Rapport du projet « Parc Advisor »

Programmation WEB

Université de Paris Dauphine

Réalisé par :

El Beze Mikhael & Lakhdhar Amira

Rapport du projet « Parc Advisor »

Programmation WEB

1- Les objectifs généraux du site, leurs critères de succès et les utilisateurs cibles

Le projet consiste à réaliser une application web « Parc Advisor » qui permettrait aux responsables de parcs de faire une publicité à leur parcs (parc aquatique, parc d'attraction, parc d'aventure…) en payant une somme forfaitaire. Les propriétaires doivent pouvoir publier, suite à la confirmation de l'administrateur de l'application, des informations décrivant leur site, pour qu'un utilisateur puisse voter ou voir la moyenne des notes attribuées par les visiteurs suite à leurs expériences dans les parcs visités. Notre application web cible les responsables de parcs et les potentiels visiteurs, habitués et fans des différents types de parcs.

2- Description de l'environnement technique

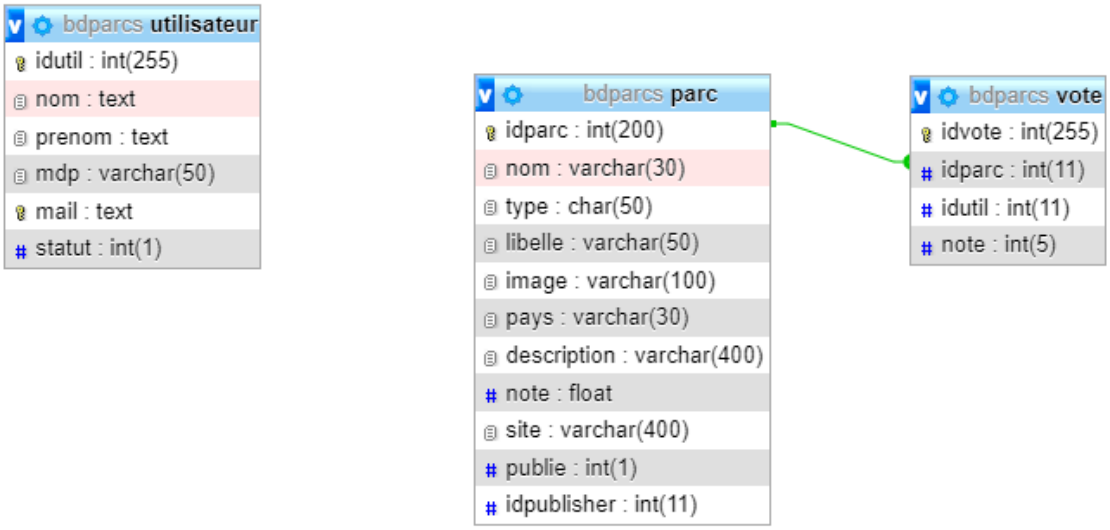
Serveur :

Pour mettre en application le site, il faut d'abord **installer XAMPP**. Vous devez vous assurer que le port **5000** est libre d'être utilisé par le server pour les connexions *http* du site *Parc Advisor.* Si le port est indisponible, vous pouvez le modifier sur **XAMPP** en suivant les instructions sur ce lien. Le style de l'application est réalisé avec Bootstrap.

Base de données :

Il faut créer la base de données qui servira à stocker les informations des utilisateurs, des parcs et des votes réalisés sur le site. Pour créer l'architecture nécessaire, il faut exécuter les requêtes SQL fournies dans le fichier « bdparcs.sql ». Enfin, il faut ouvrir le fichier d'environnement «*.env »* fournit dans le code source et modifier les informations d'accès à votre base de données (USER & PASSWORD) et si vous avez changé le port d'écoute, vous devez mettre le même port dans la variable PORT.

Architecture de la base de données du site :



On a une base de données avec 3 tables :

- **Parc** : qui répertorie les parcs avec les champs ci-dessus explicite. La variable *idparc* est unique à chaque parc et constitue la clé primaire de la table. La table montre au super user si la publication est publiée ou pas et lui affiche les informations fournies pour validation.
- **Utilisateur** : qui répertorie les différents utilisateurs, responsable d'un parc et visiteur, qui vont se connecter au site. La clé primaire est *idutil*. Un utilisateur se connecte avec son mail unique et son mot de passe, préalablement choisi par ses soins. La variable statut sera égale à 0 ou à 1, **0** représentant un **visiteur** et **1 un responsable de parc**.
- **Vote :** qui liste tous les votes faits par les utilisateurs. Elle a pour clé primaire *idvote* et pour clé étrangère *idparc*.

Vous pouvez remarquer l'absence de relation entre l'identifiant de la table *utilisateur* et les champs *parc.idpublisher* et *vote.idutil*. Ce choix est justifié par le fait que tous les utilisateurs ne peuvent pas voter pour un parc et pareillement, tous les utilisateurs ne peuvent pas publier un nouveau parc.

Dépendance :

L'application « Parc Advisor » est développée avec **Node JS et Express JS**. Il faut installer les modules suivants depuis le répertoire « *src* » :

Module	Version	Commande d'installation
nodemon	**2.0.7**	npm install nodemon -g
mysql	**2.18.1**	npm install mysql
ejs	**3.1.6**	npm install ejs
express	**4.17.1**	npm install expess
express-session	**1.17.2**	npm install express-session
dotenv	**9.0.2**	npm install dotenv
alert	**5.0.10**	npm install alert
paypal-rest-sdk	**1.8.1**	npm install paypal-rest-sdk

Vous pouvez vérifier par la suite que tous ces modules sont listés dans les dépendances du projet dans le document « *package.json* » qui se trouve dans le répertoire « *src* ».

Vous pouvez rajouter dans le même fichier : {"start": "node index.js"} pour lancer le projet par la suite avec la commande « nodemon ».

Lancement du projet :

L'environnement technique est maintenant prêt pour lancer le projet. Pour tester, merci de suivre ces instructions :

1- Start Apache et MySQL sur XAMPP

2- Lancer « nodemon » ou bien « node index.js » depuis le répertoire « src »

3- Ouvrez votre navigateur web préféré (hors Mozilla Firefox) et lancer la page http://localhost:5000/inscription

   Vous pouvez maintenant naviguer sur le site.

   Remarque : l'interface fournie est réservée aux utilisateurs cibles de l'application. Cependant, pour le bon fonctionnement du site, il est exigé d'avoir un super user qui vérifie que l'utilisation du site se passe bien, notamment en validant les nouvelles demandes de publication des sites qui seront envoyées par les responsables de ces derniers. Il est donc important que le super user accède à la base grâce à l'outils **phpMyAdmin** pour vérifier les informations transmises par les responsables et valider les publications après avoir vérifié la véracité des informations et ce tout

simplement en changeant la valeur du champ « ***parc.publie*** » attribuée par défaut aux sites de **0** à **1**. Ainsi, la publication sera validée et s'affichera dans la liste des parcs publiés pour permettre à l'utilisateur normal de voter et au responsable de recevoir le vote des visiteurs.

Également, il faudra créer un compte *sandbox* sur PayPal afin de pouvoir simuler le payement.

3- Structure du site

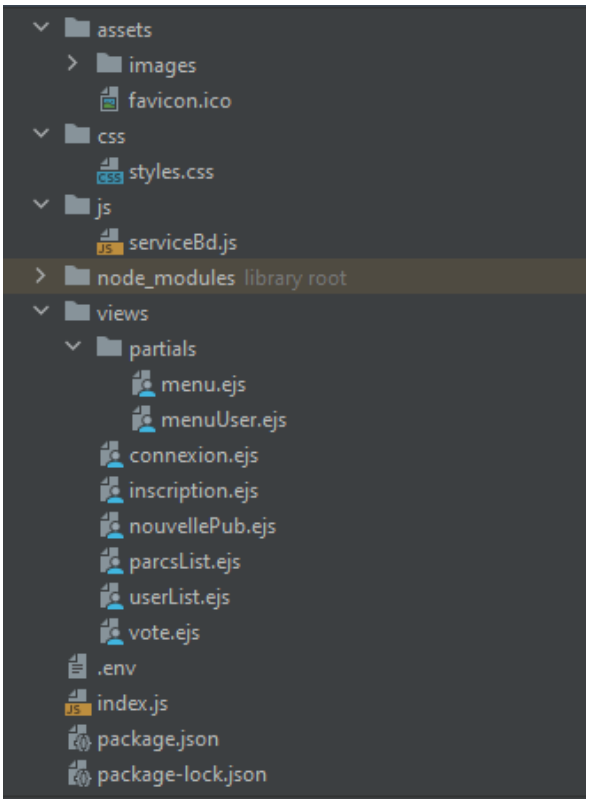
Organisation :

Amira a procédé à l'implémentation du code en node JS sauf PayPal, le style ainsi que les vues.

Mikhael s'est occupé de la base de données et l'implémentation du PayPal, ayant eu des problèmes personnels il n'a pas pu faire.

Le dépôt git du projet : https://github.com/Alakhdhar/Projet-Web-Dauphine.git

Arborescence :



Arborescence du code source

Nous avons divisé le code sur une architecture correspondante aux différentes fonctions. La feuille de style est dans le répertoire css. Les vues du site sont dans views. Comme nous avons deux barres de navigation différentes mais qui se répètent dans plusieurs pages, nous avons décidé de mettre le code associé aux barres dans des fichiers se trouvant dans views/partials et de les inclure sur les vues si nécessaire. Le fichier

index.js est notre fichier d'entrée et lancement du projet. Il organise la navigation sur le site par url. Le document serviceBd gère l'exploitation et le dialogue avec la base de données. Enfin, le répertoire assets/images regroupe les images locales du site ainsi que celles pouvant être rattachée aux parcs par les responsables lors d'une publication de nouveau parc.

But des pages :

Voici la composition des pages de l'application :

- **Page d'enregistrement :** une page affichant un formulaire à remplir par les utilisateurs, tout en précisant le statut (propriétaire ou visiteur). Deux utilisateurs ne peuvent pas avoir la même adresse mail (contrainte exigée sur la base de données où le champ « *utilisateur.mail* » est unique). Ici, deux scénarios se présentent :
    - L'utilisateur est un simple visiteur ou futur visiteur : Il peut s'inscrire gratuitement et visualiser les publications sur les parcs, les comparer en observant les notes attribuées aux sites et voter s'il le souhaite, en se basant sur ses expériences personnelles vécues dans le(s) parc(s).
    - L'utilisateur est un propriétaire ou chargé de publicité d'un parc : afin de pouvoir bénéficier du service de publicité de son bien, celui-ci doit passer après l'inscription par un lien PayPal pour payer une somme pour chaque publication de site. Ce paiement lui permet alors de publier les informations concernant son parc et de les mettre à jour si besoin au fil du temps et ce, après la validation de l'administrateur du site des informations fournies afin de garantir une publication authentique à la réalité, bannir toute tentation d'arnaque des potentiels futurs visiteurs des sites ou de triche et mise en avant du parc par rapport aux parcs concurrents en cas d'utilisation de fausses informations.

- **Page de connexion** : une page qui permet aux utilisateurs, grâce à un formulaire, de se connecter à leur compte existant. En remplissant correctement le formulaire de connexion et si la connexion est réussie, l'utilisateur, quel que soit son statut, sera redirigé vers la page listant les parcs validés

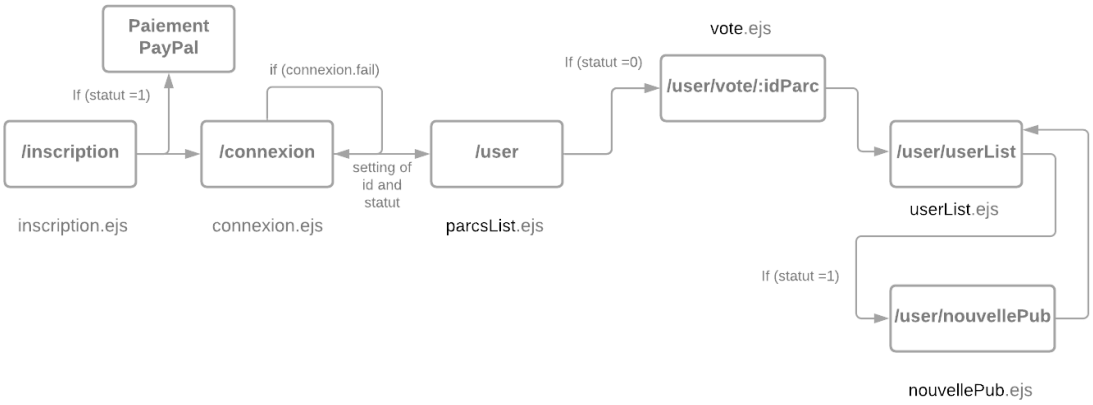
à la suite de son inscription. Dans le cas contraire, une fenêtre d'alerte s'affiche pour notifier l'utilisateur qu'il doit recommencer ou s'inscrire au site.

- **Page de paiement PayPal :** cette page permet à un responsable de parc, après son inscription sur le site, de payer l'abonnement et ainsi de pouvoir se connecter sur le site afin de pouvoir profiter des services de celui-ci.

- **Page de liste des parcs** : liste les parcs publiés. En haut de la page se trouve une barre de navigation permettant à l'utilisateur l'accès aux services correspondant à son statut. Si c'est un simple visiteur, il pourra visualiser les sites qu'il a noté et supprimer ou modifier son vote. Pour cela, il devra se diriger sur la page de votes en cliquant sur « Mes votes » de la barre de menu. Si c'est un publicateur, il pourra envoyer une demande de publication de contenu en fournissant un formulaire avec les informations nécessaires à la description des parcs (site officiel, type du parc, pays, description rapide…). Pour cela, il faudra cliquer sur l'icône « Mes parcs » de la barre de menu. Ce qui permettra une redirection vers la page de publication. Enfin, un bouton « Déconnexion » permettrait à l'utilisateur la déconnexion de sa session active et le retour à la page de connexion.
- **Page de parcs publiés par l'utilisateur** : fournie aux responsables de parcs. Elle liste les publications existantes de l'utilisateur, validées et en attentes de validation et un bouton « nouvelle publication » permettra d'en créer une nouvelle.
- **Page vote** : fournie aux visiteurs. Elle permet de visualiser les votes attribués par l'utilisateur aux différents sites, des modifier les notes attribuées aux parcs et de les supprimer si voulu.

4- Ergonomie du site

Maquette



Schémas des scénarios possibles

Découpage des pages

Les pages sont découpées selon les fonctionnalités qu'on veut fournir à l'utilisateur selon son statut :

Un utilisateur, qu'il soit responsable ou visiteur doit s'inscrire sur le site puis se connecter pour bénéficier des services du site. Une fois connecté, la barre de navigation ainsi que l'affichage s'adaptent à l'activité de l'utilisateur et à son statut. En effet, un utilisateur qui veut voter, a la possibilité de voir la liste des parcs validés sur la page « /user » avec la possibilité de choisir les parcs pour lesquels il souhaite voter. En cliquant un des boutons de vote, il est redirigé vers la page de vote du parc. Quand il envoie son vote, la page de ces votes (accessible aussi grâce à la barre de navigation) est actualisée. Cette page lui permet de se modifier un vote existant s'il change d'avis par exemple ou bien de le supprimer. Sur la page « Mes votes » les notes affichées par parc sont celle attribuées par l'utilisateur connecté, tant dis que les notes affichées dans la page « Parcs » représentent les moyennes des notes attribuées par différents utilisateurs à chaque parc.

Un utilisateur responsable de parc, est identifié dès sa connexion, la page « Parcs » lui affiche donc les mêmes informations que celles visibles par un visiteur de parc qui peut voter, seulement, on ne permet pas à un utilisateur de voter. Il peut cependant consulter sa liste des

parcs transmis et publiés si validés par le super admin. Comme on exige que les informations des parcs soient vérifiées par le super admin avant publication, on demande aux responsables de supprimer la publication et de créer une nouvelle avec les bonnes informations et ce grâce au formulaire qui se trouve dans la page « /user/nouvellePub » accessible depuis la page « Mes parcs ». Il ne peut donc pas modifier les informations d'une publication sans l'accord du super admin. On lui permet donc de supprimer une publication et d'en créer une ou plusieurs nouvelles, qui sont par la suite envoyées pour validation.

- Le paiement par PayPal : nous avons trouvé une démonstration qui semble être adaptée à notre architecture de projet. De ce fait, nous avons pu simuler le paiement via PayPal grâce un compte *sandbox* mais qui peut être changer à tout moment en un paiement réel (en remplaçant la variable sandbox par live sur le *index.js)*.

5- Plan de tests effectué pour vous assurer du fonctionnement

Liste des tests réalisés :

-Inscription en tant que simple visiteur / responsable.

-Payement via PayPal.

-Connexion avec des identifiants existants et sans compte existant.

-Connexion et inscription avec un format de mot de passe et d'adresse électronique non valide.

-Vote sur un site si connecté en tant qu'utilisateur simple.

- Voir la liste des votes ou des parcs publiés quand l'utilisateur avant toute activité de l'utilisateur.

-Nouvelle publication si responsable.

-Modifier et supprimer un vote.

-Supprimer une publication.

-Déconnexion.

6- Point de blocage

Problème rencontré – Façon d'y remédier :

Nous avons essayé d'implémenter une connexion avec création de session mais sans réussir.

Se rendant compte qu'on avait juste besoin de récupérer l'identifiant et le statut de l'utilisateur connecté pour adapter les services fournis aux besoins, nous avons choisie l'alternative d'utiliser des variables locales dans le fichier « index.js » qui sont « id » et « stat ». Ces variables sont tout au long de l'application sollicité pour l'exploitation et dialogues avec la base de données. Les bonnes valeurs sont affectées à ces valeurs locales lors de l'inscription après avoir vérifié que l'adresse électronique et le mot de passe sont existants et valides.

Problème non solutionné – Les raisons :

- L'Upload des images depuis une source extérieure au répertoire du code source de l'application : Nous avons compris comment charger une image grâce au formulaire de création d'une nouvelle publication si l'image sélectionnée se trouve dans le chemin « src/assets/images ». Nous n'avons pas trouvé de solution pour permettre l'upload des images depuis une autre source en les copiant dans le répertoire « src/assets/images » lors de la sélection.
- La gestion d'erreur actuelle est faible. Après une longue recherche sur le sujet, nous avons compris que ça nécessite plus de temps et d'expérience dans la manipulation de NodeJS qui peut être très puissant sur ce point de gestion.

7- Les évolutions à moyen et long terme :

Réalisant l'importance des sessions dans l'assurance de sécurité du site, nous pensons que c'est une priorité à implémenter pour assurer plus de sécurité au super admin ainsi qu'aux utilisateurs, notamment les responsables de parc qui sont supposés payer pour un service sécurisé. Pour la même raison, nous recommandons une meilleure gestion des erreurs qui peuvent avoir lieu lors de l'exploitation du site.