OPEN CLASSROOMS WEB/DEV-P4

MVP - ORINOCO

Appareils photos de collection

ORINOCO

Notre catalogue | Panier (1 article) | Vider le panier

APPAREILS PHOTO DE COLLECTION

Zurss 50S

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Hirsch 400DTS

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Franck JS 105

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Kuros TTS

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

ORINOCO

Notre catalogue | Panier (0 article)

Détails produit

Franck JS 105

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Prix: 2099.00 Euro

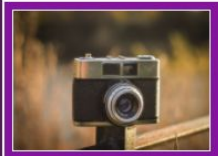
Quantité: 1 Option: 25mm 4.5 Ajouter au panier

ORINOCO

ORINOCO

Notre catalogue | Panier (1 article) | Vider le panier

Panier



	Zurss 50S	Prix U:	Qté:	+1
	Option: 35mm 1.4	499.00	1	

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Montant dû

499.00

Il y a 1 article dans le panier pour une somme de : 499.00 €

Références client

Nom*:	Tout en MAJUSCULE !
Prénom*:	1 majuscule au début...
Adresse*:	Numéro rue
CP*:	5 chiffres !
Ville*:	Tout en MAJUSCULE !
Email*:	Une adresse email !

Valider la commande

* Champ obligatoire

ORINOCO

Notre catalogue | Panier (0 article)

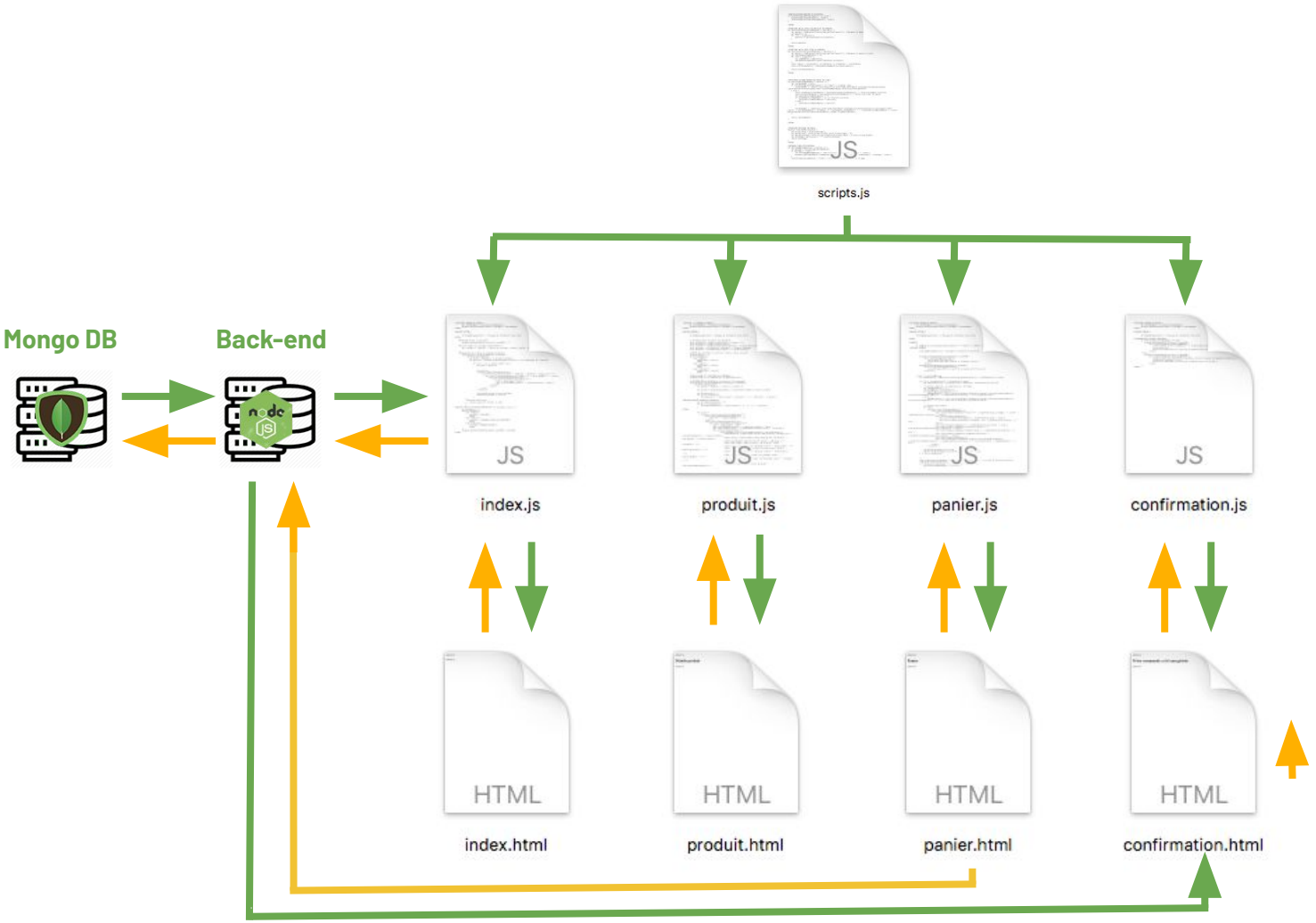
Votre commande a été enregistrée

MERCI

Confirmation de votre commande n° 47e0aac0-afbc-11ea-8e05-f55ca427fc6b

A bientôt

ORINOCO


JS
scripts.js
Mongo DB
Back-end
JS
index.js
JS
produit.js
JS
panier.js
JS
confirmation.js
HTML
index.html
HTML
produit.html
HTML
panier.html
HTML
confirmation.html

Initialisation du catalogue

Front-end

Back-end

Affichage navigateur

script.js /mémoire navigateur

Accueil
index.html

Selection du catalogue

Réponse
NODE JS
server

Initialisation
Panier vide

JSON

Affichage du catalogue
index.html

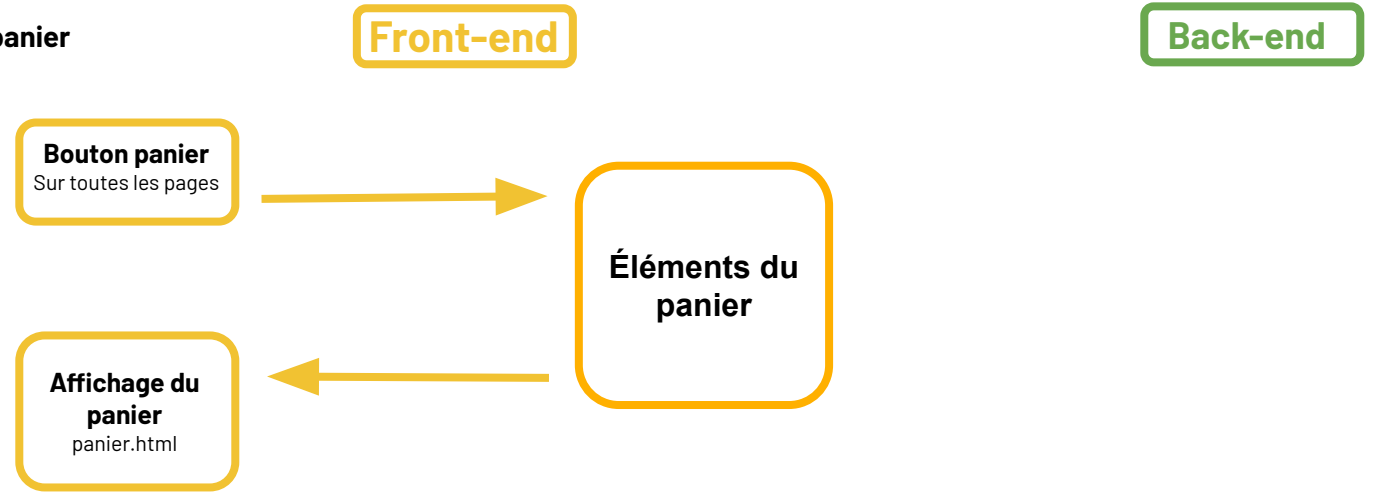
[X] id/name:price:description/imageUrl

Image cliquable
index.html

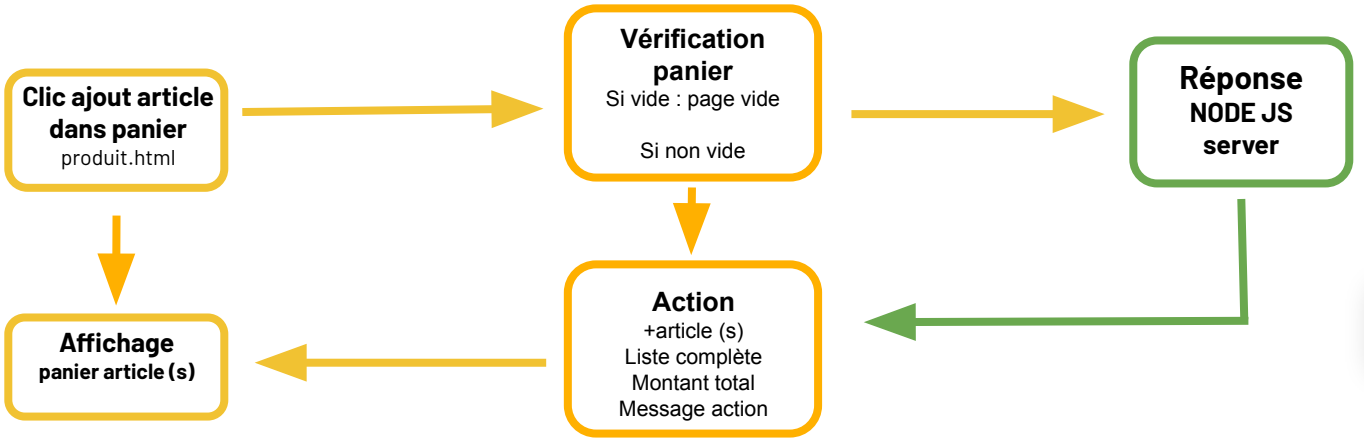
Réponse
NODE JS
server

Affichage article
produit.html

Affichage panier



Ajouter au panier



Front-end

Vidage du panier

Bouton vidage panier
produit.html
panier.html

→

Variable panier à 0

Retour
index.html

←

Affichage du catalogue

Modifications des quantités

+/- article
panier.html

→

Mise à jour du panier
Message action

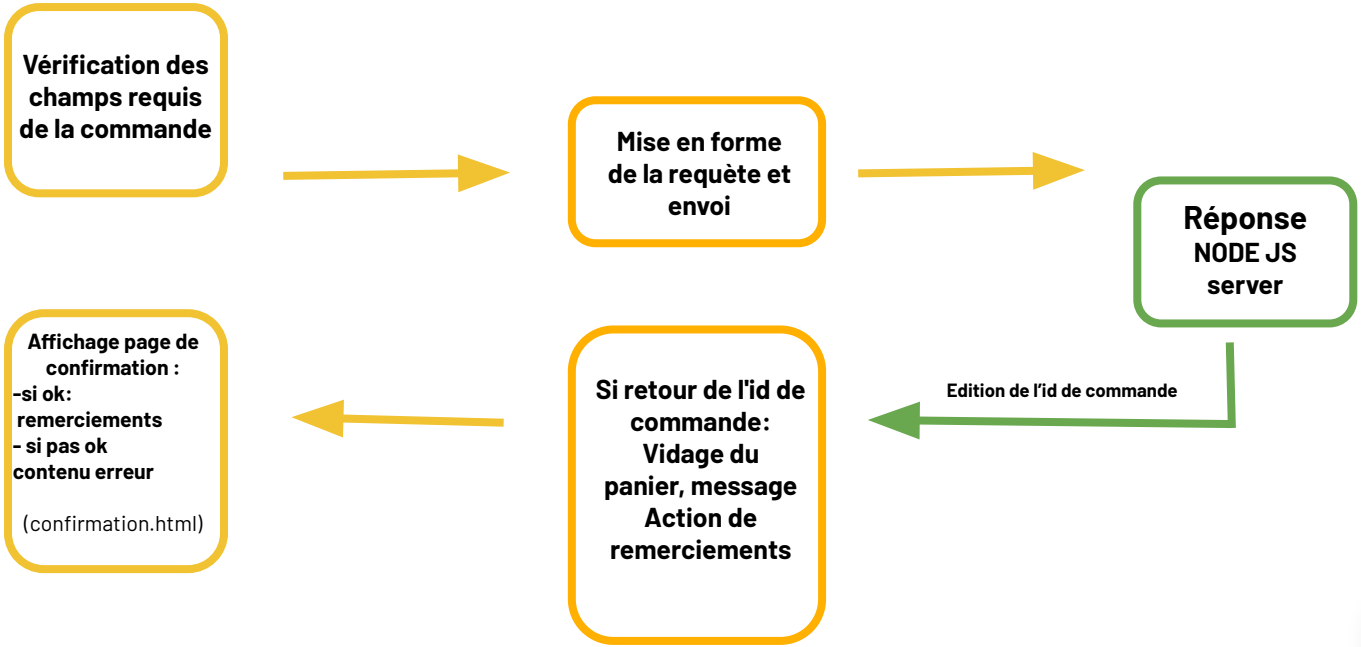
←

Mise à jour panier
calcul du montant
Message action

Front-end

Back-end

Confirmation de la commande



Plan de test ORINOCO

INTRODUCTION

Ce plan décrit l'approche de test de l' application **ORINOCO**, site de vente en ligne d'appareils photos de collection. Le but de l'application est de fournir à l'utilisateur une liste des produits disponibles sous forme de liste avec nom, photo (cliquable), et choix de différentes options liées à l'objectif. une vue personnalisée du produit, un panier dans lequel seront ajoutés l'article choisi. La page du panier permet d'augmenter ou de diminuer la quantité du produit et une fois la commande validée, il doit recevoir une confirmation avec un numéro d'enregistrement.

FONCTIONNALITÉS À TESTER

- Le panier doit contenir:
  Nom de l'article
  Image assortie
  Prix
  Montant
  Total de l'article - mis à jour en fonction du montant et du prix
  Bouton Supprimer - lorsque vous cliquez dessus, retirez l'article du panier et du stockage local

- Informations client - tous les champs sont obligatoires.
  Nom - en MAJUSCULES caractères spéciaux autorisés.
  Prénom - première lettre en Majuscules, caractères spéciaux autorisés.
  Adresse - lettres, chiffres et caractères spéciaux autorisés.
  Ville - en MAJUSCULES.
  E-mail - e-mail valide requis, @ et .XXXX dans l'adresse.

- Après la soumission réussie, l'utilisateur sera transféré vers la page de confirmation avec l'affichage de l'identifiant de la commande.

FONCTIONNALITÉS À TESTER

En arrivant sur le site, l'utilisateur doit voir les éléments disponibles dans la page d'affichage sous forme de liste. Les éléments sont chargés à partir de l'API et affichés dans un bloc individuel. L'objet comprend : l'**id du produit, le nom de l'article, le prix, une description, les options de lentilles (menu déroulant), l'image du produit (cliquable).**

Test de la connexion à l'API et retour du serveur Mongo DB
Connexion à la base **MongoDB.**
La gestion de la connexion se trouve sur le **back-end** dans **app.js (ligne 12 à 29)**
Un messsage d'erreur est renvoyé dans la console en cas d 'échec. **(ligne 20)**

```
12  mongoose.connect(
13    'mongodb+srv://will:nAcmfCoHGDgzrCHG@cluster0-pme76.mongodb.net/test?retryWrites=true',
14    { useNewUrlParser: true })
15    .then(() => {
16      console.log('Successfully connected to MongoDB Atlas!');
17    })
18    .catch((error) => {
19      console.log('Unable to connect to MongoDB Atlas!');
20      console.error(error);
21    });
22
23  app.use((req, res, next) => {
24    res.setHeader('Access-Control-Allow-Origin', '*');
25    res.setHeader('Access-Control-Allow-Headers', 'Origin, X-Requested-With, Content, Accept, Content-Type, Authorization');
26    res.setHeader('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE, PATCH, OPTIONS');
27    next();
28  });
```

Connexion à l'API

Connexion dans une promise **index.js (ligne 19)**

Un messsage d'erreur est renvoyé dans la console en cas d 'échec. **(ligne 46)**

```
18      /*Récupération du contenu du catalogue (promise)*/
19          connect("http://localhost:3000/api/"+catalogue)
20              .then(function(response){
21                  var objets = response;// si ok dans la promise
22                  var Resultats = document.getElementById("resultats"); //affichage dans id > resultats
23
24                      for (var i = 0; i < objets.length; i++) {
25                          var objet = objets[i];
26
27
28                          var article =
29                          '<article class="containerArticle">'+
30                              '<a href="produit.html?catalogue=' + catalogue + '&produit=' + objet._id + '">'+
31                                  '<div class="elementContainerArticle"><img src="' + objet.imageUrl + '" ></div>'+
32                                      '<div class="elementContainerArticle">'+
33                                          '<div class="contentDescriptionArticle">'+
34                                              '<h2>' + objet.name + '</h2>'+
35                                              '<div class="textDescription">' + objet.description + '</div>'+
36                                          '</div>'+
37                                      '</div>'+
38                              '</a>'+
39                          '</article>';
40
41                          //Affichage des articles du catalogue
42                          Resultats.innerHTML += article;
43                      }
44              })
45              .catch(function(error){
46                  console.log(error) //Erreur si echec
47              })
48
```

Connexion à l'API - récupération des données de l'article présenté
Connexion dans une promise **produit.js (ligne 35)**
Un messsage d'erreur est renvoyé dans la console en cas d'échec. **(ligne 86)**

```
33
34      /* RÉCUPÉRATION DU CATALOGUE en fonction de l'id (promise)*/
35      connect("http://localhost:3000/api/"+catalogue+"/"+idProduit)
36          .then(function(response){
37              let reponse = response; // resultat si promise OK
```

Connexion à l'API - Enregistrement et confirmation de la commande retour n°de commande
Connexion XMLHttpRequest()

```
296     };
297     xhttp.open("POST", "http://localhost:3000/api/cameras/order");
298     xhttp.setRequestHeader("Content-Type", "application/json");
299     xhttp.send(JSON.stringify(commandeToSend));
```

FONCTIONNALITÉS À TESTER

Initialisation/Fonctionnement du panier/Enregistrement de de la commande

Initialisation : vidage du panier scripts .js

```
1   /*INITIALISATION ARRIVÉE DU VISITEUR*/
2   if (localStorage.getItem("panier") === null) {
3       localStorage.setItem("panier", "vide");
4       localStorage.setItem("messagePanier", "vide");
5   }
```

Initialisation : remise à zéro du catalogue index .js

```
11    /*Nettoyage du div si non vide*/
12        document.getElementById("resultats").innerHTML = "";//remise à zéro du catalogue
13
```

Initialisation : choix du catalogue index .js

```
13
14    /*On peut changer de catalogue "manuellement"*/
15        var catalogue = "cameras"; // Options de catalogue : cameras, teddies, furniture
16
17
```

FONCTIONNALITÉS À TESTER - les fonctions

functionCalculArticlesDuPanier

```
9   /*FONCTION CALCUL DU NOMBRE D'ARTICLES TOTAL DANS LE PANIER*/
10  function functionCalculArticlesDuPanier (panier) {
11      var paniers = JSON.parse(localStorage.getItem("panier")); //Recupere le panier
12      var quantite = 0;
13      for (let x in paniers) {
14          quantite += parseInt(paniers[x].quantite);
15      }
16
17      return quantite;
18  }
```

functionCalculPrixTotalDuPanier

```
21  /*FONCTION CALCUL PRIX TOTAL DU PANIER*/
22  function functionCalculPrixTotalDuPanier (panier) {
23      var paniers = JSON.parse(localStorage.getItem("panier")); //Recupere le panier e
24      var tableauDeComptageDesPrix = [];
25      for (let x in paniers) {
26          var lignePanier = paniers[x];
27          tableauDeComptageDesPrix.push(lignePanier.prixAjour);
28      }
29      const reducer = (accumulator, currentValue) => accumulator + currentValue;
30      const prixTotalDuPanier = tableauDeComptageDesPrix.reduce(reducer);
31
32      return prixTotalDuPanier;
33  }
```

FONCTIONNALITÉS À TESTER - les fonctions

L'essentiel des fonctions sont concentrées dans scripts.js et sont indépendantes.

Fonctions de gestion de l'affichage et d'envoi des formulaires

`fonctionAffichageHeader` (ligne 36 à 55)

`affichageMessageAction` (ligne 69 à 77)

`fonctionSubmitProduit` (ligne 100 à 166)

Fonctions de gestion du panier

`transformPrice` (ligne 59 à 67)

`fonctionQuantitePlus` (ligne 186 à 206)

`fonctionQuantiteMoins` (ligne 212 à 227)

`fonctionClearPanier` (ligne 233 à 237)

`fonctionSubmitContact` (ligne 244 à 302)

FONCTIONNALITÉS À TESTER - validation des formulaires panier.js ligne 82 à 100

Nom : Lettres **MAJUSCULES** de A à Z maximum 20 caractères

```
pattern="^[A-Z]+$" maxlenght="20" placeholder="Tout en MAJUSCULE !"
```

Prénom : Lettres et accentuation possibles - Majuscule au début - Maximum 25 caractères

```
pattern="^[A-ZÀÁÂÃÄÅÇÑñÇÈÉÊËÌÍÎÏÒÓÔÕÖÙÚÛÜÝ]{1}[a-zçàáâãäåçèéêëìíîïðòóôõöøùúûüýÿ]+$" maxlength="25"
class="decoInputContact" placeholder="1 majuscule au début..."
```

Adresse : maximum 60 caractères

```
id="adresse" class="decoInputContact" placeholder="Numéro rue " maxlength="60"
```

Code postal : 5 chiffres de 0 à 9

```
id="codePostal" class="decoInputContact" pattern="[0-9]{5}" maxlength="5" placeholder="5 chiffres !"
```

Ville : Toutes lettres MAJUSCULES et accentuation possible - Maximum 25 caractères

```
id="ville" class="decoInputContact" pattern="^[A-ZÀÁÂÃÄÅÇÑñÇÈÉÊËÌÍÎÏÒÓÔÕÖÙÚÛÜÝ]+$" maxlength="30"
placeholder="Tout en MAJUSCULE !"
```

E-mail

```
id="email" class="decoInputContact" placeholder="Une adresse email !"
```