**RAPPORT DE STAGE**

DU 11/01/2021 au 02/04/2021

Formation développeur web et web mobile



Boubacar DIALLO

Formateur :

Fodé FADIKA

Marion BRUSADELLI

**Sommaire**

1. Remerciements
2. Présentation du stage
3. Objectif
4. Base de données
5. Elaboration d’un dictionnaire de données
6. Réalisation du MCD
7. Ecriture du script de la création de la base de données
8. Autorisation à la connexion
9. Ecriture du script d’insertion

b.2) programmation des procédures stockées sur le SGBD

b.3) Création de la vue

3) Développement des pages statiques et dynamique

Utilisation du Framework codeigniter

3.2) Développer des composants web d’accès aux données

La gestion du CRUD sur la table produit

1. Conclusion
2. **Remerciement**

Je remercie d’abord mes formateurs qui m’ont soutenu durant cette et tout en me poussant à être confiant.

Je remercie aussi mes camarades d’avec qui j’ai passé des meilleurs moments durant tout au long de la formation.

Enfin je remercie ma famille qui m’a soutenu à attendre mes objectifs.

1. **Présentation du stage**
2. **Objectif**

L’entreprise village green distributeur des matériels musical, souhaite développer leur vente aux particuliers. En créant un site e-commerce permettant aux clients de visualiser l’ensemble du catalogue et passer des commandes en ligne. Procéder à un paiement en ligne et éditer une facture.

1. **Base de données**

J’utilise le Système de Gestion de la Base de Données (SGBD) MySQL version 8.0.21 pour la gestion de ma base de données.

1. **Elaboration d’un dictionnaire de données**

J’ai édité un dictionnaire de données recueillies dans un tableau et leur type, que j’ai recueillis pour éditer le MCD de ma base de données.Une image contenant table

Description générée automatiquement

1. **Réalisation du MCD**

J’ai réalisé mon MCD dans l’application looping.

J’ai réalisé 10 tables qui sont les suivantes :

La table produit

La table poste

La table catégorie

La table employée

La table département

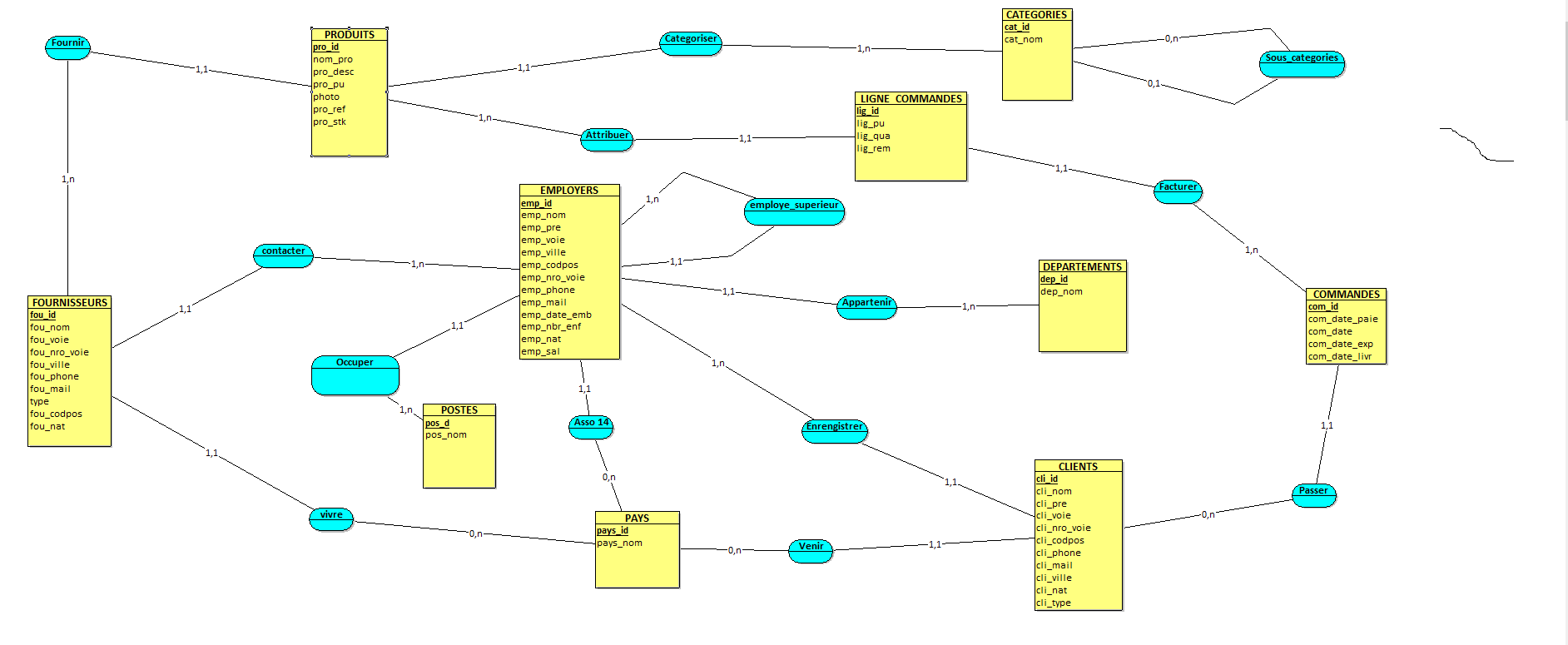
La table pays

La table commande

La table ligne de commande

La table client

La table fournisseur



1. **Script de la base de données**

J’ai traduit le Modèle Conception des Données au Modèle Logique de Données pour écrire mon script de base de données dans le vs code. La base de données s’appelle village\_green.

Je commence par créer la base de données par :

**Create database village\_green**

(Création de la base de données village\_green)

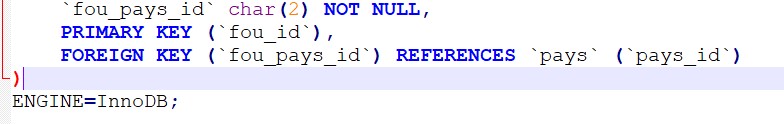
**Use village\_green**

(Utilisation de la base de données village\_green)

Une image contenant texte

Description générée automatiquement

J’ai respecté les règles clés primaire et clés étrangères lors de la traduction du MCD au MLD, la table qui reçois la clé étrangère c’est la table qui à la cardinalité maximale de 1.



Pour que je puisse importer ma base de données dans un serveur pour tester mes requêtes. J’installe WampServer qui est une plate-forme de développement Web sous Windows pour des applications Web dynamiques à l’aide du serveur Apache, du langage de scripts PHP et d’une base de données MySQL. Il possède également PHPMyAdmin pour gérer plus facilement vos bases de données.

***Apache*** *:* c'est ce qu'on appelle un serveur web. Il s'agit du plus important de tous les programmes, car c'est lui qui est chargé de délivrer les pages web aux visiteurs. Cependant, Apache ne gère que les sites web statiques (il ne peut traiter que des pages HTML). Il faut donc le compléter avec d'autres programmes.

***MySQL*** : est un logiciel de gestion de base de données dont. Il permet d'enregistrer des données de manière organisée.

PhpMyAdmin : est un logiciel gratuit écrit en langage PHP qui sert à gérer l’administration de MySQL en utilisant l’internet. Ses utilisations courantes sont :

Gestion de base de données, tables, colonnes, index.

Gestion d’utilisateurs, permissions, et privilèges de certains utilisateurs

La gestion de base de données inclus plusieurs autres sous-utilisations comme créer, consulter ou modifier des tables. Par ailleurs, on peut exécuter avec phpMyAdmin n’importe quelle requête MySQL.

Je configure mon serveur local sur le port 80.

Pour importer ma base de données, je redémarre les services du WampServer et lancer le phpMyAdmin pour pouvoir me connecter et importer le script de ma base de données.

D’abord je crée une nouvelle base de données en le nommant et définir la version de langue pour mon cas j’utilise utf8\_general\_ci et cliquer sur créer.

J’ai sélectionné ma base et cliquer sur importer je choisi la base de données dans le dossier où elle s’est trouvait puis exécuter.

1. **Autorisation de connexion**

En respectant le cahier de charge.

J’ai créé des utilisateurs : gestion, client, visiteur, administrateur

Pour le profil visiteur je lui donner des privilèges de lecture sur le catalogue qui correspond à ma table produit, et je lui ai donné le rôle de visiteur.

Pour le profil client je lui donne des privilèges de lecture sur toute la base de données, insertion et mise à jour sur les tables client et commande.

Pour le profil gestion avec des privilèges lecture et écriture sur toutes la base de données.

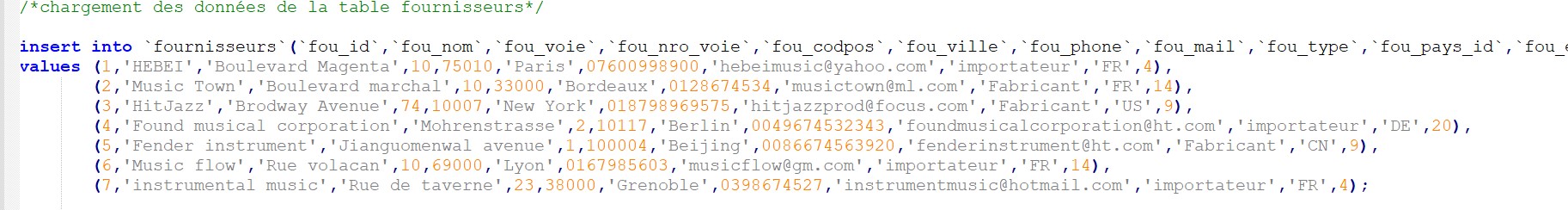
Pour l’administrateur avec tous les privilèges sur toutes la base de données et de rôle admin\_village\_user

1. **Le script d’insertion**

J’utilise toujours le vs code pour faire mon script d’insertion pour alimenter mes tables de la base de données.

La requête que j’ai utilisée pour faire mon script de la BDD est :

**INSERT INTO table (colonne1, colonne2, colonne3, etc.) VALUE (value1, value2, value3, etc.)**



Après l’insertion dans les toutes tables je les ai importés dans ma base de données en procédant la même manière dont j’ai fait importer le script de création de la base données.

**b.2) programmation des procédures stockées sur le SGBD**

J’ai fait mes procédures stockées en fonction du cahier de charge.

Les différentes procédures que j’ai faites :

Une procédure facture

Cette procédure permet de délivrer une facture lors de l’achat d’un client.

Une procédure bonne de livraison

Cette procédure permet de lister toutes les commandes passé par un client et les coordonnées du client afin d’autoriser la livraison.

Une procédure qui détermine le délai de livraison

Cette procédure qui détermine les délais de livraison d’une commande

Une procédure du retard de paiement

Cette procédure permet de lister les clients qui sont en retard de paiement à une date donnés.

Une procédure bonne de commande

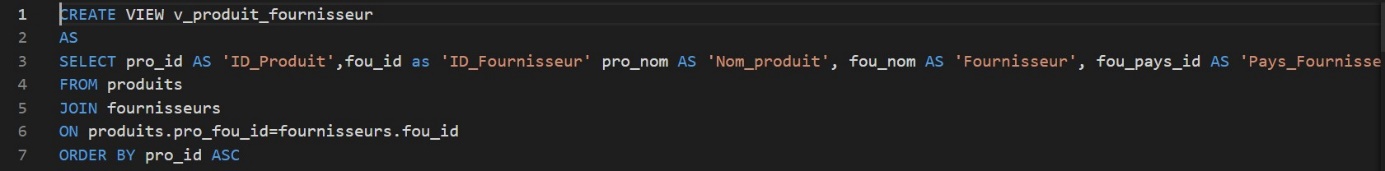
Une procédure pour lister les commandes de chaque client

Une image contenant texte

Description générée automatiquement

**b.3) Création de la vue**

J’ai créé une requête de vue correspondant à la jointure entre produit et fournisseur. Cette requête permet de lister tous les produits, leurs fournisseurs et le pays du fournisseur.



1. **Développement des pages statiques et dynamique**

**Utilisation codeigniter**

J’utilise codeigniter qui est un Framework web écrit en PHP, qui vante une conception logicielle compacte rendant le développement d'applications Web plus rapide et plus efficace. Ne maîtrisant pas bien ce Framework, il fallait jumeler entre l’apprentissage et la pratique sur mon projet. Chose qui n’était pas facile.

Je travail avec les dossiers **Controller, Models, View**

**Le Controller :** sert d’instance de médiation entre le models, la View

et toute autre ressource nécessaire ou à la création des pages web dynamique.

**View :** est partie de l’application qui est présentée à l’utilisateur final. Il s’agit en principe d’un document HTML dans lequel le contenu est intégré dynamiquement via PHP.

**Models :** Il s’agit des fonctions spéciales qui permettent d’extraire des informations d’une base de données.

J’ai téléchargé le Framework v 3.1.11 en format zip, je le dézippe dans mon serveur local (WampServer) dans le dossier wamp et le sous dossier www qui se situe dans mon C.

Ensuite je procède à la configuration du codeigniter :

Dans le fichier **$autoload.php**, je charge les bibliothèques qui se trouve à le ligne 61

$autoload['libraries'] = array('form\_validation', 'database')

**La libraries ou bibliothèque** contient de fonction qui permet l’utilisation simple, rapide et efficace l’application. Pour mon cas je charge form\_validation et database.

**Form\_validation** qui permet de contrôler la validation de la saisie de l’utilisateur avec les critères bien précis.

**Database** permet de charger directement la base de données dés le démarrage de l’application.

A la ligne 92 $autoload['helper'] = array('url','form')

Un **helper** est un ensemble de fonctions pouvant être appelé depuis le contrôleur, la vue ou le modèle. C'est simplement un fichier. PHP (sans classe) qui liste des fonctions regroupées par un même thème.

Pour mon cas je charge le url et form :

**url :** ce qui me permet de charger mes urls partout.

**Form :** le fichier form helper contient des fonctions qui facilitent l’utilisation des formulaires.

Dans le fichier **$config.php** à la page 26 je configure l’url de mon site.

$config['base\_url'] = 'http://villagegreen/’

C’est qui me permettra de me connecter via un navigateur.

Je configure ma base de données dans le fichier **$database.php.**

Les paramètres de configuration sont stockés dans un tableau multidimensionnel

$active\_group = 'default';

$query\_builder = TRUE;

$db['default'] = array(

    'dsn'   => '',

    'hostname' => 'localhost:3306',

    'username' => 'admin\_village\_user',

    'password' => '',

    'database' => 'village\_green',

    'dbdriver' => 'mysqli',

    'dbprefix' => '',

    'pconnect' => FALSE,

    'db\_debug' => (ENVIRONMENT !== 'production'),

    'cache\_on' => FALSE,

    'cachedir' => '',

    'char\_set' => 'utf8',

    'dbcollat' => 'utf8\_general\_ci',

    'swap\_pre' => '',

    'encrypt' => FALSE,

    'compress' => FALSE,

    'stricton' => FALSE,

    'failover' => array(),

    'save\_queries' => TRUE

);

En fin je charge mon Controller dans le fichier **$route.php** à la ligne 52

$route['default\_controller'] = 'action';

J’ai créé un fichier action dans mon Controller qui s’appelle action pour gérer toutes mes vues et tous mes models.

J’ai créé des fichiers que je souhaite dans mon dossier de vue.

J’ai créé une page **index.php** qui fait office de page d’accueil où je répertorie les catégories des instruments qui sont vendu par la société.

Je fais un fichier **header.php** qui fait office de l’en tête de toutes mes pages en incluant le logo, le panier, les menus.

Je fais un fichier **footer.php** que j’utilise pour toutes mes pages.

Je fais un fichier **inscription.php** pour permettre aux clients de s’inscrire et je crée un fichier action.php dans mon Controller pour faire toutes mes fonctions qui correspond à ma View et à ma Models. J’ai fait une fonction appeler contact pour valider mon formulaire d’inscription et une autre fonction appeler connexion pour afficher le formulaire.

Après je fais mon CRUD sur ma table produits.

Je commence par lister tous les produits dans un tableau.

Je crée un fichier PHP appeler **ajoutproduit** où je crée ma class **ajoutproduit** dans mon dossier Models et je vais faire ma requête. Je fais une fonction constructeur pour charger ma **base** **de** **données**. Je fais ma requête qui me permet de lister tous mes produits.

Dans mon fichier action, je fais une fonction appeler **charge** dans mon Controller où je vais charger mon Models **ajoutproduit** et ma View mon fichier PHP **chargeproduit** qui fait office de tableau. Dans ma View je fais mon tableau qui va lister tous mes produits où je fais une boucle foreach qui récupère toutes les valeurs de ma table produits et je place mes boutons de liens modifier et supprimer sur chaque ligne à la fin je fais un bouton de lien ajouter un produit.

Là je commence maintenant à faire mon CRUD par l’insertion des produits.

Je crée un fichier PHP appeler **ajout** dans mon Model ou je vais faire toutes mes fonctions du CRUD. Je crée une fonction **insert** qui prend en paramètre les données à insérer. Je passe par un tableau multidimensionnel pour donner les valeurs à insérer en base données.

Je fais une fonction dans mon Controller qui s’appelle **pro\_ajout** pour faire le lien avec ma page de formulaire d’ajout que je vais faire dans ma View et tester les données saisies par l’utilisateur et les afficher. Dans mon dossier View je crée un fichier PHP qui s’appelle **ajoutproduct** pour faire un formulaire, j’ouvre une balise PHP à l’intérieur je place ma fonction **form**\_**open** je mets en paramètre mon Controller et la fonction qui permet la validation de mon formulaire en commençant par un echo. Je ferme mon formulaire par la fonction **form**\_**close** sans de variable en paramètre. Je fais une boucle foreach pour récupérer chaque fois la **categories** en utilisant un balise HTML select avec l’option pour faire le choix. Je fais la même chose pour le **fournisseurs**. A la fin je fais un bouton submit que je vais appeler **Ajouter**.

La fonction **form\_open** permet d’ouvrir un formulaire en codeigniter

La fonction **form**\_**close** permet de fermer un formulaire en codeigniter

La mise à jour des produits ou **update**

Je fais trois fonctions le dossier Models :

Le premier **show\_produit** qui me permet de lister toutes les données de ma table, aussi **show\_produit\_id($id)** pour me récupérer les données d’un seul produit en fonction de son id, **update\_produit** permet de faire la mise à jour du produit sélectionné.

Dans mon Controller je fais 2 fonctions qui vont charger mes fonctions du Models et de ma View : la première **update**\_**id** permet de récupérer l’id envoyer à l’url, la deuxième **miseajour** permet de faire le teste le teste des différentes entrées et de les valider.

Dans ma View je crée un fichier formulaire PHP qui s’appelle **modifproduit** j’ouvre mon formulaire par la fonction **form**\_**open** de la même manière que j’ai expliqué précédemment tout en précisant la fonction de validation que je vais utiliser. Dans chaque value je précise comment je vais récupérer les valeurs à modifier, ensuite je fais une boucle foreach comme je l’ai expliqué précédemment pour la table **categories** et la table **fournisseurs** et je ferme mon formulaire avec **form\_close**. A la fin je fais un bouton submit que je vais appeler **modifier**.

Pour la suppression ou **delete**

Je crée une fonction dans mon Model appeler **supproduit** et je lui donner l’id en paramètre vu que la suppression se fait en ligne en fonction de l’id.

Dans mon Controller je fais une fonction **delete** où je vais charger ma fonction que j’ai fait dans mon Models qui va supprimer une fois qu’on sélection le bouton supprimer.

Je fais le Models pour ma table catégorie et fournisseur pour leur données que je vais charger chaque fois que j’ai besion

**4)Conclusion**

Pour conclure, ce projet a été enrichissant tout autant que la formation qui la précède. Le métier de développeur web nécessite un apprentissage constant. Ces derniers mois m’ont permis de découvrir une partie.

Les difficultés que j’ai rencontrées était, je ne connaissais pas comment faire la lecture d’un cahier de charge et mettre en place les fonctionnalités demander. Etant donné que je devais faire un site e-commerce en utilisant le MVC. Le seul Framework que je connaissais les bases c’était le codeigniter, j’ai approfondi mes connaissances pour mener bien mon projet.