**Laboratoire #11**

Gestion d’images avec Web API

**Projets de départ**

* Un projet ASP.NET Core et un projet Angular vous sont fournis. N’exécutez pas tout de suite le serveur, on doit d’abord créer une classe, faire une migration et update la base de données.
* ImageSharp est déjà installé sur le serveur.

**Étape 1 - Envoyer une image au serveur**

* L’objectif sera de permettre à l’utilisateur de sélectionner une image, l’envoyer au serveur, puis l’ajouter dans la BD et dans le File System du serveur.
* Côté client :
  + Modifiez l’input file et créez un ViewChild qui permettra d’accéder au fichier joint par l’utilisateur.
  + Produisez un FormData qui contiendra l’image et envoyez le formData au serveur dans une requête POST. (Il se peut que vous deviez modifier l’URL de la requête plus tard)
    - Attention : oui la requête sera dans le PictureService, mais construire le FormData il faut le faire dans le composant App. Vous pourrez envoyer le FormData sous forme de paramètre au service.
* Côté serveur :
  + Créez un Model pour les images. Il n’aura que 3 propriétés. Ce Model n’a absolument pas besoin d’exister côté client.
    - Une fois le Model finalisé, créez une migration et mettez à jour la base de données.
  + Dans le contrôleur relativement vide fourni, complétez l’action POST qui recevra le fichier et l’ajoutera dans la BD et le File System.
    - On veut une version originale de l’image et une version de taille réduite avec une hauteur de 200 pixels.
    - Des dossiers existent déjà pour les images, vérifiez le répertoire du serveur.
    - Attention ! Dans les notes de cours, on modifie un Birb existant. Dans ce cas-ci, **on crée** (Add) un nouvel objet dans la base de données donc le code sera différent un peu.

**Étape 2 - Le petit oiseau va sortir, ne le laissez pas s’échapper**

* L’objectif pour cette étape sera d’envoyer la liste des ids de toutes les images au client Angular. On n’a pas besoin d’envoyer les FileName et les MimeType, juste les ids.
* Côté serveur :
  + Complétez une action Get qui retourne simplement la liste de tous les ids des images dans la base de données. (Donc une List<int>)
  + Votre return pourrait ressembler à ceci …

return Ok(pictures.Select(p => p.Id))

* Côté client :
  + Assurez-vous simplement de recevoir la liste d’ids dans la fonction ngOnInit et dans la fonction updateDisplay. Ça ne sera pas suffisant pour pouvoir afficher les images, mais on y est presque.

**Étape 3 - Afficher les images**

* Nous allons permettre au serveur de retourner des fichiers images pour pouvoir les afficher dans le HTML.
* Côté serveur :
  + En vous inspirant des notes de cours, retournez le fichier associé à l’id demandé.
  + N’oubliez pas de permettre de choisir entre les tailles « large » et « small ».
* Côté client :
  + Dans le HTML, modifiez l’attribut [src] pour demander l’image au serveur en format small, pour chaque id d’image existante.
  + De plus, il faut que cliquer sur une image l’affichage en format large dans un autre onglet. (Utilisez un élément <a> et mettez la bonne requête dans son attribut href)

**Étape 4 - Elle n’est pas bonne, on va en prendre une autre**

* Pour cette étape, nous allons rendre fonctionnels les petits « X » côté Angular pour supprimer les images.
* Côté Angular :
  + Il faudra lancer une requête qui fournit simplement l’id de l’image à supprimer.
* Côté serveur :
  + Inspirez-vous de l’action Delete dans les notes de cours pour trouver l’image dans le DbContext et la supprimer. (N’oubliez pas de supprimer ses deux copies sur le disque également)
  + Simplement retourner Ok() est convenable.

**Étape 5 - Un c’est bien, mais cinquante-douze c’est mieux**

* Pour cette étape, on souhaite permettre à l’utilisateur de poster plusieurs images simultanément plutôt qu’une seule à la fois. Il y aura plusieurs changements à faire pour que ce soit possible.
* Côté Angular :
  + Dans l’<input> de type file, ajoutez l’attribut HTML **multiple** (il n’a pas de valeur), qui permet de sélectionner plusieurs fichiers à la fois.
  + Lorsque vous meublerez le formData, vous allez devoir .append() un à plusieurs fichiers maintenant.
    - Comme chaque clé doit être unique, visez quelque chose comme "image0", "image1", "image2", etc.
    - Quand il y avait un seul fichier, on allait le chercher avec …**nativeElement.Files[0]**. Pour les suivants, il suffira de remplacer le 0 par 1, puis 2, puis 3, etc.
    - Vous devrez utiliser une boucle **while** car on ne sait pas d’avance combien de fichiers seront fournis. Il faudra s’arrêter dès qu’on trouve « **null** » en faisait …**nativeElement.Files[i]**.
* Côté serveur :
  + Un peu la même idée que côté Angular : comme on va recevoir un à plusieurs fichiers, il faudra mettre une boucle **while** dans le code et faire **GetFile( … )** jusqu’à ce qu’on tombe sur « **null** ».
* Ensuite, assurez-vous que ça fonctionne !