Master MIASHS 2023

-

Programmation web côté serveur

-

Projet de création d'une application web avec le framework PHP Laravel

à rendre avant le 07 / 04 / 2023 à 18h00

Sommaire

Conseils

- Lisez bien le sujet.
- Utilisez Trello ou un autre outil de gestion de projet pour faire une liste des choses à faire avec votre binôme et ne pas vous éparpiller.
- Terminez bien le TP2, puis choisissez des fonctionnalités supplémentaires. Vous pouvez les faire ensemble avec votre binôme ou les répartir entre vous, puis les fusionner avec Git.
- Allez sur une machine de l'université ou une autre machine Linux et suivez votre guide d'installation pour vérifier que tout fonctionne bien.
- Vous avez le droit de vous entraider, mais attention pas de copie du code. Le code de vos projets sera analysé et un plagiat détecté sera sévèrement sanctionné. Si vous utilisez du code qui n'est pas le vôtre préciser le. Citez vos sources.

L'idée est que tout le monde sache faire une application web basique avec Laravel… Tout est dans le cours et la doc. Vous pouvez également trouver énormément de tutoriels sur internet.

Pour avoir 8 / 20

TP2 Complété

- Le sujet du TP2
- avec SQLite, pas de MySQL!!
- sans les exercices supplémentaires, mais avec du code commenté
- Vous n'êtes pas obligés d'utiliser le schéma de base de données fourni dans le TP, vous pouvez en définir un nouveau si vous le souhaitez.

C'est ok si je dois créer le fichier Sqlite de la base de données, modifier le fichier .env et lancer les migration avec du Seeding.

Cependant, si vous avez des données spécifiques que vous souhaitez présenter dans votre application, vous pouvez joindre la base de données à votre projet.

Github

Effectuer des commits pendant le développement de votre projet. Un Git actif et des commits clairs seront appréciés. Je veux pouvoir voir votre progression.
Je ne veux donc pas seulement un seul commit le dernier jour!

Rédaction d'un fichier texte README.md sur Github contenant:

Le fichier README.md, placé à la racine d'un dépôt Github, sera affiché par défaut sur la page du dépôt. Il fera office de votre rapport de projet.

Il doit contenir:

- Le guide d'installation de votre projet (avec toutes les commandes à exécuter et les fichiers à configurer)
- Les parties que vous avez implémentées, et pour chaque partie de brèves instructions de ce que je dois tester (URL des pages à visiter, boutons à cliquer, commandes à lancer dans un terminal, etc)
  Pour ce compte rendu, veuillez bien préciser les fonctionnalités que vous avez implémentées (ex: 4 - Ajout de rôles utilisateurs - Puis détails de ce que je dois faire pour tester ou vérifier)
- Optionnellement des remarques (Difficultés rencontrées, etc)
- Soignez la mise en forme et l'orthographe (Passez votre fichier dans un correcteur orthographique)

Rendu

Votre repository git sera votre rendu. Vérifiez bien que le nom de votre «team» est bien votre nom et celui de votre binôme. Le dernier commit doit dater d'avant le 07 / 04 / 2023 à 18h00

Ce projet est à réaliser en binôme.
Sauf exception validée avec moi, des points seront retirés si vous n'êtes pas en binôme.

Pour que le projet soit recevable, je dois aller sur votre Github puis suivre votre guide d'installation (git clone ... , composer install, npm install, modification du fichier .env, lancement des migrations, lancement du ou des serveurs, etc.).

Si le projet ne fonctionne pas, je vous fais un retour. S'il ne fonctionne toujours pas au second retour, alors votre note sera mauvaise.
N'hésitez pas à tester votre installation sur différentes machines (Je travaille sur Linux, donc assurez-vous que ça fonctionne sur Linux). En général quand ça fonctionne sous Linux, cela fonctionne sur les autres systèmes, l'inverse n'est pas vrai.

À ce stade vous avez entre 8 et 10 sur 20, selon la qualité de votre travail, de votre dépôt Git et de votre compte rendu. Si vous voulez plus, passez à la suite.

Pour avoir avoir des points supplémentaires

Les exemples de fonctionnalités qui suivent sont notés avec une indication (+)
Cela donne une idée de la difficulté mais également une idée du barème. Les fonctionnalités avec un + valent environ 2 points. C'est indicatif. En effet, cela dépend de l'implémentation. Certaines fonctionnalités peuvent être plus ou moins développées. Par exemple pour les fonctionnalités 5 ou 8 le niveau d'implémentation peut beaucoup varier. En résumé, cela dépend du travail fourni et du résultat.
Vous pouvez suivre des tutoriels trouvés sur internet mais vous devez me le préciser.
La cohérence et le travail d'ensemble est également apprécié. Je préfère peu de fonctionnalités, claires, bien exécutées, et qui fonctionnent plutôt qu'une multitude de fonctionnalités qui ne fonctionnent pas.
La clarté du code (indentation, commentaires, etc) est aussi prise en compte.

1 - Gestion des commentaires (+)

Formulaire de commentaires sous les restaurants permettant de soumettre un commentaire.
La liste de commentaires est ensuite affichée sous le formulaire.

Pour aller plus loin:

- Un captcha
- Soumission du formulaire en Ajax
- Gestion des commentaires par un administrateur (ex: Un commentaire doit être approuvé avant d'être publié, notification par email lorsqu'un commentaire est publié, possibilité de supprimer un commentaire)

2 - Gestion des tags ou du type de cuisine (+)

Les tags sont pour l'instant géré dans une colonne de tags sous forme de texte. Il serait intéressant d'avoir une vraie gestion des tags dans une table séparée avec une liaison. Cela permettra par exemple de pouvoir faire une recherche de restaurant par tags, par type de cuisine ou de pouvoir afficher des informations sur le type de cuisine.

Pour aller plus loin:

- Recherche par tags
- Affichage des informations du type de cuisine (histoire, lieu géographique, etc)

3 - Notes (+)

Possibilité de noter les restaurants (système d'étoile par exemple) avec affichage de la moyenne des notes. Possibilité de trier les restaurants par notes.

4 - CRUD des restaurants améliorés (++)

- CRUD fonctionnant en Ajax ou avec Vue.js ou avec React

5 - Identification / Authentification qui protège l'accès à l'administration (+)

https://laravel.com/docs/master/authentication
Authentification et protection de certaines parties de l'application. Si vous implémentez cette partie, vous devez générer des utilisateurs et me fournir des identifiants dans votre rapport pour que je puisse tester. Utilisation d'un starterkit breeze ou jetstream. (https://laravel.com/docs/master/starter-kits)
Aller plus loin:

- Ajout de 2 rôles ou plus, un *admin* et un *user*.
- Selon le rôles les utilisateurs n'ont pas accès au même fonctionnalité.
- Un utilisateur qui s'enregistre aura le rôle *user* par défaut.
- Seul l'administrateur peut changer le rôle d'un utilisateur.
- Le rôle *user* peut uniquement gérer ses restaurants, les commentaires de ses restaurants (si implémentés) et son profil.
- Le rôle *admin* peut accéder à toutes les parties de l'administration
- Cela signifie que vous devez créer un CRUD pour les utilisateurs
- Ajouter un CRUD pour la gestion des rôle pour pouvoir ajouter de nouveaux rôles
- Une partie de ces fonctionnalités peuvent être prises en charge par jetstream

6 - Mise en place et utilisation de Laravel Jetstream (+++)

https://jetstream.laravel.com/3.x/introduction.html

Utilisation de Tailwing CSS et Livewire ou Inertia

7 - Ajout de fichiers média pour les restaurants (+)

Pouvoir ajouter (uploader) des images à un restaurants et qu'elle s'affiche ensuite sur le frontend.
Une seule image c'est la base, mais des images affichées avec un slider c'est mieux.
Dans ce cas, un multi-upload serait encore mieux.

8 - Identification en utilisant Socialite (+)

Tout est expliqué ici: https://laravel.com/docs/master/socialite

Pour aller plus loin, ajouter l'identification avec :

- Facebook
- Google
- Github
- Twitter
- LinkedIn
- GitLab
- Bitbucket

9 - Intégration graphique responsive en utilisant Less, Sass ou Stylus et Laravel Mix (++)

Reproduisez les styles de la template utilisée dans https://foundation.zurb.com/templates.html mais convertissez le HTML/CSS/JS pour utiliser SASS.

Vous pouvez utiliser un framework comme Bootstrap ou Foundation, cependant ce sont les versions SASS qui doivent être utilisées.
Il ne faut plus voir les noms des classes CSS du framework de base dans votre HTML.

10 - Utilisation du framework Javascript VueJS, react ou angular (+++)

Réalisez l'application entièrement avec des composants Vue.js / react / angular

11 - Tests unitaires (+)

Créer des tests pour l'ensemble de votre application.
Il faut des tests "Unit", "Feature" et "Browser" en utilisant Dusk et Mockery.
Il faut lister les tests dans le compte rendu et me mettre des commandes si vous voulez que je lance des groupes spécifiques.

Si vous rencontrez des erreurs bizarres avec Dusk, installez et utilisez https://github.com/staudenmeir/dusk-updater pour mettre ChromeDriver à jour.

Dans le code de vos tests, ajoutez des commentaires pour que je puisse comprendre ce que vous faites. Dans le compte rendu, préciser ce que vous avez testé.

12 - Une application de gestion de fichiers avec catégorisation, VueJS, multi-upload, authentification (++++)

Une application complète de gestion de fichiers multimédia.

13 - Un Webchat avec Pusher et Vue.js (+++)

Ajout d'un chat à votre application.
Voici des tutoriels que vous pouvez suivre:
https://pusher.com/tutorials/how-to-build-a-chat-app-with-vue-js-and-laravel/
https://appdividend.com/2018/05/19/laravel-vue-chat-application-tutorial/

Attention à bien les intégrer dans votre application, pas de copie de leur projet dans le vôtre…

14 - Surprise!

Surprenez moi avec des fonctionnalités non listées ici.
À vos risques et périls, mais si vous faites quelque chose de puissant, ça peut payer.
N'oubliez pas de bien le préciser dans vos comptes rendus et de me donner des détails.
Dans le doute, vous pouvez me soumettre vos idées avant de commencer.