TP Cloud - Fileshare

Samuel Léobon - Eliot Louys

Choix techniques et justifications

Frontend

L'application front-end est une UI en React+Vite.

La raison de ce choix est notre expertise pré-existante avec la technologie React, ce qui permettait un développement et un debugging plus rapide et plus aisé.

Backend

Le back-end est une API Node.js programmée avec la librairie Express.
De même, Express a été choisie en raison de notre connaissance préexistante de développement d'API avec cette librairie.
De plus, Express est très approprié pour créer des APIs relativement simples sans avoir à écrire de grandes quantités de code, ce qui correspond exactement à nos besoins relatifs à ce TP.

Architecture déployée sur Azure

L'architecture déployée sur Azure se distingue en quatre parties principales :

- le App Service frontend
- le App Service backend
- le SQL Server accompagné de sa base de données SQL
- le Blob Storage

Le App Service frontend déploie le build de l'interface front-end récupéré depuis Github. Puisque le build est une page HTML, elle doit être exposée à l'aide d'une commande spécifique : `pm2 serve /home/site/wwwroot/dist --no-daemon --spa`

Le App Service backend déploie l'API Node.js en exécutant automatiquement le script "start" du package.json (qui est en l'occurrence `node server.js`). Les informations de connexion au SQL Server et au Blob Storage sont stockées dans les Variables d'Environnement.

Le SQL Server expose une base de données MySQL structurée en deux tables : une tables Users (pour les utilisateurs) et une table Items (pour lister les dossiers et fichiers, leur structure et les utilisateurs liés).

Le Blob Storage stocke les fichiers eux-mêmes. Pour assurer leur unicité, les fichiers sont stockés avec leur UUID ajoutée dans leurs noms de fichier.

Difficultés rencontrées et solutions apportées

LES VARIABLES D'ENVIRONNEMENT

Une des principales difficultés à été sur les variables d'environnement dans le front. En effet, la connexion entre backend et frontend se fait dans le code à travers les variables d'environnement globales. Et si ce système fonctionne très bien en développement à travers un .env, lors du déploiement sur azure, le code statique du front ne récupère pas les variables d'environnements azure. De la plusieurs options s'offrent à nous :

1. Hardcoder les chaînes de connexions
2. Les mettre en variable d'environnement github pour y avoir accès au build-time dans la CI
3. Injecter les variables d'environnement azure au démarrage de l'application

Nous avons opté pour l'option 3, car elle centralise les paramètres de run et de connexion back/front dans azure. Le but est aussi d'éviter les injections d'un back tiers à travers une action github malveillante. La première option expose notre url back dans le front.
Nous avons aussi eu une légère difficulté sur les variables d'environnement au démarrage du back.

Estimation des coûts Azure

Ressource Azure	**Coûts estimés**
App Service	0€ si plan gratuit, ~45€ par mois en plan de base
SQL Server	~5€/mois, mais tarification complexe et soumis à trop de facteurs pour une estimation plus précise.
Blob Storage	~0.01€ par Go par mois
App Configuration	0.101€ par store par jour