Rapport de projet web 2019

Sommaire

1. Contexte du projet

Ce projet web avait pour but de créer un site complet avec pour thème, le BDE du Cesi de Rouen. Cet objectif devait être réalisé en 2 semaines par un groupe de 4 personnes. Ce site devait comprendre plusieurs éléments, une page de connexion, une page d'inscription, une page contenant la boutique, une page par objets et vêtements vendus dans la boutique, une page contenant tous les évènements, une page par évènements, tous un espace administration permettant de faire des modifications sur les données du site et enfin une page d'accueil pouvant rediriger vers les principaux points du site.

2. Organisation du projet

Pour ce projet, le groupe composé de 4 membres, a été séparé en 2 groupes de 2 personnes. Un groupe devait s'occuper de la partie Front-end du site et l'autre groupe, de la partie Back-end. Ce système de groupe a été mis en place pendant les 10 premiers jours du projet et une fois qu'un groupe ou qu'une personne d'un groupe avait fini sa partie du projet, il proposait son aide aux autres membres du groupe de projet.

L'organisation du projet et la communication ont beaucoup été facilités par différents outils comme Trello pour la liste et l'attribution des tâches et Discord pour la communication et l'échange de données à distance.

3. HTML/CSS

Dans un premier temps nous avons fait une liste de toutes les pages que le site devait posséder, l'étape' suivante a été de se rendre sur http://www.cesi.fr pour récupérer les couleurs qui se dégageaient de ce site afin d'établir une charte graphique.

L'étape d'après a été de concevoir les mockups pour visualiser tous les éléments qui devrait figurer sur notre site. Nous avons par la suite récupéré des polices libres de droits pour les titres, texte principal, titre boutique.

Une fois cette étape terminée il ne nous resta plus qu'à faire les maquettes graphiques.

Enfin, lorsque les maquettes étaient faites, il ne nous restait plus qu'à reproduire avec du code HTML/CSS les maquettes graphiques de départ. C'est une tâche qui peut paraitre peut complexe mais qui demande tout de même un certain temps.

4. Options graphiques

Toutes ces options graphiques ont été ajoutées à l'aide de Javascript et de JQuery.

Barre de recherche

Pour rendre le site plus dynamique, nous avons décidé de créer une barre de recherche qui s'ouvre et se rétracte avec une pression sur la loupe de recherche. Cela permet de gagner en place dans le header tout en étant assez intuitif pour l'utilisateur.

Carrousel

Dans la boutique, il est possible de visionner les différents produits à l'aide d'un carrousel lorsque le site est en version portable. Cela permet de gagner en lisibilité pour chaque produit et permet un certain confort pour l'utilisateur.

Icônes des réseaux sociaux

Cette option n'est pas très importante visuellement dans le site web mais est tout de même non négligeable car elle montre à l'utilisateur que le site a été pensé dans les moindres détails. La couleur de chaque logo correspond à sa couleur d'origine lorsque l'on passe la souris dessus. Facebook et en bleu foncé, Twitter en bleu clair et enfin Instagram est en un dégradé de couleurs allant du violet au jaune.

5. Procédures stockées

Procédures liées aux activités

Les procédures liées aux activités sont principalement destinées à être utilisées par les membres du BDE, elles permettent de créer et modifier une activité. Supprimer ou modifier la visibilité d'une image et d'un commentaire des autres utilisateurs.

Pour le personnel du CESI, cela leur permet de signaler une image, une activité ou un commentaire en plus de celle des utilisateurs normaux.

Auprès des utilisateurs normaux, elles permettent d'afficher les commentaires, mais aussi de poster et supprimer leurs images et commentaires, d'aimer une image ou d'afficher et s'inscrire à une activité.

Procédures liées aux utilisateurs

Les procédures liées aux utilisateurs permettent à un visiteur de créer un compte, de le supprimer ou de se connecter avec le compte. Ce dernier permet alors de s'inscrire à des activités, poster des images, acheter des articles, poster des commentaires, etc…

Procédures liées aux produits

Les procédures liées aux produits permettent aux membres du BDE d'ajouter des produits, de les modifier, de changer leur stock, de les faire disparaitre, de créer des catégories et de leur attribuer.

Pour tous les utilisateurs connectés, d'autres procédures permettent de gérer leur panier en y ajoutant ou retirant des produits et en modifiant la quantité voulue.

Enfin, pour toute personne même non identifiée sur le site, des procédures plus génériques permettent d'afficher un produit en détail ou la liste des produits.

6. Autocomplétions

L'API Restful a été développé sur un serveur « Node.js ». Pour créer les différentes routes énumérées ci-dessous nous avons utilisé le framework « Express ».

Cladogramme des URL de l'Api Rest :

```
/
|_ Token/

|_ Produits/
    |_ Ventes/
    |_ Stoke/
    |_ Categorie/
    |_ Images/

|_ Commande/
    |_ Utilisateur/

|_ Activites/
    |_ Photo/
        |_ Commentaires/
            |_ Signaler/

        |_ Like/
        |_ Signaler/

    |_ Inscription/
    |_ Signaler/

|_ Utilisateurs/

|_ Panier/
```

Le routage « Express » permet de créer des routes avec plusieurs méthodes, cependant pour les méthodes ayant besoin de variables de formulaire nous sommes obligés d'utiliser un framework complémentaire appelé « body-parser ».

Pour créer une route nous devons indiquer au routeur la méthode à écouter, dans cette méthode nous lui indiquons la route à créer puis l'action à faire sous forme de callback, mais nous pouvons également lui définir des Middleware.

Exemple de création d'une route :

```
// On crée un nouveau routeur :
let RouterProduitsCategorie = Express.Router();

// Sans Middleware :
RouterProduitsStocke.get('/Produits/Categorie/', ProduitsCategorie.getProduitCategorie);

// Avec Middleware :
RouterProduitsStocke.post('/Produits/Images/:IdentifiantProduit', Middleware.Login,
ProduitsImages.postProduitImages);

// On ajoute les routes à l'application Express :
App.use(RouterProduitsCategorie);
```

Les middlewares sont aussi des callbacks, elles permettent d'exécuter du code avant d'exécuter l'action à faire au chargement de la route. Dans notre cas elles permettent de sécuriser l'accès, détails dans la connexion avec l'API.

Pour l'API nous avons dû aussi régler un souci de « cross-domain », c'est-à-dire que nous ne pouvions pas accès au serveur Node avec des requêtes ajax sur le serveur apache.

Nous avons ajouté les lignes suivantes dans le fichier Node.js :

```
App.use(function(Requete, Reponse, Next) {
    Reponse.header("Access-Control-Allow-Origin", "*");
    Reponse.header("Access-Control-Allow-Headers", "token, Origin, X-Requested-With, Content-Type,
Accept");
    Reponse.header("Access-Control-Allow-Methods", "PUT, DELETE, GET, POST");
    Next();
});
```

7. Développement de l'API

L'API REST a été développée sur un serveur Node.js. Pour créer les différentes routes énumérées ci-dessous. Nous avons utilisé le framework Express.

Cladogramme des URL de l'API REST :

Le routage Express permet de créer des routes avec plusieurs méthodes, cependant pour les méthodes ayant besoin de variable de formulaire nous sommes obligés d'utiliser un framework complémentaire appelé body-parser.

Pour créer une route nous devons indiquer au routeur la méthode à écouter, dans cette méthode nous lui indiquons la route à créer puis l'action à faire sous forme de callback, mais nous pouvons également lui définir des middlewares.

Exemple de création d'une route :

```
// On crée un nouveau routeur :
let RouterProduitsCategorie = Express.Router();

// Sans Middleware :
RouterProduitsStocke.get('/Produits/Categorie/', ProduitsCategorie.getProduitCategorie);

// Avec Middleware :
RouterProduitsStocke.post('/Produits/Images/:IdentifiantProduit', Middleware.Login,
ProduitsImages.postProduitImages);

// On ajoute les routes à l'application Express :
App.use(RouterProduitsCategorie);
```

Les middlewares sont aussi des callbacks, elles permettent d'exécuter du code avant d'exécuter l'action à faire au chargement de la route. Dans notre cas elles permettent de sécuriser l'accès, décrit en détail dans la partie suivante.

Pour l'API nous avons dû aussi régler un problème de « cross-domain », c'est-à-dire que nous ne pouvions pas accéder au serveur Node avec des requêtes AJAX depuis le serveur Apache.

Nous avons ajouté les lignes suivantes dans le fichier Node.js :

```
App.use(function(Requete, Reponse, Next) {
    Reponse.header("Access-Control-Allow-Origin", "*");
    Reponse.header("Access-Control-Allow-Headers", "token, Origin, X-Requested-With, Content-Type,
Accept");
    Reponse.header("Access-Control-Allow-Methods", "PUT, DELETE, GET, POST");
    Next();
});
```

8. Connexion avec l'API

Pour accéder aux informations de la base de données, on va donc utiliser les routes créées sur l'API.

En PHP, on *utilise* cURL pour effectuer ce genre de requête :

```
$ch = curl_init();
curl_setopt($ch, option: CURLOPT_URL, value: $IP_NODE."/Utilisateurs/");
curl_setopt($ch, option: CURLOPT_RETURNTRANSFER, value: true);
curl_setopt($ch, option: CURLOPT_CUSTOMREQUEST, value: "POST");
curl_setopt($ch, option: CURLOPT_POSTFIELDS, http_build_query(
        array(
            'Nom' => $_POST["surname"],
            'Prenom' => $_POST["name"],
            'Mail' => $_POST["mail"],
            'MotDePasse' => sha1($_POST["password"]),
            'Localisation' => 4
        )
));
$response = curl_exec($ch);
curl_close($ch);
$data = json_decode($response);
```

En JavaScript, on utilise AJAX :

```
$.ajax({
    dataType: "json",
    url : IPNODE+'/Panier/'+data['Message'][id]['PRO_ID'],
    type: 'PUT',
    data: {Quantite: newQty},
    headers: {'token': token},
    success: function(message){
        console.log(message);
        let stbefore = document.getElementById( elementId: "st"
        document.getElementById( elementId: "st"+id).innerText
        document.getElementById( elementId: "tt").innerText = (
    },
    error: function(resultat, statut, erreur){
        console.log(erreur);
    }
});
```

Pour une grande partie des routes, il est nécessaire de fournir un « token » d'authentification lié à l'utilisateur. Pour se faire, sur notre page de connexion, nous effectuons une requête cURL renseignant l'adresse et le mot de passe hashé (à l'aide de l'algorithme SHA1) de l'utilisateur. Ensuite, le retour de la requête nous spécifie si oui ou non l'utilisateur est identifié, et si oui on renvoie le token.

C'est alors qu'on place le token dans une variable de session PHP qui sera alors utilisée dans les autres pages à la fois pour savoir si l'utilisateur est bien connecté mais aussi pour le renseigner comme suit dans le header des requêtes vers l'API :

En PHP avec cURL :

```
curl_setopt($ch, option: CURLOPT_HTTPHEADER,
    array(
        'token:'.$_SESSION["Token"]
    ));
```

En JavaScript avec AJAX :

```
$.ajax({
    dataType: "json",
    url: IPNODE + '/Panier/' + data
    type: 'DELETE',
    headers: {'token': token},
    success: function (message) {
```

JavaScript n’ayant pas accès au token alors contenu dans la session PHP, on effectue une requête AJAX vers une page « get_token.php » renvoyant le token de l’utilisateur.

Le token est ensuite lu par l’API qui l’associe au compte qui lui correspond depuis la connexion. Les informations du compte peuvent alors être utilisées pour vérifier les permissions et accorder ou non l’accès aux routes utilisées.

Lorsque l’utilisateur se déconnecte, la session est détruite et plus aucun accès au token n’est possible.