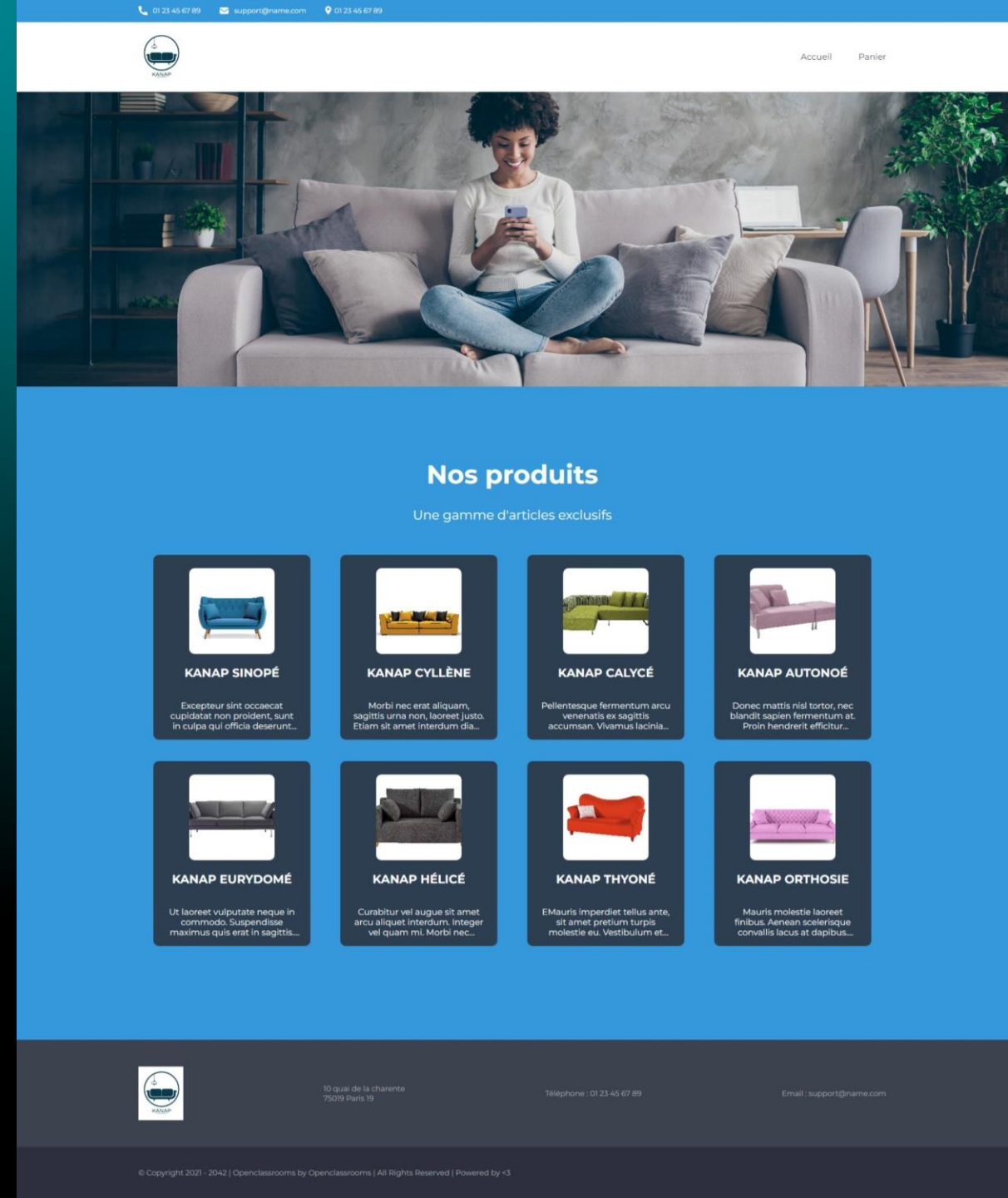


PRESENTATION DU PROJET 5

Kanap




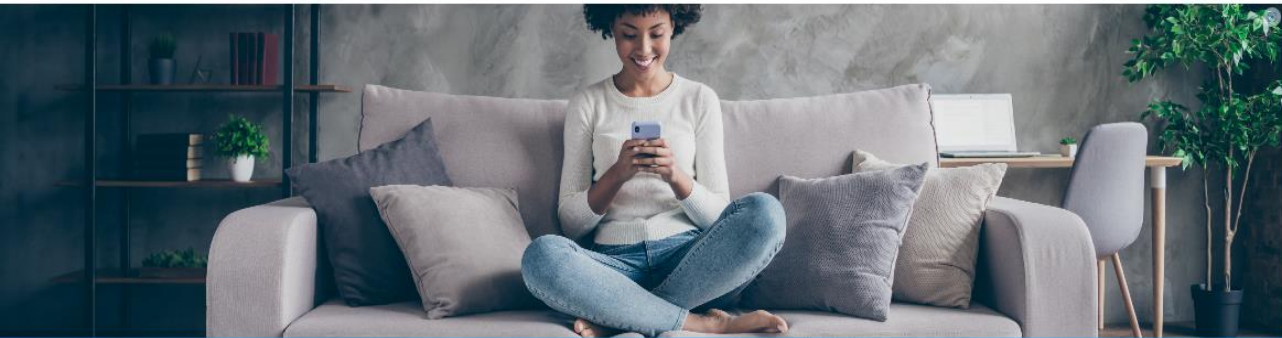
Technologies utilisées

- HTML, CSS, JavaScript.
- Visual Studio Code, Node
- Outils de développement
- Git et GitHub
- API (interface de programmation)

Page d'accueil avec nos produits


09 23 45 67 89support@name.com09 23 45 67 89

AccueilPanier




Nos produits

Une gamme d'articles exclusifs




KANAP SINOPE

Excepleur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt...




KANAP CYLLÈNE

Morbi nec erat aliquam, sagittis urna non, laoreet justo. Etiam sit amet interdum dia...




KANAP CALYCÉ

Pellentesque fermentum arcu venenatis ex sagittis accumsan. Vivamus lacinia...




KANAP AUTONOE

Donec mattis risi tortor, nec blandit sapien fermentum at. Proin hendrerit efficitur...




KANAP EURYDOMÉ

Ut laoreet vulputate neque in commodo. Suspendisse maximus quis erat in sagittis...




KANAP HÉLICÉ

Curabitur vel augue sit amet arcu aliquet interdum. Integer vel quam mi. Morbi nec...



KANAP THYONÉ

Et Mauris imperdiet tellus ante, sit amet pretium turpis molestie eu. Vestibulum et...



KANAP ORTHOSIE

Mauris molestie laoreet finibus. Aenean scelerisque convallis lacus at dapibus...

Application

- Manifeste
- Service Workers
- Stockage

Stockage

- Stockage local
 - http://127.0.0.1:5500
- Stockage de session
- IndexedDB
- Web SQL
- Cookies
- Jetons d'approbation
- Groupes d'intérêt

Cache

- Stockage en cache
- Cache précédent/avant

Services d'arrière-plan

- Extraction en arrière-plan
- Synchronisation en arrière-pl
- Notifications
- Gestionnaire de paiement
- Synchronisation périodique e
- Messagerie Push
- Rapport API

Clé

cart

Valeur

[]

Aucune propriété

Console

Problèmes

Couverture

Ressources pour les développeurs

Console réseau

2 messages

1 user message

No errors

No warnings

1 info

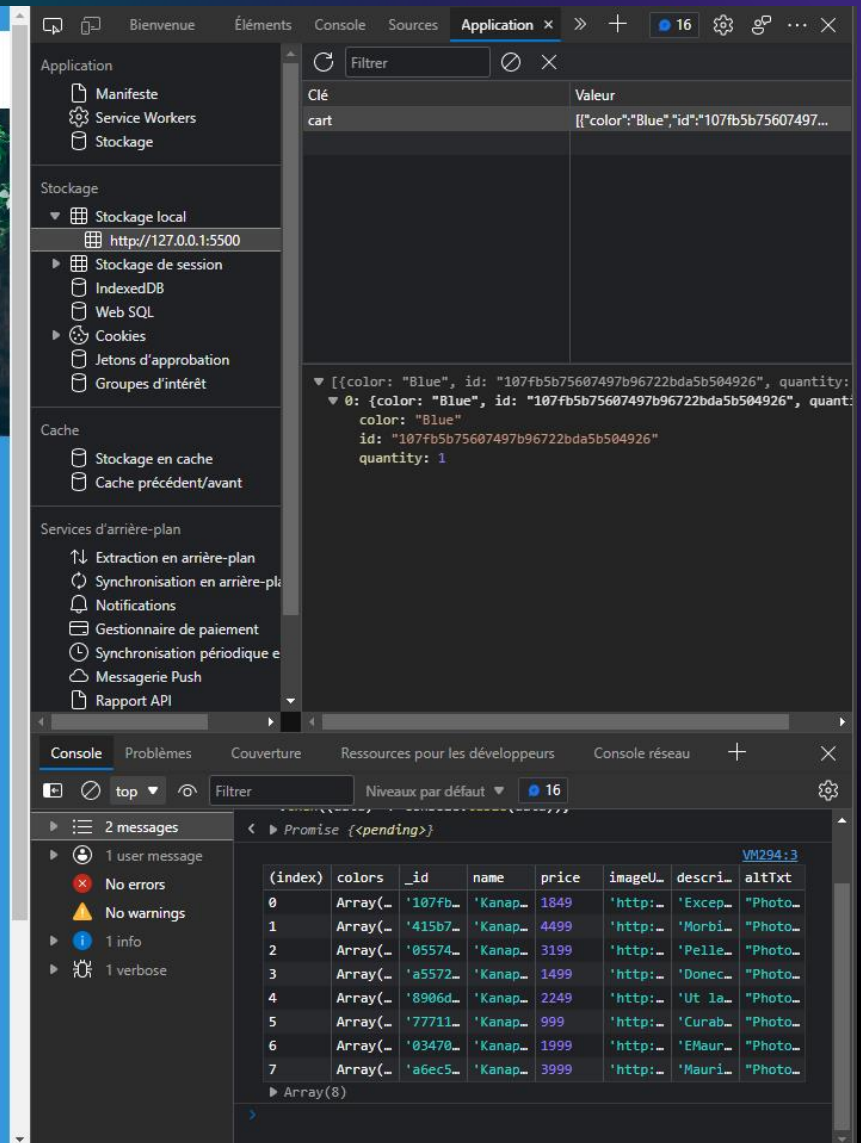
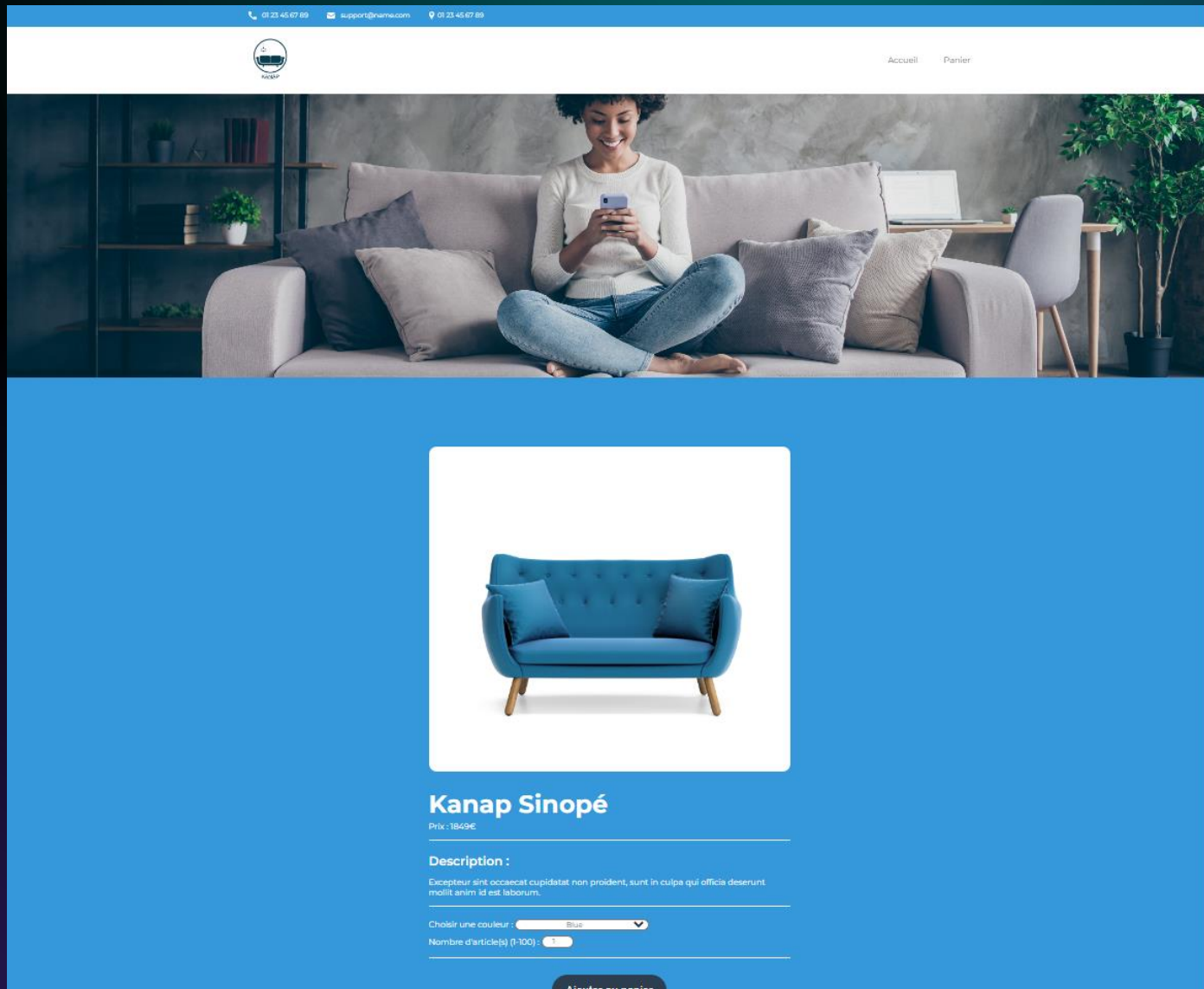
1 verbose

Promise {<pending>}

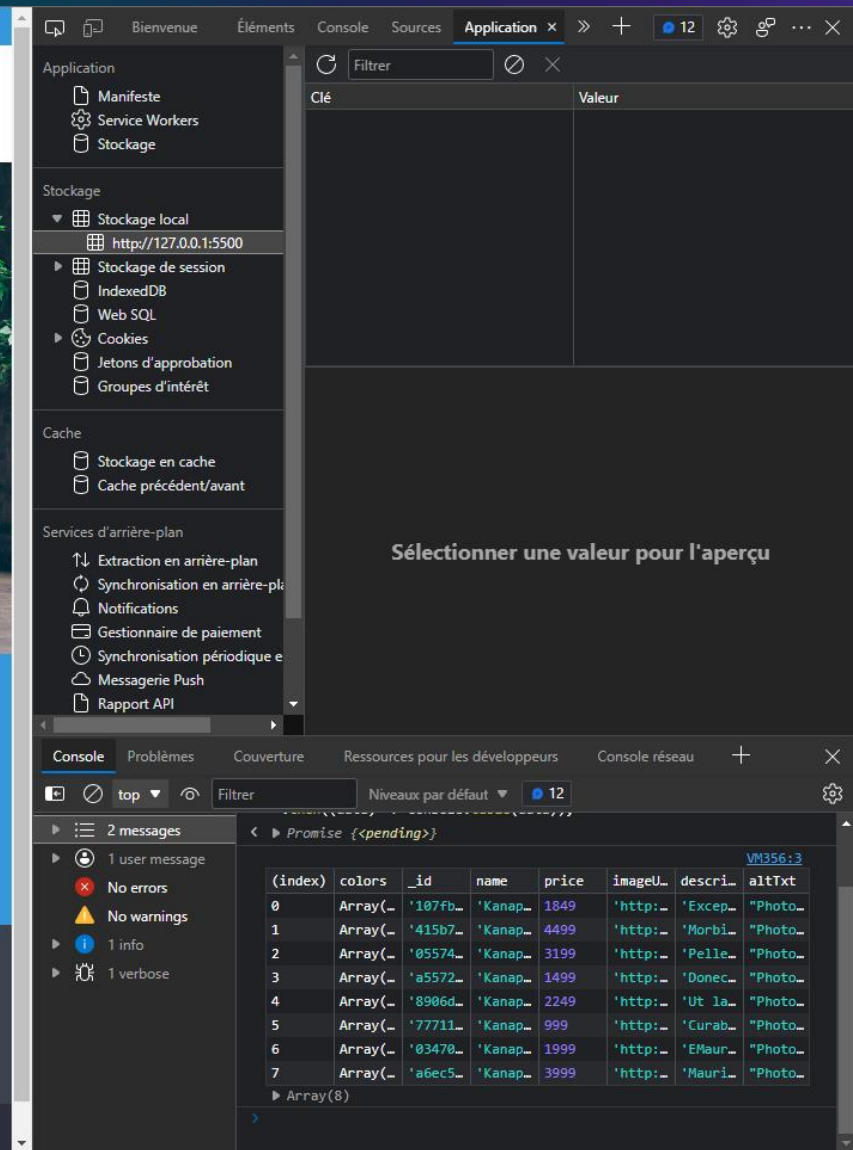
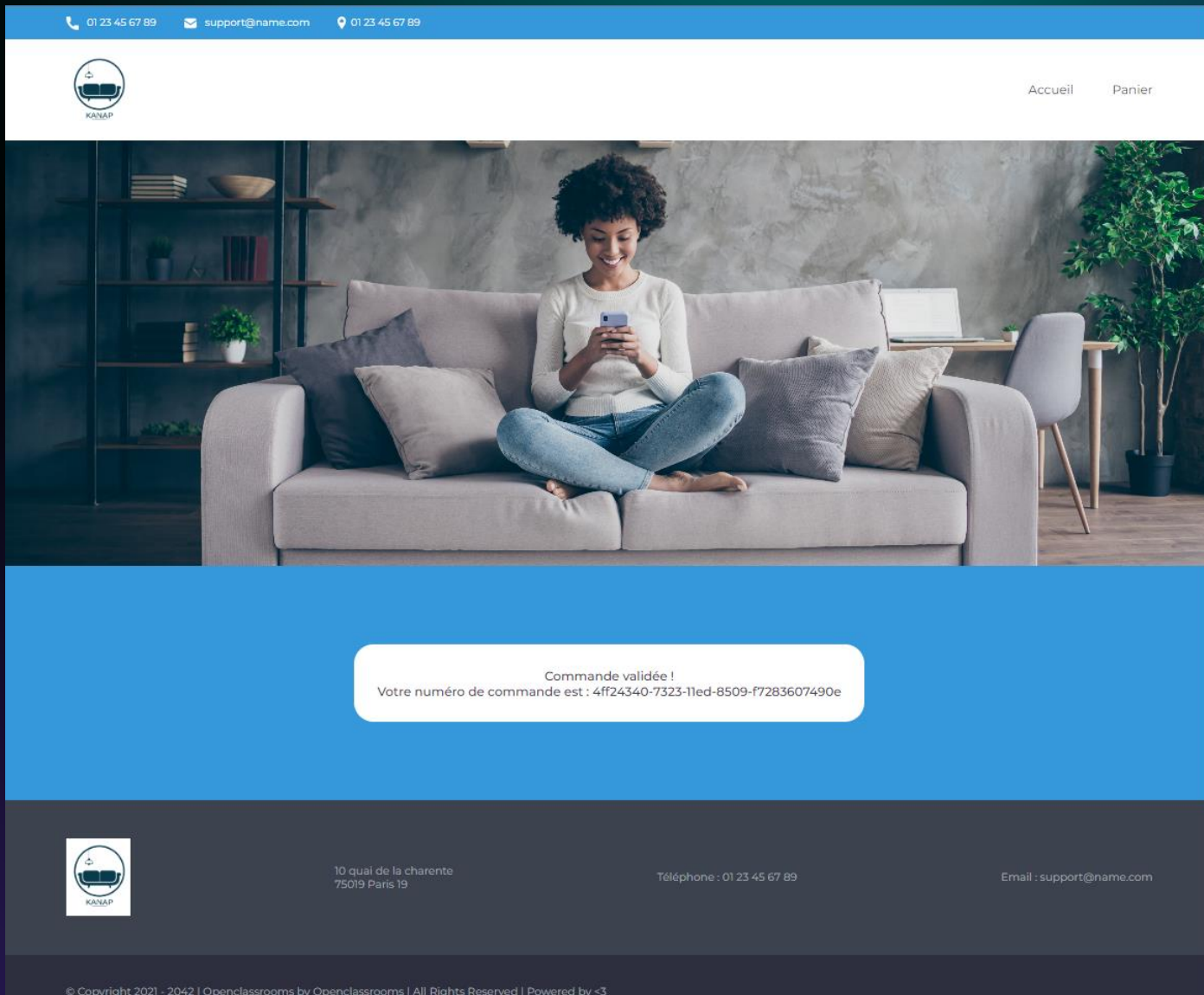
(index)	colors	_id	name	price	imageU	descri	altTxt
0	Array(_	'107fb_	'Kanap_	1849	'http:...	'Excep...	'Photo...
1	Array(_	'415b7_	'Kanap_	4499	'http:...	'Morbi...	'Photo...
2	Array(_	'05574_	'Kanap_	3199	'http:...	'Pelle...	'Photo...
3	Array(_	'a5572_	'Kanap_	1499	'http:...	'Donec...	'Photo...
4	Array(_	'8906d_	'Kanap_	2249	'http:...	'Ut la...	'Photo...
5	Array(_	'77711_	'Kanap_	999	'http:...	'Curab...	'Photo...
6	Array(_	'03470_	'Kanap_	1999	'http:...	'EMaur...	'Photo...
7	Array(_	'a6ec5_	'Kanap_	3999	'http:...	'Mauri...	'Photo...

Array(8)

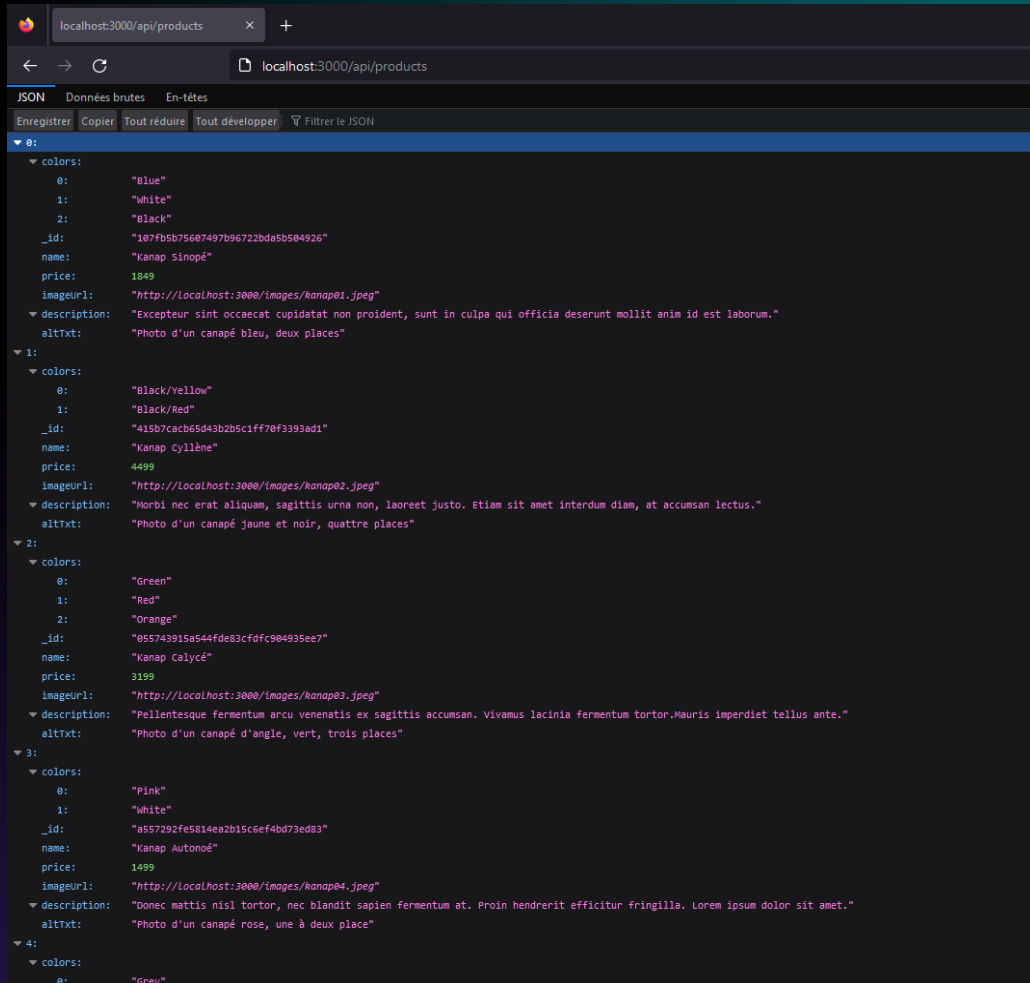
La page panier



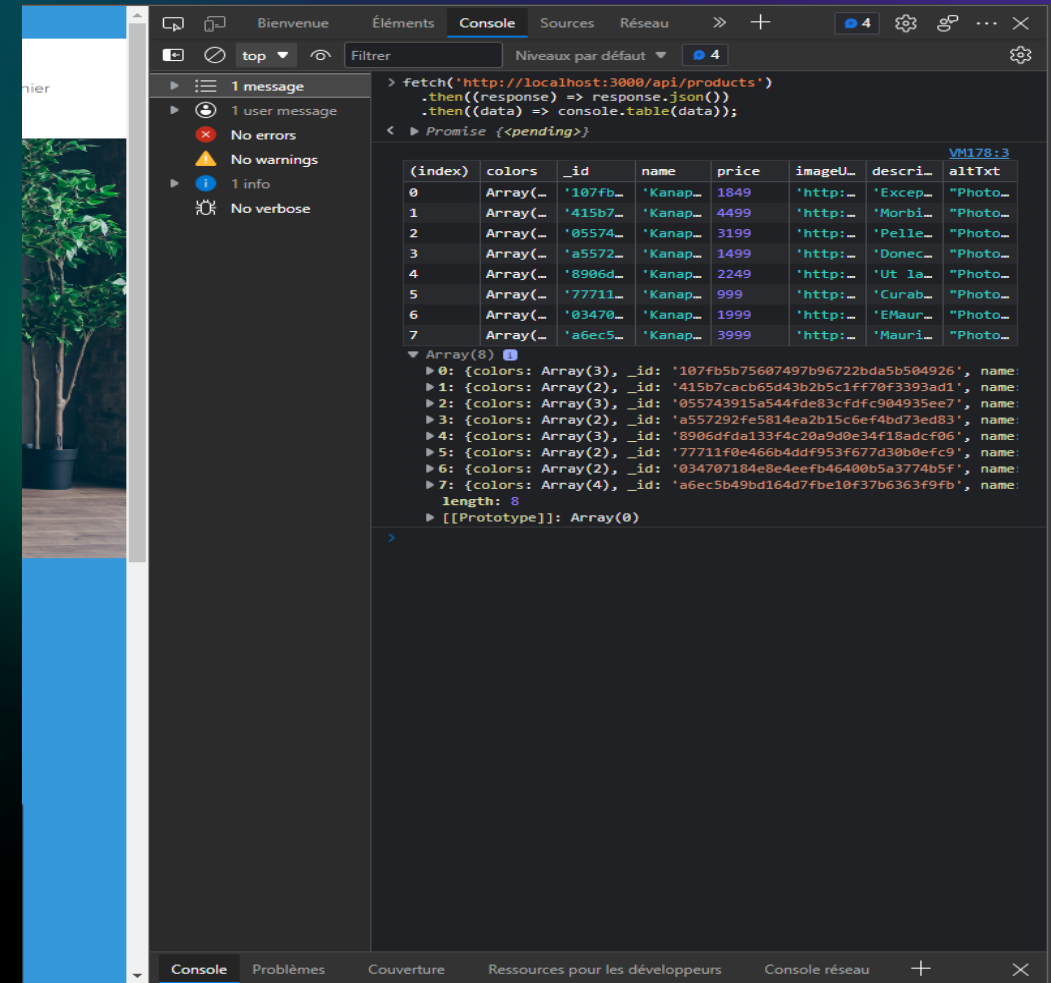
La page confirmation



Récupération de la liste de produit



La liste des produits retourné par l'API en JSON
URL des API
Catalogue de canapés :
<http://localhost:3000/api/products>



Requêter l'API avec la méthode Fetch pour
récupérer l'ensemble des produits puis
insérer chaque élément dans la page
d'accueil

Affichage des produits de façon dynamique dans le stockage local.



Affichage des produits de façon statique dans la console sous forme d'un tableau.



The screenshot shows the VS Code interface with the following components:

- Left Panel:** Contains the 'Application' view with sections for 'Stockage' (Local, Session, IndexedDB, Web SQL, Cookies, etc.) and 'Cache'.
- Right Panel (Top):** Displays the 'Local Storage' data for the URL 'http://127.0.0.1:5500'. It shows a key 'cart' with a value that is a JSON array of product objects.
- Right Panel (Bottom):** Displays the 'Console' output. It shows a message '[Violation] \'setTimeout\' handler took 202ms' and a log of a fetch request to 'http://localhost:3000/api/products'. Below the log, a table of products is displayed.

(index)	colors	_id	name	price	imageUr1	descripti...	altTxt
0	Array(3)	'107fb5b7...	'Kanap Si...	1849	'http://1...	'Excepteu...	"Photo d'...
1	Array(2)	'415b7cac...	'Kanap Cy...	4499	'http://1...	'Morbi ne...	"Photo d'...
2	Array(3)	'05574391...	'Kanap Ca...	3199	'http://1...	'Pellente...	"Photo d'...
3	Array(2)	'a557292f...	'Kanap Au...	1499	'http://1...	'Donec ma...	"Photo d'...
4	Array(3)	'8906dfda...	'Kanap Eu...	2249	'http://1...	'Ut laore...	"Photo d'...
5	Array(2)	'77711f0e...	'Kanap Hé...	999	'http://1...	'Curabitu...	"Photo d'...
6	Array(2)	'03470718...	'Kanap Th...	1999	'http://1...	'EMauris ...	"Photo d'...
7	Array(4)	'a6ec5b49...	'Kanap or...	3999	'http://1...	'Mauris m...	"Photo d'...

Quelques exemples de l'utilisation de l'Api

Requêter l'API avec la methode Get pour obtenir l'ensemble des produits ; récupérer la réponse émise, et parcourir celle-ci pour insérer chaque élément (chaque produit) dans la page d'accueil (dans le DOM).

```
function createProductsCards() {  
  fetch("http://localhost:3000/api/products")  
    .then((response) => {  
      return response.json();  
    })  
    .then((productList) => {  
      productList.forEach((product) => {  
        const productLink = createTag("a");  
        const articleTag = createTag("article");  
        const productImage = createTag("img");  
        const productName = createTag("h3");
```


Requête POST avec fetch

```
// mergeInputs() est exécutée et fusionne les valeurs entrées par l'utilisateur  
mergeInputs();  
fetch("http://localhost:3000/api/products/order", {  
  method: "POST",  
  body: JSON.stringify(orderProducts),  
  headers: {  
    "Content-Type": "application/json",  
  },  
})  
  .then((response) => {  
    return response.json();  
  })  
  .then((response) => {  
    localStorage.clear();  
    window.location.href = `confirmation.html?order=${response.orderId}`;  
  })  
  .catch((error) => {
```

Le DOM

(ajout, suppression et modification d'éléments)

```
💡 "createElement" cree un nouvel element HTML
const createInnerContent = function () {
  cartContent += `<article class="cart__item" data-id="${cart[i].id}" data-color="${productColor}">
    <div class="cart__item__img"></div>
    <div class="cart__item__content">
      <div class="cart__item__content__description">
        <h2>${productName}</h2>
        <p>${productColor}</p>
        <p>${productPrice} €</p>
      </div>
      <div class="cart__item__content__settings">
        <div class="cart__item__content__settings__quantity">
          <p>Qté : </p>
          <input type="number" class="itemQuantity" name="itemQuantity" min="1" max="100" value="${cart[i].quantity}">
        </div>
        <div class="cart__item__content__settings__delete">
          <p class="deleteItem">Supprimer</p>
        </div>
      </div>
    </div>
  </article>`;
};
```

URLSearchParams

manipuler les paramètres de
recherche dans une URL

```
// URLSearchParams manipuler les paramètres de recherche dans une URL
```

```
// la propriété textContent de l'objet orderIdSpan pour remplacer son contenu actuel par l'ID de commande stocké dans la variable  
// ce code affiche l'ID de commande de la page courante dans un élément HTML avec l'ID orderId  
// création de l'objet response
```

```
const currentPageUrl = document.location.href;  
const url = new URL(currentPageUrl);  
const orderId = url.searchParams.get("order");  
const orderIdSpan = document.getElementById("orderId");  
const response = new URL("http://127.0.0.1:5500/front/html/confirmation.html");  
console.log(response);  
orderIdSpan.textContent = `${orderId}`;
```

```
fetch("http://localhost:3000/api/products")  
  .then((response) => response.json())  
  .then((data) => console.table(data));
```


L'utilisation du local storage

```
const pushLocalStorageQuantity = function () {  
  getParentArticle = this.closest("article");  
  getProductToUpdate();  
  if (updatedProduct) {  
    const indexOfUpdatedProduct = cart.indexOf(updatedProduct);  
    updatedProduct.quantity = parseInt(  
      quantityInputField[indexOfUpdatedProduct].value  
    );  
    localStorage.setItem("cart", JSON.stringify(cart));  
    eventListener();  
    getCartTotal();  
    convertCartToArray();  
  }  
};  
  
// localStorage  
// "setItem" ajoute un element au localStorage  
// "JSON.stringify" convertit une valeur JavaScript en chaine JSON  
// "JSON.parse" convertit une chaine JSON en objet JavaScript  
// "getItem" retourne la valeur d'un element du localStorage  
💡 "removeItem" supprime un element du localStorage  
// "clear" supprime tous les elements du localStorage
```

Fin de la presentation