Rapport de projet web

Alexandre Garand, Lucas Bourel, Corentin Guillerme, Maxime Song

14 mai 2019

Table des matières

1 Présentation du projet

Notre projet est un site de réservation de trajets entre plusieurs villes (type uber). Mais au lieu de réserver un trajet en voiture, les utilisateurs réservent des trajets sur des créatures fantastiques (licornes, dragons, phénix ...).

Les utilisateurs peuvent donc s'inscrire soit pour réserver un trajet soit pour proposer un trajet en tant que monture.

Par la suite les utilisateurs qui cherchent un trajet seront appelés des clients et ceux qui proposent de faire des trajets seront appelés des vendeurs, transporteurs, créatures ou montures.

2 L'architecture

Nous avons choisi d'utiliser une structure modèle-vue-contrôleur.

Nous avons un fichier index.php qui est un routeur dirigeant vers toutes les pages du sites et qui est le seul point d'accès de ces pages.

Ensuite nous avons un fichier template.php qui donne la structure de toutes les pages du site contenant notamment l'en-tête et le pied de la page.

Ensuite pour chaque page nous avons un fichier de vue contenant le contenu spécifique de la page.

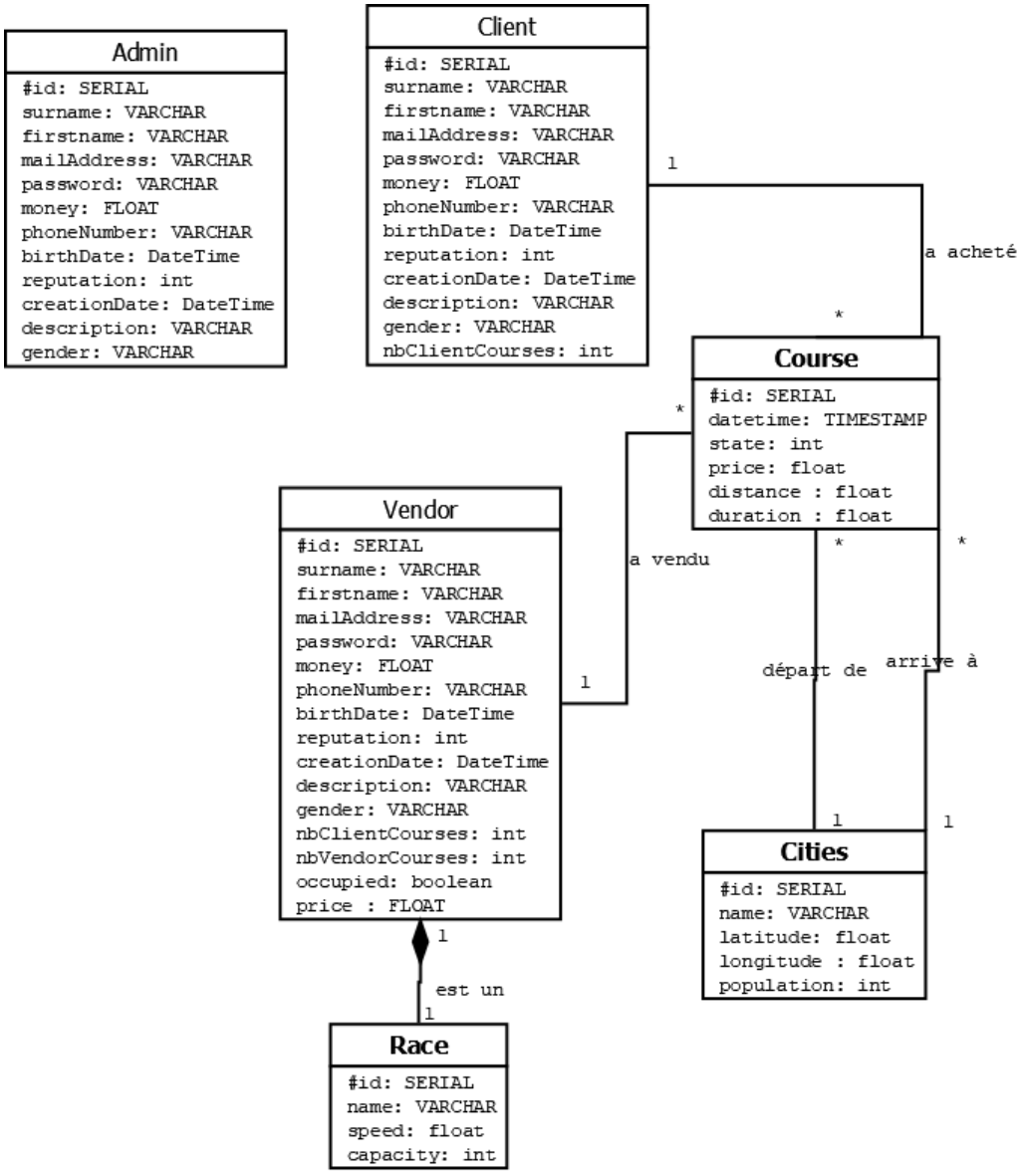
Pour chaque page non statique nous avons un fichier contrôleur qui décide quoi afficher pour ces pages en fonction du contexte.

Pour chaque table de la base de données nous avons un fichier avec une classe représentant la table avec les mêmes attributs, des accesseur et mutateurs pour modifier la classe et un fichier manager manipulant la base de donnée et la classe php pour permettre d'accéder et de traiter les données.

Le principe du site web est donc d'avoir index.php qui appelle le contrôleur de la page web à afficher, le contrôleur peut utiliser les classes du modèle pour manipuler la base de données puis il envoie la vue de la page à afficher à template.php qui affiche la page.

3 Le modèle

3.1 Les bases de données



3.1.1 Admin/Client/Vendor

Ces trois tables représentent les utilisateurs selon leur rôle et contiennent toutes les informations afin de permettre l'utilisation du site ainsi que des informations classiques sur les utilisateurs.

3.1.2 Race

Cette table contient le nom et les caractéristique de la race afin de savoir comment les clients serons servis par la race en question.

3.1.3 Course

Celle-ci contient les informations sur les transports effectués afin de permettre aux utilisateurs d'accéder à l'historique de leurs transactions ; un attribut est consacré à l'état de la course, car les courses peuvent être : disponibles, en cours de réservation, reservées, effectuées, ou annulées/refusées.

3.1.4 Cities

Celle-ci contient les informations sur les villes : les coordonnées gps pour pouvoir les placer et la population pour pouvoir les trier dans la liste des propositions pour l'auto-complétion.

3.1.5 Comment

Option de faire de donner un avis sur les Clients et Vendors, un prototype de modèle a été faite, mais cette partie n'a pas été intégrée au site.

3.2 Le programme

3.2.1 Principe général

Pour chacune des tables de la base de donnée afin de pouvoir les manipuler efficacement en php chaque table est représentée par une classe possédant les mêmes attributs que la bases de donnée, un accesseur pour chaque attribut et un mutateur pour chaque sauf ceux qui ne doivent pas être modifiés tels que l'id ou la date de création d'un compte.

3.2.2 Cas particulier de User

Une classe de plus à été ajoutée au modèle : la classe User.
Celle-ci est la classe mère de Admin, Client et Vendor permettant de ne pas avoir à réécrire la même chose trois fois pour chaque attribut en commun pour les trois.

3.2.3 Les managers

Les managers permettent de lire la base de données et de convertir une ligne de la table en une instance de la classe correspondant et vis-versa : elle permet d'ajouter, de détruire, de modifier ou d'accéder à une ligne de la base de données ou d'accéder à toutes les lignes. Il y a aussi des opérations spécifiques à certaines tables comme dans Cities qui à une fonction spécifique pour l'auto-complétion allant chercher les villes commençant par une chaîne de caractère.

3.3 Problèmes rencontrés et solutions

3.3.1 Les injections SQL

Un problème que nous avons eu est la possibilité d'effectuer des injections SQL, en effet avec la fonction query l'utilisateur pouvait utiliser la chaîne de caractères qu'il voulait pour faire faire ce qu'il voulait à la base de donnée. Pour palier à ce problème nous avons utilisé la méthode prepare de PDO qui empêche ce genre d'acte malveillant.

3.3.2 Les injections SQL

Le deuxième problème rencontré est l'héritage de User qui n'a pas pu être fait au niveau de la base de données ce qui à été pallié en écrivant trois bases de données distinctes avec des données similaires. Cette solution peu efficace à causé une duplication du code importante et un héritage peu utile au niveau des managers.

4 Réservation de trajets

4.1 Partie client

Pour réserver un trajet les utilisateurs ont à disposition un formulaire dans lequel ils entrent la ville de départ et la ville d'arrivée.

Ils arrivent ensuite sur une page avec la liste des créatures pouvant effectuer ce trajet avec des informations comme le prix et le temps de trajet.

En choisissant un trajet ils arrivent sur une page affichant des informations plus complètes sur le trajet et un lien vers le profil public du vendeur.

Sur cette page ils peuvent réserver le trajet puis sont inviter à entrer un numéro de carte bancaire pour confirmer le paiement.

Le client est alors informé par mail que son trajet à bien été réservé et obtient un lien pour accéder à des informations sur ce dernier.
Dès que le vendeur a accepté (ou refusé) le trajet le client en est informé par un mail.

Le client peut annuler un trajet à tout moment excepté si ce dernier est déjà terminé.

4.2 Partie Vendeur

Lorsqu'un client a fait une demande de réservation, le vendeur correspondant est alerté par mail qui contient un lien vers ce trajet et de là il peut soit accepter soit refuser. Le client recevra alors un mail l'informant de sa décision.

Par la suite le vendeur aura toujours la possibilité d'annuler le trajet avant que celui-ci ne se termine.

4.3 Recherche de trajets

Pour la recherche de trajets on regarde dans la base de données Sur la page de recherche de trajets on utilise Ajax pour

4.4 Envois de mails

Afin d'informer les utilisateurs sur les états des réservations la façon la plus simple étant donné qu'ils peuvent ne pas être connecté au site est de leur envoyer des mails.

Pour cela nous avons du installer PHPMailer afin de faciliter l'envoi de mails. Dans l'objectif dene pas mettre les identifiants et mots de passe du compte mail utilisé pour les envois dans le code source PHP nous avons utilisé XOAUTH2 pour se connecter.

Le compte mail utilisé est un compte GMail car il nous permet d'avoir accès à une quantité illimitée d'alias ce qui est beaucoup plus pratique pour les phases de test.

4.5 Affichage du trajet sur une carte

Pour afficher une carte avec le trajet effectué nous avons utilisé l'API Leaflet car elle est très légère, gratuite et opensource.

Les cartes proviennent d'OpenStreetMap et le chemin parcouru a été obtenu grâce à une API d'openrouteservice. Cependant ce service nécessitant une clé pour être utilisé nous avons du mettre la requête au niveau du PHP pour éviter que les utilisateurs ne voient la clé utilisée. La requète se fait à l'aide d'Ajax car cela permet de réduire le temps de réponse du serveur pour l'envoi de la page.

4.6 Formulaire de choix des villes

Afin d'avoir des villes pour les départs et les arrivées nous avons récupéré une liste de villes qui provient de geonames.
Pour faciliter le choix de la ville nous avons du en supprimer certaines de la base de bonnées pour éviter qu'il n'y ait plusieurs villes ayant le même nom (nous avons gardé celle ayant le plus d'habitants).

Pour que l'utilisateur puisse plus facilement choisir une ville nous avons ajouté un script d'autocomplétion qui affiche à l'aide d'Ajax une liste déroulante contenant des noms de villes commençant par ce que l'utilisateur a entré.
Cependant comme le serveur n'accepte que cinq connexions à la fois nous avons du ajouter un timeout avant de faire la requête et vérifier si la valeur du champ a changé pour ainsi limiter le nombre de requêtes afin ne pas faire planter le serveur.

4.7 Stockage des trajets

Il est impossible de stocker tous les trajets qui peuvent être effectués car il est de l'ordre du nombre de vendeurs sur des villes différentes multiplié pas le nombre de villes (on arrive très rapidement au dessus du million).

Pour les recherches de trajets on regarde donc quels vendeurs sont disponibles à la ville de départ et on crée un trajet pour chacun d'entre eux mais sans le rajouter à la base de donnée. Ils ne sont entrés que lorsque le client arrive sur la page de paiement, pour éviter qu'ils ne soient modifiés pendant le paiement.

4.8 Calcul de distance entre les villes

Comme l'API de recherche de chemin entre deux villes est limitée à des chemins inférieurs à 6 000 km et qui ne traversent pas les océans il faut utiliser une autre méthode pour calculer les distances entre deux villes.

C'est pour cela que les distances sont calculées à l'aide des coordonnées GPS.

4.9 Routeur API

Pour traiter les requêtes Ajax d'autocomplétion et de recherche des trajets nous avons mis en place un routeur API.php afin de séparer les requêtes de pages (index.php) des autres requêtes.

5 L'espace membre

Il a fallu gérer les deux types d'utilisateurs du site web : les transporteurs et les clients. Profitant de la configuration MVC, et puisque tout demande de passage passait par le routeur index.php, on a mis en place un petit fichier de configuration config.php pour y définir certaines variables globales utiles dans tout le reste du code.

Tout d'abord, lors de la connexion de l'utilisateur, on stocke certaines informations dans les variables de sessions. Mais, on s'était vite rendu compte qu'en y stockant l'objet PHP représentant la donnée user, on perdait la spécificité de la classe fille utilisée.
On a donc choisi plutôt en solution de seulement stocker l'id ainsi que le type d'utilisateur. Avec seulement ce couple d'information, on pouvait savoir où chercher et dans quelle table pour retrouver l'utilisateur actuel. Ce couple d'informations est donc stocké dans les variables de sessions. De plus, on charge dans le fichier de configuration config.php directement en tant que variable globale cette fois, la variable objet utilisateur correspondant à l'utilisateur actuel.

5.1 Différents utilisateurs

On distingue les trois types d'utilisateur, correspondant au trois tables du modèle. Les transporteurs (ou vendeurs), les clients, et les administrateurs. Se sont posées les problématiques des multi-comptes ainsi que celui de l'utilisateur possédant différentes casquettes.
Il a été choisi, dans le but de simplifier le modèle de données, de limiter un compte utilisateur par adresse mail, peu importe la nature de ce compte. Autrement dit, une adresse mail employée dans un compte client ne peut être utilisé pour un compte vendeurs.
On s'est toutefois laissé la possibilité de considérer les vendeurs comme des utilisateurs clients améliorées. Enfin, concernant les admins, leur particularité est surtout de leur permettre d'avoir accès à une partie du site leur procurant certaines fonctionnalité supplémentaire avec un accès sur le reste des bases de données.

5.1.1 Inscription

Seul l'inscription pour les deux premiers types d'utilisateurs sont disponible de manière publique. Puisque les données à y inscrire sont légèrement différents, deux contrôleurs différents ont été fait, l'un pour l'inscription client, l'autre pour l'inscription vendeur.

5.1.2 vérification des données

La vérification des données entrées par l'utilisateur, notamment pour le format de l'adresse mail, de la date de naissance et du numéro de téléphone ont été effectués côté serveur (à l'aide d'expression régulières). De plus,

on a rajouté un script javascript permettant de manière plus dynamique de vérifier ces entrées pendant la saisie des données. On a ainsi placé dans la vue, un message de rappel sur le format des entrées, pour les champs qui le nécessitait. Un script javascript ne rendait visible que les indications pour les champs venant d'être modifié et ne remplissant pas ses contraintes.
Bien sûr, la vérification javascript s'effectue côté client et n'est absolument pas une méthode sûre pour s'assurer des données.

5.1.3 Choix de la race pour le transporteur

Un deuxième problème s'est posé quand au choix de la race. Concernant le contrôleur associé au vendeur, il fallait pouvoir disposer des races ainsi que des informations les concernant. Soit, donc du côté php, un appel au modèle pour pouvoir disposer de la liste des races disponibles. Mais c'est encore une fois un script javascript qui a été choisi pour afficher seulement les informations relatives à la race sélectionné pendant la saisie de l'utilisateur.
Par contre, cette fois-ci, si le script n'arrive pas à s'exécuter, on considère comme non nécessaire ces informations. L'utilisateur pourra toujours choisir sa race, même si l'affichage dynamique ne lui permettra pas de voir ses caractéristiques.

5.2 La connexion

Puisqu'il a été choisi d'unifier l'unicité de l'adresse mail pour les trois types d'utilisateurs, une seul page de connexion pour ces trois utilisateurs différents suffit. C'est une fonction commune aux trois manager, permettant de vérifier la présence d'un couple d'identifiant/motdepasse dans une des trois base de données.
D'ailleurs, concernant le stockage de ces mots de passe dans la base de donnée, s'est posée également la question de sécurité. On a du bien sûr, hasher les mots de passe de nos utilisateurs, c'est à dire les passer à travers un algorithme de cryptage tel qu'il est facile de vérifier l'égalité avec un autre mot de passe en lui faisant passer le même processus mais beaucoup plus dur à partir des simples données de la base de donnée duquel on a accès retrouverez le mot de passe.
Concernant le hashage utilisé, on a décidé de se fixer sur la méthode standard défini par les normes de PHP, soit password_verify() et password_hash(), fonctions qui ont l'avantage d'évoluer régulièrement et de ne pas se limiter qu'à un seul type de hashage.

5.3 Le profil

L'ensemble des utilisateurs partageant la même base concernant leur représentation. Tous possèdent une adresse mail, un numéro de téléphone, une date de naissance, ... Ainsi, concernant la gestion de leur profil, similaire donc, un seul contrôleur a été fixé, rajoutant des champs au formulaire si besoin.
C'est donc à cette partie commune à tous les utilisateurs que la connexion commune ramène.
Le choix a été fait d'avoir accès à un formulaire dont la modification a été désactivé. Grâce à un bouton, on peut ouvrir à la modification le formulaire.
Trois choix s'offrent ensuite à l'utilisateur. Soit valider les modifications qu'il a fait, soit revenir à l'état précedent mais rester en état de modification des champs, soit annuler ses changements et revenir à l'état du formulaire figé à l'image des données utilisateurs actuelles.

Toujours concernant le vendeur, s'est posé la même question concernant le choix de la race, et le script javascript d'information associé a été renouvelé dans cette partie.

5.4 Paramètres du compte

On était proposé dans cette partie, pour tous les utilisateurs, la possibilité de changer de mot de passe ou alors de tout simplement détruire leur compte. Les liens vers lequel cet utilisateurs s'était attaché ramèneront alors vers une page pour les utilisateurs désormais inexistants. Il devient Inconnu lors des affichages.

5.5 Mes Trajets

Deux parties se distinguent alors selon les utilisateurs, bien que la vue pourtant reste identique. Du côté des utilisateurs clients, on affiche un tableau représentant l'ensemble des trajets qu'ils ont effectué en tant que client.

Du côté des vendeurs, ce sera l'ensemble des trajets effectués en tant que vendeur qui sera affiché.
Soit, seulement finalement l'appel à la base de donnée change réellement quelque chose entre ces deux types. Cet espace n'est pas particulièrement destinée à l'admin qui a le droit lui de consulter l'ensemble des trajets.

5.6 Déconnexion

Il s'agit d'une page de déconnexion commune à tous. Elle s'occupe seulement de se vider les variables de sessions, ainsi que les variables globales.

5.7 L'espace Admin

L'espace admin laisse la possibilité d'accéder à la liste des clients enregistrés sur le site. On peut modifier ce qui concerne la partie profil de cette utilisateurs. On peut également détruire ce compte, mais il s'agit d'une opération dangereuse. En effet, tel qu'implementé actuellement, il n'y a pas vérification et suppression de la présence de cet utilisateur dans les autres bases de données.
La même possibilité est laissé pour la liste des vendeurs. Toutefois, on ne peut pas modifier les caractéristiques de leur profil spécifique à leur aspect vendeur.
L'admin a également la possibilité d'accéder à l'ensemble des trajets dans la base de données. Enfin, il est le seul à pouvoir rajouter une nouvelle race dans la base de données, ainsi qu'un nouvel admin.

5.8 configuration du template

Pour pouvoir accéder à l'espace membres et les différentes pages que l'on vient de proposer, il faut que cela soit accessible. En s'aidant des variables de sessions et de celles définies dans config.php, il est facilement possible dans la page template d'y inclure un menu supplémentaire pour accéder à ses pages, selon le degré de connexion et le type de connexion , permettant à l'utilisateur connecté de disposer de son menu correspondant. Le menu vendeur et client reste le même, (car c'est le contrôleur de la modification du profil, et de la liste des trajets qui s'occupe de la distinction)

6 La vue

Au niveau du CSS, on a utilisé la grille de Bootstrap pour pouvoir bien placer les différents éléments du site, notamment pour faire le menu et le pied de page.
On a pensé le site principalement pour grand écran, mais avec l'aide de Bootstrap, le cas des petits écrans est bien traité et ne pose pas de problème apparent d'affichage.
Le design est inspiré du site de Uber, et on a repris l'écriture blanche sur fond noir pour le menu et le pied de page.
On a fait le choix également de centrer le contenu, mais on rencontre un léger soucis avec les tableaux.
Enfin, tout le travail en CSS est dans une seule feuille, qui est appelé par template.php pour que le style s'applique sur toutes nos pages.

7 Possibles améliorations

Certaines tables dans la base de données et ainsi que des manager dans le modèle existent mais ne sont pas utilisées. Elle correspondent à des fonctionnalitées envisagées au début du projet mais qui ont été abandonnées par la suite. Ainsi, il était envisagé la possibilité pour les utilisateurs de commenter les vendeurs et d'établir ainsi un score de réputation.
Dans la partie admin, la possibilité de modifier les aspects vendeur (race, position, occupé) des transporteurs. Enfin, la possibilité de demander l'ajout d'une race par un utilisateur (devant ensuite être validé par l'un des admins)

8 La répartition des rôles

Alexandre Garand : . Le modèle pour les tables Admin, Client, Vendor, Cities, Race

Lucas Bourel : Toute la partie concernant l'espèce membres (inscription, connexion, déconnexion, modification du profil et des paramètres) ainsi que l'espace admin. Au niveau du contrôleur et de la vue.

Corentin Guillerme : toute la partie réservation de trajets côté controller et vue (recherche de trajets, réservation, APIs, envois de mails...), réécriture des URLs

Maxime Song : Le modèle pour la manipulation des courses, et les commentaires sur les clients et les vendeurs (pas mis en place sur le site). Ainsi que la partie CSS.

9 Informations complémentaires

9.1 Sécurité

Afin de protéger notre site contre les utilisateurs malveillants nous avons mit en place plusieurs éléments.

— Vérification de toutes les entrées de l'utilisateur

— Vérification que l'utilisateur a bien le droit d'accéder à une page et d'effectuer une action

— injections SQL : Utilisation de la méthode prepare de PDO

— XSS : Utilisation de la fonction htmlspecialchars() sur toutes les chaines de caractères envoyées par l'utilisateur.

— authentification au compte gmail à l'aide de XOAUTH2 pour éviter de devoir stocker le mot de passe du compte dans les sources

9.2 Modules installés

Afin de pouvoir envoyer des mails nous avons du installer PHPMailer et league/oauth2-google. Pour cela nous avons inclus ces modules dans le fichier composer.json pour qu'ils soient installés automatiquement lors d'un make install

9.3 réécriture des URLs

Notre site utilise le fichier index.php comme routeur donc toutes les demandes de pages passent par ce fichier avec une adresse du type adresseDuSite/index.php ?action=truc

Afin d'avoir des URLs plus jolies nous avons modifié le fichier de configuration du serveur (.docker/nginx-php/config/site.config) pour ajouter des réécritures d'URL.

Toutes les URLs de la forme adresseDuSite/truc sont donc réécrites au niveau du serveur en adresseDuSite/index.php ?action=truc

Cela permet d'avoir de belles URLs au niveau de l'utilisateur.

9.4 bots

Le nombre de villes présentes dans la base de données est de l'ordre de 160 000. Donc pour éviter de ne pas trouver de trajet dans la grande majorité des cas il faudrait remplir la base de données avec un nombre de vendeurs du même ordre de grandeur de le nombre de villes. Pour éviter cela des vendeurs sont automatiquement ajoutés à la base de données à chaque recherche de trajet.

Ce comportement est désactivable en passant la variable $*modeDemo* à false dans le fichier config.php.

9.5 identifiants utiles

compte admin : admin@uber-licorne.com / imanadmin
comptes bots : testprojetlicorne+bot_¡id_aléatoire¿@gmail.com / imabot
compte gmail : testprojetlicorne@gmail.com / 159asxcv