



APPLICATION WEB ET WEB MOBILE

Jean-Yves PIERRE-GEROME
CEF – Promotion septembre 2023



SOMMAIRE

1. INTRODUCTION	4
2. PRESENTATION	5
3. DEVELOPPER LES FRONT END D'UNE APPLICATION WEB	6
3.1. Maquetter l'application	6
3.2. Charte graphique et logo	6
3.3. Exemple maquette	7
3.4. Réalisation d'une interface utilisateur statiques web et web mobile	10
3.5. Développer la partie dynamique des interfaces utilisateurs web ou web mobile	11
3.6. Dynamiser un site Web avec Angular	14
4. DEVELOPPER LES BACK END D'UNE APPLICATION WEB	18
4.1. Prototypage de la base de données	18
4.2. Moyen de paiement en ligne	23
4.3. Problème rencontré	24
4.4. Solution apportée	25
4.5. Développer des composants d'accès aux données SQL et NoSQL	26
4.6. Développer des composants métier coté serveur	31
4.7. Documenter le déploiement d'une application web ou web mobile	32
4.8. Le versionning (Git et GitHub)	33



Application web et web mobile

Projet réalisé dans le cadre de la présentation au
Titre Professionnel Développeur Web et Web Mobile

Présenté par

Jean-Yves PIERRE-GEROME
CEF – Promotion septembre 2023

1. Introduction

J'ai toujours été intéressé par le domaine de l'informatique, notamment par le développement. Pour approfondir mes connaissances, je me suis donc autoformé au WLangage, et dans un souci de progression continue, je suis devenu autodidacte.

Actuellement en poste dans une société éditrice de logiciel, je suis responsable du service Technique. Mon rôle consiste souvent à remonter à nos développeurs les besoins de nos clients, les bugs rencontrés ainsi que des demandes d'évolution.

Pour cela, je dois transcrire ces besoins sous forme d'un cahier des charges détaillés. Cependant malgré mes connaissances en développement, il m'arrivait parfois de ne pas saisir pleinement les retours des développeurs en particulier les problématiques complexes liées à la mise en place de ces correctifs et/ou évolution.

C'est pourquoi, j'ai décidé de suivre cette formation afin de mieux comprendre le monde du développement et d'améliorer mes connaissances. Le centre européen de formation a pu répondre à mon attente.

Dans le cadre de ma formation, j'ai eu l'occasion de réaliser un projet, il m'a donné l'opportunité de développer et de renforcer mes compétences dans plusieurs domaines de développement d'applications web. Il m'a permis d'approfondir mes connaissances théoriques, mais également de les appliquer à des situations pratique et concrètes.

En conclusion, ce projet m'a permis de me mettre en situation, de développer un esprit logique, de maîtriser un langage de programmation, d'être persévérant, de travailler en collaboration avec une équipe et surtout de respecter les bonnes pratiques.

2. Présentation

Durant ma formation, j'ai eu l'opportunité de travailler sur plusieurs projets, dont je vais détailler les principaux aspects au fil de cette présentation.

L'un de mes projets avait pour but de donner la possibilité aux utilisateurs d'acheter un cursus de cours ou des cours individuels en ligne. Pour cela, j'ai dû donner la possibilité de visualiser les cursus et le détail des cours individuels pour les utilisateurs non connectés.

Les utilisateurs pourront également créer leur compte et confirmer la création de ce dernier par l'intermédiaire d'un lien dans le mail. Une fois connectés, les utilisateurs pourront visualiser le détail des cursus ainsi que des cours. Ils pourront aussi réaliser l'achat des cursus et des cours individuels directement depuis ma plateforme. Pour cela, j'ai choisi STRIPE comme moyen de paiement en ligne en raison de sa fiabilité, de sa sécurité et de sa facilité d'intégration avec Symfony. Grâce à STRIPE, j'ai pu offrir une expérience de paiement fluide et sécurisée, permettant ainsi aux utilisateurs de finaliser leurs transactions rapidement et en toute confiance.

Une fois l'achat réalisé, l'utilisateur aura accès à l'intégralité du cursus et/ou du cours qu'il aura acheté. J'ai mis en place également un système d'obtention de certification. Le fonctionnement est le suivant, pour chaque utilisateur qui aura validé un cursus complet, une certification lui sera accordé et accessible sur sa fiche dès qu'il sera connecté.

J'ai développé une plateforme d'administration avec EasyAdmin Bundle pour la gestion des utilisateurs, les thèmes, les cursus ainsi que les cours. Les cursus et les cours seront intégrés par ma plateforme sous forme de fichier au format TXT. Quant aux utilisateurs, ils pourront être également créés par cette plateforme.

En plus de ce projet, j'ai réalisé d'autres développements comme un CV en HTML, un espace de commentaire dynamique, dynamiser un site avec le framework Angular ainsi que l'hébergement d'une application web.

3. Développer les front end d'une application web

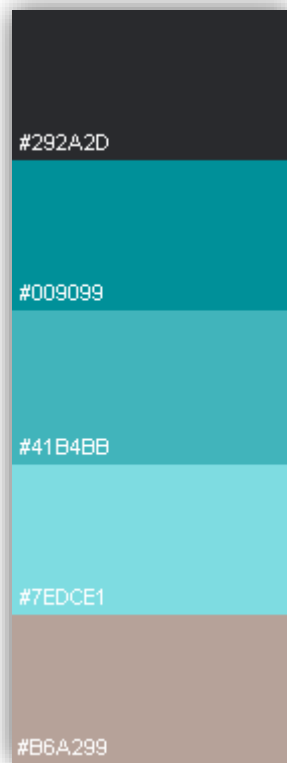
3.1. Maquetter l'application

L'une des bonnes pratiques du développement est le maquettage de l'application. Cela permet de visualiser clairement à quoi ressemblera l'application avant de commencer à coder, d'identifier rapidement les problèmes potentiels de la conception ou de l'ergonomie.

Dans ce but, j'ai réalisé cette maquette avec Figma, qui est un outil simple d'utilisation avec une bonne prise en main. J'ai créé 2 versions de maquettes, l'une au format desktop et l'autre au format mobile, pour imaginer la structure de mes pages et assurer un design responsif.

3.2. Charte graphique et logo

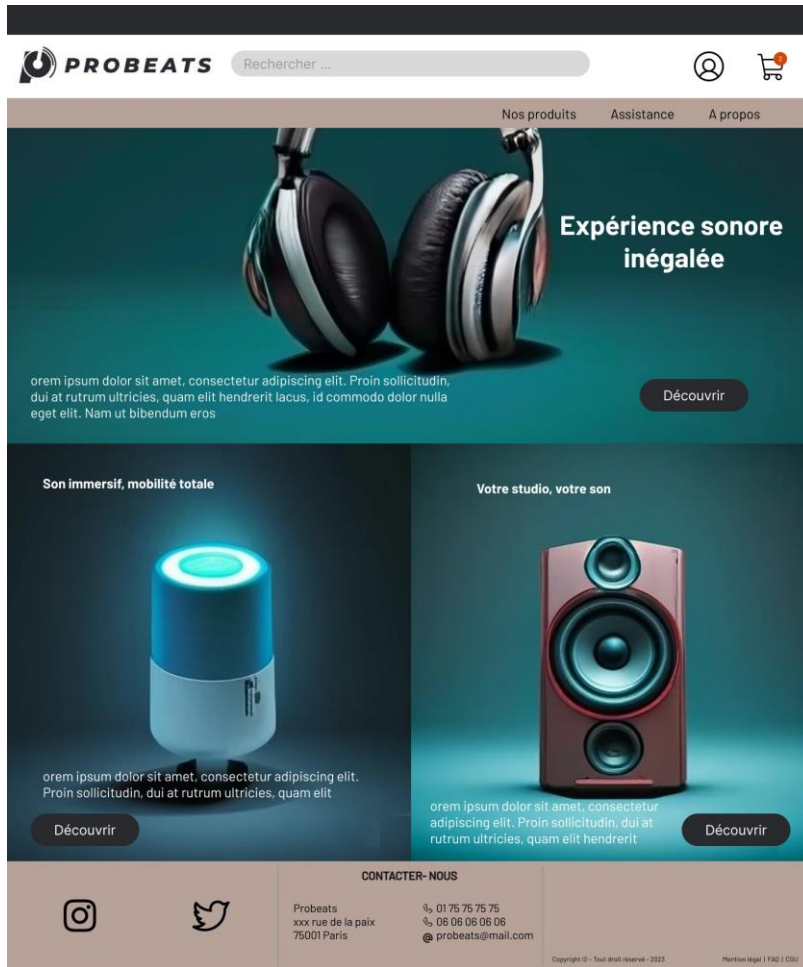
Pour ce projet, le client a souhaité que je respecte certaines contraintes. A savoir la police de caractère « **Barlow** » sur l'ensemble du site ainsi qu'une palette de couleur.



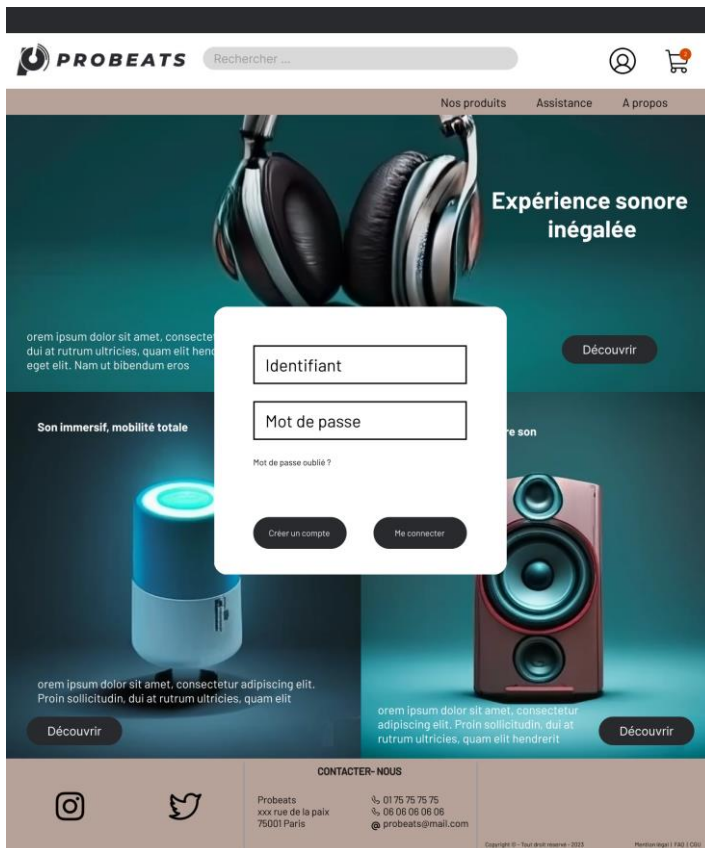
Et pour finir, le logo et la favicon a été créé spécialement pour l'occasion.



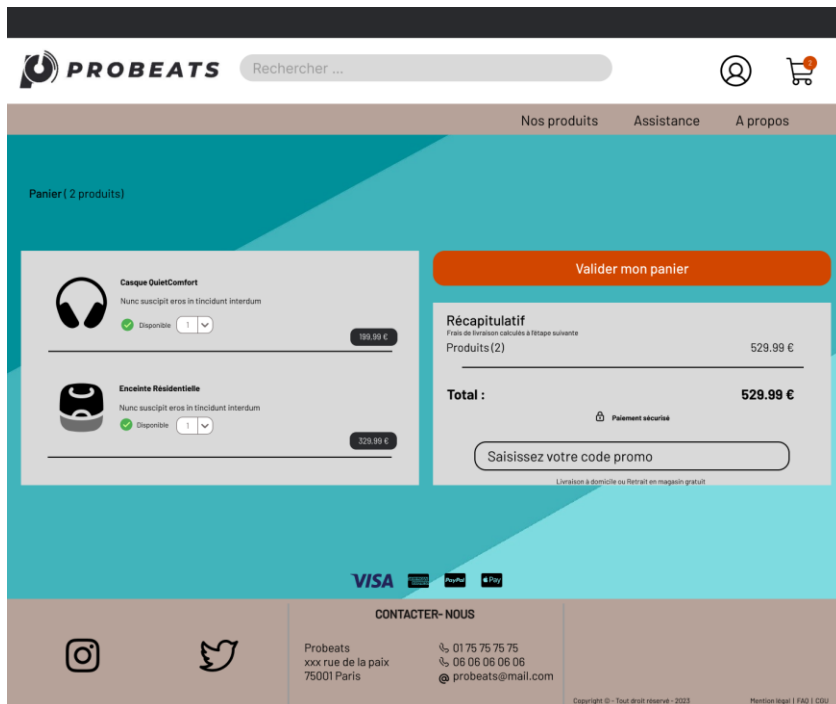
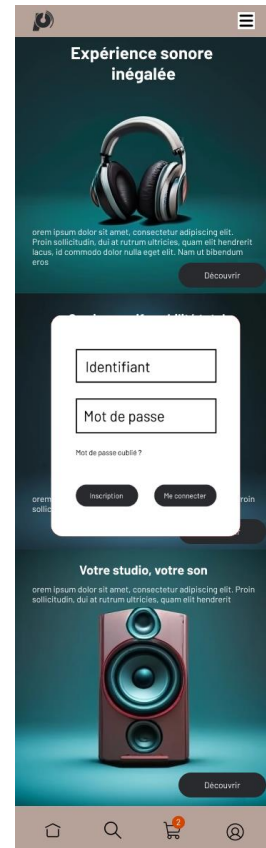
3.3. Exemple maquette



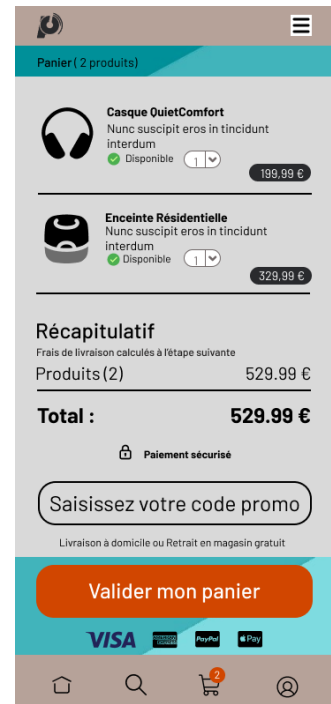
Page d'accueil



Mire de connexion



Panier utilisateur



3.4. Réalisation d'une interface utilisateur statiques web et web mobile

Lors de ma formation, il m'a été demandé de réaliser mon CV en utilisant un langage de balisage HTML pour la structure du contenu et les feuilles de styles CSS pour la présentation visuelle.

Pour faciliter le référencement naturel (SEO), j'ai utilisé différentes balises sémantiques. Elles permettent de mieux structurer et de rendre le contenu compréhensible aussi bien pour les utilisateurs, mais également pour les moteurs de recherche.

Exemple

```
<div class="nav_container">
  <div>
    <address>
      <p><a class="nav_container_link"
        target=" blank"
        href="https://www.google.fr/maps/place/12+Rue+des+Oliviers,+93600+Aulnay-sous-Bois/@48.952624,2.494328,17z/data=!3m1!4b1!4m6!3m5!1s0x47e61
        >12 rue des oliviers<br>93600 Aulnay-sous-bois</a></p>
    </address>
  </div>
  <div>
    <address>
      <p><a class="nav_container_link" href="mailto:john.doe@gmail.com">john.doe@gmail.com</a></p>
    </address>
  </div>
  <div>
    <address>
      <p><a class="nav_container_link" href="tel:0606060606">06 06 06 06 06</a></p>
    </address>
  </div>
</div>
```

J'ai utilisé la balise `<address>`, elle me permet de spécifier les informations de contact qui améliore l'accessibilité, j'ai également intégré dans une autre balise un lien hypertexte qui pointe vers une localisation sur Google Maps.

Dans la même continuité, j'ai ajouté d'autres blocs `<address>` qui contiennent un lien **mailto** ainsi qu'un lien **tel**.

Autre exemple de balise que j'ai utilisé, la balise `<article>`

```
<article>
  <h2 class="main_container_left_title">A propos</h2>
  <p class="main_container_left_p">Fort de 12 ans d'expérience en hotline informatique,
    j'ai gravi les échelons pour devenir responsable du pôle support,
    développant ainsi des compétences en gestion d'équipe et en résolution de problèmes.
    Actuellement en reconversion vers le développement web,
    je cherche à combiner mon expertise en support technique avec une nouvelle compréhension
    du développement pour créer des solutions innovantes.
    Mon parcours professionnel est guidé par une passion pour l'apprentissage continu
    et l'adaptation aux évolutions technologiques.</p>
</article>
```

Celle-ci m'a permis d'encapsuler un contenu, elle est souvent utilisée pour un article de blog ou une nouvelle, dans mon cas, j'ai souhaité mettre en avant une partie de mon expérience professionnelle.

Concernant la mise en page, j'ai utilisé une feuille de style CSS, elle s'intègre dans la balise `<head>`

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Présentation d'un CV">
  <link rel="stylesheet" href="./css/style.css">
  <title>Mon CV</title>
  <link rel="shortcut icon" href="./images/cv.png" type="image/x-icon">
</head>
```

Dans mon fichier `style.css`, j'ai pu manipuler certains éléments

```
.main_container_right_title{
  background-color: #dbc2a4;
  color: black;
  padding: 10px;
  margin-right: 20px;
  border-left: 10px solid #782221;
  width: 650px;
}
```

Exemple pour la **class** `main_container_right_title`, j'applique une couleur de fond, une couleur pour mon texte, un espacement intérieur, une marge à droite ainsi qu'une taille de texte.

3.5. Développer la partie dynamique des interfaces utilisateurs web ou web mobile

Dans le cadre de ce titre professionnel, j'ai dynamisé un espace de commentaire afin d'en faire apparaître un nouveau directement dans la liste des commentaires au moment où je valide le formulaire. Pour cela, j'utilise le langage JavaScript, ainsi que le DOM.

Le formulaire est le suivant

Mark Edwards

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Recusandae accusantium quae doloribus. Nobis ipsam libero asperiores a velit ab nemo facere molestias quisquam. Libero, repudiandae dignissimos. Doloribus hic placeat soluta?

Écrire un commentaire

Prénom

Nom

Commentaire

Max. 500 caractères

Envoyer

Le but est de pouvoir rajouter le nouveau commentaire à la suite des commentaires existant sans recharger la page.

J'ai donc créé une fonction **addComment()** qui encapsule les éléments suivants.

Je charge dans un premier temps la balise **form** dans ma variable **form**
Puis j'écoute l'événement pour savoir si mon formulaire a été soumis.
Et enfin, je désactive le rafraîchissement de la page.

```
let form = document.querySelector("form");
form.addEventListener("submit", function(e){
  e.preventDefault();
});
```

Dans l'étape suivante, je stocke mes balises par leur **id** dans leur variable respective.

```
let firstName = document.getElementById("first-name");
let lastName = document.getElementById("last-name");
let message = document.getElementById("message");
let errorMessage = document.getElementById("error-message");
```

Si aucune valeur n'a été saisie, j'affiche un message d'erreur, dans le cas contraire, il n'y a pas de message d'erreur, et j'exécute mon code.

```
if (firstName.value == "" || lastName.value == "" || message.value == "") {
  errorMessage.style.display = "block";
}else{
  errorMessage.style.display = "none";
}
```

Je stocke la balise **comment-list** par son **id** dans une variable, ce qui me permettra par la suite de créer un nouveau bloc dans cette balise.

Puis je crée différents éléments comme un paragraphe, un titre de niveau 3 et plusieurs div.

Je stocke ensuite les valeurs **firstName** et **lastName** dans ma variable titre. En utilisant **titre.innerHTML** cela permet de modifier le contenu HTML de l'élément ciblé.

```
let listComment = document.getElementById("comment-list");
let divContainer = document.createElement("div");
let divTitre = document.createElement("div");
let titre = document.createElement("h3");
let divComment = document.createElement("div");
let comment = document.createElement("p");

titre.innerHTML = firstName.value + " " + lastName.value;
titre.classList.add("font-medium", "text-gray-900");

comment.innerHTML = message.value;
```

Dans l'étape suivante, la méthode **appendChild()** me permettra d'afficher les éléments ciblés (**titre**, **comment**, **divComment**, **divTitre** et **divContainer**) comme dernier enfant d'un élément parent.

Après avoir ajouté mon commentaire, je réinitialise mon formulaire pour effacer les données saisies.

```
divTitre.appendChild(titre);
divTitre.classList.add("flex-1", "py-10", "border-t", "border-gray-200");

divComment.appendChild(comment);
divComment.classList.add("prose", "prose-sm", "mt-4", "max-w-none", "text-gray-500");

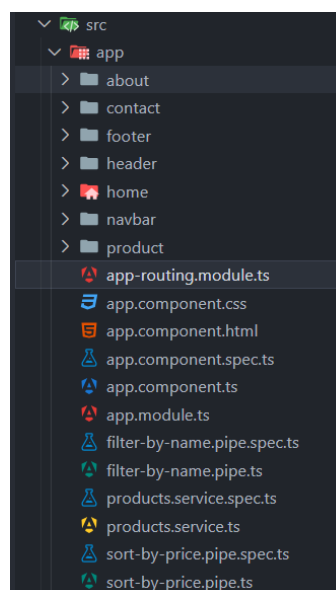
divTitre.appendChild(divComment);
divContainer.appendChild(divTitre);
listComment.appendChild(divContainer);
```

3.6. Dynamiser un site Web avec Angular

J'ai eu l'opportunité de réaliser un projet avec le framework Angular, il m'a permis de créer une application web dynamique.

L'objectif était de créer un site web responsive spécialisé dans la vente de figurine inspirées d'Astérix et Obélix. Je devais mettre en place un système de recherche et de tri dynamique par prix ou par nom de figurines.

Pour cela, j'ai utilisé Angular pour la création d'une architecture de site modulable et maintenable.



Dans ce projet, on visualise les composants (about, contact, footer, header home, etc.), le module routing dans **app-routing.module.ts** et les différents fichiers de service et de pipes.

- Le routing

J'ai utilisé des routes pour gérer la navigation entre les pages. Chaque route est définie par un **path** et un composant qui sera affiché lorsque qu'on accède à ce chemin.

Ex :

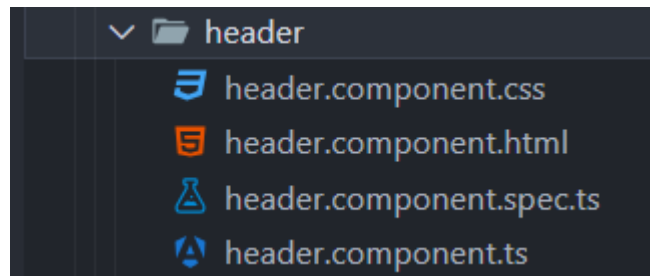
- Le chemin « **about** » est associé au composant **AboutComponent** qui affiche la page A propos.
- Le chemin « **product/:id** » définit une route dynamique, où l' :id est un paramètre dans l'URL qui est utilisé pour afficher le détail du produit
- La route **path : '**'** est une route de redirection pour toutes les URLs non définies, qui renvoie à la page d'accueil.

```
You, il y a 6 mois | 1 author (You)
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3 import { HomeComponent } from '../home/home.component';
4 import { AboutComponent } from '../about/about.component';
5 import { ContactComponent } from '../contact/contact.component';
6 import { ProductComponent } from '../product/product.component';
7
8 const routes: Routes = [
9   {
10     path: "",
11     component: HomeComponent
12   },
13   {
14     path: "about",
15     component: AboutComponent
16   },
17   {
18     path: "contact",
19     component: ContactComponent
20   },
21   {
22     path: "product/:id",
23     component: ProductComponent
24   },
25   { path: '**',
26     redirectTo: ''
27   }
28 ];
29
30 @NgModule({
31   imports: [RouterModule.forRoot(routes)],
32   exports: [RouterModule]
33 })
34 export class AppRoutingModule { }
35
```

You, il y a 14 mois | 1 author (You)

- Les composants

Les composants me permettent de structurer mon site de manière claire et réutilisable, comme le composant header, footer, home, product, about).



Exemple du composant header

Chaque composant est composé de 4 fichiers :

- *****.component.css : pour l'aspect visuel du site
- *****.component.html : ce que j'afficherais à l'écran
- *****.component.spec.ts : pour les tests unitaires
- *****.component.ts : il me permet de définir toute la logique du site.

- Les services

Dans mon projet, j'ai mis en place un service qui dans mon cas me permet est une collection d'objet Javascript qui contiennent les informations sur les figurines (nom, description, prix).

```
import { Injectable } from '@angular/core';

You, il y a 14 mois | 1 author (You)
@Injectable({
  providedIn: 'root'
})
export class ProductsService {
  private figurines = [
    {
      id: 1,
      name: "Abraracourcix",
      description: `Le village n'a qu'un seul et unique chef, même si certains ont parfois
      tendance à l'oublier et se verraient bien chef à la place du chef.
      Abraracourcix n'hésites alors pas à hausser le ton pour rappeler à
      tout le monde que le chef ici, c'est lui !`,
      price: 49.99,
      height: 17.7,
      material: "résine",
      imageUrl: "./assets/Abraracourcix.jpg"
    },
  ],
```


- Le pipes

J'ai créé également 2 fonctionnalités, l'un est un système de tri dynamique (croissant et décroissant), l'autre est une recherche par nom.

```
1 import { Pipe, PipeTransform } from '@angular/core';
2
3 You, il y a 14 mois | 1 author (You)
4 @Pipe({
5   name: 'filterByName'
6 })
7 export class FilterByNamePipe implements PipeTransform {
8   transform(value: unknown, ...args: unknown[]): unknown {
9     return null;
10  }
11 }
12 }
```

Recherche par nom

```
import { Pipe, PipeTransform } from '@angular/core';
You, il y a 14 mois | 1 author (You)
@Pipe({
  name: 'sortByPrice'
})
export class SortByPricePipe implements PipeTransform {
  //a.price - b.price tri croissant
  //b.price - a.price tri décroissant
  transform(price: any[], sortOrder: string): any[] {
    if(sortOrder === "croissant"){
      return price.sort((a:any,b:any) => {return a.price - b.price});
    }else if (sortOrder === "decroissant"){
      return price.sort((a:any,b:any) => {return b.price - a.price});
    }
    return price;
  }
}
```

Tri croissant ou décroissant

4. Développer les back end d'une application web

4.1. Prototypage de la base de données

Avant de créer une base de données pour mon application, j'ai planifié et structuré mon travail afin de prévenir toute erreur susceptible d'entraîner des problèmes de performance, de sécurité ou de maintenance sur le long terme.

Pour cela, j'ai utilisé un outil en ligne, un logiciel spécialisé dans la modélisation des systèmes d'information, pour créer les différents diagrammes MCD (Modèle Conceptuel de Donnée) et MLD (Modèle Logique de Donnée). Cet outil m'a permis de visualiser et d'identifier clairement mes entités, ainsi que leurs relations, ce qui a facilité la structure de ma base de données.

J'ai pu définir les clés primaires et étrangère ainsi que les relations entre les différentes tables de manière intuitive. Cela a assuré une modélisation cohérente et efficace, garantissant l'intégrité des données et une gestion optimale des relations entre les entités.

4.1.1 Conception du MCD : Modélisation des entités

Dans un premier temps, j'ai créé mon MCD, cela m'a permis de me concentrer sur la représentation des données du point de vue utilisateurs sans me préoccuper de l'aspect technique. Dans mon cas, les entités représentent un concept simple.

USER : les utilisateurs de la plateforme.

LESSONS : les leçons disponibles

CURSUS : les parcours de formation.

THEMES : les catégories de cours

PURCHASE : les achats effectués

Les relations entre les entités me permettent de comprendre comment elles interagissent entre elles.

Un utilisateur effectue des achats.

Un achat concerne une leçon ou un cursus.

Un cursus contient des leçons.

Un thème catégorise des cursus.

Pour chaque relation, j'ai défini les cardinalités qui précisent le nombre minimal et maximal d'occurrence entre les entités.

Entre USER et PURCHASE : un utilisateur peut effectuer un plusieurs achats.

Entre PURCHASE et CURSUS/LESSONS : un achat concerne zéro ou plusieurs cursus/leçons, car un achat peut porter soit sur un cursus soit sur une leçon.

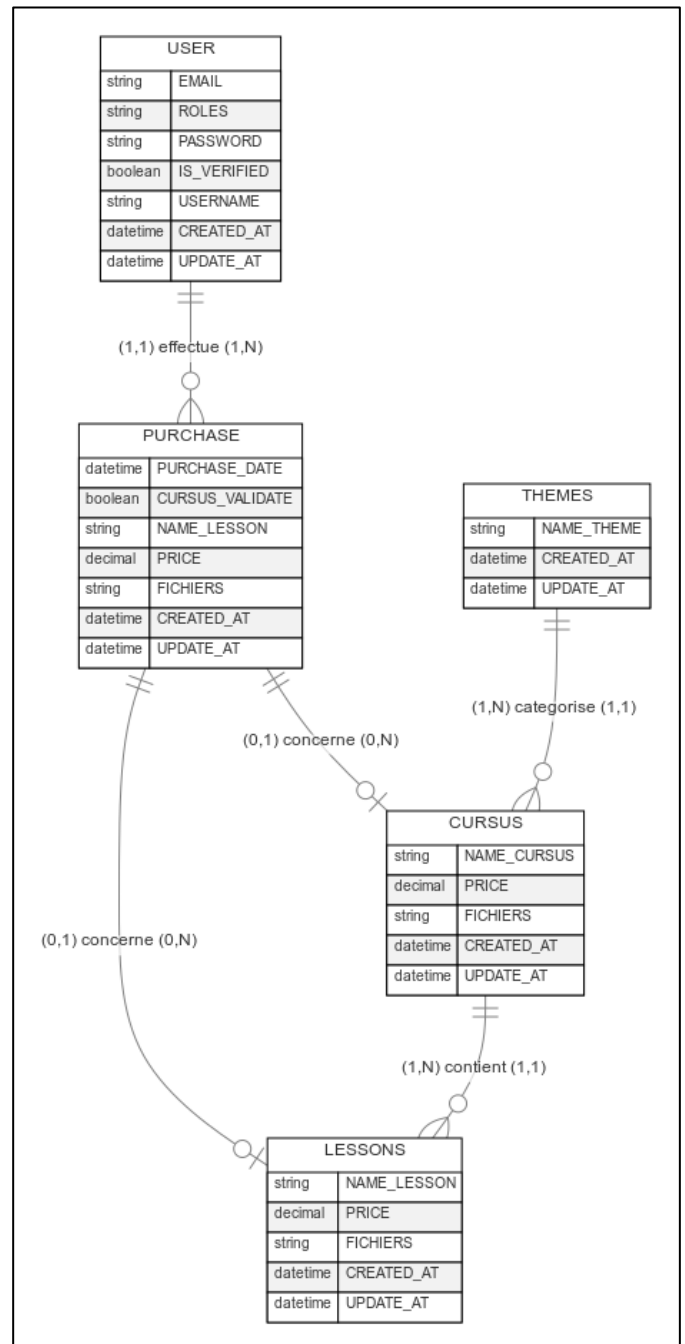
Entre THEMES et CURSUS : un thème catégorise un pour plusieurs cursus et chaque cursus appartient à un thème.

Entre CURSUS et LESSONS : un cursus contient un ou plusieurs leçons et chaque leçon appartient à un cursus.

J'y intègre également des attributs ce qui me permet de définir les caractéristiques de mes entités.

Pour un utilisateur, je renseigne son email, son username et son rôle.

Pour une leçon, son nom, son prix et ces fichiers.



4.1.2 Elaboration du MLD : Transformation du modèle conceptuel

Passons à l'aspect technique, c'est ce que le MLD m'a permis de réaliser. Cette étape transforme le concept métier du MCD en une structure proche de la base de données tout en conservant sa logique. J'intègre les clés primaires (PK) à chaque entité ainsi que les clés étrangères (FK). Les relations sont toujours présentes, mais elles sont plus détaillées.

Dans mon modèle, j'intègre les clés primaires (PK) pour garantir l'unicité de l'enregistrement.

ID dans USER pour identifier chaque utilisateur.

ID dans PURCHASE pour tracer chaque transaction.

ID dans CURSUS et LESSONS pour référencer chaque contenu pédagogique.

J'intègre également les clés étrangères (FK) pour créer les relations

user_id dans PURCHASE pour lier chaque achat à un utilisateur.

cursus_id et lessons_id (NULLABLE) dans PURCHASE pour permettre l'achat d'un cursus et/ou d'une leçon

theme_id dans CURSUS pour catégoriser chaque cursus

cursus_id dans LESSONS pour rattacher chaque leçons à un cursus.

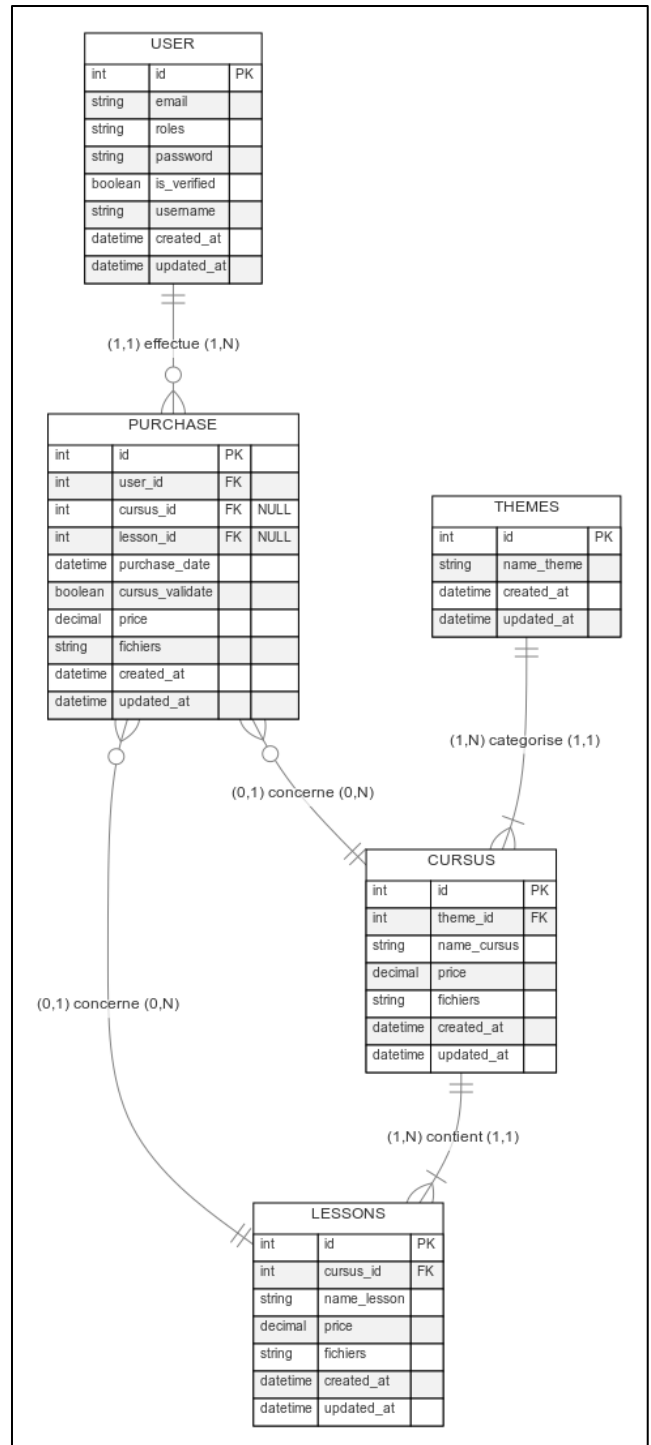
Je précise également les contraintes d'intégrité.

Dans PURCHASE, cursus_id et lesson_id, l'achat concerne soit un cursus soit une leçon.

Chaque leçon doit obligatoirement être rattachée à un cursus.

Chaque cursus doit être rattaché à un thème.

Cette structure logique me permet de garantir la cohérence des données et facilitera l'intégration dans mon SGBD MySQL.



4.1.3 Réalisation du MPD : Implémentation physique de la BDD MySQL

Pour la dernière étape, j'ai réalisé le modèle Physique de données. Cette phase est plus technique.

Il m'a permis de retranscrire les entités, les propriétés, les relations ainsi que les contraintes sous forme de script SQL, ce qui me permet de créer la base de données.

Exemple pour mon projet, je crée ma base de données. Il est important de noter que la commande **DROP DATABASE** est à utiliser avec précaution, car elle supprime la base si elle existe.

```
DROP DATABASE IF EXISTS knowledge;  
CREATE DATABASE knowledge;  
USE knowledge;
```

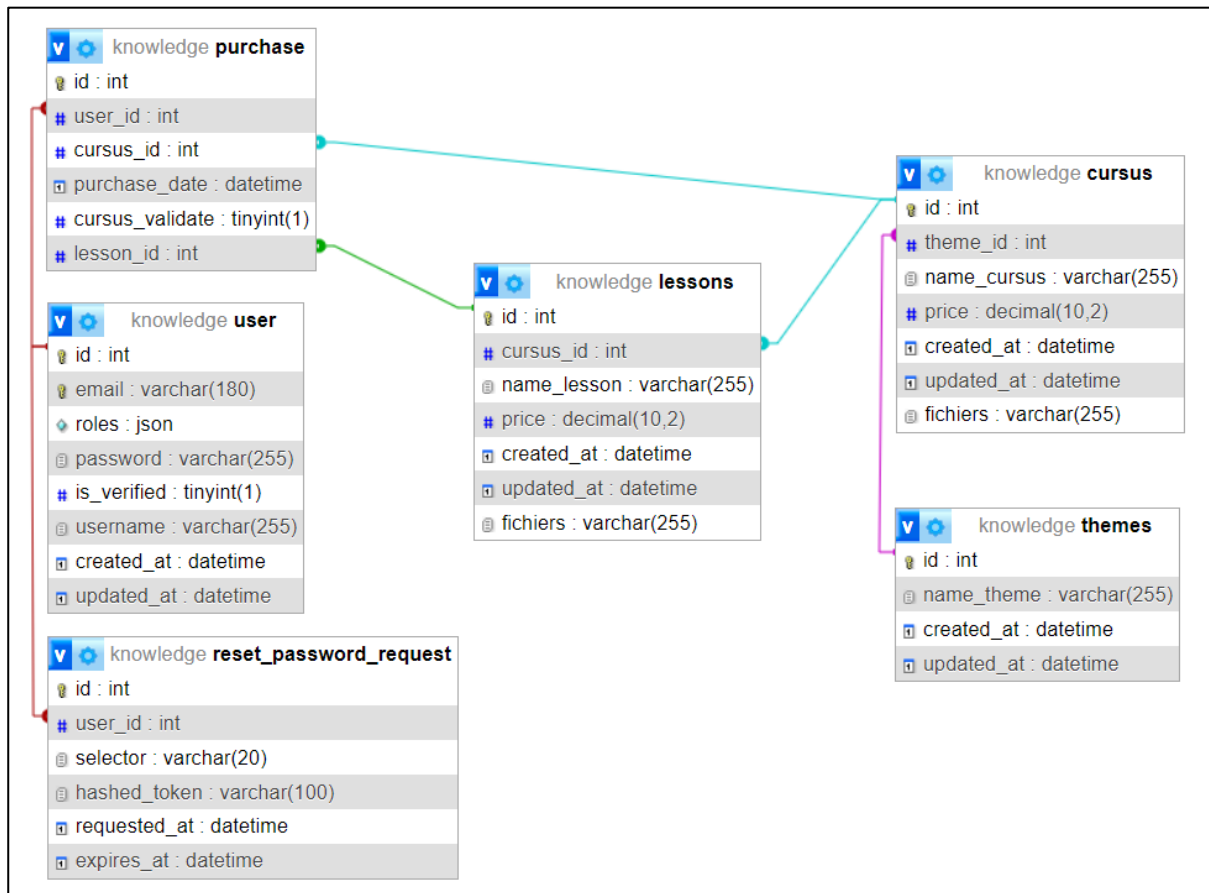
J'ai créé ensuite la table USER, avec ces différentes propriétés, telles que sa clé primaire, un entier qui s'auto-incrémente ou un email qui ne peut avoir de valeur nul.

```
CREATE TABLE `user` (  
  `id` INT PRIMARY KEY AUTO_INCREMENT,  
  `email` VARCHAR(180) NOT NULL UNIQUE,  
  `roles` JSON NOT NULL,  
  `password` VARCHAR(255) NOT NULL,  
  `is_verified` BOOLEAN NOT NULL DEFAULT FALSE,  
  `username` VARCHAR(100) NOT NULL,  
  `created_at` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,  
  `update_at` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP ON  
  UPDATE CURRENT_TIMESTAMP);
```

J'ai également mis en place la table PURCHASE qui contient des éléments, tels que des clés étrangères. Dans mon projet, elles feront le lien entre les entités, dans mon exemple, j'ai une relation One-To-Many entre la table PURCHASE et la table USER. Un utilisateur peut avoir plusieurs achats.

```
CREATE TABLE `purchase` (  
  `id` INT PRIMARY KEY AUTO_INCREMENT,  
  `purchase_date` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,  
  `cursus_validate` BOOLEAN NOT NULL DEFAULT FALSE,  
  `user_id` INT NOT NULL,  
  `lesson_id` INT,  
  `cursus_id` INT,  
  FOREIGN KEY (`user_id`) REFERENCES `user` (`id`),  
  FOREIGN KEY (`lesson_id`) REFERENCES `lessons` (`id`),  
  FOREIGN KEY (`cursus_id`) REFERENCES `cursus` (`id`));
```

Structure des tables dans PhpMyAdmin



Pour visualiser et gérer ma BDD, j'ai utilisé phpMyAdmin.

Cet outil m'a permis de :

- Visualiser la structure de ma BDD (vérifier les relations entre les tables, contrôler les types de données et s'assurer de la mise en place des contraintes d'intégrité).
- Pour mes entités principales (THEME : permet d'organiser les cursus par catégorie, CURSUS : stocker mes parcours de formation, LESSONS : contient toutes les leçons)
- Mes relations sont matérialisées :

THEMES -> CURSUS : Relation OneToMany, elle permet à un thème de contenir plusieurs cursus.

CURUS -> LESSONS : Relation OneToMany, un cursus peut regrouper plusieurs leçons

PURCHASE : L'élément central qui gère les achats, je peux lier :

Les utilisateurs aux contenus achetés.

Les achats aux CURSUS ou LESSONS.

Le suivi des certifications obtenues.

J'ai pu également tester et valider, l'intégrité des données, les contraintes des clés étrangères et la cohérence des relations entre les tables.

4.2. Moyen de paiement en ligne

La solution choisie pour le paiement en ligne a été STRIPE. Il met à disposition un mode développeur ce qui me permet de réaliser des tests en situation réel. Le code mis en place est le suivant.

```
37 #[Route('/checkout', name: 'app_checkout', methods: 'POST')]
38 public function checkout(SessionInterface $session): Response
39 {
40     Stripe::setApiKey($this->getParameter('stripe_secret_key'));
41
42     $items = $session->get('cart', []);
43     $lineItems = [];
44
45     foreach($items as $item){
46         if($item['lesson']){
47             $lineItems[] = [
48                 'price_data' => [
49                     'currency' => 'eur',
50                     'product_data' => [
51                         'name' => $item['lesson']->getNameLesson(),
52                     ],
53                     'unit_amount' => $item['lesson']->getPrice() * 100,
54                 ],
55                 'quantity' => 1,
56             ];
57         }elseif($item['cursus']){
58             $lineItems[] = [
59                 'price_data' => [
60                     'currency' => 'eur',
61                     'product_data' => [
62                         'name' => $item['cursus']->getNameCursus(),
63                     ],
64                     'unit_amount' => $item['cursus']->getPrice() * 100,
65                 ],
66                 'quantity' => 1,
67             ];
68         }
69     }
70
71     $checkout_session = Session::create([
72         'payment_method_types' => ['card'],
73         'line_items' => $lineItems,
74         'mode' => 'payment',
75         'success_url' => $this->generateUrl('payment_success', [], UrlGeneratorInterface::ABSOLUTE_URL),
76         'cancel_url' => $this->generateUrl('payment_cancel', [], UrlGeneratorInterface::ABSOLUTE_URL),
77     ]);
78
79     // Store the checkout session URL in the Symfony session
80     $session->set('checkout_url', $checkout_session->url);
81     // Redirection to intermediate route
82     return $this->redirectToRoute('checkout_redirect');
83 }
84
85 #[Route('/checkout/redirect', name: 'checkout_redirect')]
86 public function checkoutRedirect(SessionInterface $session): Response
87 {
88     // Retrieve the session URL
89     $checkoutUrl = $session->get('checkout_url');
90
91     // Return the template with the JavaScript redirection
92     return $this->render('redirect.html.twig', [
93         'checkout_url' => $checkoutUrl,
94     ]);
95 }
```

Je récupère dans la variable **\$items** tous les cursus et/ou cours présents dans mon panier via la session de l'utilisateur connecté. Ensuite, j'effectue une boucle **Foreach** sur cette collection pour traiter chaque élément, que j'assigne à la variable **\$item** à chaque itération.

Je vérifie ensuite la nature de l'article, s'il s'agit d'un cours ou d'un cursus. En fonction du résultat, je l'ajoute à la variable **\$lineItems** qui stocke les informations à envoyer à STRIPE. Pour chaque article, je renseigne son nom, je convertis son prix en centimes en le multipliant par 100, afin de respecter les recommandations de STRIPE sur les montants. Je fixe également la quantité de l'article à 1 car chaque article est acheté à l'unité.

Je crée ensuite une session de paiement avec STRIPE en utilisant la méthode **Session::create()**. Cette session contient plusieurs éléments importants pour le processus de paiement.

Je définis le type de paiement accepté avec **payment_method_types => ['card']**, ce qui signifie que seuls les paiements par cartes sont autorisés.

J'assigne à **line_items** la variable **\$lineItems** qui stocke les articles (cursus et cours) ajoutés au panier, avec son nom, son prix et sa quantité.

Le mode de paiement est fixé à **payment**, ce qui signifie qu'il s'agit d'un paiement unique. Je définis également 2 URL important pour le retour d'information après le paiement. La **success_url** est l'url vers laquelle l'utilisateur sera redirigé en cas de succès du paiement, alors que la **cancel_url** est celle dans le cas où l'utilisateur annule le paiement ou si celui-ci échoue. Ces URL sont générés automatiquement à l'aide de la méthode **generateUrl()** avec l'option **UrlGeneratorInterface::ABSOLUTE_URL** pour obtenir une URL complètes.

Je développerai cette dernière étape plus en détail dans le prochain chapitre.

4.3. Problème rencontré

Lors de la mise place de cette solution, j'ai rencontré une problématique avec l'URL qui est généré automatiquement par STRIPE. En effet, lors de validation du panier, l'utilisateur doit être redirigé sur leur plateforme afin de pouvoir renseigner ces coordonnées bancaires. Dans cette URL qui est généré, le caractère (**#**) ou fragment tronque l'adresse ce qui fait que ce lien me retourne une page introuvable. Dans le chapitre suivant, je développerais la solution que j'ai apportée.

4.4. Solution apportée

Dans un premier temps, je récupère l'URL que je stocke dans la variable `$session`.

```
// Store the checkout session URL in the Symfony session
$session->set('checkout_url', $checkout_session->url);
// Redirection to intermediate route
return $this->redirectToRoute('checkout_redirect');
```

Je la retourne ensuite dans ma méthode, puis elle est rendue sur ma template `redirect.html.twig`,

```
#[Route('/checkout/redirect', name: 'checkout_redirect')]
public function checkoutRedirect(SessionInterface $session): Response
{
    // Retrieve the session URL
    $checkoutUrl = $session->get('checkout_url');

    // Return the template with the JavaScript redirection
    return $this->render('redirect.html.twig', [
        'checkout_url' => $checkoutUrl,
    ]);
}
```

et enfin dans ma template, je passe par du JavaScript :

- j'écoute l'événement **DOMContentLoaded**.
- je récupère l'URL encodé pour le décoder
- et je retourne sur le navigateur une URL décodé.

```
<p>Redirection vers Stripe, veuillez patienter...</p>
<script type="text/javascript">
    document.addEventListener("DOMContentLoaded", function() {
        //Decode the URL before redirecting
        var encodedUrl = '{{ checkout_url }}';
        var decodedUrl = decodeURIComponent(encodedUrl);
        window.location.href = decodedUrl;
    });
</script>
```

4.5. Développer des composants d'accès aux données SQL et NoSQL

Dans le cadre de cette activité, il est essentiel de savoir comment interagir avec les bases de données que ce soit une base relationnelle (SQL) ou non-relationnelle (NoSQL).

J'ai eu l'occasion de réaliser 2 applications utilisant 2 types de base de données.

- Accès aux données SQL avec Doctrine

Dans ma première application, j'ai choisi une base **MySQL** et j'ai utilisé **Doctrine**, l'ORM de Symfony, qui m'a permis de faire le lien entre les objets PHP et ma base de données. J'ai pu manipuler mes données comme des objets PHP sans écrire de requête SQL, ce qui rend mon code plus maintenable et plus sécurisé. Dans un premier temps, j'ai renseigné dans mon fichier **.env** les informations nécessaires pour accéder à ma base de données. Le fichier est essentiel, car il contient toutes les variables d'environnement nécessaire au bon fonctionnement de l'application.

```
###> doctrine/doctrine-bundle ###
# Format described at https://www.doctrine-project.org/projects/doctrine-dbal/en/latest/reference/configuration.html#connecting-using-a-
# IMPORTANT: You MUST configure your server version, either here or in config/packages/doctrine.yaml
#
DATABASE_URL="mysql://Knowledge:9BiGhq!eTy@127.0.0.1:3306/knowledge?serverVersion=8.0.32&charset=utf8mb4"
###< doctrine/doctrine-bundle ###
```

Ensuite, j'ai créé mes entités, elles correspondent aux tables de ma base de données. Symfony facilite la création d'une entité avec la commande suivante **php bin/console make:entity**

Exemple de l'entité User

Cette commande génère une class avec des propriétés représentant les attributs de ma table.

```

class User implements UserInterface, PasswordAuthenticatedUserInterface
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    private ?int $id = null;

    #[ORM\Column(length: 180)]
    private ?string $email = null;

    /**
     * @var list<string> The user roles
     */
    #[ORM\Column]
    private array $roles = [];

    /**
     * @var string The hashed password
     */
    #[ORM\Column]
    private ?string $password = null;

    #[ORM\Column]
    private bool $isVerified = false;

    #[ORM\Column(length: 255)]
    private ?string $username = null;

    #[ORM\Column(type: Types::DATETIME_MUTABLE)]
    private ?\DateTimeInterface $created_at = null;

    #[ORM\Column(type: Types::DATETIME_MUTABLE)]
    private ?\DateTimeInterface $updated_at = null;
}

```

Les propriétés définissent les champs de la table, et chaque entité est associée à un ensemble de méthodes **Getter** et **Setter** qui me permette de lire et de modifier les données.

```
public function isVerified(): bool
{
    return $this->isVerified;
}

public function setVerified(bool $isVerified): static
{
    $this->isVerified = $isVerified;

    return $this;
}

public function getUsername(): ?string
{
    return $this->username;
}

public function setUsername(string $username): static
{
    $this->username = $username;

    return $this;
}

public function getCreatedAt(): ?\DateTimeInterface
{
    return $this->created_at;
}

public function setCreatedAt(\DateTimeInterface $created_at): static
{
    $this->created_at = $created_at;

    return $this;
}
```

Une fois l'entité créée, j'ai réalisé la migration pour mettre à jour ma base de données. Elle permet de synchroniser les changements dans mes entités.

-Préparation de la migration

php bin/console make:migrations génère le fichier contenant les instructions SQL pour appliquer le changements (ex : ajout de colonnes, modifications de types, etc...)

-Application des changements dans la base de données

php bin/console doctrine:migrations:migrate il exécute le fichier et applique les modifications à ma base de données.

- Accès aux données NoSQL

Pour le type NoSQL, j'ai utilisé MongoDB, et tout comme MySQL, j'utilise mon fichier **.env** pour renseigner les accès à ma base de données, cela me permettra de stocker les données sous forme de document au format JSON.

```
You, il y a 4 mois | 1 author (You)
NODE_ENV=template
APP_NAME=Application PortPlaisance
API_URL=127.0.0.1
URL_MONGO="mongodb+srv://tealc972:CszujSb00GAmG7mh@portplaisance.rdvbad.mongodb.net/?retryWrites=true&w=majority&appName=PortPlaisance"
SECRET_KEY=FyD9dhNRqBykoo      You, il y a 4 mois • ajout de la fonctionnalité de déconnexion
```

Dans mon fichier mongo.js, je configure ma connexion à ma base de données en utilisant Mongoose, qui est une bibliothèque propre à Node.js pour MongoDB.

```
const mongoose = require('mongoose');      You, il y a 9 mois • API de port de plaisance

exports.initClientDbConnection = async () => {
  try {
    /* ATTENTION
    On essaie de se connecté mongoDB en utilisant la variable d'environnement URL_MONGO
    Il faut donc ne pas oublier de l'ajouter au fichier .env
    URL_MONGO prends pour valeur la chaine de connexion de votre cluster mongoDB
    */
    await mongoose.connect(process.env.URL_MONGO)
    console.log('MongoDB Connecté');
  } catch (error) {
    console.log(error);
    throw error;
  }
}
```

Une fois que la connexion est établie, j'utilise des modèles pour interagir avec la base de données. Ces modèles correspondent à la couche M du modèle MVC (Modèle, Vue, Contrôleur). Elle représente une collection qui définit la structure du document (au format JSON).

Dans ce modèle, j'ai pu valider, lire et modifier les données dans MongoDB.

Exemple du modèle pour ma collection User.

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
const bcrypt = require('bcryptjs');

const User = new Schema ({
  name: {
    type: String,
    trim: true,
    require: [true, 'Le nom est requis']
  },
  email: {
    type: String,
    trim: true,
    require: [true, `L'email est requis`]
  },
  password: {
    type: String,
    trim: true
  }
},{
  timestamps: true
});

User.pre('save', function (next) {
  if(!this.isModified('password')) {
    return next();
  }
  this.password = bcrypt.hashSync(this.password, 10);
  next();
});

module.exports = mongoose.model('User', User);
```

Puis dans mon Controller, j'ai mis en place la logique qui me permet d'ajouter un nouvel utilisateur.

Je récupère les données à partir du corps de la requête (**req.body**), elles dépendent du schéma Mongoose que j'ai définie, (name, email, password). Je stocke ces éléments dans une objet temporaire **temp**. La méthode **User.create(temp)** va créer et insérer le nouvel utilisateur dans la base de données MongoDB. Si l'action est un échec ou un succès, je retourne le code 501 pour un échec ou 201 pour un succès.

```
// Ici c'est le callback qui servira à ajouter un user
exports.add = async (req, res) => {
  const temp = ({
    name      : req.body.name,
    email     : req.body.email,
    password  : req.body.password
  });

  try {
    let user = await User.create(temp);
    return res.status(201).json(user);
  } catch (error) {
    return res.status(501).json(error);
  }
}
```

4.6. Développer des composants métier coté serveur

Dans le cadre de ce projet, j'ai développé un site e-commerce permettant aux utilisateurs de créer un compte, se connecter et réaliser des achats en ligne.

La gestion des utilisateurs est un élément essentiel pour garantir la sécurité des transactions et des données personnelles des clients.

Mon projet a été codé sous Symfony, un framework PHP puissant pour la gestion des utilisateurs et des sessions. Pour gérer la création des comptes clients, j'ai commencé par utiliser la commande **php bin/console make:user**. Cette commande me crée une entité utilisateur avec des propriétés comme email, mot de passe, le rôle utilisateur (ex : ROLE_USER, pour les clients). Ces informations sont ensuite stockées dans une base de données MySQL à l'aide de Doctrine, c'est l'ORM intégré de Symfony.

Une fois, l'entité **user** créée, j'ai mis en place un système d'inscription en utilisant la commande **php bin/console make:registration**. Cela m'a permis de créer un formulaire d'inscription qui vérifie les données utilisateur avant de les stocker. Par exemple, il est essentiel que chaque adresse mail soit unique dans la base de données, ce que Symfony vérifie grâce à des contraintes de validation. J'ai aussi intégré un système de confirmation par mail pour m'assurer que l'adresse saisie par l'utilisateur est valide, ce qui est primordial pour éviter les inscriptions frauduleuses. Pour cela, j'ai utilisé le bundle Symfony **symfonycasts/verify-email-bundle**, qui facilite l'envoi d'un lien de confirmation par mail.

```

{{ form_errors(registrationForm) }}
<div class="d-flex justify-content-center align-items-center">
  <div class="card card_register" style="width: 18rem;">
    {{ form_start(registrationForm) }}
    <div class="card-body">
      <p class="card-title">S'inscrire</p>
      <hr>
      <div>
        {{ form_label(registrationForm.username, "Nom de l'utilisateur :", {'label_attr':{'class': 'form-label'}}) }}
        {{ form_widget(registrationForm.username, {'attr': {'class': 'form-control'}}) }}

        {{ form_label(registrationForm.email, "Adresse mail :", {'label_attr':{'class': 'form-label'}}) }}
        {{ form_widget(registrationForm.email, {'attr': {'class': 'form-control'}}) }}

        {{ form_label(registrationForm.deliveryAddress, "Adresse de livraison :", {'label_attr':{'class': 'form-label'}}) }}
        {{ form_widget(registrationForm.deliveryAddress, {'attr': {'class': 'form-control'}}) }}

        {{ form_label(registrationForm.plainPassword.first, "Mot de passe :", {'label_attr':{'class': 'form-label'}}) }}
        {{ form_widget(registrationForm.plainPassword.first, {'attr': {'class': 'form-control'}}) }}

        {{ form_label(registrationForm.plainPassword.second, "Confirmer le mot de passe :", {'label_attr':{'class': 'form-label'}}) }}
        {{ form_widget(registrationForm.plainPassword.second, {'attr': {'class': 'form-control'}}) }}
      </div>
      <div class="card-body">
        <button type="submit" class="btn btn-primary">Créer un compte</button>
      </div>
    {{ form_end(registrationForm) }}
    <hr>
    <p class="card-title">Déjà inscrit ?</p>
    <div class="card-body">
      <a href="/login" class="btn btn-primary">Se connecter</a>
    </div>
  </div>
</div>

```

En plus de l'inscription, j'ai configuré l'authentification des utilisateurs avec la commande **php bin/console make:auth**. Elle m'a permis de générer un contrôleur de connexion, un formulaire de login, et un **authentificateur** pour gérer la sécurité des utilisateurs. Symfony prend en charge l'encodage sécurisé des mots de passe avec un algorithme comme **bcrypt**, garantissant que les données sensibles ne sont jamais stockées en clair. Une fois connecté, l'utilisateur peut naviguer sur le site et accéder à son panier ou à son historique de commandes. La gestion de la session utilisateur est assurée par Symfony, permettant de maintenir une connexion sécurisée pendant un temps défini.

Enfin, j'ai mis en place une route pour permettre à l'utilisateur de se déconnecter en toute sécurité grâce à la fonctionnalité **logout** de Symfony, qui détruit la session active.

Grâce à ces outils, j'ai pu assurer la sécurité et la fiabilité de la gestion des utilisateurs, garantissant ainsi une expérience fluide et sécurisée pour mes clients.

4.7. Documenter le déploiement d'une application web ou web mobile

Le dernier défi d'un projet de développement est son déploiement. Pour cela, j'ai choisi d'héberger mon application sur Hostinger. J'ai opté pour cet hébergeur, car il offre des services variés pour les particuliers, les petites entreprises et les développeurs. Elle propose des tarifs compétitifs de l'ordre de quelques euros par mois. Il met à disposition différents types d'hébergement (mutualisé, VPS, cloud, WordPress). Les serveurs mis à disposition par Hostinger utilisent des disques durs SSD ce qui améliore les temps de chargement des pages ce qui est essentiel pour l'expérience utilisateur.

J'ai commencé par transférer l'intégralité de mon application via FTP, cette action a été réalisée avec un client FTP (WinSCP), une fois le transfert terminé, j'ai utilisé le Terminal d'Hostinger, j'ai réinstallé les dépendances avec la **commande `composer install --no-dev`** ce qui permet d'optimiser mon application en supprimant les ressources inutiles et d'améliorer la sécurité.

J'ai mentionné également que mon application devra être exécutée en mode production pour cela dans mon fichier **.env**, je renseigne la variable suivante **APP_ENV=prod**, elle me permet de renforcer la sécurité en désactivant certains outils de développement.

J'ai ensuite vérifié et adapté le fichier **.htaccess** situé dans le répertoire **public/**. Il permet de rediriger toutes les requêtes vers le fichier gérant ainsi toutes les redirections de mon application.

Par défaut, un site est accessible en HTTP. Pour des raisons de sécurité, j'ai choisi de sécuriser mon site avec HTTPS en utilisant **Let's Encrypt**, j'ai installé pour cela **CertBot** en exécutant la commande **`sudo apt install certbot python3-certbot-apache`**, puis j'ai lancé la commande **`sudo certbot --apache`** pour obtenir un certificat SSL/TLS gratuit. Cela me garantit une communication sécurisée entre les utilisateurs et le serveur.

4.8. Le versionning (Git et GitHub)

Dans mes différents projets, j'ai utilisé Git pour gérer le versionning de mon code, et GitHub pour l'hébergement de mes dépôts. Ce qui facilite également la collaboration avec d'autres développeurs.

- Git – gestion local

Git est un outil qui m'a permis de suivre les modifications que j'ai apporté à mon code tout au long de mon projet. Voici les principales actions que j'ai réalisées.

J'initialise mon projet avec **git init**, cela me crée un sous-dossier dans lequel j'aurai tous les fichiers nécessaires pour mes dépôts.

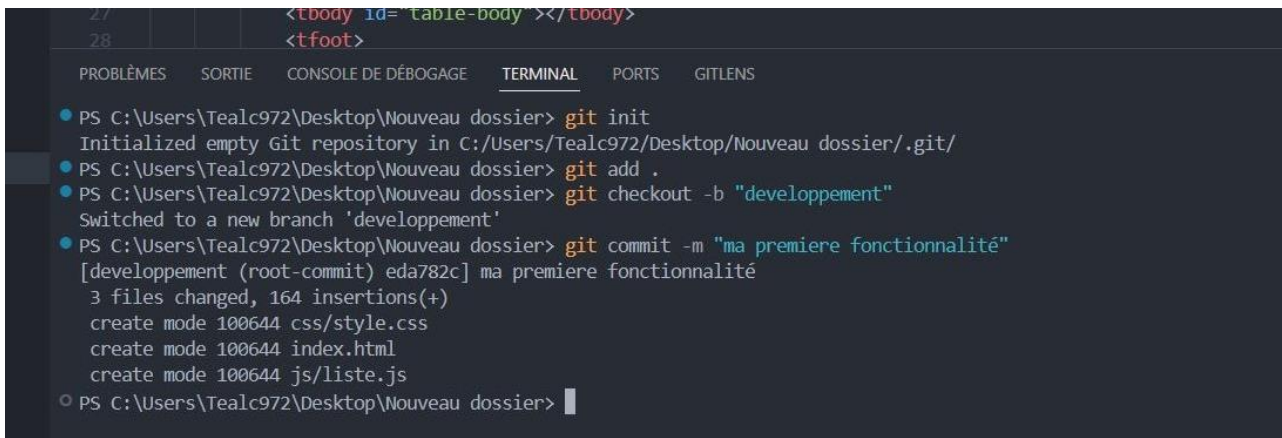
J'ajoute les modifications de mon code dans un index du Git grâce à **git add** .

J'enregistre les changements dans un historique avec la commande **git commit -m « Description de la modification »**. Cela permet de savoir exactement sur quoi porte la modification.

Je crée une nouvelle branche, car en développement la branche main ou master est utilisé pour héberger le code final, stable et prêt à être déployé en production. Elle permet

d'isoler les modifications, ce qui rend possible le travail sur une nouvelle fonctionnalité et de corriger mes bugs sans affecter le code principal. La nouvelle branche est créée avec la commande **git checkout -b « nom de la branche »**.

Etant en local, il ne me reste plus qu'à fusionner mon code sur ma branche avec **git merge**.



```
27 <tbody id="table-body"></tbody>
28 <tfoot>

PROBLÈMES  SORTIE  CONSOLE DE DÉBOGAGE  TERMINAL  PORTS  GITLENS

• PS C:\Users\Tealc972\Desktop\Nouveau dossier> git init
  Initialized empty Git repository in C:/Users/Tealc972/Desktop/Nouveau dossier/.git/
• PS C:\Users\Tealc972\Desktop\Nouveau dossier> git add .
• PS C:\Users\Tealc972\Desktop\Nouveau dossier> git checkout -b "developpement"
  Switched to a new branch 'developpement'
• PS C:\Users\Tealc972\Desktop\Nouveau dossier> git commit -m "ma premiere fonctionnalité"
  [developpement (root-commit) eda782c] ma premiere fonctionnalité
   3 files changed, 164 insertions(+)
   create mode 100644 css/style.css
   create mode 100644 index.html
   create mode 100644 js/liste.js
○ PS C:\Users\Tealc972\Desktop\Nouveau dossier> |
```

- GitHub - Hébergement

GitHub est une plateforme qui héberge tous les dépôts Git et facilite la collaboration avec d'autres développeurs.

Je l'utilise pour le stockage distant, je peux créer des dépôts privé ou publics accessible de n'importe où. Je peux également réaliser une sauvegarde de mon projet avec la commande **git push origin « nom de la branche »**.

GitHub me permet également de créer des **issues**, c'est une sorte de ticket rattaché à mon dépôt qui me permet de suivre l'avancer de mon projet. Je peux créer une issue pour chaque fonctionnalité que je veux rajouter. Documenter les bugs rencontrés tout au long du projet ou planifier les différentes tâches à réaliser.

En conclusion, ces outils mon permis de structurer et d'organiser mes projets de manière professionnelle et de travailler en toute sécurité sur plusieurs fonctionnalités.



Knowledge

Public

Pin

Unwatch 1



Your master branch isn't protected

Protect this branch from force pushing or deletion, or require status checks before merging. [View documentation.](#)

Protect this branch



master

3 Branches 0 Tags

Go to file



Add file

Code

Switch branches/tags



Find or create a branch...

Branches

Tags

✓ master

default

dev_cours

dev

View all branches

coning

migrations

public

src

templates

b1683e8 · 5 months ago

48 Commits

deploiement auto correction bug

5 months ago

Add files via upload

6 months ago

deploiement automatique

5 months ago

Add files via upload

6 months ago

Add webapp packages

7 months ago

correction de bug, erreur 500 sur le serveur, ajout de fichier f...

6 months ago

correction de bug, erreur 500 sur le serveur, ajout de fichier f...

6 months ago

correction de bug, erreur 500 sur le serveur, ajout de fichier f...

6 months ago

correction de bug, erreur 500 sur le serveur, ajout de fichier f...

6 months ago

deploiement automatique

5 months ago