

Dossier de Projet

Développeur Web et Web mobile



SOMMAIRE

Liste des compétences du référentiel couvertes par le projet	3
Résumé du projet	4
Cahier des charges	
I. Présentation d'ensemble du projet	5
II. Charte graphique	6
III. Arborescence	7
IV. Maquettage	8
IV. I Extrait de la page d'accueil du site responsive (tablette et mobile)	9
V. Spécifications fonctionnelles du projet	10
VI. Spécifications techniques du projet	10
Base de données	
I. Dictionnaire des données épurées	11
II. Model conceptuel de données	12
III. Model Relationnel de données	12
IV. Model physique de données	13
V. Connexion à la base de données	14
Réalisations	
I. Extraits de code front-end commentés	15
II. Extraits de code back-end commentés	19
Présentation du jeu d'essai	28
Veille sur les vulnérabilités de sécurité	30
Description d'une situation de travail ayant nécessité une recherche à partir de site anglophone	
I. Extrait du site	32
II. Traduction de l'extrait en français	32
Conclusion	33
Annexes	34

Liste des compétences du référentiel couvertes par le projet

Activité type 1 « Développer la partie Front-end d'une application web ou web mobile en intégrant les recommandations de sécurité »	Maquetter une application	<input checked="" type="checkbox"/>
	Réaliser une interface utilisateur web statique et adaptable	<input checked="" type="checkbox"/>
	Développer une interface utilisateur web dynamique	<input checked="" type="checkbox"/>
	Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce	<input type="checkbox"/>

Activité type 2 « Développer la partie Back-end d'une application web ou web mobile en intégrant les recommandations de sécurité »	Développer les composants d'accès aux données	<input checked="" type="checkbox"/>
	Développer la partie Back-end d'une application web ou web mobile	<input checked="" type="checkbox"/>
	Elaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce	<input type="checkbox"/>

Résumé du projet

Les commanditaires du projet, Mr et Mme PARADIS gérants de la Boulangerie Pâtisserie PARADIS ont exprimé les besoins suivants :

Développer la présence en ligne (site vitrine)

Obtenir leur identité visuelle complète et la maquette de son site internet.

Se doter d'un site Web aux fonctionnalités adaptées permettrait à La Boulangerie Pâtisserie PARADIS d'assurer la pérennité (actuellement compromise) de son établissement.

le digital devient incontournable, si la boulangerie paradis passe outre, elle risque de se retrouver de côté, invisible, le consommateur a acquis le réflexe de consulter Internet pour tous ses besoins, y compris les besoins alimentaires depuis la pandémie covid.

Si le boulanger n'y répond pas présent, ses concurrents le feront en sachant que le pain reste toujours la nourriture de base des Français.

Le site est développé principalement en:

HTML  / CSS  / TWIG  / BOOTSTRAP  / PHP(Framework SYMFONY) 

J'ai effectué toute la partie programmation grâce à l'éditeur de texte Visual Studio Code et tout le maquettage grâce à la suite adobe

Le site présenté intégrera plusieurs systèmes:

 Un système de connexion qui permettra à l'administrateur du site de gérer les produits leurs catégories et leurs taxes .

 une barre de recherche qui permettra aux visiteurs du site de chercher et trouver les différents produits .

 un formulaire de contact .

Cahier des charges

I. Présentation d'ensemble du projet

1. présentation de l'entreprise

La Boulangerie Pâtisserie PARADIS est un établissement familial de prestige situé au 1 Place Saint Louis 57000 Metz. Elle est gérée de père en fils par la famille Paradis depuis 1956.

La Boulangerie Pâtisserie offre :

Des macarons, des chocolats, des viennoiseries, les signatures, les grands classiques, la boulangerie, traiteur (réception sucrée & salée)

La maison paradis fait le choix des ingrédients en utilisant des produits de saison, frais, bio, issus d'une culture raisonnée et labellisés ,elle préfère l'utilisation des emballages biodégradables ou renouvelables.

2. Etude de la concurrence

Le secteur d'activité de cette entreprise est très concurrentiel car nous trouvons de plus en plus de boulangeries proposant les mêmes types de produits, cependant très peu possède un site web.

Les principaux concurrents dans cette zone géographique et possédant un site web sont :

Pâtisserie Bourguignon Metz <https://www.bourguignonmetz.fr/>
BOULANGERIE POULARD - Metz <https://boulangeriepoulard.eatbu.com>

3. Les cibles

la cible principale de la maison paradis est la jeune clientèle féminine de la métropole, puisque près de 29 % des femmes ont moins de 35 ans. À l'inverse, les clients hommes ont plus de 50 ans pour 65% d'entre eux.

4. Objectifs du site :

Les principaux objectifs de la création de ce site web sont :

- Augmenter au maximum la visibilité.
- Améliorer le contact entre l'entreprise et les clients.
- Se démarquer de la concurrence.
- Maximiser le chiffre d'affaire.

Les objectifs quantitatifs :

- Acquisition de nouveaux clients.
- Informations au sujet de l'entreprise.
- Promouvoir un produit ou un service.
- Augmenter ses ventes de produits.
- Acquérir 3 nouveaux clients par mois dans un premier temps (à la fin de la première année) : Objectif réaliste et temporel, puisque basée sur l'étude du marché et la concurrence.

II . Charte graphique

Le logotype



Boulangerie patisserie paradis , avec son logo rappelle les courbes qu'on retrouve dans l'art nouveau dont elle s'inspire. En y associant une typographie plus moderne, elle rend hommage à cet art et continue à le perpétrer à travers le temps. La formes des épis de blé est un rappel de la pâtisserie, le mélange d'ingédients basiques qui abouttit à un harmonieux résultat.

La typographie

Monichaa : la police du logo

Boulangerie Patisserie Paradis

Tekton Pro : La police du site internet

Boulangerie Patisserie Paradis

Les couleurs

Les couleurs que j'ai choisis pour le site :

Jaune doré (# dea505)

Appartenant au champ chromatique du jaune, le doré symbolise généralement le pouvoir, le luxe, la puissance et la richesse...



Maron Chocolat (# 5d3828)

Cette couleur a la particularité de réchauffer l'ambiance, de rassurer, de rappeler quelque chose du cocooning, malgré son apparence foncée...

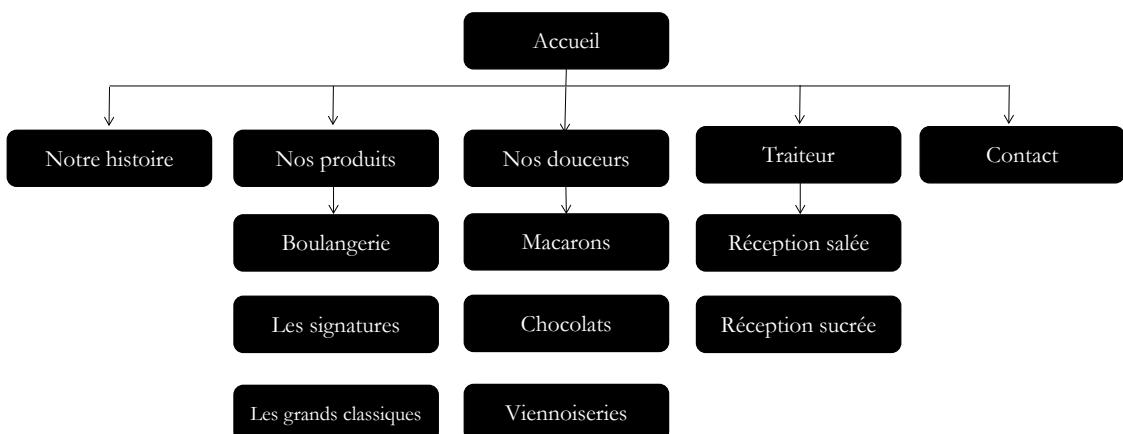


Beige (# f6efd5)

Le beige est une couleur chaleureuse qui fait référence à la douceur, l'élégance...



III. Arborescence



IV. Maquettage

— Maquette de la page d'accueil —

La maquette de la page d'accueil du site web de la boulangerie Paradies est une composition en deux colonnes. La colonne de gauche contient des images et des textes présentant les produits et la qualité de la boulangerie. La colonne de droite contient un résumé de l'histoire familiale et les coordonnées de la boulangerie.

Bar de navigation: Boulangerie Paradies, Notre histoire, Nos produits, Nos douceurs, Traiteur, Contact, Recherche.

Section Histoire: Paradies une histoire de famille
La boulangerie pâtisserie PARADIS est un établissement familial de prestige situé au 1 place Saint Louis 57000 Metz. Elle est gérée de père en fils par la famille Paradis 1956. La maison paradis a su se développer en restant fidèle aux valeurs familiales à l'origine de cette maison.

Section Produits:

- Savourez nos différentes variétés de baguettes
- Un amour de bons produits
Nos matières premières sont soigneusement sélectionnées auprès des meilleurs fournisseurs pour fabriquer dans nos boutiques comme dans notre atelier des produits de qualité. On est passionnés par notre métier on est à cœur de vous offrir le meilleur de nos produits.
- Un savoir-faire artisanal
- Savourez nos différentes variétés de pains et baguettes
- Envie d'une Gourmandise ?
- Des plus traditionnelles aux plus originales
- Au petit-déjeuner comme au goûter
- Découvrez nos compositions tout en gourmandise
- Le secret de notre gourmandise !

Bas de page:

- Copyright 2022 © All rights Reserved.
- Mentions légales
- Retrouvez nous sur les réseaux sociaux: Facebook, Instagram
- Maison PARADIS 1 Place Saint Louis 57000 Metz (+33) 6 21 45 20

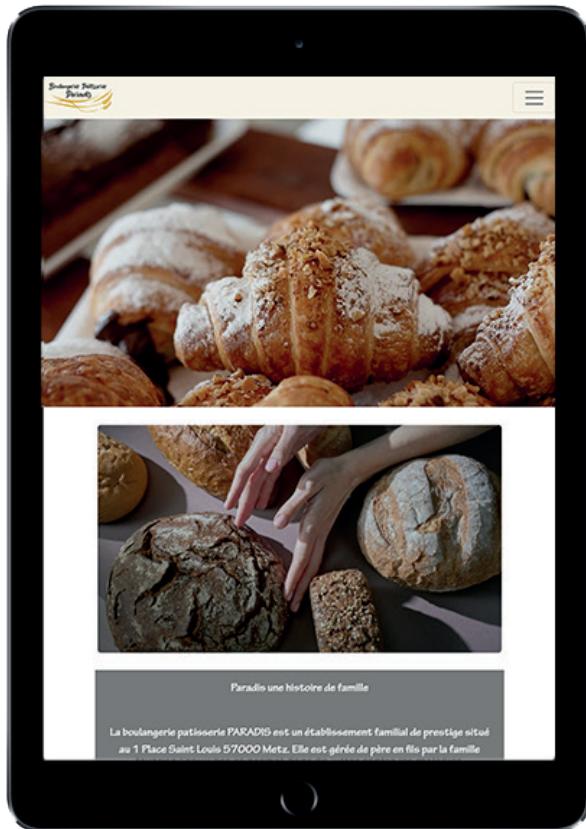
Maquette du site avec le maillage interne

— Maquettage réalisé sur Adobe XD —



Extrait de la page d'accueil du site responsive (tablette et mobile)

(min-width:480px) and (max-width:1023px)



(max-width:479px)



Spécifications fonctionnelles du projet

Le site doit permettre de présenter les différents produits de la boulangerie Patisseries classés et affichés par catégorie

Le client m'a demandé de mettre en place un site vitrine sur lequel il pourra présenter ces différents produits, le site contiendra aussi une page avec un formulaire de contact permettant d'envoyer des messages de demande d'informations et une barre de recherche sur toutes les pages du site.

Le client souhaite aussi avoir une interface admin pour pouvoir gérer les différents produits, catégories, et taxes, il pourra ajouter, modifier, supprimer si nécessaire les produits, les catégories et les taxes, cette partie doit être protégée par un accès admin (identifiant et mot de passe nécessaire).

Spécifications techniques du projet

Utilisation du langage de balisage HTML5, CSS3, TWIG pour la personnalisation des différentes pages du site.

Utilisation de Bootstrap, une bibliothèque CSS afin d'élaborer la mise en forme visuelle globale du site.

Utilisation de Symfony6 Framework PHP pour le développement des composants qui feront la liaison avec la base de données et interpréteront les différentes données.

Symfony est un framework basé sur le modèle MVC (Model View Controller). Pour faire simple, le modèle MVC va nous aider à séparer les requêtes de la base de données (Model), de la logique relative au traitement des demandes (Controller) et au rendu de la présentation (View).

Utilisation de GitHub : git permettra de conserver une partie de code source « propre » sans conflit et sans bug et de travailler sur des branches (copie de la branche principale) sans impacter sur le code déjà fonctionnel, il permet d'enregistrer son code et de le partager...
de plus lorsque l'on se retrouve sur des projets à plusieurs, git offre la possibilité de travailler chacun sur la partie qui nous incombe sans rentrer en conflit avec le reste des développeurs et de leur travail.



Comme tout bon framework, Symfony prend les questions de sécurité très au sérieux. Il intègre des mesures préventives contre les attaques les plus fréquentes (XSS, injections SQL).

Tous ces mécanismes sont mis en place de façon systématique sans intervention de l'équipe de développement, c'est pour cela que j'ai choisi de développer mon projet avec ce framework si puissant.

Base de données

I. Dictionnaire des données épurées

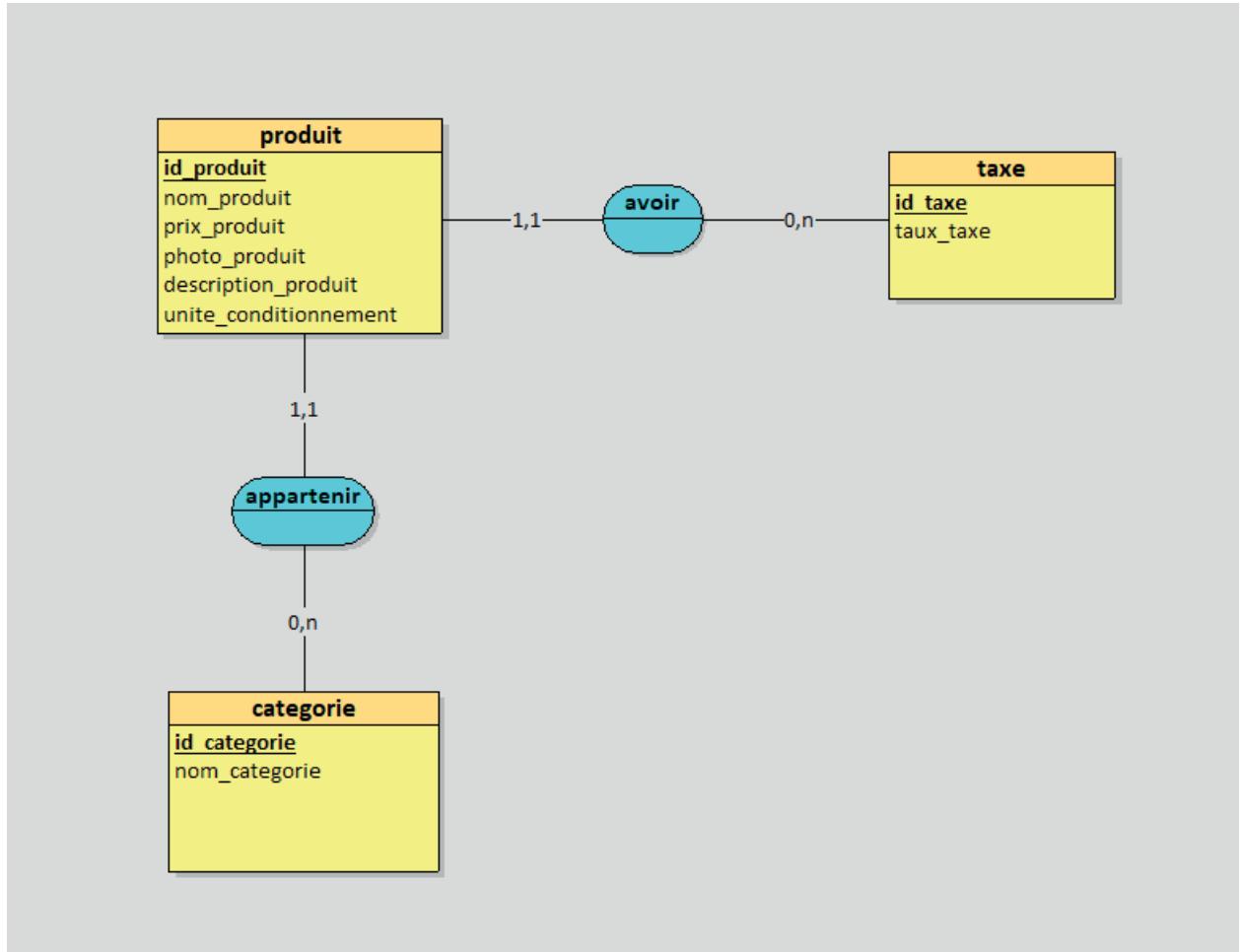
Avant toute création au niveau de ma base de données j'ai réalisé le dictionnaire des données épurées, celui-ci permet aux utilisateurs de découvrir et d'explorer tous les jeux des données disponibles, d'améliorer leur compréhension de ces données, de faciliter la collaboration avec les autres utilisateurs.

Dictionnaire de données épurées

Données				
Champs	Type	Longueur	Calculé	Elémentaire
Id_produit	Numérique			X
Nom_produit	Alphabétique	255		X
Prix_produit	Numérique			X
Unite_conditionnement	Alphabétique	255		X
Description_produit	Alphabétique	255		X
Photo_produit	Alphabétique	255		X
Id_categorie	Numérique			X
Nom_categorie	Alphabétique	255		X
Id_taxe	Numérique			X
Taux_taxe	Numérique			X
Id_admin	Numérique			X
Email	Alphabétique	180		X
password	Alphabétique	255		X

II. Conception du MCD

J'ai utilisé pour réaliser le MCD, l'application Looping qui est un logiciel de modélisation conceptuelle de données gratuit et libre d'utilisation.



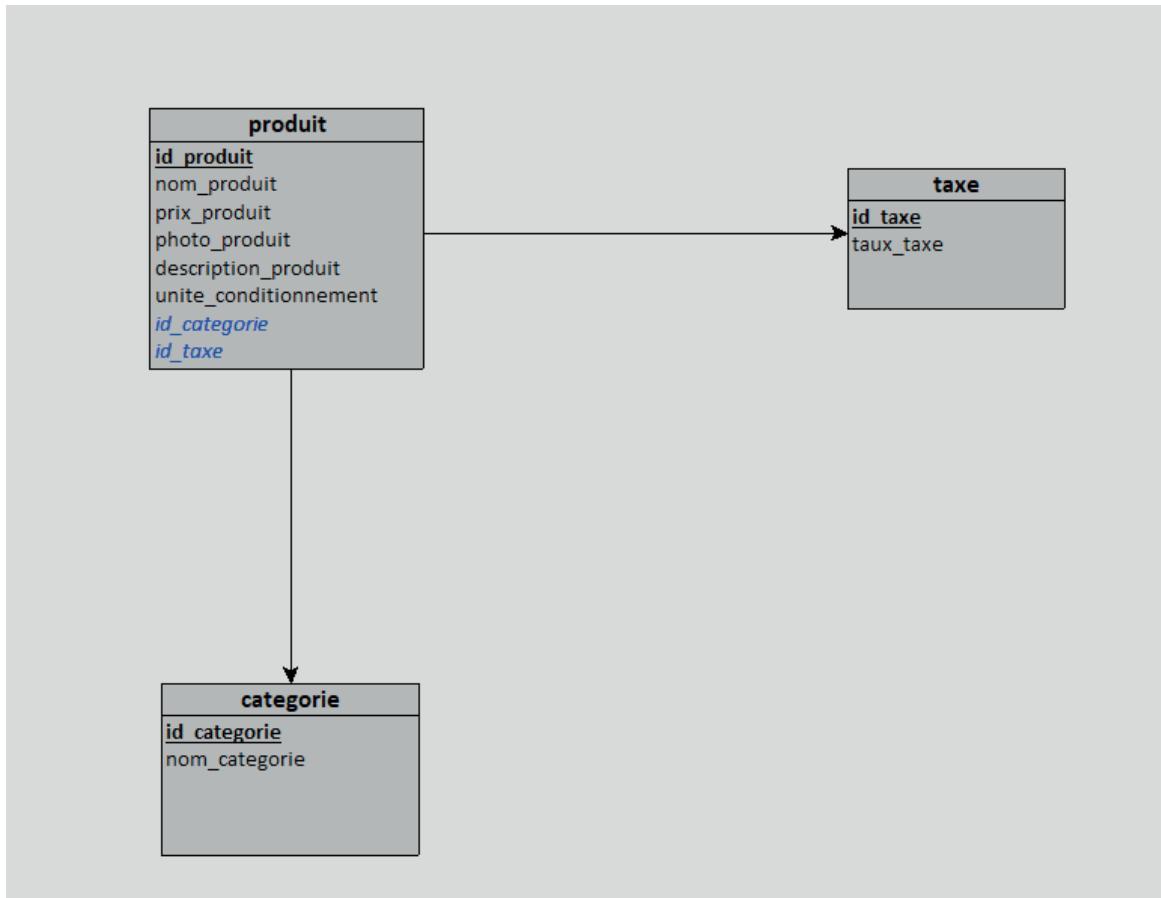
III. MRD (Modèle relationnel de données)

Categorie = (id_categorie , nom_categorie);

Taxe = (id_taxe , taxe);

Produit = (id_produit , nom_produit , prix_produit , Photo_produit , description_produit ,
unite_conditionnement, #id_categorie, #id_taxe);

IV. MPD (Model physique de données)



Règles de passage du MCD au MRD

Chaque entité du MCD devient une relation puis une table dans la base de données.

La clé de l'entité la plus forte va s'immigrer dans l'entité la plus faible.

Dans un SGBDR (système de gestion de base de données) de type relationnel:

une table est de structure tabulaire dont chaque ligne correspond aux données d'un objet enregistré.
Chaque colonne correspond à une propriété de cet objet (ça concerne uniquement la programmation orienté objet)

Une table contiendra donc un ensemble d'enregistrements.

Une ligne correspond à un enregistrement.

Une colonne correspond à un champ.

b) Son identifiant devient la clé primaire de la table.(au moment de la création de la table)

La clé primaire permet d'identifier de façon unique un enregistrement dans la table.

Les valeurs de la clé primaire sont donc uniques.

Les valeurs de la clé primaire sont obligatoirement non nulles.

V. Connexion à la base de données

La connexion à la base de données se fait grâce à la configuration de l'url de base dans le fichier de configuration «.env», les informations de connexion à la base de donnée sont stockées dans une variable d'environnement appelée DATABASE_URL

```
DATABASE_URL="mysql://db_user:db_password@127.0.0.1:3306/db_name?serverVersion=5.7"
```

Une fois les paramètres de connexion sont configurés Doctrine peut créer la base de donnée grâce à la commande :

```
php bin/console doctrine:database:create
```

Durant la période de développement les paramètres d'accès à la base de données sur le serveur localhost sont les suivants :

Utilisateur : 'root'

Mot de passe: “

```
DATABASE_URL="mysql://root@127.0.0.1:3306/BoulangerieParadis?serverVersion=5.7&charset=utf8mb4"
```

Lors du passage en production l'identifiant et le mot de passe devront être modifié pour correspondre aux informations du serveur distant.

Doctrine est l'ORM par défaut du framework Symfony, elle abrite plusieurs bibliothèques PHP principalement axées sur le stockage de bases de données et le mappage d'objets.

ORM fournit la persistance transparente des objets PHP. C'est l'interface qui permet de faire le lien ou «mapping» entre les objets et les éléments de la base de données (que gère DBAL).

Doctrine «ORM» (Object Relational Mapping) se base sur Doctrine «DBAL» (Couche d'abstraction de base de données) qui également s'appuie sur l'objet PDO (PHP Data Objects). Il est à noter que PDO existe par défaut depuis la version PHP 5.1 et ne fait pas partie de Doctrine.

PDO est une interface objet permettant d'accéder à une base de données en PHP et qui élève le niveau de sécurité (via par exemple les requêtes préparées)

Un petit schéma qui montre l'intérêt de doctrine

On peut enregistrer des objets et en récupérer en interrogeant la base de données via à doctrine .



Réalisations

I. Extraits de code front-end

Pour la réalisation, j'ai commencé par construire le squelette de mon site et pour cela je l'ai décomposé en différent composants :

Le header

La réalisation de l'aspect visuel de ce composant est gérée grâce à la bibliothèque CSS /Bootstrap.



Le header est composé du logo et de la Nav qui contient des listes déroulantes permettant de naviguer entre les différentes pages du site ...

Lorsque l'admin est connecté des options en plus apparaissent dans le header
(affichage admin connecté page 28).

Aperçu du code:

```
<header class="row">
  <nav class="navbar navbar-expand-lg navbar-light fixed-top g-0">
    <a href="{{ path('accueil') }}"></a>
    <button class="navbar-toggler" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      {% if is_granted('ROLE_ADMIN') %}
        # la liste déroulante qui va nous amener aux pages du crud sera affichée uniquement pour l'administrateur du site #
        <ul class="navbar-nav me-auto my-2 my-lg-0 navbar-nav-scroll" style="--bs-scroll-height: 1000px;">
          <li class="nav-item dropdown">
            <a class="nav-link active" href="#" id="navbarDropdownScrollingDropdown">
              <i class="bi bi-house"></i>
            </a>
            <ul class="dropdown-menu" aria-labelledby="navbarDropdownScrollingDropdown">
              <li><a class="dropdown-item" href="{{ path('produit_index') }}>Gestion des produits</a></li>
              <li><a class="dropdown-item" href="{{ path('categorie_index') }}>Gestion des catégories</a></li>
              <li><a class="dropdown-item" href="{{ path('taxe_index') }}>Gestion des taxes</a></li>
            </ul>
          </li>
        </ul>
      {% endif %}
      # histoire de la maison paradis #
      <ul class="navbar-nav me-auto my-2 my-lg-0 navbar-nav-scroll" style="--bs-scroll-height: 1000px;">
        <li class="nav-item dropdown">
          <a class="nav-link active" href="{{ path('accueil') }}" id="navbarDropdownScrollingDropdown2">
            Notre histoire
          </a>
        </li>
      </ul>
      # la liste déroulante de nos produits far Boulangerie signatures et classiques #
      <ul class="navbar-nav me-auto my-2 my-lg-0 navbar-nav-scroll" style="--bs-scroll-height: 1000px;">
        <li class="nav-item dropdown">
          <a class="nav-link active" href="#" id="navbarDropdownScrollingDropdown3">
            Nos produits
          </a>
          <ul class="dropdown-menu" aria-labelledby="navbarDropdownScrollingDropdown3">
            <li><a class="dropdown-item" href="{{ path('boulangerie') }}>Boulangerie</a></li>
            <li><a class="dropdown-item" href="{{ path('signatures') }}>Les signatures</a></li>
            <li><a class="dropdown-item" href="{{ path('classiques') }}>Les grands classiques</a></li>
          </ul>
        </li>
      </ul>
      # la liste déroulante de nos douceurs #
      <ul class="navbar-nav me-auto my-2 my-lg-0 navbar-nav-scroll" style="--bs-scroll-height: 1000px;">
        <li class="nav-item dropdown">
          <a class="nav-link active" href="#" id="navbarDropdownScrollingDropdown4">
            Nos douceurs
          </a>
          <ul class="dropdown-menu" aria-labelledby="navbarDropdownScrollingDropdown4">
            <li><a class="dropdown-item" href="{{ path('macarons') }}>Nos macarons</a></li>
            <li><a class="dropdown-item" href="{{ path('chocolats') }}>Nos Chocolats</a></li>
            <li><a class="dropdown-item" href="{{ path('viennoiseries') }}>Nos viennoiseries</a></li>
          </ul>
        </li>
      </ul>
      # la liste déroulante du traiteur #
      <ul class="navbar-nav me-auto my-2 my-lg-0 navbar-nav-scroll" style="--bs-scroll-height: 1000px;">
        <li class="nav-item dropdown">
          <a class="nav-link active" href="#" id="navbarDropdownScrollingDropdown5" role="button" data-bs-toggle="dropdown" aria-expanded="false">
            Traiteur
          </a>
          <ul class="dropdown-menu" aria-labelledby="navbarDropdownScrollingDropdown5">
            <li><a class="dropdown-item" href="{{ path('sucree') }}>Réception sucrée</a></li>
            <li><a class="dropdown-item" href="{{ path('sallee') }}>Réception salée</a></li>
          </ul>
        </li>
      </ul>
      <ul class="navbar-nav me-auto my-2 my-lg-0">
        <li class="nav-item">
          <a class="nav-link active" href="#" id="navbarDropdownScrollingDropdown6" href="{{ path('mailer') }}>Contact</a>
        </li>
      </ul>
      {% if is_granted('ROLE_ADMIN') %}
        # bouton de déconnexion qui va être affiché pour l'admin quand il est authentifié #
        <ul class="navbar-nav me-auto my-2 my-lg-0">
          <li class="nav-item">
            <button id="btnDeconnexion" href="{{ path('app_logout') }}>Déconnexion</button>
          </li>
        </ul>
      {% endif %}
      # la barre de recherche #
      <form id="formSearch" action="{{ path('search') }}" class="d-flex" method="get">
        <input id="recherche" name="search" class="form-control me-2" type="text" placeholder="Chercher un produit" aria-label="Search">
        <button id="btnRechercher" type="submit"><i class="bi bi-search fs-5"></i></button>
      </form>
    </div>
  </nav>
</header>
```

Arguments et commentaires

Afin de réaliser un site web fluide j'ai réalisé l'interface front-end grâce au framework bootstrap.
Il a été conçu pour aider les développeurs dans leurs travaux quotidiens sur les langages qui sont le HTML et le CSS.
Il fonctionne notamment sur un système de grid avec un maximum de 12 colonnes par ligne (row).
De plus grâce à une bonne utilisation du framework il est presque possible de se passer de CSS mais également des media queries pour gérer le responsive du site.
Cependant j'ai quand même utilisé du CSS afin d'embellir mon site mais également de « casser » certains code bootstrap car à un certain moment son utilisation reste parfois limitée

```
<body class="container-fluid">
|   <header class="row ">
```

Ici on peut voir que j'ai utilisé la class « container-fluid » le container-fluid va englober tout le contenu présent dans mon body et l'adapter automatiquement à la taille de l'écran, cependant son contenu lui ne sera pas totalement responsive automatiquement. Ce sera au développeur de l'adapter lui-même grâce au système de grid, j'ai choisi de le faire avec les media queries pour ce projet ...

Le terme « fluid » va lui indiquer de prendre toute la largeur de la page sans laisser de bordure contrairement à un « container » simple.

Dans cet extrait de code le terme « row » signifie que tous les éléments contenus dans le header vont être placés sur une même ligne comme on peut le voir sur le résultat du code.

Le footer



Aperçu du code:

```
<footer class="container-fluid p-0">
<div class="row" id="foot">
<div id="footerA" class="col-4 my-5 text-center d-flex flex-column">
<a href="#">Copyright 2022 ©</a>
<a href="{{ path('mentions') }}>Mentions légales</a>
<a href="#">CGU</a>
</div>

<div id="footerB" class="col-4 my-5 text-center d-flex flex-column">
<a href="#">All rights Reserved.</a>
<a href="#">Retrouvez nous sur les réseaux sociaux</a>
{# les réseaux sociaux #}
<div>
<a href="https://www.facebook.com/"><i class="bi bi-facebook"></i></a>
<a href="https://www.instagram.com/boulangerieparadis"><i class="bi bi-instagram text-danger"></i></a>
</div>
{# le lien vers la page de contact #}
<div id="footerC" class="col-4 my-5 text-center d-flex flex-column">
<a href="{{ path('mailer') }}>Maison PARADIS </a>
<a href="{{ path('mailer') }}> 1 Place Saint Louis 57000 Metz </a>
<a href="#"> 00336214520 </a>
</div>
</div>
</div>
</div>
```

Arguments et commentaires :

J'ai positionné les div qui sont dans le footer grâce aux colonnes (col-4) et j'ai centré le texte grâce à la class bootstrap : "text-center".

si j'avais pas utilisé bootstrap j'aurais appliqué un display: flex sur le container pour aligner les div, un align-items: center pour centrer le contenu verticalement, et justify-content: space-around pour répartir équitablement les div et j'aurais pu mettre tout ça dans une class bootstrap comme ceci class="d-flex align-items-center justify-content-around".

Etant donné que ces composants seront utilisés à plusieurs reprises sur notre site on les stocke dans le fichier base.html.twig qui se trouve dans le dossier templates.

Ce fichier définit les éléments communs de tous les modèles d'application, tels que :

<head>, <header>, <footer> ...

Prenons comme exemple le template de la page d'accueil du site, «index.html.twig» hérite de base.html.twig il faut juste appeler le fichier base.html.twig dans le template de cette page d'accueil et mettre le titre de la page entre deux blocs :

{% block title %} Accueil {% endblock %}

et le contenu entre deux blocs aussi :

{% block body %} Contenu {% endblock %}

Aperçu du code

```
<div id="monCarousel" class="row">
  <div id="carouselExampleControls" class="carousel slide g-0" data-bs-ride="carousel">
    <div class="carousel-inner">

      <div class="carousel-item">
        
      </div>

      <div class="carousel-item">
        
      </div>

      <div class="carousel-item active">
        
      </div>

      <div class="carousel-item">
        
      </div>

    </div>
  </div>
</div>
```

Le carrousel a été pris de la bibliothèque bootstrap je l'ai complété avec les images d'illustration de la boulangerie .

Pour définir un carrousel avec Bootstrap, il suffit d'ajouter une classe `.carousel` à un conteneur générique. On ajoutera ensuite également une classe `.slide` pour donner un effet de transition de type slide entre nos différentes diapositives.

Pour ajouter des diapositives à notre carrousel, nous allons avoir besoin d'une classe `.carousel-inner` qu'on va ajouter à un deuxième élément conteneur qui va englober toutes nos diapositives et d'une classe `.carousel-item` pour chacune des diapositives du carrousel.

Aperçu du code du main de la page d'accueil

```
<main class="row">
  <!-- le premier article histoire de la maison paradis-->
  <div class="row accueil">
    <div class="col-6 g-0" id="div1">
      
      <figcaption class="overlay px-5 text-center">
        Savourez nos différentes variétés Pains et Baguettes
      </figcaption>
    </div>
    <div class="col-6 g-0" id="text1">
      <h2 class="text-center">Paradis une histoire de famille</h2>
      <p class="text-center">
        La boulangerie patisserie PARADIS est un établissement familial de prestige situé au 1 Place Saint Louis 57000 Metz.
        Elle est gérée de père en fils par la famille Paradis 1956.
        la maison paradis a su se développer en restant fidèle au valeurs familiales à l'origine de cette maison.
      </p>
    </div>
  </div>

  <!--la deuxième partie la sources des produits de la maison paradis-->
  <div class="row accueil">
    <div class="col-6 g-0" id="text2">
      <h2 class="text-center">Un amour de bons produits</h2>
      <p class="text-center">Nos matières premières sont soigneusement sélectionnées auprès des meilleurs fournisseurs, pour fabriquer dans notre boutique comme dans notre atelier des produits de qualité.
        On est passionnés par notre métier on est à cœur de vous offrir le meilleur de nos produits.
      </p>
    </div>
    <div class="col-6 g-0" id="div2">
      
      <figcaption class="overlay px-5 text-center">
        Un savoir-faire artisanal
      </figcaption>
    </div>
  </div>

<div id="cartes" class="row">
  <!--carte boulangerie-->
  <div id="one" class="card col-3 g-0">
    
    <a href="#">
    <figcaption class="overlay px-5 text-center">
      Savourez nos différentes variétés de pains et baguettes, toutes réalisées à base de farines labellisées
    </figcaption>
  </div>
  <!--carte signatures-->
  <div id="two" class="card col-3 g-0">
    
    <a href="#">
    <figcaption class="overlay px-5 text-center">
      Envie d'une touche sucrée ?  
Faites-vous plaisir avec les pâtisseries généreuses de chez PARADIS
    </figcaption>
  </div>
  <!--carte grande classique-->
  <div id="three" class="card col-3 g-0">
    
    <a href="#">
    <figcaption class="overlay px-5 text-center">
      Des plus traditionnelles aux plus originales, nos pâtisseries raviront vos papilles de multiples saveurs.
    </figcaption>
  </div>
  <!--carte boulangerie-->
  <div id="four" class="card col-3 g-0">
    
    <a href="#">
    <figcaption class="overlay px-5 text-center">
      Au petit-déjeuner comme au goûter, la texture fondante de nos macarons ravira les gourmands.
    </figcaption>
  </div>
  <!--carte signatures-->
  <div id="five" class="card col-3 g-0">
    
    <a href="#">
    <figcaption class="overlay px-5 text-center">
      Découvrez nos compositions tout en gourmandise pour un véritable plaisir praliné et raffiné.
    </figcaption>
  </div>
  <!--carte grande classique-->
  <div id="six" class="card col-3 g-0">
    
    <a href="#">
    <figcaption class="overlay px-5 text-center">
      Le secret de notre viennoiserie ?  
Matières premières de qualité exceptionnelle...
    </figcaption>
  </div>
</div>

.card{
  border: 1px solid lightgray;
  padding: 3px;
  box-shadow: 2px 2px 2px lightgray;
  position: relative;
  /*position du parent pour que l'enfant reste dans son parent*/
}

img.card-img-top {
  width: 100%;
  height: 100%;
  /*remplir toute la div pour les images qui n'ont pas la même taille*/
  object-fit: cover; /*permet d'éviter la déformation des photos trop petite lors d'un height 100%*/
}

.card .overlay{
  /*on place le texte au dessus de la photo*/
  position: absolute;
  top: 0;
  left: 0;
  background-color: rgba(0, 0, 0, 0.74); /*couleur du filtre*/
  color: white; /*couleur du texte*/
  font-size: 1rem;
  /*on remplit tout l'espace de son parent*/
  width: 100%;
  height: 100%;
  /*on centre le texte*/
  display: flex;
  justify-content: center;
  align-items: center;
  transform: scale(0) rotate(0deg); /*echelle à 0 ce qui permet de reduire la class overlay*/
  transition: 1s;
  transform-origin: top left; /*reglage en % possible voir la doc */
}
```

Pour la partie histoire de la maison paradis tout a été positionné grâce à bootstrap j'ai juste ajusté avec le style personnel.

En cliquant sur une carte l'utilisateur sera redirigé vers la page de la catégorie choisie ...

Pour la première row dont l'identifiant est #cartes

j'ai appliqué du style personnel pour répartir les cards

```
display: flex;
justify-content: space-around;
background-color: #fffff69;
```

et pour la deuxième row dont l'identifiant est #cardss

J'ai utilisé une class bootstrap .

```
class="row d-flex justify-content-around"
```

Juste pour montrer l'intérêt d'utiliser bootstrap pour optimiser les lignes de code

Overlays pour les cartes (card)

pour animer et illustrer le site

J'ai donné aux images des cards une largeur et une hauteur de 100% avec un Object-fit : Cover qui permet d'éviter la déformation des photos ...

Une position relative au parent (.card) et une position absolute pour les enfants figcaptions (.overlay)

la figcaption va être fixé sur la card

J'ai appliqué un display flex et un align-items et un justify-content : center pour centrer le texte

II. Extraits de code Back-end

Pour la réalisation du back-end, j'ai commencé par l'installation de Symfony (détails en annexes p 36 et 37) Symfony est un framework basé sur le modèle MVC (Model View Controller). Pour faire simple, le modèle MVC va nous aider à séparer les requêtes de la base de données (Model) de la logique relative au traitement des demandes (Controller) et au rendu de la présentation (View).

ça se présente comme ceci sur symfony

Dossier SRC :

C'est le cœur du projet ! C'est le dossier qui contient:

Controller : Ce dossier contient les contrôleurs qui se chargent de rediriger vers les Manager / Service / Repository, aucun accès à la base de données ne doit se faire depuis un Controller.

Entity (Model) : Dans ce dossier nous allons définir la structure de la base de donnée au travers les classes. Chaque Entity représente généralement une table en Base de donnée.

Repository : Un Repository est toujours rattaché à une Entity, il nous permet de créer nos fonctions qui iront requêter la table de notre Entity.

Templates(view) :

Les fichiers de template Twig ont comme format monfichier.html.twig et viennent rajouter quelques fonctionnalités au HTML classique :

`{{ ... }}` : appel à une variable ou une fonction PHP, ou un template Twig parent.

`{% ... %}` : commande, comme une affectation, une condition, une boucle ou un bloc HTML.

`{# ... #}` : commentaires.

Le twig est un moteur de templates ayant des fonctionnalités far comme le contrôle de flux complexe (routage et inclusion de page), échappement automatique des caractères spéciaux, héritage des thèmes (comme les thèmes bootstrap), filtrage des variables, internationalisation(traduction du site facilité avec des systèmes de key => value)

L'utilisateur envoie une requête à l'application via le navigateur

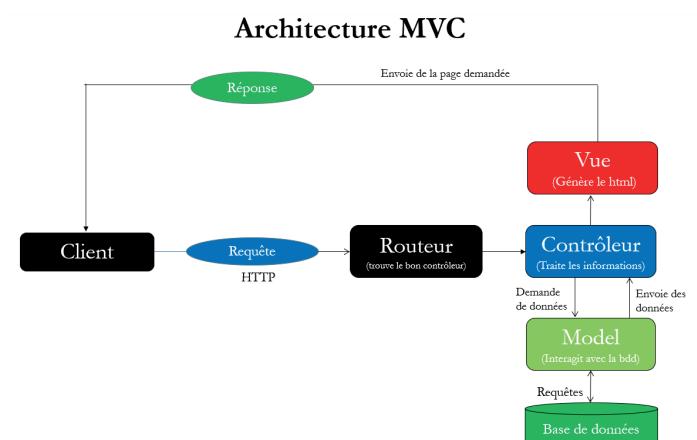
Le navigateur enverra une requête au serveur Web.

Le routeur transmet la demande au contrôleur cible là où elle se déroule toute la logique du site.

Le contrôleur interagira avec le Model, qui à son tour interagit avec Datasource via Doctrine ORM.

Une fois le processus terminé, il génère la réponse (la vue) et la renvoie au serveur Web

Enfin, la réponse sera envoyée au navigateur demandé par le serveur Web.



Présentation du CRUD des produits

Entity > Produit.php (le model)

```
<?php

namespace App\Entity;

use App\Repository\ProduitRepository;
use Doctrine\ORM\Mapping as ORM;

#[ORM\Entity(repositoryClass: ProduitRepository::class)]
class Produit
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column(type: 'integer')]
    // les propriétés de l'objet produit
    private $id;

    #[ORM\Column(type: 'string', length: 255)]
    private $nom_produit;

    #[ORM\Column(type: 'string', length: 255)]
    private $unite_conditionnement;

    #[ORM\Column(type: 'float')]
    private $prix_produit;

    #[ORM\Column(type: 'string', length: 255)]
    private $description;

    #[ORM\Column(type: 'string')]
    private $photo;
    // chaque catégorie peut être associée à plusieurs produits.
    #[ORM\ManyToMany(targetEntity: Categorie::class, inverseBy: 'produits')]
    #[ORM\JoinColumn(nullable: false)]
    private $categorie;
    // chaque taxe peut être associée à plusieurs produits
    #[ORM\ManyToMany(targetEntity: Taxe::class, inverseBy: 'produits')]
    #[ORM\JoinColumn(nullable: false)]
    private $taxe;

    // on peut juste récupérer un id c'est pour cela qu'on a juste un getter et pas de set
    public function getId(): ?int
    {
        return $this->id;
    }
    // méthodes de l'objet produit qu'on pourra utiliser en dehors de l'objet
    public function getNomProduit(): ?string
    {
        return $this->nom_produit;
    }

    public function setNomProduit(string $nom_produit): self
    {
        $this->nom_produit = $nom_produit;
    }
}
```

Src > ProduitRepository (lié à notre model)

```
<?php

namespace App\Repository;

use App\Entity\Produit;
use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
use Doctrine\Persistence\ManagerRegistry;

/**
 * @method Produit|null find($id, $lockMode = null, $lockVersion = null)
 * @method Produit|null findOneBy(array $criteria, array $orderBy = null)
 * @method Produit[] findAll()
 * @method Produit[] findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)
 */
class ProduitRepository extends ServiceEntityRepository
{
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, Produit::class);
    }
}
```

Arguments et commentaires :

Chaque entité dispose d'un repository par défaut accessible dans tous les contrôleurs de nos applications.

Voici comment accéder à un Repository depuis un contrôleur.

`use App\Repository\ProduitRepository;`

Un Repository Doctrine est un objet dont la responsabilité est de récupérer une collection d'objets.

Les repositories ont accès à deux objets principalement :

`EntityManager` : Permet de manipuler nos entités ;

`QueryBuilder` : Un constructeur de requêtes qui permet de créer des requêtes personnalisé.

ProduitController (l'affichage des produits)

```
namespace App\Controller;

use App\Entity\Produit;
use App\Form\ProduitType;
use App\Repository\ProduitRepository;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\File\Exception\FileException;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\String\Slugger\SluggerInterface;

#[Route('/produit')]
class ProduitController extends AbstractController
{
    // la récupération de la liste des produits (Read)
    #[Route('/', name: 'produit_index', methods: ['GET'])]
    public function index(ProduitRepository $produitRepository): Response
    {
        return $this->render('produit/index.html.twig', [
            'produits' => $produitRepository->findAll(),
        ]);
    }
}
```

Use `Symfony\Component\HttpFoundation\Request`;

La classe `Request` est une représentation orientée objet du message de requête HTTP.

Avec lui, on aura toutes les informations de la demande à portée de main...

Use `Symfony\Component\HttpFoundation\Response`;

Symfony fournit également une classe `Response` : une représentation PHP d'un message de réponse HTTP. Cela permettra à notre application d'utiliser une interface orientée objet pour construire la réponse qui doit être renvoyée au client...

ProduitController hérite de `AbstractController` qui sert à gerer automatiquement les routes, le `render()` ... La méthode `findAll()` dans le `produitRepository` c'est une méthode par défaut qui récupère de la base de données toutes les données du produit.
on doit injecter `ProduitRepository` dans notre fonction `index()` pour pouvoir interroger la BDD.

ProduitController (New)

```
// l'accès est seulement pour l'admin
#[IsGranted('ROLE_ADMIN')]
#[Route('/new', name: 'produit_new', methods: ['GET', 'POST'])]
public function new(Request $request, EntityManagerInterface $entityManager, SluggerInterface $slugger): Response
{
    // renvoie un message d'erreur denyAccess au cas où un utilisateur lambda tente d'accéder à l'espace de l'admin
    $this->denyAccessUnlessGranted('ROLE_ADMIN');

    $produit = new Produit();
    $form = $this->createForm(ProduitType::class, $produit);
    // le formulaire va d'abord gérer la requête pour pouvoir la traiter
    $form->handleRequest($request);
    // On vérifie si le formulaire est soumis avec des données valides
    if ($form->isSubmitted() && $form->isValid()) {
        // on va traiter la photo avant l'envoie à la base de donnée
        $photo = $form->get('photo')->getData();
        if ($photo) {
            // on récupère la photo originale et on la nettoie pour éviter toute injection extérieure notamment les balises html
            $originalFilename = $photo->getClientOriginalName(), PATHINFO_FILENAME);
            // on récupère le nom de la photo originale slugger ça sert à nettoyer mes données
            // ça remplace le strip tags dans le php natif
            $safeFilename = $slugger->slug($originalFilename);
            $newFilename = $safeFilename.'-'.$uniqid().'.'.$photo->guessExtension();

            try {
                // images directory configuré dans service .yaml
                $photo->move(
                    $this->getParameter('images_directory'),
                    $newFilename
                );
            } catch (FileException $e){
                // veuillez insérer une photo
            }
            $produit->setPhoto($newFilename);
        }
        // la préparation(persist) et l'exécution(flush) de la requête
        $entityManager->persist($produit);
        $entityManager->flush();
    }

    return $this->redirectToRoute('produit_index', [], Response::HTTP_SEE_OTHER);
}

return $this->renderForm('produit/new.html.twig', [
    'produit' => $produit,
    'form' => $form,
]);
}
```

Form > ProduitType

```
use App\Entity\Produit;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\Extension\Core\Type\FileType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;
use Symfony\Component\Validator\Constraints\File;
class ProduitType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options): void
    {
        $builder
            ->add('nom_produit')
            ->add('unite_conditionnement')
            ->add('prix_produit')
            ->add('description')
            ->add('categorie')
            ->add('taxe')
            ->add('photo', FileType::class, [
                'label' => 'Insérer une photo',
                'mapped' => false,
                'required' => true,
                'constraints' => [
                    new File([
                        'maxSize' => '4000k',
                        'mimeTypes' => [
                            'image/jpeg',
                            'image/png'
                        ],
                        'mimeTypesMessage' => 'veuillez insérer un jpeg ou png uniquement'
                    ])
                ]
            ])
        ;
    }

    public function configureOptions(OptionsResolver $resolver): void
    {
        $resolver->setDefaults([
            'data_class' => Produit::class,
        ]);
    }
}
```

Le gestionnaire d'entités (EntityManagerInterface) est un objet Doctrine qui va nous permettre de gérer nos entités, nous allons l'utiliser pour ajouter, modifier ou supprimer des lignes d'enregistrement en base de données ...

c'est l'objet le plus important de Doctrine car il est responsable de l'enregistrement et de la récupération des données dans la base de données

On instancie l'objet 'produit', on crée le formulaire d'ajout de produit qui est dans ProduitType qui est relié à l'entité produit.

Quand on appuie sur le bouton valider, la méthode handleRequest() va se charger de récupérer les données et de les réinjecter dans l'objet form pour pouvoir le traiter...

On vérifie si le formulaire est soumis avec des données valides (les données soumises seront écrites dans le formulaire) et s'il est valide (pour pouvoir effectuer le stockage en base de données), on fait le traitement de la photo,

on récupère la photo originale et on la nettoie pour éviter toute injection extérieure notamment les balises html.

Slugger ça sert à nettoyer les données, ça remplace le strip tags dans le php natif...

une fois les données de la photo sont nettoyées, on modifie le nom de la photo et on l'enregistre, enfin on prépare et on exécute la requête.

le formulaire fait une première vérification, il compare le type de données avec le type en BDD et si les données sont pas bonnes il empêche la soumission du formulaire, ça évitera les injections de balises html...

persist() méthode utilisée par symfony pour générer et préparer les requêtes

flush() dès que cette méthode est appelée Doctrine parcourt tous les objets qu'elle gère pour voir s'ils doivent être conservés dans la base de données dans notre cas les produits n'existent pas encore dans la base de données donc le gestionnaire d'entités exécute une requête INSERT.

ProduitController (Update)

```
//l'accès est seulement pour l'admin (sécurité)
#[IsGranted('ROLE_ADMIN')]
#[Route('/{id}/edit', name: 'produit_edit', methods: ['GET', 'POST'])]
public function edit(Request $request, Produit $produit, EntityManagerInterface $entityManager, SluggerInterface $slugger): Response
{
    //renvoie un message d'erreur denyAccess au cas un utilisateur lambda tente de l'espace de l'admin(sécurité en plus)
    $this->denyAccessUnlessGranted('ROLE_ADMIN');
    $form = $this->createForm(ProduitType::class, $produit);
    // le formulaire va d'abord gérer la requête pour pouvoir la traiter
    $form->handleRequest($request);
    // On vérifie si le formulaire est soumis avec des données valides
    if ($form->isSubmitted() && $form->isValid()) {
        $photo = $form->get('photo')->getData();
        if($photo){
            // on récupère la photo originale et on la nettoie pour éviter toute injection extérieure notamment les balises html
            $originalFilename = pathinfo($photo->getClientOriginalName(), PATHINFO_FILENAME);
            // on récupère le nom de la photo original slugger ça sert à nettoyer mes données
            $safeFilename = $slugger->slug($originalFilename);
            $newFilename = $safeFilename.'-'.$uniqid().'.'.$photo->guessExtension();

            try {
                $photo->move(
                    $this->getParameter('images_directory'),
                    $newFilename
                );
            } catch (FileException $e){
                // veuillez insérer une photo
            }
            $produit->setPhoto($newFilename);
        }
        // on a pas besoin de $entityManager->persist($produit); avec le flush on passe l'intégralité des données du produit dans la base
        // il sait que c'est une mise à jour
        $entityManager->flush();
        return $this->redirectToRoute('produit_index', [], Response::HTTP_SEE_OTHER);
    }
    return $this->renderForm('produit/edit.html.twig', [
        'produit' => $produit,
        'form' => $form,
    ]);
}
```

On récupère l'Id du produit grâce à la méthode GET, le formulaire créé se remplit automatiquement avec les propriétés du produit en question et on utilise la méthode handleRequest() pour que le formulaire puisse fonctionner quand on le soumet, c'est sensiblement la même chose avec le new.html.twig

Juste à la fin du traitement on utilisera seulement la méthode flush() pour l'exécution de la requête sans la méthode persist() car symfony sait que c'est une mise à jour du coup il enregistre automatiquement l'intégralité des données dans la base de données.

ProduitController (Delete)

```
//l'accès est seulement pour l'admin (sécurité)
#[IsGranted('ROLE_ADMIN')]
#[Route('/{id}', name: 'produit_delete', methods: ['POST'])]
public function delete(Request $request, Produit $produit, EntityManagerInterface $entityManager): Response
{
    $this->denyAccessUnlessGranted('ROLE_ADMIN');
    // csrf : le lancement de lien sans notre volanté ne marchera pas grâce à csrf
    // _token : vérifie que ça vient bien de notre part et ça sécurise automatiquement
    if ($this->isCsrfTokenValid('delete', $produit->getId(), $request->request->get('_token'))) {
        //remove() méthode qui informe à Doctrine que l'on veut supprimer un produit
        $entityManager->remove($produit);
        $entityManager->flush();
    }
    return $this->redirectToRoute('produit_index', [], Response::HTTP_SEE_OTHER);
}
```

On appelle la fonction remove() méthode qui informe à Doctrine que vous souhaitez supprimer une ligne d'enregistrement de la base de données.

La requête n'est pas réellement exécutée tant que la flush() méthode n'est pas appelée.
la méthode CsrfTokenValid() vérifie bien que le lien de la suppression des données est lancé par l'administrateur du site .

Template des produits ajoutés «index.html.twig» (view)

```

{# block body #}
{# notre vue qui va afficher les produits ajoutés qu'on a stocké dans la variable produits #}
<h1 class="product g-0 mt-5">Liste de produits ajoutés</h1>



| Id               | produit                  | Prix HT                         | Taxe               | prix TTC                                                                                               | Unité de conditionnement           | Description               | Catégorie               | photo                                                                                                                                                                | Action                                                                                                                                                              | Action |
|------------------|--------------------------|---------------------------------|--------------------|--------------------------------------------------------------------------------------------------------|------------------------------------|---------------------------|-------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|
| {{ produit.id }} | {{ produit.nomProduit }} | {{ produit.prixProduit }}€ / HT | {{ produit.taxe }} | {{ (produit.prixProduit + produit.prixProduit * produit.taxe.getTaxe / 100) round(2, 'ceil') }}€ / TTC | {{ produit.uniteConditionnement }} | {{ produit.description }} | {{ produit.categorie }} |  | <a class="m-2" href="{{ path('produit_show', { 'id': produit.id }) }}">Voir</a> <a class="m-2" href="{{ path('produit_edit', { 'id': produit.id }) }}">Modifier</a> |        |


```

Arguments et commentaires

Le fichier «index.html.twig» est notre vue qui va afficher les produits qu'on a stocké dans la variable \$produits dans le contrôleur, je l'ai donc modifié pour qu'il affiche les données extraites de la base de données de chaque produit grâce à la boucle {% for ... in ... %} (l'équivalent de foreach en php natif). Les données sont sous forme de tableau c'est pour ça que j'ai utilisé la syntaxe {{ produit.index }}. Au lieu d'écrire les URL des liens à la main, y'a la méthode path() une fonction qu'on utilise sur twig pour générer des URL en fonction du nom de la route donnée dans le contrôleur . #}

Extraits de quelques produits ajoutés

		id	categorie_id	taxe_id	nom_produit	unite_conditionnement	prix_produit	description	photo
<input type="checkbox"/>		333	4	7	Baguette classique	L'unité	1	Farine T65 Label Rouge. 250 grammes	baguetteB-6228d3cb532c4.jpg
<input type="checkbox"/>		334	4	7	Pain traditionnel	L'unité	1.5	Farine de blé Malté & Céréales. 250 grammes	pain-libree-6228d432e79a5.jpg
<input type="checkbox"/>		335	4	7	Pain Paradis	L'unité	1.5	Farine de Blé, de Seigle, céréales . 250 grammes	painA-6228d4b6b790d.jpg
<input type="checkbox"/>		336	4	7	Tradi-Graines	L'unité	1.8	Farine Complète T80 & graines de sésame	traditionnelle-6228d5007cb6f.jpg
<input type="checkbox"/>		337	4	7	Pain de mie	L'unité	1.2	Farine de Tradition,Lait, Œuf . 250 grammes	pain-de-mie-6228d5dea4385.jpg
<input type="checkbox"/>		338	6	8	L'Opéra	L'unité	2.6	l'Opéra au chocolat et au café	opera-6228da6ee6ceb.jpg
<input type="checkbox"/>		339	6	8	Eclairs	L'unité	2.6	Eclairs crèmeux au chocolat noir	eclairs-gratos-6228d86e95e38.jpg
<input type="checkbox"/>		340	6	8	Paris-Brest	L'unité	2.6	Crème pâtissière, praliné,amandes	parisbrest-6228d853a95e2.jpg
<input type="checkbox"/>		341	6	8	Saint honoré	L'unité	18.2	Couronnes de choux caramélisés	sainthonore-6228d9633e252.jpg
<input type="checkbox"/>		342	6	8	Mille-feuille	L'unité	10	Crème pâtissière à la vanille	millefeuille-6228d9d429616.jpg
<input type="checkbox"/>		343	6	8	Flan patissier	L'unité	2.6	Flan au bon goût de vanille	flanpatissier-6228da2bcfd91.jpg

Template de création de produits «new.html.twig»

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

{% block title %}Nouveau Produit{% endblock %}



# Créer un nouveau produit



{{ include('produit/_form.html.twig') }}



<a class="enregistrement text-black" href="{{ path('produit_index') }}">Retour à la liste</a>



{% endblock %}
```

Template de modification de produits «edit.html.twig»

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

{% block title %}Modification Produit{% endblock %}

{% block body %}



# Modifier le produit



{{ include('produit/_form.html.twig', {'button_label': 'Update'}) }}
{{ include('produit/_delete_form.html.twig') }}
<a class="enregistrement text-dark" href="{{ path('produit_index') }}">Retour à la liste des produit</a>

{% endblock %}
```

_form.html.twig

```
{{ form_start(form) }}
    {{ form_widget(form) }}
    <button class="enregistrement my-3">{{ button_label|default('Enregistrer') }}</button>
{{ form_end(form) }}
```

Arguments et commentaires :

La fonction `include()` prend en argument le chemin du template à inclure.

Le modèle inclus a accès à toutes les variables du modèle qui l'inclut.

On va inclure le `form.html.twig` le formulaire d'ajout de produit qui commence avec un `block`

`{% form_start(form)%}` et se termine par un `block` `{% form_end(form)%}`

à l'intérieur on trouve le `{% form_widget(form)%}` c'est les champs du formulaire (`input`) et le bouton Enregistrer pour pouvoir enregistrer les produit

Un aperçu des formulaires d'ajout, de modification et de suppression des produits

Créer un nouveau produit

Nom produit

Unité conditionnement

Prix produit

Description

Catégorie
 Boulangerie

Taxe
 5.5

Inserer une photo
 Aucun fichier choisi

Modifier le produit

Nom produit
 test5

Unité conditionnement
 la pièce

Prix produit
 5

Description
 produit pour tester le fonctionnement

Catégorie
 Macarons

Taxe
 20

Inserer une photo
 Aucun fichier choisi

Csrf_token : Cross site request forgery ça va nous permettre d'éviter la faille CSRF
 En fait, il s'agit d'effectuer une action visant un site ou une page précise en utilisant l'utilisateur comme déclencheur, sans qu'il en ait conscience ça peut compromettre la base de données



```
<form method="post" action="{{ path('produit_delete', { 'id': produit.id }) }}>
    {{ csrf_token('delete') ~ produit.id }}
    <input type="hidden" name="_token" value="{{ csrf_token('delete') ~ produit.id }}>
    <button class="enregistrement my-2 fs-4">Supprimer</button>
</form>
```

Id	347
Nom du produit	Tartelette aux pommes
unité de conditionnement	L'unité
Prix du produit	3€
Description	Pâte sablée croustillante
Catégorie	Les signatures

Avant de supprimer un produit on aura un message de confirmation inscrit sur l'attribut Onsubmit

127.0.0.1:8000 indique

Êtes-vous sûr de vouloir supprimer ce produit?

Un extrait de quelques produits ajoutés

produit	Prix HT	Taxe	prix TTC	Unité de conditionnement	Catégorie	photo	Action	Action
Baguette classique	1€/HT	5.5	1.06€/TTC	L'unité	Boulangerie		Voir	Modifier
Pain traditionnel	1.5€/HT	5.5	1.59€/TTC	L'unité	Boulangerie		Voir	Modifier
Pain Paradis	1.5€/HT	5.5	1.59€/TTC	L'unité	Boulangerie		Voir	Modifier
Tradi-Graines	1.8€/HT	5.5	1.9€/TTC	L'unité	Boulangerie		Voir	Modifier
Pain de mie	1.2€/HT	5.5	1.27€/TTC	L'unité	Boulangerie		Voir	Modifier

Vitrine > produit.html.twig

```
% extends 'base.html.twig'
% block title {{ titre }}% endblock %
% block body %
    <div class="row">
        <div class="col g-9">
            
        </div>
        <div class="col mt-3"> {{ titre }} </div>
        <p class="introduction"> {{ description }} </p>
    </div>
    <div id="lesProduits">
        {# pour chaque produit dans la liste des produits on récupère et on affiche les données suivantes #
        % for produit in produits %}
            <div id="leProduit" class="card mt-3">
                
                <div class="card-body text-center">
                    # le nom du produit définit dans le contrôleur #
                    <h2 class="card-title"> {{ produit.nomProduit }} </h2>
                    <p class="card-text text-black mt-3">
                        {{ produit.description }}<br>
                        {{ (produit.prixProduit + produit.prixProduit * produit.taxe.getTaxe / 100)|round(2, 'ceil') }} € TTC / {{ produit.uniteConditionnement }} </p>
                    <a href="{{ asset('/uploads/-produit photo') }}" class="btn mt-3 text-black" style="background-color: #f6efdf;">voir le produit</a>
                </div>
            </div>
        % endfor %
    </div>
% endblock %
```

Les produits sont classés selon leurs catégories spécifiques comme vous voyez sur la photo la catégorie est 'boulangerie'. toutes les pages des produits ont le même template «produit.html.twig» les variables {{ titre }} , {{ img}} , {{description}} ont été définies dans le VitrineController

{{ titre }} : titre de la page
{{ img }} : image de mise en avant de la page
{{ description }} : description de la page .

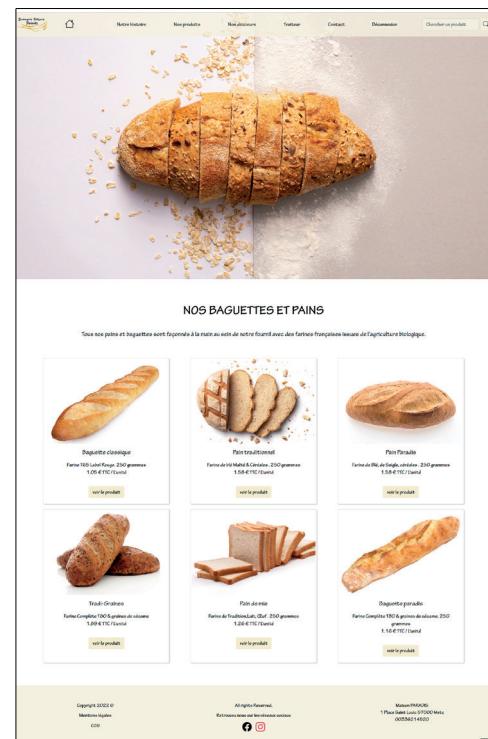
Pour chaque produit dans la liste des produits ajoutés on récupère les données définies dans le contrôleur (vitrineController) et on les affiche sur notre template .

VitrineController

```
#[Route('/boulangerie', name: 'boulangerie', methods: ['GET'])]
public function boulangerie(ProduitRepository $produitRepository): Response
{
    return $this->render('vitrine/produits.html.twig', [
        'titre' => 'NOS BAGUETTES ET PAINS',
        'img' => './images/boulangerie/pain19201088.jpg',
        'description' => 'Tous nos pains et baguettes sont façonnés à la main au sein de notre fournil avec des farines françaises issues de l'agriculture biologique.',
        'produits' => $produitRepository->findBy(['categorie' => 4])
    ]);
}
```

Les produits se mettent automatiquement dans la page concernée selon leurs catégories grâce à la fonction `findBy()` une méthode par défaut du repository qui renvoie un tableau de produits liés à la catégorie dont l'identifiant est égal à 4 pour cet exemple. il faut maintenant les envoyer à la vue.

La méthode `render()` génère le rendu du template et lui passe un tableau de variables ...



Un point sur la sécurité de la partie administrateur

Même si les internautes ne pourront pas créer leur propre compte sur le site, nous allons créer un système d'authentification entièrement fonctionnel pour l'admin. Nous n'aurons donc qu'un seul utilisateur, l'admin du site.

La première étape consiste à définir une entité User grâce à la cmd `$ symfony console make:user Admin`. Le login sera admin et nous devons générer le cryptage du mot de passe grâce à la commande suivante

`symfony console security:hash-password` On récupère le mot de passe hashé qu'on a saisi et on exécute la requête.

```
INSERT INTO admin (id, email, roles, password) VALUES (NULL, 'paradisboulangerie@gmail.com', ['ROLE_ADMIN'], '$2y$13$C/p82PLUSD6yuNYx3Z1Qc.DhXRAAm1iHejCYfA1eM70CEqkBoK3Xe')
```

J'ai donc réalisé une interface admin seulement pour l'administrateur pour accéder à cette partie il doit se connecter et donc passer par un formulaire de connexion.

```
<form method="post" class="row2 mt-5">

    {% if error %}
        <div class="alert alert-danger">{{ error.messageKey|trans(error.messageData, 'security') }}</div>
    {% endif %}

    <h1 class="h3 mb-3 font-weight-normal">Connexion</h1>

    <div id="adressMail" class="col-md-6">
        <input type="email" name="email" id="inputEmail" class="form-control" placeholder="Email" autocomplete="email">
    </div>
    <div id="password" class="col-md-6">
        <input type="password" name="password" id="inputPassword" class="form-control" placeholder="Mot de passe" autocomplete="current-password">
    </div>

    <input type="hidden" name="_csrf_token" value="{{ csrf_token('authenticate') }}>

    <a class="MdpOublie mt-3" href="{{ path('app_forgot_password_request') }}>Mot de passe oublié ?</a>

    <button class="btn btn-lg" id="connexion" type="submit"> Connexion </button>

</form>
```

Ainsi après connexion et selon les droits que l'administrateur possède il pourra voir, modifier, ajouter ou supprimer les données (CRUD) c'est une partie sécurisée ...

On refuse l'accès depuis l'intérieur d'un contrôleur et on contrôle l'accès au niveau de notre vue grâce à deux méthodes :

`IsGranted()` et `denyAccessUnlessGranted()`

qui vont donner l'accès au CRUD uniquement aux utilisateurs qui ont pour rôle « Admin »

```
#[IsGranted('ROLE_ADMIN')]
#[Route('/{id}', name: 'produit_show', methods: ['GET'])]
public function show(Produit $produit): Response
{
    $this->denyAccessUnlessGranted('ROLE_ADMIN');
    return $this->render('produit/show.html.twig', [
        'produit' => $produit,
    ]);
}
```

Pour éviter les messages d'erreur que la fonction `denyAccessUnlessGranted()` renvoie pour les visiteurs du site qui essayent d'accéder à l'espace de connexion de l'administrateur on évite de leur afficher les boutons qui seront affichés que pour l'administrateur du site quand il s'authentifie .

```
{% if is_granted('ROLE_ADMIN') %}
    <ul class="navbar-nav me-auto my-2 my-lg-0 navbar-nav-scroll" style="--bs-scroll-height: 1000px;">
        <li class="nav-item dropdown">
            <a class="nav-link active mx-5" id="navbarDropdown1">
                <i class="bi bi-house"></i>
            </a>
            <ul class="dropdown-menu" aria-labelledby="navbarDropdown1">
                <li><a class="dropdown-item" href="{{ path('produit_index') }}>Gestion des produits</a></li>
                <li><a class="dropdown-item" href="{{ path('categorie_index') }}>Gestion des catégories</a></li>
                <li><a class="dropdown-item" href="{{ path('taxe_index') }}>Gestion des taxes</a></li>
            </ul>
        </li>
    {% endif %}
```

Présentation du jeu d'essai

Afin de justifier le bon fonctionnement de l'outil, il est essentiel de procéder à l'élaboration d'un jeu d'essai.

Mon jeu d'essai se porte sur la table la plus significative, qui est la table produit.

Afin de pouvoir saisir les informations nécessaires à la création d'un produit la base de données sera préparée.

Il est essentiel de s'être déjà authentifié avec un compte administrateur afin de pouvoir interagir sur les différentes données.

Lors de la création d'un produit, il faut saisir un titre ainsi qu'une description qui permettras de présenter le poste en question. L'utilisateur devras aussi mettre en lien une image qui permettras d'illustrer l'offre.



Lors de la création d'un produit, il faut saisir le nom, l'unité de conditionnement, le prix, la catégorie, la taxe (tva) du produit ainsi qu'une description qui permettra de mieux le présenter.

L'administrateur devra aussi mettre en lien une photo qui permettra d'illustrer le produit.

On doit donc avoir dans la base de données :

l'identifiant du produit

l'identifiant de la taxe

l'identifiant de la catégorie

le nom du produit

l'unité de conditionnement du produit

le prix du produit

la description du produit

l'URL de la photo du produit

De plus ce produit doit être affiché sur le site avec la mise en forme adéquate



Données envoyées :

Réponse reçue :

Créer un nouveau produit

Nom produit

Unité conditionnement

Prix produit

Description

Catégorie

Taxe

Insérer une photo
 painchoco.jpg

test1



Id	382
Nom du produit	test1
unité de conditionnement	l'unité
Prix du produit	1.15€
Description	ceci est un test pour le jeu d'essai
Catégorie	Viennoiseries

On constate que les données reçues dans phpMyAdmin correspondent bien aux données envoyées depuis le formulaire.

	Éditer	Copier	Supprimer	id	categorie_id	taxe_id	nom_produit	unite_conditionnement	prix_produit	description	photo
<input type="checkbox"/>				379	11	9	Terrine de saumon	L'unité	18	Saumon élevé en Norvège . 500 grammes	salmon-terrine-592634-6229e95733f88.jpg
<input type="checkbox"/>				381	4	7	Baguette paradis	L'unité	1.1	Farine Complète T80 & graines de sésame. 250 gramm...	baguetteA-6228d124d96d9-62455f8e0c394.jpg
<input type="checkbox"/>				382	9	8	test1	l'unité	1.15	ceci est un test pour le jeu d'essai	painchoco-6229cb1f1afb2-625801f0d6e0e9.jpg

On vérifie ensuite sur notre site que cet ajout dans la table produit soit bien affiché sur la page de sa catégorie spécifique avec la mise en forme adéquate.

Le produit s'affiche bien dans la page des viennoiseries avec la même mise en forme des autres produits
le prix affiché est le prix TTC une donnée calculée
taxe_id = 8 correspond bien à la taxe des viennoiseries qui est égale à 10
categorie_id = 9 correspond bien à la catégorie du produit qui est viennoiseries .



test1

ceci est un test pour le jeu d'essai
1.26 € TTC / l'unité

Veille sur les vulnérabilités de sécurité

La veille est un élément essentiel, il permet de se tenir au courant des dernières vulnérabilités en termes de sécurité. Cela permet également d'anticiper les problèmes.

Plusieurs sources sont possibles pour la recherche d'informations. Les revues spécialisées ainsi que les livres sont des supports appréciés par beaucoup de personnes. Les sites ainsi que les réseaux sociaux ou encore les chaînes vidéo permettent également d'être renseigné sur les dernières failles et mises à jour. Les forums sont une autre source d'informations, certains sont alimentés par des professionnels qui sont dans le métier depuis longtemps.

Ma sélection s'est portée sur plusieurs sources. À commencer par le site de l'Agence nationale de la sécurité des systèmes d'information (ANSSI), qui publie les dernières mises à jour concernant les menaces de cybersécurité.

The screenshot shows the homepage of the ANSSI (Agence nationale de la sécurité des systèmes d'information) website. At the top, there is a navigation bar with links for social media (Facebook, Twitter, LinkedIn, YouTube), a search bar, and sections for 'DECLARATION VULNÉRABILITÉ', 'EN CAS D'INCIDENT', 'ALERTES' (highlighted in red), 'PRESSE', 'RECRUTEMENT', and 'FAQ'. Below the header, there is a banner with icons for 'VISITEZ L'AGENCE' (shield with star), 'UNE ADMINISTRATION' (building), 'VOUS ÊTES : UNE ENTREPRISE' (factory), and 'UN PARTICULIER' (house). A green banner at the bottom says 'FRANCE RELANCE'. The main content area is titled 'ALERTE DE SÉCURITÉ' in red. Below it, a sub-section says 'Les alertes sont des documents destinés à prévenir d'un danger imminent'. A table lists five security alerts:

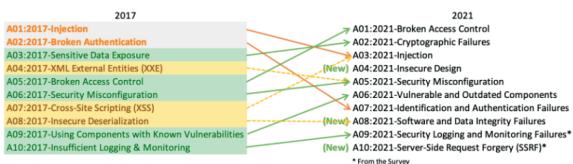
Date	Titre	Statut	Action
13 avril 2022	CERTFR-2022-ALE-003 [Maj] Vulnérabilité dans l'implémentation du protocole RPC par Microsoft	Alerte en cours	Détails
3 mars 2022	CERTFR-2022-ALE-002 Vulnérabilité dans VMware Spring Cloud Gateway	Alerte en cours	Détails
12 janvier 2022	CERTFR-2022-ALE-001 [Maj] Vulnérabilité dans Microsoft Windows	Alerte en cours	Détails
10 décembre 2021	CERTFR-2021-ALE-022 [Maj] Vulnérabilité dans Apache Log4j	Alerte en cours	Détails
8 septembre 2021	CERTFR-2021-ALE-019 [Maj] Vulnérabilité dans Microsoft Windows	Alerte en cours	Détails

L'OWASP (Open Web Application Security Project) dispose d'un projet de classification des failles les plus couramment utilisées par des utilisateurs mal intentionnés sur Internet. Ce document est accompagné d'une qualification des menaces listées et d'une explication sur l'exploitation.

Le site de l'OWASP permet également de se tenir au courant des failles actuelles grâce à une liste régulièrement tenue à jour.

The screenshot shows the OWASP Top 10 2021 page. At the top, there is a navigation bar with links for 'PROJETS', 'CHAPITRES', 'ÉVÉNEMENTS', 'À PROPOS', 'Rechercher OWASP', 'Faire un don', and 'Rejoindre'. Below the header, there is a section titled 'Top 10 de l'OWASP' with a 'Watch' button (194) and a 'Star' button (691). A sub-section titled 'Mondialement reconnu par les développeurs comme la première étape vers un codage plus sécurisé.' is shown. A note below states: 'Les entreprises doivent adopter ce document et commencer à s'assurer que leurs applications Web minimisent ces risques. L'utilisation du Top 10 de l'OWASP est peut-être la première étape la plus efficace pour changer la culture de développement logiciel au sein de votre organisation en une culture qui produit un code plus sécurisé.' To the right, there is a sidebar for the 'Fondation OWASP' with a note: 'La Fondation OWASP® s'efforce d'améliorer la sécurité des logiciels grâce à ses projets de logiciels open source dirigés par la communauté, des centaines de chapitres dans le monde, des dizaines de milliers de membres et en organisant des conférences locales et mondiales.' Below that, there is a section titled 'Renseignements sur le projet' with links for 'OWASP Top 10:2021', 'Composition du Top 10 de l'OWASP', 'Projet phare', 'Documentation', 'Constructeur', 'Défenseur', and 'Version précédente (2017)'.

Les 10 principaux risques de sécurité des applications Web
Il y a trois nouvelles catégories, quatre catégories avec des changements de nom et de portée, et une certaine consolidation dans le Top 10 pour 2021.



- A01:2021-Broken Access Control passe de la cinquième position ; 94 % des applications ont été testées pour une forme de contrôle d'accès cassé. Les 34 énumérations de faiblesses communes (CWE) mappées au contrôle d'accès interrompu avaient plus d'occurrences dans les applications que toute autre catégorie.
- A02:2021 - Les écarts cryptographiques passent d'une position à la deuxième, anciennement connue sous le nom d'exposition de données sensibles, qui était un symptôme général plutôt qu'une cause première. L'attention renouvelée ici est sur les défaillances liées à la cryptographie qui conduisent souvent à l'exposition de données sensibles ou à la compromission du système.
- A03:2021-L'injection glisse jusqu'à la troisième position. 94 % des applications ont été testées pour une forme d'injection, et les 33 CWE cartographiés dans cette catégorie ont le deuxième plus grand nombre d'occurrences dans les applications. Le Cross-Site Scripting fait désormais partie de cette catégorie dans cette édition.
- A04:2021-Conception non sécurisée est une nouvelle catégorie pour 2021, axée sur les risques liés aux défauts de conception. Si nous voulons vraiment « aller à gauche » en tant qu'industrie, cela nécessite une utilisation accrue de la modélisation des menaces, des modèles et principes de conception sécurisée et des architectures de référence.
- A05:2021-La mauvaise configuration de la sécurité passe de la 6e place à l'édition précédente ; 90 % des applications ont été testées pour une certaine forme de mauvaise configuration. Avec de plus en plus d'évolutions vers des logiciels hautement configurables, il n'est pas surprenant de voir cette catégorie monter en puissance. L'ancienne catégorie des entités externes XML (XXE) fait désormais partie de cette catégorie.

Cette faille utilise la balise HTML `<input type="file" />` et permet d'importer n'importe quel type de fichier. Un utilisateur pourrait donc télécharger un fichier .php malveillant pour prendre accès au site et sa base de donnée

La première chose à faire est de respecter quelques règles comme le stipule le site anglophone sans.org pour implémenter un système upload sécurisé :

1. Renommer le fichier
2. Ne pas enregistrer les fichiers à la racine du site
3. Vérifier la taille du fichier
4. Vérifier les extensions

Voilà comment j'ai procédé pour le traitement des fichiers photos et pour éviter la faille upload

```
//on va traiter la photo avant l'envoie à la base de donnée
$photo = $form->get('photo')->getData();
if($photo){
    // on récupère la photo originale et on la nettoie pour éviter toute injection extérieure notamment les balises html
    $originalFilename = pathinfo($photo->getClientOriginalName(), PATHINFO_FILENAME);
    // on récupère le nom de la photo original slugger ça sert à nettoyer mes données
    // ça remplace le strip tags dans le php natif
    $safeFilename = $slugger->slug($originalFilename);
    $newFilename = $safeFilename.'-'.$_uniqueid().'.'.$photo->guessExtension();

    try {
        // images directory configuré dans service .yaml
        $photo->move(
            $this->getParameter('images_directory'),
            $newFilename
        );
    } catch (FileException $e){
        // veuillez inserer une photo
    }
    $produit->setPhoto($newFilename);
}
```

On nettoie les données du fichier grâce au slugger et on la renomme .

Dans le formulaire d'ajout de produit on ajoute des contraintes pour éviter de télécharger un fichier.php ou autre fichier qui va nous compromettre notre base de donnée ...

```
parameters:
    images_directory: '%kernel.project_dir%/public/uploads'
```

Il est très compliqué de s'informer sur l'intégralité des failles de sécurités et leurs solutions vu qu'elles sont nombreuses.

Selon moi, la bonne pratique serait de cibler des failles potentiellement disponibles sur son application et d'appliquer une veille régulière sur celle-ci.

Je vais donc vous montrer une des failles que j'ai évité grâce à la solution du site anglophone sans.org qui est la

Faille Upload

```
->add('photo', FileType::class,
    'label'=>'Inserer une photo',
    'mapped'=> false,
    'required'=>true,
    'constraints'=>[
        new File([
            'maxSize' =>'4000k',
            'mimeTypes'=>[
                'image/jpeg',
                'image/png'
            ],
        ]),
    ],
);
```

Description d'une situation de travail ayant nécessité une recherche d'un site anglophone

Ayant essentiellement utilisé le framework Symfony pour toute la partie back-end de ce projet j'ai dû pas mal me documenter sur le site officiel de symfony qui est exclusivement en anglais.

J'ai aussi parfois été confronté à certains problèmes d'utilisation et les informations les plus pertinente étant en anglais j'ai donc été confronté à de nombreuses situations de travail ayant nécessité une recherche à partir d'un site anglophone.

Je vais donc vous présenter quelques bonnes pratiques du framework Symfony et les ressources anglophones qui m'ont le plus aidé tout au long de ce projet.

I. Extrait du site

The symfony framework best practices

Make your Controller Extend the AbstractController Base Controller

Symfony provides a base controller which includes shortcuts for the most common needs such as rendering templates or checking security permissions.

Extending your controllers from this base controller couples your application to Symfony. Coupling is generally wrong, but it may be OK in this case because controllers shouldn't contain any business logic. Controllers should contain nothing more than a few lines of glue-code, so you are not coupling the important parts of your application.

Use Dependency Injection to Get Services

If you extend the base AbstractController, you can only access to the most common services (e.g twig, router, doctrine, etc.), directly from the container via `$this->container->get()`. Instead, you must use dependency injection to fetch services by type-hinting action method arguments or constructor arguments.

II. Traduction de l'extrait en français

Les bonnes pratiques du framework Symfony

Faites en sorte que votre contrôleur étende le AbstractControllercontrôleur de base

Symfony fournit un contrôleur de base qui inclut des raccourcis pour les besoins les plus courants tels que le rendu des modèles ou la vérification des autorisations de sécurité.

L'extension de vos contrôleurs à partir de ce contrôleur de base couple votre application à Symfony. Le couplage est généralement erroné, mais il peut être correct dans ce cas car les contrôleurs ne doivent contenir aucune logique métier. Les contrôleurs ne doivent contenir que quelques lignes de glue-code afin de ne pas coupler les parties importantes de votre application.

Utiliser l'injection de dépendance pour obtenir des services

Si vous étendez la base AbstractController, vous ne pourrez accéder qu'aux services les plus courants (par exemple twig, router, doctrine, etc.), directement depuis le conteneur via `$this->container->get()`. Au lieu de cela, vous devez utiliser l'injection de dépendances pour récupérer les services en indiquant des arguments de méthode d'action ou des arguments de constructeur.

Pour conclure, étant novice dans le domaine du développement avant de commencer cette formation, cela m'a apporté de nombreuses connaissances et de bonnes bases pour débuter dans le métier de développeur.

Lors du stage d'immersion de 3 semaines, j'ai pu mettre en pratique ce que j'ai appris et me suis confrontée aux difficultés techniques de conception et développement pour le site de la boulangerie. Cela m'a aussi permis de comprendre plus en détail les différentes techniques de référencement.

Je tiens à remercier Mme BRICE Audrey pour sa confiance et la chance qu'elle m'a donné pour intégrer la formation que je voulais tant depuis mon arrivée en FRANCE en 2016 .

Je remercie l'ensemble des formateurs qui m'ont enseigné les différents langages de programmation et techniques de travail, et qui ont su répondre à mes interrogations tout au long de cette formation.

Je remercie également les lecteurs de ce document pour avoir pris le temps d'examiner son contenu.

Annexes

Design patterns :

Un patron de conception est un arrangement caractéristique de modules, reconnu comme bonne pratique pour résoudre un problème de conception d'un logiciel, il décrit une solution standard, utilisable dans la conception de différents logiciels.

GitHub :

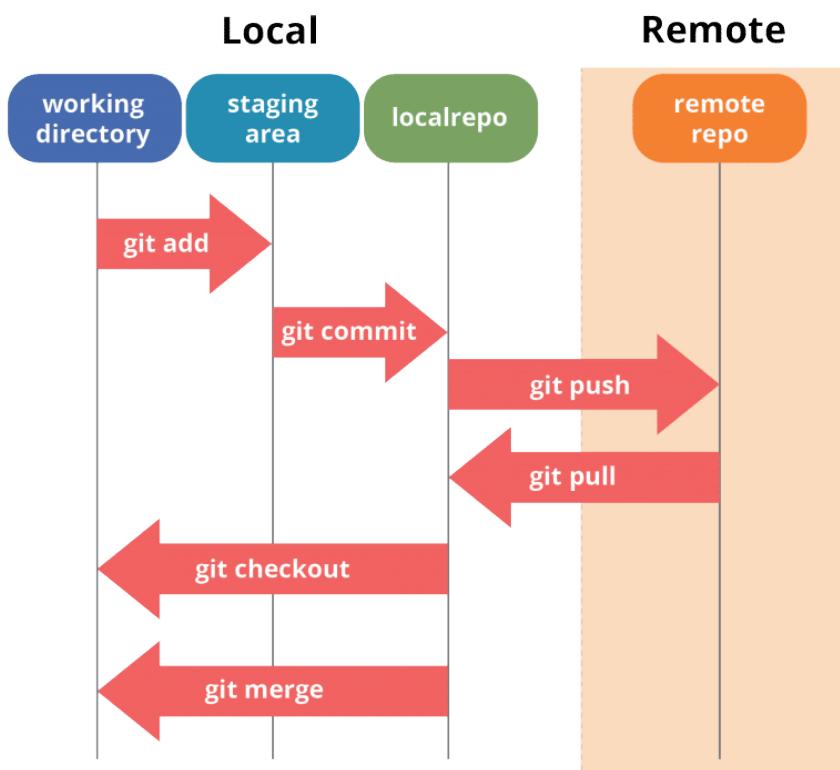
GitHub est un site de partage de code, sur lequel on peut publier des projets dont le code est géré avec le système de gestion de version Git. Par défaut, le système est open source, ce qui signifie que tout le monde peut consulter le code, l'utiliser pour apprendre ou l'améliorer et collaborer aux projets.

Cela fonctionne grâce à un système de branche, lors de la création de notre dépôt Git on disposera d'une branche « main » celle-ci va nous être vraiment utile lorsque le travail sera terminé.

Car le principe justement est de créer d'autres branches de travail afin de ne jamais toucher à notre branche principale pour pouvoir apporter des modifications à notre code sans craindre de commettre une erreur. Et enfin une fois le travail terminé on va pouvoir migrer tout ceci sur notre branche principale.

De plus afin de pouvoir détailler encore plus son travail de manière chronologique on va créer des « commits » qui sont des genres de sauvegardes via la branche sur laquelle on travaille.

On va donner un nom significatif à notre commit car lorsque l'on reviendra quelques jours plus tard sur notre travail on saura directement de quoi on parle, ceci est d'autant plus pratique lorsque l'on travail en équipe.



Installation de symfony-6

Avant de créer ma première application Symfony j'ai du installer PHP 8.0.2 et 'Composer', qui est utilisé pour installer les packages PHP.

En option, j'ai également installé Symfony CLI . Cela crée un binaire appelé `symfony` qui fournit tous les outils dont on a besoin pour développer et exécuter notre application Symfony localement.

Le `symfony`binaire fournit également un outil pour vérifier si votre ordinateur répond à toutes les exigences. en exécutant cette commande sur le terminal:

```
$ symfony check:requirements
```

Si y'a pas de messages d'erreur c'est que `symfony` est bel et bien installé

J'ai crée un nouveau projet grâce à la commande (ici mon projet s'appelle `projetB`)

```
$ symfony new boulangerie --full
```

On démarre le serveur pour voir si tout fonctionne bien grâce à la commande

```
$ symfony server:start
```

Si on veut éteindre le serveur on utilise le raccourci Ctrl+C sur le terminal ou alors on exécute la commande

```
$ symfony server:stop
```

Création des entités

Une entité est une classe PHP qui est connectée à une table de la base de données via l'ORM.

Lorsqu'une entité est liée à une table, il y a en général un fichier « repository » associé

Un repository permet la génération de requêtes simples ou complexes...

J'ai crée mes entités grâce à la commande `php bin/console make:entity`.

Une fois la commande exécutée on aura une suite de questions sur le nom de l'entité (le nom de la table) et les propriétés à créer .

Dès qu'on ajoute une propriété, `symfony` nous demande d'insérer son 'Type' sa 'taille' et si le champ peut être null dans la base de données ...

Une fois terminé, le fichier d'entité est dans le dossier `SRC/Entity/nom de notre fichier.php`

y'a automatiquement un fichier associé à l'entité qui se crée dans le dossier

`SRC/Repository/nom de notre fichierRepository.php`

Dans le fichier `nom de mon fichier.php`, on retrouve la class `nom de mon fichier` contenant ses propriétés, ses getters et setters correspondant afin d'accéder à ses propriétés de manière sécurisée.

À ce stade l'entité est créée, mais n'existe pas dans la base de données.

Pour mettre à jour ma base de données j'ai utilisé la commande suivante : `symfony console make:migration`

Cette commande va créer un fichier de migration contenant le code SQL à exécuter en fonction du système de gestion de base de données utilisées (MySQL pour ma part)
ce fichier se trouvera dans le dossier Migrations.

Pour exécuter le fichier SQL et la migration j'ai utilisé la commande

```
php bin/console doctrine:migrations:migrate
```

On vérifie PHPMyAdmin si la table (`nom de notre class`) est bien crée ...

Ensuite on passe à l'étape suivante

Créations des contrôleurs

Un contrôleur est une fonction PHP qui lit les informations depuis l'objet Request, crée et retourne une réponse HTTP (un objet Response).

Retourner une réponse signifie donc tout simplement : instancier un objet Response, disons \$response, et faire un return \$response.

La réponse peut être une page HTML, un téléchargement de fichier, une redirection ou autre... .

Le contrôleur exécute toute la logique arbitraire dont l'application a besoin pour interpréter et rendre le contenu d'une page.

Dans Symfony, la plupart des contrôleurs sont implémentés sous la forme de classes PHP.

Voici un exemple d'un contrôleur

```
namespace App\Controller;

use App\Entity\Produit;
use App\Form\ProduitType;
use App\Repository\ProduitRepository;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\File\Exception\FileException;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\String\Slugger\SluggerInterface;

#[Route('/produit')]
class ProduitController extends AbstractController
{
    //creation de view
    #[Route('/', name: 'produit_index', methods: ['GET'])]
    public function index(ProduitRepository $produitRepository): Response
    {
        return $this->render('produit/index.html.twig', [
            'produits' => $produitRepository->findAll(),
        ]);
    }
}
```

Symfony profite de la fonctionnalité d'espace de noms de PHP, pour créer un espace de noms pour toute la class controller.

L'attribut #[Route('/ ', name: 'produit_index')] est ce qui fait de la méthode index() un contrôleur.

Lorsque l'utilisateur visitera la page «/» dans un navigateur,

la fonction index() de ce contrôleur sera exécutée et une réponse sera renvoyée.

Création d'une fonction contenant toute la logique et ayant comme argument l'instanciation du fichierRepository pour pouvoir l'utiliser dans la fonction par la suite.

(dans l'exemple au dessus le produitRepository est instancié pour pouvoir utiliser la fonction findAll() qui est une requête par défaut qui existe dans le produitRepository).

Le contrôleur retourne une réponse grâce à la méthode render() qui prend en paramètres le chemin ainsi que le nom du template et un tableau de variables...

CRUD(Create Read Update Delete) des entités

Exemple: Après avoir créé l'entité produit , il faut lui implémenter le CRUD, pour gérer les différents produits.

La commande

```
php bin/console make:crud produit
```

nous génère le crud pour l'entité produit ainsi que les différents fichiers liés à son bon fonctionnement.