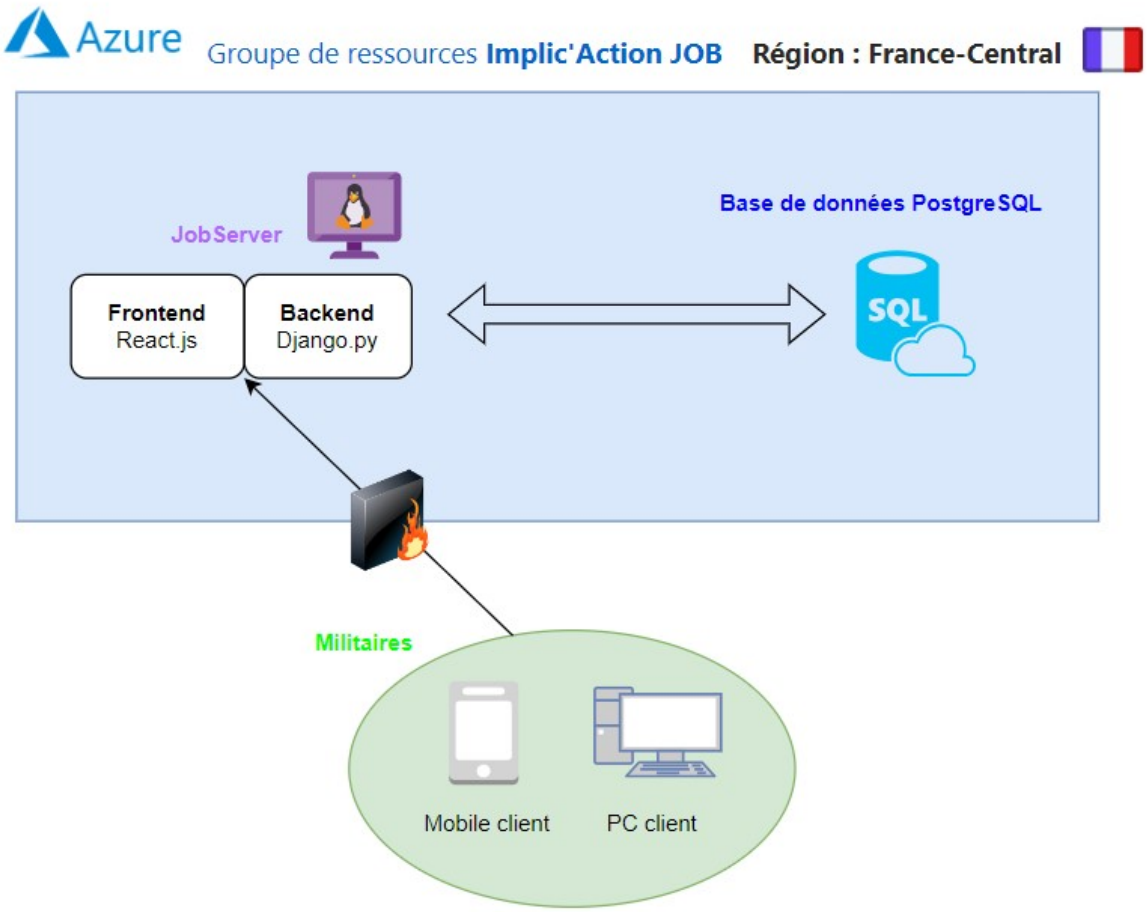
- Possibilité de partagé la conversation JobBot à un utilisateur dans une session de chat

III- Architecture de l'application

A- Infrastructure globale

L'architecture de notre application web est composée de plusieurs composants interconnectés, chacun jouant un rôle clé dans le fonctionnement global du système.



Cette architecture garantit une séparation claire des responsabilités entre le frontend, le backend et la base de données, tout en assurant une communication fluide et sécurisée entre les composants. Les utilisateurs finaux bénéficient ainsi d'une application réactive, sécurisée et fiable, accessible depuis différents types de dispositifs.

Nous avons choisi d'héberger nos serveurs en région France-Central pour que nos données restent en France.

Nous avons intégré un parefeu entre le JobServer et les endpoints clients. Cela va permettre de sécuriser les échanges de données en filtrant le trafic entrant et sortant. Le parefeu analysera les paquets de données pour détecter et bloquer les tentatives

d'accès non autorisées, les attaques potentielles et les logiciels malveillants. Il garantira également que seules les connexions légitimes et sécurisées soient établies entre les clients et le JobServer, renforçant ainsi la protection contre les cyberattaques et les violations de données.

De plus, le parefeu permettra de définir des règles spécifiques pour contrôler l'accès aux différentes ressources du serveur, en fonction des adresses IP, des ports et des protocoles utilisés. Cela contribuera à une gestion plus fine et sécurisée des permissions d'accès, assurant que les utilisateurs et les applications disposent uniquement des accès nécessaires à leurs fonctions.

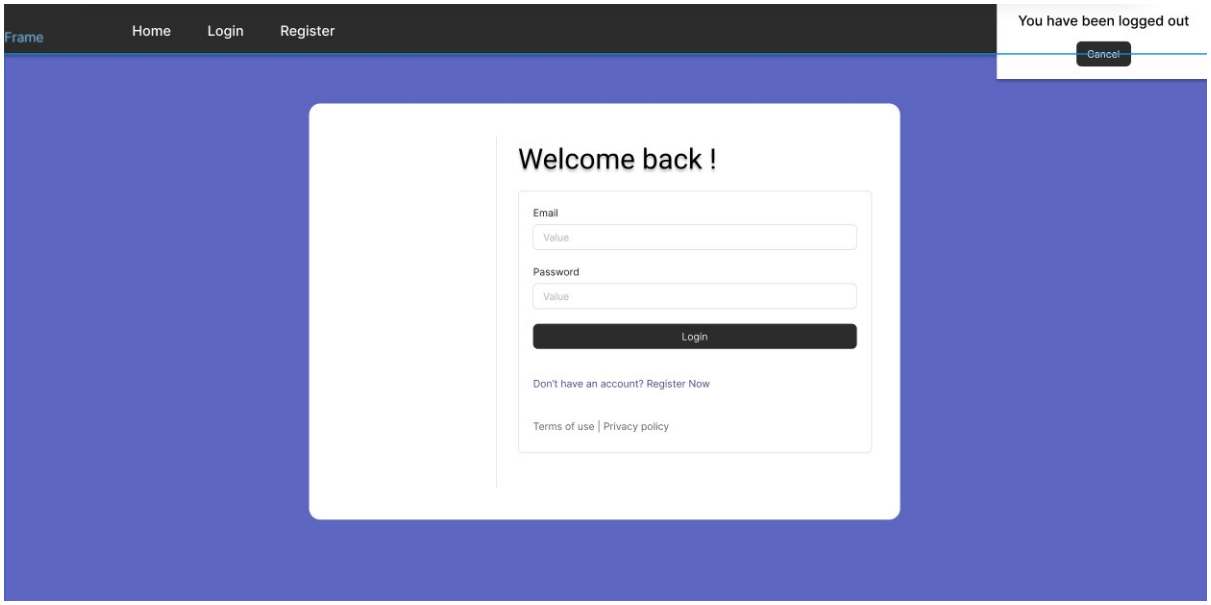
Enfin, le parefeu sera configuré pour générer des logs détaillés des activités réseau, fournissant des informations précieuses pour la surveillance et l'analyse de la sécurité. Ces logs aideront à détecter les comportements anormaux et à réagir rapidement en cas d'incident de sécurité.

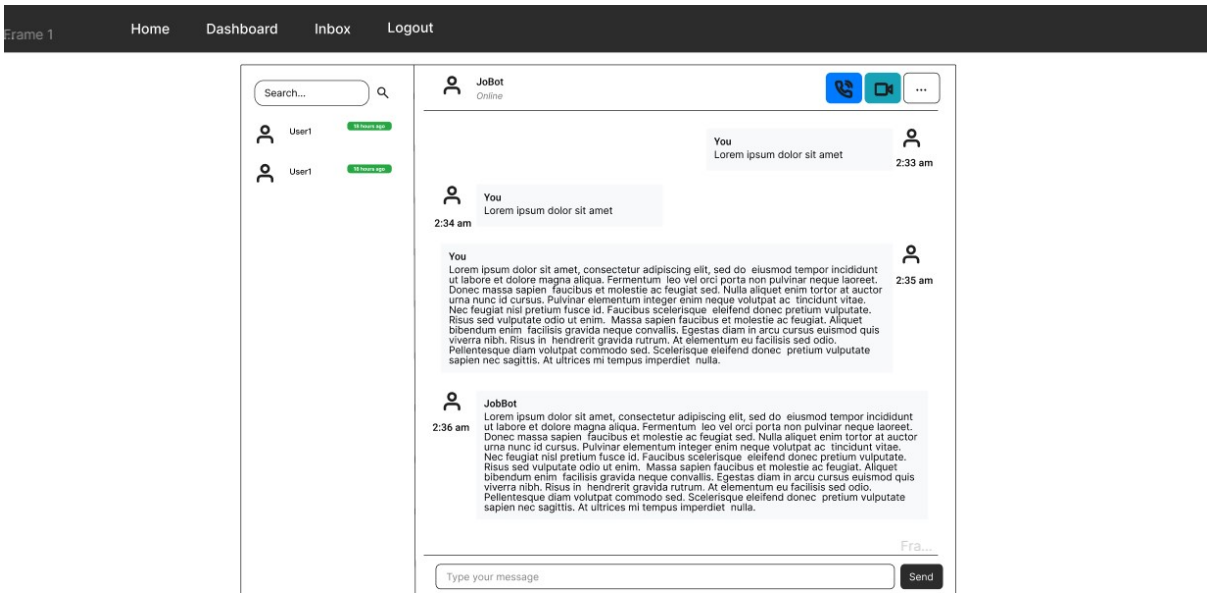
Voici les principales règles :

Nom de la Règle	Priorité	Source	Port	Protocole	Action	Description
Allow-Admin-Port-8000	100	IP Admins (x.x.x.x)	8000	TCP	Allow	Autorise l'accès au port 8000 uniquement pour les administrateurs.
Allow-Admin-SSH	200	IP Admins (x.x.x.x)	22	TCP	Allow	Autorise l'accès SSH depuis l'adresse IP des administrateurs.
Allow-App-Port-3000	300	Anywhere (0.0.0.0/0)	3000	TCP	Allow	Autorise l'accès au port 3000 depuis n'importe où.

B- Interface utilisateur

Pour assurer une expérience utilisateur fluide et intuitive, j'ai utilisé l'outil Figma pour concevoir la maquette de l'interface utilisateur de notre application web. Figma offre une plateforme collaborative idéale pour créer et itérer rapidement sur des designs interactifs, ce qui nous a permis de visualiser et de structurer efficacement chaque élément de l'interface avant sa mise en œuvre.



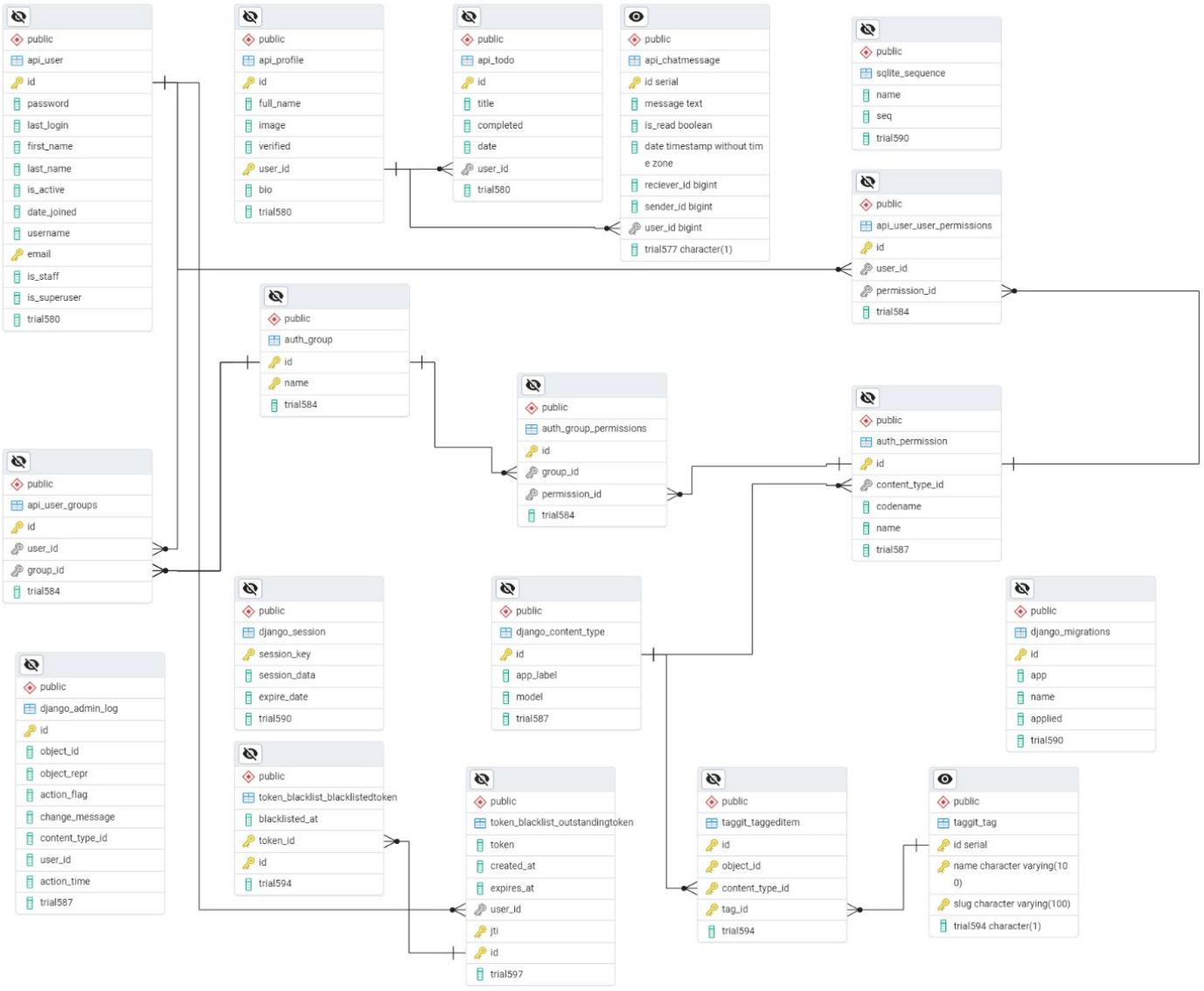


C- Base de données

Postgre SQL est un service de base de données relationnelle basé sur le cloud qui offre de nombreux avantages, notamment :

- ✓ **Performances élevées**
- ✓ **Haute disponibilité Sécurité avancée**
- ✓ **Évolutivité**

Pour illustrer les dépendances entre les différentes tables, le schéma de la base de données sera présenté ci-dessous :



Ce schéma montre les relations entre les différentes entités de la base de données, permettant de visualiser comment les données sont structurées et interconnectées pour supporter les fonctionnalités de l'application "JobBot".

IV- Technologies utilisées

A- Langages de programmation

- ✓ **Python** est choisi pour le développement de notre application web (backend) en raison de sa simplicité syntaxique qui réduit les erreurs et accélère le développement. Polyvalent, il convient aussi bien au traitement de données qu'au développement d'APIs et de sites web. Avec un large éventail de

bibliothèques et de frameworks comme **Django**, Python offre des outils robustes pour sécuriser et déployer rapidement des applications. Soutenu par une communauté active, Python garantit un support constant et des solutions efficaces, tout en offrant des performances solides pour les charges de travail importantes.

- ✓ **JavaScript** est utilisé pour le frontend en raison de sa compatibilité avec tous les navigateurs web modernes et de sa capacité à créer des interfaces utilisateur interactives et réactives. Grâce à des bibliothèques populaires comme **React.js**, JavaScript permet de construire des interfaces utilisateur dynamiques et conviviales. Son architecture orientée composants facilite la réutilisation du code et améliore la maintenabilité de l'application frontend. En outre, JavaScript bénéficie d'une large communauté de développeurs et d'une évolution rapide, garantissant l'adoption de meilleures pratiques et de technologies innovante

- ✓ **Mistral AI API :**
  Voici nos raisons pour lesquelles Mistral API à été choisi pour le projet :

1. **Open Source :** Mistral est un logiciel open source, ce qui signifie que son code source est accessible et modifiable par la communauté. Cela permet une transparence et une collaboration accrues, ainsi que la possibilité de personnaliser le logiciel selon les besoins spécifiques.
2. **Développé en France :** Mistral est développé par une entreprise française, ce qui peut être un avantage pour ceux qui préfèrent soutenir les entreprises locales ou qui ont des exigences spécifiques en matière de localisation.
3. **Sécurisé :** Mistral met l'accent sur la sécurité, ce qui est crucial lors de la gestion des workflows et des données sensibles via une API. En tant que logiciel open source, Mistral bénéficie également de l'audit et de la contribution de la communauté pour renforcer sa sécurité.
4. **Protection des données** : En tant que logiciel open source, Mistral offre un contrôle accru sur la manière dont les données sont gérées et traitées. Cela peut être important pour garantir la confidentialité et la protection des données des utilisateurs.

B- Frameworks et bibliothèques

- ✓ **Django :** Utilisé pour le développement côté serveur. Django est un framework web Python qui facilite le développement rapide et sécurisé d'applications web robustes.

- ✓ **React :** Utilisé pour le développement côté client. React est une bibliothèque JavaScript pour construire des interfaces utilisateur dynamiques et réactives.
- ✓ **Rest Framework:** Pour le développement de notre API backend, nous avons utilisé Django REST Framework (DRF), un puissant et flexible framework pour construire des APIs Web avec Django.
- ✓ **Axios :** Pour effectuer des requêtes HTTP depuis les navigateurs ou Node.js et connecter le front au back.
- ✓ **React-cookie :** React cookie est utile pour gérer les cookies, notamment pour le géré le cookie de consentement de l'utilisation des données personnelles.
- ✓ **Jazzadmin :** Jazzmin offre une interface utilisateur améliorée et moderne pour l'administration des applications Django.
- ✓ **Corsheaders :** Pour gérer les requêtes entre différents domaines et garantir la sécurité des échanges de données. Cela permet de contrôler les autorisations d'accès aux ressources entre différents domaines et de prévenir les attaques de type CORS.

C- Outils et plateformes

- ✓ **Trello :** Utilisé pour la gestion de projet. Trello permet de suivre l'avancement des tâches, de collaborer efficacement avec l'équipe et de gérer les sprints de développement.
- ✓ **Teams :** Utilisé comme moyen de communication principal. Microsoft Teams facilite la communication en temps réel, les meetings, et le partage de fichiers entre les membres de l'équipe.
- ✓ **GitLab :** Utilisé pour le dépôt de code. GitLab offre des fonctionnalités de gestion de versions, d'intégration continue (CI) et de déploiement continu (CD), assurant une gestion sécurisée et efficace du code source.
- ✓ **Azure :** Utilisé pour l'infrastructure. Azure fournit des services cloud robustes et scalables, y compris le déploiement d'applications, les bases de données, le stockage, et les outils d'analyse, permettant de gérer l'infrastructure de manière efficace et sécurisée.
- ✓ **Visual Studio Code :** Pour écrire du code nous avons utilisé l'IDE Vscode.

V- Sécurité

A- Authentification et autorisation des utilisateurs

- ✓ Les JWT sont des tokens sécurisés et auto-contenus qui peuvent être utilisés pour l'authentification. Ils sont généralement signés pour vérifier leur intégrité et peuvent être chiffrés pour garantir la confidentialité des informations qu'ils contiennent. Les JWT sont souvent utilisés dans les architectures stateless où le serveur n'a pas besoin de conserver l'état de l'authentification de session.
- ✓ Nous utilisons des cookies de session pour gérer l'authentification des utilisateurs. Lorsqu'un utilisateur se connecte, un cookie de session est créé sur son appareil. Ce cookie contient une clé de session unique qui est envoyée au serveur avec chaque requête pour vérifier l'authenticité de l'utilisateur sans avoir besoin de saisir à nouveau ses informations d'identification à chaque page visitée.
- ✓ Un mot de passe de 8 caractères minimum est requis lors de l'inscription.

B- Chiffrement des données

Toutes les communications entre le client(utilisateur) et le serveur sont sécurisés via HTTPS (HyperText Transfer Protocol Secure). HTTPS utilise le protocole TLS (Transport Layer Security) pour chiffrer les données en transit, assurant que les informations échangées (comme les identifiants de connexion et les données personnelles) ne peuvent pas être interceptées ou modifiées par des tiers non autorisés. Cela garantit une confidentialité et une intégrité des données maximales durant leur transmission sur le réseau.

Nous chiffrons les données sensibles en base de données tel que les mots de passe en sha256.

C - Protection contre les Cross Site Request Forgery

Nous avons mis en place des politiques de même origine pour restreindre les requêtes provenant de domaines externes. Nous avons mis en place des jetons CSRF uniques pour chaque formulaire sur notre site. Ces jetons sont vérifiés à chaque soumission de formulaire pour s'assurer que la requête provient d'une source légitime. Nous utilisons des méthodes HTTP sécurisées comme POST pour les actions sensibles afin de réduire les risques d'attaques CSRF.

D – Protection contre les attaques de type injection

Django utilise un ORM qui permet aux développeurs d'interagir avec la base de données en utilisant des objets Python au lieu de requêtes SQL brutes. Cela réduit considérablement le risque d'injections SQL, car les requêtes sont générées de manière sécurisée par Django.

Protection contre les injections SQL : Django intègre des mécanismes de protection contre les injections SQL en échappant automatiquement les caractères spéciaux

dans les requêtes SQL générées par l'ORM. Cela aide à prévenir les attaques d'injection SQL en garantissant que les données utilisateur sont correctement traitées.

VI Plan de Sauvegarde pour Azure PostgreSQL

Méthode 3-2-1 pour les Sauvegardes

Le plan de sauvegarde proposé respecte la méthode 3-2-1, largement reconnue pour garantir la sécurité et l'intégrité des données. Cette méthode consiste à avoir 3 copies des données sur 2 types de médias différents, avec au moins 1 copie hors site. En appliquant cette stratégie, nous assurons une protection robuste et fiable pour les données hébergées sur Azure PostgreSQL.

Sauvegardes Primaires sur Azure PostgreSQL

Justification :

- **Sauvegardes complètes quotidiennes gérées par Azure** : Azure PostgreSQL offre une gestion intégrée des sauvegardes, ce qui simplifie la maintenance et assure que les sauvegardes sont effectuées de manière fiable et sans intervention manuelle.
- **Rétention des sauvegardes : 30 jours** : Un cycle de rétention de 30 jours permet de couvrir les besoins de récupération en cas d'incidents récents, tout en optimisant l'utilisation de l'espace de stockage.

Fréquence :

- **Quotidien** : Les sauvegardes sont effectuées quotidiennement par le service Azure PostgreSQL.

Sauvegardes Secondaires sur Azure Blob Storage

Justification :

- **Quotidien : Création de sauvegardes manuelles avec pg_dump** : Utiliser pg_dump pour créer des sauvegardes manuelles offre une seconde ligne de défense. Cette méthode permet de capturer des états spécifiques de la base de données en complément des sauvegardes automatiques.

- **Quotidien : Synchronisation des sauvegardes vers Azure Blob Storage** : Stocker les sauvegardes sur Azure Blob Storage permet de bénéficier d'un stockage scalable et résilient.
- **Utilisation de la redondance géographique (GEO-RED) pour le stockage Blob** : Cette option assure que les données sont répliquées dans des régions géographiquement distantes, augmentant la résilience en cas de désastre régional.

Fréquence :

- **Quotidien** : Les sauvegardes manuelles sont effectuées quotidiennement et synchronisées immédiatement vers Azure Blob Storage.

Sauvegarde Hors Site sur Amazon S3

Justification :

- **Quotidien : Transfert des sauvegardes vers Amazon S3 via Azure Data Factory ou des scripts automatisés** : Transférer des sauvegardes vers Amazon S3 offre une couche supplémentaire de protection en cas de panne majeure sur Azure. Utiliser Azure Data Factory ou des scripts automatisés permet de gérer ces transferts de manière fiable et efficace.
- **Rétention des sauvegardes : 60 jours** : Une rétention plus longue (60 jours) sur Amazon S3 assure une couverture plus large pour la récupération de données, utile pour des besoins spécifiques de conformité ou de restauration historique.

Fréquence :

- **Quotidien** : Les sauvegardes sont transférées quotidiennement vers Amazon S3.

Surveillance et Tests

Justification :

- **Continue : Surveillance avec Azure Monitor** : Azure Monitor permet de surveiller en continu l'état des sauvegardes et de la base de données, détectant rapidement toute anomalie ou échec de sauvegarde.
- **Trimestriel : Tests de récupération pour valider l'intégrité des sauvegardes** : Tester régulièrement la récupération des sauvegardes assure que les processus de sauvegarde et de restauration fonctionnent comme prévu et que les données peuvent être récupérées en cas de besoin.

Fréquence :

- **Continue** : La surveillance est continue pour assurer une réponse rapide aux problèmes.
- **Trimestriel** : Les tests de récupération sont effectués tous les trois mois pour valider l'intégrité et la fiabilité des sauvegardes.

Ce plan de sauvegarde multifacette assure une protection robuste des données PostgreSQL hébergées sur Azure, combinant sauvegardes automatisées, redondance géographique, et tests réguliers pour garantir la disponibilité et l'intégrité des données.

VII-Plan de Déploiement pour Application React + Django sur Azure

---

1. Infrastructure sur Azure

- Réseau
- Utilisation d'un réseau virtuel (VNet)
- Création de sous-réseaux pour les composantes frontend, backend, base de données
- Stockage
- Utilisation de compte de stockage Azure pour les artefacts de déploiement
- Sécurité
- Configuration des groupes de sécurité réseau (NSG)

---

2. Configuration de la Machine Virtuelle (VM)

- Docker
- Utilisation d'une image Docker pour l'application React + Django
- Terraform
- Utilisation de modules Terraform pour la gestion de l'infrastructure

---

3. Déploiement de l'Application

- Backend (Django)
- Déploiement dans un conteneur Docker
- Frontend (React)
- Déploiement dans un conteneur Docker
- Utilisation de Nginx pour servir les fichiers statiques
- Base de Données (PostgreSQL)
- Déploiement dans une instance Azure

---

4. Gestion des Secrets et Configurations

- Utilisation d'Azure Key Vault pour les secrets
- Injection sécurisée des variables Terraform

---

5. Monitoring et Logs

Configuration de Azure Monitor et Log Analytics

---

6. Sécurité et Gouvernance

Mise en place de politiques de sécurité Azure

---

7. Plan de Sauvegarde et de Reprise Après Incident

Sauvegardes régulières de la base de données PostgreSQL

---

8. Documentation et Formation