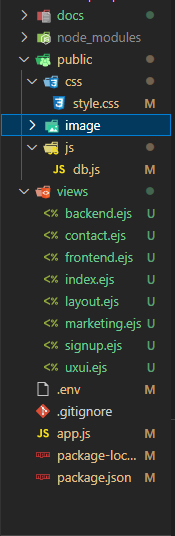
**Présentation du site**

A cause des problèmes d’accès de Nodejs à lire directement les fichiers statique sans les initier avant, notre première intervention a été de réorganiser notre dossier.

Regrouper les fichiers statiques dans un dossier « public », et nos html dans « views ».

Puis, on a décidé d’utiliser *ejs (npm i ejs, puis renommer les .html par .ejs)*pour l’interaction entre back-end et front-end. C’est une langue de template qui va nous laisser écrire du HTML dans du javascript. Rien ne change au niveau du html, on peut juste ajouter du script précisément là où il nous le faudrait.

L’attribut max dans note a été changé en "10" pour plus de diversité au cours des tests.



# Détails de chaque nouveau fichier

## public/js/db.js

C’est le fichier qui va générer (lié) notre base de données à nodejs

## views/layout.ejs

Avec express-ejs-layouts, on peut se servir d’un plan où on affichera les parties de code qui sont les mêmes pour toutes les pages (notamment notre header) pour ne pas avoir à les répéter.

## .env

Contient nos variables relatives à la base de données. On aura besoin d’installer la module *dotenv* pour lire son contenu à travers « process.env.notre\_variable »

## .gitignore

Git essayera toujours de faire un push de tous les fichiers ajouté, pour sauver un peu de temps, on préfère lui dire à l’avance quel partie ignorer

## app.js

Le paramètre de notre serveur et aussi de nos routes

# Démarche

Nous installons *mysql2* pour nous permettre à db.js de lire la base de données, et les exporte à l’aide de notre seul variable présent … pool.

Après la mise en place de nos serveurs dans app.js, on fetch pool avec database, et on met en place toute les prérequis pour lire nos fichiers statique et ejs

## Routes

app.get est utilisé pour chacune d’entre elle, ensuite, on ne rend pas tout de suite la page demandé, on exécute d’abord une requête sql pour verifier siun avis déjà présent dans la base est :

* Dans Home : la note est plus grande ou égale à 7, trié du plus grand au plus petit
* Dans les autres pages : l’avis le plus récent posté (l’id le plus grand)

## views/contact.ejs

Receuille les données entrées par l’utilisateur. Ils sont ensuite stocké dans un tableau (result) qui va être repartis selon le modules de chaque utilisateur. En appuyant sur submit, et après avoir remplis tout les inputs requis, /action\_page nous redirigera vers la page d’acceuil.

## Affichage

Les 2 premiers avis selon la condition si dessus sur la flip-box-front, et les 2 derniers sur la flip-box-back.

Il se pourrait alors qu’au début, une ou plusieurs avis soient présent des 2 côtés, ceci est causé par le fait que dans le code suivant :

connection.execute("SELECT id\_forms, firstName, testimony, note FROM forms WHERE modules='Frontend' ORDER BY id\_forms DESC", (err,result)=>{

            res.render(`${\_\_dirname + '/views/frontend.ejs'}`, {title:"Front-end", front:result})

        })

*result* nous retourne un tableau contenant nos les valeurs de notre requête, et sera l’élément examiné par *ejs* pour savoir quoi afficher sur la page

## Fichiers ejs

ejs nous permettra de rendre des scripts directement sur html. Nous utiliserons la balise <% %> et ses variantes à l’emplacement exact où on en aura besoin. Cependant, il ne va pas nous permettre de générer de nouveaux données dans un html, il nous faut donc les recupérer ailleurs.

## Bug présent

Après un certain nombre de requête, quelqu’en soit la nature, notre serveur cesse de charger une la page demandée. Il ne renvoie cependant aucune erreur pouvant nous aider à le résoudre, nous forçant à toujours le quitter puis redémarrer pour pouvoir continuer.Cela est sûrement dû à une faute de priorité sur les lignes de codes. A mettre à jour.