Sebald Pauer

PLAN DE TEST
Orinoco

Objectifs du plan de test:

Notre plan de test permet de **déceler les anomalies Javascript** et de **vérifier le bon fonctionnement** des applications «Orinoco».
Les fonctions globales sont celles à analyser,
car exploitées par les autres.

Sommaire:

Vous trouverez ces fichiers dans le dossier:
"./public/js/function"

```
async function getUrl(url){
    const settingsGet= {
    method: 'GET',
    headers: {'Content-Type':'application/json'}
    }
    return fetch(url, settingsGet)
    .then(response => response.json())
    //.then(data => console.log(data))
    //.catch(error => console.log('Une erreur est survenue:' + error));
}
```

A

Description de la fonction:
getUrl(url)

Cette fonction a pour but d'effectuer une requête HTTP en méthode GET.

Paramètre de la fonction:

- **url**: adresse de l'API pour avoir accès à sa base de données.

TEST N°1: **Vérifier la requête fetch GET**

Vérifier que la requête fetch obtient une réponse positive de l'API et que si le serveur est en maintenance, l'erreur soit affiché dans la console.

Méthode du test:

- **Promesse then:** .then(data => console.log(data))
- **Promesse catch:** .catch(error => console.log('Une erreur est survenue: ' + error))

Succès du test:

- si le contenu de l'API s'affiche dans la console au format JSON;
- si la console affiche une erreur 400, 401, 500, etc. De plus, l'erreur sera précisement décrite dans un objet.

```
function displayElements(items, ParentBoxId, productName, productItemHtml) {
    items.forEach(item => {
        let itemBox = document.createElement("div");
        document.getElementById(ParentBoxId).appendChild(itemBox);
        itemBox.classList.add(
            "col-12",
            "col-lg-4",
            "d-flex",
            "align-items-stretch"
        );
        let itemCardBox = document.createElement("div");
        itemBox.appendChild(itemCardBox);
        itemCardBox.classList.add(
            "card",
            "mb-4"
        );
        let itemCardImage = document.createElement("img");
        itemCardBox.appendChild(itemCardImage);
        itemCardImage.classList.add(
            "card-img-top",
            "h-75",
            "rounded"
        );
        itemCardImage.setAttribute("alt", productName + " " + item.name);
        itemCardImage.setAttribute("src", item.imageUrl);
        let itemCardBodyBox = document.createElement("div");
        itemCardBox.appendChild(itemCardBodyBox);
        itemCardBodyBox.classList.add(
            "card-body"
        );
        let itemButtonBox = document.createElement("div");
        itemCardBodyBox.appendChild(itemButtonBox);
        itemButtonBox.classList.add(
            "text-center",
            "mb-4"
        );
        let itemButton = document.createElement("a");
        itemButtonBox.appendChild(itemButton);
        itemButton.classList.add(
            "btn",
            "btn-light",
            "stretched-link"
        );
        itemButton.setAttribute("role", "button");
        itemButton.setAttribute("href", productItemHtml + "?" + item._id);
        itemButton. innerHTML = "Voir <em>" + item.name + "</em>";
    });
}
```

B

<u>Description de la fonction:</u>
displayElements(items, ParentBoxId, productName, productItemHtml)

Cette fonction a pour but de créer une boite (image du produit et lien vers la page du produit) pour chaque éléments présents dans les base de données présentes sur l'API.

Paramètre de la fonction:

- **items**: éléments de l'API au format JSON.
- **ParenBoxId**: identifiant id du block parent dans le fichier HTML lié.
- **productName**: «type» du produit utilisé dans la description de l'image.
- **productItemHtml**: chemin pour aller la page HTML descriptive du produit cliqué.

TEST N°2: **Vérifier l'ajout correcte des card de produits via .forEach()**

Vérifier que la fonction items.forEach() affiche correctement l'ensemble des éléments présents dans l'API.

Méthode du test:

- Créer une liste non ordonnée pour chaque nom des produits présents dans l'API:

```
const NameList = document.createElement('ul');

items.forEach( item => {
  let NameItem = document.createElement('li');
  NameList.appendChild(NameItem);
  NameItem.innerHTML = item.name;
});
```

Succès du test:

- si tous les noms s'affichent au format de liste non ordonnée.

```
function cartIconNotification_Ori(App) {
    const CartNotificationItems = JSON.parse(localStorage.getItem(`Ori${App}Cart`));
    if (localStorage.getItem(`Ori${App}Cart`)){
        const NavLinkCart = document.getElementById("CartIcon");
        NavLinkCart.innerHTML = `<i class="fas fa-shopping-cart mr-1"></i><span id="CartNotification" aria-describedby="Nombre d'éléments différents dans le panier">${CartNotificationItems.length}</span>Panier`;
    }
}
```

C

Description de la fonction:
cartIconNotification_Ori(App)

Cette fonction a pour but d'afficher une pastille numérotée (elle donne le nombre de produits différents dans le panier) lié à l'icone du panier, dans la barre de navigation.

Paramètre de la fonction:

- **App**: terme prédéfini: «teddy», «furniture» ou «camera». Il permet d'analyser le localStorage spécifique à l'application en utilisation.

TEST N°3: **Vérifier l'ajout correcte des valeurs de OriteddyCart du localStorage**

Vérifier que la fonction cartIconNotification_Ori(teddy) affiche correctement l'ensemble du nom d'éléments - présents dans le panier - dans une pastille collée à l'icon dans la barre de navigation.

Méthode du test:

- Ajout d'un élément teddy dans le panier et rafraichir la page. Si la pastille ne s'affiche pas, c'est que le localStorage OriteddyCart n'existe pas.

Succès du test:

- si la pastille apparait.

```
function displayProduct(item, ParentBoxId, OriApp){
    let _optionType = "";
    let _productType = "";
    let _productOptionLabelName = "";
    switch (OriApp) {
        case 'Oriteddy':
            _optionType += "colors";
            _productType += "Ours en peluche";
            _productOptionLabelName += "Couleur";
            break;
        case 'Oricamera':
            _optionType += "lenses";
            _productType += "Caméra";
            _productOptionLabelName += "Lentille"
            break;
        case 'Orifurniture':
            _optionType += "varnish";
            _productType += "Meuble en chêne";
            _productOptionLabelName += "Vernis";
            break;
        default:
            _optionType += "Non défini";
            _productType += "Non défini";
            _productOptionLabelName += "Non défini";
    }
    let ProductBox = document.createElement("div");
    document.getElementById(ParentBoxId).appendChild(ProductBox);
    let ProductImage = document.createElement("img");
    ProductBox.appendChild(ProductImage);
    ProductImage.classList.add(
        "card-img-top",
        "mt-4"
    );
    ProductImage.setAttribute("alt", _productType + item.name);
    ProductImage.setAttribute("src", item.imageUrl);
    let ProductBoxBody = document.createElement("div");
    ProductBox.appendChild(ProductBoxBody);
    ProductBoxBody.classList.add(
        "card-body",
        "m-md-2"
    );
    let ProductBoxTextBlock = document.createElement("div");
    ProductBoxBody.appendChild(ProductBoxTextBlock);
    ProductBoxTextBlock.classList.add("text-center");
    let ProductName = document.createElement("h1");
    ProductBoxTextBlock.appendChild(ProductName);
    ProductName.classList.add(
        "card-title",
        "h3",
        "font-weight-bold");
    ProductName.innerHTML = item.name;
    let ProductDescription = document.createElement("p");
    ProductBoxTextBlock.appendChild(ProductDescription);
    ProductDescription.classList.add(
        "h5",
        "text-justify");
    ProductDescription.innerHTML = item.description;
    let ProductPrice = document.createElement("p");
    ProductBoxTextBlock.appendChild(ProductPrice);
    ProductPrice.classList.add(
```

```
        "h3",
        "m-5");
    ProductPrice.innerHTML = item.price/100 + "€";
    let ProductOptionBox = document.createElement("div");
    ProductBoxTextBlock.appendChild(ProductOptionBox);
    ProductOptionBox.classList.add("input-group");
    let ProductOptionLabelBox = document.createElement("div");
    ProductOptionBox.appendChild(ProductOptionLabelBox);
    ProductOptionLabelBox.classList.add("input-group-prepend");
    let ProductOptionLabel = document.createElement("label");
    ProductOptionLabelBox.appendChild(ProductOptionLabel);
    ProductOptionLabel.classList.add("input-group-text");
    ProductOptionLabel.setAttribute("for", `${_optionType}Select`);
    ProductOptionLabel.innerHTML = _productOptionLabelName;
    let ProductOptionSelect = document.createElement("select");
    ProductOptionBox.appendChild(ProductOptionSelect);
    ProductOptionSelect.classList.add("custom-select");
    ProductOptionSelect.setAttribute("id", `${_optionType}Select`);
    let ProductOptionSelectValueDefault = document.createElement("option");
    ProductOptionSelect.appendChild(ProductOptionSelectValueDefault);
    ProductOptionSelectValueDefault.innerHTML = "Sans option";

    /* Boucle "for" pour afficher les différentes personnalisations disponibles pour le produit en question */

    for (let i = 0; i < item[_optionType].length; i++) {
        document.getElementById(`${_optionType}Select`).innerHTML += `<option value="${item[_optionType][i]}">${item[_optionType][i]}</option>`;
    }
};

function itemCartAddItemButton_Ori(App, data){
    let typeOption = '';
    let alertText = '';
    switch (App) {
    case 'teddy':
        typeOption += "colorsSelect";
        alertText += "L'ours en peluche";
        break;
    case 'furniture':
        typeOption += "varnishSelect";
        alertText += "Le meuble en chêne";
        break;
    case 'camera':
        typeOption += "lensesSelect";
        alertText += "L'appareil argentique";
        break;
    default:
        console.log(`Désolé, l'application Ori${App} n'existe pas.`);
    }
    /* Fonction - appelée au clic - sur le bouton "Ajouter au panier */
    document
    .getElementById("add-btn")
    .addEventListener("click", function addcart() {
        /* Variable utilisée pour étudier si le produit est déja dans le panier et gérer les différentes conditions */
        let OriAppNewItem = true;
        /* Array pour stocker les produits mis au panier */
        let OriAppClientCart = [];
        /* Variable pour la quantité sélectionnée */
```

```
        let quantity = 1;
        /* Variable pour la personnalisation sélectionnée */
        let option = document.getElementById(typeOption).value;
        /* Objet qui récupère les infos du produit et sa quantité */
        let itemSelected = {
            id: data._id,
            name: data.name,
            image: data.imageUrl,
            price: data.price,
            option,
            quantity,
        };
        /* Condition: si panier vide, alors
        on ajoute un à la quantité,
        on ajoute l'objet sélectionné au tableau et
        on stocke le tableau en local*/
        if (localStorage.getItem(`Ori${App}Cart`) === null) {
            OriAppClientCart.push(itemSelected);
            localStorage.setItem(`Ori${App}Cart`, JSON.stringify(OriAppClientCart));
            OriAppNewItem = false;
            /* Sinon si produit déja au panier, après vérification, on ajoute 1 à la
quantité du produit en question */
        } else {
            OriAppClientCart = JSON.parse(localStorage.getItem(`Ori${App}Cart`));
            OriAppClientCart.forEach((product) => {
                if ((product.id === itemSelected.id)&&(product.option ===
itemSelected.option)) {
                    product.quantity += 1;
                    OriAppNewItem = false; // pour que la dernière condition ne se
déclenche pas
                }
                localStorage.setItem(`Ori${App}Cart`,
JSON.stringify(OriAppClientCart));
            });
        }
        /* si produit non présent dans le panier alors on ajoute l'objet sélectionné
au tableau et on stocke le tableau en local */
        if (OriAppNewItem == true) {
            OriAppClientCart.push(itemSelected);
            localStorage.setItem(`Ori${App}Cart`, JSON.stringify(OriAppClientCart));
        }
        itemCartdisplayAlert(alertText);
    });
}

/*Message d'alerte indiquant la mise au panier du produit*/
function itemCartdisplayAlert(text) {
    let alertBlock = document.createElement("div");
    document.getElementById("item-added").appendChild(alertBlock);
    alertBlock.classList.add(
        "alert",
        "alert-success",
        "alert-dismissible",
        "fade",
        "show",
        "shadow"
    );
    alertBlock.setAttribute("role", "alert");
    let alertTitle = document.createElement("h5");
    alertBlock.appendChild(alertTitle);
```

```
    alertTitle.classList.add("alert-heading");
    alertTitle.innerHTML = "Nous vous remercions!";
    let alertCloseButton = document.createElement("button");
    alertBlock.appendChild(alertCloseButton);
    alertCloseButton.classList.add("close");
    alertCloseButton.setAttribute("type", "button");
    alertCloseButton.setAttribute("data-dismiss", "alert");
    alertCloseButton.setAttribute("aria-label", "Fermer la fenêtre");
    let closeIcon = document.createElement("i");
    alertCloseButton.appendChild(closeIcon);
    closeIcon.setAttribute("aria-hidden", "true");
    closeIcon.classList.add("fas", "fa-times");
    let alertCloseMessage = document.createElement("p");
    alertBlock.appendChild(alertCloseMessage);
    alertCloseMessage.innerHTML = `${text} a été ajouté à votre panier.`;
}
```

D

Description de la fonction:
displayProduct(item, ParentBoxId, OriApp)

Cette fonction a pour but d'afficher toutes les informations présentes dans l'API spécifique au produit: image, nom, description, prix, personnalisation.

Paramètre de la fonction:

- **items**: informations du produit présentes sur l'API au format JSON.
- **ParenBoxId**: identifiant id du block parent dans le fichier HTML lié.
- **OriApp**: terme prédéfini: «Oriteddy», «Orifurniture» ou «Oricamera». Il permet de déterminer trois variables importantes pour l'affichage: «_optionType» (pour le nom de la clé comprenant les personnalisations possibles), «_productType» (pour rédiger la description de l'image) et « _productOptionLabelName» (pour donner le nom du label). Donc si d'autres applications sont en développement, il est nécessaire d'actualiser les conditions switch.

Description de la fonction:
itemCartAddItemButton_Ori(App, data)

Cette fonction a pour but d'ajouter un objet au localStorage dès que le bouton d'ajout est cliqué avec son identifiant, son nom, l'adresse de son image, son prix, son option et sa quantité.

Paramètre de la fonction:

- **App**: terme prédéfini: «teddy», «furniture» ou «camera». Il permet de créer un objet dans le localStorage spécifique à l'application en utilisation et à l'objet ajouté. Il permet de déterminer trois variables importantes pour l'affichage: «typeOption» (pour donner l'identifiant de l'input à analyser) et « alertText» (pour terminer le texte du panneau alerte). Donc si d'autres applications sont en développement, il est nécessaire d'actualiser les conditions switch.
- **data**: informations du produit présentes sur l'API au format JSON.

Description de la fonction:
itemCartdisplayAlert(text)

Cette fonction a pour but d'ajouter un panneau alerte. La fonction est executée à la fin de la deuxième fonction (donc quand on clique sur ajout, cela affiche un panneau d'alerte.

Paramètre de la fonction:

- **text**: terme utilisé dans la phrase d'alerte, donnant le type de produit ajouté dans le panier de l'application.

TEST N°4: **Vérifier l'ajout correct des informations présentes dans l'API**

Vérifier que la API utilisée correspond à l'élement entré dans la fonction et que son profil correspond aux paramètres de la condition switch.

Méthode du test:
- Faire un console.log(_optionType, _productType, _productOptionLabelName) afin d'afficher si l'API utilisée existe bien dans les paramètres connus. (ex: Oriteddy)
- Les paramètres «item» et «OriApp» n'ont rien à voir (ex: Oriteddy et api/furniture/...)

Succès du test:
- si «colors», «Ours en peluche» et «Couleur» s'affichent
- si l'input Option n'aura qu'un seul paramètre et la description de l'image ne correspondra pas.

TEST N°5: **Vérifier le bon envoi du panier au localStorage**

Vérifier que la fonction « ajoutPanier() » envoie un tableau contenant les produits au LocalStorage lorsque l'utilisateur clic du bouton «Ajouter au panier».

Méthode du test:
- Ajouter au panier un ours en peluche et vérifier le localStorage du navigateur

Succès du test:
- OriteddyCart, dans le localStorage, contient les informations de l'ours en peluche ajouté.

TEST N°6: **Vérifier le message de réussite d'ajout**

Alerter le client que l'article à bien été au panier lors du clic du bouton « Ajouter au panier ».

Méthode du test:
- alert(« Article ajouté au panier! »); à la fin de la deuxième fonction

Succès du test:
- le message d'alerte s'affiche correctement après l'ajout.

```
function cart_Ori(App){
    /* Contenu du localStorage pour afficher le panier */
    const OriAppLocalStorageContent =
JSON.parse(localStorage.getItem(`Ori${App}Cart`));
    /* Afficher #EmptyCart quand le localStorage est vide et afficher #CartContent et
#Form quand il y a des éléments dans le localStorage*/
    if (localStorage.getItem(`Ori${App}Cart`) === null ||
OriAppLocalStorageContent.length == 0) {
        document.getElementById(`Ori${App}EmptyCart`).style.display = "block";
        document.getElementById(`Ori${App}CartContent`).style.display = "none";
        document.getElementById(`Ori${App}Form`).style.display = "none";
    } else {
        document.getElementById(`Ori${App}CartContent`).style.display = "block";
        document.getElementById(`Ori${App}Form`).style.display = "block";
        document.getElementById(`Ori${App}EmptyCart`).style.display = "none";
    }
    const tableCartContainer = document.getElementById(`Ori${App}CartItems`);
    let calcTotal = 0;
    /* Boucle qui itère chaque contenu du panier dans un tableau */
    /* Ne s'enclenche que si élément(s) présent(s) dans le panier */
    if (localStorage.getItem(`Ori${App}Cart`)) {
        for (let i = 0; i < OriAppLocalStorageContent.length; i++) {
            const tableCartItem = document.createElement("tr");
            tableCartContainer.appendChild(tableCartItem);
            /* Affichage de l'image du produit qui est un lien pour retourner à sa
page produit */
            const tdImage = document.createElement("td");
            tdImage.classList.add("d-none", "d-sm-table-cell");
            tableCartItem.appendChild(tdImage);
            const tdImagelink = document.createElement("a");
            const tdImageimg = document.createElement("img");
            tdImage.appendChild(tdImagelink);
            tdImagelink.appendChild(tdImageimg);
            tdImagelink.setAttribute("href", "teddy_item.html?" +
OriAppLocalStorageContent[i].id);
            tdImageimg.setAttribute("src", OriAppLocalStorageContent[i].image);
            tdImageimg.style.width = "8rem";
            /* Affichage du nom du produit */
            const tdName = document.createElement("td");
            tableCartItem.appendChild(tdName);
            tdName.innerHTML = OriAppLocalStorageContent[i].name;
            /* Affichage de la personnification du produit (par défaut) */
            const tdOption = document.createElement("td");
            tdOption.classList.add("d-none", "d-sm-table-cell");
            tableCartItem.appendChild(tdOption);
            tdOption.innerHTML = OriAppLocalStorageContent[i].option;
            /* Affichage du prix du produit multiplié par sa quantité */
            const tdPrice = document.createElement("td");
            tableCartItem.appendChild(tdPrice);
            tdPrice.innerHTML = OriAppLocalStorageContent[i].price / 100 + "€";
            /* Affichage de la quantité du produit */
            const tdQty = document.createElement("td");
            tableCartItem.appendChild(tdQty);
            tdQty.classList.add("text-center");
            tdQty.innerHTML = OriAppLocalStorageContent[i].quantity;
            if (tdQty.innerHTML == 1){
                tdQty.innerHTML += `<button id="plusButton${i+1}" class="btn btn-
outline-dark border-0 btn-sm ml-2"><i class="fas fa-plus"></i></button>`;
            } else {
```

```
                tdQty.innerHTML += `<button id="plusButton${i+1}" class="btn btn-outline-dark border-0 btn-sm ml-2"><i class="fas fa-plus"></i></button>`;
                tdQty.innerHTML += `<button id="minusButton${i+1}" class="btn btn-outline-dark border-0 btn-sm"><i class="fas fa-minus"></i></button>`;
            };
            /* Icone pour suppression du produit */
            const tdRemove = document.createElement("td");
            tableCartItem.appendChild(tdRemove);
            tdRemove.classList.add("text-center");
            const tdRemoveIconLink = document.createElement("a");
            tdRemove.appendChild(tdRemoveIconLink);
            const tdRemoveIcon = document.createElement("i");
            tdRemoveIconLink.appendChild(tdRemoveIcon);
            tdRemoveIconLink.classList.add("btn", "btn-outline-dark");
            tdRemoveIconLink.setAttribute("href", `${App}_cart.html`);
            tdRemoveIcon.classList.add("far", "fa-trash-alt");
            tdRemoveIcon.setAttribute("title", "Supprimer");
            tdRemoveIconLink.addEventListener("click", function () {
                if (OriAppLocalStorageContent.length==1){
                    localStorage.removeItem(`Ori${App}Cart`);
                } else {
                    OriAppLocalStorageContent.splice(i, 1);
                    localStorage.setItem(`Ori${App}Cart`, JSON.stringify(OriAppLocalStorageContent));
                }
                window.location.reload();
            });
            /* Affichage du prix total du panier */
            calcTotal += OriAppLocalStorageContent[i].price * OriAppLocalStorageContent[i].quantity
        }
    }
    for (let i = 0; i < OriAppLocalStorageContent.length; i++) {
        document.getElementById(`plusButton${i+1}`).addEventListener("click", function(){
            OriAppLocalStorageContent[i].quantity +=1;
            localStorage.setItem(`Ori${App}Cart`, JSON.stringify(OriAppLocalStorageContent));
            window.location.reload();
        });
        if ((OriAppLocalStorageContent[i].quantity)> 1) {
            document.getElementById(`minusButton${i+1}`).addEventListener("click", function(){
                OriAppLocalStorageContent[i].quantity -=1;
                localStorage.setItem(`Ori${App}Cart`, JSON.stringify(OriAppLocalStorageContent));
                window.location.reload();
            });
        }
    }
    /* Afficher le prix total du panier */
    const OriAppTotalPrice = document.getElementById(`Ori${App}TotalPrice`);
    OriAppTotalPrice.innerHTML = calcTotal/100 + "€";
    /* Bouton de suppression de tout le contenu du panier */
    const ClearCartButton = document.getElementById(`Ori${App}ClearCart`);
    ClearCartButton.addEventListener("click", function clearStorage() {
        localStorage.removeItem(`Ori${App}Cart`);
        window.location.reload();
    });
```

```
    /* au clique sur le bouton de commande, si tous les champs sont correctement complétés alors
    création d'un objet contact avec les infos fournis et redirection vers la page de confirmation
    sinon alert avec message invitant à remplir le formulaire */
    const SubmitOrderButton = document.getElementById("SubmitOrder");
    SubmitOrderButton.addEventListener("click", function(event) {
        event.preventDefault();
        if (
            document.getElementById("inputClientLastName").validity.valid &&
            document.getElementById("inputClientFirstName").validity.valid &&
            document.getElementById("inputClientEmail").validity.valid &&
            document.getElementById("inputClientAddress").validity.valid &&
            document.getElementById("inputClientCity").validity.valid) {
                const currentHost = window.location.origin;
                if( confirm("ATTENTION: Etant donné l'état actuel de notre application, les OPTIONS choisies NE seront PAS prises en compte à votre commande. Tout article dans votre panier sera 'Sans Option'. Veuillez nous en excusez!")){
                    const contact = {
                        firstName: document.getElementById("inputClientFirstName").value,
                        lastName: document.getElementById("inputClientLastName").value,
                        address: document.getElementById("inputClientAddress").value,
                        city: document.getElementById("inputClientCity").value,
                        email: document.getElementById("inputClientEmail").value
                    };
                    localStorage.setItem(`Ori${App}ClientContact`, JSON.stringify(contact));
                    window.location.href = currentHost + `/public/page/${App}/${App}_confirmation.html`;
                } else {
                    localStorage.removeItem(`Ori${App}Cart`);
                    window.location.href = currentHost + '/index.html';
                }
        } else {
            alert("Champs requis non complétés!");
        }
    });
}
```

E

Description de la fonction:
cart_Ori(App)

Cette fonction a pour but d'afficher une pastille numérotée (elle donne le nombre de produits différents dans le panier) lié à l'icone du panier, dans la barre de navigation.

Paramètre de la fonction:

- **App: terme prédéfini: «teddy», «furniture» ou «camera». Il permet d'analyser le localStorage spécifique à l'application en utilisation , d'afin les différents éléments en lien et de créer un localStorage contenant les coordonnées du commanditaire.**

```
function confirmation_Ori(App, PostAdress){
    let calcTotal = 0; //variable pour stocker le coût total de l'achat (X100).
    /* Stockage des id des produits présents dans le panier dans un array nommé 'products' */
    const cartContent = JSON.parse(localStorage.getItem(`Ori${App}Cart`));
    const products = [];

    cartContent.forEach((product) => {
        for (let i = 0; i < product.quantity; i++) {
        products.push(product.id);
        }
    });

    /* récupération des infos de contact dans le stockage local */
    const contact = JSON.parse(localStorage.getItem(`Ori${App}ClientContact`));

    /* création d'une variable qui regroupe les infos de contact et les id des produits commandés */
    const order = {contact, products};

    /* appel de la fonction post qui envoie les infos à l'API et récupère les infos de la commande */
    const promisePost = fetch(PostAdress, {method: 'POST', headers: {'Content-Type':'application/json'}, body: JSON.stringify(order)});
    promisePost.then(async (response) => {
        const OrderContent = await response.json();
        document.getElementById("order-id").innerHTML = "Commande n° " + await OrderContent.orderId;
        document.getElementById("name").innerHTML = contact.lastName + " " + contact.firstName;
        document.getElementById("address").innerHTML = contact.address + " à " + contact.city;
        document.getElementById("email").innerHTML = contact.email;
        /* calcul du coût total de la commande */
        for (let i = 0; i < cartContent.length; i++) {
            calcTotal += cartContent[i].price * cartContent[i].quantity
        }
        document.getElementById("order-price").innerHTML = "Total de la commande: " + (calcTotal/100) + "€";
    }).catch(error => console.log('Une erreur est survenue:', error));

    /* vide les infos stockés en local concernant la commande */
    localStorage.removeItem(`Ori${App}Cart`);
    localStorage.removeItem(`Ori${App}ClientContact`);
}
```

F

Description de la fonction:
confirmation_Ori(App, PostAdress)

Cette fonction a pour but d'afficher les informations de la commande effectuée via l'application.

Paramètre de la fonction:

- **App**: terme prédéfini: «teddy», «furniture» ou «camera». Il permet d'analyser le localStorage spécifique à l'application en utilisation , d'afin les différents éléments en lien et de créer un localStorage contenant les coordonnées du commanditaire.
- **PostAdress**: adresse de l'API utilisée pour les requète HTTP en méthode POST.

TEST N°7: **Vérifier la requête HTTP en méthode POST**

Vérifier que la requète fetch en méthode POST fonctionne correctement et renvoie une réponse avec l'identifiant de la commande.

Méthode du test:

- vérifier dans le navigateur: Network /«order»/Headers/StatutCode
- vérifier dans le navigateur: Network /«order»/Headers/Request Payload

Succès du test:

- si StatutCode = 201 (created), alors la commande à bien été envoyée.
- si "order" renvoie un objet contenant "orderId" (avec "contact" et "products").