

RAPPORT DE STAGE

Développeur web

FENEROL Miguel

9 janvier 2023 – 17 février 2023

Tuteur : Monsieur SCHNEKENBURGER
Professeur référent : Monsieur GAULIER

Etablissement : Lycée Saint Adjutor de Vernon – BTS SIO 2^{ème} année
Association : Action à Réaction Vernon

Sommaire :

Remerciements :	4
Présentation de l'entreprise :	4
Présentation du service et des moyens informatiques :	5
Présentation du projet :	6
Développement du projet :	6
Les outils utilisés :	6
Formation à la technologie Angular :	7
La création d'un projet :	7
Attribution des tâches :	8
La première base de données :	8
La création de comptes partenaires :	9
La création de comptes annonceurs :	18
L'espace partenaire :	19
Gestion du header :	22

Remerciements :

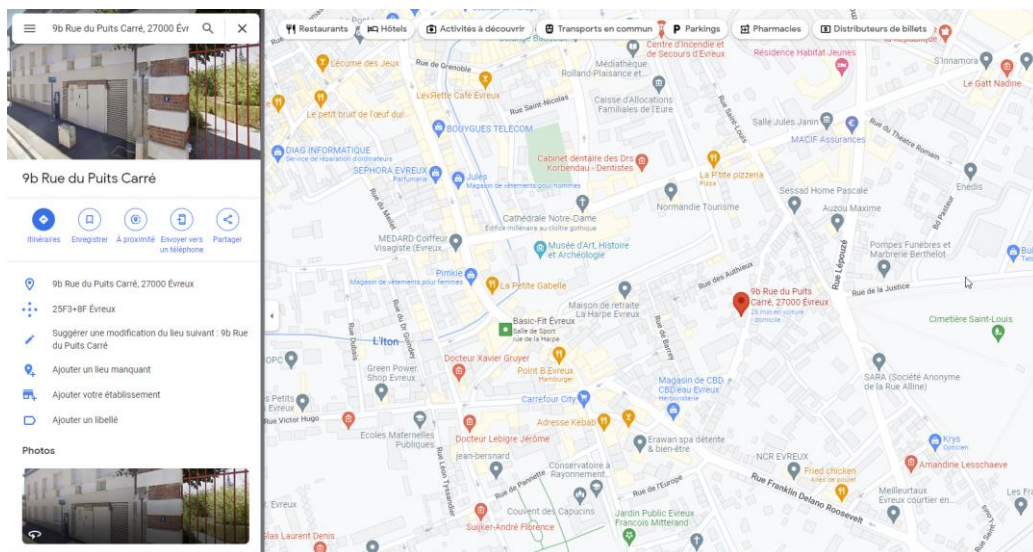
Je tiens tout particulièrement à remercier M. SCHNEKENBURGER de nous avoir accueilli pour ce stage et de nous avoir fourni un véritable projet à mener. Je le remercie également d'avoir pris le temps de s'occuper de nous et de nous avoir appris beaucoup de choses.

Je tiens également à remercier tous mes camarades qui m'ont accompagné lors de ce stage et avec qui le stage s'est bien déroulé.

Présentation de l'entreprise :

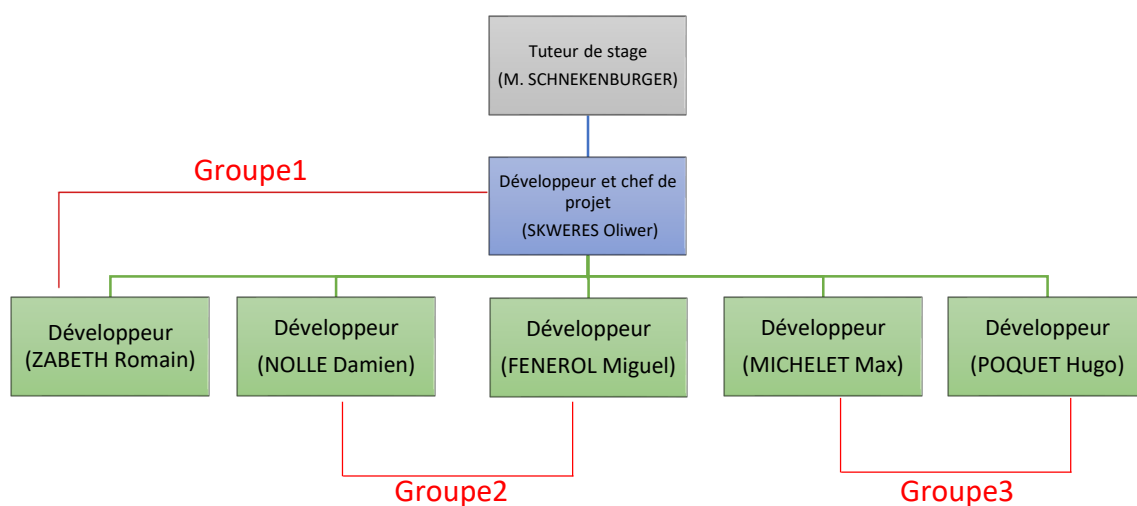
Altameos Multimedia est une entreprise de développement Web offrant ses services aux entreprises voulant concevoir leur site Web afin d'améliorer leurs visibilité.

Fondé par Monsieur SCHNEKENBURGER en 2003, l'entreprise se situe à Evreux (27000) au 9b Rue du Puits Carré.



L'entreprise est composée uniquement de Monsieur SCHNEKENBURGER car il travaille en tant qu'indépendant libéral.

Organigramme de Altameos Multimedia et de notre équipe :



Présentation du service et des moyens informatiques :

J'ai utilisé mon ordinateur personnel pour la réalisation de ce stage, voici les principales caractéristiques techniques.



Produit :	OMEN by HP 15-ce047nf
Microprocesseur :	Processeur Intel® Core™ i5-7300HQ (2,5 GHz de fréquence de base, jusqu'à 3,5 GHz avec technologie Intel® Turbo Boost, 6 Mo de mémoire cache, 4 cœurs)
Chipset	Intel® HM175
Mémoire :	8 Go de mémoire SDRAM DDR4-2400 (1 x 8 Go)
Carte graphique	Carte NVIDIA® GeForce® GTX 1060 (6 Go de mémoire GDDR5 dédiée)
Disque dur	SATA 1 To, 7200 tr/min
Écran	15.6" diagonal FHD 120 Hz IPS anti-glare WLED-backlit (1920 x 1080)

Présentation du projet :






L'objectif de ce stage est de créer un site marketplace de zéro sous Angular. Un site marketplace est un site permettant à des personnes de déposer des produits en ligne qui seront proposés à la vente aux clients. Un exemple très commun serait le site de la société Amazon. Cela se traduit par la création d'une base de données adaptée à la demande, la mise en place d'une interface graphique. De plus, il faudra gérer les différents stocks, les différents types de compte avec différents droits ainsi que les produits affichés en fonction de ces derniers éléments. En tant que site dédié à la vente, il faut aussi un moyen de paiement sécurisé et un système de génération de factures ainsi qu'une gestion des achats pour les colis.

Plus en détails dans l'organisation du site, celle-ci est composée de différents types de comptes, à savoir :

- Super Administrateur, qui est l'administrateur global du site.
- Administrateur boutique, qui s'occupe de la modération et de la gestion de la boutique en ligne.
- Administrateur Logistique, qui s'occupe qui s'occupe des commandes et des livraisons.
- Partenaire, les entreprises partenaires du site.
- Annonceur, les personnes qui vont mettre les produits en ligne pour les vendre.
- Client, les potentiels acheteurs.

Développement du projet :

Les outils utilisés :

Logiciels / Technologies	Fonctions
	Visual Studio Code en tant qu'éditeur
   	Html5 et css3 Angular -> Framework openSource basé sur le TypeScript Angular Material -> Bibliothèque de composant utilisateurs pour Angular
 	MySQL -> gestionnaire de base de données phpMyAdmin -> application visuelle pour MySQL
 	GitHub pour stocker le projet et GitKraken comme outil de versioning basé sur le Git.
	Google chrome comme moteur de recherche et navigateur de test
 	Discord et Microsoft Teams comme moyens de communication et de partage de fichiers.
	Trello, outil en ligne de suivi de tâches
 POSTMAN	Logiciel de test d'API

Formation à la technologie Angular :

Une fois la présentation globale du projet faite par notre tuteur, une semaine d'apprentissage sur la technologie Angular nous a été allouée. Pendant celle-ci, nous étions libres de la méthode d'apprentissage ainsi que le cours en ligne à suivre. Pour ma part, j'ai suivi le tutoriel d'*OpenClassroom* sur les bases d'Angular. Celui-ci m'a permis d'apprendre les commandes consoles de base pour créer et démarrer un projet ainsi que la gestion des composants qui sont la base du fonctionnement d'Angular. Ce tutoriel nous apprend également le système de routing d'Angular qui permet de changer de page sur le site sans causer le rechargement de l'application. Un component est une sorte de bloc que l'on peut appeler quand on veut sur l'application à l'intérieur d'un autre ou bien y diriger l'utilisateur via le système de routing. Une fois ces bases maîtrisées, nous nous sommes formés sur la partie CRUD/API qui est essentiel quant à la concrétisation du projet.

La création d'un projet :

Pour pouvoir créer un projet Angular, il faut déjà avoir NodeJs d'installer sur son ordinateur, ensuite il faut entrer la commande suivante depuis son terminal.

```
npm i -g @angular/cli
```

Ensuite il faut créer un projet, cela se fait via la commande suivante :

```
D:\Angular\marketplace>ng new nomProjet
```

Une fois cela fait, on peut démarrer le projet, ce qui nous amènera à la page Angular par défaut. Pour cela :

```
D:\Angular\marketplace>ng serve --open
```

Afin de commencer à coder une application il nous faut créer des components :

```
D:\Angular\marketplace>ng generate component nomComponent
```

Commande qui peut être abrégée en :

```
D:\Angular\marketplace>ng g c nomComponent
```

Un component Angular est composé de 4 fichiers :

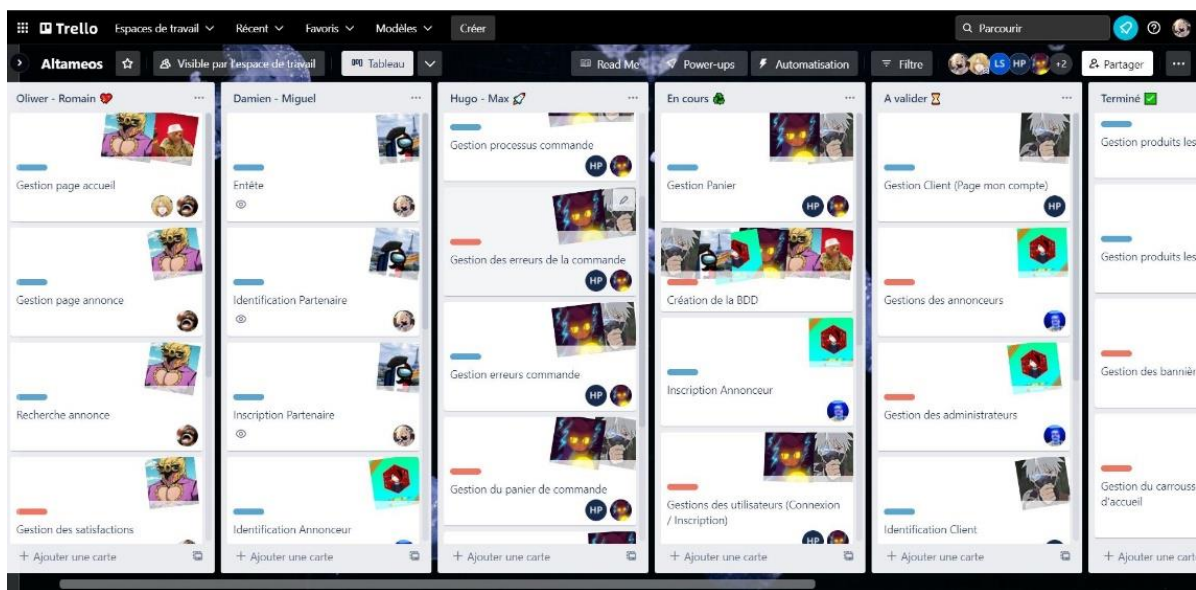
1. un fichier *nomComponent.component.html* qui comportera le code HTML de la page.
2. un fichier *nomComponent.component.scss* qui sera le style du component.
3. un fichier *nomComponent.component.spec.ts* qui comporte des tests unitaires.
4. un fichier *nomComponent.component.ts* qui contient le typeScript du component.

Attribution des tâches :

La semaine de formation étant presque achevée, nous avons étudié le projet ainsi que commencé la réflexion d'une base de données répondant aux besoins énoncés afin de pouvoir faire l'attribution des tâches. Celle-ci s'est faite sous l'attention de notre tuteur pour qu'il soit au courant de qui fait quoi. Chaque groupe a choisi ses tâches puis se les sont repartis entre les membres de ceux-ci.

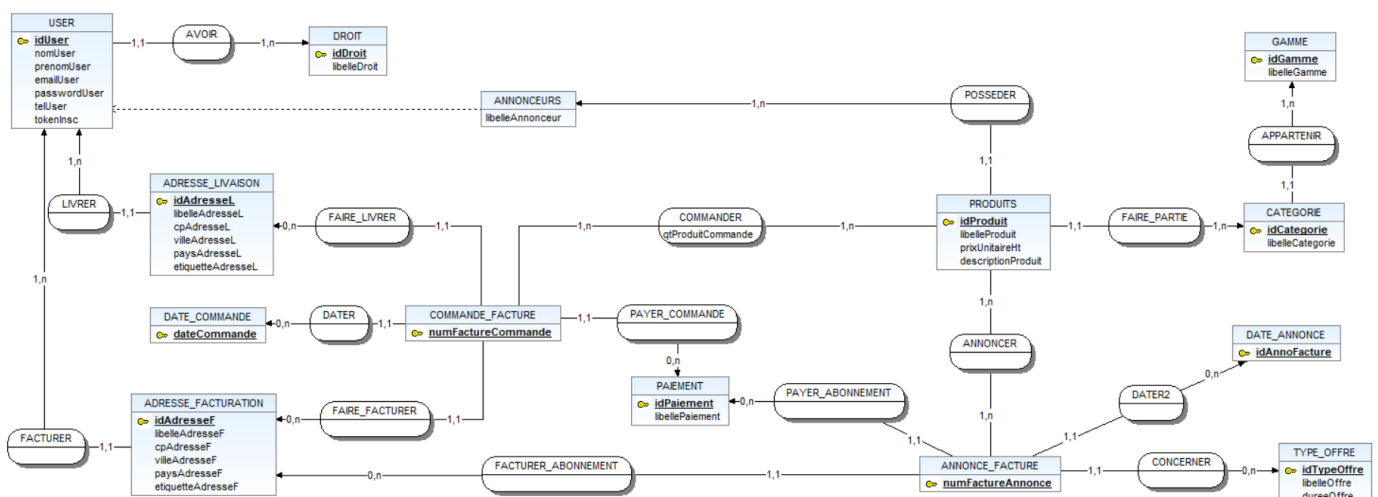
A titre personnel, je me suis vu attribué la création de comptes partenaires et de comptes annonceurs, la création de l'espace dédié aux partenaires. Ainsi que la gestion du header et du footer du site.

La gestion des tâches a été effectuée via l'outil Trello disponible en ligne, elle se présente de la manière suivante :



La première base de données :

Avant de commencer le projet, nous avons monté un début de base de données dont le MCD est ci-dessous.




On peut voir qu'on a une table USER qui est relié à une table DROITS qui nous permet de différencier les différents comptes. On a un système de produits liés à des catégories elles-mêmes liées à des gammes. Il y a également des tables liées aux adresses pour la facturation. Cette base se verra évoluer de nombreuses fois mais le fonctionnement global restera le même.

La création de comptes partenaires :

Cette première tâche, m'a pris pas mal de temps, ne serait-ce que par la découverte des formulaires sous Angular ainsi que le fonctionnement d'Angular material. De plus, celle-ci n'a cessé d'évoluer de par les attentes du maître de stage ainsi que de par le travail de mes collègues.

Afin de mener cette tâche à bien, j'ai commencé par créer un component qui comprendra le formulaire. Côté code HTML on a un formulaire réalisé avec des inputs d'Angular material. Cependant les partenaires ont des informations uniques qui les différencies des utilisateurs. En MCD, cela nous donne un héritage, ce qui nous fait une nouvelle table PARTENAIRES qui contiendra les informations suivantes :

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
<input type="checkbox"/> 1	idPartenaire 	int(11)			Non	Aucun(e)		AUTO_INCREMENT
<input type="checkbox"/> 2	nomSociete	varchar(45)	utf8mb4_general_ci		Oui	NULL		
<input type="checkbox"/> 3	adresseSociete	varchar(45)	utf8mb4_general_ci		Oui	NULL		
<input type="checkbox"/> 4	villeSociete	varchar(45)	utf8mb4_general_ci		Oui	NULL		
<input type="checkbox"/> 5	cpSociete	varchar(45)	utf8mb4_general_ci		Oui	NULL		
<input type="checkbox"/> 6	pays	varchar(45)	utf8mb4_general_ci		Oui	NULL		
<input type="checkbox"/> 7	logo	varchar(45)	utf8mb4_general_ci		Oui	NULL		
<input type="checkbox"/> 8	siret	varchar(14)	utf8mb4_general_ci		Non	Aucun(e)		
<input type="checkbox"/> 9	idEtatPartenaire	int(11)			Non	Aucun(e)		
<input type="checkbox"/> 10	titreFiche	text	utf8mb4_general_ci		Oui	NULL		
<input type="checkbox"/> 11	descriptionFiche	text	utf8mb4_general_ci		Oui	NULL		

Exemple de code d'un formulaire Angular avec Angular material :

```
src > app > add-partenaire > add-partenaire.component.html > form.grid
Go to component
1 <h2>Demande de compte partenaire</h2>
2 <form [formGroup]="insertForm" (ngSubmit)="verifPartenaire()" class="grid">
3   <mat-form-field class="input">
4     <mat-label>Nom</mat-label>
5     <input matInput formControlName="nom">
6   </mat-form-field>
7
8   <mat-form-field class="input">
9     <mat-label>Prénom</mat-label>
10    <input matInput formControlName="prenom">
11  </mat-form-field>
12  <br/>
13  <div></div>
14  <mat-form-field class="input">
15    <mat-label>Adresse</mat-label>
16    <input matInput formControlName="adresse">
17  </mat-form-field>
18
```

Après avoir créé tous les inputs nécessaires au formulaire, il faut acheminer les données jusqu'à la base, c'est à ce niveau que c'était moins une partie de plaisir. Il m'a fallu 3 jours pour enregistrer quelque chose de correct en base de données. Au début je ne savais tout simplement pas comment récupérer les données du formulaire. Il m'a fallu pas mal de recherche pour trouver et comprendre la solution.

Pour commencer il faut importer ces modules dans le fichier `ts` :

```
import { FormBuilder, FormControl, Validators } from '@angular/forms';
```

Ensuite il faut ajouter la caractéristique suivante aux inputs :

```
<mat-form-field class="input">
  <mat-label>Prénom</mat-label>
  <input matInput formControlName="leNom">
</mat-form-field>
```

Côté `ts`, il faut créer un `formBuilder`, pour cela il faut le construire après l'avoir importé.

```
constructor(private formBuilder: FormBuilder, private partenaireService: partenaireService){}
```

Le `formBuilder` :

```
updateForm = this.formBuilder.group({
  nom: new FormControl('', Validators.required),
  prenom: new FormControl('', Validators.required),
  adresse: new FormControl('', Validators.required),
  ville: new FormControl('', Validators.required),
  pays: new FormControl('', Validators.required),
  cp: new FormControl('', Validators.required),
  tel: new FormControl('', Validators.required),
  email: new FormControl('', Validators.required),
  nomSociete: new FormControl('', Validators.required),
  siret: new FormControl('', Validators.required)
});
```

Permet de rendre le champ obligatoire.

Avant d'arriver à ce résultat, j'avais essayé plein d'autres méthodes mais aucune ne fonctionnait à 100%. Il me fallait une méthode capable d'envoyer texte comme image à mes fichiers *php*. Le but ici était de pouvoir récupérer mes données et de les envoyer aux fichiers *php* en charge de communiquer avec la base de données. Je devais envoyer les données à une fonction contenue dans un fichier *service.ts* qui était chargé de réaliser toutes les communications avec les fichiers *php*.

Entre temps j'ai eu des problèmes avec les *CORS-policy* qui même en installant l'extension les bloquant, m'ont causé problème. Cela m'a pris une demi-journée à résoudre. Ensuite s'en est suivi d'un problème lié aux fichiers du CRUD, car une personne avait des fichiers différents et ça a créé des conflits lors des merges. Cela m'a pris une autre demi-journée à résoudre.

Voilà à quoi ressemble le formulaire de base :

S'identifier S'inscrire Mot de passe oublié ?

DEMANDE DE COMPTE PARTENAIRE

Nom* Prénom*

Adresse* Ville* Pays* Code Postal*

Téléphone* Email*

Ex : +(n° pays)X XX XX XX XX

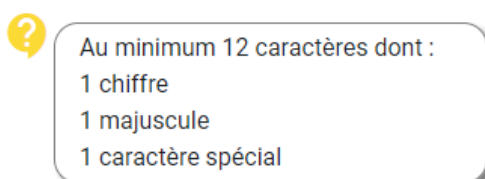
Nom de la société* SIRET*

Mot de passe* Confirmation Mot de passe*

Logo : Choisir un fichier Aucun fichier choisi Newsletter : ☐

Envoyer ma demande

On peut voir que ce formulaire comprend une insertion d'image, un bouton pour la newsletter, une *comboBox* pour les Pays. La petite bulle jaune nous donne les modalités du mot de passe lorsque l'on passe dessus.



Après consultation de ce qu'on fait les équipiers, j'ai enfin trouvé la bonne combinaison me permettant de répondre à mes besoins. J'ai utilisé un *FormData* pour stocker les données de la manière suivante :

```
const formData = new FormData();
formData.append('nom', '' + this.insertForm.value.nom);
formData.append('prenom', '' + this.insertForm.value.prenom);
formData.append('adresse', '' + this.insertForm.value.adresse);
formData.append('ville', '' + this.insertForm.value.ville);
formData.append('pays', '' + this.insertForm.value.pays);
formData.append('cp', '' + this.insertForm.value.cp);
formData.append('tel', '' + this.insertForm.value.tel);
formData.append('email', '' + this.insertForm.value.email);
formData.append('nomSociete', '' + this.insertForm.value.nomSociete);
formData.append('siret', '' + this.insertForm.value.siret);
formData.append('mdp', '' + this.insertForm.value.mdp);
formData.append('image', this.image, this.image.name);
if(''+this.insertForm.value.newsletter == 'true'){
  formData.append('newsletter', '1');
}else{
  formData.append('newsletter', '0');
}
this.partenaireService.insertPartenaire(formData, mail).subscribe((data:any)=>{});
this.router.navigate(['redirection-compte']);
```

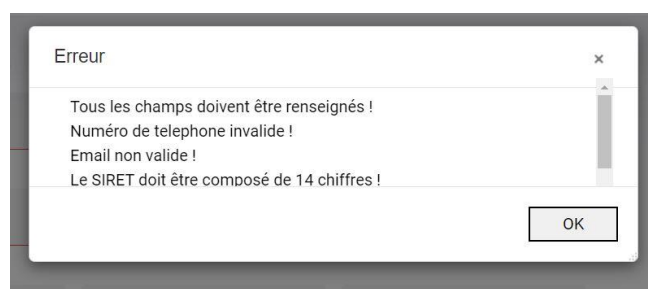
Ces données étant officielles et certaines contrôlées, il faut faire un minimum de vérifications dessus.

Par exemple le libelle du Pays qui sera renseigné via une *comboBox* pour éviter les erreurs que l'utilisateur pourrait faire. Ou encore le SIRET d'une entreprise qui doit être composé de 14 chiffres uniquement. Ainsi que respecter les conseils de le CNIL en terme de sécurité de mot de passe.

```
verifPartenaire(){
  var message = '';
  var test = true;
  if(this.insertForm.value.nom == '' || this.insertForm.value.prenom == '' || this.insertForm.value.adresse == '' || this.insertForm.value.ville == '' || this.insertForm.value.pays == '' || this.insertForm.value.cp == '' || this.insertForm.value.tel == '' || this.insertForm.value.email == '' || this.insertForm.value.nomSociete == '' || this.insertForm.value.siret == '' || this.insertForm.value.mdp == '' || this.insertForm.value.image == null){
    test = false;
    message+= 'Tous les champs doivent être renseignés !<br>';
  }
  if(!/^([0-9]{2})+([0-9]{9})$/.test(''+this.insertForm.value.tel)){
    test = false;
    message+= 'Numéro de telephone invalide !<br>';
  }
  if(!/^([a-zA-Z]{1,4})+([a-zA-Z]{1,4})+([a-zA-Z]{1,4})+([a-zA-Z]{1,4})+([a-zA-Z]{1,4})+([a-zA-Z]{1,4})+([a-zA-Z]{1,4})+([a-zA-Z]{1,4})+([a-zA-Z]{1,4})+([a-zA-Z]{1,4})+([a-zA-Z]{1,4})+([a-zA-Z]{1,4})+([a-zA-Z]{1,4})+([a-zA-Z]{1,4})$/.test(''+this.insertForm.value.email)){
    test = false;
    message+= 'Email non valide !<br>';
  }
  if(!parseInt(''+this.insertForm.value.siret) || this.insertForm.value.siret?.length != 14){
    test = false;
    message+= 'Le SIRET doit être composé de 14 chiffres !<br>';
  }
  if(this.insertForm.value.mdp != this.insertForm.value.mdp2){
    test = false;
    message+= 'Les mots de passes doivent être identiques !<br>';
  }
  if(!/^(?=.*[a-z])(?=.*[A-Z])(?=.*[!@#$%^&*])(?=.*[a-zA-Z]).{12,}$/.test(''+this.insertForm.value.mdp)){
    test = false;
    message+= 'Le mot de passe n'est pas assez fort !<br>';
  }
  if(this.image == null){
    test = false;
    message+= 'Logo non renseigné !<br>';
  }
  if(test == true){
    alertifyjs.confirm("Confirmation", "Ces informations sont-elles valides ?", ()=>{
      this.insertPartenaire();
    }, ()=>{}).set({'transition':'zoom', 'resizable': 'true', 'mouvable': 'false', 'labels':{'ok: 'Oui', cancel: 'Non'}});
  }else{
    alertifyjs.alert("Erreur ", "" + message).set({'transition':'zoom', 'resizable': 'true', 'mouvable': 'false'});
  }
}
```

Il y a également des alertes qui se modifient en fonction des erreurs commises :

Exemple d'erreurs :



Une fois que ces données sont vérifiées et mises dans le FormData, je les fais passer au service correspondant comme-ci-dessous :

```
this.partenaireService.insertPartenaire(formData, mail).subscribe((data:any)=>{});
```

Cette ligne appelle la fonction située dans le fichier *partenaire.service.ts*. Dans ce fichier, on importe le module *HttpClient* qui permet de faire des requêtes http, ainsi que le module *Injectable* afin que le service puisse fonctionner correctement. Ce module permet d'importer les dépendances nécessaires lors ce que le service est appelé plutôt que de les créer.

```
import { Injectable } from "@angular/core";
```

```
import { HttpClient } from "@angular/common/http";
```

Les fichiers *services.ts* servent à rassembler toutes les traitements que l'on veut faire sur les données de la page ainsi qu'à communiquer avec les fichiers php nous servant d'API. Dans ce cas-ci, je vais faire une requête de type POST vers mon fichier d'insert dans la base de données avec le *FormData* comprenant les données en paramètre.

```
import { Injectable } from "@angular/core";
import { HttpClient } from "@angular/common/http";

@Injectable({
  providedIn: 'root'
})
export class partenaireService{

  constructor( private http: HttpClient){}

  baseUrl: string = 'http://localhost/CRUD_marketplace/';

  insertPartenaire(partenaire: any, mail: any){
    this.http.post(this.baseUrl+'mail.php', mail).subscribe((data:any)=>{});
    return this.http.post(this.baseUrl+'insertPartenaire.php', partenaire);
  }
}
```

Comme on peut le voir sur l'image ci-dessus, la fonction *insertPartenaire* reçoit 2 paramètres, un recevant les données à insérer et l'autre les données pour un mail.

Pour ce qui est de l'insertion du partenaire :

```
C:\> wamp64 > www > CRUD_marketplace > insertPartenaire.php
1  <?php
2  error_reporting(0);
3  header("Access-Control-Allow-Origin: *");
4  header("Access-Control-Allow-Headers: access");
5  header('Access-Control-Allow-Credentials', true);
6  header('Access-Control-Allow-Methods: GET, POST, PATCH, PUT, DELETE, OPTIONS');
7  header("Content-Type: application/json; charset=UTF-8");
8  header('Access-Control-Allow-Headers: Origin, Content-Type, X-Auth-Token');
9
10 if($_SERVER['REQUEST_METHOD'] !== 'POST') {
11     http_response_code(405);
12     echo json_encode([
13         'success' => 0,
14         'message' => 'Bad Request Detected.'
15     ]);
16     exit;
17 }
18
19 require_once 'db_connect.php';
20
21 $database = new Operations();
22 $conn = $database->dbConnection();
23
24 $data = file_get_contents("php://input");
25 if(!isset($data->nom) || !isset($data->prenom) || !isset($data->adresse) || !isset($data->ville) || !isset($data->pays) || !isset($data->cp))
26     echo json_encode([
27         'success' => 0,
28         'message' => 'Enter all values.'
29     ]);
30 }
```

Dans cette première partie du fichier, on a tous les paramètres de la requête reçue avec une vérification du type de la requête afin d'éviter les bugs dans le fichier. On vérifie également que toutes les données sont présentes même si c'est déjà vérifié côté application.

Dans la deuxième partie du fichier, on passe à l'exécution du besoin, à savoir l'insertion en base de notre partenaire. On commence par récupérer les données du *FormData* qui s'envoient en méthode *POST* classique. On peut également voir que je double les caractères ' des données. Cela permet d'insérer le caractère en *SQL*. Cela implique d'écrire la requête d'une manière peu commune comme ci-dessous pour pouvoir permettre l'insertion de ce caractère qui de base sert de délimiteur aux champs de texte.

Pour commencer l'insertion, je crée un compte utilisateur avec les données qui lui sont destinées, ensuite je récupère l'identifiant ainsi créé afin de pouvoir le renseigner dans la table partenaire avec les informations uniques à un partenaire.

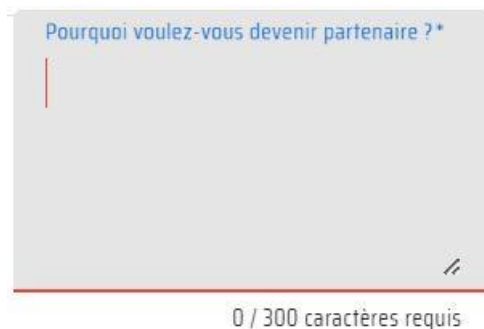
```
32 try{
33
34     $nom = str_replace("'", "\'", $_POST['nom']);
35     $prenom = str_replace("'", "\'", $_POST['prenom']);
36     $adresse = str_replace("'", "\'", $_POST['adresse']);
37     $ville = str_replace("'", "\'", $_POST['ville']);
38     $pays = str_replace("'", "\'", $_POST['pays']);
39     $cp = str_replace("'", "\'", $_POST['cp']);
40     $tel = $_POST['tel'];
41     $email = $_POST['email'];
42     $nomSociete = str_replace("'", "\'", $_POST['nomSociete']);
43     $siret = $_POST['siret'];
44     $mdp = $_POST['mdp'];
45     $newsletter = $_POST['newsletter'];
46     $file = $_FILES['image'];
47
48     $sql='INSERT INTO user (nomUser, prenomUser, emailUser, passwordUser, telUser, idDroitUser, newsUser)'
49         . 'VALUES ("', $nom, '", "', $prenom, '", "', $email, '", "', $mdp, '", "', $tel, '", 4, "', $newsletter, '")';
50     $exec=$conn->prepare($sql);
51     $exec->execute();
52
53     $sql='INSERT INTO partenaires (idPartenaire, nomSociete, adresseSociete, villeSociete, cpSociete, pays, siret, idEtatPartenaire, titreF
54         . 'VALUES ((SELECT idUser FROM user WHERE emailUser = "', $email, '" AND telUser = "', $tel, '"'), "', $nomSociete, '", "', $adresse, '"',
55     $exec=$conn->prepare($sql);
56     $exec->execute();
```

Une fois les données renseignées, on doit se charger du logo. D'abord on récupère l'identifiant du compte pour le concaténer au nom de l'image qui sera « logo-XX » selon le compte puis on la déplace dans le fichier correspondant avant d'update la table partenaires avec le nom du logo.

```
58 $sql="SELECT idPartenaire FROM partenaires WHERE idPartenaire = (SELECT idUser FROM user WHERE emailUser = '$email' AND telUser = '$tel";
59 $exec=$conn->prepare($sql);
60 $exec->execute();
61 $idP = $exec->fetch();
62 $idP = $idP['idPartenaire'];
63
64
65 $tmp_name = $file['tmp_name'];
66 $name = $file['name'];
67
68 $file_ext = explode('.', $name);
69 $file_ext = strtolower(end($file_ext));
70
71 $name = 'logo-' . $idP . '.' . $file_ext;
72
73 move_uploaded_file($tmp_name, "photos/logoPartenaires/$name");
74
75 $sql="UPDATE partenaires SET logo = '$name' WHERE idPartenaire = $idP";
76 $exec=$conn->prepare($sql);
77 $exec->execute();
78
79 }catch (Exception $e) {
80     http_response_code(404);
81     echo json_encode([
82         'success' => 0,
83         'data' => $e->getMessage()
84     ]);
85     exit;
```

Une fois le compte créé et les données bien renseignées, un mail est envoyé à la personne indiquant que son compte est bien en attente de validation. Car pour qu'un compte partenaire ou annonceur puisse être utilisé, il doit être validé par un administrateur. Ces demandes de comptes doivent être motivés par la personne.

La motivation, elle, est juste un input de 300 caractères minimum afin que les administrateurs puissent savoir pourquoi les personnes veulent devenir partenaire. Cette information est enregistrée en base et ressortie lors de la vérification de la demande via un admin.



Pour ce qui est du mail, j'ai choisi de mettre l'expéditeur, l'objet, le corps du mail et diverses informations dans un *FormData* afin de pouvoir toujours utiliser le fichier le même fichier *php* pour l'envoi. Cela se présente de la manière suivante :

```
const mail = new FormData();
mail.append('email', ''+this.insertForm.value.email);
mail.append('subject', 'Altameos.fr - Demande de compte partenaire');
mail.append('body', "Bonjour, <br><br> Votre compte au nom " + this.insertForm.value.nom + ' ' + this.insertForm.value.prenom + ' de la s'
+ " est en cours de validation. <br>Vous recevrez prochainement un mail vous informant de l'évloution de votre demande."
+ "<br><br>Cordialement, <br><br> Altameos");
```

Comme dis précédemment, la fonction *insertPartenaire* dans le service attend un paramètre qui sera envoyé en méthode *POST* au fichier *php* gérant les mails.

```
require 'src\Exception.php';
require 'src/PHPMailer.php';
require 'src/SMTP.php';

$mail = new PHPMailer(true);
try{
    //Server settings
    $mail->SMTPDebug = 0;
    $mail->isSMTP();
    $mail->Host = 'frweb7.pulseheberg.net';
    $mail->SMTPAuth = true;
    $mail->SMTPSecure = 'ssl';
    $mail->Username = ' ';
    $mail->Password = ' ';
    $mail->Port = '465';
    $mail->Charset = "UTF-8";

    //Recipients
    $mail->setFrom('no-reply@altameos.fr', 'Altameos.fr');
    $mail->addAddress($_POST['email']);
    // $mail->addReplyTo('
    // $mail->addCC('
    // $mail->addBCC('

    //Attachments
    // $mail->addAttachment('/var/tmp/file.tar.gz');
    // $mail->addAttachment('/tmp/image.jpg', 'new.jpg');

    //Content
    $mail->isHTML(true);
    $mail->Subject = ''.$_POST['subject'];
    $mail->Body = ''.$_POST['body'];
    $mail->AltBody = ''.$_POST['body'];

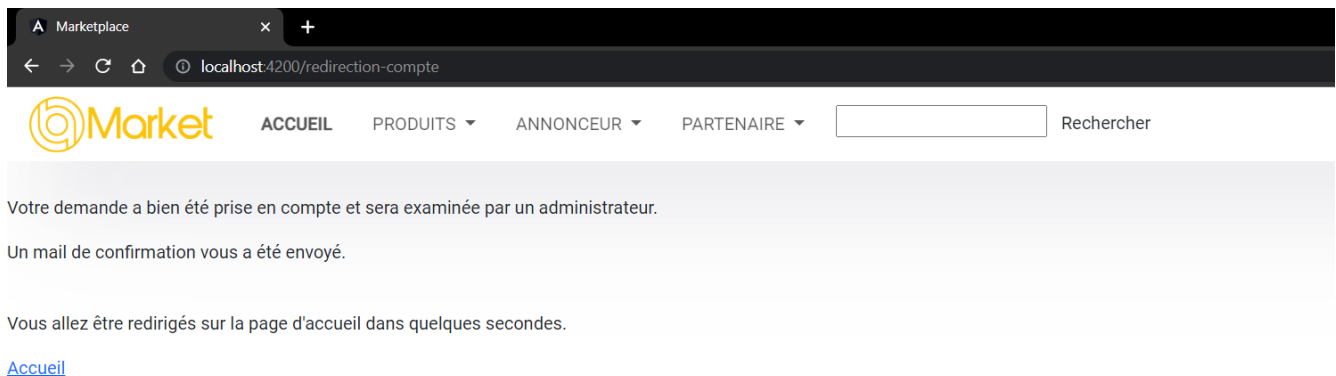
    $mail->send();
    echo 'Message has been sent';
}
```

On peut voir que l'envoi de mail nécessite plusieurs fichiers qui sont « require » en haut de l'image. Ensuite il faut définir plusieurs paramètres comme montré ici. Puis définir l'adresse d'envoi, l'adresse de réception, l'objet ainsi que le corps.

Exemple type de mail reçu :



Une fois la demande effectuée avec succès, l'utilisateur est automatiquement redirigé vers une page d'information qui redirigera automatiquement vers la page d'accueil au bout de quelques secondes. Voici un exemple de la page de redirection.



Le *typeScript* du component de redirection est composé d'un compte-à-rebours qui exécute ce qui l'y a à l'intérieur de ceux-ci une fois le temps écoulé, dans ce cas de figure, une redirection vers l'accueil est effectuée après 15 secondes.

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';

@Component({
  selector: 'app-redirection-compte',
  templateUrl: './redirection-compte.component.html',
  styleUrls: ['./redirection-compte.component.scss']
})
export class RedirectionCompteComponent implements OnInit {

  constructor(private router: Router){};

  ngOnInit(): void {
    setTimeout(() => {this.router.navigate(['/']);}, 15000);
  }
}
```

Temps en millisecondes

L'appel de ce *component* se fait par la ligne suivante se trouvant dans le *typeScript* du *component* d'insertion.

```
this.router.navigate(['redirection-compte']);
```

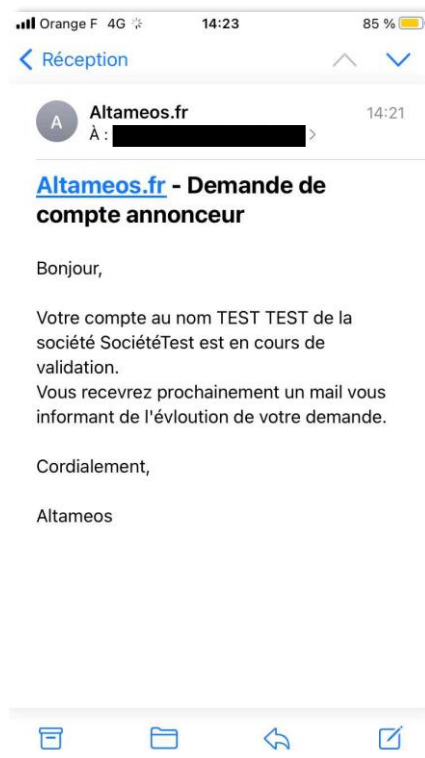
Une fois que les comptes seront validés côté administrateur, un autre mail sera envoyé pour informer le client sur l'évolution de sa demande.

La création de comptes annonceurs :

La création de comptes annonceurs suit le même procédé que la création de comptes partenaires. La seule chose qui change ce sont les requêtes SQL et le component qui n'est pas le même. Hormis ça le fonctionnement est le même car les informations à enregistrer étaient les mêmes.

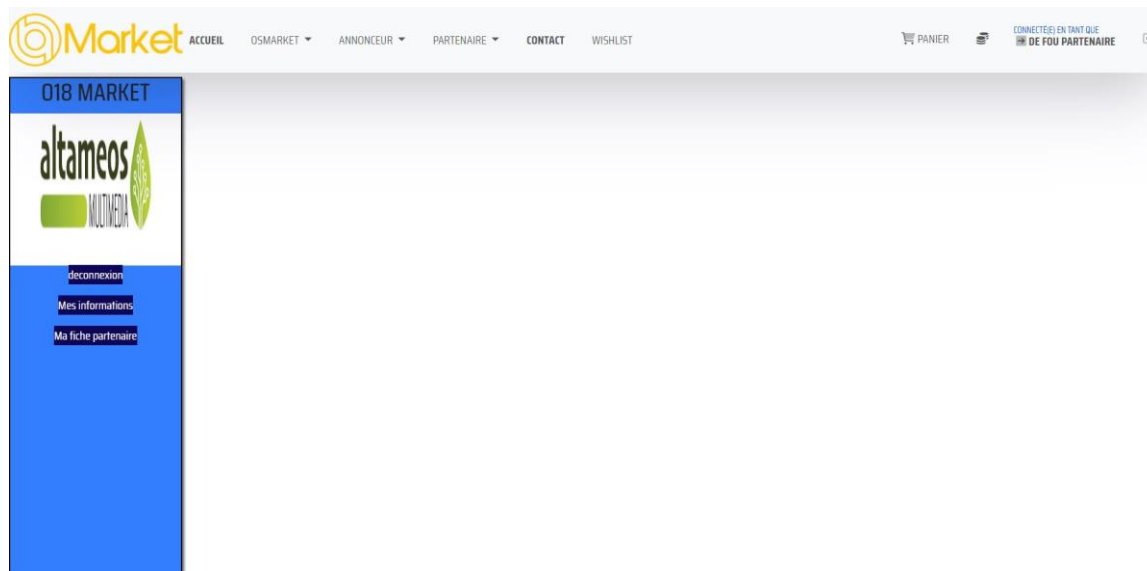
The screenshot shows a web interface for creating an advertiser account. On the left is a sidebar with the title 'Gestion utilisateurs' and three links: 'Devenir Client', 'Devenir Annonceur' (highlighted), and 'Devenir Partenaire'. The main content area has three tabs: 'S'identifier', 'S'inscrire' (selected), and 'Mot de passe oublié ?'. Below the tabs is a header 'DEMANDE DE COMPTE ANNONCEUR'. The form contains several input fields: 'Nom *', 'Prénom *', 'Adresse *', 'Ville *', 'Pays *' (a dropdown menu), 'Code Postal *', 'Téléphone *', 'Email *', 'Ex : +(n° pays)X XX XX XX XX' (a hint for the phone number), 'Nom de la société *', 'SIRET *', 'Mot de passe *', and 'Confirmation Mot de passe *'. There are character counts for 'Email' (0/5), 'SIRET' (0/14), 'Mot de passe' (0/12), and 'Confirmation Mot de passe' (0/0). A 'Logo' section has a 'Choisir un fichier' button and the text 'Aucun fichier choisi'. A 'Newsletter' toggle switch is set to 'off'. At the bottom is a blue button labeled 'Envoyer ma demande'.

Naturellement l'email reçu diffère aussi légèrement :



L'espace partenaire :

Comme son nom l'indique, l'espace partenaire est un espace dédié aux partenaires. Celui-ci permet aux partenaires de gérer leurs informations ainsi que d'éditer une fiche descriptive. Cet espace est composé d'un accueil, d'une partie pour les informations et une partie pour la fiche.



Cet accueil est un *component* qui contient d'autres *component*, le menu sur la gauche est un *component* appelé au sein de celui de l'accueil. Quant à ceux derrière les boutons, il s'agit de *component* enfants de celui de l'accueil. Pour qu'il puisse y avoir une relation parent-enfant entre les *components*, il faut les déclarer dans le fichier de *routing* et inclure une balise de *routing* à l'intérieur du *component* parent. Le logo présent dans le menu de gauche est celui qui est associé au compte lors de la création de celui-ci.

Dans le fichier *app-routing.module.ts* :

```
{ path: 'espacePartenaire',  
  component: EspacePartenaireAccueilComponent,  
  children: [  
    { path: 'fiche', component: EspacePartenaireFicheComponent },  
    { path: 'informations', component: EspacePartenaireInformationsComponent },  
    { path: '', redirectTo: 'accueil', pathMatch: 'full' }  
  ]  
},
```

Dans le composant d'accueil de l'espace partenaire, côté html :

```
<app-espace-partenaire-header [partenaire]="partenaire"></app-espace-partenaire-header>
<router-outlet></router-outlet>
```

Une balise `<router-outlet></router-outlet>` qui permet la relation parent-enfant entre les *components*.

Ensuite, une fois que l'on clique sur le bouton « mes informations », on est mené à la page suivante :

O18 MARKET

altameos MULTIMEDIA

FENEROL Miguel
Société EulaInc.

[deconnection](#)

[Mes informations](#)

[Ma fiche partenaire](#)

Vos informations

Nom*	Prenom*
FENEROL	Miguel

Adresse*	Ville*	Pays*	Code Postal*
		France	

Téléphone*	Email*	Nom de la société*	SIRET*
0606060606		EulaInc.	22222222222222

Ex : +(n° pays)X XX XX XX XX

Logo : Aucun fichier choisi

14 / 14

Ce formulaire qui sert à mettre les informations à jour est pré rempli avec les informations du compte comme on peut le voir ci-dessus. Seule l'adresse mail est fixe et ne peut être changée sans intervention d'un administrateur.

Pour ce qui est du remplissage automatique des champs, sous Angular ce n'est pas aussi simple qu'en *HTML/php*. Pour cela il faut créer un objet contenant nos données, donc en créant un modèle (qui ressemble aux classes c#) et en créant un formulaire dont les noms d'*inputs* correspondent au nom de l'objet.

```
export class updatePartenaire{
  nom?: string;
  prenom?: string;
  adresse?: string;
  ville?: string;
  pays?: string;
  cp?: string;
  tel?: string;
  email?: string;
  nomSociete?: string;
  siret?: string;
  titre?: string;
  description?: string;

  constructor(nom: string, prenom: string, adresse: string, ville: string, pays: string, cp: string, tel: string, email: string, nomSociete: string, siret: string, titre: string, description: string){
    this.nom = nom;
    this.prenom = prenom;
    this.adresse = adresse;
    this.ville = ville;
    this.pays = pays;
    this.cp = cp;
    this.tel = tel;
    this.email = email;
    this.nomSociete = nomSociete;
    this.siret = siret;
    this.titre = titre;
    this.description = description;
  }
}
```

Le module *ngOnInit* présent ci-dessous sert à exécuter une partie du code lors de l'initialisation du *component*. Ce qui permet de faire des choses avant le lancement comme ici charger les éléments dans les champs du formulaire.

```
ngOnInit(): void {
  this.partenaireService.getLesPays().subscribe((data:any)=>{(this.pays = data.data)});
  this.id = JSON.parse("" + localStorage.getItem("authUser"));
  this.partenaireService.getUserPartenaire(this.id.id).subscribe((data:any)=>{
    this.partenaire = data.data;
    let formFill = new updatePartenaire(
      this.partenaire.nomUser,
      this.partenaire.prenomUser,
      this.partenaire.adresseSociete,
      this.partenaire.villeSociete,
      this.partenaire.pays,
      this.partenaire.cpSociete,
      this.partenaire.telUser,
      this.partenaire.emailUser,
      this.partenaire.nomSociete,
      this.partenaire.siret,
      '', ''
    );
    this.updateForm.patchValue(formFill);
  });
}
```

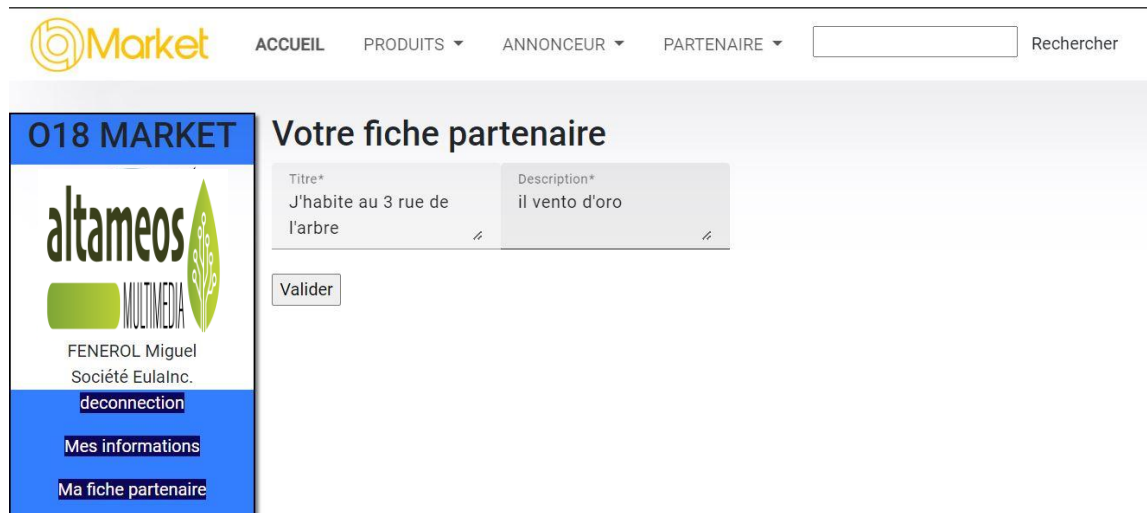
Ce qui se passe ici ; c'est que je récupère l'identifiant du compte connecté puis je récupère les informations du compte en question via la requête *SQL* ci-dessous.

```
if (isset($_GET['id']))
{
    $id = filter_var($_GET['id'], FILTER_VALIDATE_INT, [
        'options' => [
            'default' => 'all_pays',
            'min_range' => 1
        ]
    ]);
}

try
{
    $sql = "SELECT nomUser, prenomUser, emailUser, telUser, newsUser, nomSociete, adresseSociete, villeSociete, cpSociete, pays, logo,
            . "FROM user INNER JOIN partenaires ON idUser = idPartenaire WHERE idUser = $id";
    $stmt = $cnx->prepare($sql);
    $stmt->execute();
}
```

Une fois les informations récupérées, j'instancie un objet avec celles-ci puis j'utilise la méthode *patchValue* elle permet de donner les valeurs des noms correspondant à un formulaire. Contrairement à *setValue*, *patchValue* s'exécute même si tous les champs ne correspondent pas.

Ensuite il y a la page de la fiche partenaire qui fonctionne de la même manière que la précédente.



Gestion du header :

De base le header n'était pas dans un component et fait avec des balises « href » puis mis en brut sur le component accueil. Le but d'Angular est d'avoir une sorte d'application en ligne dont les pages s'affichent sans causer de rechargement afin d'avoir un site dynamique. Du coup mis le header dans un component et refais le système de route dans le header. Pour y parvenir, j'ai revu le système de routing de l'application puis j'ai remplacé les « href » par des « routerLink ».

Gestion de l'application :

Afin de prévenir le maximum de failles de sécurité, j'ai revu la navigation dans l'application afin d'y implémenter des composants dépendants d'autres afin d'empêcher que les données se baladent trop notamment du côté de l'administration.

```
{ path: 'admin', component: AdminAccueilComponent, //Administration
  children: [
    { path: 'gestionUtilisateurs', component: GestionUtilisateursComponent }, //Gestions des utilisateurs
    { path: 'gestionAdministrateurs', component: GestionAdministrateursComponent }, //Gestions des utilisateurs
    { path: 'gestionAnnonces', component: GestionAnnoncesComponent }, //Gestion des annonces
    { path: 'gestionPartenaires', component: GestionPartenairesComponent }, //Gestion des partenaires
    { path: 'listCategories', component: ListCategoriesComponent }, //Gestion des catégories
    { path: 'listGammes', component: ListGammesComponent }, //Gestion des gammes
    { path: 'gestionProduits', component: GestionProduitsComponent }, //Gestion des produits
    { path: 'gestionAvis', component: GestionAvisComponent }, //Gestion des avis
    { path: 'gestionAvis/:valid', component: GestionAvisComponent }, //Gestion des avis + tri
    { path: 'gestionOffreAnnonces', component: GestionOffresAnnoncesComponent }, //Gestion des offres d'annonces
    { path: 'newsletter', component: NewsletterComponent }, //Envoie de newsletter
    { path: 'gestionAnnonces', component: GestionAnnoncesComponent }, //Gestion des annonces
    { path: 'gestionActualites', component: GestionActualitesComponent }, //Gestion des actualités
    { path: 'gestionTutoriels', component: GestionTutorielsComponent }, //Gestion des tutos
    { path: 'gestionAdministrateurs', component: GestionAdministrateursComponent }, //Gestion des administrateurs
    { path: 'pagesConnexes', component: ListPagesConnexesComponent }, //Gestion des pages connexes
    { path: 'gestionCommandes', component: GestionCommandesComponent }, //Gestion des commandes
    { path: 'gestionInfoSite', component: FormInfositeComponent }, //Gestion des informations de la boutique
  ],
},
```


De plus, j'ai empêché la navigation via l'URL en ajoutant des conditions sur les « routerLink » et en faisant une vérification des rôles lors du chargement d'un component afin qu'un simple utilisateur ne puisse pas accéder à la page admin en tapant /admin dans sa barre de navigation par exemple. Cela correspond à la faille de sécurité « Broken Access Control » qui est une des plus exploitée de nos jours.

```
ngOnInit(): void {  
    if (localStorage.getItem('authUser')) {  
        if (JSON.parse(localStorage.getItem('authUser')).roles !== "superadmin") {  
            window.location.href = "/405";  
        } else {  
              
        }  
    } else {  
        window.location.href = "/login";  
    }  
}
```

Ajustements divers :

Durant les derniers jours de stage, j'ai mis quelques petites fonctionnalités à jour, comme le bouton du panier qui apparaissait partout et indépendamment du compte. J'ai également mis le footer dans un component et l'ai affiché au bon endroit. De plus, j'ai également supprimé beaucoup de console.log qui traînaient dans le code (utilisés lors de tests). Puis j'ai commenté certaines parties du code.

Bilan du projet :

Bien que massif malgré le temps donné, en ajoutant à cela une semaine de formation, le projet a été fini. L'idéal aurait été d'avoir une semaine supplémentaire afin de peaufiner certaines tâches et d'optimiser un peu plus l'application en général. Quant au projet en lui-même, il aurait été vraiment très intéressant et instructif. Il nous a permis d'apprendre à se servir d'une nouvelle technologie en plus de nous demander des choses jamais réalisés jusque-là. Dans l'ensemble, je n'ai pas eu trop de galères au niveau du code et le projet s'est très bien déroulé et a été bien organisé. Je remercie donc à nouveau notre maître de stage de nous avoir apporté ce projet.

Bilan du stage :

Ce stage de 6 semaines en groupe et en télétravail s'est extrêmement bien déroulé. Le télétravail n'a pas été une contrainte, ni le fait de travailler en groupe de 6. Cela a même été, au contraire, une opportunité qui nous prépare à nos futurs métiers respectifs. Les collègues étaient très aimables et coopératifs, de même pour notre tuteur qui prenait le temps de faire un suivi quotidien de l'avancement du projet. Je remercie donc à nouveau toutes les personnes qui ont participé à faire de ce stage une expérience unique.