

PROJET 4DESA

CLÉMENT HONORÉ
EWEN BOSQUET
WALID FADI

Sommaire

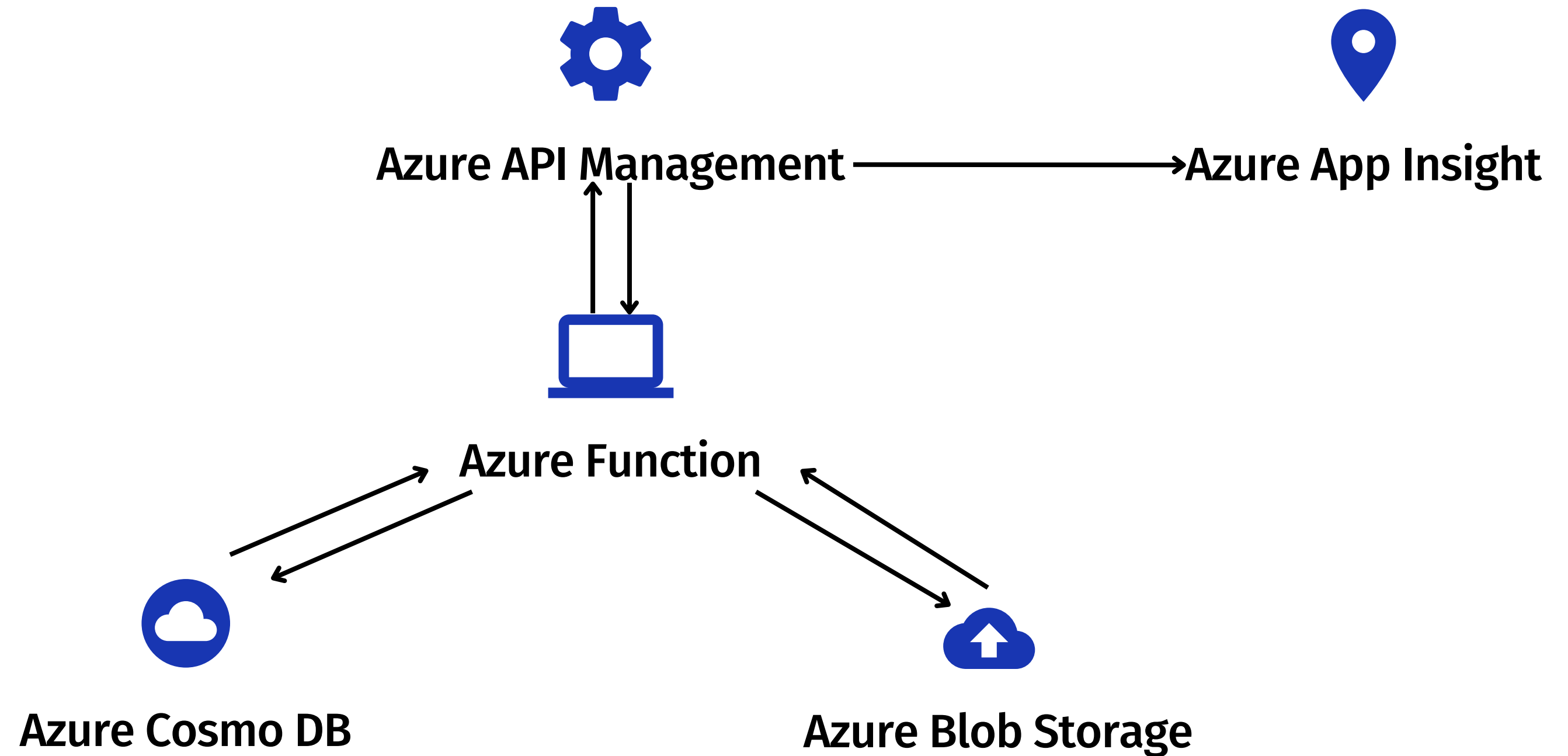
1. Présentation du projet
2. Environnement de développement
3. Les routes
4. Fonctions Azure



Presentation

Ce projet à pour but de mettre en place une plateforme back end headless pour la gestion de réseaux sociaux par le biais d'une API.

Architecture de la plateforme



Azure Function



POST - Register



POST - Login



GET/POST/DELETE - Media

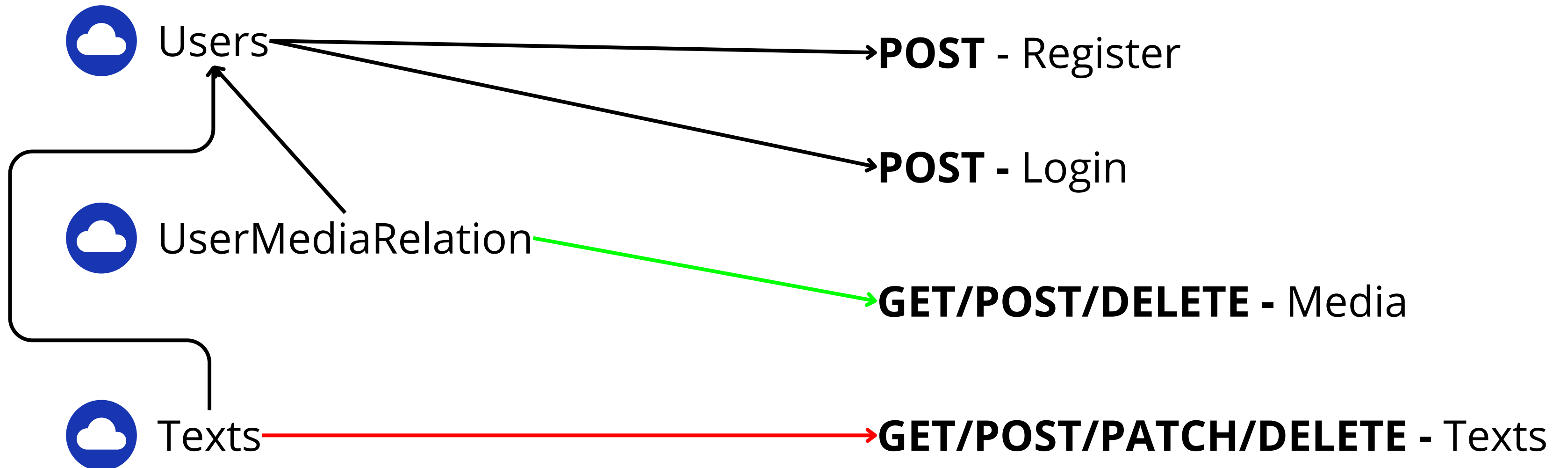


GET/POST/PATCH/DELETE - Texts

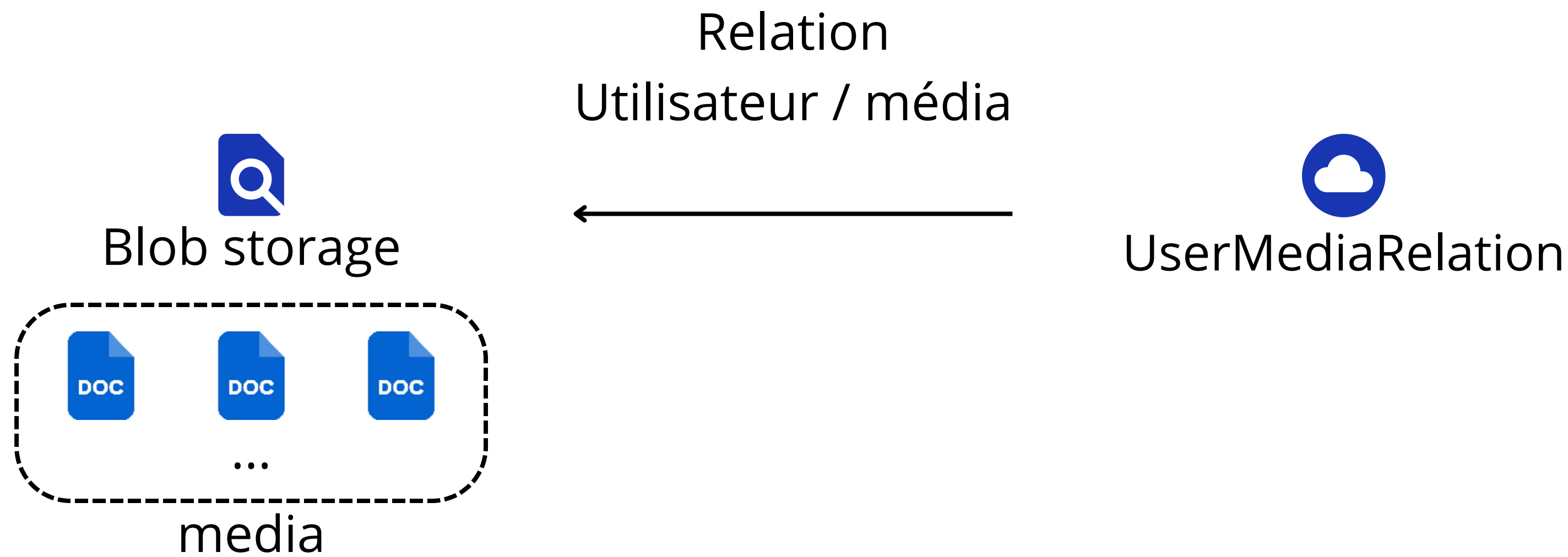
Azure Cosmo DB

Conteneur

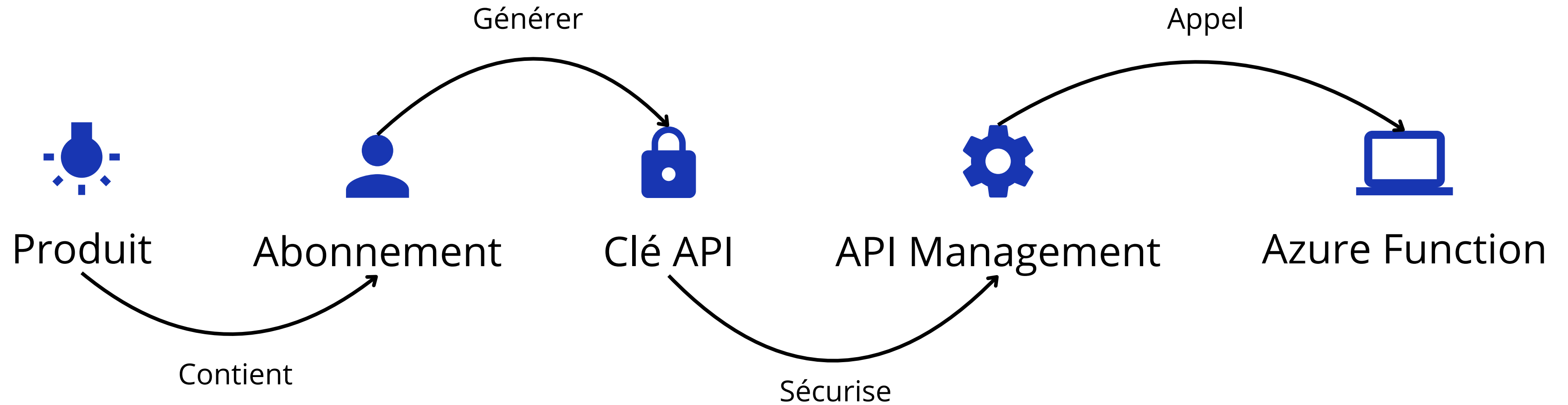
Point d'API



Azure Blob Storage



Azure API Management



Azure API Management

Endpoints

All APIs		
desaman-api	...	+ Add operation
		All operations
GET	GetUsers	...
POST	Login	...
GET	Media	...
POST	Media	...
DEL	Media	...
POST	Register	...
GET	Textes	...
DEL	Textes	...
PATCH	Textes	...
POST	Textes	...

Gestion des paramètres

Frontend

GET /media/{filename}

Frontend

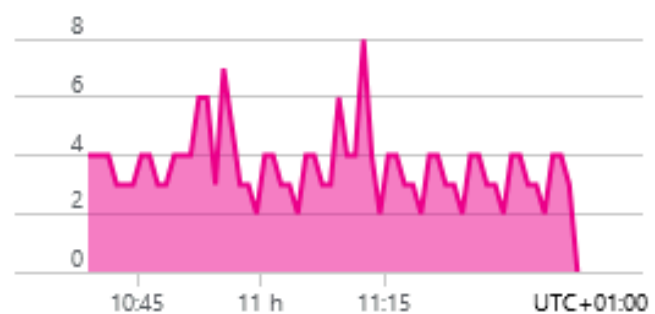
POST /media

Azure App Insight

Afficher les données du/de la dernier(ère) :

30 minutes **1 heure** 6 heures 12 heures 1 jour 3 jours 7 jours 30 jours

Demandes en échec



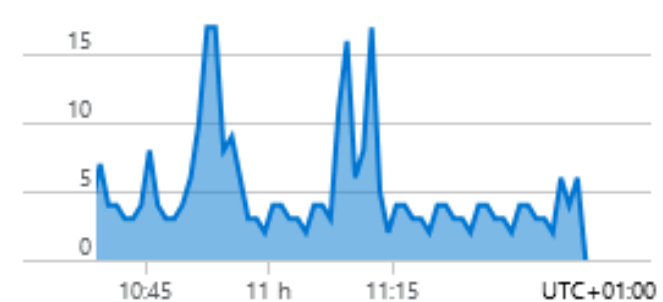
Failed requests (Nombre), desaman-... | 215

Temps de réponse du serveur



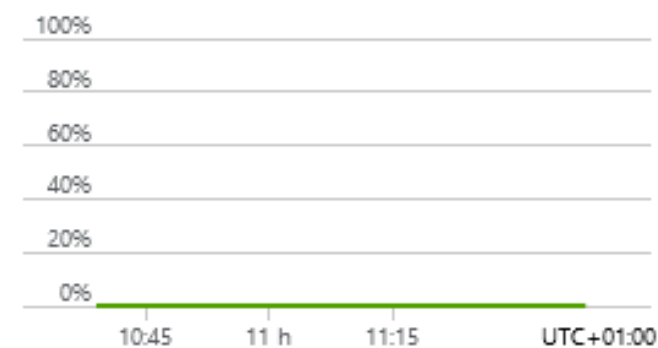
Server response time (Moy), des... | 670,28ms

Requêtes serveur



Server requests (Nombre), desaman-... | 300

Disponibilité



Availability (Moy), desaman-api | 0%

Environnement de Développement

L'environnement utilisé lors du développement des fonctions de notre API

Outil principaux



NodeJS

Nous l'avons choisie car maîtrisée par l'équipe et plus rapide pour de la conception web



VS Code

Il s'agit de notre IDE nous permettant de développer les fonctions



Azure Function

Est une extension pour VS Code permettant la gestion et le déploiement des fonctions



Postman

Nous a permis de tester les différentes routes de nos API en exécutant des requêtes HTTP

Librairie



@azure/cosmos

Permet de faire le lien entre notre base de données et notre code



@azure/storage-blob

Permet de faire le lien entre notre blob de stockage et notre code



@azure/functions

Permet de tester localement et déployé sur Azure

bcrypt

bcryptjs

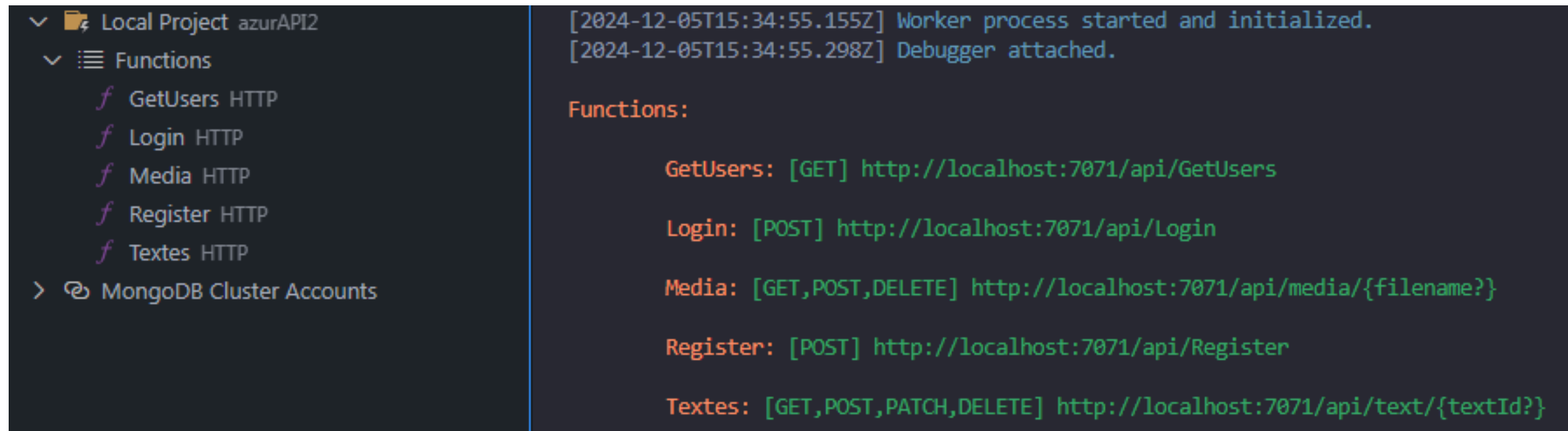
Permet de hacher des chaînes de caractères ici nos mots de passe



jsonwebtoken

Permet de générer un token d'accès unique et qui expire pour sécuriser nos routes API

Exécution des fonctions en local



```

v Local Project azurAPI2
  v Functions
    f GetUsers HTTP
    f Login HTTP
    f Media HTTP
    f Register HTTP
    f Textes HTTP
  > MongoDB Cluster Accounts

[2024-12-05T15:34:55.155Z] Worker process started and initialized.
[2024-12-05T15:34:55.298Z] Debugger attached.

Functions:

  GetUsers: [GET] http://localhost:7071/api/GetUsers

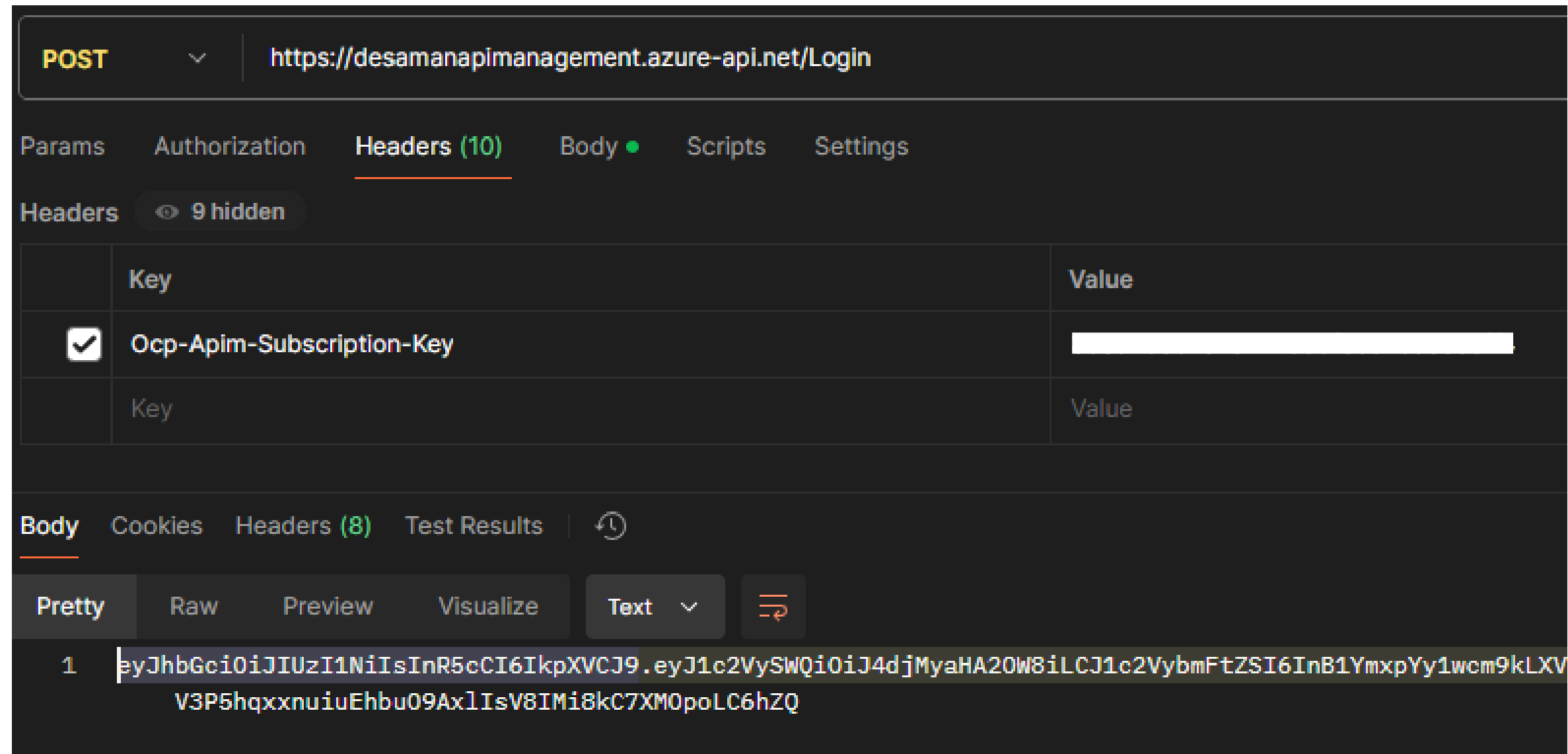
  Login: [POST] http://localhost:7071/api/Login

  Media: [GET,POST,DELETE] http://localhost:7071/api/media/{filename?}

  Register: [POST] http://localhost:7071/api/Register

  Textes: [GET,POST,PATCH,DELETE] http://localhost:7071/api/text/{textId?}
```

Postman



Les routes

Plusieurs routes sont accessibles pour effectuer différentes actions sur les données. Elles représentent les fonctionnalités du projet.

Authentication



POST - /Register

Enregistre un nouvel utilisateur dans la base de données



POST - /Login

Authentifie un utilisateur enregistré.
Retourne un jeton d'authentification JWT

Médias



POST - /media

Enregistre un nouveau média (photo, vidéo, audio, etc...) et le lie à l'utilisateur authentifié
Retourne le nom du fichier



GET - /media/{filename}

Récupère le média ayant le nom donné en paramètres
Si le média est associé à un compte privé, il n'est pas retourné
Retourne le fichier



DELETE - /media/{filename}

Supprime le média ayant le nom donné en paramètres
Seul l'utilisateur ayant enregistré le média peut le supprimer
Retourne un code HTTP 204

Textes



POST - /text

Enregistre un nouveau texte et le lie à l'utilisateur authentifié
Retourne l'identifiant du texte



GET - /text/{textId}

Récupère le texte ayant pour identifiant celui donné en paramètres
Si le média est associé à un compte privé, il n'est pas retourné
Retourne le contenu du texte



PATCH - /text/{textId}

Modifie le texte ayant pour identifiant celui donné en paramètres
Seul l'utilisateur ayant enregistré le média peut le modifier
Retourne le contenu du nouveau texte



DELETE - /text/{textId}

Supprime le texte ayant pour identifiant celui donné en paramètres
Seul l'utilisateur ayant enregistré le média peut le supprimer
Retourne un code HTTP 204

Fonctions Azure

Ici nous explorerons rapidement le code et les concepts utilisés dans les fonctions de l'API

Déclarer une fonction Azure avec Node

```
app.http('Login', {  
  methods: ['POST'],  
  authLevel: 'anonymous',  
  handler: async (request, context) => {  
    // Function code...  
  })  
});
```



/Login

```
app.http('Media', {  
  methods: ['GET', 'POST', 'DELETE'],  
  authLevel: 'anonymous',  
  route: 'media/{filename?}',  
  handler: async (request, context) => {  
    // Function code...  
  })  
});
```



/media

/media/mon-fichier

Communication avec CosmosDB

```
const client = new CosmosClient({
  endpoint: process.env.COSMOS_DB_URI,
  key: process.env.COSMOS_DB_KEY
});

const database = client.database(process.env.COSMOS_DB_DATABASE);

// Récupérer un conteneur
const getContainer = (containerName) => {
  return database.container(containerName);
};

module.exports = { getContainer };
```

Création du client CosmosDB

Connexion à une base de données
via une clé de connexion

Connexion à un conteneur spécifique

Exportation de la fonction

Login

```
const client = BlobServiceClient.fromConnectionString(process.env.STORAGE_ACCOUNT);

const getMediaBlobContainer = () => {
  return client.getContainerClient('media')
}

module.exports = { getMediaBlobContainer }
```

Création du client blob

Connexion à un conteneur

Exportation de la fonction

CONCLUSION

- Découverte des fonctionnalités Azure
- Implémentation d'une solution complexe
 - Functions app
 - CosmosDB
 - Blob storage
 - API Management
 - App insight
- Exposition de routes HTTP sécurisés



Tech et Cie.

Merci !