

# DOSSIER PROJET

## CONCEPTION D'UN SITE E-COMMERCE

Dossier Projet  
*Développeur web et web Mobile*

William David

Version 1.0, 06/09/2022

# TABLE DES MATIÈRES

1. Présentation du projet . . . . .	1
1.1. Résumé du projet . . . . .	1
1.2. Liste des compétences du référentiel . . . . .	1
1.2.1. Maquetter une application. . . . .	1
1.2.2. Concevoir une base de données. . . . .	1
1.2.3. Mettre en place une base de données . . . . .	2
1.2.4. Développer une interface utilisateur. . . . .	2
1.2.5. Développer des composants d'accès de données . . . . .	2
1.2.6. Développer des pages web en lien avec une base de données . . . . .	2
1.2.7. Mettre en œuvre une solution de e-commerce . . . . .	2
1.2.8. Développer une application simple de mobilité numérique . . . . .	3
1.2.9. Utiliser l'anglais dans son activité professionnelle en informatique . . . . .	3
1.2.10. Actualiser et partager ses compétences en développement informatique . . . . .	3
2. Cahier des charges . . . . .	4
2.1. Présentation de l'entreprise . . . . .	4
2.2. Résumé du projet . . . . .	4
2.3. Objectif du site . . . . .	4
2.4. Spécification fonctionnelle . . . . .	5
2.4.1. Front-Office . . . . .	5
2.4.2. Back-Office . . . . .	5
2.5. Modélisation . . . . .	7
2.5.1. Cas d'usage. . . . .	7
Front-Office . . . . .	7
Back-Office . . . . .	8
2.5.2. Séquences d'actions. . . . .	9
Authentification . . . . .	9

Inscription .....	10
Achat .....	11
Avis.....	13
2.5.3. Entité-Relation .....	14
2.5.4. Plan du site .....	15
2.5.5. Maquettage .....	17
Accueil .....	17
Produit .....	17
Panier .....	18
2.6. Spécification technique .....	19
2.6.1. Symfony .....	20
Grande flexibilité .....	20
Personnalisation.....	20
L'entreprise derrière la technologie .....	21
Une fiabilité éprouvée.....	21
Facile à utiliser .....	21
Support à long terme .....	21
Grande communauté .....	21
Une bonne documentation.....	21
Extensible.....	22
2.6.2. Les Contenus .....	22
2.6.3. Inventaire technique .....	22
3. Réalisation .....	24
3.1. Mise en place de Easy Admin .....	24
3.1.1. Fichier de configuration DashboardController.php .....	24
3.1.2. Exemple de fichier de configuration crud pour la table Avis. . .	25
3.2. Mise en place du module stripe.....	27
3.2.1. Pourquoi Stripe.....	27
3.2.2. Controller PaymentController.php .....	27
3.2.3. Template de succès de paiement .....	31
3.2.4. Template d'erreur de paiement.....	31

3.3. Mise en place de Mailjet .....	32
3.3.1. Présentation .....	32
3.3.2. Pourquoi Mailjet .....	32
3.3.3. Mail.php .....	32
4. Démonstration d'achat de plusieurs produits sur le site .....	34
4.1. Choix des produits .....	34
4.2. Panier .....	34
4.3. Validation du panier .....	35
4.4. Récapitulatif .....	35
4.5. Paiement avec le formulaire stripe .....	35
4.6. Confirmation de la commande .....	36
4.7. Vérification du paiement sur le site Stripe .....	36
4.8. Détail de la commande dans Stripe .....	37
5. Veille sécurité .....	38
5.1. Faille .....	38
5.1.1. Résumé .....	38
5.1.2. Solution .....	38
5.1.3. Documentation .....	38
Bulletin de security Symfony du 29 Janvier 2022 .....	38
5.1.4. Référence CVE CVE-2022-23601 .....	38
6. Recherche Création EasyAdmin .....	39
6.1. Installation .....	39
6.2. Configuration du dashboard .....	39
6.3. Configuration du menu .....	40
6.4. Ajout d'un lien .....	41
6.5. Prise .....	41
7. Traduction du site .....	42
7.1. Extrait du site anglophone .....	42
7.2. Traduction en français .....	43
8. Annexe .....	44
8.1. Source du dossier projet en asciidoctor .....	44

8.2. Glossaire et Définitions .....	75
8.3. Fichier sql de création de base.....	76

# PARTIE 1. PRÉSENTATION DU PROJET

## 1.1. RÉSUMÉ DU PROJET

Le projet a été réalisé sur la période du 1er août au 31 août, pour mon propre compte.

Il met en œuvre un site de commerce électronique de vente de vêtements, avec une vitrine, un catalogue de produits, un moteur de recherche, un système de notation, ainsi qu'un système paiement en ligne par carte bleue.

Le Framework web utilisé est le Framework PHP Symfony (<https://symfony.com/>), accompagné du module EasyAdmin (<https://symfony.com/bundles/EasyAdminBundle/current/index.html>) pour la gestion du Back-Office.

Le Responsive Design est mise en œuvre grâce à l'intégration du Framework CSS Bootstrap (<https://getbootstrap.com/>).

Les notifications par e-mail se grâce au service d'envoi d'e-mail Mailjet (<https://www.mailjet.com/>).

Le paiement en ligne est assuré par le service Stripe (<https://stripe.com/en-fr>).

## 1.2. LISTE DES COMPÉTENCES DU RÉFÉRENTIEL

Chacune des compétences citées ci-dessous, relatives au protocole de la rédaction d'un projet, sont accompagnées de précisions sur les technologies abordées avant d'expliquer leur utilisation dans les parties suivantes.

### 1.2.1. MAQUETTER UNE APPLICATION

La modélisation du projet a été réalisé avec l'outil en ligne <https://www.draw.io/>.

### 1.2.2. CONCEVOIR UNE BASE DE DONNÉES

La conception de la base de données a été réalisé en suivant le modèle Entité-Relation de la méthode Merise.

### 1.2.3. METTRE EN PLACE UNE BASE DE DONNÉES

La création de la base de données relationnelle a été faite avec l'outil d'administration phpMyAdmin.

La matérialisation des tables de données a été faite au travers de l'outil de gestion de version des structures des tables intégré à l'ORM Doctrine, qui génère des scripts de migration.

### 1.2.4. DÉVELOPPER UNE INTERFACE UTILISATEUR

L'interface utilisateur est été réalisée avec le moteur de Template Twig, intégré au Framework Symfony, et le Framework CSS Bootstrap.

### 1.2.5. DÉVELOPPER DES COMPOSANTS D'ACCÈS DE DONNÉES

Les composants d'accès aux données ont été réalisés en s'appuyant sur l'ORM Doctrine, intégré au Framework Symfony. La gestion des données au niveau du Back-Office a été intégrée au module EasyAdmin du Framework Symfony.

### 1.2.6. DÉVELOPPER DES PAGES WEB EN LIEN AVEC UNE BASE DE DONNÉES

Les pages web ont été développé dans le cadre du Framework Symfony, grâce à l'association de l'ORM Doctrine et du moteur de template Twig qui permettent de composer rapidement des pages affichant les données provenant de la base de données.

### 1.2.7. METTRE EN ŒUVRE UNE SOLUTION DE E-COMMERCE

La mise en œuvre du projet, s'appuyant sur une base de données MySql, le Framework PHP Symfony, intégrant le moteur de template Twig, auquel a été ajouté le module de gestion EasyAdmin, forme le squelette d'une solution de e-commerce.

Le paiement en ligne a été mise en œuvre en s'appuyant sur le service Stripe.

Une base de test avec avec des articles, des descriptions, des visuels libres de droit a été créé pour pouvoir présenter la solution.

### 1.2.8. DÉVELOPPER UNE APPLICATION SIMPLE DE MOBILITÉ NUMÉRIQUE

La partie Front-Office du site a été entièrement réalisée avec le Framework CSS Bootstrap. Celui-ci depuis longtemps offre les fonctionnalités nécessaire au Responsive Design.

### 1.2.9. UTILISER L'ANGLAIS DANS SON ACTIVITÉ PROFESSIONNELLE EN INFORMATIQUE

L'utilisation de l'anglais est la norme lors de la conception de programme informatique. Les Frameworks et langages de programmation utilisés ont pour base l'anglais et l'essentiel des tutoriels et documents techniques qui ont été utilisés pour la réalisation de ce projet sont en anglais.

### 1.2.10. ACTUALISER ET PARTAGER SES COMPÉTENCES EN DÉVELOPPEMENT INFORMATIQUE

La réalisation de ce projet a été pour moi l'occasion d'approfondir ce que j'avais vu en cours et de le transposer dans un autre langage.

J'ai pu pour cela m'appuyer sur l'expérience d'un collaborateur freelance qui m'a guidé et conseiller lors de la réalisation de ce projet.

Nous avons échangé par messagerie privée Signal, géré le projet grâce à Trello, suivi les versions et les problèmes sur Github, et effectué du Pair-Programming grâce à GitLive ou en présentiel.



## PARTIE 2. CAHIER DES CHARGES

### 2.1. PRÉSENTATION DE L'ENTREPRISE

La société William David (immatriculation en cours) est une auto-entreprise qui fournit des services de conception et de réalisation de sites web.

### 2.2. RÉSUMÉ DU PROJET

Ce projet met en œuvre un site de commerce électronique de vente de vêtements. Il met en place :

- une vitrine
- un catalogue de produits
- un moteur de recherche
- un système de notation
- un système de prise de commande avec panier
- un système de paiement en ligne par carte bleue
- un système d'authentification et d'autorisation
- un système de gestion des données applicatives (clients, articles, commandes, factures, avis)
- un système de gestion des droits (utilisateurs, administrateurs)

### 2.3. OBJECTIF DU SITE

Le site de vente en ligne de vêtements doit permettre d'acquérir un ou plusieurs articles, de les sélectionner, de les payer, et ce, de la façon la plus fluide et sécurisée.

Pour ce faire, un accent a été mis pour améliorer l'UX des différents scénarios développer pour l'utilisation de ce site web, de la mise en avant et de la recherche de produit, par catégorie jusqu'au paiement en ligne. Le service après vente est assuré par la présence d'un formulaire de contact.

Cependant, pour des questions évidentes de sécurité, tout achat doit se

faire à partir d'un compte existant, à créer le cas échéant.

## 2.4. SPÉCIFICATION FONCTIONNELLE

### 2.4.1. FRONT-OFFICE

L'interface présentée au client doit permettre :

- de présenter le catalogue de produits
- de naviguer de façon fluide dans le catalogue
- de rechercher des produits par différents critères
- d'accéder aux meilleurs ventes du moment
- d'ajouter des produits à son panier
- de consulter son panier
- de supprimer un article de son panier
- de valider son panier après vérification des informations (articles, prix, adresse, livraison...)
- d'accéder au paiement en ligne
- de s'inscrire sur le site
- de se connecter avec un compte existant
- de gérer son compte (adresse, mot de passe, supprimer son compte...)
- de contacter le vendeur par le biais d'un formulaire

### 2.4.2. BACK-OFFICE

L'interface présenté à l'administrateur doit permettre :

- de se connecter avec un compte d'administration
- de gérer les comptes utilisateur et leurs droits d'accès
- de gérer les données applicatives (clients, articles, commandes, factures, avis)
- téléverser de nouveaux documents (photos d'article, fiches technique etc...)
- de consulter les statistiques de fréquentation du site en nombre de

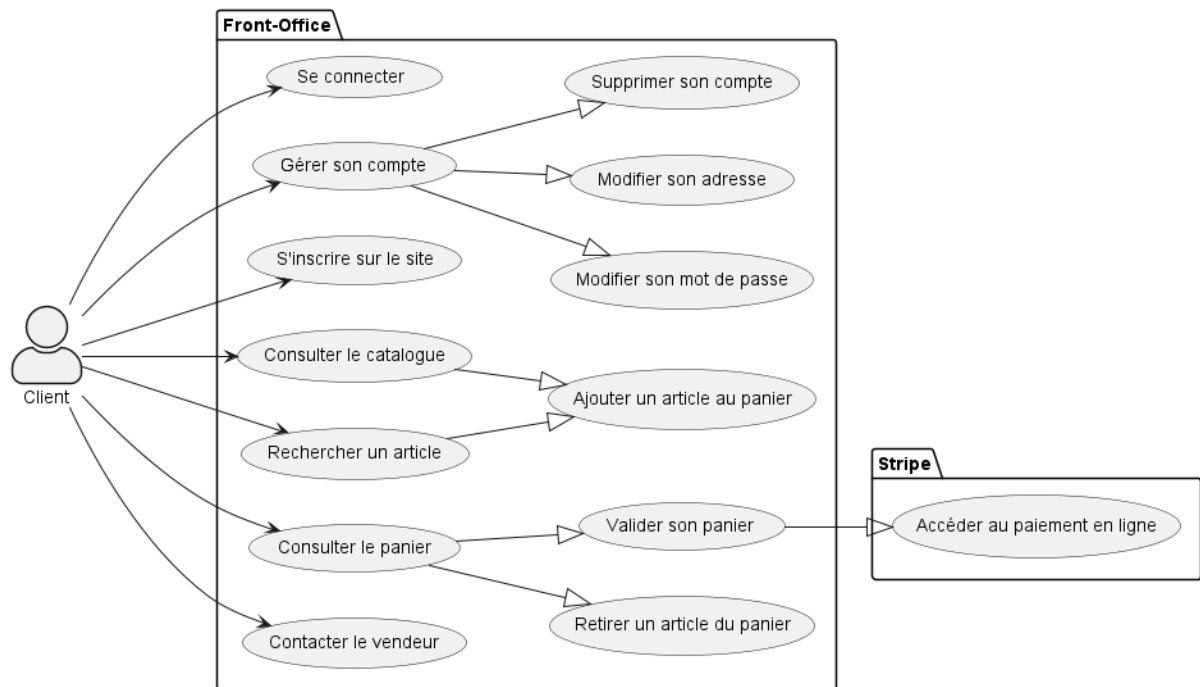
visiteurs

- de consulter les messages envoyés par le biais du formulaire de contact

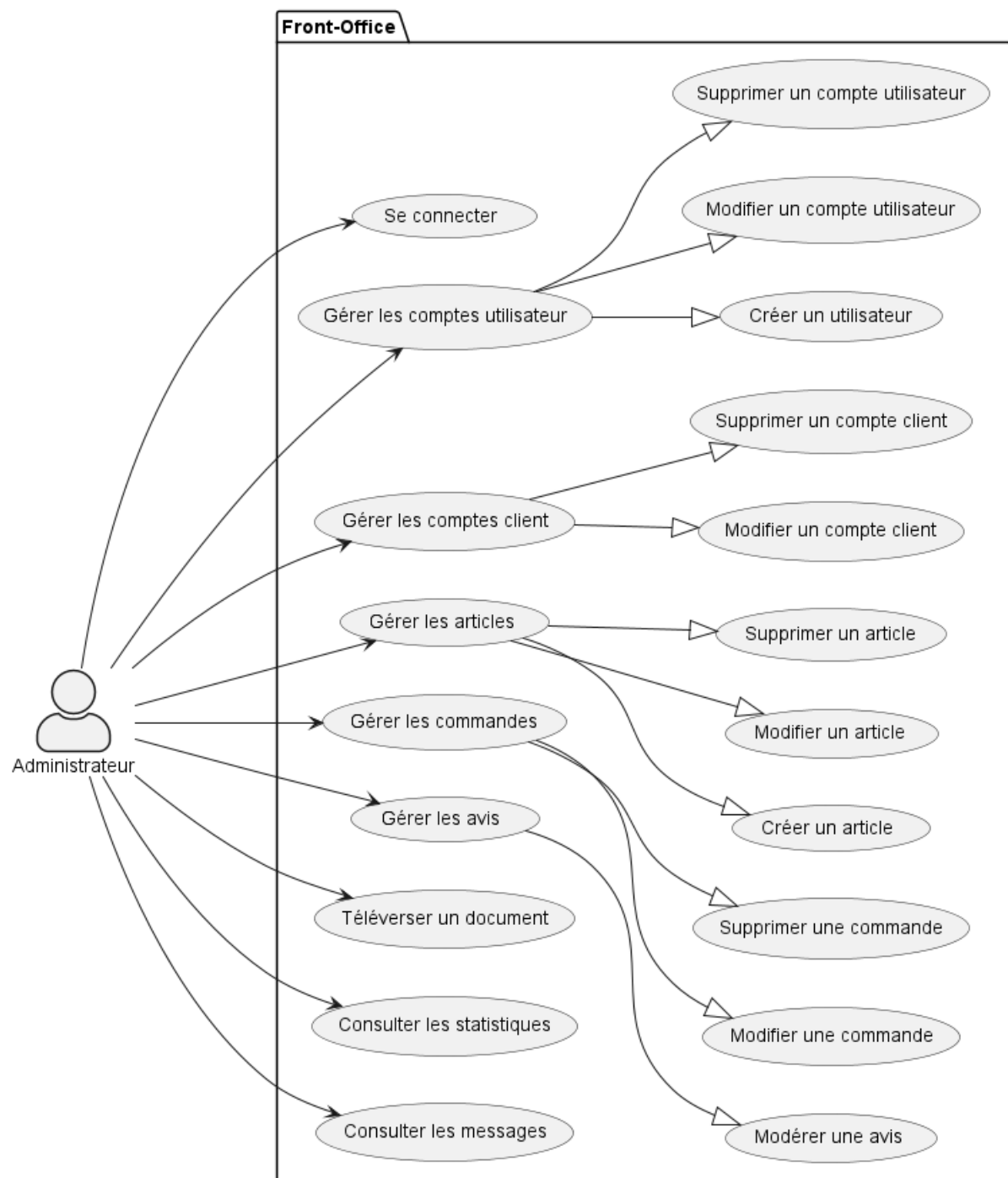
## 2.5. MODÉLISATION

### 2.5.1. CAS D'USAGE

#### FRONT-OFFICE

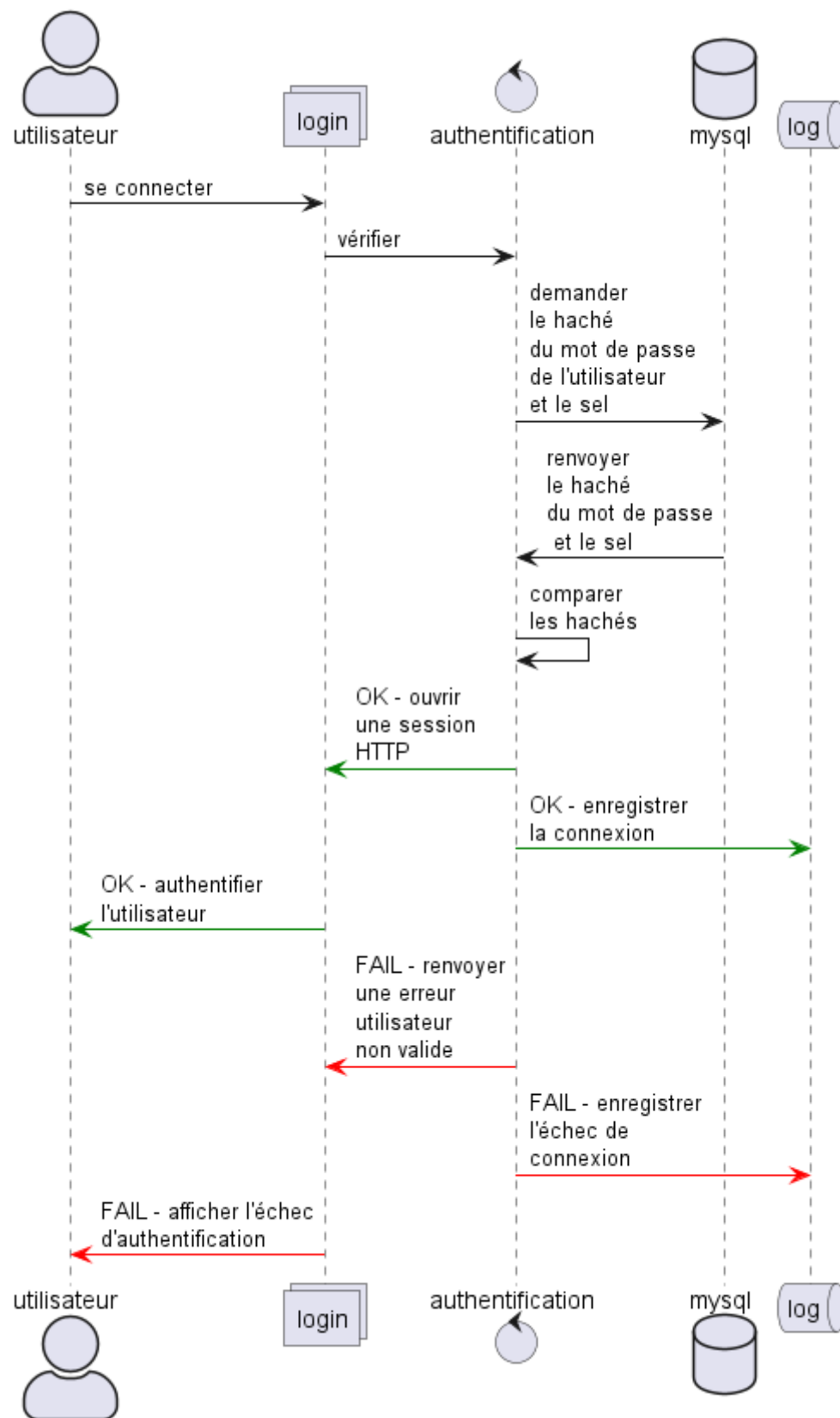


## BACK-OFFICE

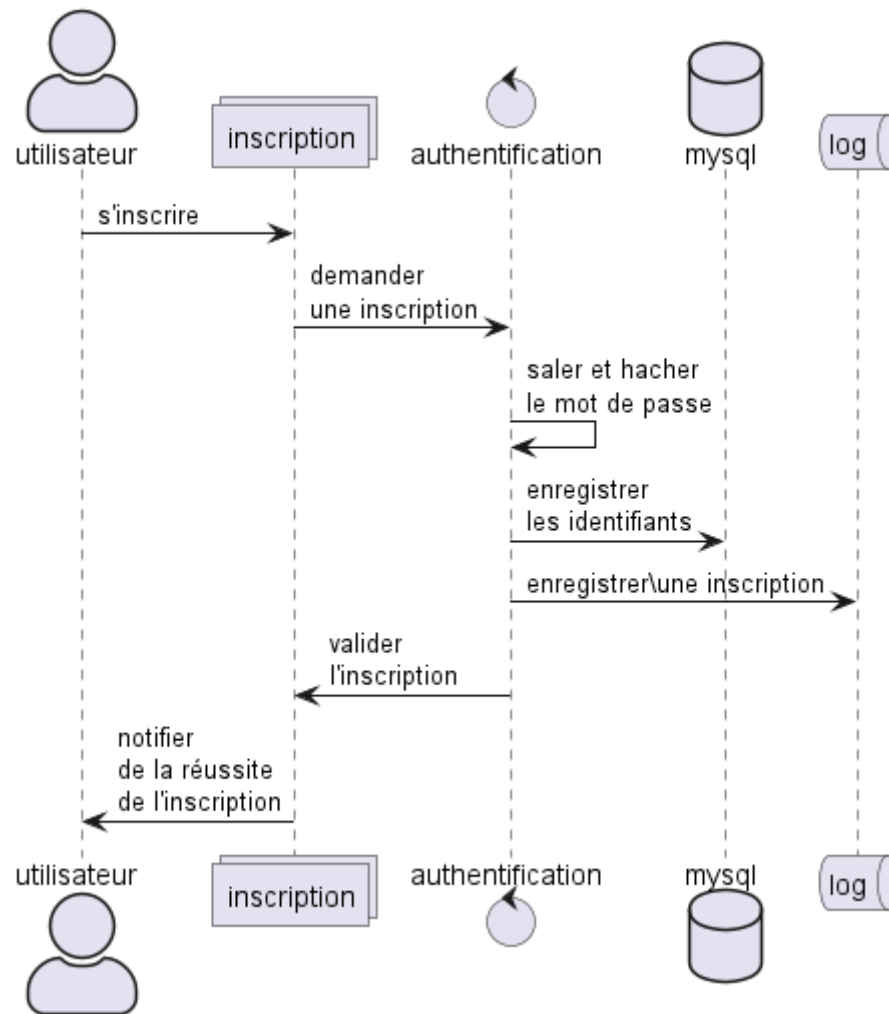


## 2.5.2. SÉQUENCES D'ACTIONS

## AUTHENTIFICATION



## INSCRIPTION

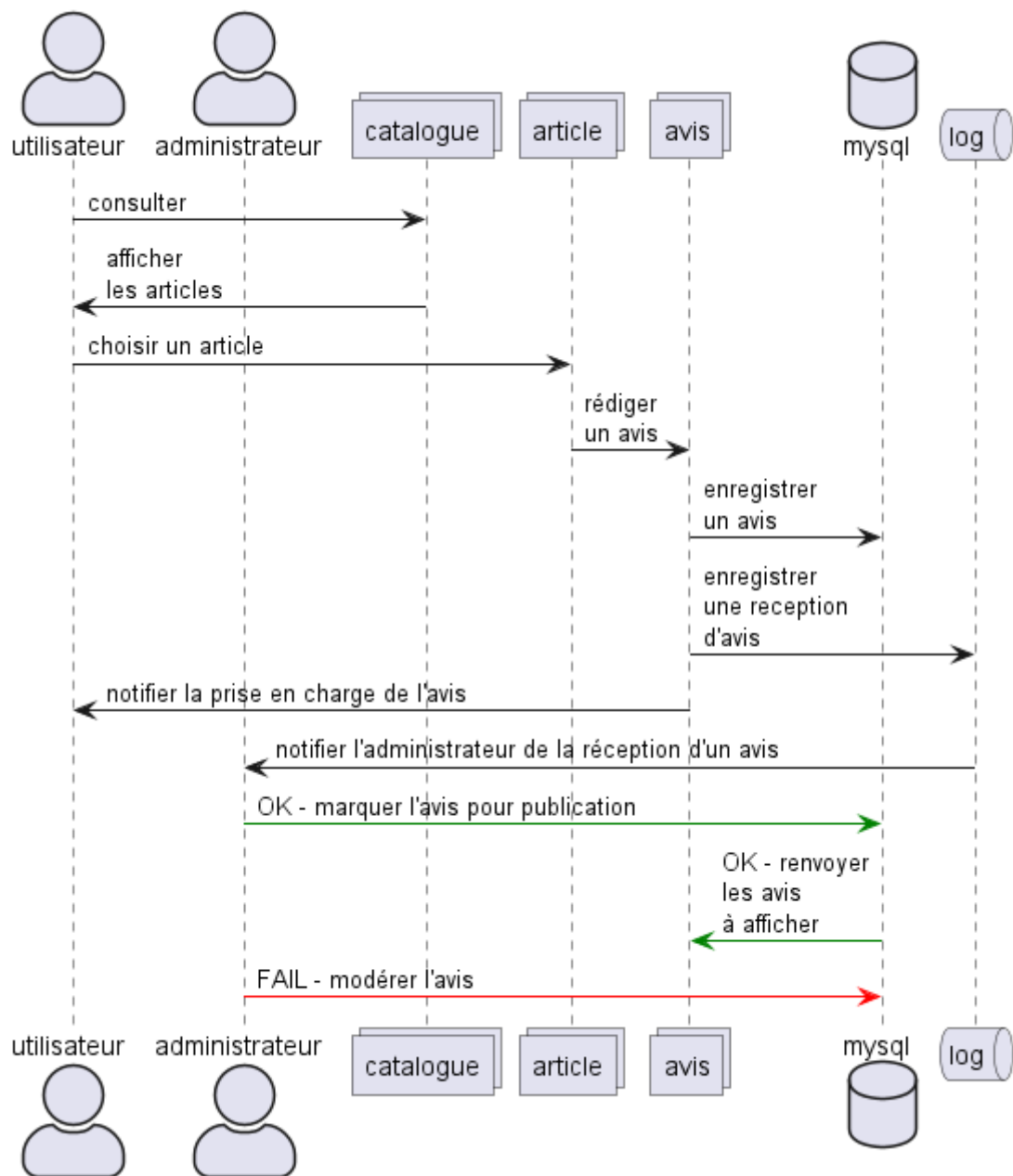


ACHAT

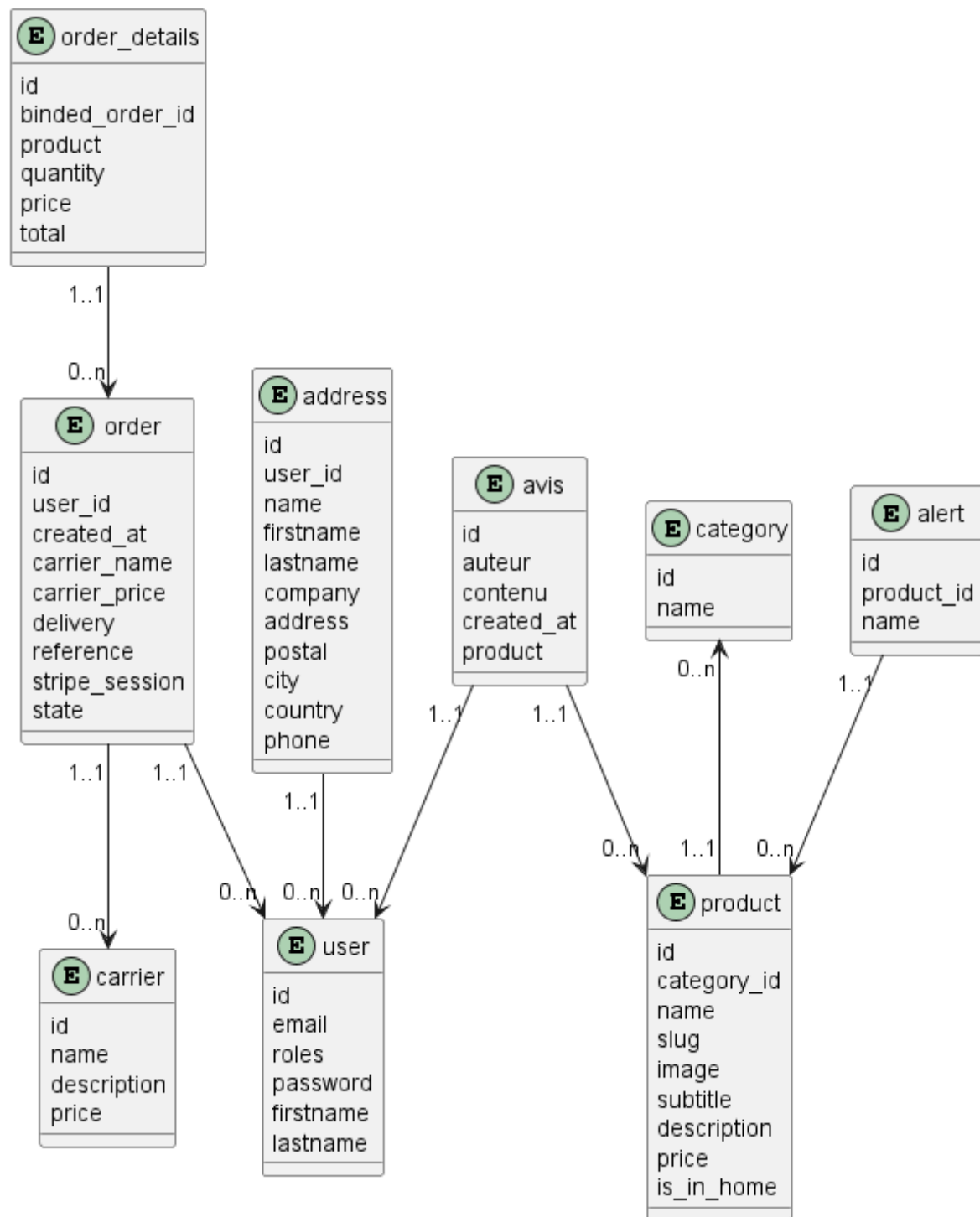




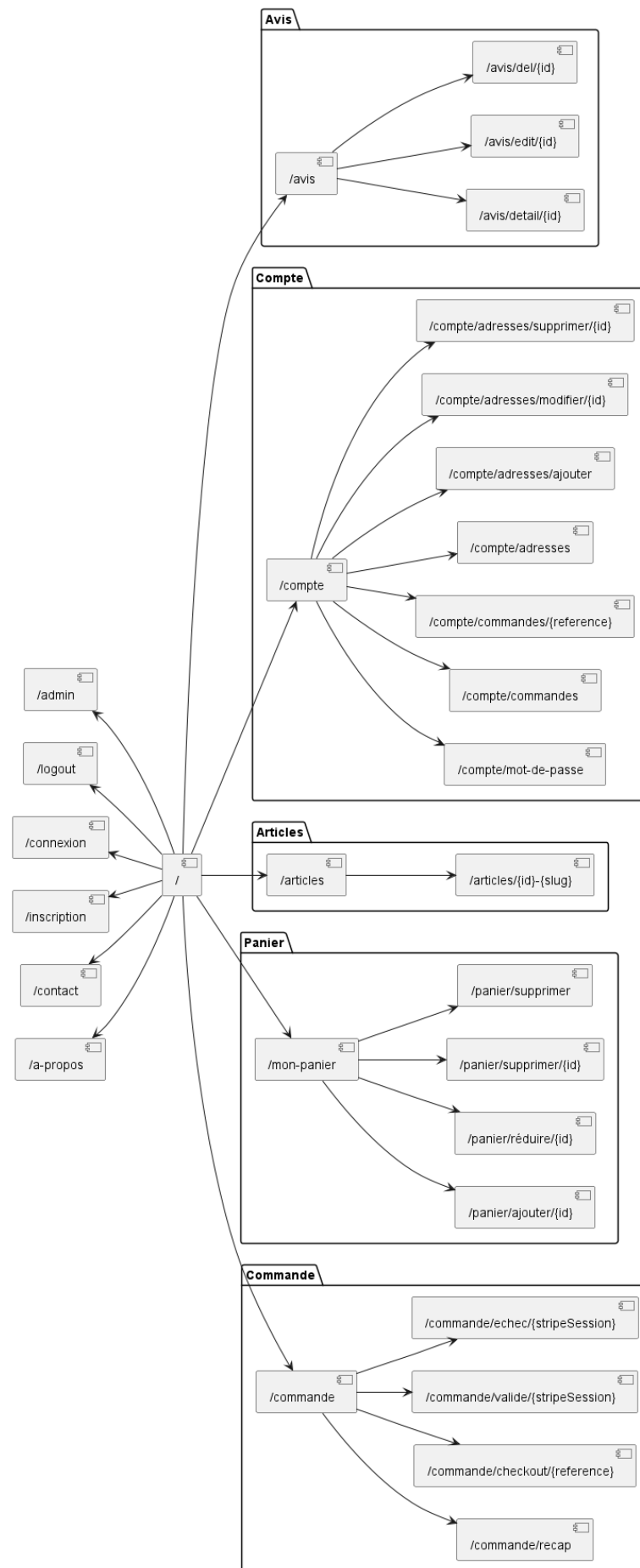
## AVIS



## 2.5.3. ENTITÉ-RELATION

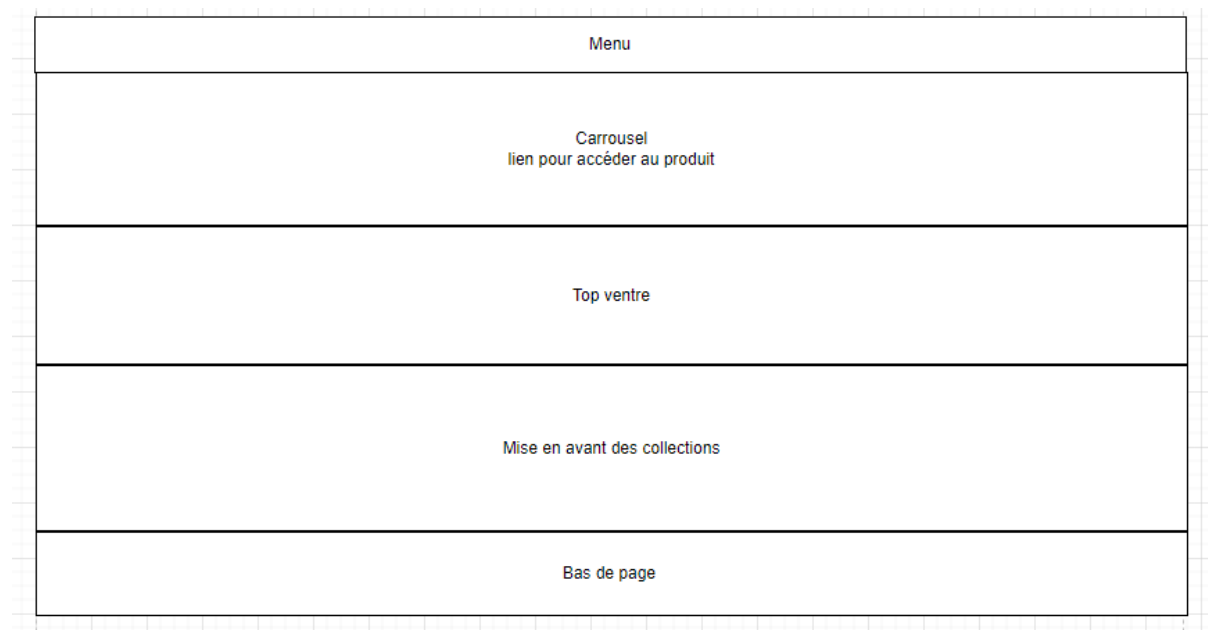


## 2.5.4. PLAN DU SITE

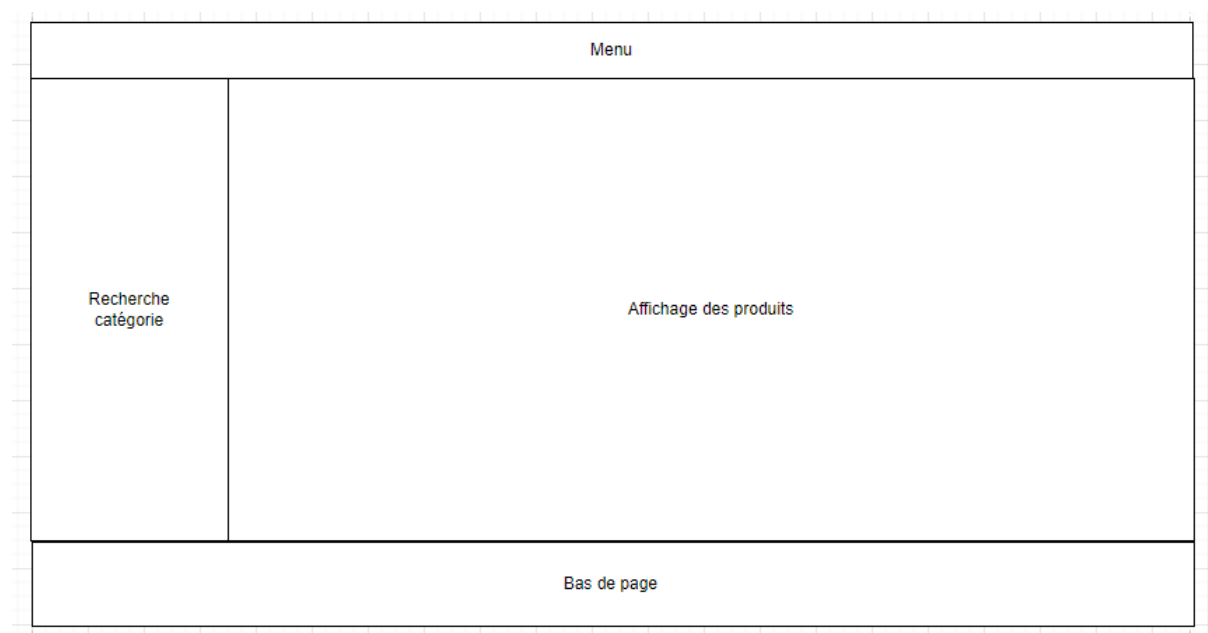


## 2.5.5. MAQUETTAGE

### ACCUEIL



### PRODUIT



## PANIER

Menu
Titre
Liste des produits du panier
Continuer achat
Valider panier
Bas de page
retour

## 2.6. SPÉCIFICATION TECHNIQUE

Le Framework web utilisé est le Framework PHP Symfony (<https://symfony.com/>), accompagné du module EasyAdmin (<https://symfony.com/bundles/EasyAdminBundle/current/index.html>) pour la gestion du Back-Office.

Le Responsive Design est mise en œuvre grâce à l'intégration du Framework CSS Bootstrap (<https://getbootstrap.com/>).

Les notifications par e-mail se grâce au service d'envoi d'e-mail Mailjet (<https://www.mailjet.com/>).

Le paiement en ligne est assuré par le service Stripe (<https://stripe.com/en-fr>).

Utilisateurs :

Le visiteur anonyme est autorisé à consulter la partie vitrine du site. L'administrateur peut se connecter au back office afin de modifier le contenu du site.

J'ai intégré Symfony au projet, et créé les entités Doctrine responsables de la lecture et de l'écriture des données dans la base. Ceci a été réalisé grâce à l'utilitaire en ligne de commande de Symfony.

Une fois les entités créées, il m'a suffit de générer et d'exécuter une migration afin de modifier la structure de la base de données en accord avec le modèle précédemment établi. Doctrine est un ORM (Object Relational Mapper) permettant d'effectuer la lecture et l'écriture des données dans une base de données.

Pour cela, il repose sur son composant DBAL, permettant de faire l'interface avec la base. DBAL permet malgré tout d'utiliser des requêtes SQL traditionnelles pour interagir avec la base de données, mais propose également un système de query builder, moyen alternatif de générer des requêtes SQL offrant, entre autres, une protection contre les injections.

Les divers Repository sont des objets fournis par Doctrine pour chaque entité, permettant de récupérer les données liées à chacune d'entre elles dans la base. Ils permettent également de définir des méthodes pour exécuter des requêtes SQL personnalisées.



## 2.6.1. SYMFONY

Pour développer ce site, j'ai choisi Symfony est l'un des Framework PHP, pour les raisons suivantes :

### GRANDE FLEXIBILITÉ

Symfony est l'un des Frameworks PHP les plus riches en fonctionnalités. Les deux avantages technologiques les plus remarquables de Symfony sont les bundles et les composants.

Le bundle est presque la même chose qu'un plugin. Considérez-le comme un ensemble de fichiers (fichiers PHP, feuilles de style, JavaScripts, images) pour la mise en œuvre d'une fonctionnalité (par exemple, un blog, un panier d'achat, etc.). Le principal avantage des bundles est qu'ils sont découplés. Vous pouvez les reconfigurer et les réutiliser pour de nombreuses applications afin de réduire le coût global de développement.

Les composants sont des fonctionnalités génériques qui réduisent les tâches de routine et permettent aux développeurs de se concentrer sur des fonctionnalités métier spécifiques. Il existe 30 composants Symfony utiles qui facilitent le processus de développement. Vous pouvez utiliser les composants de manière indépendante et ajouter vos propres modules personnalisés sans que l'architecture en pâtisse. Les composants Symfony peuvent également être utilisés de manière autonome dans d'autres frameworks (par exemple, Laravel) ou dans des solutions PHP simples.

Les bundles et les composants permettent d'éliminer les dépendances strictes dans l'architecture. Moins vous avez de dépendances, plus il sera facile d'apporter des changements sans risquer de casser d'autres parties du système. Ainsi, vous pouvez adapter la solution à toutes les exigences et à tous les scénarios d'utilisateur pour créer une application hautement flexible.

### PERSONNALISATION

Symfony offre de grandes caractéristiques et fonctionnalités de personnalisation pour les développeurs et les entreprises.

## L'ENTREPRISE DERRIÈRE LA TECHNOLOGIE

Symfony est l'un des rares frameworks bénéficiant d'un support commercial. SensioLabs, l'entreprise-créditeur et sponsor, contribue activement à sa réputation. Ils fournissent des tutoriels officiels et des certifications. Sur le site Web de l'entreprise, vous trouverez un calendrier des conférences à venir dans le monde entier. Cela montre l'ampleur et le sérieux de leurs intentions et de leurs convictions.

## UNE FIABILITÉ ÉPROUVÉE

Symfony a prouvé sa fiabilité au fil du temps alors que de nombreux autres frameworks ont échoué.

## FACILE À UTILISER

Il existe une documentation complète et détaillée. Elle est considérée comme l'une des meilleures documentations parmi les autres frameworks PHP. Chaque composant est bien expliqué et simplifié par des exemples. De plus, il bénéficie également d'un grand soutien de la communauté. Il offre une configuration facile et un mécanisme de mise en cache pour améliorer les performances des applications.

## SUPPORT À LONG TERME

Symfony est un framework stable et bien testé avec des mises à jour régulières. Les versions les plus récentes bénéficient d'un support à long terme et sont compatibles avec les versions plus récentes : jusqu'à 3 ans pour certaines versions.

## GRANDE COMMUNAUTÉ

Symfony est un open-source, avec une grande communauté. Cela signifie que les experts et les amateurs de PHP du monde entier participent à l'amélioration du code pour tout le monde. Dans la communauté, les gens coopèrent les uns avec les autres. Ils créent de nouveaux composants, essaient de résoudre les problèmes apparus, ou aident les autres avec des conseils.

## UNE BONNE DOCUMENTATION

Une documentation incomplète ou obsolète est un problème pour de

nombreuses technologies. La documentation de Symfony est considérée comme l'une des meilleures, comparée à la documentation des autres frameworks PHP. Elle est clairement écrite, bien structurée, fournie avec des exemples, et mise à jour de version en version. Vous pouvez trouver une explication de chaque composant et du processus de développement dans son ensemble.

#### EXTENSIBLE

Tout dans le framework Symfony se représente comme un bundle. Chaque bundle a une fonctionnalité unique. Vous pouvez réutiliser le bundle dans d'autres projets et le partager avec la communauté également. C'est également l'une des raisons qui le rendent populaire auprès des développeurs. La meilleure partie est que vous pouvez changer ou modifier n'importe quoi, même le noyau du système sans reconfigurer le framework complet. Vous pouvez ajouter les fonctionnalités dont vous avez besoin et étendre les caractéristiques d'une application autant que vous le souhaitez.

#### 2.6.2. LES CONTENUS

Tout le contenu de ce projet (image, photos, logo, textes) sont libres de droit et d'utilisation.

#### 2.6.3. INVENTAIRE TECHNIQUE

- Visual Studio Code
- Framework PHP Symfony
- Gestion du Back-Office avec EasyAdmin
- Moteur de Template Twig
- Accès à la base de donnée par l'ORM Doctrine
- Base de données MySql
- Gestion de la base de données avec phpMyAdmin et Doctrine
- Serveur HTTP Apache
- Gestion du paiement avec Stripe
- Gestion d'envoi de mail avec Mailjet

- Gestion du Responsive Design avec le Framework CSS Bootstrap

## PARTIE 3. RÉALISATION

### 3.1. MISE EN PLACE DE EASY ADMIN

En Utilisant le site de symfony (<https://symfony.com/bundles/EasyAdminBundle/current/dashboards.html>), j'ai pu comprendre le fonctionnement de EasyAdmin et l'implémenter dans mon projet.

#### 3.1.1. FICHIER DE CONFIGURATION DASHBOARDCONTROLLER.PHP

```
<?php

namespace App\Controller\Admin;

use App\Entity\Avis;
use App\Entity\User;
use App\Entity\Order;
use App\Entity\Carrier;
use App\Entity\Headers;
use App\Entity\Product;
use App\Entity\Category;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
use EasyCorp\Bundle\EasyAdminBundle\Config\MenuItem;
use EasyCorp\Bundle\EasyAdminBundle\Config\Dashboard;
use EasyCorp\Bundle\EasyAdminBundle Router\AdminUrlGenerator;
use EasyCorp\Bundle\EasyAdminBundle\Controller\AbstractDashboardController;

class DashboardController extends AbstractDashboardController
{
    /**
     * @Route("/admin", name="admin")
     */
    public function index(): Response
    {
        // redirect to some CRUD controller
        $routeBuilder = $this->get(AdminUrlGenerator::class);

        return $this->redirect($routeBuilder->
setController(OrderCrudController::class)->generateUrl());
    }

    public function configureDashboard(): Dashboard
    {
        return Dashboard::new()
```

```

        ->setTitle('Ma Boutique'); // Titre du Back Office

    }

    public function configureMenuItems(): iterable
    {
        // linkToDashboard permet de créer le home du menu
        yield MenuItem::linkToDashboard('Tableau de bord', 'fa fa-home');
        // linkToCrud permet de créer les menus en les reliant à une table
        yield MenuItem::linkToCrud('Utilisateurs', 'fas fa-user', User::
class);
        yield MenuItem::linkToCrud('Catégories', 'fas fa-list',
Category::class);
        yield MenuItem::linkToCrud('Produits', 'fas fa-tag', Product::class);
        yield MenuItem::linkToCrud('Transporteurs', 'fas fa-truck',
Carrier::class);
        yield MenuItem::linkToCrud('Commandes', 'fas fa-shopping-cart',
Order::class);
        yield MenuItem::linkToCrud('Avis', 'fas fa-desktop', Avis::class);
        yield MenuItem::linkToCrud('Bannières', 'fas fa-desktop',
Headers::class);
        return [ // linkToRoute permet de créer un lien pour retourner au site
            yield MenuItem::linkToRoute('Retour', 'fa fa-home', 'home')
        ];
    }
}

```

Le fichier ci-dessus, est DashboardController.php, il permet de créer le menu du back office sur le côté gauche.

### 3.1.2. EXEMPLE DE FICHIER DE CONFIGURATION CRUD POUR LA TABLE AVIS

Les commandes utilisées pour créer les entités et les scripts de migration sont :

```

php bin/console make:entity {nom de la table}
php bin/console make:migration
php bin/console doctrine:migrations:migrate

```

```

<?php

namespace App\Controller\Admin;

```

```

use App\Entity\Avis;
use EasyCorp\Bundle\EasyAdminBundle\Config\Crud;
use EasyCorp\Bundle\EasyAdminBundle\Config\Actions;
use EasyCorp\Bundle\EasyAdminBundle\Field\SlugField;
use EasyCorp\Bundle\EasyAdminBundle\Field\TextField;
use EasyCorp\Bundle\EasyAdminBundle\Field\ImageField;
use EasyCorp\Bundle\EasyAdminBundle\Field\MoneyField;
use EasyCorp\Bundle\EasyAdminBundle\Field\BooleanField;
use EasyCorp\Bundle\EasyAdminBundle\Field\DateTimeField;
use EasyCorp\Bundle\EasyAdminBundle\Field\TextareaField;
use EasyCorp\Bundle\EasyAdminBundle\Field\AssociationField;
use EasyCorp\Bundle\EasyAdminBundle\Controller\AbstractCrudController;

class AvisCrudController extends AbstractCrudController
{
    public static function getEntityFqcn(): string
    {
        return Avis::class;
    }

    public function configureFields(string $pageName): iterable
    {
        return [
            TextField::new('auteur', 'Auteur'), // Relis le champs auteur à une
            // colonne Auteur dans le tableau
            TextareaField::new('contenu')->hideOnIndex(),
            DateTimeField::new('created_at', 'Créée le')
        ];
    }

    public function configureCrud(Crud $crud): Crud
    {
        return $crud
            ->setEntityLabelInSingular('Avis')
            ->setEntityLabelInPlural('Avis')
            ;
    }
}

// --
// -- Structure de la table `avis`
// --

// CREATE TABLE `avis` (
//   `id` int(11) NOT NULL,
//   `product_id` int(11) NOT NULL,
//   `auteur` varchar(255) NOT NULL,

```

```
// `contenu` longtext NOT NULL,  
// `created_at` datetime NOT NULL  
// ) ;
```

Le fichier `AvisCrudController.php` permet de configurer le CRUD pour la table avis.

## 3.2. MISE EN PLACE DU MODULE STRIPE

### 3.2.1. POURQUOI STRIPE

Stripe est un outil efficace de paiement en ligne qui permet de transférer de l'argent du compte bancaire de votre client vers le compte de votre entreprise, par le biais de carte de crédit. Stripe est un module ergonomique qui s'harmonise parfaitement au style de votre site. De plus, il est facile d'utilisation par votre client. Cet atout vous permet d'augmenter la satisfaction de vos clients. Stripe assure un niveau de sécurité élevé, vous permettant de recevoir vos paiements en toute fiabilité. C'est aussi une solution avantageuse pour vos clients, car tous les frais sont contrôlés par son site marchand, afin d'éviter tout acte malveillant. Pour se prémunir contre les litiges avec les clients, Stripe vous offre un contrat VAD (Vente à Distance) que vous souscrivez lors de l'achat d'un abonnement Stripe. La fiabilité et la sécurité optimale de Stripe en font la solution la plus prisée par plusieurs e-commerçants qui utilisent des CMS très populaires à l'instar de Prestashop et Shopify.

### 3.2.2. CONTROLLER PAYMENTCONTROLLER.PHP

```
<?php  
  
namespace App\Controller;  
  
use App\Entity\Order;  
use App\Model\Cart;  
use App\Repository\OrderRepository;  
use App\Service\Mail;  
use Doctrine\ORM\EntityManagerInterface;  
use Stripe\Checkout\Session;  
use Stripe\Stripe;  
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;  
use Symfony\Component\HttpFoundation\Response;
```



```

use Symfony\Component\Routing\Annotation\Route;

class PaymentController extends AbstractController
{
    /**
     * Etape de vérification avant confirmation du paiement
     */
    /**
     * @Route("/commande/checkout/{reference}", name="checkout")
     */
    public function payment(OrderRepository $repository, $reference,
        EntityManagerInterface $em): Response
    {
        // Récupération des produits de la dernière commande et formattage
        dans un tableau pour Stripe
        $order = $repository->findOneByReference($reference);
        if (!$order) {
            throw $this->createNotFoundException('Cette commande n\'existe
pas');
        }
        $products = $order->getOrderDetails()->getValues();
        $productsForStripe = [];
        foreach ($products as $item) {
            $productsForStripe[] = [
                'price_data' => [
                    'currency' => 'eur',
                    'unit_amount' => $item->getPrice(),
                    'product_data' => [
                        'name' => $item->getProduct()
                    ]
                ],
                'quantity' => $item->getQuantity()
            ];
        }
        // Ajout des frais de livraison
        $productsForStripe[] = [
            'price_data' => [
                'currency' => 'eur',
                'unit_amount' => $order->getCarrierPrice(),
                'product_data' => [
                    'name' => $order->getCarrierName()
                ]
            ],
            'quantity' => 1
        ];
        // Une clé est nécessaire pour utiliser Stripe fournit dans leur site
        Stripe::setApiKey

```

```

('sk_test_51LNyQsDz6qM0cya0B1HWM8Y6a3k7rAG040C2L3qxGbI9f5XhsxzUAeqgrhhKYEmSMEH
AgZ3uI33kjfR96pZN0lpb00NMC07VJA');
    header('Content-Type: application/json');

    // $YOUR_DOMAIN = 'https://ecommerce.fr';
    $YOUR_DOMAIN = 'http://localhost:8080';

    // Création de la session Stripe avec les données du panier
    $checkout_session = Session::create([
        'line_items' => $productsForStripe,
        'mode' => 'payment',
        'success_url' => $YOUR_DOMAIN .
        '/commande/valide/{CHECKOUT_SESSION_ID}',
        'cancel_url' => $YOUR_DOMAIN .
        '/commande/echec/{CHECKOUT_SESSION_ID}',
    ]);
    $order->setStripeSession($checkout_session->id);
    $em->flush();
    return $this->redirect($checkout_session->url);
}

/**
 * Méthode appelée lorsque le paiement est validé
 */
/**
 * @Route("/commande/valide/{stripeSession}", name="payment_success")
 */
public function paymentSuccess(OrderRepository $repository,
$stripeSession, EntityManagerInterface $em, Cart $cart)
{
    $order = $repository->findOneByStripeSession($stripeSession);
    if (!$order || $order->getUser() != $this->getUser()) {
        throw $this->createNotFoundException('Commande inaccessible');
    }
    if (!$order->getState()) {
        $order->setState(1);
        $em->flush();
    }

    // Envoi mail de Confirmation
    $user = $this->getUser();

    $content = "Bonjour {$user->getFirstname()} nous vous remercions de
votre commande";
    (new Mail)->send(
        $user->getEmail(),
        $user->getFirstname(),

```

```

        "Confirmation de la commande {$order->getReference()}",
        $content
    );

    // Suppression du panier une fois la commande validée
    $cart->remove();
    return $this->render('payment/success.html.twig', [
        'order' => $order
    ]);
}

/**
 * Commande annulée (clic sur retour dans la fenêtre)
 */
/**
 * @Route("/commande/echec/{stripeSession}", name="payment_fail")
 */
public function paymentFail(OrderRepository $repository, $stripeSession)
{
    $order = $repository->findOneByStripeSession($stripeSession);
    if (!$order || $order->getUser() != $this->getUser()) {
        throw $this->createNotFoundException('Commande inaccessible');
    }

    return $this->render('payment/fail.html.twig', [
        'order' => $order
    ]);
}
}

```

Ce Contrôleur permet de mettre en place le module stripe et de lui fournir les données du panier, les informations produits et informations livraisons.

Il gère aussi le cas si le paiement réussi (méthode paymentSuccess) ou échoue (méthode paymentFail).

En déléguant le paiement à Stripe, la sécurité est gérée totalement par ce module qui est stable et connaît son domaine.

### 3.2.3. TEMPLATE DE SUCCÈS DE PAIEMENT

```
{% extends 'base.html.twig' %}

{% block title %}Ma commande - Ma Boutique{% endblock %}

{% block body %}
    <h2>Confirmation de votre commande</h2>
    <p>
        Bonjour {{order.user.firstname}} {{order.user.lastname}}, <br>
        Nous vous remercions de votre commande n°
    <b>{{order.reference}}</b>.<br>
        Une confirmation vient de vous être envoyé par mail. <br>
    </p>
    <hr>
    <p>
        Votre commande sera livrée par {{order.carrierName}} à l'adresse
        suivante: <br>
        {{order.delivery|raw}}
    </p>
    <hr>
    <p>
        Pour suivre votre commande, rendez-vous dans votre <a href="{{
        path('account_orders') }}">compte</a>.
    </p>
{% endblock %}
```

### 3.2.4. TEMPLATE D'ERREUR DE PAIEMENT

```
{% extends 'base.html.twig' %}

{% block title %}Ma commande - Ma Boutique{% endblock %}

{% block body %}
    <h2>Annulation de votre commande</h2>
    <p>
        Bonjour {{order.user.firstname}} {{order.user.lastname}}, <br>
        Votre paiement pour la commande n° <b>{{order.reference}}</b> n'a pas
        abouti.<br>
    </p>
    <hr>
    <a class="btn btn-outline-success" href="{{ path('order') }}">
    <b>Réessayer</b></a>
{% endblock %}
```

## 3.3. MISE EN PLACE DE MAILJET

### 3.3.1. PRÉSENTATION

Mailjet est un système d'envoi et de suivi d'emails basé dans le cloud<sup>5</sup>. La plateforme permet aux professionnels d'envoyer tant leurs emails marketing (newsletters, offres promotionnelles) que leurs emails transactionnels (notifications, confirmations d'inscription, de commande, factures...). Les services de Mailjet comprennent des solutions de conception d'emails, d'envoi de volumes massifs et de suivi de ces envois.

### 3.3.2. POURQUOI MAILJET

J'ai choisit Mailjet car il donne la possibilité d'envoyer des mails et leur suivis. Ensuite Mailjet me permettra de faire des newsletters, offres promotionnelles et autre. Mailjet permet de réaliser des rapports de campagnes plutôt précis. Pour chaque campagne d'email marketing que vous menez, vous pouvez connaître en temps réel le pourcentage d'emails délivrés, de clics, d'emails ouverts, de désabonnement, de signalement comme spam ou encore de messages d'erreur.

### 3.3.3. MAIL.PHP

```
<?php
namespace App\Service;

use Mailjet\Client;
use Mailjet\Resources;

class Mail
{
    private $api_key = "c2a1cd58ae21be0451736ab830498846";
    private $api_key_secret = "a8b58a736745704c0d59ee211014fe5c";

    public function send($toEmail, $toName, $subject, $content)
    {
        $mj = new Client($this->api_key, $this->api_key_secret, true, ['version'
=> 'v3.1']);
        $body =
        [
            'Messages' =>
            [
                [
```

```
'From' =>
[
    'Email' => "sportif_wd@hotmail.fr",
    'Name' => "Ma Boutique"
],
'To' =>
[
    [
        'Email' => $toEmail,
        'Name' => $toName
    ]
],
'Subject' => $subject,
'TextPart' => $content,
'HTMLPart' => $content,
]
];
$response = $mj->post(Resources::$Email, ['body' => $body]);

// Read the response / Lecture de la reponse

$response->success();
}
?>
```

# PARTIE 4. DÉMONSTRATION D'ACHAT DE PLUSIEURS PRODUITS SUR LE SITE

## 4.1. CHOIX DES PRODUITS

Cette page affiche les produits et permet de filtrer les produits par catégorie.

Ma BoutiqueProduits🛒 + 540,00 €

Mon compteMon Déconnexion

Rechercher


Mots-clés:

Votre recherche


☐ Manteaux  
☐ Echarpes  
☐ Bonnets  
☐ T-shirts  
☐ Chaussures  
☐ Lunettes  
☐ Chapeaux  
☐ Casquettes  
☐ Montres

Valider


Nos Produits




BONNET ROUGE POMPOM  
Restez au chaud avec style  
25,99 €





BONNET ROUGE ET BLEU POMPOM  
Restez au chaud avec style (encore)  
19,99 €



ECHARPE ROUGE  
Pour le ski ou la ville  
24,00 €







Nous avons sélectionné quelques produits pour la démonstration.

## 4.2. PANIER

Cette page, le panier, affiche les produits choisis afin d'être achetés.




Ma BoutiqueProduits🛒 + 540,00 €

Mon compteMon Déconnexion

Mon Panier

Voici les articles que vous avez ajoutés

Réinitialiser mon panier

Article	Prix (unitaire)	Quantité	Total
 T-Shirt Moulant	20,00 €	- 1 +	20,00 €
 Echarpe Rayée	20,00 €	- 1 +	20,00 €
 Montre	500,00 €	- 1 +	500,00 €
Total		x 3	540,00 €

Continuez mes achats

Valider mon panier

Qui sommes-nous ? Contact © 2022 Ma Boutique Mentions légales

Retour haut de page

4.1. Choix des produits

34

## 4.3. VALIDATION DU PANIER

Cette page affiche le montant de la commande, et permet d'indiquer l'adresse de livraison et le choix du transporteur, ce qui rajoute un coût de livraison.

Ma Boutique

Produits

+ 540,00 €

Mon compte

Déconnexion

Je valide ma commande

Choisissez votre adresse de livraison

[Ajouter nouvelle adresse](#)

☐ C:

1 rue de l'union  
bobigny...

Choisissez votre transporteur

☐ Chronopost:


Livraison standard à domicile en 2 jours ouvrés.  
4.00 €

☐ La Poste:


Si vous préférez trouver dans votre boîte à lettres un avis de  
passage à la place de votre commande.  
1.90 €

Passer au paiement


Récapitulatif :

T-Shirt Moulant

x1

Echarpe Rayée

x1

Montre

x1

Total

540,00 €

## 4.4. RÉCAPITULATIF

Cette page est un récapitulatif du montant de la commande et des frais de livraisons. Pour cette commande nous avons un montant de 541,90 Euro que nous devons retrouver dans le site Stripe, rubrique paiements.

Ma Boutique

Produits

+ 540,00 €

Mon compte

Déconnexion

Mon récapitulatif

Mon adresse de livraison

william william david

0650013186

sii

1 rue de l'union

93000


bobigny

FR


Transporteur

La Poste


Vos articles

T-Shirt Moulant

x1

Echarpe Rayée

x1

Montre

x1

Total articles

540,00 €

Livraison

1,90 €

Total commande

541,90 €

Payer

Qui sommes-nous ? Contact © 2022 Ma Boutique Mentions légales

Retour haut de page

## 4.5. PAIEMENT AVEC LE FORMULAIRE STRIPE

Cette page est fournie par Stripe, et on voit afficher la désignation des



produits et les prix et le montant total, ce qui doit, après validation sur le bouton payer, être afficher dans le site Stripe rubrique paiement.

← Ma Boutique **TEST MODE**

Payer Ma Boutique

## 541,90 €

T-Shirt Moulant	20,00 €
Echarpe Rayée	20,00 €
Montre	500,00 €
La Poste	1,90 €

Propulsé par **stripe** | [Conditions d'utilisation](#) [Confidentialité](#)

**Coordonnées**

E-mail

**Moyen de paiement**

Carte bancaire Bancontact iDEAL ...

**Informations de la carte**

1234 1234 1234 1234

MM / AA CVC

**Nom du titulaire de la carte**

**Pays ou région**

France

☐ Enregistrer mes informations pour le paiement sécurisé en 1 clic  
Réglez plus rapidement sur Ma Boutique et des milliers d'autres sites.

**Payer**

## 4.6. CONFIRMATION DE LA COMMANDE

Cette page s'affiche lorsque le paiement s'est bien effectué, nous irons vérifier sur le site de Stripe que les informations correspondent.

Ma Boutique Produits + 0,00 € Mon compte [Ajouter](#) Déconnexion

### Confirmation de votre commande

Bonjour william david,  
Nous vous remercions de votre commande n° 20220829151824-630cbca0c4948.  
Une confirmation vient de vous être envoyée par mail.

Votre commande sera livrée par La Poste à l'adresse suivante:  
william william david  
0650013186  
sii  
1 rue de l'union  
93000  
bobigny  
FR

Pour suivre votre commande, rendez-vous dans votre [compte](#).

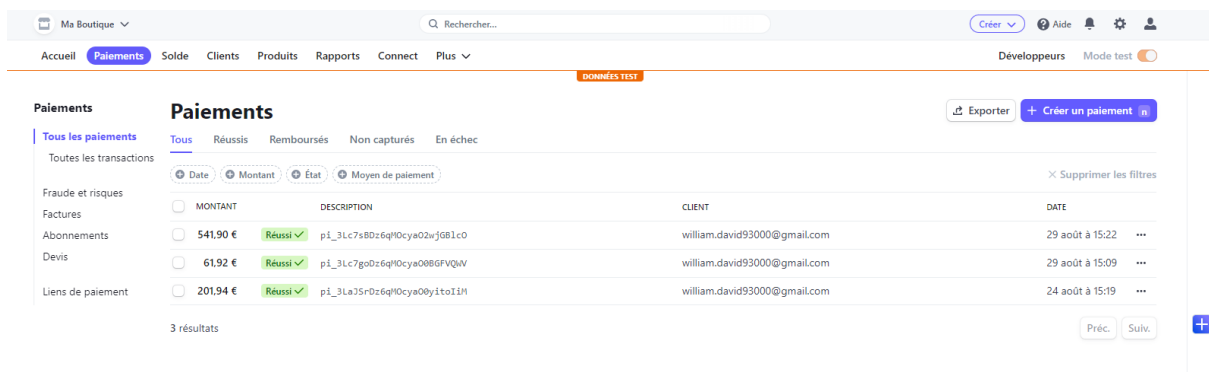
[Qui sommes-nous ?](#) [Contact](#) © 2022 Ma Boutique [Mentions légales](#)

[Retour haut de page](#)

## 4.7. VÉRIFICATION DU PAIEMENT SUR LE SITE STRIPE.

Notre vérification montre bien que la ligne de notre commande est d'un

montant de 541.90 Euro. Pour voir les détails, il faut cliquer sur la ligne ce qui affiche le détail de la commande.

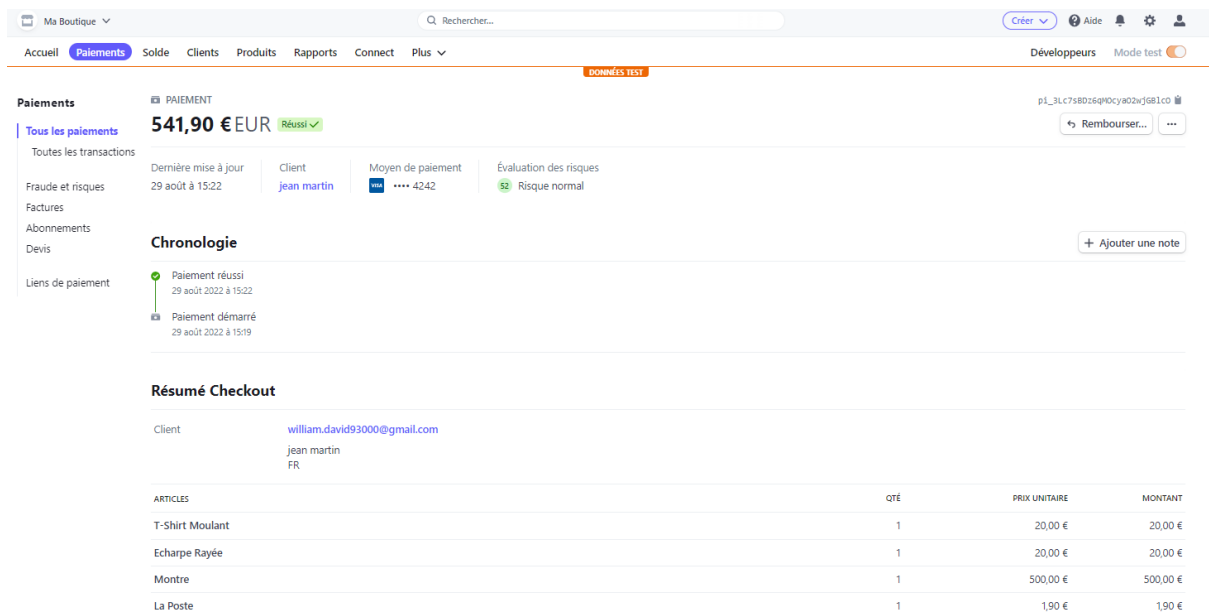


The screenshot shows the Stripe Payments interface. The top navigation bar includes 'Ma Boutique', a search bar, and buttons for 'Créer', 'Aide', and user settings. The main menu has 'Accueil', 'Paielements', 'Solde', 'Clients', 'Produits', 'Rapports', 'Connect', and 'Plus'. The 'Paielements' section is active, showing a list of transactions. The table has columns for 'MONTANT', 'DESCRIPTION', 'CLIENT', and 'DATE'. Three transactions are listed, all with a status of 'Réussi'.

MONTANT	DESCRIPTION	CLIENT	DATE
541.90 €	p1_3Lc7s8Dz6qH0cy802wJGB1c0	william.david93000@gmail.com	29 août à 15:22
61.92 €	p1_3Lc7g0z6qH0cy808GFVQW	william.david93000@gmail.com	29 août à 15:09
201.94 €	p1_3La7Sr0z6qH0cy80yito1Ij	william.david93000@gmail.com	24 août à 15:19

## 4.8. DÉTAIL DE LA COMMANDE DANS STRIPE

Dans le détail de la commande, nous avons les informations comme le montant, la date de paiement, le nom et le mail du client, et les informations sur les articles achetés.



The screenshot shows the Stripe Payment Detail interface for a specific transaction. The top navigation bar is the same as the previous screenshot. The main menu has 'Accueil', 'Paielements', 'Solde', 'Clients', 'Produits', 'Rapports', 'Connect', and 'Plus'. The 'Paielements' section is active, showing the details of a payment of 541.90 € EUR. The 'Chronologie' section shows a timeline of the payment, including 'Paielement réussi' and 'Paielement démarré'. The 'Résumé Checkout' section shows the client information and a table of the items purchased.

ARTICLES	QTE	PRIX UNITAIRE	MONTANT
T-Shirt Moulant	1	20,00 €	20,00 €
Echarpe Rayée	1	20,00 €	20,00 €
Montre	1	500,00 €	500,00 €
La Poste	1	1,90 €	1,90 €

## PARTIE 5. VEILLE SÉCURITÉ

### 5.1. FAILLE

#### 5.1.1. RÉSUMÉ

Une vulnérabilité a été découverte dans Symfony. Elle permet à un attaquant de provoquer une injection de requêtes illégitimes par rebond (CSRF). Le composant de formulaire Symfony fournit un mécanisme de protection CSRF en utilisant un jeton aléatoire injecté dans le formulaire et en utilisant la session pour stocker et contrôler le jeton soumis par l'utilisateur. Lors de l'utilisation du FrameworkBundle, cette protection peut être activée ou désactivée avec la configuration. Si la configuration n'est pas précisée, par défaut, le mécanisme est activé tant que la session est activée. Dans un changement récent dans la façon dont la configuration est chargée, le comportement par défaut a été abandonné et, par conséquent, la protection CSRF n'est pas activée sous forme lorsqu'elle n'est pas explicitement activée, ce qui rend l'application sensible aux attaques CSRF.

#### 5.1.2. SOLUTION

Symfony a restauré la configuration par défaut pour activer la protection CSRF par défaut. (<https://github.com/symfony/symfony/commit/f0ffb775febdf07e57117aabadac96fa37857f50>)

#### 5.1.3. DOCUMENTATION

BULLETIN DE SECURITY SYMFONY DU 29 JANVIER 2022

<https://github.com/symfony/symfony/security/advisories/GHSA-vvmr-8829-6whx>

#### 5.1.4. RÉFÉRENCE CVE CVE-2022-23601

<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-23601>

## PARTIE 6. RECHERCHE CRÉATION EASYADMIN

Pour ce projet, j'ai du comprendre et faire des recherches sur la mise en place du module EasyAdmin

### 6.1. INSTALLATION

Pour installer EasyAdmin, il faut lancer la ligne de commande : `php bin/console make:admin:dashboard`

### 6.2. CONFIGURATION DU DASHBOARD

Dans mon projet j'ai appelé ce fichier `DashboardController.php`. Ci-dessous, c'est un exemple pour comprendre comment on doit configurer le dashboard :

```
<?php

namespace App\Controller\Admin;

use EasyCorp\Bundle\EasyAdminBundle\Config\Dashboard;
use EasyCorp\Bundle\EasyAdminBundle\Controller\AbstractDashboardController;

class DashboardController extends AbstractDashboardController
{
    // ...

    public function configureDashboard(): Dashboard
    {
        return Dashboard::new()
            // the name visible to end users
            ->setTitle('ACME Corp.')
            // you can include HTML contents too (e.g. to link to an image)
            ->setTitle(' ACME <span class="text-small">Corp.</span>')

            // by default EasyAdmin displays a black square as its default
            favicon;

            // use this method to display a custom favicon: the given path is
            passed

            // "as is" to the Twig asset() function:
            // <link rel="shortcut icon" href="{{ asset('...') }}">
```

```

->setFaviconPath('favicon.svg')

// the domain used by default is 'messages'
->setTranslationDomain('my-custom-domain')

// there's no need to define the "text direction" explicitly
because
// its default value is inferred dynamically from the user locale
->setTextDirection('ltr')

// set this option if you prefer the page content to span the
entire
// browser width, instead of the default design which sets a max
width
->renderContentMaximized()

// set this option if you prefer the sidebar (which contains the
main menu)
// to be displayed as a narrow column instead of the default
expanded design
->renderSidebarMinimized()

// by default, users can select between a "light" and "dark" mode
for the
// backend interface. Call this method if you prefer to disable
the "dark"
// mode for any reason (e.g. if your interface customizations are
not ready for it)
->disableDarkMode()

// by default, all backend URLs are generated as absolute URLs. If
you
// need to generate relative URLs instead, call this method
->generateRelativeUrls()
;
}
}

```

## 6.3. CONFIGURATION DU MENU

Dans le fichier `DashboardController.php`, pour configurer le menu, il faut rajouter la fonction `configureMenuItems()` :

```

<?php
// ...

```

```
public function configureMenuItems(): iterable
{
    return [
        MenuItem::linkToDashboard('Dashboard', 'fa fa-home'),

        MenuItem::section('Blog'),
        MenuItem::linkToCrud('Categories', 'fa fa-tags', Category::class),
        MenuItem::linkToCrud('Blog Posts', 'fa fa-file-text', BlogPost::
class),

        MenuItem::section('Users'),
        MenuItem::linkToCrud('Comments', 'fa fa-comment', Comment::class),
        MenuItem::linkToCrud('Users', 'fa fa-user', User::class),
    ];
}
```

## 6.4. AJOUT D'UN LIEN

Pour accéder au site du back office j'ai dû trouver quel code utiliser dans la fonction `configureMenuItems()`.

Cette méthode est :

```
<?php

return [ // linkToRoute permet de créer un lien pour retourner au site
    yield MenuItem::linkToRoute('Retour', 'fa fa-home', 'home')
];
```

## 6.5. PRISE

## PARTIE 7. TRADUCTION DU SITE

### 7.1. EXTRAIT DU SITE ANGLOPHONE

Dashboards are the entry point of backends and they link to one or more resources. Dashboards also display a main menu to navigate the resources and the information of the logged in user.

Imagine that you have a simple application with three Doctrine entities: users, blog posts and categories. Your own employees can create and edit any of them but external collaborators can only create blog posts. (789 signes)

You can implement this in EasyAdmin as follows: +

- . Create three CRUD controllers (e.g. `UserCrudController`, `BlogPostCrudController` and `CategoryCrudController`); +
- . Create a dashboard for your employees (e.g. `DashboardController`) and link to the three resources; +
- . Create a dashboard for your external collaborators (e.g. `ExternalDashboardController`) and link only to the `BlogPostCrudController` resource.

Technically, dashboards are regular Symfony controllers so you can do anything you usually do in a controller, such as injecting services and using shortcuts like `$this->render()` or `$this->isGranted()`.

Dashboard controller classes must implement the `EasyCorp\Bundle\EasyAdminBundle\Contracts\Controller\DashboardControllerInterface`, which ensures that certain methods are defined in the dashboard. Instead of implementing the interface, you can also extend from the `AbstractDashboardController` class. Run the following command to quickly generate a dashboard controller:

```
$ php bin/console make:admin:dashboard
```

If you now visit the `/admin` URL of your application, you'll see the default EasyAdmin Welcome Page:

## 7.2. TRADUCTION EN FRANÇAIS

Les tableaux de bord sont le point d'entrée des backends et ils sont liés à une ou plusieurs ressources. Les tableaux de bord affichent également un menu principal pour naviguer dans les ressources et les informations de l'utilisateur connecté.

Imaginez que vous ayez une application simple avec trois entités Doctrine : utilisateurs, articles de blog et catégories. Vos propres employés peuvent créer et modifier n'importe lequel d'entre eux, mais les collaborateurs externes ne peuvent créer que des articles de blog.

Vous pouvez l'implémenter dans EasyAdmin comme suit : +

- . Créez trois contrôleurs CRUD (par exemple, `UserCrudController`, `BlogPostCrudController` et `CategoryCrudController`) ; +
- . Créez un tableau de bord pour vos employés (par exemple `DashboardController`) et un lien vers les trois ressources ; +
- . Créez un tableau de bord pour vos collaborateurs externes (par exemple, `ExternalDashboardController`) et créez un lien uniquement vers la ressource `BlogPostCrudController`.

Techniquement, les tableaux de bord sont des contrôleurs Symfony standard, vous pouvez donc faire tout ce que vous faites habituellement dans un contrôleur, comme injecter des services et utiliser des raccourcis comme `$this->render()` ou `$this->isGranted()`.

Les classes de contrôleur de tableau de bord doivent implémenter `EasyCorp\Bundle\EasyAdminBundle\Contracts\Controller\DashboardControllerInterface`, qui garantit que certaines méthodes sont définies dans le tableau de bord. Au lieu d'implémenter l'interface, vous pouvez également étendre la classe `AbstractDashboardController`. Exécutez la commande suivante pour générer rapidement un contrôleur de tableau de bord :

```
$ php bin/console make:admin:dashboard
```

Si vous visitez maintenant l'URL `/admin` de votre application, vous verrez la page d'accueil EasyAdmin par défaut :



## PARTIE 8. ANNEXE

### 8.1. SOURCE DU DOSSIER PROJET EN ASCIIDOCTOR

Code source dossier projet (fichier pdf) créer en asciidoctor, integrant plantuml pour generer les schemas.

```
:doctype: book
:chapter-signifier: Partie
:doctitle: Dossier Projet: Développeur web et web Mobile
:docdate: 06/09/2022
:docupdate: 07/09/2022
:imagesdir: images
:icons: font
:toc: auto
:toclevels: 5
:toc-title: Table des matières
:pdf-style: Dossier-Projet
:pdf-stylesdir: {docdir}
:pdf-themesdir: {docdir}/themes
:pdf-fontsdir: {docdir}/fonts
:source-highlighter: pygments
= {doctitle}
William David
v1.0, 06/09/2022

<<<
// asciidoctor-pdf -r asciidoctor-diagram --theme=Dossier-Projet-theme.yml
'.\dossier projet - développeur web et web mobile.adoc'

:sectnums:
== Présentation du projet

=== Résumé du projet

Le projet a été réalisé sur la période du 1er août au 31 août, pour mon propre compte.

Il met en œuvre un site de commerce électronique de vente de vêtements, avec une vitrine, un catalogue de produits, un moteur de recherche, un système de notation, ainsi qu'un système paiement en ligne par carte bleue.

Le Framework web utilisé est le Framework PHP Symfony (https://symfony.com/), accompagné du module EasyAdmin (https://symfony.com/bundles/EasyAdminBundle/current/index.html) pour la
```

gestion du Back-Office. +

Le Responsive Design est mise en œuvre grâce à l'intégration du Framework CSS Bootstrap (<https://getbootstrap.com/>). +

Les notifications par e-mail se grâce au service d'envoi d'e-mail Mailjet (<https://www.mailjet.com/>). +

Le paiement en ligne est assuré par le service Stripe (<https://stripe.com/en-fr>).

### === Liste des compétences du référentiel

Chacune des compétences citées ci-dessous, relatives au protocole de la rédaction d'un projet, sont accompagnées de précisions sur les technologies abordées avant d'expliquer leur utilisation dans les parties suivantes.

#### ==== Maquetter une application

La modélisation du projet a été réalisé avec l'outil en ligne <https://www.draw.io/>.

#### ==== Concevoir une base de données

La conception de la base de données a été réalisé en suivant le modèle Entité-Relation de la méthode Merise.

#### ==== Mettre en place une base de données

La création de la base de données relationnelle a été faite avec l'outil d'administration phpMyAdmin.

La matérialisation des tables de données a été faite au travers de l'outil de gestion de version des structures des tables intégré à l'ORM Doctrine, qui génère des scripts de migration.

#### ==== Développer une interface utilisateur

L'interface utilisateur est été réalisée avec le moteur de Template Twig, intégré au Framework Symfony, et le Framework CSS Bootstrap.

#### ==== Développer des composants d'accès de données

Les composants d'accès aux données ont été réalisés en s'appuyant sur l'ORM Doctrine, intégré au Framework Symfony. La gestion des données au niveau du Back-Office a été intégrée au module EasyAdmin du Framework Symfony.

#### ==== Développer des pages web en lien avec une base de données

Les pages web ont été développé dans le cadre du Framework Symfony, grâce à

L'association de l'ORM Doctrine et du moteur de template Twig qui permettent de composer rapidement des pages affichant les données provenant de la base de données.

#### ==== Mettre en œuvre une solution de e-commerce

La mise en œuvre du projet, s'appuyant sur une base de données MySQL, le Framework PHP Symfony, intégrant le moteur de template Twig, auquel a été ajouté le module de gestion EasyAdmin, forme le squelette d'une solution de e-commerce.

Le paiement en ligne a été mise en œuvre en s'appuyant sur le service Stripe.

Une base de test avec des articles, des descriptions, des visuels libres de droit a été créé pour pouvoir présenter la solution.

#### ==== Développer une application simple de mobilité numérique

La partie Front-Office du site a été entièrement réalisée avec le Framework CSS Bootstrap. Celui-ci depuis longtemps offre les fonctionnalités nécessaire au Responsive Design.

#### ==== Utiliser l'anglais dans son activité professionnelle en informatique

L'utilisation de l'anglais est la norme lors de la conception de programme informatique. Les Frameworks et langages de programmation utilisés ont pour base l'anglais et l'essentiel des tutoriels et documents techniques qui ont été utilisés pour la réalisation de ce projet sont en anglais.

#### ==== Actualiser et partager ses compétences en développement informatique

La réalisation de ce projet a été pour moi l'occasion d'approfondir ce que j'avais vu en cours et de le transposer dans un autre langage.

J'ai pu pour cela m'appuyer sur l'expérience d'un collaborateur freelance qui m'a guidé et conseiller lors de la réalisation de ce projet. +

Nous avons échangé par messagerie privée Signal, géré le projet grâce à Trello, suivi les versions et les problèmes sur Github, et effectué du Pair-Programming grâce à GitLive ou en présentiel.

#### == Cahier des charges

#### === Présentation de l'entreprise

La société William David (immatriculation en cours) est une auto-entreprise qui fournit des services de conception et de réalisation de sites web.

### === Résumé du projet

Ce projet met en œuvre un site de commerce électronique de vente de vêtements. Il met en place :

- \* une vitrine
- \* un catalogue de produits
- \* un moteur de recherche
- \* un système de notation
- \* un système de prise de commande avec panier
- \* un système de paiement en ligne par carte bleue
- \* un système d'authentification et d'autorisation
- \* un système de gestion des données applicatives (clients, articles, commandes, factures, avis)
- \* un système de gestion des droits (utilisateurs, administrateurs)

### === Objectif du site

Le site de vente en ligne de vêtements doit permettre d'acquérir un ou plusieurs articles, de les sélectionner, de les payer, et ce, de la façon la plus fluide et sécurisée.

Pour ce faire, un accent a été mis pour améliorer l'UX des différents scénarios développer pour l'utilisation de ce site web, de la mise en avant et de la recherche de produit, par catégorie jusqu'au paiement en ligne. Le service après vente est assuré par la présence d'un formulaire de contact.

Cependant, pour des questions évidentes de sécurité, tout achat doit se faire à partir d'un compte existant, à créer le cas échéant.

### === Spécification fonctionnelle

#### ==== Front-Office

L'interface présentée au client doit permettre :

- \* de présenter le catalogue de produits
- \* de naviguer de façon fluide dans le catalogue
- \* de rechercher des produits par différents critères
- \* d'accéder aux meilleures ventes du moment
- \* d'ajouter des produits à son panier
- \* de consulter son panier
- \* de supprimer un article de son panier
- \* de valider son panier après vérification des informations (articles, prix, adresse, livraison...)
- \* d'accéder au paiement en ligne
- \* de s'inscrire sur le site

```

* de se connecter avec un compte existant
* de gérer son compte (adresse, mot de passe, supprimer son compte...)
* de contacter le vendeur par le biais d'un formulaire

==== Back-Office

L'interface présentée à l'administrateur doit permettre :

* de se connecter avec un compte d'administration
* de gérer les comptes utilisateur et leurs droits d'accès
* de gérer les données applicatives (clients, articles, commandes, factures,
avis)
* téléverser de nouveaux documents (photos d'article, fiches technique etc...)
* de consulter les statistiques de fréquentation du site en nombre de
visiteurs
* de consulter les messages envoyés par le biais du formulaire de contact

<<<

=== Modélisation

==== Cas d'usage

===== Front-Office

[plantuml, target=use-case-front, format=png, align=center]

```

left to right direction skinparam actorStyle awesome

actor "Client" as client

```

package "Front-Office" { "Se connecter" as (sign-in) "Gérer son compte" as
(account) "Modifier son mot de passe" as (password) "Modifier son
adresse" as (address) "Supprimer son compte" as (delete) "S'inscrire sur le
site" as (sign-up) "Consulter le catalogue" as (catalog) "Rechercher un
article" as (research) "Ajouter un article au panier" as (add) "Retirer un
article du panier" as (remove) "Consulter le panier" as (view) "Valider son
panier" as (validate) "Contacter le vendeur" as (contact)

}

```

```

package Stripe { "Accéder au paiement en ligne" as (stripe) }

```

```

client --> (sign-in) client --> (sign-up) client --> (account) (account) --|>

```

```
(password) (account) --|> (address) (account) --|> (delete) client --→
(catalog) client --→ (research) (catalog) --|> (add) (research) --|> (add) client
--→ (view) (view) --|> (remove) (view) --|> (validate) client --→ (contact)
(validate) --|> (stripe)
```

&lt;&lt;&lt;

===== Back-Office

[plantuml, target=use-case-back, format=png, align=center]

left to right direction skinparam actorStyle awesome

actor "Administrateur" as admin

```
package "Front-Office" {
    "Se connecter" as (sign-in)
    "Gérer les comptes utilisateur" as (user)
    "Créer un utilisateur" as (user-add)
    "Modifier un compte utilisateur" as (user-update)
    "Supprimer un compte utilisateur" as (user-delete)
    "Gérer les comptes client" as (customer)
    "Modifier un compte client" as (customer-update)
    "Supprimer un compte client" as (customer-delete)
    "Gérer les articles" as (item)
    "Créer un article" as (item-add)
    "Modifier un article" as (item-update)
    "Supprimer un article" as (item-delete)
    "Gérer les commandes" as (order)
    "Modifier une commande" as (order-update)
    "Supprimer une commande" as (order-delete)
    "Gérer les avis" as (review)
    "Modérer une avis" as (review-delete)
    "Téléverser un document" as (upload)
    "Consulter les statistiques" as (analytics)
    "Consulter les messages" as (read)
}
```

```
admin --→ (sign-in)
admin --→ (user)
(user) --|> (user-add)
(user) --|> (user-update)
(user) --|> (user-delete)
admin --→ (customer)
(customer) --|> (customer-update)
(customer) --|> (customer-delete)
admin --→ (item)
(item) --|> (item-add)
(item) --|> (item-update)
(item) --|> (item-delete)
admin --→ (order)
(order) --|> (order-update)
(order) --|> (order-delete)
admin --→ (review)
(review) --|> (review-delete)
admin --→ (upload)
admin --→ (analytics)
admin --→ (read)
```

&lt;&lt;&lt;

===== Séquences d'actions

### ===== Authentification

[plantuml, target=authentication-sequence, format=png, align=center]

skinparam actorStyle awesome

actor utilisateur collections login control authentication database mysql  
queue log

utilisateur → login : se connecter  
login → authentication : vérifier  
authentication → mysql : demander\nle haché\ndu mot de passe\nde l'utilisateur\net le sel  
mysql → authentication : renvoyer\nle haché\ndu mot de passe\net le sel  
authentication → authentication : comparer\nles hachés  
authentication -[#green]> login : OK - ouvrir\nune session\nHTTP  
authentication -[#green]> log : OK - enregistrer\nla connexion  
login -[#green]> utilisateur : OK - authentifier\nl'utilisateur  
authentication -[#red]> login : FAIL - renvoyer\nune erreur\nutilisateur\nnon valide  
authentication -[#red]> log : FAIL - enregistrer\nl'échec de\nconnexion  
login -[#red]> utilisateur : FAIL - afficher l'échec\nd'authentification

<<<

### ===== Inscription

[plantuml, target=inscription-sequence, format=png, align=center]

skinparam actorStyle awesome

actor utilisateur collections inscription control authentication database mysql  
queue log

utilisateur → inscription : s'inscrire  
inscription → authentication : demander\nune inscription  
authentication → authentication : saler et hacher\nle mot de passe  
authentication → mysql : enregistrer\nles identifiants  
authentication → log : enregistrer\nune inscription  
authentication → inscription : valider\nl'inscription  
inscription → utilisateur : notifier\nde la réussite\nde l'inscription

&lt;&lt;&lt;

===== Achat

**[plantuml, target=order-sequence, format=png, align=center]**

skinparam actorStyle awesome

actor utilisateur collections catalogue collections article collections panier  
 control validation database mysql entity stripe queue livraison queue log

utilisateur → catalogue : consulter catalogue → utilisateur : afficher\nles  
 articles utilisateur → article : choisir un article article → panier :  
 ajouter\nun article panier → mysql : enregistrer l'article dans le panier  
 mysql → panier : confirmer l'enregistrement panier → utilisateur : notifier  
 de la mise à jour du panier utilisateur → panier : valider le panier panier →  
 validation : initier\nune validation validation → mysql : enregistrer\nla  
 commande validation → stripe : initier un paiement stripe → validation :  
 renvoyer\nun numéro de règlement validation → mysql : enregistrer\nle  
 numéro\nde règlement validation → log : enregistrer l'initiation d'un  
 règlement validation → utilisateur : afficher le formulaire Stripe utilisateur  
 → stripe : envoyer ses informations de paiement stripe -[#green]>  
 validation : OK - confirmer le paiement validation -[#green]> mysql : OK -  
 enregistrer\nle paiement validation -[#green]> log : OK - enregistrer le  
 paiement validation -[#green]> mysql : OK - enregistrer\nun  
 préparation\nde commande validation -[#green]> livraison : OK - mettre  
 en livraison validation -[#green]> utilisateur : OK - confirmer la prise en  
 charge de la commande stripe -[#red]> validation : FAIL - notifier de  
 l'échec du paiement validation -[#red]> mysql : FAIL -  
 enregistrer\nl'échec\nle paiement validation -[#red]> log : FAIL -  
 enregistrer l'échec le paiement validation -[#red]> utilisateur : FAIL -  
 notifier de l'échec du paiement

===== Avis

**[plantuml, target=review-sequence, format=png, align=center]**

skinparam actorStyle awesome



actor utilisateur actor administrateur collections catalogue collections  
 article collections avis database mysql queue log

utilisateur → catalogue : consulter catalogue → utilisateur : afficher\nles  
 articles utilisateur → article : choisir un article article → avis : rédiger\nun  
 avis avis → mysql : enregistrer\nun avis avis → log : enregistrer\nune  
 reception\nd'avis avis → utilisateur : notifier la prise en charge de l'avis log  
 → administrateur : notifier l'administrateur de la réception d'un avis  
 administrateur -[#green]> mysql : OK - marquer l'avis pour publication  
 mysql -[#green]> avis : OK - renvoyer\nles avis \nà afficher administrateur  
 -[#red]> mysql : FAIL - modérer l'avis

<<<

==== Entité-Relation

[plantuml, target=database-map, format=png, align=center]

entity address { id user\_id name firstname lastname company address  
 postal city country phone }

entity carrier { id name description price }

entity category { id name }

entity order { id user\_id created\_at carrier\_name carrier\_price delivery  
 reference stripe\_session state }

entity order\_details { id binded\_order\_id product quantity price total }

entity product { id category\_id name slug image subtitle description price  
 is\_in\_home }

entity user { id email roles password firstname lastname }

entity alert { id product\_id name }

entity avis { id auteur contenu created\_at product }

address "1..1" --> "0..n" user alert "1..1" --> "0..n" product order "1..1" -->  
 "0..n" user order "1..1" --> "0..n" carrier order\_details "1..1" --> "0..n" order

avis "1..1" -> "0..n" user avis "1..1" -> "0..n" product category "0..n" ←-  
 "1..1" product

```
<<<
```

```
==== Plan du site
```

```
[plantuml, target=site-map, format=png, align=center,height = 80%]
```

left to right direction

component [/] as home component [/a-propos] as a\_propos component  
 [/contact] as contact component [/inscription] as inscription component  
 [/connexion] as connexion component [/logout] as logout component  
 [/admin] as admin

package "Avis" { component [/avis] as avis component [/avis/detail/{id}] as  
 avis0 component [/avis/edit/{id}] as avis1 component [/avis/del/{id}] as  
 avis2

```
avis --> avis0
avis --> avis1
avis --> avis2
}
```

package "Compte" { component [/compte] as compte component  
 [/compte/mot-de-passe] as compte0 component [/compte/commandes]  
 as compte1 component [/compte/commandes/{reference}] as compte2  
 component [/compte/adresses] as compte3 component  
 [/compte/adresses/ajouter] as compte4 component  
 [/compte/adresses/modifier/{id}] as compte5 component  
 [/compte/adresses/supprimer/{id}] as compte6

```
compte --> compte0
compte --> compte1
compte --> compte2
compte --> compte3
compte --> compte4
compte --> compte5
compte --> compte6
```

```
}
```

```
package "Articles" { component [/articles] as articles component
[/articles/{id}-{slug}] as articles0
```

```
    articles --> articles0
}
```

```
package "Panier" { component [/mon-panier] as panier component
[/panier/ajouter/{id}] as panier0 component [/panier/réduire/{id}] as
panier1 component [/panier/supprimer/{id}] as panier2 component
[/panier/supprimer] as panier3
```

```
    panier --> panier0
    panier --> panier1
    panier --> panier2
    panier --> panier3
}
```

```
package "Commande" { component [/commande] as commande
component [/commande/recap] as commande0 component
[/commande/checkout/{reference}] as commande1 component
[/commande/valide/{stripeSession}] as commande2 component
[/commande/echec/{stripeSession}] as commande3
```

```
    commande --> commande0
    commande --> commande1
    commande --> commande2
    commande --> commande3
}
```

home -u→ a\_propos home -u→ contact home -u→ inscription home -u→ connexion home -u→ logout home -u→ admin

home -d→ avis home -d→ compte home -d→ articles home -d→ panier  
home -d→ commande

```
==== Maquettage
===== Accueil
```

```
image:1_maquette_page_accueil.png[align=center]
```

```
===== Produit
```

```
image:2_maquette_page_produit.png[align=center]
```

```
<<<
```

```
===== Panier
```

```
image:3_maquette_page_panier.png[align=center]
```

```
<<<
```

```
=== Spécification technique
```

Le Framework web utilisé est le Framework PHP Symfony (<https://symfony.com/>), accompagné du module EasyAdmin (<https://symfony.com/bundles/EasyAdminBundle/current/index.html>) pour la gestion du Back-Office. +

Le Responsive Design est mise en œuvre grâce à l'intégration du Framework CSS Bootstrap (<https://getbootstrap.com/>). +

Les notifications par e-mail se grâce au service d'envoi d'e-mail Mailjet (<https://www.mailjet.com/>). +

Le paiement en ligne est assuré par le service Stripe (<https://stripe.com/en-fr>).

Utilisateurs :

Le visiteur anonyme est autorisé à consulter la partie vitrine du site.

L'administrateur peut se connecter au back office afin de modifier le contenu du site.

J'ai intégré Symfony au projet, et créé les entités Doctrine responsables de la lecture et de l'écriture des données dans la base. Ceci a été réalisé grâce à l'utilitaire en ligne de commande de Symfony.

Une fois les entités créées, il m'a suffit de générer et d'exécuter une migration afin de modifier la structure de la base de données en accord avec le modèle précédemment établi. Doctrine est un ORM (Object Relational Mapper) permettant d'effectuer la lecture et l'écriture des données dans une base de données.

Pour cela, il repose sur son composant DBAL, permettant de faire l'interface avec la base. DBAL permet malgré tout d'utiliser des requêtes SQL traditionnelles pour interagir avec la base de données, mais propose également

un système de query builder, moyen alternatif de générer des requêtes SQL offrant, entre autres, une protection contre les injections.

Les divers Repository sont des objets fournis par Doctrine pour chaque entité, permettant de récupérer les données liées à chacune d'entre elles dans la base. Ils permettent également de définir des méthodes pour exécuter des requêtes SQL personnalisées.

#### ==== Symfony

Pour développeur ce site, j'ai choisi Symfony est l'un des Framework PHP, pour les raisons suivantes :

#### ===== Grande flexibilité

Symfony est l'un des Frameworks PHP les plus riches en fonctionnalités. Les deux avantages technologiques les plus remarquables de Symfony sont les bundles et les composants.

Le bundle est presque la même chose qu'un plugin. Considérez-le comme un ensemble de fichiers (fichiers PHP, feuilles de style, JavaScripts, images) pour la mise en œuvre d'une fonctionnalité (par exemple, un blog, un panier d'achat, etc.). Le principal avantage des bundles est qu'ils sont découplés. Vous pouvez les reconfigurer et les réutiliser pour de nombreuses applications afin de réduire le coût global de développement.

Les composants sont des fonctionnalités génériques qui réduisent les tâches de routine et permettent aux développeurs de se concentrer sur des fonctionnalités métier spécifiques. Il existe 30 composants Symfony utiles qui facilitent le processus de développement. Vous pouvez utiliser les composants de manière indépendante et ajouter vos propres modules personnalisés sans que l'architecture en pâtisse. Les composants Symfony peuvent également être utilisés de manière autonome dans d'autres frameworks (par exemple, Laravel) ou dans des solutions PHP simples.

Les bundles et les composants permettent d'éliminer les dépendances strictes dans l'architecture. Moins vous avez de dépendances, plus il sera facile d'apporter des changements sans risquer de casser d'autres parties du système. Ainsi, vous pouvez adapter la solution à toutes les exigences et à tous les scénarios d'utilisateur pour créer une application hautement flexible.

#### ===== Personnalisation

Symfony offre de grandes caractéristiques et fonctionnalités de personnalisation pour les développeurs et les entreprises.

#### ===== L'entreprise derrière la technologie

Symfony est l'un des rares frameworks bénéficiant d'un support commercial. Sensiolabs, l'entreprise-créateur et sponsor, contribue activement à sa réputation. Ils fournissent des tutoriels officiels et des certifications. Sur le site Web de l'entreprise, vous trouverez un calendrier des conférences à venir dans le monde entier. Cela montre l'ampleur et le sérieux de leurs intentions et de leurs convictions.

#### ===== Une fiabilité éprouvée

Symfony a prouvé sa fiabilité au fil du temps alors que de nombreux autres frameworks ont échoué.

#### ===== Facile à utiliser

Il existe une documentation complète et détaillée. Elle est considérée comme l'une des meilleures documentations parmi les autres frameworks PHP. Chaque composant est bien expliqué et simplifié par des exemples. De plus, il bénéficie également d'un grand soutien de la communauté. Il offre une configuration facile et un mécanisme de mise en cache pour améliorer les performances des applications.

#### ===== Support à long terme

Symfony est un framework stable et bien testé avec des mises à jour régulières. Les versions les plus récentes bénéficient d'un support à long terme et sont compatibles avec les versions plus récentes : jusqu'à 3 ans pour certaines versions.

#### ===== Grande communauté

Symfony est un open-source, avec une grande communauté. Cela signifie que les experts et les amateurs de PHP du monde entier participent à l'amélioration du code pour tout le monde. Dans la communauté, les gens coopèrent les uns avec les autres. Ils créent de nouveaux composants, essaient de résoudre les problèmes apparus, ou aident les autres avec des conseils.

#### ===== Une bonne documentation

Une documentation incomplète ou obsolète est un problème pour de nombreuses technologies. La documentation de Symfony est considérée comme l'une des meilleures, comparée à la documentation des autres frameworks PHP. Elle est clairement écrite, bien structurée, fournie avec des exemples, et mise à jour de version en version. Vous pouvez trouver une explication de chaque composant et du processus de développement dans son ensemble.

#### ===== Extensible

Tout dans le framework Symfony se représente comme un bundle. Chaque bundle a une fonctionnalité unique. Vous pouvez réutiliser le bundle dans d'autres projets et le partager avec la communauté également. C'est également l'une des raisons qui le rendent populaire auprès des développeurs. La meilleure partie est que vous pouvez changer ou modifier n'importe quoi, même le noyau du système sans reconfigurer le framework complet. Vous pouvez ajouter les fonctionnalités dont vous avez besoin et étendre les caractéristiques d'une application autant que vous le souhaitez.

#### ==== Les Contenus

Tout le contenu de ce projet (image, photos, logo, textes) sont libres de droit et d'utilisation.

#### ==== Inventaire technique

- \* Visual Studio Code
- \* Framework PHP Symfony
- \* Gestion du Back-Office avec EasyAdmin
- \* Moteur de Template Twig
- \* Accès à la base de donnée par l'ORM Doctrine
- \* Base de données MySQL
- \* Gestion de la base de données avec phpMyAdmin et Doctrine
- \* Serveur HTTP Apache
- \* Gestion du paiement avec Stripe
- \* Gestion d'envoi de mail avec Mailjet
- \* Gestion du Responsive Design avec le Framework CSS Bootstrap

#### == Réalisation

#### === Mise en place de Easy Admin

En Utilisant le site de symfony (<https://symfony.com/bundles/EasyAdminBundle/current/dashboards.html>), j'ai pu comprendre le fonctionnement de EasyAdmin et l'implémenter dans mon projet.

#### ==== Fichier de configuration DashboardController.php

[source, php]

```
<?php
```

```
namespace App\Controller\Admin;
```

```
use App\Entity\Avis; use App\Entity\User; use App\Entity\Order; use
```

```
App\Entity\Carrier; use App\Entity\Headers; use App\Entity\Product; use
App\Entity\Category;
Symfony\Component\HttpFoundation\Response;
Symfony\Component\Routing\Annotation\Route;
EasyCorp\Bundle\EasyAdminBundle\Config\MenuItem;
EasyCorp\Bundle\EasyAdminBundle\Config\Dashboard;
EasyCorp\Bundle\EasyAdminBundle Router\AdminUrlGenerator;
EasyCorp\Bundle\EasyAdminBundle\Controller\AbstractDashboardContr
oller;
```

```
class DashboardController extends AbstractDashboardController { /** *
@Route("/admin", name="admin") */ public function index(): Response { //
redirect to some CRUD controller $routeBuilder =
$this->get(AdminUrlGenerator::class);
```

```
    return $this->redirect($routeBuilder-
    >setController(OrderCrudController::class)->generateUrl());
}
```

```
public function configureDashboard(): Dashboard
{
    return Dashboard::new()
        ->setTitle('Ma Boutique'); // Titre du Back Office
```

```
}
```

```
public function configureMenuItems(): iterable
{
    // linkToDashboard permet de créer le home du menu
    yield MenuItem::linkToDashboard('Tableau de bord', 'fa fa-home');
    // linkToCrud permet de créer les menus en les reliant a une table
    yield MenuItem::linkToCrud('Utilisateurs', 'fas fa-user',
User::class);
    yield MenuItem::linkToCrud('Catégories', 'fas fa-list',
Category::class);
    yield MenuItem::linkToCrud('Produits', 'fas fa-tag', Product::class);
    yield MenuItem::linkToCrud('Transporteurs', 'fas fa-truck',
Carrier::class);
    yield MenuItem::linkToCrud('Commandes', 'fas fa-shopping-cart',
```



```

Order::class);
    yield MenuItem::linkToCrud('Avis', 'fas fa-desktop', Avis::class);
    yield MenuItem::linkToCrud('Bannières', 'fas fa-desktop',
Headers::class);
    return [ // linkToRoute permet de créer un lien pour retourner au site
        yield MenuItem::linkToRoute('Retour', 'fa fa-home', 'home')
    ];
}
}

```

Le fichier ci-dessus, est `DashboardController.php`, il permet de créer le menu du back office sur le côté gauche.

==== Exemple de fichier de configuration crud pour la table Avis

Les commandes utilisées pour créer les entités et les scripts de migration sont :

[source,shell]

```

php bin/console make:entity {nom de la table} php bin/console
make:migration php bin/console doctrine:migrations:migrate

```

[source, php]

```
<?php
```

```
namespace App\Controller\Admin;
```

```

use App\Entity\Avis;
use EasyCorp\Bundle\EasyAdminBundle\Config\Crud;
use EasyCorp\Bundle\EasyAdminBundle\Config\Actions;
use EasyCorp\Bundle\EasyAdminBundle\Field\SlugField;
use EasyCorp\Bundle\EasyAdminBundle\Field\TextField;
use EasyCorp\Bundle\EasyAdminBundle\Field\ImageField;
use EasyCorp\Bundle\EasyAdminBundle\Field\MoneyField;
use EasyCorp\Bundle\EasyAdminBundle\Field\BooleanField;
use EasyCorp\Bundle\EasyAdminBundle\Field\DateTimeField;
use EasyCorp\Bundle\EasyAdminBundle\Field\TextareaField;
use EasyCorp\Bundle\EasyAdminBundle\Field\AssociationField;

```

EasyCorp\Bundle\EasyAdminBundle\Controller\AbstractCrudController;

```
class AvisCrudController extends AbstractCrudController { public static
function getEntityFqcn(): string { return Avis::class; }
```

```
public function configureFields(string $pageName): iterable
{
    return [
        TextField::new('auteur','Auteur'), // Relis le champs auteur à une
        colonne Auteur dans le tableau
        TextareaField::new('contenu')->hideOnIndex(),
        DateTimeField::new('created_at', 'Créée le')
    ];
}
```

```
public function configureCrud(Crud $crud): Crud
{
    return $crud
        ->setEntityLabelInSingular('Avis')
        ->setEntityLabelInPlural('Avis')
        ;
}
```

```
}
```

Le fichier AvisCrudController.php permet de configurer le CRUD pour la table avis.

=== Mise en place du module stripe  
 ==== Pourquoi Stripe

Stripe est un outil efficace de paiement en ligne qui permet de transférer de l'argent du compte bancaire de votre client vers le compte de votre entreprise, par le biais de carte de crédit.

Stripe est un module ergonomique qui s'harmonise parfaitement au style de votre site. De plus, il est facile d'utilisation par votre client. Cet atout vous permet d'augmenter la satisfaction de vos clients.

Stripe assure un niveau de sécurité élevé, vous permettant de recevoir vos paiements en toute fiabilité. C'est aussi une solution avantageuse pour vos clients, car tous les frais sont contrôlés par son site marchand, afin d'éviter tout acte malveillant.

Pour se prémunir contre les litiges avec les clients, Stripe vous offre un contrat VAD (Vente à Distance) que vous souscrivez lors de l'achat d'un

**abonnement Stripe.**

La fiabilité et la sécurité optimale de Stripe en font la solution la plus prisée par plusieurs e-commerçants qui utilisent des CMS très populaires à l'instar de Prestashop et Shopify.

==== Controller PaymentController.php

[source, php]

```
<?php
```

```
namespace App\Controller;
```

```
use App\Entity\Order; use App\Model\Cart; use
App\Repository\OrderRepository; use App\Service\Mail; use
Doctrine\ORM\EntityManagerInterface; use Stripe\Checkout\Session; use
Stripe\Stripe; use
Symfony\Bundle\FrameworkBundle\Controller\AbstractController; use
Symfony\Component\HttpFoundation\Response; use
Symfony\Component\Routing\Annotation\Route;
```

```
class PaymentController extends AbstractController { / * Etape de
vérification avant confirmation du paiement */ / *
@Route("/commande/checkout/{reference}", name="checkout") */ public
function payment(OrderRepository $repository, $reference,
EntityManagerInterface $em): Response { // Récupération des produits de
la dernière commande et formattage dans un tableau pour Stripe $order =
$repository->findOneByReference($reference); if (!$order) { throw
$this->createNotFoundException('Cette commande n'existe pas'); }
$products = $order->getOrderDetails()->getValues(); $productsForStripe
= []; foreach ($products as $item) { $productsForStripe[] = [ 'price_data' => [
'currency' => 'eur', 'unit_amount' => $item->getPrice(), 'product_data' => [
'name' => $item->getProduct() ] ], 'quantity' => $item->getQuantity() ]; } //
Ajout des frais de livraison $productsForStripe[] = [ 'price_data' => [
'currency' => 'eur', 'unit_amount' => $order->getCarrierPrice(), 'product_data'
=> [ 'name' => $order->getCarrierName() ] ], 'quantity' => 1 ];
Stripe::setApiKey('sk_test_51LNyQsDz6qMOcyaOBiHWM8Y6a3k7rAGO4
OC2L3qxGbl9f5XhsxzUAeqgrhhKYEmSMEHAgZ3ul33kjfR96pZN0lpb00
NMCO7VJA'); header('Content-Type: application/json');
```

```
// $YOUR_DOMAIN = 'https://ecommerce.fr';
$YOUR_DOMAIN = 'http://localhost:8080';
```

```
// Création de la session Stripe avec les données du panier
$checkout_session = Session::create([
    'line_items' => $productsForStripe,
    'mode' => 'payment',
    'success_url' => $YOUR_DOMAIN .
'/commande/valide/{CHECKOUT_SESSION_ID}',
    'cancel_url' => $YOUR_DOMAIN .
'/commande/echec/{CHECKOUT_SESSION_ID}',
]);
$order->setStripeSession($checkout_session->id);
$em->flush();
return $this->redirect($checkout_session->url);
}
```

```
/**
 * Méthode appelée lorsque le paiement est validé
 */
/**
 * @Route("/commande/valide/{stripeSession}", name="payment_success")
 */
public function paymentSuccess(OrderRepository $repository, $stripeSession,
EntityManagerInterface $em, Cart $cart)
{
    $order = $repository->findOneByStripeSession($stripeSession);
    if (!$order || $order->getUser() != $this->getUser()) {
        throw $this->createNotFoundException('Commande inaccessible');
    }
    if (!$order->getState()) {
        $order->setState(1);
        $em->flush();
    }
}
```

```
// Envoi mail de Confirmation
$user = $this->getUser();
```

```
$content = "Bonjour {$user->getFirstname()} nous vous remercions de votre
commande";
(new Mail)->send(
```

```

$user->getEmail(),
$user->getFirstname(),
"Confirmation de la commande {"$order->getReference()}",
$content
);

```

```

// Suppression du panier une fois la commande validée
$cart->remove();
return $this->render('payment/success.html.twig', [
    'order' => $order
]);
}

```

```

/**
 * Commande annulée (clic sur retour dans la fenêtre)
 */
/**
 * @Route("/commande/echec/{stripeSession}", name="payment_fail")
 */
public function paymentFail(OrderRepository $repository, $stripeSession)
{
    $order = $repository->findOneByStripeSession($stripeSession);
    if (!$order || $order->getUser() != $this->getUser()) {
        throw $this->createNotFoundException('Commande inaccessible');
    }
}

```

```

return $this->render('payment/fail.html.twig', [
    'order' => $order
]);
}
}

```

Ce Controller permet de mettre en place le module stripe et de lui fournir les données du panier, les informations produits et informations livraisons.

Il gère aussi le cas si le paiement réussi (méthode paymentSuccess) ou échoue (méthode paymentFail).

En déléguant le paiement à Stripe, la sécurité est gérée totalement par ce module qui est stable et connaît son domaine.

<<<

**==== Template de succès de paiement****[source, php]**

```
{% extends 'base.html.twig' %}
```

```
{% block title %}Ma commande - Ma Boutique{% endblock %}
```

```
{% block body %} <h2>Confirmation de votre commande</h2> <p>
Bonjour {{order.user.firstname}} {{order.user.lastname}}, <br> Nous vous
remercions de votre commande n° <b>{{order.reference}}</b>.<br> Une
confirmation vient de vous être envoyé par mail. <br> </p> <hr> <p>
Votre commande sera livrée par {{order.carrierName}} à l'adresse suivante:
<br> {{order.delivery|raw}} </p> <hr> <p> Pour suivre votre commande,
rendez-vous dans votre <a href="{{ path('account_orders') }}">compte</a>.
</p> {% endblock %}
```

**==== Template d'erreur de paiement****[source, php]**

```
{% extends 'base.html.twig' %}
```

```
{% block title %}Ma commande - Ma Boutique{% endblock %}
```

```
{% block body %} <h2>Annulation de votre commande</h2> <p> Bonjour
{{order.user.firstname}} {{order.user.lastname}}, <br> Votre paiement pour
la commande n° <b>{{order.reference}}</b> n'a pas abouti.<br> </p>
<hr> <a class="btn btn-outline-success" href="{{ path('order') }}">Réessayer</a> {% endblock %}
```

**=== Mise en place de Mailjet****==== Présentation**

Mailjet est un système d'envoi et de suivi d'emails basé dans le cloud5. La plateforme permet aux professionnels d'envoyer tant leurs emails marketing (newsletters, offres promotionnelles) que leurs emails transactionnels (notifications, confirmations d'inscription, de commande, factures...). Les services de Mailjet comprennent des solutions de conception d'emails, d'envoi de volumes massifs et de suivi de ces envois.

#### ==== Pourquoi Mailjet

J'ai choisit Mailjet car il donne la possibilité d'envoyer des mails et leur suivis.

Ensuite Mailjet me permettra de faire des newsletters, offres promotionnelles et autre.

Mailjet permet de réaliser des rapports de campagnes plutôt précis. Pour chaque campagne d'email marketing que vous menez, vous pouvez connaître en temps réel le pourcentage d'emails délivrés, de clics, d'emails ouverts, de désabonnement, de signalement comme spam ou encore de messages d'erreur.

==== Mail.php  
[source, php]

```
<?php namespace App\Service;
```

```
use Mailjet\Client; use Mailjet\Resources;
```

```
class Mail { private $api_key = "c2a1cd58ae21be0451736ab830498846";  
private $api_key_secret = "a8b58a736745704c0d59ee211014fe5c";
```

```
public function send($toEmail, $toName, $subject, $content)  
{  
    $mj = new Client($this->api_key, $this->api_key_secret,true,['version' =>  
'v3.1']);  
    $body =  
    [  
        'Messages' =>  
        [  
            [  
                'From' =>  
                [  
                    'Email' => "sportif_wd@hotmail.fr",  
                    'Name' => "Ma Boutique"  
                ],  
                'To' =>  
                [  
                    [  
                        'Email' => $toEmail,  
                        'Name' => $toName  
                    ]  
                ],  
                'Subject' => $subject,  
                'TextPart' => $content,  
                'HTMLPart' => $content,  
            ]  
        ]  
    ];
```

```

    ]
  ]
];
$response = $mj->post(Resources::$Email, ['body' => $body]);

```

```
// Read the response / Lecture de la reponse
```

```

    $response->success();
  }
}
?>

```

== Démonstration d'achat de plusieurs produits sur le site

=== Choix des produits

Cette page affiche les produits et permet de filtrer les produits par catégorie.

image:1\_choix\_produit.png[align=center]

Nous avons sélectionné quelques produits pour la démonstration.

=== Panier

Cette page, le panier, affiche les produits choisis afin d'être achetés.

image:2\_panier.png[align=center]

=== Validation du panier

Cette page affiche le montant de la commande, et permet d'indiquer l'adresse de livraison et le choix du transporteur, ce qui rajoute un coût de livraison.

image:3\_validation.png[align=center]

=== Récapitulatif

Cette page est un récapitulatif du montant de la commande et des frais de livraisons.

Pour cette commande nous avons un montant de 541,90 Euro que nous devons retrouver dans le site Stripe, rubrique paiements.



image:4\_recapitulatif.png[align=center]

=== Paiement avec le formulaire stripe

Cette page est fournie par Stripe, et on voit afficher la désignation des produits et les prix et le montant total, ce qui doit, après validation sur le bouton payer, être afficher dans le site Stripe rubrique paiement.

image:5\_paiement\_avec\_module\_stripe.png[align=center]

=== Confirmation de la commande

Cette page s'affiche lorsque le paiement s'est bien effectué, nous irons vérifier sur le site de Stripe que les informations correspondent.

image:6\_confirmation\_commande.png[align=center]

=== Vérification du paiement sur le site Stripe.

Notre vérification montre bien que la ligne de notre commande est d'un montant de 541.90 Euro.

Pour voir les détails, il faut cliquer sur la ligne ce qui affiche le détail de la commande.

image:7\_verification\_paiement\_dans\_stripe.png[align=center]

=== Détail de la commande dans Stripe

Dans le détail de la commande, nous avons les informations comme le montant, la date de paiement, le nom et le mail du client, et les informations sur les articles achetés.

image:8\_verification\_detail\_paiement\_dans\_stripe.png[align=center]

== Veille sécurité

=== Faille

==== Résumé

Une vulnérabilité a été découverte dans Symfony. Elle permet à un attaquant de provoquer une injection de requêtes illégitimes par rebond (CSRF).

Le composant de formulaire Symfony fournit un mécanisme de protection CSRF en utilisant un jeton aléatoire injecté dans le formulaire et en utilisant la session pour stocker et contrôler le jeton soumis par l'utilisateur.

Lors de l'utilisation du FrameworkBundle, cette protection peut être activée ou désactivée avec la configuration. Si la configuration n'est pas précisée, par défaut, le mécanisme est activé tant que la session est activée.

Dans un changement récent dans la façon dont la configuration est chargée, le

comportement par défaut a été abandonné et, par conséquent, la protection CSRF n'est pas activée sous forme lorsqu'elle n'est pas explicitement activée, ce qui rend l'application sensible aux attaques CSRF.

#### ==== Solution

Symfony a restauré la configuration par défaut pour activer la protection CSRF par défaut.

(<https://github.com/symfony/symfony/commit/f0ffb775febd07e57117aabac96fa37857f50>)

#### ==== Documentation

===== Bulletin de security Symfony du 29 Janvier 2022

<https://github.com/symfony/symfony/security/advisories/GHSA-vvmr-8829-6whx>

===== Référence CVE CVE-2022-23601

<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-23601>

#### == Recherche Création EasyAdmin

Pour ce projet, j'ai du comprendre et faire des recherches sur la mise en place du module EasyAdmin

#### === Installation

Pour installer EasyAdmin, il faut lancer la ligne de commande :

```
php bin/console make:admin:dashboard
```

#### === Configuration du dashboard

Dans mon projet j'ai appelé ce fichier DashboardController.php

Ci-dessous, c'est un exemple pour comprendre comment on doit configurer le dashboard :

```
[source,php]
```

```
<?php
```

```
namespace App\Controller\Admin;
```

```
use EasyCorp\Bundle\EasyAdminBundle\Config\Dashboard; use
EasyCorp\Bundle\EasyAdminBundle\Controller\AbstractDashboardContr
oller;
```

class DashboardController extends AbstractDashboardController { // ...

```
public function configureDashboard(): Dashboard
{
    return Dashboard::new()
        // the name visible to end users
        ->setTitle('ACME Corp.')
        // you can include HTML contents too (e.g. to link to an image)
        ->setTitle(' ACME <span class="text-
small">Corp.</span>')
```

```
// by default EasyAdmin displays a black square as its default favicon;
// use this method to display a custom favicon: the given path is passed
// "as is" to the Twig asset() function:
// <link rel="shortcut icon" href="{{ asset('...') }}">
->setFaviconPath('favicon.svg')
```

```
// the domain used by default is 'messages'
->setTranslationDomain('my-custom-domain')
```

```
// there's no need to define the "text direction" explicitly because
// its default value is inferred dynamically from the user locale
->setTextDirection('ltr')
```

```
// set this option if you prefer the page content to span the entire
// browser width, instead of the default design which sets a max width
->renderContentMaximized()
```

```
// set this option if you prefer the sidebar (which contains the main menu)
// to be displayed as a narrow column instead of the default expanded design
->renderSidebarMinimized()
```

```
// by default, users can select between a "light" and "dark" mode for the
// backend interface. Call this method if you prefer to disable the "dark"
// mode for any reason (e.g. if your interface customizations are not ready
// for it)
->disableDarkMode()
```

```

        // by default, all backend URLs are generated as absolute URLs. If
you
        // need to generate relative URLs instead, call this method
        ->generateRelativeUrls()
    ;
    }
}

```

### === Configuration du menu

Dans le fichier `DashboardController.php`, pour configurer le menu, il faut rajouter la fonction `configureMenuItems()` :

[source,php]

```
<?php
```

```

public function configureMenuItems(): iterable { return [
MenuItem::linkToDashboard('Dashboard', 'fa fa-home'),

```

```

MenuItem::section('Blog'),
MenuItem::linkToCrud('Categories', 'fa fa-tags', Category::class),
MenuItem::linkToCrud('Blog Posts', 'fa fa-file-text', BlogPost::class),

```

```

        MenuItem::section('Users'),
        MenuItem::linkToCrud('Comments', 'fa fa-comment', Comment::class),
        MenuItem::linkToCrud('Users', 'fa fa-user', User::class),
    ];
}

```

### === Ajout d'un lien

Pour accéder au site du back office j'ai dû trouver quel code utiliser dans la fonction `configureMenuItems()`.

Cette méthode est :  
[source,php]

```
<?php
```

```
return [ // linkToRoute permet de créer un lien pour retourner au site
        yield MenuItem::linkToRoute('Retour', 'fa fa-home', 'home')
    ];
```

=== Prise

<<<

== Traduction du site

=== Extrait du site anglophone

[quote,EasyAdmin documentation]

Dashboards are the entry point of backends and they link to one or more resources. Dashboards also display a main menu to navigate the resources and the information of the logged in user.

Imagine that you have a simple application with three Doctrine entities: users, blog posts and categories. Your own employees can create and edit any of them but external collaborators can only create blog posts. (789 signes)

You can implement this in EasyAdmin as follows:

- . Create three CRUD controllers (e.g. `UserCrudController`, `BlogPostCrudController` and `CategoryCrudController`);
- . Create a dashboard for your employees (e.g. `DashboardController`) and link to the three resources;
- . Create a dashboard for your external collaborators (e.g. `ExternalDashboardController`) and link only to the `BlogPostCrudController` resource.

Technically, dashboards are regular Symfony controllers so you can do anything you usually do in a controller, such as injecting services and using shortcuts like `$this->render()` or `$this->isGranted()`.

Dashboard controller classes must implement the `EasyCorp\Bundle\EasyAdminBundle\Contracts\Controller\DashboardControllerInterface`, which ensures that certain methods are defined in the

dashboard. Instead of implementing the interface, you can also extend from the `AbstractDashboardController` class. Run the following command to quickly generate a dashboard controller:

```
$ php bin/console make:admin:dashboard
```

If you now visit the `/admin` URL of your application, you'll see the default EasyAdmin Welcome Page:

```
<<<

=== Traduction en français

[quote]
```

Les tableaux de bord sont le point d'entrée des backends et ils sont liés à une ou plusieurs ressources. Les tableaux de bord affichent également un menu principal pour naviguer dans les ressources et les informations de l'utilisateur connecté.

Imaginez que vous ayez une application simple avec trois entités Doctrine : utilisateurs, articles de blog et catégories. Vos propres employés peuvent créer et modifier n'importe lequel d'entre eux, mais les collaborateurs externes ne peuvent créer que des articles de blog.

Vous pouvez l'implémenter dans EasyAdmin comme suit :

- . Créez trois contrôleurs CRUD (par exemple, `UserCrudController`, `BlogPostCrudController` et `CategoryCrudController`) ;
- . Créez un tableau de bord pour vos employés (par exemple `DashboardController`) et un lien vers les trois ressources ;
- . Créez un tableau de bord pour vos collaborateurs externes (par exemple, `ExternalDashboardController`) et créez un lien uniquement vers la ressource `BlogPostCrudController`.

Techniquement, les tableaux de bord sont des contrôleurs Symfony standard, vous pouvez donc faire tout ce que vous faites habituellement dans un contrôleur, comme injecter des services et utiliser des raccourcis comme `$this->render()` ou `$this->isGranted()`.

Les classes de contrôleur de tableau de bord doivent implémenter `EasyCorp\Bundle\EasyAdminBundle\Contracts\Controller\DashboardControllerInterface`, qui garantit que certaines méthodes sont définies dans le tableau de bord. Au lieu d'implémenter l'interface, vous pouvez également étendre la classe `AbstractDashboardController`. Exécutez la commande suivante pour générer rapidement un contrôleur de tableau de bord :

```
$ php bin/console make:admin:dashboard
```

Si vous visitez maintenant l'URL `/admin` de votre application, vous verrez la page d'accueil EasyAdmin par défaut :

<<<

## == Glossaire et Définitions

**Back-Office::** Interfaces d'un service présenté aux utilisateurs finaux.

**CSS::** Cascading Style Sheets. Langage informatique de mise en forme de contenu HTML.

**Framework::** Ensemble de bibliothèques définissant un cadre de développement de logiciel.

**Front-Office::** Interfaces d'un service servant à l'administration de celui-ci.

**HTML::** Hypertext Markup Language. Langage informatique à base de balises définissant la structure et le contenu d'une page web.

**IDE::** Integrated Development Environment. Interface de développement comprenant la coloration syntaxique, la détection d'erreur ou encore la mise en forme du code.

**IHM::** Interface Homme Machine. L'ensemble des interfaces utilisées par un utilisateur humain pour communiquer avec la machine.

**ORM::** Object Relational Mapping. Bibliothèque logiciel d'accès à une base de données qui transforme les lignes de données en objet utilisable par le langage de programmation.

**Responsive Design::** Ensemble de techniques permettant aux pages web de s'adapter à la taille des écrans sur lesquels elles s'affichent.

**SGBDR::** System de Gestion de Base de Données Relationnelles.

**SQL::** Structured Query Language. Langage pour interroger les bases de données.

**URL::** Uniform Resource Locator. Adresse d'un site ou d'une page hypertexte sur internet

**UX::** User eXpérience. Expérience Utilisateur vécue dans la globalité de l'interaction avec le service, prenant en compte l'ergonomie, l'utilisabilité, l'impact émotionnel ressenti.

**Wireframe::** représentation sous forme de ligne du squelette d'une page web

**Workflow::** processus d'automatisation des tâches d'une application

## 8.2. GLOSSAIRE ET DÉFINITIONS

### Back-Office

Interfaces d'un service présenté aux utilisateurs finaux.

### CSS

Cascading Style Sheets. Langage informatique de mise en forme de contenu HTML.

### Framework

Ensemble de bibliothèques définissant un cadre de développement de logiciel.

### Front-Office

Interfaces d'un service servant à l'administration de celui-ci.

### HTML

Hypertext Markup Langage. Langage informatique à base de balises définissant la structure et le contenu d'une page web.

### IDE

Integrated Development Environment. Interface de développement comprenant la coloration syntaxique, la détection d'erreur ou encore la mise en forme du code.

### IHM

Interface Homme Machine. L'ensemble des interfaces utilisées par un utilisateur humain pour communiquer avec la machine.

### ORM

Object Relational Mapping. Bibliothèque logiciel d'accès à une base de données qui transforme les lignes de données en objet utilisable par le langage de programmation.

### Responsive Design

Ensemble de techniques permettant aux pages web de s'adapter à la taille des écrans sur lesquels elles s'affichent.



## SGBDR

System de Gestion de Base de Données Relationnelles.

## SQL

Structured Query Language. Langage pour interroger les bases de données.

## URL

Uniform Resource Locator. Adresse d'un site ou d'une page hypertexte sur internet

## UX

User eXpérience. Expérience Utilisateur vécue dans la globalité de l'interaction avec le service, prenant en compte l'ergonomie, l'utilisabilité, l'impact émotionnel ressenti.

## Wireframe

représentation sous forme de ligne du squelette d'une page web

## Workflow

processus d'automatisation des tâches d'une application

## 8.3. FICHER SQL DE CRÉATION DE BASE

```
-- phpMyAdmin SQL Dump

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";

--
-- Base de données : `e-commerce-symfo`
--

-- DROP database IF EXISTS e-commerce-symfo;
-- Create Database e-commerce-symfo;
use altrh1730611_12m9mhg;
-- -----
```

```
--
-- Structure de la table `address`
--
DROP TABLE IF EXISTS address ;

CREATE TABLE `address` (
  `id` int NOT NULL,
  `user_id` int NOT NULL,
  `name` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
NULL,
  `firstname` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci
NOT NULL,
  `lastname` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
NULL,
  `company` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci
DEFAULT NULL,
  `address` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
NULL,
  `postal` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
NULL,
  `city` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
NULL,
  `country` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
NULL,
  `phone` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

--
-- Structure de la table `carrier`
--
DROP TABLE IF EXISTS carrier ;
CREATE TABLE `carrier` (
  `id` int NOT NULL,
  `name` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
NULL,
  `description` longtext CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
NULL,
  `price` double NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

--
-- Déchargement des données de la table `carrier`
--

INSERT INTO `carrier` (`id`, `name`, `description`, `price`) VALUES
(1, 'Chronopost', 'Livraison standard à domicile en 2 jours ouvrés.', 400),
```

```

(2, 'La Poste', 'Si vous préférez trouver dans votre boîte à lettres un avis
de passage à la place de votre commande.', 190);

-----

--
-- Structure de la table `category`
--
DROP TABLE IF EXISTS category ;
CREATE TABLE `category` (
  `id` int NOT NULL,
  `name` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
  NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

--
-- Déchargement des données de la table `category`
--

INSERT INTO `category` (`id`, `name`) VALUES
(2, 'Manteaux'),
(3, 'Echarpes'),
(4, 'Bonnets'),
(5, 'T-shirts'),
(6, 'Chaussures'),
(7, 'Lunettes'),
(8, 'Chapeaux'),
(9, 'Casquettes');

-----

--
-- Structure de la table `doctrine_migration_versions`
--
DROP TABLE IF EXISTS doctrine_migration_versions ;
CREATE TABLE `doctrine_migration_versions` (
  `version` varchar(191) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
  `executed_at` datetime DEFAULT NULL,
  `execution_time` int DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3 COLLATE=utf8_unicode_ci;

--
-- Structure de la table `headers`
--
DROP TABLE IF EXISTS headers ;
CREATE TABLE `headers` (
  `id` int NOT NULL,

```

```

`title` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
`content` longtext COLLATE utf8mb4_unicode_ci NOT NULL,
`btn_title` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
`btn_url` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
`image` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

--
-- Déchargement des données de la table `headers`
--

INSERT INTO `headers` (`id`, `title`, `content`, `btn_title`, `btn_url`,
`image`) VALUES
(3, 'Accessoires pour toutes les saisons', 'Laissez vous tenter par nos
lunettes, chapeaux aussi bien que nos bonnets ou nos écharpes !', 'Nos
articles', '/articles', '2bdb8484b1ba1c36d19137a2c6b44a314cfdbb8.jpg'),
(4, 'Livraison rapide', 'Profitez de nos services de livraison express, mais
aussi de nos retours gratuits ! Essayez, adoptez ou échangez !', 'Je
commande', '/articles', '4ba7091ffffbc5fa736faa53e65ce7b1d47dd4f8e.jpg'),
(5, 'Chic et élégant', 'Nos derniers articles soldés sur la collection
automne', 'Voir', '/articles',
'a92c95e83126727c338110afae40dddb1608e9c4.jpg');

--
-- Structure de la table `order`
--

DROP TABLE IF EXISTS `order` ;

CREATE TABLE `order` (
  `id` int NOT NULL,
  `user_id` int NOT NULL,
  `created_at` datetime NOT NULL,
  `carrier_name` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci
NOT NULL,
  `carrier_price` varchar(255) CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci NOT NULL,
  `delivery` longtext CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
NULL,
  `reference` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `stripe_session` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `state` int NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

--
-- Structure de la table `order_details`
--

```

```

DROP TABLE IF EXISTS order_details ;

CREATE TABLE `order_details` (
  `id` int NOT NULL,
  `binded_order_id` int NOT NULL,
  `product` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `quantity` int NOT NULL,
  `price` double NOT NULL,
  `total` double NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

--
-- Structure de la table `product`
--
DROP TABLE IF EXISTS product ;
CREATE TABLE `product` (
  `id` int NOT NULL,
  `category_id` int NOT NULL,
  `name` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
NULL,
  `slug` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
NULL,
  `image` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
NULL,
  `subtitle` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
NULL,
  `description` longtext CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
NULL,
  `price` double NOT NULL,
  `show_top_vente` int NOT NULL DEFAULT 0
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

--
-- Déchargement des données de la table `product`
--

INSERT INTO `product` (`id`, `category_id`, `name`, `slug`, `image`,
`subtitle`, `description`, `price`, `show_top_vente`) VALUES
(2, 4, 'Bonnet Rouge Pompom', 'bonnet-rouge-pompom',
'e0b7e89b60de57f1e4451fd9831be26a102081e4.jpg', 'Restez au chaud avec style',
'Lorem ipsum dolor sit amet consectetur adipisicing elit. Maxime
mollitia,\r\nmolestiae quas vel sint commodi repudiandae consequuntur
voluptatum laborum\r\nnumquam blanditiis harum quisquam eius sed odit fugiat
iusto fuga praesentium\r\noptio, eaque rerum! Provident similique accusantium
nemo autem. Veritatis\r\nnobcaecati tenetur iure eius earum ut molestias
architecto voluptate aliquam\r\nnihil, eveniet aliquid culpa officia aut!
Impedit sit sunt quaerat, odit,\r\nntenetur error, harum nesciunt ipsum debitis

```

```

quas aliquid.', 1800, 0),
(3, 4, 'Bonnet Rouge et Bleu Pompom', 'bonnet-rouge-et-bleu-pompom',
'8bf330b14aefcec1ab92b7c3f475290fd7940275.jpg', 'Restez au chaud avec style
(encore)', 'Lorem ipsum dolor sit amet consectetur adipisicing elit. Maxime
mollitia,\r\nmolestiae quas vel sint commodi repudiandae consequuntur
voluptatum laborum\r\nnumquam blanditiis harum quisquam eius sed odit fugiat
iusto fuga praesentium\r\noptio, eaque rerum! Provident similique accusantium
nemo autem. Veritatis\r\nnobcaecati tenetur iure eius earum ut molestias
architecto voluptate aliquam\r\nnihil, eveniet aliquid culpa officia aut!',
1600, 0),
(4, 3, 'Echarpe Rouge', 'echarpe-rouge',
'9ec1111e99942243bf2157e2fbaea2aec0a485ac.jpg', 'Pour le ski ou la ville',
'Lorem ipsum dolor sit amet consectetur adipisicing elit. Maxime
mollitia,\r\nmolestiae quas vel sint commodi repudiandae consequuntur
voluptatum laborum\r\nnumquam blanditiis harum quisquam eius sed odit fugiat
iusto fuga praesentium\r\noptio, eaque rerum!', 1400, 0),
(5, 2, 'Manteau chaud', 'manteau-chaud',
'5cf449463c611b07451480da81fa048f208fc974.jpg', 'Pour les hivers rudes',
'Lorem ipsum dolor sit amet consectetur adipisicing elit. Maxime
mollitia,\r\nmolestiae quas vel sint commodi repudiandae consequuntur
voluptatum laborum\r\nnumquam blanditiis harum quisquam eius sed odit fugiat
iusto fuga praesentium\r\noptio, eaque rerum! Provident similique accusantium
nemo autem. Veritatis\r\nnobcaecati tenetur iure eius earum ut molestias
architecto voluptate aliquam\r\nnihil, eveniet aliquid culpa officia aut!
Impedit sit sunt quaerat, odit,\r\nntenetur error, harum nesciunt ipsum debitis
quas aliquid.\r\nLorem ipsum dolor sit amet consectetur adipisicing elit.',
5800, 0),
(6, 5, 'T-shirt Blanc', 't-shirt-blanc',
'ee40d9f7990aead7e8695f2eb4599ea1dbb0b1b1.jpg', 'Simple, basique', 'Lorem
ipsum dolor sit amet consectetur adipisicing elit. Maxime
mollitia,\r\nmolestiae quas vel sint commodi repudiandae consequuntur
voluptatum laborum\r\nnumquam blanditiis harum quisquam eius sed odit fugiat
iusto fuga praesentium\r\noptio, eaque rerum!', 1800, 0),
(7, 3, 'Echarpe Rayée', 'echarpe-rayee',
'2153f235c1175c595860b612a4d2657402184a04.jpg', 'La classe à Annemasse',
'Lorem ipsum dolor sit amet consectetur adipisicing elit. Maxime
mollitia,\r\nmolestiae quas vel sint commodi repudiandae consequuntur
voluptatum laborum\r\nnumquam blanditiis harum quisquam eius sed odit fugiat
iusto fuga praesentium\r\noptio, eaque rerum! Provident similique accusantium
nemo autem. Veritatis\r\nnobcaecati tenetur iure eius earum ut molestias
architecto voluptate aliquam\r\nnihil, eveniet aliquid culpa officia aut!
Impedit sit sunt quaerat, odit,\r\nntenetur error, harum nesciunt ipsum debitis
quas aliquid.', 1500, 0),
(8, 5, 'T-Shirt Moulant', 't-shirt-moulant',
'00a9e824e1a5ba119d9964cc2798fdc3e27b9a84.jpg', 'Pour les bogoss\''', 'Lorem
ipsum dolor sit amet consectetur adipisicing elit. Maxime mollitia,\r
\r\nmolestiae quas vel sint commodi repudiandae consequuntur voluptatum laborum

```

```

\r\nnumquam blanditiis harum quisquam eius sed odit fugiat iusto fuga
praesentium\r\nnoptio, eaque rerum!', 15800, 0),
(9, 7, 'Lunettes new aviator', 'lunettes-new-aviator', '
9b63b18c0cde83590b9e36eaaac34092e430712c.jpg', 'Intemporel', 'Lunettes sombres
unisex, printemps, été ou automne.', 2400, 0),
(10, 7, 'Lunettes Wayfarer style', 'lunettes-wayfarer-style',
'dc0f8d0f37e4866c5882c01e0841b7f8b271e49d.jpg', 'Promis ce sont des vraies',
'Souvent imitées, jamais égales, on a la classe ou ne l\'a pas', 9800, 1),
(11, 9, 'Casquette orange', 'casquette-orange',
'242b2cf61210f43c231c6fdec081a554c6757e95.jpg', 'Contre le soleil et le bon
gout', 'Superbe casquette orange, il faut aimer le orange.', 400, 1),
(12, 6, 'Chaussures Mike', 'chaussures-mike',
'985fc0c43bac9473286fd1f4fabe1d14c21485e7.jpg', 'Mieux que l\'originale',
'Pour faire du skate ou pour rien faire sinon c\'est bien aussi rien faire.',
14900, 1);

```

```
--
--
-- Structure de la table `user`
--
DROP TABLE IF EXISTS user ;
CREATE TABLE `user` (
  `id` int NOT NULL,
  `email` varchar(180) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
NULL,
  `roles` json NOT NULL,
  `password` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
NULL,
  `firstname` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci
NOT NULL,
  `lastname` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

--
-- Structure de la table `product`
--
DROP TABLE IF EXISTS alert ;
CREATE TABLE `alert` (
  `id` int NOT NULL,
  `product_id` int NOT NULL,
  `name` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

```
--
-- Index pour les tables déchargées
--

--
-- Index pour la table `address`
--
ALTER TABLE `address`
  ADD PRIMARY KEY (`id`),
  ADD KEY `IDX_ADRESSE_USER_ID` (`user_id`);

--
-- Index pour la table `carrier`
--
ALTER TABLE `carrier`
  ADD PRIMARY KEY (`id`);

--
-- Index pour la table `category`
--
ALTER TABLE `category`
  ADD PRIMARY KEY (`id`);

--
-- Index pour la table `doctrine_migration_versions`
--
ALTER TABLE `doctrine_migration_versions`
  ADD PRIMARY KEY (`version`);

--
-- Index pour la table `headers`
--
ALTER TABLE `headers`
  ADD PRIMARY KEY (`id`);

--
-- Index pour la table `order`
--
ALTER TABLE `order`
  ADD PRIMARY KEY (`id`),
  ADD KEY `IDX_ORDER_USER_ID` (`user_id`);

--
-- Index pour la table `order_details`
--
ALTER TABLE `order_details`
```



```
ADD PRIMARY KEY (`id`),
ADD KEY `IDX_ORDER_DETAILS_BINDED_ORDER_ID` (`binded_order_id`);

--
-- Index pour la table `product`
--
ALTER TABLE `product`
  ADD PRIMARY KEY (`id`),
  ADD KEY `IDX_PRODUCT_CATEGORY_ID` (`category_id`);

--
-- Index pour la table `user`
--
ALTER TABLE `user`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `UNIQU_USER_ID_EMAIL` (`email`);

--
-- AUTO_INCREMENT pour les tables déchargées
--

--
-- AUTO_INCREMENT pour la table `address`
--
ALTER TABLE `address`
  MODIFY `id` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=7;

--
-- AUTO_INCREMENT pour la table `carrier`
--
ALTER TABLE `carrier`
  MODIFY `id` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;

--
-- AUTO_INCREMENT pour la table `category`
--
ALTER TABLE `category`
  MODIFY `id` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=10;

--
-- AUTO_INCREMENT pour la table `headers`
--
ALTER TABLE `headers`
  MODIFY `id` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=6;

--
-- AUTO_INCREMENT pour la table `order`
```

```
--  
ALTER TABLE `order`  
  MODIFY `id` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=64;  
  
--  
-- AUTO_INCREMENT pour la table `order_details`  
--  
ALTER TABLE `order_details`  
  MODIFY `id` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=106;  
  
--  
-- AUTO_INCREMENT pour la table `product`  
--  
ALTER TABLE `product`  
  MODIFY `id` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=13;  
  
--  
-- AUTO_INCREMENT pour la table `user`  
--  
ALTER TABLE `user`  
  MODIFY `id` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=30;  
  
--  
-- Contraintes pour les tables déchargées  
--  
  
--  
-- Contraintes pour la table `address`  
--  
ALTER TABLE `address`  
  ADD CONSTRAINT `FK_ADDRESS_ID_USER_ID` FOREIGN KEY (`user_id`) REFERENCES  
  `user` (`id`);  
  
--  
-- Contraintes pour la table `order`  
--  
ALTER TABLE `order`  
  ADD CONSTRAINT `FK_ORDER_ID_USER_ID` FOREIGN KEY (`user_id`) REFERENCES  
  `user` (`id`);  
  
--  
-- Contraintes pour la table `order_details`  
--  
ALTER TABLE `order_details`  
  ADD CONSTRAINT `FK_ORDER_DETAILS_ID_ORDER_ID` FOREIGN KEY  
  (`binded_order_id`) REFERENCES `order` (`id`);
```

```
--  
-- Contraintes pour la table `product`  
--  
ALTER TABLE `product`  
  ADD CONSTRAINT `FK_PRODUCT_CATEGORY_ID` FOREIGN KEY (`category_id`)  
  REFERENCES `category` (`id`);  
COMMIT;
```