Loya Dylan
Tribouillois Julien

2022/2023

Rapport du projet de technologie du web

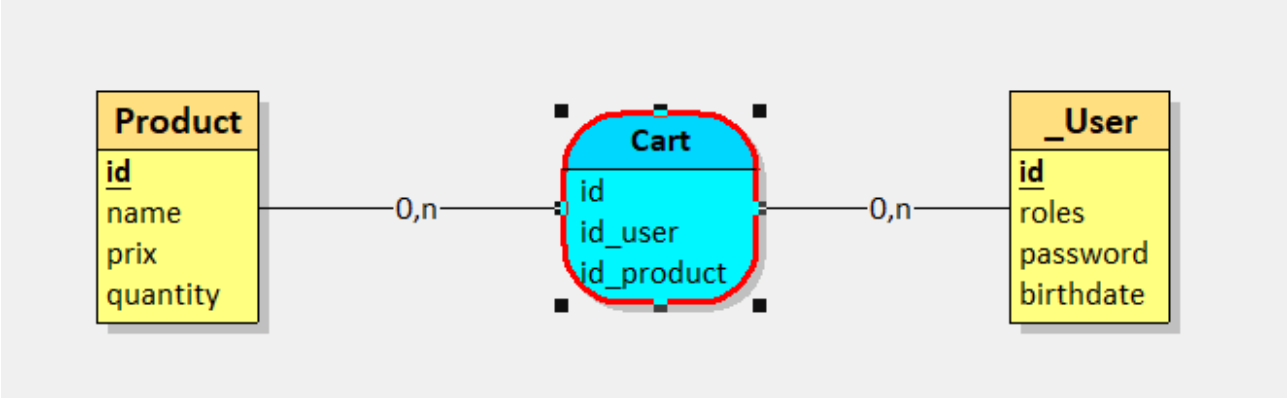
« Sac à d'eau »

FACULTÉ
DES SCIENCES
FONDAMENTALES
ET APPLIQUÉES
UNIVERSITÉ DE POITIERS

Table des matières

I. Schéma de la base :.....3
II. Organisation du code :.....3
1. Controller :.....3
2. Entity :.....3
3. Form :.....4
4. Repository :.....4
5. Security :.....4
6. Service :.....4
7. Templates :.....4
III. Créer un service dans Symfony :.....5
IV. Points particulier :.....5

I. Schéma de la base :



La base de données de notre site est composé de trois entités, à noté que dans le code la classe *_User* se nomme *User* cependant le logiciel sur lequel le schéma à été réaliser n'accepter pas le nom *User* car déjà réservé. Nous pouvons voir qu'un utilisateur peut acheter un produit en le mettant dans son panier, il peut avoir plusieurs paniers ou pas du tout, est un panier peut contenir un, plusieurs ou aucun articles.

II. Organisation du code :

1. Controller :

Pour ce projet nous avons créer quatre controllers : HomeController, ProductController, SecurityController et UserController. ProductController contient une fonction pour créer un nouveau produit si le formulaire reçu est reçu et validé. SecurityController contient deux fonction *login* et *logout* pour pouvoir se connecter et se déconnecter du site. Enfin UserController contient trois fonctions, une qui permet de créer un nouvel utilisateur si le formulaire reçu est validé, la seconde de modifier un utilisateur, et la dernière de supprimer un utilisateur.

2. Entity :

Comme écrit auparavant nous avons trois entités : *Cart*, *Product* et *User. Ils ont été créé avec Symfony et comporte les attributs de la base de donnée ainsi que divers fonctions pour accéder à ces attributs.*

3. Form :

Ce projet contient quatre formulaires : ConnectUserType, CreateUserType, ModifyUserType et ProductFormType. Les formulaire ne font rien de plus que ce qui est décrit dans leur nom.

4. Repository :

Nous avons ici trois repository nommés : CartRepository, ProductRepository et UserRepository. C'est fichiers servent à géré les classes de la base de données, donc à créer ou effacer des colonnes.

5. Security :

Afin de gérer les authentifications pour la sécurité nous disposons d'un fichier nommé LoginFormAuthenticator. Ce formulaire permet de vérifier si l'utilisateur est authentifier, il pourra alors ensuite naviguer sur le site.

6. Service :

Le dossier ne contient qu'un fichier nommé PasswordVerificator qui va permettre de créer un service afin de vérifier le mots de passe de l'utilisateur.

7. Templates :

Les templates sont séparés dans différents dossier : *cart, common, home, products, security* et *user*, nous avons aussi le template racine nommé *base,* dont certains hériterons.

Dans le dossier *cart,* il y a un fichier nommé *cart* qui permet d'afficher le panier d'un utilisateur.

Dans le dossier *common nous avons cinq templates, bandeau, erreur, flash, menu,* et *page.* Les quatre premiers vont afficher différents texte comme indiqué dans leur nom sauf *bandeau* qui lui affichera une image et *page* les regroupera dans une page du site et se dernier hérite de *base*.

Dans le dossier *home* nous n'avons qu'un template appelé *home*, il hérite de *base* et permet d'afficher la page d'accueil du site.

Dans le dossier *products* il y a deux fichier un pour afficher les détails d'un produit nommé *productDetail* et l'autre pour afficher la liste des produits *productList*.

Dans *security* nous avons un fichier qui permet d'afficher la page pour se connecter, il se nomme *login*.

Enfin dans *user* nous avons trois fichier, un pour afficher les informations d'un utilisateur, il se nomme *userDetail*. Un autre nommé *userForm* pour afficher le formulaire d'un utilisateur, et un pour afficher la liste de tout les utilisateurs nommé *userList.*

III. Créer un service dans Symfony :

Pour créer un service il faut commencer par créer un fichier .php qui contiendra notre service comme celui créer dans src/Service/PasswordVerificator.php, ensuite nous devons créer un fichier .yml puis il faut y ajouter le code suivant :

services:

nom_du_service

class:nom/classe/creer/avant

puis nous n'avons plus qu'à l'appeler dans nos contrôleurs :

$my_service = $this→container→get('nom_du_service') ;

Si on veut passer des arguments à notre service il faut mettre dans le fichier PROJET/config/services.yaml :

services :

my_service :

class:nom_du_service

arguments :

//ici la liste des argument qu'on veut («@serv» pour un autre service)

IV. Points particulier :

Le site est fonctionnel cependant la partie avec l'outil Mailer n'a pas été réaliser et par inadvertance nous n'avons pas respecter les conditions pour nommer les tables avec le préfixe «i23_».