# 1. Panoramica dell'architettura

## 1.1 Obiettivi principali

1. **Erogare un servizio RAG** (Retrieval Augmented Generation):

    - Utilizzo di un database vettoriale per la ricerca semantica.
    - Mantenimento di un database NoSQL scalabile (Azure Cosmos DB for MongoDB).
    - Esecuzione della logica applicativa in un servizio gestito (Azure App Service).

2. **Sicurezza by design**:

    - Gestione centralizzata delle identità e degli accessi con Azure AD.
    - Limitazione del traffico tramite Virtual Network e private endpoints.
    - Protezione di segreti e chiavi con Azure Key Vault.
    - Monitoraggio continuo con Azure Monitor.

---

# 2. Autenticazione e Identità

## 2.1 Managed Identities

- **System-Assigned Managed Identity** per l'App Service:
    - L'applicazione ottiene token da Azure AD.
    - Accesso a Key Vault, Cosmos DB, database vettoriale tramite RBAC e/o politiche assegnate in Azure.

---

# 3. Virtual Network (VNet) e sicurezza di rete

## 3.1 Creazione della VNet

- **VNet dedicata** (ad esempio, `rag-vnet`):
    - Collocata in una o più region Azure (in base a requisiti di resilienza).

    - Suddivisione in **subnet** per isolare i componenti (ad es. `subnet-data`, `subnet-management`).

## 3.2 Integrazione dell'App Service con la VNet

- App Service con piano che supporta l'**Azure VNet Integration** per connettersi internamente a Cosmos DB, database vettoriale e Key Vault tramite IP privati.

## 3.3 Private Endpoints

- **Cosmos DB** e **Database Vettoriale**:
    - Creare i private endpoint sulle rispettive risorse e assegnarli alla `subnet-data`.
    - In questo modo, il traffico passa solo all'interno della VNet, senza esposizione pubblica.
- **Key Vault**:
    - Configurare un private endpoint che limiti l'accesso solo ai servizi all'interno della VNet.

---

# 4. Azure App Service

## 4.1 Architettura dell'applicazione

- **Hosting**:
    - Piano con supporto VNet e prestazioni adeguate.
- **Endpoint**:
    - **HTTP/HTTPS** per ricevere richieste dagli utenti/client.

## 4.2 Accesso a servizi esterni

- **Cosmos DB**:
    - Utilizzo della **stringa di connessione** archiviata in Key Vault.
    - L'App Service recupera il segreto con la Managed Identity.
- **Database Vettoriale**:
    - Gestione delle credenziali di accesso in Key Vault.
    - Query e indicizzazioni passano via private endpoint.

## 4.3 Protezione e monitoraggio

- **Application Insights**:

    - Attivare la telemetria (richieste, errori, dipendenze) per individuare colli di bottiglia e possibili problemi di performance.
- **Deployment slot**:
    - Usare **slot** (ad es. "staging", "production") per testare nuove versioni, riducendo rischi di interruzione del servizio.
- **Always On**:
    - Abilitare l'opzione "Always On" per evitare il cold start e mantenere l'applicazione sempre reattiva.

---

# 5. Azure Cosmos DB for MongoDB

## 5.1 Configurazione

- **API MongoDB**:
    - Scegliere la versione di API supportata (ad es. 4.0 o successivi) in base alle esigenze di compatibilità con driver e framework.
- **Throughput**:
    - Impostare un livello di RU/s (Request Unit) adeguato al carico stimato; si può usare il **Serverless** se il traffico è intermittente.
- **Partitioning**:
    - Definire una chiave di partizionamento adatta per ottimizzare la distribuzione dei dati.

## 5.2 Sicurezza

- **Private Endpoint**:
    - Attivo su `subnet-data`, con accesso consentito solo all'App Service (tramite VNet Integration).
- **Authentication**:
    - Utilizzo di token e RBAC nativi di Cosmos DB.
- **Encryption at rest**:
    - Abilitata di default su Cosmos DB.

---

# 6. Database Vettoriale

## 6.1 Funzionalità e capacità

- **Indici vettoriali**:
  - Supporto per embedding ad alta dimensionalità.
  - Metriche di similarità configurabili (coseno, prodotto scalare, distanza euclidea).
- **Elaborazione dei contenuti**:
  - Capacità di gestire e indicizzare vettori generati da modelli di embedding.
  - Supporto per metadati e filtri aggiuntivi per affinare le ricerche.

## 6.2 Sicurezza e connessioni

- **Private Endpoint**:
  - Consentire solo connessioni dalla subnet interna.
- **Autenticazione e autorizzazione**:
  - Conservare le credenziali di accesso in Key Vault.
  - Ruotare periodicamente le chiavi per prevenire utilizzi non autorizzati.
  - Implementare controlli di accesso granulari quando disponibili.

---

# 7. Azure Key Vault

## 7.1 Gestione segreti e chiavi

- **Segreti**:
  - Conservare in Key Vault le stringhe di connessione di Cosmos DB, le credenziali del database vettoriale, e altri token.
- **Access Policies / RBAC**:
  - Abilitare l'accesso solo alla Managed Identity dell'App Service (Get, List).

---

# 8. Monitoraggio e sicurezza

## 8.1 Azure Monitor e Application Insights

- **Metrics & Logs**:
    - Centralizzare metriche (CPU, memoria, richieste dell'App Service, RU/s di Cosmos, performance del database vettoriale) e log (App Service logs, diagnostica).

## 8.2 Sicurezza

- **Encryption in transit**:
    - Utilizzo di TLS 1.2+ per tutte le comunicazioni.

---

# 9. Best Practice

1. **Ruotazione regolare di segreti e chiavi**:
    - Periodicamente cambiare le credenziali di accesso per tutti i servizi.
2. **Managed Identity ovunque possibile**:
    - Evitare credenziali statiche, soprattutto in code e file di configurazione.
3. **Private Endpoint per tutti i servizi critici**:
    - Ridurre la superficie di attacco e non esporre endpoint pubblici.
4. **Ottimizzazione delle query vettoriali**:
    - Configurare correttamente gli indici vettoriali per bilanciare precisione e performance.

---

# 10. Modello Linguistico e Servizi Vocali

## 10.1 Azure OpenAI Service

- **Modello LLM**:
    - Utilizzo di **GPT-4o mini** per la generazione di risposte.
    - Configurazione di parametri come temperatura e max_tokens in base alle esigenze di risposta.
- **Integrazione RAG**:
    - Implementazione del pattern di prompt RAG per incorporare informazioni dal database vettoriale.
    - Utilizzo di tecniche di chunking e retrieval ottimizzate per migliorare la pertinenza delle risposte.

## 10.2 Servizi Vocali Azure

- **Speech-to-Text**:
    - Utilizzo di **Azure Speech Service** per la trascrizione vocale in tempo reale.
    - Supporto multilingua per consentire interazioni in diverse lingue.
- **Text-to-Speech**:
    - Implementazione di voci neurali di Azure Speech Service per una sintesi vocale naturale.

---

## Conclusione

Questo documento delinea un'**architettura target** su Azure per un sistema di Retrieval Augmented Generation (RAG). L'infrastruttura sfrutta i servizi cloud (App Service, Cosmos DB for MongoDB, database vettoriale, Key Vault, Azure OpenAI e Azure Speech Services) e implementa best practice di sicurezza (VNet e private endpoint, Key Vault) e di monitoraggio (Azure Monitor) per garantire affidabilità e protezione dei dati. L'implementazione corretta di ciascun componente, con un'adeguata attenzione alla rete e alla gestione delle credenziali, produce un sistema robusto e scalabile per il chatbot RAG con capacità di interazione vocale.