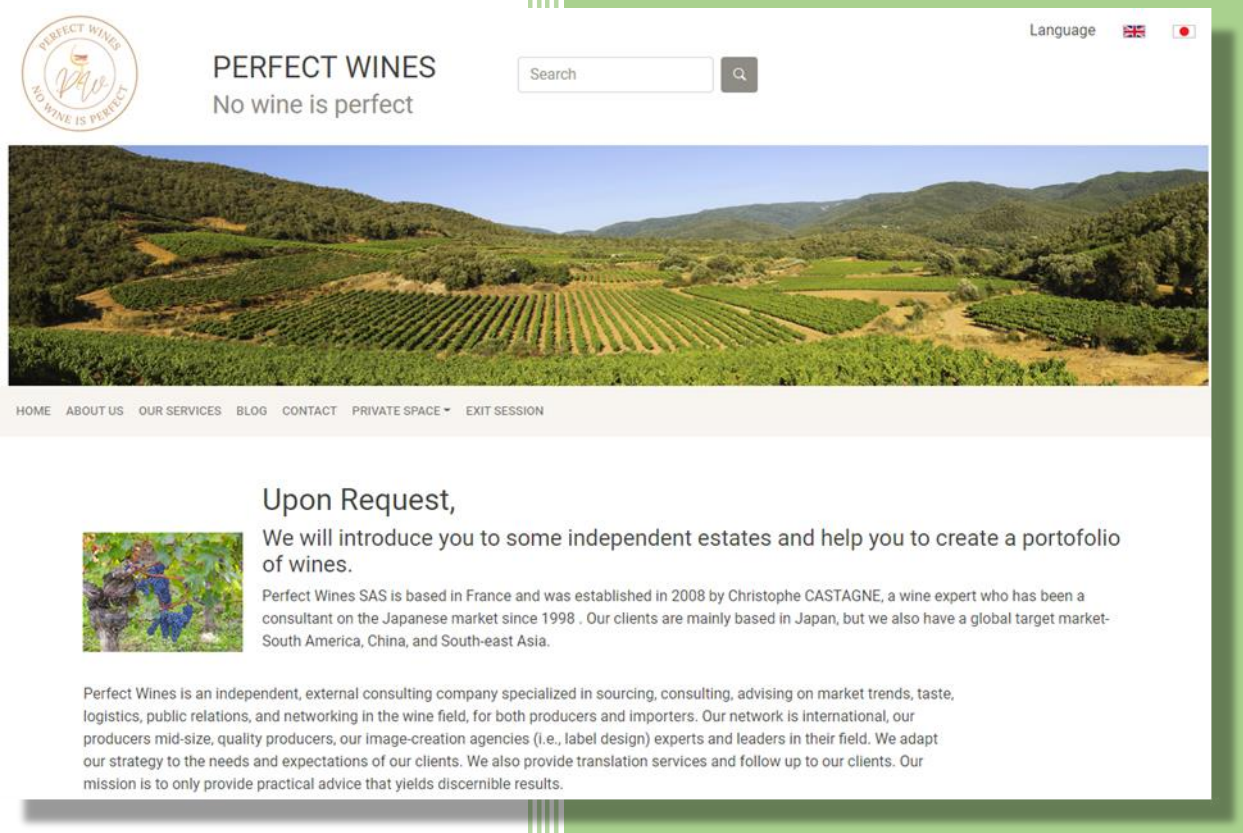
2021

Dossier de projet
www.perfectwines-export.com



TITRE : Développeur Web et Web Mobile

DENIS FARKAS

TABLE DES MATIERES

I. Compétences visées par notre projet professionnel

A. Pour l'activité type 1, « Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité » :

- Maquetter une application,
- Réaliser une interface utilisateur avec une solution de gestion de contenu.

B. Pour l'activité type 2, « Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité » :

- Développer les composants d'accès aux données,
- Elaborer et mettre en œuvre des composants dans une application de gestion de contenu.

II. Résumé

La société Perfect Wines veut adopter un nouveau site internet, pour consolider son activité de sourcing et de consulting dans le domaine des vins et spiritueux, au Japon et au niveau international et pour développer une nouvelle activité de distribution de vins en marque privée (taylor-made).

Il a pour vocation d'être un support qui permettrait d'optimiser les actions commerciales afin de promouvoir ses offres de services et de produits auprès des clients importateurs, tout en élargissant son audience auprès d'une nouvelle clientèle.

Mais, il est aussi destiné à organiser et optimiser la communication dans le suivi du traitement des commandes grâce à un tableau de bord simple, accessible par enregistrement privé avec accès restreint.

Jusqu'à présent, chaque commande nécessitait un échange permanent d'emails, souvent récurrents.

Perfect wines cherche donc à améliorer sa communication interne et externe, tout en fidélisant sa clientèle avec une interface efficiente qui confortera sa position croissante dans ce domaine d'activité.

Leurs représentants, domiciliés à Marseille et Tokyo nous ont demandé, à ma camarade Thuc-nhi Wiedenhofer et moi-même, de créer un site vitrine et un blog d'accès publique, ainsi qu'un ensemble d'interfaces back office avec gestion des droits, une pour chaque société importatrice et ses représentants et une autre pour l'administration de Perfect wines.

Nous allons décrire dans ce rapport différentes fonctionnalités de notre projet professionnel, sans pouvoir détailler la partie de suivi des traitements de commande toujours en prévision de développement, la date finale de livraison ayant été fixée par convention au 15 août 2021.

III. Spécifications fonctionnelles

A. Description de l'existant

L'entreprise dispose depuis 2014 d'un site html/css passif, non responsive, avec quelques animations JavaScript (slideshow). Sa charte graphique est de bonne qualité, mais il a été convenu dans nos entretiens de la modifier complétement.

B. Périmètre du projet

Le site est multilingue (anglais, japonais) et nous pouvons conserver les textes originaux pour la partie vitrine. Nous avons décidé d'alimenter les deux versions à partir de la base de données, chaque page disposant de sa propre table, afin de pouvoir à moyen terme proposer d'autres langues pour accompagner l'expansion de leur activité.

La partie japonaise des textes est sous la responsabilité d'une traductrice à Tokyo.

C. Cible adressée par le site internet

L'entreprise travaille actuellement avec 6 entreprises importatrices japonaises et plus de 20 prestataires export. Une association récente avec une entreprise espagnole devrait décupler ce public, avec notamment une ouverture sur les marchés d'Amérique Latine.

D. Gestion des droits

Afin de contrôler les espaces personnalisés des clients importateurs, nous avons convenu de différencier les rôles, leurs droits et leurs enregistrements sur deux niveaux.

Perfect Wines enregistre un compte entreprise pour son client et lui affecte un représentant, en créant un compte importateur à leur principal interlocuteur dans cette entreprise.

Ce « prime importer » crée ensuite dans l'espace privé de son entreprise des comptes « importers » pour ses assistants qui voudraient accéder au système.

Celui-ci se charge donc de la gestion de son équipe, avec des droits de création, lecture, modification et suppression de ces comptes.

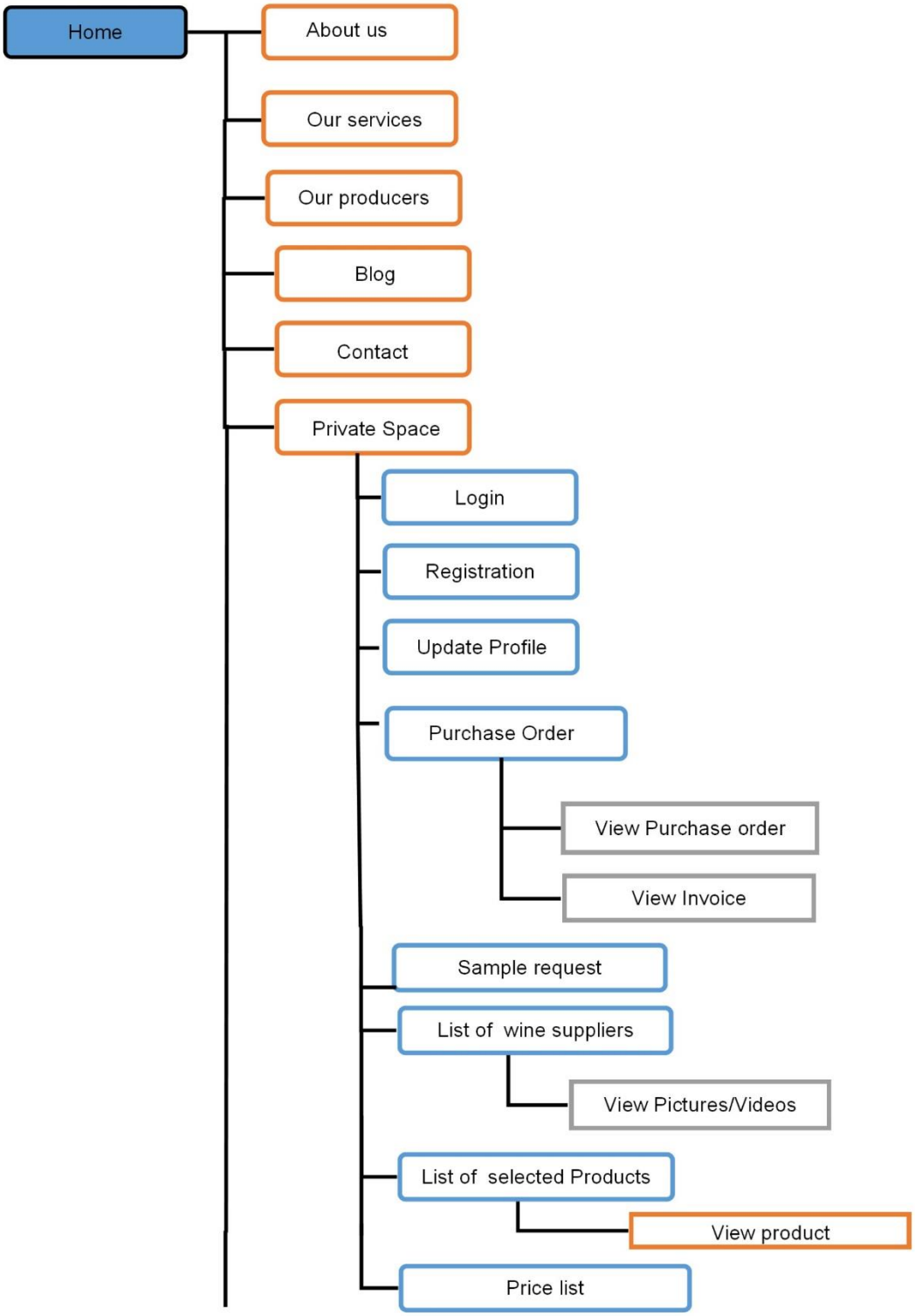
Les assistants peuvent modifier à leur guise, leurs données personnelles respectives, leurs mots de passe, mais ne peuvent en aucun cas altérer leur id ainsi que leur rôle.

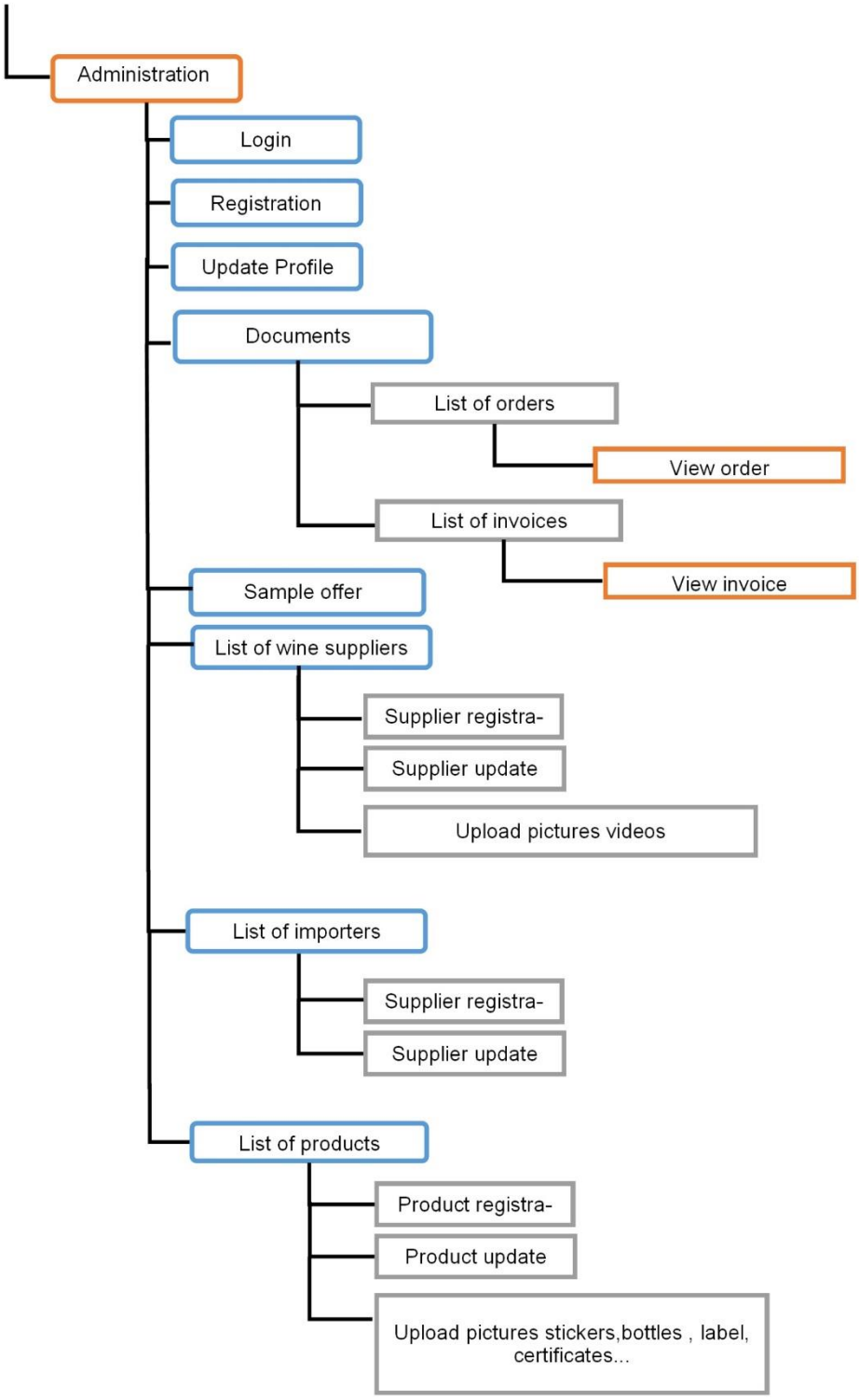
Perfect wines conserve dans son back office un contrôle total de toutes ces procédures de CRUD et se réserve le droit en cas de possible faille ou de malversation de supprimer des comptes assistants.

Les informations disponibles pour chaque compte client sont différenciées (les prix d'achats, les fournisseurs, etc…). Il est donc impératif de maintenir un cloisonnement entre chacune de ces entreprises.

Nous verrons par la suite, comment l'utilisation du framework Codeigniter nous aide dans cet objectif de sécurisation et cloisonnement.

E. Arborescence du site




Administration
Login
Registration
Update Profile
Documents
List of orders
View order
List of invoices
View invoice
Sample offer
List of wine suppliers
Supplier registra-
Supplier update
Upload pictures videos
List of importers
Supplier registra-
Supplier update
List of products
Product registra-
Product update
Upload pictures stickers,bottles , label, certificates...

IV. Description des fonctionnalités

A. Accès publique

Le public en général a accès aux pages du site vitrine, ainsi qu'au blog. Son interaction avec l'interface se limite au choix de la langue, ainsi qu'à la navigation entre les différentes pages et articles du blog. Il ne peut pas ajouter de commentaires dans ce dernier, l'entreprise n'ayant pas de temps à consacrer à la modération. Chaque page est alimentée dans une version de langue, par une requête sur sa table de bdd respective, qui contient en colonnes l'intégralité des zones de texte, des boutons, titres, etc... et en lignes les différentes langues disponibles.

Il peut toutefois laisser dans le formulaire contact, ses coordonnées et un message pour établir une communication avec l'entreprise. Ce formulaire est envoyé automatiquement par mail à l'entreprise.

B. Espace privé clients importateur

1. Page de connexion

Un simple formulaire de connexion email / password permet d'accéder à l'espace privé de l'entreprise. Cette connexion contrôle en bdd l'existence du compte ainsi que la véracité de son password crypté par password hash et met en session l'ID, l'ID de son entreprise, son email ainsi que son rôle.

Le rôle de l'importateur détermine un accès différencié à l'interface :

Le « prime importer », représentant de l'entreprise accède à un espace de back office complet ; ses assistants à un espace personnalisé plus restreint.

2. Back office Prime importer

Le représentant dispose d'un ensemble de fonctionnalités de CRUD sur les comptes assistants : il peut créer, modifier, lire et effacer chacun de ces comptes à partir d'un tableau qui liste les membres de son équipe, doté d'une fonction modifier et d'une fonction effacer. Il peut aussi modifier le compte entreprise pour les champs adresse, téléphone et email ainsi que ses propres coordonnées.

a) Formulaire d'inscription d'un compte assistant

b) Formulaire de modification d'un compte assistant

c) Formulaire de modification des coordonnées de son entreprise

d) Formulaire de modification de son compte

Chacun de ces formulaires est protégé par une procédure de vérification sur les variables sessions.

3. Espace privé importer

L'assistant de l'importateur visualise le compte entreprise, la liste des assistants ainsi que ces informations personnelles.

Son unique possibilité de modification concerne ses propres coordonnées.

a) Formulaire de modification de son compte

4. Catalogue produits et fiche produit

Les deux rôles ont accès au catalogue de produits personnalisé. Personnalisé car avec son activité de marque privée, Perfect Wines personnalise la présentation de certaines catégories de bouteilles pour chaque entreprise importatrice.

Les prix affichés sont déterminés par l'id de l'entreprise et alimentés par une table dans la bdd dédiée à cette fonction.

La page catalogue affiche une image de présentation de la bouteille, son origine, l'année de fabrication et le prix officiel. Une fonction Voir donne accès à la fiche produit.

La page produit affiche les images de présentation de la bouteille, recto-verso, les images de certification et de médailles, les images de bouchon et de capsule ainsi que les spécifications techniques (cépages, degré d'alcool, etc…).

a) Page catalogue produits

b) Fiche produit

5. Page nos fournisseurs et fiche fournisseur

La page nos fournisseurs présente un tableau récapitulatif des fournisseurs avec quelques informations succinctes sur la région de production, les appellations, etc..

La fonction voir dans ce tableau permet d'accéder à une fiche succincte sur le producteur de vin.

Avec la possibilité de visualiser des photos ainsi que des vidéos sur le producteur.

a) Page nos fournisseurs

b) Fiche producteur, vidéos et photos

6. Demande d'échantillons

Les deux types de compte ont la possibilité de remplir un formulaire de demande d'échantillons concernant les produits affichés dans le catalogue.

a) Formulaire demande d'échantillons

7. Récapitulatif des demandes d'échantillons

Un tableau permet de visualiser toutes les demandes d'échantillons effectuées dans le passé, avec l'identification des produits, leur quantité ainsi que le statut de la demande.

a) Page récapitulative des demandes d'échantillons

8. Récapitulatif des commandes

Un tableau permet de visualiser toutes les commandes effectuées dans le passé, avec possibilité de visualiser le bon de commande en format PDF et quelques informations générales, quantité, produit, montant total et statut de la commande.

a) Page récapitulative des commandes

b) Page visualisation du pdf du bon de commande

9. Page promotionnelle

Perfect wines pourra présenter dans cette page une campagne promotionnelle personnalisée pour l'entreprise.

C. Back office Administration Perfect Wines

1. Formulaire de connexion

Un simple formulaire de connexion email / password permet d'accéder à l'espace back office de Perfect wines. Cette connexion contrôle en bdd l'existence du compte ainsi que la véracité de son password crypté par password hash et met en session l'ID, ainsi que son email.

a) Formulaire de connexion

Il n'y a pas de distinction de rôles au niveau de l'équipe de Perfect wines.

2. Formulaire d'inscription

Chaque administrateur existant, a la possibilité d'enregistrer un nouvel administrateur. La création de compte s'effectue donc par cooptation. L'administrateur et ses employés sont au nombre de 4, il n'est donc pas nécessaire de complexifier la gestion des droits à ce niveau.

a) Formulaire d'inscription

3. Modification compte administrateur

Chaque administrateur a accès à un formulaire de modification de son compte.

a) Formulaire de modification de profil

4. Formulaire d'inscription entreprise importatrice

Chaque administrateur a accès à un formulaire de création de compte entreprise.

a) Formulaire d'inscription entreprise

5. Formulaire d'inscription compte Prime importer

Chaque administrateur a accès à un formulaire de création de compte prime importer, considéré comme représentant de l'entreprise.

a) Formulaire d'inscription de compte Prime importer

6. Tableau récapitulatif entreprises clientes et fiche entreprise

Chaque administrateur a accès au tableau récapitulatif des entreprises clientes avec fonction Voir la fiche entreprise et CRUD sur cette entreprise.

a) *Tableau récapitulatif entreprises*

b) *Fiche entreprise avec CRUD*

7. Tableau récapitulatif assistants du client

Chaque administrateur a accès au tableau récapitulatif des assistants de l'entreprise cliente avec fonction Voir la fiche assistant et CRUD sur ces fiches.

a) *Tableau récapitulatif assistants entreprise*

b) *Fiche assistant entreprise cliente avec CRUD*

8. CRUD fournisseurs

Création, modification des entreprises productrices de vins avec upload d'images et de vidéos.

a) *Formulaire d'inscription entreprise*

b) *Formulaire de modification entreprise*

c) *Formulaire upload images, Vidéos*

d)

9. CRUD produits

Création, modification des produits avec upload d'images.

a) *Formulaire d'inscription produit*

b) *Formulaire de modification produit*

c) *Formulaire upload images*

d) *Tableau récapitulatif produits*

10. CRUD offre d'échantillons

Création, modification des offres d'échantillons.

a) *Formulaire d'inscription offre*

b) *Formulaire de modification offre*

c) *Tableau récapitulatif offre et demande échantillons avec statut*

11. CRUD bons de commandes

Création, modification des fiches bons de commande.

a) *Formulaire d'inscription bon*

b) *Formulaire de modification bon*

c) *Formulaire upload PDF Bon de commande*

d) *Tableau récapitulatif bons de commande avec statut*

12. CRUD promotion

Création, modification des promotions avec upload d'images.

a) Formulaire d'inscription promotion
b) Formulaire de modification promotion
c) Formulaire upload images
d) Tableau récapitulatif promotions

D. BLOG

Le blog se divise en trois parties :

1. Partie publique

Une partie publique pour visualiser la page index du blog, la page de chaque article avec ses commentaires de la part des assistants entreprises clientes.

a) Page index blog
b) Page article blog avec visualisation commentaires clients

2. Partie entreprises clientes

Identique à la partie publique avec toutefois la possibilité d'ajouter des commentaires dans la page article.

a) Page article blog avec visualisation et ajout commentaires

3. CRUD Blog perfect wines

La partie création, modification, suppression des articles avec upload d'images sous la responsabilité des administrateurs Perfect Wines.

Les administrateurs ont aussi le rôle de contrôler dans un tableau récapitulatif les commentaires.

a) Formulaire création article avec upload image
b) Formulaire de modification article
c) Tableau récapitulatif articles avec fonction Voir
d) Tableau récapitulatif commentaires avec CRUD

V. Spécifications techniques

A. Technologies utilisées pour la partie back-end :

- Framework Codeigniter 3
- Php 7, PDO , POO, SQL (Looping et My SQL Workbench pour la realization des MCD, MLD et MPD)

B. Technologies utilisées pour la partie front-end :

- HTML 5
- Framework Bootstrap CSS 5
- Librairie JQuery
- Javascript

C. Environnement de développement :

- Visual code, Atom comme IDE (environnement de développement intégré),
- Google Chrome pour la visualisation et l'inspection,
- Environnement serveur de notre hébergeur Hostinger, avec Cpanel et phpMyadmin, chaque modification étant directement déployée en pré-production sur le serveur après commit git.

D. Environnement utilisée pour versionning et la collaboration :

- Github. com pour le versionning,
- GitKraken pro pour effectuer les commandes git en interface,
- l'auto déploiement git de l'hébergeur Hostinger.com avec clef ssh,
- Trello pour le suivi des taches et Slack pour une meilleure communication et intégration collaborative avec notre camarade Thuc-nhi, partie prenante de ce projet.

VI. Architecture du projet

Le projet étant développé avec le *framework* Codeigniter 3, nous utilisons une architecture *Pattern MVC (Model-View-Controller)*. Le *controller* comme chef d'orchestre définit des variables, les contrôle en fonction des actions de l'utilisateur, contient des fonctionnalités en POO comme méthodes, envoie des requêtes au *Model* qui dialogue avec la base de données et lui renvoie le résultat.

Le contrôleur traitera ce résultat pour avancer dans sa procédure et générera comme résultat une vue avec des données qui seront affichées selon l'architecture html/css de celle-ci.

Codeigniter utilise la programmation orientée objet (POO) et le Php Data Object (PDO) qui constitue une couche d'abstraction qui intervient entre l'application PHP et un système de gestion de base de données (SGDB).

Outre les fichiers de configuration de l'environnement, de la base de données, des routes et de l'*autoloader*, Codeigniter met à notre disposition tout un ensemble de fonctionnalités déjà développées dans les librairies *form_validation, session, database, email* et *table*. Ces librairies sont directement appelées dans notre *autoloader*.

Nous utilisons aussi les *helpers url, file* et html dont nous allons décrire les fonctionnalités en traduisant les ressources disponibles dans la documentation.

A. Recherche effectuée à partir d'un site anglais (documentation codeigniter) sur les helpers. (https://codeigniter.com/userguide3/general/helpers.html)

1. Texte original:

Helper Functions

Helpers, as the name suggests, help you with tasks. Each helper file is simply a collection of functions in a particular category. There are URL Helpers, that assist in creating links, there are Form Helpers that help you create form elements, Text Helpers perform various text formatting routines, Cookie Helpers set and read cookies, File Helpers help you deal with files, etc.

Unlike most other systems in CodeIgniter, Helpers are not written in an Object Oriented format. They are simple, procedural functions. Each helper function performs one specific task, with no dependence on other functions.

CodeIgniter does not load Helper Files by default, so the first step in using a Helper is to load it. Once loaded, it becomes globally available in your controller and views.

(…)

Form Helper

The Form Helper file contains functions that assist in working with forms.

Escaping field values

You may need to use HTML and characters such as quotes within your form elements. In order to do that safely, you'll need to use common function html_escape().

If you use any of the form helper functions listed on this page, the form values will be automatically escaped, so there is no need to call this function. Use it only if you are creating your own form elements.

2. Traduction :

« Les *helpers* comme le suggère le nom, vous aident dans vos tâches. Chaque fichier *helper* est simplement une collection de fonctions dans une catégorie particulière. Il y a les URL *helpers* qui vous assiste dans la création de links, les *helpers form* qui vous aide à créer des éléments de formulaire, des *helpers Text* qui appliquent différentes routines de formatage de texte, les

cookie helpers qui construisent et lisent les *cookies*, les *file helpers* qui vous aide à gérer les fichiers, etc…

A la différence de la plupart des autres systèmes de Codeigniter, les *helpers* ne sont pas écrits en POO. Ceux sont simplement des fonctions procédurales. Chaque fonction du *helper* applique une tâche spécifique, sans dépendance avec les autres fonctions.

Codeigniter ne charge pas les fichiers *Helper* par défaut, donc le premier pas pour utiliser un *helper* et de le charger. Une fois chargé, il devient globalement disponible dans vos contrôleurs et vos vues.

Form Helper :

Le fichier *helper Form* contient des fonctions qui vous assiste en travaillant avec les formulaires.

Echappement des champs *values* : Vous pouvez avoir besoin d'utiliser html et les guillemets dans vos éléments *form*. Pour pouvoir faire cela surement, vous aurez besoin d'utiliser la fonction commune *html_escape*.

Si vous utilisez n'importe laquelle des fonctions *helper* de cette page, les *values* du formulaire seront automatiquement échappées donc il n'est pas nécessaire d'appeler cette fonction. Utilisez-la simplement si vous créez vos propres éléments de formulaire. »

VII. Réalisations

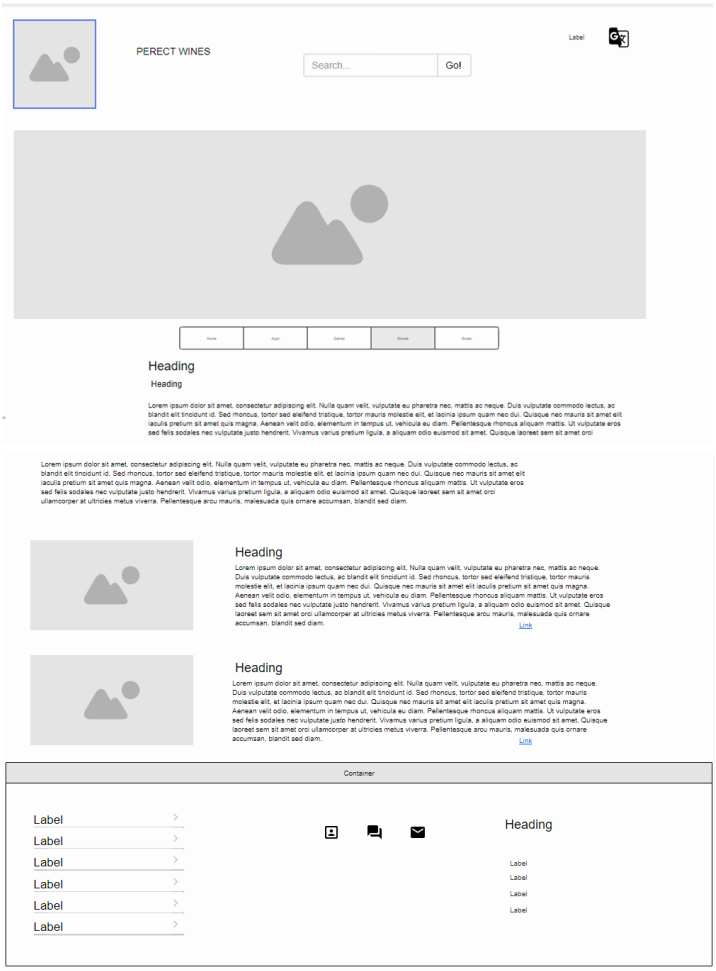
A. Maquettage

A la suite de plusieurs réunions de concertation avec les représentants de Perfect wines, dont le principal actionnaire Christophe Castagne en visite en France, nous avons identifié les différentes fonctionnalités qui devaient être développées dans le site. En ayant été séduit par l'image général de notre projet boutique « Panama Hats » Mr Castagne nous a demandé de respecter cette nouvelle image pour son propre site.

Nous devions donc réaliser le design d'interface de notre application web dans une démarche de conception itérative en passant successivement de simples schémas à la main (sketchs) à une zonification, puis des wireframes, mockups et enfin quelques prototypes.

Etant limité par la version gratuite de moqups.com, qui nous oblige à générer seulement un maximum de 200 éléments nous avons réalisé chaque page représentative, puis effectué avec l'outil capture d'écran de Windows 10 une sauvegarde de celle-ci.

Nous disposions donc de maquettes mockups pour chaque page, sans design graphique, sans dimension esthétique et sans plus de détails sur les éléments de l'interface utilisateur.



La seconde étape consistait à réaliser un prototype de nos pages en révélant l'identité et les fonctionnalités de base de l'interface finale pour un meilleur impact visuel.

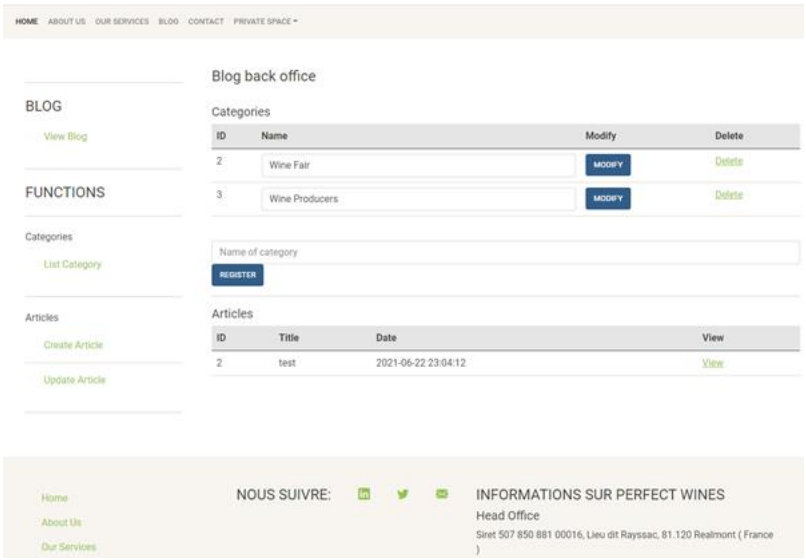
A la différence de l'exercice boutique cité au-dessus et par nécessité de préparer les dossiers de projet simultanément au développement de ce celui-ci, nous avons décidé de présenter un espace de pré-production directement accessible aux parties intéressées.

L'option n'a été rendue possible que par l'adoption précoce d'un hébergement définitif chez Hostinger.com avec interface de déploiement automatisée directement connectée à notre *repository* Git par clé ssh et webhook.

Cette méthode n'est applicable dans notre cas que grâce au caractère exceptionnel de ce projet : Mr Castagne a totalement intégré l'idée qu'il s'agit d'un exercice de mise en pratique de nos compétences professionnelles, et nous donne carte blanche pour adapter ses nécessités à notre vision des choses.

Il n'intervient pas dans la conception des pages au niveau graphique, même s'il dispose d'un accès en temps réel à leur espace de développement.

Voici par exemple une capture de l'étape précoce de pré production de la page back office administration du blog, déjà entièrement fonctionnelle et responsive, mais sans banner :



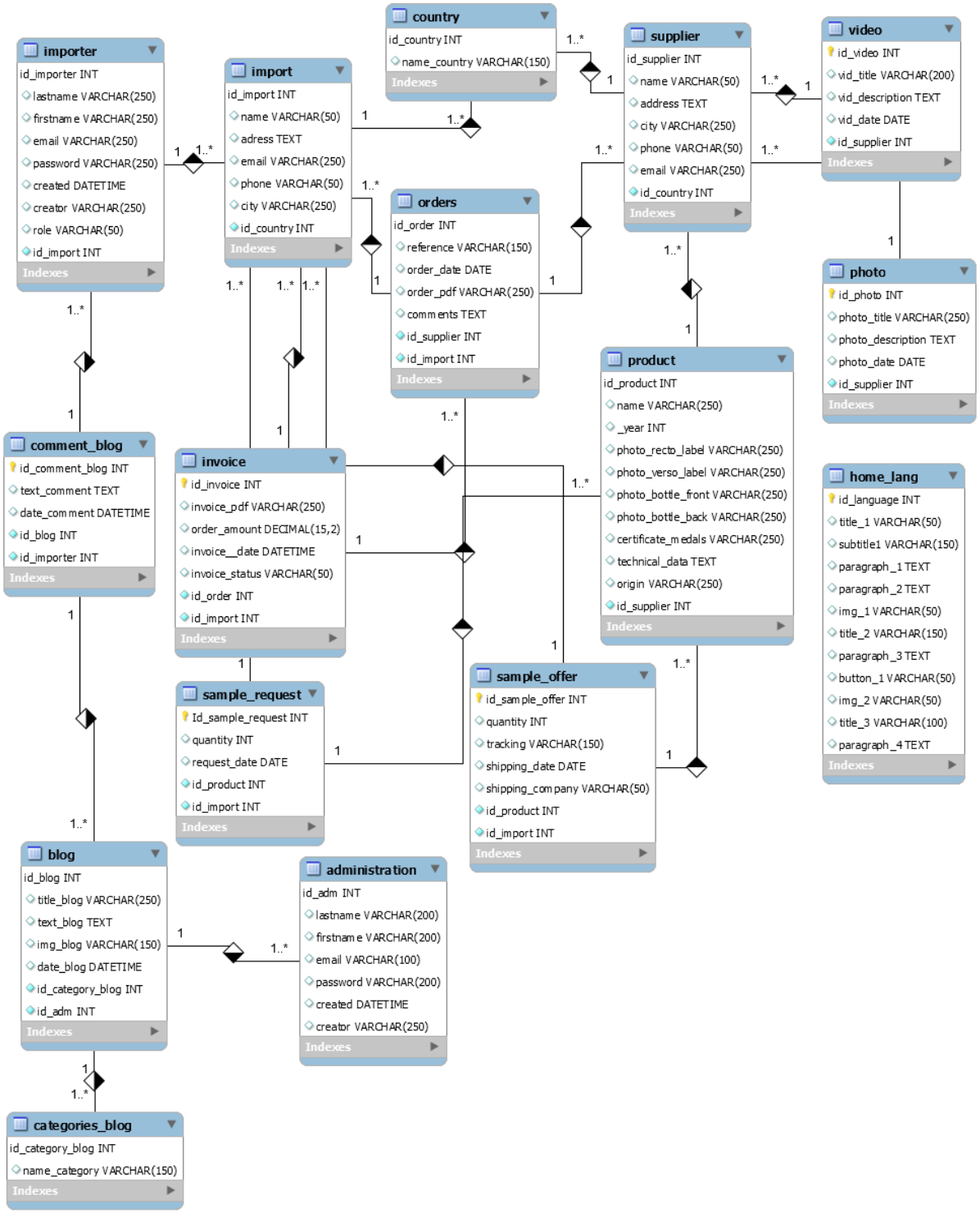
Nous avons conscience que cette liberté d'action n'est pas de mise dans une relation commerciale habituelle, les prototypes servant en quelque sorte de référence contractuelle pour l'objectif final à atteindre.

Une fois établis dans la concertation, le développeur échappe à toute tentative de modifications impromptues au cours de leur réalisation, à l'exception d'une nécessité impérieuse de modification ou de création de nouvelles fonctionnalités qui demanderont à leur tour des réunions de concertation et de prototypage.

La visualisation de cet espace de pré-production par le grand public n'est actuellement pas permise, étant protégé par un simple formulaire de connexion opérationnel, vérifiant de manière sécurisée les identifiants de l'équipe de Perfect wines en base de données.

B. Conception de la base de données

Au regard de l'ensemble des fonctionnalités requises par l'entreprise, nous avons développé la base de données suivante représentée dans un schéma de Modèle physique de données :



Le modèle conceptuel de données et le modèle logique réalisés avec le logiciel looping se trouvent en Annexes.

Ce schéma indique les tables correspondantes aux entités identifiées dans le MCD, avec leurs colonnes et le typage de chacune d'entre-elles. Il nous montre aussi les clefs étrangères qui traduisent entre chaque table une association de cardinalité 1-N dans le modèle conceptuel.

Cette base s'organise autour de trois acteurs : la table import qui représente les entreprises importatrices clientes et la table importer comprenant l'ensemble des employés de ces entreprises qui participent à notre système, identifiés par une id, l'id de leur entreprise en clé étrangère et leur rôle, avec d'autres attributs plus personnels.

La table supplier qui représente les entreprises productrices de vins à laquelle sont reliées les tables product, video et photo.

Entre ces deux acteurs apparait la table orders qui contient toutes les commandes effectuées entre ces deux groupes d'entreprises. Elle n'est pas reliée à la table product car il ne s'agit pas d'un système d'achat online, ces commandes sont juste représentées par les pdf des bons de commande.

Par contre les tables sample_request et sample_offer sont directement reliées à la table product car elles identifient le produit requis (un seul produit par demande d'échantillon).

Dans le cas où Perfect wines voudrait modifier son système sur les échantillons en permettant des envois groupés nous devrions changer les cardinalités à N-N dans le modèle conceptuel et obtenir des tables de transition dans le modèle physique.

L'administration perfect wines apparait dans le système pour deux fonctionnalité, la connexion et la création, modification des catégories et articles de blog.

Les employés des entreprises importatrices dans notre système, alimentent la table comment_blog qui est reliée à la table blog contenant les articles de blog, par la clé étrangère id_blog.

Enfin chaque page disposera de sa table _lang comme notre exemple home_lang contenant toutes les versions de langue de la page, c'est-à-dire pour l'instant anglais et japonais.

Notre base de données est déjà configurée dans l'espace serveur de pré production, elle est donc alimentée en temps réel par nos opérations et nos tests.

Nous avons créé un ensemble de comptes Test pour évaluer le fonctionnement de l'ensemble.

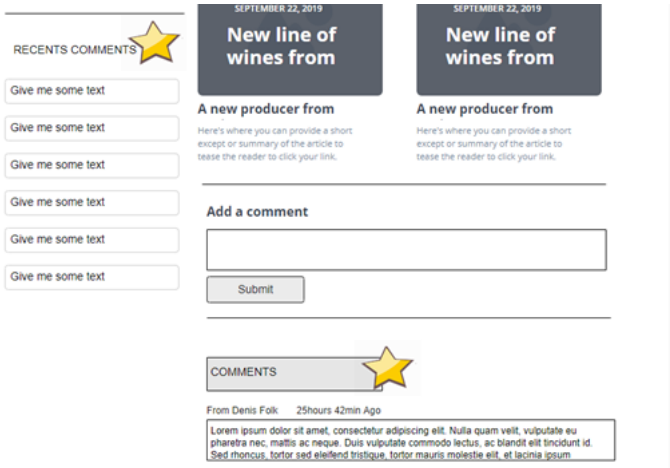
Seul la table administration contient actuellement des personnes réelles, les assistants et associés de Perfect wines.

VIII. Extraits de code significatifs

A. Partie front-end : réalisation de la page article de blog avec ajout de commentaires pour les importateurs.

1. Code MVC Codeigniter pour afficher la page article de blog

Notre *mockup* sur la page Article de Blog contient deux zones destinées à afficher les commentaires par date de publication, du plus récent au plus ancien. (Cf. étoiles) L'une se situe dans la partie aside gauche de la page, l'autre en dessous du bloc article, qui contient un affichage de l'article (image, texte, auteur, date) ainsi que deux suggestions d'articles dans la même catégorie.



Elle contient aussi une zone *Add comment* qui n'apparait que lorsque la session contient l'email et le rôle de l'importateur qui s'est connecté. Cette zone n'apparait donc pas pour le public en général, simplement du fait que Perfect wines ne dispose pas temps pour effectuer des opérations de modération des commentaires. Nous verrons ensuite comment cela est mis en place dans la vue.

Le contrôleur Blog contient la fonction article() qui utilise le helper url et sa fonctionnalité uri pour identifier le troisième segment de l'url contrôleur/méthode/paramètre. Ce segment correspond à l'id_blog dans la table blog qui contient en ligne chaque article et en colonnes chaque attribut.

Avec cette id_blog nous pouvons constituer un tableau $data['row'] en faisant appel à la méthode singleArticle du model Blog_model, qui contiendra les attributs de l'article.

```
$data['row'] =$this->Blog_model->singleArticle($id_blog);
```

La méthode singleArticle du model effectue une requête sur la table blog avec une jointure sur la table categories_blog pour ajouter le nom de la catégorie par référence à son id_category_blog contenue dans les deux tables, une jointure avec la table administration pour afficher le nom et prénom de l'auteur :

```
$sql = "SELECT * FROM blog JOIN categories_blog ON blog.id_category_blog = cat
egories_blog.id_category_blog JOIN administration ON blog.id_adm=administratio
n.id_adm WHERE id_blog = ?";
```

Le controller se charge alors d'envoyer ces données dans la vue correspondante à l'article.

```
 $this->load->view('blog/article', $data) ;
```

Nous appliquons une méthode similaire controller/model/vue pour charger une liste de commentaires correspondants à cet article.

La modalité qui suggère deux articles qui partagent la même id_category_blog est particulière.

Nous devons extraire de $data['row'] la valeur de id_category_blog pour cet article :

```
$id_category_blog = $data['row']['id_category_blog'];
```

Puis nous lançons une requête dans le model qui retournera deux articles non identiques à celui-ci mais qui partagent la même id_category_blog.

```
$data['suggest']=$this->Blog_model->suggestArticles($id_category_blog, $id_blog);
```

La requête dans le modèle est la suivante :

```
        $sql = "select * from blog WHERE id_category_blog = ? AND id_blog <> ?
 ORDER by date_blog DESC LIMIT 2";
 $query=$this->db->query($sql, array($id_category_blog, $id_blog));
```

Codeigniter nous permet pour *bind* deux paramètres de faire un *array* qui contient les deux valeurs de variables.

$data contient donc maintenant un ensemble de données qui nous permettra d'afficher chaque élément requis dans la vue, les attributs de l'article, ainsi que deux articles suggérés et dans la colonne de gauche une liste de catégories, une liste générique de commentaires et les derniers articles publiés.

```
$data['categ']=$this->Blog_model->listCateg();
$data['comments']=$this->Blog_model->listComments($id_blog);
$data['lastComments']=$this->Blog_model->lastComments();
$data['suggest']=$this->Blog_model->suggestArticles($id_category_blog, $id_blog);
$data['last']=$this->Blog_model->lastArticles();
```

L'ensemble de ces affichages à gauche dans le **aside** sont sujets au clic afin d'obtenir de nouvelles vues :

- Cliquer sur un élément de la liste **Categories** nous envoie à une vue qui liste les articles de blog qui partagent cette catégorie ;
- Cliquer sur le titre en bas de chaque commentaire dans la rubrique **Last Comments**, nous envoie à la vue article de ce titre ;
- Cliquer sur le titre ou la photo des articles dans la rubrique **Last Articles**, nous envoie aussi à la vue article de ce titre.

2. Partie du code pour ajouter un commentaire

Sous l'affichage du bloc article et articles suggérés apparait une section **Publish a comment** seulement lorsque la personne est connectée et que sa session contient un id_importer.

```
<?php
        if(isset($this->session->userdata['id_importer'])){
            echo form_open('Blog/addComment', 'class="contact_form" ');
```

Le submit de ce formulaire insère le commentaire dans la table correspondante et recharge l'intégralité de la page ce qui nous permet de voir le dernier commentaire apparaître dans la colonne de gauche sous la rubrique générique **Last Comments** , puisqu'elle liste l'ensemble des commentaires ajoutés du plus récent au plus ancien, en stipulant le titre de l'article et en href son $id_blog pour nous renvoyer sur la page article correspondante.

Le commentaire apparait aussi dans la rubrique **Comments** au centre puisqu'il correspond au dernier commentaire ajouté pour cet article.

B. Partie front-end : réalisation de la fonction search de blog avec Jquery et Ajax.

Chaque page de blog dispose dans sa colonne de gauche d'une fonctionnalité ***Search*** qui permet d'effectuer une recherche sur les titres d'articles du blog dans la table blog.

1. Partie Javascript avec librairie Jquery et Ajax

Le script que nous avons intégré directement dans chaque page de la section blog se divise en deux parties :

Lorsque le ***DOM*** de la page est entièrement chargé, Jquery écoute l'***input*** de notre champ ***search*** identifié par ***l'id searchtext*** avec la méthode ***keyup***. Si l'utilisateur introduit avec son clavier un caractère ou une chaine de caractères dans ce champ, la variable ***search*** prendra

la valeur de ce contenu et lancera la fonction ***load_data*** avec comme paramètre cette variable.

```
$('#search_text').keyup(function(){
 var search = $(this).val();
 if(search != '')
 {
  load_data(search);
 }
```

Jquery utilise alors la méthode ajax pour envoyer à notre ***controller*** Blog /méthode ***fetch*** un ***post*** ayant pour valeur notre variable ***search***. Si la réponse est un succès nous chargerons dans la ***div*** identifiée par ***l'id result*** le résultat de notre requête, formaté en html par notre ***controller***.

```
function load_data(query)
{
 $.ajax({
  url:"<?php echo base_url(); ?>Blog/fetch",
  method:"POST",
  data:{query:query},
  success:function(data){
   $('#result').html(data);
  }
 })
}
```

2. Partie Controller, Model avec Codeigniter

Dans la fonction publique ***fetch()*** , nous contrôlons si la variable envoyée par Ajax n'est pas vide et dans ce cas nous appelons la méthode de notre ***model*** pour effectuer la requête en base de données.

```
$this->db->select("*");
            $this->db->from("blog");
            $this->db->like('title_blog', $query,'after');
```

Codeigniter remplace l'expression $query% (le pourcentage ici indique ***Like*** n'importe quel caractère après la chaine de caractère $query) par l'option **after**. Si nous cherchions n'importe quel caractère avant la chaine $query nous devrions utiliser l'option **before**.

Si le résultat de cet requête n'est pas vide, nous composons par concaténation un format de sortie en liste avec un ***link*** dont le ***href*** se dirigera vers le contrôleur Blog méthode article paramètre ***id_blog*** pour afficher la page de cet article de blog et le titre du blog.

La boucle **foreach** nous permet de parcourir tous les résultats envoyés par notre ***model***.

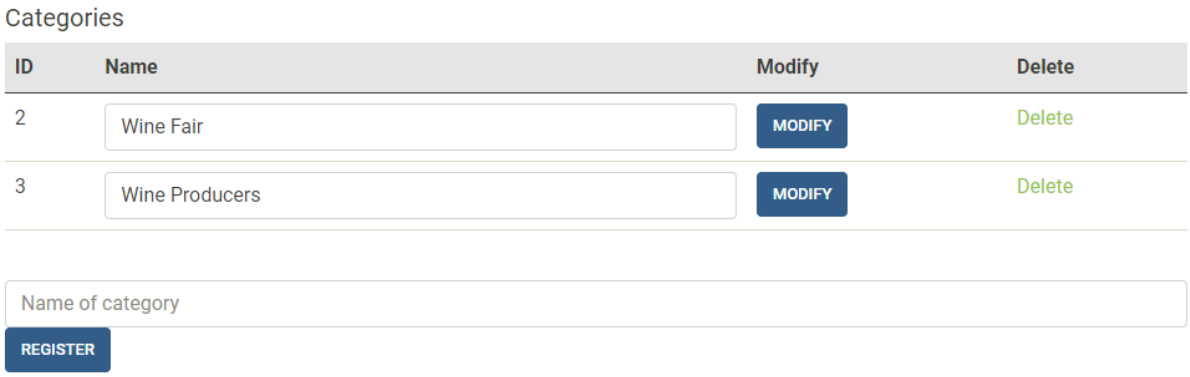
Si le résultat est vide nous affichons simplement ***No data found***.

C. Partie Back-end : Elaborer et mettre en œuvre des composants dans une application de gestion de contenu.

Les administrateurs de Perfect wines ont la possibilité, après connexion d'accéder dans leur backOffice à un back-end dédié à la gestion de la partie Blog.

Les fonctions de CRUD sont appliquées sous forme de tableaux aux catégories ainsi qu'aux articles.

1. Pour les catégories :



L'architecture MVC de codeigniter structure le code de la manière suivante :

Dans notre ***controller*** Blog, la fonction publique backBlog() vérifie tout d'abord que la personne qui veut accéder à cette partie soit effectivement un administrateur avec l'aide de la librairie session.

```
if($this->session->has_userdata('email') && $this->session->userdata('role')=='adm' )
```

Si c'est le cas notre ***controller*** demande à notre ***Model*** Blog_model de lancer deux requêtes sur la base de données pour obtenir deux listes : la liste des catégories et la liste des articles de blog.

```
$data['categ']=$this->Blog_model->listCateg();
$data['article']=$this->Blog_model->listArticle();
```

Le code du model est structure de cette manière :

```
public function listCateg()
    {
        $query=$this->db->query(
            "select * from categories_blog"
          );
          return $query->result();
```

Le ***controller*** envoie ces données avec un $data comme paramètre de la vue backBlog.

```
$this->load->view('adm/backBlog', $data) ;
```

La vue backBlog à l'aide d'un ***foreach*** lit toutes les valeurs du tableau correspondant à catégorie et affiche dans une table chaque catégorie en formulaire pour directement permettre la modification.

Ce formulaire est dirigé vers le ***controller*** Blog méthode updateCateg avec l'id catégorie en paramètre.

```
echo form_open('Blog/updateCateg/'.$row->id_category_blog, 'class="contact_form" ');
```

Nous ajoutons à coté de ce formulaire la fonction ***delete*** qui permet d'effacer une catégorie, en fonction de son id, comme paramètre.

En dessous de cette table nous avons placé un simple formulaire pour enregistrer en base une nouvelle catégorie.

2. Pour les articles de blog :

De la même manière nous présentons en table le titre et la date de création des articles de blog.

Articles

ID	Title	Date	View
4	test3	2021-06-25 10:37:12	View
3	essai 2	2021-06-23 22:53:38	View
2	test	2021-06-22 23:04:12	View

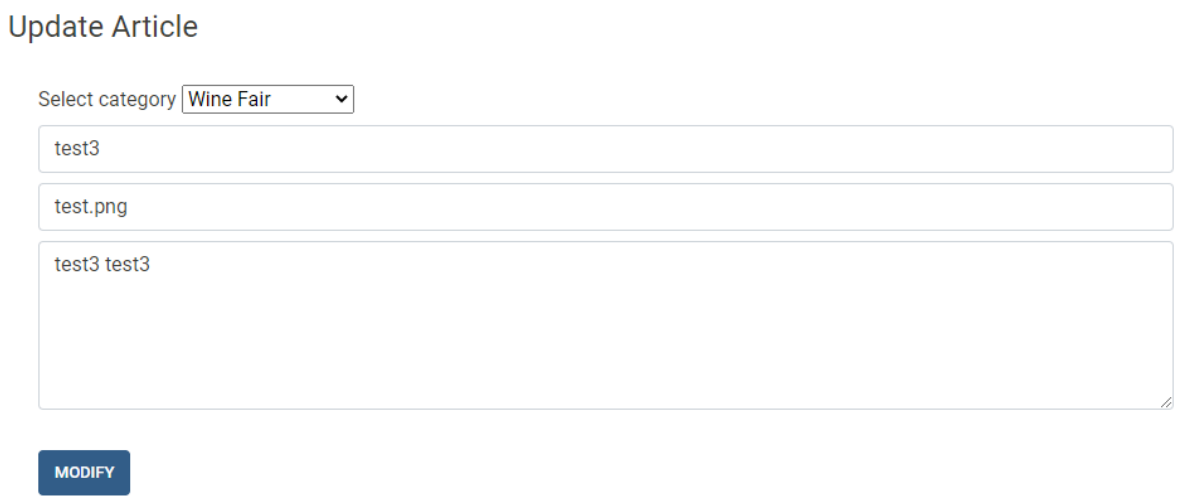
La fonction ***view*** nous dirige vers le ***controller*** Blog méthode ***update_article*** paramétre id_blog qui a pour fonction de générer une vue de l'article de blog dans un formulaire.

Nous prenons ici aussi une mesure de sécurité en contrôlant l'existence de la ***session userdata(email)*** et le fait que la ***session userdata(role)*** soit égale à ***adm***.

Le ***controller*** effectue une nouvelle requête à la base dans *le* ***model*** Blog pour reconstituer un $data contenant les attributs de l'article et une liste de catégories qui sera chargé dans chaque ***value*** des ***inputs***.

La liste de catégories est chargée dans un <select> tout en identifiant une <option> avec la valeur de catégorie existante de l'article suivi de ***selected***.

```
<select name="id_category_blog" id="Select1">
<?php
echo '<option value="'.$row['id_category_blog'].'" selected>'.$row['name_categ
ory'].'</option>';
foreach($categ as $category){
if($category->id_category_blog != $row['id_category_blog']){
echo '<option value="'.$category->id_category_blog.'">'.$category-
>name_category.'</option>';
}
} ?>
</select>
```



Ce formulaire est dirigé vers le ***controller*** Blog méthode ***modify_article***, sans paramètre puisque l'***id_blog*** est indiqué dans un ***input type hidden***.

Chaque ***input*** du formulaire est accompagné d'un ***required***, nous n'appliquons donc dans le ***controller*** qu'une mesure de sécurité de la librairie ***security : xss_clean***.

```
$text_blog = $this->security->xss_clean($this->input->post('text_blog'));
```

CodeIgniter est livré avec un filtre de prévention ***cross-site scripting***, qui recherche les techniques couramment utilisées pour déclencher JavaScript ou d'autres types de code qui tentent de détourner des cookies ou de faire d'autres choses malveillantes. Si quelque chose d'interdit est rencontré, il est rendu sûr en convertissant les données en caractères.

Nous constituons un ***array*** avec ces variables et le ***controller*** dialogue avec son ***model*** pour insérer ces nouvelles valeurs dans la table blog.

```
 $updateArticle = $this->Blog_model->updateArticle($article, $id_blog);
```

En cas de succès nous rechargeons la page backblog, en cas d'échec nous retournons au début de l'opération affichage du formulaire.

Cette partie Back-end contient aussi une méthode de création d'article qui fonctionnera selon le même principe.

D. Partie back-end clients importateurs

Comme nous l'avions expliqué dans la partie fonctionnalité, la partie back-end des clients importateurs contient deux versions :

- Une partie ***private space*** réservée au ***Prime importer***, représentant de l'entreprise qui comprend toutes les fonctionnalités CRUD pour modifier, créer, effacer les assistants qui ont accès au système, ainsi qu'une fonctionnalité pour mettre à jour les informations de son entreprise en cas de besoin.
- Une partie ***private space*** pour les assistants ***importer*** qui affiche la liste des assistants, les attributs de son entreprise, ainsi que son profil.
  La seule fonctionnalité de modification dans cette partie concerne son propre profil.

C'est lors de la connexion que la version à afficher sera déterminée par le rôle de l'utilisateur contenu dans la variable $getUserDetails['role'].

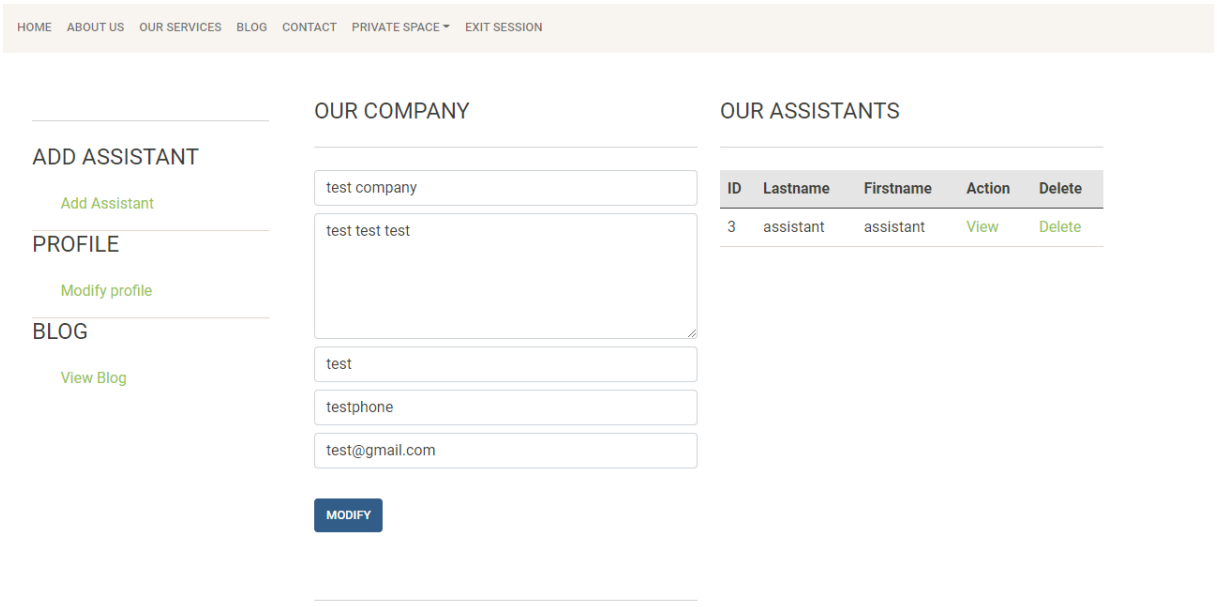
```
if($getUserDetails['role']=='importer'){
                    redirect('Importers/privateSpace');
                   }else{redirect('Importers/privateSpacePrime');}
```

Cet espace reste privé et cloisonné des autres espaces importateurs, donc des autres entreprises car lors de cette même connexion nous sauvegardons en session l'id_import de l'entreprise à laquelle appartient cet utilisateur.

```
$this->session->set_userdata('id_import', $getUserDetails['id_import']);
```

Nous vérifions dans le ***controller*** de création de cette ***view*** si l'email existe en session et si son rôle est bien ***primeImporter*** , puis nous récupérons les variables contenues en session pour préparer les tableaux de valeurs nécessaires aux CRUD :

```
$id_import= $this->session->userdata['id_import'];
$id_importer=$this->session->userdata['id_importer'];
$data['company']=$this->Import_model->ourCompany($id_import);
$data['profile']=$this->Import_model->singleImport($id_importer);
$data['assistants']=$this->Import_model->ourAssistants($id_import);
```



Les informations de son entreprise sont affichées directement dans un formulaire au cas où il faudrait apporter des modifications.

La liste des assistants s'affiche dans un tableau avec une fonction ***view*** qui lui donnera accès à un formulaire de modification de ce profil.et une fonction ***delete*** qui supprime le profil de son assistant.

Il dispose par ailleurs d'une fonctionnalité dans la colonne gauche de l'***aside*** pour modifier son propre profil.

Enfin dans cette même colonne il dispose d'une fonction Add assistant qui lui permet d'accéder à un formulaire pour ajouter à son équipe un nouvel employé de l'entreprise.

Le private space de ces employés est radicalement différent puisque les données de l'entreprise, ainsi que la liste des assistants ne sont pas modifiables.

Dans ce cas il ne peut que mettre à jour ses propres données. Pour ce faire nous contrôlons dans la fonction updateImporter() que la session id_importer soit identique à l'id_importer du formulaire.

```
If($this->session->userdata('id_importer')==$id_importer)
```

Cette page contient aussi les formulaires pour enregistrer des commandes d'échantillons, une fonctionnalité pour afficher un tableau de toutes ces commandes d'échantillons passées, une fonctionnalité pour afficher un tableau de toutes les commandes de produits passées qui permettra de visualiser sous format pdf chacune de ces commandes.

E. Partie Sécurité : Décrire les mesures de sécurité appliquées au site perfectwines-export.

1. Usage de https :

Le protocole HTTPS (HyperText Transfer Protocol Secure) a été conçu pour pallier le problème de sécurité posé par http, les échanges de données entre un serveur et un client étant ouverts à tous.

Le HTTPS, en réalité, n'est qu'un protocole HTTP auquel on a ajouté une couche sécurisée appelée TLS (Transport Layer Security). Celle-ci agit comme une clé de chiffrement qui crypte les données échangées entre le serveur et le client.

HTTPS garantit aussi l'identité du site web consulté, de façon à être sûr qu'il est bien celui dont l'URL s'affiche.

La mise en pré production précoce de notre site sur l'hébergeur Hostinger.com a été effectuée en respectant cette utilisation du https.

2. Justification de l'usage de Codeigniter :

Ce framework dispose d'un ensemble de méthodes sécurisées pour la plupart contenues dans le helper security et qui permettent de se protéger de malversations.

3. Defined BASEPATH :

D'une manière générale, l'accès direct aux ***controllers*** et ***models*** en entrant le chemin directement dans notre explorateur est interdit.

```
defined('BASEPATH') OR exit('No direct script access allowed');
```

Cette procédure est utilisée pour s'assurer que la demande est passée par index.php dans notre répertoire racine. Ceci pour s'assurer que toutes les classes de base de Codeigniter sont chargées et que certaines variables ont été définies.

4. Protection des formulaires avec le helper form :

Les valeurs des inputs sont automatiquement échappées lorsque nous utilisons ce helper, afin d'éliminer tous les caractères spéciaux html qui pourraient être ajoutés dans les champs du formulaire.

A cela s'ajoute une fonction qui assainit ces valeurs input la méthode xss_clean que nous appliquons dans nos procédures à chaque réception de $_Post.

```
$data = $this->security->xss_clean($data);
```

Nous avons vu (p.25) que CodeIgniter détecte avec ce filtre les techniques couramment utilisées pour déclencher JavaScript ou d'autres types de code qui tentent de détourner des cookies ou de faire d'autres choses malveillantes.

5. Protection contre l'injection de code SQL :

Nos ***models*** sont aussi organisés pour contrer l'injection de code Sql qui occasionne une perte de contrôle sur les données en base, ce qui peut mener à leur exfiltration, altération ou suppression. Nos requêtes sont préparées avec la méthode ***bind parameters*** de codeIgniter dans lesquelles les paramètres sont interprétés indépendamment de la requête elle-même. De cette manière, il est impossible d'effectuer des injections.
Exemple :

```
$sql = "select firstname, lastname, blog.id_blog, title_blog, img_blog, date_b
log from blog JOIN administration on blog.id_adm=administration.id_adm WHERE
id_category_blog = ?  ORDER by date_blog DESC";
        $query=$this->db->query($sql, $id_category_blog);
```

6. Le password_hash:

Nous utilisons lors de l'enregistrement d'un nouvel utilisateur une méthode d'encryptage du mot de passe de cette manière :

```
 $options = array("cost"=>4);
            $hashPassword = password_hash($password,PASSWORD_BCRYPT,$options);
```

Nous vérifions ensuite lors de la connexion le mot de passe en utilisant la méthode ***password_verify***:

```
password_verify($password, $getUserDetails['password'])
```

7. Utilisation de la librairie session :

Enfin nous multiplions les contrôles des éléments que nous avons placés en session à chaque moment opportun pour bloquer l'accès aux utilisateurs qui n'auraient pas les droits requis pour utiliser ces procédures.

```
if($this->session->has_userdata('email') && $this->session-
>userdata('role')=='adm' )
```

8. Intégration du captcha dans nos pages formulaires contact :

Nous devrions mettre en place un captcha dans notre formulaire contact pour éviter l'usage de bots et différencier le vrai du faux traffic.

Nous utiliserons Google Recaptch en nous inscrivant puis en choisissant dans l'onglet admin console la version 3 qui valide via un test les actions de l'utilisateur.

Pour afficher le captcha, nous avons besoin d'ajouter un script recaptcha/script api.js entre les deux balises <head> de notre site internet.

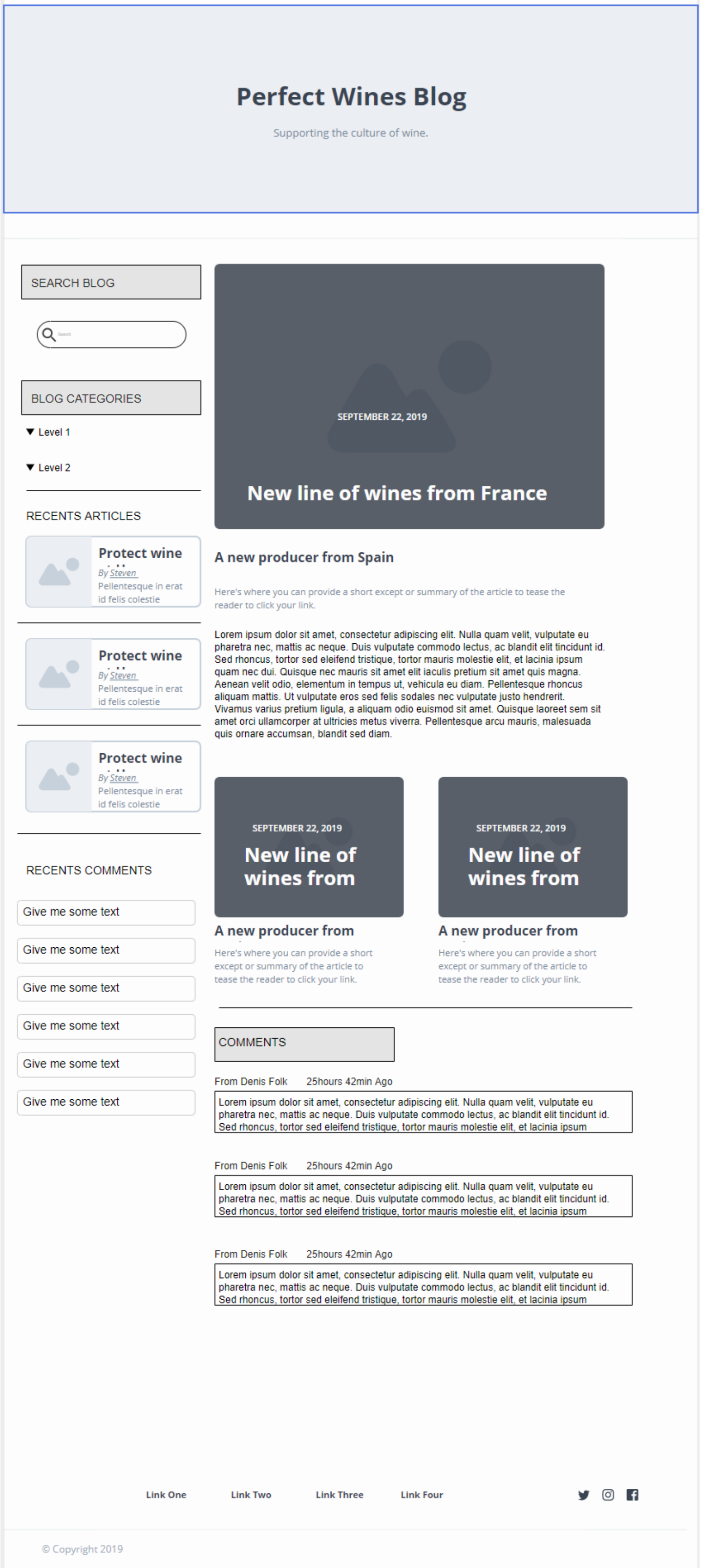
Puis, nous devons insérer une ligne de code à l'endroit où le CAPTCHA doit apparaître. La ligne comprend un espace réservé pour la clé de notre site obtenu au cours de la première étape. Copier/coller la clé entre les guillemets et enregistrer nos modifications.

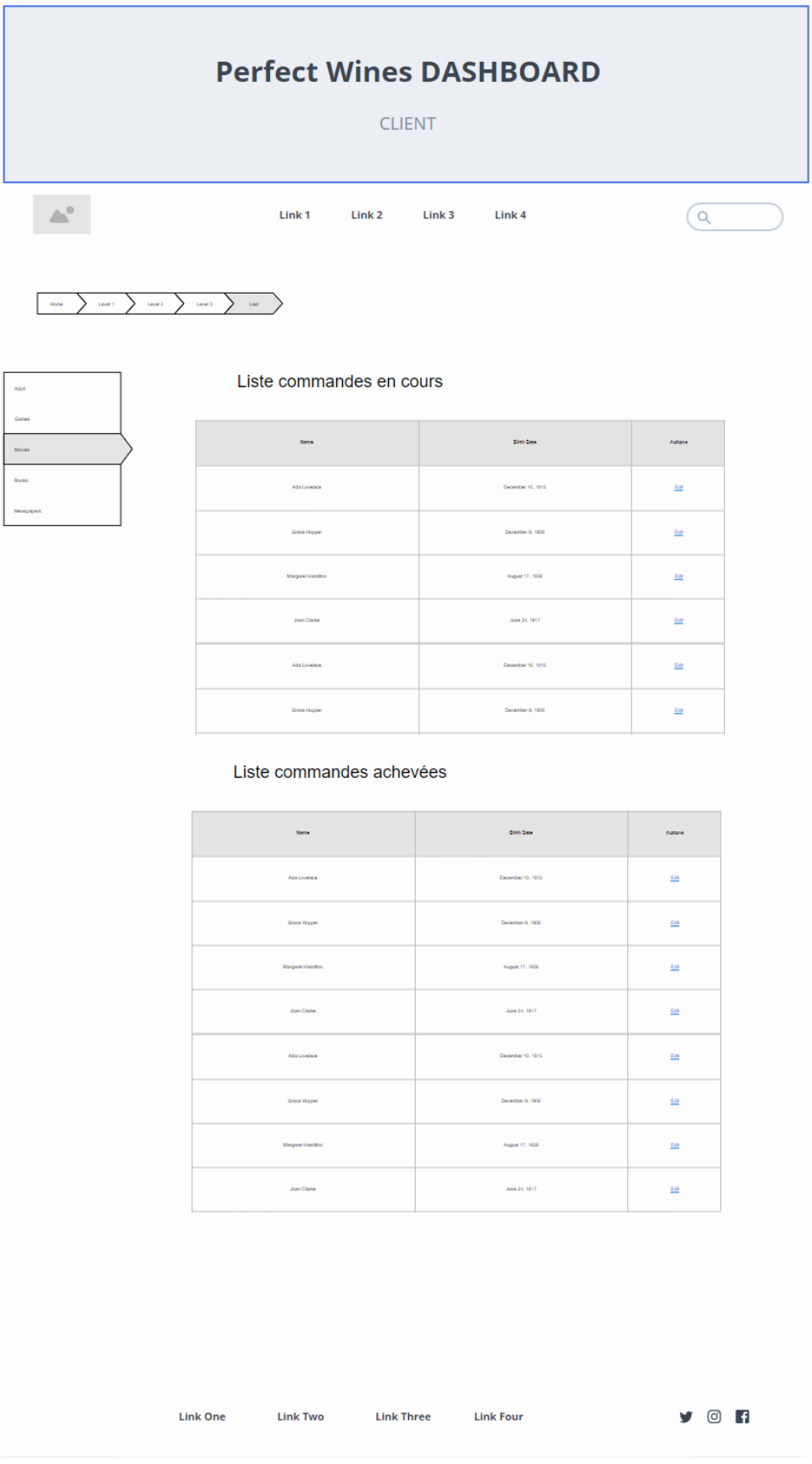
Il est à noter que l'hébergeur de notre client, hostinger.com, exécute BitNinja qui est un système de sécurité de serveur.

BitNinja est une fonctionnalité de sécurité qui détecte les adresses IP avec un comportement suspect, par exemple accéder à la même page Web à intervalles rapprochés, remplir plusieurs fois le mot de passe incorrect, etc. Un tel comportement indique que l'utilisateur utilisant cette adresse IP peut attaquer le site. Bitninja met alors l'adresse IP à un greylist et le visiteur doit se vérifier en remplissant un Captcha. Si le captcha est correct, l'adresse IP est supprimée du greylist.

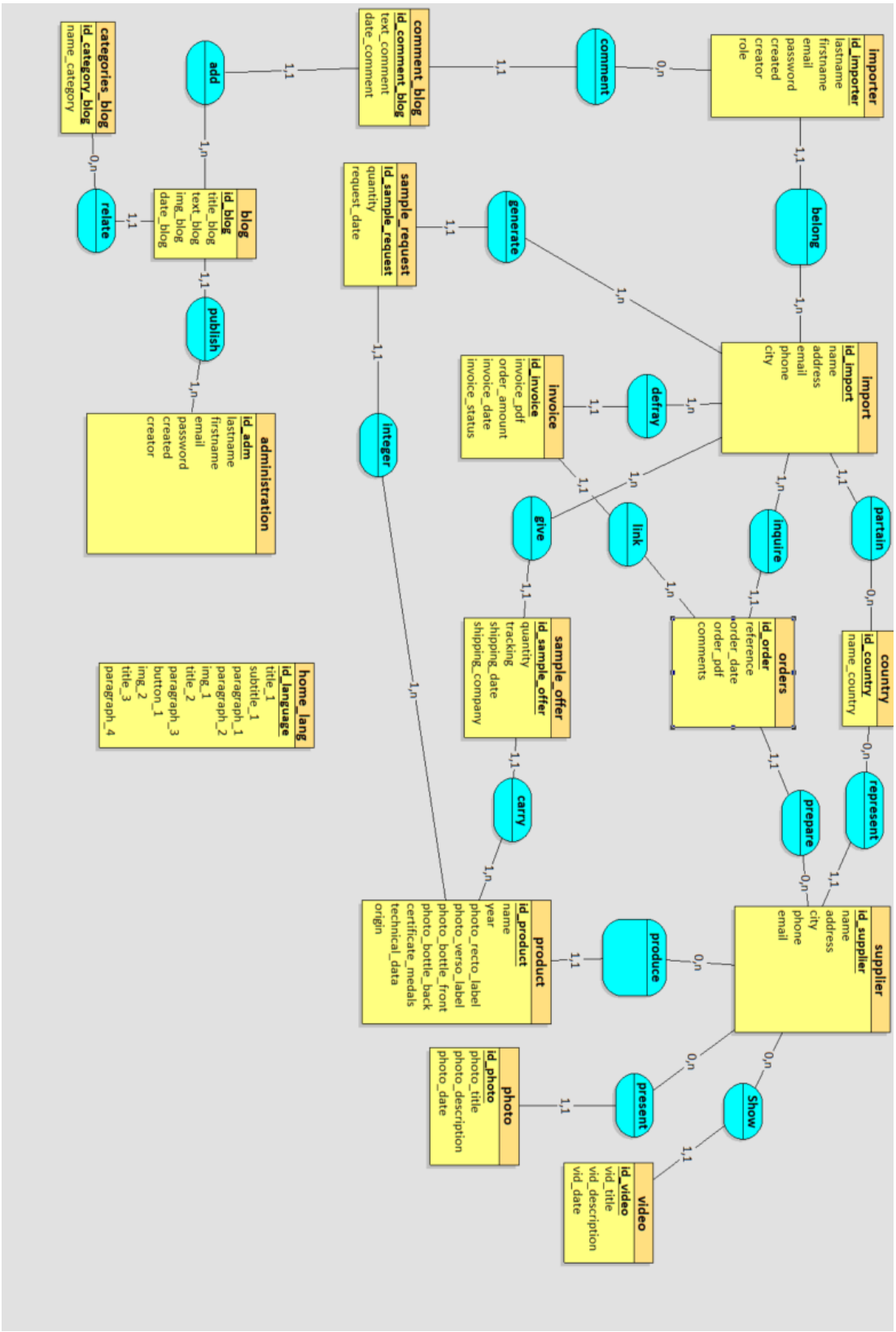
IX. ANNEXES

A. Mockups




Perfect Wines DASHBOARD
CLIENT
Link 1
Link 2
Link 3
Link 4
Home
Level 1
Level 2
Level 3
Last
Apps
Games
Movies
Books
Newspapers
Liste commandes en cours
Name
Birth Date
Actions
Ada Lovelace
December 10, 1815
Edit
Grace Hopper
December 9, 1906
Edit
Margaret Hamilton
August 17, 1936
Edit
Joan Clarke
June 24, 1917
Edit
Ada Lovelace
December 10, 1815
Edit
Grace Hopper
December 9, 1906
Edit
Liste commandes achevées
Name
Birth Date
Actions
Ada Lovelace
December 10, 1815
Edit
Grace Hopper
December 9, 1906
Edit
Margaret Hamilton
August 17, 1936
Edit
Joan Clarke
June 24, 1917
Edit
Ada Lovelace
December 10, 1815
Edit
Grace Hopper
December 9, 1906
Edit
Margaret Hamilton
August 17, 1936
Edit
Joan Clarke
June 24, 1917
Edit
Link One
Link Two
Link Three
Link Four

B. Modèle conceptuel de données



C. Modèle logique de données