| | | |
|---|---|---|
| | Page de garde des rapports de stage | Date du stage<br>du 16/06/14<br>au 19/12/14 |
| | **Stage d'expérience** ❒<br>**Stage d'exécution** ❒<br>**Stage élève – ingénieur** ❒<br>**Stage année de césure :**<br>**❒ 12 mois ❒1er semestre ❒2ème semestre**<br>**Stage projet de fin d'études : x** | |
| | | Année |

Nom et Prénom du stagiaire : **Arthur SIMON**

Spécialité : ❒ CGP **x** ETI

Confidentialité du rapport : ❒ OUI **x** NON

Titre du rapport en français :

## Développement du site Dior Mag sur le CMS Drupal 7

Titre du rapport en anglais :

## Web development for Dior against the Drupal CMS


## Nom de l'entreprise /organisme : Smile

Ville : Asnières-sur-Seine Pays : France

Nom et Prénom du maître de stage : Lucie Moineau

Fonction : Responsable de production BU Paris WebCMS

Avant tout, il convient de remercier toutes les personnes de Smile qui m’ont impressionné par leur gentillesse et leur professionnalisme en m’offrant ma première expérience réelle du monde de l’entreprise. Mention spéciale à Jérémy Bardeau.

Je remercie également très chaleureusement Mathilde Stéphan et ma famille pour leur soutien indéfectible.

Enfin, un grand merci à l’équipe de CPE, dont Françoise Perrin, Caroline Chamot-Rooke, Christine Liatard et Nicole Gache pour leur disponibilité.

## Résumé

Dior est une marque de luxe française mondialement connu. En raison du marché dans lequel ils évoluent, et de leur clientèle, les responsables de Dior sont forcés d'être absolument irréprochables sur l'image qu'ils renvoient de l'entreprise. Il est normal qu'en 2014 ils accordent alors un soin tout particulier à leurs sites en ligne.
C'est pourquoi, alors que le site Dior Mag construit via le CMS eZPublish n'est pas vieux de trois ans, ils ont décidé de faire une refonte totale du site en le passant sous le CMS Drupal 7. Un des principaux objectifs de cette refonte est également de grandement faciliter la contribution d'articles et de contenus sur le site par les responsables Dior sans perdre en rien de leurs exigences en termes de rendu graphique.
Nous étions deux développeurs chargés de l'implémentation technique du site, depuis l'installation des premiers modules de bases à la livraison du code sur les serveurs de test, en passant par l'intégration des montages HTML préalablement réalisés dans un environnement fonctionnel. Outre des problèmes inhérents à certains modules Drupal - qu'il convient de résoudre par un peu d'astuce, beaucoup d'expérience et en continuant toujours d'apprendre, puisque ceux-ci sont rarement documentés - , il nous a fallu déployer tout un arsenal d'outils et de technologie pour parvenir à développer le site Dior Mag. Comme Drupal fonctionne dans un environnement LAMP, les outils fournis par UNIX ou installés sur Linux (PC personnel ou serveur) en ligne de commande sont nombreux : tâches planifiées, administration de Drupal via une interface Shell, bases de données MySQL, scripts de livraisons, partage de l'environnement empaqueté dans une LXC, etc. Le partage du code est un incontournable du développement, et s'est fait via Git dans le cas de Dior.
Après près de deux mois de développement et à seulement une semaine de la fin, quand il ne restait qu'une page à construire, Dior décida d'arrêter les opérations en cours pour se donner le temps de redéfinir leur stratégie. Au moment de mon départ de l'entreprise commençait un nouveau projet appelé "Dior News", qui semble reprendre plusieurs fonctionnalités de Dior Mag, condensées sur une seule page.

# Sommaire

# Introduction

**Smile** est une société de services en ingénierie informatique (SSII) spécialisée dans l'intégration de solutions Open Source, et se réclame comme le leader de ce marché tant au niveau français qu'européen **(1)**. Si elle est fière de cette position, c'est surtout parce que l'Open Source est depuis quelques années très à la mode, et pas seulement dans le garage de quelques informaticiens en manque d'occupation. Les services proposés gratuitement par Google, les services d'hébergement d'Amazon et l'émergence d'une multitude de services en modèle freemium ces dernières années a quelque peu démodé le modèle de vente de licence logiciel qui avait propulsé Microsoft dans les années 1980, si bien qu'aujourd'hui "la croissance du marché Open Source est plus rapide que la croissance du marché IT" **(2)**. Les solutions propriétaires généralistes étant facilement reproductibles, on peut même entrevoir qu'à l'avenir le modèle de licence sera plus cantonné dans des niches ou dans des domaines dans lesquels une véritable expertise et un travail de fond sont nécessaires.

J'ai intégré en juin 2014 la Business Unit (BU) **Paris WebCMS** dédiée à la réalisation de sites web basés sur des systèmes de gestion de contenu (ou CMS, pour Content Management System) open source tels que Typo3, eZPublish ou Drupal. Après près de deux mois de Tierce Maintenance Applicative (TMA) passés sur des sites réalisés préalablement par l'équipe comme Total ou Vinci, nous avons, un ingénieur études et développement (IED) et moi-même, commencé le développement du site **Dior Mag** sous **Drupal 7**.

# 1 Contexte et organisation

## 1.1 Dior

Dior est une entreprise mondialement connu. Elle fait parti de l'écosystème des marques de luxes françaises, dont elle est l'une des représentantes les plus importantes : mode et textile, parfumerie, maquillage, joaillerie, etc. Elle appartient au groupe LVMH, qui contrôle également les prestigieux Louis Vuitton, Givenchy, Kenzo ou Guerlain pour ne citer qu'eux.

Etant donné le contexte dans lequel elles évoluent, ces marques sont particulièrement soucieuses de leur image. En effet, leur image, leur style, leur univers, c'est leur fond de commerce. Pas étonnant alors qu'une marque aussi importante que Dior ait développé une stratégie web ambitieuse. Celle-ci repose sur deux éléments principaux :


**Figure 1.1 – Réseaux sociaux**

- des réseaux sociaux de toute sorte, qui permettent de communiquer différement à des populations très variées : Facebook, Google+ et Twitter sont les plus généralistes ; Instagram et Tumblr permettent de toucher une population souvent jeune et branchée ; enfin le marché chinois à ses propres canaux pour accéder à un marché émergent de nombreux millionnaires.

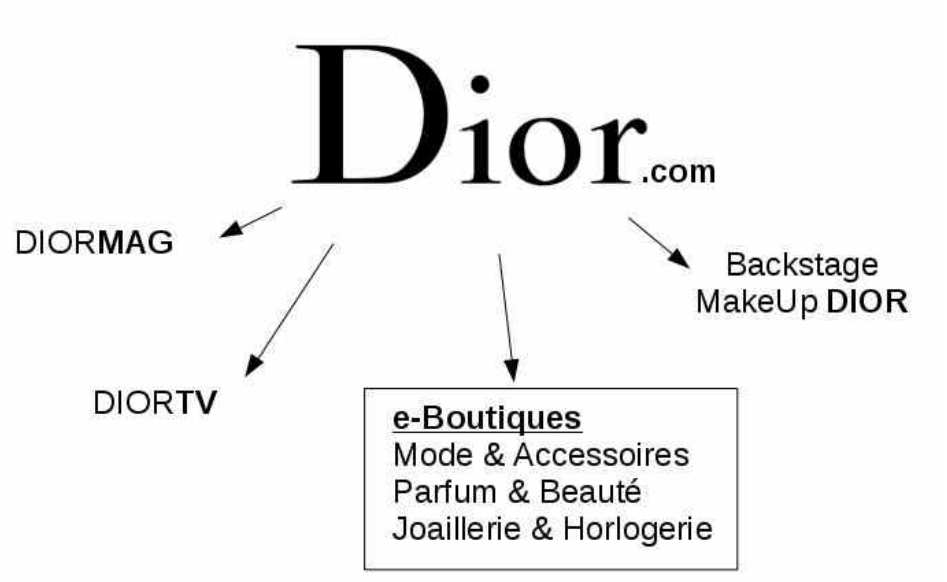



**Figure 1.2 – Dior sur le Web**

- Sur l'image ci-dessus, il est possible d'observer l'organisation actuelle des sites Dior. En effet, Dior.com doit en fait être vu comme une plateforme qui redirige vers chacun des autres sites du groupe : un site sur le maquillage, les techniques et les produits ; une webTV pour regrouper tous les supports vidéos de la marque ; un ensemble de e-boutiques pour la vente par correspondance ; et enfin Dior Mag, le site dédié aux actualités de la marque. C'est ce dernier qu'il nous a été donné de refaire.

## 1.2 Un nouveau Dior Mag

Dior Mag est le site des actualités de la marque Dior depuis 2011 : nouvelle collection, défilé, fashion week, évènement, soirée de charité ou encore dossier thématiques, toute la ligne éditoriale actuelle de la marque passe par ce biais.

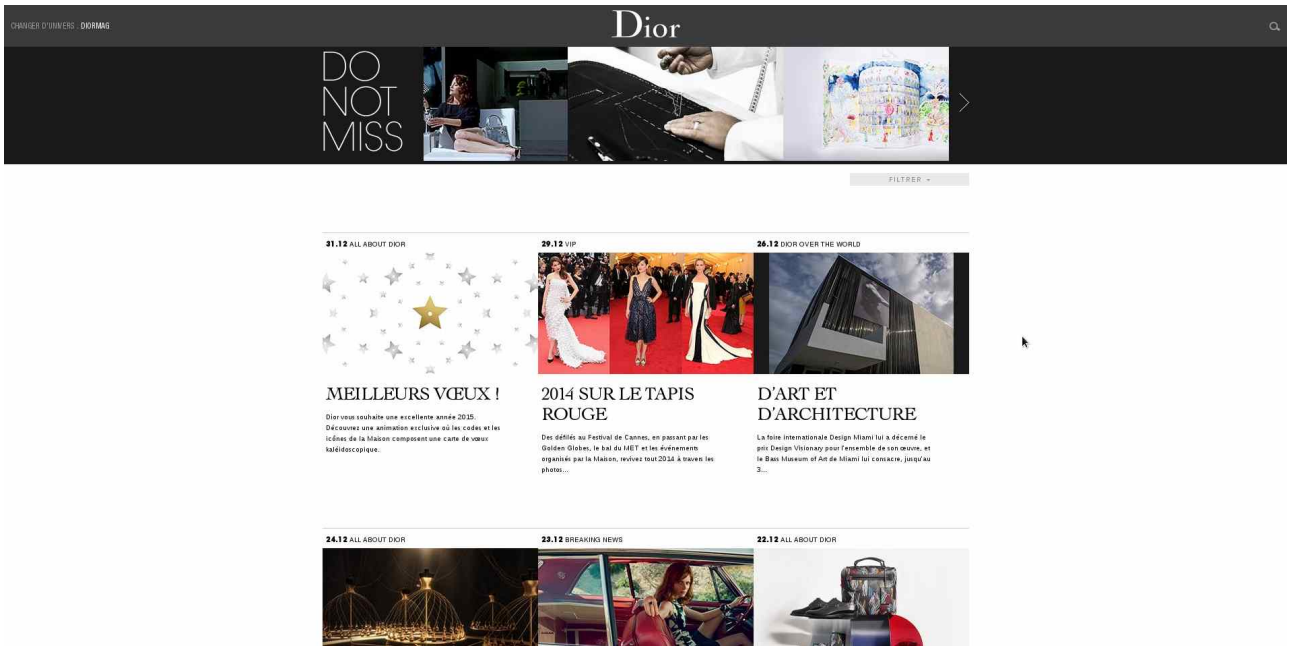



**Figure 1.3 – Dior Mag sous eZPublish**

L'ancienne version de Dior Mag, visible ci-dessus, était propulsée par le CMS eZPublish. Le design est moderne et toujours d'actualité après trois ans d'exploitation. On peut aisément imaginer le désir de Dior de changer d'interface après cette durée de vie : le monde de la mode est après tout basé sur l'éphémère et le renouveau.

Dior a fait part de sa volonté de changement d'environnement back-office, invoquant une **trop grande complexité de contribution des contenus sur eZPublish.** C'est la principale raison de l'appel d'offre gagné par Smile. Il a ainsi été décidé qu'**un nouveau site verrait le jour**. Celui-ci serait basé sur le CMS **Drupal** dans sa version 7, et serait construit “from scratch” sans reprendre aucun élément de l'ancienne plateforme.

Le choix de Drupal comme CMS est également dû à la popularité de ce dernier. Puisque celui-ci est Open Source, il a donc une communauté très active qui continue de développer et de maintenir la plateforme Drupal et toutes les fonctionnnalités qui lui sont liés. C'est donc un pari sur l'avenir et un gage d'évolutivité et de maintenance.

### 1.3 L'organisation du projet

Un autre acteur était présent pour la rédaction des spécifications avec Dior : l'agence digitale Mazarine, qui était en charge des spécifications graphiques. Leur rôle était donc de fournir des maquettes des pages du site à un niveau de détail important : taille des éléments, couleurs, polices, animations et descriptifs pour tous les formats d'appareil, puisque le site devait également être “responsive”, c'est-à-dire respecter les principes du “responsive design” pour être agréables à naviguer sur mobiles, tablettes ou desktop. Ces fichiers était généralement sous format .psd (PhotoShop Document) ou sous format vidéo.

Après la phase de spécifications graphiques vient la phase de montage : le monteur est chargé de traduire les spécifications graphiques en HTML, en s'aidant des feuilles de style CSS et du langage javascript pour les animations de la page. A la sortie, on doit obtenir un site factice, non-fonctionnel, mais dont tous les éléments sont facilement identifiables et directement récupérables pour être intégrés à la partie fonctionnelle du site fournie par Drupal.

Deux développeurs étaient chargés de l'intégration : Ludovic Beaubras, ensuite remplacé par Antoine Forgue, et moi-même. Notre mission était de paramétrer le CMS, activer toutes les fonctionnalités requises et les adapter au fonctionnel voulu par le client, page par page, fonctionnalité par fonctionnalité, puis d'appliquer les styles fournis par le monteur. Drupal est un outil très puissant qui offre de très nombreuses possibilités, mais sa complexité est un frein à l'apprentissage et une réelle connaissance de l'outil est nécessaire.

Le projet était géré par Kateryna Astafieva. L'expert technique, Alan Moreau, a une excellente connaissance du CMS et est un référent pour les développeurs dans le cas d'interrogations sur des problèmes techniques.

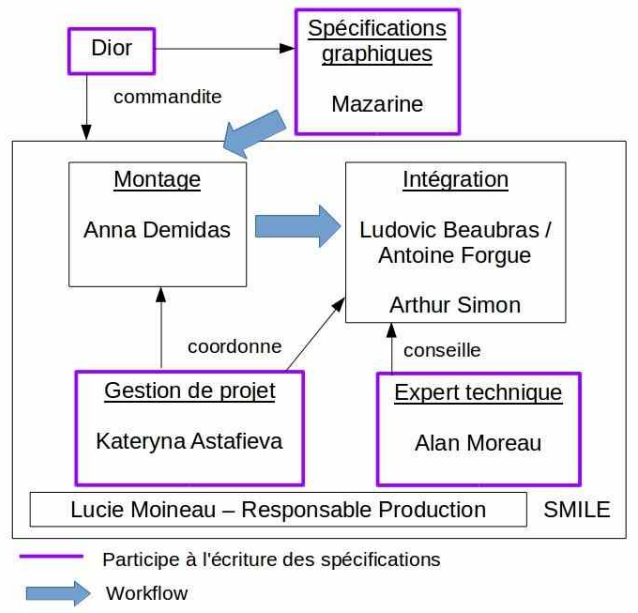



**Figure 1.4 – Organigramme**

C’est Mazarine qui était en contact avec Dior depuis longtemps et lui a vendu la solution Drupal. Smile a remporté l’appel d’offre pour l’implémentation technique, probablement car c’est une équipe de la société qui avait développé l’ancien site DiorMag avec le CMS eZPublish et dont Dior était très satisfait. D’après les comptes-rendus de réunions que j’ai eu de Kateryna Astafieva, la chef de projet de Smile sur Dior, et d’Alan Moreau, expert technique sur Drupal, la collaboration avec les personnes de chez Mazarine était souvent difficile. Ceux-ci avaient en effet une approche plus utilisation et “look-&-feel” et proposait des fonctionnalités et une expérience utilisateur sophistiquée voir complexe, sans vraiment tenir compte des contraintes et des bons usages sur Drupal, quand Smile aurait souvent privilégié la simplicité et la robustesse par le respect des bonnes pratiques de développement sur Drupal. Ce sont là deux philosophies qui s’affrontent, celle du designer et celle du technicien. Par ailleurs, la collaboration a pu être mise en difficulté dès le début car elle faisait suite à une compétition des deux protagonistes sur l’appel d’offre.

# 2 Spécifications

## 2.1 Allure globale de Dior Mag

Avant tout, les spécifications définissent les différentes pages que propose le site.

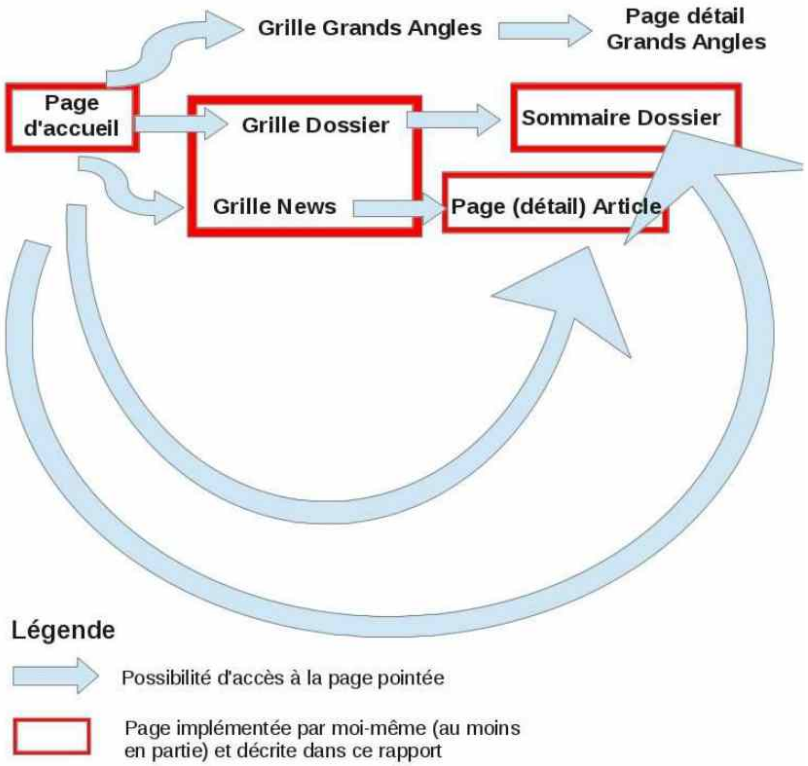



**Figure 2.1 – Schéma simplifié des interactions entre les pages de Dior Mag**

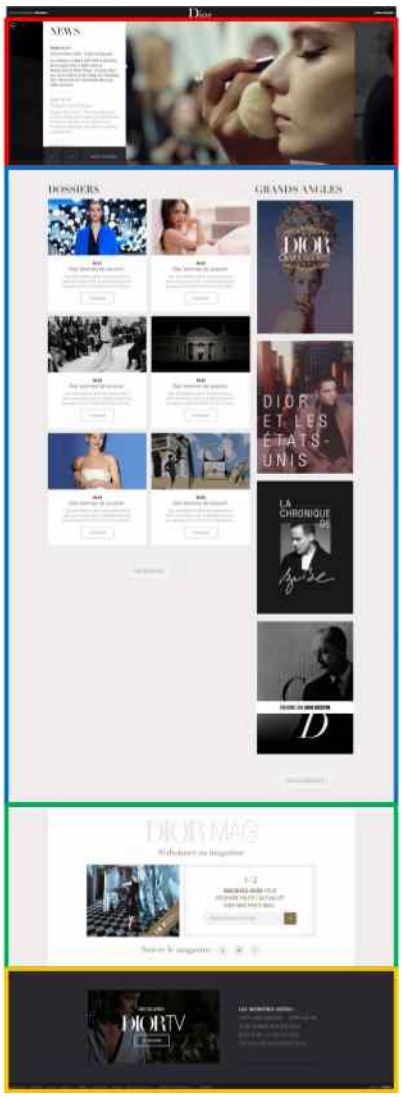
Comme le montre la figure 2.1, le site s'articule autour de trois éléments : Article, Dossier et Grands Angles. Ceux-ci sont en effet les types de contenu du site. Un type de contenu est l'entité principale définie par tout système de gestion de contenu. Nous en parlerons plus en détail dans la partie 3.

Les pages Détail Article, Sommaire Dossier et Détail Grands Angles sont les principales pages de description des contenus. Elles affichent les champs propres liés à ces entités : champs textes de toutes sortes, images ou vidéos, selon des styles définis par les spécifications graphiques et implémentés au montage.

Les pages “Grille”, quelqu'elles soient, listent les pages décrites ci-dessus. La Grille Grands Angles listent les pages Grands Angles dans l'ordre antéchronologique. Les Grilles Dossier et News listent respectivement les pages Sommaire Dossier et Article, en affichant titre, image, date et résumé (champ “Chapo”)

### 2.2 Page d'Accueil

**Figure 2.2 – Page d'accueil**




La page d'Accueil de Dior Mag est très dynamique et comporte beaucoup de contenus. Elle est visible ci-contre dans sa version pour des tailles d'écran supérieur à 1024px, ce qui corespond à des PCs. Un affichage spécifique est parfois prévu pour des écrans de plus de 1366px. Enfin un dernier affichage pour tablette est prévu (écrans entre 767px et 1024px, généralement les tablettes). Il n'y a pas réellement de version dédiée au mobile, puisqu'une application native était initialement prévue. A noter que tous les templates présentés dans la suite sont ceux des grands formats.

La page est ici découpée en 4 zones, mises en valeur par des cadres couleur. En rouge, c'est la partie qui prend la plus grande partie de l'écran au chargement de la page : le Slider de News. On voit un encadré sur fond blanc sur la gauche qui liste titres et résumés d'articles. L'article sélectionné dans cette liste affiche son visuel (image ou vidéo) dans la partie gauche qui est en fait un caroussel. Un bouton “+ de News” permet d'accéder à la page Grille News.

La zone bleue est composée de deux zones bien distinctes : la Zone dossier et la Zone Grands Angles. S'affichent respectivement dans ces zones les dossiers et les grands angles classés par ordre antéchronologique dont la case “Display on Homepage” est cochée (Cf. les spécifications des types de contenu en annexes 2.2 et 2.3). La zone Grands

Angles est limitée à 3 éléments : si cette limite est atteinte, le bouton “+ de Grands Angles” qui permet d'accéder à la zone Grands Angles s'affiche. La zone Dossier affiche systématiquement le bouton “+ de Dossiers”. A noter la présence d'une option “Display at top” dans les spécification du type de contenu Dossier (Cf annexe 2.2). Si elle est cochée, le dossier fait fi de l'ordre antéchronologique et remonte en tête de liste.

**Figure 2.3 – Elément de la zone Dossier**


Les types de contenu Dossier et Grands Angles possèdent chacun un champs nommés respectivement “Label +1” et “Label New” (Cf Annexes 2.2 et 2.3). Ceux-ci permettent d'afficher un petit encadré rouge indiquant “+1” ou “NEW” sur le dossier ou le grand angle remonté en page d'accueil.

La zone en vert est la zone de Newsletter. L'utilisateur peut y inscrire ses coordonnées pour recevoir la newsletter, choisir un réseau social pour suivre la page Facebook, le fil Twitter ou ajouter Dior Mag à ses cercles Google+.

Enfin, la zone jaune propose une prévisualisation de vidéos de la webTV de Dior – DiorTV – ainsi qu'une redirection vers celle-ci.

### 2.3 Types de contenu, page associée et page “Grille” associée

| Nom | Définition |
|---|---|
| **Article** | Contenu spécifique créé pour la gestion des articles (affichage dans le slider des articles de la page d’accueil et sur la page Grille News) |
| **Grand Angle** | Contenu spécifique créé pour la gestion des grands angles (sur la page Grille Grands Angles et dans la colonne Grands Angles sur la page d’accueil) |
| **Dossier** | Contenu spécifique créé pour la gestion des dossiers (focus) (sur les pages Grille Dossier, Sommaire Dossier et dans la zone Dossier de la page d’accueil) |

**Le détail des spécifications des types de contenu est disponible en annexe.**

### 2.3.1 Les Articles

En tant que site d'actualité, il est tout à fait normal que le premier type de contenu créé soit le type de contenu Article. Celui-ci rassemble donc des champs tels que Titre, Chapo, Date de publication, Body1, Body2, Images, etc,. Les catégories Thèmes et Univers se basent sur le système de taxonomie propre à Drupal, décrit dans la partie 3. Il permet notamment de créer des listes de vocabulaires pour attribuer des "termes", c'est-à-dire attribuer des mots-clés au type de contenu. Ces deux listes de catégorie vont notamment permettre le mécanisme de tri de la page Grille News.


**Figure 2.4 – Exemple d'une page article**

Les articles ont donc une page dédiée ("Page article" ou "Page détail article") où il est possible de voir l'intégralité de l'article et des contenus qui lui sont associés (images, vidéos). Celle-ci est accessible :

- Directement depuis la page d'accueil qui remonte les articles les plus récents
- Depuis la page dite "Grille News" qui remonte les articles dans l'ordre antéchronologique et propose un mécanisme de filtres décrit plus loin (dans la suite on se réfèrera indifféremment à News ou à Article qui définissent la même entité).

L'édition de cette page doit être la plus personnalisable possible. Les styles doivent être applicables ou non, la largeur des contenu décidée à la volée, et les contenus déplacés à la volée comme des widgets en drag-&-drop.

Dans la partie haute de la colonne de droite de l'article de la figure 2.3, on aperçoit une image avec un numéro au dessus ("1/9") : c'est le Slideshow de la page article (Cf Annexe 2.1), un champs qui permet de contribuer plusieurs images puis de les faire défiler sur la page. Il est également possible d'ouvrir une visionneuse telle que celle de la figure 2.4.

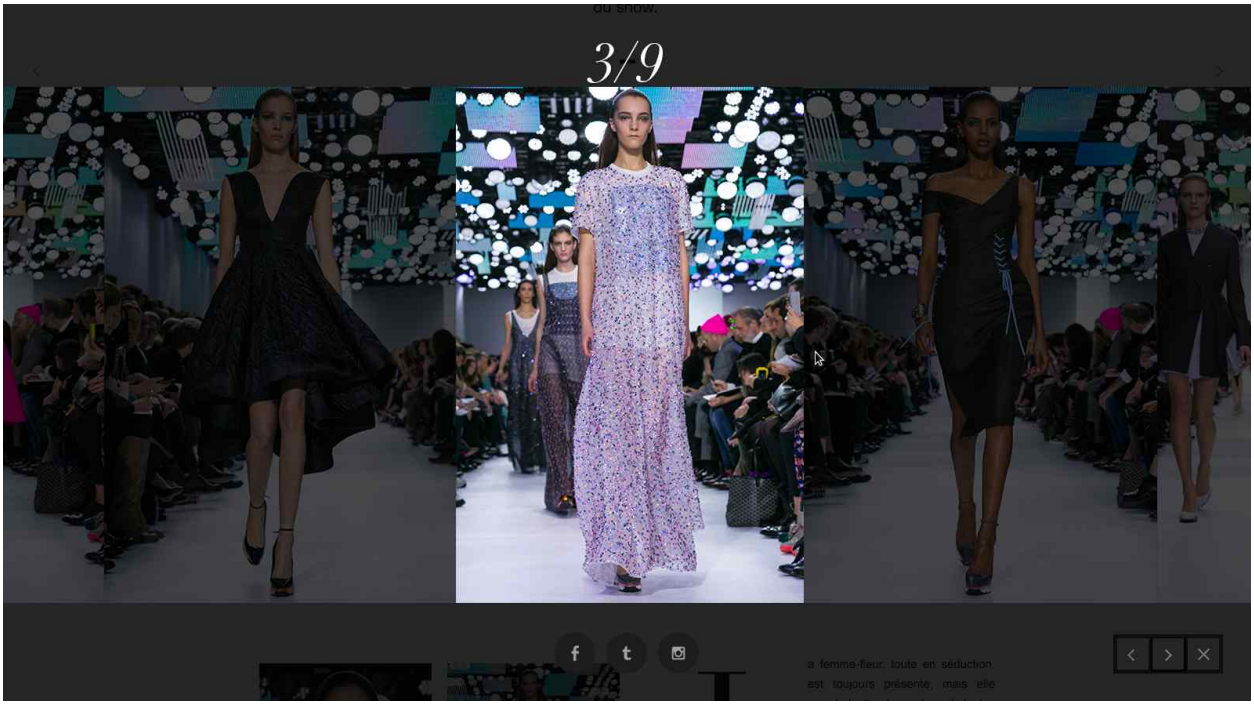



**Figure 2.5 – Visionneuse**

Au bas de la figure 2.4, on voit des boutons de partage de réseaux sociaux. Ceux-ci doivent permettre à l'utilisateur de se connecter à son compte et de partager sur ses réseaux en ouvrant une fenêtre du type de la figure 2.5.


**Figure 2.6 – Fenêtre de partage d'un article sur Facebook**

Dior souhaitait que lorsqu'un article était partagé à partir de la visionneuse, c'est l'image alors visionnée par l'utilisateur (l'image 3/9 dans l'exemple de la figure 2.4) qui soit l'image d'illustration et de prévisualisation de l'article dans le partage sur le réseau social.

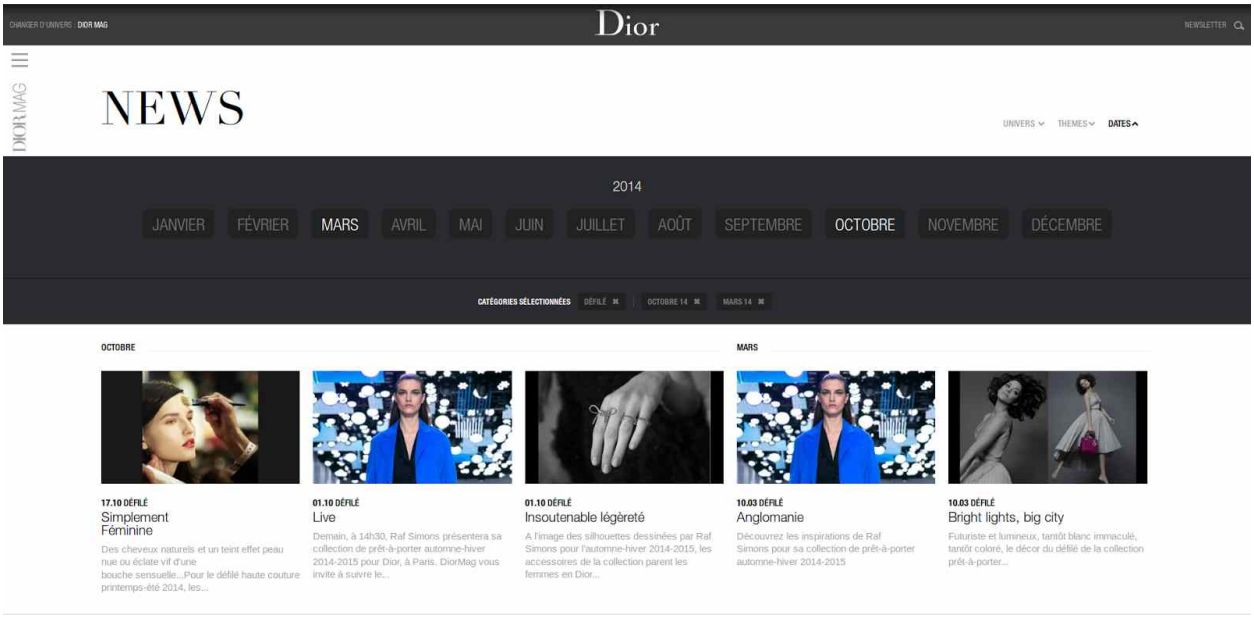



**Figure 2.7 – Page Grille News avec zone de filtre par date**

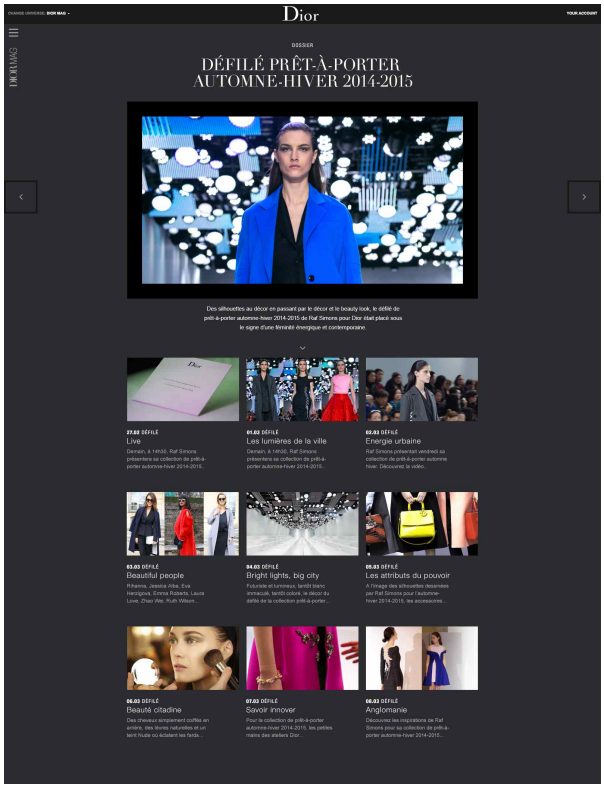
La page **Grille News** regroupe les 20 articles les plus récents, et propose un mécanisme de filtre en se basant sur les termes des catégories Thèmes et Univers, et en se basant aussi sur les dates de publication (filtre par mois et année). La conditions **ET** s'applique entre chaque catégorie. La condition **OU** s'applique à l'intérieur de chaque catégorie.

Par exemple, si je sélectionne “Septembre 2014”, tous les articles publiés durant cette période s'affichent, et seulement ces articles là. Si je sélectionne en plus “mars 2013”, tous les articles publiés durant l'une **ou** l'autre de ces périodes s'affiche. Comme le tri se fait systématiquement par ordre antéchronologique, les articles publiés en septembre seront remontés en haut de page, suivis des articles publiés en mars. En sélectionnant la catégorie “Défilé” dans Thèmes, on affiche les articles liés à cette catégorie **et** publiés en septembre 2014 **ou** en mars 2013. S'il y a plus de 20 articles à retourner, un bouton “+ de News” s'affiche. Un clic sur ce bouton permet d'ajouter les 20 articles suivants au bas des articles déjà affichés, de manière dynamique. Par défaut, au chargement de la page aucune catégorie n'est sélectionnée et tous les articles sont affichés (dans la limite des 20 articles, avec le même mécanisme lié au bouton “+ de News”).

Comme le montre la figure 2.7, les catégories et les dates sont sélectionnables au clic dans une interface prédéfinie. Une fois sélectionnée, la catégorie ou la date est mise en surbrillance et ajoutée à la liste des catégorie sélectionnées juste en dessous. La sélection de la date se fait par une liste des mois de l'année, et l'année elle-même se change grâce à un pager (flèches gauche et droite).

### 2.3.2 Les Dossiers

**Figure 2.8 – Exemple de page Sommaire Dossier**




Pour grouper les Articles et les mettre en valeur par thématique, le type de contenu Dossier présente :

- un titre
- une image (et une vidéo optionnelle)
- un chapo
- la possibilité d'être lié à un nombre infini d'articles via un champ présent dans le type de contenu Article (Cf. champ *Linked Folder* en annexe).

Les pages Sommaire Dossier sont accessibles directement depuis la *Zone Dossier* de la page d'accueil ou par la page Grille Dossier. Comme on peut le voir sur la figure 2.4, elles affichent le titre, l'illustration (image ou vidéo) et le résumé du dossier, avant de présenter la liste des articles qui lui sont liés en présentant là-aussi titre, date, image et résumé de l'article, ainsi qu'un terme de Thèmes lié à l'article s'il y en a un.

Pour permettre de naviguer facilement d'un dossier à l'autre, un "pager" est mis en place. Il est représenté par une flèche à gauche et à droite de la page. Il permet de naviguer à travers les dossiers par ordre chronologique.

La page Grille Dossier est identique à la page Grille News, hormis le fait que seul la catégorie Univers et le champs Date permettent de filtrer les Dossier. Après tout, le type de contenu Dossier ne comporte pas de champs Thèmes.

A noter une particularité du type de contenu Dossier : dans la quasi-totalité des cas, la date de publication prise en compte pour un dossier n'est pas celle du champs Date défini par le type de contenu Dossier. En pratique, Dior préfère que celle-ci soit définie par la date de publication la plus récente parmi les dates de publication des articles liés au dossier. Ceci a un impact sur la

page d'Accueil (dans la zone Dossier), sur la page Sommaire Dossier, la page Grille Dossier et la navigation entre les dossiers via le pager précédemment décrit.

#### 2.3.3 Les Grands Angles

Le type de contenu Grands Angles présente Titre, Chapo et Image pour présenter un commentaire de cette dernière en grand format et en bonne définition sur la “Page détail Grands angles”. Ils sont accessible depuis la zone Grands Angles de la page d’accueil ou depuis la Grille Grands Angles.

Le type de contenu Grands Angles est peu abordé dans ce rapport. En effet, bien que j’ai moi-même mis en place la Grille Grands Angles, elle présente un intérêt technique limité par rapport aux Grille News et Grille Dossier car sans aucun mécanisme de filtre. La “page détail Grands Angles” quant à elle était la dernière page à construire lorsque le projet fut arrêté.

### 2.4 Navigateurs cibles

Les navigateurs Desktop cibles sont les suivants :

- Internet Explorer (IE 8, 9, 10,11)
- Firefox (minimum 23)
- Chrome (minimum 28)
- Safari 6

Les navigateurs web mobiles et tablettes (responsive design) :

- IOS 6 et 7 avec Safari mobile
- Android 4.1 - 4.4 avec Android Browser et Chrome

### 2.5 Liste des acteurs utilisateurs

| Acteur | Description / rôles |
|---|---|
| **Internaute** | Un utilisateur du site |
| **Contributeur** | Une personne ayant un compte avec des droits spécifiques lui permettant de créer et modifier des contenus et médias, publier des contenus. |
| **Administrateur du site** | Une personne ayant un compte avec des droits spécifiques lui permettant d'administrer son site avec des droits restreints sur un certain nombre de fonctionnalités, cet utilisateur a uniquement les droits sur les contenus de son site. L’utilisateur a la possibilité de créer des termes de taxonomie et la possibilité de créer des utilisateurs. |
| **Administrateur global** | Une personne ayant un compte avec des droits spécifiques lui permettant d'administrer tout le site |

### 2.6 Champs communs

Tous les contenus disposent, de manière native :

- d'une ***Date de création*** au format DD/MM/YYYY HH :MM :SS
- d'une ***Date de mise à jour*** au format DD/MM/YYYY HH :MM :SS
- d'une liste de **langue** (10 langues) dans le back office : Chinois, Allemand, Espagnol, Français (langue par défaut), Italien, Japonais, Russe, Coréen, Anglais (UK), Brésilien
- 18 langues sur le front office avec les préfixes suivants :
  - Chinois : http://www.diormag.com/zh_cn/<page>
  - Taïwanais : zh_tw
  - Hong-Kong Chinois : zh_hk
  - Hong-Kong Anglais : en_hk
  - Allemand : de_de
  - Espagnol : es_es
  - Amérique latine : es_sam
  - Francais : fr_fr
  - Belge français : fr_be
  - Belge anglais : en_be
  - Italien : it_it
  - Japonais : ja_jp
  - Russe : ru_ru
  - Coréen : ko_kr
  - Anglais (UK) : en_gb
  - USA (Amérique du Nord) : en_us
  - Anglais International : en_int
  - Brésilien : pt_br
- d'un champ auteur (complété automatiquement en fonction de l'identifiant du créateur de contenu)

La gestion du multilingue se fait en revanche sur les seules 10 langues du back-office. Un contributeur pourra donc créer et éditer un contenu en Chinois. Ce contenu sera ensuite accessible à www.diormag.com/zh_hk/contenu1, www.diormag.com/zh_cn/contenu1 ou encore www.diormag.com/zh_tw/contenu1. L'adresse est personnalisée pour l'utilisateur selon sa propre localisation.

## 3 Drupal et les CMS

### 3.1 Bref historique sur le Web

En 1991, Tim Berners-Lee travaillait au Centre Européen pour la Recherche Nucléaire (CERN) où il est relié au réseau interne et à l'ARPANET et travaille avec des chercheurs originaires de toute l'Europe qui partagent leur activité entre le site de Genève et leurs pays d'origine. Il imagine alors un moyen de faciliter le partage d'informations entre les physiciens du CERN, c'est l'invention du World Wide Web et de ses 3 piliers :

- Les URL : **protocol://user:password@host:port/path?query#fragment**
- Le protocole HTTP (HyperText Transfer Protocol)
- Le langage de formatage : HTML (HyperText Markup Language)

La technologie connaît immédiatement un franc succès dans le milieu universitaire d'abord, puis les premiers business se forment autour d'elle. Des efforts sont faits pour améliorer l'expérience utilisateur côté front, notamment grâce à Netscape, l'entreprise à l'origine du célèbre navigateur et de la fondation Mozilla. Dans un premier temps, le développement du HTML est très rapide et la version 4 sort en 1997 (le HTML 5 est en cours de spécification depuis 2008 par le W3C - le World Wide Web Consortium, l'organisation de standardisation des technologies du web fondée par Tim Berners-Lee). Ensuite apparaissent les feuilles de style en cascade (CSS - Cascading Style Sheets) qui sont à la forme ce que HTML est au contenu, et définissent les styles des éléments selon la descendance et des règles de priorités assez simple. Apparaît également le Javascript, qui permet la manipulation des éléments du DOM (la structure de la page web) donc l'interaction avec les formulaires et une ergonomie plus interactive, et contient l'objet XMLHttpRequest à l'origine de l'AJAX et du Web 2.0 de la seconde partie des années 2000.

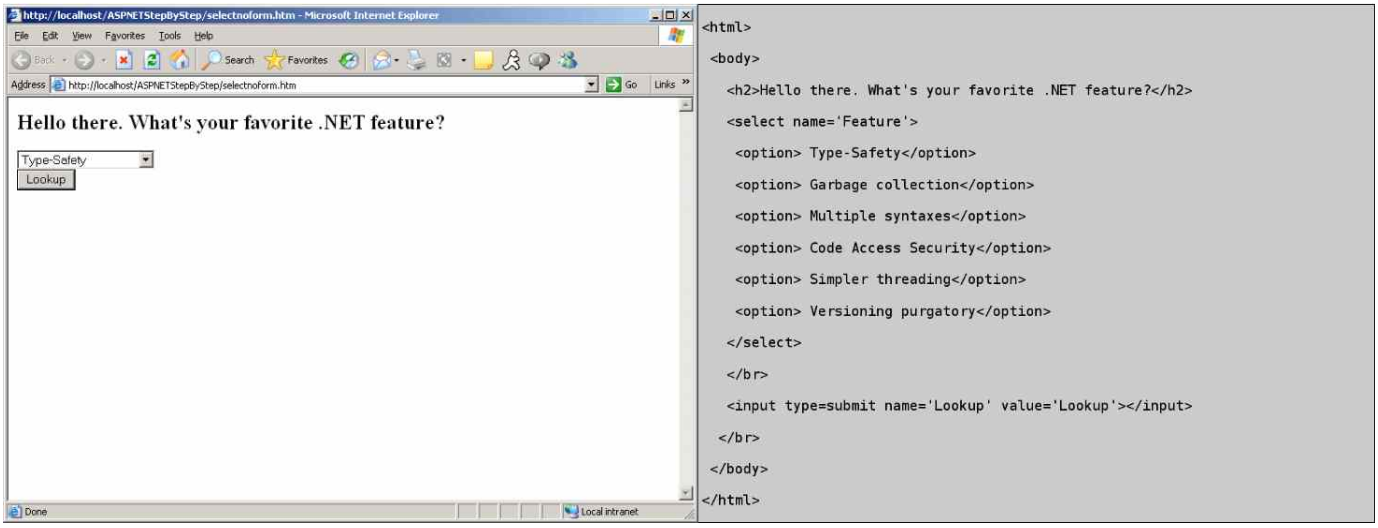



**Figure 3.1 - Exemple de page web et du fichier source HTML associé**

Parallèlement, les expérimentations des premières heures du web pour créer des pages dynamiques menèrent à la popularisation de plusieurs langages et paradigmes de programmation: Java, très tôt apprécié des entreprises pour sa maturité et sa rigueur, PHP, Python et bien d'autres. Ils génèrent dynamiquement les pages web à partir des :

- Interactions utilisateur : requêtes HTTP (visite d'une nouvelle page ou requête AJAX)
- Données stockées dans le SGBD (Sytème de Gestion de Base de Données) comme MySQL, Oracle Database ou plus récemment dans les basées de données orientés document comme MongoDB.

C'est donc là que se situe "l'intelligence" du Web dans sa forme brute, pourvu que l'application et le "métier" soient bien pensés pour fournir aux internautes des services véritablement utiles, facilitateurs, automatisés et "intuitifs".
**(3)**

### 3.2 Environnement LAMP

L'acronyme LAMP désigne un ensemble de logiciels libres qui forment ensemble l'environnement de serveur Web le plus utilisé au monde, parfois considéré comme l'élément clé de l'explosion du World Wide Web dans les années 1995 - 2005.

- Linux (GNU/Linux) : système d'exploitation libre créé par Richard Stallman et Linus Torvald au début des années 90, inspiré du système UNIX. La principale distribution Linux, Debian, est réputée pour sa stabilité, représente 58.5% des parts de marché des serveurs LAMP en octobre 2013 **(16)** et est à la base des distributions célèbres comme Ubuntu pour le desktop et les servers, Kali Linux (anciennement Backtrack) pour les tests sécurité et le hack ou récemment SteamOS pour le jeu en ligne.
- Apache : serveur HTTP le plus populaire au monde avec plus de 182 millions de sites actifs en avril 2014, soit 52% du marché **(17)**
- MySQL : système de gestion de base de données relationnelles SGBDR, aujourd'hui distribué par Oracle après l'OPA réussie sur Sun Microsystems. Ce rachat a posé des problèmes aux législateurs antitrust - notamment à la Commission européenne qui auraient finalement cédé à la pression de l'industrie et de l'administration américaine - puisqu'elle mettait Oracle dans une position de force en contrôlant 2 des 3 SGBD les plus populaires du marché **(18)**. Le créateur de MySQL a forké la version 5.1 de MySQL pour créer MariaDB.
  **(19)**
- PHP : langage de script inspiré de C et Perl, construit par la communauté web pour le web et en a fait un des langages les plus populaires du monde par sa facilité d'apprentissage. Cette croissance organique est un des principaux reproches des détracteurs de PHP : la programmation objet n'a été introduite qu'à partir de la version 4 ; le nommage et l'ordre des paramètres des fonctions n'est pas toujours cohérent ; etc. Il diffère en cela de philosophies très rigoureuses comme Java. En janvier 2013, PHP propulserait 39% des sites web. **(20)**

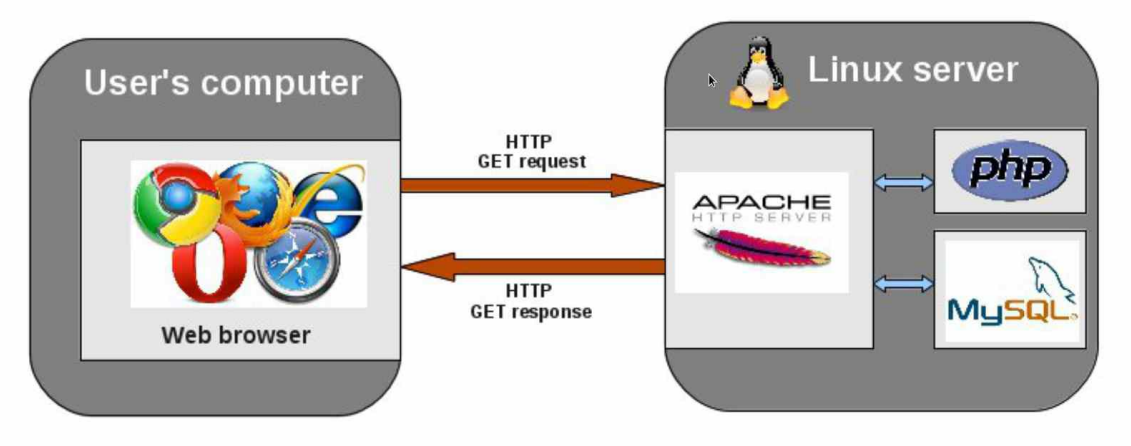



**Figure 3.2 - Interaction client - serveur LAMP**

### 3.3 Les CMS

Le CMS est un outil d'industrialisation et d'administration du site web. Il doit permettre à chacun, développeur ou non, de créer un site web fonctionnel et personnalisé assez intuitivement en proposant une interface d'administration structurante et organisée. Ce besoin s'est fait sentir dès le début du web quand des entreprises comme Amazon ont investis des millions de dollars pour développer leurs propres CMS, qui exigeaient tout de même aux contributeurs de contenu de structurer avec du HTML et, en fait, d'être très "débrouillards". **(4) (5)**

**NB1 :** Wordpress est le CMS le plus populaire au monde, propulse 25% des sites web **(6) (7)** et permet de créer un site en proposant également un hébergement gratuit à l'URL **<nom_du_site>.wordpress.com**
**NB2 :** le back-office est la partie d'un système informatique qui n'est pas accessible aux utilisateurs finaux ou aux clients, en opposition aux applications de front-office. Un exemple trivial serait de comparer le front-office à un journal ou un magazine distribués à X lecteurs, et le back-office aux locaux de la rédaction de ce journal (machines à écrire, maquettes, etc).

Tous les CMS partagent un certain nombre de propriétés et de fonctionnalités :

- séparation des opérations de gestion de la forme et du contenu : la structure HTML et le style sont appliqués sur tous les éléments d'une catégorie de contenu, les contributions se font donc sans impact ni régressions.
- contribution dans un éditeur WYSIWYG (What You See Is What You Get), les champs de formulaire bien connus qui proposent des options de formatage du texte comme dans logiciel de bureautique
- un workflow, ou chaîne de publication, pour gérer la mise en ligne des contenus
- les contenus sont organisés
- gestion des utilisateurs hiérarchisés en leur attribuant des rôles et des permissions. Exemple :
    - le rôle "visiteur anonyme" a la permission de lecture de certaines pages
    - le "visiteur authentifié" à ces mêmes permissions + écriture de commentaires
    - le "contributeur" a ces mêmes droits + création/édition de certains types de contenus
    - le "validateur" a les permissions du "visiteur authentifié" + publication des contenus
    - l'administrateur a toutes les permissions

- gestion du multilingue
- un respect (très) relatif du patron de conception MVC **(8)**

Et d'autres encore, le but n'étant pas ici de dresser une liste exhaustive mais de repérer quelques points clés représentatifs.

**(9)**

### 3.4 De l'utilité du développeur et des frameworks

Mais si tout est est éditable à partir du back-office, alors, n'importe qui peut le faire, et, dans ce cas, où est la difficulté technique? C'est d'ailleurs l'un des principaux arguments des défenseurs du CMS dans la bataille qui agite le web depuis plusieurs années : CMS vs Frameworks, les tenants des premiers prétendant pouvoir tout faire facilement et rapidement, leurs opposants affirmant être les seuls à avoir toutes les clés. En réalité, quelque soit l'outil choisi, des compétences en programmation sont toujours nécessaires pour être sûr de pouvoir réaliser toutes les fonctionnalités possibles.

Le choix entre CMS et framework est donc un éternel débat, qui doit être faits selon les particularités du projet : pour un site simple qui doit être déployé rapidement, sans grande contrainte en terme de fonctionnalités complexes voire "exotiques", les CMS simples seront privilégiés d'où l'immense popularité de Wordpress ; à l'inverse, une application web avec une partie métier moins standard et un focus sur l'expérience utilisateur et le design back-office sera généralement basé sur des frameworks qui ne font pas d'hypothèses sur l'utilisation mais sont des boîtes à outils complètes orientées développeur. A noter l'émergence des CMF, des librairies permettant de construire rapidement un back-office de type CMS à partir de son framework favori.

**(10)(11)**

### 3.5 Drupal

En 2009, la Maison Blanche a choisi d'adopter Drupal pour la gestion de son site whitehouse.gov. Cette mesure a fait couler beaucoup d'encre numérique à l'époque et fut vécu comme un signal fort par les autres administrations américaines, les entreprises et la communauté Drupal. **(12)**

Comme le présente sa documentation officielle, Drupal essaye de concilier la simplicité ET la flexibilité en proposant des outils prêts à l'emploi et une interface de programmation riche et pensée pour être personnalisable. Il prétend être à la croisée du CMS et du framework en proposant le meilleur des deux mondes. **(13)**

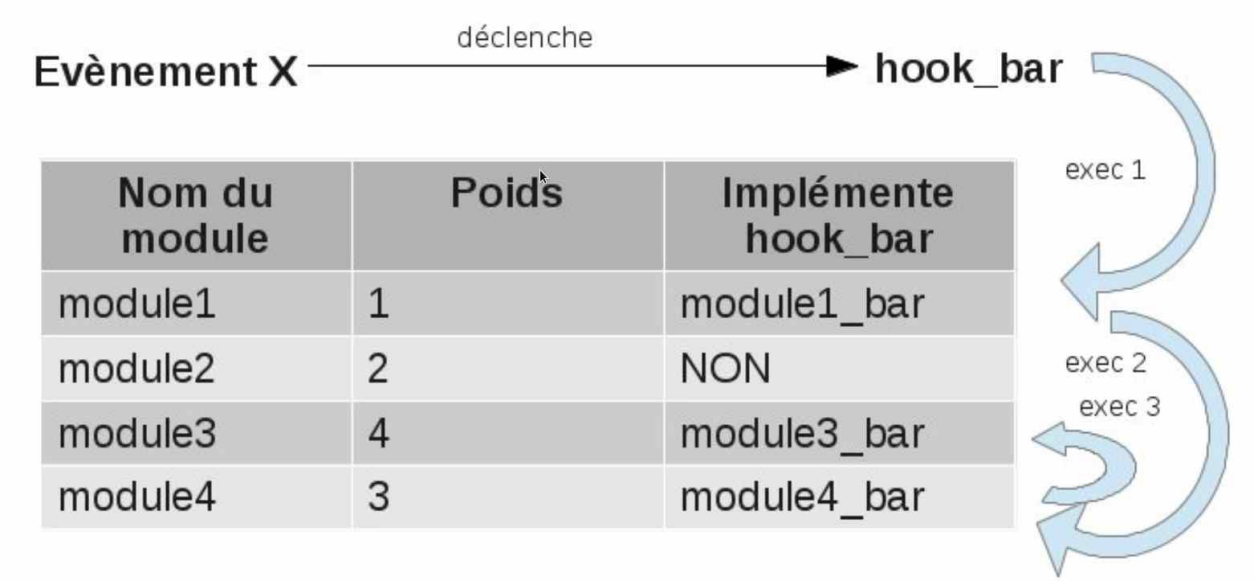
#### 3.5.1 Modularité

Drupal s'organise autour d'une installation de base plutôt simple et sans beaucoup plus de fonctionnalités qu'une plateforme de blog classique, sinon qu'elle fournit déjà de nombreux outils et une architecture capable d'absorber des projets complexes et d'une envergure certaine via l'ajout de **modules**. Les modules sont organisés en 3 catégories bien distinguées dans l'arborescence d'un projet Drupal, qui sont :

- **core** : les modules fournis avec les fichiers d'installation, présents dans tout projet Drupal.

- **contrib** : un module contrib n'est ajouté par un développeur/administrateur que lorsque celui-ci répond à un besoin clairement identifié de son site (mais probablement commun à de nombreux sites puisque mis à disposition par la communauté). Au fur et à mesure des versions de Drupal, plusieurs modules contrib (ou communautaire) ont été ajoutés dans le core Drupal, car identifiés comme standards ou quasi-standards. Cette standardisation et l'appropriation quasi-unanime de ces modules les a amenés à être largement utilisés, testés et éprouvés jusqu'à devenir très stables.
- **custom** : les modules spécifiques au projet, créés pour les besoins métiers par les développeurs du site, généralement trop spécifiques pour être utiles pour d'autres projet donc très rarement mis à disposition de la communauté. Le passage d'un module custom à un module contrib nécessite l'identification d'une réponse à un problème général grâce à ce module, et le cas échéant, un effort supplémentaire de la part des développeurs pour rendre le module en question aussi générique et réutilisable que possible.

Le système de modules de Drupal se base sur le concept des **hooks**. Un hook est une fonction PHP dont la convention de nommage est de de la forme "foo_bar" où "foo" est le nom du module qui implémente le hook et "bar" le nom du hook proprement dit. Lorsque Drupal autorise et déclenche l'intervention d'un hook, il déclenche successivement toutes les implémentations de ce hook dans les modules dans l'ordre des poids attribués à chaque module dans la table "system" de la base de données. Les hooks sont disponibles dans le core Drupal, et d'autres peuvent être créés par des modules (contrib ou custom).




| Nom du module | Poids | Implémente hook_bar |
|---|---|---|
| module1 | 1 | module1_bar |
| module2 | 2 | NON |
| module3 | 4 | module3_bar |
| module4 | 3 | module4_bar |

**Figure 3.3 - Ordonnancement d'exécution des implémentation du hook "hook_bar"**

### 3.5.2 Gestion des contenus dans Drupal

Au contraire de la plupart des autres CMS, Drupal ne gère pas ses contenus de manière hiérarchique, ce qui est l'une des sources de critiques de celui-ci.
La principale entité de description du contenu dans Drupal est le **type de contenu** ("content type"). La description et la configuration du type de contenu (qui est généralement définie en back-office) permet de regrouper des **champs** ("fields") de type variés : texte, liste de sélection, image, référence à une autre entité de contenu, etc. Un contenu créé à partir d'un type de contenu s'appelle un **node**. Un parallèle avec la programmation orientée objet serait de voir le type de contenu comme la classe, les nodes comme les objets et les champs comme les attributs et les méthodes.
Drupal possède un système de classification du contenu : la **taxonomie.** Celui-ci permet de définir des **vocabulaires** et d'y ajouter des **termes de taxonomie.** Chaque vocabulaire peut être attaché à un (ou plusieurs) type(s) de contenu, afin de catégoriser, tagger ou classifier les nodes de toutes les manières envisageables.
Citons également les **commentaires,** de petits bouts de contenu que les utilisateurs peuvent soumettre et attacher à un node en particulier. Les **menus** sont une simple collection de liens décrits de manière hiérarchique.

### 3.5.3 Templating et theming dans Drupal

Par défaut dans Drupal, une page web est divisée en **régions** qui sont classiquement header, footer, main, left sidebar, etc. Chaque région contient des **blocs,** des regroupements d'information qui s'apparentent à ce que d'autres CMS appellent "widgets", mais de manière plus généralisé, tout ce qui n'est pas contenu principal, fil d'arianne ou menus est généralement un bloc .De plus, le système de **vues** (du module Views) est largement utilisé par la plupart des sites Drupal pour la flexibilité qu'il permet, la possibilité d'afficher selon un **contexte** (présence d'un contenu en particulier sur la page, visite d'un type de page, etc) certains contenu dans le format choisi (généralement une page ou un bloc, mais également flux RSS et d'autres). Le cas d'utilisation le plus connu de Views est la création dynamique de listes de contenu mises à jour, comme la remontée des 5 ou 10 dernières actualités sur la page d'accueil du site.

Bien que l'utilisation de ce système permette une meilleure intégration à Drupal et donc une administration enrichie depuis le back-office, il n'a pas été employé pour construire et structurer les pages du site DiorMag. En effet, non seulement la mise en place d'une telle structure est d'un intérêt limité et quelque peu chronophage dans un projet au forfait qui aurait possiblement dépassé les délais s'il n'avait pas été arrêté; mais il était fonctionnellement difficile, voire impossible, de mettre en place certaines fonctionnalités du site via ce système. Dans le projet Dior, l'utilisation intensive du moteur de template a été préférée(Drupal 7 se base sur PHPTemplate) :

- soit en surchargeant les templates des types de contenu, comme le type de contenu Dossier où il a suffi de créer le fichier node--folder.tpl.php pour que celui-ci définisse la nouvelle structure HTML du node. Ensuite, le hook **hook_preprocess_node**, executé

avant chaque affichage de node, peuple les variables du template (Cf. un exemple de template en annexe).

- soit en définissant directement un élément de menu via le **hook_menu.** Celui-ci définit un lien relatif et sa fonction de callback associée retournant le contenu de la page sous format HTML. Pour ce faire, la fonction de callback fait appel à un template prédéfini via le **hook_theme** qui associe un nom du theme au template, et via la fonction **theme(<nom_du_theme>, <array_de_variables>).** Ainsi l'affichage de la homepage se fait selon la structure définie ci-dessous :

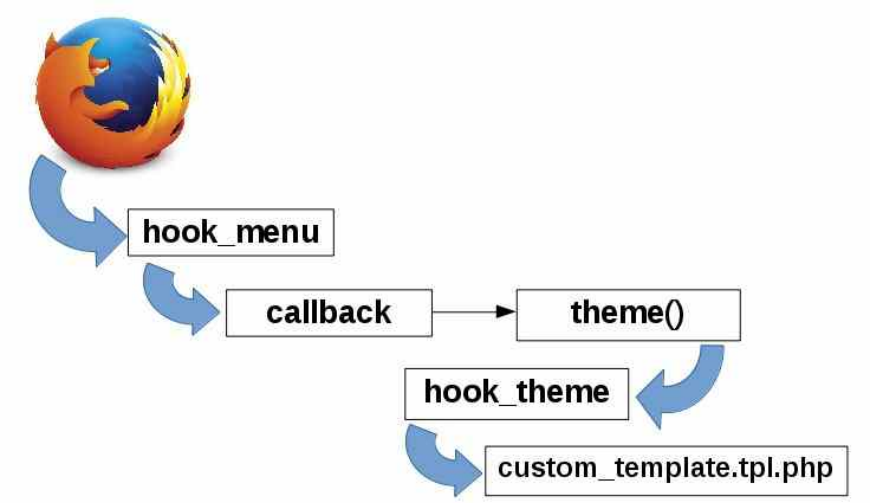



**Figure 3.4 : Structure de rendu utilisée pour DiorMag**

### 3.5.4 Remarques

Une particularité importante de Drupal par rapport aux autres CMS et frameworks, est que tout se passe en base de données : les contenus eux-mêmes bien évidemment, mais également les types de contenus qui les définissent, les vues, les blocs et tous les changements de configuration qui peuvent être faits dans le back-office sont stockés en base de données, ce qui fait de Drupal un CMS difficile à utiliser lorsque plusieurs développeurs travaillent en parallèle.. Il existe donc un module (communautaire) pour faire le lien entre base de données et code source : le module Features, largement utilisé par la communauté et l'équipe et dont la procédure est décrite dans l'article **(14)** :

1. Effectuer les changements voulus en back-office
2. Exporter la feature concernée par le back-office en téléchargeant *foo.zip*
3. Dézipper et écraser l'ancienne version du dossier *foo*
4. Commiter (via le gestionnaire de version Git pour le projet Dior)
5. Déployer le code sur le serveur d'intégration (ou de pré-production ou production)
6. Dans le back-office du serveur, sur l'interface Features, choisir l'option "revert"

7. Notifier les autres développeurs qu'ils doivent faire un "revert" sur leur propre machine pour appliquer les changements sur leur machine locale

C'est une procédure longue et fastidieuse, ce qui est frustrant quand on pense que la procédure sera souvent beaucoup plus simple (voire directe en faite) avec d'autres solutions. L'article **(14)** pointe d'autres défauts de Drupal, comme l'architecture peu élégante voire incompréhensible du fait du labyrinthe des appels de hooks aux arguments obscurs, de la surexploitation des tableaux associatifs, qui auraient en fait pu être des objets pour faire profiter à Drupal d'un peu de la puissance de la programmation objet.

A noter également que la procédure décrite ci-dessus peut être plus rapide via l'utilisation du shell Drupal : l'outil **drush,** qui permet de faire de nombreuses opérations d'administration par ligne de commande. Ainsi les opérations 2/3 et 6 peuvent être remplacées respectivement par les commandes :

```
$ drush fu <nom_de_la_feature>
```

```
$ drush fr <nom_de_la_feature>
```

L'article **(14)** fait d'ailleurs référence aux problèmes de gestion de dépendance sous Drupal : dans la plupart des projets Drupal, et tous ceux que j'ai vu à Smile en dehors de DiorMag, il est nécessaire de versionner - c'est-à-dire de d'inclure dans le gestionnaire de version - le core Drupal, les modules communautaires et tous les thèmes rajoutés, contrairement à des projets en Ruby par exemple où seuls sont versionnés le code custom et des fichiers de config. Ce versionnement du core et des modules contrib offre plus de libertés au développeur qui, par facilité, va souvent choisir de modifier directement ces fichiers sources plutôt que de faire appel à un hook dédié. C'est une mauvaise pratique, qui pose notamment problème lors des mises à jours fournies par la communauté : il faut alors éviter de mettre à jour ces fichers, sauf pour les mises à jour de sécurité où il faut alors refaire tout l'historique des modifications en espérant qu'il n'y ait aucune régression.
La commande **drush make** palie à ce problème. Elle récupère les sources de ces projets en se basant sur un fichier au format *.make* qui lui sera versionné **(15)**. Pour DiorMag, c'est la procédure que nous avons mise en place, pour la première fois chez Smile, la commande **drush make** étant présente dans le script de livraison.

# 4 Implémentation

## 4.1 Autre pré-requis techniques

### 4.1.1 Environnement de développement

**LXC : L**inu**X C**ontainer, un environnement de virtualisation très pratique, alliant les avantages du *chroot* et des virtualisations totales : c'est le *chroot* avec l'isolation, la virtualisation sans émulation matérielle, qui se base directement sur la configuration matérielle réelle du PC. Ce système allie donc sécurité réseau et performances égales (ou presque) à celles d'un système natif. C'est au coeur de ces environnements que sont installés les environnement LAMP, et donc Drupal.
Un tel système, une fois déployé, est accessible directement depuis la machine hôte dans le système de fichiers à */lxc/<nom_de_la_machine>* ou par connection SSH à *root@<adresse_ip_de_la_lxc>*. La première méthode permet d'accéder au code source custom à */lxc/dior/home/smile/dior-mag,* et permet alors d'utiliser des éditeurs de texte en interface graphique depuis le système hôte pour éditer les fichiers source dans la LXC. La seconde permet d'administrer directement le système virtualisé sans conscience du système hôte, pour administrer Drupal via **drush** par exemple, versionner via **git**, faire des requêtes SQL ou configurer le serveur Apache, le tout en ligne de commande.
Le code source, présent dans le répertoire */home* est servi par le serveur via un lien symbolique qui le pointe depuis le répertoire */var/www*, pour des questions de droits d'accès et de sécurité, et pour ne versionner que le répertoire */home.*
Pour accéder au site de la machine de développement, il suffit d'accéder à l'adresse IP de la LXC, ou à son alias défini dans le fichier */etc/hosts*.
De plus, afin de faciliter la mise en place et l'échange de ces LXC entre les développeurs, afin que ceux-ci travaillent sur un environnement identique, Smile a mis en place des scripts de création, déploiement et suppression de LXC. Pour la création, on se base simplement sur une archive et on lui donne un nom :

```
$ sudo cdeploy dior dior-almor.tgz
```

Quelques secondes après le déploiement, on peut déjà s'y connecter via :

```
$ ssh dior.lxc
```

### 4.1.2 Montage

Avec le développement des éléments graphiques dans les pages web depuis les débuts, les feuilles de styles CSS et le langage Javascript ont été mis à forte contribution. Les premières présentent cependant plusieurs inconvénients qui le rendent à la longue fastidieux à développer

et difficile à maintenir “as is”. On citera notamment l’absence de variables et l’impossibilité de réutiliser du code, les préfixes propres à chaque navigateur (-moz, -webkit, -ie) ou encore le manque de respect de la hiérarchie du document HTML dans lequel est chargée la feuille de style. Pour pallier à cela ont donc été inventés les **préprocesseurs CSS**, les plus populaires étant **LESS** et **SASS**, qui permettent entre autres de définir variables et fonctions, de gérer la compatibilité “trans-navigateur” (on dit généralement “cross-browser”) automatiquement (le développeur n’a qu’à se conformer au standard W3C), et la hiérarchie HTML est respectée grâce au “nesting” des sélecteurs CSS.

Les monteurs Web utilisent également des bibliothèques comme jQuery pour faciliter le développement Javascript.

Un workflow classique de développement front-end est donc :

1. Compiler les fichiers .sass, .scss ou .less en .css
2. Concaténer les fichiers .css
3. Concaténer les .js
4. Minifier le fichier résultant de l’action précédente

Les fichiers résultant de l’opération (.js minifiés, .css) sont les fichiers fonctionnels du site, mais ce sont les fichiers sources (.js source, .scss) qui sont partagés les développeurs, qui devront répéter le workflow décrit ci-dessus. C’est pour automatiser ce processus qu’a été créé **Grunt**, un module de **NodeJS**, la célèbre solution qui introduisit le javascript comme un langage serveur, basé sur le moteur javascript V8 créé pour Chrome par Google. Une fois tout installé, il suffit d’une simple commande pour lancer le processus :

```
$ grunt server
```

Grunt s’établit alors comme un serveur sur la machine locale, et chaque fois qu’un de ces fichiers q’il gère (.scss, .js, .html, etc) est modifié, il le détecte et exécute le workflow prédéfini (semblable à celui décrit ci-dessus) en seulement quelques secondes.

### 4.1.3 Livraison

Quelque soit le projet, sa livraison, c’est-à-dire la copie de tous ses fichiers source et leur installation dans un environnement correctement configuré qui sert déjà les pages web aux utilisateurs autorisés, se fait généralement par un “diff” entre le code dans le gestionnaire de version et celui déjà présent sur le serveur cible. Comme l’utilisation de Git chez Smile était totalement nouvelle, les scripts de livraison n’avaient pas été fournis par la Direction Technique, c’est pourquoi ils nous fallait au départ du projet faire des copies directes de notre poste de travail vers le serveur en **scp.**
Finalement, Alan Moreau a mis en place l’outil *charon* à partir du framework python *Fabric*. En plus de copier les sources en “diff”, celui-ci créait le tag git. Le reste du script de livraison (qui

commence par lancer charon) se charge de “reverter” les features (cf. 1.b.iv), d’importer les fichiers de traduction et de vider les caches.

### 4.2 Types de fichiers

#### 4.2.1 Images

Pour certains champs de type Image dans les types de contenu décrits en annexe, on lit “Le bon format d’image média sera appelé sur le front office en fonction du device”. En fait, pour chaque image, le contributeur a la possibilité d’uploader différentes versions : la version principale, mais aussi les versions qui s’afficheront sur le site sur mobile, le site sur tablette, l’application mobile Android, l’application tablette Android, l’application mobile iOS et enfin l’application tablette iOS.

Pour ce faire nous avons créé un nouveau type de fichiers : Responsive Image. Les types de fichiers sont similaires aux types de contenu puisqu’ils semblent tous deux être des regroupements de champs. Les types de fichiers sont orientés gestion des fichiers, tandis que les types de contenu seront généralement affichés sous forme de pages web dédiées. Par défaut, Drupal fourni le champs fichier, mais nous avons privilégié un champs de *Sélection Media* via le module Media, qui propose à chaque attribution/contribution d’image la possibilité de parcourir la médiathèque et les fichiers précédemment contribués ou d’uploader une nouvelle image .

Enfin, comme Dior est une marque très soucieuse de son image, et pour faciliter la contribution, j’ai mis en place le module Manualcrop qui permet de recadrer les images à partir de Drupal.

#### 4.2.2 Vidéos

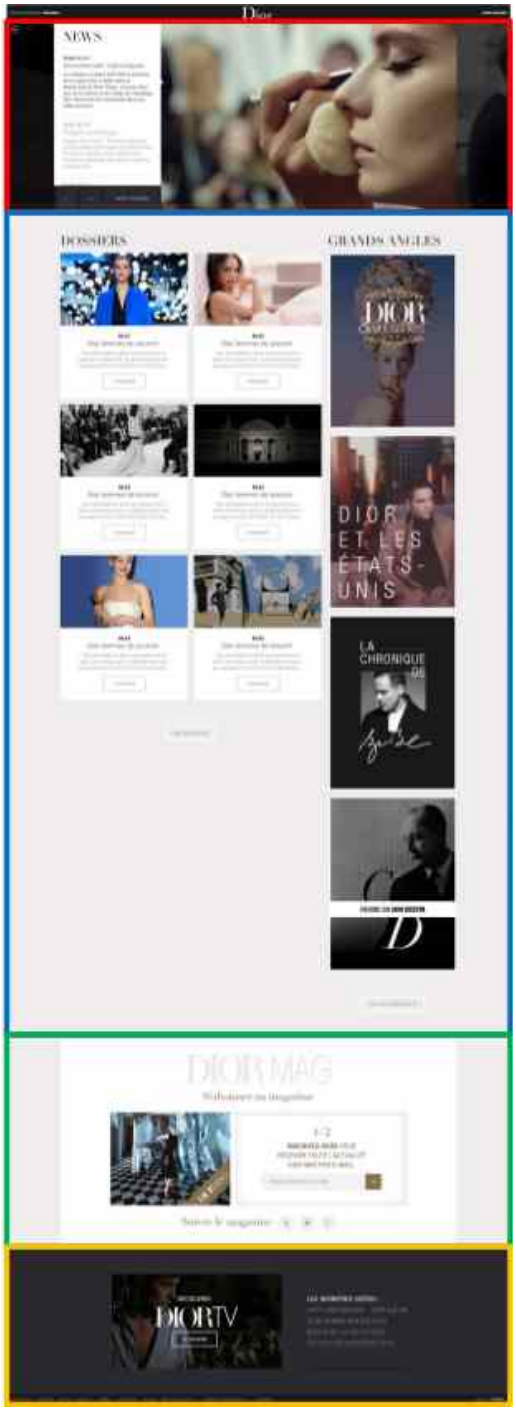
On trouvera 2 type de vidéos sur le site de Dior :

- Responsive video : un abus de langage dont l’origine est le type de fichiers Responsive Image précédemment défini. Ce type de fichier permet de contribuer localement les vidéos dans 3 formats : MP4, Webm et OGV. L’utilisation de ses 3 formats et de la balise HTML5 <video> qui donne la possibilité de définir plusieurs sources possibles assure que la vidéo sera visible dans tous les navigateurs (par exemple, Firefox ne supporte pas officiellement le format MP4). Pour les navigateurs plus anciens qui ne supportent pas HTML5 (Internet Explorer 8 en tête), nous nous sommes servi de la bibliothèque javascript Video.js **(23)** qui modifie la structure HTML pour revenir à des techniques plus traditionnelles d’embarquement des vidéos grâce à la balise <object>.
- MassMotion Video : une partie des vidéos, qui sont déjà utilisées par Dior.com et la version précédente de DiorMag, sont fournies par MassMotionMedia **(24)**, un hébergeur de contenus médias haute définition, haute disponibilité et multi-device. Nous avons donc mis en place un cron, c’est-à-dire une tâche planifiée chargée de lancer le webservice pour récupérer les nouvelles vidéos MassMotion de Dior.

**NB :** le terme Responsive Image, quant à lui, vient du terme de “responsive design”, ou plus précisément “responsive web design”, définit comme “une notion de conception de sites web qui regroupe différents principes et technologies dans laquelle un site est conçu pour offrir au visiteur une expérience de consultation optimale facilitant la lecture et la navigation. L'utilisateur peut ainsi consulter le même site web à travers une large gamme d'appareils (moniteurs d'ordinateur, smartphones, tablettes, TV, etc.) avec le même confort visuel et sans avoir recours au défilement horizontal ou au zoom avant/arrière sur les appareils tactiles notamment, manipulations qui peuvent parfois dégrader l'expérience utilisateur.” **(25)**

## 4.3 Page par page

**Figure 4.1 - Page d’accueil**




### 4.3.1 Page d’accueil

Pour rappel, la page d’accueil du site DiorMag est divisée en 4 catégories principales :

- le slider de News en haut en rouge
- le corps de la page avec la zone Dossiers et la zone Grands Angles, qui remontent les nodes de ces 2 types de contenu selon des règles déjà définies
- la zone de newsletter pour soumettre son email
- la zone DiorTV qui proposent des vidéos et redirigent vers le site DiorTV

Si le slider a été principalement mis en place par le monteur, et la newsletter et la zone DiorTV intégrée par Ludovic Bernard, j’ai été chargée de mettre en place les zones Dossier et Grands Angles, encadrées en bleu dans l’image ci-contre.

#### 4.3.1.1 Zone Grands Angles

La mise en place du second était très simple : elle impliquait la remontée des nodes de type Grands Angles avec leurs champs Cover grands angles (affiché par défaut), Titre et Chapo (affiché au “roll-over”, c’est-à-dire au survol de la souris), selon les règles suivantes :

- classés par ordre décroissant de date de publication
- Affiché uniquement si la case “Display on Homepage” a été cochée par l’utilisateur
- Affiché s'il n’y a pas plus de 3 résultats

Si trois résultats remontent , le bouton “Tous les Grands Angles” doit être affiché en bas de cette zone pour rediriger les utilisateurs vers la page Grands Angles.

```
SELECT title.title_field_value, chapo.field_chapo_value, img.field_image_fid
FROM node n
  INNER JOIN field_data_title_field AS title
    ON n.nid = title.entity_id
  INNER JOIN field_data_field_image AS img
    ON n.nid = img.entity_id
  INNER JOIN field_data_field_chapo AS chapo
    ON n.nid = chapo.entity_id AND title.language = img.language
  INNER JOIN field_data_field_publication_date AS date
    ON n.nid = date.entity_id
  INNER JOIN field_data_field_display_homepage AS checkbox
    ON n.nid = date.entity_id
WHERE checkbox.field_display_homepage_value = 1
  AND node.status = 1
  AND title.language = 'fr'
ORDER BY date.field_publication_date.value DESC, node.created DESC,
LIMIT 3;
```

La requête SQL ci-dessus serait fonctionnellement valable pour récupérer les contenus nécessaires pour la zone Grand Angle telle que décrit plus haut (pour des contenus en français). Cependant, outre que celle-ci est théoriquement vulnérable aux injections SQL, elle est tout de même assez difficile à lire pour une requête sensée être simple. C'est aussi pour cela que Drupal, comme tout CMS ou framework qui se respecte, propose une API pour interagir plus facilement avec les bases de données : db_select, db_insert, etc. qui retournent eux-mêmes des objets SelectQuery ou InsertQuery, et dont les méthodes associées correspondent à la définition du langage SQL : addField (select), range(offset, limit), condition(where), etc. Puisqu'il s'efforce de maintenir la compatibilité entre les SGBDR, MySQL et PostgreSQL notamment, Drupal n'implémente pas les fonctions propres à PostgreSQL (un SGBD beaucoup plus complet que MySQL).

Toutefois, comme l'application est ici très simple et ne concerne qu'un et un seul type de contenu, il était préférable d'utiliser une abstraction encore plus simple de Drupal pour accéder au contenu : la classe EntityFieldQuery.

"*This class allows finding entities based on entity properties (for example, node->changed), field values, and generic entity meta data (bundle, entity type, entity id, and revision ID).* ***It is not possible to query across multiple entity types.*** *For example, there is no facility to find published nodes written by users created in the last hour, as this would require querying both node->status and user->created." Documentation officielle* **(26).**

Ainsi le code correspondant à la même requête (sauf la sélection des champs du node, puisqu'EntityFieldQuery retourne le node dans son entier) :

```
global $language;

$query = new EntityFieldQuery();

$query->entityCondition('entity_type','node')
        ->entityCondition('bundle', 'panorama')
        ->propertyCondition('status', NODE_PUBLISHED)
        ->fieldLanguageCondition('title_field', $language->language)
        ->fieldOrderBy('field_publication_date','value','DESC')
        ->propertyOrderBy('created', 'DESC')
        ->addTag('node_access');

$query->fieldCondition('field_display_homepage','value', 1);
$query->range(0, 3);
$result = $query->execute();
```

Le code est ici bien plus compréhensible. De plus, la requête étant construite graduellement via des méthodes, et exécutée tout à la fin, on n'a pas besoin de respecter l'ordre de la syntaxe SQL. Cela permet la factorisation du code : une requête très similaire sera lancée pour la page Grille Grands Angles, avec pour seule différence les deux lignes en caractères gras ci-dessus. La partie au-dessus de ces lignes sera donc mise en commun, puis des structures conditionnelles (*if, else …*) se chargeront de personnaliser le code selon la page visitée.

#### 4.3.1.2 Zone Dossier

**Figure 4.2 – Zone dossier**


La mise en place de cette zone fut plus ardue. On voit sur l'image ci-contre (1) le titre de la zone, (2) liste des dossiers remontés, (3) le bouton "Tous les dossiers" qui redirige vers la page Grille Dossiers. Pour chaque élément, on distingue une zone image dans la partie haute, et une zone contenu dans la partie basse avec date, titre, chapo (tronqué) et un encadré contenant le nombre d'articles liés au dossier affiché.

Rappelons deux particularités qui, en plus du compte des articles liés au dossier, vont considérablement complexifier la requête :

- la date de publication affichée et prise en compte pour le tri des dossiers n'est pas celui indiqué dans le node lui-même. C'est en fait la date de publication la plus récente parmi les articles qui sont liés au dossier.
- la zone image affiche par défaut l'image "Dossier cover" définie dans le dossier. Au roll-over, on voit défiler les images "Grid image" (une seule image "Grid image" contribuée par article) de tous les articles liés au dossier.

**Etat de fait**
Au tout début du projet, l'expert technique Alan Moreau m'avait conseillé l'utilisation des EntityFieldQuery pour les zones Grands Angles ET Dossiers, me conseillant d'utiliser une seule requête pour chaque zone. C'était oublier que la documentation des EntityFieldQuery précise : *"It is not possible to query across multiple entity types"* **(26).** Pour la zone Dossiers, il faut en effet des informations provenant à la fois des nodes Dossiers et Articles, intimement liés d'un point de vue fonctionnel et dont les requêtes ne peuvent donc être séparées.

**Adaptation au vu des contraintes de l'industrialisation**
A la vue du travail que j'ai effectué pour la zone Dossier, Ludovic Bernard, l'autre développeur qui travaillait avec moi sur DiorMag, a quant à lui remarqué que si mon système était performant, il serait difficile à aborder pour les collaborateurs en TMA. Son choix aurait été d'effectuer une requête par Dossier à afficher.

**Proposition**
Il eût été intéressant de vérifier les performances d'un tel système dans un site, et surtout une page d'accueil, qui présentent déjà beaucoup de fonctionnalités et d'images, ce qui n'en fait pas un site internet des plus légers.

Presque comme un compromis entre la performance voulue par Alan Moreau, et la maintenabilité qu'aurait préférée Ludovic Bernard, j'ai choisi d'effectuer ce travail en 2 requêtes SQL : une première requête pour charger les Dossiers qui remplissent les conditions énoncées ci-dessus, avec le titre associée, le chapo, l'image "Dossier cover", les ID des articles associés avec leur nombre et la date de publication du plus récent d'entre eux ; et une seconde requête pour charger les images "Grid image" des articles liés, en se basant sur les résultats (ID) de la première requête.

**Implémentaion**

```
$lang = $language->language;

$query = db_select('field_data_field_linked_folder', 'link');

$query->innerJoin('node', 'folder_node', '
        folder_node.nid = link.field_linked_folder_target_id AND
        folder_node.status = 1
');
$query->innerJoin('node', 'article_node', '
        article_node.nid = link.entity_id AND
        article_node.status = 1
');
$query->innerJoin('field_data_field_etiquette', 'etiquette', '
        etiquette.entity_id = link.field_linked_folder_target_id AND
        etiquette.language = link.language
');
$query->innerJoin('field_data_field_publication_date','date', '
        date.entity_id = link.entity_id AND
```

```
        date.language = link.language
');
$query->innerJoin('field_data_field_actual_title','title', '
        title.entity_id = link.field_linked_folder_target_id AND
        title.language = link.language
');
$query->innerJoin('field_data_field_sticky','sticky','
        sticky.entity_id = link.field_linked_folder_target_id AND
        sticky.language = link.language
');
$query->innerJoin('field_data_field_image','img_cover', '
        img_cover.entity_id = link.field_linked_folder_target_id
');
$query->leftJoin('field_data_field_chapo','chapo', '
        chapo.entity_id = link.field_linked_folder_target_id AND
        chapo.language = link.language
');

$query->addField('title','field_actual_title_value','title');
$query->addField('etiquette','field_etiquette_value','etiquette');
$query->addField('chapo','field_chapo_value','chapo');
$query->addField('link','field_linked_folder_target_id','folder');
$query->addField('img_cover', 'field_image_fid', 'img_cover');

$query->condition('link.language', $lang);

$query->addExpression('COUNT(DISTINCT link.entity_id)','count'); /***/
$query->addExpression('MAX(date.field_publication_date_value)','publication_date'); /***/

$query->groupBy('link.field_linked_folder_target_id'); /***/

$query->innerJoin('field_data_field_display_homepage', 'display','
        display.entity_id = link.field_linked_folder_target_id AND
        display.language = link.language AND
        display.field_display_homepage_value = 1
');

$query->orderBy('sticky.field_sticky_value','DESC');
$query->orderBy('publication_date','DESC');
$query->orderBy('folder_node.created', 'DESC');

$query->havingCondition('count', 2, '>=');
```

Ci-dessus la première requête qui fait le lien entre articles et dossiers, en passant notamment par la table field_data_field_linked_folder qui correspond au champs “Linked folder” du type de contenu Article. Son schéma est le suivant :

```
mysql> desc field_data_field_linked_folder;
+-------------------------------+------------------+------+-----+---------+-------+
| Field                         | Type             | Null | Key | Default | Extra |
+-------------------------------+------------------+------+-----+---------+-------+
| entity_type                   | varchar(128)     | NO   | PRI |         |       |
```

```
| bundle                        | varchar(128)    | NO   | MUL |              |          |
| deleted                       | tinyint(4)      | NO   | PRI | 0            |          |
| entity_id                     | int(10) unsigned | NO   | PRI | NULL     |          |
| revision_id                   | int(10) unsigned | YES  | MUL | NULL     |          |
| language                      | varchar(32)     | NO   | PRI |      |          |
| delta                         | int(10) unsigned | NO   | PRI | NULL     |          |
| field_linked_folder_target_id | int(10) unsigned | NO   | MUL | NULL     |          |
+-------------------------------+------------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

Le propos de cette table est de faire correspondre 1 ou plusieurs (ou 0, mais il n'y pas d'entrée en base dans ce cas ) dossier(s) à chaque article : *entity_id* indique l'ID de l'entité à laquelle est lié le champs, c'est-à-dire l'article ; *field_linked_folder_target_id* est la valeur réelle du champs, donc l'ID du dossier indiqué par le contributeur. Le reste de la requête s'articule autour de ces 2 valeurs. Les éléments clés de la requête au-dessus sont marqués d'étoiles ***/***/*** :

- groupBy : tous les Dossiers ET tous les Articles sont à prendre en compte, mais ce sont les premiers qui regroupent les informations et doivent être affichés ici
- count : au sein de chaque regroupement, on compte le nombre d'articles (*entity_id*) différents.
- max : la date de publication du Dossier est celle du plus récent article qui lui est lié, donc la plus grande d'entre elles.

Les lignes indiquées en gras sont les lignes spécifiques à la page d'accueil. Là encore, le reste sera encapsulé dans une fonction de "préparation de requête" et réutilisé pour la page Grille Dossier et la zone Dossier de la page détail Article (à voir plus loin).

### 4.3.2 Grille news et grille dossier

La page Grille News liste tous les articles publiés par ordre antéchronologique, et propose de les filtrer selon 3 critères : catégorie Univers ; catégorie Thèmes ; date (mois et année).

**Figure 4.3 - Elément News de la Grille**

Notons la présence du mois de parution en haut du dernier article paru ce mois-là, comme la mention "Novembre" dans la figure 4.3. Si l'article a affiché à la suite a également été publié en novembre, la mention n'apparaitra pas. S'il a été publié avant novembre, le mois correspondant s'affichera. En raison des filtres et du bouton "+ de News" (décrit ci-dessous), on verra que ce point de changement oblige à une mécanique assez complexe pour le rendu de template.

**Bouton « + de News » : offset et rendu**

Le rôle du bouton “+ de News” est d’afficher les 20 résultats suivants en dessous des précédents sans rechargement de la page via une requête AJAX. On a donc un script Javascript qui a des états par défaut (des tableaux univers, themes et date tous vides, un offset à 0). A partir de ces états, l’évènement “click” sur le bouton “+ de News” déclenche une requête HTTP GET grâce à la méthode jQuery **getJson (27)** vers une URL définie dans le **hook_menu**. La fonction de callback retourne toutes les données formatées via la fonction **drupal_json_output**. Quand le javascript reçoit la réponse du serveur dans un format valide, il insère les éléments nécessaires et incrémente l’offset.

**Difficultés autour du bouton « + de News »**

La difficulté du rendu a principalement résidé dans le mois de la publication. Par exemple, si la dernière publication affichée a été publiée en août, et que le plus récent des articles obtenus via “+ de News” a également été publié en août, on doit ajouter cet article dans la même catégorie que les derniers déjà affichés. Au contraire, si le premier de ces articles a été publié en juillet, on ne se soucie pas de ce qui a été fait avant et on insère tout à la fin.

**Spécifications des filtres**

Les filtres de la page sont affichés sous la forme de 3 boutons en haut à gauche : “Univers”, Thèmes” et “Date”. Quand on veut filtrer par date et qu'on clique sur un mois, celui s’affiche dans la liste des catégories sélectionnées (le bas de l’encadré noir) et lance une requête AJAX pour restreindre les éléments affichés aux mois et aux catégories sélectionnés. Comme les mois déjà sélectionnés ne sont plus cliquables et sont affichés avec une police plus claire, il a fallu gérer le système du “pager” d’année : si je sélectionne “Septembre” en 2014, “Septembre” s’affiche en clair, je passe à 2013, “Septembre” s’affiche en sombre, je repasse en 2014 et “Septembre” s’affiche de nouveau en clair. La conservation des états dans le javascript était donc également utile pour cette fonctionnalité.

**Particularités des filtres**

La requête AJAX des filtres et des dates fonctionne de la même manière que pour le bouton “+ de News”, à ceci près que :

- l’évènement se déclenche sur les boutons de la zone filtre
- le nouvel élément de filtrage est ajouté au tableau correspondant du scrip Javascript. Il est identifié par :
    - “mois” et “année” pour la date via “data-month” et “data-year” en attribut de l’élément HTML (le format *data-<nom_custom>* est valide W3C pour créer des attributs HTML personnalisés
    - “tid” via “data-tid” pour les catégories Univers et Thèmes, qui correspond à l’ID du terme de taxonomie attribué par Drupal.
- l’offset est réinitialisé
- le nouveau contenu n’est pas ajouté à la suite de l’ancien mais remplace ce dernier

Les règles de recherche croisées via les filtres sont les suivantes :

- condition ET entre filtres de catégories différentes : par exemple si on a sélectionné le thème “Défile”, l’univers “Mr Dior” et le mois de mai 2014, on affiche seulement les articles parus au mois de mai 2014 et appartenant à ces 2 catégories, sans dérogation possible à une seule des conditions.
- condition OU entre filtres de même catégorie.

La page Grille Dossier, également accessible depuis la page d’accueil, est similaire à la page Grille News, à ceci près que seuls les filtres “univers” et “dates” sont présents.

#### 4.3.3 Page article

**Principe général**

Pour les pages Articles, qui sont le principal contenu du site Dior Mag, Dior souhaitait avoir la capacité de personnaliser les pages au maximum. Il a donc été imaginé une solution basée sur le module Panelizer. Celui-ci permet en effet de configurer des “layouts”, c’est-à-dire des affichages pré-configurés pour un type de contenu. On peut donc définir plusieurs layouts pour chaque type de contenu, et choisir celui qui convient le mieux, si ce n’est pas celui par défaut. Une fois le layout choisi pour un node donné, on peut effectuer certaines modification : le layout est alors considéré comme “custom”.Pour Dior, il a été imaginé 5 layouts de base, dont certains sont visibles en annexe. Si le corps de l’article fait toujours la même largeur, soit 960px (pour les résolutions au dessus de 1024, sinon le responsive design s’applique), les éléments sont organisés via des colonnes de tailles et de dispositions variées. Pierrick Cadet, expert technique, a créé un module *custom* pour permettre l’édition de l’article et l’ajout des éléments directement depuis le *front* de l’article, dans les colonnes prédéfinies. La taille de ces colonnes est également personnalisable, auquel cas le layout est considéré “custom”.

**Le module Panelizer**

Dans Panelizer, et donc dans la page Article de DiorMag, les éléments du layout et des colonnes sont appelés des *panes*. Ce sont principalement (et c’est ce qui nous intéresse ici) des champs du node, quoique Panelizer propose également d’autres contenus. Ainsi, après la contribution du contenu, l’auteur de l’article doit mettre celui-ci en forme en ajoutant au layout une/des image(s) du champs à sélection multiple Images, ou en ajoutant Body1, Slideshow, Exergue, etc (cf les spécifications et la définition des types de contenu en 2.c.v. En revanche, pour l’application des styles, on ne peut passer par le theming classique (décrit en 1.b.iii) comme on l’a fait pour les autres pages, puisque c’est Panelizer qui définit l’affichage.

**Personnalisation générale de la page**

Heureusement, Panelizer définit **template_preprocess_panelizer_view_mode**. Les fonctions de type **template_preprocess_<nom_theme>** sont comme des fonctions **hook_preprocess_<nom>** : elles peuvent être implémentées plusieurs fois dans des modules différent et appelées les unes à la suite des autres. Le **<nom_theme>** dans les fonctions **template_preprocess** fait lui référence à un thème définit dans le **hook_theme**. Ici on se sert de **template_preprocess_panelizer_view_mode** pour afficher pour tous les articles sa date de publication, un terme de taxonomie Thèmes associé et son titre.

**Personnalisation localisée « pane par pane »**

Le module Panels, sur lequel se base Panelizer, définit également la fonction **template_preprocess_panels_pane.** Comme la plupart des fonctions de preprocess dans Drupal, celle-ci possède un argument *$variables,* qui est en fait un tableau (*array*) peuplé par des variables à afficher dynamiquement dans le template correspondant, des classes CSS, des options de rendu, etc. Celle qui nous occupe ici contient notamment un objet *pane* dont l'attribut *subtype* indique clairement le nom du champs dont il s'agit. On obtient donc la structure suivante :

```
function dm_article_preprocess_panels_pane(&$vars) {
  if($vars['pane']->subtype == 'node:field_chapo') {
    //process and populates $vars
  }
  elseif($vars['pane']->subtype == 'node:field_body_1') {
    //...
  }
  elseif ($vars['pane']->subtype == 'node:field_slideshow') {
    //..
  }
  //etc...
}
```

Par ce biais, on va principalement attribuer des classes CSS aux éléments simples (Quote, exergue). C'est également par là qu'on va personnaliser le partage des réseaux sociaux : ceux-ci peuvent également être ajoutés sous forme de *pane* (comme sur la figure 3.7), ou sont de toutes façon automatiquement attribués dans la visionneuse d'image.

**Personnalisation du partage de réseaux sociaux**

La difficulté résidait à personnaliser le partage de réseaux : lorsque l'utilisateur clique sur le lien *Facebook* de la visionneuse, il partage sur son Mur l'article qu'il est en train de lire avec le titre, le chapo comme description et l'image qu'il est en train de visionner. On a donc mis en place au préalable les balises métas selon la recommandations Open Graph mise en place par Facebook, mais devenu un standard ~~officiel~~ officieux dans le monde des réseaux sociaux. On trouve donc en en-tête des balises de la forme :

```
<meta property="og:title" content="Premières impressions"></meta>
```

Les scripts de partage sont eux spécifiques à chaque plateforme. Pour Facebook, l'URL du bouton de partage est **http://facebook.com/sharer.php?u=<URL>.** L'URL fournie est celle du contenu à partager sur le réseau, à savoir ici l'URL de l'article. Le script *sharer.php*, à chaque fois qu'il est appelé, lance une requête GET sur *<URL>* pour obtenir le contenu désiré, grâce aux balises *meta* telle que celle ci-dessus si elles existent, ou en tentant de *parser* directement le contenu de la page sinon. Il était possible avant avril 2014 de fournir l'argument **p[images][0]** comme argument après **u** pour définir l'image à afficher dans le partage, mais cette option a été

retirée du script pour des raisons de stabilité et d'efficacité de la plateforme Facebook **(28).** Les autres réseaux accepte peut-être un tel argument, mais on a préféré mettre en place une solution générale indépendante des choix faits par ces plateformes.

Pour pallier à ce problème et proposer une solution stable et généraliste, j'ai créé un argument **sni** (pour "Social Network Image")en paramètres de requête HTTP de la page Article. Plutôt que de fournir l'image dans l'URL du bouton sous la forme **http://facebook.com/sharer.php?u=<URL>&p[images][0]=<URL_img>**, le nom de l'image est en paramètre de l'URL elle-même en argument, donc sous la forme **http://facebook.com/sharer.php?u=<URL>?sni=<nom_image>.** A chaque visite de la page, Drupal appelle **hook_html_head_alter,** un hook dédié à l'altération des paramètres de l'en-tête <head>. Les images de la page sont listés et apparaissent avec la balise suivante dans l'ordre défini dans **hook_html_head_alter.**

```
<meta property="og:image" content="http://diormag.clients.smile.fr/sites/default/files/52ea7e392c408_4.jpg">
```

Il importe donc, dans le cas où **sni** est défini, de s'assurer que l'image correspondante est bien en tête de toutes les balise méta *og:image.* Quand le script vient faire une requête sur l'URL avec l'argument donné, il trouve la bonne image partagée par l'utilisateur.

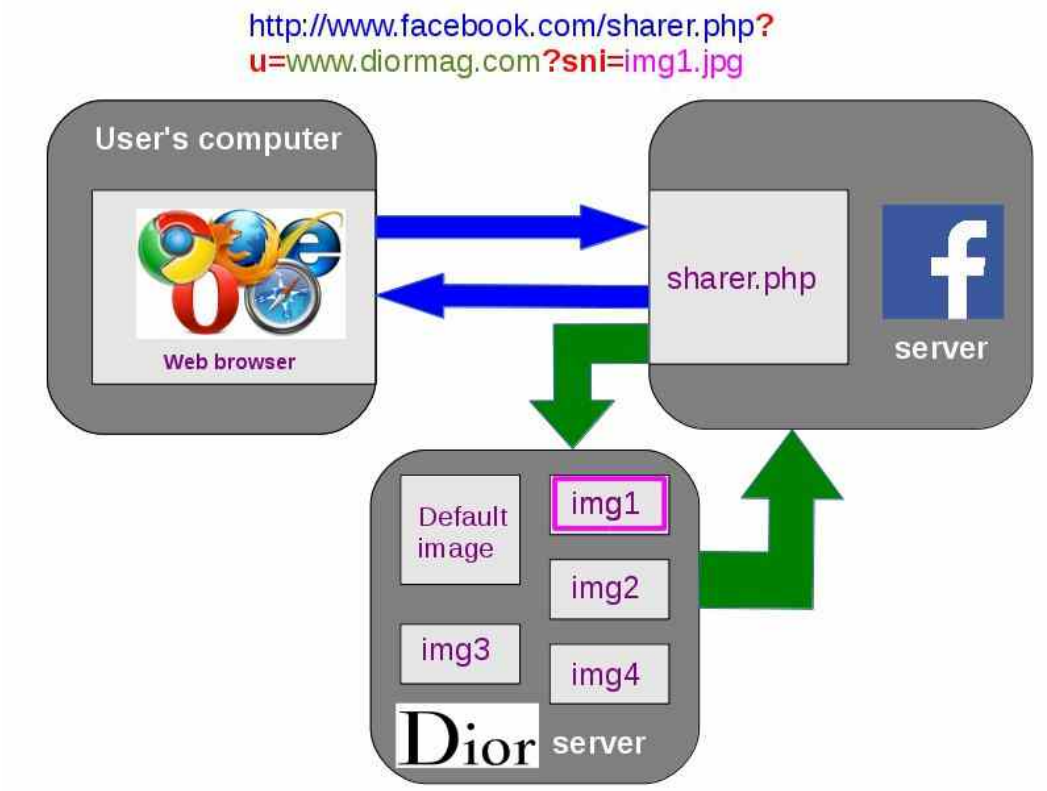



**Figure 4.4 – Schéma de personnalisation à la volée de l'image de partage**

Enfin, dernier élément de la page qui s'affiche tout en bas de celle-ci, une zone Dossier. Ici s'affiche le titre du dossier lié à l'article dont le poids est le plus fort, configurable en back-office

en “glisser-déposer” (*drag-&-drop*). S’affiche ici le titre du Dossier et la liste de tous les articles liés à ce dossier sauf l’article courant, sous le même format que celui trouvé pour la page Grille News.

### 4.3.4 Sommaire dossier

La page Sommaire Dossier affiche le titre du dossier ; le visuel *Dossier cover,* ou *Dossier video cover* si ce dernier est défini ; le chapo du dossier ; et finalement la liste des articles liés au dossier.

On a ici 2 requêtes SQL. Une des requêtes sert à récupérer les articles liés au dossier, d’une manière assez similaire à ce qui a été fait précédemment, le fonctionnel étant ici très simple.

L’autre requête concerne les flèches visibles de chaque côté de la page qui vers un autre dossier, l’une vers le précédent dossier dans l’ordre chronologique, l’autre le dossier suivant, en se basant encore une fois sur la date de publication du plus récent article lié au dossier. Là encore on réutiliser le code préalablement, et le pager de dossier réutilise ici la majeure partie de la requête de la zone dossier de la page d’accueil.

Au chargement de la page, si l’écran fait au moins 1024px (valable donc pour les tablettes à l’horizontal), tout l’écran doit être occupé par le haut de la page, c’est à dire par le titre, le visuel, le chapo et la petite flèche que l’on peut distinguer au-dessous. Au clic sur celle-ci, la page est scrollée (“défilée”) pour que le haut de la page se cale exactement sous la flèche. Si le scroll était bien implémenté, le redimensionnement des éléments du haut de page ne l’était pas. Or, l’intégratrice Anna Demidas était déjà en dépassement de budget à ce moment du projet et n’a pu s’en occuper.
J’ai donc dû m’assurer moi-même via CSS et jQuery que le visuel, qu’il soit de type Image, Responsive Video ou MassMotion Video, occupe bien toute la place disponible entre le titre et le chapo sur toutes les résolutions (1024, 1366) sur tous les devices (desktop, iPad, tablette Android) et sur tous les navigateurs (Firefox, Chrome, Safari, Internet Explorer 8 à IE11). Etant donné mes compétences en technologies front et les contraintes imposées ici, ce fut difficile à résoudre et jamais complètement terminé.
En effet, si tout semblait marcher sur mon PC en local en redimensionnant les fenêtres des navigateurs, le rendu sur iPad lui n’était pas correct pour les vidéos MassMotion, qui je le rappelle étaient fournies sous format Flash, format non reconnu par Apple. Il aurait fallu débugger via un Mac, mais le projet ayant été arrêté alors que je commençais seulement à résoudre les problèmes sur desktop et les navigateurs classiques, il n’était pas intéressant de poursuivre le projet aussi loin après.

## Conclusion

Bien qu'il soit très frustrant voire incompréhensible qu'un projet comme Dior Mag soit arrêté aussi brutalement qu'il le fût, travailler à la mise de ce site fut une expérience enrichissante tant d'un point de vue technique que managérial.

Drupal est un très bon outil pour répondre à toutes sortes de besoins, car malgré ses points faibles qu'on aura parfois noté au fil de ce compte-rendu, il présente des atouts forts, qui sont une faible courbe d'apprentissage (bien que plus importante que les autres CMS), une large communauté pour augmenter Drupal et proposer de l'aide, et une modularité désirée et conceptualisée pour faciliter le développement additionnel. Cet outil était donc relativement bien adapté pour Dior Mag qui, tout en étant un simple site d'actualités à priori gérable par tout CMS, présente également des fonctionnalités moins communes, comme la personnalisation des pages Article via Panelizer ou la possibilités d'éditer les images en lignes grâce au module Manualcrop. C'est pour ce genre de fonctionnalités qu'il est intéressant de disposer de la puissance d'un outil comme Drupal, dont la communauté aura toujours une partie de solution à proposer.

Puisque c'est ma première expérience dans une unité de production avec des attentes fortes de la part de ma hiérarchie, ce projet est évidemment assez unique voire fondateur dans ma conception de la gestion de projet. S'insérer dans des projets et dans une équipe au fonctionnement bien rodé n'est jamais chose aisée, d'autant moins quand on a peu d'expérience de l'entreprise. Pour ma part, je suis ravi d'avoir pu développer mes compétences techniques et mon savoir-être dans la BU Paris WebCMS. Les gens qui y travaillent m'ont beaucoup appris, le plus souvent avec le sourire, et pour cela je les remercie très chaleureusement.

# Annexes

```
<section class="l-content-top">
  <div class="inner">
    <div class="block-media-sommaire">
      <div class="zone-top">
        <p class="date-rubric"><strong><?php print $date?></strong> <?php print t('folder')?></p>
        <h2 class="title"><?php print $title_h2?></h2>
      </div>
      <div class="zone-media">
        <div class="media-container">
          <?php print $media?>
        </div>
      </div>
      <div class="zone-description">
        <div class="text">
          <?php print $chapo?>
        </div>
        <span class="btn-go js-btn-go"><span>go to list</span></span>
      </div>
    </div>
  </div>
</section>

<section class="l-content-main">
  <div class="inner">
    <!--section role="main" class="column" id="content"-->
    <div class="large-1col">
      <div class="list-sommaire">
        <div class="container-grid-960">
          <?php print $articles?>
        </div>
        <?php if (isset($sommaire_prev_next)):?>
          <?php print $sommaire_prev_next?>
        <?php endif?>
      </div>
    </div>
    <!--/section-->
  </div>
</section>
```

**Annexe 1.1- node—folder.tpl.php**

## Annexe 2.1 - Type de contenu Article

| Nom | Obligatoire | Commentaire |
|---|---|---|
| **System title** | oui | Ligne de texte : champ système pour le back-office et l'indexation |
| **Title** | oui | Éditeur de texte WYSIWYG avec la possibilité d'appliquer des styles : <i>, <br>, , <b>, espaces insécables et interlignage augmenté |

| | | |
|---|---|---|
| **Language** | oui | Liste déroulante comportant la liste des langues du site. Français est pré choisi par défaut. |
| **Publication date** | oui | Champ date |
| **Catégorie Univers** | | Liste déroulante de termes du vocabulaire de taxonomie Univers |
| **Catégorie Thèmes** | | Liste déroulante de termes du vocabulaire de taxonomie Thèmes. C'est un attribut à afficher sur la page de détail d'un article |
| **Grid Image** | oui | Référence image media, permettant d'uploader une image. |
| **Images** | | Référence image média multiple permettant d'uploader plusieurs images à afficher sur la page détail Article.<br><br>Le bon format d'image média sera appelé sur le front office en fonction du device |
| **Slider Image** | | Référence image media, permettant d'uploader une image à afficher dans le Slider de la page d'accueil.<br><br>Slider Image est prioritaire sur Slider Video. |
| **Slider Video** | | Référence video média à afficher dans le slider de la page d'accueil + prévoir une image « Poster ». |
| **Vidéo** | | Référence vidéo média multiple. Vidéos à afficher sur l'article détail |
| **Slideshow** | | Champ référence image multiple permettant d'uploader plusieurs images à afficher dans le slider sur la page Article |
| **Chapô** | oui | Éditeur de texte WYSIWYG permettant de saisir le chapô d'un article. Le champ autorise les espaces insécables. Les mêmes règles de limitation de nombre de caractères que sur Dior.com seront appliquées. |
| **Body 1** | | Éditeur de texte WYSIWYG permettant d'appliquer les styles prédéfinis par la charte graphique |
| **Body 2** | | Éditeur de texte WYSIWYG permettant d'appliquer les styles prédéfinis par la charte graphique |
| **Quote text** | | Éditeur de texte WYSIWYG permettant de mettre le texte important avec le style prédéfini par la charte graphique |
| **Exergue** | | Ligne de texte permettant de saisir le texte avec le style |

| | | |
|---|---|---|
| | | Exergue prédéfini |
| **Linked folder** | | Champ d'auto complétion multiple permettant d'associer un article à un/plusieurs dossiers |
| **Background color** | | Champ de sélection de couleur du fond : blanc/gris |

## Annexe 2.2 - Type de contenu Dossier

| Nom | Obligatoire | Commentaire |
|---|---|---|
| **System title** | oui | Ligne de texte |
| **Title** | oui | Éditeur de texte WYSIWYG avec la possibilité d'appliquer des styles : <i>, <br>, , <b>, espaces insécables et interlignage augmenté |
| **Language** | oui | Liste déroulante comportant la liste des langues du site. Français est pré choisi par défaut. |
| **Publication date** | oui | Champ date |
| **Catégorie Univers** | | Liste déroulante de termes du vocabulaire de taxonomie Univers |
| **Dossier cover** | oui | Référence image media, permettant d'uploader une image. Le bon format d'image média sera appelé sur le front office en fonction du device.<br><br>Cette image s'affiche sur le Sommaire Dossier, dans un push dossier et sur la page d'accueil et Grille Dossiers. |
| **Dossier video cover** | | Référence video media. Vidéo à afficher sur le Sommaire Dossier. Si ce champ n'est pas rempli, le champ Dossier cover sera affiché. |
| **Chapô** | oui | Éditeur de texte WYSIWYG. Le champs autorise les espaces insécables. |
| **Label +1** | | Case à cocher permettant d'afficher l'étiquette « +1» sur le dossier |
| **Display on Homepage** | | Case à coher, pré-cochée par défaut, permettant d'afficher un dossier sur la page d'accueil |
| | | Case à cocher permettant de remonter un dossier en première position sur la page d'accueil et la page Grille |

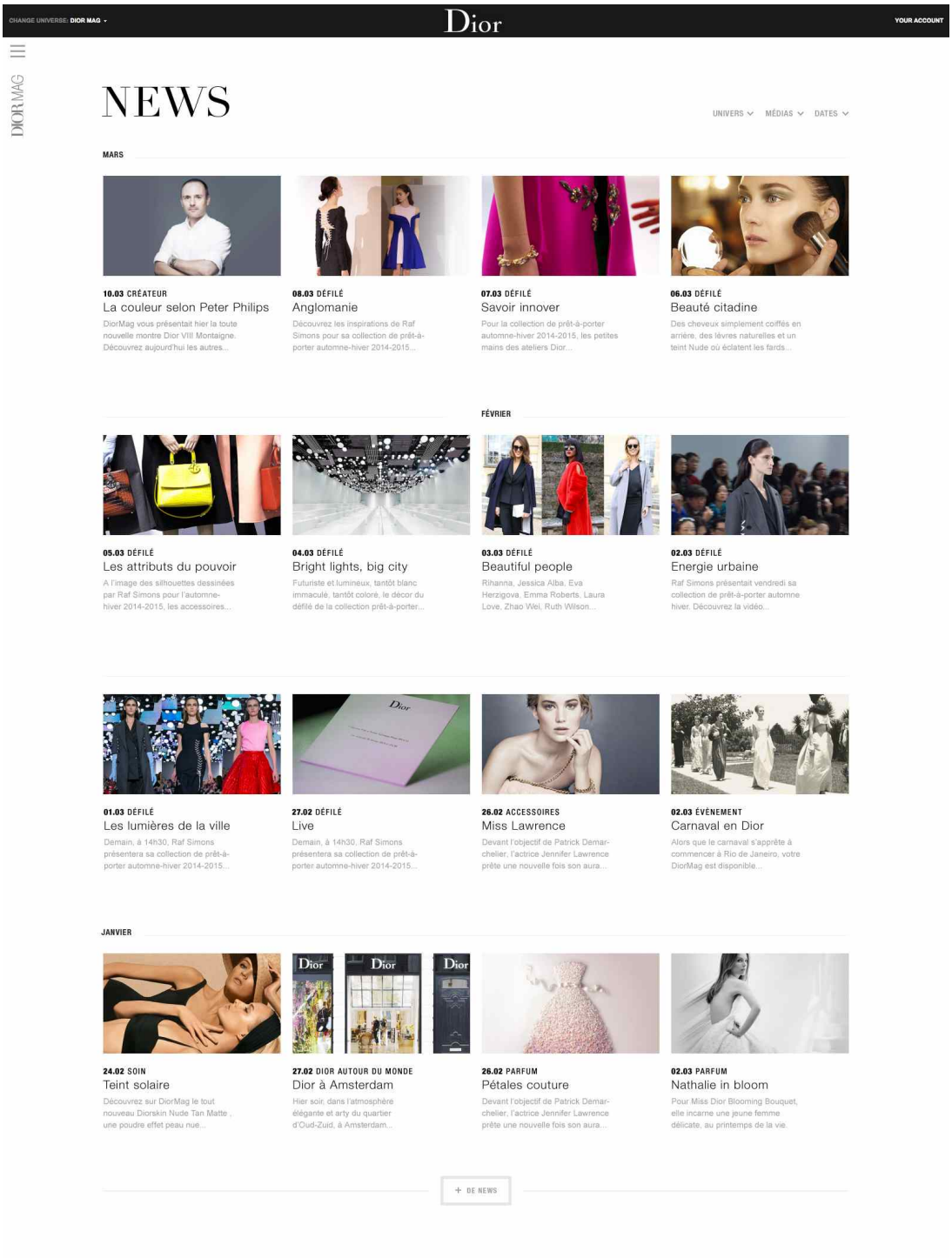
| | | |
|---|---|---|
| **Display at top** | | Dossier |

## Annexe 2.3 - Type de contenu Grands Angles

| Nom | Obligatoire | Commentaire |
|---|---|---|
| **Title** | oui | Ligne de texte |
| **Language** | oui | Liste déroulante comportant la liste des langues du site. Français est pré choisi par défaut. |
| **Chapô** | | Editeur de texte WYSIWYG. Le champ autorise les espaces insécables. |
| **Cover grand angle** | oui | Référence image média permettant d'uploader une image |
| **Label New** | | Case à cocher permettant d'afficher l'étiquette « New » sur le dossier |
| **Display on Homepage** | | Case à cocher permettant d'afficher un Grand Angle sur la page d'accueil |

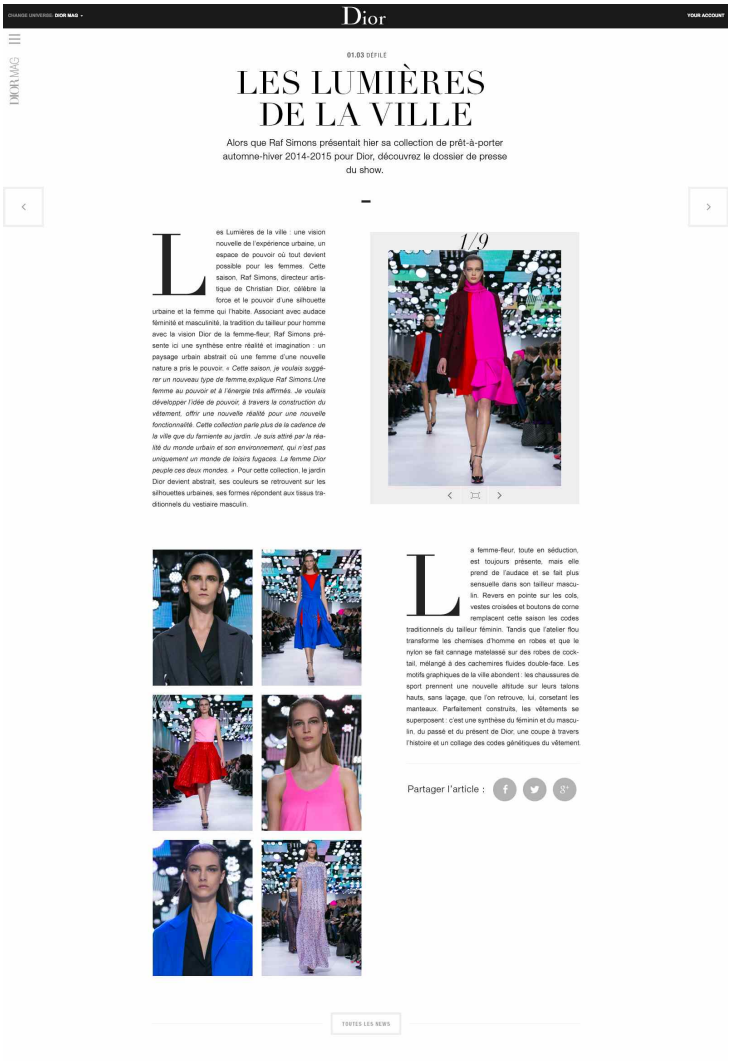
Remarques :

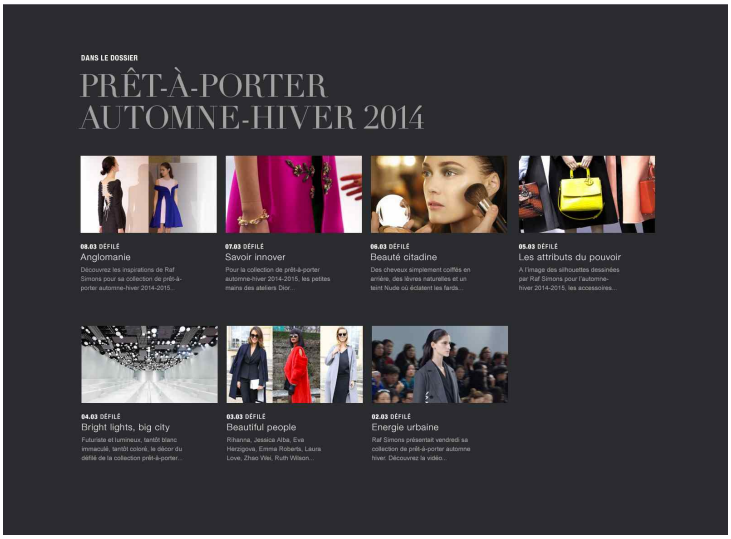
- Pour les types de contenu Article et Dossier, il y a 2 champs *Title* car l'un est nécessaire pour l'indexation dans système Drupal et l'autre sert aux contributeurs pour rajouter des styles qui s'afficheront sur la page du node et les pages Grille.
- Les espaces insécables ont été mis en place par Ludovic Bernard, IED, pour les champs titre et chapo : les parties entourées de crochets carrés ("[" et "]") seront nécessairement sur la même ligne.




**Annexe 3.1 – Grille News**

## Annexe 3.2 – Page article avec zone dossier associée en bas de page




## Annexe 3.3 – Autre layout de la page Article

# Index

**AJAX** : technologie basée sur l'objet XmlHttpRequest du langage javascript, qui permet notamment de lancer des requêtes HTTP, de recevoir des réponses à ces requêtes puis de modifier le contenu affiché sur la base de ces réponses sans jamais recharger la page. C'est l'un principaux éléments qui permit l'introduction du Web2.0.
**attribut HTML** : au format "clé=valeur", permet de définir des propriétés via les attributs pré-existants définis dans la spécification W3C du HTML, comme pour définir le chemin de l'image via l'attribut "src" (*<img src="path/to/img.jpg"></img>*) ou l'URL d'un lien (*<a href="www.domaine.com/page">Lien </a>*). On peut également définir des attributs personnalisés dans toute balise, de la forme "data-nom=valeur".
**back-office :** C'est la partie d'un système informatique qui n'est pas accessible aux utilisateurs finaux ou aux clients, par opposition à une application de front office.
**callback (fonction de) :** une fonction de rappel, généralement appelé et exécutée lorsqu'un évènement se déclenche. On dit alors que cette fonction est la fonction de callback de cet évènement. Dans le cas du hook_menu dans Drupal, le hook_menu défini un élément de menu lié à une URL relative. Quand une requête HTTP est faite sur cette URL, la fonction de callback est exécutée.
**chroot** : est une commande des systèmes d'exploitation UNIX permettant de changer le répertoire racine d'un processus de la machine hôte.
**cross-browser :** est la possibilité pour toute application web, sous format HTML ou programmée avec un langage de script s'exécutant côté client de supporter plusieurs navigateurs web
**drag-&-drop :** ou "glisser-déposer", est dans une interface graphique une méthode consistant à utiliser une souris, pavé ou écran, pour déplacer d'un endroit à un autre un élément graphique présent sur l'écran d'un smartphone, tablette ou ordinateur.
**fork (nom) :** un nouveau logiciel créé à partir du code source d'un logiciel existant. Cela suppose que les droits accordés par les auteurs le permettent : ils doivent autoriser l'utilisation, la modification et la redistribution du code source.
**forker :** créer un fork
**gestionnaire de versions :** un logiciel qui permet de stocker un ensemble de fichiers en conservant la chronologie de toutes les modifications qui ont été effectuées dessus. Les plus connus sont SVN, Git, Mercurial et Bazaar.
**jQuery :** est une bibliothèque JavaScript libre et multi-plateforme créée pour faciliter l'écriture de scripts côté client dans le code HTML des pages web
**chemin relatif/absolu :** dans une arborescence de fichier, le chemin relatif va se baser sur un contexte pour trouver le chemin du fichier ou dossier recherché en descendant dans l'arborescence à partir de ce contexte. Le chemin absolu commence à la racine, que ce soit dans un système UNIX (avec un "slash" initial pour indiquer la racine) ou sur le web (lien complet http://www.domaine.com/chemin)
**MVC :** pour Modèle-Vue-Controlleur, désigne un patron de conception de programmation informatique destiné à séparer les opérations de gestion de la forme (Vue) et du contenu (Modèle), avec une gestion séparée des évènements et interactions utilisateurs (Controlleur).

**HTTP (serveur) :** est un logiciel servant des requêtes respectant le protocole de communication client-serveur Hypertext Transfer Protocol (HTTP), qui a été développé pour le World Wide Web.
**Open Graph :** protocole permettant à n'importe quelle page web de devenir l'objet enrichi d'un graphe social. Par exemple, il est utilisé sur Facebook pour permettre à une page web de bénéficier des mêmes fonctionnalités que n'importe quel autre objet sur Facebook
**pager :** interface généralement composée de 2 flèches permettant de passer d'un élément à l'autre dans une catégorie d'élément donné. Le nom vient du fait que ces catégories sont souvent des pages (web) comme sur les moteurs de recherche.
**pop-in :** à l'inverse de la pop-up, qui une fenêtre ouverte par le site web visité par l'utilisateur sur le système d'exploitation de celui-ci, la pop-in est vue comme moins invasive car une surcouche du site web, qui assombrit (mais laisse en transparence) les zones de la page n'appartenant pas à la pop-in elle-même.
**responsive design :** technique de montage HTML/CSS permettant aux sites web conçus par ce biais de proposer une interface intuitive et revisitée pour chaque format d'écran (mobile, tablette, desktop).
**roll-over :** action de placement du curseur de la souris sur un élément considéré. On en parle souvent quand un événement javascript est déclenché par cette action.
**SQL (injection) :** technique visant à insérer du code SQL dans un champ pour modifier les requêtes par défaut et pirater le site web. Il est possible de s'en prémunir via des requêtes préparées, qui vont échapper et ne jamais exécuter le contenu provenant des utilisateurs comme un code SQL, mais comme un simple contenu.
**SVN :** système de gestion de version centralisé, c'est-à-dire que toute opération de versionnement doit se faire autour d'un serveur central.
**Template :** en français « gabarit » ou « patron », est un patron de mise en page où l'on place images et textes. Dans le cas du web, on parle généralement de templates HTML dans lesquels la structure de la page est organisée et où seuls les contenus (nom d'utilisateur, chemin de l'image, etc.) sont à insérer, souvent par le moteur de template (PHPTemplate pour Drupal).
**template (moteur de) :** application de rendu proposant souvent une API pour faciliter le rendu du contenu dans la structure HTML. Il s'intègre dans la partie Vue du patron de conception MVC.
**TMA : Tierce Maintenance Applicative,** est la maintenance appliquée à un logiciel ( « applicative » ) et assurée par un prestataire externe dans le domaine des technologies de l'information et de la communication.
**Virtualisation :** consiste à faire fonctionner un ou plusieurs systèmes d'exploitation / applications comme un simple logiciel, sur un ou plusieurs ordinateurs - serveurs / système d'exploitation, au lieu de ne pouvoir en installer qu'un seul par machine

# Bibliographie


(1) www.smile.fr/Societe
(2) http://www.developpez.com/actu/63310/La-croissance-du-marche-de-l-open-source-devrait-elle-alarmer-les-editeurs-de-logiciels-proprietaires-En-France-le-marche-pese-2-5-milliards-d/
(3) source Wikipédia et culture personnelle
(4) http://www.contegro.com/info-center/designers-blog/blog-article/_thread_/a-brief-history-of-cms-development
(5) http://bahai-library.com/what_is_cms
(6) http://www.techofafrica.com/wordpress-a-10-ans-decouvrez-les-chiffres-et-les-grandes-dates-du-celebre-cms/
(7) http://w3techs.com/technologies/overview/content_management/all
(8) MVC : http://fr.wikipedia.org/wiki/Mod%C3%A8le-vue-contr%C3%B4leur
(9) http://www.cmswiki.com/tiki-index.php?page=HistoryOfCMS + Wikipédia
(10) http://www.web-and-development.com/a-framework-or-a-cms-what-is-better-to-choose/
(11) http://www.grafikart.fr/blog/framework-php-bon-ou-pas
(12) source : bruits de couloir
(13) https://www.drupal.org/getting-started/before/overview
(14) http://www.robozen.com/does-drupal-still-suck/
(15) http://blog.alantondelier.net/2014/03/19/simplifiez-et-accelerez-vos-installations-drupal-avec-drush-make
(16) http://www.softpanorama.org/WWW/lamp.shtml
(17) http://news.netcraft.com/archives/2014/04/02/april-2014-web-server-survey.html
(18) http://db-engines.com/en/ranking
(19) http://en.wikipedia.org/wiki/Sun_acquisition_by_Oracle
(20) http://news.netcraft.com/archives/2013/01/31/php-just-grows-grows.html
(21) https://github.com/blog/1470-five-years
(22) www.mazarine.com
(23) http://www.videojs.com/
(24) http://www.massmotionmedia.com/
(25) http://fr.wikipedia.org/wiki/Site_web_adaptatif
(26) https://api.drupal.org/api/drupal/includes!entity.inc/class/EntityFieldQuery/7
(27) http://api.jquery.com/jQuery.getJSON/
(28) https://developers.facebook.com/x/bugs/357750474364812/