# Rapport de stage

*Altameos Multimédia*

ROMAIN ZABETH

*2023*

# Rapport de stage

Développeur web

ROMAIN ZABETH

Janvier 2023 à Février 2023

Tuteur de stage : Lucas Schnekenburger

Etablissement : Saint-Adjutor - BTS SIO 2ème année option SLAM

Entreprise d'accueil : Altameos Multimédia

# REMERCIEMENTS

Je tiens à remercier toutes les personnes qui ont contribué au succès de mon stage.

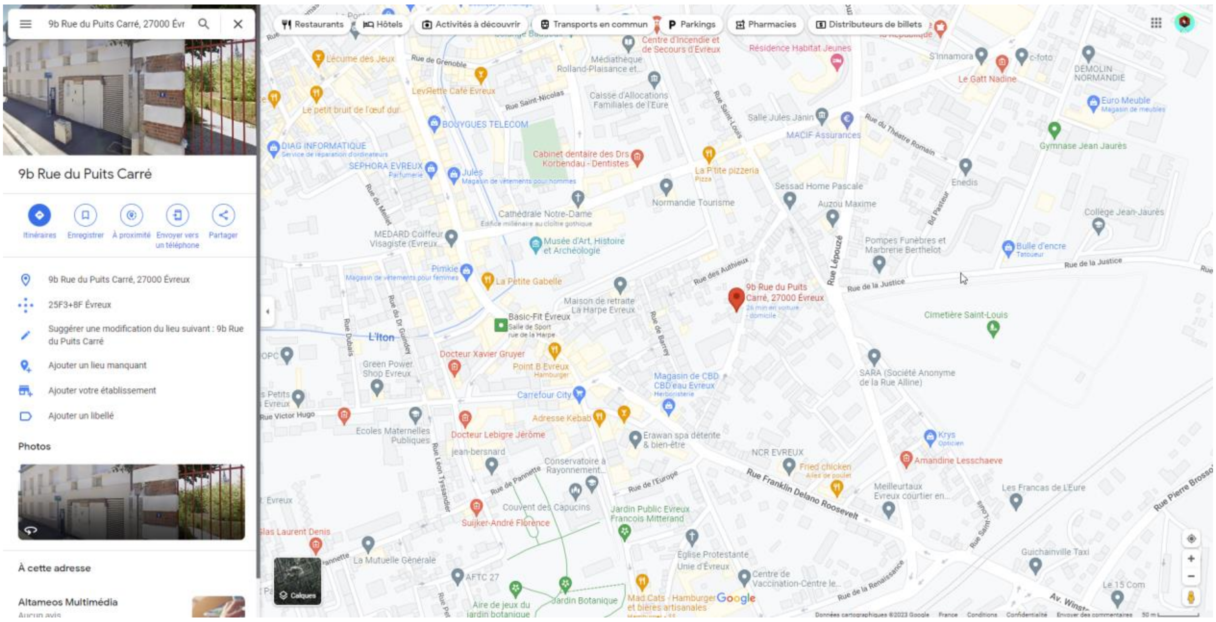
Tout d’abord, je tiens à remercier mon maître de stage, M. Schnekenburger, pour m’avoir accepté en tant que stagiaire. Grâce à lui, cette expérience professionnelle en télétravail aura été une réussite et surtout une très bonne expérience.

Je tiens aussi à remercier les stagiaires qui étaient avec moi pendant ce stage, Oliwer, Damien, Max, Hugo et Miguel avec qui j’ai collaboré et travaillé sur le même projet, dans une bonne ambiance.

<u>Sommaire</u>

# I. Présentation de l’entreprise

Mon stage se passe dans l’entreprise Altameos Multimédia. Altameos Multimédia a été créé le 10 octobre 2003. L’entreprise se situe à Evreux (27000) au 9b Rue du Puits Carré.



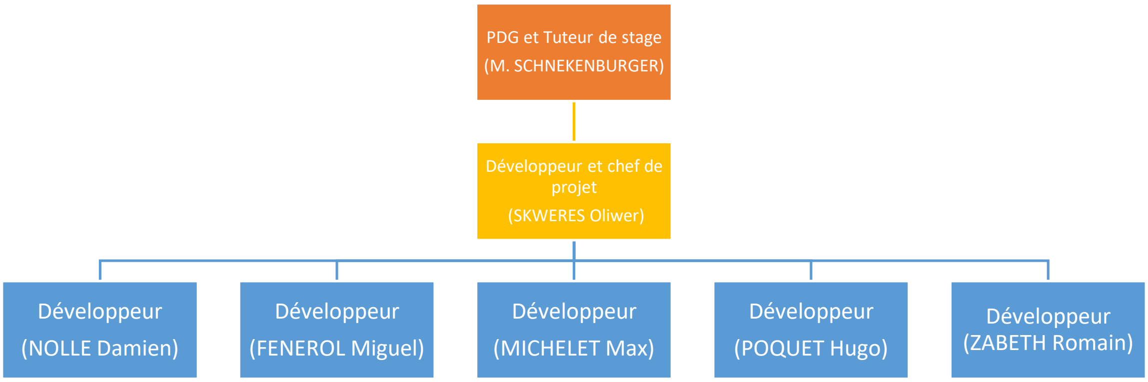

## a) Les secteurs d’activité

Altameos Multimédia est une agence numérique née dans le but d’aider les gens dans de diverses tâches concernant les sites internet. Cela peut être du web design, du référencement comme des conseils numériques. Altameos Multimédia est donc là avant tout pour apporter son soutien aux diverses personnes dans le besoin.

## b) Les membres de l’association

L’entreprise est composée d’une unique personne qui est Lucas Schnekenburger, en tant que travailleur indépendant.

Dans l’organisation d’un projet, l’entreprise peut avoir des partenaires.



# II. Les conditions de travail

Ce stage se passe en distanciel. Je travaille donc de chez moi, et ce, pendant l'entièreté du stage. Nous sommes 6 stagiaires en tout dans ce stage.

## a) Les moyens informatiques matériels

Aucun matériel informatique n'a été fourni, j'effectue donc le stage en travaillant avec mon ordinateur personnel.

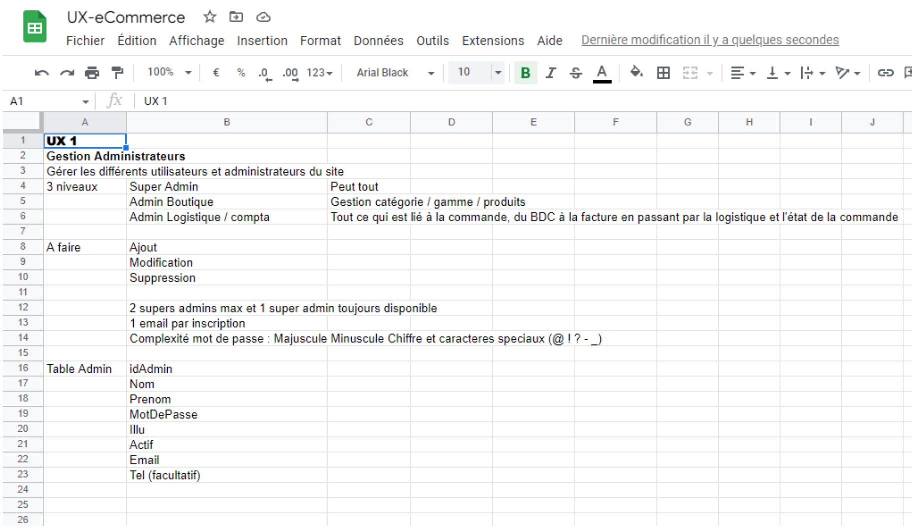
## b) Les moyens informatiques logiciels

Pour effectuer ce stage, j'utilise Wampserver64 pour gérer la base de données ainsi que Visual Studio Code 2022 pour programmer le site.

Le rapport de stage quant à lui, est fait sur Word 2019.

## c) Les ressources fournies

Le maître de stage nous a fourni 2 fichiers Excel. Un est une fiche UX, l'autre est un Dashboard.


UX-eCommerce

Fichier Édition Affichage Insertion Format Données Outils Extensions Aide Dernière modification il y a quelques secondes

A1 UX 1

| | A | B | C |
|---|---|---|---|
| 1 | **UX 1** | | |
| 2 | **Gestion Administrateurs** | | |
| 3 | Gérer les différents utilisateurs et administrateurs du site | | |
| 4 | 3 niveaux | Super Admin | Peut tout |
| 5 | | Admin Boutique | Gestion catégorie / gamme / produits |
| 6 | | Admin Logistique / compta | Tout ce qui est lié à la commande, du BDC à la facture en passant par la logistique et l'état de la commande |
| 7 | | | |
| 8 | A faire | Ajout | |
| 9 | | Modification | |
| 10 | | Suppression | |
| 11 | | | |
| 12 | | 2 supers admins max et 1 super admin toujours disponible | |
| 13 | | 1 email par inscription | |
| 14 | | Complexité mot de passe : Majuscule Minuscule Chiffre et caracteres speciaux (@ ! ? - _) | |
| 15 | | | |
| 16 | Table Admin | idAdmin | |
| 17 | | Nom | |
| 18 | | Prenom | |
| 19 | | MotDePasse | |
| 20 | | Illu | |
| 21 | | Actif | |
| 22 | | Email | |
| 23 | | Tel (facultatif) | |
| 24 | | | |
| 25 | | | |
| 26 | | | |

| Etat | Par | FRONT | BACK | Spécificité | UX liées | nb jours |
|---|---|---|---|---|---|---|
| | | | Gestion des administrateurs | | UX 1 | 1 |
| | | Mise en place structure Angular de base | | | | 2 |
| | | | Gestion des utilisateurs | | UX 2 | 1 |
| | | | Gestion des annonceurs | | UX 3 | 1 |
| | | | Gestion des partenaires | | UX 4 | 1 |
| | | Sécurisation et accès sécurisé | | | | 4 |
| | | | Gestion des catégories | | | 0,5 |
| | | | Gestion des gammes | | | 0,5 |
| | | | Gestion des produits | | | 1 |
| | | | Gestion des Offres pour annonceurs / packs | | | 1 |
| | | | Gestion des annonces | | | 1 |
| | | | Gestion des informations de la boutique | | | 0,5 |
| | | | Gestion des bannières publicitaires | | | 1 |
| | | | Gestion du carroussel de la page d'accueil | | | 1 |
| | | | Gestion de pages connexes (mentions / crédits / CGV...) | | | 1 |
| | | | Gestion du panier de commande | | | 1 |
| | | | Gestion de la commande | | | 1 |
| | | | Gestion des états de commande | | | 1 |
| | | | Gestion des erreurs commandes | | | 1 |
| | | | Gestion logistique commande | | | 1 |
| | | | Gestion Satisfaction | | | 1 |
| | | | Gestion Actualités | | | 1 |
| | | | Gestion Tutoriels en ligne | | | 1 |
| | | Entête | | | | 0,5 |
| | | Footer | | | | 0,5 |
| | | Recherche produits | | | | 1 |
| | | Inscription Client | | | | 0,5 |
| | | Identification Client | | | | 0,5 |
| | | Gestion Client | | | | 0,5 |
| | | Inscription Annonceur | | | | 0,5 |
| | | Identification Annonceur | | | | 0,5 |
| | | Gestion Annonceurs | | | | 0,5 |
| | | Gestion Annonceurs – Produits | | | | 0,5 |
| | | Gestion Annonceurs – Packs choisis | | | | 1 |
| | | Inscription Partenaire | | | | 0,5 |
| | | Identification Partenaire | | | | 0,5 |
| | | Gestion Panier | | | | 1 |
| | | Gestion Processus commande | | | | 1 |
| | | Gestion Page Catalogue | | | | 0,5 |
| | | Gestion Page Produits | | | | 0,5 |
| | | Gestion Page Accueil | | | | 1 |
| | | Gestion Pages du site | | | | 1 |
| | | Gestion Actualités et détail | | | | 1 |
| | | Gestion Tutoriaux et détail | | | | 1 |
| | | Gestion Newsletter | | | | 0,5 |
| | | Gestion Contact | | | | 0,5 |
| | | Gestion Produits les plus vendus | | | | 0,5 |
| | | Gestion Produits notation | | | | 0,5 |
| | | Gestion Produits les plus vus | | | | 0,5 |
| | | Gestion Wishlist | | | | 1 |
| | | Gestion Erreurs Commande / Livraison / reclam. | | | | 1 |
| | | Installation sur serveur Test | | | | 1 |
| | | | | | | |
| | | | | Total Jours projets | | 36,5 |
| | | | | Temps de travail par binome | | 12,166667 |

## d) La communication

J’utilise l’application Teams ainsi que Discord pour communiquer avec mes coéquipiers ainsi qu’avec mon maître de stage si j’ai des questions à lui poser ou des retours à faire.

Un appel vocal est effectué avec le tuteur chaque début et fin de journée pour tenir compte de ce qu’on va réaliser et de ce qu’on a réalisé pendant la journée.

# III. Présentation du projet

## a) Le contexte

Notre maître de stage M. Schnekenburger aimerait réaliser un site d'e-commerce pour pouvoir le revendre.

## b) Notre objectif

Notre objectif est donc de créer un site d'e-commerce en Angular. Dans ce site, des annonceurs pourront rejoindre la plateforme et payer une offre pour afficher leurs annonces pour la vente de leurs produits. Des partenaires pourront aussi rejoindre, leur permettant d'afficher leur publicité. Sans oublier les clients qui pourront commander les produits venant des annonceurs. Puis les administrateurs (superadmins, admins boutiques et admins logistique), qui pourront modérer le site que ça soit pour les commentaires, les validations d'annonces ou encore la gestion des différentes données du site comme les gammes ou les produits (ajouts, modifications, suppressions). Un système de paiement doit être mis en place pour permettre aux différents clients d'acheter des produits ou des offres pour les annonceurs.

J'ai bien évidemment la possibilité d'exposer mes idées pour une quelconque amélioration du site.

## c) L'objectif du site

L'objectif du site est d'en faire un site d'e-commerce viable. C'est-à-dire qu'à la fin de notre stage, le site doit pouvoir être utilisé par les clients pour acheter des produits. Les produits vendus seront des produits aidant les personnes atteintes de handicap.

# IV. Le développement du projet

## a) Les outils utilisés

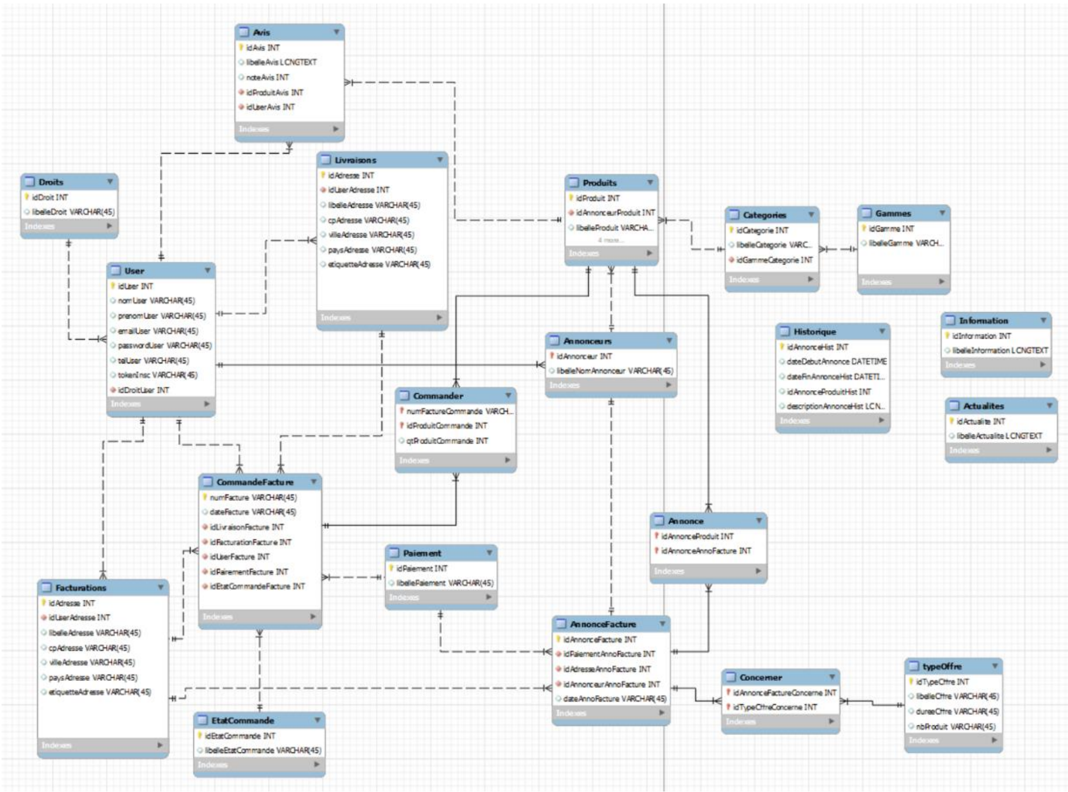
Pour programmer mon site, j'utilise le logiciel Visual Studio Code 2022. Pour interagir avec la base de données, j'utilise WampServer64.

Pour communiquer avec mes coéquipiers, j'utilise principalement discord. Mais j'utilise aussi Microsoft Teams pour communiquer avec notre maître de stage. Pour fusionner le travail de toute l'équipe on a utilisé l'outil de versioning git kraken GitKraken .

J'utilise comme navigateur Opera GX , pour les recherches aussi bien que pour tester le site en local. Pour nous organiser dans toutes les tâches que nous devions réaliser et suivre leur avancement, nous avons utilisé le site Trello .

## b) La base de données

Nous avons dû créer la base de données de 0, pour nous aider nous avions le contexte, les fichiers Excel sur lesquels il y avait déjà écrit quelques tables et enfin l'aide de notre tuteur.

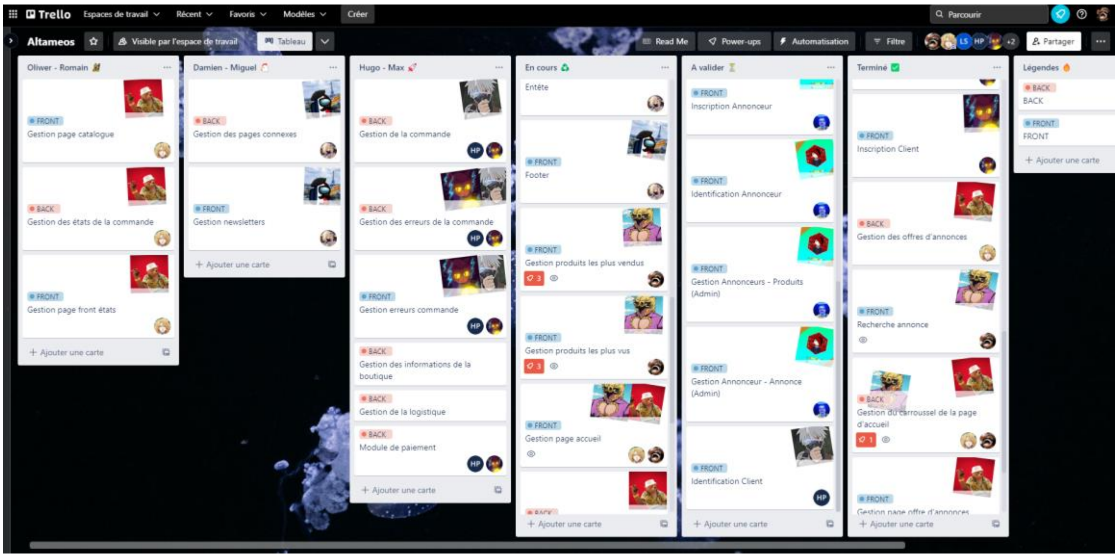


Pendant la confection du site, on s’est retrouvé à devoir changer la base de données pour y rajouter des tables, des modifications ou même des suppressions.

### c) Attribution des tâches

Pour réaliser ce site de commerce, nous avons dû nous répartir le travail. Cette répartition s’est organisée en tâches. Me concernant, j’ai dû m’occuper des tâches suivantes :

- Gestion des gammes (affichage, ajout, modification, suppression)
- Gestion des catégories (affichage, ajout, modification, suppression)
- Gestion du carrousel de la page d’accueil (affichage)
- Gestion des bannières publicitaires
    - Affichage côté client
    - Gestion des publicités (affichage, ajout, modification, suppression)
- Recherche de produit à partir de la page d’accueil
    - Par nom
    - Par gamme
    - Par catégorie
- Filtre des produits
    - Les plus vues
    - Les plus vendus
    - Les plus notés



## d) Le déroulement du stage

### 1) La semaine de formation

Avant de commencer à programmer le site, nous avons eu 1 semaine pour nous former sur le Framework Angular, vu que le site doit être réalisé en utilisant celui-ci. On s'est bien évidemment partagé les tutoriaux qu'on trouvait de notre côté pour accélérer notre apprentissage. L'apprentissage s'est bien passé de mon côté, j'ai pu commencer le projet sereinement.

Ce que je retiens d'Angular c'est que chaque page à son component contenant un fichier pour son affichage(.html), un fichier pour son style graphique(.scss) et un fichier pour la gestion des données (.ts).

```
header
  header.component.html
  header.component.scss
  header.component.spec.ts
  header.component.ts
```

On a aussi un fichier contenant toutes les routes utilisées.

```
{ path: 'addPartenaire', component: AddPartenaireComponent },
{ path: 'add-product', component: AddProduitComponent },
{ path: 'addGamme', component: AddGammesComponent },
{ path: 'addCategorie', component: AddCategoriesComponent },
{ path: 'addOffreAnnonce', component: AddOffreAnnonceComponent},
{ path: 'addAnnonceur', component: AddAnnonceurComponent},
{ path: 'addPublicites', component: AddPublicitesComponent},
{ path: 'redirection-compte', component: RedirectionCompteComponent},
```

Ainsi que des fichiers contenant des services que l'on appellera dans chaque component. Ces services contiennent des fonctions directement stockées dans un api.

```
export class PubService {

  constructor( private http: HttpClient) { }

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

  getRandomPub() {
    return this.http.get(this.baseUrl+'get-random-pub.php')
  }

  getPub() {
    return this.http.get(this.baseUrl+'viewPublicites.php')
  }

  getSinglePub(id:any) {
    return this.http.get(this.baseUrl+'view-one-pub?id='+id);
  }

  createPub(Pub:any) {
    //  console.log(id);
      return this.http.post(this.baseUrl+'insertPub.php', Pub);
    }

  editPub(Pub:any) {
  //  console.log(id);
    return this.http.post(this.baseUrl+'edit-publicites.php', Pub);
  }

  deletePub(id:any) {
    return this.http.delete(this.baseUrl + 'deletePublicites.php?id=' + id);
  }

}
```

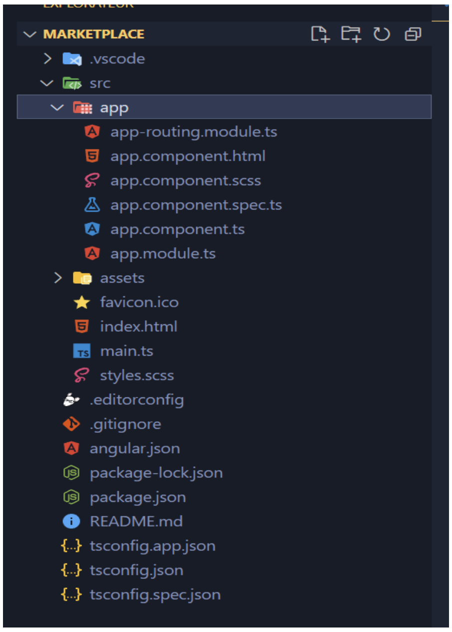
L'api contient les différentes requêtes permettant de récupérer les données de la base de données. L'api utilise le langage de programmation php .

Angular permet aussi d'ajouter plein de bibliothèques permettant un grand nombre de possibilités sur tout. La gestion des données, la navigation ou encore les formulaires. Permettant de faciliter la vie d'un développeur.

Au début, on se perd dans la structure car il y a pas mal de redirection, mais une fois maitrisé, il est un atout de taille.

## 2) Le début du projet

Après avoir créé le Trello pour la gestion des tâches, nous avons choisi les tâches que nous voulions faire. Je me suis donc retrouvé avec les tâches citées plus haut. S'ensuit la création du dépôt GitHub sur lequel se trouve la base du projet. Voici à quoi elle ressemble :




## 3) La gestion des gammes

Pour la première tâche, il fallait en choisir une qui ne dépende pas d'autres tâches. J'ai donc opté pour la gestion des gammes. Sachant qu'une gamme contenait assez peu d'informations, j'ai trouvé judicieux de commencer pour cette tâche.

| idGamme | libelleGamme |
|---|---|
| 1 | Smartphone |
| 2 | Ordinateur |

L'affichage s'est fait assez rapidement même si quelques petites erreurs se sont manifestées.

Concernant l’insertion, Angular possède ses propres formulaires permettant une gestion assez efficace de ceux-ci.

Front :

```
<form [formGroup]="addForm" novalidate class="form">
    <p class="libelle">Libelle Gamme :</p>
    <input type="text" class="input" name="libelleGamme" formControlName="libelleGamme" required #libelleInput>
    <div class="row">
        <div class="col-md-12">
            <button class="cssbuttons-io-button"   (click)="onSubmit()">
```

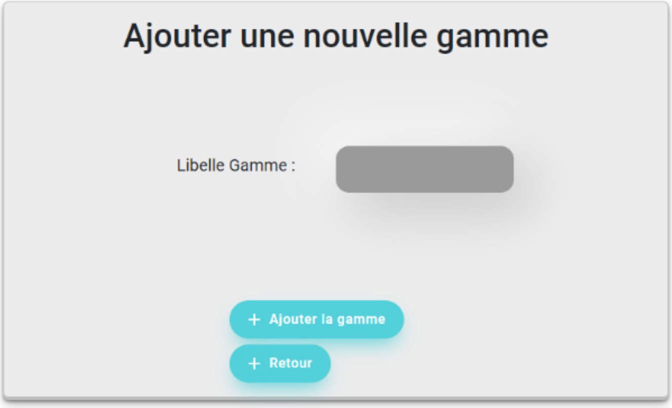
Back :

```
this.addForm = this.formBuilder.group({
  libelleGamme: ['', Validators.required],
})
```

Du côté de l’api l’insertion des données se présente comme ceux-ci.

```
try {

    $libelleGamme = $data->libelleGamme;


    $query = "INSERT INTO `gammes`(
    libelleGamme
    )
    VALUES(
    :libelleGamme
    )";

    $stmt = $conn->prepare($query);


    $stmt->bindValue(':libelleGamme', $libelleGamme, PDO::PARAM_STR);


        if ($stmt->execute()) {

            http_response_code(201);
            echo json_encode([
                'success' => 1,
                'message' => 'Data Inserted Successfully.'
            ]);
            exit;
        }
```

Voici l’interface d’ajout d’une nouvelle gamme.




Le tuteur de stage se chargeait du css, mais j’ai quand même préféré en faire un peu pour que ce soit regardable.

Pour la modification elle se présente comme celle-ci :


La valeur est bien évidemment pré rempli avec la valeur de la gamme que l’on va modifier.

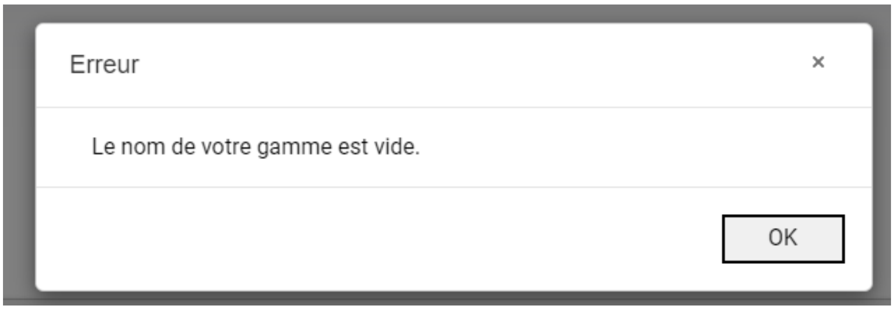
Cette fois, il fallait récupérer l’id de la gamme à modifier, dans le fichier, on recherchera donc l’id avant de faire la modification avec celui-ci.

```php
try {

    $fetch_post = "SELECT * FROM `gammes` WHERE idGamme=:id";
    $fetch_stmt = $conn->prepare($fetch_post);
    $fetch_stmt->bindValue(':id', $data->id, PDO::PARAM_INT);
    $fetch_stmt->execute();

    if ($fetch_stmt->rowCount() > 0) :
     //echo 'AAA';
        $row = $fetch_stmt->fetch(PDO::FETCH_ASSOC);
        $libelleGamme = isset($data->libelleGamme) ? $data->libelleGamme : $row['libelleGamme'];

       $update_query = "UPDATE `gammes` SET libelleGamme = :libelleGamme
        WHERE idGamme = :id";

        $update_stmt = $conn->prepare($update_query);

        $update_stmt->bindValue(':libelleGamme', htmlspecialchars(strip_tags($libelleGamme)), PDO::PARAM_STR);


        $update_stmt->bindValue(':id', $data->id, PDO::PARAM_INT);
```

Des petites fautes de frappe dans le code m'ont fait perdre un peu de temps, mais rien de critique.

En faisant la modification, l'envie de gérer les erreurs commençait déjà à se manifester. J'ai donc trouvé une bibliothèque remplissant parfaitement ce rôle. Elle se nomme Alertify JS.

Voici ce qui se passe si je clique sur le bouton Valider avec le nom de la gamme vide :

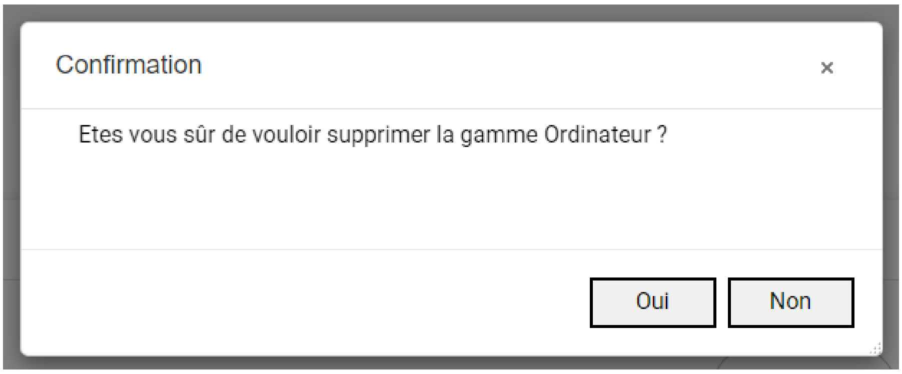



Alertify JS permet un affichage dynamique, de plus, il est simple d'utilisation.

```js
if(this.addForm.value.libelleGamme == "")
{
  alertifyjs.alert("Erreur","Le nom de votre gamme est vide.");
}
```

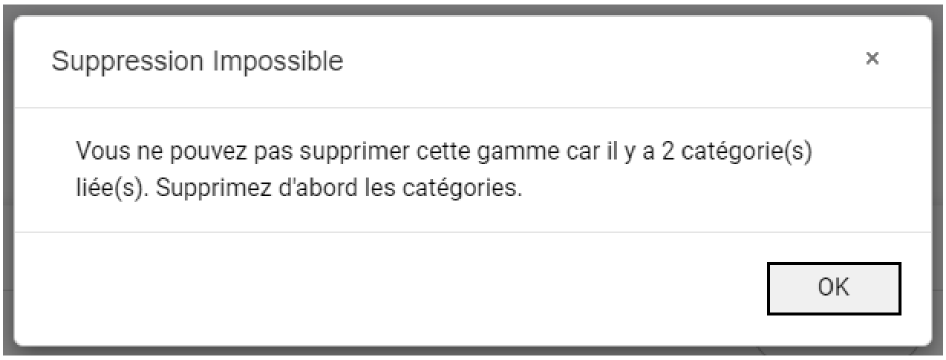
C'est pour cela qu'il sera utilisé dans l'entièreté du site.

Il est même possible de faire des messages de confirmation. Notamment lors de la suppression.




Concernant la suppression il n'y a eu aucun problème majeur lors de sa création.

J'ai aussi réussi à mettre un message d'erreur dans le cas où la gamme était déjà liée et donc non supprimable. Cependant pour cela j'ai eu besoin d'une aide externe car je n'arrivais pas à renvoyer les données voulues. Le problème était que je cherchais à faire une requête SQL plus compliqué que nécessaire.

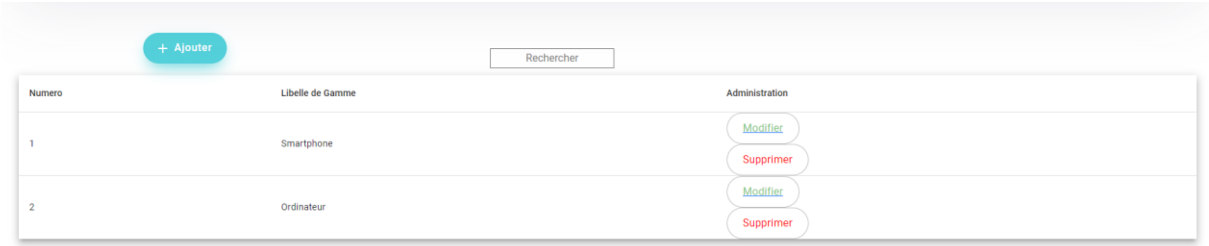



Voici son rendu dans le code :

```
alertifyjs.confirm("Confirmation", "Etes vous sûr de vouloir supprimer la gamme " + gamme.libelleGamme + " ?",()=>{
  this.gammeservice.deleteGammes(gamme.idGamme).subscribe((data:any) =>{

   if(data.data) {
     let nbCate = data.data.count;
     alertifyjs.alert("Suppression Impossible","Vous ne pouvez pas supprimer cette gamme car il y a " + nbCate + " catégorie(s) liée(s).
    } else {
     window.location.reload();
    }
```

L'affichage a été la première chose que j'ai faite pour pouvoir y ajouter au fil du temps les fonctionnalités citées au-dessus c'est-à-dire l'ajout, la modification ainsi que la suppression.

Voici un aperçu de la page :



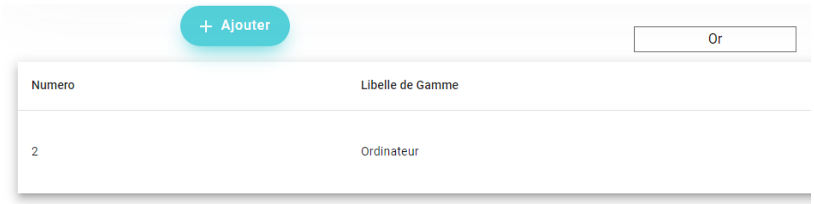

Le tableau a été fait grâce à Angular material, notre tuteur de stage a insisté sur le fait d'utiliser Angular Materials car cela lui permet de modifier le style plus facilement.

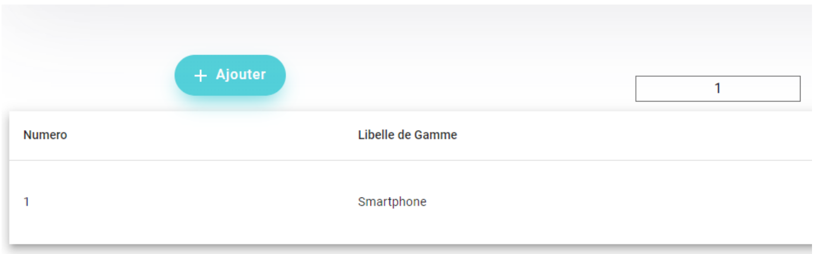
J'ai ajouté une fonction de recherche dynamique grâce à une bibliothèque se nommant ng2-search-filter

Grâce à cette bibliothèque, les recherches prennent en compte ce que retourne la requête d'affichage (select). Il suffit d'une ligne pour l'appliquer.

```
<table mat-table [dataSource]="gammes | filter:searchText" class="mat-elevation-z8">
```

Dans cet exemple, on sera capable de rechercher par nom de gamme comme par id.



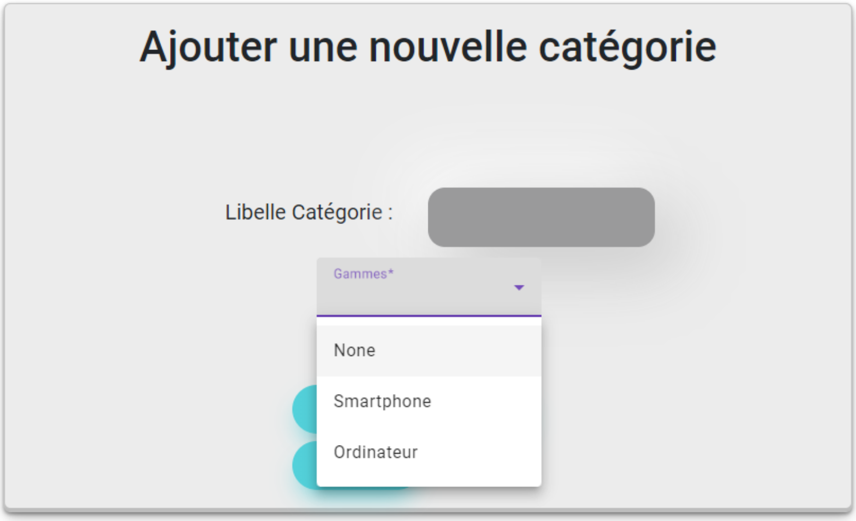




J'ai donc pu faire tout ce que je voulais concernant la gestion des gammes.

## 4) La gestion des catégories

J’ai décidé de partir sur cette tache ensuite car elle ressemblait à celle-ci fait précédemment.

Concernant l’ajout, voici à quoi il ressemble :



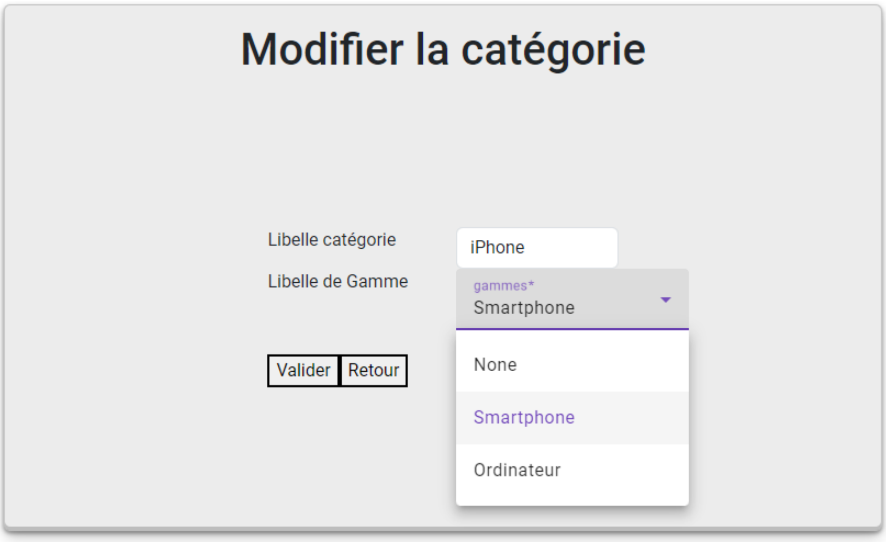

On reste sur globalement le même affichage.

Pour cette tâche j’ai hésité entre mettre id de la gamme en textbox ou mettre une liste déroulante, à ce moment-là, j’étais parti sur la textbox au cas où il y aurait trop de gammes, je n’arrivais pas aussi à utiliser Angular Materials à ce moment-là. J’ai posé la question à mon tuteur et il était d’accord mais c’était une erreur d’incompréhension. Il m’a fait part du changement quelques tâches plus tard. À ce moment-là, j’ai réussi à bien les utiliser les listes déroulantes et le changement a été assez simple à réaliser.

```
<p class="libelle">Libelle Catégorie :</p>
<input type="text" class="input" name="libelleCategorie" formControlName="libelleCategorie" required #libelleCategorieInput>
    <mat-form-field appearance="fill" class="cbGamme">
    <mat-label>Gammes</mat-label>
    <mat-select name="idGamme" formControlName="idGamme">
        <mat-option value="None">None</mat-option>
        <mat-option *ngFor="let gamme of gammes" [value]="gamme.idGamme" >
        {{gamme.libelleGamme}}
        </mat-option>
    </mat-select>
    </mat-form-field>
```

| idCategorie | libelleCategorie | idGammeCategorie |
|---|---|---|
| 1 | iPhone | 1 |
| 3 | Macbook | 2 |
| 4 | Windows | 2 |

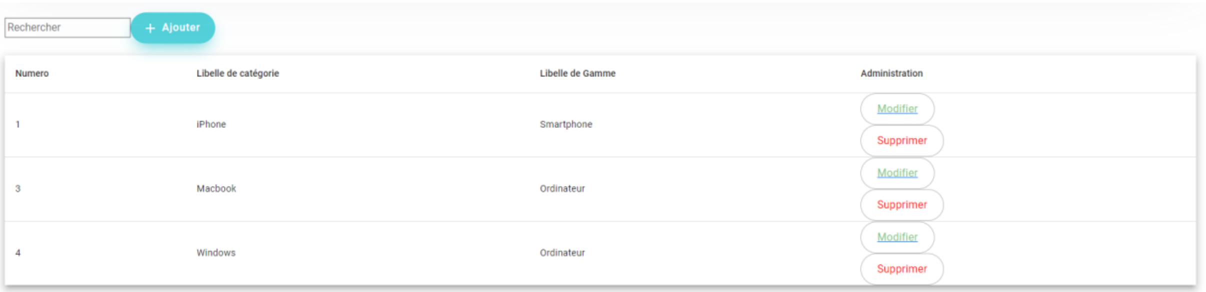
J’avais donc fait la même chose pour la modification, je l’ai changé sans problème elle aussi.




Néanmoins pour la première réalisation de celle-ci, j’avais fait une erreur de frappe dans mon api qui m’a bien ralenti. J’ai mis du temps à la trouver.

Je n’ai pas eu de problème pour la suppression.

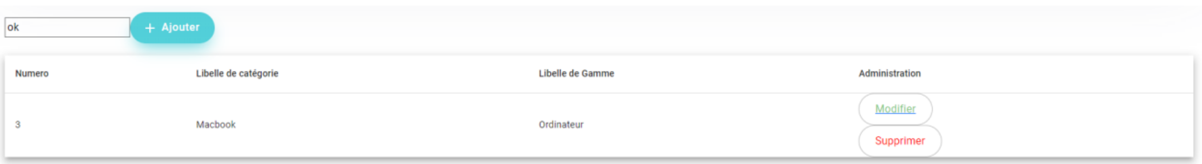
L’affichage reste le même aussi. Des messages d’erreur ont été aussi programmés comme la tâche précédente.



Rechercher | + Ajouter

| Numero | Libelle de catégorie | Libelle de Gamme | Administration |
|---|---|---|---|
| 1 | iPhone | Smartphone | Modifier Supprimer |
| 3 | Macbook | Ordinateur | Modifier Supprimer |
| 4 | Windows | Ordinateur | Modifier Supprimer |

La recherche fonctionne avec tous les critères.



ok | + Ajouter

| Numero | Libelle de catégorie | Libelle de Gamme | Administration |
|---|---|---|---|
| 3 | Macbook | Ordinateur | Modifier Supprimer |

J’ai donc pu faire tout ce que je voulais pour la recherche des catégories.

Suite à la fin de cette tâche, notre chef de projet a dû réaliser la première fusion du projet, je l'ai donc soutenu en adaptant au mieux mon code (arrangement de fichier, règle de nommage). Le système d'authentification étant aussi opérationnel, j'ai pu sécuriser mes pages admin.

```
if(!localStorage.getItem('authUser')) {
  this.router.navigate(['/']);
} else {
  let user = JSON.parse(localStorage.getItem('authUser')!)
  if(user.roles != "admin_boutique") {
    this.router.navigate(['/405']);
```

On va d'abord vérifier si l'utilisateur est connecté puis on va vérifier s'il a le bon rôle. Sinon, on le fait retourner à la page d'accueil.

### 5) La gestion du carrousel

La gestion du carrousel a été un peu plus compliqué même s'il a été plus rapide.

Le problème étant la recherche d'un carrousel fonctionnel, au début j'ai essayé plusieurs carrousels sans qu'aucun n'arrive à s'installer. Quand j'en ai fait part au tuteur de stage, il a recherché un peu de son côté et m'a envoyé des liens, j'avais déjà visité la plupart des sites. Mais j'ai décidé de consacrer plus de temps sur un en particulier : ngx-owl-carousel-o. C'est le seul que j'ai réussi à au moins installer, j'ai donc tout misé sur lui.

Voici ce qu'il donne quand il est intégré au site.


Dans ce carrousel sont affichées les différentes pubs du site, on peut slider directement dessus ou utiliser les flèches directionnelles en dessous pour changer de pub. En cliquant dessus, cela nous redirigera vers le site externe du client. On peut voir des infos sur la pub, ces infos seront développées dans la prochaine tâche.

Côté code, voilà à quoi ressemble le carrousel :

```
customOptions: OwlOptions = {
  loop: true,
  mouseDrag: true,
  touchDrag: true,
  pullDrag: false,
  dots: false,
  navSpeed: 700,
  navText: ['<', '>'],
  responsive: {
    0: {
      items: 1
    },
  },
  nav: true
}
```

```
<owl-carousel-o [options]="customOptions" >


  <ng-container *ngFor="let pub of pubs">
    <ng-template carouselSlide [id]="pub.idPub">
      <a (click)="redirectToPub(pub.idPub)">
         <div class="titreProduit">
          <h3>{{pub.libellePub}}</h3>
          <h6>{{pub.ligne1}}</h6>
          <h6>{{pub.ligne2}}</h6>
          <h6>{{pub.ligne3}}</h6>
        </div>
        <img class="carousel img-carrousel"  src="http://localhost/api/photos/pubs/{{pub.images[0].libelleImagePub}}"
      </a>

    </ng-template>
  </ng-container>

</owl-carousel-o>
```

Au début il y a eu un problème d'incompréhension et j'ai affiché les produits au lieu des pubs. Ce problème a bien évidemment été réglé.

Pour les autres carrousels du site, mes coéquipiers ont repris ma méthode. La seule chose que je n'ai pas réussi à faire, c'est de faire défiler automatiquement les pubs. Mais quelqu'un plus tard a réussi à trouver la solution.

### 6) La gestion des pubs

Comme dit précédemment, j'ai affiché des produits au lieu des pubs. Le fait est que la base de données ne contenait pas de table publicités. J'ai donc dû la créer.

Voici la composition de la table publicités :

| | # | Nom | Type | Interclassement | Attributs | Null | Valeur par défaut | Commentaires | Extra | Action | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ☐ | 1 | **idPub** | int | | | Non | *Aucun(e)* | | AUTO_INCREMENT | Modifier | Supprimer | Plus |
| ☐ | 2 | **libellePub** | varchar(255) | utf8mb4_0900_ai_ci | | Non | *Aucun(e)* | | | Modifier | Supprimer | Plus |
| ☐ | 3 | **ligne1** | varchar(255) | utf8mb4_0900_ai_ci | | Oui | *NULL* | | | Modifier | Supprimer | Plus |
| ☐ | 4 | **ligne2** | varchar(255) | utf8mb4_0900_ai_ci | | Oui | *NULL* | | | Modifier | Supprimer | Plus |
| ☐ | 5 | **ligne3** | varchar(255) | utf8mb4_0900_ai_ci | | Oui | *NULL* | | | Modifier | Supprimer | Plus |
| ☐ | 6 | **lien** | varchar(225) | utf8mb4_0900_ai_ci | | Non | *Aucun(e)* | | | Modifier | Supprimer | Plus |

Le libelle fait office de titre, de plus notre tuteur m'a demandé de mettre 3 lignes distinctes, cela a pour but de faciliter l'affichage. On a ensuite le lien vers le site externe qui sera utilisé pour rediriger vers celui-ci.

De plus, il fallait que l'on puisse mettre plusieurs images pour une publicité. J'ai donc dû créer une 2ème table, la table imagepub.

Voici la composition de la table imagepub :

| # | Nom | Type | Interclassement | Attributs | Null | Valeur par défaut | Commentaires | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|
| 1 | idImagePub | int | | | Non | *Aucun(e)* | | AUTO_INCREMENT | Modifier Supprimer Plus |
| 2 | libelleImagePub | varchar(225) | utf8mb4_0900_ai_ci | | Non | *Aucun(e)* | | | Modifier Supprimer Plus |
| 3 | idPubliciteImage | int | | | Non | *Aucun(e)* | | | Modifier Supprimer Plus |

Bien évidemment, il me fallait faire la gestion des publicités pour l'admin.

Pour cette même gestion, je me suis grandement inspiré de mon coéquipier qui avait un peu prête la même composition pour les produits.


Concernant l'affichage côté client, on peut les retrouver tout autour du carrousel et sur la gauche des listes des produits.

Chaque publicité est récupérée via une requête SQL qui affiche les publicités dans un ordre aléatoire.

```php
$sql = "SELECT * FROM publicites ORDER BY RAND() limit 1";
$stmt = $conn->prepare($sql);
$stmt->execute();


if($stmt->rowCount() > 0) {


        $dataR = $stmt->fetch(PDO::FETCH_ASSOC);



    $id = $dataR['idPub'];


    $sql = "SELECT idImagePub, libelleImagePub, idPubliciteImage FROM imagepub WHERE idPubliciteImage = $id";
    $stmt = $conn->prepare($sql);

    $stmt->execute();
        $dataR['imagesRand'] = $stmt->fetchAll(PDO::FETCH_ASSOC);



    echo json_encode([
        'success' => 1,
        'dataR' => $dataR,
    ]);
```

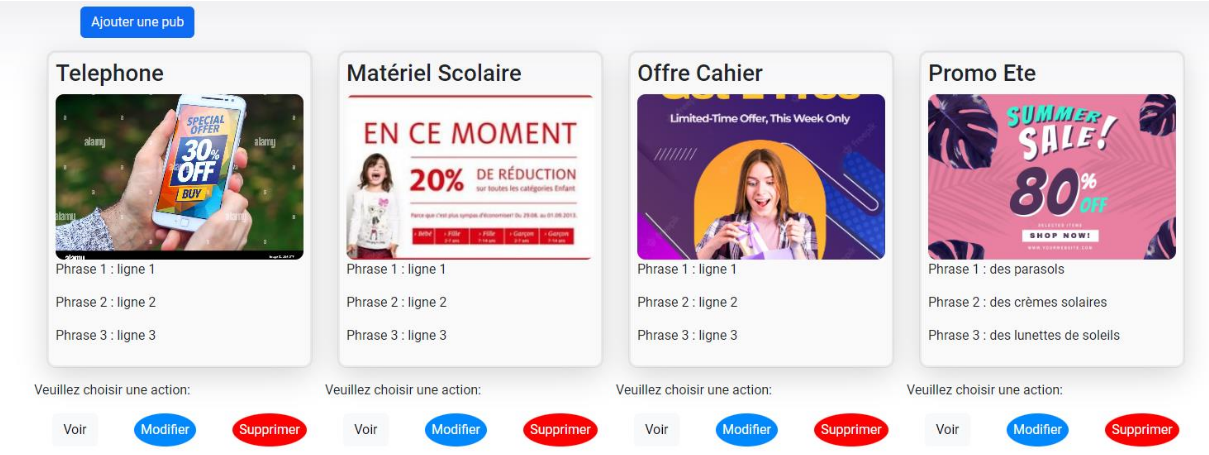
Ma requête retourne bien des objets publicités aléatoire.

```
▼Array(6)
 ▶0: {idPub: 28, libellePub: 'Telephone', ligne1: 'ligne 1', ligne2: 'ligne 2', ligne3: 'ligne 3', …}
 ▶1: {idPub: 29, libellePub: 'Promo Ete', ligne1: 'des parasols', ligne2: 'des crèmes solaires', ligne3: 'des lunettes de soleils', …}
 ▶2: {idPub: 27, libellePub: 'Offre Cahier', ligne1: 'ligne 1', ligne2: 'ligne 2', ligne3: 'ligne 3', …}
 ▶3: {idPub: 27, libellePub: 'Offre Cahier', ligne1: 'ligne 1', ligne2: 'ligne 2', ligne3: 'ligne 3', …}
 ▶4: {idPub: 29, libellePub: 'Promo Ete', ligne1: 'des parasols', ligne2: 'des crèmes solaires', ligne3: 'des lunettes de soleils', …}
 ▶5: {idPub: 27, libellePub: 'Offre Cahier', ligne1: 'ligne 1', ligne2: 'ligne 2', ligne3: 'ligne 3', …}
  length: 6
 ▶[[Prototype]]: Array(0)
```

Côté HTML, l'affichage se fait en prenant les index du résultat pour que la publicité affichée ne soit pas toutes les mêmes

```html
<div class="col-12">
    <!-- Ici Publicité zone 3 - sans Caption (Slide en boucle) -->
    <a href="{{randomPub[2].lien}}"><img class="pub2" src="http://localhost/api/photos/pubs/{{randomPub[2].imagesRand[0].l
    <!-- <a href="#"
        style="background: url('https://ce1.uicdn.net/fd1/76ac2a7243337b63c75022501aa2b/webapp/35222-221339_TV-Campaign-FR
</div>
<div class="col-12">
    <!-- Ici Publicité zone 4 - sans Caption (Slide en boucle) -->
    <a href="{{randomPub[3].lien}}"><img class="pub2" src="http://localhost/api/photos/pubs/{{randomPub[3].imagesRand[0].l
    <!-- <a href="#"
        style="background: url('data:image/jpeg;base64,/9j/4AAQSkZJRgABAQAAAQABAAD/2wCEAAoHCBERDxEPDxIREhAPERAPEQ8PDxERDw8
</div>
```

Concernant l'affichage côté admin, voici à quoi il ressemble :




J'ai opté pour mettre des données dans des cards vu que cette fois, il y a pas mal d'informations. En matière de nouvelles fonctionnalités, on peut voir la pub sélectionnée pour avoir plus de détails sur lui. Dans ce cas-là, cela nous de voir le lien assigné à la pub.


De plus, cela nous permet de voir les plusieurs images assignées à la publicité. De base avec la requête assignée, je ne pouvais pas mettre les images dans l'objet publicité. Il a donc fallu ajouter les images dans l'objet d'une manière un peu spéciale.

```
this.pubService.getPub().subscribe((data:any) => {
  data = data.data;
  for(let i = 0; i < data.images.length; i++) {

    if(this.imagePub.has(data.images[i].idPubliciteImage)) {
      this.imagePub.get(data.images[i].idPubliciteImage).push(data.images[i]);
    } else {
      this.imagePub.set(data.images[i].idPubliciteImage, []);
      this.imagePub.get(data.images[i].idPubliciteImage).push(data.images[i]);
    }
  }
  for (let y = 0; y < Object.keys(data).length -1; y++) {
    data[y].images = this.imagePub.get(data[y].idPub);
    this.pubs.push(data[y]);
  }
  console.log(this.pubs)
})
```

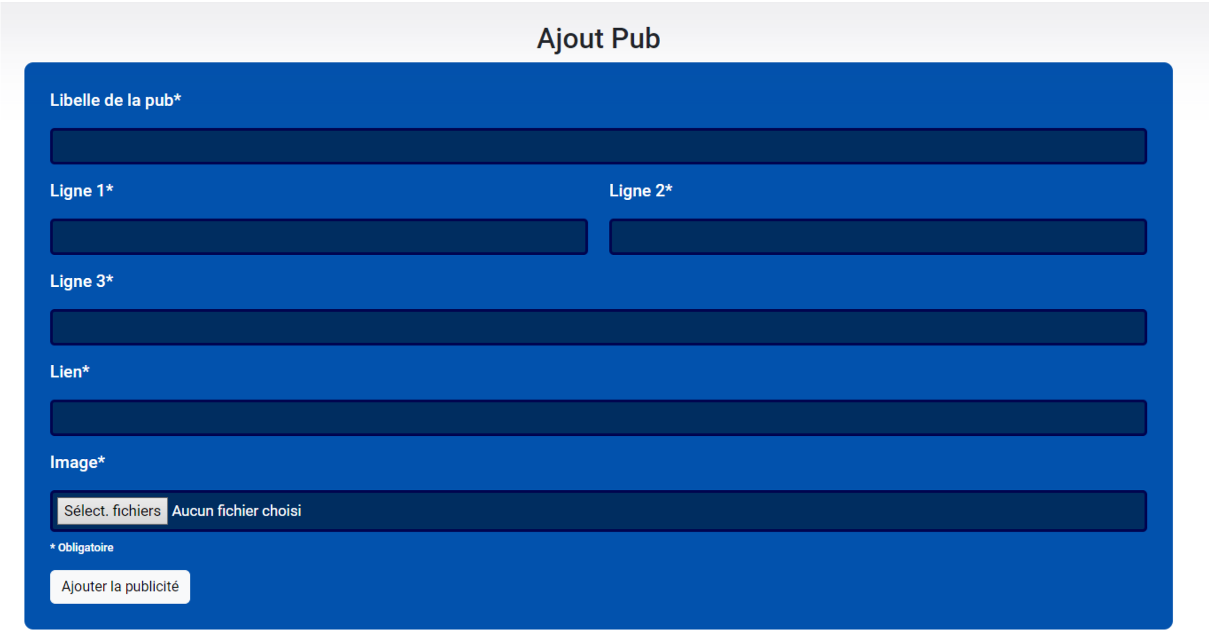
On va d'abord récupérer les images à leur emplacement initial puis les assigner à chaque objet en fonction de leur id.

Ce qui nous permet d'arriver à ce résultat :

```
▼4:
   idPub: 30
  ▼images: Array(2)
    ▶0: {idImagePub: 28, libelleImagePub: '30-0.jpg', idPubliciteImage: 30}
    ▶1: {idImagePub: 29, libelleImagePub: '30-1.png', idPubliciteImage: 30}
     length: 2
    ▶[[Prototype]]: Array(0)
   libellePub: "Promo Ete"
   lien: "www.amazon.com"
   ligne1: "des parasols"
   ligne2: "des lunettes de soleil"
   ligne3: "etc"
```

Cela facilite donc la manipulation des informations pour afficher les valeurs.

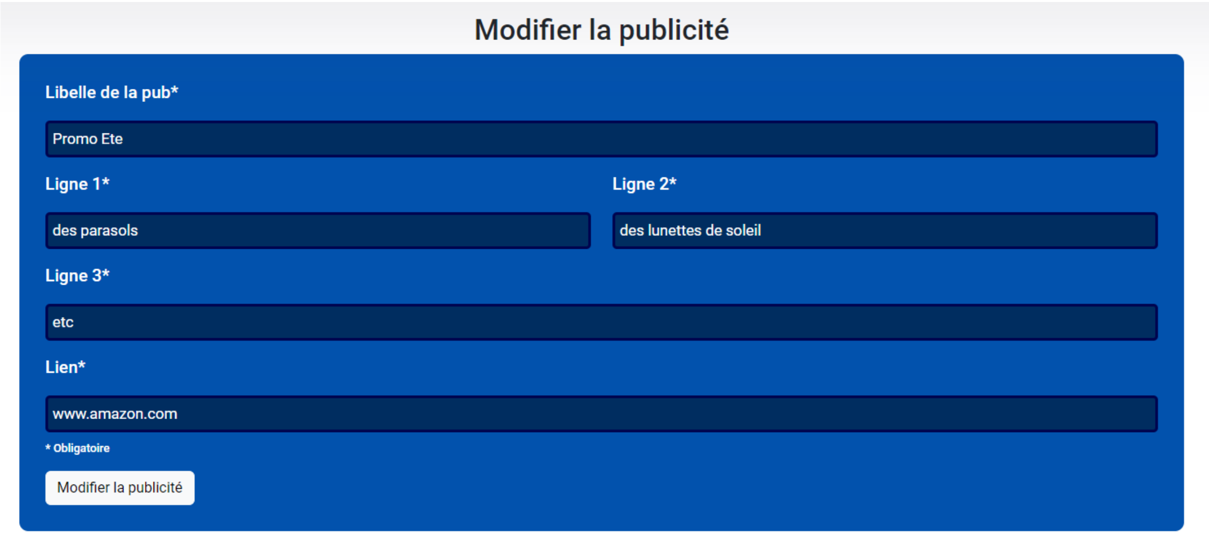
Voici l'interface de l'ajout :




Dans le code, cela donne lieu évidemment à un formulaire plus long.

```
if(this.selectedImages.length > 0) {
  const formData = new FormData();
  for (let i = 0; i < this.selectedImages.length; i++) {
      formData.append('imagePub' + i, this.selectedImages[i], this.selectedImages[i].name);
  }
  formData.append('libellePub', this.addPubForm.value.libellePub);
  formData.append('ligne1', this.addPubForm.value.ligne1);
  formData.append('ligne2', this.addPubForm.value.ligne2);
  formData.append('ligne3', this.addPubForm.value.ligne3);
  formData.append('lien', this.addPubForm.value.lien);
  console.log(formData)
  this.PubService.createPub(formData).subscribe((data:any) => {
  })
  this.router.navigate(['/view-publicites']);
}
```

Voici l’interface de la modification :




On ne peut malheureusement pas modifier les images.

```
this.modifPubForm = this.formBuilder.group({
  idPub:[],
  libellePub: ['', Validators.required],
  ligne1: ['', Validators.required],
  ligne2: ['', Validators.required],
  ligne3: ['', Validators.required],
  lien: ['', Validators.required],

})
```

Pour cette tâche, j’ai dû changer de méthode dans l’api en utilisant des méthodes POST. Les images étaient plus compliquées à gérer avec l’ancienne méthode.

```
$libellePub = str_replace("\'", "\'\'",$_POST['libellePub']);
$ligne1 = str_replace("\'", "\'\'",$_POST['ligne1']);
$ligne2 = str_replace("\'", "\'\'",$_POST['ligne2']);
$ligne3 = str_replace("\'", "\'\'",$_POST['ligne3']);
$lien = str_replace("\'", "\'\'",$_POST['lien']);

$imagePub = array();

if(isset($_FILES['imagePub0'])) {
    array_push($imagePub, $_FILES['imagePub0']);
}
if(isset($_FILES['imagePub1'])) {
    array_push($imagePub, $_FILES['imagePub1']);
}
if(isset($_FILES['imagePub2'])) {
    array_push($imagePub, $_FILES['imagePub2']);
}

if(!isset($libellePub) || !isset($lien)) {
    echo json_encode([
        'success' => 0,
        'message' => 'Enter all values.'
    ]);
}
```

Avec cette méthode, les apostrophe ne pouvaient pas être rentrés, pour pallier ce problème, j’ai utilisé des **str_replace** pour pouvoir rendre cet ajout possible.

Pour la suppression, il n’y a pas eu de problème.

### 7) Les recherches de produit à partir de la page d’accueil

Pour cette tâche j’étais chargé de créer les recherches respectives par nom, catégorie et gamme.

Malheureusement, la bibliothèque utilisée sur les autres pages ne pouvait pas marcher, car les données à filtrer n’étaient pas sur la bonne page.

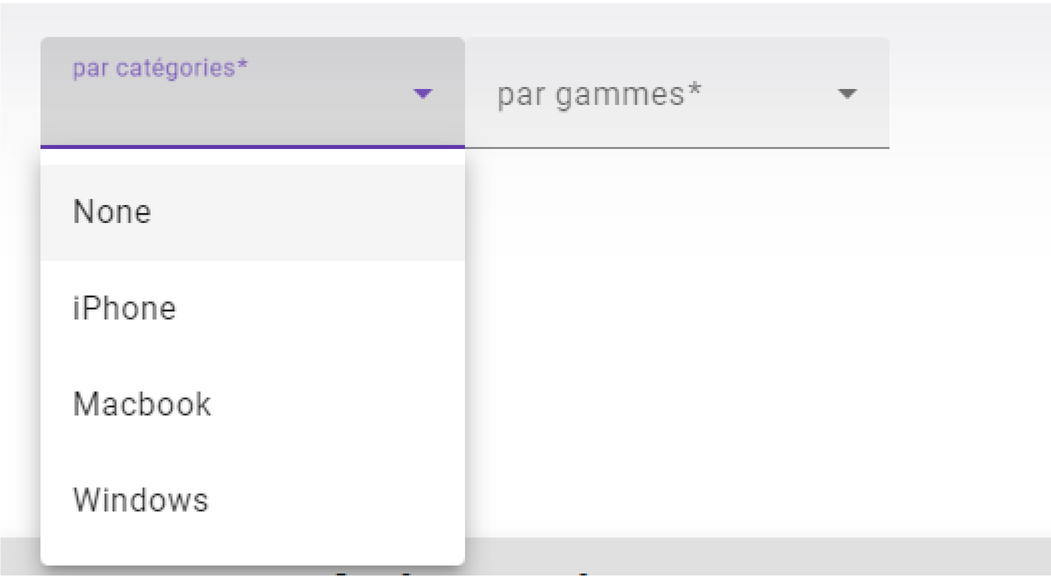
J’ai commencé par la recherche par nom, car c’était la plus simple en plus d’être la plus importante. J’ai quand même eu quelques problèmes étant donné que je devais modifier une page que je n’avais pas faite, celle des produits.


Voici le fichier permettant de récupérer les données :

```
try
{

    $libelleProduit = $_GET['libelleProduit'];

    $sql = "SELECT idProduit, libelleProduit, qtStock, descriptionProduit, prixUnitHT, idCategorieProduit
            FROM `produits` WHERE libelleProduit
            LIKE '%$libelleProduit%'";
    $stmt = $cnx->prepare($sql);
    $stmt->execute();


    if ($stmt->rowCount() > 0) :

        $data = null;
        if (is_numeric($annonceur_id)) {
            $data = $stmt->fetch(PDO::FETCH_ASSOC);
        } else {
            $data = $stmt->fetchAll(PDO::FETCH_ASSOC);
        }

        $sql = "SELECT idImage, libelleImage, idProduitImage
                FROM image
                INNER JOIN produits ON idProduitImage = idProduit
                WHERE libelleProduit LIKE '%$libelleProduit%'";
        $stmt = $cnx->prepare($sql);
        $stmt->execute();
        $data['images'] = $stmt->fetchAll(PDO::FETCH_ASSOC);

        echo json_encode([
            'success' => 1,
            'data' => $data,
        ]);
```

On regarde si la recherche correspond à un des produits.

Les deux autres recherches fonctionnent par listes déroulantes.



Voici comment les listes déroulantes sont affichées :

```
<form [formGroup]="nameForm" novalidate class="form">
    <mat-form-field appearance="fill">
    <mat-label>par catégories</mat-label>
    <mat-select name="idCategorieProduit" formControlName="idCategorieProduit">
        <mat-option value="0">None</mat-option>
        <mat-option *ngFor="let categorie of categories" [value]="categorie.idCategorie
        {{categorie.libelleCategorie}}
        </mat-option>
    </mat-select>
    </mat-form-field>

    <mat-form-field appearance="fill">
        <mat-label>par gammes</mat-label>
        <mat-select name="idGammeCategorie" formControlName="idGammeCategorie">
            <mat-option value="0">None</mat-option>
            <mat-option *ngFor="let gamme of gammes" [value]="gamme.idGamme">
            {{gamme.libelleGamme}}
            </mat-option>
        </mat-select>
    </mat-form-field>
</form>
<button class="btn" (click)="redirectToAllProductCategorieGamme()">Rechercher</button>
```

Les problèmes qui sont arrivés derrière sont les routes. En effet, ces 3 recherches retournaient vers la même route et donc même si le paramètre changeait, celui-ci restait à la même place.
Pour pallier ce problème, j'ai décidé de créer une route contenant les 3 informations.

```
{ path: 'view-annonces/:libelleProduit/:idCategorieProduit/:idGammeCategorie'
```

Sauf que je me suis retrouvé avec un problème de condition m'obligeant à utiliser cette route même quand aucun paramètre n'est entré. J'ai laissé passer quelques jours avant de régler ce problème. Je l'ai réglé avec la condition suivante :

```
else if(this.url.snapshot.params['libelleProduit'] == undefined
        && this.url.snapshot.params['idCategorieProduit'] == undefined
        && this.url.snapshot.params['idGammeCategorie'] == undefined)
        {
```

Ce qui me permet de mettre une route contenant moins d'informations, renforçant la sécurité.

| Avant | | Après |
|---|---|---|
| localhost:4200/view-annonces/undefined/0/0 | → | localhost:4200/view-annonces |

## 8) Filtre des produits

Pour cette tâche, je dois filtrer les produits par nombre de notes, nombre de vues, mais aussi avec le nombre de ventes.

J'ai dû le faire 3 fois en tout pour chaque tri. Un dans la page des produits, les 2 autres dans la page d'accueil (1 pour les mises en avant/ 1 pour les mises en avant de catégorie). J'ai commencé par la page des produits. Voici les requêtes :

Par nombre de vues :

```
$sql = "SELECT idProduit, idAnnonceurProduit, libelleProduit, qtStock, descriptionProduit, prixUnitHT,
idCategorieProduit, libelleNomAnnonceur, AVG(CASE WHEN Avis.valid = 1 THEN Avis.noteAvis ELSE NULL END) as averageNote,
MAX(Image.libelleImage) as image, count(idProduitView) as NbVues
FROM produits
LEFT JOIN annonceurs ON idAnnonceur = idAnnonceurProduit
LEFT JOIN Avis ON Produits.idProduit = Avis.idProduitAvis
LEFT JOIN Image ON Produits.idProduit = Image.idProduitImage
LEFT JOIN views ON idProduitView = idProduit
WHERE idProduit IN (SELECT idAnnonceProduit FROM Annonce WHERE idEtatAnnonce = 2)
OR adminProduit = 1
GROUP BY idProduit
ORDER BY NbVues DESC";
```

Par nombre de ventes :

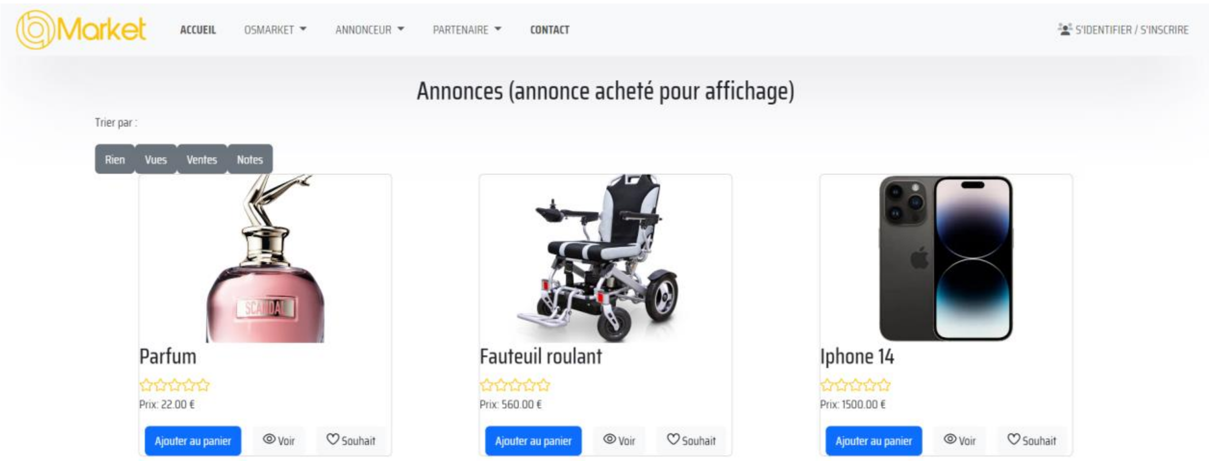
```
$sql = "SELECT idProduit, idAnnonceurProduit, libelleProduit, qtStock, descriptionProduit, prixUnitHT,
idCategorieProduit, libelleNomAnnonceur, AVG(CASE WHEN Avis.valid = 1 THEN Avis.noteAvis ELSE NULL END) as averageNote,
MAX(Image.libelleImage) as image, SUM(qtProduitComm) as NbVendu
FROM produits
LEFT JOIN annonceurs ON idAnnonceur = idAnnonceurProduit
LEFT JOIN Avis ON Produits.idProduit = Avis.idProduitAvis
LEFT JOIN Image ON Produits.idProduit = Image.idProduitImage
LEFT JOIN commander ON idProduitComm = idProduit
WHERE idProduit IN (SELECT idAnnonceProduit FROM Annonce WHERE idEtatAnnonce = 2)
OR adminProduit = 1
GROUP BY idProduit
ORDER BY NbVendu DESC";
```

Par nombre de notes :

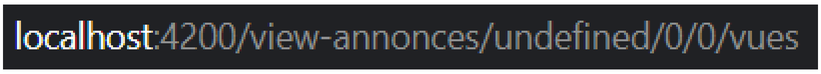
```
$sql = "SELECT idProduit, idAnnonceurProduit, libelleProduit, qtStock, descriptionProduit, prixUnitHT,
idCategorieProduit, libelleNomAnnonceur, AVG(CASE WHEN Avis.valid = 1 THEN Avis.noteAvis ELSE NULL END) as averageNote,
MAX(Image.libelleImage) as image, count(idProduitAvis) as NbNotes
FROM produits
LEFT JOIN annonceurs ON idAnnonceur = idAnnonceurProduit
LEFT JOIN Avis ON Produits.idProduit = Avis.idProduitAvis
LEFT JOIN Image ON Produits.idProduit = Image.idProduitImage
WHERE idProduit IN (SELECT idAnnonceProduit FROM Annonce WHERE idEtatAnnonce = 2)
OR adminProduit = 1
GROUP BY idProduit
ORDER BY NbNotes DESC";
```

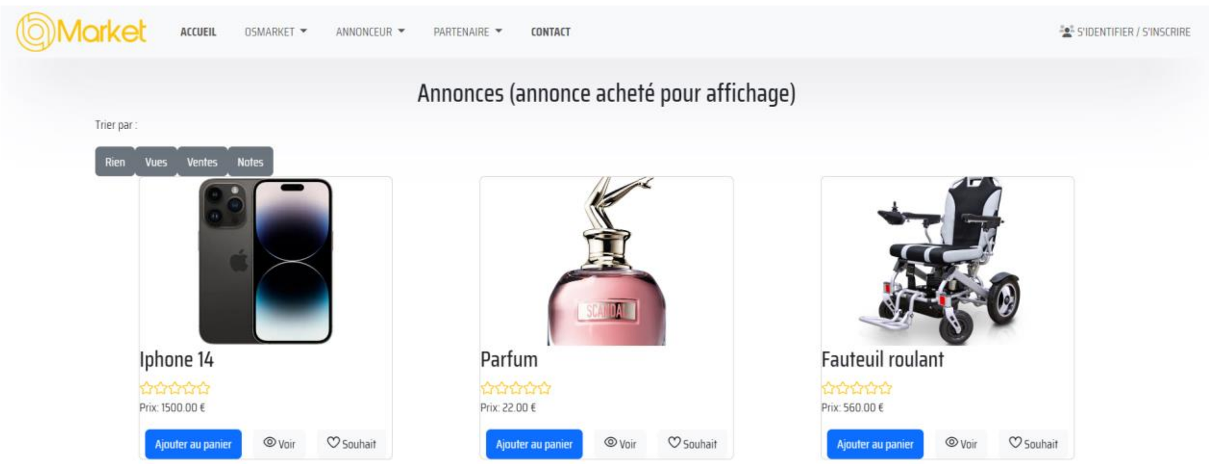
Ces requêtes sont pratiquement les mêmes sur la page d'accueil.

On peut choisir quel tri effectuer grâce à ces boutons.


Je varie les requêtes utilisées grâce à la route empruntée.
Pour les plus vus, voici un aperçu.



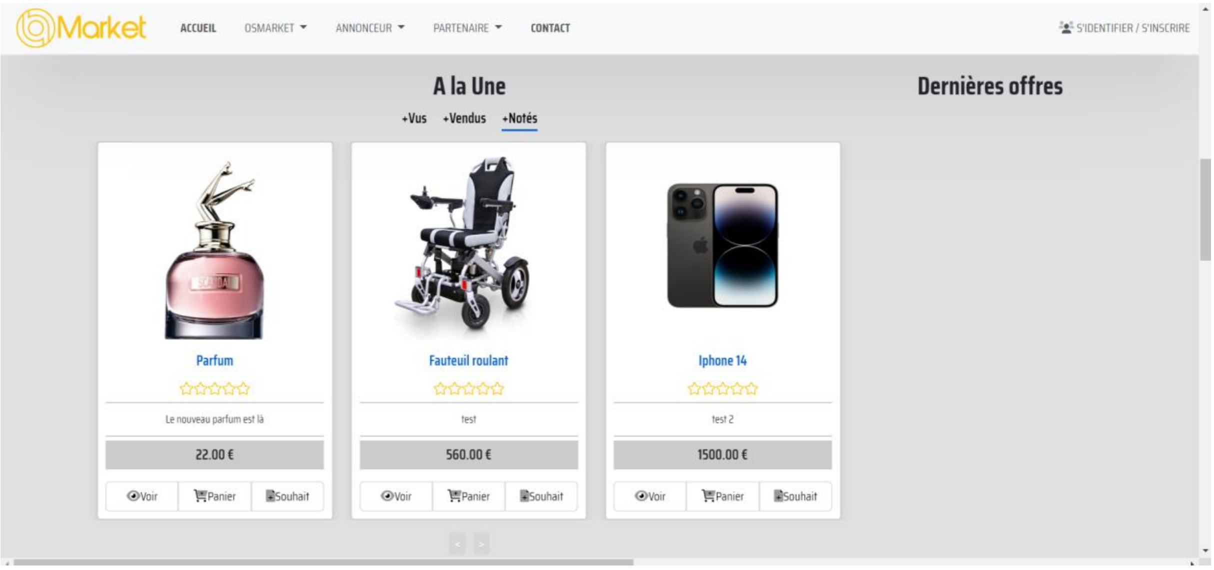

Si on trie par vues, on constate que l'ordre ressorti n'est pas le même.

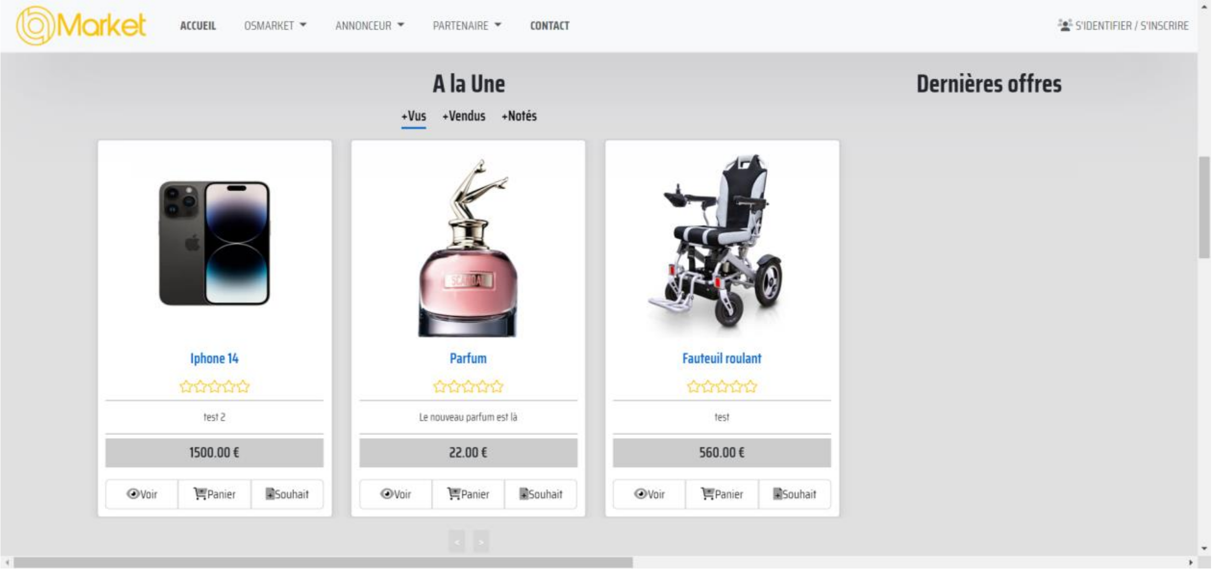
Concernant la partie page d'accueil, voici comment on sélectionne l'ordre.


On peut constater que le type de tri actif se voit souligné.
Cela est possible grâce aux class dynamiques. En fonction de la valeur de la variable isActive, la classe active s'activera, permettant le soulignage du tri sélectionné.

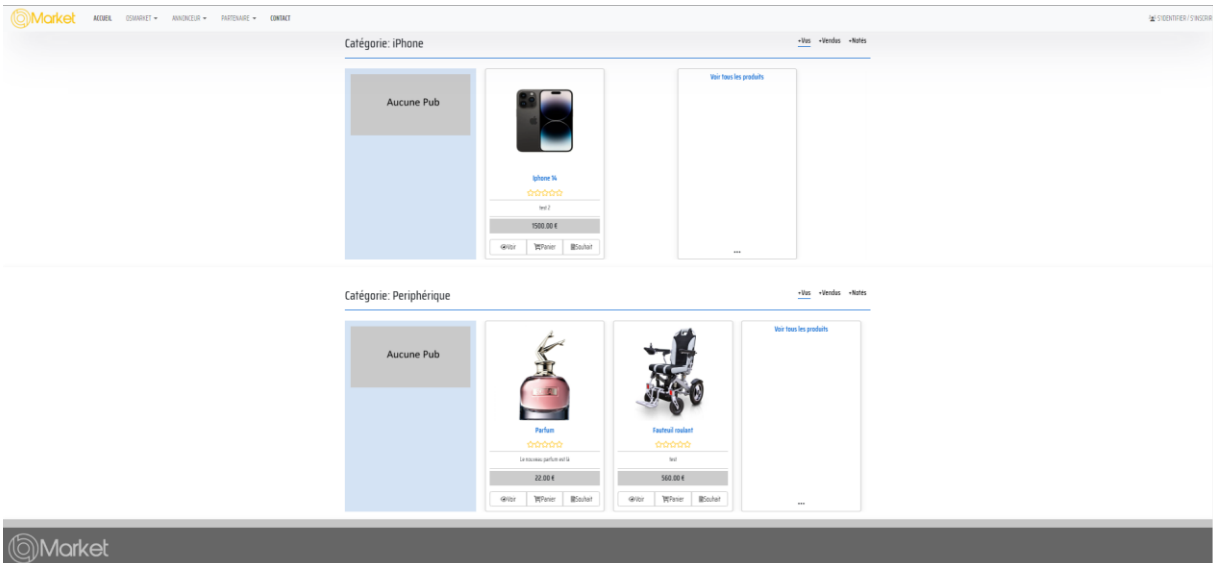
```
<ul>
    <li><a [ngClass]="{'active': isActive == 'vues'}"   href="/tri/vues">+Vus</a></li>
    <li><a [ngClass]="{'active': isActive == 'vendus'}"   href="/tri/vendus">+Vendus</a></li>
    <li><a [ngClass]="{'active': isActive == 'notes'}"   href="/tri/notes">+Notés</a></li>
</ul>
```

Voici plusieurs exemples des produits à la une.






Et voici à quoi ressemble la partie catégorie, fonctionnant de la même manière que celles au-dessus concernant le tri.




Cette tâche était assez redondante, car c'était globalement les mêmes requêtes à chaque fois, mais j'ai quand même dû réfléchir sur le moyen de transférer les données sur les bonnes pages.

Quelques jours avant la fin du stage, notre tuteur nous a demandé de finalement supprimer les messages d'erreurs sous alertifyjs pour les remplacer par des alerts Angular Materials pour qu'elles soient plus simples à styliser pour lui.

## V. Le bilan

Le projet global a été fini à environ 90%. Malheureusement, notre tuteur nous a fait part de tâches supplémentaires le dernier jour du stage. Ils étaient donc compliqués pour les personnes concernées de tout faire dans les temps.

Concernant ma partie, les seuls regrets que j'ai sont de ne pas avoir pu m'occuper de la modification des images par manque de temps. Mais tout le reste est bien fonctionnel. Néanmoins, la non-finition du projet laisse un goût amer.

Ce stage m'aura permis d'apprendre une nouvelle techno de zéro : Angular. J'ai eu moins de mal que prévu à l'apprendre. A défaut de ne pas être en présentielle, ce fut une belle expérience en télétravail. J'en retiendrai un travail sérieux, accompagné d'une bonne ambiance ainsi qu'une bonne équipe. Comparé à mon stage de l'année dernière réalisé dans une association, j'ai trouvé ce stage beaucoup plus professionnel. Un stage qui me rapproche bien plus du monde du travail.