Rapport de gestion de projet

1. Introduction

Contexte de la mission précédente

Puls-Events est une entreprise proposant une plateforme web permettant aux utilisateurs de découvrir et suivre des événements culturels en temps réel. Les données sont collectées à partir de sources externes telles qu'**OpenAgenda** et personnalisées selon les préférences des utilisateurs (localisation, période, thématiques).

Un **Proof of Concept (POC)** a été réalisé afin de démontrer la faisabilité d'un moteur de recherche sémantique intégrant un chatbot basé sur un système de **Retrieval Augmented Generation (RAG).** Ce chatbot utilise des techniques de traitement du langage naturel et une base de données vectorielle afin de fournir des recommandations pertinentes.

Objectif de la mission actuelle

L'objectif de ce projet est de transformer ce POC en un **Minimum Viable Product (MVP)** scalable, robuste et exploitable en production, tout en maîtrisant les coûts et en garantissant la qualité des réponses fournies aux utilisateurs.

2. Analyse et synthèse des besoins formulés par l'équipe

Synthèse du contexte

Ce POC, réalisé dans un environnement **Python**, repose sur un **pipeline** complet de RAG intégrant la **collecte de données, leur transformation, une recherche vectorielle et la génération de réponses** en langage naturel.

Le système permet de :

- Collecter automatiquement des événements culturels via l'API OpenAgenda ;
- Nettoyer, normaliser et structurer des données ;
- Transformer les descriptions textuelles en embeddings vectoriels ;
- Indexer ces embeddings à l'aide de FAISS pour effectuer des recherches sémantiques rapides ;
- Générer des réponses contextualisées via un modèle de langage (Mistral).

Le chatbot est capable de répondre à des requêtes variées telles que :

- des recherches par type d'événement (concert, théâtre, exposition) ;
- des recherches géographiques implicites (ex. « événements à Paris ») .

Objectifs métiers confirmés par le POC

Les résultats obtenus lors du POC ont permis de confirmer plusieurs hypothèses métiers initiales.

Tout d'abord, la recherche conversationnelle basée sur la sémantique répond efficacement au besoin des utilisateurs de **trouver rapidement des événements pertinents**, sans parcourir de longues listes ni manipuler des filtres complexes.
Ensuite, l'utilisation de données réelles issues d'OpenAgenda permet de produire des recommandations crédibles, ancrées dans un contexte réel et vérifiable.

Les objectifs métiers retenus pour la suite du projet sont donc :

- Proposer une expérience de recherche plus naturelle et intuitive que les mécanismes traditionnels ;
- Améliorer la pertinence des recommandations grâce à la compréhension du langage naturel ;
- Valoriser un volume important de données événementielles publiques ;
- Poser les bases d'un service évolutif pouvant être enrichi fonctionnellement (interface web, historique utilisateur, couverture géographique élargie).

Contraintes techniques identifiées

Le POC a également permis d'identifier des contraintes techniques structurantes pour la suite du projet.

1. Contraintes liées aux données et au NLP

Le volume potentiel des données OpenAgenda est important (plus de 15 000 événements sur une année). Afin de rester compatible avec les quotas d'API du modèle d'embeddings Mistral, le POC a volontairement limité le nombre d'événements vectorisés, tout en conservant un échantillon représentatif.

Lors de l'exécution du POC, des erreurs liées aux limites de quotas de l'API Mistral ont été rencontrées, notamment lors de la génération d'embeddings sur des volumes plus importants. Ces limitations ont confirmé la nécessité d'intégrer, dès la phase MVP, des mécanismes de contrôle des volumes, d'optimisation des appels NLP et de maîtrise des coûts.

Cette contrainte met en évidence la nécessité, pour le MVP, de prévoir des mécanismes d'optimisation des coûts et de gestion des volumes.

2. Contraintes d'architecture

Le POC repose sur une architecture locale (stockage CSV, embeddings .npy, index FAISS local) et une interface en ligne de commande.
Si cette approche est adaptée à une démonstration technique, elle n'est pas suffisante pour un usage réel. Le MVP devra intégrer :

- une architecture plus modulaire ;
- une séparation claire entre ingestion, indexation et exposition des services ;
- des mécanismes de déploiement et de supervision adaptés à un environnement de production.

3. Contraintes fonctionnelles

Certaines fonctionnalités restent limitées à ce stade :

- absence d'interface utilisateur graphique ;
- gestion temporelle encore perfectible (expressions comme « ce week-end », « fin d'année ») ;
- absence de mémoire conversationnelle persistante.

Ces limites ne remettent pas en cause la validité du POC, mais constituent des axes d'amélioration clairs pour la phase MVP.

Utilisateurs cibles et usages observés

Le POC a été principalement conçu pour valider des usages orientés **utilisateur final**, à savoir :

- la formulation de requêtes en langage naturel ;
- la recherche d'événements par thème, localisation ou période ;
- l'obtention de réponses synthétiques et contextualisées.

Analyse et choix d'une solution cloud adaptée

Le choix du cloud provider doit privilégier la **flexibilité**, la **simplicité de mise en œuvre** et les **services managés**, plutôt qu'une optimisation fine prématurée.

1. Veille sur les principales solutions du marché

Une veille comparative a été menée sur les principaux fournisseurs de cloud public (Azure, AWS et Google Cloud Platform) afin d'identifier la solution la plus adaptée au déploiement du MVP, selon des critères de simplicité, de maîtrise des coûts, de rapidité de mise en œuvre et de cohérence avec une démarche MVP.

Tableau – Comparaison des cloud providers pour le MVP RAG

Critère	Microsoft Azure	Amazon Web Services	Google Cloud Platform
Positionnement	Généraliste, orienté entreprise	Très large, très granulaire	Data & ML centric
Simplicité de mise en œuvre (MVP)	Élevée	Moyenne à faible	Moyenne
Services managés conteneurs	App Service / Container Apps simples	ECS / EKS plus complexes	Cloud Run efficace
Observabilité native	Azure Monitor intégré	CloudWatch à configurer	Cloud Monitoring
Lisibilité des coûts (faible charge)	Bonne	Plus complexe	Correcte
Adaptation à une démarche MVP	Très bonne	Moyenne	Bonne
Scalabilité long terme	Élevée	Très élevée	Élevée
Temps de delivery	Court	Plus long	Moyen

2. Choix retenu : Microsoft Azure

Le cloud provider retenu pour le MVP est **Microsoft Azure**.

AWS, bien que très complet et particulièrement adapté aux architectures complexes à grande échelle, présente une **complexité de configuration et de tarification plus élevée**, moins adaptée à un MVP à faible charge.
Google Cloud Platform offre d'excellentes capacités en data et machine learning, mais son écosystème est davantage orienté vers des usages analytiques avancés que vers un MVP applicatif complet.
Azure offre un **compromis équilibré** entre simplicité, services managés, observabilité et coûts, tout en restant compatible avec une évolution future vers des architectures plus complexes.

À l'issue de cette analyse, **Microsoft Azure** a été retenu comme cloud provider pour le MVP.
Ce choix repose non pas sur une supériorité technologique absolue, mais sur sa **meilleure adéquation avec une phase de MVP**, combinant simplicité de déploiement, services managés cohérents, observabilité intégrée et coûts maîtrisables à faible charge.

3. Positionnement par rapport à la démarche MVP

Le choix d'Azure s'inscrit dans une logique pragmatique :

- valider rapidement la **valeur métier** du produit ;
- limiter les risques techniques et financiers ;
- éviter un sur-dimensionnement de l'infrastructure.

Cette approche est cohérente avec les principes de la démarche MVP, où l'infrastructure doit **servir l'expérimentation et l'itération**, et non constituer un frein à l'évolution du projet.

3. Plan de projet

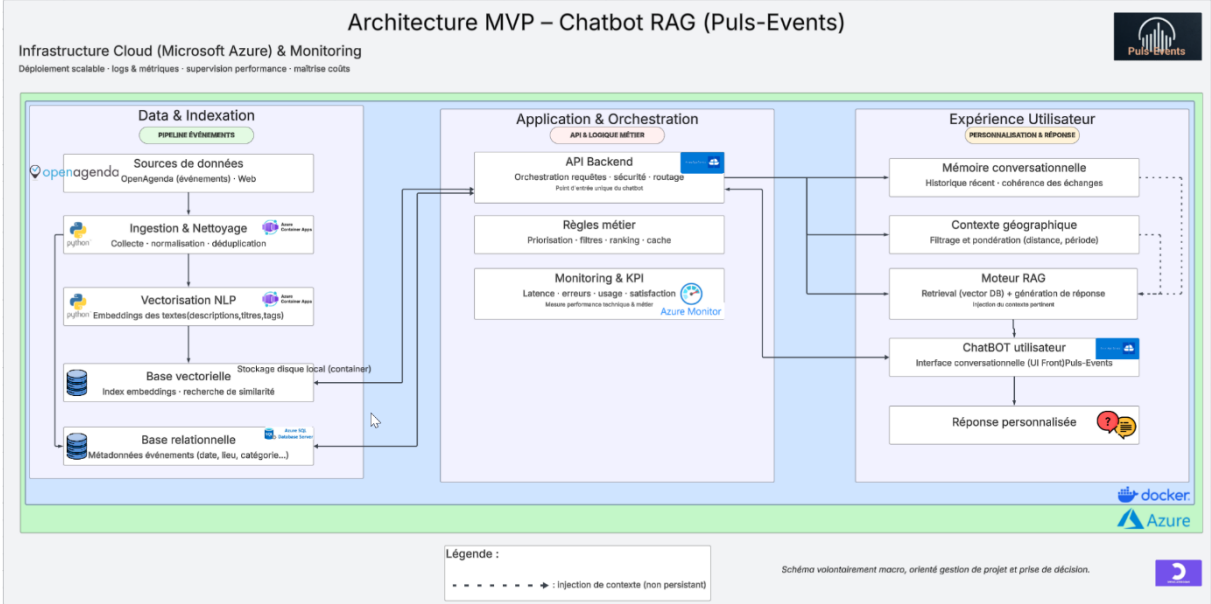
Ce plan permet un pilotage itératif du projet avec des points de validation réguliers afin d'ajuster les priorités en fonction des contraintes techniques et métiers.

4. Macro backlog des fonctionnalités

Fonctionnalité	Description	Priorité	Complexité	Risques identifiés	Mitigation
Ingestion OpenAgenda	Collecte automatisée des événements via API OpenAgenda	Must-Have	Moyenne	Volume important	Pagination, filtres
Nettoyage & normalisation	Nettoyage texte, normalisation champs, déduplication	Must-Have	Moyenne	Données hétérogènes	Règles de nettoyage
Stockage relationnel	Base relationnelle pour métadonnées (date, lieu, catégorie…)	Must-Have	Moyenne	Modèle mal adapté	Schéma simple & évolutif
Chunking des descriptions	Découpage des textes longs avec overlap	Must-Have	Faible	Perte de contexte	Overlap contrôlé
Génération embeddings	Embeddings NLP avec Mistral	Must-Have	Moyenne	Quota API	Limitation volume
Base vectorielle	Index FAISS pour recherche sémantique	Must-Have	Moyenne	Performance	Index simple
API Backend	Point d'entrée unique du chatbot	Must-Have	Élevée	Couplage fort	Architecture modulaire
Moteur RAG	Retrieval + génération de réponse	Must-Have	Élevée	Hallucinations	Contexte contrôlé
Chatbot utilisateur	Interface conversationnelle (CLI / UI simple)	Must-Have	Faible	UX limitée	MVP assumé
Règles métier	Filtres date / ville / catégorie / ranking	Nice-to-Have	Moyenne	Complexité logique	Règles simples
Contexte géographique	Pondération par distance	Nice-to-Have	Moyenne	Données imprécises	Géoloc simplifiée
Monitoring & KPI	Logs, erreurs, latence	Nice-to-Have	Faible	Manque visibilité	Logs centralisés

Les risques principaux (coûts NLP, performance, charge) ont été anticipés dès la phase de conception.

5. Architecture technique détaillée



6. Estimation des coûts Build & OPEX

Les estimations sont réalisées à un niveau macro, conformément au périmètre MVP et aux informations disponibles.

Dans l'hypothèse d'un développement réalisé par un **Data Engineer freelance**, le coût de BUILD du MVP est estimé à partir d'un **taux journalier moyen (TJM) de 500 €**, correspondant à un profil junior–confirmé sur le marché français.
Les charges de développement sont évaluées sur la base du **macro-backlog du projet**, en jours de travail, et visent à couvrir l'ensemble des activités nécessaires à la livraison d'un MVP fonctionnel.

Coûts BUILD (développement initial)

Poste de coût	Description	Nature du coût	Charge estimée	Coût
Cadrage & conception	Analyse POC, définition MVP, architecture cible	Ponctuel	3 jours	1 500 €
Pipeline data	Ingestion OpenAgenda, nettoyage, normalisation	Ponctuel	5 jours	2 500 €
Base relationnelle	Modélisation et mise en place des métadonnées	Ponctuel	3 jours	1 500 €
Vectorisation NLP	Génération des embeddings et indexation	Ponctuel	3 jours	1 500 €
Base vectorielle	Mise en place FAISS et tests de performance	Ponctuel	2 jours	1 000 €
API Backend	Orchestration, sécurité, règles métier	Ponctuel	6 joiurs	3 000 €
Moteur RAG	Retrieval + génération contrôlée	Ponctuel	5 jours	2 500 €
Interface chatbot	Interface utilisateur simple (CLI / UI web)	Ponctuel	2 jours	1 000 €
Tests & validation	Tests fonctionnels et cas d'usage	Ponctuel	2 jours	1 000 €
Documentation	Documentation technique & projet	Ponctuel	2 jours	1 000 €
Total			33 jours	16 500 €

Sur la base de ces estimations, Le coût total de BUILD est ainsi estimé à environ 16 500 €, correspondant à un investissement initial.

Coûts OPEX (exploitation)

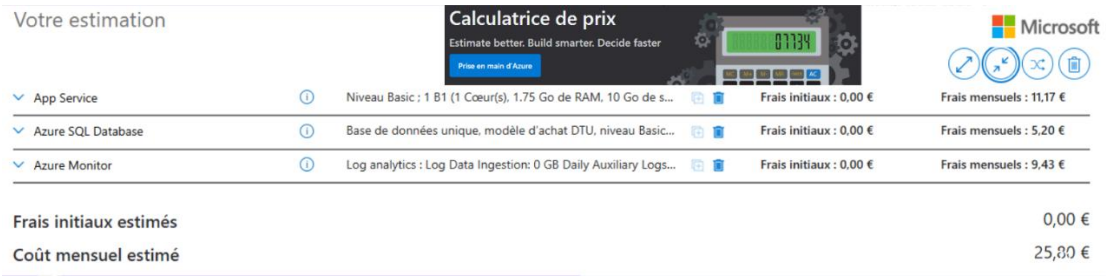
Les coûts d'exploitation (OPEX) correspondent aux **charges récurrentes** nécessaires au fonctionnement du MVP une fois déployé en environnement cloud.
Les estimations présentées ci-dessous sont établies sur la base d'un **usage mensuel modéré**, cohérent avec une phase MVP, et reposent sur les **tarifs publics des services cloud et des modèles de langage utilisés**.

Hypothèses d'usage

- 1 000 utilisateurs / mois
- 5 requêtes chatbot / utilisateur
- 5 000 requêtes RAG / mois
- 50 000 événements OpenAgenda

[Outil de calcul de tarification | Microsoft Azure]

Estimation globale des services Azure :



Estimation détaillée des services Azure :

Microsoft Azure Estimate

Votre estimation

Service category	Service type	Custom name	Region	Description	Estimated monthly cost	Estimated upfront cost
Calcul	App Service		France Central	Niveau Basic : 1 B1 (1 Cœur(s), 1.75 Go de RAM, 10 Go de stockage) x 730 Heures ; Système d'exploitation Linux ; 0 SNI/SSL Connexions ; 0 SSL IP Connexions ; 0 Domaines personnalisés ; 0 Certificats SSL standards ; 0 Certificats SSL génériques	€11.17	€0.00
Bases de données	Azure SQL Database		France Central	Base de données unique, modèle d'achat DTU, niveau Basic, B : 5 DTU, 2 Go d'espace de stockage inclus par base de données, 1 bases de donnée(s) x 730 Heures, 0 Go de stockage, Redondance du stockage de sauvegarde RA-GRS, 0 x 5 Go rétention à long terme	€5.20	€0.00
DevOps	Azure Monitor		France Central	Log analytics : Log Data Ingestion: 0 GB Daily Auxiliary Logs without processing, 2 GB Daily Auxiliary Logs with processing, 0 GB Daily Basic logs, 0 GB Daily Analytics logs ingested, 1 months of Interactive Retention, 0 months of Retention, 0 GB data restored for 0 days, 0 queries per day with 0 GB data scanned per query, 0 GB of Log Data Exported per day, Platform Log Data Processed per day: 0 GB with Destination to Storage or Event Hub and 0 GB with Destination to Marketplace Partners, 0 Search job Queries per day with 0 GB data scanned per query : 0 points de terminaison SCOM MI ; Prometheus managé : utilisation de la méthode d'estimation de collecte par défaut (avec un cluster de 0 nœuds Linux, 0nœuds Windows, 0 containers et 0 pods), 0 nombre moyen d'utilisateurs quotidiens de tableaux de bord, 7 tableaux de bord, 50000 échantillons de données interrogés par tableau de bord, 25 règles d'alerte promql, 25 règles d'enregistrement promql ; Application Insights : 0 Go de journaux analytiques quotidiens ingérés, 3 mois de conservation des données, 0 Tests Web Standard, fréquence d'exécution 5 minutes, exécution pendant 730 heures ; 0 ressources surveillées X 1 série chronologique d'indicateur de performance surveillée par ressource, Fréquence du signal de journal 5 minutes avec 0 signaux de journal surveillée et 1 séries chronologiques par signal	€9.43	€0.00
Support			Support		€0.00	€0.00
			Licensing Program	Microsoft Customer Agreement (MCA)		
			Billing Account			
			Billing Profile			
			Total		€25.80	€0.00

Disclaimer

All prices shown are in Euro Zone – Euro (€) EUR. This is a summary estimate, not a quote. For up to date pricing information please visit https://azure.microsoft.com/pricing/calculator/
This estimate was created at 1/26/2026 6:20:08 PM UTC.

Poste de coût	Détail du périmètre couvert	Hypothèses retenues	Type de coût	Coût mensuel estimé
Hébergement cloud	Hébergement de l'API Backend, du moteur RAG et de l'interface utilisateur via des conteneurs (Azure App Service / Container Apps)	Charge faible à modérée (phase MVP, trafic limité)	Récurrent	12 €
Stockage relationnel	Base de données pour les métadonnées structurées des événements (dates, lieux, catégories, descriptions courtes)	Volume < 1 Go, faible nombre d'écritures	Récurrent	6 €
Stockage vectoriel	Stockage local de l'index FAISS et des embeddings vectoriels	FAISS embarqué, stockage disque inclus dans l'hébergement	Récurrent	0 €
Appels NLP / LLM	Génération des réponses du chatbot (RAG) et appels aux modèles de langage	~5 000 requêtes/mois, ~1 000 tokens par requête	Variable	60 €
Réseau	Trafic entrant/sortant, appels API externes (OpenAgenda, LLM)	Volumes faibles, inclus dans l'offre cloud	Récurrent	Inclus
Monitoring & logs	Collecte des métriques de performance, logs applicatifs, erreurs et supervision	Azure Monitor / Application Insights, faible volumétrie	Récurrent	10 €
Maintenance opérationnelle	Ajustements mineurs, surveillance, correctifs légers	Maintenance minimale en phase MVP	Récurrent	Inclus
Total				88 €

L'OPEX correspond aux **coûts récurrents** liés à l'exploitation du MVP une fois déployé.

Les coûts OPEX sont ainsi estimés à environ 85-90€ par mois, un niveau compatible avec une exploitation MVP et une montée en charge progressive.

Optimisation budgétaire

Levier	Description	Effet
Limitation du périmètre MVP	Fonctionnalités essentielles uniquement	Réduction BUILD
Réduction volume embeddings	Vectorisation ciblée	Réduction OPEX
Cache des réponses	Réutilisation résultats fréquents	Réduction appels LLM
Batch processing	Traitements groupés	Réduction coûts NLP
Montée en charge progressive	Ajustement à l'usage réel	Maîtrise OPEX
Architecture modulaire	Remplacement facile des briques	Pérennité

7. Bilan

7.1 Du POC au MVP

Le projet a débuté par la réalisation d'un **POC** visant à valider la faisabilité d'un chatbot de recommandation d'événements basé sur une approche **RAG**.
Ce POC a permis de confirmer la pertinence de la recherche sémantique et de la génération de réponses à partir de données réelles.

Sur cette base, le projet a évolué vers un **MVP**, en suivant une démarche structurée : analyse des limites du POC, définition d'un périmètre fonctionnel réaliste, conception d'une architecture cible et priorisation des fonctionnalités.
Cette approche progressive a permis de transformer une preuve de concept technique en une solution exploitable et orientée usage.

7.2 Choix techniques et méthodologiques

Les choix techniques reposent directement sur les enseignements du POC.
L'architecture **RAG** a été retenue pour combiner recherche sémantique et génération de réponses tout en limitant les hallucinations. La séparation entre **base relationnelle** et **base vectorielle** permet de distinguer clairement les données structurées et la recherche par similarité.

Le recours à une **API Backend centrale** facilite l'orchestration, la maintenance et l'évolution du système.
D'un point de vue méthodologique, l'approche **MVP** a été privilégiée afin de maîtriser la complexité, les coûts et les risques, tout en préparant les évolutions futures.

7.3 Défis et solutions

Les principaux défis concernent la gestion des volumes de données, les coûts liés aux modèles de langage et la qualité des réponses générées.

Ces enjeux ont été traités par une limitation du périmètre MVP, une vectorisation ciblée, l'usage de règles métier et une architecture pensée dès l'origine pour être **scalable, modulaire et observable**.

8. Annexes

A. Portfolio professionnel

Le portfolio professionnel est accessible via GitHub et regroupe l'ensemble des projets réalisés dans le cadre du parcours Data Engineer.

Lien vers le portfolio GitHub : *Portfolio of Antoine Kelleni*

B. Liens externes et références utiles

- Documentation OpenAgenda (API et données événementielles) : Documentation OpenAgenda
- Documentation des outils de vectorisation et de bases vectorielles :

  FAISS – Facebook AI Similarity Search (Meta AI)
  https://github.com/facebookresearch/faiss

- Azure App Service
  https://azure.microsoft.com/pricing/details/app-service/
- Azure SQL Database
  https://azure.microsoft.com/pricing/details/azure-sql-database/
- Azure Monitor – Tarification
  https://azure.microsoft.com/pricing/details/monitor/
- Mistral AI – Tarification API
  https://docs.mistral.ai/pricing/