

Présentation de projet

**Soutenance Titre Professionnel
Développeur Web & Web mobile**
Présenté par Yoann Lecavelier



SOMMAIRE

- 1. CONTEXTE DU PROJET**
- 2. RÉALISATION DU CÔTÉ FRONT-END**
- 3. RÉALISATION DU CÔTÉ BACK-END**
- 4. PRÉSENTATION DES TECHNOLOGIES**
- 5. PRÉSENTATION D'UN ACHAT D'ESSAI**
- 6. CONCLUSION**



CONTEXTE DU PROJET

POURQUOI FAIRE UN SITE E-LEARNING?

FONCTIONNALITÉES CLIENT ?

FONCTIONNALITÉES À PRÉVOIR ?

RÉALISATION DU CÔTÉ FRONT-END

FONCTIONNALITÉES À PRÉVOIR ?

CONDITIONS À RESPECTER ?

CHARTE GRAPHIQUE ?

SCHÉMA DES FONCTIONNALITÉS

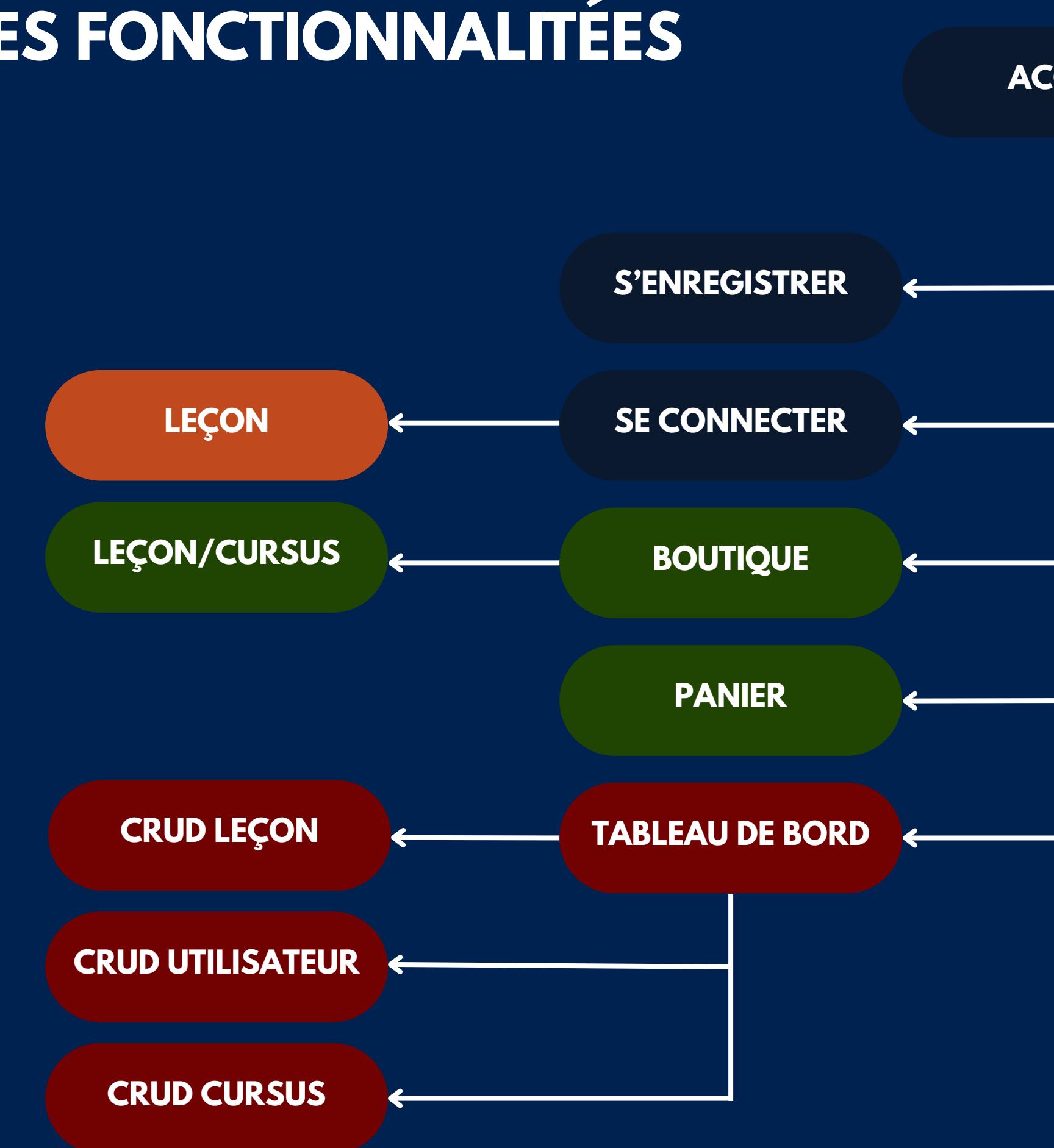
ACCEUIL

Accès à un utilisateur non connecté(e)

Accès à un utilisateur connecté(e)

Accès à un utilisateur administrateur

Accès à un utilisateur ayant effectué un achat



PAGE “ACCEUIL”

Vue bureau

The screenshot shows a desktop view of the Knowledge website. At the top, there is a navigation bar with links for "Accueil", "Boutique", "Panier", and "Se déconnecter". Below the navigation bar is the website's logo, which features a stylized book icon above the word "KNOWLEDGE". A message in French encourages users to create an account and verify their email to access services. At the bottom left, there are two links: "J'ai un compte" and "Je m'inscris".

Vue mobile

The screenshot shows a mobile view of the Knowledge website. The layout is similar to the desktop version, featuring the logo at the top right and a navigation menu icon (three horizontal lines) at the top right. A message in French encourages users to create an account and verify their email. At the bottom left, there are two links: "J'ai un compte" and "Je m'inscris".

CONTIENT :

- Connexion
- Inscription

ACCESSIBILITÉ:

- Utilisateur non identifié
- Etudiant
- Administrateur
- Utilisateur identifié

PAGE “ACCEUIL”

TEMPLATE

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <link rel="stylesheet" href="{{ asset('app.css') }}>
    <link rel="icon" type="image/png" href="{{ asset('favicon.png') }}>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Home</title>
  </head>
  <body>
    <div class="nav">
      <nav>
        <div class="navlinks">
          <ul class="navbar">
            <li><a href="/">Accueil</a></li>
            {% if app.user %}
              <li><a href="/formations">Boutique</a></li>
              <li><a href="/cart">Panier</a></li>
            {% if is_granted('ROLE_STUDENT') %}
              <li><a href="{{ path('app_student') }}>Élèves</a></li>
            {% endif %}
              <li><a href="{{ path('app_logout') }}>Se déconnecter</a></li>
            {% if is_granted('ROLE_ADMIN') %}
              <li><a href="{{ path('app_admin') }}>Tableau de bord</a></li>
            {% endif %}
            {% else %}
              <li><a class="active" href="/register">S'inscrire</a></li>
              <li><a href="/login">Se connecter</a></li>
            {% endif %}
            </ul>
          </div>
          
        </nav>
        
      </div>
      <br />
    </div>
    <div style="background-color: #f1f8fc; padding: 20px; text-align: center;">
      <h1 style="font-size: 15px; font-weight: 200;">Pour accéder à nos services, pensez à créer un compte et vérifier votre mail. <br/> Dans le cas contraire, l'accès à nos services vous sera refusé.</h1>
      <a href="/login" style="font-weight: bold;">J'ai un compte</a> <br/> <a href="/register" style="font-weight: bold;">Je m'inscris</a>
    </div>
    </div>
    <script>
      const menuHamburger = document.querySelector(".menu-hamburger");
      const navLinks = document.querySelector(".navlinks");

      menuHamburger.addEventListener("click", () => {
        navLinks.classList.toggle("mobile-menu");
      });
    </script>
  </body>
</html>
```

CSS

```
* {
  font-family: "Comic Sans MS";
}

body {
  display: table;
  width: 99%;
  height: 99%;
  background-color: #f1f8fc;
}

.nav {
  border-bottom: 1px solid #0074c7;
}
.navbar {
  list-style: none;
  padding: 0;
  margin: 0;
  display: flex;
  justify-content: flex-end;
}
.navbar li {
  padding: 10px 20px;
}

.menu-hamburger {
  display: none;
  width: 50px;
  position: absolute;
  top: 35px;
  right: 15px;
}

@media screen and (max-width: 360px) {
  *
  width: 90%;
}
body {
  width: 359px;
  height: 600px;
}
.navbar {
  padding: 0;
}
.navlinks {
  top: 0;
  left: 0;
  position: absolute;
  background-color: #0074c7a;
  backdrop-filter: blur(7px);
  width: 100px;
  height: 100vh;
  display: flex;
  justify-content: center;
  align-items: center;
  margin-left: -100px;
  transition: all 0.5s ease;
}
.navlinks ul {
  display: flex;
  flex-direction: column;
  text-align: center;
}
.navlinks.mobile-menu {
  margin-left: 0;
}
.navbarLogo {
  width: 150px;
}
.menu-hamburger {
  display: block;
}

.navlinks ul li {
  margin: 25px 0;
  font-size: 20px;
}
```

CONTIENT:
Le code pour la navbar
Lien hypertexte avec un titre.
Script pour la navbar

-

PAGE “INSCRIPTION”

Vue bureau

The screenshot shows a desktop browser window with a white header containing the KNOWLEDGE logo and navigation links for Accueil, S'inscrire, and Se connecter. Below the header is a large, light gray rectangular input field labeled "S'inscrire". Inside this field are four input boxes: "Nom d'utilisateur", "Adresse email", "Mot de passe", and "Confirmer le mot de passe". Below these is a blue "Register" button. At the bottom of the input field is a link "Déjà inscrit ?" followed by a blue "Se connecter" button.

Vue mobile

The screenshot shows a mobile device displaying the registration page. The top bar includes the KNOWLEDGE logo and a three-line menu icon. The main content area is identical to the desktop version, featuring the "S'inscrire" input field with its four required fields and "Register" button, along with the "Déjà inscrit ?" link and "Se connecter" button at the bottom.

CONTIENT:

- Formulaire d’inscription
- Connexion

ACCESIBILITÉ:

- Utilisateur non identifié

PAGE “INSCRIPTION”

TEMPLATE

CSS

```
● ● ●  


S'inscrire

<{{ form_errors(registrationForm) }}  
{{ form_start(registrationForm) }}  


<{{ form_row(registrationForm.username, {  
    label: "Nom d'utilisateur"  
})}}  
{{ form_row(registrationForm.email) }}  


<{{ form_row(registrationForm.password.first) }}  
{{ form_row(registrationForm.password.second) }}  
Register  


Déjà inscrit ?

Se connecter<{{ form_end(registrationForm) }}  
}}


```

```
● ● ●  
.title-register {  
    color: grey;  
    border-bottom: 1px solid #0074c7;  
}  
  
.form-register {  
    padding: 10px;  
    display: grid;  
    border: 1px solid #384050;  
    text-align: start;  
    height: 65px;  
    width: 250px;  
}  
  
.register_input {  
    padding: 10px;  
    color: #00497c;  
}  
.btn-register {  
    margin: 10px;  
    border: 0px;  
    width: 150px;  
    height: 30px;  
    background-color: #0074c7;  
    color: #fff;  
    border-radius: 5px;  
}
```

CONTIENT:
Input pour le formulaire d'inscription
Lien vers la page de connexion.

-
-

PAGE “CONNEXION”

Vue bureau

Accueil S'inscrire Se connecter

KNOWLEDGE

Se connecter

Email
email

Mot de passe
mot de passe

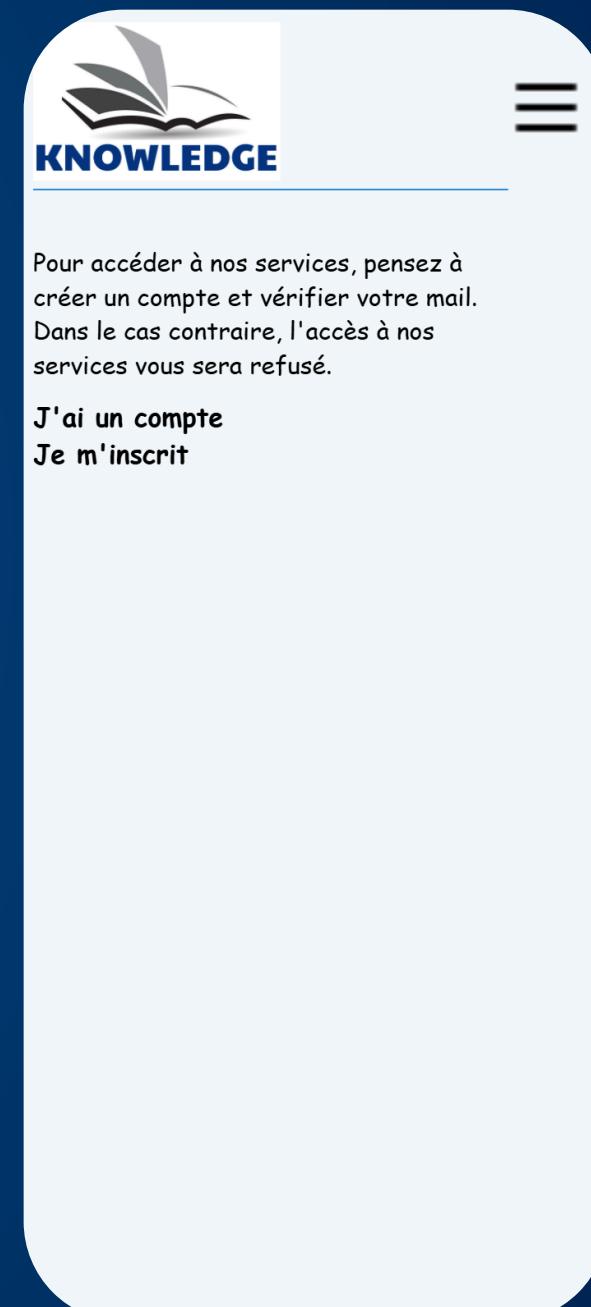
Se connecter

Mot de passe oublié

Pas encore inscrit ?

Créer un compte

Vue mobile



CONTIENT:

- **Formulaire de connexion**
- **Inscription**

ACCESSIBILITÉ:

- **Utilisateur non identifié**
- **Etudiant**
- **Administrateur**
- **Utilisateur identifié**

PAGE “CONNEXION”

TEMPLATE

```
● ● ●
<div class="form">
  <div class="form-login">
    {# block body %}
    <form method="post">
      {# if error %}
      <div class="alert alert-danger">
        {{ error.messageKey|trans(error.messageData, 'security') }}
      </div>
      {# endif %} {# if app.user %}
      <div class="mb-3">
        You are logged in as {{ app.user.userIdentifier }},<br>
        <a href="{{ path('app_logout') }}>Logout</a>
      </div>
      {# endif %}
      <p class="title-form">Se connecter</p>
      <label for="inputEmail">Email</label> <br>
      <input
        type="email"
        value="{{ last_username }}"
        name="email"
        id="inputEmail"
        class="form-control"
        autocomplete="email"
        required
        autofocus
        placeholder="email"
      />
      <br />
      <label for="inputPassword">Mot de passe</label> <br>
      <input
        type="password"
        name="password"
        id="inputPassword"
        class="form-control"
        autocomplete="current-password"
        required
      />
      <input
        type="hidden"
        name="csrf_token"
        value="{{ csrf_token('authenticate') }}"
      />
      <br>
      <button class="btn-login" type="submit">Se connecter</button> <br>
      <a href="/passwordrecovery">
        Mot de passe oublié</a>
      <p class="title-form" style="color:cadetblue">Mot de passe oublié</p><a href="/account_create">
        Pas encore inscrit ?</a>
      <button class="btn-login">Créer un compte</button>
      </div>
    </div>
  {# endblock %}
</div>
```

CSS

```
● ● ●
/*Formulaire Login*/
.title-form {
  color: grey;
  border-bottom: 1px solid #0074c7;
}
.btn-login {
  margin: 10px;
  border: 0px;
  width: 150px;
  height: 30px;
  background-color: #0074c7;
  color: #fff;
  border-radius: 5px;
}
.form-recovery {
  color: grey;
  border-bottom: 1px solid #0074c7;
}
.form {
  display: flex;
  align-items: center;
  margin: 0 auto;
  padding-top: 50px;
  width: 250px;
}
.form-login {
  display: flex;
  border: 1px solid #384050;
  text-align: start;
  height: 350px;
  width: 220px;
  padding: 10px;
}
```

CONTIENT:

Appel à l'entité User (pour afficher à l'utilisateur qu'il est déjà connecté si cela est le cas)

Input pour le formulaire de connexion

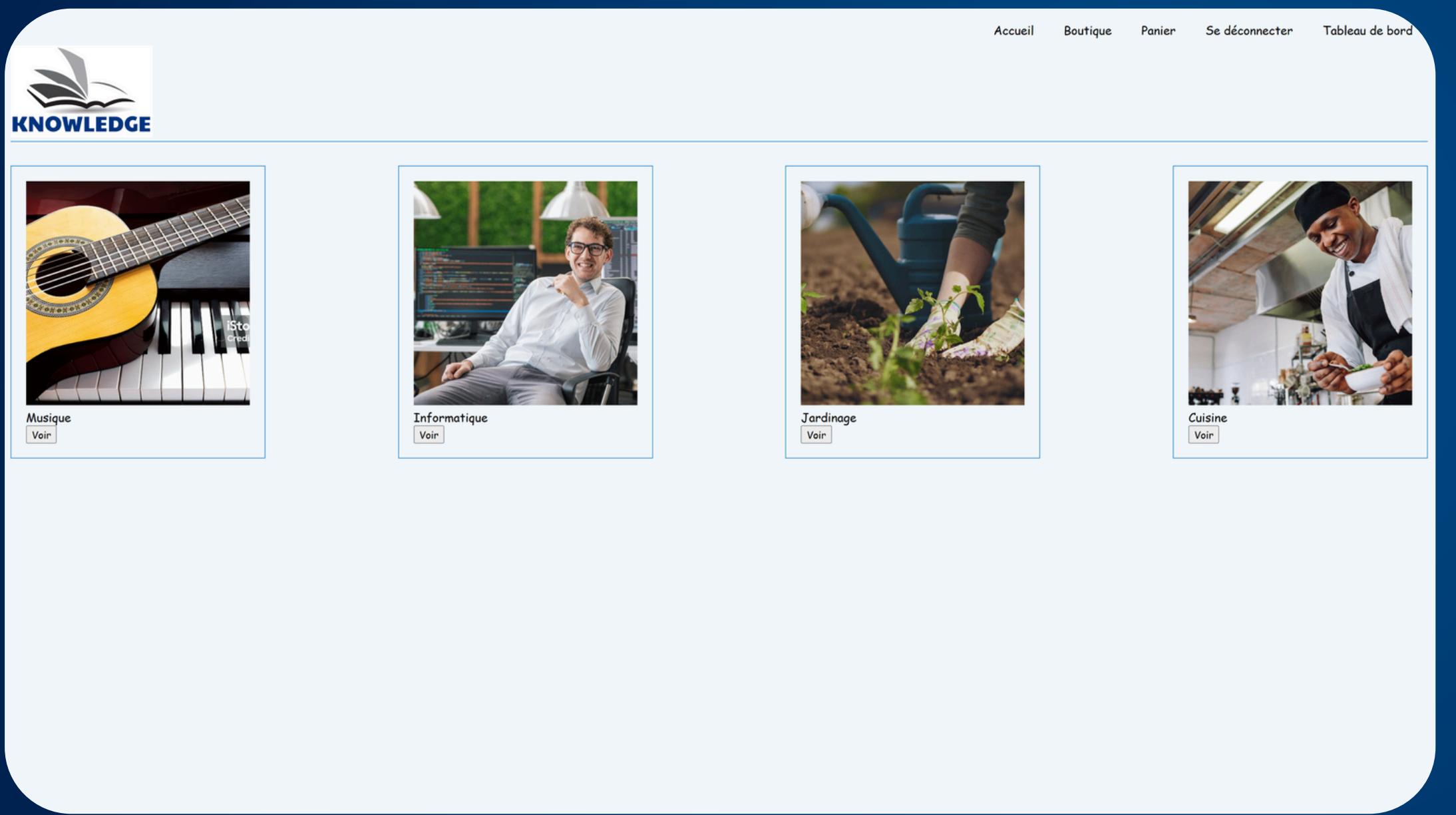
Token CSRF (Sinon le login ne marche pas)

Lien vers la page d'inscription .

-
-
-

PAGE “FORMATIONS”

Vue bureau



Vue mobile



CONTIENT :
• **Toutes les formations.**

ACCESSIBILITÉ:

- **Etudiant**
- **Administrateur**
- **Utilisateur identifié**

PAGE “FORMATIONS”

TEMPLATE

```
<section>
  <div class="show_product2">
    <div class="product_card">
      {% for formation in formation %} {% if formation.id == 1 %}
      <tr>
        <td>
        <br />
        <td>{{ formation.title }}</td>
        {% if app.user %}
          <button class="show_product"><a href="{{ path('formation_show', {'id': formation.id}) }}>Voir</a>
        {% endif %}
        <br />
      </tr>
      {% endif %}
      {% endfor %}
      <br>
    </div>
    <div class="product_card">
      {% for formation in formation %} {% if formation.id == 2 %}
      <tr>
        <td>
        <br />
        <td>{{ formation.title }}</td>
        {% if app.user %}
          <button class="show_product"><a href="{{ path('formation_show', {'id': formation.id}) }}>Voir</a>
        {% endif %}
        <br />
      </tr>
      {% endif %}
      {% endfor %}
      <br>
    </div>
    <div class="product_card">
      {% for formation in formation %} {% if formation.id == 3 %}
      <tr>
        <td>
        <br />
        <td>{{ formation.title }}</td>
        {% if app.user %}
          <button class="show_product"><a href="{{ path('formation_show', {'id': formation.id}) }}>Voir</a>
        {% endif %}
        <br />
      </tr>
      {% endif %}
      {% endfor %}
      <br>
    </div>
    <div class="product_card">
      {% for formation in formation %} {% if formation.id == 4 %}
      <tr>
        <td>
        <br />
        <td>{{ formation.title }}</td>
        {% if app.user %}
          <button class="show_product"><a href="{{ path('formation_show', {'id': formation.id}) }}>Voir</a>
        {% endif %}
        <br />
      </tr>
      {% endif %}
      {% endfor %}
      <br>
    </div>
  </div>
</section>
```

CSS

```
/*Showing formation page*/
.showing_card {
  display: flex;
  justify-content: center;
  height: 500px;
}

.details_card {
  display: flex;
  border: 1px solid #384050;
  height: 600px;
  width: 1200px;
}

.card_img {
  padding: 10px;
  display: flex;
  height: 250px;
  width: 250px;
}

.product_details {
  display: flex;
  height: 600px;
  width: 500px;
  justify-content: center;
}

.add_to_card {
  width: 200px;
  height: 40px;
  border-radius: 5px;
  border: 1px solid #384050;
  margin-top: 20px;
  margin-right: 20px;
  justify-content: end;
}
```

CONTIENT:

- Appel à l’entité “Formation” afin de récupérer les informations sur les formations (Titre, Imagine,Prix ...)
- Possibilité de voir la page détails pour les utilisateur connecté.

PAGE “DÉTAILS DE LA FORMATIONS”

Vue bureau

The screenshot shows a desktop browser window with the 'KNOWLEDGE' logo at the top left. The main content area displays a course titled 'Informatique'. It includes a thumbnail image of a smiling person sitting at a desk with a computer monitor, a 'Retour aux formations' link, and sections for 'Leçons de cette formation' and 'Cursus de cette formation'. Each section lists items with price information and 'Ajouter la leçon au panier' or 'Ajouter le cursus au panier' buttons.

The screenshot shows a mobile device displaying the 'KNOWLEDGE' platform. The top navigation bar includes a logo, a menu icon, and links for 'S'inscrire', 'Nom d'utilisateur', 'Adresse email', 'Mot de passe', 'Confirmer le mot de passe', and 'Register'. Below this is a 'Déjà inscrit?' link and a 'Se connecter' button.

CONTIENT :

- **Leçon de la formation**
- **Cursus de la formation**
- **Détails (Prix, titre...)**

ACCESIBILITÉ:

- **Etudiant**
- **Administrateur**
- **Utilisateur identifié**

PAGE “DÉTAILS DE LA FORMATIONS”

TEMPLATE

```
<div class="showing_card">
  {% if formation.id %}
    <div class="details_card">
      <div class="card_img">
        
      </div>
      <br />
      <div class="product_details">
        <div class="center_details">
          <div>
            <p class="details">{{ formation.title }}</p>
            <a href="/formation"><strong>Retour aux formations</strong></a>
            <br />
            <br />
            {% if lessons is defined and lessons|length > 0 %}
              <h3>Leçons de cette formation :</h3>
              <ul>
                {% for lesson in lessons %}
                  <li>{{ lesson.title }} : {{ lesson.price }} €</li>
                {% endfor %}
              </ul>
              <form method="get" action="{{ path('app_cart_add', { id: lesson.id }) }}"\>
                <button class="add_to_card">Ajouter la leçon au panier</button>
              </form>
            {% else %}
              <p>Aucune leçon disponible pour cette formation.</p>
            {% endif %}

            {% if cursus is defined and cursus|length > 0 %}
              <h3>Cursus de cette formation :</h3>
              <ul>
                {% for cursus in cursus %}
                  <li>{{ cursus.title }} : {{ cursus.price }} €</li>
                {% endfor %}
              </ul>
              <form method="get" action="{{ path('app_cart_add', { id: cursus.id }) }}"\>
                <button class="add_to_card">Ajouter le cursus au panier</button>
              </form>
            {% else %}
              <p>Aucun cursus disponible pour cette formation.</p>
            {% endif %}
            </div>
            <div class="select_details">
              <form method="get" action="{{ path('app_cart_add', { id: formation.id }) }}"\>
                <button class="add_to_card">Ajouter la formation entière au panier</button>
              </form>
            </div>
          </div>
        </div>
      </div>
    </div>
  {% else %}
    <p>Pas de formation pour le moment</p>
  {% endif %}
</div>
```

CSS

BUREAU

```
/*Showing formation page*/
.showing_card {
  display: flex;
  justify-content: center;
  height: 500px;
}

.details_card {
  display: flex;
  border: 1px solid #384050;
  height: 600px;
  width: 1200px;
}

.card_img {
  padding: 10px;
  display: flex;
  height: 250px;
  width: 250px;
}

.product_details {
  display: flex;
  height: 600px;
  width: 500px;
  justify-content: center;
}

.add_to_card {
  width: 200px;
  height: 40px;
  border-radius: 5px;
  border: 1px solid #384050;
  margin-top: 20px;
  margin-right: 200px;
  justify-content: end;
}
```

MOBILE

```
/* Mobile view for details formation page*/
section {
  display: grid;
  height: 100%;
  width: 100%;
}
.show_product2 {
  width: 100%;
  gap: 10px;
  flex-direction: column;
  border-radius: 10px;
  border-color: #384050;
  padding-top: 10px;
}

.showing_card {
  margin: 0;
  padding: 0;
  width: 360px;
}

.details_card {
  margin: 0;
  flex-direction: column;
  width: 360px;
  height: auto;
  border: none;
  border-top: 1px solid #384050;
}

.card_img {
  margin: 0;
  width: 50px;
  height: auto;
  justify-content: center;
  padding: 10px 0;
}

.card_img img {
  margin: 0;
  max-width: 100%;
  max-height: 200px;
}

.center_details {
  margin: 0;
  padding: 0;
  width: 200px;
}

.center_details > div {
  padding: 0;
  margin: 0;
}

.add_to_card {
  width: 100%;
  font-size: 13px;
}

ul {
  padding-left: 15px;
  width: 100%;
  margin-bottom: 10px;
}

li {
  margin-bottom: 10px;
  width: 100%;
}

form {
  width: 100%;
}

.details {
  font-size: 18px;
  font-weight: bold;
  margin-bottom: 10px;
}

.product_details {
  height: 600px;
  width: 360px;
}
```

CONTIENT:

- Appel à l'entité “Formation” afin de récupérer les informations sur les formations (Titre, Imagine,Prix ...)
- Appel des entités (Cursus,Leçon,Formation) afin de pouvoir les ajouter au panier.

PAGE “PANIER”

Vue bureau

The screenshot shows a desktop browser window with the 'KNOWLEDGE' logo at the top left. The top navigation bar includes links for Accueil, Boutique, Panier, Se déconnecter, and Tableau de bord. The main content area is titled 'Mon Panier' and features a product card for 'Leçon n°1 : Les langages Html et CSS'. The card includes a small thumbnail image of a person, the product name, its type ('technology'), its price ('Prix : 32 €'), and two buttons: 'Retirer du panier' and 'FINALISER MA COMMANDE'. A large red circle highlights the 'FINALISER MA COMMANDE' button.

Vue mobile

The screenshot shows a mobile phone displaying the 'Mon Panier' page from the Knowledge website. The layout is similar to the desktop version, showing the product card for 'Leçon n°1 : Les langages Html et CSS'. The 'FINALISER MA COMMANDE' button is again highlighted with a large red circle.

CONTIENT :

- DÉTAILS DES ARTICLES AJOUTÉES
(PRIX, NOM, ETC..)

ACCESIBILITÉ:

- Etudiant
- Administrateur
- Utilisateur identifié

PAGE “PANIER”

TEMPLATE

```
<h1>Mon Panier</h1>

<div class="form-container">
    {% for item in cart %}
        <div class="bloc_cart">
            
            <h2 class="form-title">{{ item.lesson.title }}</h2>
            <p>Type : {{ item.lesson.type }}</p>
            <p>Prix : {{ item.lesson.price }} €</p>
            <div class="btn_cart">
                <a
                    class="btn_delete_cart"
                    href="{{ path('app_cart_remove', { id: item.lesson.id }) }}"
                >Retirer du panier</a>
            </div>
        {% else %}</div>
    {% endfor %}
    <p>Votre panier est vide.</p>
<div class="cart_finish">
    <div class="cart_choice">
        <div class="total_price">
            <h2 class="price">
                TOTAL : {{ totalPrice|number_format(2, ',', '') }} €
            </h2>
            <button id="checkout-button" class="btn_pay">
                FINALISER MA COMMANDE
            </button>
            {% block javascripts %}
            <script type="text/javascript">
                var stripe = Stripe(
                    "pk_test_51Q2uEVCybVMxBZRKyHMIvOJ78qmmrsj7Xwabaxk3GhMKt@id1SS1s4N410jSL5t7W3eieDHyMwckNFHc0LPhaLE00AqRlhkXg"
                );
                var checkoutButton = document.getElementById("checkout-button");
                checkoutButton.addEventListener("click", function () {
                    fetch("/create-checkout-session", {
                        method: "POST",
                    })
                    .then(function (response) {
                        return response.json();
                    })
                    .then(function (session) {
                        console.log(session.id);
                        return stripe.redirectToCheckout({ sessionId: session.id });
                    })
                    .then(function (result) {
                        if (result.error) {
                            console.error("Error:", error);
                        }
                    });
                });
            </script>
            {% endblock %}
        </div>
    </div>
</div>
```

CSS

BUREAU

```
.cart_img {
    border: 1px solid black;
    width: 75px;
    height: 75px;
}

.btn_cart {
    padding-top: 20px;
}

.btn_delete_cart {
    text-align: center;
    background-color: #f1f8fc;
    border: 1px solid #384050;
    border-radius: 5px;
    font-weight: bold;
    padding: 10px 10px;
    margin: 10px;
}

.bloc_cart {
    display: flex;
    justify-content: space-between;
    width: 100%;
}

.cart_finish {
    margin-top: 30px;
    height: 15px;
    width: 400px;
}

.btn_pay {
    text-align: center;
    background-color: #cd2c2e;
    font-size: large;
    opacity: 70%;
    border: 2px solid #00497c;
    height: 60px;
    width: 300px;
    border-radius: 20px;
}

.btn_continue {
    background-color: #82b864;
    height: 60px;
    width: 300px;
    opacity: 70%;
    font-size: large;
    border: 2px solid #00497c;
    border-radius: 20px;
}

.price {
    font-size: 18px;
}

.total_price {
    text-align: center;
    height: 60px;
    width: 300px;
    border: 2px solid #384050;
}

.cart_choice {
    display: flex;
    gap: 10px;
}

.choice {
    display: block;
    padding-left: 310px;
}
```

MOBILE

```
.cart_finish {
    width: 360px;
}

.cart_choice {
    width: 360px;
}

.total_price {
    width: 200px;
}

.form-container {
    margin: 0px;
    padding: 0px;
}
```

CONTIENT:

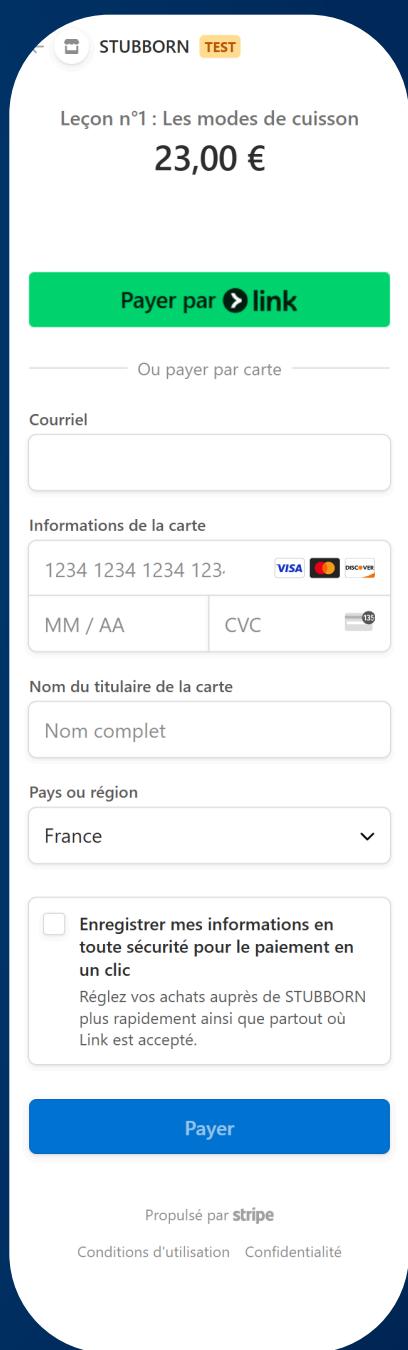
- Appel à l'entité “Formation” afin de récupérer les informations sur les formations (Titre, Imagine,Prix ...)
- Appel des entités (Cursus,Leçon,Formation) afin de pouvoir les ajouter au panier.

PAGE “ACHAT”

Vue bureau

The screenshot shows a payment interface for a lesson titled "Leçon n°1 : Les langages Html et CSS" priced at 32,00 €. At the top left is the logo "STUBBORN TEST MODE". A large green button at the top right says "Payer par link". Below it is a smaller link "Ou payer par carte". The form includes fields for "Courriel" (Email), "Informations de la carte" (Card details) showing sample card number 1234 1234 1234 1234 and logos for VISA, Mastercard, American Express, and JCB, and fields for "MM / AA" (Month/Year) and "CVC". There is also a field for "Nom du titulaire de la carte" (Cardholder's name) and a dropdown for "Pays ou région" (Country/Region) set to "France". A checkbox option "Enregistrer mes informations en toute sécurité pour le paiement en un clic" (Save my information for one-click payment) is present with explanatory text about Link payment. At the bottom is a blue "Payer" (Pay) button.

Vue mobile



CONTIENT :

- DÉTAILS DES ARTICLES AJOUTÉES
(PRIX, NOM, ETC..)

ACCESIBILITÉ:

- Etudiant
- Administrateur
- Utilisateur identifié

PAGE “ÉLÈVE”

Vue bureau

The screenshot shows a student dashboard with a white header containing navigation links: Accueil, Boutique, Panier, Élèves, and Se déconnecter. Below the header is a logo for "KNOWLEDGE" featuring an open book icon. The main content area displays a welcome message: "Bonjour STUDENT, bienvenue sur la page élève !". It also includes a note: "Ici tu retrouveras tout t'es cours ! clique sur "GO !" Afin de commencer un chapitre !". Below this, there is a section titled "FORMATION : INFORMATICIEN". Three blue rectangular buttons represent assignments: "Devoir 1" (Débutant), "Devoir 2" (Intermédiaire), and "Devoir 3" (Expert), each with a "GO !" button. At the bottom left, there is a green rectangular button labeled "Finir ma formation" (Quizz) with a "GO !" button.

Vue mobile

The screenshot shows the same student dashboard as the desktop version, but it is displayed on a mobile phone screen with rounded corners. The layout is similar, with the "KNOWLEDGE" logo at the top, followed by the welcome message and assignment buttons. The "Finir ma formation" button is also present. The overall design is responsive, adapting to the smaller screen size.

CONTIENT :

- Contenu des formations, cursus, leçon.
- Obtention du diplôme.

ACCESIBILITÉ:

- Etudiant

PAGE “ÉLÈVE”

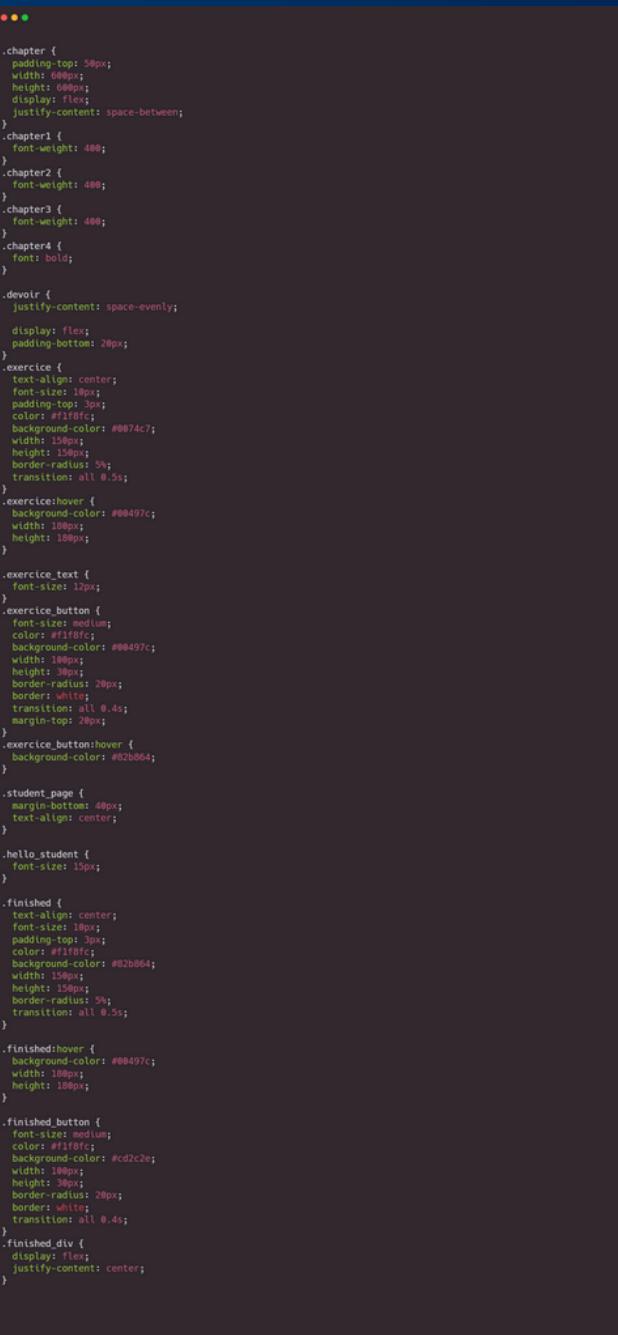
TEMPLATE



```
% if is_granted('ROLE_CERTIFICATE') %
<h1>Certifié : <strong>Knowledge Learning </strong></h1>
{% endif %}

<div class="student_page">
  {% if user %}
    <h1 class="hello_student">
      Bonjour {{ user.username }}, bienvenue sur la page élève !
    </h1>
    <br />
    <p>
      Ici tu retrouveras tout t'es cours ! <br />
      clique sur "GO !" Afin de commencer un chapitre !
    </p>
    {% if is_granted('ROLE_CUISINIER') %}
      <td>FORMATION : CUISINIER</td>
    {% endif %}{% if is_granted('ROLE_INFORMATIQUE') %}
      <td>FORMATION : INFORMATICIEN</td>
    {% endif %} {# if is_granted('ROLE_JARDINIER') %}
      <td>FORMATION : JARDINIER</td>
    {% endif %} {# if is_granted('ROLE_MUSIQUE') %}
      <td>FORMATION : MUSICIEN</td>
    {% endif %} {# endif %}
  </div>
  <div class="devoir">
    <div class="exercice">
      <h1>Devoir 1</h1>
      <p class="exercice_text">Débutant</p>
      <button class="exercice_button">GO !</button>
    </div>
    <div class="exercice">
      <h1>Devoir 2</h1>
      <p class="exercice_text">Intermédiaire</p>
      <button class="exercice_button">GO !</button>
    </div>
    <div class="exercice">
      <h1>Devoir 3</h1>
      <p class="exercice_text">Expert</p>
      <button class="exercice_button">GO !</button>
    </div>
  </div>
  <div class="finished_div">
    <div class="finished">
      <h1>Finir ma formation</h1>
      <p class="exercice_text">Quizz</p>
      <button class="finished_button">GO !</button>
    </div>
  </div>
</div>
```

CSS



```
.chapter {
  padding-top: 50px;
  width: 1000px;
  height: 600px;
  display: flex;
  justify-content: space-between;
}
.chapter1 {
  font-weight: 400;
}
.chapter2 {
  font-weight: 400;
}
.chapter3 {
  font-weight: 400;
}
.chapter4 {
  font-weight: bold;
}
.devoir {
  justify-content: space-evenly;
  display: flex;
  padding-bottom: 20px;
}
.exercice {
  text-align: center;
  font-size: 18px;
  padding-top: 30px;
  background-color: #0074c7;
  width: 150px;
  height: 150px;
  border-radius: 5px;
  transition: all 0.5s;
}
.exercice:hover {
  background-color: #00497c;
  width: 180px;
  height: 180px;
}
.exercice_text {
  font-size: 12px;
}
.exercice_button {
  font-size: medium;
  color: #f1f0fc;
  background-color: #00497c;
  width: 100px;
  height: 30px;
  border-radius: 20px;
  border: white;
  transition: all 0.4s;
  margin-top: 20px;
}
.exercice_button:hover {
  background-color: #02b864;
}
.student_page {
  margin-bottom: 40px;
  text-align: center;
}
.hello_student {
  font-size: 15px;
}
.finished {
  text-align: center;
  font-size: 18px;
  padding-top: 30px;
  color: #f1f0fc;
  background-color: #02b864;
  width: 150px;
  height: 150px;
  border-radius: 5px;
  transition: all 0.5s;
}
.finished:hover {
  background-color: #00497c;
  width: 180px;
  height: 180px;
}
.finished_button {
  font-size: medium;
  color: #f1f0fc;
  background-color: #cd2c2e;
  width: 100px;
  height: 30px;
  border-radius: 20px;
  border: white;
  transition: all 0.4s;
}
.finished_div {
  display: flex;
  justify-content: center;
}
```

CONTIENT :

- Contient un appel à l'entité “Role” afin de savoir quel métier à étais choisis.

ACCESSIBILITÉ:

- Etudiant, Administrateur.

PAGE “TABLEAU DE BORD”

Vue bureau



Accueil Boutique Panier Se déconnecter Tableau de bord

Liste des utilisateurs

CLIQUEZ ICI POUR GERER LES UTILISATEURS

Nom	Email	Création	Mis à jour	
STUDENT	student@dev.com	24/02/2025 27/02 14:18	24/02/2025 27/02 14:18	Modifier
ADMIN	admin@dev.com	24/02/2025 27/02 14:17	24/02/2025 27/02 14:17	Modifier
User	user@dev.com	24/02/2025 27/02 14:17	24/02/2025 27/02 14:17	Modifier
Yoann	yoannlecavelier18@gmail.com	28/02/2025 Jamais		Modifier
CertificateStudent	Certificat@gmail.com	28/02/2025 Jamais		Modifier
Nylun	kawzex@outlook.fr	01/04/2025 Jamais		Modifier

Contenus

ID Formation Crédit Mis à jour Crée par Mis à jour par

1	Musique	27/02/2025 27/02 14:12	x	
2	Informatique	27/02/2025 27/02 14:12	x	
3	Jardinage	27/02/2025 27/02 14:12	x	
4	Cuisine	27/02/2025 27/02 14:12	x	

Achats

Liste des Paiements

ID du paiement	Montant	Statut	Date
pi_3QxQIUcVbVMxBZRK1bscSt27	52 EUR	succeeded	2025-02-28 10:55:58
pi_3QxQbzCzbVbVMxBZRKOENEczbz	26 EUR	succeeded	2025-02-28 10:51:19
pi_3QxQDNcybVbVMxBZRKO0twzHt	26 EUR	succeeded	2025-02-28 10:50:41
pi_3QxQ7iczbVbVMxBZRKOqlhgvgf9	78 EUR	succeeded	2025-02-28 10:44:53
pi_3QxQ75GybVbVMxBZRKOlrKXieh	78 EUR	succeeded	2025-02-28 10:44:34
pi_3QxQ50CzbVbVMxBZRK1s4qEnA	52 EUR	succeeded	2025-02-28 10:42:26
pi_3QxQ3TCzbVbVMxBZRKEf8JZmf	26 EUR	succeeded	2025-02-28 10:40:27
pi_3QxQ25CzbVbVMxBZRKEwWrpv	26 EUR	succeeded	2025-02-28 10:39:01
pi_3QxPznOybVbVMxBZRK1QG6Sxxce	52 EUR	succeeded	2025-02-28 10:36:39
pi_3QxPzJzbVbVMxBZRKO2NbHz0	26 EUR	succeeded	2025-02-28 10:36:09

Utilisateur Certifiée

ID	Certifié	Mis à jour
Student		

CONTIENT :
CRUD USER
Liste des users
CRUD CONTENUES
Liste des contenus
Liste des achats
Liste des Paiements
Liste des élèves ayant obtenu une certification

Vue mobile



Liste des utilisateurs

CLIQUEZ ICI POUR GERER LES UTILISATEURS

Nom	Email	Création	Mis à jour	
STUDENT	student@dev.com	24/02/2025	24/02/2025	Modifier
ADMIN	admin@dev.com	24/02/2025	24/02/2025	Modifier
User	user@dev.com	24/02/2025	24/02/2025	Modifier
Yoann	yoannlecavelier18@gmail.com	28/02/2025	28/02/2025	Modifier
CertificateStudent	Certificat@gmail.com	28/02/2025	28/02/2025	Modifier
Nylun	kawzex@outlook.fr	01/04/2025	01/04/2025	Modifier

Contenus

ID	Formation	Création	Mis à jour	
1	Musique	27/02/2025 27/02 14:12	x	
2	Informatique	27/02/2025 27/02 14:12	x	
3	Jardinage	27/02/2025 27/02 14:12	x	
4	Cuisine	27/02/2025 27/02 14:12	x	

ID	Type	Leçon	Prix	Création	Mis à jour	Crée par	Mis à jour par
1	guitar	Leçon n°1 : Découverte de l'instrument	26 €	27/02/2025 27/02 14:18	x		
2	guitar	Leçon n°2 : Les accords et les gammes	26 €	27/02/2025 27/02 14:18	x		
3	piano	Leçon n°1 : Découverte de l'instrument	26 €	27/02/2025 Jamais	x		
4	piano	Leçon n°2 : Les accords et les gammes	26 €	27/02/2025 Jamais	x		
5	technology	Leçon n°1 : Les langages Html et CSS	32 €	27/02/2025 Jamais	x		
6	technology	Leçon n°2 : Dynamiser votre site avec Javascript	32 €	27/02/2025 27/02 14:18	x		
7	gardening	Leçon n°1 : Les outils du jardinier	16 €	27/02/2025 Jamais	x		
8	gardening	Leçon n°2 : Jardiner avec la lune	16 €	27/02/2025 Jamais	x		
9	kitchen	Leçon n°1 : Les modes de cuisson	23 €	27/02/2025 27/02 14:18	x		
10	kitchen	Leçon n°2 : Les saveurs	23 €	27/02/2025 27/02 14:18	x		
11	kitchen	Leçon n°1 : Mettre en œuvre le style de l'assiette	26 €	27/02/2025 27/02 14:18	x		
12	kitchen	Leçon n°2 : Harmoniser un repas à quatre plats	26 €	27/02/2025 Jamais	x		

Achats

Liste des Paiements

ID du paiement	Montant	Statut	Date
pi_3QxQIUcVbVMxBZRK1bscSt27	52 EUR	succeeded	2025-02-28 10:55:58
pi_3QxQbzCzbVbVMxBZRKOENEczbz	26 EUR	succeeded	2025-02-28 10:51:19
pi_3QxQDNcybVbVMxBZRKO0twzHt	26 EUR	succeeded	2025-02-28 10:50:41
pi_3QxQ7iczbVbVMxBZRKOqlhgvgf9	78 EUR	succeeded	2025-02-28 10:44:53
pi_3QxQ75GybVbVMxBZRKOlrKXieh	78 EUR	succeeded	2025-02-28 10:44:34
pi_3QxQ50CzbVbVMxBZRK1s4qEnA	52 EUR	succeeded	2025-02-28 10:42:26
pi_3QxQ3TCzbVbVMxBZRKEf8JZmf	26 EUR	succeeded	2025-02-28 10:40:27
pi_3QxQ25CzbVbVMxBZRKEwWrpv	26 EUR	succeeded	2025-02-28 10:39:01
pi_3QxPznOybVbVMxBZRK1QG6Sxxce	52 EUR	succeeded	2025-02-28 10:36:39
pi_3QxPzJzbVbVMxBZRKO2NbHz0	26 EUR	succeeded	2025-02-28 10:36:09

Utilisateur Certifiée

ID	Certifié	Mis à jour
Student		

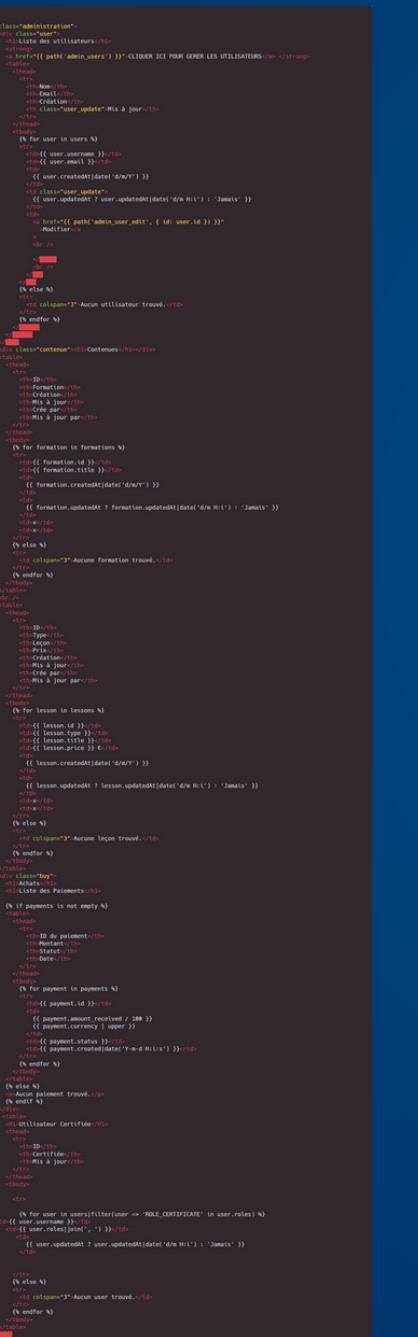
- Administrateur

ACCESIBILITÉ:

ID	Certifié	Mis à jour
Student	ROLE_CERTIFICATE	ROLE_USER

PAGE “TABLEAU DE BORD”

TEMPLATE



```
<!-- class="administration" -->
<!-- class="user_update" -->
<!-- class="formation" -->
<!-- class="lesson" -->
<!-- class="order" -->
<!-- class="user" -->
<!-- class="certificate" -->
<!-- class="role" -->
<!-- class="payment" -->
<!-- class="user-role" -->
<!-- class="user-relations" -->


<a href="#">Gérer les utilisateurs


<table border="1">
<thead>
<tr>
<th>Nom
<th>Prénom
<th>Role
<th>Mis à jour

</thead>
<tbody>
<tr>
 John | Doe | Administrateur | 10 minutes ago |

<tr>
 Jane | Doe | Administrateur | 10 minutes ago |

<tr>
 Bob | Smith | Utilisateur | 10 minutes ago |
<table border="1">
<thead>
<tr>
<th>Nom
<th>Type
<th>Leçon
<th>Création
<th>Mis à jour

</thead>
<tbody>
<tr>
 Formation A | Formation | 10 leçons | 10 minutes ago | 10 minutes ago |

<tr>
 Formation B | Formation | 10 leçons | 10 minutes ago | 10 minutes ago |
<table border="1">
<thead>
<tr>
<th>Nom
<th>Type
<th>Création
<th>Mis à jour

</thead>
<tbody>
<tr>
 Leçon A | Formation | 10 minutes ago | 10 minutes ago |

<tr>
 Leçon B | Formation | 10 minutes ago | 10 minutes ago |
<table border="1">
<thead>
<tr>
<th>ID
<th>Montant
<th>Statut
<th>Date

</thead>
<tbody>
<tr>
 P001 | 100€ | En attente | 10 minutes ago |

<tr>
 P002 | 200€ | En attente | 10 minutes ago |
<table border="1">
<thead>
<tr>
<th>Nom
<th>Prénom
<th>Role
<th>Mis à jour

</thead>
<tbody>
<tr>
 John | Doe | Administrateur | 10 minutes ago |

<tr>
 Jane | Doe | Administrateur | 10 minutes ago |
```

CONTENU :

- Appel à l’entité “User”
- Appel à l’entité “Formation”
- Appel à l’entité “Lesson”
- Appel à l’entité “Order”

CSS



```
.administration {
  width: 360px;
  height: auto;
  font-size: 10px;
}

.administration.table {
  width: 360px;
}

.user_update {
  display: none;
}

.modify_user {
  width: 360px;
  height: auto;
}
```

ACCESIBILITÉ:

- Administrateur

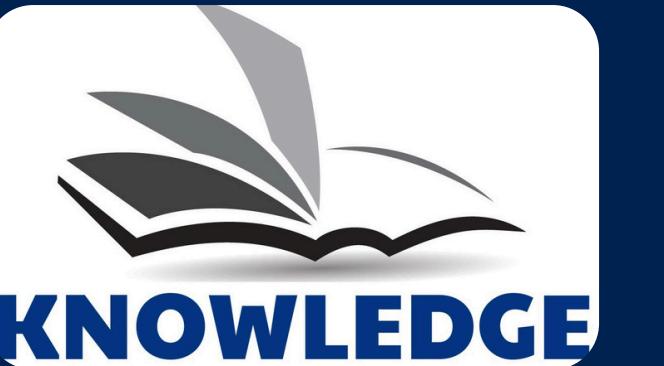
CIBLE DU PROJET & USER-STORIES :

En tant que	Je souhaite	Afin de (besoin/nécessité)
Visiteur	Me créer un compte afin de voir les formations disponible.	Voir la liste de formation disponible
Utilisateur vérifié	Voir la liste de formation disponible.	Réalisé un achat si une formation/leçon me plaît
Utilisateur étudiant	Accéder à ma mes achats	Accéder à mes leçons ou ma formation
Administrateur	Modifier Supprimer Ajouter ou Mettre à jour	Maintenir le site web et ajouté des nouveautés.

ROUTES FRONT :

URL	Title	Content
/	Accueil	Acceuil du site web
/formations	Formations	Boutique de formation
/formations/{id}	Détails de la formation	Affiche les détails sur la formation
/cart	Panier	Affiche le contenu du panier
/login	Connexion	Affiche le formulaire de connexion
/register	Inscription	Affiche le formulaire d'inscription
/student	Elève	Affiche la page d'apprentissage
/logout	Déconnexion	Déconnecte l'utilisateur et retourne à l'accueil

CHARTE GRAPHIQUE :



RÉALISATION DU CÔTÉ BACK-END

CONDITIONS À RESPECTER ?

FONCTIONNALITÉES À PRÉVOIR ?

SCHÉMA DES FONCTIONNALITÉS

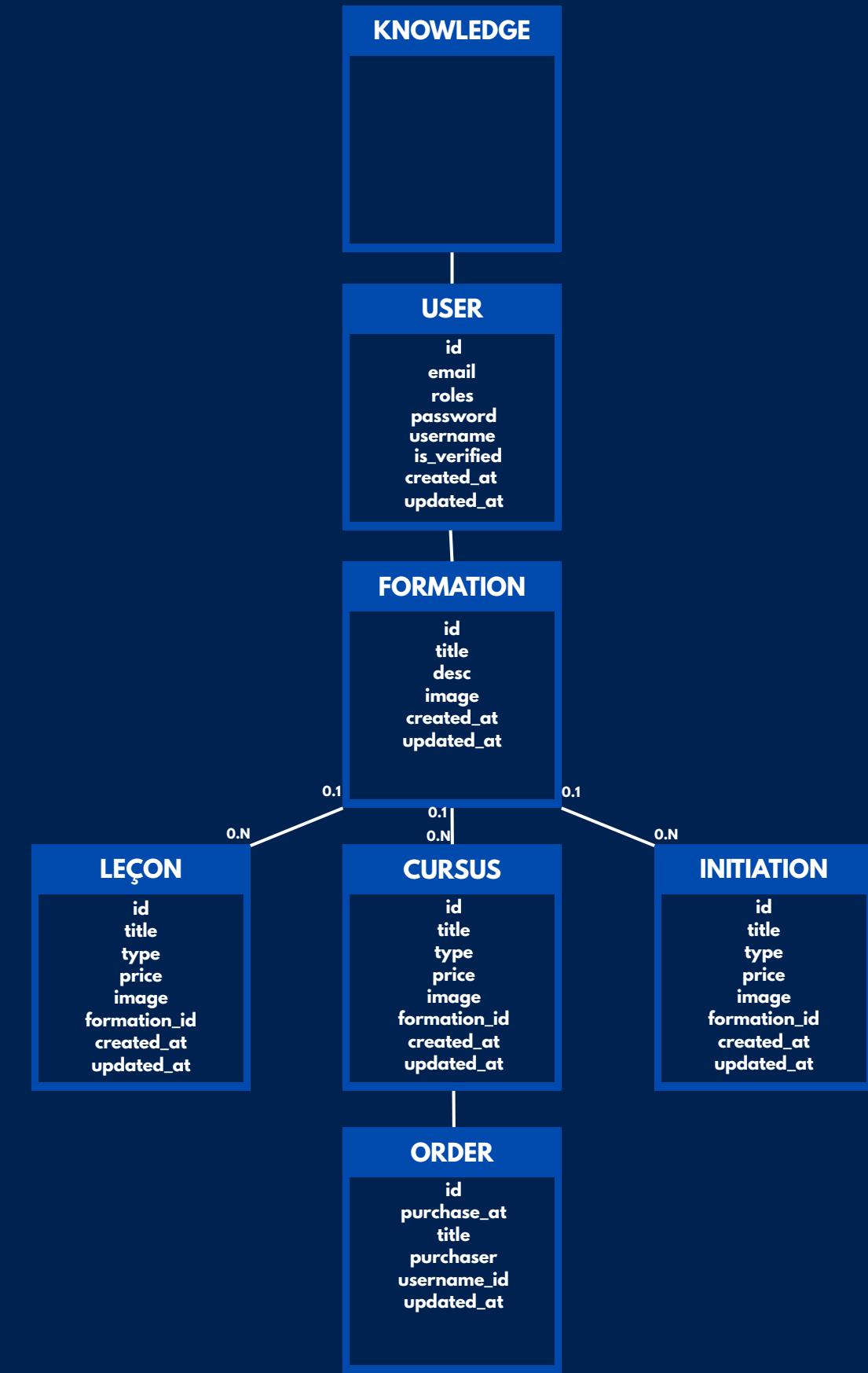
ACCEUIL

Entités

Formulaire



DIAGRAMME DE LA BASE DE DONNÉES



0.1 → 0.N
One to many

0.N → 0.1
Many to one

PRÉSENTATION DES ENTITÉES

Une entité est une classe PHP qui représente un modèle de données.

FONCTIONS DE L'ENTITÉ “USER”

EXTRAIT DE ENTITY.PHP

```
class User implements UserInterface, PasswordAuthenticatedUserInterface
{
    /**
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     * @Column(type="integer")
     * @var int $id = null;
     */
    private ?int $id = null;

    /**
     * @ORM\Column(length=100, unique=true)
     * @Column(name="email")
     * @var string $email = null;
     */
    private ?string $email = null;

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

    /**
     * @var string The hashed password
     */
    private ?string $password = null;

    /**
     * @var string $username
     */
    private ?string $username = null;

    /**
     * @var bool $isVerified = false;
     */

    /**
     * @ORM\Column(type="datetime_immutable")
     * @Column(name="created_at")
     * @var \DateTimeImmutable $createdAt = null;
     */
    private ?\DateTimeImmutable $createdAt = null;

    /**
     * @ORM\Column(type="datetime", nullable=true)
     * @Column(name="updated_at")
     * @var \DateTime $updatedAt = null;
     */
    private ?\DateTime $updatedAt = null;

    public function __construct()
    {
        $this->createdAt = new \DateTimeImmutable();
    }

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

    public function getUpdatedAt(): ?\DateTime
    {
        return $this->updatedAt;
    }

    /**
     * @param string $username
     */
    public function setUsername(string $username): void
    {
        $this->username = $username;
    }

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

    public function getId(): ?int
    {
        return $this->id;
    }

    public function getEmail(): ?string
    {
        return $this->email;
    }

    public function setEmail(string $email): void
    {
        $this->email = $email;
    }

    /**
     * A virtual identifier that represents this user.
     */
    public function getIdentifier(): string
    {
        return (string) $this->email;
    }

    /**
     * @see UserInterface
     * @return list<string>
     */
    public function getRoles(): array
    {
        $roles = $this->roles;
        if (!in_array('ROLE_USER', $roles)) {
            $roles[] = 'ROLE_USER';
        }
        return array_unique($roles);
    }

    /**
     * @param list<string> $roles
     */
    public function setRoles(array $roles): void
    {
        $this->roles = $roles;
    }

    public function getPassword(): ?string
    {
        return $this->password;
    }

    public function setPassword(string $password): void
    {
        $this->password = $password;
    }

    /**
     * @see UserInterface
     */
    public function eraseCredentials(): void
    {
        // If you store any temporary, sensitive data on the user, clear it here
        // $this->plainPassword = null;
    }

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

    public function setIsVerified(bool $isVerified): void
    {
        $this->isVerified = $isVerified;
    }

    public function __toString()
    {
        return $this->username;
    }
}
```

**ENTITÉ GÉNÉRÉE AUTOMATIQUEMENT LORS DE L'IMPORT DU FORM LOGIN
(USERNAME,PASSWORD,EMAIL...)**

FONCTION IS_VERIFIED/SETISVERIFIED POUR VÉRIFIER SI L'EMAIL À ÉTAIS VALIDÉ (BOOLEAN)

FONCTION CREATED_AT POUR AVOIR LA DATE DE CRÉATION DU COMPTE (DATETIME_IMMUTABLE)

FONCTION UPDATED_AT POUR AVOIR LA DATE DE MISE À JOUR DU COMPTE (DATETIME_MUTABLE)

**FONCTION GETCREATEDAT/GETUPDATEDAT/SETUPDATEDAT
AFIN DE RETOURNER ET DE METTRE À JOUR LA DATE DE MISE À JOUR DE L'USER (DATETIME_MUTABLE)**

FONCTION GETUSERNAME/SETUSERNAME AFIN DE RETOURNER ET MODIFIER LE NOM D'UTILISATEUR (STRING)

**FONCTION GETID/GETEMAIL/SETEMAIL AFIN DE RETOURNER L'ID ET L'EMAIL ET MODIFIER L'EMAIL
(INTEGER POUR L'ID ET STRING POUR L'EMAIL)**

FONCTION GETROLES ET SETROLES AFIN DE RETOURNER LE RÔLE DE L'USER ET LUI ATTRIBUER UN RÔLE (ARRAY)

FONCTIONS DE L'ENTITÉ “FORMATION”

EXTRAIT DE FORMATION.PHP

```
<?php  
namespace App\Entity;  
  
use App\Repository\FormationRepository;  
use Doctrine\ORM\Mapping as ORM;  
use Doctrine\Common\Collections\ArrayCollection;  
use Doctrine\Common\Collections\Collection;  
use Doctrine\ORM\Mapping\JoinTable;  
  
#ORM\Entity(repositoryClass: FormationRepository::class)  
class Formation  
  
{  
  
    #ORM\Id  
    #ORM\GeneratedValue  
    private ?int $id = null;  
  
    public function getId(): ?int  
    {  
        return $this->id;  
    }  
  
    // Column for the title of the formation  
    private ?string $title = null;  
  
    public function getTitle(): ?string  
    {  
        return $this->title;  
    }  
  
    // Column for the description of the formation  
    private ?string $desc = null;  
  
    public function getDesc(): ?string  
    {  
        return $this->desc;  
    }  
  
    // Adding image  
    #ORM\Column(type: string, length: 255, nullable: true)  
    private ?string $image = null;  
  
    public function getImage(): ?string  
    {  
        return $this->image;  
    }  
  
    public function setImage(?string $image): self  
    {  
        $this->image = $image;  
        return $this;  
    }  
  
    //Adding datetime for created_at and update_at  
    #ORM\Column(type: datetime_immutable )  
    private ?\DateTime $createdAt = null;  
  
    #ORM\Column(type: datetime, nullable: true)  
    private ?\DateTime $updatedAt = null;  
  
    /**  
     * @var Collection<int, Lesson>  
     */  
    #ORM\ManyToOne(targetEntity: Lesson::class, mappedBy: 'formation')  
    private Collection $lessons;  
  
    public function __construct()  
    {  
        $this->createdAt = new \DateTimeImmutable();  
        $this->lessons = new ArrayCollection();  
    }  
  
    /**  
     * @return Collection<int, Lesson>  
     */  
    public function getLessons(): Collection  
    {  
        return $this->lessons;  
    }  
  
    public function addLesson(Lesson $lesson): static  
    {  
        if ($this->lessons->contains($lesson)) {  
            $this->lessons->add($lesson);  
            $lesson->setFormation($this);  
        }  
  
        return $this;  
    }  
  
    public function removeLesson(Lesson $lesson): static  
    {  
        if ($this->lessons->removeElement($lesson)) {  
            // set the owning side to null (unless already changed)  
            if ($lesson->getFormation() === $this) {  
                $lesson->setFormation(null);  
            }  
        }  
  
        return $this;  
    }  
  
    public function getCreatedAt(): ?\DateTimeImmutable  
    {  
        return $this->createdAt;  
    }  
  
    public function getUpdatedAt(): ?\DateTime  
    {  
        return $this->updatedAt;  
    }  
  
    #ORM\PreUpdate  
    public function setUpdatedAt(): void  
    {  
        $this->updatedAt = new \DateTime();  
    }  
  
    public function getRoleForUser()  
    {  
        if ($this->getId() == 1){  
            return 'ROLE_BASIC';  
        } elseif ($this->getId() == 2){  
            return 'ROLE_ADVANCED';  
        }  
  
        return 'ROLE_USER';  
    }  
}
```

ENTITÉ CRÉER MANUELLEMENT DEPUIS L'INVITÉ DE COMMANDE

FONCTION CREATED_AT POUR AVOIR LA DATE DE CRÉATION DE LA FORMATION (DATETIME_IMMUTABLE)

FONCTION UPDATED_AT POUR POUVOIR METTRE À JOUR LA FORMATION ET AVOIR LA DATE DE LA DERNIÈRE MISE À JOUR (DATETIME_MUTABLE)

FONCTION GETCREATEDAT/GETUPDATEDAT/SETUPDATEDAT
AFIN DE RETOURNER LA DATE DE MISE À JOUR DE LA FORMATION (DATETIME_MUTABLE)

FONCTION GETTITLE POUR AVOIR ET RETOURNER LE TITRE DE LA FORMATION (STRING)

FONCTION GETIMAGE POUR AVOIR L'IMAGE DE LA FORMATION (STRING)
FONCTION SETIMAGE POUR RETOURNER L'IMAGE DE LA FORMATION (STRING)

FONCTION GETLESSONS AFIN DE RETOURNER LES LEÇONS PRÉSENTE DANS LA FORMATION

FONCTION ADDLESSONS AFIN D'AJOUTER DES LEÇONS DANS UNE FORMATION

FONCTION REMOVELESSONS AFIN DE RETIRER DES LEÇONS DANS UNE FORMATION

FONCTION GETROLEFORUSERS AFIN DE DETERMINER QUEL RÔLE À L'UTILISATEUR

FONCTION GETID AFIN DE RETOURNER UN ID PAR FORMATION

FONCTIONS DE L'ENTITÉ “LESSON”

EXTRAIT DE LESSON.PHP

```
<?php  
namespace App\Entity;  
  
use App\Repository\LessonRepository;  
use Doctrine\Common\Collections\ArrayCollection;  
use Doctrine\Common\Collections\Collection;  
use Doctrine\ORM\Mapping as ORM;  
  
#[ORM\Entity(repositoryClass: LessonRepository::class)]  
class Lesson  
  
{  
  
    #[ORM\Column(type: 'datetime_immutable')]  
    private ?\DateTimeImmutable $createdAt = null;  
  
    #[ORM\Column(type: 'datetime', nullable: true)]  
    private ?\DateTime $updatedAt = null;  
  
    public function __construct()  
    {  
        $this->createdAt = new \DateTimeImmutable();  
    }  
  
    public function getCreatedAt(): ?\DateTimeImmutable  
    {  
        return $this->createdAt;  
    }  
  
    public function getUpdatedAt(): ?\DateTime  
    {  
        return $this->updatedAt;  
    }  
  
    #[ORM\PreUpdate]  
    public function setUpdatedAt(): void  
    {  
        $this->updatedAt = new \DateTime();  
    }  
  
    #[ORM\Id]  
    #[ORM\GeneratedValue]  
    #[ORM\Column]  
    private ?int $id = null;  
  
    public function getId(): ?int  
    {  
        return $this->id;  
    }  
  
    //Column for the title of the lesson  
    #[ORM\Column]  
    private ?string $title = null;  
  
    public function getTitle(): ?string  
    {  
        return $this->title;  
    }  
  
    //Column for the type of the lesson  
    #[ORM\Column]  
    private ?string $type = null;  
  
    public function getType(): ?string  
    {  
        return $this->type;  
    }  
  
    #[ORM\Column]  
    private ?string $image = null;  
  
    public function getImage(): ?string  
    {  
        return $this->image;  
    }  
  
    #[ORM\Column]  
    private ?string $price = null;  
  
    #[ORM\ManyToOne(inversedBy: 'lessons')]  
    private ?Formation $formation = null;  
  
    //Column for the price of the lesson  
    public function getPrice(): ?string  
    {  
        return $this->price;  
    }  
  
    public function getFormation(): ?Formation  
    {  
        return $this->formation;  
    }  
  
    public function setFormation(?Formation $formation):  
    {  
        $this->formation = $formation;  
        return $this;  
    }  
  
    public function setType(?string $type): self  
    {  
        $this->type = $type;  
        return $this;  
    }  
}
```

ENTITÉ CRÉER MANUELLEMENT DEPUIS L'INVITÉ DE COMMANDE

FONCTION CREATED_AT POUR AVOIR LA DATE DE CRÉATION DE LA LEÇON (DATETIME_IMMUTABLE)

FONCTION UPDATED_AT POUR POUVOIR METTRE À JOUR LA LEÇON ET AVOIR LA DATE
DE LA DERNIÈRE MISE À JOUR (DATETIME_MUTABLE)

FONCTION GETCREATEDAT/GETUPDATEDAT/SETUPDATEDAT
AFIN DE RETOURNER LA DATE DE MISE À JOUR DE LA LEÇON (DATETIME_MUTABLE)

FONCTION GETTITLE POUR AVOIR ET RETOURNER LE TITRE DE LA LEÇON (STRING)

FONCTION GETTYPE POUR AVOIR LE TYPE DE LA LEÇON (STRING)
FONCTION SETTYPE POUR RETOURNER LE TYPE DE LA LEÇON

FONCTION GETIMAGE POUR AVOIR ET RETOURNER L'IMAGE DE LA LEÇON (STRING)

FONCTION GETPRICE AFIN D'AVOIR ET DE RETOURNER LE PRIX DE LA LEÇON

FONCTION GETFORMATION AFIN DE SAVOIR LA FORMATION A LAQUELLE EST LIÉE LA LEÇON
FONCTION SETFORMATION AFIN DE RETOURNER LA FORMATION A LAQUELLE EST LIÉE LA LEÇON

FONCTION GETID AFIN DE RETOURNER UN ID PAR LEÇON

FONCTIONS DE L'ENTITÉ “CURSUS”

EXTRAIT DE CURSUS.PHP

```
...  
<?php  
  
namespace App\Entity;  
  
use App\Repository\CurusRepository;  
use Doctrine\Common\Collections\ArrayCollection;  
use Doctrine\Common\Collections\Collection;  
use Doctrine\ORM\Mapping as ORM;  
  
#(ORM\repositoryClass: CurusRepository::class)  
class Curus  
{  
  
    #ORM\Column(type: 'datetime_immutable')  
    private ?\DateTimeImmutable $createdAt = null;  
  
    #ORM\Column(type: 'datetime', nullable: true)  
    private ?\DateTime $updatedAt = null;  
  
    public function __construct()  
    {  
        $this->createdAt = new \DateTimeImmutable();  
    }  
  
    public function getCreatedAt(): ?\DateTimeImmutable  
    {  
        return $this->createdAt;  
    }  
  
    public function getUpdatedAt(): ?\DateTime  
    {  
        return $this->updatedAt;  
    }  
  
    #(ORM\Preload)  
    public function setUpdatedAt(): void  
    {  
        $this->updatedAt = new \DateTime();  
    }  
  
    #ORM\id  
    #ORM\GeneratedValue  
    #ORM\Column  
    private ?int $id = null;  
    public function getId(): ?int  
    {  
        return $this->id;  
    }  
  
    //Column for the title of the lesson  
    #ORM\Column  
    private ?string $title = null;  
    public function getTitle(): ?string  
    {  
        return $this->title;  
    }  
  
    //Column for the type of the lesson  
    #ORM\Column  
    private ?string $type = null;  
    public function getType(): ?string  
    {  
        return $this->type;  
    }  
  
    #ORM\Column  
    private ?string $image = null;  
    public function getImage(): ?string  
    {  
        return $this->image;  
    }  
  
    #ORM\Column  
    private ?string $price = null;  
    #ORM\ManyToOne(targetEntity: 'Formation')  
    private ?Formation $formation = null;  
    //Column for the price of the lesson  
    public function getPrice(): ?string  
    {  
        return $this->price;  
    }  
    public function getFormation(): ?Formation  
    {  
        return $this->formation;  
    }  
    public function setFormation(?Formation $formation): static  
    {  
        $this->formation = $formation;  
        return $this;  
    }  
  
    public function setType(?string $type): self  
    {  
        $this->type = $type;  
        return $this;  
    }  
}
```

ENTITÉ CRÉER MANUELLEMENT DEPUIS L'INVITÉ DE COMMANDE

FONCTION CREATED_AT POUR AVOIR LA DATE DE CRÉATION DU CURSUS (DATETIME_IMMUTABLE)

**FONCTION UPDATED_AT POUR POUVOIR METTRE À JOUR LE CURSUS ET AVOIR LA DATE
DE LA DERNIÈRE MISE À JOUR (DATETIME_MUTABLE)**

**FONCTION GETCREATEDAT/GETUPDATEDAT/SETUPDATEDAT
AFIN DE RETOURNER LA DATE DE MISE À JOUR DU CURSUS (DATETIME_MUTABLE)**

FONCTION GETTITLE POUR AVOIR ET RETOURNER LE TITRE DU CURSUS (STRING)

**FONCTION GETTYPE POUR AVOIR LE TYPE DU CURSUS (STRING)
FONCTION SETTYPE POUR RETOURNER LE TYPE DU CURSUS**

FONCTION GETIMAGE POUR AVOIR ET RETOURNER L'IMAGE DU CURSUS (STRING)

FONCTION GETPRICE AFIN D'AVOIR ET DE RETOURNER LE PRIX DU CURSUS

**FONCTION GETFORMATION AFIN DE SAVOIR LA FORMATION A LAQUELLE EST LIÉE AU CURSUS
FONCTION SETFORMATION AFIN DE RETOURNER LA FORMATION A LAQUELLE EST LIÉE AU CURSUS**

FONCTION GETID AFIN DE RETOURNER UN ID PAR CURSUS

FONCTIONS DE L'ENTITÉ “INITIATION”

EXTRAIT DE INITIATION.PHP

```
***  
<?php  
  
namespace App\Entity;  
  
use App\Repository\InitiationRepository;  
use Doctrine\ORM\Mapping as ORM;  
  
#[ORM\Entity(repositoryClass: InitiationRepository::class)]  
class Initiation  
{  
  
    // Column for the date of creation of the cursus  
    #[ORM\Column(type: 'datetime_immutable')]  
    private ?\DateTimeImmutable $createdAt = null;  
  
    #[ORM\Column(type: 'datetime', nullable: true)]  
    private ?\DateTime $updatedAt = null;  
  
    public function __construct()  
    {  
        $this->createdAt = new \DateTimeImmutable();  
    }  
  
    public function getCreatedAt(): ?\DateTimeImmutable  
    {  
        return $this->createdAt;  
    }  
  
    // Column for the date of the up to date cursus  
    public function getUpdatedAt(): ?\DateTime  
    {  
        return $this->updatedAt;  
    }  
  
    #[ORM\Preupdate]  
    public function setUpdatedAt(): void  
    {  
        $this->updatedAt = new \DateTime();  
    }  
  
    #[ORM\Id]  
    #[ORM\GeneratedValue]  
    #[ORM\Column]  
    private ?int $id = null;  
    public function getId(): ?int  
    {  
        return $this->id;  
    }  
  
    //Column for the title of the cursus  
    #[ORM\Column]  
    private ?string $title = null;  
  
    public function getTitle(): ?string  
    {  
        return $this->title;  
    }  
  
    //Column for the type of the cursus  
    #[ORM\Column]  
    private ?string $type = null;  
  
    public function getType(): ?string  
    {  
        return $this->type;  
    }  
  
    public function setType(?string $type): self  
    {  
        $this->type = $type;  
        return $this;  
    }  
  
    //Column for the price of the cursus  
    #[ORM\Column]  
    private ?string $price = null;  
  
    public function getPrice(): ?string  
    {  
        return $this->price;  
    }  
  
    // Column for the image of the cursus  
    #[ORM\Column]  
    private ?string $image = null;  
  
    public function getImage(): ?string  
    {  
        return $this->image;  
    }  
  
    // Column for the formation_id  
    #[ORM\ManyToMany(mappedBy: 'initiation')]  
    private ?Formation $formation = null;  
  
    public function getFormation(): ?Formation  
    {  
        return $this->formation;  
    }  
  
    public function setFormation(?Formation $formation): static  
    {  
        $this->formation = $formation;  
        return $this;  
    }  
}
```

ENTITÉ CRÉER MANUELLEMENT DEPUIS L'INVITÉ DE COMMANDE

FONCTION CREATED_AT POUR AVOIR LA DATE DE CRÉATION DE L'INITIATION (DATETIME_IMMUTABLE)

**FONCTION UPDATED_AT POUR POUVOIR METTRE À JOUR L'INITIATION ET AVOIR LA DATE
DE LA DERNIÈRE MISE À JOUR (DATETIME_MUTABLE)**

**FONCTION GETCREATEDAT/GETUPDATEDAT/SETUPDATEDAT
AFIN DE RETOURNER LA DATE DE MISE À JOUR DE L'INITIATION (DATETIME_MUTABLE)**

FONCTION GETTITLE POUR AVOIR ET RETOURNER LE TITRE DE L'INITIATION (STRING)

**FONCTION GETTYPE POUR AVOIR LE TYPE DE L'INITIATION (STRING)
FONCTION SETTYPE POUR RETOURNER LE TYPE DE L'INITIATION**

FONCTION GETIMAGE POUR AVOIR ET RETOURNER L'IMAGE DE L'INITIATION (STRING)

FONCTION GETPRICE AFIN D'AVOIR ET DE RETOURNER LE PRIX DE L'INITIATION

**FONCTION GETFORMATION AFIN DE SAVOIR LA FORMATION A LAQUELLE EST LIÉE DE L'INITIATION
FONCTION SETFORMATION AFIN DE RETOURNER LA FORMATION A LAQUELLE EST LIÉE DE L'INITIATION**

FONCTION GETID AFIN DE RETOURNER UN ID PAR INITIATION

FONCTIONS DE L'ENTITÉ “ORDER”

EXTRAIT DE ORDER.PHP

```
● ● ●
<?php
namespace App\Entity;

use App\Repository\OrderRepository;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\ORM\Mapping as ORM;

#[ORM\Entity(repositoryClass: OrderRepository::class)]
#[ORM\Table(name: 'order')]
class Order
{

    #[ORM\Column(type: 'datetime_immutable')]
    private ?\DateTimeImmutable $purchaseAt = null;

    public function __construct()
    {
        $this->purchaseAt = new \DateTimeImmutable();
    }

    public function getPurchaseAt(): ?\DateTimeImmutable
    {
        return $this->purchaseAt;
    }

    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    private ?int $id = null;

    public function getId(): ?int
    {
        return $this->id;
    }

    private ?string $article = null;

    public function getArticle(): ?string
    {
        return $this->article;
    }

    public function getPurchaser(?User $username): Collection
    {
        return $this->$username;
    }

    public function getPurchaseDate(): ?\DateTimeImmutable
    {
        return $this->purchaseAt;
    }

    #[ORM\Column]
    private ?string $Purchaser = null;

    #[ORM\Column]
    private ?string $title = null;

    public function getTitle(): ?string
    {
        return $this->title;
    }

    #[ORM\ManyToMany(mappedBy: 'order')]
    private ?User $username = null;

    public function getPurchaser(): ?User
    {
        return $this->username;
    }

    public function setPurchaser(?User $user): static
    {
        $this->username = $user;
        return $this;
    }

}
```

ENTITÉ CRÉER MANUELLEMENT DEPUIS L'INVITÉ DE COMMANDE

FONCTION GETPURCHASEAT POUR AVOIR LA DATE DE L'ACHAT (DATETIME_IMMUTABLE)

FONCTION GETPURCHASEDATE AFIN DE RETOURNER LA DATE DE L'ACHAT (DATETIME_IMMUTABLE)

FONCTION GETPURCHASER AFIN D'OBTENIR L'ACHETEUR (RELATION MANY TO ONE AVEC LA TABLE USER | COLLECTION)
FONCTION SETPURCHASER AFIN DE RETOURNER L'ACHETEUR (STATIC)

FONCTION GETID AFIN DE RETOURNER UN ID PAR ORDER

PRÉSENTATION DES CONTROLEURS

Un contrôleur Symfony est une classe qui gère les requêtes des utilisateurs, exécute du code, et retourne une réponse.

CONTROLEUR “HOMECONTROLLER”

EXTRAIT DU CONTROLEUR

```
●●●
<?php
namespace App\Controller;
use App\Entity\Formation;
use Doctrine\Persistence\ManagerRegistry;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
final class HomeController extends AbstractController
{
    #[Route('/', name: 'app_home')]
    public function index(ManagerRegistry $manager): Response
    {
        return $this->render('home/index.html.twig', []);
    }
}
```

CONTROLEUR GÉNÉRÉE MANUELLEMENT AVEC L'INVITÉ DE COMMANDE.

ROUTE /

FUNCTION INDEX PERMETTANT DE RETOURNER LA TEMPLATE “HOME/TEMPLATE.HTML.TWIG”

CONTROLEUR “LOGINCONTROLLER”

EXTRAIT DU CONTROLEUR

```
● ● ●  
<?php  
  
namespace App\Controller;  
  
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;  
use Symfony\Component\HttpFoundation\Response;  
use Symfony\Component\Routing\Annotation\Route;  
  
final class LoginController extends AbstractController  
{  
    #[Route('/login', name: 'app_login')]  
    public function index(): Response  
    {  
        return $this->render('login/index.html.twig', [  
            'controller_name' => 'LoginController',  
        ]);  
    }  
}
```

CONTROLEUR GÉNÉRÉE MANUELLEMENT AVEC L’INVITÉ DE COMMANDE.

ROUTE /LOGIN

FUNCTION INDEX PERMETTANT DE RETOURNER LA TEMPLATE “LOGIN/LOGIN.HTML.TWIG” ET LA VARIABLE “CONTROLLER_NAME” AFIN DE PASSER LES VALEURS DU FORMULAIRE DE LOGIN

ROUTE /LOGOUT
PERMETTANT DE SE DÉCONNECTER SANS RIEN RETOURNER

CONTROLEUR “REGISTRATIONCONTROLEUR”

EXTRAIT DU CONTROLEUR

```
<?php  
namespace App\Controller;  
  
use App\Entity\User;  
use App\Form\RegistrationFormType;  
use App\Repository\UserRepository;  
use App\Security\EmailVerifier;  
use App\Security\UserPasswordHasherInterface;  
use Symfony\Bridge\Twig\Mime\TemplatedEmail;  
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;  
use Symfony\Component\HttpFoundation\Request;  
use Symfony\Component\HttpFoundation\Response;  
use Symfony\Component\Intl\Util\Address;  
use Symfony\Component\PasswordHasher\User\UserPasswordHasherInterface;  
use Symfony\Component\VerifyEmail\Exception\VerifyEmailExceptionInterface;  
use Symfony\Component\VerifyEmail\VerifierInterface;  
use Symfony\Component\VerifyEmail;  
  
class RegistrationController extends AbstractController  
{  
    public function __construct(private EmailVerifier $emailVerifier)  
    {}  
  
    #[Route('/register', name: 'app_register')]  
    public function register(Request $request, UserPasswordHasherInterface $userPasswordHasher,  
    EntityManagerInterface $entityManager, MailerInterface $mailer): Response  
    {  
        $user = new User();  
        $form = $this->createForm(RegistrationFormType::class, $user);  
        $form->handleRequest($request);  
  
        if ($form->isSubmitted() && $form->isValid()) {  
            $user->string($password);  
            $password = $form->get('password')->getData();  
            // encode the plain password  
            $user->setPassword($userPasswordHasher->hashPassword($user, $password));  
            $entityManager->persist($user);  
            $entityManager->flush();  
  
            // generate a signed url and mail it to the user  
            $this->get('emailVerifier')->handleEmailConfirmation('app_verify_email', $user,  
            (new TokenEmail($user))  
                ->from(new Address('stubborn@blabla.com', 'stubborn'))  
                ->to($request->get('email'))  
                ->subject('Veuillez Confirmer votre email')  
                ->htmlTemplate('registration/confirmation_email.html.twig')  
            );  
            // do anything else you need here, like send an email  
            return $this->redirectToRoute('app_home');  
        }  
        return $this->render('registration/register.html.twig', [  
            'registrationForm' => $form,  
        ]);  
    }  
  
    #[Route('/verify_email', name: 'app_verify_email')]  
    public function verifyUserEmail(Request $request, UserRepository $userRepository, EntityManagerInterface $entityManager): Response  
    {  
        $id = $request->query->get('id');  
        if (null === $id) {  
            return $this->redirectToRoute('app_register');  
        }  
  
        $user = $userRepository->find($id);  
        if (null === $user) {  
            return $this->redirectToRoute('app_register');  
        }  
  
        // validate email confirmation link, sets User::isVerified=true and persists  
        try {  
            if ($user->validateEmailConfirmation($id)) {  
                $this->get('emailVerifier')->handleEmailConfirmation($request, $user);  
                $user->setRole(array_merge($user->getRoles(), ['ROLE_VERIFIED']));  
                $entityManager->flush();  
                $entityManager->flush();  
                $this->addFlash('success', 'Votre adresse email a été vérifiée avec succès et vous avez maintenant  
accès aux fonctionnalités');  
            } catch (VerifyEmailExceptionInterface $exception) {  
                $this->addFlash('verify_email_error', $exception->getReason());  
            }  
        } catch (\Exception $e) {  
            $this->addFlash('verify_email_error', $e->getMessage());  
        }  
        return $this->redirectToRoute('app_register');  
    }  
    return $this->redirectToRoute('app_home');  
}
```

CONTROLEUR GÉNÉRÉE AUTOMATIQUEMENT AVEC L’INVITÉ DE COMMANDE LORS DE LA CRÉATION DU FORMULAIRE D’ENREGISTREMENT.

ROUTE /REGISTER

FUNCTION REGISTER PERMETTANT DE RETOURNER LA TEMPLATE “REGISTRATION/REGISTER.HTML.TWIG”, IMPORTER LE FORMULAIRE D’ENREGISTREMENT, ET D’HACHER LE MOT DE PASSE, PUIS DE VERIFIER SI LES CHAMPS SONT VALIDÉES, SI OUI, L’UTILISATEUR EST ENREGISTRÉ ET UN MAIL DE CONFIRMATION D’INSCRIPTION LUI EST ENVOYÉ PUIS IL EST REDIRIGÉ À LA PAGE D’ACCEUIL.

ROUTE /VERIFY/EMAIL

FUNCTION VERIFYUSEREMAIL, PERMETTANT DE RÉCUPERER L’ID DE L’UTILISATEUR, ET DE LUI ATTRIBUER UN RÔLE CLIENT SI IL VALIDE SON EMAIL, PUIS LE REDIRIGE À LA PAGE D’ACCEUIL EN LE CONNECTANT.

CONTROLEUR “PRODUCTCONTROLLER”

EXTRAIT DU CONTROLEUR

```
<?php  
  
namespace App\Controller;  
  
use App\Entity\Formation;  
use App\Entity\Lesson;  
use App\Repository\CursusRepository;  
use App\Repository\FormationRepository;  
use App\Repository\LessonRepository;  
use Doctrine\Persistence\ManagerRegistry;  
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;  
use Symfony\Component\HttpFoundation\Response;  
use Symfony\Component\Routing\Annotation\Route;  
use Doctrine\ORM\EntityManagerInterface;  
  
final class ProductsController extends AbstractController  
{  
    #[Route('/formations', name: 'app_formation')]  
    public function index(ManagerRegistry $manager, FormationRepository $formation): Response  
    {  
  
        $formation = $manager->getRepository(Formation::class)->findAll();  
  
        return $this->render('products/index.html.twig', ['formation' => $formation]);  
    }  
  
    #[Route('/formation/{id}', name: 'formation.show')]  
    public function show(int $id, EntityManagerInterface $entityManager, LessonRepository $lessonRepository, CursusRepository $cursusRepository, FormationRepository $formationRepository, EntityManagerInterface $entitymanager): Response  
    {  
        $formation = $formationRepository->find($id);  
  
        if (!$formation) {  
            throw $this->createNotFoundException('Cette formation n\'existe pas.');  
        }  
  
        $lessons = $lessonRepository->findBy(['formation' => $formation]);  
        $cursus = $cursusRepository->findBy(['formation' => $formation]);  
  
        return $this->render('products/show.html.twig', [  
            'formation' => $formation,  
            'lessons' => $lessons,  
            'cursus' => $cursus  
        ]);  
    }  
}
```

CONTROLEUR GÉNÉRÉE MANUELLEMENT AVEC L’INVITÉ DE COMMANDE.

ROUTE /FORMATIONS

**FUNCTION INDEX PERMETTANT DE RETOURNER LA TEMPLATE “PRODUCTS/INDEX.HTML.TWIG”
ET D’Y AJOUTER LES PROPRIÉTÉES DE L’ENTITÉ FORMATION**

ROUTE /FORMATIONS/{ID}

**FUNCTION SHOW PERMETTANT DE RETOURNER LA TEMPLATE “PRODUCT/SHOW.HTML.TWIG ” EN Y
AJOUTANT L’ID DE LA FORMATION AVEC LES PROPRIÉTÉES DE L’ENTITÉ FORMATION | LESSON | CURSUS
AFIN D’Y AFFICHER TOUS LES DETAILS SUR LA FORMATION SÉLECTIONNÉE**

CONTROLEUR “STUDENTCONTROLLER”

EXTRAIT DU CONTROLEUR

```
●●●
<?php
namespace App\Controller;

use App\Entity\Formation;
use App\Repository\FormationRepository;
use App\Repository\LessonRepository;
use Doctrine\Persistence\ManagerRegistry;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
use App\entity\User;
use App\entity\Student;

final class StudentController extends AbstractController
{
    #[Route('/student', name: 'app_student')]
    public function index(ManagerRegistry $manager, LessonRepository $lessonRepository, FormationRepository $formationRepository): Response
    {
        $formation = $manager->getRepository(Formation::class)->findAll();
        if (!$formation) {
            throw $this->createNotFoundException('Cette formation n\'existe pas.');
        }

        $lessons = $lessonRepository->findBy(['formation' => $formation]);
        $user = $this->getUser();

        return $this->render('student/index.html.twig', [
            'formation' => $formation,
            'lessons' => $lessons,
            'user' => $user
        ]);
    }

    #[Route('/student/{id}', name: 'formation_type')]
    public function LearningPage(int $id, ManagerRegistry $manager, LessonRepository $lessonRepository, FormationRepository $formationRepository): Response
    {
        $formation = $formationRepository->find($id);
        $formation = $manager->getRepository(Formation::class)->findAll();
        $lessons = $lessonRepository->findBy(['formation' => $formation]);

        if (!$formation) {
            throw $this->createNotFoundException('Cette formation n\'existe pas.');
        }

        $lessons = $lessonRepository->findBy(['formation' => $formation]);
        return $this->render('student/show.html.twig', [
            'formation' => $formationRepository->find($id),
            'lessons' => $lessons
        ]);
    }
}
```

CONTROLEUR GÉNÉRÉE MANUELLEMENT AVEC L'INVITÉ DE COMMANDE.

ROUTE /STUDENT

FUNCTION INDEX PERMETTANT DE RETOURNER LA TEMPLATE “STUDENT/INDEX.HTML.TWIG” EN Y IMPORTANT LES DONNÉES DES ENTITÉES FORMATION | LESSON | USER, IL AJOUTE AU FORMATION LES LEÇONS CORRESPONDANTE, ET VERIFIE SI L’USER EST BIEN ÉTUDIANT OU ADMIN AFIN D’ACCÉDER AU PANEL ÉTUDIANT.

ROUTE /STUDENT/{ID}

FUNCTION LEARNINGPAGE PERMETTANT DE RETOURNER LA TEMPLATE “STUDENT/SHOW.HTML.TWIG” EN Y IMPORTANT LES DONNÉES DES ENTITÉES FORMATION | LESSON | USER CORRESPONDANT À L’ID DE LA FORMATION.

CONTROLEUR “CHAPTERCONTROLLER”

EXTRAIT DU CONTROLEUR

```
<?php  
namespace App\Controller;  
  
use App\Entity\Formation;  
use App\Repository\FormationRepository;  
use App\Repository\LessonRepository;  
use Doctrine\ORM\EntityManagerInterface;  
use Doctrine\Persistence\ManagerRegistry;  
use Symfony\Component\HttpFoundation\Response;  
use Symfony\Component\Routing\Annotation\Route;  
use Symfony\Component\HttpFoundation\RedirectResponse;  
  
final class ChapterController extends AbstractController  
{  
    #[Route('/chapter', name: 'app_chapter')]  
    public function index(EntityManagerInterface $manager): Response  
    {  
        $formation = $manager->getRepository(Formation::class)->find(1);  
        return $this->render('chapter/index.html.twig', [ 'formation' => $formation,  
    ]);  
    }  
  
    #[Route('/chapter/{id}', name: 'chapter_by_id', id: ManagerRegistry::getManager(), LessonRepository $lessonRepository, FormationRepository $formationRepository)]  
    public function chapterById($id, ManagerRegistry $manager, LessonRepository $lessonRepository, FormationRepository $formationRepository): Response  
    {  
        //get the formation by the id  
        $formation = $formationRepository->find($id);  
  
        //get the lesson corresponding with the formation  
        $lessons = $lessonRepository->findBy(['formation' => $formation]);  
  
        return $this->render('chapter/index.html.twig', [  
            'formation' => $formation, //formation detail  
            'lessons' => $lessons, //lesson of this formation  
        ]);  
    }  
  
    // Path to give the certification  
    #[Route('/finalchapter', name: 'final_chapter')]  
    public function final(): Response  
    {  
        $user = $this->getUser();  
  
        return $this->render('chapter/finalchapter.html.twig', [ 'user' => $user ]);  
    }  
  
    #[Route('/student/certificate', name: 'student_certificate')]  
    public function addRole(EntityManagerInterface $entityManager): RedirectResponse  
    {  
        $user = $this->getUser(); // localize the connected user  
  
        if ($user) {  
            $this->addFlash('error', 'Vous devez être connecté pour obtenir la certification.');  
            return $this->redirectToRoute('app_login');  
        }  
  
        if (in_array('ROLE_CERTIFICATE', $user->getRoles())) {  
            $roles = $user->getRoles();  
            array_push($roles, 'ROLE_STUDENT');  
            $user->setRoles($roles); // new role  
            $entityManager->persist($user);  
            $entityManager->flush();  
            $this->addFlash('success', 'Vous avez obtenu votre diplôme !');  
        }  
        return $this->redirectToRoute('app_student');  
    }  
}
```

CONTROLEUR GÉNÉRÉE MANUELLEMENT AVEC L’INVITÉ DE COMMANDE.

ROUTE /CHAPTER

FUNCTION INDEX PERMETTANT DE RETOURNER LA TEMPLATE “CHAPTER/INDEX.HTML.TWIG”
ET D’AJOUTER LES DONNÉES RELIÉES À L’ENTITÉ TEMPLATE

ROUTE /CHAPTER/{ID}

FUNCTION CHAPTERBYID PERMETTANT DE RETOURNER LA TEMPLATE “CHAPTER/INDEX.HTML.TWIG”
ET D’AJOUTER LES DONNÉES RELIÉES À L’ENTITÉ FORMATION ET LESSONS AFIN DE POUVOIR
ÉTUDIER CELLE-CI

ROUTE /FINALCHAPTER

FUNCTION FINAL PERMETTANT D’AFFICHER LA TEMPLATE “CHAPTER/FINALCHAPTER.HTML.TWIG”
AVEC LES PROPRIÉTÉS DE L’UTILISATEUR AFIN D’ATTRIBUER LA CERTIFICATION

ROUTE /STUDENT/CERTIFICATE

FUNCTION ADDROLE PERMETTANT D’ATTRIBUER A L’UTILISATEUR AYANT FINI ENTIÈREMENT SA FORMATION
UNE CERTIFICATION GRÂCE AU ROLE_CERTIFICATE

CONTROLEUR “ADMINCONTROLLER”

EXTRAIT DU CONTROLEUR

```
...  
* @Route("/admin", name="app_admin")  
public function index(ManagerRegistry $manager): Response  
{  
    $formations = $manager->getRepository(Formation::class)->findAll();  
    $users = $manager->getRepository(User::class)->findAll();  
    $lessons = $manager->getRepository(Lesson::class)->findAll();  
    $payments = $this->stripeService->getPaymentsIntents();  
  
    return $this->render('admin/index.html.twig', ['users' => $users,  
    'formations' => $formations, 'lessons' => $lessons, 'payments' => $payments]);  
}  
  
#Route('/users', name='admin_user')  
public function listUsers(UserRepository $userRepository): Response  
{  
    return $this->render('admin/user.html.twig', [  
        'users' => $userRepository->findAll()  
    ]);  
}  
  
#Route('/user/edit/{id}', name='admin_user_edit')  
public function editUser(User $user, Request $request, EntityManagerInterface $em): Response  
{  
    $form = $this->createForm(UserType::class, $user);  
    $form->handleRequest($request);  
    if ($form->isSubmitted() && $form->isValid()) {  
        $em->flush();  
        $this->addFlash('success', 'Utilisateur mis à jour avec succès');  
        return $this->redirectToRoute('admin_users');  
    }  
  
    return $this->render('admin/edit_user.html.twig', [  
        'form' => $form->createView(),  
        'user' => $user,  
    ]);  
}  
  
#Route('/user/delete/{id}', name='admin_user_delete', methods: ['POST'])  
public function deleteUser(User $user, EntityManagerInterface $em): Response  
{  
    $em->remove($user);  
    $em->flush();  
    $this->addFlash('success', 'Utilisateur supprimé avec succès');  
    return $this->redirectToRoute('admin_users');  
}
```

CONTROLEUR GÉNÉRÉE MANUELLEMENT AVEC L’INVITÉ DE COMMANDE.

ROUTE /ADMIN

**FUNCTION INDEX PERMETTANT DE RETOURNER LA TEMPLATE “ADMIN/INDEX.HTML.TWIG”
PERMETTANT D’AJOUTER TOUTES LES DONNÉES DE TOUTES LES ENTITÉES AFIN D’AVOIR UN PANEL
ADMINISTRATEUR POUVANT EFFECTUER UN CRUD**

ROUTE /USERS

**FUNCTION LISTUSERS PERMETTANT DE RETOURNER LA TEMPLATE “ADMIN/USER.HTML.TWIG”
ET D’AJOUTER TOUTES LES DONNÉES DE L’ENTITÉES USER AFIN D’AFFICHER LA LISTE DES UTILISATEURS
SUR LA PAGE.**

ROUTE /USERS/EDIT/{ID}

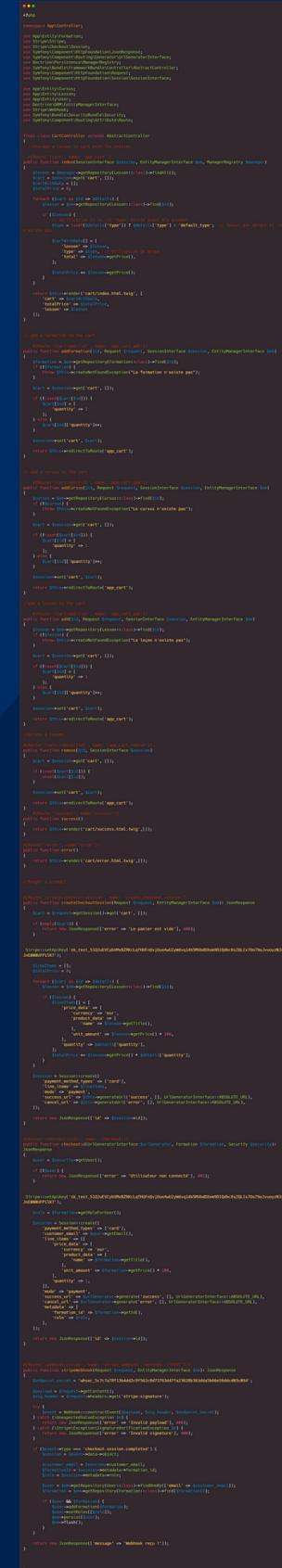
**FUNCTION EDITUSER PERMETTANT DE RETOURNER LA TEMPLATE “ADMIN/EDIT_USER.HTML.TWIG”
ET D’AJOUTER LES DONNÉES DE L’ENTITÉES USER AFIN DE POUVOIR MODIFIER LES DONNÉES DES
UTILISATEURS**

ROUTE /USERS/DELETE/{ID}

**FUNCTION DELETEUSER PERMETTANT DE RETOURNER LA TEMPLATE “ADMIN/DELETE_USER.HTML.TWIG”
ET D’AJOUTER LES DONNÉES DE L’ENTITÉES USER AFIN DE POUVOIR SUPPRIMER UN UTILISATEUR**

CONTROLEUR “CARTCONTROLLER”

EXTRAIT DU CONTROLEUR



```
class CartController extends Controller
{
    public function __construct(private EntityManagerInterface $entityManager, private SecurityManager $security, private Session $session)
    {
        parent::__construct();
    }

    public function indexAction(): Response
    {
        $cart = $this->session->get('cart');
        if ($cart === null) {
            $cart = new ArrayCollection();
            $this->session->set('cart', $cart);
        }
        $formations = $this->entityManager->getRepository(Formation::class)->findAll();
        $lessons = $this->entityManager->getRepository(Lesson::class)->findAll();
        $cursus = $this->entityManager->getRepository(Cursus::class)->findAll();

        return $this->render('cart/index.html.twig', [
            'formations' => $formations,
            'lessons' => $lessons,
            'cursus' => $cursus,
            'cart' => $cart
        ]);
    }

    public function addFormationAction(Request $request): Response
    {
        $formationId = $request->get('formation_id');
        $formation = $this->entityManager->getRepository(Formation::class)->find($formationId);

        if ($formation === null) {
            return $this->redirectToRoute('cart_index');
        }

        $cart = $this->session->get('cart');
        if ($cart === null) {
            $cart = new ArrayCollection();
            $this->session->set('cart', $cart);
        }

        $cart->add($formation);

        return $this->redirectToRoute('cart_index');
    }

    public function addCursusAction(Request $request): Response
    {
        $cursusId = $request->get('cursus_id');
        $cursus = $this->entityManager->getRepository(Cursus::class)->find($cursusId);

        if ($cursus === null) {
            return $this->redirectToRoute('cart_index');
        }

        $cart = $this->session->get('cart');
        if ($cart === null) {
            $cart = new ArrayCollection();
            $this->session->set('cart', $cart);
        }

        $cart->add($cursus);

        return $this->redirectToRoute('cart_index');
    }

    public function addLessonAction(Request $request): Response
    {
        $lessonId = $request->get('lesson_id');
        $lesson = $this->entityManager->getRepository(Lesson::class)->find($lessonId);

        if ($lesson === null) {
            return $this->redirectToRoute('cart_index');
        }

        $cart = $this->session->get('cart');
        if ($cart === null) {
            $cart = new ArrayCollection();
            $this->session->set('cart', $cart);
        }

        $cart->add($lesson);

        return $this->redirectToRoute('cart_index');
    }

    public function removeAction(Request $request): Response
    {
        $id = $request->get('id');

        $cart = $this->session->get('cart');
        if ($cart === null) {
            $cart = new ArrayCollection();
            $this->session->set('cart', $cart);
        }

        $cart->removeElement($id);

        return $this->redirectToRoute('cart_index');
    }

    public function successAction(): Response
    {
        $cart = $this->session->get('cart');
        if ($cart === null) {
            $cart = new ArrayCollection();
            $this->session->set('cart', $cart);
        }

        $this->session->remove('cart');

        return $this->redirectToRoute('cart_index');
    }

    public function checkoutSessionAction(): Response
    {
        $stripeSecretKey = env('STRIPE_SECRET_KEY');
        $stripePublicKey = env('STRIPE_PUBLIC_KEY');

        $stripe = Stripe::create([
            'secret_key' => $stripeSecretKey,
            'public_key' => $stripePublicKey
        ]);

        $cart = $this->session->get('cart');
        if ($cart === null) {
            $cart = new ArrayCollection();
            $this->session->set('cart', $cart);
        }

        $totalPrice = 0;
        foreach ($cart as $item) {
            $totalPrice += $item->getPrice();
        }

        $checkoutSession = $stripe->checkout->sessions->create([
            'line_items' => [
                [
                    'price_data' => [
                        'currency' => 'EUR',
                        'unit_amount' => $totalPrice * 100
                    ],
                    'quantity' => 1
                ]
            ],
            'mode' => 'payment',
            'success_url' => 'http://localhost:8000/cart/success',
            'cancel_url' => 'http://localhost:8000/cart/error'
        ]);

        $this->session->remove('cart');

        return $this->redirectToRoute('cart_checkout', [
            'id' => $checkoutSession->id
        ]);
    }

    public function webhookStripeAction(): Response
    {
        $stripeSecretKey = env('STRIPE_SECRET_KEY');

        $stripe = Stripe::create([
            'secret_key' => $stripeSecretKey
        ]);

        $event = $stripe->webhooks->events->retrieve($_GET['event']);

        if ($event->type === 'checkout.session.completed') {
            $checkoutSession = $event->data->object;

            $cart = $this->session->get('cart');
            if ($cart === null) {
                $cart = new ArrayCollection();
                $this->session->set('cart', $cart);
            }

            $cart->clear();

            $this->session->remove('cart');

            return $this->redirectToRoute('cart_index');
        }

        return $this->redirectToRoute('cart_index');
    }
}
```

CONTROLEUR GÉNÉRÉE MANUELLEMENT AVEC L'INVITÉ DE COMMANDE.

ROUTE /CART

FUNCTION INDEX PERMETTANT DE RETOURNER LA TEMPLATE “CART/TEMPLATE.HTML.TWIG” ET D’Y AJOUTER LES DONNÉES DES ENTITÉES FORMATION | LESSONS | CURSUS CRÉATION D’UN \$CART AVEC UNE SESSION AFIN DE GARDER DANS LA SESSION LES ARTICLES AJOUTÉES AU PANIER.

CRÉATION D’UN TABLEAU \$CARTWITHDATA AFIN D’Y AJOUTER LES DONNÉES LORS D’UN AJOUT AU PANIER, CRÉATION D’UN \$TOTALPRICE CALCULANT LA TOTALITÉ DU PRIX DES ARTICLES AJOUTÉES DANS LE PANIER.

ROUTE /CART/ADD/{ID}

FUNCTION ADDFORMATION AYANT LES DONNÉES DE L’ENTITÉ FORMATION AFIN DE POUVOIR L’AJOUTÉE AU PANIER
FUNCTION ADDCURSUS AYANT LES DONNÉES DE L’ENTITÉ CURSUS AFIN DE POUVOIR L’AJOUTÉE AU PANIER
FUNCTION ADD AYANT LES DONNÉES DE L’ENTITÉ LESSON AFIN DE POUVOIR L’AJOUTÉE AU PANIER

ROUTE /CART/REMOVE/{ID}

AFIN DE POUVOIR RETIRER LE PRODUIT SOUHAITER DANS LE PANIER ET REDIRIGER VERS LE PANIER.

ROUTE /SUCCESS

EN CAS DES SUCCÈS DE PAIEMENT, PERMET DE REDIRIGER VERS LA TEMPLATE CART/SUCCESS.HTML.TWIG

ROUTE /ERROR

EN CAS D’ÉCHEC DE PAIEMENT, PERMET DE REDIRIGER VERS LA TEMPLATE CART/ERROR.HTML.TWIG

ROUTE /CREATE-CHECKOUT-SESSION

WEBHOOK STRIPE PERMETTANT DE CRÉER UNE SESSION D’ACHAT

ROUTE /CHECKOUT/{ID}

WEBHOOK STRIPE PERMETTANT DE VOIR LE CHECKOUT CORRESPONDANT À L’ID

ROUTE /WEBHOOK/STRIPE

WEBHOOK STRIPE PERMETTANT D’ATTRIBUER LE RÔLE ÉTUDIANT AINSI QUE LA FORMATION QUE L’UTILISATEUR À PAYÉ

PRÉSENTATION DES DISPOSITIFS DE SÉCURITÉ



DISPOSITIF MIS EN PLACE AVEC SYMFONY



. GESTION DES UTILISATEURS ET AUTHENTIFICATION :

- **SECURITY BUNDLE :** SYMFONY UTILISE LE SECURITYBUNDLE POUR GÉRER LA SÉCURITÉ DES UTILISATEURS. IL PERMET DE CONFIGURER DES SYSTÈMES D'AUTHENTIFICATION (COMME FORMULAIRE D'AUTHENTIFICATION, AUTHENTIFICATION PAR JETON, ETC.).
- **LOGIN SÉCURISÉ :** SYMFONY OFFRE UNE AUTHENTIFICATION BASÉE SUR DES SESSIONS, DES COOKIES SÉCURISÉS (AVEC L'OPTION HTTPONLY ET SECURE POUR LES COOKIES DE SESSION), ET LA POSSIBILITÉ DE CONFIGURER DES MÉCANISMES D'AUTHENTIFICATION PERSONNALISÉS.
- **GESTION DES RÔLES :** SYMFONY PERMET DE DÉFINIR DES RÔLES D'UTILISATEURS ET DES CONTRÔLES D'ACCÈS EN FONCTION DE CES RÔLES AVEC LA CONFIGURATION DE FIREWALLS ET DE ACCESS_CONTROL.
-

PROTECTION CONTRE LES ATTAQUES CSRF (CROSS-SITE REQUEST FORGERY) :

- SYMFONY PROTÈGE AUTOMATIQUEMENT LES FORMULAIRES CONTRE LES ATTAQUES CSRF EN GÉNÉRANT UN JETON UNIQUE POUR CHAQUE FORMULAIRE. CE JETON EST ENSUITE VÉRIFIÉ LORS DE LA SOUMISSION DU FORMULAIRE, EMPÊCHANT AINSI LES ATTAQUES QUI TENTENT DE SOUMETTRE DES FORMULAIRES MALVEILLANTS À L'INSU DE L'UTILISATEUR.
-

3. VALIDATION ET ASSAINISSEMENT DES DONNÉES :

- **VALIDATIONS CÔTÉ SERVEUR :** SYMFONY OFFRE DES MÉCANISMES DE VALIDATION DES DONNÉES, CE QUI PERMET DE VÉRIFIER ET DE FILTRER LES DONNÉES ENTRANTES AVANT DE LES ENREGISTRER EN BASE DE DONNÉES.
- **PROTECTION CONTRE LES INJECTIONS SQL :** SYMFONY UTILISE L'ORM DOCTRINE, QUI PROTÈGE CONTRE LES ATTAQUES PAR INJECTION SQL EN PARAMÉTRANT AUTOMATIQUEMENT LES REQUÊTES.
-

PROTECTION CONTRE LES ATTAQUES XSS (CROSS-SITE SCRIPTING) :

- SYMFONY ÉCHAPPE AUTOMATIQUEMENT LES SORTIES DANS LES VUES TWIG POUR ÉVITER LES ATTAQUES XSS. PAR EXEMPLE, LES VARIABLES AFFICHÉES DANS LES TEMPLATES SONT AUTOMATIQUEMENT ÉCHAPPÉES, SAUF SI ELLES SONT EXPLICITEMENT MARQUÉES COMME SÛRES.

SÉCURISATION DES SESSIONS :

- SYMFONY FOURNIT UNE GESTION ROBUSTE DES SESSIONS, Y COMPRIS LA POSSIBILITÉ DE CONFIGURER LA DURÉE DE VIE DES SESSIONS, DE SÉCURISER LES COOKIES DE SESSION (VIA LES OPTIONS SECURE ET HTTPONLY), ET DE FORCER L'UTILISATION DE COOKIES UNIQUEMENT VIA HTTPS.

CONFIGURATION DES FIREWALLS ET DES ZONES DE SÉCURITÉ :

- SYMFONY PERMET DE CONFIGURER DES FIREWALLS ET DE DÉFINIR DES ZONES DE SÉCURITÉ, CE QUI PERMET DE PROTÉGER CERTAINES PARTIES DE L'APPLICATION (COMME LES ZONES ADMINISTRATIVES) PAR UN CONTRÔLE D'ACCÈS STRICT.

ENCODAGE DES MOTS DE PASSE :

- SYMFONY UTILISE DES ALGORITHMES DE HACHAGE DE MOTS DE PASSE ROBUSTES, COMME BCRYPT OU ARGON2, POUR GARANTIR QUE LES MOTS DE PASSE DES UTILISATEURS SONT STOCKÉS DE MANIÈRE SÉCURISÉE.



PAGE “REGISTER”



POUR S'INSCRIRE SUR LE SITE, PLUSIEURS VERIFICATION SON NÉCESSAIRE.

- L'EMAIL DOIT-ÊTRE UNIQUE.
- LE MOT DE PASSE DOIT-ÊTRE RÉPÉTÉE DEUX FOIS ET DOIT CORRESPONDRE

SI SES VÉRIFICATIONS SONT OBTENUES L'UTILISATEUR RECEVRA UN MAIL AFIN DE VALIDER SON INSCRIPTION ET ACCÉDER AUX SERVICES.
SI L'UTILISATEUR NE REMPLIT PAS SES CONDITIONS IL NE SERA PAS INSCRIT.

LORS DE L'INSCRIPTION LE MOT DE PASSE EST HACHÉ AFIN D'ÊTRE IMPOSSIBLE D'ÊTRE RÉCUPÉRER.



PAGE “LOGIN”

POUR SE CONNECTER AU SITE L’UTILISATEUR DOIT :

- **AVOIR LE BON IDENTIFIANT**
- **AVOIR LE BON MOT DE PASSE**

MAIS IL DOIT AUSSI AVOIR :

- **UN CSRF_TOKEN**

**CELUI-CI EST CACHÉ DANS LE FORMULAIRE ET UNIQUE À CHAQUE UTILISATEUR AFIN D’ÉVITER LES ACTIONS NE PROVENANT PAS DU FORMULAIRE,
POUVANT-ÊTRE MALVEILLANTES (SE CONNECTER...)**

PAGE “ADMIN”

**POUR ACCÉDER À LA PAGE ADMIN AINSI QUE TOUTES LES AUTRES PAGES DU PANEL ADMINISTRATEUR
L’UTILISATEUR DOIT ÊTRE OBLIGATOIREMENT ROLE_ADMIN, SINON, IL SE VERRA ÊTRE REDIRIGÉ VERS LA PAGE D’ACCUEIL.**

PAGE “STUDENT”

**POUR ACCÉDER À LA PAGE STUDENT AINSI QUE TOUTES LES AUTRES PAGES DU PANEL ETUDIANT
L'UTILISATEUR DOIT ÊTRE OBLIGATOIREMENT ROLE_STUDENT, SINON, IL SE VERRA ÊTRE REDIRIGÉ VERS LA PAGE D'ACCEUIL.**

PAGE D'ACHAT WEBHOOK STRIPE

AFIN D'ACHETER LES ARTICLES STRIPE À MIS EN PLACE UN DISPOSITIF DE SÉCURITÉ

**STRIPE_SECRET_KEY: CLÉ UNIQUE POUR L'API AFIN D'ÉVITER LES ACTIONS MALVEILLANTES
LORS D'UN ACHAT.**

Technologies utilisées

FRONT-END



Symfony



TWIG



VERSIONNING



BACK-END



Symfony



RÉALISATION D'UN ACHAT TEST

COMMENT FAIRE ?

QUE CE PASSE T'IL LORS D'UN ACHAT ?

RÉALISATION D'UN ACHAT TEST

ÉTAPE 1: SÉLECTION DE L'ARTICLE À ACHETER

The screenshot shows a user interface for an e-commerce site. At the top, there is a navigation bar with links: Accueil, Boutique, Panier, Élèves, and Se déconnecter. Below the navigation is a logo featuring a stylized book icon and the word "KNOWLEDGE". The main content area is titled "Mon Panier". It displays a single item in the cart: "Leçon n°1 : Les langages Html et CSS" (Lesson 1: HTML and CSS Languages). The item details include Type : technology and Prix : 32 €. To the right of the item, there are two buttons: "Retirer du panier" (Remove from cart) and a larger button labeled "TOTAL : 32,00 €". Below these buttons is a red button labeled "FINALISER MA COMMANDE" (Finalize my order).

Ici, j'ai choisis d'acheter la leçon 1 de la formation en informatique.
Puis, je clique sur “Finaliser ma commande”

RÉALISATION D'UN ACHAT TEST

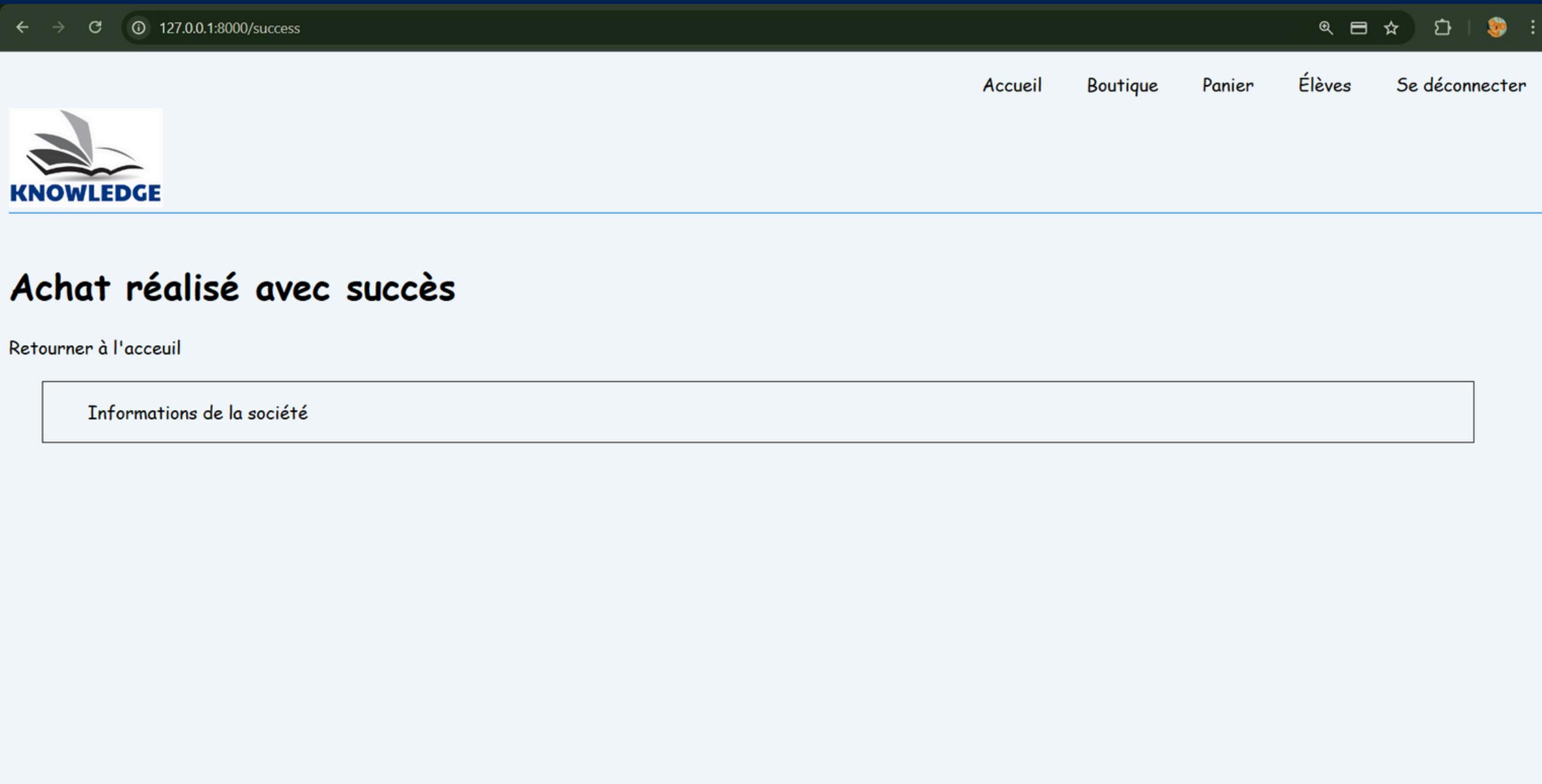
ÉTAPE 2 : ENTRER LES DONNÉES DE TEST.

The screenshot shows a payment interface for a course titled "Leçon n°1 : Les langages Html et CSS" priced at 32,00 €. The interface includes a "TEST MODE" indicator. On the right, there is a green button labeled "Payer par link". Below it, there are fields for "Courriel" (kawzex@outlook.fr), "Informations de la carte" (4242 4242 4242 4242, VISA logo), and "Nom du titulaire de la carte" (Yoann). A dropdown menu for "Pays ou région" is set to "France". At the bottom, there is a checkbox for "Enregistrer mes informations en toute sécurité pour pouvoir payer mes prochains achats en un clic" with explanatory text about Stripe's documentation. A blue "Payer" button is at the bottom right.

Ici, nous aperçevons qu'il y a bien l'article sélectionnée en haut à gauche, au bon montant.
(Et surtout que le mode TEST est activé, celui-ci permet d'utiliser les cartes bancaires factice
fournis avec la documentation STRIPE)

RÉALISATION D'UN ACHAT TEST

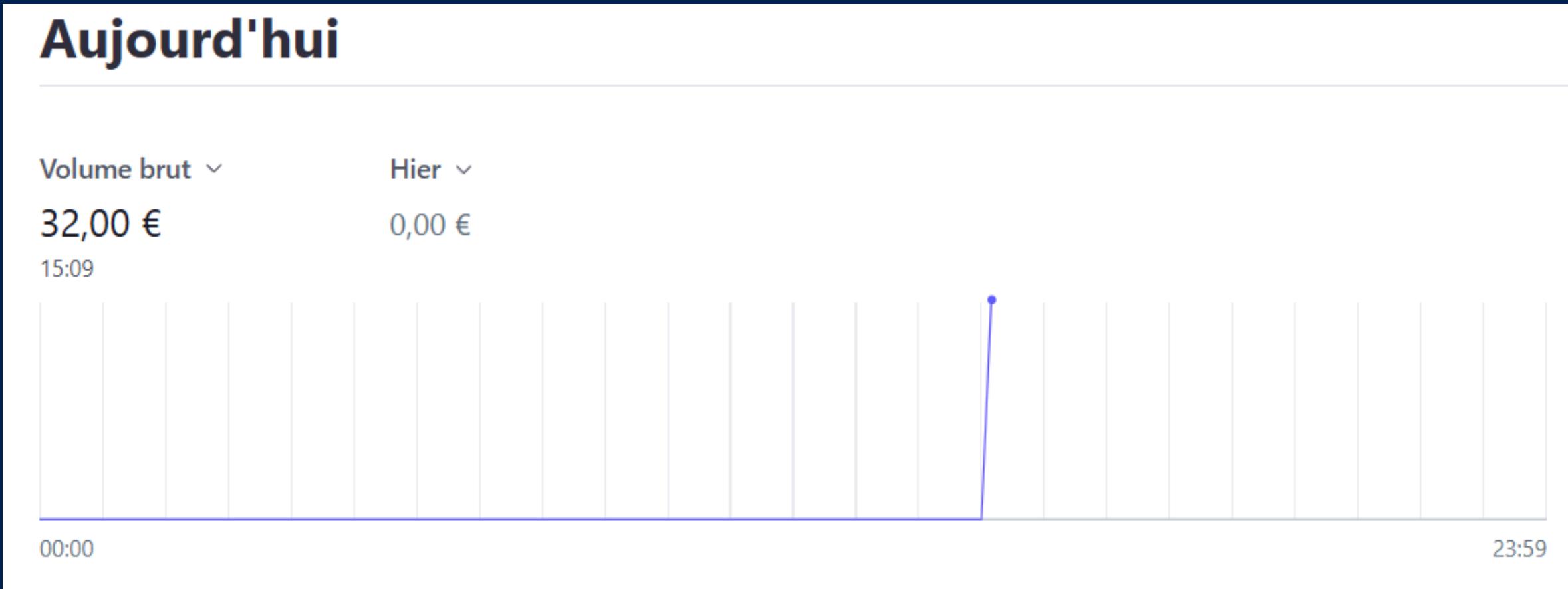
ÉTAPE 3 : L'ACHAT À ÉTAIS RÉALISER AVEC SUCCÈS.



Ici, l'achat à étais validé donc nous sommes rediriger vers l'url /success

RÉALISATION D'UN ACHAT TEST

VÉRIFICATION SI L'ACHAT À BIEN ÉTAIS PRIS EN COMPTE SUR LE TABLEAU DE BORD STRIPE.



La somme à étais versé, il à bien étais réaliser.

RÉALISATION D'UN ACHAT TEST

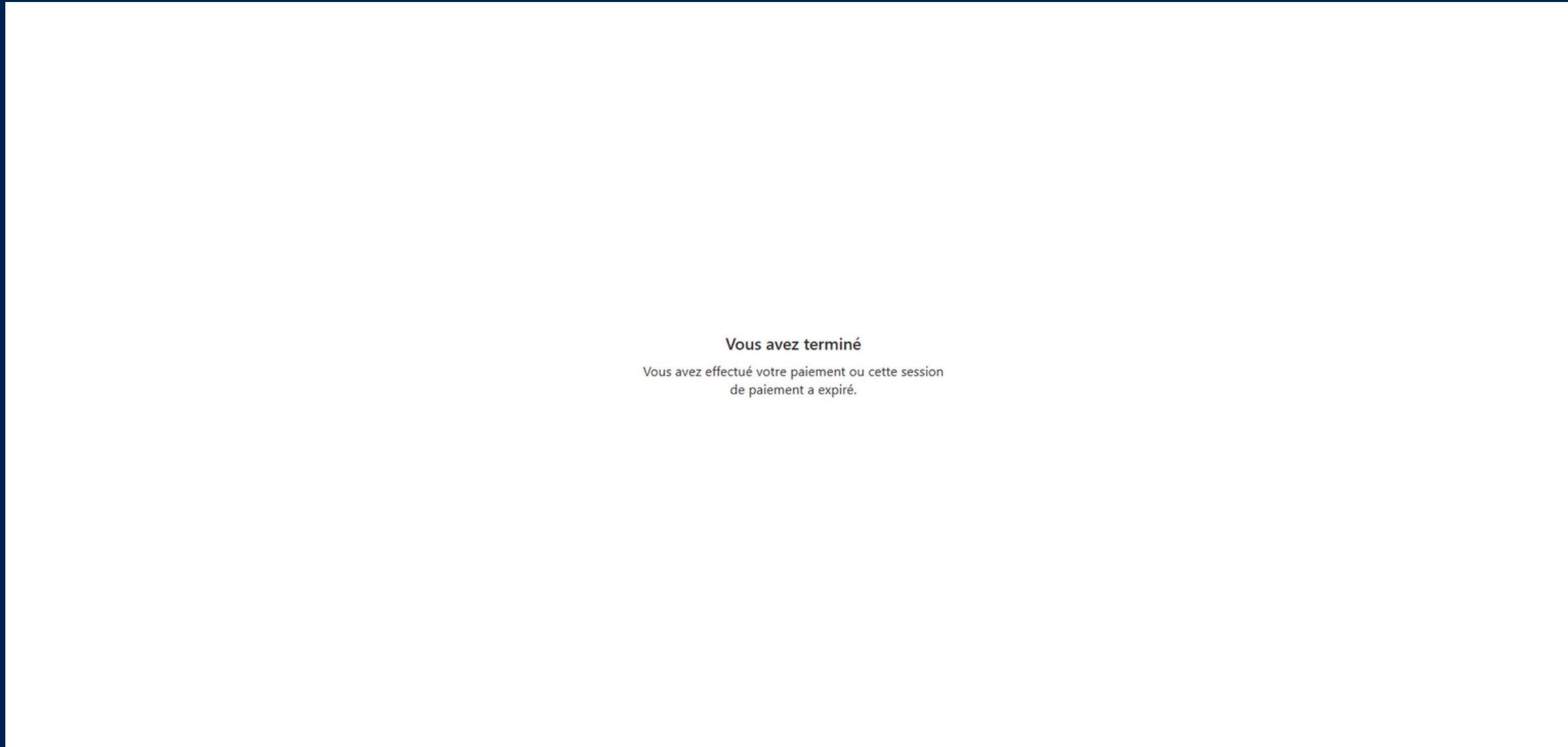
QUE SE PASSE T'IL EN CAS D'ÉCHEC ?

The screenshot shows a payment interface for a purchase of 16,00 € for a lesson titled "Leçon n°1 : Les outils du jardinier". The top left corner indicates "STUBBORN TEST MODE". On the right, there is a green button labeled "Payer par link" and an "Ou" option for email input, which contains "kawzex@outlook.fr". Below this is a section for card information with fields for number ("4242 4242 4242 4248"), expiration ("02 / 27"), and CVC ("777"). A red error message states "Votre numéro de carte n'est pas valide." Below these fields are fields for "Nom du titulaire de la carte" (Yoann) and "Pays ou région" (France). At the bottom, there is a checkbox for saving payment info and a blue "Payer" button.

Ici, la carte bancaire n'est pas bonne, donc impossibilité de cliquer sur "Payer"

RÉALISATION D'UN ACHAT TEST

QUE SE PASSE T'IL EN CAS D'ÉCHEC ?



Ici l'utilisateur est parti puis revenu sur la session OU sa connexion à échoué/expiré.
Le paiement ne s'effectuera pas, l'utilisateur devra recommencé la manipulation.

RÉALISATION D'UN ACHAT TEST

QUE SE PASSE T'IL EN CAS D'ÉCHEC ?

The screenshot shows a payment interface for a purchase of 16,00 € for a lesson titled "Leçon n°1 : Les outils du jardinier". The top right corner indicates "TEST MODE". On the right, there is a green button labeled "Payer par link". Below it, there is a "Courriel" field containing "kawzex@outlook.fr". Under "Informations de la carte", the card number is shown as "4000 0000 0000 9995", with the last four digits "9995" highlighted in red. The expiration date is "10 / 26" and the CVV is "625". A message below the card details states: "Votre carte de crédit a été refusée en raison de fonds insuffisants. Essayez de payer avec une carte de débit." In the "Nom du titulaire de la carte" field, the name "Yoann" is entered. The "Pays ou région" dropdown is set to "France". At the bottom, there is a checkbox for "Enregistrer mes informations en toute sécurité pour pouvoir payer mes prochains achats en un clic" with the explanatory text: "Réglez plus rapidement vos achats auprès de STUBBORN et de tous les autres professionnels qui acceptent Link.". A blue "Payer" button is at the bottom.

Ici l'utilisateur n'as plus les fonds nécessaires pour payer avec sa carte..
Donc il devra utiliser une autre carte.

RÉALISATION D'UN ACHAT TEST

QUE SE PASSE T'IL EN CAS D'ÉCHEC ?

The screenshot shows a payment interface for a purchase of "Leçon n°1 : Les outils du jardinier" worth 16,00 €. The top left shows the STUBBORN logo with "TEST MODE" indicated. The top right features a green button labeled "Payer par link". Below it, there's an "Ou" option followed by a "Courriel" field containing "kawzex@outlook.fr". Under "Informations de la carte", a card number "4000 0000 0000 9979" is entered, along with the expiration date "10 / 26" and the CVV "625". A red error message "Votre carte a été refusée." is displayed above the card fields. Further down, the "Nom du titulaire de la carte" is set to "Yoann", "Pays ou région" is set to "France", and there's a checkbox for "Enregistrer mes informations en toute sécurité pour pouvoir payer mes prochains achats en un clic". A blue "Payer" button is at the bottom.

Ici, la carte à étais déclarer comme volé, donc non valide pour acheter.
Elle est immédiatement refusé.

CONCLUSION

CE QU'IL FAUT RETENIR ?

POINTS FORT DE L'APPLICATION ?

POINTS FAIBLE ?

POINTS FORT DE KNOWLEDGE

- **FACILE D'UTILISATION**
- **CODE COMMENTÉ DONC FACILE À ÊTRE MIS À JOUR.**
- **SYSTÈME DE RÔLE**
- **SYSTÈME D'AUTORISATION**
- **SYSTÈME D'E-LEARNING**
- **SYSTÈME D'ACHAT**
- **SYSTÈME DE COOKIES ET TOKENS**
- **SYSTÈME DE VÉRIFICATION DE COMPTE**
- **CRUD FORMATIONS | LEÇON | CURSUS | INITIATION | USERS**

POINTS FAIBLE DE KNOWLEDGE

- NÉCESSITE UNE SÉCURITÉ PLUS APPROFONDIE (ANTI-DDOS).
- NÉCESSITE UN HEBERGEUR ET UNE BASE DE DONNÉES SOLIDE.

CE QU'IL FAUT RETENIR DE KNOWLEDGE

KNOWLEDGE EST UNE APPLICATION SIMPLE D'ACCÈS AVEC UNE PROTECTION CLASSIQUE, PERMETTANT À UN UTILISATEUR VÉRIFIÉ D'EFFECTUER UN ACHAT, ET D'ÉTUDIER EN TOUTES SÉCURITÉS. ELLE PERMET AUSSI À UN UTILISATEUR ADMINISTRATEUR DE MAINTENIR À JOUR L'APPLICATION DEPUIS SON PANEL ADMINISTRATEUR.



MERCI DE M'AVOIR
ÉCOUTÉ



KNOWLEDGE