Examen Back

Objectifs

- Être capable de créer un projet avec un framework PHP (Laravel ou Symfony).
- Versionner son code avec Git.
- Savoir manipuler les routes et les contrôleurs.
- Savoir créer et exploiter les modèles ou entités.
- Savoir créer des formulaires et gérer les données utilisateurs avec la validation.
- Maîtriser les migrations.
- Être à l'aise avec un moteur de template PHP (Blade ou Twig).

Modalités

Ce projet est à réaliser sur les journées du 27/09/22 au 29/09/22. Vous devrez versionner votre projet avec git et le rendre disponible sur Github pour le formateur. Vous ferez un maximum de commits avec chaque étape de vos travaux. Idéalement, vous pouvez travailler avec les Pull Requests sur Github mais ce n'est pas une obligation (Création de branches et *merge* dans la branche *master*).

Réalisation

Nous souhaitons créer un mini site e-Commerce. Vous avez quelques maquettes à disposition déjà intégrées en HTML avec Bootstrap 5. Vous êtes libre d'intégrer vous-même les maquettes si vous souhaitez concevoir votre propre intégration. Vous pouvez également utiliser Tailwind, Bootstrap 5… Mais vous serez obligé de faire le HTML / CSS de zéro.

Bien entendu, vous devrez créer un nouveau projet Laravel / Symfony pour réaliser tout cela.

Pages principales

Vous créerez les routes suivantes ainsi que les contrôleurs et méthodes associées :

- /
- /produits
- /produits/12-iphone-xs
- /categorie/12-smartphone
- /contact

Il est évident que chaque contrôleur devra renvoyer une réponse correcte ainsi que la vue associée. Commencez d'abord par intégrer tranquillement l'ensemble des pages avec le HTML / CSS sans vous embêter avec la base de données ou autre. Attention de bien concevoir votre structure avec un layout et une ou plusieurs sections / blocs.

BONUS Configuration

Vous remarquerez que le titre du projet est « Ecommerce » un peu partout sur le site. Vous devez modifier cette valeur par le nom de votre boutique et surtout stocker cette valeur dans la variable d'environnement APP_NAME. Vous utiliserez ensuite la fonction config pour afficher cette valeur là où elle doit être affichée. https://laravel.com/docs/9.x/configuration

Il faudrait également que les informations de contact (Footer et page de contact) soient stockées dans la configuration, par exemple dans le fichier config/services.php (tableau) https://linuxhint.com/how-to-create-config-file-in-laravel/

L'avantage de faire cela est que l'on pourra modifier facilement les informations de société (Adresse, téléphone) pour notre client.

On pourra faire la même chose avec Symfony.
https://symfony.com/doc/current/templating/global_variables.html

Base de données

Nous aurons besoin d'une table pour les produits, les catégories et les couleurs. Il faudra également un modèle *Product,* un modèle *Category* et un modèle *Color*. Ce modèle Product contiendra les attributs suivants :

- ID
- Nom (iPhone X)
- Description
- Prix
- Slug (iphone-x)
- Date de création
- Coup de cœur (permet de savoir si le produit est un coup de cœur ou non)
- Une liste de couleurs (à stocker grâce à une relation ?)
- Image (On devra pouvoir uploader une image)
- Promotion (entier contenant une promotion en pourcentage à calculer avec le prix)
- Un champ category_id pour lier une catégorie au produit

Le modèle Category contiendra seulement un nom et un slug. Le modèle Color contiendra un nom et un code hexadécimal.

Vous vous aiderez d'une factory et d'un seeder pour remplir la base de données une fois les modèles créés : https://laravel.com/docs/9.x/seeding

Avec Symfony, vous ferez plutôt des fixtures.
https://symfony.com/bundles/DoctrineFixturesBundle/current/index.html

Vous êtes libre dans la création de produits, de catégories et de couleurs.

Fonctionnalités Front Office

- La page *d'accueil* affichera 3 produits aléatoires de la base de données dans un carrousel. Elle affichera également un produit coup de cœur aléatoire de la BDD. On affichera aussi les derniers produits créés ainsi que les meilleurs produits. Les meilleurs produits seront ceux avec les meilleures notes donc vous devez attendre de faire le modèle Review avant de faire cela.
- La page « *produits* » affichera tous les produits dans la base de données. Une pagination sera présente, on affiche 6 produits par page. Une *sidebar* sur la gauche permettra de filtrer les produits par couleurs (On restera bien sur la même page, c'est un formulaire). La liste de catégories affichera des liens vers chaque catégorie de la boutique, vous devez utiliser le modèle Category. On affichera le dernier produit créé dans la BDD. On pourra afficher les couleurs avec le code hexadécimal sur chaque fiche produit ou dans les filtres.
- La page *produits/12-iphone-xs* affichera le produit iPhone XS de la base de données. Pensez à afficher toutes les données : Image, prix calculé avec la promotion, liste des couleurs dans la base de données avec le nom et code hexadécimal. La description du produit sera du markdown transformé en HTML. Regardez https://laravel.com/docs/9.x/helpers#method-str-markdown pour avoir un indice. On pourrait stocker le markdown dans la base de données et le transformer avec Blade sur la page, cela permettra au client d'afficher son texte en gras etc… Avec Symfony, ce sera plutôt https://twig.symfony.com/doc/3.x/filters/markdown_to_html.html
- La page *categorie/12-smartphone* affichera également la liste des produits mais seulement de la catégorie concernée.

Sur la liste des produits, la description affichée sera la description tronquée :
https://laravel.com/docs/9.x/helpers#method-fluent-str-limit

BONUS Email

- La page contact est un simple formulaire de contact où l'utilisateur saisit les informations. On les valide et on envoie un mail à l'administrateur du site. On devra valider l'email, le nom (3 caractères minimum) et le message (15 caractères minimum). On affichera les erreurs proprement.

Pour pouvoir envoyer des emails, il faudra installer et configurer Maildev :
https://medium.com/@dao.houssene/maildev-tester-lenvoie-de-vos-mails-avec-maildev-cce3115594fd

Il faudra ensuite créer un email : https://laravel.com/docs/9.x/mail#writing-mailables
Et bien sûr, il faudra envoyer cet email : https://laravel.com/docs/9.x/mail#sending-mail
Pour Symfony : https://symfony.com/doc/current/mailer.html

Fonctionnalités Back Office

Certaines routes ne sont accessibles que par un administrateur. Vous allez déjà vous occuper de créer les routes nécessaires.

- /admin
- /admin/produits
- /admin/produits/nouveau
- /admin/produits/1/modifier
- /admin/produits/1/supprimer (uniquement en DELETE)

Il faudra que les vues de ces routes héritent d'un nouveau layout : admin.blade.php par exemple ou admin.twig.html. Vous allez donc pouvoir créer un gabarit de page totalement différent, vous avez la maquette du back office qui est fournie.

- La page */admin/produits/creer* affichera un formulaire permettant d'ajouter un produit dans la base de données. Il faudra valider le formulaire en affichant les erreurs s'il y en a. Le nom du produit devra contenir au moins 3 caractères et la description au moins 10 caractères. Le prix du produit devrait être un entier numérique compris entre 99 et 1000. Le slug du produit sera généré automatiquement à partir du nom du produit. Vous pouvez utiliser le helper https://laravel.com/docs/9.x/helpers#method-str-slug de Laravel ou le slugger de Symfony. On pourra choisir la liste des couleurs associées au produit avec des checkboxes et la catégorie avec un select ou un radio. Vous n'oublierez pas de gérer l'upload de l'image.
- La page */admin/produits/1/modifier* permettra de modifier le produit dont l'*id* est située dans l'URL. Vous utiliserez le même formulaire.
- La page */admin/produits/1/supprimer* permettra de supprimer un produit. Cette page ne sera accessible que par la méthode DELETE de la requête HTTP. On devra donc passer par un formulaire pour supprimer le produit et pas par un simple lien.

Si vous avez le temps, vous ferez la même chose pour les catégories.

BONUS Sécurité

Vous allez implémenter la partie sécurité avec l'inscription d'un utilisateur et le login. A la connexion, l'utilisateur verra son prénom apparaître dans le menu et pourquoi pas son avatar ? Vous pouvez utiliser Laravel UI pour vous aider à faire cela. Attention, il faudrait bien combiner votre layout avec celui de Laravel UI. Les maquettes de login et d'inscription n'ont pas été fournies, vous devrez donc adapter le code de Laravel UI avec le thème ou créer tout cela de zéro. Il nous faudra également un système de mot de passe oublié.

On utilisera le middleware auth pour vérifier que l'utilisateur est bien connecté afin d'accéder à l'administration : https://laravel.com/docs/9.x/authentication#protecting-routes

On pourrait également s'assurer que l'utilisateur est bien un administrateur.

Conseils

Pour la partie graphique, vous êtes libre de modifier les maquettes. La liste des fonctionnalités n'est pas exhaustive et vous êtes libre d'ajouter ou modifier des fonctionnalités en fonction de votre imagination.

Le but n'est pas de terminer le TP au plus vite mais bien de manipuler Laravel / Symfony et d'être à l'aise avec le framework. Chacun avancera comme il le peut sur chaque partie.

BONUS Commentaires

Le modèle Review contiendra le nom d'un utilisateur, la date de création, la note, une description et bien sûr elle sera liée à l'entité Product. A partir de la fiche produit, on pourra ajouter une Review. L'administrateur du site pourra gérer les Review à partir du back office. A vous de voir comment gérer cela avec les routes etc… Vous pouvez vous aider de ce TP : https://github.com/dwwm-bethune/php/tree/main/13-tp-avis

BONUS Panier

Vous implémenterez un système de panier en utilisant les sessions. Le panier devra donc être conservé durant la visite d'un utilisateur et celui-ci pourra ajouter / modifier ou supprimer un produit du panier. L'ajout au panier se fera via le bouton « Ajouter au panier », on aura besoin d'une route afin de pouvoir ajouter le produit en question dans la session. À partir de la page du panier, on pourra supprimer les produits du panier.

- /panier **=** Afficher le panier
- /panier/{product}/{quantity}/{couleur} **=** Ajouter un produit dans la session avec une quantité et la couleur
- /panier/{product} **=** Supprimer un produit du panier

https://laravel.com/docs/9.x/session#interacting-with-the-session
https://symfony.com/doc/current/session.html

BONUS Recherche

Vous allez implémenter la possibilité de rechercher un produit. Cela sera un formulaire redirigeant vers la liste des produits avec un paramètre dans l'URL. On utilisera ce paramètre pour réaliser un where et un like sur le modèle Product.

Bon courage.