Arnaud Parot 2019-2021

Fiche de production

Système de connexion/inscription

Table des matières

1) Partie base de données/SQL

1.1) Définition des besoins et du MCD

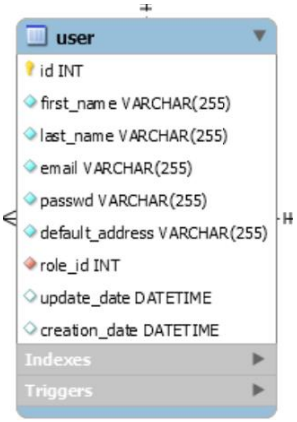
La base de données devra contenir une table de façon à sauvegarder les données des utilisateurs lors de l'inscription et de pouvoir se connecter à un compte qui existe déjà à l'aide d'une adresse e-mail et d'un mot de passe. Elle sera donc munie d'une table que l'on appellera « User » qui aura les attributs suivants :

- Id
- Nom
- Prénom
- Adresse mail
- Mot de passe
- Adresse
- Rôle

Elle disposera également d'une date de modification et d'une date d'ajout afin de savoir quand l'utilisateur a été créé et quand a-t-il été modifié pour la dernière fois.

Le rôle sera une clé étrangère vers la table « Rôle » qui contiendra tous les rôles disponibles ainsi que leurs informations, cette table est uniquement composée d'un id et d'un libelle.

Nous veillerons également que le mot de passe soit crypté afin de ne pas le laisser apparaître en clair dans la BDD, ce qui pourrai poser de gros problème de confidentialité et surtout de piratage et faciliterais les actions malveillantes.



1.2) Création du script SQL

Afin de permettre la création des tables au mieux possible, nous avons créé un fichier de script SQL appelé « Script.sql » contenant l'intégralité du script de création de la base de données (Tables, Association, Triggers, etc…) ainsi qu'un script permettant de mettre des données de base dans la BDD, ces données par défaut vont permettre le bon affichage du site lors d'une présentation, car lorsqu'il n'y a aucune donnée, rien ne s'affiche sur certaines pages web rendant la présentation de celle-ci moins tape a l'œil et agréable.

Ce fichier permet également le déploiement facile et rapide de la base de données étant donné qu'il n'y a besoin que d'un seul fichier et que nous avons juste besoin d'importer le script de création via PHPMyAdmin ou tout autre moyen d'import de script a la base de données.

La partie concernant la table User est la suivante, ce code n'est qu'une partie du script entier :

```
-- -----------------------------------------------------
-- Table `dendo_jitensha`.`user`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dendo_jitensha`.`user` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `first_name` VARCHAR(255) NOT NULL,
  `last_name` VARCHAR(255) NOT NULL,
  `email` VARCHAR(255) NOT NULL,
  `passwd` VARCHAR(255) NOT NULL,
  `default_address` VARCHAR(255) NOT NULL,
  `role_id` INT NOT NULL,
  `update_date` DATETIME NULL,
  `creation_date` DATETIME NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC),
  INDEX `fk_user_role_idx` (`role_id` ASC),
  CONSTRAINT `fk_user_role`
    FOREIGN KEY (`role_id`)
    REFERENCES `dendo_jitensha`.`role` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

1.3) Création de la base de données

Dans le cadre de notre projet nous avons utilisé PHPMyAdmin pour importer la BDD depuis le script de créations que nous avons réalisé auparavant, pour l'importer, nous nous rendons sur la page principale de PHPMyAdmin et utilisons l'onglet « Importer » prévu à cet effet.

Il nous suffit ensuite de sélectionner notre fichier « Script.sql » et de cliquer sur le bouton exécuter pour que la création se fasse automatiquement si aucune erreur de syntaxe n'est présente, a noté que s'il y a des erreurs de syntaxe, le script s'exécutera jusqu'à l'erreur mais ne continuera pas après.

Nous avons maintenant créé notre base de données nécessaire a la l'inscription et la connexion des utilisateur sur le site.

	Table	Action	Lignes	Type	Interclassement	Taille	Perte
☐	category	Parcourir Structure Rechercher Insérer Vider Supprimer	3	InnoDB	utf8_general_ci	32,0 kio	-
☐	orders	Parcourir Structure Rechercher Insérer Vider Supprimer	3	InnoDB	utf8_general_ci	48,0 kio	-
☐	order_line	Parcourir Structure Rechercher Insérer Vider Supprimer	8	InnoDB	utf8_general_ci	64,0 kio	-
☐	payment	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8_general_ci	48,0 kio	-
☐	pictures	Parcourir Structure Rechercher Insérer Vider Supprimer	11	InnoDB	utf8_general_ci	48,0 kio	-
☐	product	Parcourir Structure Rechercher Insérer Vider Supprimer	11	InnoDB	utf8_general_ci	48,0 kio	-
☐	role	Parcourir Structure Rechercher Insérer Vider Supprimer	2	InnoDB	utf8_general_ci	32,0 kio	-
☐	status	Parcourir Structure Rechercher Insérer Vider Supprimer	2	InnoDB	utf8_general_ci	32,0 kio	-
☐	user	Parcourir Structure Rechercher Insérer Vider Supprimer	3	InnoDB	utf8_general_ci	48,0 kio	-
☐	user_address	Parcourir Structure Rechercher Insérer Vider Supprimer	3	InnoDB	utf8_general_ci	48,0 kio	-
	10 tables	**Somme**	**46**	**InnoDB**	**utf8_general_ci**	**448,0 kio**	**0 o**

Extrait de la table User :

id	first_name	last_name	email	passwd	default_address	role_id	update_date
1	Admin	Admin	admin@admin.fr	admin	Admin	1	2021-03-05 14:44:
2	Arnaud	Parot	arnaudparot@outlook.fr	$2y$10$WCKpRNPz6eLaK46FP.nZWeGu4fEa.YXLpS3lcbjStyf...	4 Rue des Saules	2	2021-03-05 18:06:
3	Jean	Dupon	duponjean@mail.fr	$2y$10$yOV7d5Mr6a9h/zD9l9XjoOFqqkwQXlGveKwMEfcuJoS...	Là bas	1	2021-03-09 17:28:

Nous pouvons par ailleurs voir sur cette capture l'encryptions des mots de passes

2) Partie PHP/Backend

Lors de la réalisation de la partie principale de site web, nous avons décidé de suivre une architecture MVC pour notre code PHP, cette architecture a pour avantage de simplifier un peu le codage mais surtout de bien organiser notre code, nous avons mis en place un système performant permettant d'ajouter facilement de nouveau attribut si besoin en BDD ou d'autre tables sans altérer le fonctionnement de notre code et en rendant l'intégration de ces nouveaux attributs et nouvelles tables très facile.

2.1) Création des model

Les modèles suivent tous un modèle par défaut en héritant de celui-ci, il contient notamment une fonction permettant d'attribuer automatiquement les attributs d'une table au model correspondant via ses getters et setters appelé automatiquement, cette fonction est appelée fonction d'hydratation.

Le constructeur des model hérite donc aussi de celui du model principal et appel donc automatiquement la fonction quand on leurs passent un tableau avec les attributs, il dispose également d'un attribut id toujours présent dans chaque autre model.

```php
public function __construct(array $donnees){
    $this->hydrate($donnees);
}

private function hydrate(array $donnees)
{
    foreach ($donnees as $key => $values) {
        $methode = 'set_'.$key;
        if (method_exists($this, $methode)) {
            $this->$methode($values);
        }
    }
}
```

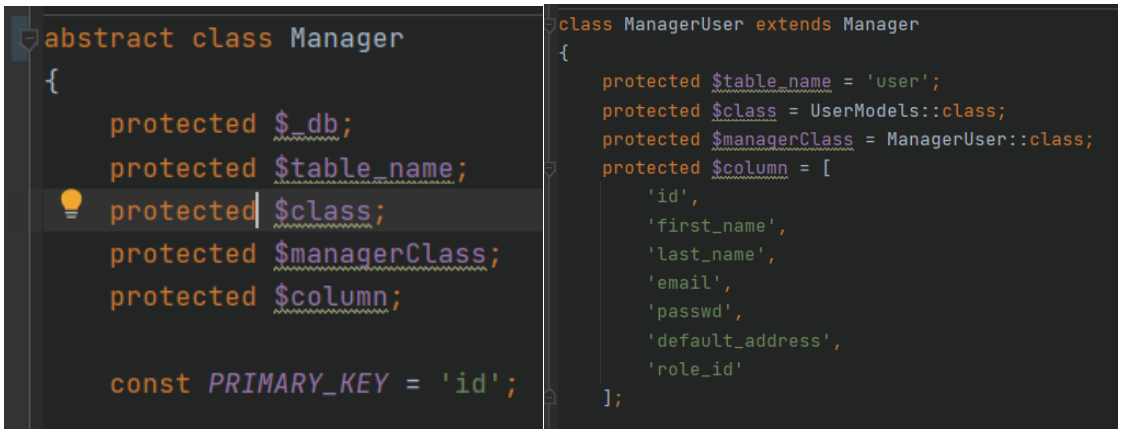
Le model qui contiens les informations des utilisateurs hérite donc du model principal et dispose de ses propres attributs avec getters et setters associé à chacun des attributs.

```php
class UserModels extends Models
{
    private $_first_name;
    private $_last_name;
    private $_email;
    private $_passwd;
    private $_default_address;
    private $_role_id;
```

2.2) Création de contrôleur pour lier à la base de données

Tous nos contrôleurs sont également basés sur un contrôleur en class abstraite qui contient quelques attributs de base mais aussi les fonctions de base permettant d'effectuer les actions principales sur la BDD, à savoir un INSERT, un UPDATE, un DELETE et un SELECT. Ce contrôleur dispose aussi d'attribut commun à chaque autre contrôleur :

- Db : Contiens un objet PDO permettant la connexion à la BDD
- Table_name : Nom de la table a laquelle est lié le manager
- Class : Le model associé à la table
- ManagerClass : Le nom du manager actuel
- Column : Le nom de chaque attribut présent dans la table



```php
abstract class Manager
{
    protected $_db;
    protected $table_name;
    protected $class;
    protected $managerClass;
    protected $column;

    const PRIMARY_KEY = 'id';
```

```php
class ManagerUser extends Manager
{
    protected $table_name = 'user';
    protected $class = UserModels::class;
    protected $managerClass = ManagerUser::class;
    protected $column = [
        'id',
        'first_name',
        'last_name',
        'email',
        'passwd',
        'default_address',
        'role_id'
    ];
```

Les fonctions SELECT, INSERT, UPDATE et DELETE sont construite tous de la même façon, elles utilisent les attributs définit dans le manager pour créer automatiquement la requête correspondante à exécuter dans la base de données en remplissant également tous les paramètres de la requête automatiquement.

```php
public function insert($objet){
        $q = $this->_db->prepare('INSERT INTO
'.$this->table_name.'('.implode(',
',array_slice($this->column, 1)).') VALUES ('.implode(',
', array_map(function ($values){return ':'.$values;},
array_slice($this->column, 1))).')');
        foreach (array_slice($this->column, 1) as $row){
            $methode = 'get_'.$row;
            $q->bindValue(':'.$row, $objet->$methode());
        }
        $q->execute();
    }
```

2.3) Inscrire un utilisateur

Lors de la connexion d'un utilisateur, nous récupérons d'abord un model utilisateur créer à partir des informations d'inscription donnée, puis nous envoyons ce model au contrôleur afin d'ajouter en base de donnée l'utilisateur via la fonction « create() » situé dans le contrôleur de base, prenant en paramètre ce dit model.

La fonction va ensuit générer la requête SQL d'ajout d'utilisateur puis y associé les informations entrées par l'utilisateur via le model en appelant automatiquement les getters associés à ce model, en cas de succès, l'utilisateur sera également connecté directement pour lui éviter de devoir se connecter manuellement après l'inscription.

```php
public function insert($objet){
    $q = $this->_db->prepare('INSERT INTO '.$this->table_name.'('.implode(', ',array_slice($this->column, 1)).')
VALUES ('.implode(', ', array_map(function ($values){return ':'.$values;}, array_slice($this->column, 1))).')');
    foreach (array_slice($this->column, 1) as $row){
        $methode = 'get_'.$row;
        $q->bindValue(':'.$row, $objet->$methode());
    }
    $q->execute();
}
```

Notre utilisateur est alors créé et enregistrer en base de données, il peut donc se connecter et déconnecter librement mais aussi commander des produits, modifier son profil, ainsi qu'accéder aux options du site qui nécessite la création d'un compte.

2.4) Connecter un utilisateur

Lors de la connexion d'un utilisateur, nous récupérons d'abord un model utilisateur créer à partir des informations de connexion donnée, puis nous envoyons ce model au contrôleur afin de tester celui-ci grâce à une fonction « connect() » prenant en paramètre ce dit model.

La fonction va ensuit vérifier si un utilisateur utilisant cette adresse mail existe, on considère que chaque adresse est unique. Si cette adresse existe, on récupère l'utilisateur associé puis on regarde si le met de passe correspond bien au mot de passe associé à ce compte, si c'est le cas, on connecte l'utilisateur en démarrant une session et stockant des informations dans cette session.

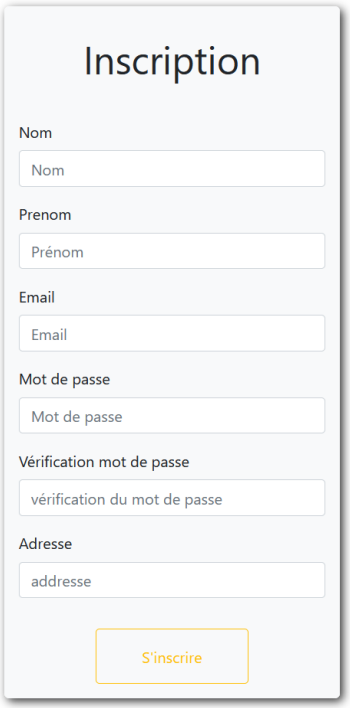
```php
public function connect(UserModels $user){
    $sql = 'SELECT id, first_name, last_name, passwd, role_id FROM '.$this->table_name.' WHERE email = :mail';
    $q = $this->_db->prepare($sql);
    $q->bindValue(':mail', $user->get_email());
    $q->execute();
    $result = $q->fetch(PDO::FETCH_ASSOC);
    if (!empty($result)) {
        if (password_verify($user->get_passwd(), $result['passwd'])){
            session_start();

            $_SESSION['is_connected'] = true;
            $_SESSION['first_name'] = $result['first_name'];
            $_SESSION['last_name'] = $result['last_name'];
            $_SESSION['id'] = $result['id'];
            $_SESSION['role_id'] = $result['role_id'];
        }
    }
}
```

Notre utilisateur est maintenant connecté et peux accéder aux options uniquement disponibles lorsqu'on est connecté.

3) Partie HTML/CSS (Formulaires de connexions/inscription)

3.1) Formulaire d'inscription

Le formulaire d'inscription doit être agréable visuellement et compréhensible, il devra permettre de saisir toute l'information nécessaire à l'inscription d'un utilisateur et de les transmettre au serveur via une requête POST qui va ensuit créer un model d'utilisateur et l'envoyer au contrôleur pour une insertion.



La mise en forme de notre formulaire sera accompagnée de Bootstrap pour fournir un affichage responsif sur mobile et tout type d'écran. Il sera composé d'un conteneur puis d'un form permettant d'envoyer les données.

```
<div class="container col-md-3 py-5">
    <div class="formulaireInscription">
        <h1 class="text-center py-3">Inscription</h1>
        <form method="post" class="pt-3">
            <div class="form-group">
```

Les champs du formulaire comprendront un nom, prénom, une adresse mail, un mot de passe, une vérification de mot de passe et une adresse. Tous les champs seront conçus de la même manière comme suit.

```
<div class="form-group">
    <label for="nom" class="display-5" >Nom</label>
    <input type="text" required="required" id="nom" class="form-control display-3" name="last_name" placeholder="Nom">
</div>
```

Une fois le formulaire envoyé nous créons le model nécessaire à l'inscription grâce aux informations contenues dans la variable $_POST

```
if (isset($_POST) && !empty($_POST) && $_POST["passwd"] == $_POST["verif_password"]){
    $user = new UserModels([
            "first_name" => $_POST["first_name"],
            "last_name" => $_POST["last_name"],
            "email" => $_POST["email"],
            "passwd" => $_POST["passwd"],
            "default_address" => $_POST["default_address"],
            "role_id" => 2
    ]);
    $manager = new ManagerUser();
    $manager->insert($user);
    $manager->connect(new UserModels([
        "email" => $_POST["email"],
        "passwd" => $_POST["passwd"]
    ]));
    header( header: "Location: address.php");
}
```

Notre utilisateur est maintenant créé et peut se connecter librement au site

3.2) Formulaire de connexion

Le formulaire de connexion doit également être agréable visuellement et compréhensible, il devra permettre de saisir toute l'information nécessaire à la connexion d'un utilisateur et de les transmettre au serveur via une requête POST qui va ensuit créer un model d'utilisateur et l'envoyer au contrôleur pour une connexion.

La mise en forme de notre formulaire sera accompagnée de Bootstrap pour fournir un affichage responsif sur mobile et tout type d'écran. Il sera composé d'un conteneur puis d'un form permettant d'envoyer les données.

```
<div class="container col-md-3 py-5">
    <div class="formulaireInscription">
        <h1 class="text-center py-3">Connexion</h1>
        <form method="post" class="pt-3">
```

Les champs du formulaire comprendront une adresse mail associé à un compte et un mot de passe qui seront définit comme suit.

```
<div class="form-group">
    <label for="email" class="display-5" >Email</label>
    <input type="email" required="required" id="email" class="form-control display-3" name="email" placeholder="Email">
</div>
```

Les informations seront ensuite récupérées par le biais de la variable $_POST et formeront ensuit un model qui sera envoyé au contrôleur pour tester les informations et connecter l'utilisateur.

```
if (isset($_POST) && !empty($_POST)){
    $manager = new ManagerUser();
    $manager->connect(new UserModels([
            "email" => $_POST["email"],
            "passwd" => $_POST["password"]
    ]));
    if (isset($_SESSION) && !empty($_SESSION)){
        header( header: "Location: index.php");
    }
}
```

Si les informations sont correctes, l'utilisateur est maintenant connecté au site et peux accéder librement à son compte ainsi qu'au fonctionnalité du site qui nécessite d'être connecté.