Modal INF471W

Site de la course au drapeau

Table des matières

Informations préliminaires

Je n'ai pas pu créer les utilisateurs demandés car mon site prend comme identifiant une adresse mail et non un identifiant. De plus, je protège contre les mots de passe faibles. Voici les deux compte « extérieur » créés :

alice@alice.fr

mdp : Lapin?2021

bob@bob.fr

mdp : Renard?2021

Projet

Le but de ce projet est de créer un site de référence pour la course au drapeau, une course organisée entre Bordeaux et l'X pour symboliser un événement historique (plus d'informations à ce propos dans la section A propos du site web).

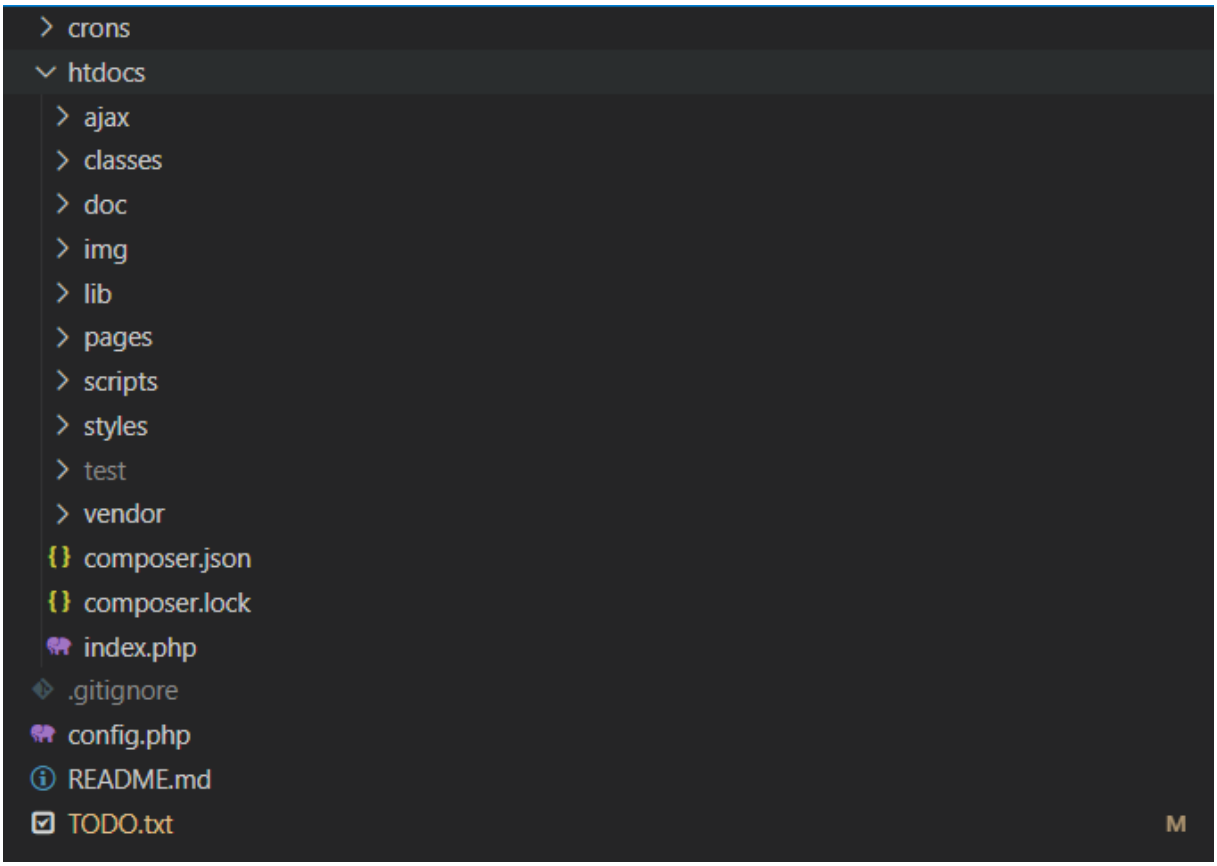
Il est dédié aux participants pour qu'ils trouvent les informations nécessaires et aux organisateurs pour centraliser les inscriptions.

Ce projet a été pensé pour pouvoir être ouvert aux extérieurs à tout moment. Néanmoins, il ne le sera probablement jamais vis-à-vis de sa symbolique.

Le site web est en ligne et disponible sur le réseau eduroam, https://course-drapeau.binets.fr/

Il est n'est pas fini mais a atteint un état convaincant (et heureusement car la course est dans deux semaines).

Organisation des fichiers



Configuration serveur

J'ai rajouté dans mon projet un container htdocs afin de me rapprocher autant que possible de la configuration serveur afin de faciliter le processus CI/CD. Le seul fichier qui diffère est le fichier config.php qui contient les credentials de la base de données du serveur. Il me suffit d'un pull depuis git pour intégrer de nouvelles modifications.

J'ai également défini un cronjob dont l'objectif est de vider le fichier htdocs/tmp (que l'on ne voit pas ci-dessus).

Librairies externes

Les librairies sont de deux types. Les librairies importés de manière traditionnelle dans le dossier lib. On trouve :

Bootstrap (éléments de style et de frontend js)

Jquery (éléments de js pour simplifier la syntaxe)

Leaflet : librairie open source pour afficher des cartes. Le donnée de cartes proviennent d'openstreetmap. Cette API est très simple d'utilisation et bien documentée.

Leaflet-gpx : libraire qui s'appuie sur leaflet pour afficher des traces gps et visualiser des parcours. Également très simple d'utilisation.

SimpleExcel : librairie pour importer des données depuis un tableur excel. Je voulais l'utiliser pour importer facilement des données vers la base de données en ligne. Néanmoins la librairie n'était pas correctement maintenue et permet d'ouvrir uniquement les CSV et pas les XLS.

Il y a également les librairies importées de manière plus efficace avec composer. On trouve ces librairies sur packagist et on n'a pas besoin de se soucier des dépendances. Les librairies que j'utilise sont spécifiées dans composer.json et sont ensuite téléchargés et compilés dans le dossier vendor.

Composer permet aussi d'autoload les classes php en utilisant les namespace mais cette fonctionnalité n'est pas utilisée.

On trouve :

phpoffice/phpspreadsheet : librairie pour ouvrir toutes les extensions excel, CSV et xls

apereo/phpcas : utiliser le CAS mis en place par la dsi pour l'authentification des X

GIT

Pour travailler efficacement et garder les différentes versions du projet, j'ai utilisé github. Le projet est en public à l'adresse https://github.com/pe712/Site-web-php-course-Bordeaux-X.

Le fichier .gitignore spécifie les fichiers non suivis par git.

Ressources

Les dossiers img et doc contiennent respectivement des images et des documents (essentiellement excel) utilisés par le site.

Styles

Le dossier styles contient le fichier de style css. J'ai choisi de faire un unique fichier de style commun à toutes les pages. Cela a des avantages : gain de temps car c'est toujours le même fichier et facilité pour partager des styles entre les pages. Cela a aussi des inconvénients : le fichier est très long et illisible (ce qui n'est pas gênant en production car on ne modifie que des petites parties de ce fichier que l'on peut rechercher).

Par conséquent il est nécessaire de bien organiser ce code. Les class et id sont regroupés par pages proprement séparés par un commentaire.

Le nom des classes et id sont choisis pour être facilement identifiables (sauf les premiers créés). J'ai choisi une convention de la forme [page]-[type d'élément html]-[localisation (si besoin)]

Ajax

Ajax a été introduit tard dans le projet donc peu de fonctionnalités sont asynchrones. L'avantage de ne pas utiliser les fonctions asynchrones est de coder majoritairement côté serveur et ne pas avoir à alterner du code serveur et du code client.

Toutes les requêtes asynchrones envoyés par les méthodes ajax en utilisant jquery (donc XMLHttpRequest) sont traitées sur une unique page en passant en paramètre GET la fonctionnalité cherchée. La page vérifie les autorisations du client.

Scripts

Scripts contient deux fichiers : map.js dédié spécifiquement à la création des cartes et à l'affichages des traces GPX dessus.

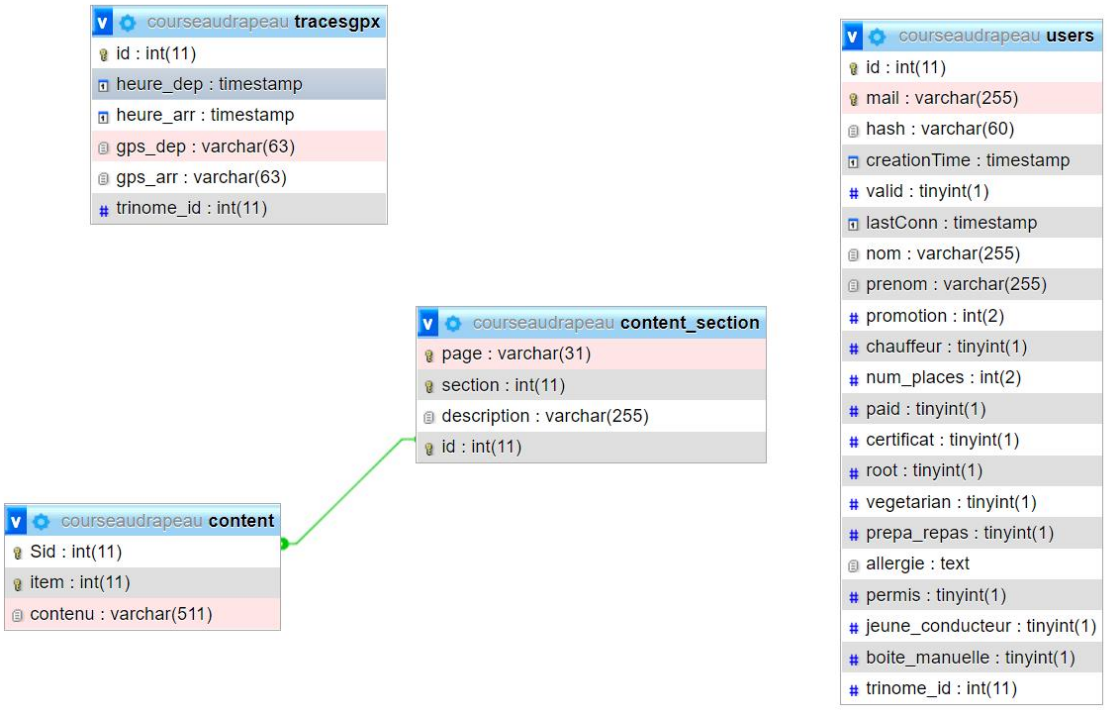
script.js contient tous les autres scripts js. Les fonctions sont regroupés à nouveau par page (ou certaines fonctionnalités particulières demandant un peu plus de code) séparées par des commentaires. La plupart des fonctions codés s'appuie sur le jqclick de jquery et consistent juste à afficher ou masquer des éléments html.

Classes, pages et index.php

Classes regroupent les classes qui servent à gérer tout le backend et pages regroupe la partie php plutôt orienté production du html pour le client.

Je reviendrai plus en détail sur la structuration du site en php dans la section suivante.

Base de données :



Données serveur :

Les données rentrées par l'administrateur du serveur sont stockées sur les tables tracesgpx, content et content_section.

Tracesgpx : Cette table contient les données permettant de visualiser les parcours et de transmettre les données sur la course pour le client. On retrouve heure de départ et d'arrivée, points GPS correspondants et numéro du trinôme qui court.

Contenu dynamique :

Comme vous l'aviez conseillé j'ai initialement voulu que le contenu puisse être entièrement stocké dans la base de données afin de pouvoir le modifier sans toucher aux fichiers php. Cette feature est fonctionnelle

et m'a demandé beaucoup de temps mais finalement pas forcément pertinente. En effet, le but était qu'une autre édition puisse entièrement changer le contenu sans jamais avoir à rentrer dans du code php ou même html. Or cela limite grandement la capacité de modification du site web d'une année sur l'autre. Il y a toujours des nouveautés qui demandent une fonctionnalité supplémentaire ça ou là et qui obligent l'utilisateur à rentrer dans le code php. En effet l'utilisateur ne peut même pas changer le style du texte qu'il rentre. Je ne peux pas non plus permettre à l'administrateur (même s'il est administrateur) de rentrer du texte html. Cela présenterait une faille de sécurité trop grande. Je me suis débrouillé pour qu'il puisse insérer des liens avec une syntaxe particulière mais là aussi l'intérêt est limité et de plus cela présente toujours une faille de sécurité car il faudrait vérifier que le lien n'est pas externe, si tenté que l'administrateur ne cherche pas justement à créer un lien externe… Finalement cette fonctionnalité est peu convaincante et je vais la retirer du site à terme.

Pour en revenir à la base de données, pour chaque page les éléments de contenu sont regroupés en unités de sens que j'ai appelé section et qui possèdent une description. A chaque section est associés plusieurs items qui ont un contenu.

Données utilisateur

La table **users** contient toutes les données sur l'utilisateur, elle est remplie au fur et à mesure du remplissage de son espace personnel. Ainsi la plupart des champs ont une valeur par défaut. La clé primaire d'un compte est son adresse mail.

Pour les X, le champ **hash** est inutile, les champs **nom**, **prénom** et **promo** sont remplis automatiquement à l'aide des données du CAS.

CreationTime et **LastConn** permettent d'avoir un suivi sur la date d'inscription et sur les connexions des clients.

Le champ **valid** permet de savoir si un compte est valide. Cette fonctionnalité n'a pas encore été mise au point. Elle ne concerne que les extérieurs et consiste en une vérification du mail. Il n'y a pas non plus de possibilité de réinitialiser le mot de passe pour eux pour l'instant.

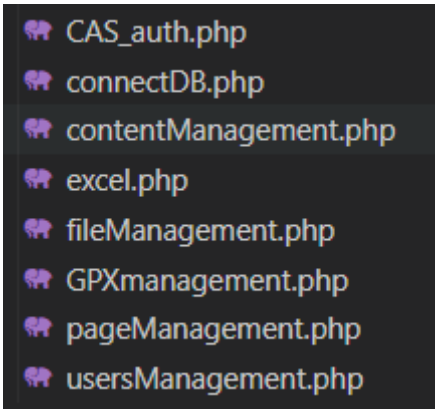
chauffeur, num_places, paid, certificat, vegetarian, prepa_repas, allergie, permis, jeune_conducteur, boite_manuelle : sont les champs que l'utilisateur remplit pour permettre l'organisation de la course.

Le champ **root** permet de spécifier les utilisateurs ayant les droits root.

Le champ **trinome_id** permet de renseigner le numéro de trinôme pour l'attribution des tronçons aux coureurs. Ce champ est renseigné manuellement car la constitution des trinômes est soumise à beaucoup de contraintes.

Structure php

Classes

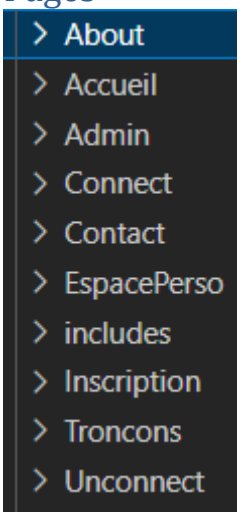


Je pense que les noms sont plutôt explicites. Je ne rentrerai pas dans le détail des classes sauf celles de pageManagement.php

C'est ce script qui détermine l'organisation de tous les fichiers du dossier pages. En effet, le fichier index.php va appeler une page à charger sous la forme de l'instanciation d'un objet d'une classe fille de la classe Page. Cette dernière va faire un prétraitement et en fonction décider d'appeler une autre page à charger récursivement ou alors afficher un contenu. Le contenu est situé alors dans le fichier [pagename]content.php et il est chargé avec un buffer de la sortie.

Cette structuration des pages permet de bien dissocier le traitement php dédié à décider du contenu à afficher et du traitement php dédié à l'affichage (ou a des actions de base de données). Cela m'a demandé du temps de bien structurer le site de cette manière. Il aurait été beaucoup plus simple d'utiliser dès le début un framework (comme symfony) qui fait exactement cela de manière beaucoup plus propre.

Pages



Le dossier pages regroupe en dossier toutes les pages. Ce sont tout les pagename qu'il est possible de passer dans l'url en méthode GET.

Le dossier includes regroupe en fichier distincts des éléments à include (notamment les éléments présents sur toutes les pages qui sont insérés dans index.php).

Je mets ici une brève description de chaque fichier :

About	Page à propos, c'est une FAQ dédié à fournir toutes les informations aux participants

Admin	C'est la page d'administration, elle n'est pas finie
Connect	Formulaire de connexion
Contact	Information sur le binet
EspacePerso	C'est de loin la page la plus grosse. Je l'ai scindé en sous fichiers dans le dossier cards pour bien s'y retrouver. C'est ici que l'on retrouve les certificats médicaux upload (mais ce n'est pas ici qu'on les download, c'est dans /tmp)
Inscription	Formulaire d'inscription pour les extérieurs
Tronçons	Données sur le parcours de la course
Unconnect	Page de déconnexion (redirection uniquement)

Annexes

Visuel du site web