PROJET JARDINIER

BTS SIO 2ème année

El hamouli, Wassim
BTS SIO 2ère année

documentation technique

Projet : Le P'tit Jardinier

Cette documentation présente l'architecture et les fonctionnalités du projet Le P'tit Jardinier, une application développée avec Symfony. Elle décrit les différentes étapes de configuration, de gestion des données, de création de vues et de contrôleurs, ainsi que la gestion des utilisateurs et des devis.

Routage des Pages:

Définition des Routes:

Les routes déterminent le chemin d'accès aux différentes pages de l'application. Elles sont configurées soit dans des fichiers de configuration dédiés, soit directement via des
annotations dans les contrôleurs.

Création d’une Vue et d’un Contrôleur

Vue (Twig)

Les vues sont écrites en utilisant Twig, un moteur de template intégré à Symfony. Elles permettent de rendre dynamiquement les données envoyées par le contrôleur et de les
afficher dans le navigateur de l'utilisateur.

Contrôleur

Un contrôleur est une classe chargée de traiter les requêtes HTTP et de renvoyer une réponse appropriée. Les actions des contrôleurs sont associées à des routes via des annotations. Chaque action peut ensuite transmettre des données à une vue

Récupération d'une Variable POST:

Gestion des Requêtes

Les données envoyées par un formulaire HTML sont récupérées à l'aide de l'objet Request. Une fois les données extraites, elles sont validées et traitées avant d'être enregistrées ou utilisées pour effectuer des opérations spécifiques.

Gestion des Variables de Session:

Utilisation des Sessions

Symfony gère les sessions via son composant Session, permettant de stocker temporairement des informations liées à l'utilisateur, telles que les données soumises
par un formulaire ou l'état de connexion.

Configuration de la Base de Données:

Connexion à la Base

La configuration de la base de données est définie dans le fichier .env. Symfony utilise Doctrine pour gérer la connexion et l'interaction avec la base de données relationnelle.
Exemple de Configuration :
DATABASE_URL=mysql://user:password@127.0.0.1:3306/nom_de_la_base Voici la configuration que j'ai utilisé :
DATABASE_URL="mysql://root:@127.0.0.1:3306/jardinier?serverVersion=mariadb-11.1.0"

Conception d’un Modèle Objet:

Entités

Les entités sont des classes PHP annotées qui représentent les tables de la base de données, leurs colonnes et leurs relations. Elles sont utilisées par Doctrine pour générer les schémas de base de données et exécuter les opérations CRUD.

Création, Lecture, Modification, Suppression de Données (CRUD):

Création

Les nouvelles entrées sont ajoutées à la base de données via l’Entity Manager en utilisant les méthodes persist et flush.
Lecture Le Doctrine Repository permet de rechercher des données dans la base de données en utilisant des méthodes pré-définies ou personnalisées.

Modification

Les données existantes sont mises à jour en modifiant l’objet de l’entité, puis en
appelant la méthode flush pour appliquer les changements.

Suppression

Les objets sont supprimés de la base de données via l’Entity Manager avec la méthode
remove.

Utilisation des Repositories:

Définition

Les Repositories contiennent la logique nécessaire pour interagir avec les entités et la base de données. Ils permettent d'exécuter des requêtes personnalisées ou d'utiliser des méthodes prédéfinies telles que find, findAll ou findBy.

Génération de Formulaires à Partir de Modèles Objets:

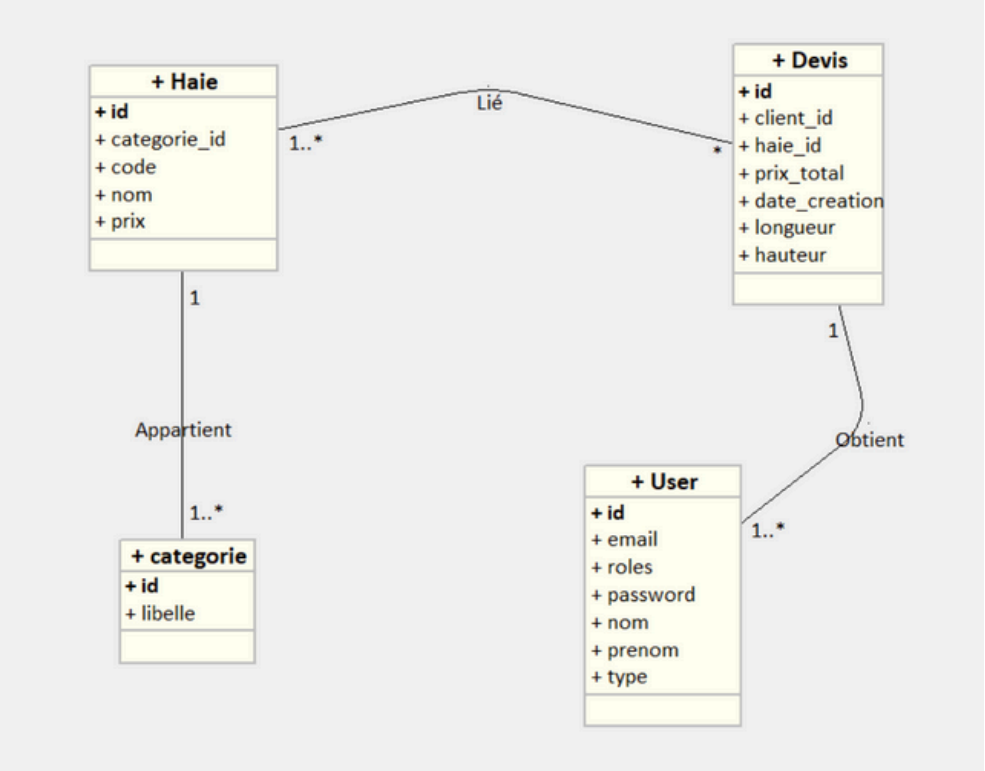
Formulaires Symfony

Symfony permet la génération automatique de formulaires à partir des entités. Ces formulaires peuvent être personnalisés pour inclure des champs spécifiques ou appliquer des styles selon les besoins.

Jointure entre modèles de données:

Symfony Entity

Symfony propose une génération automatique d’entité via une ligne de commande :
php bin/console make :entity



Puis il faut mettre à jour le schéma de base de données :
php bin/console make:migration Grâce aux accesseurs définis dans les modèles il est facile d'accéder aux données liées.

Génération automatique:

Sur Symfony on peut directement générer des entités et un CRUD :
Générer les Getters & Setters : php bin/console make:entity --regenerate App
Générer un CRUD : php bin/console make:crud
J'ai utilisé le CRUD pour les users, cela a facilité de le fait de créer/lire/modifier/supprimer les users.

Gestion des devis et des clients:

Symfony propose un système d'authentification rapide grâce à des commandes simples, telles que symfony console make:user.
Chaque utilisateur se voit attribuer un rôle (particulier ou entreprise), et ce rôle influence directement les prix des devis. Par exemple, un utilisateur avec le rôle "entreprise" bénéficiera d'une réduction de 10% sur son devis.
Chaque utilisateur peut consulter ses propres devis, tandis que l'administrateur a accès
à tous les devis, quelle que soit leur origine. Des restrictions d'accès sont donc
appliquées : certaines routes sont inaccessibles aux utilisateurs simples, tandis que
l'administrateur a la possibilité de gérer les clients, modifier ou supprimer des devis, et
effectuer d'autres actions administratives