

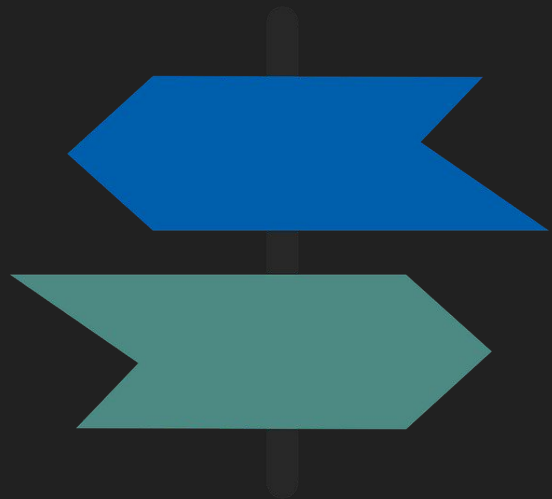


audiophile



Sommaire

1. Présentation du projet
2. Conception
3. Fonctionnalités
4. Démonstration
5. Réalisations et extraits de code
6. Situation de travail ayant
7. Conclusion



1

Présentation du projet

Le projet audiophile



Présentation du projet



Les besoins du client

- Haute qualité sonor
- Connexion sans fil
- Livraison à l'adresse de son choix
- Matériel professionnel et grand public



Les personnes ciblées

- Les audiophiles (entre 25 ans et plus)
- Les voyageurs (entre 18 ans et 50 ans)
- Les sportifs (entre 18 ans et 35 ans)
- Les streamers (entre 18 ans et 35 ans)



Les objectifs

- Accroître la visibilité
- Rendre les produits plus facilement accessible
- Augmenter le nombre de vente
- Acquérir de nouveaux clients
- Fidéliser la clientèle
- Mieux connaître la clientèle



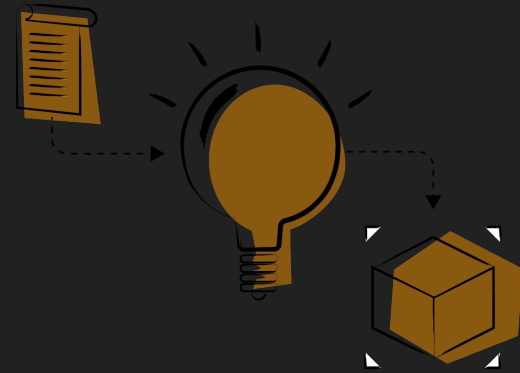
Les principaux concurrents

- amazon.fr
- fnac.fr
- darty.fr
- boulanger.fr
- bose.fr

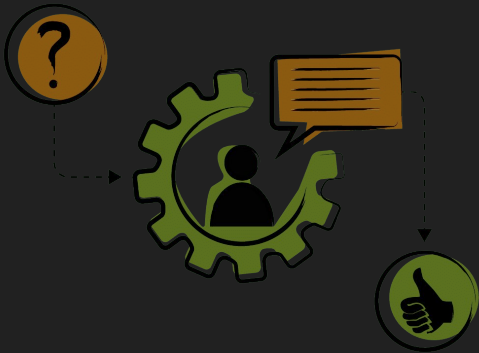




2



Conception



Conception

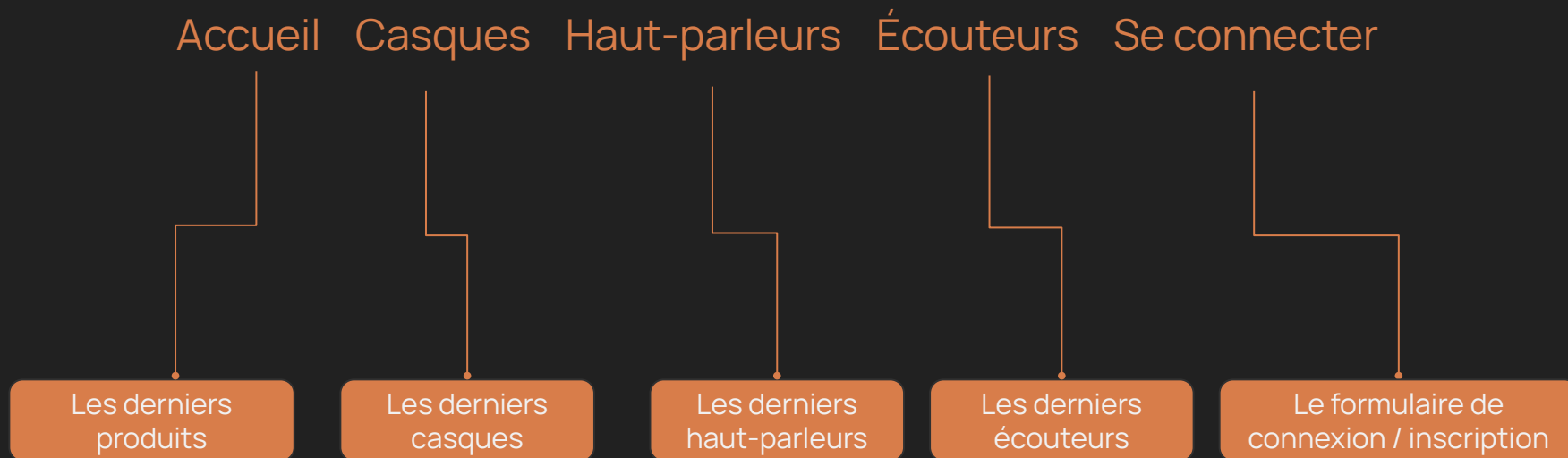
Plan du site

Identité graphique

Wireframe

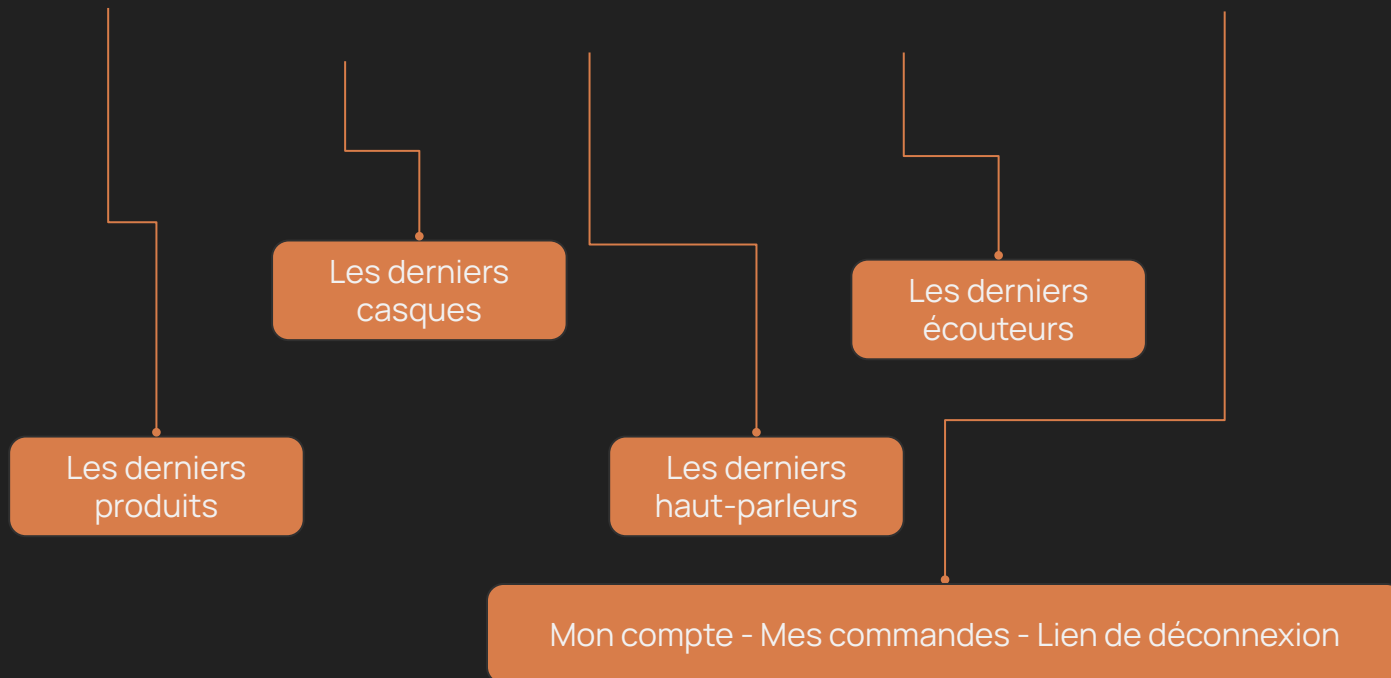


Plan du site - les utilisateurs non connectés



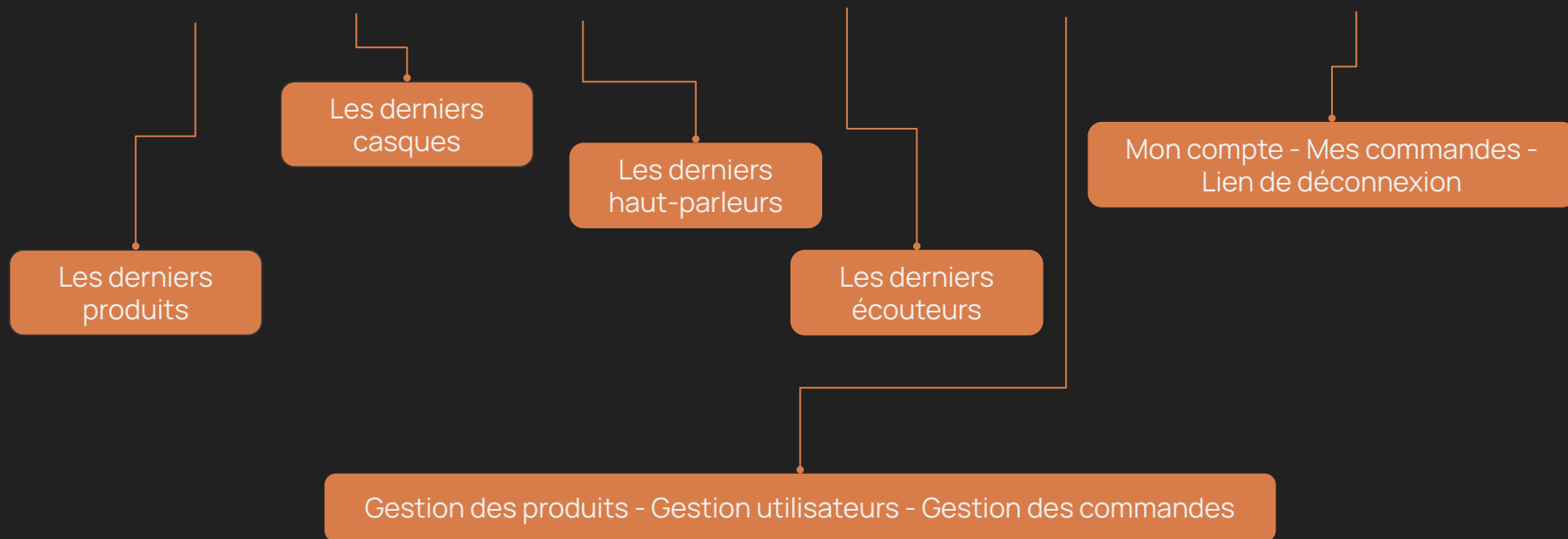
Plan du site - les utilisateurs connectés

Accueil casques haut-parleurs écouteurs nomUtilisateur



Plan du site - les administrateurs

Accueil Casques Haut-parleurs Écouteurs Espace admin nomUtilisateur



L'identité graphique

Couleurs

#D87D4A

#101010

#F1F1F1

#FAFAFA

#FBAF85

#FFFFFF

#000000

Boutons

VOIR PRODUIT

VOIR PRODUIT

VOIR PRODUIT

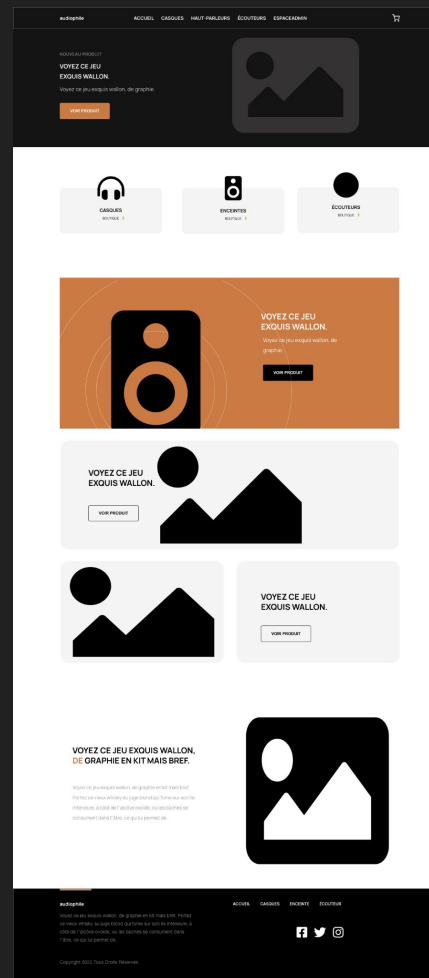
VOIR PRODUIT

Typographie

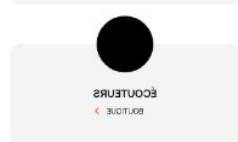
MANROPE

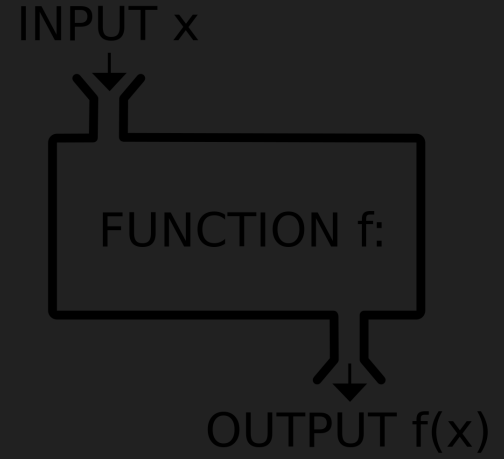
AZERTYUIOPQSDFGHJKLMWXCVRN
azertyuiopqsdfghjklmwxvbn
0123456789!@#\$%^&*()_+

Wireframe desktop



Wireframe mobile





Fonctionnalités

$$C = \frac{2\pi\epsilon l}{\ln\left(\frac{R_2}{R_1}\right)}$$

Fonctionnalités

Principales fonctionnalités

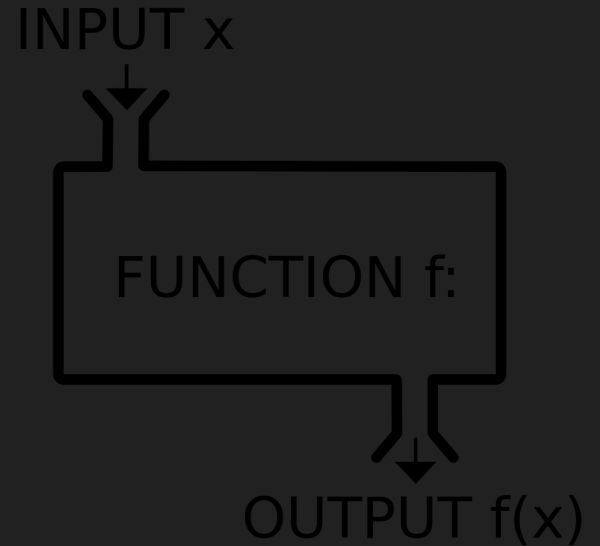


Le MCD

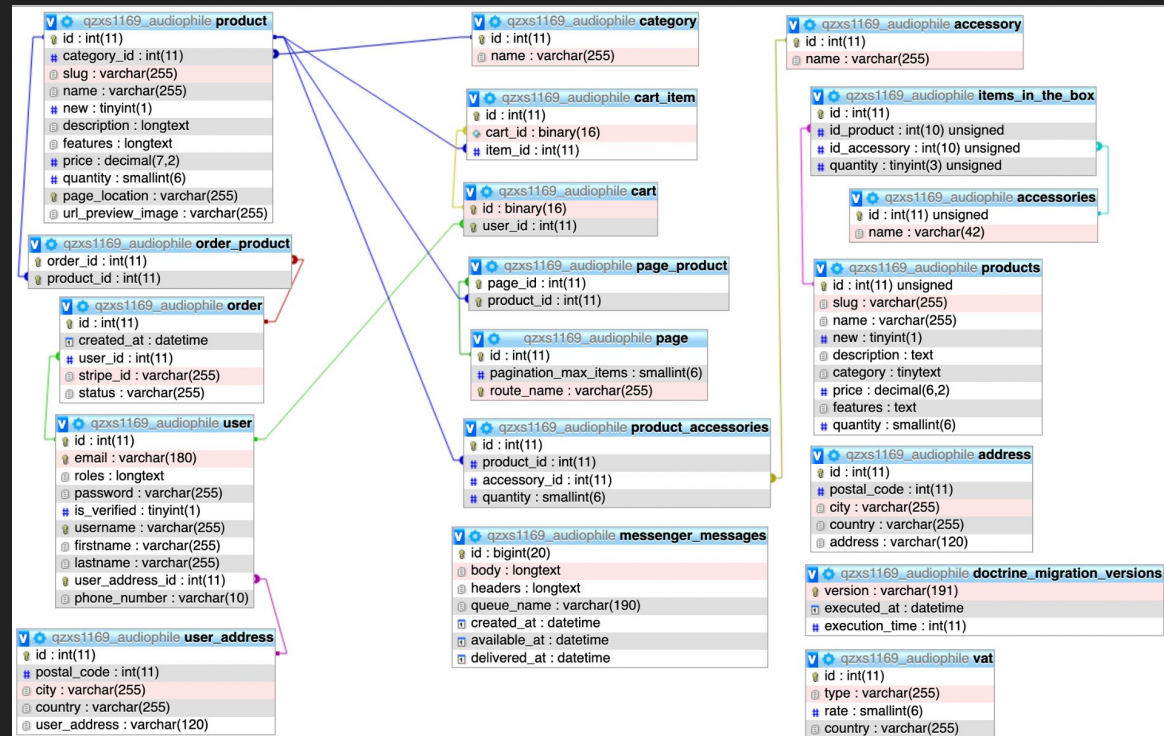


Les fonctionnalités principales

- Formulaire d'inscription/connexion
- Espace client
- Espace administrateur
- Formulaire de contact
- Panier
- Paiement



Le MCD



4

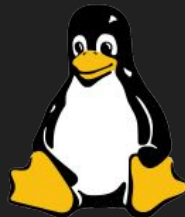
Démonstration





5

Extraits du code source



Extraits du code source

Front-end



Back-end



Front-end - index des produits

L'attribut `pageLocation` détermine l'emplacement du produit sur la page et `fileNameSuffix` détermine s'il s'agit d'un mobile ou d'un desktop.

```
{# Produit| vedette. #}

{% for product in products %}
  {% if product.pageLocation == 'featured-product' %}
    <section class="index-section-featured-product">
      <article
        style="background-image: url('{{ 'upload/images/products/index/' ~ product.slug ~ fileNameSuffix }}');"
        {# Première colonne. #}

        <div class="col">
          {% if product.new %}
            <strong class="tag-new-product-light-gray">{% trans %}new product{% endtrans %}</strong>
          {% endif %}
          <h2>{{ product.slug|replace({'-': ' '}) }}</h2>
          <p>{{ product.description }}</p>
          <a class="btn btn-default-1" href="{{ path('app_product_detail', { 'productSlug': product.slug }) }}">{% trans %}see product{% endtrans %}</a>
        </div>

      </article>
    </section>
  {% endif %}
{% endfor %}
```


Front-end - produits indexés par catégorie

```
{% if products is not null and category is not null %}
<div id="products-by-category-title">{{ category.name|trans({'%headphones%':category.name }) }}</div>
<div class="products-by-category-container">
    {% for product in products|reverse %}
        <article id="{{ product.id }}" class="products-by-category">
            
            {#  #}
            <div class="products-by-category-content-container">
                {% if product.new %}<strong class="tag-new-product-orange">{% trans %}new product{% endtrans %}</strong>{% endif %}
                <h2>{{ product.name }}</h2>
                <p>{{ product.description }}</p>
                <a class="btn btn-default-1" href="{{ path('app_product_detail', { 'productSlug': product.slug }) }}">{% trans %}see product{% endtrans %}</a>
            </div>
        </article>
    {% endfor %}
</div>
{% endif %}
{% include "_includes/_section-company-presentation.html.twig" %}
```

Back-end

```
<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class UserController extends AbstractController
{
    #[Route('/{_locale}/{username}/profil', name: 'app_user_index', requirements: ['_locale' => 'en|fr'])]
    public function profile(): Response
    {
        return $this->render('user/base.html.twig', [
            'controller_name' => 'UserController',
        ]);
    }

    #[Route('/{_locale}/{username}/commandes', name: 'app_user_orders_index', requirements: ['_locale' => 'en|fr'], defaults: ['_locale' => 'en|fr'])]
    public function ordersIndex(): Response
    {
        return $this->render('user/base.html.twig', [
            'controller_name' => 'UserController',
        ]);
    }
}
```

Back-end

Le service pour calculer la TVA

```
!<?php
```

```
namespace App\Service;
```

```
class TavManager
```

```
{
```

```
    const STANDARD_RATE = (20 / 100);
```

```
    const INTERMEDIATE_RATE = (10 / 100);
```

```
    const REDUCED_RATE = (5.5 / 100);
```

```
    const PARTICULAR_RATE = (2.1 / 100);
```

```
    /**
```

```
     * Obtenir le prix TTC.
```

```
     *
```

```
     * @return void
```

```
     */
```

```
    public function getPriceIncludingTax($price, $rate)
```

```
    {
```

```
        return $price * ($rate / 100);
```

```
    }
```

```
    /**
```

```
     * Obtenir le prix HT.
```

```
     *
```

```
     * @return void
```

```
     */
```

```
    public function getPriceExcl($price, $rate)
```

```
    {
```

```
        return $price / ($rate / 100);
```

```
    }
```

```
}
```

Back-end

L'ajout de nouvelles fonctions à Twig

src > Twig > AppExtension.php > AppExtension > getPriceIncludingTax

```
27
28 /**
29  * Retourne la variable d'environnement demandée.
30  *
31  * @param String $varname
32  * @return String
33  */
34 public function getEnvironmentVariable($varname)
35 {
36     return $_ENV[$varname];
37 }
38
39 /**
40  * Retourne le prix TTC d'un produit.
41  *
42  * @param [type] $price
43  * @param [type] $rate
44  * @return void
45  */
46 public function getPriceIncludingTax($price, $rate)
47 {
48     return $this->tavManager->getPriceIncludingTax($price, $rate);
49 }
50
51 /**
52  * Retourne le prix HT d'un produit.
53  *
54  * @param [type] $price
55  * @param [type] $rate
56  * @return void
57  */
58 public function getPriceExcl($price, $rate)
59 {
60     return $this->tavManager->getPriceIncludingTax($price, $rate);
61 }
62 }
```

Back-end

CRUD - Create

```
/**
 * Méthode utilisée pour créer le panier.
 * L'appel à cette méthode est réalisé via la méthode load du contrôleur CartController.
 */
* @return void
*/
private function createCart()
{
    // Récupère un entityManager et l'utilisateur courant ou null si l'utilisateur n'est pas connecté.

    $em = $this->doctrine->getManager();
    $user = $this->getUser();

    // Crée une nouvelle instance (instancie) de la classe Cart.

    $cart = new Cart;

    // Crée une nouvelle instance de la classe Uuid en utilisant le constructeur pour déterminer quel type d'uuid associer à l'instance.

    $uuid = new Uuid(Uuid::v1());

    // Fix l'id du nouveau panier.

    $cart->setId($uuid->toBinary());

    // Si l'utilisateur est connecté, alors associe le panier à l'utilisateur, sinon persiste le panier directement.

    if ($user !== null) {
        $cart->setUser($user);
    }

    $em->persist($cart);

    $em->flush();

    // Retourne une réponse au client en lui indiquant que la requête a été un succès et que le panier a bien été créé.

    $response = new Response(Response::HTTP_CREATED);

    // Utilise les entêtes pour créer un nouveau cookie pour stocker l'id du panier.

    $response->headers->setCookie(Cookie::create('cartId', $uuid));

    $response->send();

    return $response;
}
```

Back-end

CRUD - Read

```
#[Route('/load', name: 'get_cart', methods: ['GET'])]  
public function load(Request $request): JsonResponse  
{  
    $cartRepository = $this->doctrine->getRepository(Cart::class); // Récupère le repository du panier.  
  
    $cartId = $request->cookies->get('cartId'); // Récupère l'uuid du panier depuis le cookie.  
  
    $cart = $cartRepository->findOneBy(['id' => $cartId]); // Récupère le panier avec l'uuid du cookie.  
  
    if ($cart !== null && $cart->getUser() === null) { // Si le panier existe déjà et qu'il n'est pas associé à un utilisateur.  
        $user = $this->getUser();  
  
        if ($user !== null) { // Si l'utilisateur est connecté.  
            $em = $this->doctrine->getManager();  
  
            $cart->setUser($user);  
  
            $em->persist($cart);  
            $em->flush();  
        }  
  
        /*  
        | Covertir le uuid du panier en binaire pour le passer la requête SQL.  
        */  
  
        $cartId = new Uuid($cartId);  
  
        $cartId = $cartId->toBinary();  
  
        /*  
        | Récupère les produits du panier.  
        */  
  
        $cartItems = $cart->getCartItems();  
  
        $json = array();  
  
        /*  
        | Récupère les données liées aux produits du panier et compte le nombre de fois qu'un produit est présent au panier.  
        */  
        foreach ($cartItems as $cartItem) {  
            $query = $this->doctrine->getConnection()->prepare('SELECT * FROM cart_item WHERE cart_id = ? AND item_id = ?');  
  
            $itemId = $cartItem->getItem()->getId();  
  
            $query->bindParam(1, $cartId);  
            $query->bindParam(2, $itemId);  
            $stmt = $query->execute();  
  
            $product['id'] = $cartItem->getItem()->getId();  
            $product['name'] = $cartItem->getItem()->getName();  
            $product['log'] = $cartItem->getItem()->getLog();  
            $product['price'] = $cartItem->getItem()->getPrice();  
            $product['category']['name'] = $cartItem->getItem()->getCategory()->getName();  
            $product['quantity'] = $stmt->rowCount();  
  
            array_push($json, $product);  
        }  
  
        /*  
        | Supprime les doublons.  
        */  
  
        $json = array_map("unserialize", array_unique(array_map("serialize", $json)));  
  
        $newJson = array();  
  
        foreach ($json as $product) {  
            array_push($newJson, $product);  
        }  
  
        return $this->json($newJson);  
    }  
  
    /*  
    | Vérifi si l'utilisateur a déjà un panier.  
    */  
  
    $cartFromUser = $this->getUser() ? $cartRepository->findOneBy(['user' => $this->getUser()]) : null; // Recherche le panier par son utilisateur, retourne le panier s'il existe ou nul dans le cas contraire.  
  
    if ($cart === null || !isset($cart) && $cartFromUser !== null) {  
        $this->createCart();  
    }  
}
```

Back-end

CRUD - Update

```
#[Route('/add/product', name: 'add_product', methods: ['POST'])]
public function add(Request $request): Response
{
    $productId = (int) $request->get('productId'); // Récupère l'id du produit.
    $cartId = $request->cookies->get('cartId'); // Récupère l'id du panier depuis le cookie.

    $uuid = new Uuid($cartId);

    $uuid = $uuid->toBinary(); // Converti l'uuid en binaire.

    $cartRepository = $this->doctrine->getRepository(Cart::class); // Récupère le repository du panier.
    $cart = $cartRepository->findOneBy(['id' => $uuid]); // Recherche le panier par son uuid.

    if ($cart !== null) { // Si le panier existe.

        $productRepository = $this->doctrine->getRepository(Product::class); // Récupère le repository de l'entity product.

        $product = $productRepository->findOneBy(['id' => $productId]); // Récupère le produit à ajouter au panier.

        if ($product !== null) { // Si le produit existe, alors ajoute le au panier.

            $cartItem = new CartItem(); // Crée une nouvelle instance de CartItem pour créer une relation de type ManyToMany sans les contraintes d'unicités afin de pouvoir gérer la quantité.
            $cartItem->setCart($cart);
            $cartItem->setItem($product);

            $cart->addCartItem($cartItem);

            $em = $this->doctrine->getManager();

            $em->persist($cartItem);

            $em->flush();

            return new Response('', Response::HTTP_CREATED);
        }
    }

    return new Response('', Response::HTTP_NOT_FOUND);
}
```

Back-end

CRUD - Delete

```
#[Route('/empty', name: 'empty', methods: ['DELETE'])]
public function empty(Request $request)
{
    $cartId = $request->cookies->get('cartId'); // Récupère l'id du panier depuis le cookie.

    $uuid = new Uuid($cartId);

    $uuid = $uuid->toBinary(); // Converti l'uuid en binaire.

    $cartRepository = $this->doctrine->getRepository(Cart::class); // Récupère le repository du panier.

    $cart = $cartRepository->find($uuid);

    if ($cart !== null) { // Si le panier existe.
        $em = $this->doctrine->getManager(); // Récupère le service entityManager.

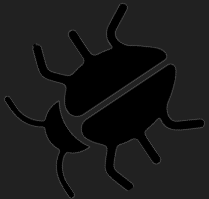
        $em->remove($cart); // Fait appel à la méthode remove de l'entityManager en lui passant en paramètre le panier à supprimer.

        $em->flush(); // Commit les modifications pour mettre à jour la base de données.

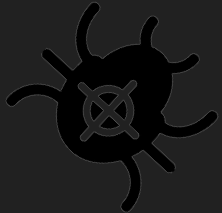
        return new Response('Panier supprimé.', Response::HTTP_OK);
    }

    $response = new Response();

    return new Response('', Response::HTTP_OK);
}
```

Situations de travail ayant nécessité une recherche



Situations de travail ayant nécessité une recherche

Le problème

Recherches

Solution




Le problème

L'attribut lastname de l'entité user est inaccessible du fait que l'utilisateur n'est pas connecté

[Symfony Exception](#)[Symfony Docs](#)

RuntimeExceptionHTTP 500 Internal Server Error

Impossible to access an attribute ("lastname") on a null variable.

ExceptionLogs1Stack Trace

Twig\Error\RuntimeException

in templates/stripe/checkout.html.twig (line 38)

```
33.         <section>
34.             <div class="checkout-form-title">{% trans %}form{% endtrans %}</div>
35.             <strong>{% trans %}identity and contact details{% endtrans %}</strong>
36.
37.             <label for="lastname">{% trans %}last name{% endtrans %}</label>
38.             {{ form_widget(form.lastname, {'attr': {'name': 'lastname', 'id': 'lastname', 'value': app.user.lastname}})
39.             <div class="checkout-form-error">
40.                 {{ form_errors(form.lastname) }}
41.             </div>
42.
43.             <label for="firstname">{% trans %}first name{% endtrans %}</label>
```

in var/cache/dev/twig/62/62901271686923adc8d7e173972d767d.php twig_get_attribute (line 188)

in vendor/twig/twig/src/Template.php -> block_body (line 171)

in var/cache/dev/twig/59/591fe71bd42902a65e8ee6efc73980e8.php -> displayBlock (line 88)

in vendor/twig/twig/src/Template.php -> doDisplay (line 394)

in vendor/twig/twig/src/Template.php -> displayWithErrorHandling (line 367)

Solution

Ajouter une règle au firewall de Symfony pour restreindre l'accès aux utilisateurs non connectés

```
- { path: ^/(%app.locales%)/stripe, roles: ROLE_USER }
```



Conclusion



Améliorations prévues

- Factoriser le code
- Améliorer l'espace utilisateur et l'espace admin
- Améliorer le panier



Merci !