Projet de fin de module

**Accueil**

Thème : magasin en ligne

Date de fin du projet : dimanche 26 avril 2020

Défense orale (individuelle) : mercredi 29 avril, 6 et 13 mai 2020

Procédure à suivre pour la remise du projet :

* L’entièreté du code doit être dans un répertoire github (accessible par l’utilisateur FBruynbroeck).
* Un dump complet de la base de données mysql doit se trouver dans le répertoire github.
* Analyses, diagrammes, documentations du projet seront également présents dans le répertoire github.
* Un fichier README à la racine du projet contiendra la procédure à suivre afin d’installer le projet.

|  |
| --- |
| Le projet fonctionne avec la version 5.6 de php. Editer les variables $db\_url, $db\_login, $db\_pass dans le fichier config.php. $db\_url: chemin vers votre db MySQL $db\_login: utilisateur ayant accès en lecture/écriture à la db. $db\_pass: mot de passe de l'utilisateur utilisé pour la connexion à la db ($db\_login). Créer une base de données MySQL ayant pour nom "projet" et procéder au restore du dump du fichier projet.sql accessible à la racine du répertoire. |

Envoyer l’url de votre projet + le thème (exemple : Magasin de jeux vidéo) à l’adresse francois.bruynbroeck@hotmail.com avant le 26 avril (=dès que possible)  
Attention, le sujet du mail doit commencer par: [PROJET IPAM] + Nom + Prénom.  
Exemple: [PROJET IPAM] Bruynbroeck François

La récupération de votre projet depuis github se fera le 26 avril à minuit. Il ne sera donc plus possible de modifier votre projet après cette date.

Pensez à bien push votre code (régulièrement) et vérifiez que votre répertoire github possède bien l’entièreté du projet (accédez à votre répertoire depuis l’interface web github si vous avez un doute).

**Limitations**

Le site web permettra aux visiteurs de :

* Consulter la liste des articles
* Voir le détail des articles
* De s’inscrire sur le site
* De s’authentifier

Le site web permettra (en plus) aux clients (personnes authentifiées) de :

* Ajouter des articles dans son panier
* Valider une commande (confirmer son panier)
* Consulter la liste de ses commandes.

Le site web permettra aux administrateurs de :

* Ajouter des articles
* Editer des articles
* Supprimer des articles
* Lister les commandes
* Statistique des commandes

SEO : La structure des URLs doit être soignée.

Les URLs comportent toujours un mot ou une phrase identifiant la nature de la page (pour une phrase, les mots seront séparés par des tirets).

Exemple : l’utilisateur souhaitant accéder au détail d’une chaise de jardin accédera à l’url :

Monprojet.be/articles-jardin/chaise

Libre à vous de structurer les URLs comme vous le souhaitez

Il est également demandé d’utiliser à un moment le langage javascript dans votre projet.

Quelques idées d’utilisation :

* Un formulaire pour rajouter des articles via un fichier json avec une prévisualisation (preview via javascript)
* Drag and drop des articles dans le panier.
* Prévisualisation du rendu final d’un article en temps réel quand on complète un formulaire de création.
* Message d’avertissement quand on encode un article déjà créé (avant même de confirmer le formulaire)
* …

*Ce sont des exemples ! Vous ne devez pas implémenter l’entièreté de la liste !*

Concernant les articles, le minimum à afficher est :

* Un titre
* Un prix

La gestion d’un stock n’est donc pas obligatoire (on suppose que le stock est illimité)

* Ou validation d’un membre du personnel du magasin : OK.

Concernant les commandes, le minimum à afficher est :

* Le nom du client qui passe la commande.
* Les articles commandés.
* Le prix total de la commande.

Concernant les statistiques des commandes, l’administrateur doit (en un clin d’œil) pouvoir visualiser les articles les plus vendus grâce à un graphique.

Le site sera développé en PHP (minimum v5.6) ET JavaScript sans aucun framework.

Les données sont stockées dans une base de données MySQL.

Aucune limitation au niveau des librairies que vous souhaitez utiliser (Ex : jQuery).

Attention cependant à utiliser des librairies récentes (vous êtes pro, et donc, vous devez utiliser des outils à jour).

Côté skin (CSS), le site doit être agréable à visiter.

Vous devez donc structurer le rendu et respecter les balises (<h1…2…3>, <div>, <label>, <table>, <th>, etc)

Pensez également à la résolution des écrans pouvant varier.

Conseil : utilisez des librairies comme Bootstrap (<https://getbootstrap.com>) afin de limiter les risques et garantir ainsi votre site responsive.

**Structure**

Le développeur doit respecter l’architecture MVC.

Il est important de séparer vos fichiers en (minimum) 3 dossiers :

* Model
* Views
* Controllers

Un fichier «index.php » à la racine du projet fera office de routeur.

En fonction de l’avancement du cours, le développeur devra présenter les modèles sous forme d’objet PHP qui fera alors office de mapper (ORM). Si la notion Orienté Objet n’est pas acquise, le développeur devra alors au minimum travailler avec des fonctions PHP au niveau des modèles.

N’oublier pas de sécuriser vos dossiers et de configurer apache (via .htaccess) afin de toujours faire pointer les requêtes vers votre routeur.

Attention : il est possible que votre client git n’ajoute pas les fichiers cachés (exemple : .htaccess). Vérifier donc cela (configurez votre client pour qu’il le fasse ou ajoutez les fichiers depuis l’interface web de github).

**Par où commencer ?**

Avant de foncer dans le code, vous devez reprendre les différentes demandes du client sous forme de point.  
Ensuite, préparez votre analyse.

Pour faciliter votre développement, vous pouvez représenter les différents cas d'utilisation (via un diagramme des cas d'utilisation).

Exemple:  
En tant que visiteur, je peux:

* Voir la liste des jeux vidéo
* Voir le détail d'un jeu vidéo
* M'identifier
* Créer un nouveau compte utilisateur

En tant qu'administrateur, je peux:

* Ajouter un jeu vidéo
* Importer des jeux vidéo
* Editer un jeu vidéo
* Supprimer un jeu vidéo
* Editer mon profil
* Voir le listing des commandes
* Me déconnecter

En tant que simple utilisateur connecté, je peux:

* Ajouter des articles dans un panier
* Voir mes commandes
* Editer mon profil
* Me déconnecter

Grâce à ce genre de diagramme, vous avez une vision plus claire du projet. Il sera alors plus facile de développer sans oublier quelque chose.

Ensuite concevez votre base de données en fonction de votre projet.  
Là aussi, une analyse est nécessaire afin de ne pas oublier une table, une colonne, une relation.  
Préparez un diagramme (via UML par exemple) afin d'avoir un visuel de votre base de données.  
Il sera alors plus facile de créer votre base de données sur base de votre diagramme.

Une fois que la base de données est prête, commencez par coder les différentes pages qui seront utilisées (aucune logique, juste un titre).  
Attention: Ne pas oublier de bien respecter la structure MVC !

Ensuite, implémentez votre page de listing qui est l'étape la plus simple.

Sans le savoir, en procédant de cette façon, vous travaillez avec une méthode "agile".  
Le principe ici est de développer par étape en ayant un rendu visuel "rapide".  
De cette façon, vous pouvez revenir vers votre client en lui montrant l'état d'avancement du projet.  
Quelques avantages:

* Vous voyez un avancement rapide.
* Le client voit que son projet avance.
* Le client peut revenir sur des détails que l'on peut alors adapter/corriger rapidement (ce qui -serait plus difficile une fois le projet terminé).

Continuez l'implémentation des autres pages.

N'oubliez pas que des outils/plugins/librairies existent ! Il ne faut pas refaire la roue.

**Extra**

Rendez votre client heureux

Les gens ne savent pas ce qu'ils veulent avant que vous le leur montriez - Steve Jobs

C'est la même chose pour votre client.  
Soyez imaginatif. Pensez logique et facilité.

Pourquoi ne pas lui proposer un formulaire d'inscription à une newsletter pour ses visiteurs ? De cette façon, on pourrait envoyer de la publicité.  
Mais aussi, un formulaire pour éditer le profil, rajouter un avatar aux utilisateurs, permettre l'ajout d'images pour les articles, rajouter une gestion du stock, etc.

Aucune obligation pour cette partie. Il s'agit d'Extra.  
Mais gardez en tête qu'un client satisfait est un client fidèle.  
Si vous allez plus loin que sa demande, il vous sera reconnaissant.

**Idée de structure DB**

Vous retrouverez ici la structure d’une base de données « générique » correspondant au thème du projet.

Description

L’idée est donc d’enregistrer les utilisateurs, les articles ainsi que les commandes.

Pour cela, nous allons d’abord nous préoccuper de la partie « utilisateur ».

Nous devons authentifier les utilisateurs sur base d’un rôle (Administrateur, Client,…)

Il faut donc deux tables :

* Une table rôle
* Une table utilisateur

Rôle (role) contiendra au minimum 2 colonnes :

* Id (INT, PK avec incrémentation automatique) <= identifiant unique d’un rôle.
* Name (VARCHAR) <= nom d’un rôle (exemple : admin)

Utilisateur (user) contiendra au minimum 4 colonnes :

* Id (INT, PK avec incrémentation automatique) <= Identifiant unique d’un utilisateur.
* Login (VARCHAR, de préférence unique) <= login d’un utilisateur
* Password (VARCHAR, Attention à la taille en fonction de votre méthode de hashage) <= password d’un utilisateur.
* Role\_id (INT, FK de la colonne id de la table Role) <= relation permettant de récupérer le rôle de l’utilisateur.

Nous avons donc terminé pour la partie utilisateur.

Passons maintenant aux articles.

Une seule table est nécessaire : une table article.

Article (item) contiendra au minimum 3 colonnes :

* Id (INT, PK avec incrémentation automatique) <= Identifiant
* Title (VARCHAR, éventuellement unique) <= Titre de l’article
* Price (FLOAT) <= Prix de l’article

Vous pouvez également rajouter une table « Catégorie » et rajouter une relation à la table Article pour pouvoir les trier par catégorie (même principe qu’avec la table Utilisateur et Rôle).

Et maintenant, la partie un peu plus complexe : les commandes !

Pour cela, nous allons avoir besoin de trois tables (ou deux tables si on ne compte pas gérer le statut d’une commande) :

* Une table statut
* Une table commande
* Une table commande\_article

Statut (status) contiendra 2 colonnes :

* Id (INT, PK avec incrémentation automatique, ou éventuellement un VARCHAR en PK) <= Identifiant unique d’un statut
* Name (VARCHAR) <= nom d’un statut (exemples : En attente, Traité, Refusé,…)

Commande (book) contiendra au minimum 3 colonnes :

* Id (INT, PK avec incrémentation automatique) <= identifiant unique d’une commande (numéro de la commande)
* User\_id (INT, FK de la colonne id de la table Utilisateur) <= relation permettant de récupérer l’utilisateur (client) de la commande.
* Status\_id (INT, FK de la colonne id de la table Statut) <= relation permettant de récupérer le statut actuel de la commande.

Commande\_article (Book\_item) contiendra au minimum 4 colonnes :

* Id (INT, PK avec incrémentation automatique) <= identifiant unique (ne sera normalement pas utlisé. Mais toujours bien d’avoir une PK)
* Book\_id (INT, FK de la colonne id de la table Commande) <= relation permettant de faire le lien avec la commande
* Item\_id (INT, FK de la colonne id de la table Article) <= relation permettant de récupérer un article.
* Price (FLOAT) <= prix de l’article. Nous copions ici le prix pour garder une trace du prix de l’article au moment ou il a été commandé.

Dump de la db

Voir fichier annexe (dump\_projet)

Diagramme de la db

Voir image en lien (image de la db pour le programme (prof))

**Défense orale :**

A définir (29-04, 06-05, 13-05) : 17h30, 18h10, 18h50, 19h30, 20h10

* Mon jour