UNIVERSITÉ DE FRIBOURG SUISSE
UNIVERSITÄT FREIBURG SCHWEIZ

Projet : Technologies Web

Alina Petrescu
Nevena Radovanovic

20 décembre 2013

Projet Technologies Web SA13 - Rapport

Département d'Informatique • Université de Fribourg - Universität Freiburg • Boulevard de Pérolles 90 • 1700 Fribourg • Switzerland

http://unifr.ch/diuf/

Table des matières

1 Avant-propos

Notre projet représente un vrai travail de groupe qui nous a tenues occupées durant plusieurs semaines parsemées de quelques nuits blanches inévitables mais inoubliables. La collaboration pour la partie programmation a été facilitée par le système de stockage en ligne *Dropbox* et par le serveur Web du *DIUF* qui héberge notre projet. De plus, nous avons utilisé la base de données du *DIUF* et son interface graphique *phpMyAdmin*. Toute cette infrastructure, qui nous a été mise à disposition gracieusement par l'Université, a été complétée par l'utilisation, sans modération, des réseaux sociaux. Par contre, aux séances du cours, nous avons participé en chair et en os et avec beaucoup de plaisir.

Concernant le timing, il n'y a pas eu de problèmes grâce à l'efficacité de notre équipe, d'un côté, et à la qualité de l'encadrement dont nous avons pu bénéficier, d'un autre côté.

L'esprit d'équipe a aussi persisté durant la rédaction du rapport final. Nous avons décidé de faire ce rapport exclusivement en français pour faciliter la propagation de notre travail dans le monde francophone.

2 Introduction

2.1 But du projet

Étant toutes deux des filles passionnées de mode et d'achats en ligne, nous avons décidé d'utiliser notre savoir-faire informatique pour la mise en place d'un site de e-commerce spécialisé en mode féminine. Parmi les choix de vêtements, nous avons gardé, à titre d'exemple, les robes en tout genre. Puis nous avons ajouté une page de chaussures, souvent des talons, et finalement une page d'accessoires qui regroupe les colliers, les bracelets et les bagues. De fil en aiguille, nous avons finalement nommé le site **Nevali - Your Online Luxury Shop** où l'acronyme *Nevali* provient de nos deux prénoms.

2.2 Organisation du document

À part les deux premières sections, l'avant-propos et l'introduction, ce rapport contient encore cinq sections et la bibliographie.

La section 3 présente les fonctionnalités du site propres à chacun des trois niveaux d'utilisation.

La section 4 donne des informations concernant l'architecture du projet et l'implémentation de ses fonctionnalités.

La section 5 passe en revue les principaux aspects d'ergonomie.

La section 6 met en évidence des détails de conception qui donnent de la personnalité à notre site.

La section 7 fait la synthèse de notre travail, évoque les problèmes rencontrés au long du projet ainsi que les solutions apportées et suggère de possibles améliorations ultérieures.

2.3 Motivations

L'objectif de ce projet a été de se familiariser avec les technologies du Web vues dans le cours. Notre tâche principale a été de mettre en oeuvre un Content Management System (CMS). Ce travail nous a permis d'utiliser les notions théoriques qui nous ont été présentées et d'approfondir, plus particulièrement, des aspects pratiques liés à la construction de sites web et à leur complexité.

2.4 Outils

Pour réaliser ce projet, nous avons utilisé :

- le système d'exploitation Linux (Ubuntu 13.10) ;
- l'éditeur *gedit* pour écrire le code (HTML, PHP, JavaScript et CSS) ;
- le client Cisco AnyConnect pour établir une connexion sécurisée VPN au réseau de l'université ;
- un point de montage par protocole SFTP pour accéder et travailler sur les fichiers du site web à distance ;
- le protocole SSH pour manipuler commodément les droits d'accès aux divers fichiers du site ;
- le service Dropbox pour le partage des fichiers et le contrôle des versions ;
- l'infrastructure fournie par le *DIUF* pour faire tourner le site web (le système LAMP, l'interface *phpMyAdmin* et la base de donnée SQL) ;
- le navigateur Firefox pour tester le site web, avec l'extension *FireBug* (très utile à la fois pour examiner les éléments de la page générée par le serveur et reçue par le navigateur ainsi que pour expérimenter des retouches concernant les feuilles de style CSS) ;
- les réseaux sociaux Facebook et Whatsapp pour communiquer en temps réel et pour synchroniser à distance notre travail ;
- le logiciel WinEdt 8.0 et la distribution MiKTeX sous Windows 8 pour la rédaction de ce rapport en LaTeX.

3 Utilisation du site

Concernant la navigation sur notre site, on peut distinguer trois types d'utilisateurs, à savoir :

- le **visiteur** qui correspond aux personnes pas encore enregistrées ou aux personnes enregistrées et qui ne se sont pas encore loguées, ou à celles qui se sont déloguées ;
- l'**utilisateur enregistré** qui correspond aux personnes ayant un compte sur le site et loguées ;
- l'**administrateur** qui, après s'être logué avec un compte lui donnant des droits supplémentaires, peut éditer, ajouter ou supprimer du contenu.

Après plus d'une quinzaine de versions successivement améliorées, l'interface **publique** du site web final (accessible aux visiteurs) a été structurée en huit pages correspondant aux sujets et aux fichiers `.php` suivants :

- l'accueil `home.php` ;
- les nouvelles `News.php` ;
- les robes `Dresses.php` ;
- les chaussures `Shoes.php` ;
- les accessoires `Accessorize.php` ;
- des informations générales et sur les créateurs du site `Us.php` ;
- les commentaires `Contact.php` ;
- les résultats de recherche `SearchRes.php`.

En outre, par rapport à un simple visiteur, chaque utilisateur enregistré a accès à trois pages supplémentaires concernant :

- son profil `profile.php` ;
- son panier `BasketMod.php` ;
- la liste de ses achats `By.php`.

En plus, l'administrateur peut aussi accéder à deux autres pages afin de remplir son rôle :

- la page de gestion des utilisateurs `UserMan.php` ;
- la page de l'historique des achats de chaque utilisateur `History.php`.

3.1 Le visiteur

Le visiteur n'a aucun privilège particulier sur le site. Il peut *naviguer* parmi les huit pages principales (voir Figure 1) afin de : *consulter* les produits et leurs photos, *lire* ou *écrire* des commentaires sur le site, ou faire une *recherche* de produit. Il n'a aucun profil ni accès aux fonctions d'achat ou d'administration. En revanche, il peut devenir un utilisateur enregistré en *se loguant* soit sur un compte existant soit sur un nouveau compte créé à cet effet.

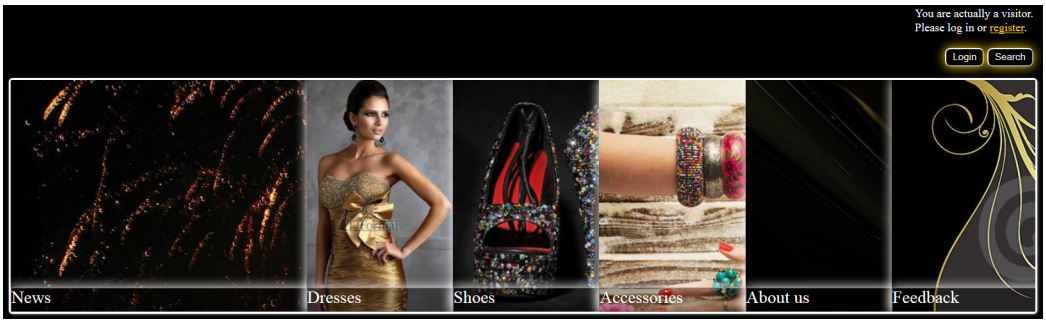

FIGURE 1 – Page d'accueil du *visiteur*

3.2 L'utilisateur enregistré

L'utilisateur enregistré a plus de privilèges qu'un visiteur. Après s'être logué, il a le droit de faire tout ce que fait le visiteur et, de plus, il peut *acheter* des produits, *consulter son profil* (changer son mot de passe ou son nom d'utilisateur, revoir ses achats), *voir son panier*, faire des *recherches* et *se déloguer*. Après chaque achat validé, il reçoit aussi une suggestion de produits qui pourraient éventuellement l'intéresser.

Si l'utilisateur décide d'acheter un produit, il doit ensuite préciser les détails souhaités, à savoir la couleur et la taille choisies parmi les disponibilités. Une fois le produit ajouté au panier, l'acheteur peut valider et payer sa commande ou continuer sa navigation sur le site. Il a toujours la possibilité de voir le contenu et le coût de son panier, et aussi de supprimer les produits qu'il ne veut plus. Pour payer, l'utilisateur est obligé d'entrer des informations concernant le mode de paiement, le lieu de la livraison, etc. Après l'envoi de ces informations, l'acheteur est informé si son paiement a été accepté ou non (le refus intervenant si les informations saisies sont incomplètes ou erronées).

3.3 L'administrateur

À part le fait que l'administrateur n'a pas de profil et ne peut pas faire d'achats, il a tous les droits d'un utilisateur enregistré. De plus, il peut effectuer des modifications sur la plupart des pages du site en *créant*, *éditant* ou *supprimant* du contenu. Il accède à ces fonctions à l'aide de l'*admin-bar* (voir Figure 2), une boîte qui surgit et se positionne sous la barre du menu principal, dans les pages modifiables par l'administrateur. Cette boîte peut contenir les boutons `Delete`, `Create` et `Edit`. On donne ci-dessous des précisions

FIGURE 2 – L'*admin-bar*

concernant la réalisation des tâches d'administration.

3.3.1 Bouton *Create*

Sur la page des nouvelles, par exemple, ce bouton active l'affichage d'un formulaire *POST* qui permet d'introduire la nouvelle souhaitée dans la base de données (la date est générée automatiquement).

Sur chaque page de produits, le bouton active l'affichage d'un formulaire *POST* permettant d'introduire toutes les informations relatives à un nouveau produit, y compris un champ de fichier pour charger l'image associée (voir Figure 3).

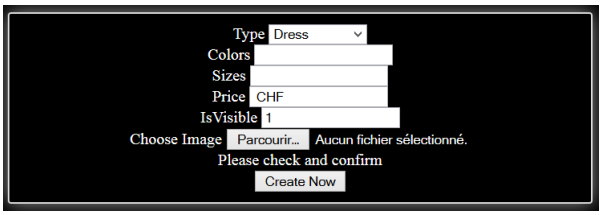

FIGURE 3 – Mode Create

3.3.2 Bouton *Edit*

Visible sur les pages de produits, ce bouton provoque le réaffichage de la page courante (méthode *POST*) sous une forme plus complète, afin de permettre la modification des informations concernant les produits (par exemple la couleur, la taille ou le prix). Plus précisément, ces informations apparaissent, cette fois, sous la forme de champs de formulaire déjà pré-remplis et modifiables (voir Figure 4). Au bas de chaque tel formulaire, un bouton ad-hoc permet l'actualisation de la base de données avec les nouvelles informations pour le produit concerné (via PHP : requête SQL *UPDATE* dans la table *products* de notre base de données). Il convient de mentionner que dans l'affichage obtenu avec le bouton `Edit` se trouvent aussi les éventuels produits cachés aux utilisateurs ordinaires et qui peuvent éventuellement être rendus à nouveau visibles par l'administrateur (en mettant le champ `IsVisible` de zéro à la valeur 1).

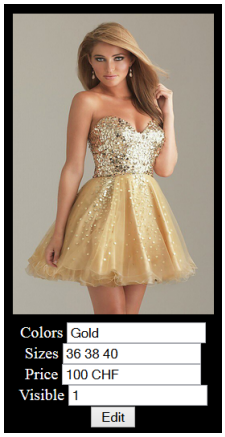

FIGURE 4 – Mode Edit

FIGURE 5 – Mode Delete

Sur la page de gestion des utilisateurs, un même mécanisme permet de remplacer le simple affichage des données utilisateurs par des champs de formulaire et, le cas échéant, de modifier ces données (prénom, nom, identifiant et mot de passe).

3.3.3 Bouton *Delete*

Dans une page de produits, un clic sur ce bouton réaffiche la page en ajoutant en-dessous de chaque produit l'unique élément d'un formulaire *POST* sous la forme d'un bouton qui permet la suppression du produit souhaité (voir Figure 5). Afin d'éviter une suppression par inadvertance, avant que cette opération soit effective, on demande une confirmation supplémentaire de la part de l'administrateur (qui doit appuyer un bouton `Confirm Delete product x` apparaissant dans l'*admin-bar*). Il convient de mentionner que les produits supprimés ne sont pas effacés de la base de données mais rendus seulement invisibles (en mettant le champ `IsVisible` de 1 à la valeur zéro).

Sur la page de gestion des utilisateurs, on peu utiliser le même mécanisme, mais avec une suppression réelle de l'utilisateur dans la base données (c'est-à-dire dans la table des utilisateurs `users`, dans la table des historiques d'achats `History` et dans la table des commandes en cours `Command`).

Sur la page de nouvelles, le bouton `Delete` provoque le réaffichage de la page avec un bouton supplémentaire à côté de chaque nouvelle, afin de permettre la suppression de celle-ci.

Sur la page de commentaires, il n'y a pas d'*admin-bar*, car la seule action possible est la suppression d'un commentaire. Par conséquent, pour l'administrateur, chaque commentaire apparaît accompagné d'un bouton supplémentaire permettant sa suppression.

3.3.4 Bouton *User Management*

L'accès à la page de gestion des utilisateurs se fait via le bouton `User Management` (voir Figure 6) qui s'insère dans la boîte contenant les boutons pour la gestion du logout et de la recherche de produits, et qui est placée au-dessus de la barre de menus, tout à droite. La page de gestion présente un tableau avec les données de tous les utilisateurs. De plus, grâce au bouton `See history of buyings`, l'administrateur peut accéder à la page d'historique des achats de chaque utilisateur.

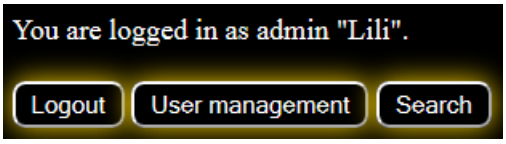

FIGURE 6 – Le bouton *User Management*

4 Architecture du projet

La réalisation de ce projet met en place des technologies basées sur les langages HTML, PHP, CSS, JavaScript et SQL, et qui peuvent intervenir soit au niveau du serveur (qui produit les pages web et gère les données du site), soit du côté du client (qui utilise un navigateur pour télécharger et afficher des pages web afin de les consulter ou d'interagir avec le serveur).

4.1 Côté serveur

C'est le serveur du DIUF à l'Université de Fribourg qui gère la base de données, abrite les pages web statiques et produit les pages web dynamiques.

4.1.1 Base de données

Une base de données SQL est utilisée pour stocker les données relatives au site : les utilisateurs, les produits, les achats, les commentaires, etc.

- La table `users` contient la liste des utilisateurs enregistrés (et qui ne sont pas donc des administrateurs). Un champ `username` et un champ `password` servent au processus de login. Des champs supplémentaires `firstname` et `lastname` permettent de personnaliser l'expérience de l'utilisateur (page de profil, par exemple). Le champ `UserID` est la clé principale de la table qui permet la jointure avec les autres tables.
- La table `admin` contient la liste des administrateurs du site. On y retrouve à nouveau les champs `username` et `password` pour le *login*.
- La table `products` est la plus conséquente car elle regroupe la liste de tous les produits proposés sur le site. La clé primaire `ProductID` générée automatiquement (avec la propriété unique) sert d'identifiant principal de chaque produit. Le champ `Type` de type texte définit la catégorie du produit : `dresses`, `shoes` ou `accessories` (pour les robes, chaussures et accessoires respectivement). Il permet la sélection des produits concernés dans chacune des pages correspondant du site (de type `dresses` pour la page `dresses.php`, etc). Le champ `Color` permet de spécifier les couleurs dans lesquelles est disponible chaque produit (champ texte, sous la forme "*bleu, rouge, rose*" par exemple). Le champ `Size` concerne les tailles disponibles pour chaque produit. Le champ `Price` indique le prix d'un produit (champ texte, au format "*300 CHF*" par exemple). Enfin, un champ `Photo` de type `blob` permet de stocker l'image qui sera affichée pour chaque produit.
- La table `news` stocke les nouvelles affichées dans la page de nouvelles. Un champ `Date` est rempli automatiquement à la création de chaque nouvelle depuis le CMS. Les champs `Title` et `Content` sont remplis via un formulaire ad-hoc du CMS et donnent le titre de la nouvelle et son texte respectivement.

- La table `comments` contient les commentaires affichés dans la page *Feedback* et qui ont été soumis via le formulaire contenu dans cette page. Le champ `name` correspond au nom indiqué par l'auteur du commentaire et le champ `comment` contient le commentaire en lui-même.
- La table `Command` représente le panier "global" du site, à un moment donné. Le champ `CommandID` est utilisé pour différencier les commandes. Ce champ est incrémenté automatiquement. Le champ `userName` contient le nom d'utilisateur enregistré ayant procédé à l'achat. Le champ `ProductID` est l'identifiant du produit acheté, le champ `Size` correspond à la taille choisie pour ce produit et le champ `Color` à la couleur. Une fois que l'utilisateur a payé son panier, tous les produits payés sont effacés de cette table et envoyés à la table `History`.
- La table `History` contient les mêmes champs que la table `Command`, et ici sont enregistrées toutes les commandes déjà payées. Nous utilisons cette table pour afficher l'historique des achats de l'utilisateur (lorsque l'administrateur clique sur le bouton See history of buyings).
- La table `thumbs` contient deux champs : `up` et `down`. Ils sont incrémentés à chaque fois qu'un visiteur ou un utilisateur enregistré clique sur l'une des figures correspondant à ces champs dans la page *Feedback* (afin d'exprimer son accord ou son désaccord vis-à-vis du site, respectivement).

4.1.2 PHP

La génération des pages web envoyées au client et affichées par son navigateur est réalisée par le serveur et programmée en PHP. Des informations stockées côté serveur peuvent ainsi être transmises au client. Par contre, pour récupérer des informations du client, on se sert de formulaires utilisant la méthode `POST` ou la méthode `GET` (transmises dans l'entête de la requête HTTP, pour la méthode `POST`, et dans l'URL, pour la méthode `GET`).

Gestion des sessions et des droits d'accès Pour le processus de *login* et les informations de session, on utilise la méthode `POST`, plus sécurisée pour ce type d'échange. Les informations de session sont stockées dans des variables PHP, donc côté serveur, à savoir les variables de session correspondant à un tableau associatif `$SESSION['']`. Plus précisément, `$SESSION['login_user']` pour l'identifiant de login et `$SESSION['user_type']` pour les niveaux de privilège : *visitor*, *user* ou *admin*.

La portée des variables de session s'étend à tout le code PHP, mais il faut d'abord appeler la fonction `session_start()`. Celle-ci initialise une session, si aucune session n'a été ouverte, ou récupère les variables de la session en cours, si une telle session existe déjà. Cette fonction est exécutée au tout début du script PHP. Elle est en fait contenue dans le fichier `phpsessionheader.php` et correspond à l'*en-tête* qui est inclus au début de presque tous les fichiers `.php` écrits. Juste après, on initialise par défaut la variable `$SESSION['type_user']=''visitor''` si celle-ci n'est pas déjà initialisée (comme dans le cas d'une ouverture de session où on accède pour la toute première fois au site ou si on vient de se déloguer).

Concernant le *logout*, celui-ci est effectué du côté utilisateur par un formulaire `POST` réduit au bouton *logout*. Ce dernier recharge la page et l'*en-tête* intercepte ce message `POST`, puis appelle la fonction `session_destroy()` qui détruit toutes les variables de session. Il ne reste plus qu'à recharger la page pour compléter sa déconnexion (on recharge en

fait la *homepage*, car un utilisateur qui se déconnecte ne souhaite pas forcément que son navigateur se souvienne de la dernière page visitée).

Pour exécuter le processus de *login*, il faut que la page ait été chargée via le formulaire correspondant à la connexion : champs *username*, *password* et bouton *login*. C'est un formulaire `POST` (comme mentionné ci-dessus). L'*en-tête* intercepte le message `POST` de ce formulaire, récupère les variables *username* et *password* transmises via POST et effectue une recherche SQL dans les tables `users` et `admin` pour vérifier la présence d'un tel couple *username/password*. Selon la table où l'on a trouvé le couple, on affecte à la variable de session `$SESSION['user_type']` la valeur `user` ou `admin`. Dès lors, le reste du code pourra accéder à cette variable pour bénéficier des fonctionnalités associées à l'utilisateur enregistré ou à l'administrateur.

Remarquons que le code permettant l'activation des privilèges utilisateur et administrateur est en fait la réunion des portions de page de code PHP qui sont incluses dans l'*en-tête*. Nous avons regroupé ces blocs de code dans le dossier *login*. C'est un dossier critique au niveau de la sécurité. On veillera particulièrement à bien positionner les droits d'accès et d'écriture sur ce dossier (comme on doit de toute façon le faire sur l'ensemble du site).

Génération des pages La génération dynamique des pages web repose sur un nombre restreint d'éléments fondamentaux associés les uns aux autres :

- des **tests if** permettant d'adapter une page aux droits d'utilisateurs (à l'aide des variables de sessions) et à des actions et des formulaires soumis (grâce aux variables POST ou GET) ;
- des **boucles while** qui permettent des itérations sur des listes de données ;
- des **requêtes SQL** qui alimentent les boucles *while* ;
- des **include** de pages PHP (ou Javascript).

Concernant le dernier élément, nous avons segmenté notre code en beaucoup d'unités associées à des fonctionnalités particulières (un bouton, un formulaire, une présentation d'un seul produit, la barre d'administration, de login, une fenêtre de dialogue, du javascript, etc.) et qui sont assemblées par des `include`. Des variables appropriées ont permis de paramétrer le rendu de ces `include` (produit, type de produit, nom d'utilisateur, etc.).

4.2 Côté client

Une fois les pages web générées et envoyées au client, celles-ci ne restent pas tout à fait figées, mais gardent un comportement dynamique car elles peuvent réagir à certaines actions de l'utilisateur. Le code PHP a déjà été exécuté du côté serveur. C'est le code Javascript, exécuté du côté client, qui donne cette réactivité (car il est plus rapide de ne pas devoir repasser par le serveur et recharger la page).

Nous avons privilégié le code Javascript pour les fonctionnalités du site qui ne requièrent pas un échange avec le serveur. C'est le cas, par exemple, du bouton qui active l'affichage d'un formulaire (comme le bouton `Create` de l'`admin-bar`). Par contre, le bouton qui soumet le formulaire doit communiquer avec le serveur.

Des fonctionnalités avancées d'affichage comme les fenêtres de dialogue intégrées à la page ou le diaporama des photos agrandies des produits ont été rendues disponibles en faisant appel aux librairies *JQuery* et *JQueryUI* écrites en Javascript. À ce propos, la dernière version de *JQueryUI* entre en conflit avec la toute dernière version de *JQuery*.

Cependant, comme mentionné plus tard, nous avons su trouver des versions compatibles et offrant les différentes fonctions dont nous avons eu besoin.

4.2.1 Javascript

Nous avons créé plusieurs fonctions qui permettent l'apparition des boîtes de dialogue spécifiques (voir les figures 7, 8, 9) et ces fonctions se trouvent dans le fichier `ScriptPopupDialog.js` du dossier `login`. Comme chaque boîte de dialogue, par l'option `modal`, bloque le reste de la page (qui devient grisée, et avec laquelle l'utilisateur ne peut plus interagir), il n'est plus possible d'ouvrir deux boîtes à la fois. Dans une version antérieure du site, il fallait que ces fonctions gèrent également la dissimulation des autres boîtes de dialogue. Ces fonctions sont activées par des boutons ou par des

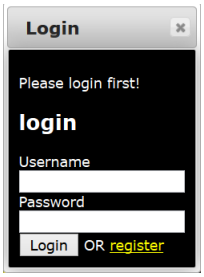

FIGURE 7 – Popup Login

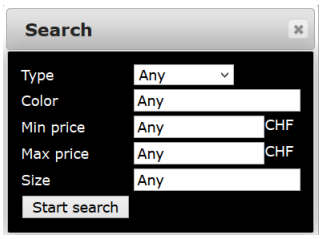

FIGURE 8 – Popup Search

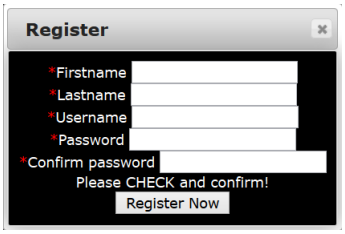

FIGURE 9 – Popup Register

liens (balises d'ancres <a>). Pour cela, on peut utiliser soit l'évènement `onclick` par l'intermédiaire de l'option `onclick=fonction-a-lancer()` pour les balises <button> soit l'option `href=javascript:fonction()` pour les balises d'ancres.

Par rapport au PHP, le langage Javascript est très avantageux pour faire apparaître et disparaître des éléments de l'interface graphique. En pratique, les solutions Javascript sont nettement plus réactives (délais imperceptibles pour l'utilisation du Javascript, contre souvent une seconde pour une requête POST adressée au serveur du DIUF, en passant par le client VPN). Le Javascript est également moins intrusif : on ne réinitialise pas la page et les informations peuvent être dans des formulaires partiellement remplis et non soumis. Tout cela améliore l'ergonomie pour l'utilisateur, sa bonne perception de la fluidité du site, son confort, et, peut-être, favorisera son envie d'achat.

Nous avons également utilisé le langage Javascript pour profiter au mieux des possibilités offertes par la librairie *JQueryUI*. À la création des boîtes de dialogue (dans le fichier `logininfoboxDialogs.php` du dossier `login`), nous avons pu paramétrer précisément le comportement de ces boîtes via les options de la classe *Dialog* de *JQueryUI*. C'est le cas, par exemple, de la boîte de dialogue d'inscription `register`. On donne ci-dessous quelques exemples de telle options :

- boîte non affichée par défaut : $("#dialogRegister").dialog({autoOpen :false}) ;
- boîte non déplaçable, par l'option `draggable:false` ;
- boîte dimensionnée à son contenu, par les options `height` et, `width` à `auto` ;
- boîte disparaissant avec un effet de fondu, par l'option `hide` à `fade` ;
- boîte désactivant (grisant) le reste de la page, avec l'option `modal` à `true` ;
- boîte non redimensionnable par l'utilisateur (la taille automatique convient), par l'option `resizable` à `false` ;

- boîte en haut de la page (le milieu de la page pose des soucis d'ergonomie), par l'option `position` à `top`.

La documentation sur ces options a certes demandé beaucoup de temps, mais les tester et les choisir précisément nous a permis d'économiser du travail sur les feuilles de style CSS et sur les fonctions de gestion des boîtes.

D'une manière générale, dans la mesure du possible, nous avons essayé de résoudre les problèmes rencontrés et de réaliser les tâches proposées à l'aide de solutions "home-made". Cependant, nous n'avons pas poussé ce choix à l'extrême en voulant à tout prix "réinventer la roue". Par exemple, l'agrandissement de la photo d'un produit a été résolu d'abord par nos propres moyens mais la variante finale retenue fait appel à la libraire *JQueryUI*. Ceci n'a pas apporté forcément un grand gain de temps, car découvrir l'existence de cette librairie et la compréhension de son but et de son utilisation a demandé de longues heures de recherche, lecture et réflexion. Par contre, la variante finale est plus riche en fonctionnalités et de qualité supérieure.

4.3 Content Management System - CMS

L'essentiel des fonctionnalités administrateur pour le CMS sont centralisées dans l'`admin-bar` implémentée principalement dans le fichier `admin-bar.php` qui inclut :

- les boutons `Delete`, `Edit` et `Create` ainsi que les actions associées, et l'affichage de `Mode:Delete` ou `Mode:Edit` ;
- le fichier `ButtonConfirmDelete.php` qui gère l'étape de confirmation de suppression (d'un produit ou d'un utilisateur) avant d'effectuer l'opération (protégeant ainsi contre l'appui sur un bouton `Delete` par inadvertance) ;
- le fichier `DeleteProduct.php` qui effectue la mise à jour de la base de données SQL (requête *UPDATE* du champs `IsVisble` car ce n'est pas une suppression qu'en apparence) ;
- le fichier `DeleteClientSQL.php` qui effectue la suppression du client de la base de données (requête *DELETE* sur les trois tables concernées : `users`, `Command` et `History`) ;
- le fichier `EditProduct.php` qui effectue la mise à jour d'un produit dans la base de données (requête *UPDATE*) ;
- le fichier `EditClientSQLUpdate.php` qui effectue la mise à jour des informations d'un client dans la base de données (requête *UPDATE*) ;
- le fichier `Createform.php` qui contient le formulaire de création d'un produit ; ce formulaire n'est pas affiché par défaut mais on peut le faire apparaître en cliquant sur le bouton `CREATE`.

Vu que les requêtes SQL demandent les identifiants d'accès à la base de données, il a fallu inclure (par des instructions `include`) le fichier `config.php` du dossier `login` dans les fichiers utilisant de telles requêtes. De plus, un test adéquant associé à un choix *if* permet d'afficher un message d'erreur en cas de non succès de la requête SQL.

Une autre partie du CMS est implémentée dans le fichier `UserMan.php` par l'`include` du fichier `ClientEdit.php`, qui contient le formulaire d'édition des informations pour un utilisateur. Ce formulaire est chargé dans la boucle *while* du fichier `UserMan.php`.

Les pages `Dresses.php`, `Shoes.php` et `Accessorize.php` incluent chacune le même fichier `TableProducts.php`, mais après avoir initialisé la variable `$producttype` à une valeur propre à chacune des pages. L'exécution du fichier `TableProducts.php` dresse et

affiche la table des produits de type `$producttype`. À chaque itération de la boucle `while` du fichier `TableProducts.php`, le fichier `PictureProductEdit.php` est chargé si l'utilisateur est un administrateur en `Mode:Edit` (tandis que le fichier `PictureProduct.php` est chargé pour un utilisateur ordinaire et ceci produit un affichage "normal"). En fait, `PictureProductEdit.php` contient un formulaire d'édition. Pour la fonctionnalité de suppression de contenu correspondant à un administrateur en `Mode:Delete`, on se sert du fichier `ButtonDelete.php` qui affiche le bouton correspondant au sein de la présentation d'un produit.

4.4 CSS - Feuilles de style

Il convient de reconnaître que sans CSS, il n'y aurait pas de site aussi joli et fonctionnel. En effet, les feuilles de style ont été essentielles pour la conception de notre site d'achats et nous n'avons utilisé que cinq fichiers `.css` pour les cent et quelques fichiers `.php` écrits, à savoir :

- Le fichier `Nevali.css` est le fichier principal pour la gestion de l'affichage du site.
- Le fichier `galerie-home.css` gère le style de la barre de navigation (*navbar* grande) de la *homepage*.
- Le fichier `galerie-menu.css` gère le style de la barre de navigation (*navbar* petite) de toutes les autres pages, excepté la page d'accueil.
- Le fichier `admin.css` gère l'affichage de l'`admin-bar` et de la boîte de dialogue de création lorsqu'on clique sur le bouton `Create`.
- Le fichier `print.css` gère l'aperçu avant impression et l'impression.

Nous avons décidé de n'utiliser que des feuilles de style CSS pour la barre de navigation et rien d'autre (pas de Javascript, etc). Ce fût un joli défi et nous en sommes très fières.

Nous avons aussi constamment utilisé les balises `div` pour structurer les pages. Elles regroupent le contenu en blocs et nous facilitent la mise en page avec le CSS ensuite.

Nous avons préféré utiliser les tailles relatives (en `%`) plutôt que les tailles absolues (en `px`) car elles gèrent mieux le redimensionnement des pages.

De plus, dans la mesure du possible, nous avons choisi des tailles en `em` plutôt qu'en pixel car elles gèrent mieux le redimensionnement du texte.

Voici les trois types de positions que nous avons utilisés :

- `absolute` : positionnement par rapport au conteneur, sans réservation de place dans le flot ;
- `relative` : positionnement par rapport à la place dans le flot ;
- `fixed` : positionnement par raport à l'écran, indépendamment de tout le reste.

Finalement, ces méthodes de conception de feuilles de style CSS nous ont permis d'obtenir une mise en page qui s'adapte non seulement aux écrans normaux d'ordinateurs, mais aussi aux formats des smartphones et tablettes.

5 Ergonomie

Durant les tests d'ergonomie, nous avons profité des avis de plusieurs personnes. Chacune d'entre elles nous a proposé des modifications intéressantes à faire. Nous avons beaucoup appris durant ces moments-là et nous en gardons un très bon souvenir.

- Lors des premiers tests, plusieurs personnes ne trouvaient pas ce qu'il fallait faire pour acheter un produit. Il n'y avait pas de bouton *Buy* sous les photos et ce n'était pas clair qu'il fallait d'abord se connecter pour ensuite acheter. Nous avons donc ajouté un bouton *Buy* qui, de plus, fait apparaître une fenêtre de *login* pour les *visiteurs*.
- Toujours au début, sur la page d'accueil, il n'y avait que le titre **Nevali**, sans aucune indication concernant le but du site. Nous avons donc ajouté la précision **Your Online Luxury Shop** afin de compléter la description.
- Presque tous les testeurs ont été incommodés par notre ancienne version de zoom sur les photos des produits. En effet, on pouvait cliquer deux fois pour agrandir une image et une fois pour qu'elle revienne en miniature. Cependant, comme la taille des photos agrandies était trop importante, la suite de l'affichage passait à la ligne et on ne la voyait plus à l'écran. Nous avons donc opté pour une solution plus professionnelle qui fait appel aux librairies *JQuery* et *JQueryUI* et à leurs fonctionnalités.
- Lors d'un test en classe, un de nos collègues nous a fait remarquer qu'il n'y avait pas d'adresse de livraison pour la marchandise payée. Nous avons donc ajouté cette fonctionnalité lors du paiement.
- Après les tests en classe, nous avons aussi ajouté des étoiles rouge à côté des champs obligatoires.
- Toujours grâce aux testeurs, nous avons pu découvrir et corriger plusieurs petites erreurs au niveau du code accédant à la base de données (surtout au niveau des requêtes).
- Sur la page *About Us*, nous avons ajouté des informations sur nous-mêmes ainsi que nos photos.
- Nous avons principalement effectué nos tests de rendus avec les navigateurs Firefox et Chrome, sous Ubuntu. Mais quelques tests sous Windows et sous Android ont également été réalisés.
- Nous avons bien fait attention aux re-dimensionnements lors des zooms par "clic roulette", et lors des changement du cadre des pages (fenêtres plein écran, petites ou sur la moitié de l'écran). Dans la partie CSS nous avons donné plus de détails sur ces aspects. Ainsi, nous avons pu penser le regroupement des divers éléments de la page en fonction du comportement par rapport à ce type de tests.
- De manière générale, l'utilisation de *JQuery* et *JQueryUI* pouvait poser quelques soucis de compatibilité par rapport au re-dimensionnement et par rapport aux styles CSS appliqués. Nous avons dû regarder en détail la documentation et revoir toutes les options CSS afin d'obtenir, après beaucoup d'essais, un comportement ergonomique.
- Finalement, les professeurs membres du jury d'évaluation de notre travail ont recommandé l'ajout d'une gestion de tous les utilisateurs et ceci a été le dernier élément avec lequel nous avons complété le site.

6 Points forts du site

La qualité de notre site réside aussi dans l'attention spéciale accordée aux détails. Nous avons gardé à l'esprit le fait que l'utilisateur doit prendre possession du site avec facilité et y revenir avec plaisir. Voici quelques exemples :

- Lorsqu'un visiteur (non-authentifié) clique sur un bouton *Buy*, une fenêtre "popup" contenant le formulaire d'inscription s'affiche.
- Nous avons créé une feuille de style `print.css` qui permet d'afficher et d'imprimer seulement les photos et les descriptions des produits.
- Nous avons protégé le code "sensible" avec des *if* relatifs à la variable `$SESSION['user_type']` de sorte qu'un pirate ne puisse pas y avoir accès ou deviner des variables ; nous avons aussi bien fixé les droits d'accès aux fichiers ; nous avons préféré les commentaires entre balises PHP par rapport aux commentaires HTML qui sont envoyés au navigateur.
- Tous les titres ont été réalisés "à la main" (à l'aide d'un éditeur approprié) ainsi que mis en fond transparent pour l'esthétique du site.
- Lorsqu'on clique sur la photo d'un produit, une fenêtre "popup" apparaît et affiche l'image en grande taille. Il est ensuite possible de faire défiler les photos agrandies des autres produits soit à l'aide des touches (gauche, droite, haut et bas) du clavier soit à l'aide de la roulette de la souris.
- Nous avons ajouté une "favicon" (comportant la lettre N en couleurs jaune et or) sur l'ensemble du site.
- Nous avons prévu aussi un *footer* (bas de page) à la fois pour rendre les pages plus jolies et pour permettre aux utilisateurs de nous contacter facilement.
- Après chaque paiement effectué, l'utilisateur se voit proposer d'autres produits que le site lui recommande.
- Les administrateurs peuvent gérer agréablement les comptes des utilisateurs.
- Nous avons ajouté une image *.gif* (avec des flocons de neige) en fond de titre sur toutes les pages du site (pour rappeler la période des fêtes).
- Le code a été divisé en plus d'une centaine de fichiers regroupés en plusieurs dossiers, et il a été abondamment commenté. Ceci a rendu sa lisibilité meilleure et sa compréhension plus facile. Le découpage intelligent du code l'a rendu plus flexible (on peut ainsi facilement lui ajouter de nouvelles fonctionnalités) et sa maintenance est devenue plus aisée (la gestion simultanée de nombreuses fonctionnalités du site se retrouve simplifiée). Par exemple, l'ajout des flocons de neige en fond de titre a été réalisé sans difficulté et on peut même envisager une refonte du style pour chaque saison (grâce un fichier de style global qui gère l'ensemble du site).

7 Conclusions

7.1 Synthèse

Dans le cadre de ce projet, nous avons réalisé l'implémentation d'un CMS qui nous a permis la création d'un site web de e-commerce professionnel. Nous avons investi un temps assez conséquent à la réalisation de ce site mais nous ne le regrettons pas un instant. Au tout début, nous étions "perdues" car nous ne savions ni comment démarrer ni ce qu'on attendait précisément de nous. Mais, une fois le canevas du site posé et grâce à l'excellent encadrement dont nous avons bénéficié, nous nous sommes lancées dans l'écriture effective du code et les questions ont commencé à se répondre d'elles-mêmes. Nous sommes vraiment très fières de ce que nous avons réussi à faire et nous espérons que le site vous plaira tout autant.

7.2 Problèmes rencontrés

Durant notre travail, nous avons dû résoudre des problèmes "normaux" liés aux implémentations choisies. Cependant, à part ces problèmes, nous avons rencontré toute une série de situations particulières pour lesquelles il a fallu trouver des solutions ad-hoc. Ci-dessous, on présente quelques exemples :

- Sans être des fans de la plateforme Windows, nous avons essayé au début d'utiliser ce système d'exploitation pour réaliser le projet. Après plusieurs essais infructueux et grâce à la patience et au pouvoir de persuasion des nos camarades, nous avons continué l'aventure dans le monde merveilleux de Linux.
- Nous avons dû montrer beaucoup de patience afin de régler maintes détails qui ont finalement assuré un affichage agréable et fluide (et la plupart des solutions font intervenir les feuilles de style CSS).
- Au début, la synchronisation du travail nous a posé quelques soucis (car on la faisait manuellement, par des opération "copier/coller" depuis Dropbox sur nos espaces de travail respectifs). Il a fallu nous imposer une stricte discipline de travail afin d'éviter la perte de données suite à un travail simultané sur un même fichier.
- Nous avons rencontré des conflits entre les dernières versions de la librairie *JQueryUI* et respectivement *JQuery*. Par exemple, nous ne pouvions pas à la fois réaliser le diaporama des images agrandies des produits et afficher les boîtes de dialogue. Finalement, après avoir consulté de nombreux forums dédiés aux librairies Javascript, nous avons trouvé les "bonnes" librairies compatibles.
- Pour les boîtes de dialogue, nous avons choisi au début un affichage au milieu de la page mais celui-ci s'est avéré inapproprié, surtout pour les pages dont les dimensions dépassaient la taille de l'écran. De plus, le contenu des fenêtres de dialogue redimensionnables par l'utilisateur n'arrivaient pas à suivre correctement les changements de la taille de ces fenêtres. Pour résoudre ces soucis, nous avons revu intégralement la documentation officielle pour les fenêtres de dialogue afin de trouver précisément des fonctions et des options utiles. Dans la solution retenue, les boîtes de dialogue ne sont plus redimensionnables et sont modales et affichées en haut de l'écran.
- Pour tester les modifications apportées aux fichiers CSS, il a fallu bien prendre garde à recharger chaque fois la "nouvelle" page (celle affectée par ces modifications). Plus précisément, il faut provoquer le renouvellement du cache afin d'éviter que la page emploie toujours des feuilles de style ou des images issues de la mémoire du navigateur. Concrètement, à la place d'une simple ré-actualisation par l'appui de la touche F5, on doit utiliser la combinaison de touches CTRL+SHIFT+R.

7.3 Améliorations futures

Nous pensons que les objectifs fixés dans le cadre de ce projet ont bien été réalisés et espérons que le jury aura la même opinion. Cependant, nous sommes conscientes que des améliorations peuvent encore être apportées et nous mentionnons ci-dessous quelques exemples :

- permettre aux administrateurs du site d'ajouter de nouvelles pages et pas seulement d'éditer les pages existantes ;
- faire plusieurs tables de produits, chacune associées à un certain type de produit ;
- donner la possibilité aux administrateurs d'ajouter des étiquettes aux produits, en cas de soldes ou pour les nouveautés (comme *-50%* ou *new* par exemple) ;

- améliorer le formulaire de connexion en ajoutant, par exemple, une fonctionnalité du type *Forgot password ?* ou une question de sécurité pour aider l'utilisateur à récupérer son mot de passe ;
- ajouter une flèche en bas de l'écran qui permettrait de remonter immédiatement tout en haut de page ;
- rendre la barre de menu "collante" par rapport au défilement haut-bas ;
- pouvoir enregistrer des préférences de l'utilisateur (afin de personnaliser le style global du site en fonction de l'utilisateur) ;
- déployer le site sur un serveur accessible au grand public et implémenter la possibilité de faire de vraies transactions.

7.4 Remerciements

Maintenant que nous approchons la fin de ce projet, nous pouvons dire que nous avons eu beaucoup de plaisir à travailler dessus. Même lorsque nous avons rencontré des difficultés, nous les avons regardées comme un défi et nous avons pu les surmonter.

Tout d'abord, nous voulons remercier Messieurs les Professeurs Gérald COLLAUD et Jacques MONNARD pour leur excellent cours *Projet : Technologies Web*. Leurs réponses précises et rapides à toutes nos questions ont contribué de manière significative à faciliter notre apprentissage et ont rendu ce projet plus compréhensible et agréable.

Le présent devoir nous a permis la découverte et l'appréhension de plusieurs langages et technologies web ainsi que la joie du travail en équipe. De plus, la réalisation d'un projet informatique d'envergure qui résout de manière professionnelle un "vrai" problème métier nous a rendues plus confiantes en nos capacités et nous a apporté une précieuse expérience.

Plus particulièrement, nous sommes reconnaissantes vis-à-vis des membres de nos familles pour leurs soutient et encouragement.

Références

[1] Projet : Technologies Web, *Gérald Collaud et Jacques Monnard*, Centre NTE.

[2] PHP & MySQL, *Super Poche, Versions 4 et 5*, Seconde Édition - Avril 2008.

[3] Le langage PHP 5, *OSYX, Version 1.6*, 2008.

[4] `http://www.php.net/` - La documentation en ligne du langage PHP.

[5] `http://www.w3schools.com` - Souvent utilisé pour les feuilles de style (CSS).

[6] `http://fr.flamingtext.com/` - Le site utilisé pour tous les titres du site.

[7] `http://jquery.com/` - Manipulation des boîtes de dialogue.

[8] `http://jqueryui.com/` - MAnipulation du zoom pour les photos.

[9] `http://fr.wikipedia.org` - Souvent utilisé pour des informations générales et pour des explications simples.