

Documentation d'Architecture pour FlyAway Airlines

Crée par :

Hamza Belyahaioui

Saad Chabba

Sommaire

1. Introduction
   * Contexte de projet
   * Objectifs
   * Portée du projet
2. Vue ensemble de la solution
   * Architecture globale
   * Principes directeurs
   * Chois technique principale
3. Architecteur de traitement
   * Architecteur micro-service
   * Containerisation et orchestration
   * Model de déploiement
4. Solution de stockage et bases de donnes
   * Architecture basse de données polyglotte
   * Stratégie de persistance
   * Stockage des fichiers et document
   * Stratégie de sauvegarde et reprise après sinistre
5. Architecture Font
   * Application web
   * Application mobile
   * Architecture UI
   * Architecture de communication
6. Architecture Back
   * Api Gateway
   * Communication interservices
   * Gestion des donnes transactionnelles
   * Intégrations externes
7. Sécuritaire et conformité
   * Stratégie d’authentification
   * Gestion des autorisations
   * Protection des donnes
   * Conformité règlementaire
   * Sécurité opérationnelle
8. Scalabilité et haut disponibilité
   * Architecture multi-régions
   * Design pour la résilience
   * Gestion de charge
9. Optimisation des performances
   * Stratégie de mise en cache
   * Optimisation des requêtes
   * Optimisation Frontend
   * CI/CD
   * Stratégie de déploiement
   * Monitoring et Observabilité
10. Calcul des Coûts Azure
11. Main d'Œuvre et Compétences Requises

* Équipe de Projet
* Compétences Techniques Requises
* Phases de Déploiement

1. Conclusion

Introduction

### Contexte du projet

FlyAway Airlines a récemment fait appel à nos services suite à une insatisfaction concernant leur site web nouvellement implémenté et leur architecture de microservices. Les performances n'ont pas répondu à leurs attentes, et notre équipe est chargée de redessiner et d'optimiser entièrement leur écosystème informatique.

Notre mission est de concevoir une solution complète comprenant le frontend, le backend, les bases de données et les systèmes de stockage. Nous proposons une solution hébergée sur Microsoft Azure, comme illustré dans notre schéma d'architecture.

### Objectifs

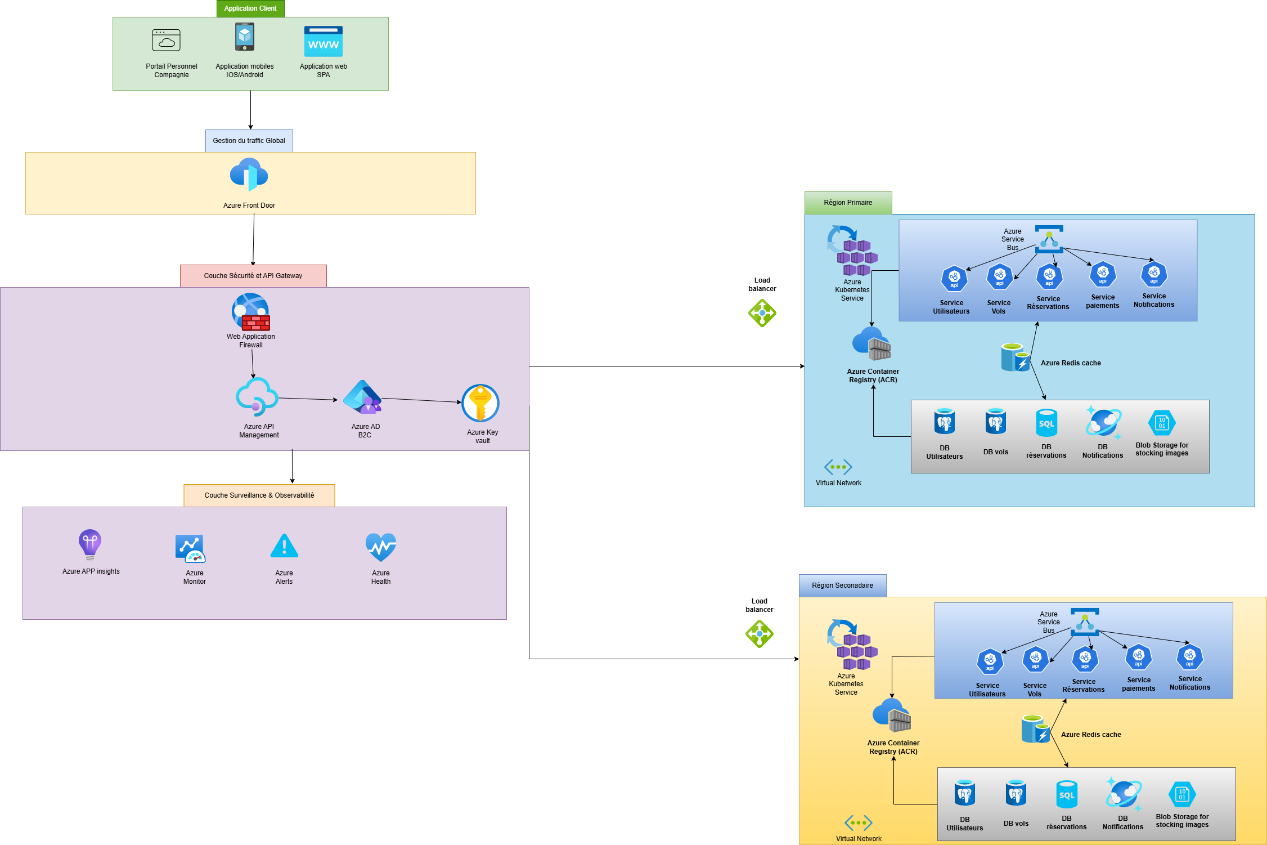
L'objectif principal de ce projet est de concevoir une architecture performante et évolutive sur Microsoft Azure qui garantira une disponibilité maximale du système. Nous nous engageons à assurer la sécurité des données et la conformité réglementaire dans toutes les régions où FlyAway Airlines opère. L'architecture optimisera l'expérience utilisateur sur tous les appareils et facilitera la maintenance et les évolutions futures grâce à des choix technologiques stratégiques et une conception modulaire.

### Portée du projet

La refonte couvrira l'ensemble de l'écosystème IT de FlyAway Airlines. Nous repenserons l'architecture frontend pour les applications web et mobiles en mettant l'accent sur la performance et l'ergonomie. L'architecture backend sera restructurée autour de microservices bien définis, communiquant via des interfaces standardisées. Les bases de données et systèmes de stockage seront sélectionnés selon les besoins spécifiques de chaque type de données. L'infrastructure réseau et la sécurité seront conçues selon les meilleures pratiques du secteur, et nous établirons des processus d'intégration continue et de déploiement modernes pour accélérer les mises en production.

Vue d'ensemble de la solution

# Architecture globale



Notre architecture est basée sur une approche cloud-native utilisant les services Azure. Elle s'articule autour de plusieurs composants clés:

- **Application Client**: Interfaces utilisateur (web, mobile, portal personnel)

- **Azure Front Door**: Pour la gestion du trafic global et la distribution géographique

- **API Gateway**: Pour l'orchestration et la sécurisation des APIs

- **Microservices**: Déployés dans Azure Container Registry/AKS

- **Azure AD**: Pour l'authentification et l'autorisation

- **Azure Key Vault**: Pour la gestion sécurisée des secrets

- **Stockage de données**: Combinaison de bases de données relationnelles et NoSQL

- **Monitoring**: Surveillance complète avec Azure Monitor, Alerts et Health

non structurées, Azure Cache for Redis pour les données à accès fréquent, et Azure Cognitive Search pour la recherche avancée de vols.

### Principes directeurs

Notre architecture est guidée par plusieurs principes fondamentaux. Nous visons une haute disponibilité grâce à une conception multi-régions intégrant des mécanismes de failover automatique. La scalabilité horizontale est assurée par la capacité d'ajouter des ressources à la demande selon les besoins. La sécurité est intégrée dès la conception à tous les niveaux de l'architecture (security by design). Nous adoptons une approche cloud-native en utilisant les services managés d'Azure lorsque c'est pertinent, tout en conservant la flexibilité nécessaire pour les besoins spécifiques. L'observabilité complète est garantie par des solutions de monitoring et logging centralisés qui offrent une visibilité totale sur l'état et les performances du système.

# Choix technologiques principaux

Pour assurer une solution performante, scalable et maintenable, nous avons sélectionné les technologies suivantes:

### Frontend

**Web Application** : React.js avec TypeScript pour une application SPA robuste et typée

**Mobile iOS** : Swift 5 avec SwiftUI pour une expérience native optimale

**Mobile Android** : Kotlin avec Jetpack Compose pour une interface moderne

**Framework CSS** : Tailwind CSS pour une interface utilisateur réactive et élégante

**State Management** : Redux pour la gestion d'état côté web, SwiftUI/Combine pour iOS

### Backend

**Langage Principal** : C# .NET 8 pour les microservices critiques (haute performance)

**Langages Secondaires** : Node.js avec TypeScript pour certains microservices non-critiques

**Framework API** : ASP.NET Core pour les microservices C#, Express.js pour Node.js

**Communication** : gRPC pour les communications inter-services à haute performance

**API Gateway** : API Management avec Ocelot pour le routage et la gestion des politiques

**Authentification** : IdentityServer4 intégré avec Azure AD

### Bases de Données et Stockage

**Données Relationnelles** : Azure SQL Database (SQL Server) pour les données structurées

**Données NoSQL** : Azure Cosmos DB avec API SQL pour les données de réservation

**Cache** : Redis pour la mise en cache distribuée

**Stockage d'Objets** : Azure Blob Storage pour les documents et images

**Indexation et Recherche** : Azure Cognitive Search pour la recherche avancée de vols

### DevOps et Infrastructure

**Containerisation** : Docker pour tous les microservices

**Orchestration** : Kubernetes géré via AKS

**CI/CD** : Azure DevOps Pipelines pour l'intégration et le déploiement continus

**IaC** : Terraform et ARM Templates pour l'infrastructure as code

**Monitoring** : Application Insights, Prometheus et Grafana pour la surveillance

Cette pile technologique a été choisie pour maximiser la performance, la fiabilité et la maintenabilité du système, tout en tirant parti des compétences existantes et des meilleures pratiques de l'industrie.

## Architecture de traitement

### Architecture microservices

L'architecture backend est organisée en microservices fonctionnels, chacun dédié à un domaine métier spécifique. Le service d'authentification et gestion des utilisateurs s'appuie sur Azure Active Directory B2C pour gérer les identités et les accès, avec support multi-facteur et SSO, ainsi que la gestion des rôles et permissions. Cette approche centralise la sécurité tout en offrant une expérience utilisateur fluide.

Le service de gestion des profils prend en charge les informations personnelles des utilisateurs, le stockage et la gestion des documents sensibles, ainsi que les préférences utilisateur. Ces données sont stockées de façon sécurisée dans Azure Cosmos DB, avec des mécanismes de chiffrement appropriés.

Pour la recherche de vols, nous avons conçu un service spécifique exploitant Azure Cognitive Search comme moteur de recherche optimisé. Ce service gère également la géolocalisation, la cartographie, ainsi que le filtrage et le tri des résultats, offrant ainsi une expérience de recherche riche et performante.

Le service de réservation est au cœur du métier de FlyAway Airlines. Il gère les disponibilités en temps réel, implémente le verrouillage temporaire des sièges pendant la réservation pour éviter les conflits, et traite les changements et annulations avec une logique métier sophistiquée.

Pour les transactions financières, le service de paiement assure le traitement sécurisé des transactions, l'intégration avec les fournisseurs de paiement, et la gestion des remboursements. Ce service est conçu avec une attention particulière à la sécurité et à l'atomicité des transactions.

Enfin, le service de gestion des vols fournit une interface pour le personnel de la compagnie, permettant la validation des voyageurs et la gestion des embarquements. Ce service interne est crucial pour les opérations quotidiennes de la compagnie aérienne.

### Conteneurisation et orchestration

Tous les services sont conteneurisés avec Docker et orchestrés via Azure Kubernetes Service (AKS). Cette approche assure une gestion efficace des ressources, permettant d'allouer dynamiquement CPU et mémoire selon les besoins de chaque service. Le scaling automatique basé sur la charge permet d'ajuster le nombre d'instances en fonction du trafic. AKS offre également une résilience accrue avec des capacités d'auto-healing qui redémarrent automatiquement les conteneurs défaillants. Cette architecture garantit un déploiement cohérent entre les différents environnements, de développement à production.

### Modèle de déploiement

Nous proposons un modèle de déploiement hybride avec un environnement de production principal sur Azure Kubernetes Service (AKS) déployé en multi-zones de disponibilité dans les régions principales. L'auto-scaling des pods est configuré en fonction des métriques de charge pour optimiser les ressources.

Pour assurer une redondance maximale, un environnement de production secondaire est maintenu dans le datacenter propriétaire de FlyAway Airlines. Ce cluster Kubernetes on-premise est géré par Azure Arc, permettant une gestion unifiée avec l'environnement cloud. Azure Site Recovery assure la synchronisation continue entre les deux environnements.

Les environnements de développement et test sont également déployés sur Azure Kubernetes Service, avec des instances de taille réduite mais dans une configuration identique à la production. Des templates Azure Resource Manager (ARM) garantissent la parité des environnements, évitant ainsi les problèmes liés aux différences de configuration.

## Solutions de stockage et bases de données

### Architecture de données polyglotte

Notre approche de persistance polyglotte combine plusieurs types de bases de données Azure pour répondre aux différents besoins de l'application. Azure SQL Database constitue notre base de données relationnelle principale pour les données transactionnelles comme les réservations et paiements, les informations de vol et disponibilité des sièges, ainsi que la configuration système et les paramètres. Cette base de données est configurée en mode hyper-scale avec géo-réplication pour garantir haute disponibilité et performances.

Azure Cosmos DB est utilisé comme base de données NoSQL pour les profils utilisateurs et préférences, l'historique de navigation et recherches, ainsi que les données semi-structurées et évolutives. Sa configuration multi-région avec distribution globale permet une faible latence d'accès partout dans le monde.

Pour optimiser les performances, Azure Cache for Redis sert de cache distribué pour les sessions utilisateurs, le cache de recherche de vols, les verrouillages temporaires pour les réservations et les files d'attente légères. Cette couche de cache réduit significativement la charge sur les bases de données principales.

Azure Cognitive Search est notre moteur de recherche spécialisé pour l'indexation des vols et destinations, la recherche géospatiale pour la cartographie, la recherche full-text avec facettes, et l'analyse en temps réel des tendances de recherche. Ce service managé offre des capacités de recherche avancées sans nécessiter une infrastructure complexe.

### Stratégie de persistance

Notre stratégie de persistance intègre des patterns modernes adaptés à une architecture distribuée. Nous utilisons le pattern Saga pour gérer les transactions distribuées et maintenir la cohérence entre services. Ce mécanisme permet de coordonner des opérations complexes comme les réservations qui impliquent plusieurs services.

Azure Data Factory met en œuvre le Change Data Capture (CDC) pour synchroniser les données entre différents systèmes, assurant ainsi la cohérence des informations à travers l'écosystème. Pour les opérations critiques comme les réservations et paiements, nous implémentons l'Event Sourcing avec Azure Event Grid, permettant un audit complet et la possibilité de reconstruire l'état du système à n'importe quel point dans le temps.

Nous adoptons également le pattern CQRS (Command Query Responsibility Segregation) pour séparer les opérations de lecture et d'écriture. Cette approche permet d'optimiser chaque type d'opération et d'améliorer les performances globales du système.

### Stockage des fichiers et documents

Le stockage des fichiers et documents est réparti entre plusieurs solutions selon la nature et la sensibilité des données. Azure Blob Storage sert de système principal pour le stockage des documents et médias, notamment les copies de passeports et documents d'identité, les photos de profil et autres médias. Ce service offre une réplication géographique pour assurer la disponibilité des données.

Azure Files est utilisé pour le partage de fichiers entre services, le stockage temporaire pour les traitements de documents et le partage de configurations. Cette solution s'intègre parfaitement avec les conteneurs et services Azure.

Pour les données hautement sensibles, nous conservons un stockage on-premise dans l'infrastructure propriétaire de FlyAway Airlines. Ces données bénéficient d'un cryptage au repos et en transit, ainsi que de politiques de rétention et d'archivage conformes aux réglementations applicables.

### Stratégie de sauvegarde et reprise après sinistre

Notre plan de continuité d'activité repose sur une stratégie de sauvegarde robuste avec Azure Backup pour des sauvegardes incrémentales toutes les heures et des sauvegardes complètes quotidiennes. Azure Site Recovery assure la réplication cross-région des données critiques, permettant un basculement rapide en cas de défaillance d'une région entière.

Le plan de reprise après sinistre est conçu avec un RTO (Recovery Time Objective) inférieur à 1 heure et un RPO (Recovery Point Objective) inférieur à 5 minutes, garantissant une reprise rapide des opérations avec une perte de données minimale. Des tests de restauration réguliers sont menés dans Azure DevTest Labs pour valider les procédures et s'assurer que l'équipe est préparée en cas d'incident.

## Architecture frontend

### Applications web

Notre architecture web repose sur le Server-Side Rendering (SSR) avec Next.js pour améliorer significativement les performances perçues par l'utilisateur et optimiser le référencement (SEO). Cette approche supporte également le rendu hybride, combinant SSR et Client-Side Rendering (CSR) pour offrir la meilleure expérience possible.

Les applications web sont également conçues comme des Progressive Web Apps (PWA), offrant des fonctionnalités hors ligne basiques, la possibilité d'installation sur l'écran d'accueil et des notifications push. Ces caractéristiques rapprochent l'expérience web d'une application native.

Pour optimiser les temps de chargement, nous implémentons une stratégie sophistiquée incluant le lazy loading des composants, le code splitting par route et le prefetching intelligent des ressources. Ces techniques réduisent considérablement le temps de chargement initial et améliorent la réactivité de l'application.

L'hébergement des applications web est réparti entre Azure Static Web Apps pour les composants statiques et Azure App Service pour le backend Next.js. Cette séparation permet d'optimiser les coûts tout en maintenant d'excellentes performances.

### Applications mobiles

Les applications mobiles FlyAway Airlines sont développées avec React Native pour offrir une interface utilisateur fluide et native sur iOS et Android. Cette approche permet d'accéder aux fonctionnalités natives du device comme l'appareil photo et le GPS, tout en partageant une grande partie du code avec les applications web.

Nous implémentons une stratégie offline-first avec synchronisation en arrière-plan, mise en cache des données fréquemment utilisées et files d'attente pour les actions offline. Les utilisateurs peuvent ainsi consulter leurs réservations et cartes d'embarquement même sans connexion internet, ce qui est crucial pour une application de voyage.

Les notifications push sont gérées via Azure Notification Hubs, permettant d'envoyer des alertes personnalisées concernant les changements de vol, les offres spéciales ou les informations d'embarquement.

### Architecture UI/UX

Un design system centralisé et cohérent constitue la base de notre approche UI/UX. Cette bibliothèque de composants réutilisables, associée à des thèmes et styles standardisés, garantit une expérience unifiée sur toutes les plateformes. Une documentation interactive accompagne ce design system pour faciliter son adoption par les équipes de développement.

L'accessibilité est une priorité, avec une conformité WCAG 2.1 niveau AA. Nos interfaces supportent les lecteurs d'écran, proposent une navigation clavier complète et respectent les exigences de contraste et de taille pour accommoder tous les utilisateurs.

L'internationalisation est intégrée nativement avec support multi-langue, formats de date, heure et devise localisés, ainsi que du contenu adapté aux spécificités culturelles des différentes régions où opère FlyAway Airlines.

### Architecture de communication

Notre architecture de communication frontend-backend utilise plusieurs approches complémentaires. Les API REST sont employées pour les opérations CRUD standard, offrant simplicité et compatibilité. Pour les requêtes complexes et l'optimisation du trafic mobile, nous implémentons GraphQL avec Azure API Management, permettant aux clients de spécifier exactement les données dont ils ont besoin.

Les mises à jour en temps réel comme la disponibilité des sièges ou les alertes de prix sont gérées via Azure SignalR Service, remplaçant avantageusement les WebSockets traditionnels avec une solution managée. Cette approche permet des expériences interactives sans surcharger le serveur avec des requêtes de polling.

## Architecture backend

### API Gateway

Azure API Management constitue le point d'entrée unifié pour toutes nos APIs. Ce service assure le routage intelligent des requêtes vers les microservices appropriés, implémente des politiques de throttling et quotas pour prévenir les abus, et propose une mise en cache des réponses pour améliorer les performances. L'authentification centralisée simplifie la sécurité en appliquant des politiques cohérentes à toutes les APIs.

La gestion des versions des APIs est un aspect crucial de notre architecture. Azure API Management supporte plusieurs versions d'APIs simultanément, permettant des migrations progressives sans perturber les clients existants. Une stratégie de dépréciation claire est définie pour accompagner l'évolution des services.

### Communication inter-services

La communication entre services utilise des approches synchrones et asynchrones selon les besoins. Pour les requêtes nécessitant une réponse immédiate, nous privilégions la communication synchrone via REST. Les APIs sont documentées avec OpenAPI (Swagger) pour faciliter leur utilisation.

Pour les opérations distribuées, nous utilisons la communication asynchrone via Azure Service Bus et Event Grid. Cette approche permet la publication d'événements de domaine, le traitement en arrière-plan et assure un bon découplage des services. Les services peuvent évoluer indépendamment tant qu'ils respectent les contrats d'événements.

### Gestion des données transactionnelles

Les transactions distribuées représentent un défi majeur dans une architecture microservices. Nous utilisons le pattern Saga orchestré par Azure Durable Functions pour coordonner les transactions multi-services et implémenter des mécanismes de compensation en cas d'échec partiel.

Nos consistency patterns sont adaptés aux différents types de données. Nous appliquons une cohérence éventuelle pour les données non critiques, réduisant ainsi la latence et améliorant la disponibilité. Pour les transactions financières, nous maintenons une cohérence forte afin de garantir l'intégrité des données sensibles.

### Intégrations externes

Azure API Management gère également nos APIs pour les partenaires et fournisseurs, incluant les GDS (Global Distribution Systems), les fournisseurs de paiement et les systèmes d'assurance voyage. Cette approche uniformise la sécurité et le monitoring des intégrations externes.

Notre architecture d'intégration emploie Azure Logic Apps pour les intégrations complexes, avec des adaptateurs standardisés pour chaque système externe. Nous implémentons des circuit breakers pour isoler les défaillances, des politiques de retry avec exponential backoff pour la résilience, et des dead letter queues pour gérer les messages problématiques.

## Sécurité et conformité

### Stratégie d'authentification

Azure Active Directory B2C constitue le cœur de notre stratégie d'authentification. Ce service supporte les flux OAuth 2.0 modernes comme Authorization Code avec PKCE et permet l'intégration avec des fournisseurs d'identité externes tels que Google et Facebook. Les jetons JWT sont signés et chiffrés pour garantir leur intégrité et confidentialité.

L'authentification multi-facteur est disponible via plusieurs méthodes: TOTP (Time-based One-Time Password), push notifications sécurisées et biométrie pour les applications mobiles. Cette approche en couches renforce significativement la sécurité des comptes utilisateurs.

### Gestion des autorisations

Le contrôle d'accès est implémenté via RBAC (Role-Based Access Control) natif d'Azure, définissant des rôles utilisateurs comme client, agent ou administrateur avec des permissions granulaires. Cette approche simplifie la gestion des droits d'accès à grande échelle.

Pour les cas plus complexes, nous utilisons ABAC (Attribute-Based Access Control) qui permet des restrictions basées sur le contexte comme la géolocalisation ou le type d'appareil. Ces politiques dynamiques sont mises en œuvre avec Azure AD Conditional Access, offrant une sécurité contextuelle avancée.

### Protection des données

Le chiffrement est omniprésent dans notre architecture, avec TLS 1.3 pour le chiffrement en transit et AES-256 pour le chiffrement au repos. La gestion des clés est centralisée dans Azure Key Vault, simplifiant l'administration des secrets et permettant une rotation régulière des clés.

Les données sensibles comme les cartes de crédit font l'objet d'une tokenisation pour limiter l'exposition des informations confidentielles. Le masquage dynamique des données avec Azure SQL Database Dynamic Data Masking permet d'afficher des informations différentes selon les profils utilisateurs, limitant l'accès aux données sensibles.

### Conformité réglementaire

Notre architecture est conçue pour respecter les principales réglementations applicables au secteur du transport aérien. Pour le GDPR, nous maintenons un registre des traitements, implémentons des mécanismes de consentement explicites et supportons le droit à l'oubli et la portabilité des données.

La conformité PCI DSS est assurée par une segmentation réseau rigoureuse avec Azure Virtual Network, une gestion sécurisée des données de paiement et un système complet de logging et d'audit avec Azure Monitor et Log Analytics. Des audits réguliers valident cette conformité.

Nous prenons également en compte les réglementations locales spécifiques à chaque région d'opération de FlyAway Airlines, en adaptant nos processus et contrôles en conséquence.

### Sécurité opérationnelle

Azure Web Application Firewall constitue notre première ligne de défense, offrant une protection contre les attaques OWASP Top 10 et autres menaces web courantes. Azure DDoS Protection Standard protège l'infrastructure contre les attaques par déni de service, tandis que des filtres géographiques limitent l'accès depuis des régions non desservies.

Notre surveillance continue s'appuie sur Azure Security Center pour l'analyse comportementale et la détection d'anomalies. Azure Sentinel fournit des capacités SIEM (Security Information and Event Management) intégrées, centralisant la collecte et l'analyse des logs de sécurité pour identifier rapidement les incidents potentiels.

## Scalabilité et haute disponibilité

### Architecture multi-régions

Notre architecture multi-régions repose sur un déploiement actif-actif dans plusieurs régions Azure. Azure Traffic Manager assure le routage géographique intelligent des utilisateurs vers la région la plus proche, optimisant ainsi les temps de latence. Les données critiques bénéficient d'une réplication synchrone entre régions, tandis que les données secondaires sont répliquées de manière asynchrone pour un meilleur équilibre entre cohérence et performances.

La stratégie de basculement prévoit une bascule automatique en cas de défaillance d'une région, avec des tests réguliers des procédures via Azure Site Recovery. Pour les services critiques, nous visons un RTO (Recovery Time Objective) inférieur à 1 minute, garantissant une continuité de service presque imperceptible pour les utilisateurs.

### Design pour la résilience

Les circuit breakers sont implémentés avec la bibliothèque Polly pour .NET, permettant d'isoler les défaillances, d'offrir un fallback gracieux et de faciliter l'auto-récupération des services. Cette approche évite les cascades d'erreurs qui pourraient affecter l'ensemble du système.

Le pattern Bulkhead nous permet d'isoler les ressources par service, prévenant ainsi la propagation des défaillances. Chaque service dispose de ses propres ressources dédiées, évitant qu'un pic de charge sur un service n'affecte les autres.

Nos retry patterns incluent l'exponential backoff pour espacer progressivement les tentatives, le jitter pour éviter les effets de synchronisation, et des dead letter queues dans Azure Service Bus pour gérer les messages qui ne peuvent pas être traités après plusieurs tentatives.

### Gestion de charge

Azure CDN distribue le contenu statique à travers un réseau mondial de points de présence, réduisant la latence et soulageant les serveurs d'application. Le edge caching des réponses d'API améliore encore les performances pour les requêtes fréquentes.

Le load balancing est géré à plusieurs niveaux: Azure Application Gateway pour le trafic HTTP/HTTPS avec des fonctionnalités avancées comme le SSL termination et le routage basé sur le contenu; Azure Load Balancer pour les protocoles non-HTTP nécessitant un équilibrage de charge de niveau 4; et Azure Traffic Manager pour l'optimisation du routage global entre régions.

Le rate limiting est implémenté dans Azure API Management avec des quotas par utilisateur et par IP, ainsi que des mécanismes de throttling pour prévenir les abus. Ces protections assurent que les ressources restent disponibles pour tous les utilisateurs légitimes, même en cas de pics de trafic.

## Optimisation des performances

### Stratégie de mise en cache

Notre stratégie de mise en cache multi-niveaux comprend le cache navigateur et application mobile pour les données locales, Azure CDN pour le edge caching, Azure API Management pour le caching des réponses d'API, Azure Cache for Redis pour le cache applicatif distribué, et les mécanismes de cache intégrés aux bases de données. Chaque niveau est optimisé pour des types de données et des patterns d'accès spécifiques.

L'invalidation intelligente du cache est basée sur les événements avec Azure Event Grid, permettant de rafraîchir les données uniquement lorsque nécessaire. Nous utilisons également un TTL adaptatif qui ajuste la durée de vie des objets en cache selon leur fréquence d'accès et de modification.

### Optimisation des requêtes

Azure SQL Database Query Store nous permet d'analyser en continu les performances des requêtes et d'optimiser les index en conséquence. Nous utilisons aussi Azure SQL Database Intelligent Insights pour identifier automatiquement les problèmes de performance et recevoir des recommandations d'optimisation.

Les read replicas avec Azure SQL Database geo-replicas permettent de séparer les charges de lecture et d'écriture, améliorant ainsi les performances des deux types d'opérations. La réplication asynchrone pour les lectures offre un bon équilibre entre cohérence et performance.

Pour les bases de données volumineuses, nous implémentons le sharding avec partitionnement par région géographique et par ID client. Azure SQL Database elastic pools optimise l'utilisation des ressources en partageant les capacités entre plusieurs bases de données selon leurs besoins.

### Optimisation frontend

L'optimisation des assets comprend la minification et le bundling du code JavaScript et CSS, la compression avec Brotli et Gzip, ainsi que l'utilisation de formats d'image optimisés comme WebP. Ces techniques réduisent considérablement la taille des ressources à télécharger.

Nos stratégies de chargement incluent l'extraction du Critical CSS pour un rendu rapide de la page, le preload des ressources critiques pour anticiper les besoins, et le lazy loading des images et composants non-critiques pour prioriser l'affichage du contenu principal. Cette approche progressive améliore considérablement l'expérience utilisateur perçue.

### CI/CD Pipeline

Notre approche GitOps s'appuie sur Azure DevOps ou GitHub Actions pour automatiser les workflows de build, test et déploiement. Le déploiement continu est appliqué pour les environnements de développement, tandis que les environnements de production bénéficient d'un déploiement progressif avec validation à chaque étape.

L'Infrastructure as Code est mise en œuvre via Azure Resource Manager (ARM) templates et Bicep pour la provision des ressources cloud. Les déploiements applicatifs utilisent des manifests Kubernetes et des Helm charts pour garantir la cohérence et la répétabilité des déploiements.

### Stratégie de déploiement

Pour les changements majeurs, nous utilisons la stratégie Blue/Green deployments avec Azure DevOps, permettant de basculer rapidement entre deux environnements identiques. Les Canary releases sont employées pour tester les fonctionnalités risquées sur un sous-ensemble d'utilisateurs via Azure App Service deployment slots.

Les feature flags permettent l'activation progressive des fonctionnalités, offrant un contrôle fin sur le déploiement et la possibilité de désactiver rapidement une fonctionnalité problématique sans déploiement complet.

### Monitoring et observabilité

Notre logging centralisé repose sur Azure Log Analytics, avec structured logging incluant des contextes de corrélation pour faciliter le debugging. La rétention des logs est configurée selon les exigences réglementaires et les besoins opérationnels.

Le monitoring combine Azure Monitor pour les métriques système et applicatives, Azure Application Insights pour le monitoring applicatif détaillé, Azure Workbooks pour la visualisation personnalisée, et Azure Alerts pour la gestion proactive des alertes.

Le tracing distribué avec Application

CALCULE DES COUTS AZURE

## LieN VERS L’excel pour la calcule des Coûts -> [Télécharger](https://github.com/chabbasaad/4ARCL/raw/refs/heads/master/ExportedEstimate_4ARCL_saad_hamza.xlsx)

MAIN D’OUVRE ET COMPETENCES REQUISES

### Equipe de projet

|  |  |  |
| --- | --- | --- |
| Rôle | Nombre | Responsabilités |
| Architecte Cloud | 1 | Conception de l'architecture globale, choix technologiques |
| DevOps Engineer | 2 | Configuration et maintenance de l'infrastructure cloud, CI/CD |
| Frontend Developer | 3 | Développement des interfaces web et mobiles |
| Backend Developer | 4 | Implémentation des microservices et APIs |
| Database Administrator | 1 | Conception et optimisation des bases de données |
| Security Specialist | 1 | Implémentation des mesures de sécurité et conformité |
| QA Engineer | 2 | Tests et assurance qualité |

### Compétences Techniques Requises

* Expertise Azure (AKS, Azure SQL, Cosmos DB, etc.)
* Développement .NET Core / Node.js pour les microservices
* React / Angular pour le frontend web
* Swift / Kotlin pour les applications mobiles
* Connaissances en sécurité cloud et conformité RGPD
* Expérience avec Kubernetes et Docker
* Maîtrise des pratiques DevOps et CI/CD

### Phases de Déploiement

|  |  |  |
| --- | --- | --- |
| Phase | Durée | Description |
| Phase 1: Analyse et Conception | 4 semaines | Analyse détaillée des besoins, conception de l'architecture |
| Phase 2: Mise en place de l'Infrastructure | 3 semaines | Configuration de l'environnement Azure, mise en place du réseau et de la sécurité |
| Phase 3: Développement des Microservices | 8 semaines | Implémentation des différents microservices |
| Phase 4: Développement Frontend | 6 semaines | Développement des interfaces utilisateur web et mobiles |
| Phase 5: Intégration et Tests | 4 semaines | Tests d'intégration, de charge et de sécurité |
| Phase 6: Migration des Données | 2 semaines | Migration des données depuis l'ancien système |
| Phase 7: Déploiement en Production | 1 semaine | Mise en production progressive |
| Phase 8: Stabilisation et Optimisation | 4 semaines | Surveillance, résolution des problèmes, optimisation des performances |

*Durée totale estimée* : **32 semaines** (environ 8 mois)

Et pour la section de références et ressources documentaires en vous recommende de visite les lien suivant : <https://github.com/chabbasaad/4ARCL>

## Conclusion

L'architecture Azure proposée pour FlyAway Airlines offre une solution robuste, évolutive et performante qui répond aux exigences actuelles et futures de l'entreprise. En combinant une approche microservices moderne, des technologies cloud-native Microsoft et une stratégie de déploiement multi-régions, nous assurons une disponibilité maximale et des performances optimales à l'échelle mondiale.

Les choix technologiques ont été guidés par les besoins spécifiques de chaque composant du système, en privilégiant l'écosystème Microsoft Azure et les meilleures pratiques de l'industrie. La sécurité et la conformité ont été intégrées dès la conception pour garantir la protection des données sensibles et le respect des réglementations internationales.

L'intégration native des services Azure permet une gestion simplifiée et une cohérence accrue de l'ensemble de l'infrastructure. Les outils de monitoring et d'analyse intégrés offrent une visibilité complète sur les performances et la santé du système.

La mise en œuvre de cette architecture permettra à FlyAway Airlines de surmonter les limitations actuelles de son système et d'offrir une expérience utilisateur exceptionnelle à ses clients dans le monde entier, tout en bénéficiant de la fiabilité et de la scalabilité de la plateforme Microsoft Azure.