



wallhaven

ANGULAR
S4B1

PROJET OUR WALLPAPER

1 PRÉSENTATION DE L'API	p.3
2 PROBLÉMATIQUE	p.3
3 MAQUETTE	p.4
4 SITE FINAL	p.5
5 UTILISATION DE L'API	p.7
6 MISE EN PAGE HTML ET UTILISATION DE BOOTSTRAP	p.8
7 LES DIFFICULTÉES RENCONTRÉES	p.10
8 PISTES D'AMÉLIORATION	p.10

Présentation de l'api

Nous avons décidé d'utiliser le plugin du site WallHaven, un site qui répertorie des fonds d'écran pour ordinateur. Il dispose d'une base de données de plus de 650 000 fonds d'écran. C'est une API qui ne nécessite pas d'API Key mis à part pour les contenus NSFW.



Problématique initiale :

Après consultation du site de WallHaven, nous nous sommes rendus compte que le site n'était pas très esthétique, avec beaucoup trop d'informations, nous avons donc voulu recréer un site plus sobre et actuel, en intégrant des fonctionnalités basiques qui permettent de ne pas perdre l'utilisateur dans sa navigation.

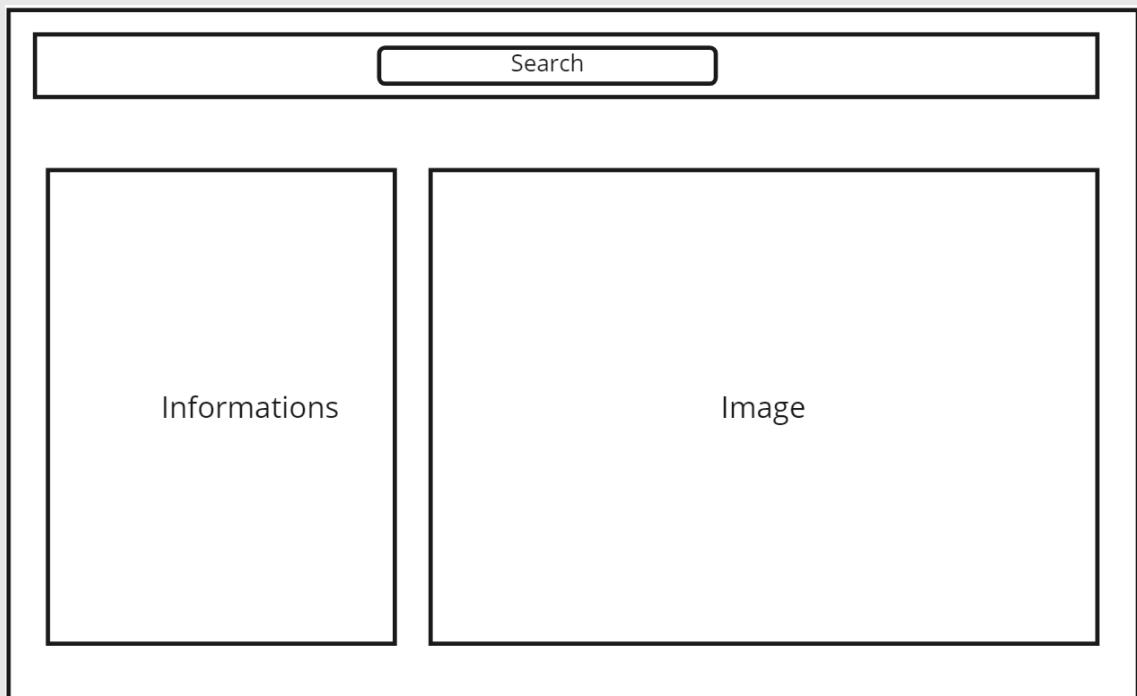
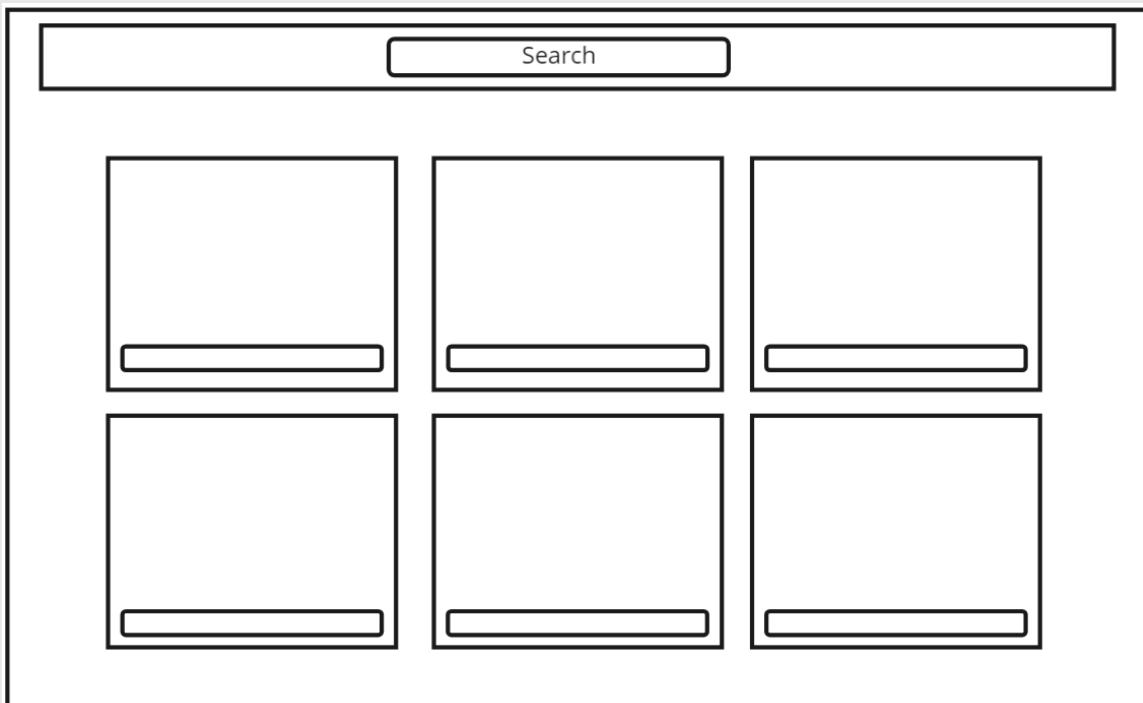
Maquette

Voici nos premières maquettes, nous voulions deux pages principales, la première qui répertorie tous les fonds d'écran, avec une barre de recherche et des informations sur ceux-ci et la seconde qui nous affiche le fond d'écran choisi en plus grand avec ses caractéristiques.

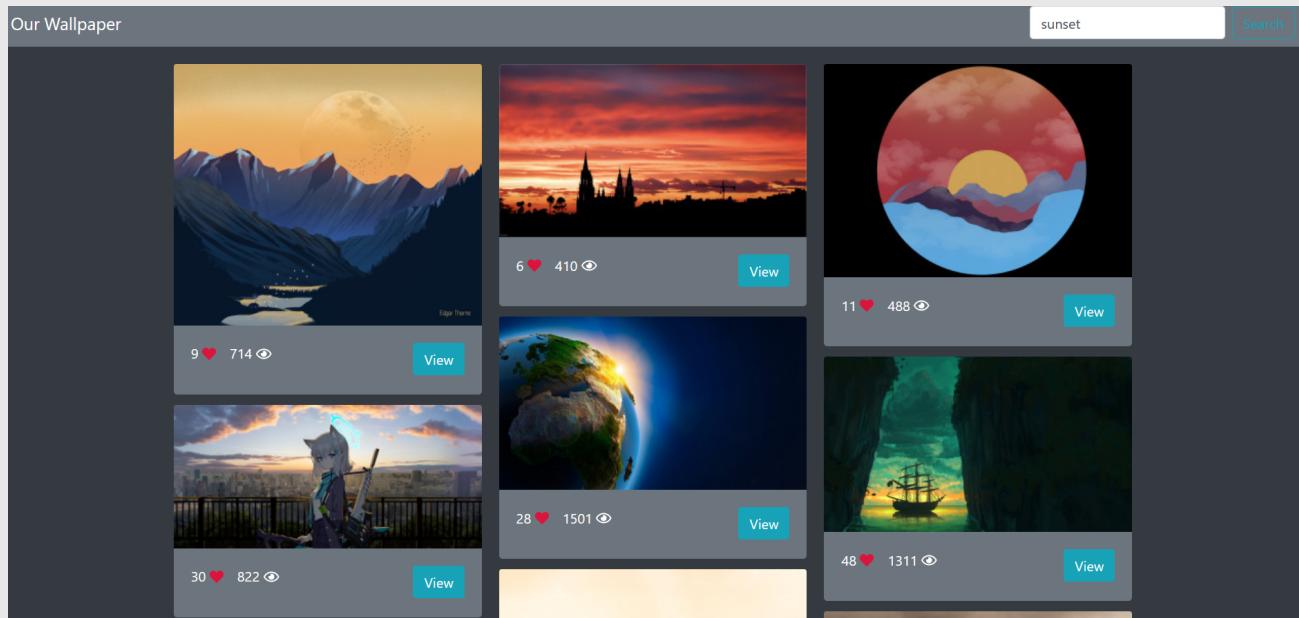
Lien du projet :

GITHUB : <https://github.com/FlippyArt/Projet-angular/tree/main/Projet>

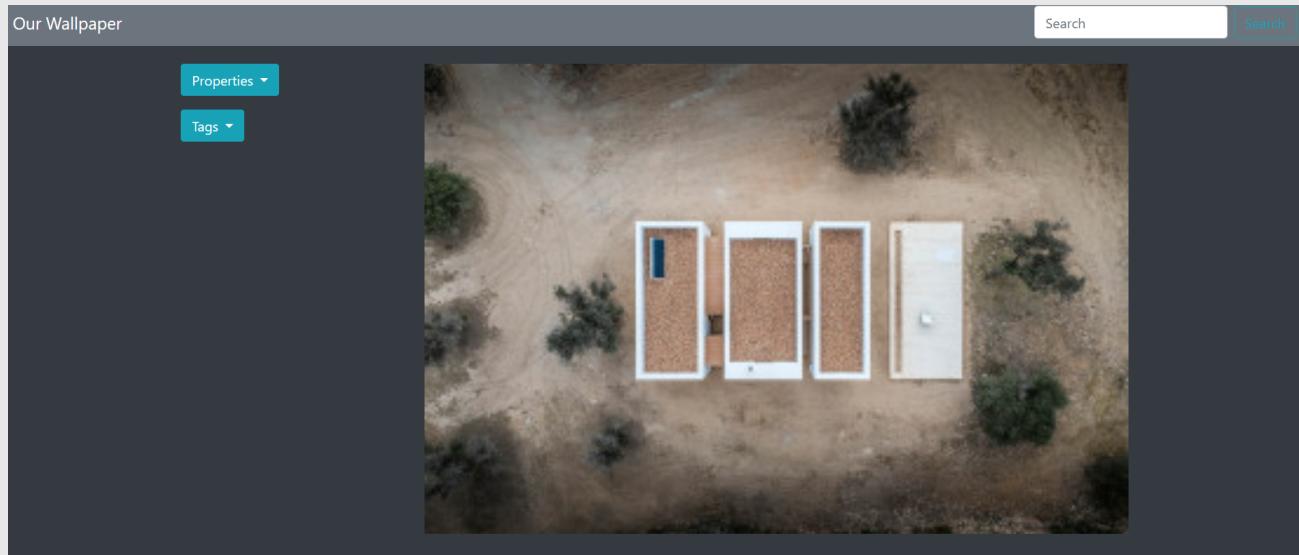
Maquette



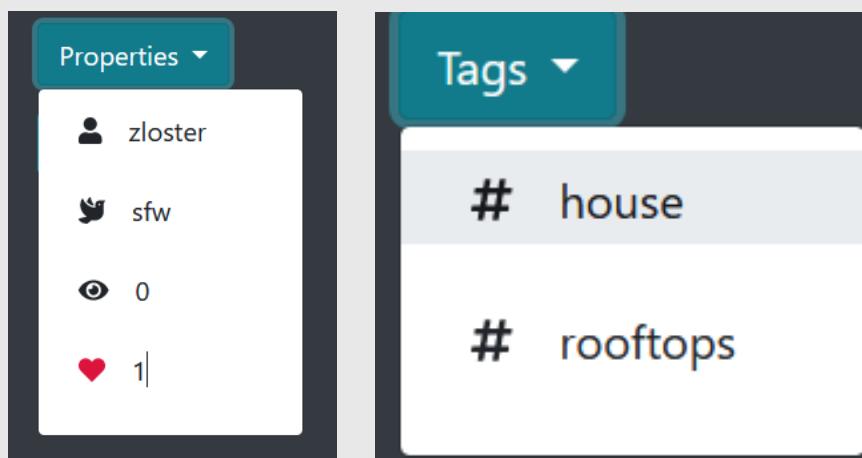
SITE FINAL



Page d'accueil



Page pour chaque fond d'écran



Site final

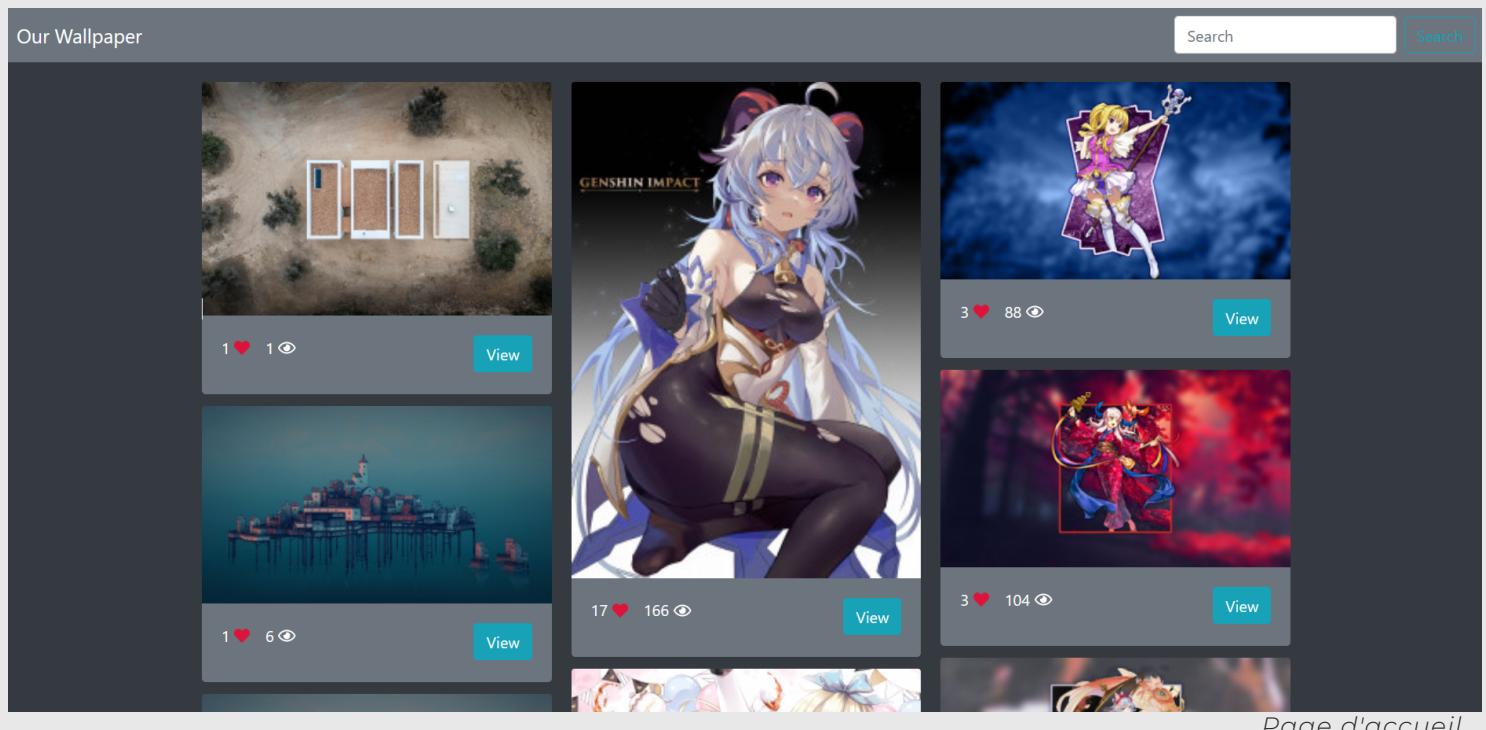
Le site est plutôt sobre pour se conformer aux designs actuels, nous retrouvons donc une page principale qui répertorie 24 fonds d'écrans par page (Limitation de l'API), sur chacun, nous retrouvons le nombre de like ainsi que le nombre de vues.

Sur cette page principale, nous avons intégré une barre de recherche.

Quand on clique sur le bouton "view" d'un des fonds d'écrans, il apparaît en plus grand sur une page différente avec des informations sur la gauche de celui-ci.

Quand on déroule le bouton "properties" on retrouve le nom du propriétaire, la catégorie de l'image, le nombre de vues et de like.

Quand on déroule le bouton "Tags", on retrouve les différents tags de l'image.



Page d'accueil

Utilisation de l'API

Tout d'abord nous avons décidé d'utiliser l'api de wallhaven, voici un petit lien si vous voulez voir leur documentation.

Cette API nous donne accès à la base de données du site wallhaven.cc et ainsi d'accéder à tout son contenu tel que les fonds d'écrans.

Pour pouvoir créer la base de notre application, nous avons procédé en deux étapes principales, la récupération et l'affichage d'un fond d'écran puis la récupération et l'affichage de plusieurs fond d'écran sous forme de grille.

On a donc commencé comme dit précédemment par l'affichage d'un wallpaper. Pour cela nous avons créé un composant nommé single-wallpaper et un modèle nommé wallpaper.

```
export interface Wallpaper {
  id: string;
  url: string;
  short_url: string;
  uploader: {
    username: string;
    group: string;
    /*avatar: {
      200px: string;
      128px: string;
      32px: string;
      20px: string;
    };*/
  };
  views: number;
  favorites: number;
  //source: string;
  purity: string;
}
```

Ici le modèle wallpaper est basé sur les informations reçues par l'API via une requête avec un ID.

Et dans le composant single-wallpaper nous faisons appel à une fonction getWallpaperById à laquelle on s'abonne pour pouvoir recevoir le contenu sous forme d'une réponse.

```
constructor(
  private wallpapersRepository: WallpaperRepository,
  private activatedRoute: ActivatedRoute
) {}

wallpaper?: Wallpaper = undefined;

ngOnInit(): void {
  if (this.activatedRoute.snapshot.params.id) {
    this.wallpapersRepository.getWallpaperById(String(this.activatedRoute.snapshot.params.id))
      .subscribe((reponse) => {this.wallpaper=reponse.data});
  }
}
```

Car pour accéder aux données telles que l'image, le créateur ou encore le nombre de vues, il faut appeler l'url suivante : <https://wallhaven.cc/api/v1/w/<ID here>> avec l'ID de l'image à la place du <ID here>.

```
getWallpaperById(id: string) {
  return this.httpClient.get<{data: Wallpaper}>(`/api/v1/w/${id}?apikey=gC0X9sAYo1rzgd87GJJXkvHa9GyU70Qi`);
}
```

La fonction va ainsi faire une requête avec l'id en paramètre pour nous renvoyer les données du fond d'écran. Ici on peut remarquer qu'il y a une apikey, de base elle n'est pas nécessaire mais car elle débloque seulement le contenu nsfw. Nous l'avons mise dans un premier temps pour essayer de résoudre un bug d'affichage mais au final on l'a gardée pour avoir un peu plus de contenu sur le site.

Utilisation de l'API

La deuxième étape a été de faire une page regroupant plusieurs fonds d'écrans, ici nous en afficherons 24 car ce sont les rendus de base de l'api. Pour pouvoir faire cette page nous avons décidé de refaire un composant et un modèle.

```
import { Wallpaper } from './Wallpaper';

export interface WallpapersList {
  results: Wallpaper[];
}
```

Tout d'abord nous avons fait le modèle WallpapersList qui informe l'application que le retour sera une liste de Wallpaper.

Ensuite nous avons fait le composant all-wallpapers qui appelle deux fonctions, la fonction principale est getAllWallpapers:

```
getAllWallpapers() {
  return this.httpClient.get<{data: Wallpaper[]; meta: Meta}>(`/api/v1/search`);
}
```

Dans cette fonction nous utilisons l'url fournie par l'api qui est de base une url de recherche mais que l'on a utilisé pour faire apparaître 24 fonds d'écran.
La seconde fonction nous permet de faire une vraie recherche toujours via une url mais cette fois-ci avec en paramètre le mot recherché.

```
getWallpapersBySearch(text: string) {
  return this.httpClient.get<{data: Wallpaper[]; meta: Meta}>(`/api/v1/search?q=${text}`);
}
```

Voici les fonctionnalités de base de notre application. Nous allons maintenant vous parler de notre mise en page mais aussi de l'intégration d'Angular dans notre HTML.

Mise en page HTML et utilisation de bootstrap

Pour designer notre site nous avons choisi de nous tourner vers bootstrap car nous avons plus d'expériences avec lui qu'avec Angular Material.

Tout d'abord nous avons, dans le html principal, mis en place un menu et appelé notre composant all-wallpaper.

```
<nav class="navbar fixed-top navbar-expand-lg navbar-dark bg-secondary" >
  <a class="navbar-brand" href="#">Our Wallpaper</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent"
    aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>

  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
      <!--li class="nav-item active">
        <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>
      </li>-->
    </ul>
    <div class="form-inline my-2 my-lg-0">
      <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search" [FormControl]="search" (keyup.enter)="onEnter()"
        >
      <a [routerLink]="'wallpapers-search/' + search.value" class="btn btn-outline-info my-2 my-sm-0" >Search</a>
    </div>
  </div>
</nav>

<div class="container">
  <router-outlet></router-outlet>
</div>
```

Mise en page HTML et utilisation de bootstrap

Vous pouvez aussi remarquer que nous avons utilisé du property-binding et que l'on a mis en place du routing.

```
<main id=main>

  <section>
    <div class="card-columns">
      <div class="card text-white bg-secondary border-0" *ngFor="let wallpaper of allWallpapers">
        <img [id]="wallpaper.id" [src]="wallpaper.thumbs.original" alt="" class="card-img-top h-100">
        <div class="card-body row">
          <div class="col-6">
            <p style="display: inline-block; margin-right: 1em;">{{wallpaper.favorites}} <i class="fas fa-heart" style="color: #crimson ;"></i></p>
            <p style="display: inline-block;">{{wallpaper.views}} <i class="far fa-eye"></i>
          </div>
          <div class="col-6 text-right">
            <a [routerLink]="'wallpaper/' + wallpaper.id" class="btn btn-info">View</a>
          </div>
        </div>
      </div>
    </div>
  </section>

</main>
```

En ce qui concerne l'HTML du all-wallpapers on est restés simple avec une grid de cards via bootstrap, vous pouvez encore remarquer l'utilisation de routing pour accéder à la page du fond d'écran qui nous intéresse et du property-binding pour récupérer les infos de vues et de likes. Pour ce qui est des icônes nous avons utilisé la librairie de FontAwesome.

Et enfin pour finir, nous avons single-wallpaper dans lequel nous utilisons un *ngIf pour vérifier s'il ya un fond d'écran en entrée.

```
<main id=main *ngIf="wallpaper" >
```

Ensuite, via bootstrap nous avons une grille avec une première colonne, composée de deux boutons dropdown. Le premier contient les propriétés principales du fond d'écran il les avec du property-binding. On récupère donc le nom du propriétaire, la pureté de l'image, le nombre de vues et de j'aime.

```
<div class="dropdown" style="margin-bottom: 1em;">
  <button class="btn btn-info dropdown-toggle" type="button" id="dropdownMenu" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
    Properties
  </button>
  <div class="dropdown-menu" aria-labelledby="dropdownMenu">
    <p class="dropdown-item"><i class="fas fa-user" style="margin-right: 1em;"></i>{{wallpaper.uploader.username}}</p>
    <p class="dropdown-item"><i [class]="'fas fa-' + purityToClass(wallpaper.purity)" style="margin-right: 1em;"></i>{{wallpaper.purity}}</p>
    <p class="dropdown-item"><i class="fas fa-eye" style="margin-right: 1em;"></i>{{wallpaper.views}}</p>
    <p class="dropdown-item"><i class="fas fa-heart" style="color: #crimson ; margin-right: 1em;"></i>{{wallpaper.favorites}}</p>
  </div>
</div>
```

Vous pouvez remarquer que pour la pureté, l'icône change en fonction de celle-ci via une petite fonction utilisant un switch.

```
purityToClass(purity: string): string{
  switch (purity) {
    case "sfw":
      return "dove";
    case "sketchy":
      return "exclamation-circle";
    case "nsfw":
      return "exclamation-triangle";
    default:
      return "dove";
  }
}
```

```
<div class="dropdown">
  <button class="btn btn-info dropdown-toggle" type="button" id="dropdownMenu" data-toggle="dropdown" aria-haspopup="true"
  aria-expanded="false">
    Tags
  </button>
  <div class="dropdown-menu" aria-labelledby="dropdownMenu">
    <p class="dropdown-item" *ngFor="let tag of wallpaper.tags"><i class="fas fa-hashtag" style=" margin-right: 1em;"></i>
      {{tag.name}}</p>
  </div>
</div>
```

Le deuxième dropdown nous donne tous les tags de l'image grâce à un *ngFor et à du property-binding.

Puis de l'autre côté de la grille, dans l'autre colonne, nous affichons le fond d'écran via son ID, toujours en utilisant du property-binding.

```
<div class="col-9">
  <div>
    <img [id]="wallpaper.id" [src]="wallpaper.thumbs.original" width="100%" height="100%" alt="">
  </div>
</div>
```

Vous pouvez aussi remarquer que nous avons utilisé du property-binding et que l'on a mis en place du routing.

Les difficultés rencontrées

Pour ce projet nous avons rencontré pas mal de difficultés car ce n'était que la première fois qu'il codait en Angular sans avoir d'exemples ni de fil directeur. Une des principales difficultés était donc due à la "découverte" du framework et de ses fonctionnalités telles que les Observables et les notions de *ngIf / *ngFor ou encore de routing. Nous avons aussi rencontré des erreurs de syntaxe car nous n'avions pas encore assez d'expérience.

Nous pensons que la plus grande difficulté a été de faire le routing mais surtout de faire le système de recherche car nous avons rencontré un bon nombre de problèmes avec l'API. Le fait étant que nous avons dû modifier un bout de proxy.conf.json pour pouvoir lancer l'application sur une adresse locale.

L'incorporation de Bootstrap et de FontAwesome a aussi été compliquée car nous ne l'avions pas vu et que les docs étaient pas toutes fonctionnelles.

Nous aurions aimé avoir un système de pagination pour afficher plus de fond d'écrans mais nous n'avons pas réussi à le déployer.

Nos pistes d'améliorations

Nous aurions aimé avoir un système de recherche plus poussé comme avec un color picker ou par tags. Mais aussi un système de filtre reprenant ces données et la pureté de l'image.

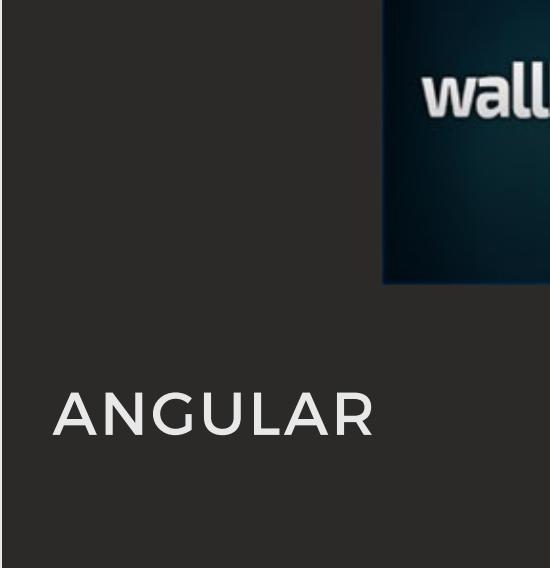
Il serait aussi intéressant de mettre en place un système de like et de galerie pour retrouver les fonds d'écran que l'on a aimé.

Et enfin, pourquoi pas rajouter une option de téléchargement du fond d'écran.

Pour ce qui est du design, on pourrait améliorer la page d'affichage d'un fond d'écran pour avoir plus d'options mais surtout une image moins floue.



wallhaven



ANGULAR

PROJET OUR WALLPAPER