0 - Sommaire

1) Prérequis et fonctionnalités

a) Identifiants et connexion à l'API

Les identifiants de connexion à l'API sont propres à chaque casse automobile. Ceux-ci sont fournis exclusivement par Opisto. Ils sont composés des données suivantes :

- ID de casse (chiffre entier)
- Nom d'utilisateur
- Mot de passe
- Clé secrète

Ces 4 valeurs doivent être insérées dans le fichier suivant : utils/const.php (API_CASSE_ID / API_USER / API_PASSWORD / API_SECRET_ID). Les autres valeurs, notamment API_HOST, ne doivent pas être modifiées.

ATTENTION : ces valeurs sont valables pour utiliser l'API en préproduction. Pour la mise en ligne du site, Opisto doit nous fournir des identifiants spécifiques à la production (voir partie n°4 : Mise en production).

b) Dépendances de développement

Le dossier est composé de nombreux fichiers nécessaires au bon fonctionnement de l'application. Certains seront automatiquement inclus par le biais du gestionnaire de paquets

NPM et de l'automatiseur de tâches GULP, mais certaines propriétés et fonctions devront être copiées/collées à la main.

Certains fichiers peuvent déjà exister dans votre projet, comme par exemple includes/header.php. Le code à récupérer pour faire fonctionner l'application correctement est situé entre les commentaires " START DEVDEP " et " END DEVDEP ".

Attention : certains commentaires contiennent des remarques pour plus de compréhension. Il peut également y avoir plusieurs sections à récupérer au sein d'un seul et même fichier, il faut donc faire preuve de vigilance lors de l'installation de cette solution sur un projet !

c) Configuration requise

Pour fonctionner correctement, l'application a besoin d'un serveur exécutant PHP (version 7.2 minimum), ainsi qu'une base de données MySQL.

/!\ Lors de la création de la base de données, penser à sélectionner le jeu de caractères : utf8_general_ci.

Un modèle de base de données est fourni dans le dossier (sql/opisto_tests.sql).

Le serveur de production doit impérativement être équipé d'un certificat SSL (protocole HTTPS obligatoire pour la communication entre l'appli web et l'API). Ceci est également valable pour le serveur de préproduction.

Un certificat temporaire est fourni (répertoire /certs) pour assurer le fonctionnement de l'application en local. **Ce dossier ne doit pas être uploadé sur le serveur (preprod / prod).**

d) Éléments à fournir par le client

Conditions générales de vente : **à remplacer dans le répertoire /docs** (le mieux étant de remplacer simplement le fichier, mais lui laisser le même nom).

Les réductions particulières à appliquer pour les clients professionnels (si existantes).

Les réductions particulières à appliquer sur les frais de port (si elles existent).

Les réductions particulières à appliquer à tous les clients (si elles existents).

Les réductions sont présentes dans le fichier **utils/const.php**.

Elles se présentent sous la forme d'un tableau, permettant de différencier les réductions à appliquer sur les frais de port (delivery) et sur les pièces (parts).

Chacune de ces catégories peut posséder une ou plusieurs réductions, qui contiennent 3 index :

- Label : c'est le libellé de la réduction, pour pouvoir l'afficher sur le site
- percent : c'est la valeur (en %) de la réduction. 20 = 20% de réduction
- min : c'est le nombre minimum de pièce(s) à partir duquel on applique la réduction.

Exemple : min = 3 >> la réduction s'appliquera à partir de 3 pièces dans le panier

Il faut également récupérer les différents tarifs de livraison, que ça soit pour la France mais également pour l'étranger (si la casse a autorisé l'envoi dans d'autres pays).

Pour cela, il faut commencer par créer une commande, et aller jusqu'au processus de paiement pour la créer sur l'API.

Ensuite, il faut aller dans "mon compte" puis "mes commandes" et afficher cette commande.

Ouvrir le fichier "controllers/accountController.php", puis rechercher la ligne "require 'views/accountDetailsOrderView.php';"

Juste avant cette ligne, insérer "var_dump($order_api->Casse->ShippingCountries)".

Cette ligne va permettre de récupérer l'intégralité des livraisons possibles (vers la france, ou autre), ainsi que les délais possibles (standard / express / autre) mais également le coefficient à appliquer à chaque mode de livraison, et la liste des ID de categories concernées.

Attention à bien supprimer la ligne une fois cette manipulation effectuée.

Ces données doivent ensuite être retranscrites dans la constante "SITE_DELIVERY_DELAYS" du fichier "utils/const.php".

2) Installation et paramétrage

a) Récupération des ressources

ETAPE 1 : Récupérer les dépendances

Ouvrir le fichier " package.json " situé à la racine du répertoire.

Récupérer le contenu de "dependencies" et le copier/coller dans votre fichier "package.json", à la racine de votre projet.

Ouvrir la console > cd + chemin vers votre projet > npm install pour télécharger et installer les dépendances manquantes (Slick et bootstrap-select).

ETAPE 2 : Cloner les fichiers du répertoire source vers votre projet

Si certains fichiers présents dans le répertoire sont déjà présents dans votre projet, ne pas les écraser !

Lire avec attention la partie qui va suivre avant de procéder à la copier des fichiers.

Copier les répertoires et fichiers du répertoire "app" pour les inclure à votre application. **Si des répertoires existent déjà au sein de votre projet**, copier simplement le contenu du répertoire dans le vôtre (ex. /img ou /js).

Si certains fichiers présents dans le répertoire sont déjà présents dans le répertoire du même nom dans votre projet : récupérer **uniquement** les parties comprises entre les commentaires intitulés **"START DEVDEP" et "END DEVDEP", à l'intérieur de ces fichiers.**

Penser également à récupérer le contenu du fichier .htaccess, en prenant soin de

supprimer les éventuels doublons avec votre configuration existante (RewriteEngine on, Options +FollowSymLinks etc.)

Ne pas modifier le nom des fichiers : les fichiers compris dans la solution doivent impérativement garder le même nom. Si les fichiers sont renommés, la solution risque de présenter des problèmes et ne fonctionnera pas.

Certains fichiers sont uniquement utilisés pour la création d'un design générique pour le développement de l'application, et sont donc facultatifs lors du déploiement sur votre projet. Voici les fichiers concernés :

- Répertoire "fonts" : attention, des icônes sont nécessaires pour le fonctionnement de l'appli. Penser à récupérer le répertoire "dev".
- Répertoire "sass" : pour savoir quels sont les fichiers à récupérer dans ce répertoire, ouvrir le fichier /sass/app.scss et consulter les commentaires "START DEVDEP" et "END DEVDEP" pour connaître les fichiers à récupérer et à inclure à votre projet.
  /!\ S'il n'y a pas de balise START DEVDEP / END DEVDEP dans le fichier, cela signifie qu'il ne faut rien récupérer (il sert simplement de design générique pour le développement de l'application, mais n'a aucune utilité sur le site final).
- Répertoire "includes" : de la même manière que précédemment, récupérer uniquement le contenu situé entre les commentaires "START DEVDEP" et "END DEVDEP". Des remarques complémentaires ont été ajoutées pour plus de compréhension.

Des adaptations de design en SCSS peuvent bien évidemment être appliquées sur la solution si nécessaire. Il n'est pas impératif de garder le design générique ayant été utilisé pour le développement de la solution, il est tout à fait possible de faire des adaptations de celui-ci pour que le design global de l'application soit en accord avec celui du site vitrine.

Quelques précisions, fichier par fichier :

- includes/form-search.php doit être intégralement remplacé par celui fourni dans la solution

- includes/header.php : supprimer "session_start()" au début du fichier, le démarrage de session se fait dans le MainController.

- sass/ : pour éviter toute confusion avec votre projet, les composants importants pour l'affichage de la solution ont été stockés dans des fichiers de noms différents (exemple : les icones ne sont pas dans components/_fonts.scss mais dans components/_icons.scss). Hormis pour le fichier "app.scss" ou vous devez récupérer une partie du code, pour les sous-répertoires du répertoire "sass", vous ne devez récupérer que les fichiers qui ne sont pas présents dans votre projet.

Exemple : ne récupérez pas "partials/_headers.scss" > il ne contient aucune propriété CSS nécessaire à l'affichage de l'application.

- Le fichier "composer.json" ainsi que "composer.lock" ne sert à rien pour votre projet, inutile de le copier/coller.

- includes/helper.php : penser à bien récupérer le "default" à la fin du switch ($PAGE_name). Laisser des chaines vides (enlever "Développement API …").

- includes (header.php et footer.php) : attention à bien récupérer les fonctions ob_start() (header) et ob_end_flush(footer) > elles servent à mettre en mémoire tampon la page HTML le temps que les calculs PHP soient effectués, et évitent les erreurs de type : cannot modify header.

- includes/helper.php : il faut définir le title / description des pages panier et compte.

b) Création de la base de données

Une copie de la structure de la base de données est présente dans le répertoire /sql, il est nécessaire de l'installer sur le serveur local pour faire tourner le site.

Penser à mettre à jour les informations de connexion à celle-ci. Ces informations sont présentes dans le fichiers /utils/const.php (fichier rassemblant les constantes utilisées sur le site).

c) Fichiers de configuration

Avant de commencer à lancer votre projet pour vérifier le fonctionnement de celui-ci, vous devez éditer le fichier /utils/const.php. Ce fichier contient les principales constantes utilisées sur le site, c'est un fichier de configuration qui va permettre de déterminer sur quelle casse automobile vous allez travailler, mais également le nom du site, le nom du domaine, différentes url pour la création de compte / commandes.

Voici le détail des informations qu'il contient :

DB_HOST : le nom d'hôte de la base de données

DB_NAME : le nom de la base de données

DB_USER : le nom d'utilisateur pour se connecter à la base de données

DB_PASS : le mot de passe pour se connecter à la base de données

API_HOST : l'hôte de l'API d'Opisto. En preprod, ne pas le modifier.

API_USER : le nom d'utilisateur pour se connecter à l'API (fourni par Opisto, différent en préprod et en prod)

API_CASSE_ID : l'identifiant de la casse automobile (fourni par opisto, ne change pas en preprod et en prod).

API_PASSWORD : le mot de passe de connexion à l'API (fourni par Opisto, différent en préprod et en prod).

API_SECRET_ID : le secret ID de connextion à l'API (fourni par opisto, différent en préprod et en prod)

SITE_SALT : Clef secrète permettant de générer des tokens de sécurité. Elle peut ne pas être modifiée, mais il est fortement recommandé de générer une nouvelle clef pour chaque nouveau site. La clef se compose d'une suite de 16 caractères (chiffres / lettres / majuscules), et doit être entrée à la suite de 5.

SITE_SALT_KEY : C'est la clef de déchiffrage des token. Elle doit correspondre à SITE_SALT, mais avec un "$" à la fin.

SITE_BASE : l'URL de base (absolue) du site. Doit être modifié en local, en preprod, ainsi qu'en production. Elle correspond à l'URL absolue du dossier où se trouve le fichier "index.php".

SITE_NAME : Le nom de la société (Exemple : Mazard Pièces Auto)

SITE_DOMAIN : le domaine utilisé, sans http ni les "www" (Exemple : piecesauto.fr)

SITE_LOGO : l'URL absolu du logo de la société (utilisé notamment dans les envois d'email). Il faut impérativement que ce soit une URL accessible en ligne (pas d'url relative).

SITE_MAIL : L'adresse email du propriétaire du site.

SITE_PHONE : Le numéro de téléphone de la casse Auto

SITE_STREET : Le numéro et nom de rue de l'adresse de la casse auto.

SITE_STREET_ADDITIONNAL : Le complément d'adresse (Batiment, Etage, Porte etc.) de l'adresse de la casse automobile

SITE_ZIPCODE : le code postal de l'adresse de la casse auto

SITE_CITY : la ville de l'adresse de la casse auto

SITE_COUNTRY : le pays de l'adresse de la casse auto

SITE_CONTACT : l'URL (absolue) de la page contact du site

SITE_VALID_ACCOUNT : l'URL (absolue) de la page de validation de compte client

SITE_VALID_EMAIL : l'URL (absolue) de la page de validation de changement d'email

SITE_RESET_PASSWORD : l'URL (absolue) de la page de réinitialisation de mot de passe.

SITE_DELETE_ACCOUNT : l'URL (absolue) de la page de suppression de compte

SITE_CANCEL_PRO : l'URL (absolue) de la page d'annulation de passage en compte pro

SITE_DETAIL_ORDER : l'URL (absolue) de la page de détail d'une commande

ATTENTION : pour les URL absolues, ne pas modifier les valeurs !

SITE_PRO_REDUCTIONS : un tableau contenant les réductions appliquées aux clients professionnels (pièces & frais de port)

SITE_REDUCTIONS : un tableau contenant les réductions appliquées à tous les clients (sur les pièces, et sur les frais de port).

SITE_DELIVERY_DELAYS : un tableau contenant les délais possibles de livraison disponible lors de la commande d'une pièce.

d) Création du cache

Pour éviter de surcharger l'API avec des requêtes trop nombreuses, il est nécessaire de créer des fichiers de mise en cache pour certaines données, comme les marques et modèles de véhicules.

Pour cela, commencez par créer un répertoire "cache" à la racine de votre projet (dans le répertoire app). Une fois votre projet démarré, accédez à l'URL suivante : votre-nom-de-projet/app/utils/cacheControl.php. Cela aura pour effet de mettre en cache les marques de voitures, les gammes et les modèles pour chaque marque, ainsi que les catégories de pièces de véhicules.

ATTENTION : cette opération est très longue et va prendre plusieurs minutes à être exécutée. Veillez à ne pas mettre l'ordinateur en veille ou à ne pas fermer le navigateur durant ce laps de temps, ce qui aurait pour effet de stopper la mise en cache, et impliquera donc de relancer son exécution.

Autre information importante : cette mise en cache devra être renouvelée tous les 3

mois pour permettre de récupérer les nouvelles marques / gammes et modèles de véhicules. Il faudra donc programmer, sur l'hébergement, une tâche CRON qui s'exécutera tous les 3 mois, de préférence au milieu de la nuit pour éviter une surcharge du serveur à des heures de grande fréquentation.

3) Mise en préproduction / en production

Fichiers à modifier pour mise en production / préproduction :

- Créer la base de données pour la preprod / prod, et importez le modèle (sur le SQL Privé sur OVH, possibilité d'utiliser la base " opisto_tests ").

- Ne pas uploader sur le serveur les répertoires suivants : /certs et /sql.

- utils/fonctions.php > enlever les certificats SSL sur les fonctions callAPI_get et callAPI_post (les lignes à supprimer sont commentées).

- utils/const.php > mettre à jour les valeurs de ce fichier pour que l'application puisse fonctionner. Voir les détails ci-dessus.

- includes/head.php > mettre à jour le contenu de la balise "<base>" : mettre l'URL du site internet

- models/professionalManager.php : supprimer les lignes commentées "supprimer en prod"

- Mettre à jour le fichier includes/mangopay_config.php (uniquement pour mise en production) et remplacer l'url de la sandbox par : api.mangopay.com.
Mettre également le nom d'utilisateur Mangopay (opisto à la place de opistodevelopment)

- Mettre à jour l'url de l'API Opisto dans utils/const.php (uniquement pour la mise en prod, décommenter les lignes "Prod" et commenter les lignes "Dev").

- Mise en production uniquement : fichier /utils/cacheControl.php > mettre à jour partout ou il est présent : /4 à remplacer par /9 (pour utiliser CatData, qui est le modèle des marques / modèles / gammes utilisé en production, alors que 4 est celui utilisé en préprod).
/!\ ATTENTION : consigne non valable à partir de la v2.15 de l'API > celle-ci est passée directement sur le 9 (en preprod et en prod).