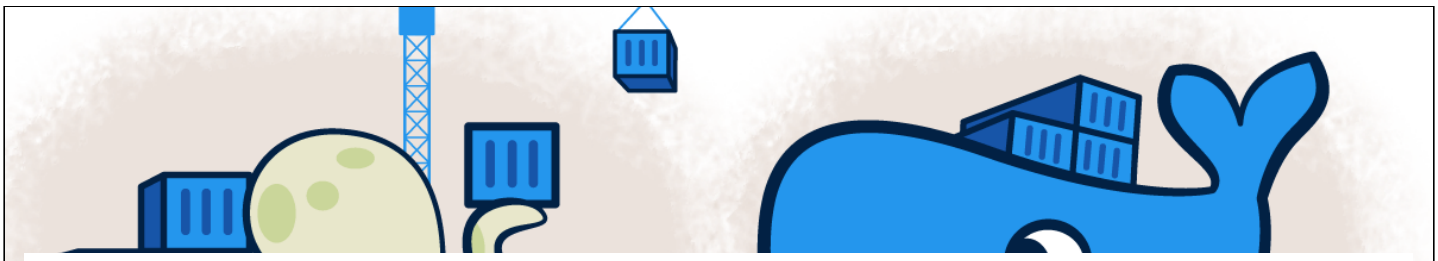


Créer une App pour Azure (partie 2)

Cloud, CSS, Docker, HTML, PHP

Langage Compétences : 5

La transition digitale a poussé les éditeurs de logiciels et les entreprises à migrer leurs applications sur le web. Les services du cloud Azure ont offert la plateforme idéale, les services adéquats, ainsi que les outils, pour réaliser un développement vers le tout numérique en gardant le contrôle total sur les données. Cette publication propose la création d'une application simple en découvrant certains de ces services et outils.

Publié mercredi 22 juin 2022, 17h33
Modifié lundi 26 août 2024, 10h04

By Olivier Paudex

Mise en place de l'environnement de travail

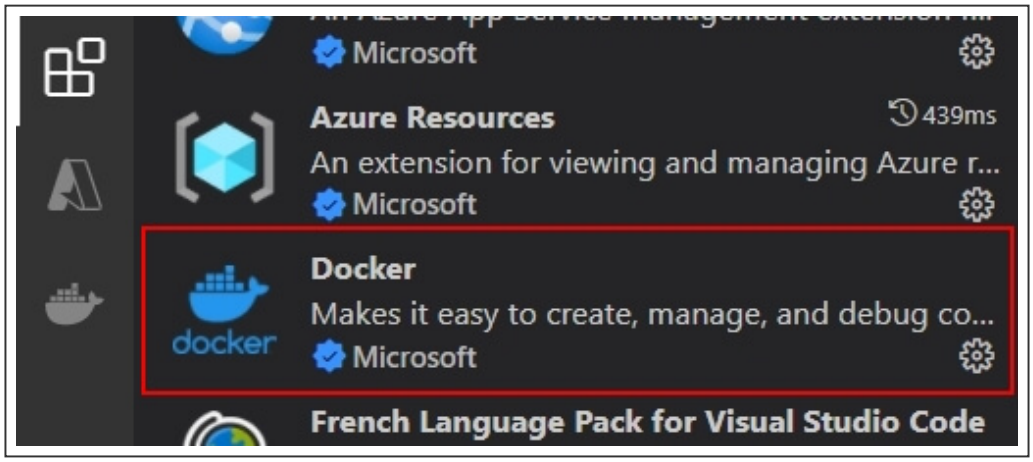
Pour travailler avez le cloud Azure et les containers, il faut installer certains outils dont quelques uns sont fortement recommandés.

> Cette publication est composée de plusieurs partie. Ceci étant la 2ème partie.

L'outil de travail est ici un PC de type Window 11. Il est tout à fait possible de le faire depuis un Mac.

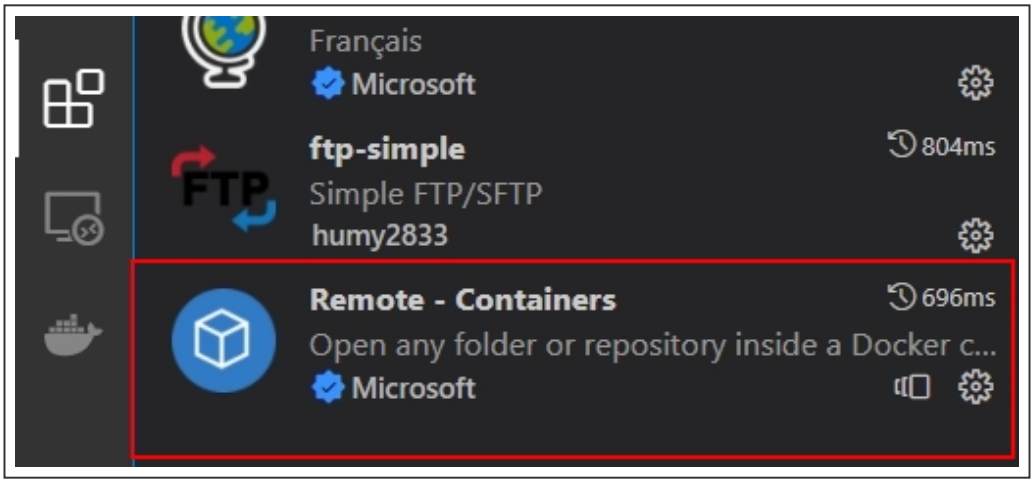
Si vous ne possédez pas de compte Azure, il est temps de le faire. L'ouverture d'un compte et l'utilisation des différentes services est gratuit les 30 premiers jours.

- Installation de Visual Studio Code : https://code.visualstudio.com/download.
- Installation de Docker sur Windows : https://docs.docker.com/desktop/windows/install/.
- Installation du plugin Docker pour VSCode.



Le plugin Docker pour VSCode

- Installation du plugin **"Remote – Containers"** pour VSCode pour éditer directement le code du container.



Le plugin Remote – Containers pour VSCode

- Installer PowerShell : https://docs.microsoft.com/en-us/powershell/.

- Installer Azure CLI : https://docs.microsoft.com/en-us/cli/azure/.
- Depuis l'interface du VSCode, ouvrez un terminal Powershell et effectuez la mise à jour de Azure CLI si nécessaire

```
az --version
```

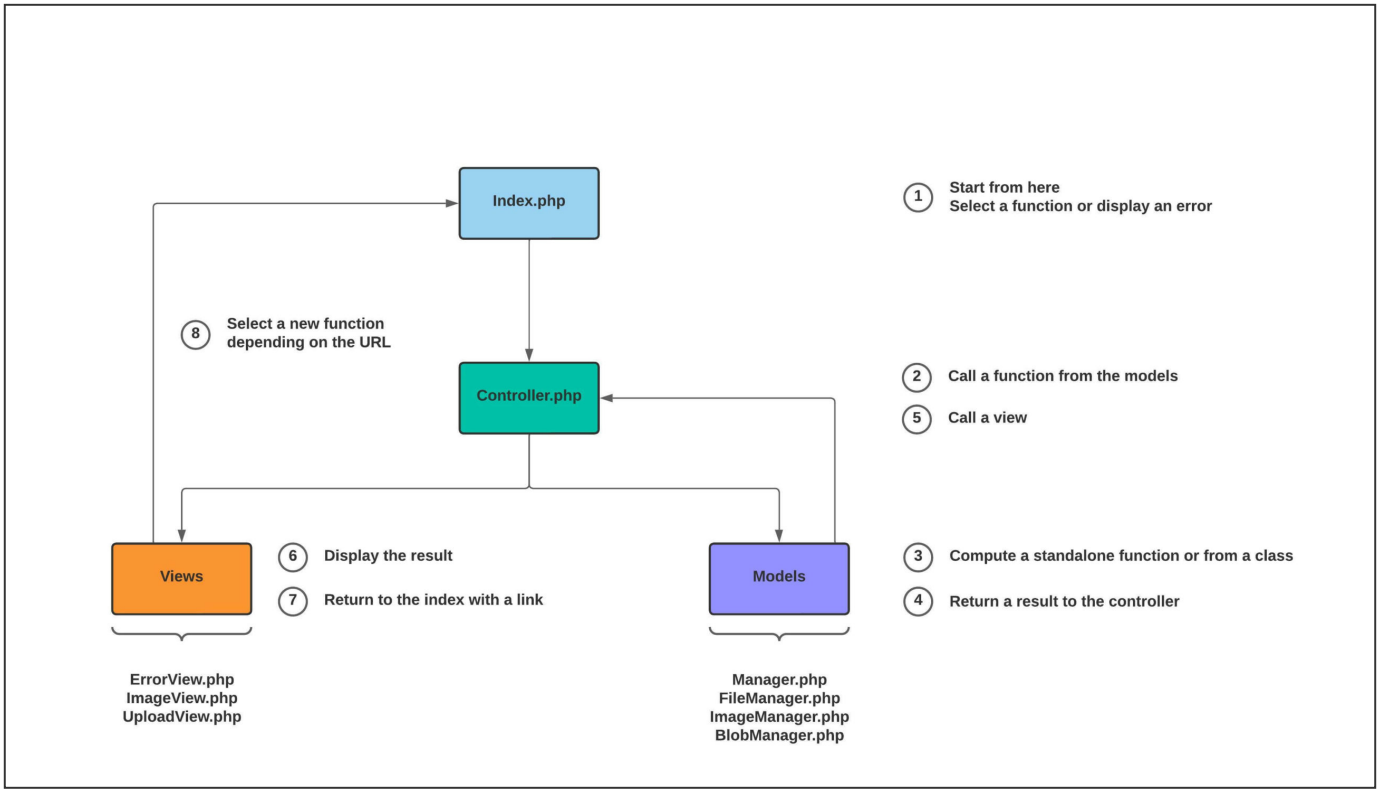
puis

```
az upgrade
```

Avant-projet

La méthode MVC

Avant de démarrer un projet, il est toujours intéressant de mettre ses idées à plat, de dessiner les écrans, de choisir un langage de programmation. Pour ce projet, le **PHP**, le **HTML** et le **CSS** ont été choisi. Pour le code, la méthodologie **MVC**, axée sur les modèles et les vues, sera utilisée pour garder une vision la plus simpliste possible. Ci-dessous, un plan du projet au format MVC.

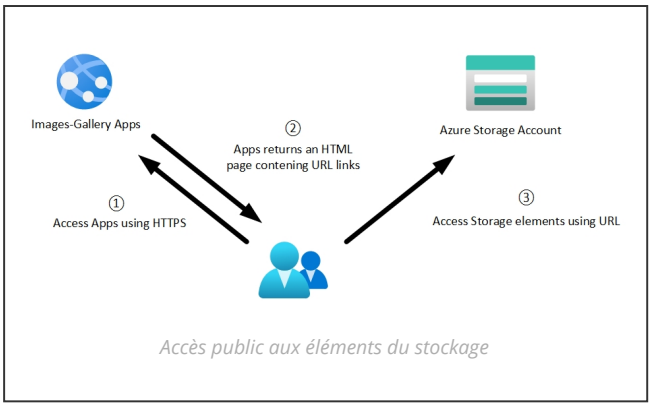


Le modèle MVC de l'application

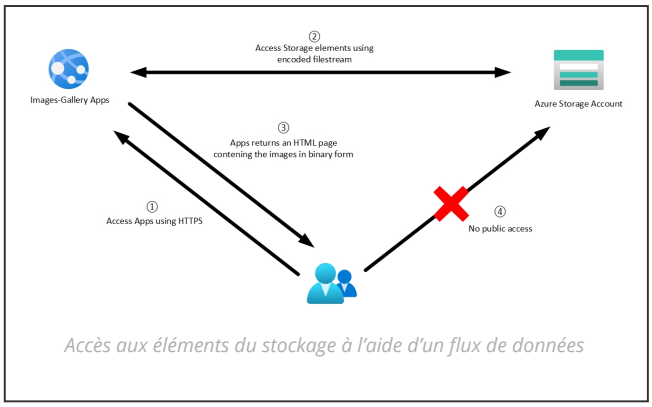
Accès au stockage de données

Voici une explication sur l'accès aux données du stockage de manière sécurisée et non-sécurisée. L'application va bien entendu utiliser la seconde manière.

Sur le cloud Azure, pour accéder au stockage de données, on utilise l'URL d'un élément pour accéder à celui-ci. Sauf que dans le cas d'une application cloud, l'accès public au service de stockage doit être fermé. Ce qui ne permet pas d'utiliser l'URL des éléments pour y accéder. L'application **Images Gallery** va utiliser une lecture binaire des images stockées pour y accéder, et non pas son URL.



Accès public aux éléments du stockage



Accès aux éléments du stockage à l'aide d'un flux de données

- Dans l'image de gauche, l'accès aux éléments du stockage se fait en utilisant l'URL.
- Dans l'image de droite, l'accès aux éléments du stockage se fait en utilisant un flux de données interne depuis l'application. Ce qui permet de fermer l'accès public au compte de stockage du cloud.

Ci-dessous, la fonction qui permet de récupérer une image du compte de stockage. Elle se trouve dans le fichier **BlobManager.php**.

> Téléchargez le code de cette application en cliquant sur le lien "Pièce jointe" tout en haut de la publication.

```
// Get blob content from Azure storage
public function getBlobContent($containerName) {
  // Storage connexion
  $blobClient = $this->storageConnect();

  // Get blob file and mime property
  $blob = $blobClient->getBlob($containerName, $this->Name);
  $properties = $blobClient->getBlobProperties($containerName, $this->Name);
  $mimeType = $properties->getProperties()->getContentType();
  // Get base64 encoded blob file stream
  $stream = stream_get_contents($blob->getContentStream());
  $fileEncode = base64_encode($stream);
  // Return the source stream
  $src = 'data: '. $mimeType . ';base64,' . $fileEncode;
  return $src;}
```

Conclusion

Voici quelques bonnes idées de départ concernant les logiciels à utilisés. Une réflexion sur la méthodologie à aussi été évoquée, méthode qu'il est très conseillé d'adopter quand on veut créer une application web.

Enfin, une explication sur le concept utilisé lors de l'accès à un compte de stockage est venu s'ajouter. Il était très important d'en souligner les grandes lignes, car il y a peu ou pas de documentation concernant un accès sécurisé non public au stockage Azure.

Le prochain chapitre va parler du projet, du code de l'application, de l'infrastructure en containers avec Docker et des librairies utilisées.