

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	10
Avis.....	11
2.5.3. Entité-Relation	13
2.5.4. Plan du site	14
2.6. Spécification technique	16
2.6.1. Symfony	17
Grande flexibilité	17
Personnalisation.....	17
L'entreprise derrière la technologie	18
Une fiabilité éprouvée.....	18
Facile à utiliser	18
Support à long terme	18
Grande communauté	18
Une bonne documentation.....	18
Extensible.....	19
2.6.2. Les Contenus	19
2.6.3. Inventaire technique	19
3. Réalisation	20
3.1. Mise en place de Easy Admin	20
3.1.1. Fichier de configuration DashboardController.php	20
3.1.2. Exemple de fichier de configuration crud pour la table Avis. . .	21
3.2. Mise en place du module stripe.....	23
3.2.1. Pourquoi Stripe.....	23
3.2.2. Controller PaymentController.php	23
3.2.3. Template de succès de paiement	26
3.2.4. Template d'erreur de paiement.....	26
4. Démonstration d'achat de plusieurs produits sur le site.....	27
4.1. Choix des produits	27
4.2. Panier.....	27
4.3. Validation du panier	28

4.4. Récapitulatif	28
4.5. Paiement avec le formulaire stripe	28
4.6. Confirmation de la commande.....	29
4.7. Vérification du paiement sur le site Stripe.....	29
4.8. Détail de la commande dans Stripe.....	30
4.9. ??? Autre ???	30
5. Veille sécurité	31
5.1. Faille	31
5.1.1. Résumé.....	31
5.1.2. Solution.....	31
5.1.3. Documentation.....	31
Bulletin de security Symfony du 29 Janvier 2022	31
5.1.4. Référence CVE CVE-2022-23601.....	31
6. Recherche Création EasyAdmin.....	32
6.1. Installation.....	32
6.2. Configuration du dashboard	32
6.3. Configuration du menu	33
6.4. Ajout d'un lien.....	34
6.5. Prise.....	34
7. Traduction du site	35
7.1. Extrait du site anglophone	35
7.2. Traduction en français	36
8. Glossaire et Définitions.....	37

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

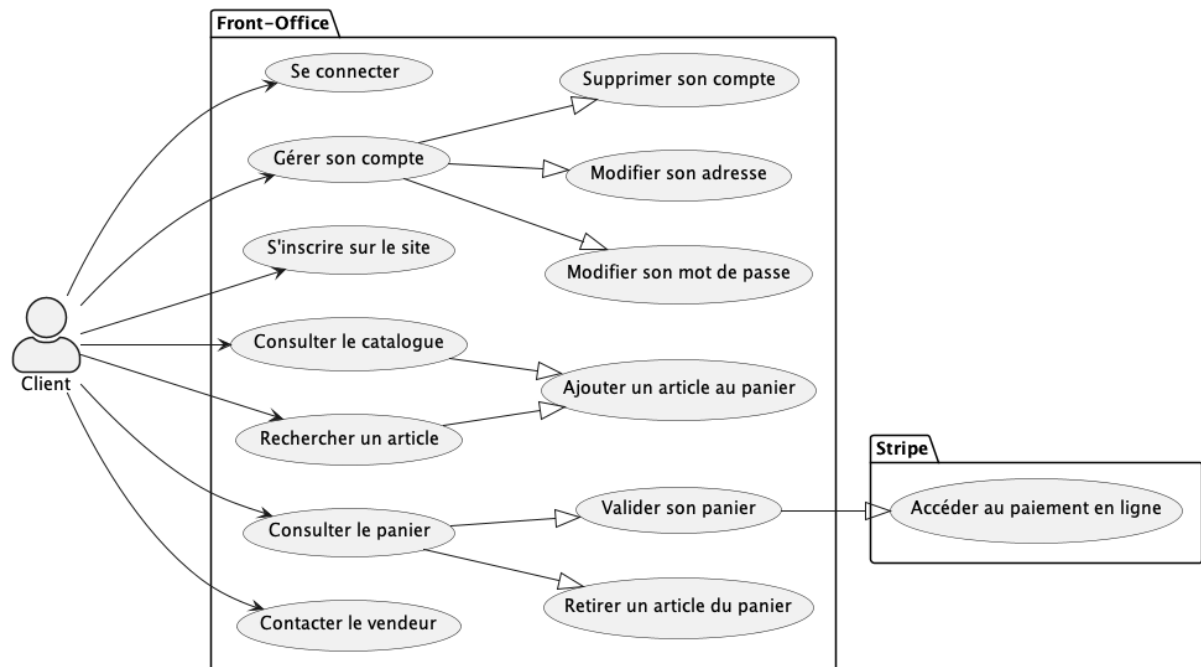
visiteurs

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

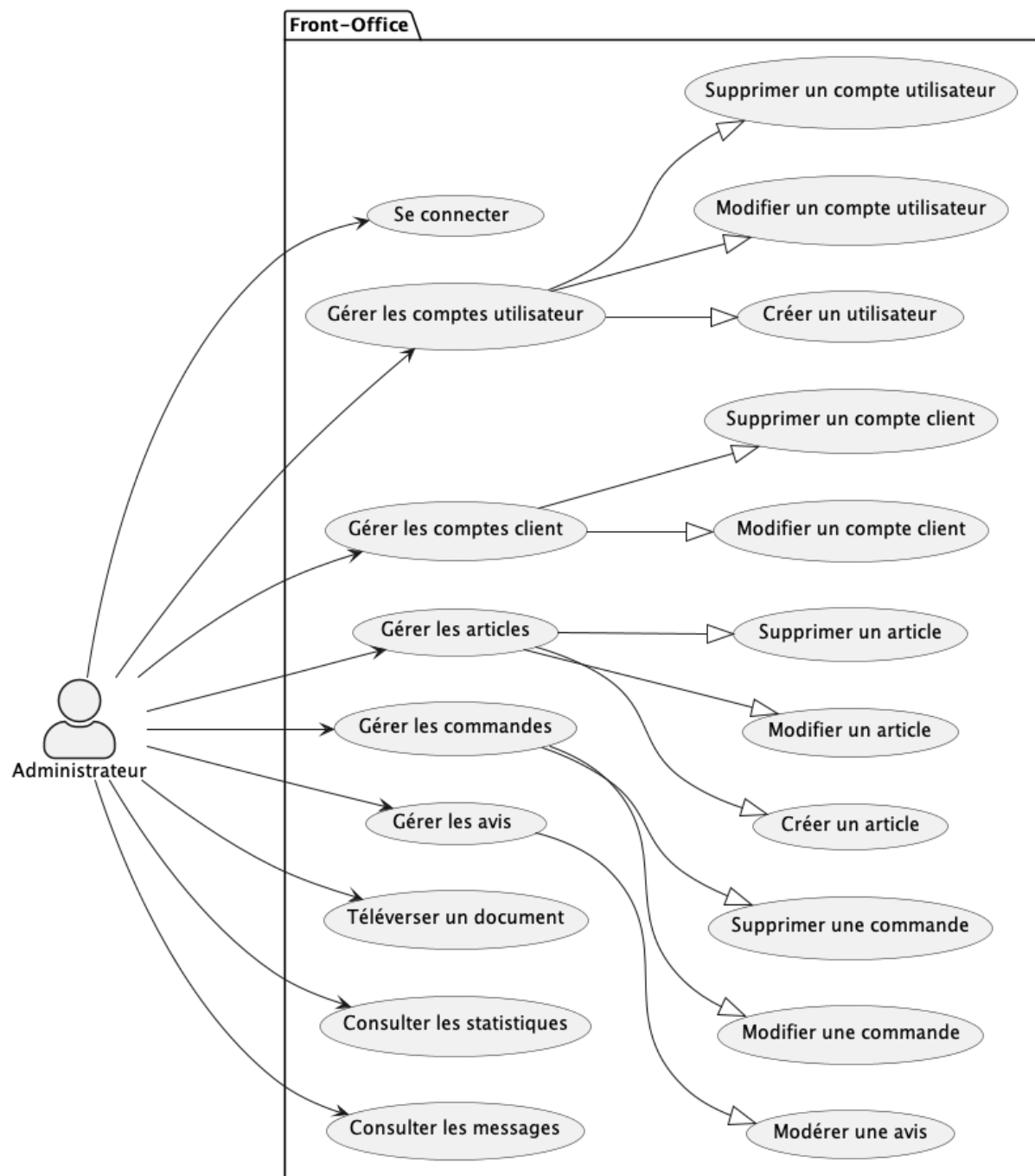
2.5. MODÉLISATION

2.5.1. CAS D'USAGE

FRONT-OFFICE

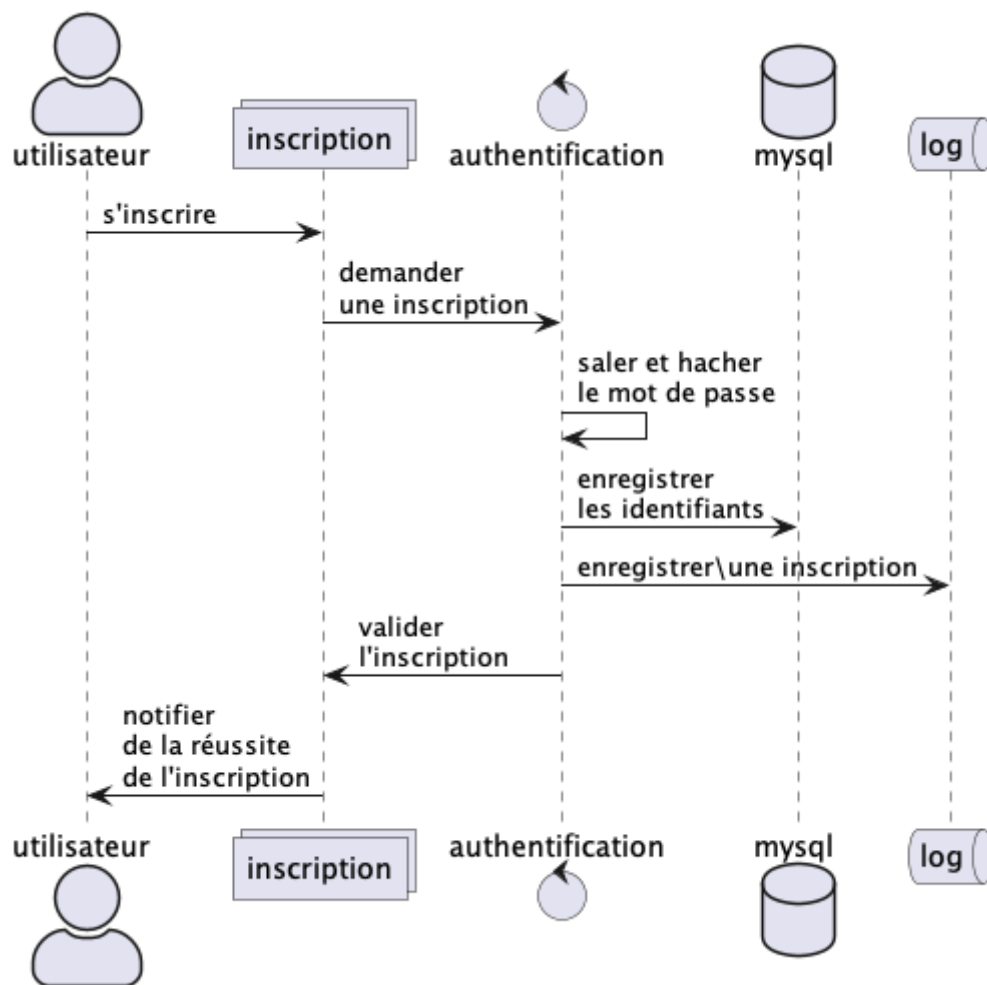


BACK-OFFICE

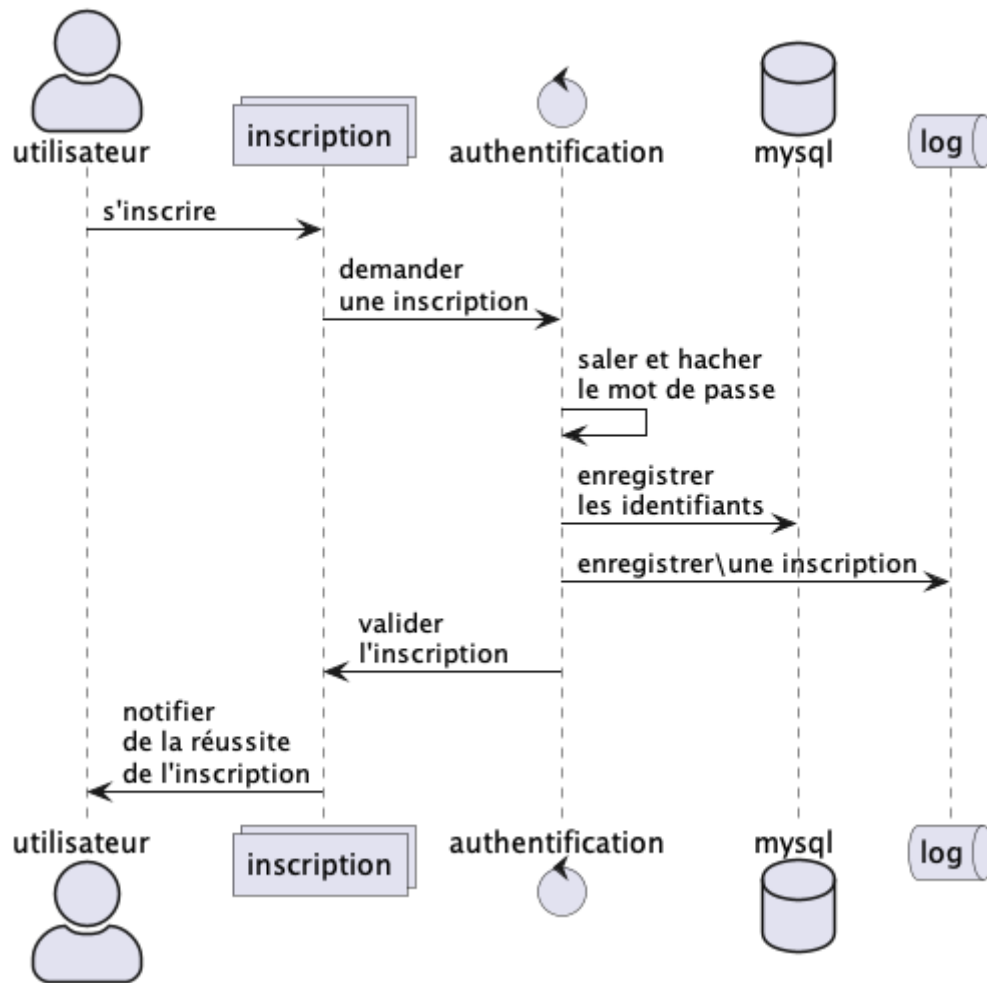


2.5.2. SÉQUENCES D' ACTIONS

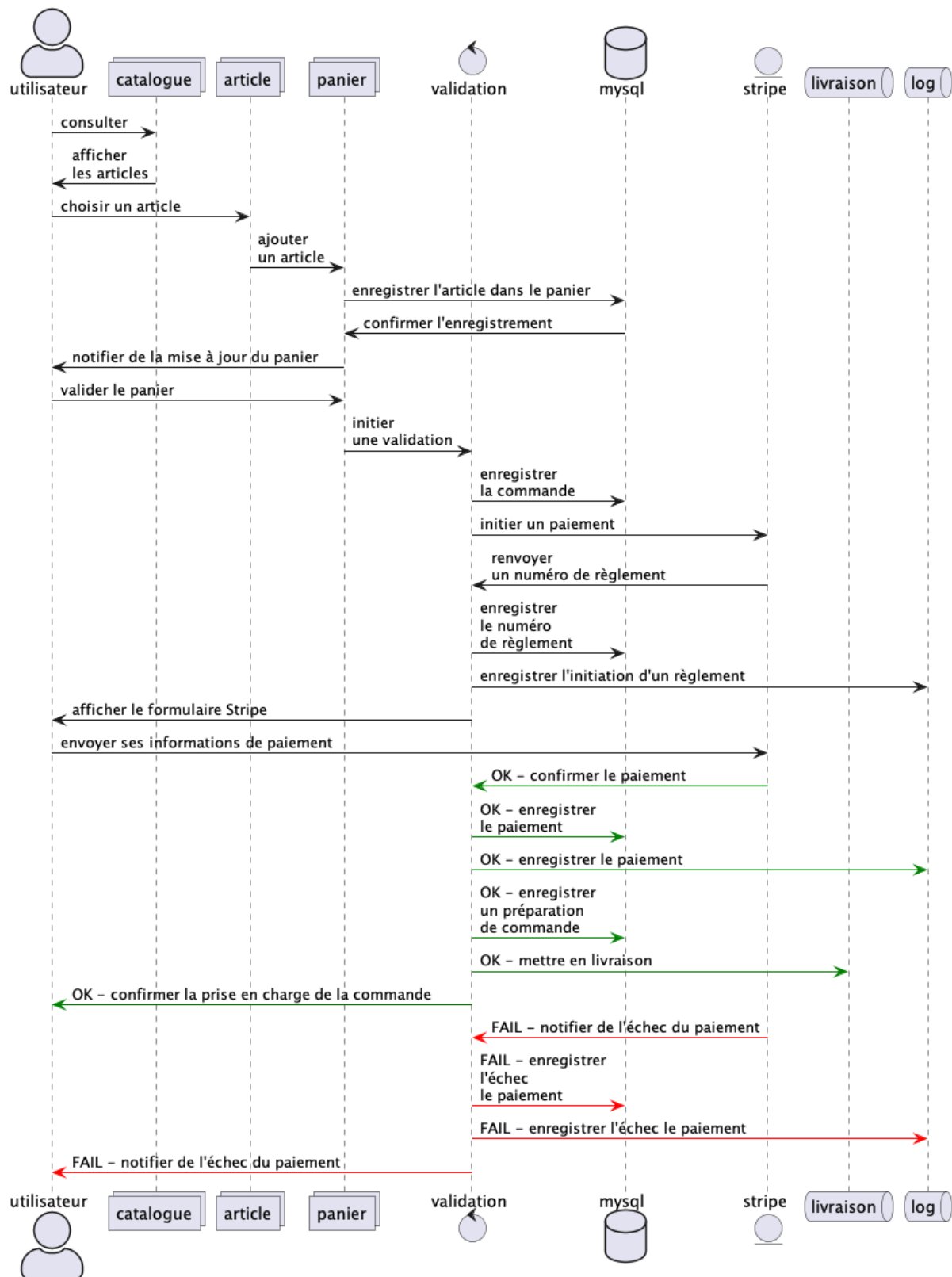
AUTHENTIFICATION



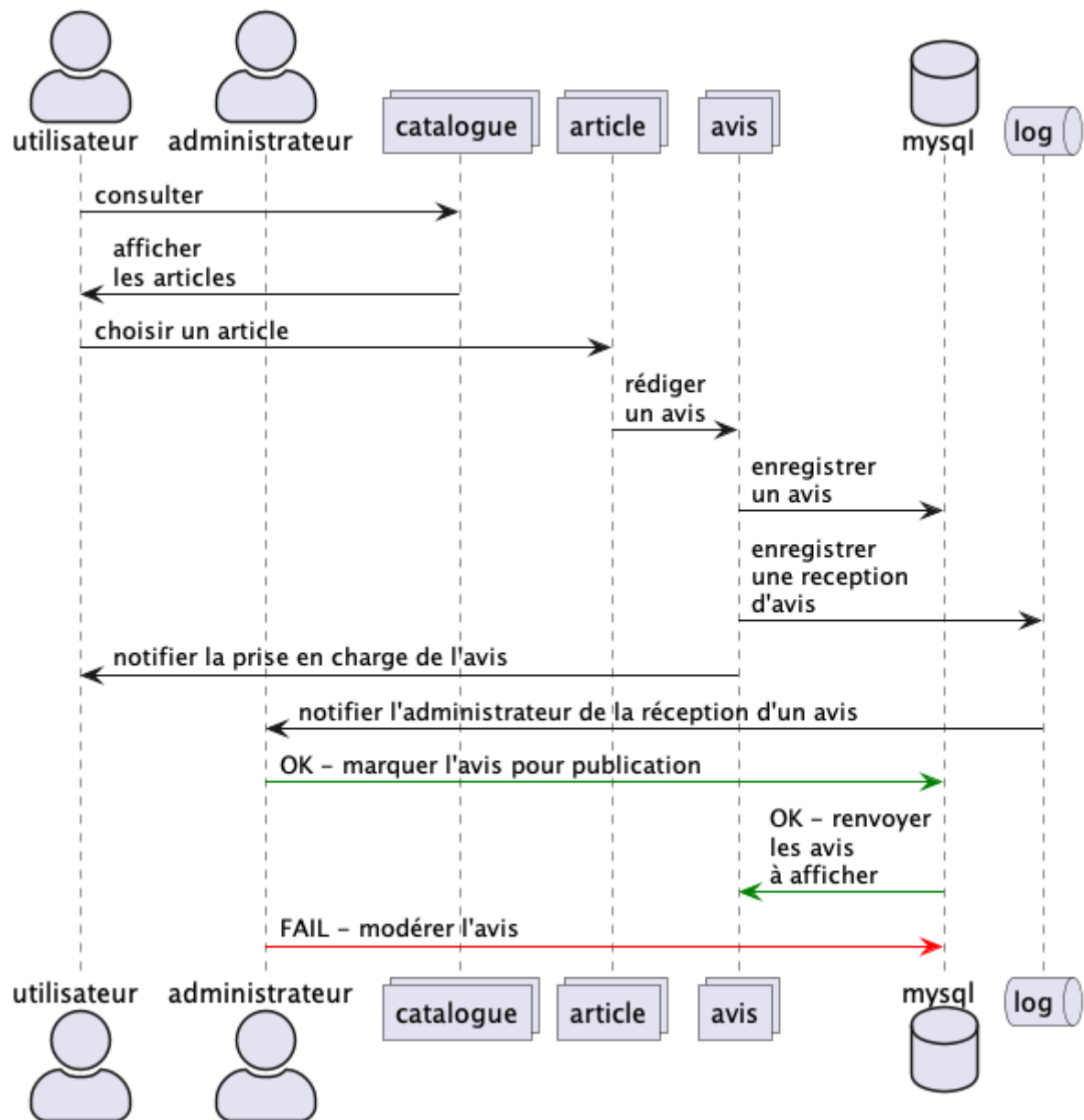
INSCRIPTION



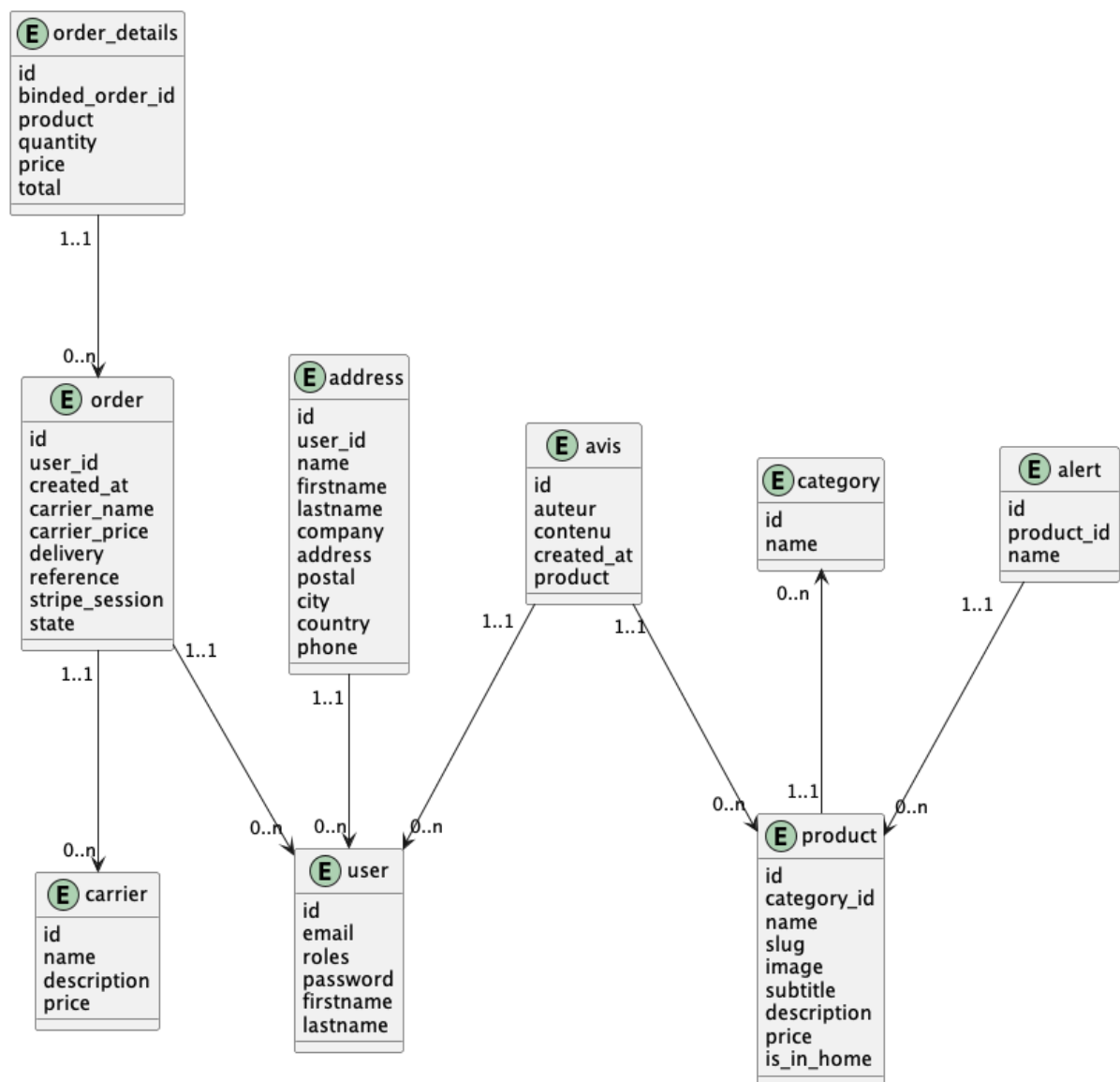
ACHAT



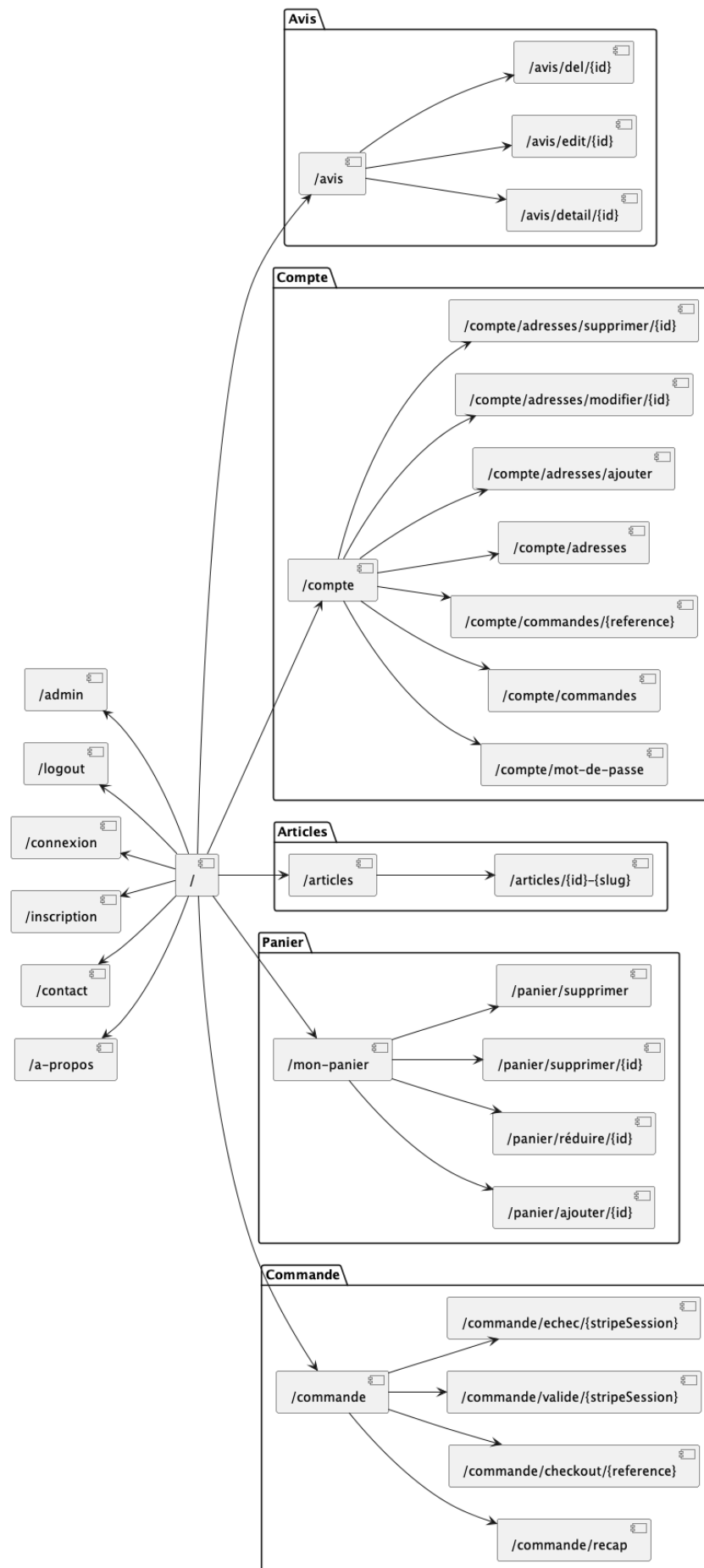
AVIS



2.5.3. ENTITÉ-RELATION



2.5.4. PLAN DU SITE



2.6. 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 suffi 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-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.

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

- 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 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.

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_51LNyQsDz6qM0cyaOB1HWM8Y6a3k7rAG040C2L3qxGbI9f5Xhsx
zUAeqgrhhKYEmSMEHAgZ3uI33kjfR96pZN01pb00NMC07VJA');
    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.

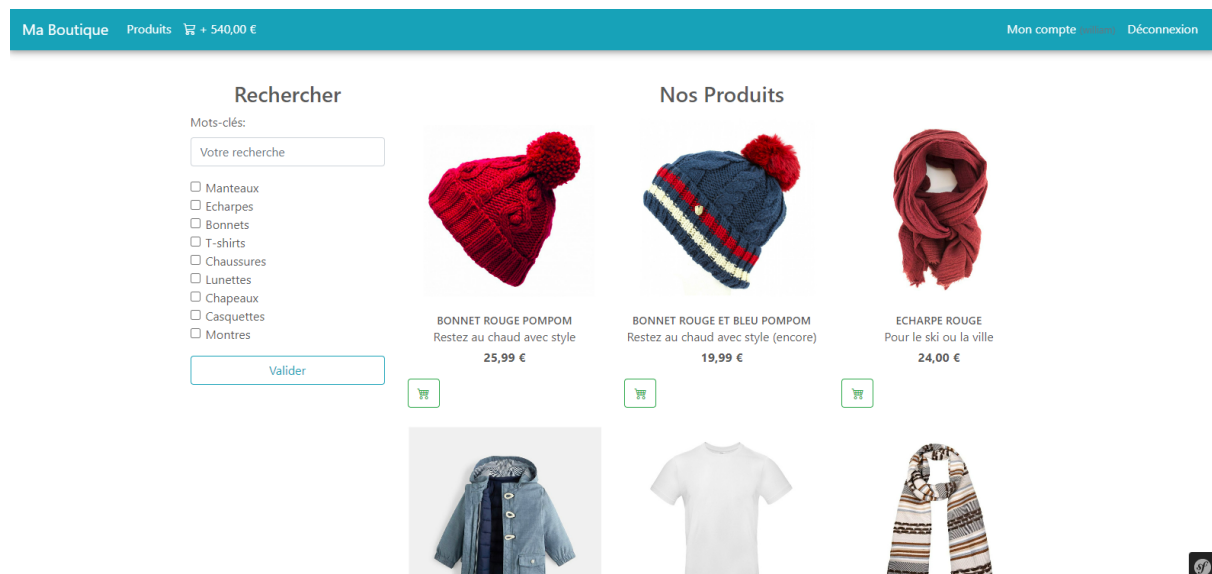
3.2.3. TEMPLATE DE SUCCÈS DE PAIEMENT

3.2.4. TEMPLATE D'ERREUR DE PAIEMENT

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

4.1. CHOIX DES PRODUITS

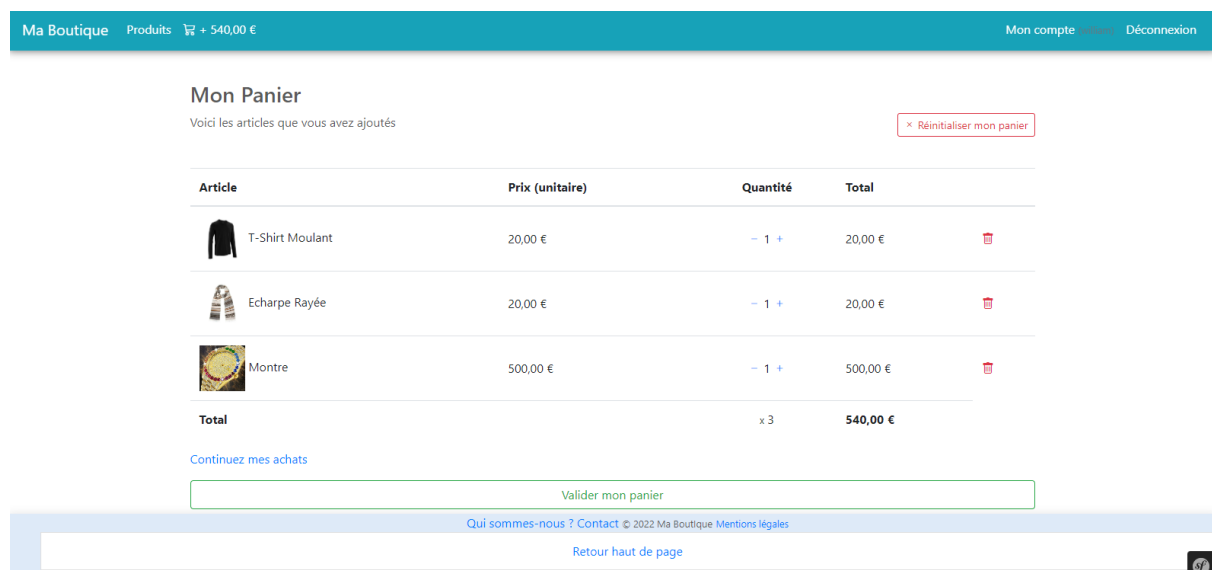
Cette page affiche les produits et permet de filtrer les produits par catégorie.



Nous avons sélectionné quelques produits pour la démonstration.

4.2. PANIER

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



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
sili
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 VISA

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 [Mon compte](#) 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](#).

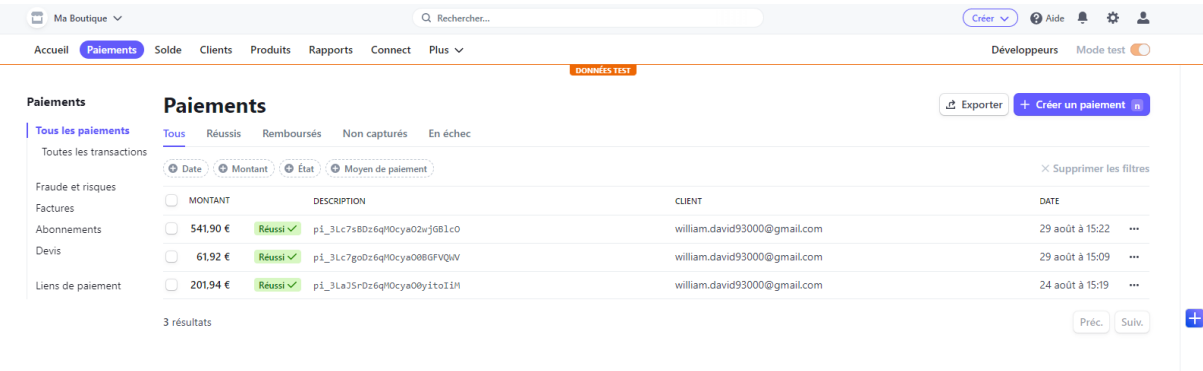
[Qui sommes-nous ?](#) [Contact](#) © 2022 Ma Boutique [Mentions légales](#)

[Retour haut de page](#)

4.7. VÉRIFICATION DU PAIEMENT SUR LE SITE STRIPE.

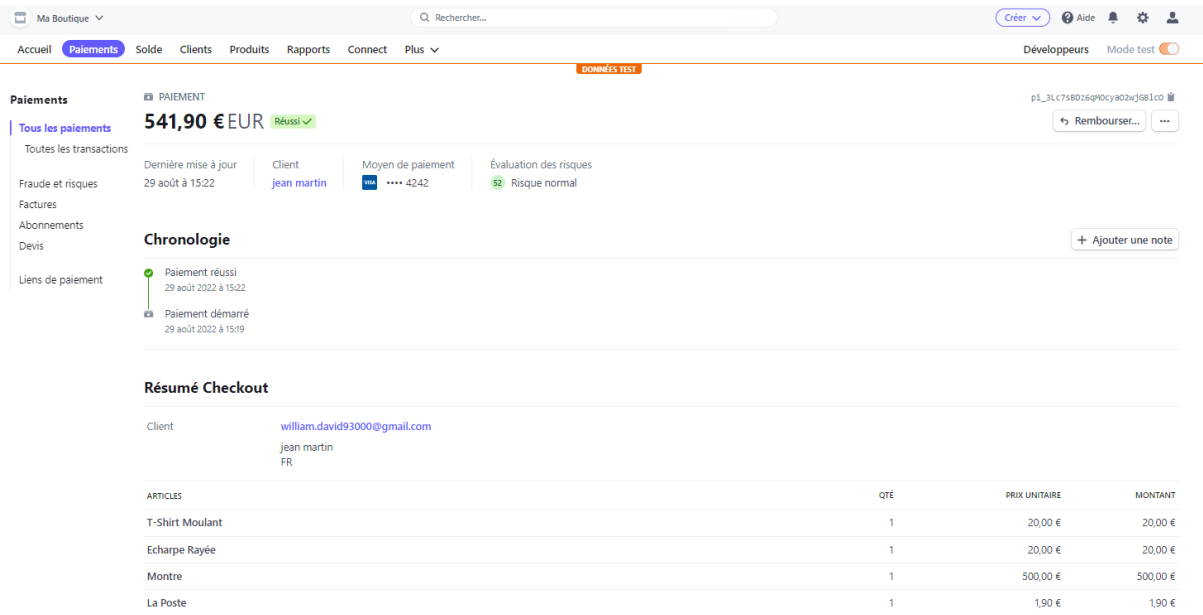
Notre vérification montre bien que la ligne de notre commande est d'un

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



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.



4.9. ??? AUTRE ???

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. 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