

Gestionnaire transport

Dossier de projet

Septembre 2020



Présenté par Adrien Gonzalez

Sommaire

Introduction	3
1.1 Résumé du dossier de projet	3
Présentation	
2.1 Présentation personnelle	3
2.2 Présentation de La Plateforme	4
Projet entreprise	4
3.1 Présentation de l'entreprise	4
3.2 Présentation du projet	5
3.3 Compétences utilisées	6
3.4 Cahier des charges	8
3.5 Organisation	10
3.6 Spécifications techniques	11
Développement	11
4.1 Réalisations	11
4.1.1 Maquettage du site	11
4.1.2 Développement de la partie front-end	12
4.1.3 Développement de la partie back-end	15
4.2 Jeu d'essai	26
4.3 Veille et sécurité	27
4.4 Difficultés rencontrées	30
Conclusion	32
Annexes	33

Introduction

1.1 Résumé du dossier de projet

Ce dossier présente une partie du travail réalisé lors de ma formation à La Plateforme à Marseille, à savoir le développement d'un site pour une entreprise regroupant un ensemble de compétences pour mon titre RNCP de niveau 5.

Vous y retrouverez donc un panel de compétences mises en avant lors de ce projet d'une durée de 5 mois, ainsi que son déroulement, les attentes attendues par la personne en charge de ce projet et sa réalisation finale.

Présentation

2.1 Présentation personnelle

Je m'appelle Adrien Gonzalez, j'ai 21 ans et j'habite à Marseille dans le 9ème arrondissement.

Sortant d'un BTS SNIR (**S**ystèmes **N**umériques option **I**nformation et **R**éseaux) au lycée Jean Perrin, je voulais continuer mes études dans le domaine du développement web que j'affectionne particulièrement. C'est pourquoi, en cherchant sur internet, je suis tombé sur l'école de La Plateforme qui propose une formation en tant que développeur web / web mobile sur 2 ans (une première année à l'école et la deuxième en alternance).

2.2 Présentation de La Plateforme

La Plateforme est une école du numérique et des nouvelles technologies co-fondée avec le Club Top 20 réunissant les grandes entreprises de la Métropole Aix Marseille. Elle comprend une offre de formations diversifiées destinées à former des codeurs et développeurs web, des experts en sécurité, des ingénieurs spécialisés en Intelligence Artificielle, et des cadres d'entreprises au travers de cycles de formations continues.

Je suis actuellement la formation "Coding School" qui s'adresse à tous les passionnés sans diplômes et sans limite d'âge. C'est une formation gratuite qui offre donc ses chances à toutes et à tous.

Projet entreprise

3.1 Présentation de l'entreprise

Global Prestations annexes est une holding de transport de marchandises proposant un accompagnement structurel au nouveau créateur d'entreprise transport en leur proposant :

- La gestion intégrale de l'administratif (facture, comptabilité, litiges...)
- La mise à disposition du gestionnaire transport (capacité professionnelle – attestataire transport)
- Logiciels de suivi activité (géolocalisation, lettre de voiture dématérialisée, contrôle des heures conducteurs...)
- Proposition de marché et accompagnement vers développement stratégique avec des affrètements transports, contrat client

Lorsqu'une personne souhaite devenir transporteur il faut au préalable avant de s'immatriculer au niveau des greffes de tribunal de commerce, faire une demande d'autorisation d'exercer la profession de transporteur routier de marchandises à la DREAL ou DRIEA suivant les régions. La DREAL (Direction Régionale de l'Environnement, de l'Aménagement et du Logement) ou DRIEA (Direction régionale et interdépartementale de l'Équipement et de l'Aménagement) est une autorité dépendante du Préfet de chaque Région qui contrôle et surveille que les activités dans le domaine du transport respecte les lois entrée en vigueur, auquel cas la DREAL ou DRIEA a le pouvoir de faire cesser toute activité ou mettre en demeure de conformité les établissements en infraction.

3.2 Présentation du projet

Afin de faciliter la mise en relation entre entrepreneurs et gestionnaires, commissionnaires transport, l'idée d'une plateforme, façon "Le Bon Coin" est née du constat qu'aucun site gratuit spécialisé à cette effet n'existait !

Ce projet professionnel consiste donc à créer une plateforme ***gestionnairetransport.com*** par laquelle la mise en relation entre gestionnaires, commissionnaires et entrepreneurs transports est facilitée. Des utilisateurs inscrits pourront déposer des annonces et fournir une attestation certifiant leur éligibilité (vérifié par l'admin avant d'être postée).

La plateforme a pour principal but d'être simple d'utilisation et de faciliter au maximum les échanges entre les différents utilisateurs. A l'arrivée sur le site l'utilisateur pourra voir l'ensemble des annonces postées ainsi qu'un filtre permettant de faciliter sa recherche mais également un système de favoris afin de sauvegarder des annonces pour y revenir plus tard.

Pour terminer, ce site possède une interface administrative accessible au gérant du site permettant :

- la consultation de l'ensemble des annonces en attente de validation
- l'archivage de la base des utilisateurs sous forme de liste (possibilité de ban un utilisateur 30 jours / permanent)
- le pilotage des services en les ajoutant et/ou en supprimant (comme l'option payante de mise en avant des annonces)
- la visualisation et la gestion des paiements effectués par les utilisateurs.

3.3 Compétences utilisées

Pour réaliser ce projet, de nombreuses connaissances étaient nécessaires car le but de ce projet étant de mettre en pratique l'ensemble des compétences / outils utilisé(e)s pendant l'année.

Langages HTML et CSS

J'ai utilisé HTML / CSS comme structures des pages à développer et pour l'intégration. Le CSS m'a aidé pour le responsive des pages grâce aux médias queries ainsi que pour le design complet du site.

Langages PHP et SQL

Le PHP est utilisé pour les requêtes avec la base de données et l'affichage des informations récupérées. De nombreuses requêtes avec la base sont réalisées pour les formulaires de connexion / inscription, de dépôt d'annonce, de la partie profil mais également pour la réalisation de nombreuses fonctionnalités comme les filtres, les favoris, la messagerie personnelle...

Langages JavaScript / jQuery / JSON

Le JavaScript est utilisé pour le responsive (navbar) mais également pour la vérification des formulaires, si un champ n'est pas rempli ou incorrect, une erreur sera affichée sans rafraichissement de la page.

Pour cela, j'utilise Ajax, une communication est faite lorsque j'appuie sur un bouton de validation du formulaire avec un fichier php, qui lui va exécuter une requête sql et renvoyer le résultat sous forme de tableaux json.

En fonction du résultat j'affiche un contenu différents dynamiquement en JavaScript.

Utilisation du framework Bootstrap

Bootstrap est une collection d'outils utiles à la création du design de sites et d'applications web.

J'ai utilisé ce framework afin de faciliter le responsive du site et de m'aider dans le design de celui-ci.

Stripe

Stripe est une société américaine d'origine irlandaise, destinée au paiement par internet pour professionnels.

J'ai utilisé Stripe pour mon site car celui-ci propose une documentation très complète concernant son utilisation. Mon site propose un système de mise en avant lorsqu'on dépose une annonce et c'est pourquoi je devais mettre en place un système de paiement en ligne.

De plus le panel administratif communique avec Stripe via l'api afin d'ajouter des services et/ou d'en supprimer, de voir les paiements et de les rembourser lorsqu'il est nécessaire.

3.4 Cahier des charges

Les objectifs du site Internet :

Proposer un outil gratuit qui permettrait au créateur de société transport de pouvoir utiliser la capacité de transport d'une tierce personne en :

- facilitant la mise en relation entre futur transporteur et gestionnaire transport.
- recruter nos futures filiales (intégration d'un groupe avec mise à disposition du gestionnaire transport).

La cible :

- Créateur de société transport
- Société transport en développement
- Gestionnaire transport pour mise à disposition de leur capacité
- Expert comptable et avocat pour pub

L'architecture du site Internet :

ACCUEIL:

proposer des profils directement comme présenté ci-dessous Filtrer un attestataire pour la capacité marchandises/voyageur/commissionnaire Permettre le dépôt d'annonce pour les gestionnaires transport Intégration latérale bandeau pub (avec renvoi sur site correspondant) + Modèle de document officiel payant Proposition des cookies / CGV / Contact

Les annonces des gestionnaires :

Accueil : Visuel synthétique :

- Photo
- Pseudo
- Lieu d'action
- Type de capacité
- Prix de la mise à disposition
- Téléphone
- Email

Dépôt d'annonce des gestionnaires :

- Nom
- Prénom
- Pseudo
- Date naissance
- Adresse
- Email
- Tel
- Type de capacités avec possibilité d'intégrer les attestations pour vérification
- Tarif € ou à négocier
- Lieu d'action (France entière, paris, Bouches du Rhône) Remplissage libre
- Descriptif de parcours / expériences /objectifs
- Téléchargement photos Jpeg/ PDF
- Possibilité de faire payer comme le principe de LEBONCOIN avec remonter en tête de liste / Mise en évidence / Rajout de photo... Modérateur : Global Prestations Annexes se réserve le droit après contrôle des pièces d'accepter ou refuser toute diffusion sur le site

3.5 Organisation

Concernant l'organisation du projet, nous étions 3 au départ puis nous avons finis à 2 (car la deuxième personne a été basculé en deuxième promotion). Afin de se répartir le mieux les tâches, nous avons fait en fonction des compétences de chacun, moi pour la partie back-end et la deuxième personne pour la partie front-end.

Nous avons utilisé GitHub afin de pouvoir envoyer nos fichiers versionnés dessus et de pouvoir les récupérer depuis n'importe quel appareil mais surtout de pouvoir mettre une description lors d'un commit / push d'un fichier afin de savoir ce qui a été fait pour chaque push de fichier.

GitHub m'a été très utile notamment lors du confinement pour que je puisse récupérer les fichiers sur mon ordinateur personnel et continuer à travailler sur un écran plus grand que l'ordinateur portable.

De plus cela est pratique afin que chacun puisse récupérer des fichiers, travailler dessus, mettre en commun les différents codes...

Toutes les 2 semaines, nous avons organisé avec la cliente, des réunions via google meet afin qu'elle puisse voir notre avancé, donner son avis en voyant notre site via le partage d'écran. Ainsi nous avons ajouté des fonctionnalités en fonction de ce qu'elle demandait.

3.6 Spécifications techniques

Versioning et stockage

Pour ce projet nous avons utilisé Git comme logiciel de versionning, pour stocker notre code versionné sur un serveur distant afin de pouvoir travailler plus facilement dessus chacun de notre côté. Pour cela nous avons créé plusieurs branches sur GitHub, avec nos noms puis la branche master qui elle est réservée pour le projet final. Pour chaque nouvelle tâche effectuée, je faisait un push sur ma branche via l'outil TortoiseGit afin de mettre à jour mon avancé.

Technologies

- PHP : Langage back-end
- HTML / CSS : Structure du site
- JavaScript / jQuery : Création contenu dynamique
- PHPmyAdmin : Base de données
- Bootstrap : Responsive et design du site
- Stripe : Mise en place d'un système de paiement

Développement

4.1 Réalisations

4.1.1 Maquettage du site

En début de projet, j'ai commencé à réaliser une maquette du site avec PhotoFiltre 7 afin de faire un aperçu du design et du jeu de couleur de celui-ci. Je n'ai pas choisis Balsamiq ici car au niveau du design, il était pour moi moins représentatif.

Je voulais, avant de passer au code, donner un aperçu au client concernant le design de son site afin qu'il puisse me dire ses préférences et pouvoir avancer dans sa direction.

(Voir annexes 1 page 31)

Après que le client est validé la maquette, j'ai commencé cette fois-ci à réaliser la maquette côté base de données. Pour cela j'ai choisi GenMyModel qui est un outil permettant de créer plusieurs type de diagrammes / modélisations dont une pour les bases de données. J'ai donc pu construire plusieurs tables avec les données qu'elles devront avoir afin de faire marcher l'ensemble du site.

(Voir annexe 2 pages 33)

4.1.2 Développement de la partie front-end

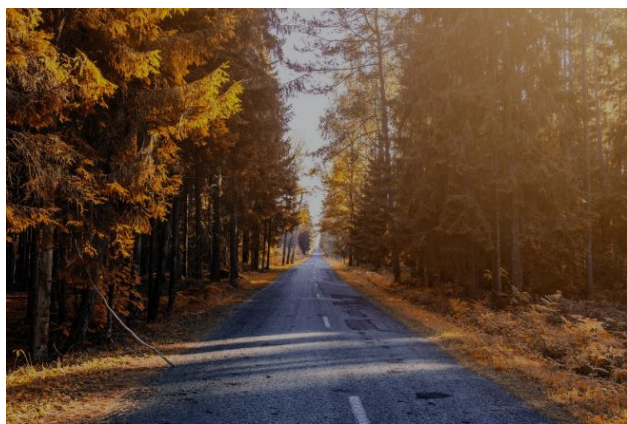
J'ai développé ce projet en gardant à l'esprit que l'utilisateur devrait pouvoir se servir du site dans n'importe quel navigateur, et sur n'importe quel périphérique.

En conséquence, j'ai développé le CSS du site en me servant des *media-queries*, afin de pouvoir modifier l'affichage en fonction de la taille de l'écran et donc du périphérique utilisé. J'ai également fait en sorte que le comportement du site soit adapté à un maximum de périphériques différents. De plus j'utilise également le display flex afin de disposer les éléments de façon à ce qu'ils soient adaptable selon la taille de l'écran avec un "flex-wrap".

```
@media (max-width: 991px) {  
  .option{  
    display: flex;  
    flex-direction: column;  
    align-items: center;  
    height: 100%;  
  }  
}
```

Par exemple ici, je modifie la disposition de ma div "option" pour une taille d'écran inférieur ou égale à 991px.

Concernant le design, j'ai suivi la maquette qui a précédemment été réalisé et j'ai commencé par la page d'accueil qui affichera l'ensemble des annonces postées par les utilisateurs ainsi que les filtres pour trier ceux-ci. Suite à ça, j'ai fait les pages d'inscription / connexion en partant d'un template de formulaire trouvé sur internet que je trouvais très complet. J'ai ensuite utilisé ce formulaire pour chaque page qui en avait besoin. (connexion, inscription, profil ...). Les formulaires sont différents de ceux réalisés dans la maquette mais ils sont plus complet et ont donc été choisis.



CONNEXION

Login

Mot de passe

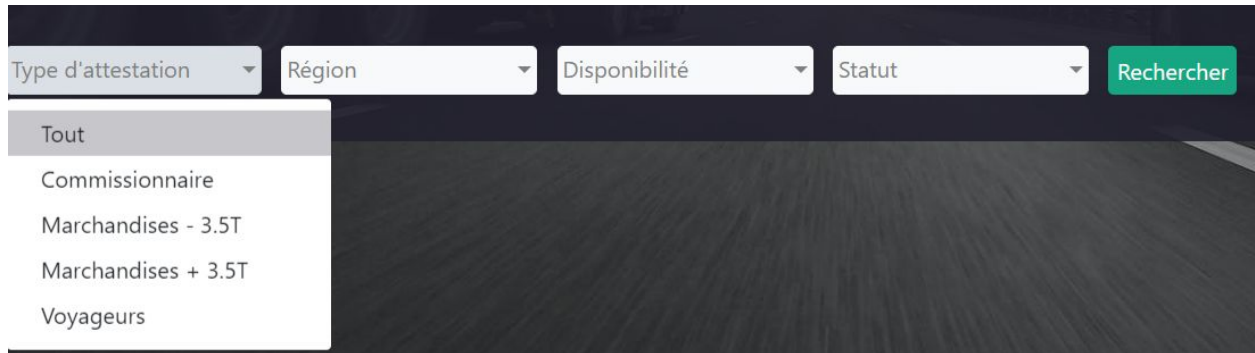
VALIDER

Besoin d'un compte ? INSCRIVEZ-VOUS
Mot de passe oublié ? CLIQUEZ ICI

L'ensemble du site possède le panel de couleur gris et vert de code hexadécimal suivant : #17A681 ■ , #0D836E ■ et #21202E ■.

Comme expliqué précédemment, mon JavaScript permettra de vérifier si les formulaires sont bien remplis lors de leur validation et si c'est le cas, d'effectuer une requête avec la base de données et d'afficher du contenu dynamiquement en fonction du résultat. (ex: login ou mot de passe incorrect, encadrement des input en rouge...).

Mon JavaScript sert également à la personnalisation des filtres présents sur ma page d'accueil.



Ou encore pour changer le css d'un élément du site comme les étoiles servant à mettre en favori une annonce lorsqu'on clique dessus.



De plus ma navbar est réalisé grâce à Bootstrap qui possède une partie de JavaScript intégré afin de la rendre responsive.



4.1.3 Développement de la partie back-end

Arborescence de l'application

Accueil

➤ Inscription

Permet à l'utilisateur de s'inscrire

➤ Connexion

Permet à l'utilisateur de se connecter

➤ Dépôt d'annonce

Permet à un utilisateur inscrit de déposer une annonce

➤ Annonce

Permet à un utilisateur connecté ou non de visualiser une annonce

➤ Messagerie

Permet à un utilisateur inscrit d'envoyer un message à un autre utilisateur

➤ Profil utilisateur

Permet à un utilisateur de consulter un profil d'un autre utilisateur avec ses nombres d'annonces, date d'inscription...

➤ Mes annonces

Permet à l'utilisateur inscrit de consulter ses annonces postées

➤ Mes favoris

Permet à l'utilisateur inscrit de consulter ses annonces mises en favoris

➤ Paramètre

Permet à un utilisateur inscrit de pouvoir consulter et/ou modifier ses informations personnelles

➤ Contact

Permet de contacter le support

Panel d'administration

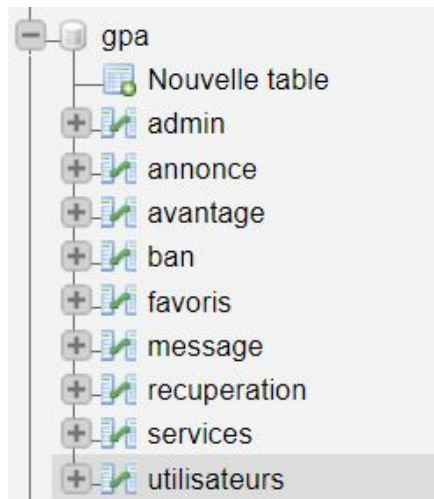
- Afficher les annonces en attente de validation
- Afficher une liste des utilisateurs inscrits
- Modifier ses informations personnelles
- Ajouter un service et/ou supprimer
- Voir liste des achats effectués avec détail

Pour la partie back-end, j'ai d'abord commencé à créer ma base de données avec phpmyadmin. J'ai créé les différentes tables que j'avais maquettée au préalable avec les différentes informations qu'elles contiennent.

Pour exemple, voici la table utilisateurs qui contiendra donc toutes les informations correspondantes aux inscrits du site.

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
<input type="checkbox"/>	1 id	int(11)			Non	Aucun(e)		AUTO_INCREMENT	Modifier Supprimer Plus
<input type="checkbox"/>	2 genre	varchar(5)	latin1_swedish_ci		Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/>	3 nom	varchar(20)	latin1_swedish_ci		Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/>	4 prenom	varchar(20)	latin1_swedish_ci		Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/>	5 adresse	varchar(255)	latin1_swedish_ci		Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/>	6 email	varchar(255)	latin1_swedish_ci		Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/>	7 naissance	varchar(10)	latin1_swedish_ci		Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/>	8 login	varchar(15)	latin1_swedish_ci		Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/>	9 password	varchar(255)	latin1_swedish_ci		Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/>	10 profil	varchar(255)	latin1_swedish_ci		Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/>	11 date	date			Oui	NULL			Modifier Supprimer Plus

La base de données finale comprend 9 tables :



Une fois la base de données créée, j'ai commencé à développer la partie connexion / inscription afin d'inscrire des utilisateurs. Lorsqu'un utilisateur s'inscrit, il doit renseigner l'ensemble des informations suivantes : genre, nom, prénom, adresse, email, login, mot de passe et photo de profil (optionnel). Une fois les données renseignées, je récupère le contenu des champs en JavaScript afin de les passer en paramètre via Ajax pour pouvoir les insérer dans la base avec des requêtes SQL.

Tout d'abord, je regarde si le contenu des champs n'est pas vide, si c'est le cas, je stocke leur contenu dans des variables.

```

function inscription(){
    $(".wrap-input100").css({"border": "solid 1px #E6E6E6"})
    $(".genre").css({"color" : "black"})
    if($("#nom").val() != "" && $("#prenom").val() != "" && $("#adresse").val() != "" &&
        $("#email").val() != "" && $("#naissance").val() != "" && $("#login").val() != "" &&
        $("#password1").val() != "" && $("#password2").val() != "" &&
        $('#homme').is(':checked') || $('#femme').is(':checked') )
    {
        $(".champ_vide").remove()
        genre = document.querySelector('input[name="genre"]:checked').value;
        nom = $("#nom").val()
        prenom = $("#prenom").val()
        adresse = $("#adresse").val()
        email = $("#email").val()
        naissance = $("#naissance").val()
        login = $("#login").val()
        password1 = $("#password1").val()
        password2 = $("#password2").val()
    }
}

```

Une fois les vérifications faites, je passe en données le contenu des champs et les envoie à un autre fichier php qui va effectuer les requêtes d'insertion en base de données.

Ici, vous pouvez voir qu'il y a un ajax dans un ajax car le deuxième permet lui de communiquer avec un autre fichier php qui va faire un upload de l'image de profil de l'utilisateur dans le dossier correspondant. J'ai dû faire un autre ajax car pour pouvoir faire l'upload ça ne marchait pas si je passais des données avec ajax. C'est pour cela qu'il y a écrit "processData : null" car la data doit être à null.

Le deuxième ajax s'exécute si l'utilisateur a choisi une image de profil sinon une image par défaut lui sera donnée.

```
$.ajax({
  url: '../fonctions/fonction_inscription.php',
  type: 'POST',
  data: {genre: genre, nom: nom, prenom: prenom, adresse: adresse, email: email, naissance: naissance,
    login: login, password1: password1, password2: password2},

  success: function(data)
  {
    if(data == "ok")
    {
      var fd = new FormData();
      var files = $('#fileToUpload')[0].files[0];
      fd.append('file', files);

      if(files != undefined)
      {
        $.ajax({
          url: '../fonctions/upload_inscription.php',
          type: 'post',
          data: fd,
          contentType: false,
          processData: false,
          success: function(response){
            }
          }
        }
      }
    }
  }
});
```

En revanche si les champs sont vides, je touche au css pour montrer qu'il faut remplir le champ (exemple pour le champ login) :

```
$("#login").val("");
$('.login').css({"border": "1px solid #C0392B"});
```

Une fois inscrit, j'envoie l'utilisateur sur la page de connexion (même fonctionnement que pour l'inscription).

Maintenant, pour le dépôt d'annonce, plusieurs informations sont demandées par l'utilisateur afin de respecter ce qui été demandé dans le cahier des charges, dont le plus important et l'attestation prouvant que la personne peut exercer son activité. Lorsque la personne dépose son annonce, il est envoyé vers une page lui proposant de pouvoir payer afin de mettre en tête de liste son annonce pendant un certain nombre de jours, ou alors il peut déposer son annonce sans avantages.

Bien évidemment, avant que l'annonce soit postée sur le site et donc visible sur la page d'accueil, elle est d'abord en attente de validation par l'administrateur. L'administrateur possède un panel afin de voir l'ensemble des annonces en attente avec leurs informations, (descriptions, attestations au format pdf...).

Après validation, l'annonce s'affiche sur la page d'accueil, et pour cela j'ai effectuée une requête qui va sélectionner l'ensemble des annonces validées et en fonction des filtres choisis (aucun par défaut).

Afin de récupérer l'ensemble des filtres, j'utilise la méthode "GET", quand un utilisateur va mettre un filtre, il s'affiche dans l'url et grâce à une boucle, je vais récupérer l'ensemble des informations contenu dans l'url et construirai donc ma requête avec ceci.

Dans "\$cond" je stocke le nom du filtre, donc je dis que s'il est vide ou égale à Tout / Toutes, il n'en prend pas compte.

```
foreach ($_GET as $get => $val){
    $cond = $val;
    if($cond != "" && $cond != "Toutes_les_régions" && $cond != "Toutes" && $cond != "Tout")
    {
        if($get == "type_attestation")
        {
            $val = str_replace("de", " ", $val);
            $val = str_replace("moins", "-", $val);
            $val = str_replace("plus", "+", $val);
            $val = str_replace("-", ".", $val);
            $type = $get.'='.'.'.$val.'';
        }
    }
}
```

Enfin je récupère tout dans une variable "\$condition" et la met dans la requête.

"SELECT annonce.id, profil, nom, prenom, region, type_attestation, tel, email, prix, disponibilite FROM utilisateurs INNER JOIN annonce on \$condition WHERE annonce.id_utilisateur=utilisateurs.id and verif = '1' ORDER BY date_annonce DESC LIMIT \$limit OFFSET \$offset"

Ici le offset et limit correspondent au nombre d'annonce affichées par page, \$limit = 16. J'ai mis en place un système de pagination afin de naviguer entre les différentes annonces.

Ensuite, une fois les annonces récupérées, il me faut les afficher, alors pour ça j'ai effectué une boucle while et j'ai affiché le tout dans des div.

```

while($data = mysqli_fetch_array($execute_req_annonces))
{
    ?>
    <div class="liste_profil Marchandises - 3.5T" id="<?php echo $data['id'];?>">
    <img class="image_profil rounded-circle" id="<?php echo $data['id'];?>" src="<?php echo "img/profil/" . $data['profil'];?>" width="125">
    <div id="name_24"><?php echo $data['nom'] . ' ' . $data['prenom'];?></div>
    <div id="region_24"><?php echo $data['region']; ?></div>
    <div class="attestation" id="attestation_24"><?php echo $data['type_attestation'];?></div>
    <div style="font-weight: bold;" id="tarif_24"><?php echo $data['prix'] . ' €';?></div>
    <div class="<?php echo str_replace(' ', '', $data['disponibilite']);?>" id="dispo_24"><?php echo $data['disponibilite'];?></div>

```

Maintenant que les annonces sont visibles sur le site, d'autres utilisateurs peuvent les consulter en cliquant dessus et seront redirigés vers une autre page. Ils retrouveront alors toutes les informations de l'annonce en question récupérer grâce à un \$_GET['id'], id que je stocke dans un input de type hidden pour chaque annonce.

Depuis cette page, il est également possible d'accéder au profil de la personne où on y verra toutes ses annonces postées, sa date d'inscription... De plus si l'utilisateur est intéressé, il pourra soit contacter par téléphone, soit directement depuis la messagerie du site.

De plus chaque utilisateur possède une page "paramètre" dans laquelle ils pourront modifier leurs informations personnelles. (nom, prénom, mot de passe, photo de profil...).

Pour résumer, les utilisateur peuvent :

- S'inscrire / se connecter
- Déposer une annonce
- Consulter les annonces en ligne
- Consulter les profils utilisateurs
- Envoyer un message via la messagerie du site
- Mettre des annonces en favoris
- Consulter ses annonces postées et/ou les supprimer
- Consulter ses annonces en favoris et/ou les supprimer

Maintenant en ce qui concerne la partie admin, il est possible d'y accéder en ajoutant dans l'url "/gt-admin", gt = gestionnaire transport. Une fois arriver sur la page d'administration, il sera demandé un mot de passe.

Une fois connecté l'administrateur pourra :

- Valider les annonces en attente
- Consulter les profils des utilisateurs
- Modifier ses informations (login, mot de passe)
- Ajouter des services et/ou en supprimer
- Visualiser les 50 derniers paiements effectués et/ou les rembourser

Pour la partie concernant les services, j'ai utilisé l'api Stripe afin de communiquer avec celui-ci. Stripe possède une documentation très complète ce qui simplifie grandement la tâche. Tout d'abord il faut choisir quel langage utilisé, plusieurs sont proposés (PHP, Node.js, JAVA..). J'ai donc choisi le PHP et il m'a fallu télécharger un dossier via GitHub (le lien était donné vers celui-ci).

Enfin ma première tâche était de créer un produit, j'ai donc consulté la doc et j'ai trouvé ceci :

```
$stripe = new \Stripe\StripeClient(
    'sk_test_51H2BsNKouFi2buoUWmyNEiPXAfPA7hNDpZH6vbsPJ8cWRnWESRXEZCR8p1qSUS1RdEKq
    35YnFy68eZ4fygBYmSUf00kTzuIVfe');

$stripe->products->create([

    'name' => 'nom du produit',
]);
```

Le "sk_test" correspond à la clé de test donné lors de l'inscription à Stripe, il faut la renseigner à chaque fois qu'on utilise "/Stripe/StripeClient". Après il faut tout simplement

appelé “products->create” et renseigner le nom du produit (obligatoire) mais il est également possible de renseigner une description et des metadata qui seront des données en plus attachées au produit et d’autres paramètres qui me seront pas utiles ici. Une fois un produit créé, Stripe renvoie le résultat sous forme de tableau json comme ceci :

```
{
  "id": "prod_HeOI743151EB7A",
  "object": "product",
  "active": true,
  "attributes": [],
  "created": 1594801689,
  "description": "Met en tête de liste votre annonce pendant 60 jours.",
  "images": [],
  "livemode": false,
  "metadata": {
    "categorie": "Annonce"
  },
  "name": "nom produit",
  "statement_descriptor": null,
  "type": "service",
  "unit_label": null,
  "updated": 1594801689
}
```

Quand je créer un produit, je renseigne dans le metadata la catégorie de celui-ci. Pour le moment, il y en a qu’un qui est “Annonce”, ce sont les services qui seront proposés lors du dépôt d’annonce et il est possible de renseigner la durée de celui-ci (7, 30 ou 60 jours), données que j’enregistre dans ma base de données comme le prix. Stripe prendra comme données le nom, la description et les metadata.

Maintenant sur la page où l’utilisateur choisit s’il veut utiliser un service, je fais une requête qui sélectionne l’ensemble des services avec la catégorie “Annonce”. Comme mes services sont stockés dans la base de données je fais simplement une requête SQL pour récupérer ceux-ci et les affiche dans une boucle.

Mise en avant (7 jours)

Met en tête de liste votre annonce pendant 7 jours

Acheter 5 €

Mise en avant (30 jours)

Met en tête de liste votre annonce pendant 30 jours.

Acheter 10 €

Mise en avant (60 jours)

Met en tête de liste votre annonce pendant 60 jours.

Acheter 15 €

Déposer mon annonce sans avantages

Mise en avant (7 jours)

Met en tête de liste votre annonce pe...

✕

✉ Adresse e-mail

🗑 Numéro de carte

📅 MM / AA

🔒 CVV

☐ Se souvenir de moi

Payer 5,00 €

Si l'utilisateur choisit de déposer sans avantages, l'annonce est déposée normalement sinon s'il choisit un avantage, il lui sera demandé de payer et une fois le paiement validé, je vais enregistrer dans la table avantage l'id utilisateur ainsi que les informations de l'avantage qu'il a acheté.

Sur la page d'accueil, je fais une requête qui permettra de sélectionner les id des annonces possédant des avantages et selon la date de l'avantage, sa durée et la date actuelle, je vais update la date de façon à la remettre à jour pour que l'annonce reste en

tête de liste (date étant différente de la date qui sert à déterminer le temps de l'annonce qui au bout de 60 jours se retrouve supprimée).

```
date_default_timezone_set('Europe/Paris');
$now = new DateTime();
$date=$now->format('Y-m-d');

//DELETE ANNONCE > 60 JOURS
$delete_annonce = "DELETE from annonce where DATEDIFF('$date' , date_validite) > 60";
mysqli_query($base, $delete_annonce);

// AVANTAGE BOOST ANNONCE (7, 30 ou 60 jours)
$select_avantage = "SELECT date_avantage, duree, id_service FROM avantage INNER JOIN services on id_service = services.id";
$execute_req_avantage = mysqli_query($base, $select_avantage);
$nombre_avantage = mysqli_num_rows($execute_req_avantage);

if($nombre_avantage != 0)
{
    while($resultat_select_avantage = mysqli_fetch_array($execute_req_avantage))
    {
        $duree = $resultat_select_avantage['duree'];
        $id_service = $resultat_select_avantage['id_service'];
        $delete_avantage = "DELETE from avantage where DATEDIFF('$date', date_avantage) > '$duree' and id_service = '$id_service'";
        mysqli_query($base, $delete_avantage);
    }
}

// MET DATE A JOUR POUR ANNONCES QUI ONT DES AVANTAGES POUR METTRE EN TETE DE LISTE
$select_annonce_avantage = "SELECT annonce.id FROM annonce INNER JOIN avantage on annonce.id = id_annonce";
$execute_annonce_avantage = mysqli_query($base, $select_annonce_avantage);
$nombre_annonce = mysqli_num_rows($execute_annonce_avantage);
```

De plus il est possible par l'admin de pouvoir voir comme dit précédemment les 50 derniers paiements effectués et de les rembourser si nécessaires :

```
$stripe=new\Stripe\StripeClient(
```

```
'sk_test_51H2BsNKouFi2buoUWmyNEiPXAfPA7hNDpZH6vbsPJ8cWRnWESRXEZCR8p1qSUS1RdEKq35YnFy68eZ4fygBYmSUf00kTzuIVfe');
```

```
$stripe->charges->all(['limit' => 50]);
```

Et pour le remboursement :

```
$stripe->refunds->create([
```

```
    'charge' => 'ch_1H4kXLKouFi2buoUqCG9fEiA',
]);
```

4.2 Jeu d'essai

Je vais vous présenter un jeu d'essai réaliser sur une fonctionnalité qui pour moi est essentiel au fonctionnement du site. Cette fonctionnalité est l'ajout d'annonce par les utilisateurs inscrits.

Jeu d'essai (Dépôt d'une annonce)

Ordre	Scénarios	Cas de test : valeurs	Cas de test : résultats attendus
1	Scénario nominal	Type d'attestation Région Description Attestation format pdf Disponibilité Statut Téléphone Tarif	Retour page d'accueil si annonce bien déposée
2	Scénario d'exception n°1 <i>Description</i>	Type d'attestation Région Description Attestation format pdf Disponibilité Statut Téléphone Tarif	Description trop courte < 150 caractères, espaces non compris.
3	Scénario d'exception n°2 <i>Document</i>	Type d'attestation /Région Description / Attestation format pdf Disponibilité / Statut Téléphone / Tarif	Document pdf non valide ou non renseigné
4	Scénario d'exception n°3 <i>Téléphone</i>	Type d'attestation Région Description Attestation format pdf Disponibilité Statut Téléphone Tarif	Numéro de téléphone non valide

4.3 Veilles et sécurité

Concernant la sécurité, il est tout d'abord important de vérifier les données rentrées par l'utilisateur dans les formulaires, que ce soit pour s'inscrire, se connecter ou déposer une annonce. Car ses données sont ensuite récupérées et insérées dans la base de données, il faut donc faire attention à ce qui est récupéré. De plus, ceci doit se faire sans perturber la navigation au sein du site.

C'est pourquoi j'utilise JavaScript afin de vérifier que tous les champs soient bien remplis dans un premier temps.

Côté utilisateur :



CONNEXION

*Des champs sont vides

Login

Mot de passe

VALIDER

Besoin d'un compte ? INSCRIVEZ-VOUS

Mot de passe oublié ? CLIQUEZ ICI

Cette erreur correspond aux champs non renseignés par l'utilisateur.

Côté code :

```
if ($("#login").val() != "" && $("#password").val() != "")
{
    login = ($("#login").val())
    password = ($("#password").val())
    $.ajax({
        url: 'fonctions/fonction_connexion.php',
        type: 'POST',
        data: {login: login, password: password},

        success: function(data){
            data=data.trim()
            if(data == "erreur")
            {
                $(".champ_vide").remove()
                $(".login100-form-title").after('<div class="champ_vide">*Login ou mot de passe incorrect</div>')
            }
            else
            {
                document.location.reload(true);
            }
        }
    });
}
else
{
    $('#login').css({"border":"1px solid #C0392B"})
    $('#password').css({"border":"1px solid #C0392B"})
    $(".champ_vide").remove()
    $(".login100-form-title").after('<div class="champ_vide">*Des champs sont vides</div>')
```

De plus, côté inscription, lorsqu'un utilisateur renseigne son login et son email, je vérifie si ceux-là n'existent pas déjà dans ma base de données car ils doivent être unique à chaque utilisateur. Concernant l'email je regarde grâce à une fonction si celle-ci possède le bon format "...@... .fr/.com". Sinon j'indique à l'utilisateur que le format n'est pas le bon. De même en ce qui concerne le numéro de téléphone (les deux formats acceptés étant 06 et 07).

Une fois tous les champs requis complétés et validés par mon JavaScript je les envoie via Ajax à mon fichier php qui lui va exécuter les requêtes SQL. Pour ajouter encore une couche de sécurité, j'utilise plusieurs méthodes. Tout d'abord, j'utilise le `htmlspecialchars` qui va convertir les caractères spéciaux en entités HTML.

Je l'effectue pour toutes mes variables :

```
// SECURITE
$genre      = trim(htmlspecialchars($genre));
$nom        = trim(htmlspecialchars($nom));
$prenom     = trim(htmlspecialchars($prenom));
$email      = trim(htmlspecialchars($email));
$naissance  = trim(htmlspecialchars($naissance));
$login      = trim(htmlspecialchars($login));
$password1  = trim(htmlspecialchars($password1));
$password2  = trim(htmlspecialchars($password2));
```

Et j'effectue mon htmlspecialchars dans un "trim" qui va lui enlever tous les espaces de ma chaîne de caractères. De plus pour le login je regarde si celui-ci ne possède pas de caractères spéciaux, pour cela j'ai rentré tous les caractères spéciaux non acceptés dans un tableau, puis je compare simplement ma variable au tableau afin de savoir s'il y a des similitudes, s'il y en a, c'est que ma variable possède un caractère spécial, dans ce cas je retourne une erreur.

En parallèle, je regarde si les mots de passes correspondent lors de l'inscription (mot de passe et confirmation mot de passe), si la taille du mot de passe est supérieure ou égale à 8 (pour plus de sécurité) et si le login lui est supérieur ou égale à 5 caractères.

Si toutes les conditions sont remplies, je vais crypter le mot de passe de l'utilisateur afin de stocker dans la base de données un mot de passe que je ne pourrai pas voir moi-même, et que personne pourra voir non plus. Pour cela j'utilise la fonction "password_hash" qui va donc créer une clé de hachage pour le mot de passe d'une longueur de 60 caractères et qui ressemblera à ceci :

"\$2y\$12\$h16.bpWoWCkgn1UU.Sd4eAO9xNN5so2ihRDyYYp7ZG8slJwKIMK"

Après afin de vérifier la correspondance entre un mot de passe crypté et un mot de passe renseigné par l'utilisateur lors d'une connexion, j'utilise la fonction "password_verify".

4.4 Difficultés rencontrées

Les délais

Tout d'abord, ce que j'ai trouvé assez difficile pour ce projet était d'établir des dates pour la réalisation de celui-ci car c'était la première fois que je réalisais un projet de cette taille avec des fonctionnalités que je n'avais jamais mises en place et donc qui était inconnu pour moi (système de paiement, de remboursement, mise en tête de liste des annonces...).

Il était donc assez difficile pour moi d'élaborer des dates précises de début / fin pour chaque tâche. J'ai donc réalisé mes tâches une par une sans avoir spécialement de délai pour chacune d'elle.

De plus, par le confinement, il m'a fallu trouver dans les premiers temps une autre façon de m'organiser et de trouver la motivation d'avancer dans le projet. Car en plus du projet professionnel, je devais également réaliser des projets à côté pour l'école car j'apprenais encore de nouveaux langages au début de celui-ci (JavaScript / jQuery / json).

Malgré ça, j'ai quand même réussi à trouver la motivation et à suivre un certain rythme afin d'avancer le plus possible dans mes projets, un jour sur 2 je travaillais sur les projets d'école et l'autre jour, projet professionnel.

Difficultés niveau techniques

Lors de ce projet, comme dis précédemment, j'ai mis en place un bon nombre de fonctionnalités dont certaines m'ont posés de nombreux problèmes, de nombreuses difficultés. Par exemple, pour mettre en place le système de paiement, cela était totalement nouveaux pour moi, je ne connaissais pas du tout Stripe avant cela.

J'ai donc suivi la documentation de l'api tout en anglais expliquant comment créer des produits, des paiements, remboursement...

Extrait de la documentation en anglais, (ici pour créer un produit) :

Create a product

Creates a new product object.

Parameters

id optional

An identifier will be randomly generated by Stripe. You can optionally override this ID, but the ID must be unique across all products in your Stripe account.

name **REQUIRED**

The product's name, meant to be displayable to the customer. Whenever this product is sold via a subscription, name will show up on associated invoice line item descriptions.

active optional

Whether the product is currently available for purchase. Defaults to `true`.

description optional

The product's description, meant to be displayable to the customer. Use this field to optionally store a long form explanation of the product being sold for your own rendering purposes.

metadata optional associative array

Set of **key-value pairs** that you can attach to an object. This can be useful for storing additional information about the object in a structured format. Individual keys can be unset by posting an empty value to them. All keys can be unset by posting an empty value to `metadata`.

```
POST /v1/products

1 $stripe = new \Stripe\StripeClient(
2   'sk_test_51H2BsNKouFi2buoUmyNEiPXAfPA7hNDpZH6vbsP38cWnWESRxEZCR8'
3 );
4 $stripe->products->create([
5   'name' => 'Gold Special',
6 ]);
```

```
RESPONSE

{
  "id": "prod_He01743151EB7A",
  "object": "product",
  "active": true,
  "attributes": [],
  "created": 1594801689,
  "description": "Met en tête de liste votre annonce pendant 60 jours.",
  "images": [],
  "livemode": false,
  "metadata": {
    "categorie": "Annonce"
  },
  "name": "Gold Special",
  "statement_descriptor": null,
  "type": "service",
  "unit_label": null,
  "updated": 1594801689
}
```

La documentation indique les paramètres à rentrer lors de la création d'un produit, tous les paramètres sont optionnels sauf le nom qui lui est obligatoire.

Dans les paramètres on y trouve :

- L'id qui est l'identifiant du produit, si non renseigné, Stripe en génère un automatiquement.
- Le nom du produit (obligatoire)
- Active, pour savoir si le produit est disponible ou non à l'achat, par défaut à "oui"
- La description du produit afin de décrire le produit renseigné pour que les utilisateurs puissent la voir.
- Metadata, un ensemble de données que l'on peut attacher à un objet, ce sont des informations supplémentaires comme par exemple pour mon projet la catégorie des services ajoutés.

D'autres paramètres sont disponibles mais je ne les ai pas tous énumérés.

De plus, lorsque j'avais des difficultés à réaliser certaines choses, je me rendais souvent sur <https://stackoverflow.com/> qui répond souvent à mes questions.

Conclusion

J'étais vraiment motivé à l'idée de réaliser un projet que je trouvais en plus de ça vraiment très intéressant. Nous avons 2 propositions de projet et c'est celui-ci que nous avons choisis (le deuxième était un site e-commerce de ventes de sacs pour ordinateurs).


Cependant, je ne me rendais pas compte ce que cela impliquerait en terme de temps et j'ai passé beaucoup de temps à coder sur ce projet, à l'école comme à la maison, la semaine comme le week-end.


Finalement, ce projet m'a dans un premier temps permis de mettre en pratique mes acquis que j'ai pu avoir durant mon année de formation et de les consolider. De plus, j'ai pu apprendre à travailler avec un certain rythme et une certaine assiduité. La réalisation de ce projet permet de savoir à quoi s'attendre en ce qui concerne le monde du travail car il fallait créer un site en fonction d'une demande d'un client, d'un temps donné (4 à 6 mois) et donc d'un cahier des charges, c'était donc très formateur.

Annexes

1 - Maquette site web Gestionnaire transport réalisé avec PhotoFiltre 7 (quelques exemples)

Accueil / Messagerie personnelle du site

[Déposer une annonce](#) [Accueil](#) [Contact](#) [A propos](#) 




Type d'attention
Commissionnaire
Marchandises + 3.5T
Marchandises
Voyageurs

Région
Auvergne - Rhône
- alpes
Auvergne-Rhône
- alpes
Bretagne
Centre-Val de Loire
Hauts-de-France
Île-de-France
La Réunion


Disponibilité
Immédiate
Sous 3 mois

Prix max
[XX Résultats](#)


Bandeau de pub




GT Stéphanie
Lyon
Capacité +3.5T
07.67.40.48.28.
contact@global-
pa.fr




GT Fabien
Lyon
Capacité +3.5T
07.67.40.48.28.
contact@global-
pa.fr




GT Fabien
Lyon
Capacité +3.5T
07.67.40.48.28.
contact@global-
pa.fr




GT Stéphanie
Lyon
Capacité +3.5T
07.67.40.48.28.
contact@global-
pa.fr



GT Fabien
Lyon
Capacité +3.5T
07.67.40.48.28.
contact@global-
pa.fr










GT Fabien
Lyon
Capacité +3.5T
07.67.40.48.28.
contact@global-
pa.fr




Global-PA

[Conditions d'utilisations](#) [Contactez-nous](#)

[Accueil](#) [Contact](#) [A propos](#) 

	Nom utilisateur
Utilisateur 1	
Utilisateur 2	
Utilisateur 3	
Utilisateur 4	
	
	
<div>Ecrivez votre message Envoyer</div>	



Global-PA

[Conditions d'utilisations](#) [Contactez-nous](#)

Ajout d'annonce / paramètre utilisateur

Accueil Contact A propos

Nom : Prénom :

Pseudo : Date de naissance :

Adresse : Email :

Tel : Photo :

SUIVANT

Après validation

Type de capacité Tarif - A négocier

Lieu d'action

Descriptif de parcours / expériences / objectifs

SUIVANT

Après validation

Possibilité de faire payer pour remonter de liste

SUIVANT

Global-PA

Conditions d'utilisations Contactez-nous

Accueil A propos Contact

Informations personnelles

Mes documents

E-mail

Mot de passe

Informations modifiables données lors de l'inscription

Attestations

E-mail

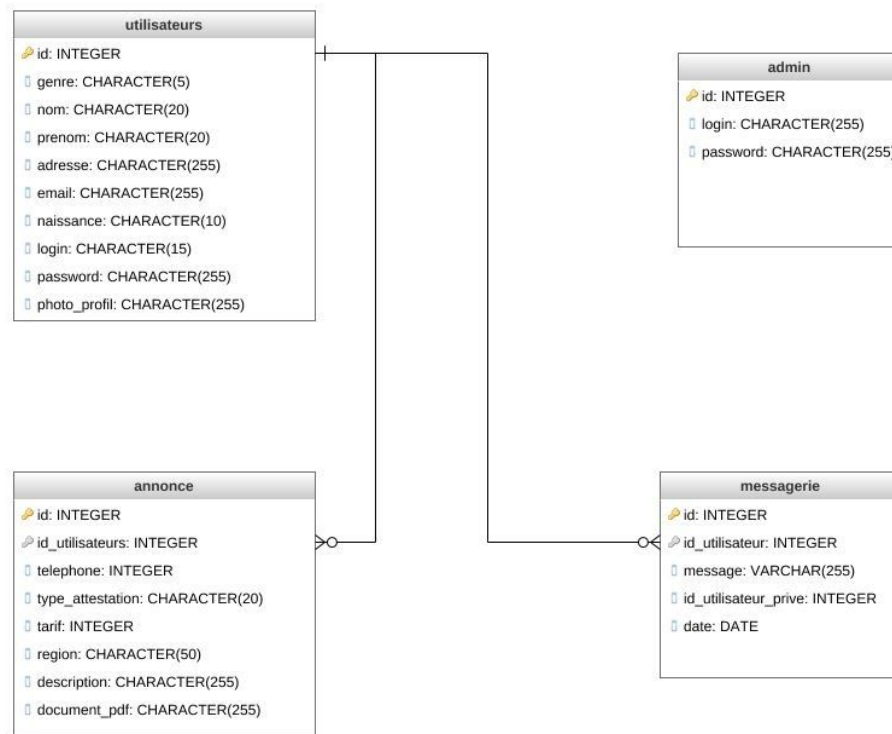
Mot de passe

Global-PA

Conditions d'utilisations Contactez-nous

2 - Maquette de la base de données avec GenMyModel

Ceci est la base de données conçu au début, avec l'avancement du projet, j'ai rajouté par la suite de nouvelles tables (services, ban, avantages...).



3 - Extrait de code

Upload d'une image

```
<?php
session_start();
require "config.php";

$uploadOk = 1;
$valid_extensions = array("jpg","jpeg","png");
$imageFileType = pathinfo($_FILES['file']['name'],PATHINFO_EXTENSION);

if(!in_array(strtolower($imageFileType),$valid_extensions) ) {

    echo "wrong_format";
}
else
{
    $uploadOk = 0;
    $name = sha1(session_id().microtime().'.jpg');
    $filename=$name.".jpg";
    $location = "../img/profil/".$filename;

    move_uploaded_file($_FILES['file']['tmp_name'],$location);
    echo "uploadOk";
}

// SELECT ID
$req_id="SELECT id FROM utilisateurs ORDER BY id DESC";
$execute_req_id=mysqli_query($base, $req_id);
$result_req_id=mysqli_fetch_array($execute_req_id);
$id = $result_req_id['id'];

// INSERT PHOTO DE PROFIL
$insert_image = "UPDATE utilisateurs SET profil = '$filename' WHERE id = '$id'";
mysqli_query($base, $insert_image);
?>
```

Messagerie en JavaScript (affiche les messages en fonction de l'utilisateur)

```
var timeout = 1500;
function affichages_messages(){

    var conv_user = $('.active_chat').attr('id').substr(10)

    $.ajax({
        url: '../fonctions/fonction_messages.php',
        type: 'POST',
        data: {conv_user: conv_user},

        success: function(data){
            if(document.getElementsByClassName('outgoing_msg').length + document.getElementsByClassName('incoming_msg').length > JSON.parse(data).length)
            {
                $(".outgoing_msg").remove()
                $(".incoming_msg").remove()
            }
            for(i=0; i < JSON.parse(data).length; i++)
            {
                var result = JSON.parse(data)[i];
                for(j=0; j < Object.keys(result).length; j++)
                {
                    var id = Object.keys(result)[0]
                    var login = Object.keys(result)[1]
                    var profil = Object.keys(result)[2]
                    var message = Object.keys(result)[3]
                    var date = Object.keys(result)[4]
                    var user_on = Object.keys(result)[5]
                }
                if(result[login] === result[user_on] && $('#outgoing_msg_'+result[id]).length === 0)
                {
                    $(".msg_history").append('<div id="outgoing_msg_'+result[id]+'"' class="outgoing_msg"></div>')
                    $('#outgoing_msg_'+result[id]).append('<div id="sent_msg_'+result[id]+'"' class="sent_msg"></div>')
                    $('#sent_msg_'+result[id]).append('<p id="message_'+result[id]+'"'>'+result[message]+'</p>')
                    $('#message_'+result[id]).after('<span id="time_date_'+result[id]+'"' class="time_date">'+result[date]+'</span>')
                    $('#time_date_'+result[id]).append('<svg id="svg_'+result[id]+'"' class="bi bi-trash" width="1em" height="1em" viewBox="0 0 16 16" fill=
                    var div = $('.msg_history');
                    var height = div[0].scrollHeight;
                    div.scrollTop(height);
                }
                else if(result[login] !== result[user_on] && $('#incoming_msg_'+result[id]).length === 0)
                {
                    $(".msg_history").append('<div id="incoming_msg_'+result[id]+'"' class="incoming_msg"></div>')
                    $('#incoming_msg_'+result[id]).append('<div id="incoming_msg_img_'+result[id]+'"' class="incoming_msg_img"></div>')
                    $('#incoming_msg_img_'+result[id]).append('')
                    $('#message_'+result[id]).after('<span id="time_date_'+result[id]+'"' class="time_date">'+result[date]+'</span>')
                    $('#incoming_msg_img_'+result[id]).after('<div id="received_msg_'+result[id]+'"' class="received_msg"></div>')
                    $('#received_msg_'+result[id]).append('<div id="received_withd_msg_'+result[id]+'"' class="received_withd_msg"></div>')
                    $('#received_withd_msg_'+result[id]).append('<p id="message_'+result[id]+'"'>'+result[message]+'</p>')
                    $('#time_date_'+result[id]).after('<span class="time_date">'+result[date]+'</span>')
                    var div = $('.msg_history');
                    var height = div[0].scrollHeight;
                    div.scrollTop(height);
                }
            }
        }
    });
    setTimeout(affichages_messages, timeout);
};
```



Salut

2020-06-04 15:00:00



ça va ?

2020-06-04 16:00:00

cc

2020-06-15 16:07:23



Type a message

Envoyer