuis sa version 5, PHP intégre la programmation orientée objet avec tous les concepts qui en découlent (classe, encapsulation, interface, héritage, exception, etc.).

1.3.6.1. Concepts de base

Du script PHP est intégré dans du code HTML à l'intérieur des balises « < ?php ?> ». Comme tout langage de programmation, PHP offre des fonctionnalités basiques : variables, tableaux, fonctions, conditions boucles, opérateurs, expressions régulières, manipulation de fichiers, etc. Aussi, il possède ses propres concepts afin de pouvoir dynamiser une page web :

- Inclusion de code HTML ;
- Transmission de variables de page en page. Pour cela il existe différentes méthodes :
    - Méthode GET où l'on envoie les variables dans l'URL même de la requête HTTP ;
    - Méthode POST où l'on envoie les variables dans le corps de la requête HTTP donc invisibles dans l'URL ;
    - Les cookies qui sont des informations stockées sur la machine du client ;
    - Les sessions qui sont des informations stockées sur la machine du serveur ;
- Connexion à une base de données et manipulation de bases de données. En effet, PHP fait l'intermédiaire entre l'utilisateur et MySQL. Cela peut se faire avec :
    - L'extension « mysql_ » qui regroupe des fonctions d'accès à une base de données. Elle est ancienne et non recommandée actuellement ;
    - L'extension « mysql_i » qui sont des fonctions améliorées et qui proposent des fonctionnalités qui sont plus à jour ;
    - L'extension PDO qui est un outil complet d'accès à une base de données ;
- Génération et manipulation d'images grâce à la librairie GD ;
- Génération de fichiers PDF grâce à la librairie FPDF ;
- Création, écriture et lecture d'un fichier stocké sur le disque dur du serveur avec paramétrage du droit d'accès. [14][15][16][17]

1.4. Bases de données

1.4.1. Définitions

Une base de données est un ensemble organisé et structuré de données rassemblées, stockées et à mémoriser de manière permanente pour être facilement exploitable par un ensemble d'utilisateurs. Un Système de Gestion de Bases de Données ou SGBD, quant à lui, est un programme de haut niveau permettant aux utilisateurs d'interagir avec une base de données. Cela en vue de structurer, d'insérer, de modifier et de rechercher de manière efficace des données spécifiques. Il permet entre autres le partage de la base de données entre les différents utilisateurs et applications y accédant, mais aussi sa protection, son entretien et son évolution.

La mise en place des bases de données est devenue incontournable dans le but de gérer de plus en plus de quantité de données en mémoire qui seront utilisées par un très grand nombre d'utilisateurs. Effectivement, le volume de données n'arrête pas de monter sous la poussée des nouvelles technologies du Web. [16][17]

1.4.2. Architectures de base de données

Au cours des années, les bases de données se sont orientées vers différentes architectures :

- **Modèle hiérarchique :** Les informations sont organisées sous une structure d'arborescence. Ici, chaque enregistrement ne possède qu'un seul parent donc il est impossible d'avoir de relation « n vers n » ;
- **Modèle réseau :** Contrairement au modèle hiérarchique, ce modèle est en mesure de lever de nombreuses difficultés du modèle hiérarchique grâce à la possibilité d'établir des liaisons de type « n vers n » en définissant des associations entre tous les types d'enregistrements. Les liens entre objets peuvent exister sans restriction. Cependant, pour retrouver une donnée, il faut connaître le chemin d'accès (les liens), rendant les programmes dépendants de la structure de données ;
- **Modèle relationnel :** L'intérêt du modèle relationnel se situe dans sa simplicité et ses fondations mathématiques (théorie des ensembles et logique du premier ordre). Ces facteurs ont apporté un considérable progrès dans la représentation et la manipulation de données. Son principe repose sur le stockage des données hétérogènes dans des tables, avec la possibilité d'établir des relations entre elles. Ces vues relationnelles permettent à chaque utilisateur de personnaliser sa vision des données et les contraintes d'intégrité

complètent la description de l'information. C'est ce modèle qui s'impose sur le marché actuel par le biais du langage SQL ;

- Modèle objet : Il constitue une tentative de réponse à des nouveaux besoins pour des applications plus complexes nécessitant des structures d'objets plus complexes, de nouveaux types de données pour le stockage de fichiers, ou encore la possibilité de définir des opérations non standards qui sont spécifiques aux applications. Des exemples d'applications complexes sont le CAD/CAM (Computer-Aided Design/Computer-Aided Manufacturing), les bases de données géographiques (GIS : Geographic Information Systems) et les bases de données multimédia. Ces systèmes reprennent les concepts utilisés dans les langages de programmation orientés objet, ajoutés aux spécificités des systèmes de bases de données : persistance des données, transactions, etc. Ce modèle pourrait probablement être ajouté au modèle relationnel pour avoir le modèle objet-relationnel afin de bénéficier des avantages des deux approches. [16]

1.4.3. Modèle Conceptuel de Données (MCD)

Le MCD ou Modèle Conceptuel de Données est un formalisme graphique pour la modélisation de données. C'est une étape intermédiaire avant l'intégration du modèle obtenu vers un SGBD pour pouvoir manipuler les données dans une application informatique.

Son but est de décrire le monde réel par l'intermédiaire de concepts d'entités et d'associations. Ce concept permet de distinguer les entités et les associations entre les entités qui vont constituer la base de données après avoir effectué une analyse des données avec la liste des contraintes. Ainsi, l'on pourra avoir une structure de qualité indispensable pour surmonter les problèmes de gestion de données. Ces problèmes sont l'apparition des redondances et le non respect des contraintes. Le modèle obtenu sera ensuite converti en un modèle exploitable par un SGBD, par exemple en une base de données relationnelle.

1.4.4. Langage SQL

Le langage SQL ou Structured Query Language ou langage d'interrogation structuré est un langage de requête utilisé pour interroger, manipuler et définir les données, mais aussi pour contrôler l'accès aux bases de données qui utilisent le modèle relationnel.

Il a été conçu par IBM dans les années 70 et a ensuite été adopté comme norme de langage de manipulation de bases de données pour devenir le langage standard des Systèmes de Gestion de

Bases de Données Relationnelles (SGBDR), tel MySQL. Ce dernier est un SGBD faisant partie des systèmes les plus utilisés actuellement. MySQL est basé sur une architecture client/serveur c'est-à-dire que les clients doivent s'adresser au serveur. C'est ce serveur qui gère et contrôle les accès aux données, en communiquant par des requêtes avec réponses. Un serveur MySQL gère pour cela une ou plusieurs bases de données qui contiennent chacune différents types d'objets : tables, index, fonctions, etc. Le type d'objet le plus représenté est la table qui est la forme de stockage des relations. Chaque table est caractérisée par une ou plusieurs colonnes appelés champs ou attributs définis dans un type de données qui spécifient le domaine dans lequel l'attribut est défini (int, varchar, text, datetime, etc.).

Le SGBD MySQL permettra à des utilisateurs de créer et maintenir une base de données à l'aide de langages de requêtes SQL :

- Langage de Description des Données ou LDD : Il permet de décrire la structure de ces objets avec les contraintes à imposer. Ses verbes d'action sont :
    - CREATE : Création d'objet comme une table ;
    - ALTER : Modification d'un objet ;
    - DROP : Suppression d'un objet ;
- Langage de Manipulation des Données ou LMD : C'est le langage qui permet de d'accéder et de modifier les données contenues dans les tables sous forme de lignes. Ses verbes d'action sont :
    - SELECT : extraction de valeurs ;
    - INSERT : ajout de lignes ;
    - UPDATE : mise à jour de lignes ;
    - DELETE : suppression de lignes ;
- Langage de Contrôle de Données ou LCD : Il permet de gérer les droits d'accès aux tables dont l'accès est restreint à des utilisateurs connus et identifiés par un nom et un mot de passe du SGBD. Les commandes SQL correspondantes sont :
    - GRANT : accorder une autorisation sur un objet ;
    - REVOKE : supprimer une autorisation sur un objet ;
    - SET PASSWORD : modifier le mot de passe d'un utilisateur. [19][20]

1.5. Conclusion

Le web est actuellement envahi par différents types de sites web, allant du simple site informatif aux réseaux sociaux. Si ils diffèrent par leurs caractéristiques, ils sont quasiment les mêmes si on regarde l'architecture du web. De manière basique, l'utilisateur saisit une URL dans un navigateur, le serveur d'applications se charge de fournir les données en faisant éventuellement appel à un serveur de base de données pour fournir les données. Les langages côtés serveurs (PHP, ASP, etc.) étant exécutés au niveau du serveur web, les langages côtés clients (HTML, CSS, JS, etc.) étant interprétés au niveau du navigateur. Maintenant, dans un contexte plus restreint, nous allons dans le chapitre suivant, parler de notions sur la vente en ligne

CHAPITRE 2
VENTE EN LIGNE

2.1.Généralités sur le e-commerce [21]

2.1.1. Définitions

Le e-commerce ou commerce électronique est une forme de commerce dont les transactions se font en ligne ou à distance par le biais d'interfaces électroniques et digitales, sans contact visuel entre l'acheteur et le vendeur. Il englobe essentiellement les transactions commerciales s'effectuant entre les réseaux informatiques, notamment Internet, à partir des différents types de terminaux tels que les ordinateurs, tablettes, smartphones, TV connectées, etc.

Les biens et les services font l'objet de commandes effectuées via le réseau. Pour le paiement et la livraison ultime, ils peuvent être effectués par des méthodes traditionnelles.

Il est à noter que les commandes reçues par téléphone, par télécopieur et par le courrier électronique ne sont pas considérées comme étant du commerce électronique car ces outils ne permettent pas une automatisation complète des transactions commerciales. Les transactions bancaires et financières ne relèvent pas non plus du commerce électronique.

2.1.2. Différents types de e-commerce

L'e-commerce étant basé sur une certaine nature de relation vendeur-acheteur, on peut citer :

- **Le B2G (Business to Government) :** Echange électronique entre les entreprises privées et le gouvernement ;
- **Le B2B (Business to Business) :** Commerce électronique entre entreprises
- **Le B2E (Business to Employee) :** Echange électronique entre une entreprise et ses employés, aussi appelé Intranet ;
- **Le B2C (Business to Consumer) :** Commerce électronique à destination des particuliers. Il s'agit de sites web marchands ;
- **Le C2C (Consumer to Consumer) :** Commerce électronique entre particuliers. Il s'agit de sites web permettant la vente entre particuliers.

Pour le commerce électronique entre particuliers, il existe trois (3) systèmes d'échanges :

- Les ventes aux enchères ;
- Les tiers de confiance ;
- Les petites annonces.

Parmi les principaux biens et services que l'on peut vendre par internet, on peut citer les biens culturels (livres, CD et DVD, etc.), les appareils technologiques (PC, smartphones, chaînes hi-fi, etc.), le tourisme et les voyages (billets, locations, etc.), les produits de grande consommation avec les supermarchés en ligne, les produits d'imprimerie (cartes de visites, plaquettes, supports commerciaux), les produits d'habitats, les vêtements, la puériculture, etc. Il est aussi possible d'effectuer des systèmes de vente spécialement adaptés au monde internet tels le développement de photographies numériques, le téléchargement de musique, la location de DVD par internet et la VOD ou vidéo à la demande. En outre, plusieurs entreprises proposent également des services sur internet, qu'ils soient payant ou non. On peut citer la banque en ligne, l'assurance en ligne ou la presse en ligne.

2.1.3. Avantages et inconvénients de la vente en ligne

Le e-commerce offre aux nouveaux vendeurs sur Internet de nombreux avantages :

- Les PME bénéficient de l'amélioration de leur image avec une présentation moderne de leur société ;
- Il ouvre un nouveau canal de distribution qui complémente les produits et services de l'entreprise ;
- Il offre une zone d'attraction commerciale très élargie voire même transfrontalière pour un coût de présence relativement faible ;
- Il favorise l'interaction car il permet de développer une relation plus personnelle du vendeur avec le client. Cela facilite ainsi vente « one to one » (personnalisée) et le sur-mesure et permet d'envisager des politiques de fidélisation du client à travers une offre de services et à forte valeur ajoutée ;
- Il facilite grandement les transactions car il évite aux clients de se déplacer et ainsi leur faire gagner du temps ;
- Les données sont enregistrées facilement via un back office et peuvent être traitées aisément (filtres de recherches, suppression, modification, listing, etc.) ;
- Il peut recueillir toutes sortes d'informations sur les habitudes et les besoins de l'internaute. Ainsi, au fur et à mesure de la visite d'un utilisateur, on apprend à connaître ses habitudes par des séries de clics. Cela permet aussi d'adapter le emarketing du site en fonction des profils des consommateurs pour en retirer le maximum de profits ;

- C’est un formidable outil de présélection pour rechercher les meilleurs prix, les meilleurs produits, les derniers modèles, etc. ;
- Le client n’a plus à affronter une quelconque pression de la part des vendeurs ;
- Un site internet e-commerce peut devenir un véritable marché aux puces à grande échelle voire à échelle mondiale.

Malgré ces nombreux avantages, certains facteurs freinent encore le développement de l’e-commerce. Ceci est principalement causé par une résistance psychologique dûe aux faits suivants :

- Le manque de confiance sur la sécurisation des moyens de paiement, même si les méthodes de cryptage de données actuelles assurent une confidentialité quasi parfaite lors de la transaction ;
- La résistance des intermédiaires (grossistes et distributeurs) qui craignent une destruction d’emplois suivie d’une perte de chiffre d’affaires ;
- Les cookies informatiques qui retracent les habitudes des clients ;
- La crainte des clients d’avoir affaire à des marchands malhonnêtes ;
- Il existe toujours des internautes peu expérimentés qui ne se sentent pas à l’aise devant un ordinateur ;
- Le manque de contact avec le produit sans possibilité de le voir en vrai ;
- Les difficultés de recours en cas d’ennuis.

2.2.Approches du e-commerce

Il existe plusieurs stratégies à choisir pour un entrepreneur quand il intègre une solution e-commerce. Néanmoins il y a des principes à respecter pour la création d’un site et aussi des facteurs clés de succès.

2.2.1. Stratégies du e-commerce

Cinq (5) stratégies permettent à l’entreprise d’intégrer le e-commerce :

- **Le « Clic & Mortar »** ou « clic et mortier » : Utilisé par des entreprises qui combinent l’utilisation de la vente par internet avec le magasin traditionnel. Elle offre donc plusieurs possibilités aux clients comme le repérage du produit sur internet et l’achat en magasin, ou encore le repérage du produit en magasin et l’achat sur le net, etc. On peut citer FNAC, SNCF et M&M’s ;

- **La vitrine commerciale :** Il s'agit d'un site institutionnel présentant de manière statique les différents produits de l'entreprise. En d'autres termes, c'est une sorte de catalogue en ligne ;
- **Site marchand ou boutique :** C'est le plus compliqué à réaliser car en plus d'être une vitrine en ligne, un site marchand propose également une solution de paiement en ligne ;
- **Galerie en ligne :** C'est un centre commercial virtuel qui regroupe plusieurs entreprises non concurrentes. Il permet ainsi aux utilisateurs de trouver à la même adresse une offre élargie et complémentaire ;
- **E-comptoire ou plateforme télécommerce :** Il s'agit d'un système clé en main ou package complet en faveur des TPE (Très Petites Entreprises) par un grand opérateur (par exemple Yahoo Store) moyennant un droit d'entrée et une commission sur les ventes. Le client bénéficie en plus d'une animation réalisée par l'hébergeur qui se charge également du paiement sécurisé, de la gestion des bons de commande, des réclamations, etc.

2.2.2. Principes à prendre en compte

L'ouverture d'un site marchand requiert les principes suivants :

- Plus un produit est unique, plus il existe un avantage concurrentiel ;
- Un facteur clé considérable de la réussite du site est d'offrir en complément du produit un ou plusieurs avantages distinctifs. On peut citer l'éventail du choix, une livraison plus rapide, un service personnalisé, ou un prix plus bas que dans le commerce traditionnel ;
- Plus un produit est « dématérialisable », plus il est fait pour le e-commerce comme par exemple : l'édition, l'information, la vente par correspondance, la musique, la vidéo, le tourisme, les voyages, les loisirs, les réservations, les ventes aux enchères, etc. ;
- L'important n'est pas uniquement de concevoir un site riche et grandiose. Il faut aussi être capable d'investir suffisamment et continuellement en promotion, ainsi que d'instaurer une organisation entièrement tournée vers une logistique impeccable.

En plus de ces principes, il est important de prendre en compte les facteurs clés de réussite suivants :

- **Proposer un catalogue complet :** Il y a plus de chances qu'un internaute utilise le site et revienne plus souvent si on met en ligne différentes sortes de produits ou prestations. Ne pas le faire équivaudrait à ne remplir que la moitié des rayons dans une boutique classique ;

- **Rassurer les clients :** La confiance de l'utilisateur envers le site est importante afin qu'il puisse envoyer sa commande. Il faut absolument que les informations restent confidentielles et que le système de paiement soit parfaitement sécurisé ;
- **Proposer un site ergonomique :** Il serait trop contraignant pour l'utilisateur de le retenir trop longtemps avec une interface où il est difficile de se retrouver. Etant exigeant et pressé, il doit pouvoir trouver ce qu'il souhaite rapidement et précisément car tout cyberacheteur fait en général le tour de la concurrence ;
- **Fidéliser le client :** Il faut adopter un bonne politique de fidélisation pour que les clients ne quittent pas facilement le site pour aller vers la concurrence. Parmi les exemples de politique de promotion offensive, il y a les propositions des prix intéressants, les réductions sur les nouveautés, la mise en place d'un programme de fidélisation alléchant ;
- **Faire de la pub en ligne :** Il faut acheter de l'espace publicitaire en ligne, notamment sous forme de bandeaux publicitaires interactifs légers communicant un message clair et attrayant ;
- **Se bouger en permanence :** Il faut sans cesse chercher à améliorer l'attraction du site ou aussi à renouveler son contenu afin de faire revenir le plus souvent. Pour cela on vise la qualité, l'originalité, le choix et l'attrait.

2.3.Points indispensables pour la création de projet e-commerce [22] [23]

2.3.1. Immatriculation

De façon générale, toute personne désirant se lancer dans la vente en ligne doit se doter d'un statut juridique l'y autorisant : auto-entrepreneur, entreprise individuelle, société, etc. La vente d'objets nécessite de s'immatriculer et l'administration fiscale estime que les revenus doivent être déclarés. Le vendeur ne peut pas également délivrer de facture sans statut légal. Le régime de l'auto-entrepreneur est particulièrement adapté à cette activité.

2.3.2. Définition des biens et services vendus

Il faut savoir que tout ce que l'on vend sur un territoire doit respecter la législation locale. On doit ainsi respecter le droit de la propriété industrielle (marques, brevets, dessins et modèles, droit d'auteur, etc.) mais aussi les normes du territoire (règlementation des lotos en ligne, de la vente de voyages, de produits pharmaceutiques, etc.). En outre, il existe des produits qui ne peuvent pas être vendus en dehors de leur réseau de distribution sélective. Par exemple, il y a certains parfums

qui ne peuvent pas être vendus en dehors des points de vente préalablement agréés par le fabriquant.

En somme, il faut toujours se méfier quant aux produits que l'on vend car, même si on n'est pas le fabricant, on peut être jugé responsable si les biens vendus sur le site sont des contrefaçons de droits de tiers.

2.3.3. Trouver les bons fournisseurs

Le choix de fournisseurs, de produits, etc., et du tarif peut poser certaines problématiques. En effet, dans toutes les professions il peut exister des fournisseurs indélicats (prix qui augmentent à la dernière minute, délais de livraison trop longs, frais de ports minimum très élevés, etc.). Cela pourrait être catastrophique car c'est l'image du site qui sera salie quand un client sera déçu.

Les fournisseurs à privilégier sont ceux qui ont un bon suivi de leurs produits (suivi des stocks minimums, des produits épuisés, des arrivages, etc.) et qui tiennent le vendeur informé au bon (moment comme pour les promotions et le changement de prix).

Petit plus, ce serait une bonne idée de demander une réunion-bilan régulièrement pour renégocier les tarifs, obtenir des cadeaux, etc.

2.3.4. Gestion du stock

La gestion du stock peut être un métier usant et long, mais important. Il y a tout d'abord la réception des commandes fournisseurs avec les erreurs de livraisons, les problèmes de références, etc. Il faut toujours vérifier que tout soit bien là, voire vérifier l'état des produits un par un.

Ensuite, il faut faire une classification des produits dans le site avec la gestion des emplacements. Mieux, on peut faire des inventaires réguliers pour être sûr que les stocks sont à jour.

2.3.5. Insertion des mentions légales

Une boutique en ligne ne doit pas être anonyme, et un visiteur ou un client doit pouvoir identifier le vendeur, et savoir comment le contacter. On doit afficher sur la page des mentions légales les renseignements légaux tels que :

- le nom de la société, l'adresse du siège, le pays ;
- le nom du gérant ;
- les moyens de contact du marchand ;
- le numéro et les données indispensables de l'entreprise ;

- le capital de l'entreprise ;
- le nom de l'hébergeur du site ;
- le(s) numéro(s) de déclaration juridique et/ou administratif.

2.3.6. Choix du mode de paiement

Pour la vente en ligne, il est possible de proposer différents types de paiement. Il y a entre autres le paiement par chèque, paiement crypté, par carte bleue (CB), par porte-monnaie électronique, par SMS, par le fournisseur d'accès à internet (FAI), etc. Au responsable du site ensuite de choisir parmi ces modes de paiement proposés. Le paiement de la commande peut être effectué lors de la commande ou à la livraison contre remboursement. Il est à noter que dans ce dernier cas, des frais supplémentaires peuvent être appliqués.

2.3.7. Gestion de la responsabilité

La loi pose un principe de présomption de responsabilité du cybervendeur en énonçant qu'il est « responsable de plein droit à l'égard de l'acheteur de la bonne exécution des obligations résultant du contrat, que ces obligations soient à exécuter par elle-même ou par d'autres prestataires de services, sans préjudice de son droit de recours contre ceux-ci ». Cette protection est renforcée par une responsabilité totale du vendeur dans toute la chaîne d'exécution de la vente, y compris la livraison. Si le produit n'arrive pas chez le client, sa commande doit être livrée à nouveau. Sinon, il doit par exemple se retourner contre le transporteur si le colis a été perdu en cours de livraison.
La limite à cette responsabilité est la mauvaise foi de l'acheteur, l'endommagement du produit ou une mauvaise utilisation du service par l'acheteur. D'où, une obligation d'information importante vis-à-vis des clients, avec des sanctions pénales prévues en cas de non-respect.

2.3.8. Respect du droit de la consommation

En dehors des obligations de droit commun qui pèsent sur les commerçants de manière générale, il existe une législation spécifique qui contraint le vendeur en ligne à respecter un certain nombre de règles :

- L'information préalable du client ;
- Le droit de rétractation des particuliers. C'est une possibilité reconnue aux particuliers de revenir sur leur consentement exprimé lors de la conclusion du contrat. Cela leur permet de retourner un bien commandé sans justification. Toutefois le délai est limité en quelques

jours. Le droit de rétractation s'applique aussi pour les produits soldés, d'occasion ou déstockés ;

- L'obligation d'indiquer une date limite de livraison, date limite à laquelle il devra s'engager à livrer le bien. Il y a aussi la possibilité d'annuler la vente en cas de retard de livraison ou l'indisponibilité du produit par lettre recommandée avec accusé de réception ou par email de résolution de la vente. Au cas où le bien commandé est indisponible, le vendeur doit en informer l'acheteur qui doit pouvoir être remboursé sans délai ou à qui il faudra fournir un bien ou service de qualité et prix équivalents.

2.4.Paypal [26]

2.4.1. Définition

Paypal est un service de paiement en ligne. Il permet à des particuliers et entreprises de payer des achats, de recevoir des paiements, ou d'envoyer et de recevoir de l'argent en ligne. Tout utilisateur peut profiter de ces services en fournissant une adresse de courrier électronique, un mot de passe et en transmettant diverses coordonnées financières à Paypal comme le numéro de carte de crédit. Par la suite, les transactions sont effectuées sans avoir à communiquer de coordonnées financières. Paypal précise aussi qu'en Europe, *« Paypal (Europe) S.à r.l. & Cie, S.C.A est un établissement de crédit sous licence du Luxembourg, conformément à l'article 2 de la loi du 5 avril 1993 sur le secteur financier et son amendement, et est soumis au contrôle de l'autorité luxembourgeoise en vigueur, la Commission de Surveillance du Secteur Financier, sis L-1150 à Luxembourg. Le service se limitant à la monnaie électronique, ce qui ne le qualifie pas de service de dépôt ou d'investissement aux termes de la loi, les clients de Paypal ne sont pas protégés par les programmes de garantie des dépôts financiers du Luxembourg, assurés par l'Association pour la Garantie des Dépôts Luxembourg (AGDL) »*.

2.4.2. Historique

Paypal est une fusion de deux start-ups nommée « Confinity » et « X.com ». En Septembre 1998, Peter Thiel et Max Levchin lancent Fieldlink, permettant aux utilisateurs de stocker des informations cryptées sur Palm Pilots et autres PDA. Ceci dans le but de faire de leurs appareils de véritables portefeuilles numériques. En décembre, ils créent Confinity qui est spécialisé dans la cryptographie et le transfert d'argent entre PDA. En Octobre 1999, un ingénieur de Confinity

développe une démo en ligne qui permet d'envoyer un paiement par e-mail. Quand à X.com, c'est une start-up fondée en mars 1999 par Elon Musk proposant des services de banque en ligne.
En Janvier 2000, Paypal adapte rapidement son fonctionnement pour permettre des paiements sur eBay car les utilisateurs de ce dernier affichent le logo Paypal sur leurs pages de ventes aux enchères afin d'encourager d'autres internautes à créer leurs propres comptes. La communauté eBay adopte ce service et Paypal atteint 100 000 comptes. En 2002, PayPal a été rachetée par eBay du fait qu'environ la moitié des transactions du site utilisaient PayPal et que le système interne à eBay n'était pas aussi populaire.

2.4.3. Services pour les acheteurs et vendeurs

Paypal propose aux acheteurs de payer en ligne sans communiquer ses données bancaires. L'utilisateur peut simplement s'identifier avec son adresse électronique et son mot de passe. Il n'est pas nécessaire d'alimenter son compte Paypal à l'avance. La source d'approvisionnement que l'utilisateur a choisie (carte de paiement ou compte bancaire) est automatiquement débitée au moment de la transaction. En outre, Paypal permet de transférer des fonds d'un compte vers un autre internaute qui dispose d'un compte Paypal.
Néanmoins, les internautes ne peuvent payer avec leur compte Paypal que sur les sites qui acceptent ce mode de paiement. Paypal peut aussi servir à accepter des paiements par carte (CB, Visa, MasterCard, American Express, Aurore, Cofinoga, 4étoiles, Privilèges). Ces paiements sont envoyés directement aux vendeurs sous forme d'emails et peuvent mettre jusqu'à 24h à être visibles sur le compte PayPal par Carte Bancaire, où aucune annulation n'est possible.
L'installation de PayPal est gratuite. Contrairement à la majorité des solutions de paiement proposées par les banques, PayPal ne nécessite pas l'obtention d'un contrat de vente à distance. PayPal se rémunère exclusivement en prélevant une commission sur chaque transaction.
Depuis 2009, Paypal a mis à disposition de sa communauté d'utilisateurs des API pour leur permettre d'innover sur les moyens de paiement de demain. En 2011, Paypal a édité un blog à destination des TPE /PME. Le blog Tiroir-Caisse regroupe des conseils et des articles à destination des sites de commerce électronique.

2.4.4. Principe de fonctionnement de Paypal

Les étapes d'une transaction typique s'opèrent comme suit :

1. L'acheteur se trouve sur la page web du site où le produit ou service peut être acheté via le bouton Paypal intégré ;
2. Après avoir cliqué sur le bouton de paiement, l'acheteur est redirigé sur le site PayPal. Il a le choix entre s'identifier pour utiliser son compte Paypal ou alors saisir ses coordonnées bancaires s'il n'a pas de compte ;
3. Après la saisie des coordonnées, l'acheteur va avoir un récapitulatif de sa commande sur une nouvelle page sur Paypal. Il aura le choix entre confirmer ou annuler sa commande
4. Une fois le paiement validé, un message de validation est affiché sur le compte Paypal. En parallèle à cela, Paypal lance une requête IPN (Instant Paiement Notification) sur notre site d'origine pour que l'on puisse nous aussi traiter le paiement ;
5. Enfin l'acheteur retourne sur le site initial grâce à un lien sur la page précédente.

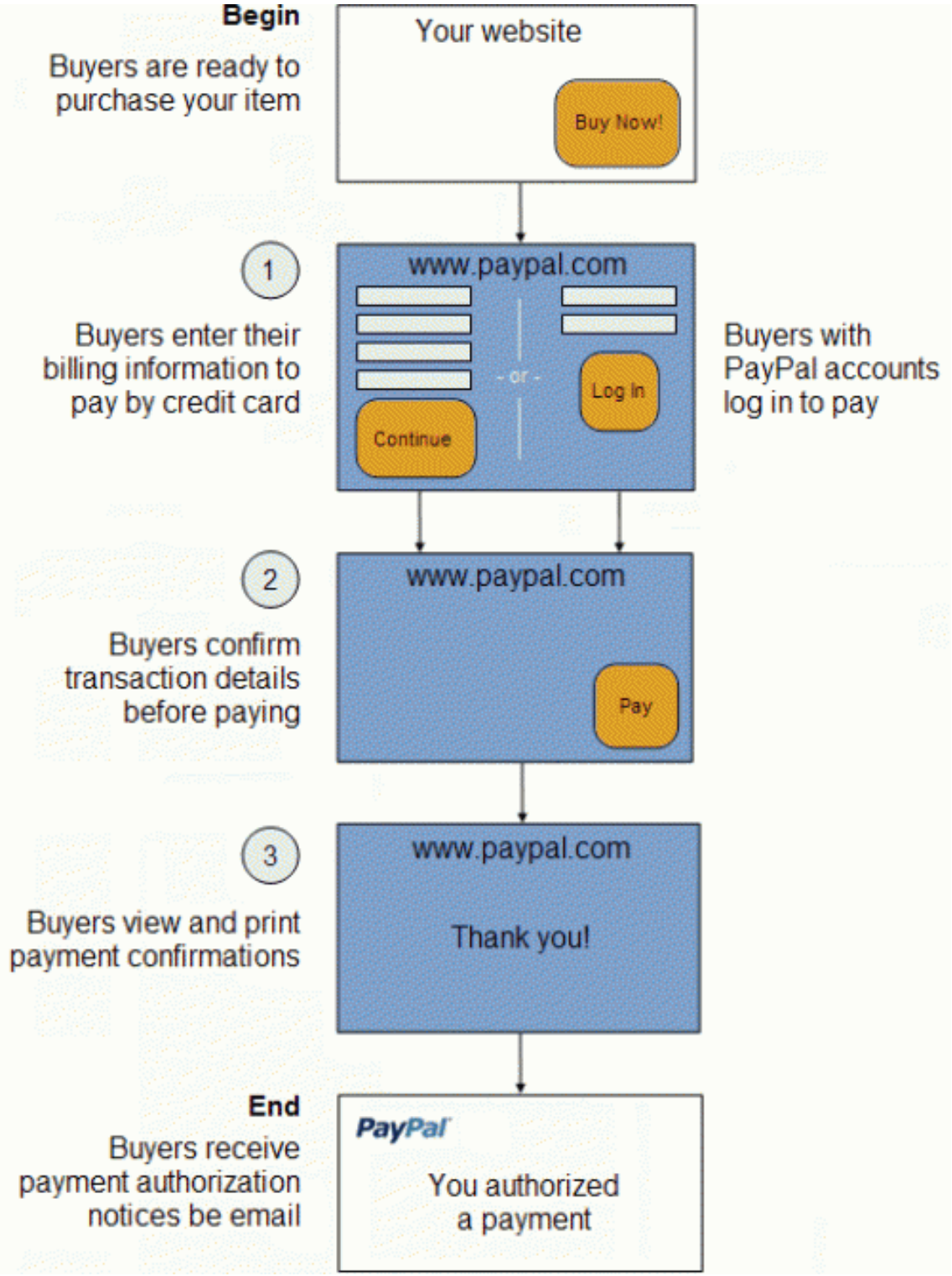


Figure 2.1 : Transaction typique sur Paypal

Outre la page qui va contenir le formulaire de paiement pour l'achat du produit ou du service, on aura besoin d'une autre page qui va traiter la requête IPN.

2.5.E-business (business électronique)

2.5.1. Définition

Le e-business ou « affaires électroniques », correspond à une vaste notion que l'on pourrait définir par l'utilisation de moyens électroniques (particulièrement des Technologies de l'Information et de la Communication) pour réaliser des affaires (du business). Cela pour améliorer le fonctionnement de l'entreprise afin de créer de la valeur pour elle-même, pour ses clients, et pour ses partenaires. Il couvre également tous les processus impliqués dans la chaîne de valeur : les achats électroniques (ou « e-procurement » en anglais), la gestion de la chaîne d'approvisionnement avec le traitement électronique des ordres, le service à la clientèle, et les relations avec les partenaires.

Le e-business s'applique donc aussi bien aux organisations virtuelles sur internet qu'aux entreprises traditionnelles. [24] [25]

2.5.2. Différences entre e-commerce et e-business

Une solution e-commerce présente simplement les solutions permettant à une personne de vendre en ligne ses produits. De ce fait, elle ne garantit pas le succès du site en lui-même. Elle n'offre que les fonctionnalités d'un système e-commerce, en l'occurrence le panier, la gestion du catalogue en ligne, etc.

L'e-business, quant à lui, traite les problèmes fondamentaux liés aux développements d'une communauté verticale autour du site, la fidélisation de la clientèle, l'instauration de la confiance entre vendeur et acheteur, l'optimisation des ventes, l'analyse du comportement du client, etc. Il est tout ce qui peut être mis en œuvre en amont pour concrétiser une vente et par la suite assurer la fidélisation client. Une solution e-business est donc un ensemble d'outils qui permettent d'une part de créer un site de vente en ligne et d'autre part de mettre à disposition du marchand tous les moyens nécessaires pour prospecter, transformer et fidéliser les clients (listes de cadeaux, points de fidélité, remises en espèces (cash back), chèques cadeau, coupons de remise, parrainage, affiliation, etc.). Le e-business est composé de « relations d'échanges » à savoir le mailing, les actions de fidélisation, les promotions, support, le service après-vente, etc.).

Le terme e-commerce ne désigne en réalité qu'une facette du e-business car le e-commerce couvre uniquement l'utilisation d'un support électronique pour la relation commerciale d'une entreprise avec des particuliers. [24] [25]

2.5.3. Caractérisation d'une entreprise

Une entreprise peut être vue comme une entité qui se charge de fournir des produits ou services à des clients dans un environnement en constante évolution. Le fonctionnement d'une entreprise peut être modélisé selon un ensemble de fonctions en interaction. Ces fonctions sont réparties en trois (3) catégories :

- **Les fonctions de réalisation :** Elles représentent le cœur de l'activité de l'entreprise c'est-à-dire la production de biens ou de services. On parle aussi de cœur de métier. Elles concernent les activités de production, de gestion des stocks et de l'approvisionnement (fonction achat) ;
- **Les fonctions de management :** Elles regroupent toutes les fonctions stratégiques de gestion de l'entreprise : direction générale, gestion des ressources humaines et gestion financière et comptable ;
- **Les fonctions support :** Elles servent d'appui aux fonctions de réalisation pour permettre le bon fonctionnement de l'entreprise. Il s'agit de l'ensemble des activités liées à la vente, ainsi que l'ensemble des activités transversales à l'organisation, telle que la gestion des infrastructures technologiques (fonction IT).

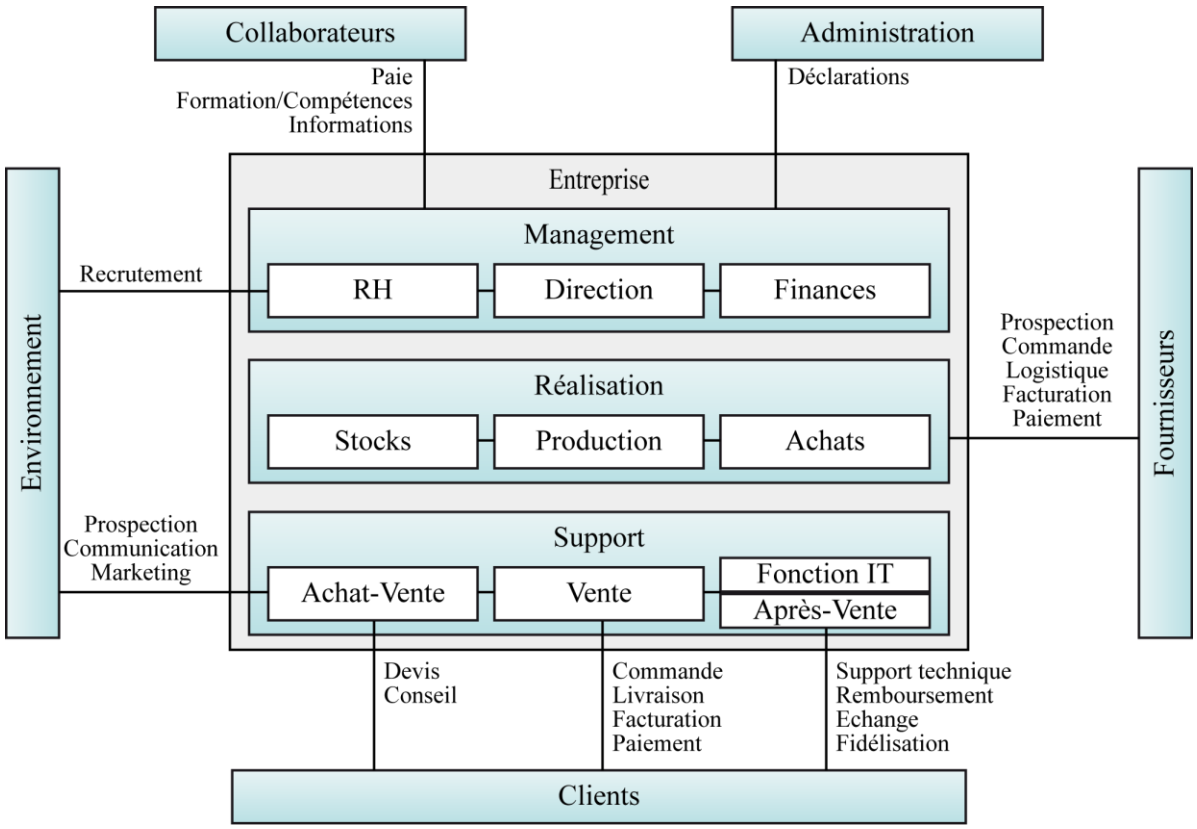


Figure 2.02 : Caractérisation de l'entreprise

2.5.4. Notion de Back Office et Front Office

Les termes Front Office et Back Office désignent littéralement et respectivement les termes « boutique » et « arrière-boutique ». Elles décrivent les parties du système d'information de l'entreprise dédiées respectivement à la relation directe avec le client et à la gestion propre de l'entreprise.

- Le Front Office, aussi appelé Front Line est la partie dédiée à la relation directe avec le client. En d'autres termes, c'est la partie frontale de l'entreprise visible à la clientèle ;
- Le Back Office, à l'inverse, est la partie dédiée à la gestion propre de l'entreprise. Il s'agit de l'ensemble des parties du système d'information auxquelles l'utilisateur final n'a pas accès du fait que c'est l'ensemble des processus internes à l'entreprise (production, stocks, logistique, vente, comptabilité, gestion des ressources humaines, etc.).

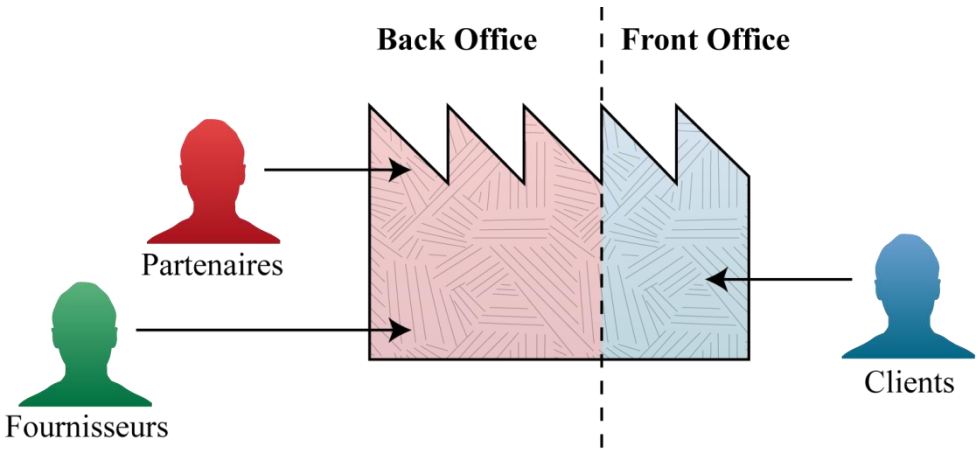


Figure 2.03 : Illustration du Front Office et du Back Office

2.5.5. Présentation des différents concepts

La mise en place d'une démarche e-Business implique la mise en place d'un réseau d'entreprise. C'est via ce réseau que les services spécifiques à l'entreprise seront accessibles en mode client-serveur. Ce système est généralement consultable via une interface web, que l'on peut voir avec un simple navigateur.

Néanmoins, la mise en place d'outils informatique n'est pas suffisante. On considère qu'une entreprise commence réellement un projet e-Business lorsqu'elle met en œuvre une nouvelle organisation tirant partie des nouvelles technologies.

La notion de e-Business est néanmoins très souple et englobe toute les utilisations possibles des technologies de l'information et de la communication (TIC) pour tout ou partie des activités suivantes :

- Rendre plus efficaces les relations de l'entreprise avec ses clients et différents partenaires (fournisseurs, administrations, etc.) ;
- Développer de nouvelles opportunités d'affaires ;
- Fluidifier la circulation de l'information en interne ;
- Mettre sous contrôle les différents processus de l'entreprise (production, stocks, achats, vente, ressources humaines, etc.).

Il s'agit donc de mettre en œuvre des canaux de communication privilégiée entre l'entreprise et son environnement et de les articuler avec ses processus internes afin de maîtriser au mieux les coûts internes et externes.

Voici entre autres les principales technologies du marché :

- Intranet / Extranet ;
- Groupware ;
- Gestion des processus métiers ;
- E-Commerce ;
- Portails d'entreprise ;
- Intégration des applications de l'entreprise (EAI, pour Enterprise Application Integration) ;
- Echange de données informatisées (Enterprise Data Interchange) ;
- Gestion de la Relation Client (CRM) ;
- Gestion de la connaissance (Knowledge Management) ;
- Gestion de la chaîne logistique, notée SCM pour Supply Chain Management ;
- Progiciel de gestion intégré (PGI), appelé également ERP (Enterprise Resource Planning) ;

2.6.Intégration de Paypal sur un site internet [27] [28]

2.6.1. Introduction

La solution de paiement en ligne proposée par Paypal est une solution de paiement facile à intégrer sur un site internet. Elle facilite aussi la vie de bon nombre de sites internet. Dans ce paragraphe, il s’agira de comprendre le fonctionnement de l’intégration de PayPal dans sa configuration basique.

Pour ajouter la fonctionnalité de paiement Paypal sur un site internet, il faut suivre les étapes suivantes :

- Créer un compte développeur et des comptes de tests ;
- Paramétrer le compte vendeur ;
- Ajouter le code source d'intégration Paypal au site internet ;
- Mettre en place l'IPN.

2.6.2. Création et paramétrage des comptes

Avant toute intégration, il est nécessaire de créer un compte développeur sur Paypal en s'inscrivant sur https://developer.paypal.com. C'est cela qui va nous permettre de créer des comptes acheteur et vendeur afin de pouvoir tester le système mis en place. En effet, Paypal a mis en place une sandbox qui permet de tester l'intégration du service et simuler des paiements sans qu'aucun fonds ne soient versés.

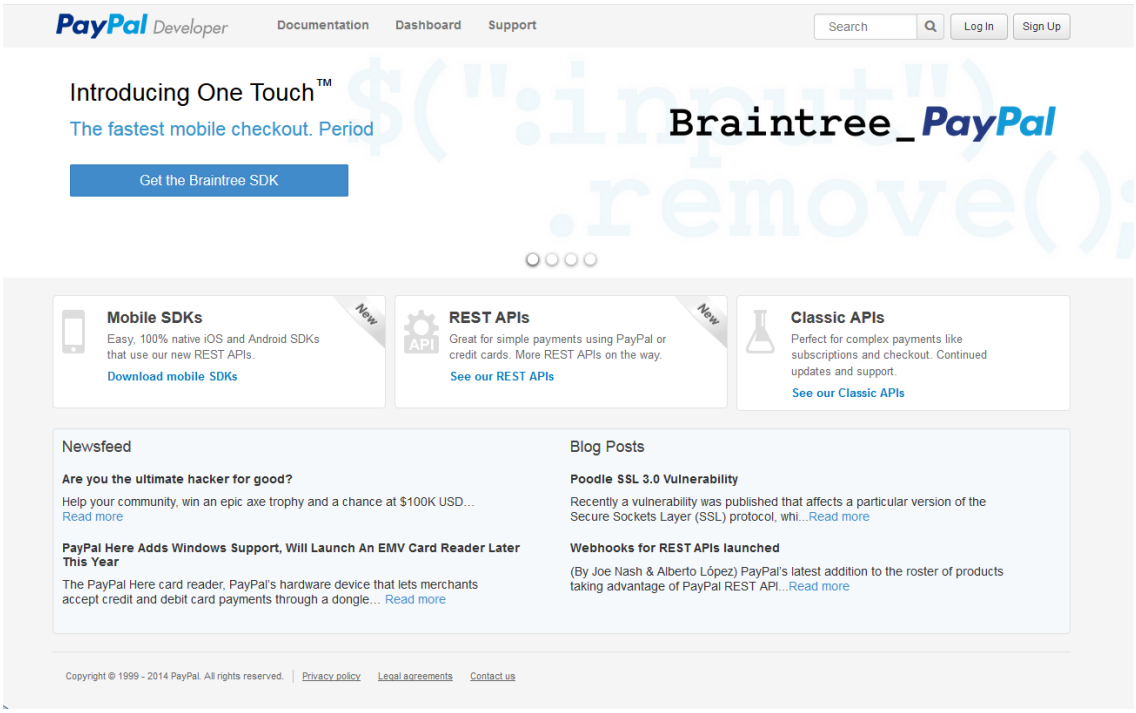


Figure 2.04 : Page d'accueil du site développeur de Paypal

Une fois connecté, il faut créer les comptes de test en accédant au menu « Dashboard » et dans la partie « Sandbox ». C'est dans la sandbox que l'on crée les comptes de test à partir du lien « Créer un compte pré-configuré ».

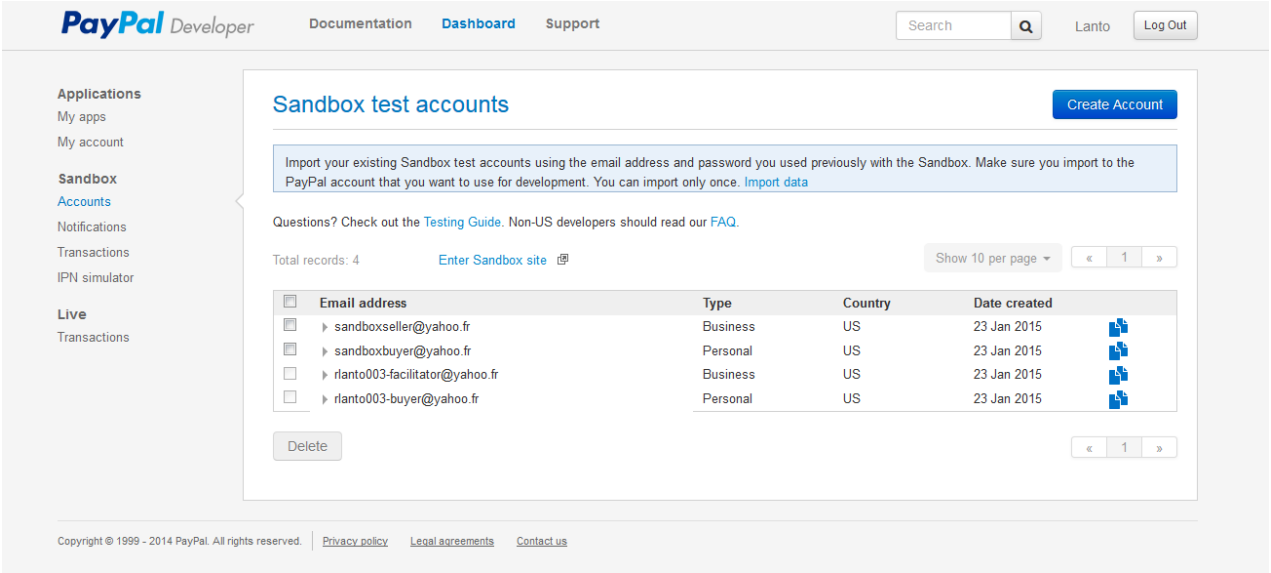


Figure 2.05 : Exemple de sandbox de Paypal

- Le compte vendeur est destiné à simuler les paramètres du vendeur ;
- Le compte acheteur est destiné pour effectuer les achats virtuels.

Il est à noter que ces deux (2) comptes sont en réalité des sous-comptes du compte développeur. Pour se connecter à l'un de ces comptes de test, on accède à l'option « Test Accounts ».

Après la création des comptes, il faut paramétrer le compte vendeur dans www.sandbox.paypal.com. D'abord, il faudra activer l'IPN en allant dans « My Account > Profile > Instant Payement Notification Preferences ».

Maintenant que le compte vendeur est opérationnel, on va choisir un bouton correspondant au mode de vente. Cela se fait dans l'onglet « Merchand Services ». Par exemple, si on veut vendre un simple produit, on clique sur le lien « sell single item ». Différentes options explicites sont ensuite montrées, il suffit de paramétrer le bouton selon les besoins. Les champs principaux étant :

- La description du produit ou du service ;
- Le prix ;
- Redirection vers une URL spécifique du site à l'issu du paiement.

Concernant le prix, il est toujours recommandé de voir la grille des commissions Paypal pour éviter les mauvaises surprises (https://www.paypal-france.fr/marchands/solutions-paiement/tarifs/).

Après la validation du formulaire, on aura à notre disposition un code source que l'on va copier et coller à l'emplacement souhaité dans le site pour l'intégration du bouton.

2.6.3. Analyse du code source d'intégration Paypal

Le code source du formulaire de paiement fourni par Paypal est composé d'une balise HTML <form> avec à l'intérieur différents balises <input> de type hidden et dont les noms sont cités ci-après.

```
<form action="https://www.sandbox.paypal.com/cgi-bin/webscr" method="post">
    <input type='hidden' value="Montant_Achat" name="amount" />
    <input name="currency_code" type="hidden" value="EUR" />
    <input name="shipping" type="hidden" value="0.00" />
    <input name="tax" type="hidden" value="0.00" />
    <input name="return" type="hidden" value="http://votredomaine/paiementValide.php" />
    <input name="cancel_return" type="hidden" value="http://votredomaine/paiementAnnule.php" />
    <input name="notify_url" type="hidden" value="http://votredomaine/validationPaiement.php" />
    <input name="cmd" type="hidden" value="_xclick" />
    <input name="business" type="hidden" value="votre_emailtest_biz@domaine" />
    <input name="item_name" type="hidden" value="Nom de votre produit" />
    <input name="no_note" type="hidden" value="1" />
    <input name="lc" type="hidden" value="FR" />
    <input name="bn" type="hidden" value="PP-BuyNowBF" />
    <input name="custom" type="hidden" value="ID_ACHETEUR" />
    <input
        alt="Effectuez vos paiements via PayPal : une solution rapide, gratuite et sécurisée"
        name="submit"
        src="https://www.paypal.com/fr_FR/FR/i/btn/btn_buynow_LG.gif"
        type="image" />
    <img src="https://www.paypal.com/fr_FR/i/scr/pixel.gif" border="0" alt="" width="1" height="1" />
</form>
```

Figure 2.06 : Exemple de formulaire de paiement

D'abord, la balise <form> englobante possède les attributs suivants :

- « action » : https://www.sandbox.paypal.com/cgi-bin/webscr. C'est la page de Paypal qui va traiter les données que l'on va lui envoyer ;
- « method » : post

Concernant les champs du formulaire qui sont de type « hidden », ils prendront des noms (attribute « name ») spécifique à Paypal.

Attibut « name »	**Explication**	**Valeurs possibles**
Monnaie		
currency_code	Choix de monnaie	EUR, USD, etc.
Shipping	Frais de transport	Nombre entier ou décimal
Tax	Taxe	Nombre entier ou décimal

Pages de retour		
Return	Si le paiement est accepté	URL du site internet
cancel_return	Si la transaction est annulée	URL du site internet
notify_url	Traitement de l'IPN	URL du site internet
Informations sur les acteurs		
Business	Adresse du compter à créditer (compte vendeur)	Adresse e-mail au format valide
Custom	Identifiant de l'acheteur sur le site pour un éventuel traitement lors de la requite IPN envoyée par Paypal	Dépend du site
Commande		
Cmd	Type de bouton cliqué	_xclick : Pour un item unique _cart : Pour des items multiples (panier d'items)
Upload	indique l'utilisation du panier de commande (items multiples)	0 ou 1
item_name	Nom du produit	Chaîne de caractères
Amount	Montant de la transaction	Nombre entier ou décimal
Quantity	Quantité	Nombre entier
Divers		
no_note	Ne pas autoriser l'ajout de commentaire par l'acheteur	0 ou 1
Lc	Langue par défaut	FR, EN, etc.
Bn	Paramètre pour le bouton	PP-BuyNowBF

Tableau 2.01 : Champs du formulaire de paiement de Paypal

Pour utiliser une liste (panier) de produits, il suffit d'ajouter le suffixe « _x » sur les champs relatifs à un produit (item_name, amount, quantity, etc.). « x » est le numéro du produit.

2.6.4. Mise en place de l'IPN

2.6.4.1. Introduction

Une fois qu'un acheteur valide un paiement, Paypal va automatiquement appeler la page renseignée dans le paramètre « notify_url » du formulaire. L'IPN envoie un certain nombre de paramètres POST qu'il va falloir récupérer dans cette page afin de pouvoir traiter le paiement sur notre site.

2.6.4.2. Traitement de l'IPN

Dans cette page de traitement de l'IPN, il faudra suivre les étapes suivantes :

1. Récupérer les données POST reçues venant de Paypal et les stocker dans une variable. Cette dernière va nous permettre de renvoyer toutes les données à Paypal pour vérifier l'intégrité des données. Cette variable est de la forme « &clé=valeur&clé=valeur&clé=valeur » justement pour que le système Paypal reçoive nos données dans l'URL et ainsi retourner une réponse sur notre page de paiement en fonction de l'état du paiement ;
2. Ouvrir une connexion avec le serveur sandbox Paypal (www.sandbox.paypal.com) avec le header adéquat pour obtenir la vérification de l'intégrité des données reçues ;
3. Paypal va ensuite à son tour envoyer des données POST vers notre page. On récupère ces données que l'on va traiter. Il en existe plusieurs mais voici les plus importantes :
    - $_POST["custom"] : Identifiant de l'utilisateur sur le site ;
    - $_POST["txn_id"] : Identifiant de la transaction ;
    - $_POST['payment_status'] : Statut du paiement. Il servira à voir si le paiement est complété (vaut « Completed ») ;
    - $_POST['mc_gross'] : Montant du paiement ;
    - $_POST['mc_currency'] : Monnaie utilisée ;
4. Vérifier si la connexion au service Paypal a réussi. Si c'est le cas, on lui envoie le header précédemment préparé puis on récupère l'information renvoyée par Paypal pour vérifier :
    - si la transaction est valide (le code renvoyé est « VERIFIED ») auquel cas les données envoyées et récupérées sont correctes ;
    - si la transaction est invalide (le code renvoyé est « INVALID ») auquel cas les données envoyées et récupérées sont incorrectes.

2.6.4.3. Validation et enregistrement du paiement en PHP

Il faudra enfin traiter le paiement sur notre site si la transaction est valide, c'est-à-dire à l'étape (4) du traitement de l'IPN quand le code renvoyé est « VERIFIED ». C'est ici que l'on utilise les données POST récupérées à l'étape (3). En effet, elles pourraient être utiles pour des actions dans notre base de données ; par exemple pour mettre à jour le statut des produits du panier de l'utilisateur vu que l'on connaît l'identifiant de l'utilisateur. On pourrait aussi par exemple ajouter une commande en base de données avec le numéro de transaction venant de Paypal.

Cette étape paraît facile mais elle comprend quelques subtilités. En effet, il y a quelques tests à effectuer au préalable :

- Vérifier que le statut du paiement (payment_status) soit bien égal à « Completed » ;
- Vérifier que le numéro de transaction (txn_id) n'a pas déjà été traité ;
- Vérifier que le montant du paiement est correct dans le cas où le prix est fixe.

2.6.4.4. Code source standard

Il existe un code source standard proposé par Paypal, code qui sera intégré à notre page de traitement de l'IPN. Voir la figure ci-après.

```php
// 1. construire la variable avec les données provenant du système PayPal (ajouter "cmd")
$req = "cmd=_notify-validate";
foreach ($_POST as $key => $value) {
    $value = urlencode(stripslashes($value));
    $req .= "&$key=$value";
}

// 2. renvoyer au système PayPal pour validation
$header .= "POST /cgi-bin/webscr HTTP/1.0\r\n";
$header .= "Content-Type: application/x-www-form-urlencoded\r\n";
$header .= "Content-Length: " . strlen($req) . "\r\n\r\n";
$fp = fsockopen ("www.sandbox.paypal.com", 80, $errno, $errstr, 30);

// 3. récupérer les données POST
$item_name = $_POST['item_name'];
$item_number = $_POST['item_number'];
$payment_status = $_POST['payment_status'];
$payment_amount = $_POST['mc_gross'];
$payment_currency = $_POST['mc_currency'];
$txn_id = $_POST['txn_id'];
$receiver_email = $_POST['receiver_email'];
$payer_email = $_POST['payer_email'];
$id_user = $_POST['custom'];

// 4. si la connexion à Paypal est réussie, on vérifie l'état de la transaction
if (!$fp) {
    // erreur HTTP
} else {
    fputs ($fp, $header . $req);
    while (!feof($fp)) {
        $res = fgets ($fp, 1024);
        if (strcmp ($res, "VERIFIED") == 0) {
            // transaction valide
        }
        else if (strcmp ($res, "INVALID") == 0) {
            // transaction invalide
        }
    }
    fclose ($fp);
}
```

Figure 2.07 : Code source standard pour le traitement de l'IPN

Avant de conclure ce chapitre, une importante remarque s'impose. Avant la mise en production du site, il ne faut absolument pas oublier de changer les liens vers la sandbox de paypal (www.sandbox.paypal.com) par paypal (www.paypal.com).

2.7.Conclusion

Clic & Mortar, vitrine commerciale, site marchand, etc., toute entreprise a à sa disposition plusieurs stratégies pour mettre en place son système e-commerce et la bonne solution e-business qui va avec. Cependant, un des problèmes actuels du e-commerce est le règlement des paiements. Le manque de confiance venant de l'utilisateur vis-à-vis de la sécurité de paiement reste un grand facteur de blocage. Pour y remédier, Paypal offre un moyen sécurisé de paiement en ligne tant pour les acheteurs pour effectuer leurs achats avec sérénité, mais aussi pour les développeurs car l'intégration de Paypal sur un site internet est standard et facile à mettre en œuvre.

CHAPITRE 3
MODELES DE PROGRAMMATION

3.1.Open Source

3.1.1. Introduction

La désignation « Open Source » ou « code source ouvert » s'applique aux logiciels dont la licence respecte des critères établis par l'Open Source Initiative. Ce dernier donne la possibilité de redistribuer, d'accéder au code source et d'en créer des travaux dérivés. Le code source est ainsi à la disposition des programmeurs qui collaborent et améliorent ensemble le code source pour les partager au sein de la communauté, qui pourra à son tour y contribuer.

La définition de l'Open Source est une déclaration des droits de l'utilisateur d'ordinateur. Elle définit certains droits que la licence d'un logiciel doit garantir aux utilisateurs pour pouvoir être qualifiée d'Open Source.

En somme, Les droits fournis par l'Open Source permettent aux volontaires qui coopèrent pour créent des produits de garantir les droits suivants :

- Le droit de faire des copies du programme, et d'en distribuer des copies ;
- Le droit d'accéder au code source du logiciel, un préliminaire nécessaire pour pouvoir y apporter des modifications ;
- Le droit d'améliorer le programme. [29]

3.1.2. Historique

Le concept de « logiciel libre » est ancien. C'est le projet GNU, initié par Richard Stallman et développé par des hackers, qui est à l'origine des logiciels libres, dès 1984. Le principe de Stallman est que tout le monde devrait avoir et apprécier de plus de liberté. Il a mis sur pied un ensemble de droits dont il estimait que tout utilisateur devait pouvoir jouir, et les a codifiés au sein de la licence publique générale de GNU, ou GPL. Petit coup de malice, Stallman a intitulé cette licence « copyleft », car au lieu de l'interdire, elle donne le droit de copier.

Stallman a lui-même développé de grands travaux en matière de logiciels libres, tels que :

- Le compilateur de langage C du projet GNU ;
- GNU Emacs, un éditeur disposant d'attraits tels que certains en parlent comme d'une religion.

Ses travaux ont inspiré de nombreux autres développeurs à proposer du logiciel libre selon les conditions de la GPL.

Depuis, la définition de l'Open Source reprend de nombreuses idées de Stallman, et on peut la considérer comme une dérivation de ses propres travaux. Cette désignation « Open Source » a été suggérée par Christne Peterson. Depuis, de très nombreuses communautés de développeurs (dont des fondations) se sont créées et ont ensuite évolué vers des ensembles de contributeurs constitués en société.

3.1.3. Comparaison avec « logiciel libre »

L'emploi de manière interchangeable des termes « Open Source » et « Logiciel libre » est devenu courant. En effet, les licences d'utilisation sont souvent les mêmes donc on ne se soucie pas toujours de la distinction.

Le terme « logiciel libre » désigne des logiciels qui respectent la liberté des utilisateurs qui ont la liberté d'exécuter, copier, redistribuer, modifier et améliorer ces logiciels. Ils supportent le mouvement de la Free Software Definition rédigée par Richard Stallman. La notion d'Open Source repose également sur des définitions précises, mais afin de lever l'ambiguïté sémantique du mot anglais « free » signifiant « libre d'accès » mais aussi « gratuité ». Or l'Open Source n'opère que sur la liberté d'accès au fonctionnement du logiciel.

En pratique, la plupart des licences de l'Open Source satisfont aux critères du libre selon la Free Software Foundation. Les différentes subtilités qui les distinguent étant principalement d'ordre philosophique et commercial, pour rappeler aux utilisateurs qu'un logiciel peut avoir un coût. En outre, en utilisant la désignation « free software », on tient à mettre en avant la finalité philosophique et politique de la licence, tandis que la désignation « open source » met l'accent sur la méthode de développement et de diffusion du logiciel.

3.1.4. Définition de l'Open Source

Pour qu'un logiciel puisse être qualifié d'Open Source, il faut que toutes les conditions suivantes soient remplies. Il faut par exemple qu'elles s'appliquent aux versions dérivées d'un programme aussi bien qu'au programme original. Il ne suffit pas de n'en appliquer que quelques-unes, et il ne suffit pas de ne les appliquer que dans certaines périodes.

- **Libre redistribution**. Cela signifie qu'on peut faire autant de copies du logiciel qu'on le souhaite, les vendre, les donner. On peut aussi donner le logiciel en tant que composant d'un ensemble de programmes de diverses origines. Tout cela sans devoir donner d'argent à qui que ce soit pour bénéficier de ce privilège ;
- **Code source**. Le but est de faire en sorte que le code source soit distribué aux côtés de la version initiale et de tous les travaux qui en dériveront. Il n'est pas autorisé de proposer un code source rendu difficile à comprendre. Le code source est la forme la plus adéquate pour qu'un programmeur modifie ou corrige le programme. Il n'est pas autorisé de proposer des formes intermédiaires, comme ce qu'engendre un préprocesseur ou un traducteur automatique ;
- **Travaux dérivés**. La licence doit autoriser les modifications, les évolutions (correction des bogues, ports vers des nouveaux systèmes, apport d'améliorations), les travaux dérivés, et leur distribution sous les mêmes conditions que celles qu'autorise la licence du programme original. Cependant, on n'exige pas que le producteur d'un travail dérivé utilise les mêmes conditions de licence. On leur donne juste la possibilité de le faire ;
- **Intégrité du code source de l'auteur**. La licence ne peut restreindre la redistribution du code source sous forme modifiée que si elle autorise la distribution de fichiers de correction ou « patch » aux côtés du code source. Ceci dans le but de modifier le programme au moment de la construction. Cette clause donne à l'auteur original la possibilité d'imposer que les modifications soient bien distinctes de leur propre travail, sans pour autant interdire toute modification. La licence peut exiger que les travaux dérivés portent un nom différent ou un numéro de version distinct de ceux du logiciel original. Par exemple, la société Netscape peut insister sur le fait qu'elle seule a le droit de donner à une version du programme le nom de Netscape Navigator, alors que les versions libres du programme doivent porter un nom comme Mozilla ;
- **Pas de discrimination contre des personnes ou des groupes**. La licence ne doit opérer aucune discrimination à l'encontre de personnes ou de groupes de personnes ;
- **Pas de discrimination contre des domaines d'application**. La licence ne doit pas limiter le champ d'application du programme. Par exemple, elle ne doit pas interdire l'utilisation du programme dans le cadre d'une entreprise ou pour la recherche génétique ; le programme doit pouvoir être utilisé aussi bien par une clinique qui pratique des avortements que par une organisation militant contre le droit à l'avortement ;

- **Distribution de licence**. Les droits attachés au programme doivent s'appliquer à tous ceux à qui il est redistribué, sans obligation pour ces parties de remplir des conditions d'une licence supplémentaire. La licence doit s'appliquer automatiquement, sans exiger une quelconque signature ;
- **La licence ne doit pas être spécifique à un produit**. Les droits attachés au programme ne doivent pas dépendre du fait qu'il fasse partie d'une distribution logicielle spécifique. Si le programme est extrait de cette distribution spécifique et est utilisé ou distribué selon les conditions de la licence du programme, toutes les parties auxquelles le programme est redistribué doivent bénéficier des droits accordés lorsque le programme est au sein de la distribution originale de logiciels. Cela signifie que le programme doit rester libre, même séparé de la distribution logicielle avec laquelle il a été fourni ;
- **La licence ne doit pas restreindre d'autres logiciels**. La licence ne doit pas imposer de restrictions sur d'autres logiciels distribués avec le programme licencié. Par exemple, la licence ne doit pas exiger que tous les programmes distribués sur le même support soient aussi Open Source. Par exemple, une version de GhostScript (programme de rendu de PostScript) exige que le support sur lequel est distribué ce programme ne contienne que des logiciels libres. Les licences Open Source ne permettent pas cela. Ainsi, l'auteur du programme GhostScript distribue une autre version (un peu plus ancienne) de ce programme, couverte par une licence vraiment Open Source. [30]

3.1.5. Compétition par rapport aux solutions propriétaires

Les logiciels Open Source sont développés selon un mode de travail collaboratif, avec des contributions des membres de la communauté. Le code source peut être relu et amélioré par tous, ce qui peut permettre notamment la correction de problèmes de sécurité. De ce fait, la qualité des logiciels produits est mise en avant.

En revanche, ceux qui ne proposent pas des programmes Open Source se trouvent face à une concurrence de plus en plus rude et doivent savoir que les utilisateurs apprécient et exigent les droits qu'ils auraient dû avoir depuis toujours. Par exemple, des programmes comme ceux qui constituent le système d'exploitation Gnu/Linux et le navigateur web de Netscape ont acquis une grande popularité, en prenant la place de logiciels couverts par des licences plus restrictives.

Les sociétés qui utilisent un logiciel Open Source profitent des avantages liés à un modèle de développement très réactif. Ce modèle est souvent mis en place par plusieurs sociétés qui

coopèrent. Une grande partie du travail est fournie par des individuels qui ont tout simplement besoin d'une amélioration qui répond mieux à leurs besoins.

3.2.Programmation orientée objet [31]

3.2.1. Historique

La POO (Programmation Orientée Objet) est un style de programmation élaboré au début des années 1960 par Ole-Johan Dahl et Kristen Nygaard, poursuivi ensuite par Alan Kay.

Le langage Simula-67 pose les constructions qui seront celles des langages orientés objet à classes : classe, polymorphisme, héritage, etc. Mais c'est réellement par et avec Smalltalk 71 puis Smalltalk 80 (Dan Ingalls), inspiré en grande partie de Simula 67 et de Lisp, que les principes de la programmation par objets (résultat des travaux d'Alan Kay) sont véhiculés : objet, encapsulation, messages, typage et polymorphisme, etc. Smalltalk aussi plus qu'un langage à objets, c'est un environnement graphique interactif complet.

3.2.2. Définition

La Programmation Orientée Objet est un concept de programmation puissant et pratique qui consiste en la définition et l'interaction de briques logicielles appelées « objets ». Un objet représente un concept, une idée ou toute entité du monde physique (par exemple une voiture, une personne, etc.). Cet objet possède une structure interne et un comportement, et il sait interagir avec ses pairs. Il s'agit donc de représenter ces objets et leurs relations. L'interaction entre les objets via leurs relations permet de concevoir et réaliser les fonctionnalités attendues, et de mieux résoudre les problèmes.

3.2.3. Procédural et orienté objet

Il existe deux (2) grands types de programmation :

- La **programmation procédurale ou structurée** (comme le C). Dans cette programmation, on définit des fonctions et des procédures qui sont destinées à s'appeler. Chaque fonction effectue un traitement particulier. Globalement, le procédural travaille sur l'action ou le verbe. Prenons exemple sur une voiture : pour calculer la vitesse d'une voiture, la programmation procédurale amènera à appeler une fonction « calculerVitesse(voiture) », et cela partout où cela est nécessaire. Le code faisant référence à une voiture est donc « fondu » et dispersé dans l'ensemble des programmes, pouvant mener rapidement à des difficultés en cas de modification de la structure des données ;
- La **Programmation Orientée Objet** amène à manipuler uniquement des objets contenant un ensemble groupé de variables et de méthodes. Dans cette configuration, toujours dans notre exemple sur la voiture, le sujet est prépondérant : disposant d'un objet « voiture », on effectuera spontanément « voiture.calculerVitesse() » . Du coup, tout le code touchant à l'objet « voiture » se trouve ainsi regroupé.

3.2.4. *Notion de classe et d'objet*

Une classe est une structure de données qui regroupe des propriétés et des comportements. Par exemple, une classe Voiture peut avoir des propriétés (couleur, immatriculation, puissance, etc.) et des comportement (avancer(), reculer(), stopper(), etc.).

En orienté objet, les propriétés sont appelées « variables d'instance » et les comportements sont appelés « méthodes ».

Un objet est quant à lui une instance de classe. En d'autres termes, c'est un exemplaire créé à partir de la classe en donnant des valeurs à certaines propriétés. Une classe peut être vue comme une moule qui, quand on le remplit, donne un objet ayant la forme du moule ainsi que toutes ses caractéristiques. Par conséquent, on déclare comme type une classe et on déclare comme variables de ce type des objets. Par exemple, une voiture « Peugeot 206 » et une autre « BMW M3 » sont des instances de la classe « Voiture », c'est-à-dire des voitures avec chacune des propriétés spécifiques.

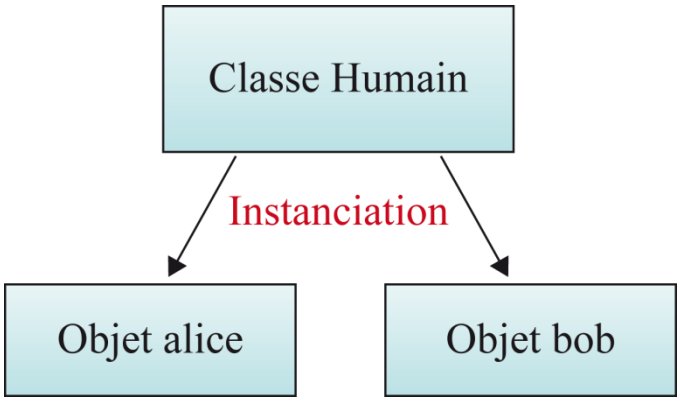


Figure 3.01 : Objets instances d'une classe « Humain »

3.2.5. ***Notion d'encapsulation***

Pour une classe, les attributs et/ou méthodes peuvent être soit visibles, soit cachés aux yeux d'un programmeur extérieur : c'est le principe « d'encapsulation ». Cela permet de masquer l'ensemble des procédures et fonctions destinées à la gestion interne de l'objet. Par l'encapsulation, le programme peut modifier la structure interne des objets ou leurs méthodes associées sans avoir d'impact sur les utilisateurs de l'objet.

L'encapsulation permet de garder une cohérence dans la gestion de l'objet, tout en assurant l'intégrité des données qui ne pourront être accédées qu'au travers des méthodes visibles.

3.2.6. ***Notion d'héritage***

L'héritage est un mécanisme qui permet de spécialiser une classe. Ainsi, la classe qui en découle (appelée « classe fille ») possédera non seulement les propriétés et méthodes de sa mère mais également d'autres méthodes spécifiques ou redéfinies. On parler de faire « dériver » la classe en une classe fille.

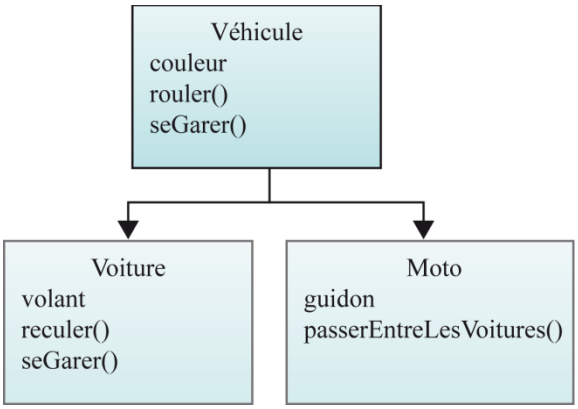


Figure 3.02 : Les classes « Voiture » et « Moto » dérivant de la classe Véhicule.

Dans cet exemple, la classe « Voiture » :

- conserve la propriété de la classe mère « couleur » et la méthode « rouler() » ;
- redéfinit la méthode de la classe mère « seGarer() » ;
- possède une nouvelle propriété « volant » et une nouvelle méthode « reculer ».

3.2.7. *Notion de polymorphisme et de transtypage*

Dans le cas d'un héritage, les objets sont dits « polymorphes » car ils possèdent plusieurs types :

- Le type de leurs classes ;
- Mais également le type de leurs classes mères.

De ce fait, le « polymorphisme » est le fait qu'un objet puisse appartenir à plusieurs types. Par exemple, un objet de type « Voiture » est aussi de type « Véhicule ».

Il est possible alors de forcer le programme à voir un objet comme un type différent de son type initial là où est attendu en objet d'un certain type. C'est le « transtypage » ou « cast ». A noter seulement qu'il y a transtypage de la fille vers la mère et non l'inverse : une classe mère n'est pas du type de sa fille.

En outre, comme l'objet hérite des attributs et méthodes de ses ancêtres, il garde toujours la capacité de pouvoir appeler ou même redéfinir une méthode afin de la réécrire ou de la compléter pour un besoin spécifique.

3.2.8. *Notion de classes abstraites et d'interfaces*

Une classe abstraite est identique à une classe normale. Sa particularité est que c'est une classe que l'on ne peut instancier. Elle sert à définir une classe mère (superclasse) que l'on va hériter. Ses enfants pourront ainsi utiliser les attributs et redéfinir méthodes jugés comme communes.

Une interface, quant à elle, est une classe intégralement abstraite. C'est un ensemble de constantes et de déclarations de méthodes (sans aucun corps). C'est une sorte de standard auquel une classe peut répondre. En effet, toutes les classes qui implémentent cette interface possèdent les méthodes et les constantes déclarées dans celle-ci. Une interface ne fait que donner une liste de méthodes qui seront à définir dans les classes qui implémentent l'interface.

3.3.Architecture multicouches

3.3.1. Problématique

Lors de la démocratisation de l'informatique, dans les premiers programmes de gestion, l'interface utilisateur faisait tout sur la base de données. Il lisait, modifiait la base de données et enregistrait celle-ci.

De ce fait, avec l'évolution de ces logiciels, les développeurs ont très vite aperçu les limites de cette technique. Il a fallu appliquer la notion de travail sur des couches dans la programmation fonctionnelle. Grâce à un modèle multicouche d'une application, on peut rendre son code modulaire et facile à maintenir. Qui plus est, il faut savoir que les questions d'architecture sont importantes. Il est primordial de connaître l'architecture à utiliser avant de se lancer dans des développements effrénés. Cela permet d'éviter un véritable « pêle-même » dans l'écriture du code et permet de faire des développements plus méthodiques, avec des classes plus indépendantes et permet ainsi une facilité de maintenance de l'application.

3.3.2. Architecture 2-tiers et 3-tiers [34]

L'environnement de base est le modèle client/serveur qui est en 2-tiers. Il désigne aussi un mode de dialogue dans un réseau. Le serveur est celui qui contient les ressources qu'un ou plusieurs clients peuvent ou non consulter selon les privilèges attribués. Le Client sollicite le serveur, qui, quant à lui, attend les requêtes du (des) client(s) et envoie des réponses. En général, le client désigne la machine sur laquelle est exécuté le logiciel client d'accès au serveur, et le serveur est la machine sur laquelle est exécuté le logiciel serveur contenant les ressources.

Le modèle 2-tiers se schématise ainsi :

- **Couche client :** Contient l'interface utilisateur qui se connecte à l'adresse du serveur pour pouvoir effectuer des requêtes ;
- **Couche serveur :** Contient toutes les ressources (fichiers, base de données, etc), reçoit les requêtes et renvoie des données.

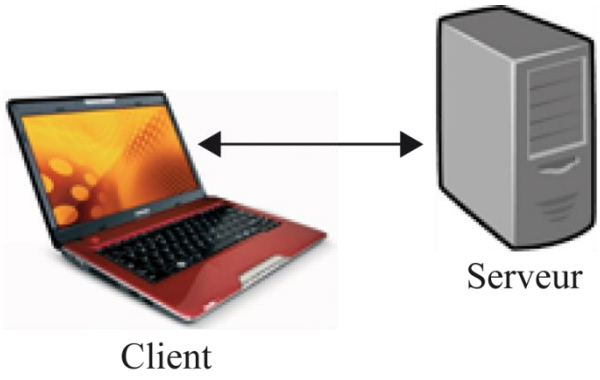


Figure 3.03 : Architecture client/serveur

L'architecture 3-tiers a pour but de séparer une application en trois (3) couches logicielles, dont le rôle est clairement défini :

- **Tiers présentation des données :** Contient l'interface utilisateur ou IHM (Interface Homme Machine) qui correspond à l'affichage, la restitution sur le poste de travail, le dialogue avec l'utilisateur ;
- **Tiers métier :** Correspond à la mise en œuvre de l'ensemble des règles de gestion et de la logique applicative. C'est à ce niveau que se trouvent toutes les règles de gestion, et toute l'intelligence de la démarche ;
- **Tiers d'accès aux données :** Correspond aux données qui sont destinées à être conservées. Les programmes du deuxième tiers font appel à ce dernier pour consulter ou mettre à jour les données issues de la base de données.

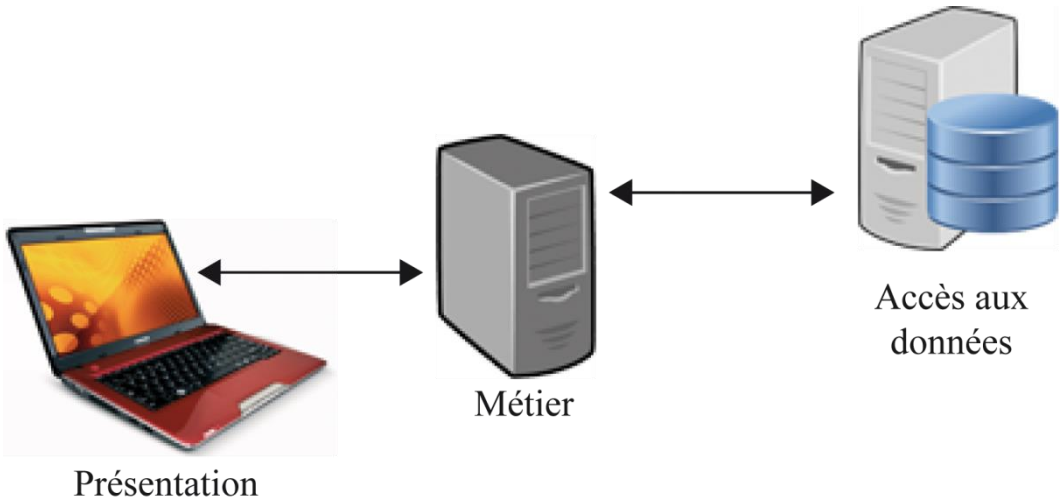


Figure 3.04 : Architecture 3-tiers

3.3.3. Architecture n-tiers [31] [33] [34]

Il existe plusieurs ouvrages qui montrent des techniques de séparation des couches pour une application. Mais au final, le but est de faire le plus d'abstraction possible d'une couche à une autre. En général, chaque couche est le client d'une autre et est le serveur d'une autre ; chaque couche peut offrir ses ressources à une autre et peut puiser celles d'une autre.

A partir du schéma 3-tiers, on tire l'architecture n-tiers en rajoutant des couches supplémentaires pour obtenir une architecture à 4, 5, …, n-tiers. Chaque couche regroupe des objets ayant les mêmes rôles pour avoir des classes découplées reposant le moins possible sur d'autres classes.

Voici un exemple d'architecture à 5 couches :

- **Couche présentation ou IHM :** Contient les composants de l'Interface Homme Machine avec le code propre à l'affichage des données ;
- **Couche application :** Contient tous les contrôleurs de cas d'utilisation de l'application. Cette couche assure le lien entre la couche présentation et la couche métier ;
- **Couche objets ou BO (Business Object) :** Contient les objets de données pures (objets métiers ou entités) qui transitent entre toutes les autres couches ;
- **Couche d'accès aux données ou DAL (Data Access Layer) :** Contient les composants permettant d'alimenter en données les différents objets métiers manipulés par l'application, et de permettre le stockage de ces objets en base de données, fichiers bruts, XML, etc. ;
- **Couche métier ou BLL (Business Logic Layer) :** Contient tous les composants métier qui fournissent un moyen d'accéder aux données fournies par la couche DAL, sans avoir à connaître la technologie de stockage employée.

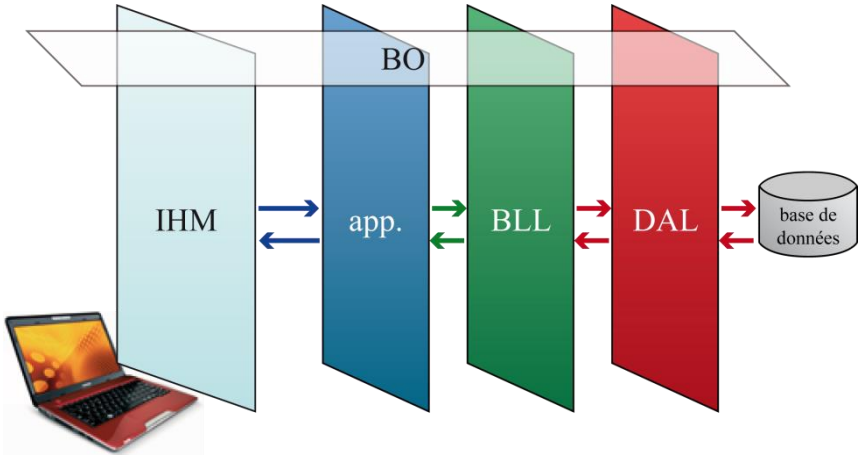


Figure 3.05 : Exemple d'architecture n-tiers

3.4.Pattern MVC [32] [34]

3.4.1. Définition

Le patron MVC (Modèle Vue Contrôleur) est, comme tout patron de conception, un modèle destiné à répondre aux besoins des applications interactives. Il sépare les problématiques liées aux différents composants au sein de leur architecture respective.

Le MVC offre une façon d'organiser et de structurer le code source d'un programme en distinguant trois (3) entités distinctes qui sont, le Modèle, la Vue et le Contrôleur ayant chacun un rôle précis dans l'interface. Le cadre est alors normalisé pour structurer l'application et facilite ainsi le dialogue entre les concepteurs.

Il a été introduit par Trygve Reenskaug en 1978-1979 pour proposer une solution générale aux problèmes d'utilisateurs manipulant des données volumineuses et complexes.

3.4.2. Découpage

Dans une architecture MVC, les trois (3) entités sont les suivantes :

- Le Modèle ;
- La Vue ;
- Le Contrôleur.

3.4.2.1.Modèle

Le Modèle représente les données manipulées par le programme. Il décrit les données manipulées, assure le traitement de ces données, garantit leur intégrité et les interactions avec la base de données. S'il y a une base de données, c'est le Modèle qui la contient. Le Modèle offre des méthodes pour mettre à jour (insertion, suppression, mise à jour) et récupérer les données.

Les résultats renvoyés par le Modèle ne s'occupent pas de la présentation. Ainsi découplé, il n'a aucun lien direct vers le Contrôleur ou la Vue.

3.4.2.2.Vue

La Vue fait l'interface avec l'utilisateur. Sa première tâche est d'afficher les données renvoyées par le Modèle par l'intermédiaire du Contrôleur. Sa seconde tâche est de recevoir toutes les actions de l'utilisateur (clic de souris, sélection d'une entrée, bouton, etc.). Pour les traitements des requêtes, ces différents événements sont envoyés au Contrôleur. La Vue n'effectue pas de traitement mais se contente juste d'afficher les données reçues et d'interagir avec l'utilisateur.

3.4.2.3.Contrôleur

Le Contrôleur est chargé de la synchronisation du Modèle et de la Vue. Il contient toute la logique du code ; on y inclut le Modèle pour manipuler les données pour ensuite renvoyer une Vue à l'utilisateur. En fonction de la requête de l'utilisateur, il enclenche les actions à effectuer et met en relation la Vue et le Modèle. Si une action nécessite un changement des données, le Contrôleur demande la modification des données au modèle et avertit ensuite la Vue que les données ont changé pour que celle-ci se mette à jour.

3.4.3. Interactions

Les interactions entre le modèle, la vue et le contrôleur sont résumées la figure suivante :

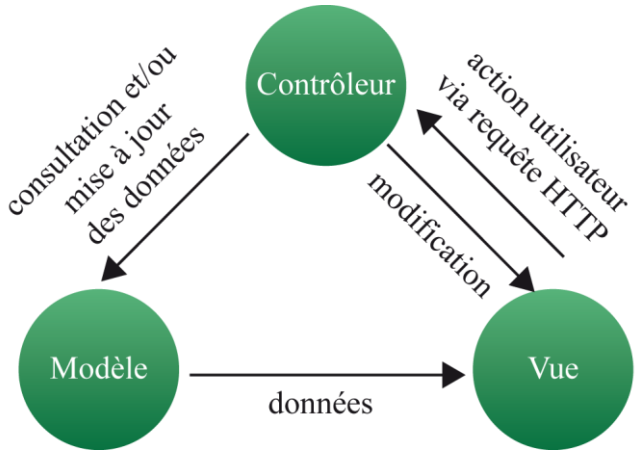


Figure 3.06 : Interactions entre le modèle, la vue et le contrôleur

3.4.4. Avantages

Le modèle MVC impose une architecture claire, ce qui simplifie la maintenance et l'amélioration sur un projet. Par exemple, la modification des traitements ne change en rien la vue si on décide de changer le stockage de données (XML vers SQL), en changeant uniquement les traitements d'interaction avec la base de données dans la couche Modèle.

3.5.Pattern MVVM (Model-View-View Model) [35]

3.5.1. Définition

Le concept MVVM pour Model View View Model est une variation du patron de conception MVC.

Il est apparu en 2004 et est originaire de Microsoft, très souvent utilisé par des bibliothèques Javascript et des technologies Microsoft telles que Silverlight. Cette méthode a pour but de séparer la vue de la logique d'accès aux données et ainsi simplifier l'écriture des interfaces graphiques.

3.5.2. Principe

Le modèle MVVM est destiné pour des applications où le Modèle n'est pas directement exploitable par la Vue. Pour cela les classes de Modèle de Vue sont en charge de fournir des propriétés qui seront directement exploitables depuis la Vue, sans faire aucun traitement.

Plutôt que d'effectuer les traitements sur un Modèle dans une Vue, c'est le Modèle de Vue qui est en charge de faire les traitements et d'accéder au Modèle. Une Vue ne doit faire qu'afficher les données et en aucun cas faire de quelconques traitements. Ce Modèle de Vue prépare alors les données afin qu'elles soient affichables par la Vue. En d'autres termes, le Modèle de Vue peut être vu comme un adaptateur entre la Vue et le Modèle.

3.5.3. Découpage

Les composants du MVVM sont :

- **Model** ou le Modèle (les données). Il s'agit, comme en MVC, de la couche de données métiers, généralement de classes qui contiennent des données sans avoir à connaître la technologie de remplissage des données, (XML, base de données, service web, etc.) ;
- **View** ou la Vue. Elle correspond à ce qui est affiché dépendamment des données du Modèle ;
- **View Model** ou littéralement Modèle de Vue. C'est le lien entre le Modèle et la Vue. Il s'agit d'une classe qui fournit une abstraction de la Vue en s'occupant des liaisons de données et des éventuelles conversions. Il s'appuie sur la puissance du binding pour mettre à disposition de la Vue les données du Modèle.

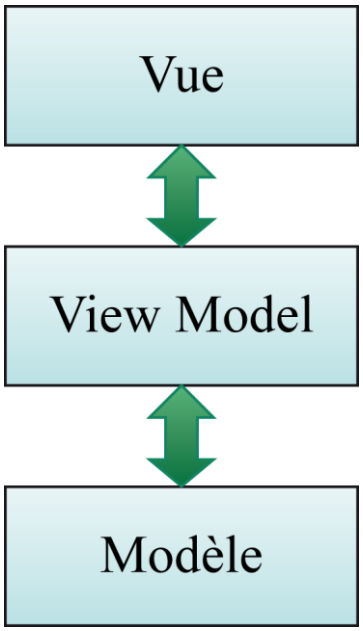


Figure 3.07 : Schéma du design pattern MVVM

3.6.Généralités sur un framework

3.6.1. Définitions

Le mot « framework » est formé de deux (2) mots anglais :

- « frame » : cadre ;
- « work » : travail.

Littéralement, un framework est un « cadre de travail ». Concrètement, il s'agit d'un ensemble de composants qui servent à créer les fondations, l'architecture et les grandes lignes d'un logiciel. Ces composants vont servir aux développeurs comme base pour leurs projets.

Un framework est donc une sorte de boîte à outils conçue par un ou plusieurs développeurs à destination d'autres développeurs. Contrairement à certains scripts tels que WordPress, un framework n'est pas utilisable tel quel ; il n'est pas fait pour être utilisé par les utilisateurs finaux. Un framework est plutôt conçu pour un développeur qui, à son tour va s'en servir pour construire son projet à lui.

3.6.2. Objectifs

L'objectif premier d'un framework est d'améliorer la productivité des développeurs finaux qui l'utilisent. Un framework offre la possibilité au développeur final d'en utiliser certains composants pour lui faciliter le développement. Il pourra ainsi se concentrer uniquement sur le plus important

de son travail en utilisant l'architecture qui lui est offert. Autrement dit, le framework s'occupe de la forme et permet au développeur de se concentrer sur le fond.

3.6.3. Avantages

Les avantages principaux d'un framework sont :

- Le gain en productivité ;
- La lisibilité du code ;
- L'amélioration du travail ;
- L'aide de la communauté.

Un framework permet aux développeurs de réaliser un code bien organisé et bien architecturé. De ce fait, le projet devient facilement maintenable et évolutif. De plus, un framework offre des composants prêts à être utilisés, ce qui permet d'éviter de réinventer la roue à chaque projet. On économisera ainsi des heures de développement.

Ensuite, un framework améliore la façon de travailler en équipe (avec d'autres développeurs PHP et des designers). En particulier, un framework architecturé en MVC sépare le code PHP du code HTML. Ainsi, un designer peut travailler sur des fichiers différents que les développeurs. D'autre part, un framework a une structure et des conventions de code connues. Ainsi, le code devient lisible pour un autre développeur et il s'intégrera très rapidement au projet.

Enfin, le dernier avantage est la communauté soutenant chaque framework. C'est elle qui fournit les tutoriaux, cours et diverses documentations, de l'aide sur les forums, et bien sûr les mises à jour. Ces mises à jour sont très importantes car on pourra utiliser un code complet sur une fonctionnalité sur laquelle on pourrait être bloqué.

3.6.4. Inconvénients

Le plus grand inconvénient d'un framework est l'apprentissage. Pour pouvoir utiliser un framework, il faut un certain temps d'apprentissage pour en maîtriser le fonctionnement. Chaque composant d'un framework a sa propre complexité qu'il faudra appréhender une par une. Néanmoins, on peut voir l'apprentissage d'un framework comme un investissement : il y a un certain effort à fournir au début, mais les résultats se récoltent ensuite sur le long terme.

3.7.Conclusion

Les différents concepts de programmation présentés dans ce chapitre ne sont pas tous obligatoires, certes, mais sont fortement conseillés. Ils forment un ensemble de conseils raisonnables de présentation des programmes, ainsi que des conseils de programmation qui ont recueilli l'assentiment de programmeurs chevronnés. Chaque concept porte son lot d’avantages, et fournit les bonnes pratiques à adopter pour un code optimisé et maintenable. Le temps passé à taper les programmes est négligeable par rapport au temps passé à les lire, c'est pourquoi il est primordial de réaliser des programmes dont la lisibilité est optimale.

CHAPITRE 4
CONCEPTION ET REALISATION DU FRAMEWORK

4.1.Introduction

Non pas dans le but de fournir une complète documentation – ce qui prendrait certainement un nombre considérable de pages - ce chapitre est plutôt écrit dans le but de présenter succinctement le framework que l'on a créé pour la réalisation de ce présent mémoire. D'abord, nous verrons dans ce paragraphe quelques notions utiles pour la compréhension de son mécanisme expliqué dans la suite de l'ouvrage. Nous verrons entre autres :

- La raison d'être du framework ;
- Les bundles ;
- Le MVC appliqué au framework ;
- Le routing.

4.1.1. Raisons d'être du framework

Le langage PHP a maints avantages tels la gratuité, la facilité de prise en main et d'intégration. Actuellement, PHP figure parmi les trois (3) langages de développement les plus populaires (source langpop.com) donc des millions d'utilisateurs développent avec PHP. Il peut alors devenir victime de son succès. Souvent, faire du PHP natif équivaut à faire du « code spaghetti » soit du code non structuré et pratiquement en désordre. En outre, au cours de chaque projet, les tâches deviennent répétitives et réutiliser le code devient difficile, voire impossible. Nous nous permettons alors de proposer une architecture que tout développeur final pourra ensuite utiliser pour réaliser ses projets PHP afin d'avoir un code lisible, réutilisable et maintenable. Nous allons voir que le framework est divisé en plusieurs composants, chaque composant remplissant un rôle précis.

4.1.2. Bundles

4.1.2.1.Définition

De manière simple, par définition, un bundle correspond à une fonctionnalité du site. Ainsi, une fonctionnalité est regroupée dans un même endroit - c'est-à-dire en un bundle - ce qui permet d'avoir une bonne séparation de code, de découper les fonctionnalités et de mieux s'y retrouver en

cas de maintenance. Par exemple, on peut avoir un bundle dédié pour les utilisateurs du site, un autre pour les produits en vente, un autre pour les articles et nouvelles, etc.

De plus, plusieurs bundles peuvent s'interagir et fonctionner ensemble. Bien sûr cela est totalement géré par le framework. Toujours dans notre exemple, un bundle dédié pour les utilisateurs est lié à un bundle dédié pour les articles si on considère le fait qu'un utilisateur écrit un article.

A part le fait que le code est mieux organisé, un bundle peut être réutilisable pour un autre projet. Cela présente un énorme avantage car cela permet d'accélérer le temps de développement du fait de la réutilisabilité du code.

4.1.3. MVC appliqué au framework

Concernant l'architecture MVC appliqué à notre framework, il faut savoir d'ores et déjà que ce n'est pas l'application toute entière qui est organisée en MVC. A vrai dire, c'est chaque bundle qui est organisé en MVC. Autrement dit, le code source d'un bundle est composé de fichiers de contrôleurs, de fichiers de modèles et de fichiers de vues.

Pour fournir une page web, le mécanisme est assez simple :

1. La requête venant de l'utilisateur appelle un contrôleur contenu dans le bundle demandé ;
2. C'est une méthode (ou action) de ce contrôleur qui est exécutée ;
3. Cette action de contrôleur se charge, si besoin est, d'interagir avec le modèle ;
4. La méthode du contrôleur renvoie alors une vue avec les données qu'on souhaite lui passer (données venant du modèle, variables préalablement crées, etc.).

Très succinctement, dans le modèle, on va retrouver plusieurs couches :

- Les modèles qui représentent la couche objets c'est-à-dire contenant les données pures (objets métiers) qui vont transiter entre les autres couches ;
- Le QueryBuilder qui représente la couche d'accès aux données ;
- Le Core qui représente la couche métier.

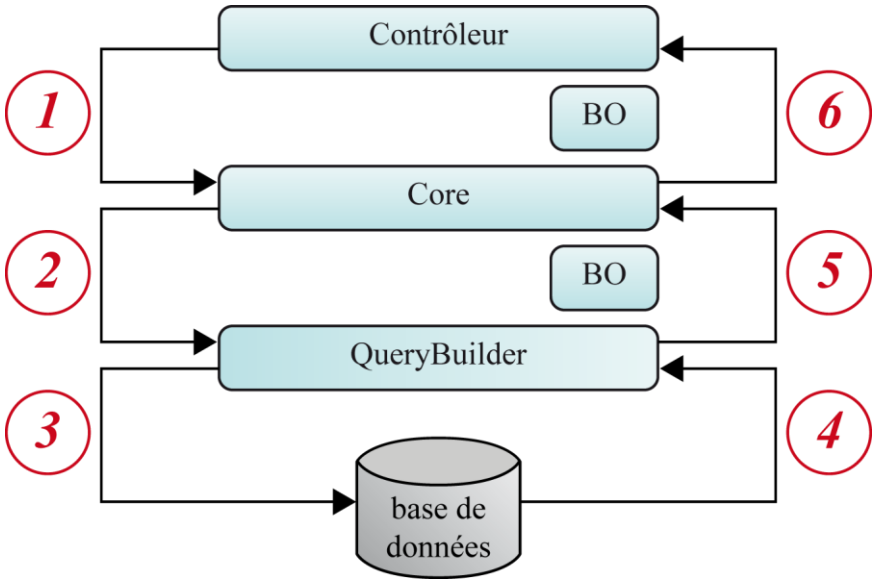


Figure 4.01 : Couches modèles sollicitées par le contrôleur

Cette figure, qui représente les différentes couches du modèle, est décrite de la manière suivante :

1. Le contrôleur sollicite le Core pour fournir et traiter les données, la plupart du temps des objets métiers (BO) venant de la base de données ;
2. Le Core sollicite le QueryBuilder qui est le seul à pouvoir accéder à la base de données ;
3. Le QueryBuilder interagit avec la base de données pour récupérer ou modifier les lignes d'entrées adéquats ;
4. La base de données retourne les lignes d'entrées au QueryBuilder ;
5. Le QueryBuilder effectue le mapping entre ces lignes d'entrées en objets métiers (BO) et les envoie au Core ;
6. Le Core s'occupe des éventuels traitements sur les objets reçus et les lègue finalement au contrôleur.

4.1.4. Routing

Nous avons mentionné dans le paragraphe précédent que la requête venant de l'utilisateur appelle une action d'un contrôleur, qui lui, est contenu dans le bundle demandé pour renvoyer une vue. Mais la problématique est comment traduire une requête utilisateur (URL) en une action de contrôleur.

Conçu à cet effet, le routing est le mécanisme qui permet de traduire une URL demandée par l'utilisateur en une action d'un contrôleur.

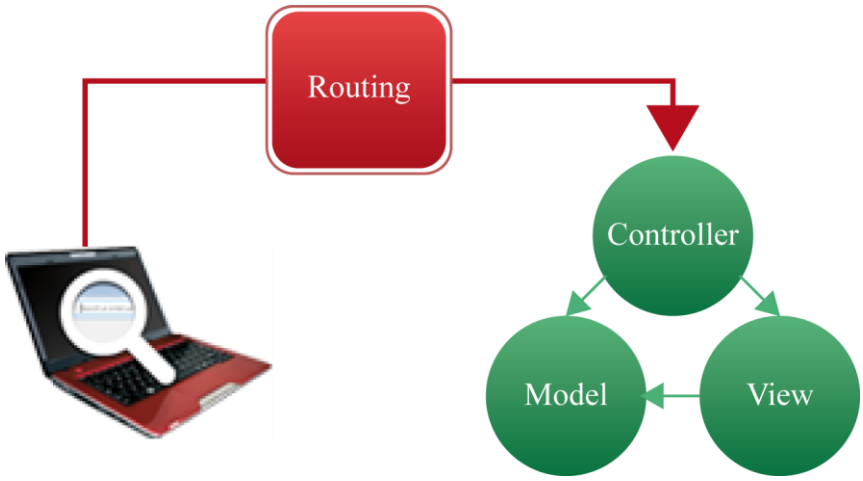


Figure 4.02 : Mécanisme du routing

Nous verrons de plus amples détails comme la structure de fichiers de routing et leur création dans un paragraphe spécialement dédié dans la suite de l'ouvrage.

4.2.Organisation des répertoires

Après de nombreux essais de configurations, on a établi une architecture de dossiers pour organiser les fichiers du framework. Cette organisation devra être suivie par tous les projets qui découleront du framework. Ceci dans le but de séparer les fichiers selon leurs fonctionnalités et aussi de facilement s'y retrouver lors du développement des programmes. Cette norme permettra donc de faciliter le partage de tâches entre plusieurs développeurs, qui est un des objectifs d'un tel framework.

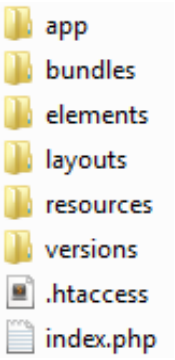


Figure 4.03 : Répertoires du framework

4.2.1. Répertoire « app »

Ce répertoire concerne le cœur du framework proprement dit. Les fichiers qui s'y trouvent permettent la liaison entre les fichiers que l'on va créer pour nos fonctionnalités, les classes mères

que l'on va hériter pour nos propres fichiers, les configurations, le routage, les variables globales, etc.

Néanmoins, il existe dans ce dossier des fichiers que les développeurs peuvent et/ou doivent éditer. Ces fichiers sont entre autres :

- bundles.php : Ce fichier contient un tableau PHP nommé $GLOBALS["bundles"]. Il doit contenir tous les noms des dossiers de bundles qui sont utilisés pour un projet. En d'autre termes, c'est dedans que l'on enregistre tous nos bundles qui vont composer le site. Dans notre cas, on a les bundles :
    - « default » : Pour les pages d'accueil ;
    - « product » : Pour les produits ;
    - « user » : Pour la gestion d'utilisateurs ;
    - « stand » : Pour la gestion des boutiques ;
    - « newsletter » : Pour la gestion des newsletter.
- functions.php : Fichier où l'on peut ajouter des fonctions PHP ;
- hooks.php : Contient la liste des hooks (expliqué plus tard dans un paragraphe spécialement dédié) ;
- vars.php : Contient des initialisations de variables globales. En effet, il est parfois pratique de créer des variables globales auxquelles on veut accéder partout dans le code.

4.2.2. Répertoire « bundles »

Ce dossier sera le plus utilisé pour mettre nos codes sources. On organise le code en ce qu'on appelle « bundle ». Chaque fonctionnalité sera alors écrite dans un dossier du nom (de notre choix) mais toujours dans le dossier « bundle ».

Par exemple, on pourrait avoir un bundle nommé « produits » contenant le code source concernant les rubriques sur les produits en stock (ajout, édition, suppression, listage et affichage, ajout au panier, etc.). Ou aussi un bundle nommé « utilisateurs » si l'on veut mettre un mécanisme d'authentification sur le site web (gestion de la session, listage, blocage, profil, etc.).

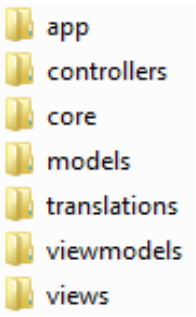


Figure 4.04 : Répertoires d'un bundle

Pour y voir plus clair, voici le contenu d'un dossier de fonctionnalité contenu dans le dossier « bundle » :

- Dossier « app » : Contient le fichier de configuration de routing du bundle ;
- Dossier « controllers » : Contient les fichiers de contrôleurs, soit le « C » du MVC ;
- Dossier « core » : Contient les fichiers relatifs au BLL du bundle ;
- Dossier « models » : Contient les fichiers de modèles, soit le « M » du MVC ;
- Dossier « translations » : Contient les fichiers de traductions si l'on veut utiliser un site multi-langues ;
- Dossier « viewmodels » : Contient les fichiers pour les classes de modèles de vue ;
- Dossier « views » : Contient les fichiers de vues, soit le « V » du MVC.

Ainsi, nous pouvons en déduire que le contenu d'un bundle est organisé de façon à remplir une fonctionnalité, et c'est dans ces dossiers que l'on placera notre code source.

On rappelle toujours que quand on crée un nouveau bundle, il faudra ajouter son nom dans le tableau $GLOBALS["bundles"] du fichier « App/bundles.php ».

4.2.3. Répertoire « layouts »

Ce répertoire quant à lui contient les fichiers de mise en page général du site. C'est le design global qui contient les parties répétitives comme le header, la barre de navigation, le footer, etc. C'est dans un fichier de layout que l'on va implémenter des « blocs » où viendront se greffer les vues de nos bundles.

L'avantage est que les développeurs et les designers pourront travailler séparément puisqu'on peut créer des layouts sans connaissance en PHP.

4.2.4. Répertoire « resources »

Enfin, c'est ici que l'on placera les fichiers liés à la présentation globale. Images, fichiers CSS, fichiers JavaScript, polices de caractères, etc.

4.2.5. Fichiers à la racine

A la racine du projet, nous avons deux (2) fichiers :

- « .htacess » : C'est un fichier .htaccess clasique permettant la sécurisation d'un dossier sur un serveur web ;
- « .htpasswd » : Fichier contenant les couples logins/mots de passe de restriction d'accès ;
- « index.php » : Dans un projet PHP basique, c'est un fichier de page web mais ici, il ne fait aucun affichage. Il s'agit du « contrôleur frontal ». C'est le point d'entrée de l'application qui se charge de faire les différents appels et liaisons entre les fichiers du framework quand l'utilisateur demande une page via un URL.

4.3.Routeurs et routes

4.3.1. Rôle des routes et du routeur

Le mécanisme de routing met en scène deux (2) intervenants :

- Le routeur ;
- Les routes.

Pour faire simple, les routes contiennent la configuration pour une URL précise, une configuration qui se résume à *« quelle méthode de quel contrôleur de quel bundle appeler pour une forme d'URL précise ? »*. Le routeur quant à lui contient juste les routes.

4.3.2. Création d'une route

Nous savons désormais que pour un bundle, c'est dans le fichier « routing.php » que sont définies les routes.

Chaque route et routeur sont en fait tous deux des tableaux en PHP :

- Un routeur est un tableau contenant des routes ;
- Une route est une ligne du tableau à 4 éléments correspondant à la configuration d'une route.

Voici la syntaxe générale d'une route :

```
array(
        "<nom_de_la_route>",
        "<URL_pattern>",
        "<nom_du_bundle>:<Nom_du_contrôleur>:<nom_de_la_méthode>",
        [ "rôles_requis" ]
);
```

- **1ere ligne :** Nom de la route qui doit être unique ;
- **2ème ligne :** Pattern URL. Il correspond à une forme d'URL saisie par l'utilisateur ;
- **3ème ligne :** Cette ligne est importante car elle précise le nom de la méthode et le contrôleur et bundle à appeler ;
- **4ème ligne :** Restrictions d'accès car il se peut que l'on permette l'accès à la page à seulement une catégorie de personnes.

Enfin, il est à noter que le framework nous permet d'insérer des paramètres dynamiques dans la route sous la forme {<*nom_de_la_variable*>}, paramètre qui sera ensuite récupéré en paramètre de l'action du contrôleur. Ces paramètres sont dynamiques du fait que leurs valeurs peuvent changer et l'utilisateur pourra définir leur contenu. Par exemple, on peut avoir une URL « produit/categorie/{id} » pour voir la liste des produits d'une certaine catégorie à partir de l'identifiant de la catégorie dans la base de données.

4.3.3. Appel d'une route

On sait désormais que chaque bundle a son propre routeur dans leurs fichiers « routing.php ». Aussi, chaque bundle est enregistré dans « app/bundles.php ». Le framework gère alors toutes les routes que l'on crée. Ainsi, quand une requête arrive - et quand une forme d'URL correspond à un pattern d'URL dans le routeur - le framework appellera automatiquement la méthode de contrôleur précisé. Tout cela grâce aux configurations de routes et à l'automatisme du framework. Ce dernier va parcourir le liste de routes jusqu'à trouver une correspondance avec un pattern URL ; une erreur sera affichée si aucune route ne correspond à la requête de l'utilisateur.

4.3.4. Avantages

L'utilisation des routes présente plusieurs avantages dont les suivantes :

- La génération d'un lien se fait via le nom de route. Si la forme d'un lien sera changée plus tard, il ne sera plus nécessaire de modifier une à une les valeurs de tous attributs « href », il suffira de changer le pattern URL dans la définition de route ;
- Grâce à la variable $GLOBALS["context"] contenue dans le fichier « vars.php », on ne va plus modifier un par un les chemins des liens si jamais on utilise un contexte (préfixe d'URL) pour le site ;
- Comme on a un paramètre qui permet de restreindre une page aux utilisateurs disposant d'un rôle précis, les restrictions d'accès sont mieux gérées ;
- Les intrusions dans les dossiers du site ne seront plus permises. Avant, il a fallu mettre dans chaque dossier un fichier « index.php » sinon la liste des fichiers qui s'y trouvent seront visibles et accessibles. Maintenant, si un utilisateur malveillant tente d'accéder aux dossiers, il y aura une « erreur 404 » car la route n'a pas été définie pour l'URL qu'il a saisie ;
- La réécriture des URLs est un grand facteur qui permet le référencement d'une page web.

4.4.Vue

4.4.1. Définition

Les vues sont conçues afin de séparer les traitements de données de l'affichage. Une vue ne traitera donc que la présentation de l'information destinée à l'utilisateur. Dans une vue, le code PHP est alors réduit, plutôt que d'avoir une longue page classique où le code se mélange. Néanmoins, on aura toujours besoin d'un peu de code dynamique pour juste afficher les variables et faire des boucles si l'on a des tableaux de valeurs à parcourir.

4.4.2. Mises en page (layouts) et vues (views)

Pour le rendu d'une page, nous avons deux (2) éléments :

- Un layout ;
- Une vue.

Les layouts sont des fichiers de mise en page général du site. C'est le design global qui contient les parties répétitives comme le header, la barre de navigation, le footer, etc. En outre, c'est dans un fichier de layout que l'on va implémenter des « blocs » où viendront se greffer les vues de nos

bundles. En d'autres termes, le layout est aussi une vue à la différence qu'il est destiné à contenir d'autres vues.

Un fichier de layout est stocké dans le dossier « layouts ». Tandis qu'un fichier de vue est stocké dans le dossier « views » du bundle.

Les schémas suivants montrent un layout sur lequel est placé un bloc nommé « content » (en vert). Sur chacune des deux (2) images figure une vue différente :

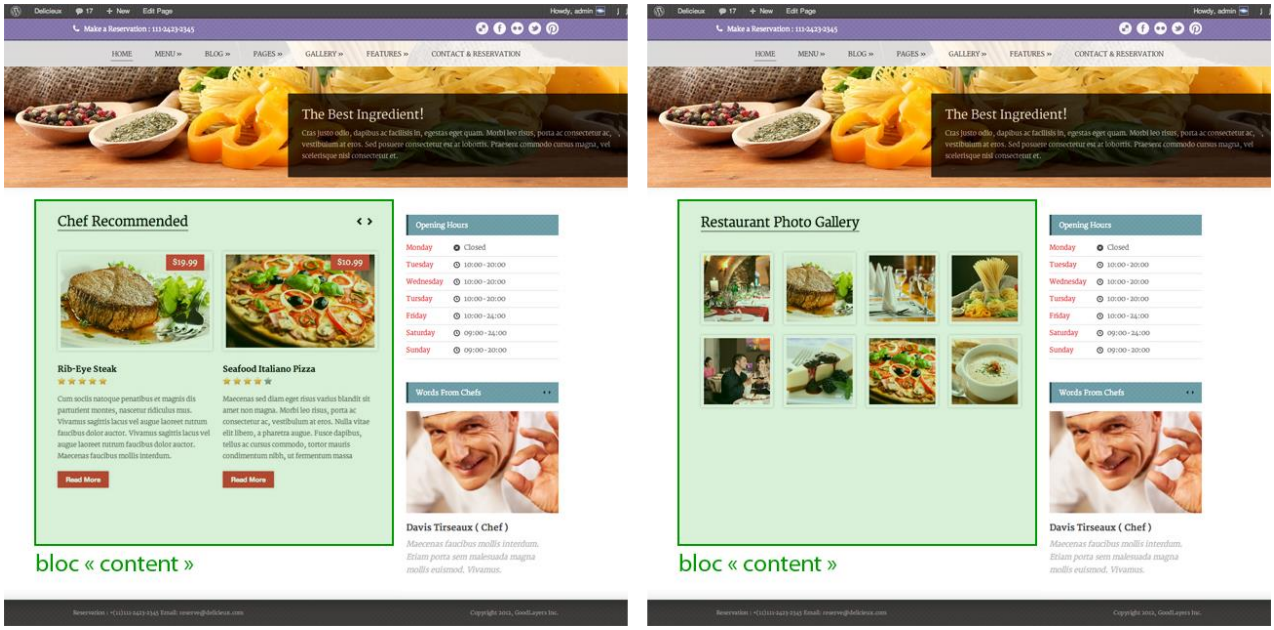


Figure 4.05 : Layout avec deux (2) vues implémentées dans le bloc nommé « content »

4.4.3. Blocs

Il a été dit qu'un bloc sert à accueillir une vue sur un layout.

Pour créer un bloc sur une page de layout, il suffit de mettre le code PHP dont la syntaxe est la suivante :

```
place("<nom_du_bloc>");
```

4.4.4. Inclusion de vue

Dans la plupart des cas, on a des blocs qui se répètent sur plusieurs pages du site internet. Cependant, on doit minimiser le plus possible des « copier-coller ». C'est justement pour cela qu'on a l'inclusion de vue.

Une remarque importante, à l'intérieur d'une vue incluse, on peut utiliser les variables renvoyées par le contrôleur.

La fonction à utiliser est includeView(), pour faire différence avec include() qui est propre à PHP :

```
includeView(
        "<nom_du_bundle>":"<nom_du_dossier>":"<nom_vue.php>",
        <tableau_variables_de_vue>
);
```

Dans la fonction includeView(), on spécifie :

- L'emplacement du fichier de vue ;
- Le tableau contenant les variables à envoyer à la vue.

4.4.5. Quelques fonctions liées à une vue

- asset("*<chemin>*") : Génération de chemin absolu vers un fichier contenu dans le dossier « resources » ;
- path("*<nom_route>*") : Génération d'une URL à partir du nom d'une route associé ;
- place("*<nom_bloc>")* : Indique le nom d'un bloc sur lequel va se greffer une vue renvoyée par le contrôleur ;
- render("*<nom_bundle>*:*<nom_contrôleur>:<nom_action>")* : Inclusion de contrôleur ;
- trans("*<texte_source>")* : Fonction de translation, utile pour les traductions ;
- displayFlashBag() : Permet d'afficher les flashbags ;
- paginate(*<nombre>)* : Permet d'afficher des liens de pagination en fonction du nombre total d'éléments affichables. Le nombre de tronçons à afficher est défini dans « vars.php » sous la variable $GLOBALS["paginationSlice"].

4.5.Modèle

4.5.1. Définition

C'est une classe qui va contenir les attributs avec leurs getters et setters.

Trois (3) éléments entrent en jeu lors de la manipulation d'un modèle :

- La classe de modèle ;
- Le gestionnaire de modèle ou ModelManager ;
- Le générateur de requêtes ou le QueryBuilder.

4.5.2. Structure du modèle

La classe de modèle de base est située dans le dossier « app/core ».

Model
check() generate() getId() setId($id) getQueryBuilder() findBy($criteriasArray) findOneBy($criteriasArray)

Durant l'écriture de la classe, la génération des getters et setters est une longue opération longue. Pour pallier à cela, on a la fonction generate(). Comme en le voit sur la figure ci-après, il suffit juste d'écrire la classe avec les données membres et le mapping dans le constructeur (à gauche), puis, en appelant la fonction generate(), les getters et setters sont affichés sur la page (à droite), il suffit de copier le code dans notre classe.

Figure 4.06 : Résultat de la fonction generate()

4.5.3. Création d'un modèle

On crée un modèle via une classe PHP héritant de la classe mère « Model ».

Il n'est pas nécessaire de préciser le champ « id » puisque le framework l'inclut par défaut pour chaque modèle créé.

4.5.4. Mapping

Dans le constructeur de la classe de modèle, il faut préciser la relation entre les attributs de classe elle-même avec les champs de la base de données. Entre autres, on doit définir :

- **Le nom de la table correspondant à la classe :** En effet, on peut avoir un nom de table différent du nom de la classe de modèle associé ;
- **Les references :** En matière de classe, ce sont les attributs qui font référence à un autre objet. En matière de base de données relationnelles, ce sont les attributs qui font référence à une autre ligne d'une autre table ;
- **Les referencers :** C'est en quelque sorte l'inverse des références. En matière de classe, ce sont les attributs qui sont des références à un autre objet. En matière de base de données relationnelles, ce ne sont pas des attributs de table, mais ce sont les attributs d'une autre table qui font référence à une ligne de la table courante ;
- **Les champs :** En effet, il faut lier les attributs de la classe à leurs valeurs dans la base de données.

Après avoir fait le mapping, on n'est jamais sûr que toutes les correspondances aient été faites. De ce fait, pour vérifier si des champs ont été oubliés lors du mappage, on a la fonction check() qui est appliqué au modèle.

4.5.5. QueryBuilder

4.5.5.1.Définition

Le QueryBuilder est une classe PHP héritant de la classe QueryBuilder. Son rôle est de centraliser toutes les actions qui concernent la manipulation des modèles. C'est uniquement ici que l'on fait du CRUD (Create, Read, Update, Delete) à l'aide de différentes méthodes de la classe QueryBuilder.

4.5.5.2.Utilité

le QueryBuilder ne s'occupe que de l'accès aux données afin de gérer le contenu dynamique d'une application de type CRUD. Elle ne fait qu'interagir avec la base de données pour sélectionner, insérer, mettre à jour et supprimer des entrées.

Le but de l'utilisation d'une classe de QueryBuilder est d'avoir une classe d'accès aux données pour chaque objet métier. En effet, il existe un QueryBuilder associé à chaque modèle, dont le

nom est celui du modèle préfixé de « QueryBuilder ». Par exemple, pour un modèle nommé « Personne », on a le QueryBuilder associé nommé « PersonneQueryBuilder ». Ce dernier va contenir la construction de la requête et éventuellement la récupération des résultats retournés.
Les classes de QueryBuilder se trouvent dans le dossier « models » du bundle.

4.5.5.3.Structure du QueryBuilder

Ici, les classes héritent de la classe mère QueryBuilder. Pour communiquer avec la base de données, on ne fait aucune requête SQL classique mais on utilise des méthodes équivalentes aux langages de requêtes SQL. Par exemple, un SELECT standard est équivalent à la méthode select(). Un autre avantage est qu'il est possible de construire la requête en plusieurs fois. Cela se fait en créant des méthodes différentes.
La construction d'une requête se fait par étape :

1. On écrit d'abord les clauses de la requête grâces à des méthodes comme select(), where(), limit(), orderBy(), etc. ;
2. On construit ensuite la requête pour générer la requête SQL résultant des clauses écrites. Cela se fait avec la méthode build() ;
3. Enfin, s'il s'agit d'une récupération dans la base de données, on récupérer le(s) résultat(s) grâce à des méthodes comme getResults() et getOneResult.

QueryBuilder

`select($fieldsList)`
`join($inverse, $proprietor, $direction)`
`referencer($inverse, $proprietor, $whereList)`
`where($criteriasList)`
`addWhere($criteriasList)`
`limit($offset, $slice)`
`orderBy($field, $mode="ASC")`
`build($booleanTest=false)`
`getOneResult()`
`getResults()`
`getOneArrayResult()`
`getArrayResults()`
`findOneBy($criteriasList)`

4.5.6. Model manager

Le Model manager est la classe qui va nous permettre de gérer la persistance des objets que l'on manipule. Pour l'utiliser, il suffit d'instancier la classe ModelManager et d'appeler les méthodes qui y sont définies :

- delete(<*objet*>) : Supprime l'objet c'est-à-dire supprime l'entrée correpondante en base de données ;
- persist(<*objet*>) : Enregistre la ligne correspondante en base de données, ou effectue la mise à jour si l'objet existe déjà en base de données.

Model
`delete($model)` `persist($model)`

4.5.7. View Models

Un bundle possède aussi un dossier nommé « viewmodels » qui est destiné à contenir des modèles de vue. Le contenu d'une telle classe est similaire à celui d'un simple objet métier, sauf qu'il remplit le rôle de modèle de vue c'est-à-dire contenir des propriétés (avec leurs accesseurs et mutateurs) afin d'être directement utilisable par la vue, sans nécessité de faire des traitements.

4.6.Core

4.6.1. Définition et rôle

On sait que chacune des classes de modèle - définies dans la couche métier – contient uniquement des propriétés avec ses accesseurs et mutateurs et ne gère strictement aucun traitement. En outre, le QueryBuilder ne s'occupe que de l'accès à la base de données ; il ne contient que des méthodes d'accès aux données pour sélectionner, modifier ou supprimer des entrées. Quant aux traitements, ils ne sont gérés uniquement dans le Core, ce dernier s'occupant de la couche logique métier du bundle.

4.6.2. Structure du Core

Les classes du Core se trouvent dans le dossier « core » du bundle. Les classes qui s'y trouvent héritent de la classe mère Core. En pratique, les méthodes du Core appellent des méthodes de la couche DAL, cette dernière étant la seule à pouvoir interagir avec la base de données. En outre, elles peuvent aussi éventuellement retourner des views models qui seront directement utilisables par le contrôleur.

Model
getModel($model) getViewModel($model)

4.7.Contrôleur

4.7.1. Rôle du contrôleur

Le contrôleur est celui qui contient toute la logique de l'application. Entre autres, il gère les arguments de route passés en paramètre, appelle le modèle, effectue plus ou moins des traitements voulus, utilise les fonctions et retourne un résultat comme une vue, des variables, du JSON, etc à partir de nos données.

4.7.2. Structure du contrôleur

Un contrôleur est une simple classe PHP qui hérite de la classe « Controller ». Il va contenir une multitude de méthodes. C'est dans chaque méthode qu'il va faire tout son travail comme dit précédemment (appel de modèles, utilisation des fonctions, envoi de vue, etc.).

Controller
executeOnCore() getCore($core) getModel($model) getViewModel($viewModel) render($layout, $arrayViews) place($arrayViews) sendMail($from, $to, $subject, $message) uploadImage($width, $height, $destPath) uploadImages($width, $height, $destPath) uploadFile($destPath, $extensionsList) uploadFiles($destPath, $extensionsList)

4.7.2.1.Traitement des paramètres de route

Dans le paragraphe consacré aux routes, il a été dit qu'on peut envoyer des paramètres dynamiques dans la définition de routes. Ces paramètres sont écrits entre accolades sous la forme « {*<nom_du_paramètre>*} ».

Ces paramètres seront ensuite écrits en arguments de la méthode du contrôleur, avec le même nom de variable que le nom du paramètre de route. On pourra de ce fait récupérer leurs valeurs dans la méthode.

Nous pouvons voir sur la figure suivante, une route avec le paramètre « url » qui sera récupéré dans la méthode associée avec une variable nommée « $url ».

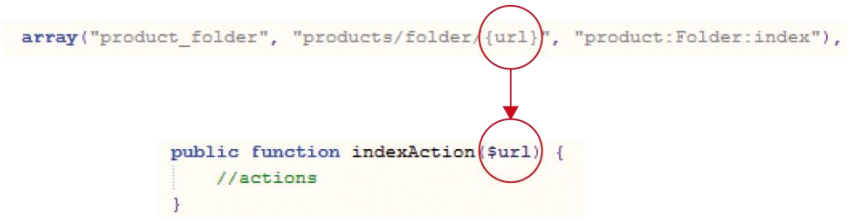


Figure 4.07 : Route et méthode associée

4.7.2.2.Appel d'une classe Core

getCore("*<nom_du_bundle>*:*<Nom_du_modèle>*")

Cette méthode est conçue pour pouvoir accéder à un Core contenu dans n'importe quel bundle. En effet, en plus du nom de la classe, le nom du bundle est précisé.

4.7.2.3.Retour de vue

Pour renvoyer une vue, on utilise la méthode render() dont la syntaxe est la suivante :

```
$this->render(
        "<layout_php>",
        array(
                array(
                        "<nom_bundle>:<nom_contrôleur>:<vue_php>:<nom_bloc>",
                        <tableau_variables_de_vue>
                )
        )
);
```

Nous pouvons voir que dans la fonction render(), on spécifie :

- Le layout ;
- Le fichier de vue avec le nom du bloc d'emplacement ;
- Le tableau contenant les variables à envoyer à la vue.

A l'image de cette fonction render(), on a aussi une fonction place() qui consiste à envoyer une vue partielle donc qui n'a pas de layout. A titre de rappel, une vue partielle est une portion de vue répétitive à plusieurs pages, c'est pour cela que l'on inclut cette vue partielle au layout général.

4.7.3. Inclusion de contrôleur dans une vue

Il y a des cas où on veut inclure un bloc avec du contenu dynamique (ex : liste des derniers articles, publicité aléatoire, etc.). La problématique est que le code est contenu dans une méthode de contrôleur, qui à son tour renvoie la vue.

Grâce à une fonction nommée render() - mais cette fois-ci à écrire dans une vue – la vue correspondante va s'inclure dans le layout.

La syntaxe de la fonction étant :

```
render("<nom_du_bundle>:<Nom_du_contrôleur>:<nom_de_la_méthode>");
```

4.7.4. Quelques méthodes du contrôleur

- getCurrentUser() : Retourne l'utilisateur courant (une instance de la classe User) s'il y en a ;
- redirect("*<nom_de_la_route>*", *<tableau_de_paramètres_de_route>*) : Redirection vers une autre action ;
- sendMail("*<destinateur>*", "*<destinataires>*", "*<sujet>*", "*<message>*") : Envoi de mail ;
- uploadImage("*<largeur>*", "*<hauteur>*", "*<chemin_dossier>*") : Envoi d'une image sur le serveur à partir d'un formulaire sur le site, et enregistrement de la miniature associée ;
- uploadFile("*<chemin_dossier>*", "*<liste_extensions_permises>*") : Envoi d'un fichier sur le serveur à partir d'un formulaire sur le site ;

4.8.Hooks (crochets)

4.8.1. Définition

Un élément important du framework est le « hook ». Analogue à l'inclusion de vue dans un bloc de layout, un hook est un point d'ancrage que l'on place n'importe où dans le code pour y inclure du code, du texte, une fonction, etc. Après avoir déclaré ce point d'ancrage, on pourra plus tard « déclencher » le hook en définissant une fonction qui sera exécutée pour cet évènement. Ainsi, chaque déclenchement d'un hook est associé à une fonction qui sera exécutée.

Un exemple de l'utilisation du hook est pour le titre de la page en haut du navigateur.

Ce titre doit changer à chaque page (donc à chaque méthode) pour informer l'utilisateur. Pour ce faire :

1. On « déclare » un hook dans la balise <title> du layout pour indiquer que l'on va y placer plus tard un titre ;

2. Ajouter le hook dans la pile de hooks et créer la fonction à déclencher ;
3. Ensuite, dans chaque méthode, on « déclenche » le hook pour y écrire un texte qui remplacera le titre de la page.

4.8.2. Déclaration du hook

La déclaration du hook se fait via la fonction hook*("*<*nom_du_hook*>"). L'unique argument de cette fonction est le nom du hook qui est arbitrairement choisi par le développeur. Ainsi, quand on va déclencher le hook, ce nom va servir pour désigner le hook à déclencher.
Dans notre exemple du titre de page, cette fonction est à mettre à l'intérieur de la balise <title> avec un nom comme « hook_title ».

4.8.3. Ajout du hook dans la pile de hooks

Après avoir déclaré un hook à l'endroit voulu de notre code source, la prochaine étape consiste à ajouter le hook correspondant à la pile de hooks. Cette pile sert à stocker tous les hooks définis par le développeur.
addHook("<*nom_du_hook*>", "<*nom_fonction*>", "<*tableaux de variables*>")
Les arguments de cette fonction sont :

- <*nom_du_hook*> : Désigne le nom du hook à déclencher ;
- <*nom_fonction*> : Désigne le nom de la fonction qui sera exécutée ;
- <*tableaux de variables*> : Il s'agit d'un tableau variables au cas où l'on voudrait mettre des paramètres en argument de la fonction de hook. Ainsi, une fonction de hook aura la forme suivante : <*nom_fonction*> (<*tableau_de_variables*>) { <*actions*> }

4.8.4. Déclenchement du hook

Le déclenchement du hook consiste à remplacer le contenu d'un hook par du contenu que l'on définit nous-même.
Cela se fait en appelant la fonction :

```
replaceHook("<nom_du_hook>", "<nom_fonction>", "<notre_contenu>");
```

4.9.Gestion des utilisateurs

4.9.1. Bundle « user »

Le bundle « user » est en charge de la gestion des utilisateurs. Il est fourni par défaut par le framework. Si l'on regarde dans le fichier de routing, on peut constater que plusieurs routes y sont générées :

```
$routes = array(
    array("user_login", "login", "user:Authentication:login"),
    array("user_logout", "logout", "user:Authentication:logout"),
    array("user_register", "register", "user:Registration:register"),
    array("user_confirm", "confirm/{param}", "user:Registration:confirm"),
    array("user_settings", "settings", "user:Profile:settings", "LOGGED"),
    array("user_profile", "profile/{id}", "user:Profile:profile", "LOGGED"),
    array("user_admin_index", "admin/users", "user:Admin:index"),
    array("user_admin_user", "admin/user/{username}", "user:Admin:user"),
);
```

Figure 4.08 : Fichier de routing du bundle « user »

Voici entre autres les routes qui y sont répertoriées, avec leurs fonctions :

- « user_login » : Gère la connexion d'un utilisateur à son compte personnel ;
- « user_logout » : Permet à un utilisateur connecté de se déconnecter pour mettre fin à sa session ;
- « user_register » : Gère le formulaire d'inscription de l'utilisateur ;
- « user_confirm » : Gère la confirmation d'inscription d'un utilisateur ;
- « user_settings » : Permet à l'utilisateur de changer ses informations personnelles sur le site ;
- « user_profile » : Permet de consulter le profil d'un utilisateur en connaissant son nom d'utilisateur ;
- « user_admin_index » : Administration de tous les utilisateurs inscrits sur le site ;
- « user_admin_user : Permet aux administrateurs de voir le profil d'un utilisateur et de le gérer.

4.9.2. Modèle User

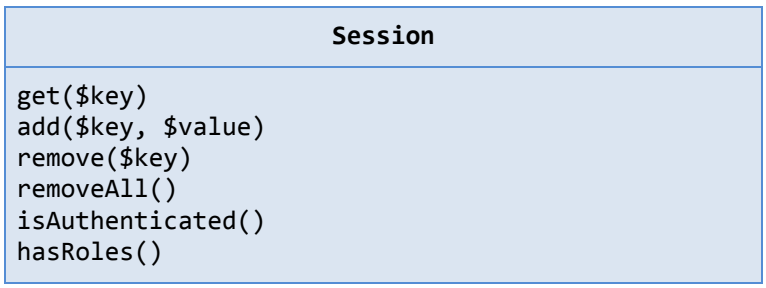
Le modèle User représente tout utilisateur qui s'est inscrit au site. Les propriétés en sont :

- $username : Nom d'utilisateur ;
- $email : Adresse e-mail ;
- $password : Mot de passe haché en md5 ;
- $lastname : Nom de famille ;

- $firstname : Prénom ;
- $sex : Sexe ;
- $image : Nom di fichier pointant vers la photo de profil ;
- $date : Date d'inscription;
- $roles : Chaîne au format JSON contenant la liste des rôles de l'utilisateur ;
- $lastLogin : Timestamp représentant le temps de la dernière connexion au site ;
- $registered : Booléen pour savoir si l'inscription a été confirmée par email (1) ou non (0) ;
- $blocked : Booléen pour savoir si le compte est bloqué (1) ou non (0) ;
- $published : Booléen pour savoir si le compte est activé (1) ou non (0) ;
- $token : Chaîne de caractères auto-générés lors de l'inscription de l'utilisateur. Elle est utile pour la confirmation d'inscription par email de l'utilisateur.

4.9.3. Classe Session

Le framework possède une classe nommée « Session ». Son rôle est, comme son nom l'indique, de gérer la session utilisateur. Pour cela, elle contient différentes méthodes que nous expliciterons ci-dessous :



- get($key) : Récupère la valeur d'une variable de session via sa clé ;
- add($key, $value) : Ajoute une variable de session grâce à sa paire clé/valeur ;
- remove($key) : Supprime une variable de session via sa clé ;
- removeAll() : Supprime toutes les variables de session ;
- isAuthenticated() : Retourne un booléen pour savoir si l'utilisateur en cours s'est authentifié sur le site ;
- hasRoles($roles) : Retourne un booléen pour savoir si l'utilisateur possède le(s) rôle(s) désignés en paramètres. Ce paramètre est une chaîne de caractère avec le nom du rôle. Pour renseigner plusieurs rôles, on les sépare par une virgule.

4.10. Gestion des formulaires

4.10.1. Classe Request

Le paragraphe de la gestion des formulaires est une bonne occasion de présenter la classe Request. Son rôle est de manipuler les données reçues lors de la requête qui a généré la page en cours. Elle ne contient que des méthodes statiques, répertoriées dans le tableau suivant :

Request
isPost() hasOneEmpty($arrayKeys) get($key) getPost($key) bind(Model $model)

- isPost() : Retourne un booléen pour savoir si la page en cours a été appelée via la méthode POST (1) ou non (0) ;
- hasOneEmpty($arrayKeys) : Permet de savoir si un champ de formulaire parmi la liste fournie n'a pas été rempli. L'argument de cette méthode est un tableau de chaîne de caractères représentant les noms des champs ;
- get($key) : Récupère la valeur d'une variable envoyée par la méthode GET via sa clé ;
- getPost($key) : Récupère la valeur d'une variable envoyée par la méthode POST ;
- bind($model) : Gère le binding entre les champs du formulaire et un objet préalablement instancié. Les explications viendront dans le sous-paragraphe suivant lié à la gestion de la soumission du formulaire.

4.10.2. Sécurisation des variables

Toutes les variables transmises par les méthodes GET et POST sont sécurisées quand on accède à leurs valeurs via l'objet Request. On applique par défaut les fonctions htmlspecialchars() et mysql_real_escape_string($value) en vue de la protection contre les failles XSS et les injections SQL. Il est alors inutile de protéger manuellement chaque variable. Cependant, au cas où l'on veut afficher volontairement une variable qui contient du HTML, il existe pour cela la fonction raw() qui va afficher la valeur de la variable en brut.

4.10.3. Gestion de la soumission du formulaire

Plutôt que de traiter les données POST reçues sur une autre action – au risque d'alourdir le nombre de fonctions, la requête est traitée dans la méthode même en testant si la requêtes actuelle est un POST.

En outre, plus d'explications s'imposent sur le binding. En général, les champs d'un formulaire sont faits pour remplir les propriétés d'un objet vide préalablement créé. Cela est géré par le formulaire de la façon suivante : si le nom de champ du formulaire est le même que celui de la propriété de l'objet vide, au binding, la valeur de cette propriété sera automatiquement assignée par celle du champ correspondant. Prenons exemple sur un objet d'une classe Personne (instancié via un objet $personne). Elle pourrait contenir les champs « nom », « prénom » et « sexe ». Du côté du formulaire, on aura des champs avec respectivement les mêmes noms. Quand on fera quelque chose comme Request->bind($personne), les propriétés de l'objet $personne seront automatiquement remplies.

La gestion d'un formulaire se fait en trois (3) étapes :

1. Vérifier si la requête est de type POST, auquel cas on peut traiter l'envoi du formulaire ;
2. Lier notre objet aux variables POST. C'est ici que l'on fait le binding sur le modèle vide afin de remplir automatiquement les propriétés de l'objet vide par les valeurs des champs. Pour les autres variables qui n'ont pas de lien avec l'objet, il faudra les traiter manuellement ;
3. Manipuler ensuite l'objet obtenu (afficher, persister en base de données, etc.).

4.11. Gestion des langues

4.11.1. *Principe*

Il est fortement utile d'avoir plusieurs langues disponibles sur le site. Pour cela, on utilise des dictionnaires de translation, plus précisément un fichier de dictionnaire par langue. On parle de dictionnaire car il se chargera de traduire une chaîne de caractères source d'une langue à une autre, en la comparant avec un ensemble de possibilités en fonction de la langue choisie. Par exemple, toute chaîne de caractères source correspondant à « helloworld » sera traduite en :

- « Bonjour le monde » en français ;
- « Hello world » en anglais ;
- « Hola el mundo » en espagnole ;
- etc.

4.11.2. Structure d'un fichier de dictionnaire

Chaque dossier de bundle contient un dossier nommé « translations » qui va contenir nos fichiers de dictionnaire.

Tout d'abord, parlons du nom de fichier. C'est un simple fichier PHP qui respecte la convention de nom suivante : « dictionary.<*locale*>.php », où <*locale*> représente la locale pour la langue voulue. Evidemment, ce fichier contiendra la liste des chaînes de caractères correpondant à cette langue. Par exemple, on pourra avoir :

- dictionary.fr.php pour les chaînes de caractères en français ;
- dictionary.en.php pour les chaînes de caractères en anglais ;
- dictionary.es.php pour les chaînes de caractères en espagnol ;
- etc.

Ensuite, le contenu du fichier est un simple tableau associatif dont le nom de variable est « $dictionary ». Chaque ligne d'entrée du tableau est une paire clé/valeur où :

- La clé est le texte source ;
- La valeur est le texte traduit dans la langue propre au fichier de dictionnaire.

Il faut aussi savoir que l'on peut aussi inclure des paramètres dynamiques dans le texte traduit. Ces paramètres sont écrits dans la chaîne en les entourant du symbole « % ». C'est ainsi lors de la traduction que l'on assignera des valeurs à ces paramètres.

La figure suivante montre un exemple de contenu du fichier de traduction. En haut le fichier pour la langue française et en bas celui pour la langue anglaise.

```php
//fichier "dictionary.fr.php"
$dictionary = array(
    "page_title" => "Mon site web",
    "page_copyright" => "Tous Droits Réservés %year%",
    "nav_home" => "Accueil",
    "nav_about" => "A propos",
    "nav_contact" => "Contact",
    "user_greeting" => "Bonjour %user%",
    "user_infos" => "Votre nom est %firstname% %lastname%",
);

//fichier "dictionary.en.php"
$dictionary = array(
    "page_title" => "My website",
    "page_copyright" => "All Rights Reserved %year%",
    "nav_home" => "Home",
    "nav_about" => "About",
    "nav_contact" => "Contact",
    "user_greeting" => "Hello %user%",
    "user_infos" => "Your name is %firstname% %lastname%",
);
```

Figure 4.09 : Exemples de contenu du fichier de traduction

4.11.3. Locale

La locale représente le code de langue voulu par l'utilisateur (en, fr, mg, s, etc.). Pour stocker la locale de l'utilisateur, le framework possède une variable globale nommée $GLOBALS["locale"]. Le mécanisme se fait comme suit : à chaque fois qu'une URL possède la variable GET « lang », sa valeur sera prise comme valeur de $GLOBALS["locale"] et stockée dans le cookie $_COOKIE["locale"]. D'autre part, à chaque changement de page, le framework vérifie toujours s'il y a une valeur dans $_COOKIE["locale"] et remplace avec cette valeur celle de $GLOBALS["locale"]. C'est comme si, à chaque changement de page, le framework mémorise la locale choisie par l'utilisateur.

C'est en fonction de la valeur de la locale que le framework va choisir les fichiers de dictionnaire correspondants.

Enfin, dans le fichier « vars.php », on a une variable nommée $GLOBALS["locales"] qui est un tableau répertoriant les locales permises. Ceci pour éviter le cas où l'utilisateur écrit directement dans l'URL une locale qui ne correspond à aucun fichier de dictionnaire.

4.11.4. Traduction

La traduction se fait via la fonction trans("<*texte_source*>", "<*tableau_variables*>"). En utilisant cette fonction, le framework va récupérer la correspondance dans le tableau du fichier de dictionnaire pour afficher le vrai texte en function de la langue choisie. Voici les details sur ces paramètres :

- <*texte_source*> : Correspond au texte source ;
- <*tableau_varables*> : Tableau associatif à paire clé/valeur pour les paramètres dynamiques. La clé étant le nom du parametre qui, dans le texte traduit a été entouré du symbole « % ». La valeur étant la valeur du paramètre.

4.12. Conclusion

Dans ce mémoire, nous avons non seulement développé un site de vente en ligne mais aussi un framework PHP MVC. Ce framework Open Source va être proposé à tout développeur final pour lui servir d'outil de base afin d'éviter de réinventer la roue à chaque projet rencontré. Ce chapitre a présenté les grandes lignes et les composants qui constituent le framework. Il y a notamment le concept de bundles, de routing, de translations, de layouts, de hooks, etc. Pour mettre en pratique le fonctionnement de notre framework, passons directement au dernier chapitre qui va parler du site proprement dit.

CHAPITRE 5
CONCEPTION ET REALISATION DU SITE A PARTIR DE NOTRE FRAMEWORK

5.1.Introduction

5.1.1. Présentation du projet

Le projet de ce mémoire consiste à réaliser un site internet de vente en ligne multi-boutiques avec paiement en ligne. Ce site a été conçu à partir d'un framework PHP que l'on a élaboré, dans le but de définir une structure de code organisée, découpée, lisible, maintenable et réutilisable. Le but de ce projet n'est donc pas non seulement de développer un site internet de vente en ligne, mais aussi d'exploiter et de mettre en exergue les fonctionnalités du framework dont les grandes lignes ont été détaillées dans le chapitre précédent.

En s'inscrivant sur le site, tout utilisateur pourra voir les différents produits qui s'y trouvent. Ces produits sont non seulement classés par catégorie, mais aussi par boutique. En effet, chacun peut demander à ouvrir une boutique sur le site et à y placer ses produits.

On verra dans la suite de l'ouvrage que le site propose plusieurs fonctionnalités aux utilisateurs désirant faire leurs courses en ligne, tout cela avec une navigation aisée et dans un cadre harmonieux.

5.1.2. Analyse des besoins et objectifs

Actuellement à Madagascar, les magasins et les boutiques prolifèrent dans tous les coins de rue de la ville. Seulement, devant cet embarras du choix, il peut être contraignant pour les consommateurs de passer des heures à rechercher tel ou tel produit. Concevoir ce site permettra, d'une part, aux clients de faciliter leurs courses et de leur faire gagner du temps, et d'autre part, aux entreprises commerciales de profiter des bénéfices qu'apporte Internet pour pouvoir vendre leurs produits en ligne et se faire connaître du grand public.

Les produits exposés sur le site doivent être bien organisés pour être aisément accessibles aux visiteurs. Ainsi, chaque produit doit appartenir à une catégorie et un clic sur une catégorie doit montrer les produits correspondant. Mieux, on doit prévoir un formulaire de recherche pour que l'utilisateur puisse accéder à des produits spécifiques au lieu de fouiller chaque lien sur le site. On doit également afficher les informations pertinentes concernant un produit. Entre autres le nom, le prix, le stock disponible, la boutique à laquelle il appartient, la catégorie, les descriptions, etc. L'intégration de boutiques sur le site permet aussi de plus en plus de choix sur la diversité des

produits. Toute personne désirant ouvrir une boutique sur le site pourra faire une demande d'ajout en se rendant sur une page conçue à cet effet. Elle devra fournir certaines informations (nom de la boutique, image, etc.) qui seront transmises à l'administrateur à la soumission du formulaire. Une fois notifié, ce dernier décidera ensuite si la boutique virtuelle pourra être mise en ligne ou non. Si elle est publiée, le responsable de la boutique pourra ensuite y ajouter tous les produits qu'il veut. Pour cela, il est mis à la disposition de tout vendeur un panel d'administration pour gérer et suivre son compte vendeur et les produits qui s'y trouvent. Cependant, l'administrateur peut se donner le droit d'annuler la publication d'une boutique en cas de fraude ou de non-respect du contrat.

Pour pouvoir faire des achats, l'acheteur doit ajouter des produits sur son panier virtuel. La commande pourra être validée après avoir effectué la validation par paiement. Pour cela, le site propose le paiement en drive, par monnaie électronique, ou encore avec Paypal. A chaque validation de commande, les vendeurs doivent être notifiés. On met alors à leur disposition un panel sur le site afin de consulter les produits en cours de livraison, les produits livrés, etc.

Il faudra aussi suivre certaines traces sur l'acheteur lors de ses visites. Ceci dans le but de voir ses habitudes pour avoir un aperçu de ses préférences et ainsi proposer des produits similaires.

Enfin, devant l'arrivée en masse de tous les terminaux à l'image des smartphones et des tablettes, il est impératif de s'adapter à ces types d'appareils pour conquérir le maximum de clients et offrir différentes interfaces de navigation pour chacun d'eux.

5.1.3. Cibles et intervenants

5.1.3.1.Intervenants externes

Les principales cibles du projet sont :

- **Les simples visiteurs :** Le site est ouvert à tout le monde. Chacun peut visiter les produits et les boutiques qui y sont exposés. Il faudra toutefois s'inscrire pour pouvoir effectuer des commandes ;
- **Abonnés :** Ce sont les utilisateurs qui se sont inscrits sur le site internet et qui pourront pleinement profiter des services offerts dans le front office ;
- **Les détenteurs de boutiques :** A part les acheteurs, les vendeurs sont des acteurs principaux sur le site. Ce sont eux qui détiennent les produits à exposer sur le site internet. A chacun d'eux est attribué un compte sur le site pour pouvoir gérer ses produits.

5.1.3.2.Staff interne

Le staff interne du site doit être bien organisé pour mener à bien les transactions entre les clients et les vendeurs car nous savons que toutes les opérations passent par le site. Pour cela nous avons besoin des services minimums suivants :

- **Développeur :** Chargé de maintenir le site pour le maintenir et le faire évoluer ;
- **Opérateurs de commande :** Chargés de faire un suivi des commandes effectuées et à effectuer ;
- **Contrôleurs de stock :** Chargés de vérifier les stocks auprès des boutiques et ainsi voir les boutiques qui ne respectent pas la clause selon laquelle elles doivent faire un suivi de leurs stocks et les mettre à jour sur le site ;
- **Livreurs :** Chargés de récupérer les produits chez le vendeur, de régler les paiements, et de livrer les produits.

5.1.4. Workflow des services

Le site étant multi-boutiques, on y verra une multitude de sociétés. Le règlement des paiements et des livraisons peut alors devenir un grand problème. Parce que ces opérations sont des plus délicates, nous partons du principe qu'aucune relation directe ne doit être permise entre un vendeur et un acheteur lors d'un paiement ou d'une livraison. Toutes les opérations doivent passer par le staff du site. En plus, au niveau du paiement, tous les détenteurs de boutiques ne disposent pas tous d'un compte Paypal, raison de plus pour qu'on fasse office d'intermédiaire avec les acheteurs. Voici comment cela se passe :

1. L'utilisateur paie le montant de la commande au nom du site ;
2. Le vendeur est juste notifié qu'un paiement sur sa boutique a été effectué ;
3. Le site s'occupe de prendre le produit chez le vendeur et de régler le paiement par facture ;
4. Le vendeur est notifié dans son panel d'administration que le paiement pour sa boutique a été effectué ;
5. Le site livre le produit à l'acheteur qui signe la facture de livraison ;
6. Le site notifie l'acheteur et le vendeur que le(s) produit(s) commandé(s) ont été livrés.

5.2.Modélisation de la base de données

5.2.1. *Règles de gestion*

- A 1 utilisateur correspond plusieurs produits de panier et à 1 produit de panier correspond 1 seul utilisateur ;
- A 1 produit correspond plusieurs produits de panier et à 1 produit de panier correspond 1 seul produit ;
- A 1 produit correspond plusieurs descriptions et à 1 description correspond 1 seul produit
- A 1 produit correspond un seul dossier et à 1 dossier de produit correspond plusieurs dossiers ;
- A 1 dossier correspond 1 seule catégorie et à 1 catégorie correspond plusieurs dossiers ;
- A 1 produit correspond plusieurs images et à 1 image correspond un produit ;
- A 1 dossier correspond plusieurs boutiques et à 1 boutique correspond plusieurs dossiers ;
- A 1 boutique correspond plusieurs administrateurs (utilisateurs) et à 1 administrateur (utilisateur) correspond plusieurs boutiques ;
- A 1 produit correspond plusieurs vues et à 1 vue correspond 1 seul produit ;
- A 1 utilisateur correspond plusieurs vues et à 1 vue correspond 1 seul utilisateur ;
- A 1 dossier correspond plusieurs vues et à 1 vue correspond 1 seul dossier ;
- A 1 boutique correspond plusieurs vues et à 1 vue correspond 1 seul boutique ;

5.2.2. *Modèle Conceptuel de Données*

De cette règle de gestion découle le modèle conceptuel de données suivant, formé par plusieurs classes d'objets modèles.

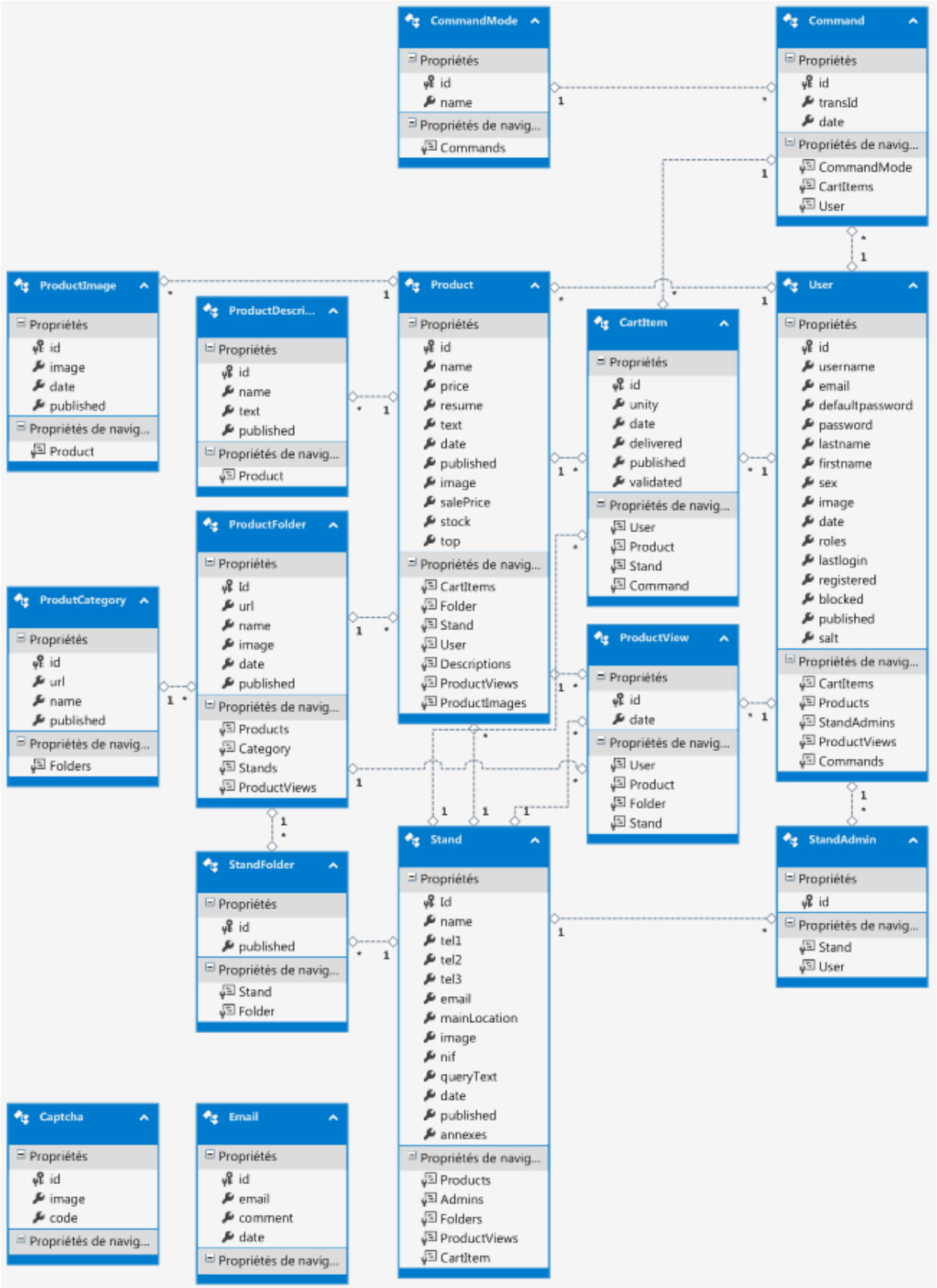


Figure 5.001 : Modélisation de la base de données

Le tableau suivant liste les classes de modèles utilisées dans ce projet :

Classe	Donnée physique représentée
User	Utilisateur s'étant inscrit au site
Product	Produit existant sur le site
Stand	Boutique existant sur le site
ProductDescription	Description d'un produit
ProductFolder	Dossier de produits
ProductCategory	Catégorie de dossiers de produits
ProductView	Vue sur un produit, un dossier ou une boutique
ProductImage	Image liée à un produit
StandFolder	Dossier de produits disponibles sur une boutique
StandAdmin	Utilisateur qui est administrateur d'une boutique
CartItem	Elément de panier d'un utilisateur
CommandMode	Mode de règlement de la commande
Command	Une commande faite par un utilisateur
Email	Adresse email pour la newsletter
Captcha	Codes captchas

Tableau 5.01 : Modèles utilisés dans le projet

5.3.Présentation des interfaces

Figure 5.002 : Design général du front office pour desktop

1. La barre d'utilisateur contenant :
    a. Les liens relatifs au compte personnel (connexion, déconnexion et paramètres) ;
    b. Les liens vers le panier ;
    c. Les liens vers les back offices (le panel vendeur et l'administration) ;
    d. Les liens de langue ;
2. Le header ;
3. La bannière publicitaire ;
4. La barre latérale comprenant :
    a. Le formulaire de recherches ;
    b. Les liens vers l'espace vendeur ;
    c. Les publicités latérales ;
    d. Les produits suggérés.
5. Le bloc « content » sur lequel va s'implanter les différentes vues ;
6. Le footer comprenant :
    a. Le formulaire d'inscription à la newsletter ;
    b. Les informations de contact ;
    c. Des informations relatives au site (réseaux sociaux, paiements acceptés, partenaires, etc.).

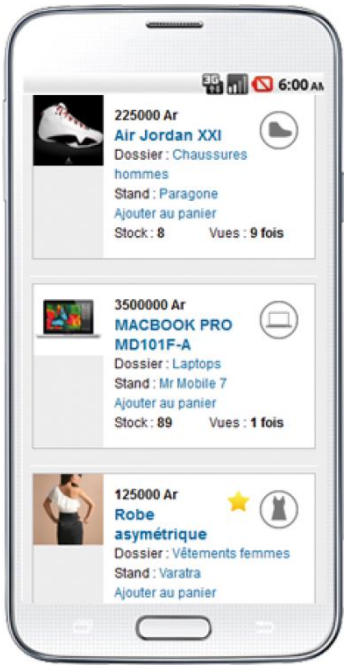


Figure 5.*003* : Design général sur mobile

L'interface d'administration, quant à elle, comporte la barre des menus à gauche et le bloc « content » identique à celui de l'interface du front office :

Figure 5.004 : Design général du back office

5.4.Utilisateurs

5.4.1. Gestion des privilèges

Le site internet est ouvert à tous. Un simple utilisateur peut visiter les différents produits et boutiques qui s'y trouvent. Cependant, il est nécessaire de créer son propre compte et de s'y connecter à l'aide du nom d'utilisateur et du mot de passe choisis pour pouvoir profiter des fonctionnalités réservées aux personnes connectées. Ces fonctionnalités sont entre autres :

- L'espace vendeur pour la demande d'ajout de boutique et l'ajout de produit dans sa boutique ;
- La gestion du panier ;
- La validation de commande par paiement pour livraison.

Un simple utilisateur ne pourra donc pas interagir avec les produits tant qu'il ne s'est pas authentifié sur le site.

5.4.2. Inscription

Pour s'inscrire sur le site, un utilisateur devra remplir un formulaire dans l'URL « /register ». Les données des champs et leurs caractéristiques sont :

- Le nom : Chaîne de caractères alphabétiques ;
- Le prénom : Chaîne de caractères alphabétiques ;
- Le nom d'utilisateur : Chaîne de caractères alphanumériques, sans caractères spéciaux, vingt-cinq (25) lettres au maximum et unique. C'est le pseudonyme qui permettra, avec le mot de passe, de se connecter sur son compte personnel ;
- Le mot de passe : Chaîne de caractères. C'est ce qui permettra, avec le nom d'utilisateur, de se connecter sur son compte personnel. Il est dupliqué sur le formulaire pour confirmer le mot de passe ;
- Le numéro de la Carte d'Identité Nationale ;
- L'adresse de livraison ;
- L'adresse email : Chaîne de caractères au format d'email. Il est unique dans la base de données et va servir à confirmer l'inscription grâce à un lien envoyé à l'utilisateur par email et à lui communiquer diverses informations (confirmation de l'ajout de boutique, blocage du compte, newsletter, etc.) ;
- Le sexe : Choix entre « Masculin » et « Féminin » ;
- Le code captcha : Pour sécuriser du site des robots qui sont capables d'envoyer plusieurs requêtes d'inscription factices ;
- Bouton d'acceptation des termes du contrat. En effet, en bas du formulaire se trouve le contrat d'inscription qui annonce les règles d'utilisation du site et les dispositions qui peuvent être prises par les administrateurs. Il est obligatoire de cocher cette case pour s'inscrire sur le site.

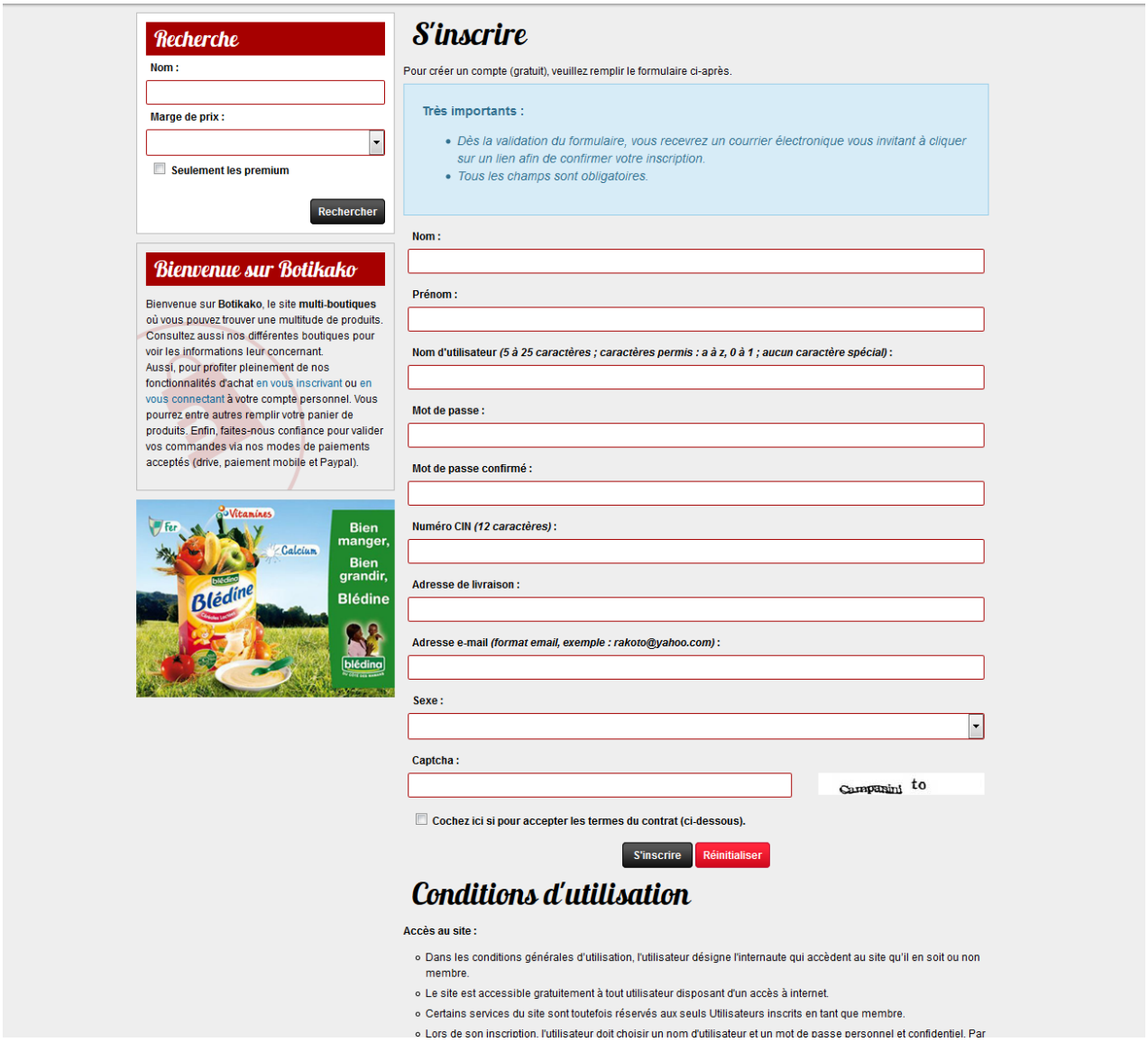


Figure 5.005 : Formulaire d'inscription

Quand l'utilisateur s'est bien inscrit - donc a rempli correctement le formulaire – il va recevoir un email lui disant de cliquer sur un lien pour confirmer l'inscription. Ce lien est de la forme « confirm/{*nom_d_utilisateur*}-{*token*} » où :

- {*nom_d_utilisateur*} est le nom d'utilisateur de la personne ;
- {*token*} est une chaîne dont les caractères sont générés au hasard. En effet, quand un utilisateur s'inscrit, on lui attribue ce token dans la base de données.

5.4.3. Connexion et déconnexion

Un utilisateur qui possède un compte personnel sur le site peut à tout moment se connecter grâce au formulaire de connexion à l'adresse « /login ». C'est là que l'utilisateur saisit son nom d'utilisateur et son mot de passe. Bien sûr, il faut interdire l'accès dans les cas suivants :

- Si au moins un champ est vide ;
- Si les données saisies ne correspondent à aucune entrée dans la base de données ;
- Si le compte n'a pas été validé par e-mail ;
- Si le compte n'a pas été bloqué par l'administrateur.

Au contraire, si tout va bien, il faudra enregistrer certaines données de l'utilisateur en variable de session comme son identifiant. Ceci afin de le maintenir comme étant loggé durant toute la navigation, et aussi pour pouvoir récupérer l'utilisateur courant à tout moment.

Enfin, au terme de sa visite, il pourra cliquer sur le lien de déconnexion pour terminer sa session. Les variables de sessions précédemment créées seront alors détruites et il faudra à nouveau se connecter pour accéder à son compte personnel.

Figure 5.006 : Formulaire de connexion

5.5.Produits

5.5.1. Introduction

Tous les produits qui ont été mis en ligne par les vendeurs peuvent être consultés par tout le monde. Sur la page d'accueil, on liste juste les produits qui ont été ajoutés récemment. Mais sur la barre latérale, on a la liste des produits que le site suggère à l'utilisateur. Aussi, les utilisateurs peuvent voir les produits spécifiques à un dossier (ex : Tables, Laptops, Baskets, T-shirts, etc.).

Enfin, il y a des produits que l'administrateur considère comme « premium ». Une icône d'étoile est affichée sur un produit considéré comme tel.

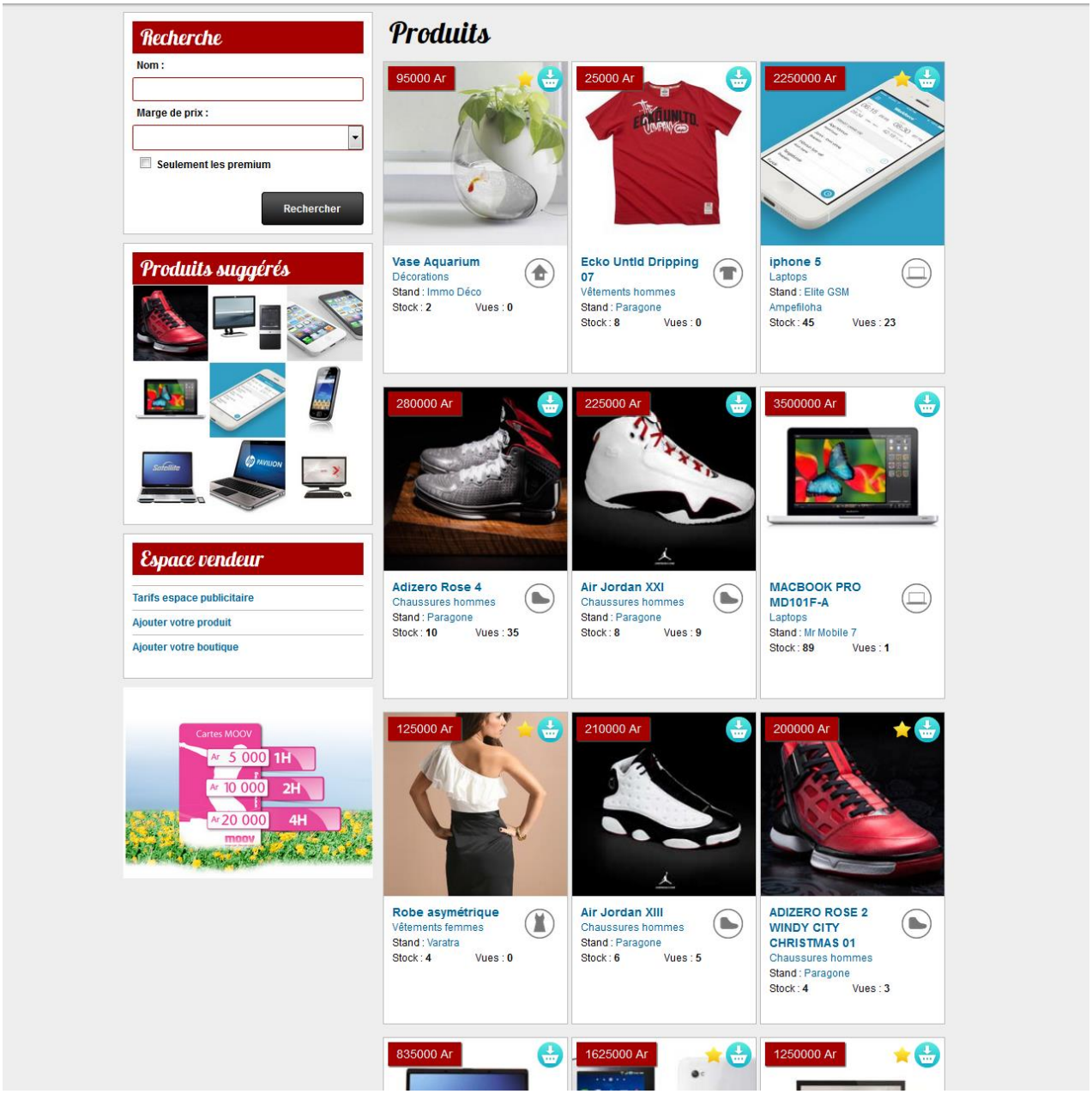


Figure 5.007 : Liste des produits

Pour voir un produit, il suffit de cliquer le lien correspondant. On affiche alors toutes les informations et détails relatifs à ce produit. On a entre autres les images, le texte descriptif, les descriptions et le nombre de vues.

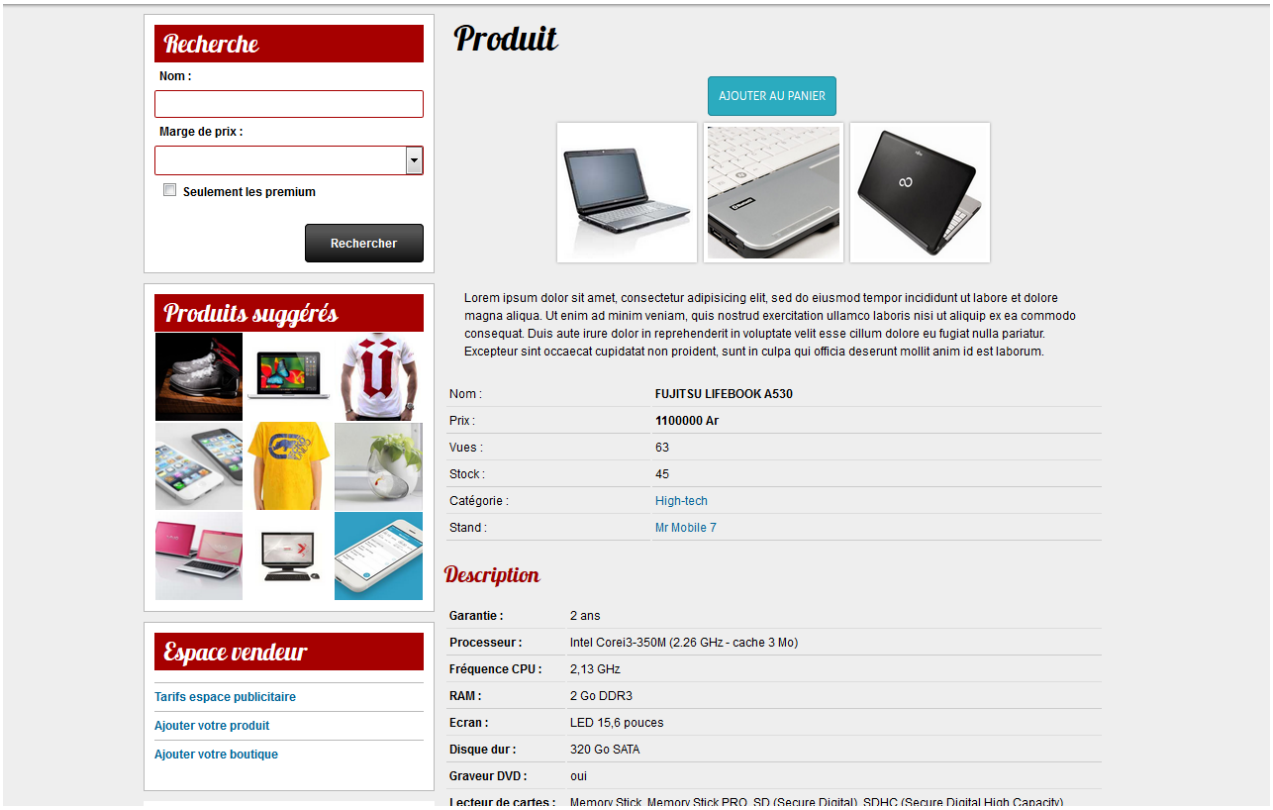


Figure 5.008 : Détails sur un produit

5.5.2. *Recherche de produits*

Devant toute une liste de produits de caractéristiques variées, il est plus ergonomique de pouvoir faire des filtres. Pour cela, une vue partielle destinée à un formulaire de recherche est intégrée à droite du layout. Ainsi, un utilisateur pourra faire les recherches en fonction :

- D'un mot clé pour le nom du produit ;
- D'une marge de prix.

Evidemment, dans les champs de texte, les utilisateurs vont remplir ce qu'ils veulent. Il est alors indispensable de traiter les valeurs reçues pour donner des résultats les plus précis que possible :

- La recherche doit être insensible à la casse ;
- La recherche doit être insensible aux accents ;
- L'ordre des mots ne compte pas. Les espaces doivent être pris en compte. L'ordonnancement des mots fournis par l'utilisateur ne sont pas forcément celui du nom du produit dans la base de données. Par exemple, « Samsung Galaxy » doit être équivalent à « Galaxy Samsung » ;

- Les mots manquants peuvent être omis. La recherche fournie par l'utilisateur doit suffire à correspondre à un produit dont le nom est plus long mais certains termes correspondent. Par exemple, un produit « Apple iPhone 5 16Go » peut correspondre à la recherche « iPhone 5 » ou « 16 Go ». L'utilisateur peut même dire uniquement « Apple » pour avoir tous les produits « Apple ». Néanmoins, on montre plus en haut de liste les produits dont le nom correspond au plus grand nombre de termes de la recherche ;
- La longueur maximale d'un terme de la recherche est de 3 caractères. Sinon, par exemple, une recherche « a » va montrer tous les produits contenant la lettre « a », ce qui peut contredire le terme de filtre ;
- Il n'est pas obligatoire de remplir tous les champs, mais au moins un champ doit être rempli. On peut par exemple juste fournir la marge de prix et aucun nom.

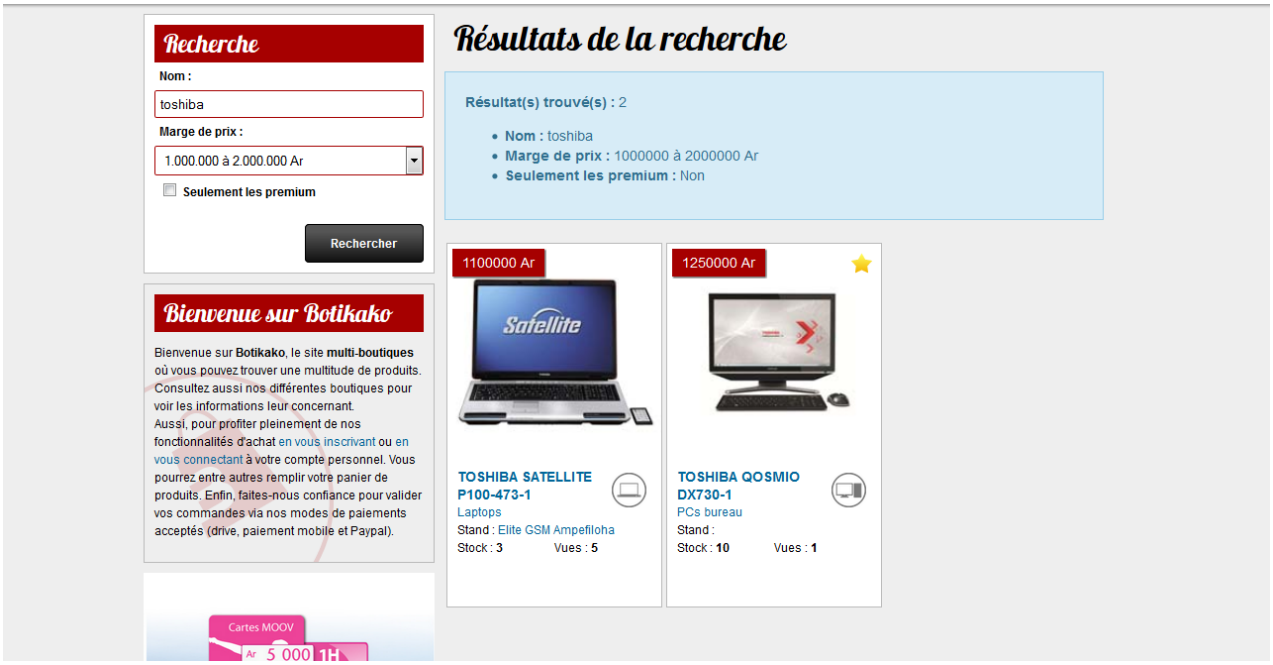


Figure 5.009 : Exemple de recherche sur le site

5.5.3. Ajout de produit

Chaque vendeur peut ajouter un produit sur sa boutique. Cela se fait en cliquant sur le lien « Ajouter votre produit » dans la barre latérale. Il sera ainsi redirigé vers le formulaire d'ajout que nous pouvons voir sur la figure suivante. Nous constatons qu'en haut du formulaire, on affiche à l'utilisateur quelles boutiques il possède sur le site. Au cas où il n'en possède aucune, on le signale et on interdit l'accès au formulaire.

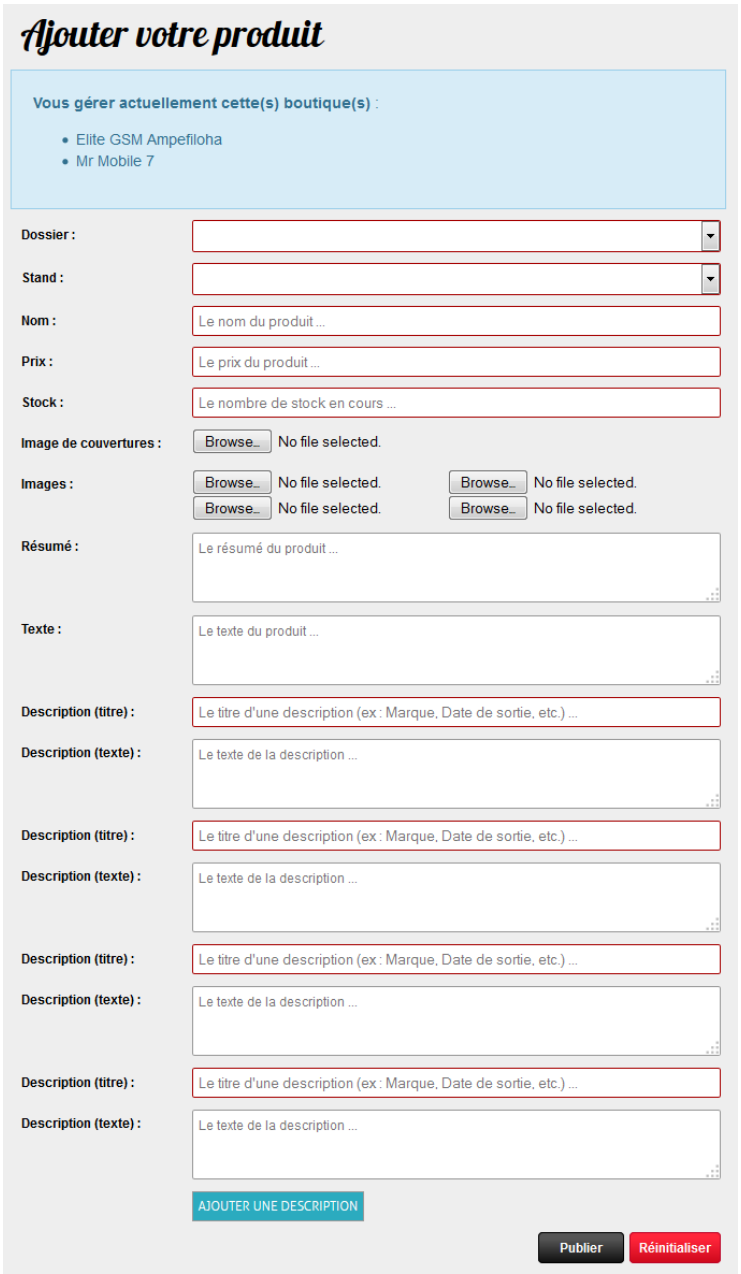


Figure 5.010 : Formulaire d'ajout de produit

Les informations à compléter sont les suivantes :

- Le dossier dans lequel placer le produit ;
- La boutique dans laquelle placer le produit ;
- Le nom du produit ;
- Son prix en Ariary ;
- Le stock disponible ;
- L'image de couverture et les éventuelles autres images ;
- Le texte résumant le produit ;
- Les descriptions. L'ajout de description est illimitée, il suffit juste de cliquer sur le bouton « ajouter une description ».

Au cas où des informations seraient oubliées, le vendeur peut à tout moment accéder à son panel pour modifier voire même supprimer des produits.

5.5.4. Analyse des habitudes de l'utilisateur

Une des fonctionnalités importantes du site est que le programme analyse les préférences d'un utilisateur en fonction du contenu qu'il a l'habitude de consulter. A chaque fois qu'un utilisateur connecté visite un produit, une catégorie ou un stand, les informations sont enregistrées. Les informations essentielles relatives à ces données sont enregistrées en base de données, et ce, de page en page.

Ainsi on connaîtra approximativement les goûts du visiteur et on pourra de ce fait proposer des pubs et des produits similaires pour qu'il puisse naviguer de manière efficace. En outre, quand quelqu'un insère un quelconque produit, ce dernier sera plus mis en avant dans les nouveaux produits.

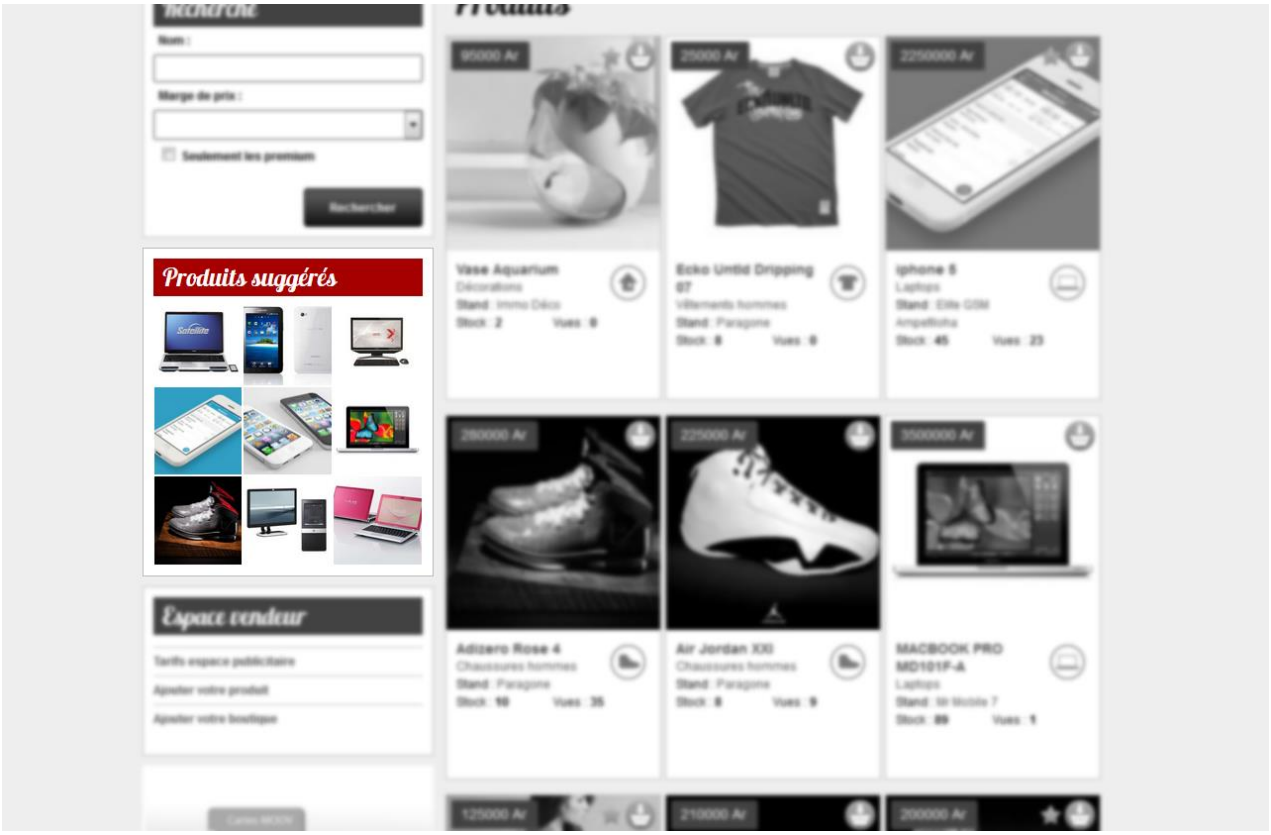


Figure 5.011 : Produits suggérés

Aussi, quand l'administrateur consulte le profil d'un utilisateur, il pourra aussi voir ses vues et ses dossiers les plus visités. Voir la figure ci-après.

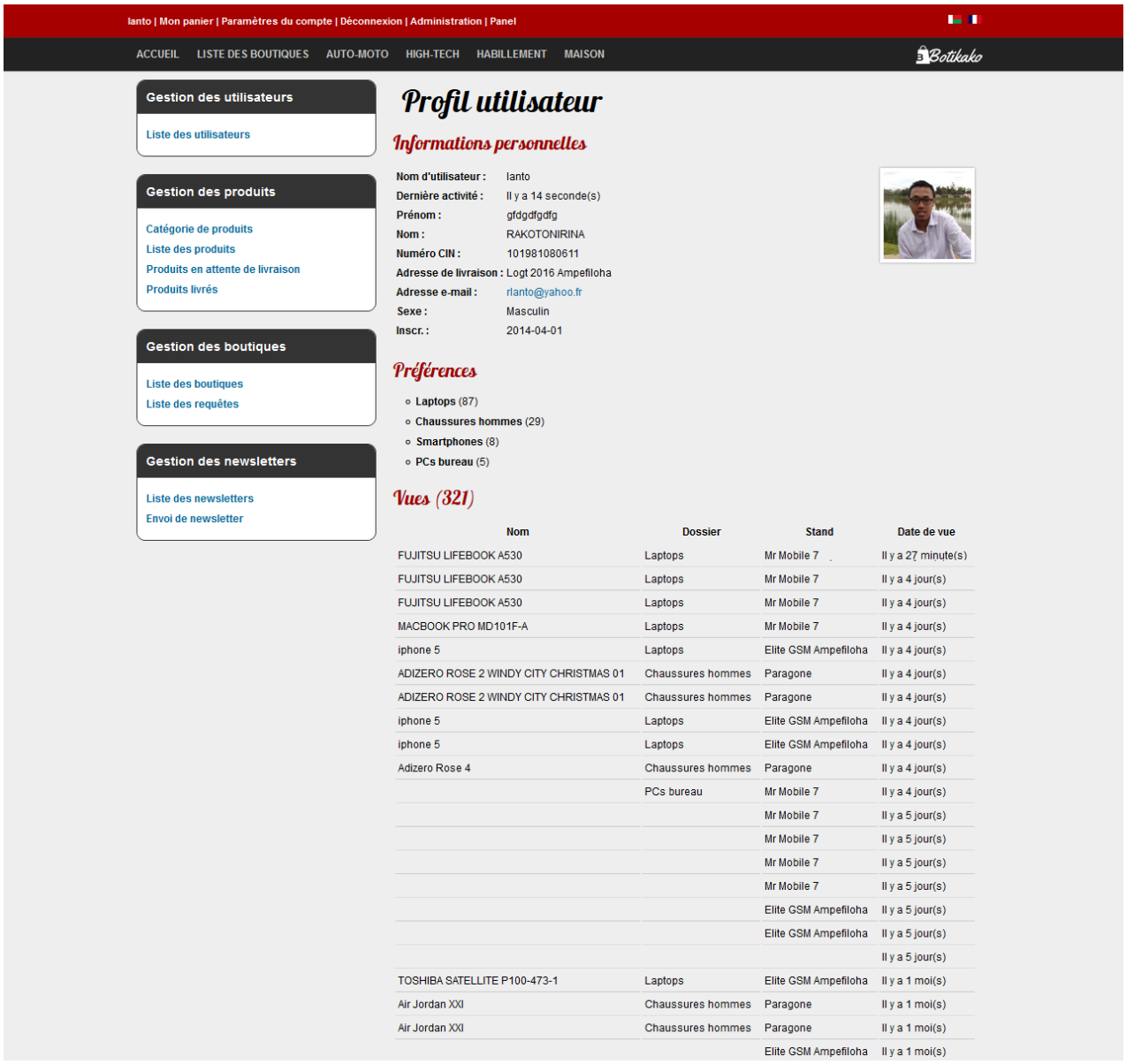


Figure 5.012 : Statistiques sur les habitudes d'un client

5.6.Panier

5.6.1. Paiements disponibles

Avant de voir le contenu du panier d'un acheteur, il est nécessaire de voir les modes de paiement proposés sur le site. Après avoir rempli et paramétré son panier, un utilisateur va devoir valider en payant sa commande pour récupérer les produits. Pour cela, le site met à disposition de l'acheteur trois (3) méthodes :

- **Drive :** Permet à l'utilisateur de réserver la commande sur le site et de payer et récupérer plus tard les produits. Ici, l'utilisateur doit savoir que les produits ne lui seront pas

réservés. Ainsi, s'il y a rupture de stock jusqu'à la date où il récupère ses biens, la commande est annulée et aucune somme ne sera versée ;

- **Monnaie électronique :** Permet à l'utilisateur de payer la somme de la commande par son téléphone mobile, à partir des opérateurs agréés. Le processus ici se déroule en deux (2) étapes : L'acheteur envoie la somme par son téléphone mobile, ensuite il valide sa commande sur le site grâce à un formulaire dédié. Nous verrons plus de détails dans un prochain paragraphe concernant la simulation de ce mode de paiement ;
- **Paypal :** Permet à l'utilisateur de régler la somme instantanément via Paypal.

5.6.2. Panier de commandes

Le panier de commandes liste tous les produits que l'utilisateur désire acheter. Son contenu change à chaque session car chaque utilisateur connecté possède son propre panier.

Comme on peut le voir sur la figure suivante, le panier contient la liste des produits à commander avec :

- Leur image et leur nom ;
- Leur boutique d'origine ;
- Leur prix unitaire ;
- Le stock disponible correspondant ;
- La quantité souhaitée qui est un champ éditable par l'acheteur ;
- Le prix total en fonction de la quantité (produit du prix et de la quantité) ;
- Une case à cocher pour retirer le produit du panier.

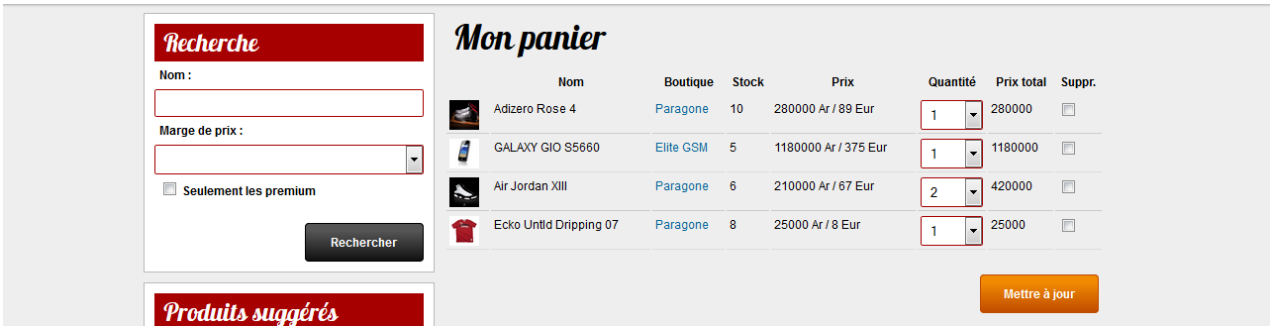


Figure 5.013 : Panier virtuel

Il existe aussi en bas du panier un tableau qui répertorie les produits en attente de livraison et/ou récupération. On parle de produits en cours de « livraison » quand le paiement par monnaie électronique ou par Paypal a été validé. Aussi, dans ce cas, le stock va être modifié car on est sûr. De l'autre côté, on dit que pour les produits sont en cours de « récupération » si l'acheteur a choisi

le drive comme mode de paiement. Dans ce cas, en tant que vendeur, on n'a aucune assurance que l'acheteur va venir récupérer les produits. Alors on ne modifie le stock restant pour le produit qu'au moment du paiement en face à face. Dans le tableau des produits en attente de livraison/récupération, on a, mis à part les mêmes informations que dans le panier :

- Le numéro de transaction ;
- La date de la validation ;
- Le mode de paiement.

Figure 5.014 : Produits en attente de livraison/récupération

5.7.Boutiques

5.7.1. Liste des boutiques

Le lien « Toutes les boutiques » affiche la liste des boutiques disponibles sur le site, avec le nom de la boutique qui est un lien menant vers celle-ci, et le nombre de vues.

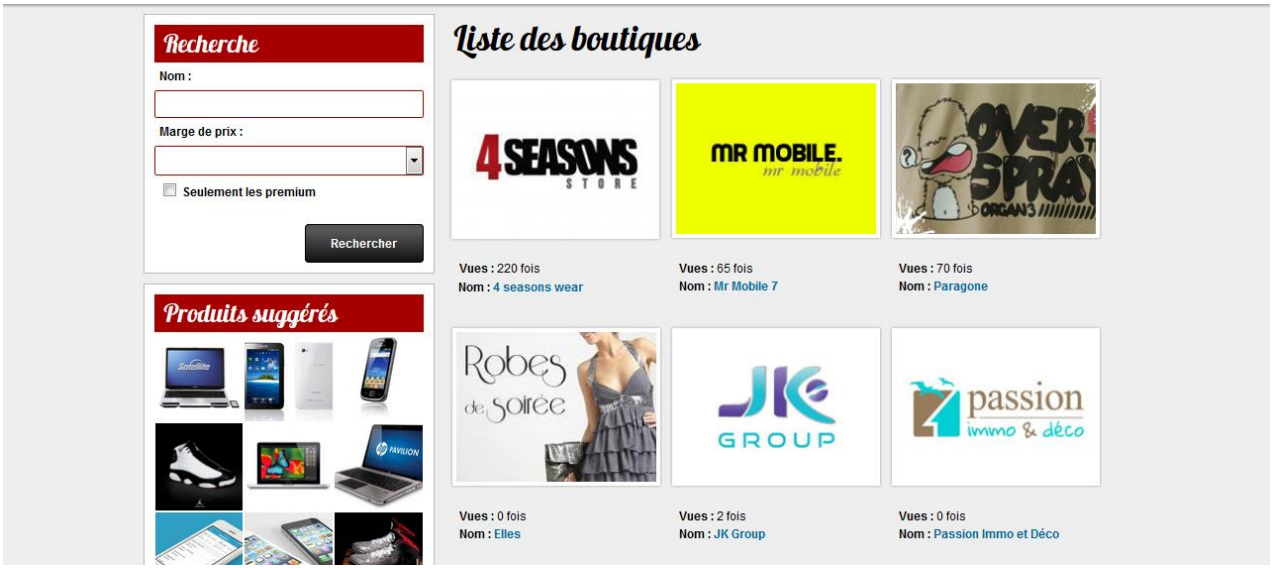


Figure 5.015 : Liste des boutiques

5.7.2. Administration des boutiques

Toutes les boutiques créées sur le site sont listées sur un tableau dans le back office de l'administrateur. Ce dernier pourra ainsi effectuer des actions comme :

- Voir les informations relatives à une boutique en cliquant sur le lien correspondant ;
- Annuler la publication d'une boutique ;
- Envoyer un message.

En consultant une boutique, il pourra voir qui en est l'administrateur, etc.

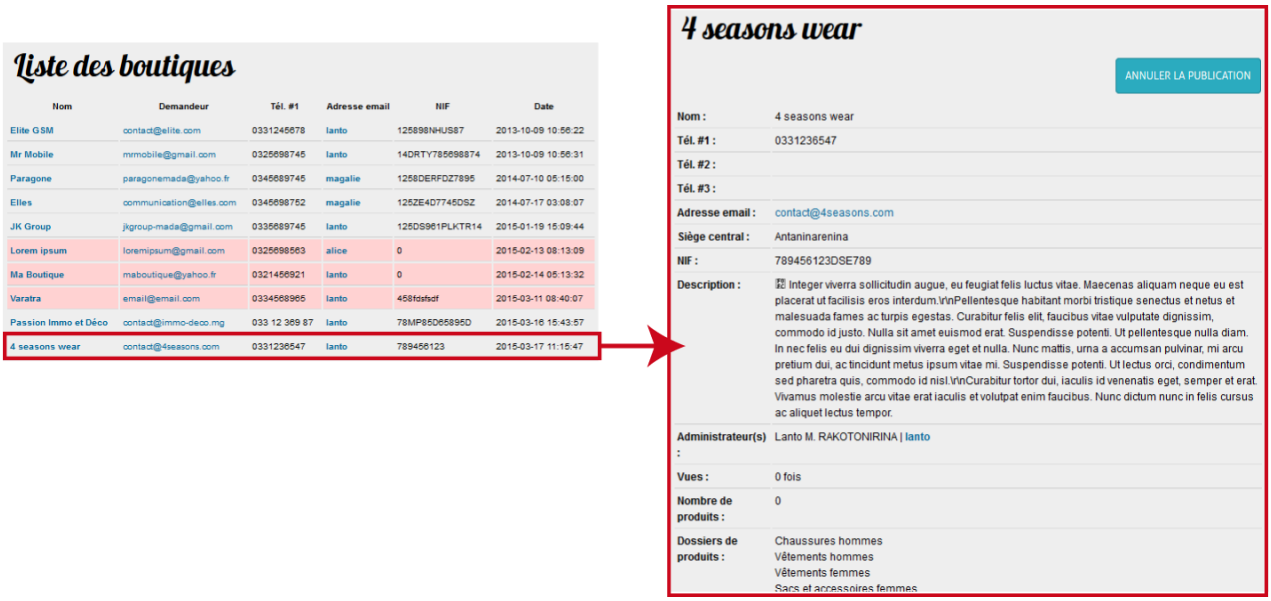


Figure 5.016 : Administration de boutiques

5.7.3. Ajout de boutique

Pour pouvoir ajouter sa boutique virtuelle, il faut avoir un compte personnel sur le site. C'est ce compte qui sera défini par défaut comme administrateur de la boutique. Cela pour permettre plus tard d'y ajouter des produits, de les modifier, de modifier les informations sur la boutique, etc.

L'ajout de boutique se fait à partir d'un simple formulaire en cliquant sur le lien « Ajouter votre boutique » du bloc « espace vendeur ».

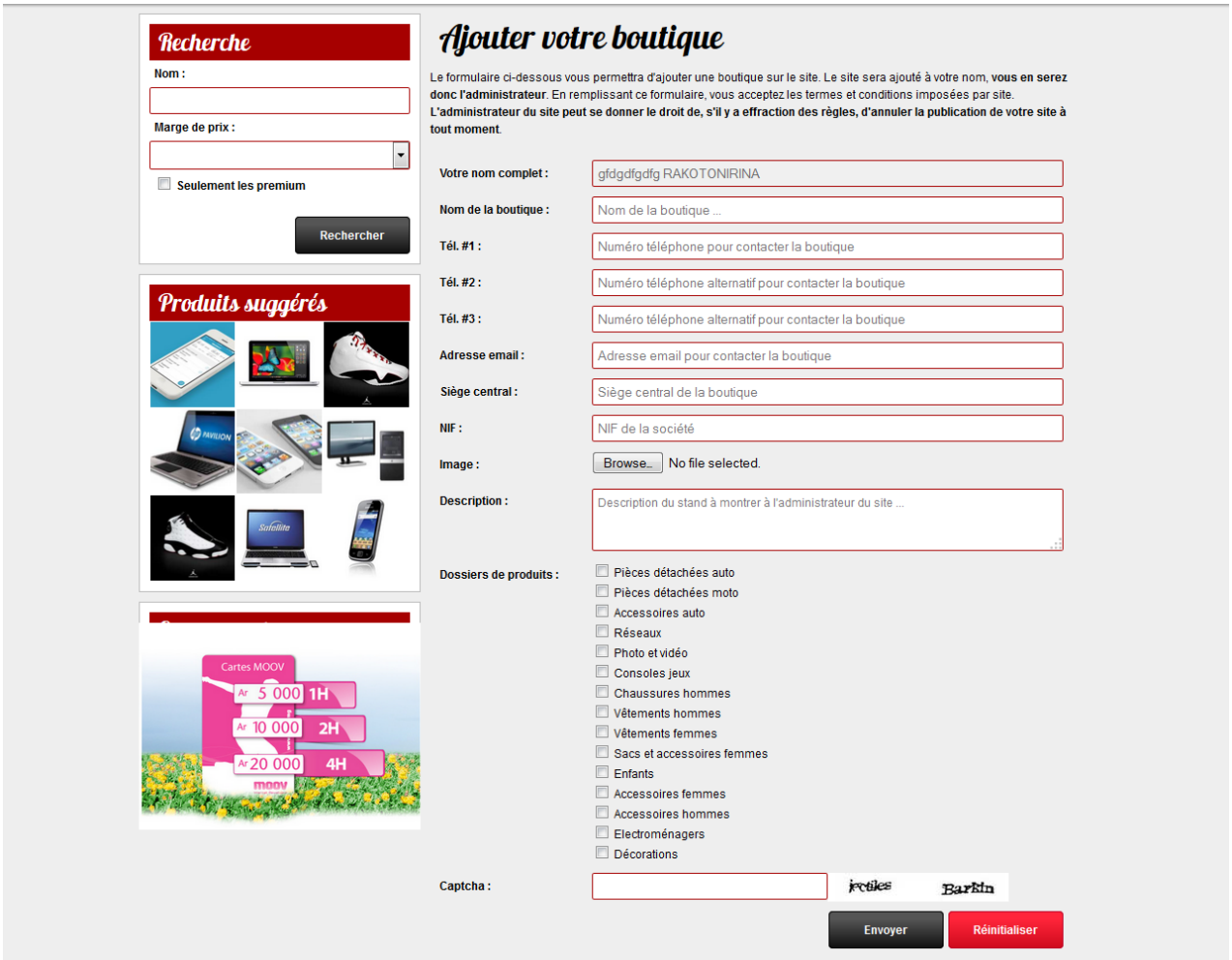


Figure 5.017 : Page d'ajout de boutique

Les champs du formulaire sont :

- Le nom complet du vendeur. Ce champ est par défaut rempli par le nom de l'utilisateur en cours (qui effectue la requête) ;
- Le nom de la boutique ;
- Une image représentative de la boutique, qui sera souvent le logo ;
- La description. Ce champ permet au vendeur de décrire en quelques phrases la boutique et permettra ainsi à l'administrateur d'en avoir une vision plus claire ;
- Les dossiers de produits (cases à cocher). Ces cases à cocher permettent au vendeur de choisir les dossiers de produits qu'il vend ;
- Le code captcha.

Une fois le formulaire envoyé, cette requête aboutira dans la liste des requêtes de l'administrateur, seul décideur pour publier ou non la boutique.

Figure 5.018 : Liste des requêtes d'ajout de boutique

En cliquant sur une ligne de cette liste, l'administrateur pourra voir les détails de la boutique. En bas de ces informations se trouvent un lien de publication de la boutique qui, une fois cliqué, mettra la boutique dans le front office pour être consultable par les acheteurs.

5.8.Simulation d'envoi de monnaie électronique

5.8.1. Introduction

A l'image de l'IPN proposé par Paypal pour PHP, nous allons simuler un paiement par monnaie électronique. Pour cela, nous allons créer un autre projet qui va servir de serveur contenant les transactions monétaires par mobile. Nous allons l'appeler STMM pour Service de Transaction Monétaire Mobile.

Premièrement, le STMM va permettre d'envoyer de l'argent mobile, transaction qui sera enregistrée en base de données avec les informations suivantes :

- Montant de la transaction ;
- Numéro du destinataire ;
- Numéro de transaction ;
- Date d'envoi de la monnaie.

Ensuite, elle servira de web service qui permettra à un autre site internet d'interroger si une transaction existe.

Ce mode de paiement s'effectue en deux (2) temps : d'abord l'acheteur envoie la somme via leur téléphone, ensuite il saisit le numéro qui lui a servi à payer cette somme et le numéro de transaction dans le formulaire correspondant. Nous allons voir par la suite le principe de ces deux (2) processus.

Ci-après le formulaire de validation du paiement.

Figure 5.019 : Formulaire de validation du paiement

5.8.2. Principe de fonctionnement de la simulation du paiement

Pour pouvoir simuler le paiement, on a créé une page qui présente un téléphone mobile permettant de faire l'envoi de monnaie électronique.

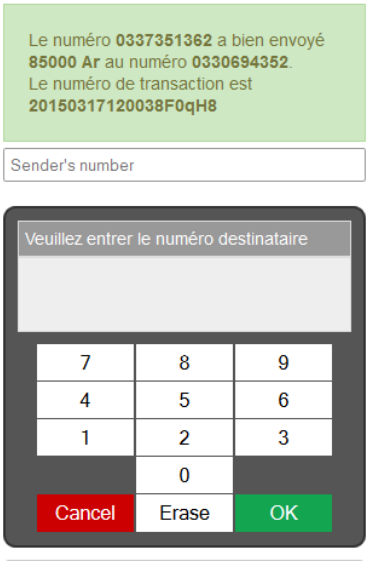


Figure 5.020 : Terminal de simulation d'envoi de monnaie électronique

Il faudra d'abord fournir le numéro de téléphone de l'émetteur qui doit respecter un bon format de numéro téléphone. Ensuite, on saisit à l'aide du clavier virtuel le numéro du destinataire. Quand ce numéro est valide, on saisit la somme à envoyer. Les données de la transaction sont alors insérées dans la base de données.

5.8.3. Principe de fonctionnement de la vérification

Le principe de fonctionnement du paiement par monnaie électronique consiste aussi en l'intégration d'un formulaire standard sur le site dont les champs sont listés dans le tableau suivant :

Attibut « name »	Type	Valeur
url	Hidden	Adresse URL pour traiter les données retournées par le service du paiement
Dst	Hidden	Numéro vers lequel la monnaie a été envoyée
Src	Text	Numéro qui a servi à envoyer la transaction. C'est sensé celui que l'utilisateur a utilisé pour régler le paiement
Transid	Text	Numéro de transaction

Tableau 5.02 : Champs du formulaire

Ce formulaire pointe vers le site distant du paiement. La soumission du formulaire va vérifier que la transaction dont les coordonnées ont été fournies (le numéro du destinateur et le numéro de transaction) existe ou non. Pour cela, le serveur du paiement, après le traitement des données reçues, va appeler la page du site fournie par le champ « url ». Voyons tout cela par la figure ci-après.

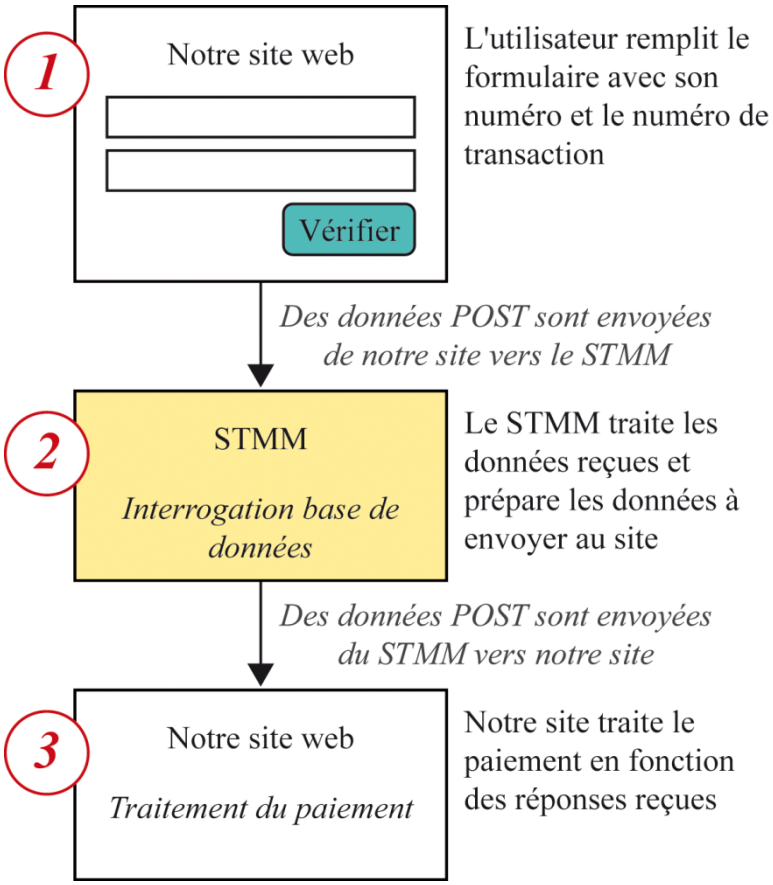


Figure 5.021 : Vérification du paiement par monnaie électronique

Les données retournées par le serveur distant sont des données POST dont les clés sont :

- **exists :** Booléen indiquant si le paiement existe (true) ou non (false). Si le paiement n'existe pas, il est inutile de récupérer les informations ci-dessous ;
- **dst :** Numéro de téléphone du destinataire ;
- **src :** Numéro de téléphone du destinateur, soit le numéro ayant servi à envoyer la somme ;
- **transid :** Numéro de transaction ;
- **amount :** Montant ;
- **date :** Date d'envoi.

5.9.Newsletter

5.9.1. Introduction

La newsletter est un très bon moyen pour communiquer avec les clients. Elle permet d'envoyer par courrier électronique aux adresses emails des internautes qui s'y sont inscrits les activités, les nouveautés, les offres, les promotions, etc. sur le site. De plus, la lettre d'information n'est pas

éditée uniquement qu'en mode texte, son habillage peut se faire en HTML et ainsi envoyer une bonne présentation graphique aux abonnés.

5.9.2. Ajout d'adresses email

Tous les internautes peuvent s'abonner à la newsletter en fournissant leur adresse email sur le formulaire d'abonnement situé sur le footer de la page. Dans le cas où le format d'email est valide et que l'email ne figure pas encore dans la base de données, alors la nouvelle adresse sera enregistrée. Par la suite, l'utilisateur recevra les newsletters prochaines envoyées par l'administrateur.

Figure 5.022 : Formulaire d'abonnement à la newsletter

En outre, l'administrateur peut aussi ajouter manuellement une nouvelle adresse dans la page « Liste des newsletters ».

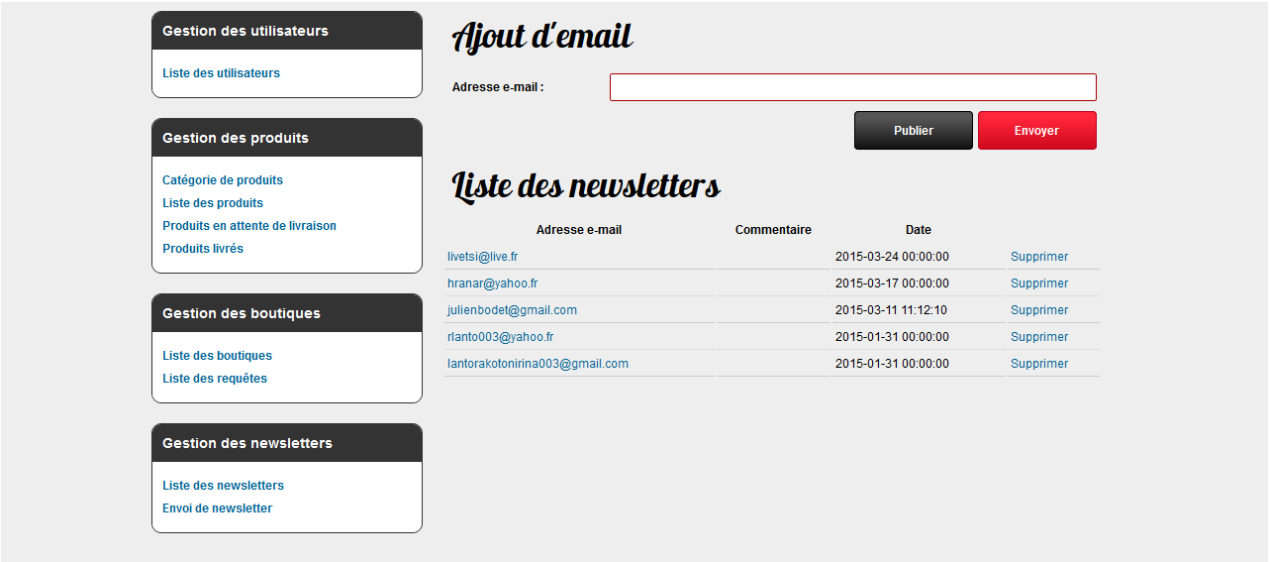


Figure 5.023 : Formulaire d'ajout d'adresse email par l'administrateur

5.9.3. Envoi de newsletter

Ci-après la page d'envoi de newsletter, uniquement accessible par un administrateur. L'interface se décompose comme suit :

1. Lien « Envoi de newsletter » qui ramène à la page conçue à cet effet ;
2. Objet du mail ;
3. Code source HTML du mail à envoyer ;
4. Aperçu du mail. Quand le code source est copié, le rendu est instantanément affiché sur cette section de la page.

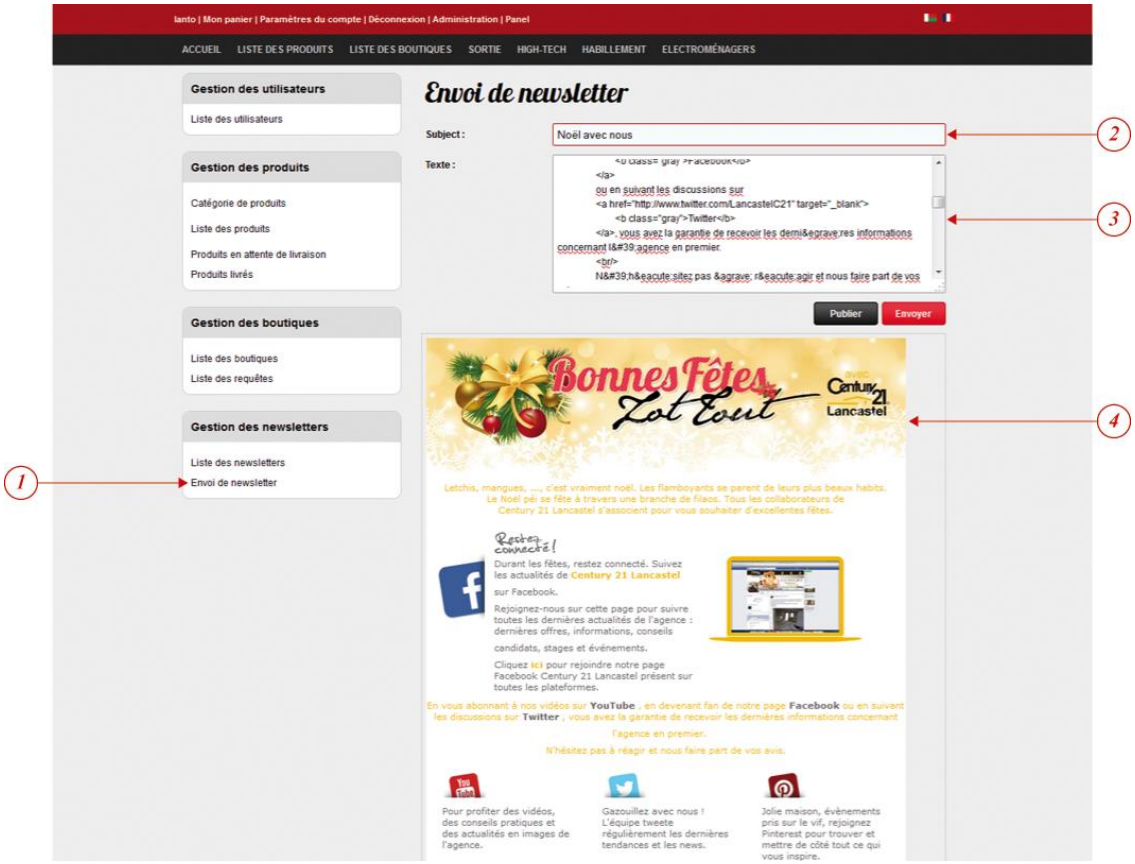


Figure 5.024 : Envoi de newsletter

A l'envoi du formulaire, tous les abonnés qui figurent dans la liste vont recevoir cette lettre d'informations.

5.10. Extrait de codes source du site avec le framework

```
class Stand extends Model {

    protected $url;
    protected $name;
    protected $tel1;
    protected $tel2;
    protected $tel3;
    protected $email;
    protected $mainLocation;
    protected $image;
    protected $nif;
    protected $queryText;
    protected $date;
    protected $published;
    protected $user;
    protected $admins;
    protected $likes;
    protected $products;
    protected $productFolders;
    protected $views;

    public function __construct() {
        //constructeur du parent
        parent::__construct();
        //mapping
        $this->tableName = "stand";
        $this->fields["url"] = "url";
        $this->fields["name"] = "name";
        $this->fields["tel1"] = "tel1";
        $this->fields["tel2"] = "tel2";
        $this->fields["tel3"] = "tel3";
        $this->fields["mainLocation"] = "mainlocation";
        $this->fields["email"] = "email";
        $this->fields["image"] = "image";
        $this->fields["nif"] = "nif";
        $this->fields["queryText"] = "querytext";
        $this->fields["date"] = "date";
        $this->fields["published"] = "published";
        $this->references = array(
            "user"=>"user:User"
        );
        $this->referencers = array(
            "admins"=>"stand:StandAdmin",
            "likes"=>"stand:StandLike",
            "products"=>"product:Product",
            "productFolders"=>"stand:StandProductFolder",
            "views"=>"product:ProductView",
        );
        //initialisations (ex : referencers new ArrayList())
        $this->admins = new ArrayList();
        $this->likes = new ArrayList();
        $this->products = new ArrayList();
        $this->productFolders = new ArrayList();
        $this->views = new ArrayList();
```

Figure 5.025 : Code source de modèle représentant une boutique

```php
class CartItemViewModel {

    protected $product;

    protected $id;
    protected $image;
    protected $name;
    protected $price;
    protected $salePrice;
    protected $resume;
    protected $stock;
    protected $deliveryDate;
    protected $published;

    protected $category;
    protected $folder;
    protected $stand;
    protected $command;

    protected $quantity;
    protected $totalPrice;

    public function __construct() {

    }

    public function getProduct() {
        return $this->product;
    }
    public function setProduct(Product $product) {
        $this->product = $product;
        $this->id = $product->getId();
        $this->image = $product->getImage();
        $this->name = $product->getName();
        $this->price = $product->getPrice();
        $this->resume = $product->getResume();
        $this->stock = $product->getStock();
        $this->published = $product->getPublished();
        $this->category = $product->getCategory();
        $this->folder = $product->getFolder();
        $this->stand = $product->getStand();
    }
    public function getQuantity() {
        return $this->quantity;
    }
    public function setQuantity($quantity) {
        $this->quantity = $quantity;
        $this->totalPrice = $this->price * $quantity;
    }
    public function getCommand() {
        return $this->command;
    }
    public function setCommand(Command $command) {
        $this->command = $command;
    }
    public function getTotalPrice() {
        return $this->totalPrice;
    }

    public function getId() {
        return $this->id;
    }
    public function getImage() {
        return $this->image;
```

Figure 5.026 : Code source de modèle de vue représentant un produit de panier

```
$routes = array(
    array("user_login", "login", "user:Authentication:login"),
    array("user_logout", "logout", "user:Authentication:logout", "LOGGED"),

    array("user_register", "register", "user:Registration:register"),
    array("user_confirm", "confirm/{param}", "user:Registration:confirm"),

    array("user_settings", "settings", "user:Profile:settings", "LOGGED"),
    array("user_profile", "profile/{id}", "user:Profile:profile", "LOGGED"),

    array("user_admin_index", "admin/users", "user:Admin:index", "ADMIN"),
    array("user_admin_user", "admin/user/{username}", "user:Admin:user", "ADMIN"),
    array("user_admin_toggle", "admin/user/toggle/{id}", "user:Admin:toggle", "ADMIN"),
);
```

Figure 5.027 : Code source de fichier de routing du bundle « user »

```
class StandQueryBuilder extends QueryBuilder {

    public function findAllStands() {
        $this->select();
        $this->referencer("stand:Stand", "product:ProductView");
        $this->join("user:User", "stand:Stand", "LEFT");
        $this->build();
        return $this->getResults();
    }
    public function findStands() {
        $this->select();
        $this->where(array("stand:Stand.published = 1"));
        $this->referencer("stand:Stand", "product:Product");
        $this->referencer("stand:Stand", "product:ProductView");
        $this->referencer("stand:Stand", "product:ProductView");
        $this->build();
        return $this->getResults();
    }

    public function findStand($mode, $url) {
        $this->select();
        if ($mode == "byUrl")
            $this->where(array("stand:Stand.url = '$url'"));
        else if ($mode == "byId") {
            $this->where(array("stand:Stand.id = '$url'"));
        }
        $this->join("user:User", "stand:Stand");
        $this->referencer("stand:Stand", "product:Product");
        $this->referencer("stand:Stand", "product:ProductView");
        $this->build();
        return $this->getOneResult();
    }

    public function findRequests() {
        $this->select();
        $this->where(array("stand:Stand.published = '0'"));
        $this->join("user:User", "stand:Stand", "LEFT");
        $this->referencer("stand:Stand", "stand:StandProductFolder");
        $this->build();
        return $this->getResults();
    }
```

Figure 5.026 : Code source de QueryBuilder correspondant à une boutique

```php
class DeliveryCore extends Core {

    public function getPendingItem($id) {
        //model & repo
        $cartItemQB = $this->getModel("product:CartItem")->getQueryBuilder();
        //cartItem
        $cartItem = $cartItemQB->findPendingItem($id);
        return $cartItem;
    }

    public function getPendingItems($user=null) {
        //model & repo
        $productQB = $this->getModel("product:Product")->getQueryBuilder();
        $cartItemQB = $this->getModel("product:CartItem")->getQueryBuilder();
        //user
        $userQB = $this->getModel("user:User")->getQueryBuilder();
        $user = $userQB->findCurrentUser();
        //list
        $list = array();
        $listTmp = $cartItemQB->findPendingItems($user);
        //viewModels
        if ($listTmp) {
            foreach($listTmp as $cartItemTmp) {
                //datas
                $product = $productQB->findOneProduct($cartItemTmp->getProduct()->getId());
                //quantity
                $quantity = $cartItemTmp->getUnity();
                //vieww models
                $productVM = $this->getViewModel("product:CartItem");
                $productVM->setProduct($product);
                $productVM->setCartItem($cartItemTmp);
                $productVM->setQuantity($quantity);
                $productVM->setCommand($cartItemTmp->getCommand());
                $list[$cartItemTmp->getId()] = $productVM;
            }
        }
        //return
        return $list;
    }
    public function getDeliveredItems($user=null) {
        //model & repo
        $productQB = $this->getModel("product:Product")->getQueryBuilder();
        $cartItemQB = $this->getModel("product:CartItem")->getQueryBuilder();
        //user
        $userQB = $this->getModel("user:User")->getQueryBuilder();
        $user = $userQB->findCurrentUser();
        //list
        $list = array();
        $listTmp = $cartItemQB->findDeliveredItems($user);
        //viewModels
        if ($listTmp) {
            foreach($listTmp as $cartItemTmp) {
                //datas
                $product = $productQB->findOneBy(array("id"=>$cartItemTmp->getProduct()->getId()));
                //quantity
                $quantity = $cartItemTmp->getUnity();
                //vieww models
                $productVM = $this->getViewModel("product:CartItem");
                $productVM->setProduct($product);
                $productVM->setCartItem($cartItemTmp);
                $productVM->setQuantity($quantity);
                $productVM->setCommand($cartItemTmp->getCommand());
                $list[$cartItemTmp->getId()] = $productVM;
            }
        }
        //return
        return $list;
    }

}
```

Figure 5.027 : Code source de Core correspondant aux livraisons

```php
class ProductController extends Controller {

    public function _suggestedAction() {
        //core
        $productCore = $this->getCore("product:Product");
        $list = $productCore->getUserProductsPreferences($this->getCurrentUser(), 15);
        //render
        $this->place(
            array(
                array(
                    "product:shared:_suggested.php",
                    array(
                        "list" => array_slice($list, 15-10),
                    )
                ),
            )
        );
    }

    public function _homeAction() {
        //core
        $productCore = $this->getCore("product:Product");
        $list = $productCore->getProducts();
        //place
        $this->place(
            array(
                array(
                    "product:product:_home.php",
                    array(
                        "totalItemCount"=>$productCore->getProductsCount(),
                        "list"=>$list,
                    )
                ),
            )
        );
    }

    public function indexAction() {
        //core
        $productCore = $this->getCore("product:Product");
        $list = $productCore->getProducts();
        //render
        $this->render(
            "layout.php",
            array(
                array(
                    "product:product:index.php:right",
                    array(
                        "totalItemCount"=>$productCore->getProductsCount(),
                        "list"=>$list,
                    )
                ),
            )
        );
    }
    public function oneAction($id) {
```

Figure 5.028 : Code source de contrôleur gérant l'affichage des produits sur le Front Office

```php
<!-- titre -->
<h1><?php echo trans("stand_stand_index_title"); ?></h1>
<!-- liste vide -->
<?php if (!$list) { ?>
    <!-- message -->
    <div class="alert alert-danger">
        <?php echo trans("default.error.emptyList"); ?>
    </div>
<?php } else { ?>
    <?php foreach($list as $stand) { ?>
        <!-- case -->
        <div class='coln-4 colm-12'>
            <div class="box-stand">
                <div class="row">
                    <!-- image -->
                    <div class="coln-12 colm-4">
                        <div class="pad-5">
                            <a href="<?php echo path("stand_stand_one", array("url"=>$stand->getUrl())); ?>
                                <img class='img-polaroid img-full-width' src="<?php echo asset("photos/stan
                            </a>
                        </div>
                    </div>
                    <div class="coln-12 colm-6">
                        <div class="stand-infos">
                            <!-- vu -->
                            <b><?php echo trans("default_views_title"); ?> :</b>
                            <?php echo trans("default_views_times", array("times"=>count($stand->getViews()
                            <br/>
                            <!-- nom -->
                            <b><?php echo trans("stand_info_name"); ?> :</b>
                            <b><a href="<?php echo path("stand_stand_one", array("url"=>$stand->getUrl()));
                            <br/>
                        </div>
                    </div>
                </div>
            </div>
            <hr class="resp-show"/>
        </div>
    <?php } ?>
    <!-- pagination -->
    <div class="clear-both"></div>
    <hr/>
    <?php paginate($totalItemCount); ?>
<?php } ?>
```

Figure 5.029 : Code source de vue qui affiche la liste de boutiques

5.11. Conclusion

Nous pouvons constater que ce site internet peut contribuer à simplifier les démarches des acheteurs. Vitrine virtuelle, recherche de produits, validation de commande, gestion de boutiques, etc., plusieurs fonctionnalités y sont implantées aussi bien pour les simples utilisateurs que pour les vendeurs. La tâche la plus délicate est le règlement des paiements, tâche achevée par une bonne coordination des opérations ; toute transaction doit passer par notre site sauf dans le cas de la validation de commande par drive. Evidemment, ce projet est amené à évoluer toujours dans le but d'améliorer la navigation des internautes et d'y apporter des fonctionnalités encore plus poussées. Enfin, de par la réalisation de ce projet, les possibilités offertes par le framework spécialement créé ont largement été vérifiées et testées.

CONCLUSION GENERALE

Ces travaux de mémoire ont pour but de faire étalage des technologies offertes par le web afin de mettre à disposition des acheteurs une plateforme multi-boutiques de vente de produits en ligne. Mais avant tout, ces technologies nous ont permis de concevoir un framework PHP. Ce dernier va être une solution offerte à tous les développeurs finaux afin de créer leurs futurs projets. Avoir un code efficient, lisible et maintenable est primordial. C'est pourquoi nous avons décidé de développer ce framework en exploitant les avantages offerts PHP et son support de la programmation orientée objet depuis sa version 5.

Nous avons pu voir que Paypal propose une solution de paiement en ligne relativement facile à mettre en place. Elle facilite grandement la vie de nombreux sites internet, avec une configuration basique et la possibilité de faire des tests sur des comptes dans la sandbox. Néanmoins, d'autres alternatives comme le paiement par drive et le paiement par monnaie électronique sont également proposées. Concernant ce dernier, on a mis en place une simulation d'envoi de monnaie dans lequel chaque transaction est enregistrée en base de données.

Dans notre site internet, plusieurs fonctionnalités ont été mises en place pour les acheteurs, les vendeurs - qui feront office de fournisseurs - et les administrateurs. De plus, la mise en place de boutiques virtuelles permet aux internautes de voir un catalogue complet de produits. Ces fonctionnalités nous ont démontré la puissance et les possibilités offertes par notre framework.

Les deux (2) projets, bien que complexe en termes de fonctionnalités, trouveront certainement des améliorations dans l'avenir, le site pour plus de fonctionnalités, le framework pour plus de possibilités, d'automatisations de tâches et pour une architecture toujours plus optimale. D'autant plus que la progression des diverses technologies du web ne cesse de s'affiner.

ANNEXE 1
HOTES VIRTUELS SOUS APACHE

Les hôtes virtuels ou virtual hosts sont très utiles pour faire fonctionner plusieurs sites web sur un même serveur web, plutôt que de les mettre derrière le site « localhost » par défaut. Aussi, il ne sera plus nécessaire de placer les fichiers de projet dans le dossier « www » de WAMP, mais dans un dossier au choix.

Figure A1.01 : Liste des sites dans « localhost »

On « héberge » plusieurs sites ou hôtes virtuels sur le même serveur Apache à l'aide de ces fichiers de configuration :

- « vhosts.conf » : Fichier qui va contenir les hôtes ;
- « httpd.conf » : Fichier de Apache ;
- « hosts » : Fichier de windows.

Le fichier « vhosts.conf » est un fichier que l'on crée dans le dossier d'Apache pour y intégrer les nouveaux hôtes virtuels. Le dossier où sera situé le fichier est « vhosts », à la racine de wamp. Une ligne d'entrée d'hôte virtuel est semblable à la figure suivante :

```
<VirtualHost *:80>
    DocumentRoot "D:/PROJETS/WEB/monsite/"
    ServerName monsite.local
    <Directory "D:/PROJETS/WEB/monsite/">
        Options FollowSymLinks
        AllowOverride All
        Allow from all
    </Directory>
</VirtualHost>
```

Figure A1.02 : Une ligne d'entrée de « vhosts.conf »

Les valeurs à fournir sont :

- Le chemin vers le projet PHP (ici « D:/PROJETS/WEB/monsite/) ;
- Le nom de domaine voulu (ici « monsite.local).

Aussi, on doit modifier le fichier « httpd.conf ». Cela consiste à ajouter une ligne pour qu'Apache prenne en compte le fichier d'hôtes virtuels. A partir du dossier d'installation de wamp, le dossier où se trouve le fichier correspond à « bin\apache\apache2.2.8\conf ». La ligne à ajouter est « include conf/vhosts.inc.conf ».

Enfin, on modifie le fichier « hosts » de Windows, situé dans « C:\WINDOWS\system32\drivers\etc\ ». Il permet signaler au système d'exploitation qu'on va assigner des noms de domaine au réseau interne (localhost), c'est en gros une sorte de DNS. La ligne à rajouter vaut « 127.0.0.1 <*nom_de_domaine*> ».

A chaque ajout d'hôte virtuel, il faut relancer Apache pour prendre en compte les modifications.

ANNEXE 2

SECURISATION D'UN SITE PAR AUTHENTIFICATION HTTP

Quand on met en ligne un site web, il se peut que l'on veuille fermer certaines zones de l'application au grand public. Ces parties restreintes sont par exemple les portails d'administration, les fichiers de données utilisateurs, la version en bêta test, etc. Pour cela, il existe une méthode simple et efficace qui permet de sécuriser tout un répertoire présent sur un serveur Web Apache au moyen d'une authentification HTTP. Cette identification sera assurée par couple un login / mot de passe.

Le principe consiste à créer deux (2) fichiers de configuration qui définissent les règles de protection ainsi que les couples login / mot de passe des utilisateurs autorisés à pénétrer dans le répertoire. Les fichiers sont :

- « .htaccess » : Permet de surcharger la configuration originale du serveur Apache sur lequel le site est hébergé ;
- « .htpasswd » : Contient les couples logins / mots de passe.

Nous voyons ci-dessous le contenu du fichier « .htaccess » qui permet de sécuriser le répertoire :

```
AuthUserFile /Users/Emacs/Sites/ApprendrePHP/administration/.htpasswd
AuthGroupFile /dev/null
AuthName Administration
AuthType Basic
require alice bob cesar
```

Voyons une à une les lignes du fichier :

- 1 : Indique le chemin canonique absolu menant au fichier « .htpasswd » qui contient les couples logins / mots de passe. Il est à noter que la fonction PHP realpath() permet de fournir le chemin canonique d'un fichier placé en paramètre ;
- 2 : Précise qu'on n'utilise pas le fichier de groupe ;
- 3 : Intitule la boîte de dialogue de l'authentification ;
- 4 : Le type « basic » indique la méthode qui sera utilisée pour l'authentification ;
- 5 : Liste les utilisateurs (séparés par un espace) ayant le droit d'accéder au répertoire après identification.

Quant au fichier « .htpasswd », sont contenu est très sommaire. Il contient uniquement les couples logins / mots de passe des utilisateurs autorisés, les éléments de cette paire étant séparés par « : ».

ANNEXE 3
REFERENCEMENT (SEO)

Le SEO pour Search Engine Optimisation consiste à employer un certain nombre de techniques dans le but de donner de la visibilité à un site internet sur les moteurs de recherche. En d'autres termes, c'est pour faire en sorte qu'un site puisse apparaître dans les premières pages (voire la première), lorsque qu'un internaute effectue une recherche sur une requête précise.

La finalité du SEO est donc de faire apparaître une ou plusieurs pages d'un site dans les premiers résultats proposés par les moteurs de recherche, lorsque l'internaute compose une requête comprenant des mots clés correspondants.

Il est à noter que les résultats organiques ou naturels sont bien différents des résultats commerciaux. En effet on parle de résultats organiques dans le cadre d'un référencement naturel et non payant, et de résultats commerciaux lorsque le référenceur investit financièrement pour apparaître sur une requête. Google Adwords est le principal outil de référencement payant.

Concernant le SEO (référencement naturel) il existe deux types de facteurs permettant d'optimiser la visibilité d'un site :

- Les facteurs internes au site lui-même (on page) :
    - Les pages doivent avoir un titre ;
    - Penser à renseigner les balise méta ;
    - Donner une description aux images en utilisant l'attribut « alt » ;
    - Hiérarchiser le contenu du site ;
    - Réécrire les URLs ;
    - Élaborer un contenu éditorial riche en mot clés ;
    - Préférer un nom de domaine simple et non concurrentiel ;
- Les facteurs externes au site (off page) :
    - La quantité d'hyperliens menant au site (linking) ;
    - La qualité de ces liens (conception du lien lui-même dans le html, qualité du site émetteur de ce lien, etc.) ;
    - Ancienneté du site et du nom de domaine .

Toutes ces méthodes n'ont qu'un seul objectif final, celui de créer du traffic sur un site internet. Plus un site internet connaîtra un trafic important, mieux il sera référencé par les moteurs de recherche.

ANNEXE 4

EXTRAITS DE CODES SOURCE DU FRAMEWORK

```php
require_once("app/session.php");
require_once("app/bundles.php");
require_once("app/visits.php");



/**
 * VERSION PROD
 */
$isProd = $GLOBALS["isProd"];

/**
 * ROUTING
 */

//paramètres
@$queryURL = $_GET["query"];
$controllerURL = "";
$bundleURL = "";
$actionURL = "";
$arrayArguments = array();
//on continue si on n'est pas sur le chemin "/"
if ($queryURL) {
    //pour savoir s'il n'y a eu aucun résultat
    $hasResult = 0;
    //boucle sur les routes
    foreach($routes as $routeParametres) {
        //paramètres
        $routePath = @$routeParametres[1];
        $routeAction = @$routeParametres[2];
        $routeNeedsLogin = @$routeParametres[3];
        //traitement des "{}" pour avoir une regex
        $routePathRegexed = preg_replace("/\{[a-z0-9]*\}/i", "([a-z0-9_-]*)", $routePath);
        $routePathRegexed = preg_replace("#\/$#i", "\/*", $routePathRegexed); //pour les valeurs facultatifs : ex :
        //si la route en cours existe dans le tableau, on break
        if (preg_match_all("'^".$routePathRegexed."$'i", $queryURL, $out1)) {

            //si le format de route n'est pas respecté
            if (!preg_match_all("/^[a-zA-Z0-9]*:[a-zA-Z0-9]*:[_a-zA-Z0-9]*$/i", $routeAction, $out2))
                if (!$isProd)
                    die("Route format is not valid. Must be \"bundle:ControlerName:methodName\".");
                else
                    die("404 error");
            //sécurité
            else if ($routeNeedsLogin == "LOGGED" && !$_SESSION["isAuthenticated"]) {
                addFlashBag("danger", trans("default.error.needsLogin"));
                redirect(path("user_login"));
            }
            else if ($routeNeedsLogin && $routeNeedsLogin != "LOGGED" && !Session::hasRoles($routeNeedsLogin)) {
                //redirect("/404");
                die("THIS PAGE NEEDS PRIVILEGE");
            }
            //si tout va bien
            else {
                //hachage du format de route
                $routeAction = explode(":", $routeAction);
                //extraction des données GET : contrôleur, action, arguments d'actions
                $bundleURL = $routeAction[0];
                $controllerURL = $routeAction[1];
                $actionURL = $routeAction[2];
                foreach($out1 as $k=>$v) {
                    if ($k != 0)
                        $arrayArguments[] = $v[0];
                }
            }
            //hasResult
            $hasResult++;
            //break
            break;
        }
```

Figure A4.01 : Code source dans « index.php »

```php
//chargement du core
public function getCore($core) {
    //hachage de l'argument "bundle:Model"
    $core = explode(":", $core);
    $bundleDirName = $core[0];
    $coreFileName = $core[1];
    //import du fichier et instanciation
    if (file_exists("bundles/$bundleDirName/core/$coreFileName"."Core.php")) {
        require_once("bundles/$bundleDirName/core/$coreFileName"."Core.php");
        $className = $coreFileName."Core";
        return new $className();
    }
    else {
        if (!$this->isProd)
            die("Controller::getCore() No Core class found in : bundles/$bundleDirName/core/$coreFileName"."Cor
        else {
            redirect("/404.php");
        }
    }
}
//chargement du modele
public function getModel($model) {
    //hachage de l'argument "bundle:Model"
    $model = explode(":", $model);
    $bundleDirName = $model[0];
    $modelFileName = $model[1];
    //import du fichier et instanciation
    if (file_exists("bundles/$bundleDirName/models/$modelFileName.php")) {
        require_once("bundles/$bundleDirName/models/$modelFileName.php");
        return new $modelFileName();
    }
    else {
        if (!$this->isProd)
            die("Controller::getModel() No Model class found in : bundles/$bundleDirName/models/$modelFileName.
        else {
            redirect("/404.php");
        }
    }
}
//chargement du modele de vue
public function getViewModel($model) {
    //hachage de l'argument "bundle:Model"
    $model = explode(":", $model);
    $bundleDirName = $model[0];
    $modelFileName = $model[1];
    //import du fichier et instanciation
    if (file_exists("bundles/$bundleDirName/viewmodels/$modelFileName"."ViewModel.php")) {
        require_once("bundles/$bundleDirName/viewmodels/$modelFileName"."ViewModel.php");
        $className = $modelFileName."ViewModel";
        return new $className();
    }
    else {
        if (!$this->isProd)
            die("No Model class found in : bundles/$bundleDirName/viewmodels/$modelFileName"."ViewModel.php");
        else {
            redirect("/404.php");
        }
    }
}
//chargement de la vue 1 : render
public function render($layout, $arrayViews=array()) {
    //inclusion des templates enfants
    $arrayTemplatesChildren = array();
    foreach ($arrayViews as $arrayView) {
        //paramètres de chaque élément du tableau (vue & position & dataViews)
        $view = $arrayView[0];
        $dataViews = @$arrayView[1];
```

Figure A4.02 : Code source dans la classe Controller

```
    }
}
public function select($fields=array()) {
    //erreur : argument non tableau
    if (!is_array($fields)) {
        die("QueryBuilder::select argument has to be an Array.");
    }
    //"SELECT"
    $sqlText = "SELECT ";
    //si $fields est vide, on prend tous les champs
    if (!count($fields)) {
        foreach($this->tableFields as $fieldObject=>$fieldBDD) {
            $sqlText .= $this->model->getTableName().".$fieldBDD AS ".$this->model->getTableName()."__$fieldBDD, ";
        }
    }
    //sinon on prend les champs indiqués
    else {
        //ajouter automatiquement le champ "id" sinon aucune données de references ou referencers n'est retournée si on ne le précise pas
        $sqlText .= $this->tableName.".id"." AS ".$this->tableName."__id, ";
        //boucle sur les champs
        foreach($fields as $field) {
            //si c'est un champ normal
            if (preg_match_all("/^[a-z0-9]+\:[a-z0-9\._]*$/i", "$field", $out)) {
                //extraction des informations
                $field = explode(".", $field);
                $modelName = $field[0];
                $fieldObject = $field[1];
                //nom de la table et du champ correspondant en BDD
                $modelTmp = $this->callClass($modelName);
                $tableName = $modelTmp->getTableName();
                $fieldsObject = $modelTmp->getFields();
                $fieldBDD = $fieldsObject[$fieldObject];
                //$sqlText
                $sqlText .= "$tableName".".$fieldBDD"." AS $tableName"."__$fieldBDD, ";
            }
            //si c'est une fonction "COUNT(x.x) AS x ; CONCAT('x', x.x) AS x"
            else if (preg_match_all("/^([a-z]*) *\(([a-z0-9]+\:[a-z0-9]*)\.([a-z0-9]*)\) *AS *([a-z0-9\._]*)$/i", "$field", $out)) {
                //extraction des informations
                $fonction = $out[1][0];
                $modelName = $out[2][0];
                $fieldObject = $out[3][0];
                $alias = $out[4][0];
                //nom de la table et du champ correspondant en BDD
                $modelTmp = $this->callClass($modelName);
                $tableName = $modelTmp->getTableName();
                $fieldsObject = $modelTmp->getFields();
                $fieldBDD = $fieldsObject[$fieldObject];
                //$sqlText
                $sqlText .= "$fonction($tableName.$fieldBDD) AS $alias, ";
            }
            else {
                die("QueryBuilder::select field <b>\"$field\"</b> format is unknown");
            }

        }
    }
    //substr() pour enlever la virgule
    $sqlText = substr($sqlText, 0, -2);
    //modification de la valeur de $this->select()
    $this->sqlTextSelect = $sqlText;
}
public function join($objectInverse, $objectProprietor, $direction="") {
    //$sqlText
    $sqlText = "";
    //instanciation de la classe
    $object = $this->callClass($objectInverse);
    $objectTableName = $object->getTableName();
    //référence
        //instanction et nom de table
        $reference = $this->callClass($objectProprietor);
        $referenceTableName = $reference->getTableName();
    //mise à jour de $this->sqlTextSelect
    $objectTableProperties = $object->getFields();
    foreach($objectTableProperties as $fieldObject=>$fieldBDD) {
```

Figure A4.03 : Code source de la méthode select() dans la classe QueryBuilder

BIBLIOGRAPHIE

[1] E. Randriarijaona, *« Réseaux TCP/IP »*, Cours 3me année Licence Professionnelle, Dép. Tél. - E.S.P.A., A.U. : 2010-2011

[2] *www.commentcamarche.com*, « Client serveur »

[3] Olivier Caron, *« Les architectures 3-tiers »*, Polytech'Lille

[4] Jean-Yves Antoine, *« Ergonomie des Interfaces Homme-Machine »*, Cours 3me année Licence Informatique, Université François Rabelais Campus Blois

[5] Philippe Gaussier & Alexandre Pitti, *« Interface Homme-Machine Cours 1 »*, Laboratoire ETIS UCP – ENSEA

[6] Fabien Duchateau, *« Conception des IHM »*, Université Claude Bernard Lyon 1, A.U. : 2014-2015

[7] A. Seigneur, *« Bases de la conception de site web »*, UFR Mathématiques et Informatique Paris 5, Déc. 2004.

[8] M. Ravonimanantsoa, *« Web et XHTML »*, Cours 3me année Licence Professionnelle, Dép. Tél. - E.S.P.A., A.U. : 2010-2011.

[9] O. Boébion, *« Initiation PHP-MySQL : HTML, HTTP, URL, PHP, Vocabulaire, principes et premiers pas »*, 12 Février 2004.

[10] *http://www.siteduzero.com/tutoriel-3-309961-dynamisez-vos-sites-web-avec-javascript.html*, « Dynamisez vos sites web avec Javascript »

[11] *http://jlguillaume.free.fr/www/documents/teaching/ntw1011/LI385_C2_Ajax.pdf*, « Cours Nouvelles Technologies du web, Ajax »

[12] h*ttp://www.siteduzero.com/tutoriel-3-160891-jquery-ecrivez-moins-pour-faire-plus.html*, « JQuery, écrivez moins pour faire plus »

[13] *http://www.json.org/jsonfr.html*, « Présentation de JSON »

[14] M. Ravonimanantsoa, *« Développement d'application d'entreprise »*, Cours 3me année Licence Professionnelle, Dép. Tél. - E.S.P.A., A.U. : 2010-2011

[15] D. Ducrocq C.N.T.E Liévin Hénin Beaumont, *« Initiation à la création de sites Web dynamiques »*, Janvier 2002

[16] M. Ravonimanantsoa, *« Bases de données »*, Cours 3me année Licence Professionnelle, Dép. Tél. - E.S.P.A., A.U. : 2010-2011

[17] A. Cornuéjols, *« Bases de données concepts et programmation »*, AgroParisTech, Spécialité Informatique (2009-2010), 19 Octobre 2009

[18] M. Cabaré, L. Lallias, « PHP Introduction Cours », Novembre 2001 version 2.0.

[19] *http://www.phpsources.org/*, « Syntaxe SQL ».

[20] *http://erci.no-ip.com/origine/divers/mysql/*, « Syntaxe SQL ».

[21] h*ttp://www.oeconomia.net/private/cours/economieentreprise/themes/ecommerce.pdf*, « Le e-business (ou e-commerce) »

[22] *http://www.assistant-juridique.fr/creer_site_commerce.jsp*, « Les 10 étapes pour créer son site e-commerce ».

[23] *http://www.conseilsmarketing.com/e-marketing/les-7-etapes-pour-lancer-son-site-ecommerce*, « Les 7 étapes pour lancer son site ecommerce »

[24] *http://www.commentcamarche.net/contents/311-e-business*, « e-Business ».

[25] h*ttp://searchcio.techtarget.com/definition/e-business*, « What is e-business »

[26] *http://fr.wikipedia.org/wiki/PayPal*, « Paypal ».

[27] *http://www.lafermeduweb.net/billet/tutorial-integrer-paypal-a-son-site-web-en-php-partie-1-275.html*, « Intégrer Paypal à son site web en PHP ».

[28] *http://www.eric-couchelou.net/installer-integrer-paypal-site/*, « Intégrer un système de paiement Paypal sur son site ».

[29] *http://www.linux-france.org/article/these/the_osd/fr-the_open_source_definition_monoblock.html*, « La definition de l'Open Source »

[30] *http://fr.wikipedia.org/wiki/Open_Source_Definition*, « Open Source Definition »

[31] *E*. Randriarijaona, *« Programmation Orienté Objet »*, Cours 3me année, Dép. Tél. - E.S.P.A., A.U. : 2011-2012

[32] vincent1870, *« Adopter un style de programmation clair avec le modèle MVC »*, Openclassrooms, 30 Octobre 2013

[33] Immobilis, *« L'architecture multicouche »*, developpez.com, 20 octobre 2010.

[34] *E*. Randriarijaona, *« Développement d'applications d'entreprise »*, Cours 5me année, Dép. Tél. - E.S.P.A., A.U. : 2013-2014

[35] *https://www.dotnetdojo.com/mvvm/*, « MVVM ? En théorie c'est simple »

FICHE DE RENSEIGNEMENTS

Auteur :

Nom ***:*** RAKOTONIRINA

Prénoms ***:*** Lantoniaina Mampionona

E-mail ***:*** *rlanto003@yahoo.fr*

Téléphone ***:*** 033 73 513 62

Adresse ***:*** Logt 216 Ampefiloha Cité Antananarivo 101

Titre du mémoire :

« CONCEPTION ET REALISATION D'UN FRAMEWORK PHP MVC ET D'UN SITE INTERNET DE VENTRE EN LIGNE MULTI-BOUTIQUES »

Nombre de pages ***:*** 139

Nombre de tableaux ***:*** 5

Nombre de figures ***:*** 63

Directeur de mémoire :

Nom : RANDRIARIJAONA

Prénoms : Lucien Elino

Tél. : +261 32 04 747 95

E-mail : elrandria@yahoo.com

RESUME

La simplicité et la puissance du langage PHP nous ont permis de créer notre framework, qui dans un second temps, ont permis de créer les parties Front Office et Back Office de notre site internet de vente en ligne multi-boutiques. Le développement de ce dernier a bel et bien prouvé que notre framework pourra être utilisé par tous les développeurs finaux pour servir de base solide pour de futurs projets. Ce mémoire, en plus de rendre service aux développeurs PHP, est une réelle occasion de mettre en œuvre notre e-business via le site conçu. Il vise tous les consommateurs, mais aussi les sociétés commerciales, qui sont à la fois nos fournisseurs de produits. L'exposition de plusieurs boutiques sur le site donne aux acheteurs une multitude de choix de produits, avec l'avantage de rester chez soi. En plus de plusieurs fonctionnalités très utiles, le site propose plusieurs solutions de paiement pour la livraison des commandes effectuées.

Mots clés : site internet, MVC, framework PHP, e-commerce, Paypal

ABSTRACT

The simplicity and the power of PHP language both permitted us to create our framework. This framework, then, truly helped us to create the Front Office and the Back Office part of our e-commerce and multi-shop website. This website proved the usefulness of our Framework because it can be used by every developer to have a strong base for their future projects. This work, in addition to offer a great benefit to PHP developers, is also a great opportunity to start our e-business achieved with our website. It aims every consumer, but also commercial enterprises, which are products providers. The existence of many shops gives to buyers a wide choice of products, staying at home. In addition to all the useful features, the website proposes many payment solutions to treat commands delivery.

Keywords : website, MVC, PHP framework, e-commerce, Paypal