Documentation Technique

Table des matières

Introduction

Ce document décrit l'architecture technique du projet e-commerce de NEW VET, une plateforme de vente en ligne pour une société de prêt-à-porter. Le projet comprend la création d'un site web e-commerce, un back-office pour la gestion des produits et des commandes, ainsi que l'intégration de services tiers (Azure Blob Storage, Mailjet). Le développement est réalisé en utilisant PHP, HTML, CSS et JavaScript, avec XAMPP comme environnement de développement local et MySQL pour la gestion des bases de données.

Architecture Générale

Le projet est structuré en trois parties principales :

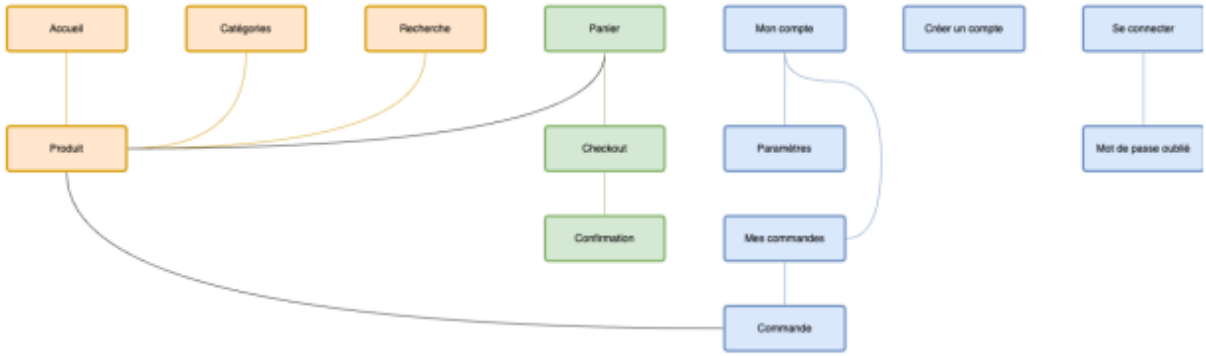
1. Front-End :
    - Développé en HTML, CSS et JavaScript pour créer une interface utilisateur responsive.
    - Utilisation de JavaScript pour gérer les interactions et appels AJAX vers le back-end.
2. Back-End :
    - Développé en PHP, le back-end fournit des accès à la base de données pour le front-end et un panneau d'administration pour la gestion des produits, des commandes et des utilisateurs.
    - Le serveur fonctionne sous Apache (via XAMPP) avec une base de données MySQL et un système de gestion de base de données phpMyAdmin
3. Services tiers :
    - Azure Blob Storage pour la gestion des fichiers (images de produits).
    - Mailjet pour l'envoi d'e-mails transactionnels.

Technologies Utilisées

- PHP : Langage côté serveur pour la gestion des requêtes HTTP, des API propre a newvet et du back-office.
- HTML/CSS : Utilisés pour structurer et styliser les pages du site web.
- JavaScript : Gère les interactions dynamiques du site, notamment les appels AJAX pour la mise à jour du panier, les formulaires, etc.

- MySQL : Base de données relationnelle pour stocker les informations sur les produits, utilisateurs, et commandes.
- Azure Blob Storage : Stockage des images et autres fichiers lourds.
- Mailjet : Gestion des e-mails transactionnels (confirmations de commande, inscription, etc.).
- XAMPP : Environnement de développement local avec Apache, PHP et MySQL.
- Composer : Gestionnaire de dépendances pour les bibliothèques PHP.

Structure du Projet



1. Front-End

Le front-end est une Single Page Application (SPA) légère avec les pages suivantes :

- Accueil : Présente un carrousel des produits vedettes et un accès rapide aux catégories.
- Catégorie : Liste des produits filtrés par catégorie.
- Produit : Page de détail des produits avec option d'ajout au panier.
- Panier : Vue récapitulative des articles ajoutés avant le passage à la commande.
- Checkout : Processus de validation de commande avec paiement.
- Connexion/Inscription : Pages pour la gestion des comptes utilisateurs.

Le site est entièrement responsif, assurant une compatibilité sur mobile et desktop.

2. Back-End

Le back-end, développé en PHP, gère les fonctionnalités suivantes :

- Back-office : Interface d'administration pour gérer les produits, les catégories, les utilisateurs et les commandes.

- Authentification : Système de connexion pour les utilisateurs et les administrateurs avec gestion des sessions.

Exemple de route API en PHP pour récupérer les produits d'une catégorie :

```php
function GetProductsByCategory($pdo, $categoryID, $activeOnly = 1) {
    $query = "SELECT * FROM produits WHERE categorie_id = :categoryID AND est_actif >= :activeOnly";
    $statement = $pdo->prepare($query);
    $statement->bindParam(':categoryID', $categoryID, PDO::PARAM_INT);
    $statement->bindParam(':activeOnly', $activeOnly, PDO::PARAM_INT);

    if (!@$statement->execute()) {
        $errorInfo = $statement->errorInfo();
        $errorMessage = json_encode($errorInfo[2]);
        echo "<script>console.error($errorMessage);</script>";
        return false;
    }

    return $statement->fetchAll(PDO::FETCH_ASSOC);
}
```

Déploiement

XAMPP doit être installé sur votre machine. Si ce n'est pas le cas, vous pouvez le télécharger depuis apachefriends.org.

Sur le répertoire **Github** (https://github.com/Simoniaque/E-commerce), téléchargez le dossier du projet et une fois dézippé déplacé le sous *C:\xampp\htdocs*.

Le site est hébergé sur un serveur compatible PHP/MySQL avec un système de gestion des fichiers via Azure Blob Storage accessible depuis l'adresse http://localhost/E-commerce-dev/ en local. Si vous avez renommé le dossier autrement, modifiez également l'adresse suivant le nom du dossier.

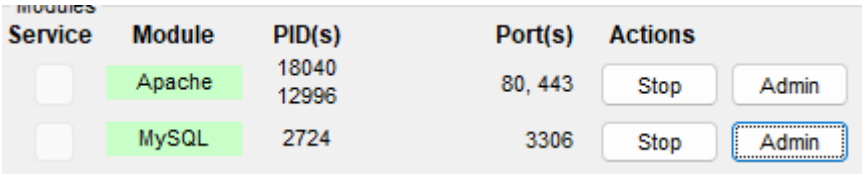
Base de Données

La base de données **MySQL** est structurée en plusieurs tables pour gérer les utilisateurs, produits, commandes, catégories, et autres entités du projet e-commerce.

Installation de la Base de Données avec XAMPP

1. Lancer XAMPP :

- Ouvrez l'application **XAMPP**.
- Démarrez les services **Apache** et **MySQL** en cliquant sur les boutons **Start** dans le panneau de contrôle.



2. Accéder à phpMyAdmin :

- Dans votre navigateur, rendez-vous à l'adresse suivante : *http://localhost/phpmyadmin* ou cliquez sur le bouton **Admin** du service MySQL dans le panneau de contrôle.
- Cela ouvrira l'interface de **phpMyAdmin**, un outil pour gérer **MySQL** via une interface graphique.

3. Créer une base de données :

- Dans **phpMyAdmin**, cliquez sur l'onglet **Bases de données**.
- Dans le champ **Créer une base de données**, entrez le nom newvet (ou un autre nom si vous le souhaitez) puis sélectionnez le codage **utf8_general_ci** et cliquez sur **Créer**.

4. Importer le schéma de la base de données :

- Après avoir créé la base de données, sélectionnez-la dans la liste des bases de données sur le côté gauche de l'interface.
- Cliquez ensuite sur l'onglet **Importer**.
- Cliquez sur le bouton **Choisir un fichier** et sélectionnez le fichier **newvet.sql**.
- Cliquez sur **Exécuter** pour importer les tables et données dans votre base de données.

5. Vérification :

- Une fois l'importation terminée, vous devriez voir les tables de la base de données (ex. utilisateurs, produits, commandes, etc.) dans **phpMyAdmin**.
- Si tout s'est bien passé, la base de données est maintenant configurée et prête à être utilisée avec votre projet.
- Vérifier le peuplement des tables notamment celle des utilisateurs

Schéma de la base de données

Voici les principales tables et leur structure.

- **Utilisateurs** : Stocke les informations des utilisateurs, y compris les administrateurs. Les mots de passe sont hashés pour des raisons de sécurité.
- **Adresses des Utilisateurs** : Gère les adresses de facturation et de livraison des utilisateurs.
- **Produits** : Contient les détails des produits, leur prix, description, et stock.
- **Commandes** : Détaille les commandes passées par les utilisateurs, incluant les adresses de facturation et de livraison.
- **Catégories** : Regroupe les produits par catégories, avec un statut pour indiquer si la catégorie est active.
- **Paniers** : Stocke les articles que les utilisateurs ajoutent à leur panier avant de passer commande.
- **Détails du Panier** : Gère les produits ajoutés dans les paniers.

Exemple de structure MySQL :

```sql
--
-- Structure de la table `adresses_utilisateurs`
--

CREATE TABLE `adresses_utilisateurs` (
  `id` int(11) NOT NULL,
  `utilisateur_id` int(11) NOT NULL,
  `voie` varchar(255) NOT NULL,
  `ville` varchar(100) NOT NULL,
  `code_postal` varchar(20) NOT NULL,
  `pays` varchar(100) NOT NULL,
  `date_creation` timestamp NOT NULL DEFAULT current_timestamp(),
  `est_actif` tinyint(1) NOT NULL DEFAULT 1
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_general_ci;

-- --------------------------------------------------------

--
-- Structure de la table `categories`
--

CREATE TABLE `categories` (
  `id` int(11) NOT NULL,
  `nom` varchar(100) NOT NULL,
  `description` varchar(255) NOT NULL,
  `est_actif` tinyint(1) NOT NULL DEFAULT 1
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Relations Clés

- **Utilisateurs → Adresses Utilisateurs** : Un utilisateur peut avoir plusieurs adresses.
- **Commandes → Utilisateurs** : Une commande est liée à un utilisateur.
- **Produits → Catégories** : Un produit appartient à une catégorie.

Les tables incluent des contraintes de clés étrangères pour assurer l'intégrité des données ainsi que des index pour optimiser les performances lors des requêtes.

Données Insérées

Quelques exemples de données insérées :

- **Catégories** :

```
INSERT INTO `categories` (`id`, `nom`, `description`, `est_actif`) VALUES
(1, 'Robes', 'Découvrez notre collection de robes qui combine élégance, confort et tendance. Que vous cherchiez une robe fluide pour l\'été, une robe de soiré
(2, 'Hauts', 'Explorez notre sélection de hauts, conçus pour s\'adapter à toutes vos envies et à chaque moment de la journée. Du t-shirt basique au chemisier
(3, 'Pantalons', 'Nos pantalons allient style et fonctionnalité pour vous offrir une allure impeccable. Que vous préfériez les coupes ajustées, les pantalons
(4, 'Chaussures', 'Complétez votre tenue avec notre gamme de chaussures, alliant confort et style pour chaque pas. Des baskets tendances aux escarpins élégant
```

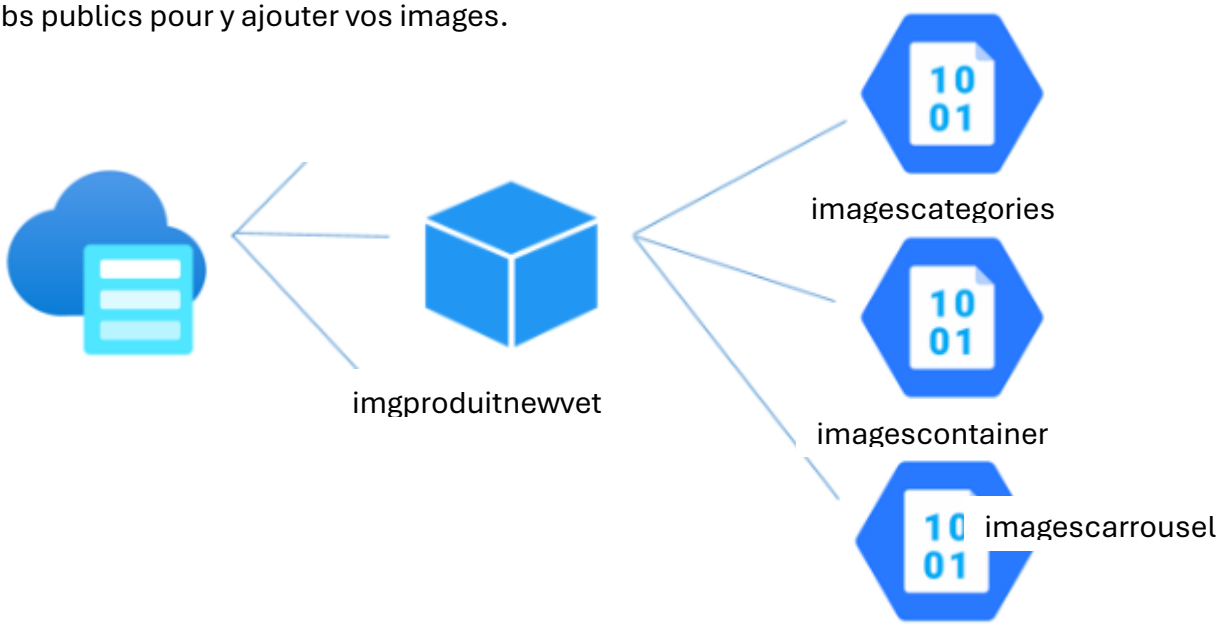
- **Produits** :

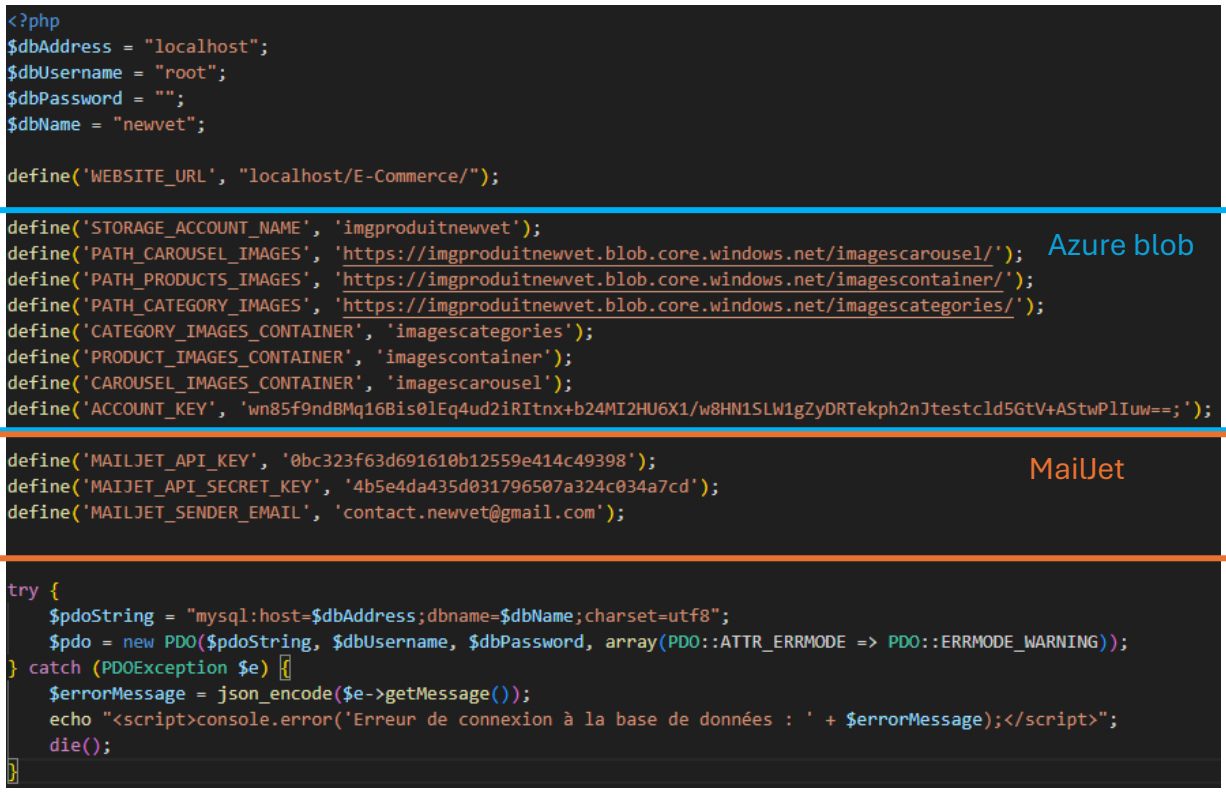
```
INSERT INTO `produits` (`id`, `categorie_id`, `nom`, `description`, `prix`, `stock`, `date_ajout`, `en_priorite`, `est_actif`) VALUES
(1, 1, 'Robe Zébrée', 'Robe zébrée élégante pour toutes les occasion', 40.00, 30, '2024-08-13', 0, 1),
(2, 1, 'Robe Maxi Florale', 'Robe longue avec motif floral, parfaite pour l&#039;été.', 79.99, 0, '2024-08-09', 0, 1),
(3, 2, 'Chemisier à Manches Longues', 'Chemisier féminin à manches longues.', 29.99, 0, '2024-04-24', 0, 1),
(4, 2, 'Débardeur Basique', 'Débardeur confortable pour une tenue décontractée.', 14.99, 0, '2024-02-16', 0, 1),
(5, 3, 'Jean Skinny Stretch', 'Jean stretch et ajusté pour un look moderne.', 49.99, 5, '2024-08-21', 0, 1),
```

Configuration

Dans le projet PHP sur le fichier **config.php**, vous pouvez configurer les informations de connexion à la base de données, l'url, l'accès au container Azure … ou laisser par défaut.

Mais avant cela vous aurez besoin d'accéder à azure pour créer un container avec trois blobs publics pour y ajouter vos images.






```php
<?php
$dbAddress = "localhost";
$dbUsername = "root";
$dbPassword = "";
$dbName = "newvet";

define('WEBSITE_URL', "localhost/E-Commerce/");

define('STORAGE_ACCOUNT_NAME', 'imgproduitnewvet');
define('PATH_CAROUSEL_IMAGES', 'https://imgproduitnewvet.blob.core.windows.net/imagescarousel/');
define('PATH_PRODUCTS_IMAGES', 'https://imgproduitnewvet.blob.core.windows.net/imagescontainer/');
define('PATH_CATEGORY_IMAGES', 'https://imgproduitnewvet.blob.core.windows.net/imagescategories/');
define('CATEGORY_IMAGES_CONTAINER', 'imagescategories');
define('PRODUCT_IMAGES_CONTAINER', 'imagescontainer');
define('CAROUSEL_IMAGES_CONTAINER', 'imagescarousel');
define('ACCOUNT_KEY', 'wn85f9ndBMq16Bis0lEq4ud2iRItnx+b24MI2HU6X1/w8HN1SLW1gZyDRTekph2nJtestcld5GtV+AStwPlIuw==;');

define('MAILJET_API_KEY', '0bc323f63d691610b12559e414c49398');
define('MAIJET_API_SECRET_KEY', '4b5e4da435d031796507a324c034a7cd');
define('MAILJET_SENDER_EMAIL', 'contact.newvet@gmail.com');


try {
    $pdoString = "mysql:host=$dbAddress;dbname=$dbName;charset=utf8";
    $pdo = new PDO($pdoString, $dbUsername, $dbPassword, array(PDO::ATTR_ERRMODE => PDO::ERRMODE_WARNING));
} catch (PDOException $e) {
    $errorMessage = json_encode($e->getMessage());
    echo "<script>console.error('Erreur de connexion à la base de données : ' + $errorMessage);</script>";
    die();
}
```

Pour MailJet il suffit de se créer un compte et de renseigner des données de votre profil. L'API utilise votre API et vos clés secrètes pour l'authentification.

Intégrations Externes

Le projet utilise **Composer**, un gestionnaire de dépendances pour PHP, afin d'intégrer et de gérer automatiquement des bibliothèques externes comme **Azure Blob Storage** et **Mailjet**. Ces bibliothèques sont nécessaires pour le bon fonctionnement de certaines fonctionnalités, comme le stockage de fichiers et l'envoi d'e-mails.

Étapes d'installation des dépendances

Avant de procéder à l'installation des dépendances, assurez-vous d'avoir Composer installer sur votre machine. Si ce n'est pas le cas, vous pouvez le télécharger et l'installer depuis getcomposer.org.

1. Installation des dépendances : Après avoir cloner le projet, exécutez la commande suivante à la racine du projet pour installer toutes les dépendances listées dans le fichier **composer.json** :

composer install

Cette commande va lire le fichier **composer.lock** (si présent) pour installer les versions exactes des bibliothèques. Si **composer.lock** n'existe pas, Composer installera les

dernières versions compatibles des bibliothèques spécifiées dans **composer.json** et créera le fichier **composer.lock**.

2. Mise à jour des dépendances (facultatif) : Si vous souhaitez mettre à jour les bibliothèques à leur dernière version compatible avec les contraintes définies dans **composer.json**, vous pouvez exécuter la commande :

composer update

Cette commande mettra à jour les versions des bibliothèques et régénérera le fichier **composer.lock** avec les nouvelles versions installées.

Exemples de bibliothèques installées

Voici une description des principales intégrations externes utilisées.

1. Guzzle HTTP Client

- **But** : Guzzle est utilisé comme client HTTP pour effectuer des requêtes API externes (par exemple, pour envoyer des demandes via Mailjet ou Azure Blob).
- **Version** : 7.9.2
- **Description** : Guzzle est une bibliothèque HTTP qui permet de simplifier les appels API avec gestion des promesses et PSR-7.
- **Lien de documentation** : Guzzle

2. Mailjet

- **But** : Mailjet est utilisé pour l'envoi d'e-mails transactionnels, comme les confirmations d'inscription ou de commande.
- **Version** : v1.6.3
- **Description** : La bibliothèque PHP de Mailjet permet de gérer facilement les envois d'e-mails via leur API.
- **Lien de documentation** : Mailjet API PHP

3. Azure Blob Storage

- **But** : Utilisé pour stocker les fichiers volumineux tels que les images de produits sur le service de stockage cloud Azure.
- **Version** : 1.5.4
- **Description** : Cette bibliothèque permet d'interagir avec les blobs stockés dans Azure. Elle gère la création, la suppression et la récupération des fichiers stockés.

- **Lien de documentation** : Azure Blob PHP SDK

4. PSR-7 et PSR-18 (HTTP Messages et Clients)

- **But** : PSR-7 et PSR-18 sont des standards utilisés pour gérer les requêtes et réponses HTTP de manière interopérable. Ils sont utilisés par Guzzle et d'autres composants.
- **Version** : 2.0 (PSR-7), 1.0.3 (PSR-18)
- **Description** : Ils définissent des interfaces pour les messages HTTP (PSR-7) et pour les clients HTTP (PSR-18), ce qui permet de standardiser les interactions HTTP dans l'application.

Lien de documentation :

- PSR-7
- PSR-18

5. Symfony Components

Plusieurs composants Symfony sont utilisés dans ce projet pour améliorer la compatibilité avec les normes PHP, notamment :

- **symfony/polyfill-ctype** et **symfony/polyfill-mbstring** : Fournissent des alternatives pour des fonctions PHP manquantes sur certaines configurations d'hébergement.
- **symfony/validator** : Utilisé pour la validation des entrées de formulaire (par exemple, pour vérifier les adresses e-mail ou les informations de paiement).

Version :

- Symfony Validator : v7.1.3
- Symfony Polyfill : v1.30.0

Sécurité

1. Authentification : Gestion des sessions utilisateurs avec chiffrement des mots de passe.
2. Protection des données : Conformité RGPD, chiffrement des données sensibles.
3. Paiement sécurisé : Intégration de passerelles de paiement certifiées PCI DSS pour protéger les informations de paiement.
4. Accès au back-office : Sécurisé par un système d'accès basé sur des rôles.
5. Aucun produit, catégories, commandes ... ne sont totalement supprimé

Administration

En dehors du système de gestion de base de données, depuis le Back-office accessible sur la page du profil newvet, il est possible pour les comptes admins d'accéder au CRUD des articles, catégories ... et d'administrer le site.

Dans le script de la base de données un premier Administrateur est créé.

Email : admin@newvet.fr

Mot de passe (a modifier): Admin123*