rapport Écrit projet AWS

Site De Covoiturage

UVSQ – Versailles Année 2019/2020

Maher Attouche – Ismail Benammar – Shadé Alao A. – Clément Carré – Yasmine Keskes

Table des matières

[I. Cahier des charges 2](#_Toc39061831)

[a. Définition 2](#_Toc39061832)

[b. Projet 2](#_Toc39061833)

[c. Contraintes 2](#_Toc39061834)

[d. Exigences non fonctionnelles 2](#_Toc39061835)

[e. Exigences fonctionnelles 2](#_Toc39061836)

[f. Fonctionnalités attendues 3](#_Toc39061837)

[g. Conclusion 3](#_Toc39061838)

[II. Présentation du groupe 4](#_Toc39061839)

[III. Moyens de communication 4](#_Toc39061840)

[IV. Outils utilisés 5](#_Toc39061841)

[V. Produit livré 6](#_Toc39061842)

[h. Accès Local 6](#_Toc39061843)

[i. Accès Online 6](#_Toc39061844)

[j. Fonctionnalités développées 7](#_Toc39061845)

[VI. La sécurité 8](#_Toc39061846)

[k. Les failles les plus connues 8](#_Toc39061847)

[l. Les mesures de prévention 8](#_Toc39061848)

[VII. Explication du code 9](#_Toc39061849)

[a. Le contrôleur 9](#_Toc39061850)

[b. Le Module .ejs 9](#_Toc39061851)

[c. La base de données 9](#_Toc39061852)

[d. Les pages html 10](#_Toc39061853)

[e. Inscription 10](#_Toc39061854)

[f. La messagerie 11](#_Toc39061855)

[g. Rechercher un trajet 14](#_Toc39061856)

[h. Gestion des demandes 16](#_Toc39061857)

[i. Le côté responsive 17](#_Toc39061858)

[VIII. Conclusion 18](#_Toc39061859)

[IX. Webographie 18](#_Toc39061860)

# Cahier des charges

* 1. Définition

Covoiturage : Utilisation d’une même voiture particulière par plusieurs personnes effectuant le même trajet (notamment afin d’alléger le trafic routier et de partager les frais de transport). [Larousse]

Sécurité informatique : La sécurité informatique consiste à protéger un système informatique contre toutes tentatives d’attaque. Une sécurité est jugée satisfaisante si le coût de l’attaque est supérieur à ses bénéfices.

* 1. Projet

Nous sommes amenés à développer un site internet, qui aura pour but de faciliter le covoiturage entres inconnus. Il devra permettre de s’inscrire et se connecter, rechercher ou proposer un trajet, candidater pour un trajet, pouvoir accepter ou refuser une demande, accéder aux profils des autres usagers, communiquer avec eux via un système de messagerie, ainsi que donner son avis sur les divers utilisateurs de la plateforme.

* 1. Contraintes
* Contraintes temporelles : respecter le calendrier du projet (05/03/20 au 30/05/20)
* Respect des divers rôles en fonction des semaines
* Communication entre les différents membres du groupe
* Le site doit être accessible sur smartphone, tablettes et ordinateurs et optimisé pour ceux-ci
* Les messages envoyés lors de précédentes conversations doivent toujours être visible
  1. Exigences non fonctionnelles
* Portée du produit :

Tous les utilisateurs auront les mêmes droits, ils pourront tous exercer les mêmes actions en se servant du site.

* Exigences du client :

Le site doit être sécurisé afin d’empêcher l’accès aux comptes d’autres utilisateurs dans le but d’usurper leur identité ou de voler des données. Il a également été demandé d’être codé en JavaScript.

* 1. Exigences fonctionnelles
* la prise en main du site doit être facile
* L’apparence du site doit être agréable
* Toute action doit être rapide à effectuer
  1. Fonctionnalités attendues
* Gestion du profil : inscription/connexion/modification, le profil contient l’identité de l’usager, les avis sur lui, ainsi qu’un moyen de le contacter.
* Rechercher un trajet
* Proposer un nouveau trajet
* Accéder au profil d’un autre usager
* Avis : donner un avis ou pouvoir lire les avis sur un utilisateur
* Candidater à un trajet
* Accepter/refuser une candidature
* Rappel des trajets prévu pour chaque utilisateur
* Messagerie : communiquer avec d’autres utilisateurs de la plateforme
  1. Conclusion

Nous avons présenté dans ce cahier des charges tous les éléments nécessaires à la réalisation du projet. Le produit qui sera livré le 30/04/2020, devra donc être fonctionnel et devra au moins présenter toutes les fonctionnalités citées précédemment.

# Présentation du groupe

Malgré un échange de rôle en début de projet, les nouvelles contraintes de travail à distance nous ont plus ou moins forcés à garder des rôles stables tout au long du développement de notre site de covoiturage.

En majeur partie Maher s’occupait du Back-End, en étroite collaboration avec Ismail.

Shadé ainsi qu’Ismail se sont chargés du Front-End.

Yasmine s’est focalisée sur la partie recherche concernant l’aspect sécurité du site, et au moment de son application, elle a également contribué à sa mise en place.

Quant à Clément, il a été force de proposition concernant les différentes fonctionnalités à développer. Il a également contribué avec Maher à la vérification du code et aux commentaires.

# Moyens de communication

Nous savons tous que la communication dans un groupe est l’un des piliers essentiels pour mener à bien les différents projets. Communiquer régulièrement sur l’avancée de tout un chacun, s’entraider pour débloquer un coéquipier permettent en autre de maintenir une bonne cohésion de groupe et une motivation constante. Ce travail d’équipe a d’autant plus été important durant cette période de confinement puisqu’il n’y avait plus de période de TD obligatoire hebdomadaire.

Pour ceci, nous avons en grande partie utilisé Discord. Cette application nous a permis de rester en contact, de discuter sur les tâches que chacun doit réaliser, de réorganiser des réunions grâce aux canaux vocaux de ce dernier et de partager des fichiers, de la documentation et certaines recherches réalisées.

Dans un premier temps **glitch** (<https://glitch.com/edit/#!/bolder-spotless-steed?path=index.html%3A1%3A0>) nous a permis de partager le projet. Cet outil a été choisi car il est gratuit, en ligne, simple d’utilisation ou l’on peut voir en direct sur internet le résultat du produit et il permet également la modification du code instantanément par plusieurs personnes, ce qui évite d’avoir des conflits. Mais l’une des faiblesses que nous avons pût rencontrer au fils du temps est sa vitesse d’exécution. En effet les pages mettaient un peu trop de temps à charger. C’est principalement pour cette raison, que nous avons décidés unanimement da basculer sur **GitHub** (<https://github.com/MaherAtt/CoivoitAws>). Nous avons choisi cet outil pour collaborer car il était connu de tous, il est simple d’utilisation mais nous devions constamment veiller à régler correctement tous les conflits lors d’une fusions afin de ne pas effacer le contenu que chacun a pu ajouter au fils du temps.

# Outils utilisés

Différents outils nous ont permis d’arriver à produire ce site.

Les éléments de base servant à la création d’un site vitrine ont été utilisés :

* **HTML**
* **CSS**

Il nous a été imposé dans les contraintes du projet à programmer en **JavaScript**.

Concernant le développement du front-end, certains membres du groupe avaient déjà utilisés **Bootsrap** qui est un outil open source servant entre autre à faciliter la réalisation du CSS et à rendre le site responsive.

Grace à nos différentes recherches, il nous est paru évident d’utiliser **Nodejs** pour le développement du site puisqu’il a été en 2019 le framework javascript le plus utilisé (<https://www.konstantinfo.com/blog/angularjs-vs-nodejs-vs-reactjs/>). Etant étudiants et en formation pour le monde du travail, se former à un outil fréquemment utilisé est donc primordial pour nous.

**NodeJs** nous permet d’utiliser JavaScript côté client et serveur aussi. Comparé à PHP, Nodejs est plus performant et plus rapide côté serveur grâce au moteur d’exécution V8 de Google et à son fonctionnement non bloquant. D’ailleurs c’est ce qui a poussé cette technologie à être employé et éprouvé par les géants du web.

De plus, cette technologie a été utilisée avec le Framework **Express.js** pour le routage des différentes pages. Ce qui nous a permis l’utilisation du Template **EJS** (Embedded JavaScript).

**EJS** nous sert à appeler du code HTML depuis NodeJS dans le but de séparer le HTML du JS et d’avoir un code propre ceci en le sauvegardant sous l’extension *.ejs* dans le dossier *./views*

Concernant la base de données (BDD), nous avons utilisé le système de gestion de bases de données relationnelles MySQL.

L’application a été déployée grâce à **Heroku** qui est une plateforme permettant le déploiement d’applications en s'appuyant sur Git. Cet outil nous a facilité le travail car il suffisait juste de faire un push sur Git et directement l’application étais mise à jour. La simplicité d’utilisation, la gratuité, la performance ainsi que la malléabilité ont été des ingrédients essentiels du choix de l’outil. En effet un certain temps de réflexion s’est imposé à nous entre : <https://www.hubside.com/fr_FR/>, <https://www.hostinger.fr/>, <https://www.heroku.com/>.

En commençant le projet, la première idée était d’utiliser l’api Google Maps, mais celle-ci étant devenue payante depuis juin 2018 et ne rentrant pas dans nos frais, nous a obligé à trouver une tierce solution. La géolocalisation a donc été gérée grâce au module **node Geocoder** qui est une API OpenCage. Elle nous permet de soustraire un certains nombres de données de géolocalisation via les coordonnées gps ou via une adresse postale.

# Produit livré

* 1. Accès Local

Le site est accessible de différentes manière. Dans un premier temps si vous le souhaitez, vous pouvez cloner le projet sur GitHub <https://github.com/MaherAtt/CoivoitAws>.

Ensuite lancez le projet en suivant ces étapes :

Npm install (permet de mettre à jour les différents modules utilisés)

Node bin/www pour lancer le projet

Puis ouvrez un navigateur web (Google Chrome de préférence) : <http://localhost:8000/>

Si vous utilisez déjà le port 8000 pour un autre logiciel, vous pourrez bien évidemment le changer dans le fichier ./bin/www



À vous de jouer, suivez les étapes simples de d’accès (Inscription – Connexion – Ajouter trajet – Rechercher trajet (etc.) – Déconnexion)

À savoir que différents compte existent déjà si vous souhaitez uniquement voir les jeux de données :

|  |  |
| --- | --- |
| Identifiant | Mot de passe |
| [maher@gmail.com](mailto:maher@gmail.com) | maher |
| [smail@gmail.com](mailto:smail@gmail.com) | smail |
| [alao.a.s@gmail.com](mailto:alao.a.s@gmail.com) | alao |

Si vous souhaitez voir la version mobile, il est préférable de suivre les instructions de la section suivante.

* 1. Accès Online

Le site est également accessible en ligne, cela permet donc de le voir sous différentes formes (taille d’écran).

Connectez-vous sur ce site <https://young-peak-38493.herokuapp.com/>

La BDD étant hébergée en ligne, les identifiants de connexion restent les mêmes que ceux évoqués dans la section précédente.

* 1. Fonctionnalités développées

Fonctionnalités réalisées :

* Réalisation d’un site web responsive
* Inscription de l’utilisateur tout en gardant la sécurité de ces informations et la sécurité de notre site notamment le hachage du mot de passe et la prévention contre les attaques XSS et l’injection SQL. Une partie concernant les attaques est consacrée dans le chapitre suivant.
* Connexion et déconnexion de l’utilisateur
* La visualisation et la modification des informations de son profil, par exemple la modification de la photo du profil
* Donner un avis et une note à un utilisateur
* La recherche et la visualisation des profils contenant les différents avis ainsi que la moyenne d’évaluation laissée par les autres utilisateurs
* Intégration d’une messagerie entre utilisateurs. Le stockage des messages se fait en temps réel dans la base de données.
* La recherche d’un trajet, elle est basée sur la date et les positions de départ et destination. Une liste des adresses est proposée lors de la saisie d’une position. Les adresses des trajets affichées sont au maximum 1km plus loin de l’adresse souhaitée
* Une demande de rejoindre un trajet est mise en attente jusqu’à la validation par son chauffeur
* Une proposition d’ajout d’un trajet est validée si les informations entrées sont cohérentes
* Le chauffeur visualise ces trajets proposés qui sont soit validé soit refusé soit en attente de validation
* Message d’alerte quand on veut accéder à une page hors connexion

Fonctionnalités non réalisées :

* Calcule du temps du trajet : contrairement à la carte géographique qui est gratuite dans certains sites, l’API pour le calcul de la distance et du temps du trajet est payante, ce qui nous a obligé à faire un calcul de distance approximative à vol d’oiseau
* Vérification de l’existence réelle d’une adresse email

# La sécurité

De nos jours, maintenir la sécurité de nos applications est devenu primordiale afin d’éviter au maximum des attaques menaçantes. Mais avant de penser à la sécurité il faut dans premier temps connaitre ce qui nous menace. En d'autres termes quels sont les « failles » que peuvent contenir une application web.

Dans ce qui suit, nous allons lister les failles les plus connues ainsi que les contres mesures prisent en prévention de possibles attaques.

* 1. Les failles les plus connues
* *Injection SQL :* Ceci consiste à détourner le comportement de l’application en injectant du texte via les formulaires et donc modifier la requête exécutée. Par exemple on peut envoyer une requête qui nous enverra le mot de passe d’un utilisateur X ou alors afficher la liste de tous les clients ainsi que toutes leurs coordonnées.
* *Attaque par force brute :* Sur les mots de passe en testant tout bêtement toutes les combinaisons possibles pour un mot de passe, jusqu’à l’obtention de la bonne combinaison. Ce type d’attaque dépend des performances de la machine ainsi que la taille du mot de passe. Cette attaque est réalisable en pratique tant que la complexité de calcul ne dépasse pas les 280 (d’après la loi de Moore).
* *Les Failles XSS :* Ce sont des attaques faites par le biais des formulaires, en envoyant un bout de script HTML/JavaScript ayant une fonctionnalité malveillante ceci peut être fait en insérant directement le code dans un formulaire ou en le sauvegardant dans un fichier et faire appel à ce dernier.
* *Les Failles include (ou URL) :* Beaucoup de sites en PHP utilisent un paramètre passé dans l’URL (en GET) pour afficher la page souhaitée et le hacker peut tout simplement modifier ce paramètre et il pourra rediriger le client vers un autre site malveillant.
  1. Les mesures de prévention
* Élargir le domaine de caractère utilisé dans les mots de passe ainsi que leur longueur (utiliser des lettres minuscule, des lettres majuscule, des chiffres et des caractères spéciaux) permet d’augmenter la complexité d’attaque par force brute.
* À l’inscription, On stock la signature d’un mot de passe (nous avons utilisé la fonction de hachage sha256). Ce qui permet de ne pas connaitre le mot de passe lui-même et de l’utiliser pour l’authentification. Par exemple on compare la signature des mots de passe pour un utilisateur donné.
* Nous avons utilisé des requêtes préparées pour éviter les injections SQL et donc on récupère ce que l’utilisateur entre dans le formulaire, on le stock dans des variables qu’on insèrera dans la requête préparée.
* En utilisant NodeJs on évite l’utilisation du *PHP* coté serveuret ça nous évite les failles URL.
* Pour éviter les failles XSS on passe par une étape de vérification de paramètre en le récupérant du formulaire avant de l’envoyer. La vérification peut être sous forme de filtre (qui échappe les caractères spéciaux <, &, ”...etc.).
* A noter que, la sécurité d’un site web n’est pas garantie à 100% car il peut toujours y avoir d’autre failles sur l’application qui peuvent être exploité par un hacker mais notre objectif est de réduire les faiblesses qu’ils peuvent y avoir.

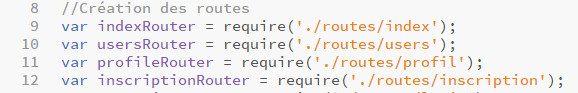
# Explication du code

Le projet a été implémenté grâce aux différents modules que pouvait nous fournir nodejs.

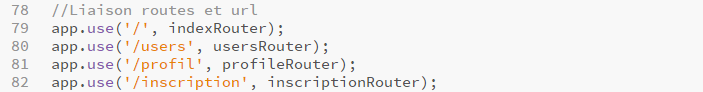
* 1. Le contrôleur

C’est le fichier ***./app*** qui fait la liaison entre les différentes type de pages.

* Les routes : Côté serveur 🡪 pages .js



Capture d’écran fichier ./app.js

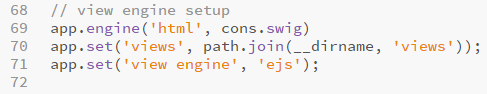
* Les vues : Côté client 🡪 pages.ejs 

Capture d’écran fichier ./app.js

* C’est également à ce niveau que nous défissions le dossier ***./Public*** comme espace où nous allons stocker tous les documents annexes (feuille de style , javascript, images)

Capture d’écran fichier ./app.js

* 1. Le Module .ejs



Capture d’écran fichier ./app.js

Les pages html ont pour extension ejs, cela permet entre autre de rendre le contenu dynamique

* 1. La base de données



Capture d’écran fichier ./app.js

Toujours dans le fichier ***./app.js*** nous nous connectons à la base de données qui est stockée en ligne, sur le site d’heroku

* 1. Les pages html

Elles ont toutes en commun : le head, le header, le début du body, le menu, le footer



Capture d’écran fichier ./views/messagerie.ejs

Cela facilite le développement du projet et permet également de maintenir une cohérence entre les différentes pages.

Seules les fonctionnalités spécifiques à une page donnée y sont ajoutées en plus dans cette dernière.

* 1. Inscription

Quelques vérifications basiques des informations sont effectuées lors de l’inscription



Capture d’écran fichier ./public/javascript/register.js

Le mot de passe est également haché lorsqu’il est inséré dans la BDD

Capture d’écran fichier ./routes/register.js

* 1. La messagerie

Au niveau du HTML nous affichons via une boucle for toutes les personnes avec qui l’utilisateur connecté (émetteur) a déjà conversé (récepteur).

L’identifiant du récepteur est enregistré au niveau de l’attribut href.



Capture d’écran fichier ./views/messagerie.ejs

Cela permet, lorsqu’un clique est effectué au niveau de la classe **conversation** de récupérer le href, correspondant (c’est-à-dire l’id du récepteur). De le placer au niveau de **cet id** via la commande suivante : *$(".afficheConversation").attr("id",res)* où *res* correspond à un sous ensemble de *this.href*.

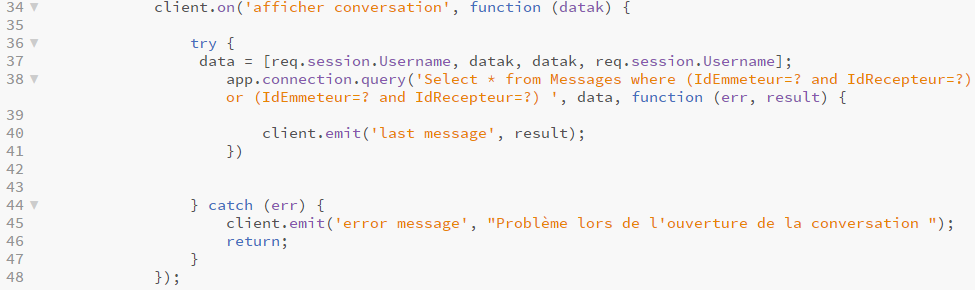
Ensuite, s’il s’agit d’un passage d’une conversation à une autre, je vide le contenu de la liste **#messages** en appliquant cette méthode : *$("#messages").empty()*.

Pour finir, j’envoie une requête au serveur en appelant : *socket.emit('afficher conversation', res);* qui se chargera de remplir / d’afficher (à nouveau) la liste **#messages**. C’est-à-dire les échanges entre l’émetteur et le récepteur



Capture d’écran fichier ./views/messagerie.ejs

Échanges client / serveurs : Afficher un fils de discussion

Après avoir sélectionné une conversation donnée côté client, le serveur se charge de rechercher dans la BDD tous les messages échangés avec le récepteur précédemment sélectionné. 

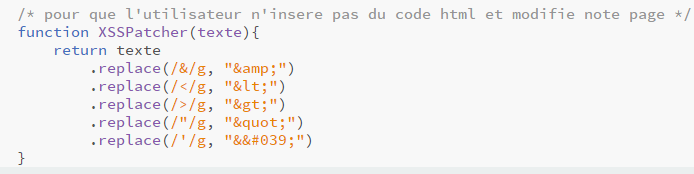
Capture d’écran fichier ./routes/messagerie.js

Ensuite on les affiche un a un côté client.



Capture d’écran fichier ./public/javascript/messagerie.js

Il est important de noter que chaque message est filtré de manière à ce qu’un utilisateur ne modifie pas la page html



Capture d’écran fichier ./public/javascipt/messagerie.js

Échanges client / serveurs : Envoyer un nouveau message

L’utilisateur connecté écrit un message via ce formulaire :



Capture d’écran fichier ./views/messagerie.ejs

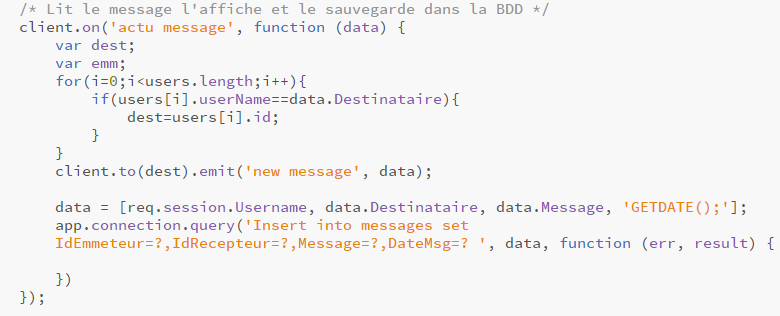
Le client stocke tous les champs nécessaire du formulaire dans une liste **data**

Dans un premier temps, le message est affiché dans le html au niveau de la liste **#messages** puis ce dernier est inséré dans la base de données via une **requête envoyée au serveur**.



Capture d’écran fichier ./public/javascript/messagerie.js

Ici, le serveur enregistre le message dans la BDD via une requête SQL. Pour finir **j’affiche le message dans la page html du destinataire**.



Capture d’écran fichier ./routes/messagerie.js

* 1. Rechercher un trajet

Cette partie est sollicitée lorsqu’une personne souhaite se rendre sur un lieu donné.

L’utilisateur rentre les informations dans le formulaire se situant dans la page *./views/rechercher.ejs*

À noter : une aide à la saisie a été implémentée pour les adresses ainsi que pour le jour et l’heure d’arrivée souhaitée.

Aide à la saisie adresse :



Capture d’écran fichier ./views/rechercher.ejs



Capture d’écran fichier ./views/rechercher.ejs

Une fois la recherche lancée, c’est le serveur qui prend la relève au niveau du fichier *./routes/rechercher.js*

Le module **geocoder** nous permet de récupérer les coordonnées GPS (longitude et latitude) à partir d’une adresse postale. Ces coordonnées seront nécessaire ultérieurement pour le calcul du temps de trajet et la distance entre le point de départ et d’arriver.

La requête SQL retourne un résultat si les points suivants sont **vérifiés** :

* La date et l’heure souhaitée d’arrivée est ultérieure à la de départ prévu

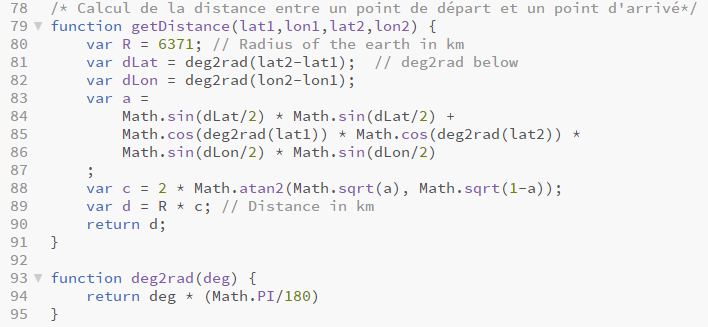
Du trajet + le temps de trajet.

* Il reste encore de la place dans le trajet



Capture d’écran fichier ./routes/rechercher.js

Calcul de la distance entre deux adresses :



Capture d’écran fichier ./routes/rechercher.js

* 1. Gestion des demandes

Correspond au moment où un utilisateur souhaite participer à trajet. Ce dernier formule donc un souhait de participation vers le conducteur, et c’est à celui que revient la charge d’accepter ou de refuser cette proposition.

Pour faire le lien avec la section précédente (recherche d’un trajet), lorsqu’un participant réserve un voyage, cette demande de réservation s’affiche dans la partie « Mes réservations » du conducteur et l’état affiché est « en attente ».

Lorsque le conducteur veut répondre à la demande il clique sur le bouton en attente. Cette action entraine **l’ouverture d’une modale** qui prend en paramètre certains éléments essentiels concernant le trajet et les participants.



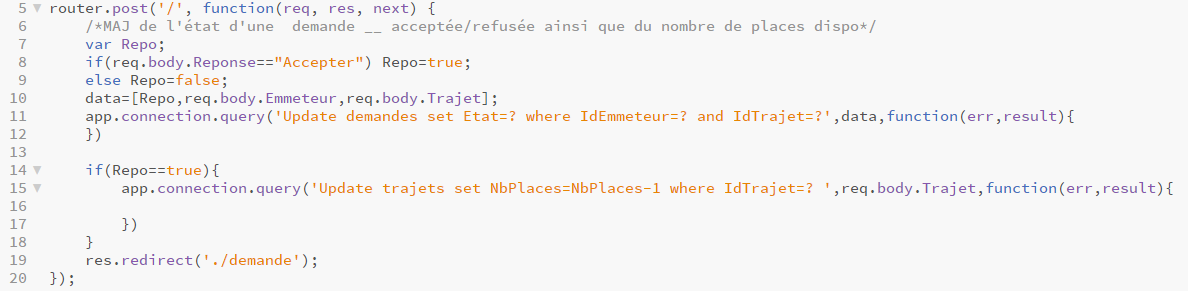
Capture d’écran fichier ./views/demande.ejs

En fonction de la réponse du chauffeur deux actions différentes seront exécuté au niveau du serveur dans le fichier ***./Routes/Repondre.js***



Capture d’écran fichier ./views/demande.ejs

Si la réponse est positive 🡺 Accepter, on met à jour le nombre de places disponibles pour le trajet. Et dans tous les cas : on modifie l’état de la demande dans la BDD

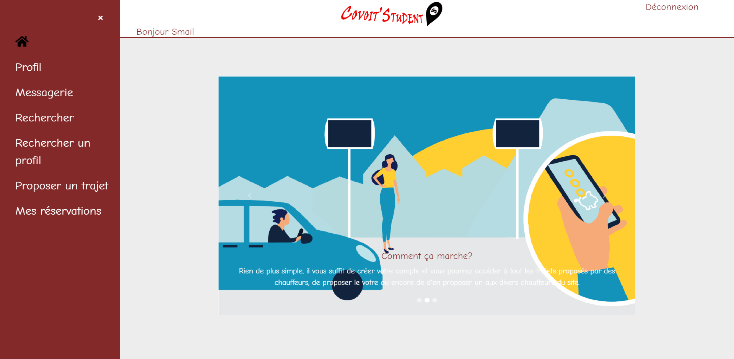


Capture d’écran fichier ./routes/repondre.js

* 1. Le côté responsive

Certaines fonctionnalités du produit changent en fonction de la taille de l’écran.

Ici nous mettons donc en avant le menu 🡺 affichage des icônes ou du texte selon l’écran

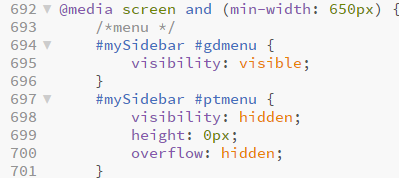




Capture d’écran fichier ./views/partials/menu.ejs

Les medias queries nous ont permis de faire facilement la bascule d’un écran à un autre





Capture d’écran fichier ./public/stylesheets/style.css

Capture d’écran fichier ./public/stylesheets/style.css

# Conclusion

La réalisation de ce site web nous a tous appris. Il nous a fait gagner en compétences techniques (javascript, Node, Api maps etc.) Il nous également permit de tester nos capacités de réadaptation en fonction des différents aléas. En effet le confinement dû au corona virus nous a obligé à réfléchir et à se réorganiser autrement.

La communication à distance est devenue primordiale car nous nous sommes retrouvés dans l’incapacité de nous réunir pour faire des points sur l’évolution du projet.

De plus il fallait trouver un moyen de s’entraider à distance et d’expliquer son point de vu, ce qui n’a pas toujours été chose simple.

Concernant le produit livré, nous sommes assez fières d’avoir pu vous livrer un site fonctionnel.

Certes se dernier comportent quelques petites erreurs que nous aurions pu éviter**.(notamment : )**.

Nous sommes aussi conscients que certaines fonctionnalités auraient pû être mieux (par exemple : ).

Une possibilité d’amélioration de se site web est peut-être la création de son application pour téléphone (sous android).

Réalisé dans des conditions quelques peut contraignantes, nous pensons et nous avons fait notre maximum afin d’atteindre au mieux les objectifs fixés.

# Webographie

<https://devcenter.heroku.com/articles/deploying-nodejs>

<https://hackernoon.com/why-use-nodejs-as-your-backend-4gpa31qk>

<https://www.geeksforgeeks.org/use-ejs-as-template-engine-in-node-js/>

<https://scotch.io/tutorials/use-ejs-to-template-your-node-application>

<https://www.devidia.net/les-5-failles-de-securite-les-plus-courantes/>

<https://mdbootstrap.com/> 🡺 Animations css

<https://getbootstrap.com/docs/4.4/getting-started/introduction/> 🡺 CSS

<https://api.jquery.com/> 🡺 jquery

<https://www.npmjs.com/package/ejs> 🡺 pour beaucoup de modules (express, ejs etc.)

<https://opencagedata.com/tutorials/geocode-in-nodejs> 🡺 module geocoder

<https://induste.com/threads/realiser-un-chat-instantanee-avec-nodejs.583595/> 🡺 messagerie

<http://defeo.lu/aws/> 🡺 javascript

<https://www.w3schools.com> 🡺 programmation en générale

<https://community.algolia.com>