DOSSIER DE

PROJET

Une image contenant texte

Description générée automatiquement

*Formation développeur web et web mobile*

**2021-2022**

**MUSTAFAJ Fabrice**

Table des matières

[I. Présentation 5](#_Toc103606388)

[II. Liste des compétences du référenciel 6](#_Toc103606389)

[1. CCP1 - Developper la partie Frond-end d’une application Web ou Web mobile en intégrant les recommandations de sécurité 6](#_Toc103606390)

[a. Maquetter une application 6](#_Toc103606391)

[b. Réaliser une interface web statique et dynamique 6](#_Toc103606392)

[c. Développer une interface utilisateur web dynamique 6](#_Toc103606393)

[2. CCP2 - Developper la partie Back-end d’une application Web ou Web mobile en intégrant les recommandations de sécurité 6](#_Toc103606394)

[a. Créer une base de données 6](#_Toc103606395)

[b. Développer les composants d’accès aux données 6](#_Toc103606396)

[c. Développer la partie backend d’une application 6](#_Toc103606397)

[III. Résumé 7](#_Toc103606398)

[IV. Cahier des charges 8](#_Toc103606399)

[Objectifs : 8](#_Toc103606400)

[Support visés : 8](#_Toc103606401)

[Charte graphique : 8](#_Toc103606402)

[Arborescence du site partie Front-end : 9](#_Toc103606403)

[Arborescence du site partie Back-office : 10](#_Toc103606404)

[a) Développement du projet 11](#_Toc103606405)

[b) Methode MERISE 12](#_Toc103606406)

[**i.** Modèle conceptuel des données 12](#_Toc103606407)

[**ii.** Modèle Logique des données 12](#_Toc103606408)

[**iii.** Modèle physique des données 12](#_Toc103606409)

[a. Front-end 13](#_Toc103606410)

[b. Back-end 14](#_Toc103606411)

[VII. Extraits de code significatifs 16](#_Toc103606412)

[1. Générateur de vêtement via le produit cartésiens des différents attributs 16](#_Toc103606413)

[2. Création d’un compte utilisateur 31](#_Toc103606414)

Table des matières

1. Présentation **……………………………..………………………………………………………....... 4**
2. Liste des compétences du référenciel **…………………………………………………………….. 5**
   1. **CCP-1** Développer la partie Front-end d’une application Web et Web Mobile en intégrant les recommandations de sécurité …………………………………………………………..……………………… 5
   2. **CCP-2** Développer la partie Back-End d’une application Web et Web Mobile en intégrant les recommandations de sécurité …………………………………………………..……………………………… 6
3. Résumé du projet **………………….………………………………...…………………………….... 7**
4. Cahier des charges **….……………………….………………………...…………………………… 8**
   1. Objectifs …………………..…………………………………………………………………………….. 8
   2. Charte graphique ……….………………….…………………………………………………………... 9
   3. Arborescence du site partie Front-end …………………………………………………………….. 10
   4. Arborescence du site partie Back-end ……………………………………………………………… 11
5. Spécifications techniques
   1. Développement du projet ……………………………………………………………………………. 12
   2. Mehode MERISE ...…………………………………………………………………………………… 13
   3. Technologies utilisées ………………………………………………………………………………... 14
   4. Conception de base de données
6. Extraits de code significatifs
   1. Génération des vêtements avec déclinaisons (inspiré de Prestashop)
   2. Gestion de contenu CRUD
7. Jeux d’essai
   1. Présentation
   2. Plan du jeu d’essai
   3. Resultat des tests
8. Vulnérabilité et sécurité
   1. Failles XSS
   2. Injection SQL
9. Extrait du site anglophone et traduction
   1. Extrait Anglophone\*
   2. Traduction
10. Annexes

1. Présentation

Je m’appelle MUSTAFAJ Fabrice, j’ai 28 ans, avant de me lancer dans ma reconversion professionnelle, j’ai travaillé dans le domaine de la sécurité au Luxembourg durant 5 ans.

J’occupait le poste au pc de sécurité et durant mes nuits j’ouvrais un fichier excel et je m’était fixé comme objectif de réaliser un planning général dynamique sur excel où tous mes collègues pouvaient entrer leurs différents postes et cela en calculant les heures préstées, les heures supplémentaires ainsi que les heures mis dans le POT tout en dissociant les heures préstées de nuit, je n’avait aucune connaissance au language de programmation.

Lorsque j’ai réussi, cela m’a apporté une énorme satisfaction de ce fait voyant que le métier d’agent de sécurité ne me correspondait plus et ayant toujours eu ce regret d’avoir arrété brusquement mes études, j’ai commencé à m’interesser au développement Web, je me suis finalement renseigné sur les formations diplomantes, dans le domaine du développement web, disponible dans le secteur,

Puis j’ai quitté mon travail afin de poursuivre se domaine qui me passionne.

1. Liste des compétences du référenciel
2. CCP1 - Developper la partie Frond-end d’une application Web ou Web mobile en intégrant les recommandations de sécurité
   1. Maquetter une application
   2. Réaliser une interface web statique et dynamique
   3. Développer une interface utilisateur web dynamique
3. CCP2 - Developper la partie Back-end d’une application Web ou Web mobile en intégrant les recommandations de sécurité
   1. Créer une base de données
   2. Développer les composants d’accès aux données
   3. Développer la partie backend d’une application
4. Résumé

Durant la formation notre formateur nous a demandé de réaliser un site d’e-commerce afin de le présenter pour l’examen, car le projet réalisé en stage est soumis à une close de confidentialité et par conséquent je suis dans l’incapacité de le présenter, j’avais libre-choix pour le thème, j’ai choisi le thème des vêtements.

Durant ce projet j’ai essayé de maximiser mes compêtences et de sortir de ma zone de confort en utilisant des bibliothèques, tout en suivant la documentation.

J’ai, pour ce projet, de ce fait pris le rôle du client et du développeur.

1. Cahier des charges

## Objectifs :

* Réaliser un site d’e-commerce

## Support visés :

* Ordinateurs de bureau
* Ordinateurs portables
* Tablettes
* Mobile

## Charte graphique :

## 

* Roboto (police)
* Regular mode
  + #FFFFFF
  + #FF850C
  + #F8EDED
* Dark mode
  + #FF850C
  + #252525
  + #5A5A5A

## Arborescence du site partie Front-end :

* Accueil
* Connexion / S’enregistrer
* Femme
  + Pull
    - Product
  + Pantalon
    - Product
  + T-shirt
    - Product
* Homme
  + Pull
    - Product
  + Pantalon
    - Product
  + T-shirt
    - Product
* Curvy
  + Pull
    - Product
  + Pantalon
    - Product
  + T-shirt
    - Product
* Enfant
  + Pull
    - Product
  + Pantalon
    - Product
  + T-shirt
    - Product
* Contact
* A propos

## Arborescence du site partie Back-office :

* Dashboard
* Tables
  + Top-catégories (Éditions / Suppression)
  + Sous-catégories (Éditions / Suppression)
  + Attributs (Éditions / Suppression)
  + Produits (Ajout / Éditions / Suppression)
* **Forms**
  + Top-catégories (Ajout catégories)
  + Sous-catégories (Ajout sous-catégories)
  + Attributs (Ajout attributs / valeurs)
  + Produits (Ajout produits)

1. Spécifications techniques
2. Développement du projet

Pour le développement du site internet j’ai commencé par observer les différents site d’e-commerce dans le secteur des vêtements afin de pouvoir m’inspirer, j’ai commencé par réaliser le logo du site, j’ai déterminé la charte graphique ensuite j’ai effectué le maquettage.

Pour le projet j’ai décidé de le réaliser en programmation objet en respectant la structure MVC (Modèle, Vue, Controller). Ayant travaillé avec Symfony (framework PHP) pendant ma période de stage, j’ai voulu reproduire le concept, cela me permet également de booster mes connaissances en programmation objet et de mieux comprendre Symfony dans le fond.

j’ai installé composer (gestionnaire de dépendance PHP) sur le projet, j’ai installé par la suite la bibliothèque DOCTRINE ORM (Mapping object-relationnel) puis j’ai installé la librairie AltoRouteur (creation de route relative), tout les dépendances se trouvent dans un dossier créer par composer s’appelant « vendor ».

1. Methode MERISE

Durant la formation de remise à niveau des métiers du numérique, nous avions abordé la méthode MERISE, qui est une méthode permettant l’analyse, la conception et la réalisation d’un système d’information. La Methode MERISE consiste à séparer les données et les traitements et se fait en 3 niveaux (modèles) conceptuel, logique et physique. L’utilisation d’une méthode dans l’analyse, la conception et la réalisation d’un système d’information est primordial, dans un premier temps pour éviter toute anomalie dans le système d’information (ex : redondance, synonyme, polysème, etc.) et afin de garantir une maintenance optimale.

1. Modèle conceptuel des données

Le modèle conceptuel à pour objectif de décrire les données qui seront utilisées, ces données sont représentées sous forme d’entités on indique les relations entre elles s’il y en a, puis on défini la cardinalité de chaque relation.

1. Modèle Logique des données

Le modèle logique à pour objectif de transformer tout les entitées en tables en observant les cardinalités de chaque relation pour pouvoir ajouter les clés étrangères dans les dites tables.

1. Modèle physique des données

Le modèle physique consiste à créer la base de données dans un SGBDR (système de gestion de base de données relationnel)

1. Technologies utilisées
   1. Front-end
      * HTML5

HTML5 (HyperText Markup Language) est un language de balisage permettant de faire la structure du site internet.

* CSS3

CSS3 (Cascading Stylesheet) est un language permettant d’appliquer du style sur un élément HTML.

* JavaScript

JavaScript (ECMAscript) est un language de script orienté objet coté Front-end (mais peut être aussi utilisé côté back-end notamment grace à Node.js) qui permet de manipuler les éléments du DOM (Document Object Model).

* AJAX

AJAX (asynchronus JavaScript and XML) est un ensemble de techniques permettant d’envoyer des données au serveur de façon asynchrone, c’est-à-dire sans rechargement de page.

1. Back-end

* PHP 8.0

PHP (PHP HyperText Preprocessor) est un language de programmation libre côté server, orienté objet, il permet la création de page web dynamiques. Son utilisation est possible via XAMPP (serveur local) ou bien sur un serveur HTTP où PHP est installé.

* DOCTRINE ORM

DOCTRINE ORM (Objet-relational mapping) est une bibliothèque PHP permettant de créer et d’interagir sur une base de données via des class PHP, les class sont la représentation des tables de la base de données et elles contiennent des méthodes pouvant gérer la création, la lecture, l’édition ou la suppression des données.

* MySQL

SQL (Structured query language) est un language pour exploiter les bases de données relationnel.

* AltoRouteur

AltoRouteur est une bibliothèque PHP permettant de créer des routes virtuelles ce qui me permet de faire appel aux différents controller du site selon la route prédéfinie.

* Make

Make est un programme que j’ai installé pour le projet me donnant la possibilité de créer un fichier Makefile qui permet l’execution de plusieurs commandes de terminaux, très utile lorsqu’on utilise DOCTRINE est que l’ont doit réinitialiser la base de données et générer les fixtures pour des tests.

1. Extraits de code significatifs
   1. Générateur de vêtement via le produit cartésiens des différents attributs

Pour la création des produits, j’ai voulu m’inspirer de prestashop, j’ai donc réalisé un générateur de produit, qui génère le produit ainsi que ses déclinaisons selon les différents attributs séléctionnés (taille, couleur, matière, etc).

En effet lorsque l’on souhaite créer un produit le formulaire est récuperer via Javascript grâce à la classe native **FormData**, permettant de récuperer les données d’un formulaire, cela est transmis au serveur via l’**AJAX** grâce à la classe native **XMLHttpRequest** (on peut utiliser la fonction **Fetch** ou la librairie **AXIOS** églament). Cela est traité par le serveur en arrière-plan qui va crée un produit parent ainsi que les différentes déclinaisons via une fonction qui calcule le produit cartésien de tous les attributs séléctionnés.

Le serveur renvoi une réponse qui est récupéré par Javascript comportant un tableau avec toutes les déclinaisons, on a la possibilité par la suite d’indiquer le stock de chaque déclinaison, l’impact sur le prix (donnant la possibilité de modifier le prix d’une ou des déclinaisons souhaitées) ainsi que les différentes images pour chaque déclinaison de produit, pour le moment aucun produit n’apparait sur la base de données, une fois le stock et les photos indiquées, le formulaire est de nouveau renvoyé via la fonction FormData et cette fois-ci j’ajoute (set) au FormData la quantité, l’impact sur le prix ainsi que les images des différentes déclinaisons et cela est finalement ajouté en base de données.

Puisque j’envoie deux fois le formulaire, une fois pour générer les produits et une autre fois pour valider la création des produits, j’ai créé la fonction sendForm() qui prend comme paramétre *values* qui sera le formulaire à envoyer au serveur. Je crée une variable qui contiendra l’objet XMLHttpRequest(), cet objet contient des méthods dont «onreadystatechange », lorsque l’on utilise l’XMLHttpRequest pour envoyer des données à un fichier celui-ci passe par quatre étapes, appelé le ‘readyState ‘, celui-ci va de 1 à 4, lorsque celui-ci est à la quatrième étape cela veux dire que le processus est arrivée a son terme, il y’a aussi le status qui est un code HTTP renvoyé par le serveur, 200 losrqu’il a réussi à communiquer avec le fichier en question, 404 lorsque qu’il n’a pas trouvé le fichier avec lequel il devait communiquer donc j’ai une condition indiquant le traitement à effectuer dans le cas ou le readystate est à son terme et lorsque le code du status indique que tout s’est bien passé. Dans methods open j’indique en premier paramétre le method d’envoi (*POST* ou *GET*), le chemin (path) du fichier dans lequel nous souhaitons communiquer ainsi qu’un troisième paramètre qui est un booléen *TRUE* ou *FALSE* pour indiquer si cela est envoyé de façon asynchrone ou non. J’indique via le responseType dans quel type de retour je souhaite recevoir la réponse, puis j’utilise la methode send qui prendra le paramétre de la fonction sendForm comme valeur à envoyer.

*ProductForm.js*



Lorsque le formulaire est submit donc envoyé je récupére via l’interface **Event** l’évènenement en question, l’interface **Event** présente une méthode **PreventDefault** qui permet d’annuler les actions par defaut du navigateur lors de l’évènement.

J’instancie la classe **FormData** dans la constante **formData**, comme paramétre j’inclue le formulaire ayant été pointé via l’id, j’ajoute la clé generateProduct au formulaire avec comme valeur ‘true’ cela me servira pour le traitement coté serveur et je fini par envoyer le formulaire avec la fonction sendForm avec en paramétre la constante formData.

*ProductForm.js*

Une image contenant texte, écran, capture d’écran, fermer

Description générée automatiquement

Dans le fichier **manageTraitement.php** qui executera le traitements des données reçues par **JavaScript** via la technique **AJAX**, j’ai utilisé la fonction **require\_once** afin de charger le fichier contenant la classe **Database** qui contient les paramètres de connexion à la base de données ainsi que l’**entityManage**r de **DOCTRINE**.

L’**entityManager** est une classe inherente à **DOCTRINE** permettant de se connecter à une base de données **SQL**, elle contient les méthodes necessaires pour valider les actions sur la base de données ainsi que pour lire les tables de notre base de données, j’utilise aussi **require\_once** pour charger les modèles (tables de la bdd sous forme de classe PHP utilisées par DOCTRINE) qui seront necessaires durant le traitement.

*ManageTraitement.php*Une image contenant texte

Description générée automatiquement

J’ouvre ma conditions en stipulant avec la fonction isset que dans le cas où la variable **$\_POST[‘generateProduct’]** ou **$\_POST[‘validProduct’]** est différent de null alors cela renvoie un booleen qui execute le traitement dans le cas ou elle renvoie true, je refait ensuite une condition avec la fonction isset sur le champs du formulaire afin d’executer le traitement lorsque tous les champs sont saisie.

Je récupère l’entityManager de **DOCTRINE** grâce a la methods « **getEntityManager** » se trouvant dans la classe **Database.php**, je récupère la value du selectbox contenant les sous-categories en déclarant dans quelle type doit être la variable (sécurité), je récupère la sous-catégorie à affilié au produit via l’entityManager grâce à la méthodes « **getRepository** » qui permet de récupérer des collections d’objets (c’est-à-dire les Models), la méthode **findOneBy** me permet de récupérer une données sur la base de données, cela revient à faire « WHERE id = VALUES LIMIT 1», je récupère tout les attributs se trouvant dans la base de données, je récupère les autres champs, sachant qu’il s’agit de champs en chaîne de caractère j’utilise la fonction **htmlspecialchars** et **trim** avant de l’affecter à une variable, la fonction htmlspecialchars me permet d’enlever tout les chevron, les simples et doubles quotes, ce qui constitue une sécurité notemment pour les attaques **XSS** (cross site scripting), consistant à envoyer des balises script avec du code à éxecuter via les champs de formulaire et qui sera par la suite interprété par le serveur.

La fonction trim est utilisé pour retirer les espaces en début de chaîne et en fin de chaîne. Je fini par récupérer le nombre d’attribut présent dans le tableau **$attributes** et je déclare la variable **$iterator** qui sera à 0, ces variables me serviront dans le traitement plus tard.

*ManageTraitement.php*

Une image contenant texte

Description générée automatiquement

Je commence par construire ma table en le concatenant au fur et à mesure avec le ‘**.=**’.

Je créer la variable **$parentProduct** qui aura pour valeur un nouvel objet **Products**, la classe **Products** est le reflet de la table **Products** se trouvant dans la base de données, celle-ci contient également les méthodes pour la gestion de l’objet en question, j’entre le nom de produit sur le produit parent ainsi que la description, la date de création et la sous-catégories dans lequelle il sera affilié, je finit par stipuler une conditions qui me servira plus tard dans le traitement.

J’indique que si **$\_POST[‘quantity’]** qui contient plusieur valeurs sous forme de liste n’est pas égale à null, alors je créer la variable quantity qui aura pour valeur l’explode de **$\_POST[‘quantity’]** j’ai indiquer le délimiter qui est une virgule ce qui me créer un tableau avec les différents éléments de la liste, j’utilise la fonction **array\_filter** qui enlève tout les éléments sans valeur d’un tableau. Je refait une condition en indiquant que si le taille du tableau n’égale pas 0 alors je persist l’objet **$parentProduct**, ce qui l’inclue dans un cache de **DOCTRINE** sinon cela voudrais dire qu’aucun stock n’a été indiqué dans aucune déclinaisons de ce fait la création du produit est annulé puis avec return je met fin au traitement.

*ManageTraitement.php*

Une image contenant texte

Description générée automatiquement

Ensuite je créer une boucle foreach, qui se traduit part, pour tout les attribute dans le tableau attributes alors tu boucle et attribute renvoie à l’élément du tableau, à chaque itérations je récupère le nom de l’attribut, j’utilise la fonction **strtr** pour remplacer des caractères dans une chaîne de caractères, la fonction **strtolower** me sert comme sont nom l’indique à mettre la chaîne de caractère en minuscule, puis la fonction **ut8\_decode** me sert a remplacer les accents sur les ‘e’ , j’ai déclaré également une condition indiquant que pour chaque itération si **$\_POST[‘attributeName’]** n’ égale pas null alors créer les « td » contenant le nom de l’attribut présent pour la table puis je récupère la valeur dans le tableau **$arrays**, de ce fait toute les valeurs des checkbox de chaque attribut se trouve dans le tableau **$arrays** , la variable **$attributeChooseCount** me sert à récupérer le nombre d’attributs séléctionnés qui me servira pour la condition plus bas, dans lequelle je stipule que si le compte des attributs séléctionné égale 0, alors je renvoi le message d’erreur « aucun attribut de séléctionné » puis j’utilise return pour mettre fin au traitement, je termine par concaténer les « td », je referme le « tr » puis je déclare la variable **$combinations** qui contiendra toute les déclinaisons selon les attributs et leurs valeurs associées ayant été rendu possible grâce à la fonction cartesian que j’ai déclarer plus bas me servant à calculer le produits cartésien de chaque valeurs des attributs choisis.

*ManageTraitement.php*

Une image contenant texte

Description générée automatiquement

Je déclare une nouvelle boucle foreach, cette fois-ci avec le tableau **$combinations** qui contient un tableau avec à l’interieur un tableau avec les valeurs des attributs, donc pour chaque **$combination** dans le tableau **$combinations**, je créer une des déclinaisons du produit, à cet déclinaisons j’ajoute la référence via la methode pour avoir une référence unique j’utilise la fonction uniqid() qui générera comme son nom l’indique, un identifiant unique et j’affecte le produit parent à la déclinaison.

Je crée l’element « tr » devant contenir les « td » contenant les informations de la déclinaison créer, j’utilise la variable **$iterator**, ayant été déclaré en amont, dans une ternaire, celle-ci vu qu’elle est incrémenté à chaque itérations en fin de boucle foreach de chaque décinaisons de produit me permet de switch de couleur sur le « tr » après chaque itérations grâce au modulo, dans la boucle des **$combinations** j’ai refait une boucle foreach cette fois-ci en stipulant que pour chaque **$value** (qui contient l’id de la table contenant les valeurs des attributs) dans le tableau **$combination,** je récupère dans la boucle la valeurs de l’attribut en le récuperant grâce à l’**entityManager** que j’ajoute à la déclinaison.

J’effectue une ternaire où je stipule grâce à la fonction **preg\_match()** qui permet de savoir si la valeur correspond à un code hexadecimal, ce qui me permet de déterminer si la valeur est une couleur et lorsque la valeur est une couleur je créer une « span » dans la balise « td » avec comme background le code hexadecimal en question ce qui me permet de rendre l’affichage des déclinaisons beaucoup plus soigné au niveau des données, dans le cas contraire cela indique juste la valeur dans la balise  « td ».

*ManageTraitement.php*



Je continue la construction de mon « tr » contenant la déclinaison, j’ajoute les éléments « td » qui vont contenir le champ pour la quantité, le champ pour l’impact du prix ainsi que le champ pour ajouter les images, j’écris une condition que si **$\_POST[‘validProduct’]** n’égale pas null et que si la variable **$\_POST[‘priceImpact’]** n’ égale pas null alors je créer un tableau de la même maniére qu’avec la variable **$\_POST[‘quantity’],** la variable **$iterator** me sert cette fois-ci à pointer la valeur (si elle a été indiquée) du champ impact du prix de la déclinaison en question, dans le cas ou une valeur est présente je remplace la valeur du prix de base par la valeur qui sera spécifié dans le champ impact prix et je l’ajoute à la déclinaison, dans le cas contraire c’est le prix de base.

*ManageTraitement.php*



Je fais ensuite deux ternaires la première indiquant que si l’élément du tableau **$quantity** à l’index de de la valeur de la variable **$iterator** alors tu ajoute la quantité à la déclinaison sinon rien, j’en refait une autre où cette fois-ci j’indique à **DOCTRINE** de mettre la déclinaison en cache si la quantité est stipulée ou de ne rien faire dans le cas contraire. J’effectue également une condition cette fois-ci pour gérer le cas des images en stipulant que si la variable **$\_FILES[‘photo0($iterator)’]** n’égale pas null.

Je déclare la variable $dossier que sera le dossier dans lequel seront ajoutées les images, je créer un tableau **$extension\_valides** qui contient toute les extensions autorisées. Je créer une boucle foreach, pour chaque clé => valeur dans la variable **$\_FILES[‘photo0’][‘name’],** je récupère l’extension grâce à la fonction **substr** et **strrchr** du fichier en question ce qui me permet de faire une condition plus bas pour indiquer avec la fonction in\_array si l’extension du fichier correspond bien à une extension présente dans le tableau des extensions autorisées, si c’est le cas je concatène la variable **$dossier** avec la variable **$photoName** afin d’avoir le path (chemin) du dossier qui va accueillir les photos.

Je déclare la variable **$result** qui va prendre comme valeur le retour de la fonction **move\_uploaded\_file,** la valeur de retour est une booléenne, la fonction **move\_uploaded\_file** permet de d’envoyer un fichier d’un point A vers un point B, j’utilise la variable **$key** pour ciblé la photo en question, puis je met comme deuxième paramètre le **$path** (chemin) du dossier qui va accueillir la photo, puisque le retour de la fonction **move\_uploaded\_file** est une booleene je m’en sert dans une condition encore plus bas et déclarant que si la variable **$result** egale true alors je créer l’objet photo, qui contiendra le nom, le chemin (pour l’affichage dans les balises img) ainsi que la déclinaison de produit pour laquelle la photo est affiliée puis j’utilise la fonction persist pour mettre tout sa en cache. Dans le cas ou la variable **$\_POST[‘photo0’]** est null alors cela va créer un objet photo avec comme nom et path « undefined ».

*ManageTraitement.php*



Je referme ma table, j’indique dans une condition que se la variable **$\_POST[‘validProductToAdd’],** **$\_POST[‘quantity’]** n’égale pas null et que la variable **$\_POST[‘quantity’]** n’egale pas rien alors j’utilise la methode flush de l’entityManager qui va ajouter tout les objets mis en cache par la fonction persist dans la base de données

*ManageTraitement.php*



* 1. Création d’un compte utilisateur

Pour la création du compte utilisateur, j’ai un controller s’appelant UserController.php qui contient une route dont le chemin est « /register » en avec la methode « GET », lorsque l’on accéde à cet route cela active la méthode registerUser, où j’éxecute à l’interieur la methode render hérité (extends) de la classe abstraite, AbstractController.php, comme dans symfony l’AbstractController me permet d’utiliser la méthode render qui me permet de charger la vue à afficher.

*UserController.php*

