14/06/2016

WebRails - Projet

Domaine de Haute-Cour

David Villa et Marc Pellet

[HEIG-VD]

Table des matières

[Introduction 2](#_Toc453635525)

[Cahier des charges 2](#_Toc453635526)

[Type de site et objectifs 2](#_Toc453635527)

[Cas d’utilisation et définition des droits 2](#_Toc453635528)

[Structure de la base de données 2](#_Toc453635529)

[Schéma conceptuel 2](#_Toc453635530)

[Schéma relationnel 2](#_Toc453635531)

[Structure du logiciel 2](#_Toc453635532)

[Implémentation 2](#_Toc453635533)

[Ajax 2](#_Toc453635534)

[Librairies utilisées 2](#_Toc453635535)

[Gestion de projet 3](#_Toc453635536)

[Etat des lieux 3](#_Toc453635537)

[Conclusion 3](#_Toc453635538)

[Annexes 3](#_Toc453635539)

[Mode d’emploi 3](#_Toc453635540)

[Indication pour l’installation 3](#_Toc453635541)

# Introduction

Dans le cadre du cours WebRails il nous a été demandé de trouver un projet à réaliser afin d’assoir les connaissances acquises durant la phase d’apprentissage et à travers les laboratoires.

Le projet en question devait être un site utilisant Ruby on Rails et Ajax. La base de données devait comporter au moins 5 tables ayant au moins une relation N-N.

Au moins 3 acteurs différents devaient pouvoir utiliser le site. Ce dernier devait donc permettre la création de compte de manière sécurisée (encryptage) ainsi que la gestion dudit compte.

L’administrateur devait avoir accès à une zone d’administration pour gérer le site.

L’utilisation de plugin était encouragée pour effectuer ces différentes tâches.

# Cahier des charges

## Type de site et objectifs

Notre projet consiste en un site pour vignerons mettant à disposition un catalogue de marchandises (vins) permettant à l’utilisateur de passer des commandes, une page de contact ainsi qu’une liste d’évènement auquel participera le vigneron.

De son côté le vigneron (administrateur) peut gérer la marchandise du catalogue ainsi que les commandes passées.

Si l’on souhaiterait résumé et catégoriser le type de site concerné par le projet, l’on pourrait dire qu’il s’agit d’une boutique en ligne de vins d’un domaine viticole.

## Cas d’utilisation et définition des droits

**Acteurs :**

Trois types d’acteur ont été mis en place.

Le premier acteur est le visiteur. Un visiteur peut accéder au catalogue des vins ainsi qu’à la liste des évènements. De plus il peut créer un compte ou s’authentifier au cas où il possède déjà un compte.

Le second acteur est le membre. Un membre peut accéder à la liste des évènements ainsi qu’au catalogue des vins. A la différence du visiteur, un membre peut ajouter des bouteilles disponibles dans le catalogue à son panier. Il peut aussi créer une commande à partir des marchandises présentes dans son panier.

De plus un membre peut accéder à une page lui permettant d’éditer les informations de son compte tel que son adresse, son nom d’utilisateur, mot de passe, etc. et d’afficher les commandes passées.

Le dernier acteur est l’administrateur. En plus de pouvoir effectuer toutes les actions d’un visiteur, un administrateur a accès à la zone d’administration, lui permettant de modifier les marchandises présentes dans le catalogue, ajouter, modifier ou supprimer des évènements ainsi que gérer les commandes disponibles.

## Structure de la base de données

### Schéma conceptuel

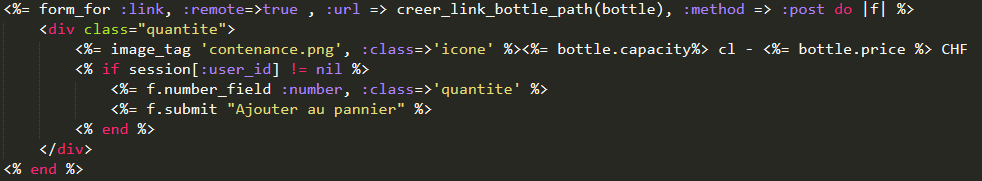
### Schéma relationnel

# Structure du logiciel

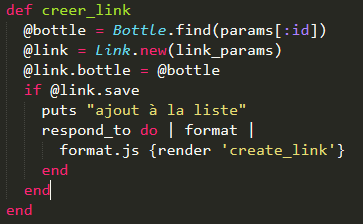
# Implémentation

## Ajax

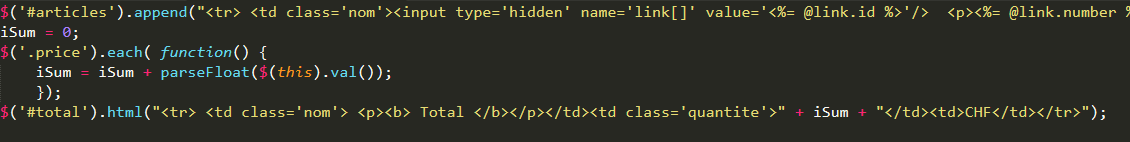
Nous avons utilisé la technologie Ajax pour implémenter un panier de commande dynamique. Quand un utilisateur ajoute un vin à son panier, l’aperçu du contenu du panier est automatiquement mis à jour.



Le code ci-dessus est la partie qui affiche les formulaires permettant d’ajouter des bouteilles au panier. Afin que le post soit traité par Ajax nous avons rajouté l’attribut :remote=>true . Le formulaire est traité dans la fonction créer\_link du bottle.controller

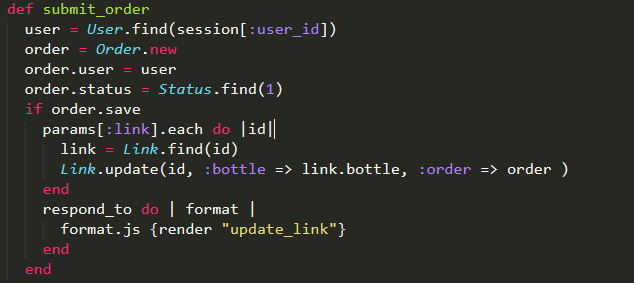


Dans cette fonction on récupère la bouteille ainsi que le nombre désiré afin de créer un nouveau lien. A la création de ce lien, aucune commande ne lui est associée. Au moment où la commande est validée, les liens créés sont mis à jour avec le numéro de la nouvelle commande. Après avoir créé le lien, l’affichage est délégué au fichier create\_link.js



On peut voir que la première ligne est coupée mais son but est d’ajouter à la liste des articles dans le panier le nouvel article que l’utilisateur vient de sélectionner. Ensuite on calcule le prix total en parcourant la liste des articles. Et finalement on affiche le total mis à jour.

Le panier en lui-même est aussi un formulaire traité à l’aide d’Ajax, les données sont cette fois transmises à la fonction submit\_order de order.controller.



La plus grosse bidouille de notre site se trouve dans ces lignes. En effet nous créons tout d’abord une nouvelle commande puis nous rajoutons cette commande aux link dont les id étaient conservée dans un champ caché du formulaire de panier. Nous chargeons finalement update\_link.js de vider la liste des bouteilles dans les paniers afin de pouvoir commencer une nouvelle commande

## Librairies utilisées

BCrypte – Permet de crypter le mot de passe avant de le stocker dans la base de données et donc de le garder secret même pour l’administrateur du site

# Gestion de projet

# Etat des lieux

# Conclusion

Malgré les quelques bugs restants, nous sommes dans l’ensemble satisfait du résultat obtenu. Nous avons réussi à utiliser Ruby on Rails pour créer un site internet à peu près complet et mettre en pratique les concepts vu en cours pendant le semestre. Nous avons pu constater que malgré le fait que nous avions fait tous les laboratoires il était difficile d’appliquer ceci dans un projet non encadré dont le cadre était totalement différent que les exercices. En résumé, nous avons trouvé que Rails est une technologie offrant des possibilités intéressantes, mais peu devenir vite compliqué. Pour certaines parties du projet, il nous a fallu avancer à tâtons en essayant toutes les possibilités jusqu’à ce que ça fonctionne. Au final on a plus l’impression d’être en train de bidouiller que de développer de manière sérieuse.

# Annexes

## Mode d’emploi

## Indication pour l’installation

1. Créer une base de données nommée hautecour
2. Modifier database.yml afin que le nom d’utilisateur et le mot de passe correspondent
3. Bundle install
4. Rake db :migrate
5. Rake db :fixtures :load
6. Accéder au site à l’adresse localhost:3000