

RAPPORT DE PROJET DE FIN D'ÉTUDES

Présenté en vue de l'obtention du
Diplôme National d'ingénieur
Spécialité : Ingénierie du Développement du Logiciel(IDL)

Par

Ons CHAHED

Conception et développement d'une plateforme analytique pour les métadonnées de la plateforme Snowflake

Encadrant professionnel : **Monsieur Nassim JALLOUD**

Team leader Dev/Data

Encadrant académique : **Monsieur Sahbi BAHROUN**

Maître Assistant

Réalisé au sein de Avaxia Group



RAPPORT DE PROJET DE FIN D'ANNÉE

Présenté en vue de la validation de
Diplôme National d'ingénieur
Spécialité : Ingénierie du Développement du Logiciel(IDL)

Par

Ons CHAHED

Conception et développement d'une plateforme analytique pour les métadonnées de la plateforme Snowflake

Encadrant professionnel : **Monsieur Nassim JALLOUD**

Team leader Dev/Data

Encadrant académique : **Monsieur Sahbi BAHROUN**

Maître Assistant

Réalisé au sein de Avaxia Group



J'autorise l'étudiant à faire le dépôt de son rapport de stage en vue de la validation du stage d'été.

Encadrant professionnel, **Monsieur Nassim JALLOUD**

Signature et cachet

J'autorise l'étudiant à faire le dépôt de son rapport de stage en vue de la validation du stage d'été.

Encadrant académique, **Monsieur Sahbi BAHROUN**

Signature

Dédicaces

À mes chers parents, la lumière de ma vie,

Puisque certaines dettes sont difficiles à rembourser, peu importe le temps que cela prend, je dédie ce travail signe de reconnaissance et de dévouement à mes chers parents qui m'ont donné la vie et la tendresse. Leur joie n'est que le succès de leurs enfants. Leur amour, leur soutien et leurs sacrifices ont abordé toutes les limites. J'espère avoir répondu même partiellement aux espoirs que vous avez fondés en moi.

À Mes chères grands parents,,

Ceci est ma profonde gratitude pour votre éternel amour, que ce rapport soit le meilleur cadeau que je puisse vous offrir.

À mon chére frère,

Tu as été à mes côtés pendant toutes les étapes de ce travail, je t'en suis très reconnaissante. Aucune dédicace ne peut exprimer la profondeur des sentiments fraternel, d'amour et d'attachement que j'éprouve à ton égard. Je vous dédie ce travail en témoignage de ma profonde affection.

À mes meilleurs amis,

Parfois, j'ai oublié de remercier les personnes qui font ma vie si merveilleuse à bien des égards. Aujourd'hui c'est le jour où j'aimerais leur dire que je vous remercie pour être là, à mes côtés durant cette période critique. Vous étiez ma source de motivation et je suis et je serais toujours fier de notre amitié ! Je suis impatiente de partager encore beaucoup d'autres moments fantastiques avec vous.

Ons CHAHED

Remerciement

Après avoir rendu grâce à Dieu tout puissant et miséricordieux, je tenu à remercier vivement tous ceux qui, de près ou de loin ont participé à l'achèvement du projet ou à la rédaction de ce document.

Je tiens tout particulièrement à remercier **Monsieur Nassim Jallooud**, mon encadrant professionnelle pour son accueil de joindre son personnel durant ce stage et de m'offrir l'opportunité de faire ce travail. Pour sa disponibilité, sa compréhensibilité, sa rigueur scientifique et son sens d'écoute et d'échange.

Je remercie **Monsieur Nassim Jallooud**, mon encadrant académique pour le temps consacré, les astuces et les conseils précieuses prodiguées durant ces années d'étude de cycle ingénieur.

Enfin, je n'oublie pas tous mes enseignants de l'**ISI** qui ont contribué à ma formation, qu'ils trouvent ici toute ma gratitude.

Ons CHAHED

Table des matières

Introduction générale	1
1 Cadre du projet	2
Introduction	3
1.1 Context général du projet	3
1.2 Organisme d'accueil	3
1.2.1 Présentation générale de Avaxia Group	3
1.2.2 Les services de Avaxia Group	3
1.3 Étude du projet	4
1.3.1 Problématique	5
1.3.2 Analyse et critique de l'existant	5
1.3.3 Solution proposée	9
1.4 Choix méthodologique	10
Conclusion	10
2 Analyse et spécification des besoins	11
Introduction	12
2.1 Identification des besoins	12
2.1.1 Besoins fonctionnels	12
2.1.2 Besoins non fonctionnels	13
2.2 Flux de travail	14
2.3 Le backlog du produit	16
Conclusion	17
3 Architecture et conception	18
Introduction	19
3.1 Étude architecturale	19
3.1.1 Architecture logique	19
3.1.2 Architecture physique	24
3.2 Étude conceptuelle	26

3.2.1	Conception globale	26
3.2.2	Conception detaillée	30
	Conclusion	41
4	Réalisation	42
	Introduction	43
4.1	Choix technologiques	43
4.1.1	Frameworks	43
4.1.2	Bases de données	43
4.1.3	Outils de communication	44
4.1.4	Bibliothèques pour l'analyse et la manipulation des données	45
4.1.5	Bibliothèques d'interaction avec le système d'exploitation	45
4.1.6	Outils de test	46
4.1.7	Outils de versionning	47
4.1.8	Outils de documentation	47
4.2	Réalisation	48
4.2.1	Micro-service « Authentification_Service »	48
4.2.2	Micro-service « Warehouse_Monitoring »	50
4.2.3	Micro-service « Query_Performance »	54
4.2.4	Micro-service « Databases_Statistics »	59
4.2.5	Micro-service « Access_Statistics »	63
4.2.6	Micro-service « DAG_Monitoring »	66
	Conclusion	68
	Conclusion générale	69
	Bibliographie	71

Table des figures

1.1	Snowflake account usage dashboard	6
1.2	Snowflake account usage dashboard	7
1.3	Tableau de bord de «Google BigQuery»	7
1.4	Tableau de bord de "Amazon Redshift"	8
2.1	Processus métier du projet	14
3.1	Architecture logique de «Snowflake Monitoring Application»	21
3.2	Architecture physique de «Snowflake Monitoring Application»	25
3.3	Diagramme de paquetages de «Snowflake Monitoring Application»	27
3.4	Diagramme de classe de «Snowflake Monitoring Application»	28
3.5	Diagramme de cas d'utilisation globale «Snowflake Monitoring Application»	29
3.6	Diagramme d'activité du micro-service «ETL-service»	33
3.7	Diagrammme de séquence de cas d'utilisation «Visualiser DAG»	35
3.8	Diagrammme de séquence de cas d'utilisation «Lister les détaills d'une tâche»	37
3.9	Diagrammme de séquence de cas d'utilisation «Créer un compte»	39
3.10	Diagrammme de séquence de cas d'utilisation «S'authentifier»	40
4.1	FastAPI logo [18]	43
4.2	React logo [19]	43
4.3	PostgreSQL logo [20]	44
4.4	MQTT logo [21]	44
4.5	REST logo [22]	44
4.6	Snowflake logo [23]	45
4.7	Psycopg logo [24]	45
4.8	Pandas logo [25]	45
4.9	Postman logo [30]	46
4.10	MQTTX logo [31]	46
4.11	Git logo [32]	47
4.12	Github logo [33]	47

4.13 Swagger logo [34]	47
4.14 Latex logo [35]	47
4.15 Draw.io logo [36]	48
4.16 Liste des APIs du microservice «Athentification_service»	48
4.17 Interface de Création de compte «Sign up»	49
4.18 Interface d'authentification «Login»	50
4.19 «Logout»	50
4.20 Liste des APIs du microservice «Warehouse_Monitoring»	51
4.21 Interface de la liste des entrepôts de données	52
4.22 Tableau de bord du surveillance d l'entrepôts des données «Monitoring_warehouse»	52
4.23 Tableau de bord du surveillance d l'entrepôts des données «Compute_WH»	53
4.24 Changement du role de l'utilisateur de «ACCOUNT ADMIN» vers «ORGADMIN»	53
4.25 Changement de seuil de credits vers «90\$»	54
4.26 Liste des APIs du microservice «Warehouse_Monitoring»	55
4.27 Interface de la liste des entrepôts de données	56
4.28 Tableau de bord du surveillance des entrepôts des données	57
4.29 Tableau de bord du surveillance des entrepôts des données	57
4.30 Tableau de bord du surveillance de l'entrepôt des données «Monitoring_Warehouse»	58
4.31 Tableau de bord du surveillance de l'entrepôt des données «Compute_WH»	58
4.32 Liste des APIs du microservice« Databases_Statistics »	60
4.33 La liste des bases de données	61
4.34 Liste des schémas de base de données	61
4.35 Liste des objets dans la base de données	62
4.36 Tableaux de bord du surveillance des bases de données	62
4.37 Liste des APIs du microservice « Access_Statistics »	63
4.38 Historique des accées	64
4.39 Tableau de bord du surveillance des accéess de l'utilisateur «ONS CHAHED»	65
4.40 Tableau de bord du surveillance des accéess de l'utilisateur «SNOWFLAKE»	65
4.41 Résultat du command «Show tasks»	66
4.42 Etat initial du DAG	67
4.43 Lancement des tâches	67

Liste des tableaux

2.1	Backlog de Produit	17
3.1	description textuelle de cas d'utilisation «Lister les entrepôts de données»	30
3.2	description textuelle de cas d'utilisation «Visualiser DAG»	31
3.3	description textuelle de cas d'utilisation «Visualiser DAG»	32

Liste des abréviations

- **API** = Application Programming Interface
- **CQRS** = Command Query Responsibility Segregation
- **DAG** = Directed Acyclic Graph
- **DAO** = Data Access Object
- **DFS** = Depth First Search
- **DOM** = Domain Object Model
- **EMQ** = Erlang MQTT Broker
- **ETL** = Extract Transform Load
- **JSON** = JavaScript Object Notation
- **MQTT** = Message Queuing Telemetry Transport
- **REST** = REpresentational State Transfer
- **SGBD** = Système de Gestion de Base de Données
- **SQL** = Structured Query Language
- **SRP** = Single Responsibility Principle

— URL = Uniform Resource Locator

Introduction générale

Dans un paysage technologique en constante évolution, la gestion efficace des données est devenue un impératif pour la compétitivité des entreprises. Avec l'avènement des plateformes de data warehousing cloud telles que Snowflake, les entreprises bénéficient désormais d'outils puissants pour gérer, stocker et analyser leurs données à grande échelle, ouvrant ainsi de nouvelles perspectives de croissance et d'innovation.

Cependant, exploiter pleinement les capacités de Snowflake requiert une surveillance constante et une optimisation proactive des opérations effectuées sur cette plateforme. C'est dans ce contexte exigeant que s'inscrit le projet de surveillance des opérations Snowflake.

L'objectif principal de cette initiative est de fournir à Avaxia Group, ainsi qu'à ses clients utilisant Snowflake, une solution exhaustive de surveillance et d'analyse des opérations sur la plateforme. Concrètement, cette solution permettra non seulement de surveiller en temps réel les performances des entrepôts de données, mais également d'analyser de manière approfondie les métriques clés telles que la charge de travail, les temps de réponse et la disponibilité des ressources. En identifiant rapidement les goulets d'étranglement, les tendances émergentes et les variations de charge, elle offrira la possibilité d'anticiper les défis potentiels et de prendre des mesures correctives proactives pour optimiser l'utilisation de Snowflake.

Grâce à cette approche proactive de surveillance et d'optimisation, Avaxia Group vise à renforcer la compétitivité de ses clients en assurant une exploitation optimale de Snowflake et en garantissant une valeur ajoutée maximale à leurs activités.

En outre, cette solution inclura des fonctionnalités avancées telles que des alertes en temps réel, des tableaux de bord personnalisables et des analyses prédictives pour aider les équipes opérationnelles à prendre des décisions éclairées et à maximiser leur efficacité.

CHAPITRE 1

CADRE DU PROJET

Plan

Introduction	3
1 Context général du projet	3
2 Organisme d'accueil	3
3 Étude du projet	4
4 Choix méthodologique	10
Conclusion	10

Introduction

Dans ce premier chapitre, nous plongeons dans le contexte global de notre projet au sein d'Avaxia Group. Cette section établit les bases en présentant le cadre général du projet, en soulignant les objectifs que nous nous efforçons d'atteindre.

1.1 Context général du projet

Ce projet s'inscrit dans le cadre du stage de fin d'études d'ingénieur au sein de la société Avaxia Group, se déroulant de Février 2024 à Mai 2024, dans l'objectif de l'obtention de diplôme national d'ingénieur de l'Institut Supérieur d'Informatique (ISI). Notre mission centrale consiste à contribuer à la réalisation du projet intitulé ******, en apportant notre expertise et notre contribution essentielle

1.2 Organisme d'accueil

1.2.1 Présentation générale de Avaxia Group

Avaxia Group, une entreprise de conseil en technologie créée en 1998 et ayant son siège à Dubaï, étend actuellement sa présence mondiale avec des bureaux au Japon et en Tunisie, et de futurs emplacements prévus au Canada, en France et à Dakar. Avaxia Group se spécialise dans les solutions middleware, la gestion des infrastructures système, ainsi que dans le conseil fonctionnel, couvrant des domaines tels que l'architecture, l'intégration et les opérations [1].

1.2.2 Les services de Avaxia Group

Avaxia Group opère sur le vaste marché international, avec une présence dans de nombreux pays. L'entreprise propose une gamme diversifiée de services, notamment les suivants[2] :

- **Expertise Technique en SAP :**

- Gestion de l'infrastructure.
- Optimisation et amélioration des performances.
- Conception et architecture du paysage SAP.
- Installation des systèmes, configuration et mises à niveau.
- Assistance technique 24/7.

• **Conseil Fonctionnel :**

- Gestion fonctionnelle du déploiement et planification des tests.
- Conception de solutions métier pour le déploiement de nouveaux modules SAP.

• **Solutions Logicielles Personnalisées :**

- Conception et développement de solutions logicielles utilisant des technologies telles que Java J2E, DELL BOOMI Flow, Salesforce et SAP Fiori.

• **Intégration des Systèmes :**

- Intégration de données et gestion des flux de données.
- Intégration d'applications métier.
- Développement et gestion d'API : DELL BOOMI, Atmosphère.

• **Ingénierie des Données :**

- Modélisation, découverte, traitement, visualisation et exploration des données, ainsi que gestion du Big Data.

• **Nouvelles Technologies :**

Création et déploiement de solutions logicielles d'entreprise en utilisant les dernières technologies, notamment :

- Apprentissage automatique (Machine Learning).
- Réalité virtuelle (VR).
- Réalité augmentée (AR).
- Internet des objets (IoT).

Cette gamme diversifiée de services reflète l'engagement d'Avaxia Group à fournir des solutions techniques avancées et à rester à la pointe de l'innovation pour répondre aux besoins variés de ses clients à l'échelle mondiale.

1.3 Étude du projet

Dans cette section nous explorons en détail l'étude de ce projet toutes en commençant par les raisons fondamentales ayant motivé le développement de notre solution et les spécifications de cette dernière ainsi que son objectif.

1.3.1 Problématique

Avec l'avènement des technologies de cloud computing et des entrepôts de données modernes tels que Snowflake, les entreprises disposent désormais d'une flexibilité et d'une évolutivité inégalées pour la gestion et l'analyse de leurs données. Cependant, cette transition vers le cloud présente des défis spécifiques en termes de performance, de coûts, et de surveillance des opérations.

Dans ce contexte, Avaxia Group se trouve confrontée à des problématiques particulières liées à l'utilisation de Snowflake, son principal entrepôt de données telque :

- **Gestion des performances :**

- Avaxia Group constate des délais dans l'exécution des requêtes et des temps de traitement prolongés lors de l'utilisation de Snowflake. Ces problèmes impactent directement la productivité des équipes et la satisfaction des utilisateurs finaux.
- Les requêtes SQL complexes ou mal optimisées peuvent engendrer des goulets d'étranglement et des inefficacités dans l'utilisation des ressources de Snowflake, nécessitant une surveillance et une optimisation constantes.

- **Maîtrise des coûts :**

- Les coûts associés à l'utilisation de Snowflake peuvent rapidement augmenter en raison d'une gestion inadéquate des ressources. Avaxia Group doit donc trouver des moyens de maîtriser ces coûts tout en garantissant des performances optimales pour ses opérations.

- **Surveillance et analyse des opération :**

- Pour assurer le bon fonctionnement de ses opérations Snowflake, Avaxia Group a besoin d'une solution de surveillance avancée pour détecter les anomalies, identifier les goulets d'étranglement et optimiser les performances.

De plus, une analyse approfondie des métriques opérationnelles telles que le temps d'exécution des requêtes, l'utilisation des ressources et les performances des entrepôts est essentielle pour optimiser l'efficacité et l'efficience des opérations.

1.3.2 Analyse et critique de l'existant

Dans cette section, nous passerons en revue les fonctionnalités, les avantages et les limitations de chaque solution, en mettant en lumière les lacunes qui ont conduit au développement de notre propre solution de monitoring des opérations Snowflake chez Avaxia Group.

1.3.2.1 Analyse de l'existant

L'analyse de l'existant est une étape cruciale dans le processus de développement de notre solution de monitoring des opérations sur Snowflake. C'est pour cela, nous allons entamer l'analyse les outils existants utilisés pour surveiller et analyser les opérations sur les divers plateformes de l'analyse des données et le data warehousing cloud

- **Snowflake account usage dashboard** : est un outil intégré qui fournit des informations sur l'utilisation de Snowflake, telles que le nombre de sessions utilisateur, la quantité de données stockées et le temps CPU utilisé. Il offre une vue rétrospective des opérations passées, permettant aux utilisateurs de comprendre l'utilisation historique de la plateforme.

La figure 1.1 suivante illustre une partie de cette dashboard :

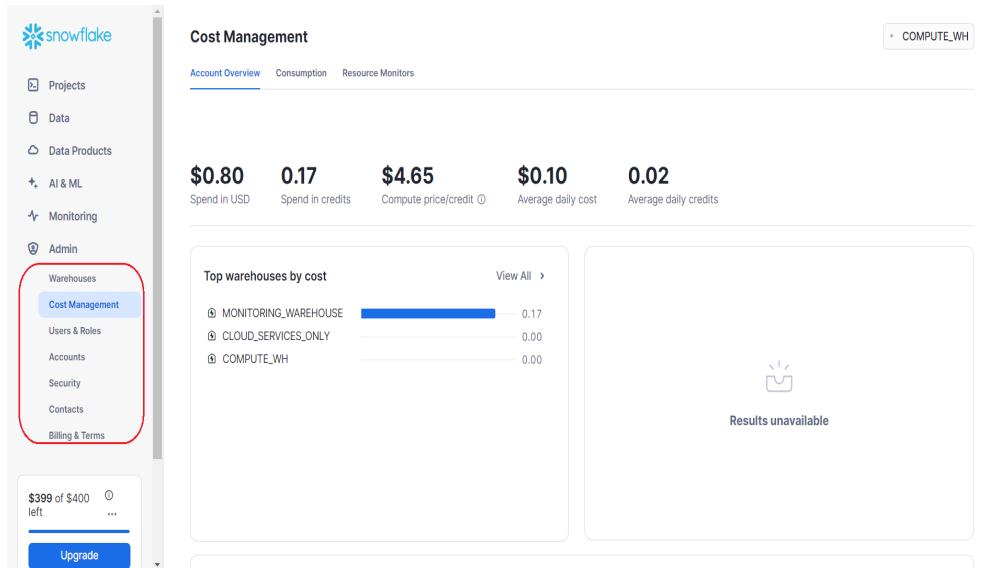


FIGURE 1.1 : Snowflake account usage dashboard

- **Snowflake Information Schema** : est une collection de vues système qui fournissent des métadonnées sur les objets et les opérations effectuées dans un compte Snowflake. Ces vues offrent une granularité élevée pour examiner les détails des requêtes SQL exécutées, les performances des entrepôts de données et d'autres aspects des opérations Snowflake.

La figure 1.2 suivante illustre une partie de cette dashboard :

Chapitre 1. Cadre du projet

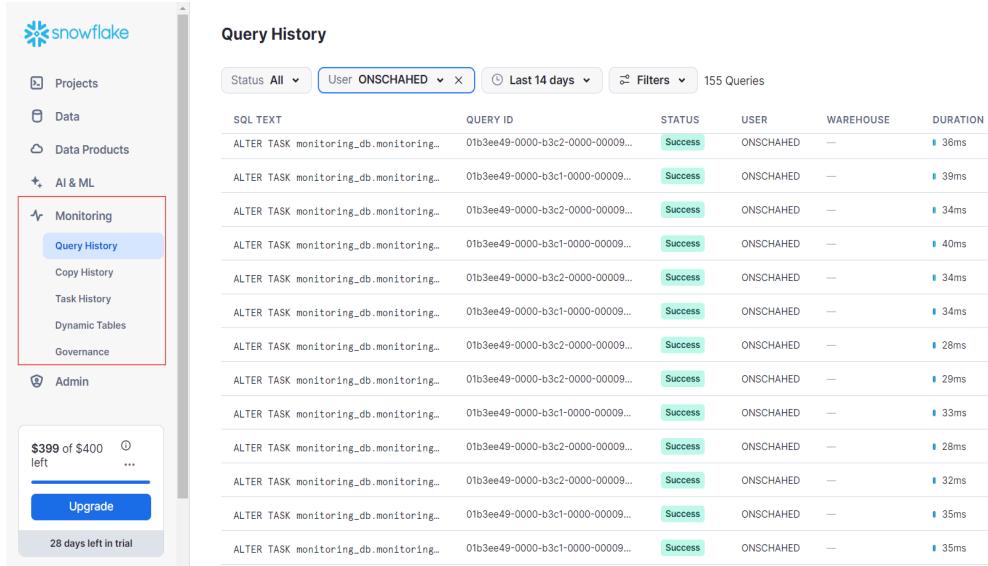


FIGURE 1.2 : Snowflake account usage dashboard

- **Google BigQuery** : est une autre option populaire pour le stockage et l'analyse des données dans le cloud. Il offre des fonctionnalités avancées telles que le traitement massivement parallèle et la capacité à exécuter des requêtes SQL complexes sur de grands ensembles de données.

La figure 1.3 suivante illustre une partie de tableau de bord de BigQuery :

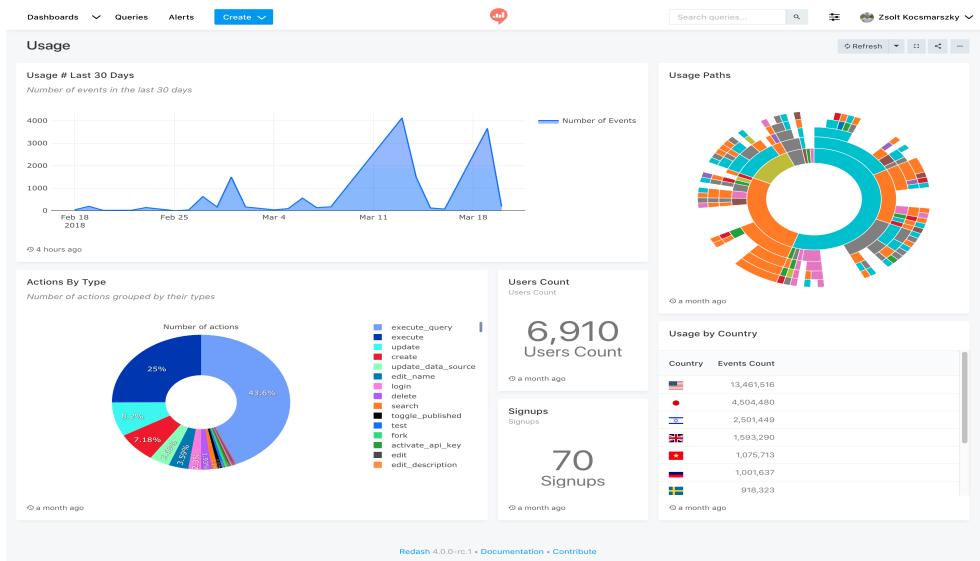


FIGURE 1.3 : Tableau de bord de «Google BigQuery»

- **Amazon Redshift** : est un entrepôt de données cloud basé sur PostgreSQL, conçu pour gérer de gros volumes de données et exécuter des analyses complexes. Il offre des performances élevées et une extensibilité, mais son modèle de tarification basé sur l'utilisation des ressources peut entraîner des coûts supplémentaires pour les entreprises

La figure 1.4 suivante illustre le tableau de bord de ResShift :

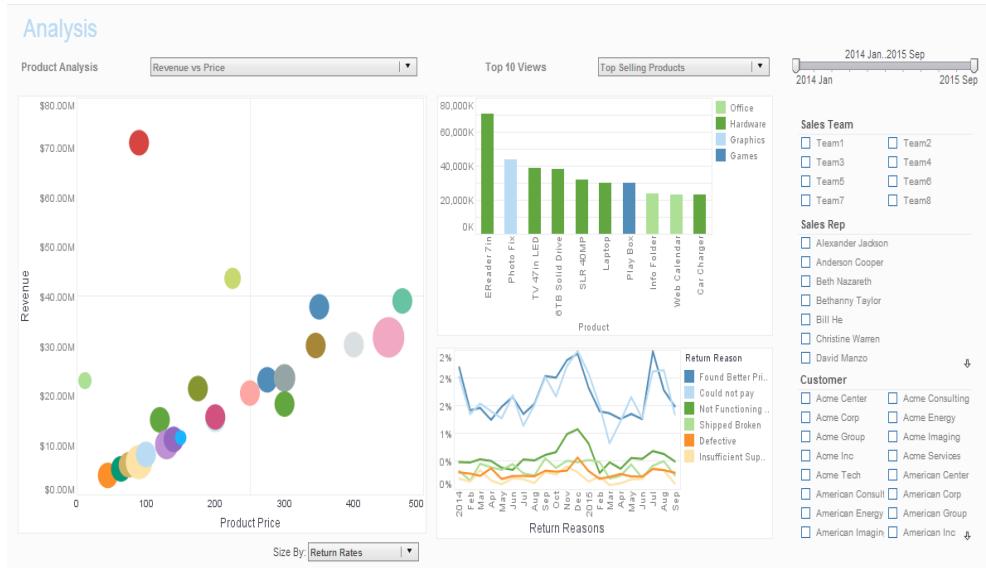


FIGURE 1.4 : Tableau de bord de "Amazon Redshift"

1.3.2.2 Critique de l'existant

Dans cette section, nous allons examiner de manière critique les outils actuellement utilisés dans le domaine de l'analyse de données, afin de fournir une base solide pour concevoir une solution qui surmonte les limitations et offre une valeur ajoutée significative à Avaxia Group et à ses clients.

- **Manque d'analyse prédictive :** l'un des principaux défauts des outils actuellement disponibles est leur incapacité à fournir une analyse prédictive des performances de Snowflake. Les tableaux de bord existants, tels que le Snowflake Account Usage Dashboard, offrent une vue rétrospective des opérations passées, mais ne fournissent pas d'informations sur les tendances futures ou les possibles goulots d'étranglement.
- **Complexité des outils :** les outils de surveillance existants, comme le Snowflake Information Schema, sont souvent complexes à utiliser et nécessitent des compétences techniques avancées pour interpréter les données fournies. Cette complexité peut rendre difficile la compréhension des métriques et la prise de décisions éclairées par les équipes opérationnelles. (par exemple dans un workflow si une certaine tache est échouée, les indicateurs disponibles sur snowflake ne peuvent pas identifier ou exactement le workflow est suspendu de façon à rendre les choses plus compliqué pour les utilisateurs de Snowflake de détecter les anomalies.)
- **Personnalisation limitée :** les options de personnalisation offertes par les outils existants sont souvent limitées. Par exemple, Google BigQuery propose des fonctionnalités avancées,

mais la personnalisation des tableaux de bord et des rapports est restreinte. Cela peut être un obstacle pour les entreprises ayant des besoins spécifiques en matière de surveillance et d'analyse des performances.

- **Coûts supplémentaires :** enfin, certains outils, comme Amazon Redshift, peuvent entraîner des coûts supplémentaires importants pour les entreprises. La tarification basée sur l'utilisation des ressources peut rapidement augmenter, surtout si les entreprises ne surveillent pas activement leur utilisation. Cela peut constituer une barrière financière pour les petites et moyennes entreprises souhaitant utiliser ces outils de surveillance.

1.3.3 Solution proposée

Notre solution pour résoudre les défis auxquels Avaxia Group est confronté dans l'utilisation de Snowflake repose sur la conception et l'implémentation d'un système de surveillance et d'optimisation des opérations exhaustif. Cette solution se décompose en plusieurs composants clés, chacun ciblant des aspects spécifiques des problématiques identifiées :

- Développement d'algorithmes d'optimisation des requêtes SQL afin de réduire les temps d'exécution et de minimiser les goulets d'étranglement dans l'utilisation des ressources de Snowflake.
- Mise en place de stratégies de monitoring en temps réel cela permettra d'identifier rapidement les problèmes de performance et de prendre des mesures correctives efficaces.
- Développement d'outils d'analyse avancée des coûts Cette analyse approfondie aidera nos utilisateurs finaux à mieux comprendre ses dépenses et à trouver des opportunités d'optimisation pour réduire les coûts tout en maintenant des performances élevées.
- la mise en place de politiques de gestion des ressources pour optimiser l'utilisation des ressources de Snowflake. Des stratégies telles que le dimensionnement automatique des entrepôts de données et l'optimisation de l'utilisation des crédits de calcul seront explorées.
- Développement des tableaux de bord interactifs et riches en informations pour permettre à Avaxia Group et à ses clients de surveiller en temps réel les performances de leurs opérations Snowflake. Ces tableaux de bord fourniront des indicateurs clés de performance, des graphiques et des visualisations pour faciliter la détection des anomalies et la prise de décisions éclairées.
- Utilisation de techniques d'analyse avancée des données opérationnelles pour permettre aux utilisateurs d'identifier les opportunités d'amélioration et d'optimisation de leurs opérations,

renforçant ainsi leur compétitivité et leur efficacité.

1.4 Choix méthodologique

Le choix de la méthode Kanban pour la gestion de notre projet est le fruit d'une réflexion stratégique minutieuse, loin d'être aléatoire. Kanban, qui tire son origine du terme japonais signifiant "tableau", se démarque par sa flexibilité, sa transparence et sa focalisation sur l'amélioration constante.

Dans le cadre de notre projet, Kanban s'impose naturellement comme le choix optimal. Cette méthode permet une gestion visuelle et en temps réel des tâches et des flux de travail, une caractéristique cruciale pour un projet centré sur l'analyse de données et la visualisation. Chaque étape de notre processus, de la collecte des données à la présentation des résultats, trouve une représentation claire et concise dans le tableau Kanban.

De surcroît, Kanban promeut une approche itérative et progressive, parfaitement en harmonie avec notre objectif d'amélioration continue. Elle autorise une gestion fluide et adaptable des tâches, offrant la souplesse nécessaire pour ajuster notre plan en fonction des découvertes et des besoins changeants du projet.

En optant pour Kanban, nous mettons l'accent sur la transparence et la communication au sein de l'équipe. Chacun dispose d'une vision limpide de l'état d'avancement du projet, favorisant ainsi la collaboration et l'engagement de tous les membres.

En résumé, la méthode Kanban s'impose comme un choix stratégique judicieux pour notre projet, offrant un cadre solide pour une gestion efficace, transparente et itérative, tout en servant l'objectif fondamental d'amélioration continue de la performance de l'équipe Avaxia.

Conclusion

À la clôture de ce chapitre initial, nous avons jeté les fondements pour la compréhension complète de la sphère du projet. Cette phase est cruciale pour cadrer les enjeux et les perspectives qui guident notre travail. Dans le prochain chapitre, nous explorerons en détail l'analyse et la spécification des besoins, une étape essentielle pour la réalisation de notre projet.

CHAPITRE 2

ANALYSE ET SPÉCIFICATION DES BESOINS

Plan

Introduction	12
1 Identification des besoins	12
2 Flux de travail	14
3 Le backlog du produit	16
Conclusion	17

Introduction

Le deuxième chapitre se consacre à l'analyse et à la spécification des besoins. Nous examinons en profondeur les besoins fonctionnels et non fonctionnels, tout en définissant clairement le flux de travail et le backlog du produit. Ces éléments constituent la base essentielle de la phase suivante de planification et de développement.

2.1 Identification des besoins

Dans cette section on va focaliser sur les besoins fonctionnels et non fonctionnels de notre projet.

2.1.1 Besoins fonctionnels

Cette section décrit en détail les besoins fonctionnels du projet de monitoring des opérations Snowflake. Ces besoins ont été identifiés à partir des exigences de l'entreprise et des utilisateurs, et sont essentiels pour le développement d'une solution efficace et adaptée aux besoins de l'entreprise.

- **Collecte automatique des données de performance de Snowflake** : Le système doit être capable de collecter automatiquement les données de performance de Snowflake, y compris les temps de réponse des requêtes, l'utilisation des ressources (CPU, mémoire, stockage) et les statistiques sur les entrepôts de données.
- **Surveillance en temps réel des requêtes SQL** : Le système doit surveiller en temps réel les requêtes SQL exécutées sur Snowflake, enregistrant les temps de réponse, les erreurs éventuelles, et en identifiant les requêtes lentes ou mal optimisées.
- **Surveillance en temps réel des entrepôts de données** : Le système doit surveiller en temps réel les métriques des entrepôts de données, y compris l'utilisation de l'espace disque, la répartition de la charge de travail et la consommation de ressources.
- **Tableaux de bord interactifs pour les métriques clés** : Le système doit fournir des tableaux de bord interactifs permettant de visualiser les métriques clés de performance de Snowflake, y compris les temps de réponse des requêtes, l'utilisation des ressources et les statistiques sur les entrepôts de données.
- **Suivi des tâches et des workflows (DAG monitoring)** : Le système doit permettre le suivi des tâches et des workflows exécutés sur Snowflake, enregistrant les étapes effectuées, les temps d'exécution et les erreurs éventuelles.

- **Surveillance de l'utilisation des crédits** : Le système doit surveiller l'utilisation des crédits de calcul sur Snowflake, enregistrant les crédits utilisés par chaque requête ou chaque tâche, ainsi que les tendances d'utilisation au fil du temps.
- **Surveillance des utilisateurs et des accès** : Le système doit suivre l'activité des utilisateurs sur Snowflake, en enregistrant les requêtes exécutées, les tables accédées et les autorisations utilisées.
- **Alertes et notifications en cas d'anomalies** : Le système doit générer des alertes et des notifications en temps réel en cas d'anomalies ou de situations critiques, telles qu'une augmentation soudaine du temps de réponse des requêtes ou une utilisation anormale des ressources.
- **Personnalisation des tableaux de bord** : Le système doit permettre aux utilisateurs de personnaliser les tableaux de bord en fonction de leurs besoins spécifiques, en sélectionnant les métriques à afficher et en configurant les seuils d'alerte.

2.1.2 Besoins non fonctionnels

Cette section détaille les besoins non fonctionnels du projet de monitoring des opérations Snowflake. Ces besoins concernent principalement les aspects de performance, de sécurité, d'évolutivité et d'expérience utilisateur de la solution. Ils sont essentiels pour garantir que le système répond aux attentes de l'entreprise en termes de qualité et de fiabilité.

- **Besoins en Performance**

1. **Temps de réponse** : Le système doit fournir des temps de réponse rapides pour l'affichage des tableaux de bord et la génération des rapports, afin de garantir une expérience utilisateur fluide.
2. **Évolutivité** : Le système doit être capable de gérer une grande quantité de données et de trafic sans compromettre ses performances, en s'adaptant de manière transparente à l'augmentation de la charge.

- **Besoins en Sécurité**

1. **Confidentialité des données** : Le système doit garantir la confidentialité des données collectées, en assurant leur cryptage lors du stockage et de la transmission.

2. **Authentification et autorisation** : Le système doit mettre en place des mécanismes d'authentification robustes pour vérifier l'identité des utilisateurs et des administrateurs, ainsi que des contrôles d'autorisation pour limiter l'accès aux données sensibles.

- **Besoins en Disponibilité**

1. **Disponibilité** : Le système doit être disponible en permanence, avec un temps de fonctionnement maximal et une reprise rapide en cas de panne.
2. **Tolérance aux pannes** : Le système doit être capable de résister aux pannes matérielles ou logicielles, en assurant la redondance des composants critiques et la sauvegarde des données.

- **Besoins en Expérience Utilisateur**

1. **Facilité d'utilisation** : Le système doit être intuitif et convivial, avec une interface utilisateur bien conçue et une navigation facile.
2. **Personnalisation** : Le système doit permettre aux utilisateurs de personnaliser leur expérience, en configurant les préférences d'affichage et les paramètres de notification.

- **Archivage des données** : Les données historiques doivent être archivées de manière efficace pour garantir l'intégrité des données.

2.2 Flux de travail

Durant la réalisation de notre projet, Ce processus a été notre fil conducteur, nous permettant de passer par plusieurs étapes clés pour atteindre notre objectif. La figure 2.1 suivante illustre notre processus métier :

Le processus métier du projet



FIGURE 2.1 : Processus métier du projet

Dans cette partie, nous allons explorer en détail le flux de travail que nous avons suivi, en soulignant l'importance de chaque étape et en expliquant comment elles se sont complétées pour aboutir à une

solution complète. Ensuite, nous examinerons chaque étape de manière approfondie pour comprendre comment elles ont contribué à notre succès global.

Passons maintenant à l'examen détaillé de chacune de ces étapes.

- **Extraction des données :**

Cette étape consistait à recueillir des informations provenant des différents vues du système de Snowflake.

L'objectif principal de cette étape était de collecter les données nécessaires à notre analyse.

Les données brutes sont la matière première de toute analyse, et leur extraction était le point de départ de notre projet.

- **Transformation des données :**

L'objectif principal de cette étape était de nettoyer et de structurer les données. Les données brutes extraites pouvaient contenir des erreurs, des doublons, des incohérences et des valeurs manquantes. La transformation des données avait pour but de les rendre utilisables, fiables et cohérentes.

- **Chargement des données :**

Une fois les données nettoyées et préparées, elles sont acheminées vers une base de données ou un entrepôt de données, comme PostgreSQL dans notre projet. Le chargement vise à stocker les données de manière centralisée pour une accessibilité et une analyse ultérieures.

- **Analyse des données :**

Cette phase consiste à explorer les données pour en extraire des informations significatives. On utilise des méthodes statistiques, des techniques de modélisation, des calculs de corrélation, etc., pour comprendre les tendances et les relations entre les données.

- **Visualisation des données :**

L'analyse des données est souvent présentée de manière visuelle sous forme de tableaux de bord interactifs, de graphiques, de diagrammes et d'autres représentations graphiques. Ces visualisations permettent aux utilisateurs de comprendre rapidement les résultats de l'analyse.

Ces phases constituent un flux de travail complet pour gérer les données, les transformer en informations exploitables et les présenter de manière efficace pour aider nos utilisateurs finaux à surveiller leurs données d'une manière concrète.

2.3 Le backlog du produit

Le backlog de produit est destiné à recueillir tous les besoins du client pour l'équipe projet. Il contient la liste des fonctionnalités incluses dans un produit, ainsi que les éléments nécessitant l'intervention de l'équipe projet. Le backlog Scrum classe les éléments par priorité pour indiquer leur ordre de réalisation.[3].

Le backlog en KANBAN n'est pas si différent d'un backlog scrum en réalité. La différence réside surtout sur les règles de sa gestion. Bien que le scrum reste évasif sur le sujet, il rappelle quelques règles importantes[4].

la table **2.1** suivante présente le backlog de produit de notre projet.

ID	User Story	Priorité	Estimation (points)
US1	En tant qu'utilisateur, je veux un tableau de bord global pour surveiller les performances générales de l'entrepôt de données, y compris les temps de réponse des requêtes, l'utilisation des ressources et les erreurs éventuelles.	Haute	8
US2	En tant qu'administrateur, je veux être en mesure de suivre en temps réel l'exécution des requêtes SQL sur la plateforme Snowflake, y compris les temps d'exécution, le nombre de lignes retournées et les plans d'exécution.	Haute	10
US3	En tant qu'administrateur, je veux surveiller l'utilisation des ressources (CPU, stockage, etc.) de l'entrepôt de données pour identifier les goulots d'étranglement et les problèmes de capacité.	Haute	9
US4	En tant qu'administrateur, je veux recevoir des alertes en temps réel en cas de dégradation des performances de l'entrepôt de données afin de pouvoir réagir rapidement et résoudre les problèmes.	Haute	8
US5	En tant qu'utilisateur, je veux générer des rapports de charge pour analyser la consommation de crédits, le coût par requête et l'utilisation des ressources au fil du temps.	Moyenne	6

US6	En tant qu'administrateur, je veux suivre l'activité des utilisateurs sur la plateforme Snowflake, y compris les requêtes exécutées, les tables accédées et les autorisations utilisées.	Moyenne	7
US7	En tant qu'administrateur, je veux être capable de gérer les sessions utilisateur actives, y compris la déconnexion des sessions inactives et la surveillance des sessions gourmandes en ressources.	Haute	8
US8	En tant qu'administrateur, je veux pouvoir définir des alertes personnalisées basées sur des seuils de performance spécifiques pour les requêtes, les ressources et les sessions.	Moyenne	7
US9	En tant que développeur, je veux accéder à des recommandations d'optimisation des requêtes SQL pour améliorer les performances et réduire les coûts.	Faible	9
US10	En tant qu'administrateur, je veux surveiller l'exécution des tâches planifiées (comme les pipelines de chargement) pour détecter les retards ou les échecs.	Moyenne	6
US11	En tant qu'utilisateur, je veux pouvoir analyser les tendances historiques des performances pour identifier les schémas et les anomalies.	Moyenne	8
US12	En tant qu'utilisateur, je veux s'authentifier d'une façon sécurisée pour accéder à la plateforme de monitoring	Moyenne	5
US13	En tant qu'administrateur, je veux suivre en temps réelle l'exécution des tâches programmées	haute	13

TABLEAU 2.1 : Backlog de Produit

Conclusion

Au terme de ce chapitre, nous avons obtenu une vision approfondie des besoins inhérents au projet. Cette analyse détaillée assure que notre solution est en phase avec les attentes et les exigences d'Avaxia Group. Dans le prochain chapitre, nous explorerons les aspects architecturaux et conceptuels de notre projet.

CHAPITRE 3

ARCHITECTURE ET CONCEPTION

Plan

Introduction	19
1 Étude architecturale	19
2 Étude conceptuelle	26
Conclusion	41

Introduction

Le troisième chapitre explore l'architecture du projet et sa étude conceptuelle. Nous allons justifier notre choix architectural et détaillrons la conception qui nous a mené à la réalisation de notre solution.

3.1 Étude architecturale

L'étude architecturale joue un rôle central dans la conception et le déploiement d'une solution robuste et évolutive.

C'est dans ce contexte,

3.1.1 Architecture logique

Dans cette section, nous examinerons en profondeur l'architecture du système, couvrant à la fois ses dimensions logiques et physiques. Nous discuterons également des défis techniques et logiques rencontrés et des solutions déployées pour assurer une infrastructure solide et efficace.

3.1.1.1 Justification du choix de l'architecture micro-services

Le choix de l'architecture Le choix de l'architecture d'une application revêt une importance capitale dans le processus de conception d'un projet. Il détermine comment les diverses parties de l'application interagiront pour atteindre les objectifs fixés.

Pour la mise en place de ce projet, nous avons délibérément opté pour **une architecture micro-services**, une décision dictée par des critères spécifiques qui s'accordent parfaitement avec les exigences de notre projet ainsi que les différentes parties prenantes. Dans cette section, nous explorerons en détail les raisons pour lesquelles nous avons opté pour cette architecture, en soulignant sa pertinence pour notre projet.

- **Flexibilité et scalabilité :** nous avons choisi l'architecture microservices pour sa flexibilité et sa capacité à évoluer facilement. En adoptant une approche basée sur des services indépendants, nous pouvons développer, déployer et mettre à l'échelle chaque composant de manière autonome, ce qui facilite l'adaptation aux changements de charge et aux besoins évolutifs de notre système,
- **Isolation des fonctionnalités :** grâce aux microservices, le déploiement et la gestion des applications sont simplifiés. Chaque service peut être déployé de manière indépendante, ce qui

réduit les risques et accélère les cycles de déploiement. De plus, cela facilite la gestion des mises à jour et des correctifs, car seuls les services concernés sont impactés,

- **Déploiement et gestion simplifiés** : grâce aux microservices, le déploiement et la gestion des applications sont simplifiés. Chaque service peut être déployé de manière indépendante, ce qui réduit les risques et accélère les cycles de déploiement. De plus, cela facilite la gestion des mises à jour et des correctifs, car seuls les services concernés sont impactés,
- **Évolutivité et résilience** : les microservices favorisent une architecture résiliente et évolutive. En cas de panne d'un service, les autres services peuvent continuer à fonctionner normalement, ce qui réduit les interruptions. De plus, cette approche permet une meilleure répartition de la charge, assurant des performances optimales même lors de pics de trafic.

En conclusion, l'architecture microservices offre une solution flexible, évolutive et facilement maintenable pour notre projet de monitoring des opérations sur Snowflake. Elle nous permet de répondre efficacement aux besoins changeants de nos utilisateurs tout en garantissant des performances et une fiabilité optimales de notre système.

3.1.1.2 Architecture logique adoptée

Comme nous avons déjà mentionné, l'architecture logique de notre solution adopte une approche modulaire reposant sur les microservices, afin de constituer le socle sur lequel repose la réalisation de l'objectif du projet. Elle met en lumière les différents composantes fonctionnelles et explique comment ces composantes interagissent pour fournir une expérience utilisateur fluide et des analyses précises[5].

Notre architecture est illustrée par la figure 3.1 suivante :

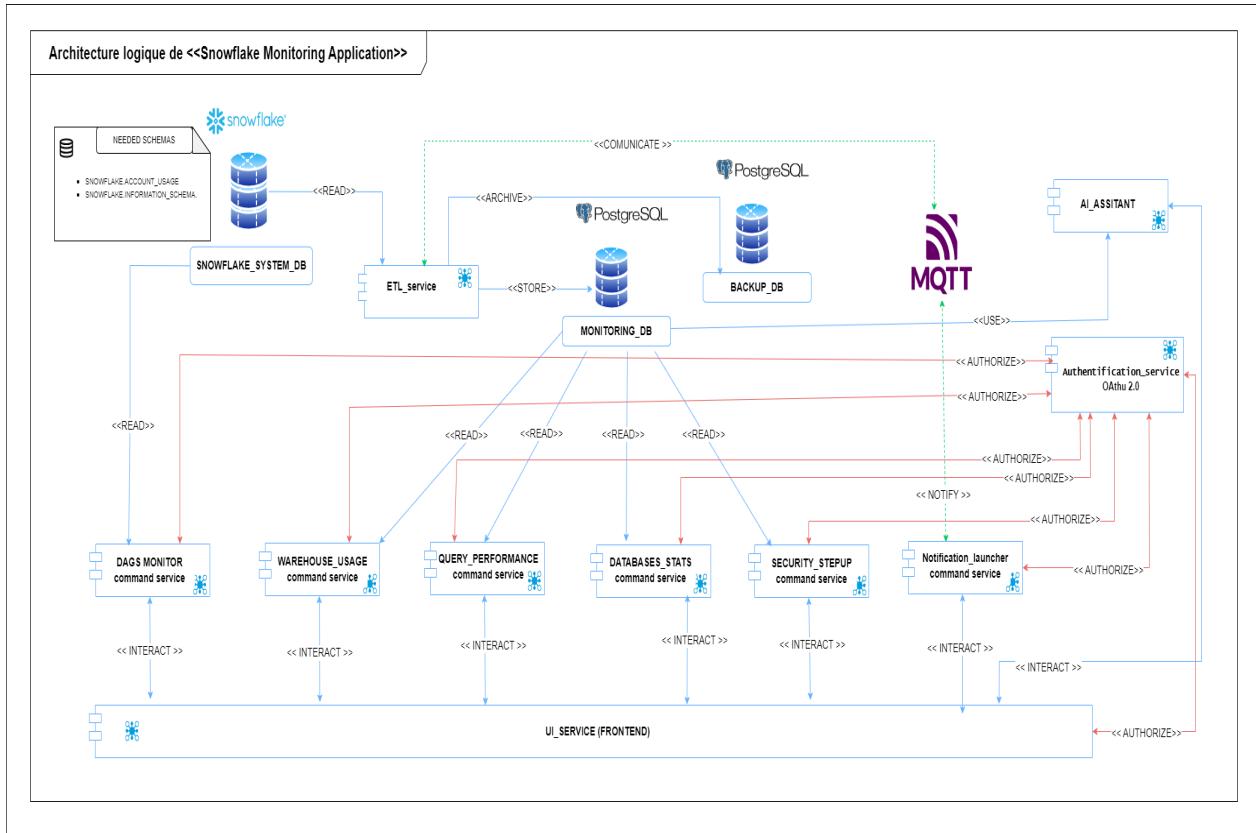


FIGURE 3.1 : Architecture logique de «Snowflake Monitoring Application»

Notre architecture est repartie comme suit :

1. Couche des données :

- **Snowflake** : ce composant représente les vues systèmes de Snowflake telles que «Account_usage» et «information_schema» qui regroupent tous les données à monter.
- **Monitoring_DB** : c'est une base de données PostgreSQL stockant les données de monitoring collectées.
- **BACKUP_DB** : c'est une base de données PostgreSQL qui permet de garantir la durabilité et la redondance des données, en cas de panne ou de besoin de restauration des données de MONITORING_DB principale.

2. Couche applicative « micro-services » :

- **Etl_Service** : service responsable de l'extraction, de la transformation et du chargement des données depuis Snowflake vers la base de données de monitoring,
- **Dags_Monitoring_Service** : service responsable du suivi et de la surveillance des workflows «DAG» Snowflake,

- **Warehouse_Usage_Service** : service chargé de la surveillance de l'utilisation des entrepôts de données Snowflake,
- **Query_Performance_Service** : service d'analyse des performances des requêtes Snowflake,
- **Databases_Stats_Service** : service de collecte des statistiques sur les bases de données Snowflake,
- **Access_Stats_Service** : service de collecte des statistiques sur les access au differents comptes et entrepôts de données Snowflake,
- **notification_launcher** :service responsable de la gestion des notifications. Il s'abonne aux événements publiés par le service ETL_service via le système MQTT, et se charge ensuite de transmettre ces notifications aux utilisateurs finaux par les canaux de communication appropriés,
- **Authentification_service** : service d'authentification et d'autorisation basé sur OAuth 2.0.
- **AI_ASSISTANT_service** : un service d'assistance IA intégré dans l'architecture pour apporter une dimension d'intelligence artificielle aux fonctionnalités de surveillance et d'analyse.

Cette couche assure la gestion de la logique métier de l'application en traitant et examinant les données collectées. Elle interagit avec l'interface utilisateur pour fournir des informations pertinentes, des fonctionnalités d'analyse, et des résultats.

3. Couche de Communication :

- **MQTT** : système de messagerie basé sur le protocole MQTT, utilisé pour la communication asynchrone et découpée entre les différents services.
- **MQTT Broker** : composant central du système MQTT, chargé de la distribution des événements publiés par les services.
- **REST** : la communication entre les microservices ce fait via des API REST.

4. Couche de Présentation :

- **UI_Service** : ce servise fournit des tableaux de bord et des interfaces utilisateur pour visualiser les données de monitoring.

Il offre une expérience utilisateur conviviale et interactive pour la visualisation des données de performance et les résultats d'analyse.

3.1.1.3 Patrons de conception

Les patrons de conception représentent des solutions éprouvées à des problèmes fréquents en conception logicielle. Ce sont comme des modèles ou des structures que l'on peut adapter pour résoudre des problèmes récurrents dans notre code[6].

Lors de l'élaboration de cette architecture, nous avons focaliser sur le fait de respecter les patrons de conception afin d'avoir une architecture solide et structurée.

Nous avons utilisé les patrons de conception suivants :

- **Le patron «Command Query Responsibility Segregation (CQRS)»** : il est aussi un modèle d'architecture qui sépare les deux parties de traitement (Écriture) et de réponse (Lecture) [7].

L'architecture applique le modèle CQRS, avec une séparation des responsabilités entre le services de commande (ETL_service) et les services de requête (tout les autres services). Cette séparation optimise les performances des flux de lecture et d'écriture de manière indépendante.

- **Le patron «Façade»** : c'est un modèle de conception structurel qui fournit une interface simplifiée pour accéder à une bibliothèque, un framework ou à tout ensemble complexe de classes[8].

Le service «ETL_service» fait office de façade, en encapsulant la complexité de l'interaction avec Snowflake, de façon que les autres services n'ont pas besoin de connaître les détails de l'interaction avec Snowflake.

- **Le patron «Observateur»** : c'est un modèle de conception comportemental qui permet d'établir un mécanisme de souscription pour envoyer des notifications à plusieurs objets concernant des événements liés aux objets qu'ils observent[9].

Le service «MQTT_broker» fait office d'observateur, car il implémente un modèle de publication/souscription, permettant aux services de s'abonner aux événements qui les intéressent. Cela découpe les services et facilite l'ajout de nouveaux services.

- **Le principe «de responsabilité unique (SRP)»** : ce principe dénonce que chaque composant (classe, service, module, etc.) ne doit avoir qu'une seule raison de changer. Cela se traduit par une séparation claire des responsabilités au sein de l'architecture[10].

Chaque microservice (DAGS MONITOR, WAREHOUSE_USAGE, QUERY_PERFORMANCE, etc.) est responsable d'une fonctionnalité spécifique du monitoring.

De plus, la séparation entre la base de données Snowflake (données surveiller) et la base de données Monitoring_DB (données de monitoring) respecte le SRP.

- **Le patron «Objet d'accès aux données (DAO)» :** ce modèle représente une façon d'organiser le code pour gérer la communication entre le programme et la base de données. Il permet de conserver un code propre et de séparer la logique d'interaction avec les données du reste de l'application[11].

L'accès à la base de données de monitoring (Monitoring_DB) est géré via un modèle DAO, séparant la logique d'accès aux données du reste de l'application. Cela améliore la maintenabilité et permet une indépendance entre la couche de données et la logique métier.

3.1.2 Architecture physique

L'architecture physique d'un projet constitue la matérialisation concrète de son architecture logique. Elle se consacre aux aspects matériels et aux ressources nécessaires pour assurer le bon fonctionnement de l'application[12].

Pour la mise en place de ce système de monitoring Snowflake, nous avons suivi une approche en n-tiers afin de garantir une séparation claire des responsabilités, une meilleure évolutivité et une plus grande résilience de l'ensemble du système. Elle permet une gestion efficace des ressources, une répartition optimale des charges de travail et une sécurisation renforcée de l'infrastructure.

la figure 3.2 suivante illustre l'architecture adopté dans ce projet :

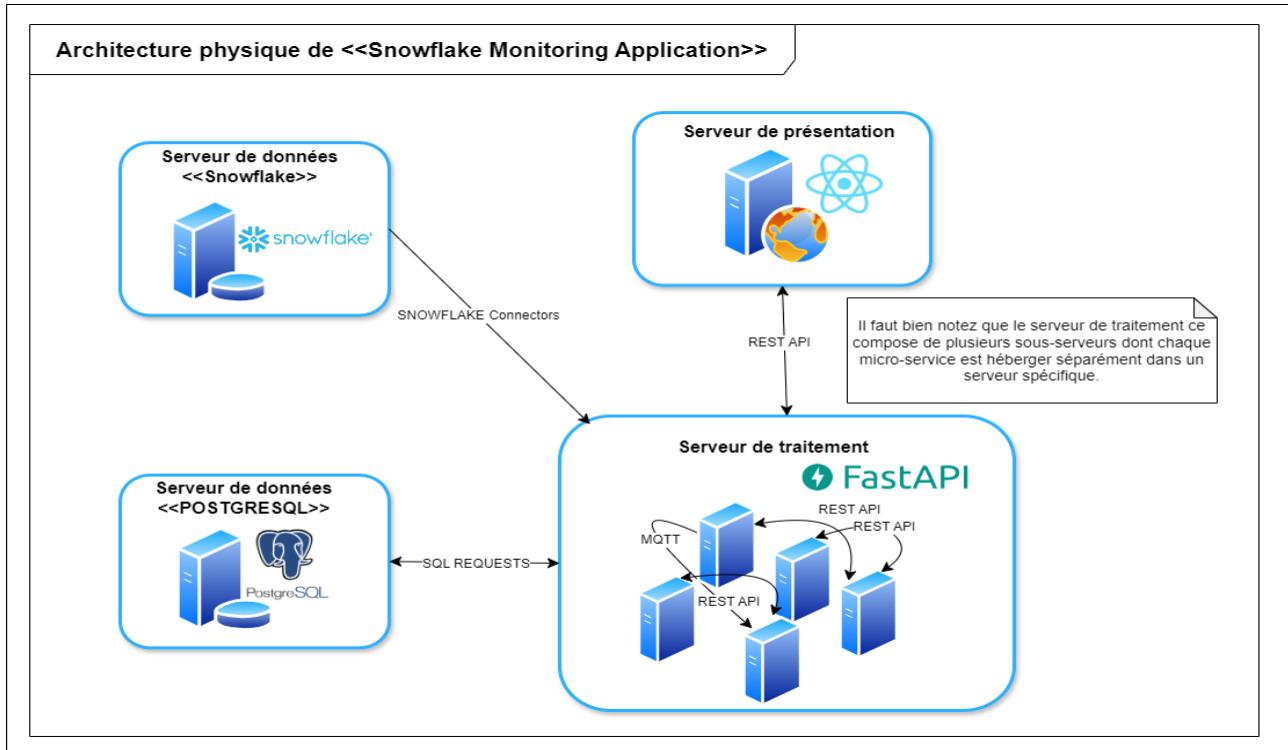


FIGURE 3.2 : Architecture physique de «Snowflake Monitoring Application»

Notre architecture physique est composée de plusieurs éléments essentiels qui travaillent en tandem pour permettre la collecte, le traitement, le stockage et la présentation des données de manière optimale. Voici un aperçu détaillé de ces composants :

- 1- **Tier des Services (Tier Application)** : cet tier comprend les différents serveurs d'application hébergeant les microservices du système.

Il inclut des serveurs pour les services DAGS_MONITOR, WAREHOUSE_USAGE, QUERY_PERFORMA ETL_service, AUTH_service, etc.

Ces serveurs sont responsables de la logique métier et du traitement des données.

- 2- **Tier des Données** : cet tier est composé du serveur de base de données hébergeant la base de données MONITORING_DB.

Il est chargé du stockage et de la gestion des données de monitoring collectées.

Le serveur Snowflake, hébergeant la base de données SNOWFLAKE_SYSTEM_DB, fait également partie de ce tier des données.

- 3- **Tier de présentation** : cet tier comprend le serveur frontend hébergeant le service ui_service (Frontend).

Il est responsable de la présentation des données de monitoring aux utilisateurs via les tableaux

de bord et les interfaces.

4- **inter-communication** : la communication est représentée modes de communication utilisés entre les composants de l'architecture, via les requêtes SQL, les API REST, les connecteurs Snowflake et le système de messagerie MQTT.

Cette architecture en n-tiers offre plusieurs avantages clés, tels que la séparation des préoccupations, l'évolutivité, la résilience et la sécurité renforcée du système. Chaque tier étant responsable d'une fonction spécifique, il devient plus aisément de maintenir, de mettre à l'échelle et de sécuriser les différents composants de manière indépendante.

3.2 Étude conceptuelle

Dans cette section nous allons nous intéresser à la modélisation de notre projet à l'aide des différents modèles et diagrammes qui vont nous permettre de mieux comprendre le flux de travail ainsi que la nature des données et les traitements qui circulent dans notre système.

3.2.1 Conception globale

Dans cette section, nous allons nous intéresser à présenter les différents diagrammes illustrant la conception globale de notre solution.

3.2.1.1 Diagramme de paquetages

Les diagrammes de paquetages sont des diagrammes structurels qui illustrent l'organisation et la disposition de différents éléments modélisés sous forme de packages[13].

Il sert à grouper des éléments en un ensemble cohérent, et à fournir un espace de noms pour ces éléments.

La figure 3.3 qui suit représente le diagramme de paquetages de notre système :

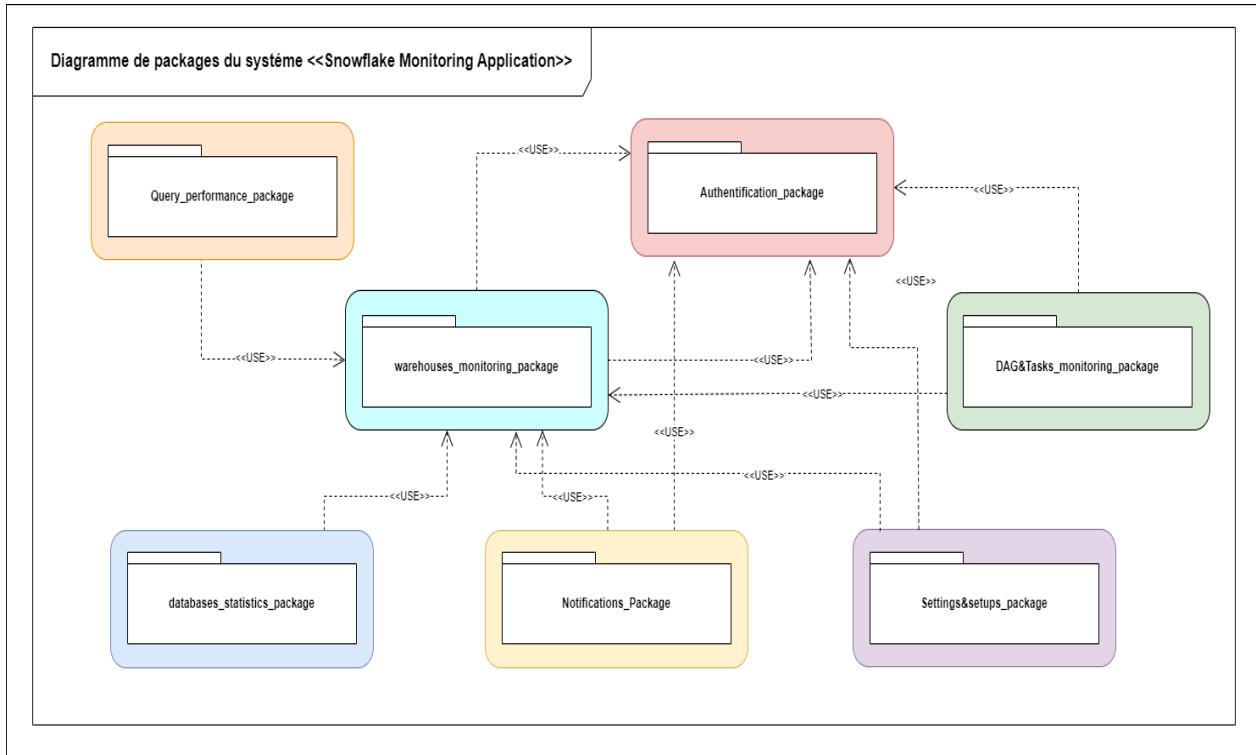


FIGURE 3.3 : Diagramme de paquetages de «Snowflake Monitoring Application»

3.2.1.2 Diagramme de classe

Les diagrammes de classes fournissent une représentation détaillée de la structure d'un système spécifique. Ils modélisent les classes du système, leurs attributs, leurs opérations ainsi que les relations entre les objets[14].

Ce diagramme permet de visualiser clairement la composition du système, facilitant ainsi la compréhension des interactions et des dépendances entre les différents éléments. En représentant les attributs et les méthodes des classes, ils aident également à définir les responsabilités et les comportements des objets dans le contexte global du système.

La figure 3.4 qui suit représente le diagramme de classe de notre système :

Chapitre 3. Architecture et conception

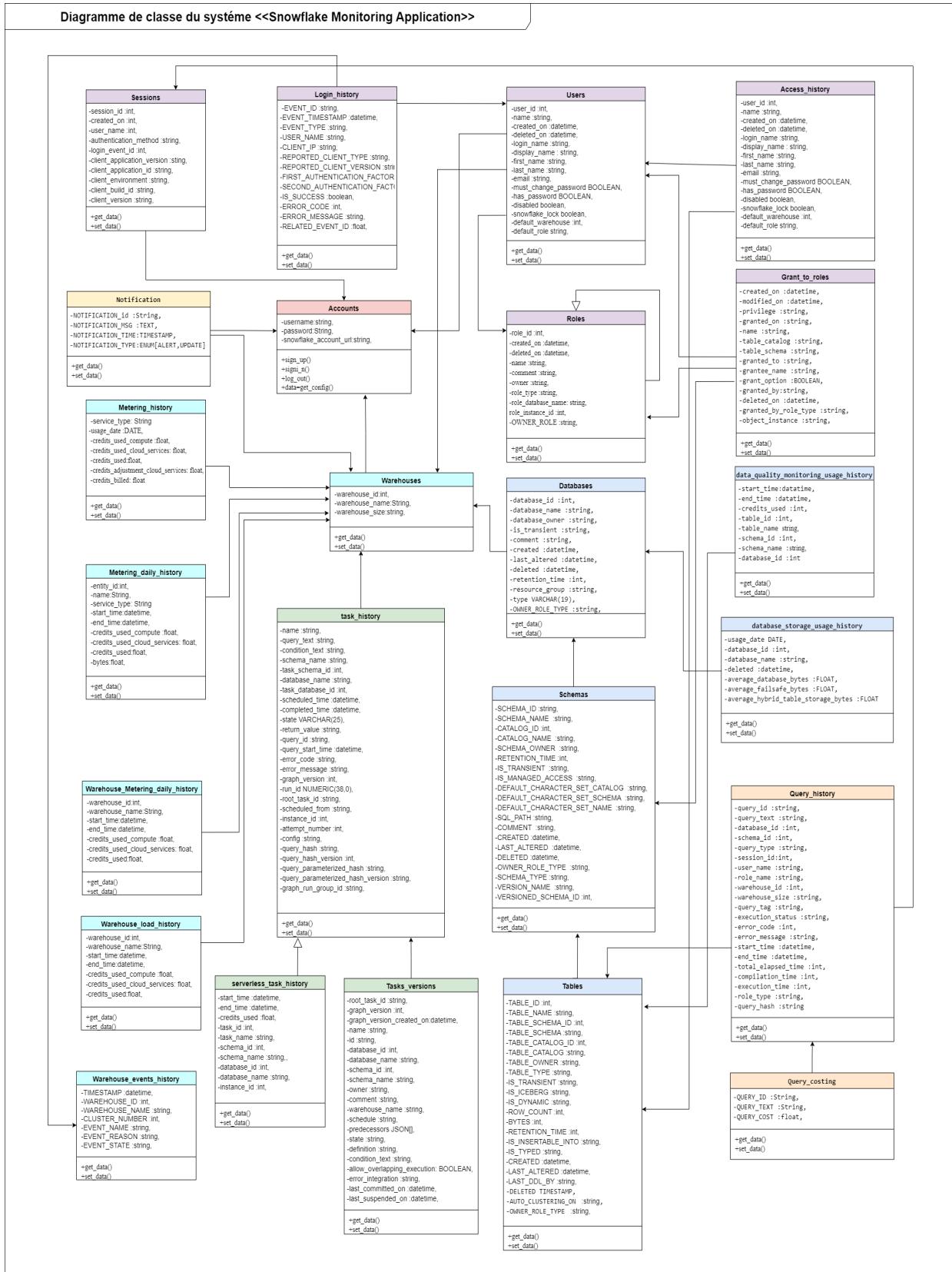


FIGURE 3.4 : Diagramme de classe de «Snowflake Monitoring Application»

3.2.1.3 Diagramme de cas d'utilisation global

Le but fondamental d'un diagramme de cas d'utilisation UML consiste à illustrer les diverses interactions entre un utilisateur et un système [15].

La figure 3.5 qui suit représente le diagramme de cas d'utilisation global de notre système :

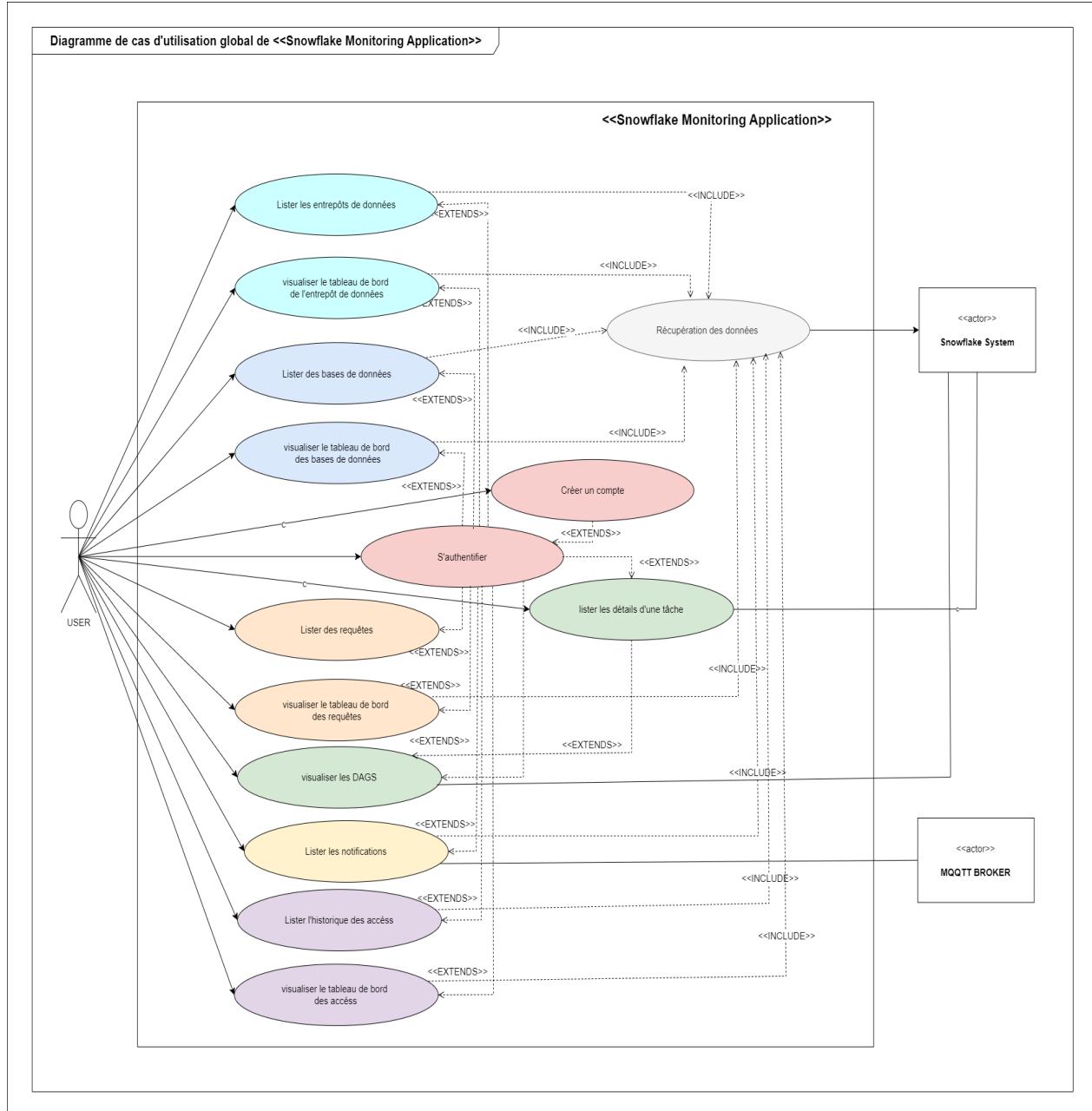


FIGURE 3.5 : Diagramme de cas d'utilisation globale «Snowflake Monitoring Application»

3.2.2 Conception détaillée

Dans cette section, nous allons s'intéresser à présenter les diagrammes illustrant la conception détaillée de chaque micro-service présent dans solution.

3.2.2.1 Description textuelle des cas d'utilisation

c'est une description claire et précise des interactions entre les acteurs et le système pour mener à bien le cas d'utilisation. Cette description identifie les acteurs impliqués ainsi que les autres cas d'utilisation pertinents[16].

1. Description textuelle du cas d'utilisation «Lister les entrepôts de données» :

Titre	Lister les entrepôts de données
Résumé	L'utilisateur souhaite de lister les données de ses entrepôts de données'
Acteurs	Utilisateur
Pré conditions	L'utilisateur est authentifié et autorisé, via une token, à accéder à cette fonctionnalité
Scénarios Nominaux	<ol style="list-style-type: none"> 1. L'utilisateur sélectionne l'option «Warehouses», 2. l'application récupère la liste des entrepôts et leurs données de la base de données Monitoring_DB, 3. l'application affiche les informations détaillées sur chaque entrepôt (nom, taille, performances, etc.), 4. l'utilisateur peut consulter les détails de chaque entrepôt.
Scénario alternatif	3.a. [«Pas de warehouses»] : Le système retourne un tableau indiquant que la liste est vide
Scénarios d'exceptions	[«Token expirée»] : Le système signale l'erreur et redirige l'utilisateur vers la page de «login».
Post conditions	L'utilisateur a accès à la liste des entrepôts de données et peut consulter leurs informations détaillées.

TABLEAU 3.1 : description textuelle de cas d'utilisation «Lister les entrepôts de données»

2. Description textuelle du cas d'utilisation «Visualiser DAG» :

Titre	Lister les entrepôts de données
Résumé	L'utilisateur souhaite de lister les données de ses entrepôts de données'
Acteurs	Utilisateur
Pré conditions	L'utilisateur est authentifié et autorisé, via une token, à accéder à cette fonctionnalité
Scénarios Nominaux	<ol style="list-style-type: none"> 1. L'utilisateur sélectionne l'option «DAG monitoring», 3. Le système effectue une connexion directe au compte snowflake à partir du service d'authentification en utilisant le nom d'utilisateur fourni dans la token, 2. Le système récupère la configuration des tâches, 4. Le système effectue un parcours en profondeur (DFS) du DAG pour identifier les nœuds racines, les nœuds feuilles et les liens entre les nœuds, 5. Le système prépare tout les informations récupérées sous le format d'une arbre, 6. Le système affiche chaque tâche ainsi que son état et ces liens avec les autres tâches.
Scénario alternatif	4.a. [«Aucune DAG signalée»] : Le système retourne le repère en indiquant que il y aucune DAG.
Scénarios d'exceptions	[«Token expirée»] : Le système signale l'erreur et redirecte l'utilisateur vers la page de «login».
Post conditions	L'utilisateur reçoit les informations détaillées sur le DAG des tâches Snowflake, lui permettant de visualiser et d'analyser la structure du workflow.

TABLEAU 3.2 : description textuelle de cas d'utilisation «Visualiser DAG»

2. Description textuelle du cas d'utilisation «Visualiser DAG» :

Titre	Lister les entrepôts de données
Résumé	L'utilisateur souhaite de lister les données de ses entrepôts de données'
Acteurs	Utilisateur
Pré conditions	L'utilisateur est authentifié et autorisé, via une token, à accéder à cette fonctionnalité
Scénarios Nominaux	<ol style="list-style-type: none"> 1. L'utilisateur sélectionne l'option «DAG monitoring», 3. Le système effectue une connexion directe au compte snowflake à partir du service d'authentification en utilisant le nom d'utilisateur fourni dans la token, 2. Le système récupère la configuration des tâches, 4. Le système effectue un parcours en profondeur (DFS) du DAG pour identifier les nœuds racines, les nœuds feuilles et les liens entre les nœuds, 5. Le système prépare tout les informations récupérées sous le format d'une arbre, 6. Le système affiche chaque tâche ainsi que son état et ces liens avec les autres tâches.
Scénario alternatif	4.a. [«Aucune DAG signalée»] : Le système retourne le repère en indiquant que il y aucune DAG.
Scénarios d'exceptions	[«Token expirée»] : Le système signale l'erreur et redirekte l'utilisateur vers la page de «login».
Post conditions	L'utilisateur reçoit les informations détaillées sur le DAG des tâches Snowflake, lui permettant de visualiser et d'analyser la structure du workflow.

TABLEAU 3.3 : description textuelle de cas d'utilisation «Visualiser DAG»

3.2.2.2 Diagramme d'activité du micro-service «ETL-service» :

Les diagrammes d'activités démontrent la logique d'un algorithme. De plus, ils fournissent une vue détaillée du comportement d'un système en décrivant la séquence d'actions au sein d'un processus[17].

Le diagramme d'activité du micro-service ETL-service est représenté dans la figure 3.6 suivante :

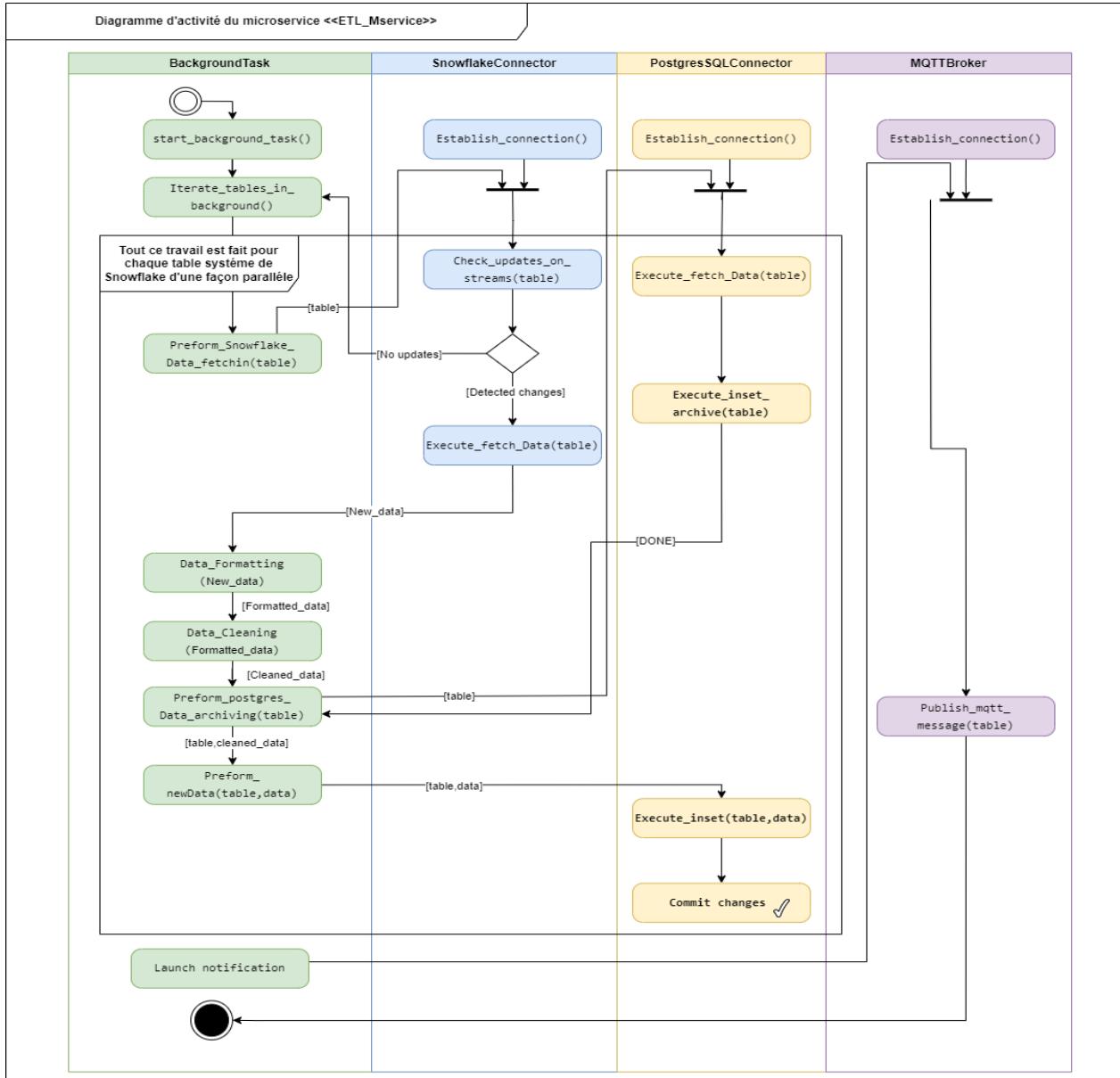


FIGURE 3.6 : Diagramme d'activité du micro-service «ETL-service»

Le traitement du micro-service ETL commence par l'appel de la méthode «`start_background_task()`», qui est une méthode programmée automatiquement chaque 24 heures. En lançant cette méthode, elle parcourt les tables du système Snowflake, dont nous voulons projeter, en utilisant la méthode «`iterate_tables_in_background()`».

À ce stade, le connecteur Snowflake établit une connexion avec le système Snowflake via la méthode «`establish_connection()`», qui à ce point connecte notre service avec le compte Snowflake dont nous souhaitons montrer ces données plus tard.

Ensuite, le contrôleur système vérifie s'il y a des mises à jour sur les tables Snowflake en appelant

la méthode «`check_updates_on_streams(tables)`».

Si il y des changements détectés, la méthode «`execute_fetch_data(table)`» est exécutée pour récupérer les nouvelles données de la table Snowflake.

Les nouvelles données récupérées sont ensuite transmises à l'étape de formatage des données «`Data_Formatting()`», où elles sont mises en forme(typage, longeur, format etc.). Après cela, l'étape de nettoyage des données «`Data_Cleaning()`» intervient pour préparer et nettoyer les données formatées.

Postérieurement, l'étape d'archivage des données dans PostgreSQL «`Preform_Data_archiving`», là où nous allons transformer les anciennes données de cette table vers une base de données d'archivage afin d'avoir toujours une traçabilité des données.

Parallèlement, le connecteur PostgreSQL établit une connexion avec les base de données PostgreSQL, `monitoring_db` et la base d'archive `backup_db`, via la méthode «`establish_connection()`». tout en exécutant la méthode «`execute_insert_archive(table, data)`».

Par la suite, le système déclanche la méthode «`Preform_newData(table,data)`» pour insérer les nouveaux données nettoyées dans la base de données de monitoring. Après l'insertion des données, le microservice effectue un commit des changements dans la base de données.

il faut notez bien que, ce flux de travail ce repête éventuellement pour chaque table, séparément ou/et séquentiellement pour les tables qui ont une relation entre eux, dans un «`thread`» à part.

Après la mise à jour de tout les tables, il publie un message MQTT via la méthode «`publish_mqtt_message()`» afin de notifier le système de l'ajout de nouvelles données.

Cette approche garantit que toutes les étapes sont correctement coordonnées et que chaque composant est informé des mises à jour de manière efficace et en temps réel.

Enfin, le microservice ETL-service termine son traitement en attendant son prochain déclanchement.

3.2.2.3 Diagramme de séquence de cas d'utilisation «Visualiser DAG» :

Les diagrammes d'activités démontrent la logique d'un algorithme. De plus, ils fournissent une vue détaillée du comportement d'un système en décrivant la séquence d'actions au sein d'un processus[17].

Le diagramme de séquence de cas d'utilisation «Visualiser DAG» est représenté dans la figure 3.7 suivante :

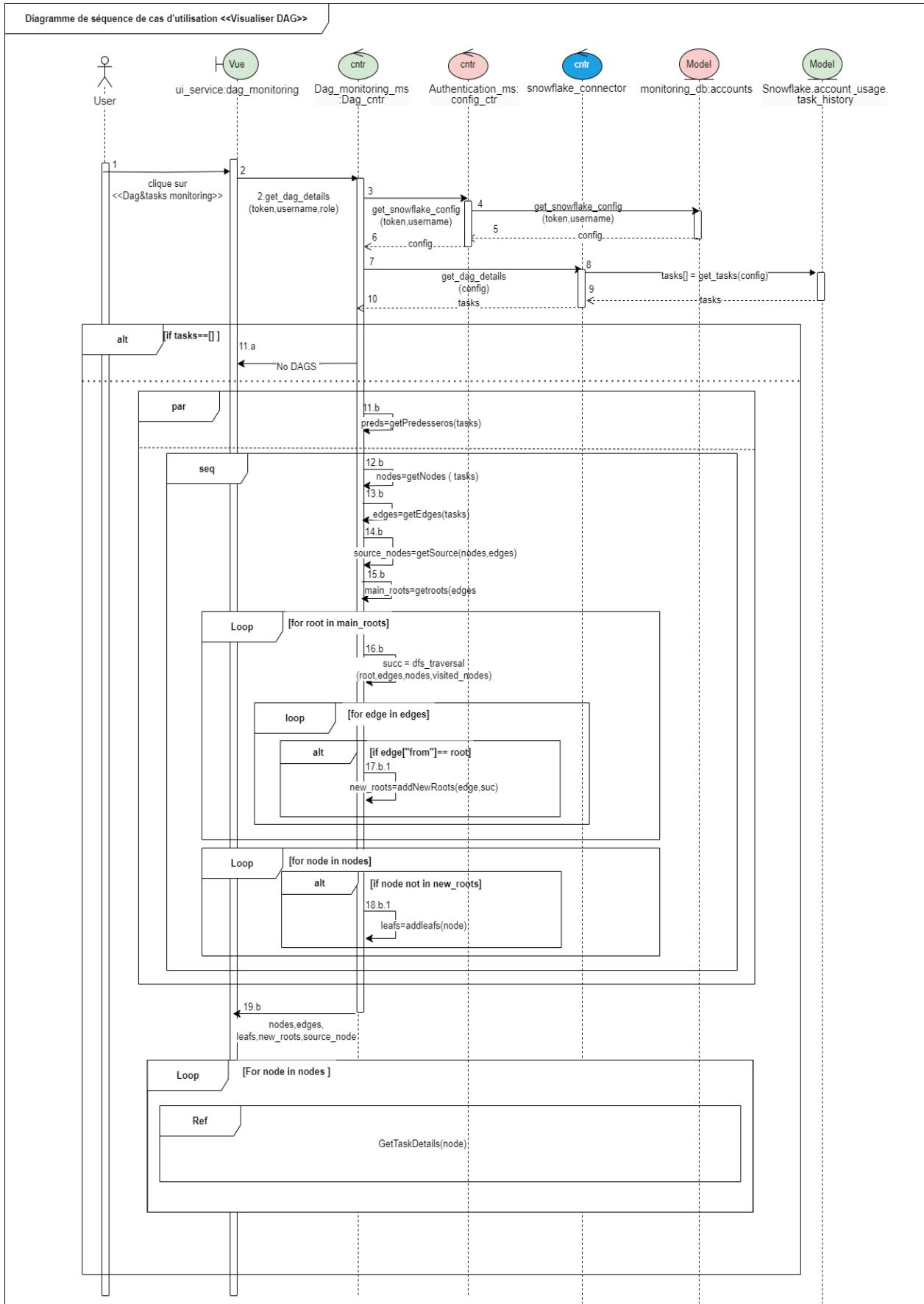


FIGURE 3.7 : Diagrammme de séquence de cas d'utilisation «Visualiser DAG»

Ce diagramme de séquence décrit le cas d'utilisation «**Visualiser le DAG**» où un utilisateur souhaite visualiser les détails du Directed Acyclic Graph (DAG) des tâches Snowflake en temps réel. Le traitement **ce** lance lorsque l'utilisateur clique sur le bouton «Task&Dag Monitoring» de l'interface, cette action lance un appel au contrôleur du micro service «DAG_monitoring» pour récupérer les DAGs. Une fois l'appel est lancé, le contrôleur en question envoie à son tour une requête au micro-service d'authentification pour avoir les configurations de compte Snowflake associés au nom d'utilisateur avec la token **d'autorization**.

Le contrôleur snowflake config_ctrl du service d'authentification renvoie les configurations nécessaires au contrôleur principal.

Une fois les configurations sont retournées, le contrôleur de DAG_monitoring fait appel au snowflake_connector, qui est le point de communication direct entre l'application et Snowflake, pour accéder au compte Snowflake de l'utilisateur et récupérer la liste des tâches qui lui sont associées.

le contrôleur snowflake_ctrl effectue les opérations nécessaires sur Snowflake et retourne la liste des tâches au DAG_ctrl :

- **Si [la liste est vide]** ; le contrôleur DAG_ctrl renvoie un message au service du frontend en indiquant qu'il y a aucune DAG pour cette utilisateur,
- **Sinon** ;

-le contrôleur DAG_ctrl commence le traitement sur les tâches retournées, en récupérant la liste des prédecesseurs de chaque tâches.

-Parallèlement, dans un autre thread séparément, il récupère les noeuds (nodes), les arêtes (edges), les noeuds sources (source_nodes), les routes principales (main_roots).

-Ensuite, pour chaque route il fait un parcours en profondeur (DFS_transferral) afin marquer les noeuds visités et collecter les successeurs.

-Puis, il vérifie pour chaque arête, si l'arête part de la racine actuelle et si c'est le cas, il ajoute le noeud de destination au new_roots qui sera l'ensemble des nouvelles racines pour la prochaine itération de la boucle principale.

-Après tout cela, il parcourt tous les noeuds pour identifier les noeuds feuilles, ceux qui ne sont pas dans new_root.

-Finalement, il renvoie toutes les informations traitées aux services frontend comme étant un objet JSON.

Enfin, le service frontend va lui-même parcourir les noeuds de l'objet retourné, en dessinant le

DAG finale en faisant appel aux contrôleur Task_details inclus du même service pour afficher en details chaque noeud du DAG.

NB : Il faut bien préciser que le déclenchement de cet cas d'utilisation nécessite l'authentification de l'utilisateur et qu'il a les permissions et les rôles qui lui permet de visualiser les DAG.

3.2.2.4 Diagramme de séquence du cas d'utilisation «Lister les détails d'une tâche» :

La figure 3.8 illustre le diagramme de séquence du cas d'utilisation «Lister les détails d'une tâche» :

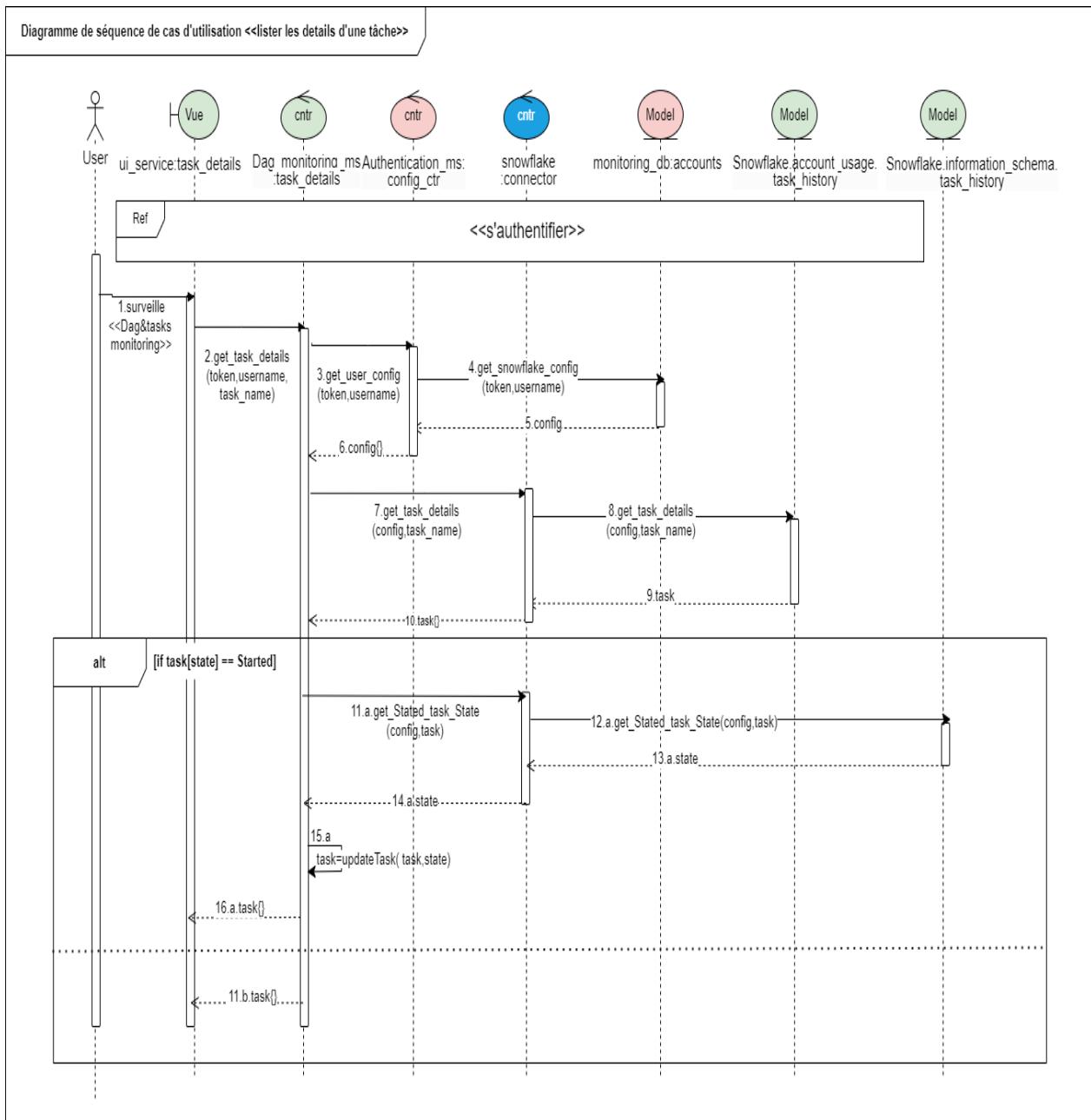


FIGURE 3.8 : Diagramme de séquence de cas d'utilisation «Lister les détails d'une tâche»

Pour Lister les details d'une tâche, et après avoir vérifier que l'utilisateur est authentifié, le service du frontend lance un appel au contrôleur task_details pour récupérer les données souhaitées.

En lançant cet contrôleur, il envoit à son tour une requête au micro-service d'authentification pour avoir les configurations de compte Snowflake assosiés au nom d'utilisateur avec la token d'autorization.

Le contrôleur snowflake config_ctrl du service d'authentification renvoie les configurations nécessaires au contrôleur principal.

Une fois les configurations sont retournées, le contrôleur de DAG_monitoring fait appel au snowflake_connector, qui est le point de communication direct entre l'application et Snowflake, pour accéder au compte Snowflake de l'utilisateur et récupérer les détails de tâche en question de la vue **Task_History** du schéma système de snowflake **Account_Ussage**, qui a des informations global sur les tâches.

le contrôleur snowflake_ctrl effectue les opérations nécessaires sur Snowflake et retourne les détails du task demandé sous le format JSON.

Le contrôleur task_details traite le résultat retourné ;

- **Si [L'état du tâche est «Started»] :**

- le contrôleur task_details relance un autre appel au snowflake_connector,
 - ce dernier il va accéder à nouveau à Snowflake mais cette fois si il va opérer la vue **Task_History** du schéma **Information_schema** de Snowflake, qui est la vue qui à les informations sur les tâches commencées en temps réelle, pour récupérer l'état de cette tâche en temps réelle qui peut être (planifiée, entrain de s'exécuter ou échouée),
 - une fois le nouveau état est retourné task_details met à jour les données en modifiant l'état «Started» par le nouveau état du tâche et l'envoyer au service du frontend en fomat JSON.

- **Sinon (L'état sera «suspended») :** le contrôleur va retourner directement les données du tâche au service de frontend sans faire aucune modification.

3.2.2.5 Diagramme de séquence du cas d'utilisation «Créer un compte» :

La figure 3.9 illustre le diagramme de séquence du cas d'utilisation «Créer un compte» :

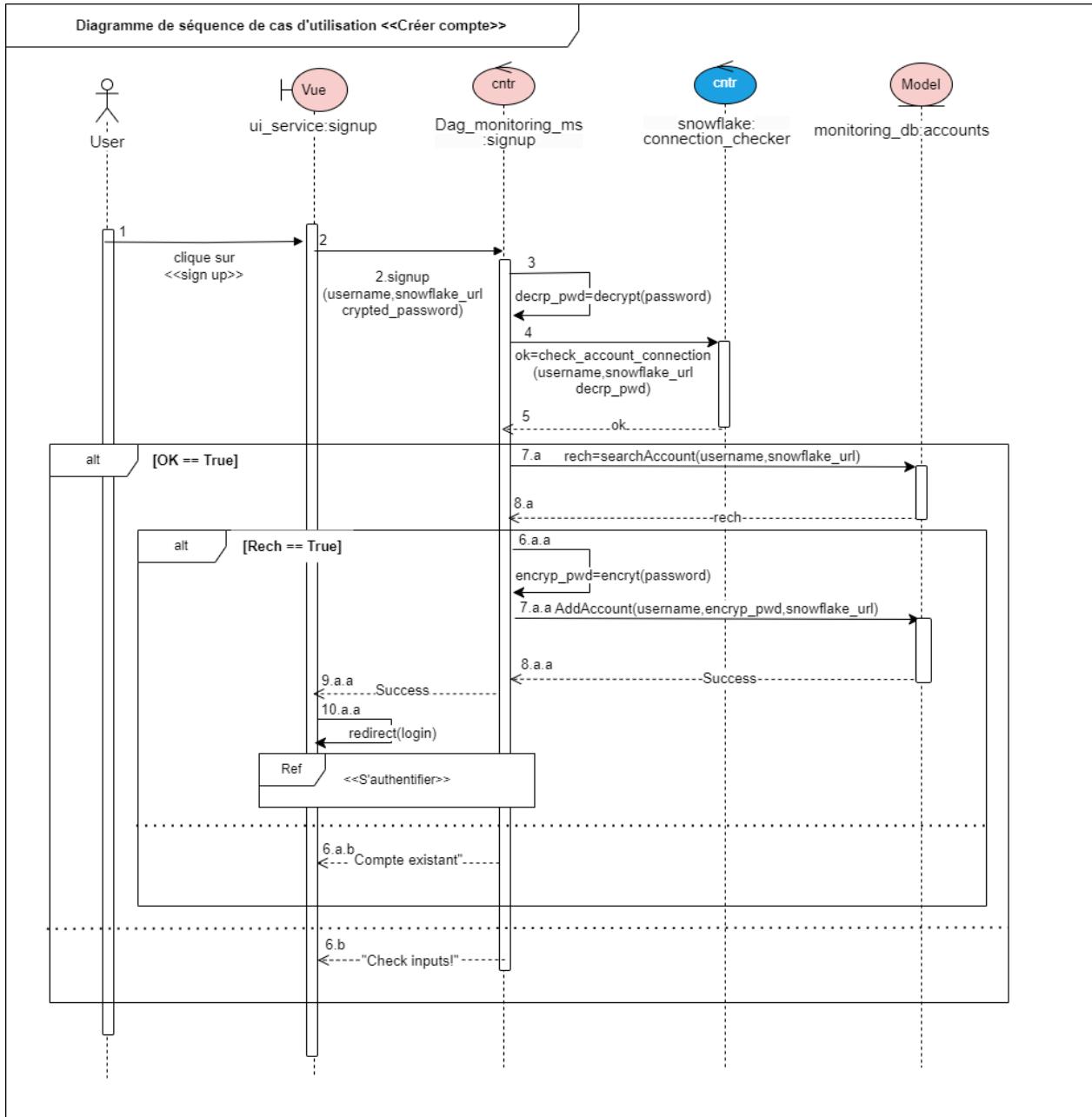


FIGURE 3.9 : Diagrammme de séquence de cas d'utilisation «Créer un compte»

Afin de pouvoir inscrire dans «Snowflake Monitoring Application», l'utilisateur accède à l'interface de création de compte de l'application en saisissant son username, mot de passe, ainsi que l'URL d'accès à son compte Snowflake.

En cliquant sur «signup», le contrôleur signup_ctrl récupère les données saisies du frontend et décrypte le mot de passe.

Ensuite, il émettre un appel au contrôleur de snowflake pour que ce dernier vérifie l'existance de ce compte dans Snowflake en retournant une réponse vers le contrôleur principal.

- [ok == True] :

-le contrôleur lance une recherche dans la table de monitoring_DB **accounts**

-si l'**account existe**, le contrôleur renvoie une erreur au frontend.

-sinon, le contrôleur va encrypter le mot de passe et crée un compte dans la table **accounts** avec les données saisies.

Enfin, il va renvoyer un message de succès au frontend qui va à son tour rediger l'utilisateur vers la page de l'authentification.

- [Sinon] :

-Le contrôleur **renvoie** un message d'erreur au frontend en indiquant que les données **saisites n'appartient** à aucun compte Snowflake.

3.2.2.6 Diagramme de séquence du cas d'utilisation «S'authentifier» :

La figure 3.10 illustre le diagramme de séquence du cas d'utilisation «S'authentifier» :

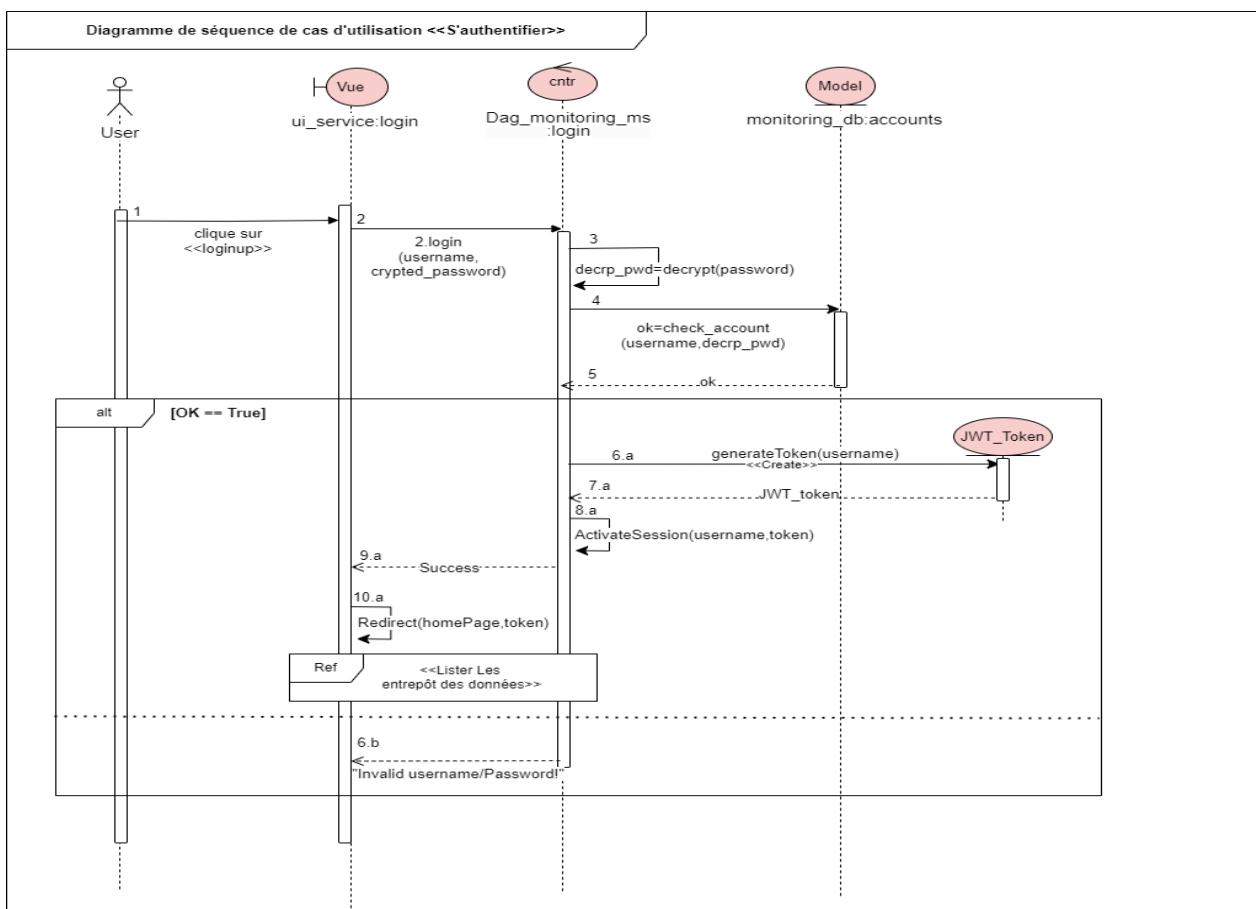


FIGURE 3.10 : Diagramme de séquence du cas d'utilisation «S'authentifier»

Pour accéder à l'application, l'utilisateur accède à l'interface de «login» en **saisissant** les données

d'authentification (username,mot de passe).

En cliquant sur «login», le contrôleur login_ctrl récupère les données saisies du frontend et décrypte le mot de passe saisi.

Ensuite, il lance une recherche dans la table **Accounts** pour vérifier l'existance du compte ;

- [ok == True] : cela signifie que le compte existe déjà, et là le contrôleur crée une instance de l'objet **JWT Token**, qui est le moyen d'accès à toutes les informations et parties de l'application. Une fois l'instance est créée et retourné le contrôleur active la session de connexion de l'utilisateur et retourne la token ainsi que un message de succès au frontend.
Finalement, le frontend redirige l'utilisateur vers la page d'accueil de l'application.
- [Sinon (compte inexistant)] : le contrôleur renvoie un message d'erreur vers le frontend en indiquant que les données saisites sont invalides.

Conclusion

Ce chapitre a établi les bases pour la mise en œuvre technique de notre projet. Les choix architecturaux et conceptuels s'avèrent essentiels pour assurer la stabilité et la performance de notre solution. Dans le prochain chapitre, nous plongerons dans la réalisation concrète de notre projet.

CHAPITRE 4

RÉALISATION

Plan

Introduction	43
1 Choix technologiques	43
2 Réalisation	48
Conclusion	68

Introduction

Le quatrième chapitre se consacre à la mise en œuvre concrète du projet. Nous explorons les choix techniques que nous avons adoptés et détaillons le travail effectué pour donner vie à notre solution.

4.1 Choix technologiques

Dans cette partie, nous explorerons les décisions stratégiques que nous avons prises concernant les technologies et les outils que nous avons choisis pour la réalisation de notre solution.

4.1.1 Frameworks

- **FastAPI :**

FastAPI est un framework web moderne et rapide (à haute performance) pour la création d'API avec Python, basé sur les annotations de type standard de Python[18].



FIGURE 4.1 :
FastAPI logo [18]

Dans la réalisation de ce projet, nous avons opté pour le choix de FastAPI comme notre backend framework vue ces performances extrêmement élevées, comparables à celles de NodeJS et Go, grâce à l'utilisation de Starlette et Pydantic. FastAPI est l'un des frameworks Python les plus rapides.

Sa Robustesse aussi, car il permet de produire un code prêt pour la production avec une documentation interactive et automatique.

- **React :**

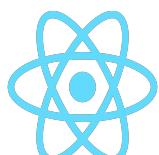


FIGURE 4.2 :
React logo [19]

React est une bibliothèque JavaScript destinée à la création d'interfaces utilisateur (UI) [19].

Dans la réalisation de ce projet, nous avons opté pour React comme étant notre frontend framework car il permet de créer des interfaces utilisateur dynamiques et réactives avec une efficacité accrue grâce à sa gestion intelligente des mises à jour via le DOM virtuel.

4.1.2 Bases de données

- **PostgreSQL :**

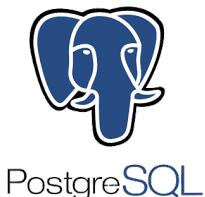


FIGURE 4.3 :
PostgreSQL logo [20]

PostgreSQL est un puissant système de base de données relationnelle objet open source, développé activement depuis plus de 35 ans, qui lui a valu une solide réputation en termes de fiabilité, de robustesse des fonctionnalités et de performances[20].

Dans la réalisation de ce projet, nous avons opté pour le choix de PostgreSQL comme étant notre SGBD vue qu'il est souvent perçu comme un système plus fiable et plus robuste que MySQL pour gérer de grands volumes de données sans risque de corruption.

De plus, il permet de stocker des informations de manière plus structurée grâce à une meilleure gestion des relations et des contraintes.

4.1.3 Outils de communication

- **MQTT :**



FIGURE 4.4 :
MQTT logo [21]

MQTT est un protocole de messagerie basé sur des normes, ou un ensemble de règles, utilisé pour la communication de machine à machine [21].

Dans la réalisation de ce projet, nous avons opté pour MQTT comme étant un système de notification car il MQTT facilite le chiffrement des messages via TLS et l'authentification des clients à l'aide de protocoles d'authentification modernes, tels que OAuth.

De plus, sa Légèreté et efficacité car les clients MQTT sont très petits, ils nécessitent peu de ressources. Les en-têtes des messages MQTT sont de petite taille afin d'optimiser la bande passante du réseau.

- **RESTful APIs :**



FIGURE 4.5 :
REST logo [22]

REST (REpresentational State Transfer) constitue un style architectural et un mode de communication fréquemment utilisé dans le développement de services Web [22].

Dans la réalisation de ce projet, nous avons opté pour RESTful API comme étant le mode de communication entre les différents microservices du système car ils sont faciles à utiliser avec la plupart des outils.

De plus REST s'impose progressivement comme le standard pour l'interaction entre systèmes.

4.1.4 Bibliothèques pour l'analyse et la manipulation des données

- **Snowflake-connector-python :**



FIGURE 4.6 :
Snowflake logo [23]

Le connecteur Snowflake pour Python fournit une interface pour le développement d'applications Python qui peuvent se connecter à Snowflake et effectuer toutes les opérations standard.[23].

Il a été conçu pour gérer les ressources principales de Snowflake, y compris les bases de données, les schémas, les tables, les tâches et les entrepôts, sans utiliser le langage SQL.

- **psycopg :**



FIGURE 4.7 :
Psycopg logo [24]

Psycopg est l'adaptateur de base de données PostgreSQL le plus populaire pour le langage de programmation Python. Il est écrit en C et fournit un moyen d'effectuer toute la gamme des opérations SQL sur les bases de données PostgreSQL[24].

Il a été conçu pour les applications fortement multithreadées qui créent et détruisent de nombreux curseurs et effectuent un grand nombre d'« INSERT « s ou d » »UPDATE « s simultanés. C'est le principal raison pour lequel nous avons choisi cet librairie pour toute interaction avec postgresql.

- **Pandas :**



FIGURE 4.8 :
Pandas logo [25]

Pandas est une bibliothèque de manipulation de données Python qui fournit des structures de données flexibles pour l'analyse des données[25].

Il a été conçu pour faciliter l'importation, la manipulation et l'analyse des données, ce qui en fait un choix idéal pour notre projet.

- **GMQTT :**

Gmqt is une bibliothèque de broker MQTT flexible et performante qui implémente entièrement le protocole MQTT V3.x et V5 en goLang [26].

4.1.5 Bibliothèques d'interaction avec le système d'exploitation

- **OS :**

Un module intégré à Python qui permet d'interagir avec le système d'exploitation sous-jacent sur lequel Python est exécuté. Il peut être utilisé pour manipuler le système de fichiers, gérer

les variables d'environnement, contrôler les processus, et bien plus encore[27].

- **Scheduler :**

C'est une bibliothèque simple d'ordonnancement en python avec asyncio, threading ainsi que la prise en charge des fuseaux horaires.

Elle permet de planifiez des tâches en fonction de leurs cycles temporels, d'heures fixes, de jours de la semaine, de dates, de poids, de décalages et de comptes d'exécution, et automatisez les tâches[28].

- **Requests :**

est un standard Python permettant d'effectuer des requêtes HTTP. Elle dissimule les subtilités des requêtes derrière une API simple, ce qui vous permet de vous concentrer sur l'interaction avec les services et la consommation de données dans votre application[29].

4.1.6 Outils de test

- **Postman :**



FIGURE 4.9 :
Postman logo [30]

Postman agit en tant que client et constitue un excellent outil pour tester des API RESTful. Il offre une interface utilisateur élégante pour effectuer des requêtes HTTP, sans avoir à écrire beaucoup de code uniquement pour tester les fonctionnalités d'une API.

Postman permet également d'analyser les réponses et de visualiser clairement les en-têtes, le corps et le statut HTTP [30].

- **MQTTX :**



FIGURE 4.10 :
MQTTX logo [31]

MQTTX est un client de bureau MQTT 5.0 open-source et multiplateforme initialement développé par EMQ, qui peut fonctionner sur n'importe quel système d'exploitation.

il rend le développement et le test d'applications MQTT plus rapides et plus faciles[31].

4.1.7 Outils de versionning

- Git :



FIGURE 4.11 :
Git logo [32]

Git est, de loin, le système de contrôle de version le plus largement utilisé aujourd’hui. C’est un projet open source avancé, activement maintenu.

Grâce à sa structure décentralisée, Git illustre parfaitement ce qu’est un système de contrôle de version décentralisé (DVCS)[32].

- Github :



FIGURE 4.12 :
Github logo [33]

GitHub est un service d’hébergement Open-Source, permettant aux programmeurs et aux développeurs de partager le code informatique de leurs projets afin de travailler dessus de façon collaborative. On peut le considérer comme un Cloud dédié au code informatique[33].

4.1.8 Outils de documentation

- Swagger :



FIGURE 4.13 :
Swagger logo [34]

Swagger est un outil spécialisé qui génère automatiquement la documentation des API RESTful de votre application.

Son principal avantage réside dans la possibilité de consulter tous les points de terminaison de l’application et de les tester immédiatement en envoyant des requêtes et en recevant des réponses[34].

- Latex :



FIGURE 4.14 :
Latex logo [35]

LaTeX est à la fois un langage de balisage et un logiciel de composition de documents, largement utilisé par la communauté scientifique.

Il facilite la rédaction de documents volumineux comme les mémoires et les thèses[35].

- Draw.io :



FIGURE 4.15 :
Draw.io logo [36]

Draw.io est une application en ligne gratuite, accessible via un navigateur web (protocole https), permettant de dessiner des diagrammes et des organigrammes.

Cet outil offre la possibilité de concevoir toutes sortes de diagrammes et de dessins vectoriels, de les enregistrer au format XML, puis de les exporter [36].

4.2 Réalisation

Dans cette section, nous allons présenter la réalisation de ce projet en illustrant chaque partie avec des captures explicatifs.

4.2.1 Micro-service « Authentification_Service »

Le micro-service « Authentification_Service » est responsable du système d'authentification de notre application

- **La liste des APIs :**

La liste des APIs présents dans le micro-service d'authentification, documentée par Swagger, sont représentés par la figure 4.16 suivante :

The screenshot shows the FastAPI Swagger UI interface. At the top, it displays "FastAPI 0.1.0 OAS 3.1" and a link to "/openapi.json". Below this, there's a "default" section with three POST methods:

- /signup Signup
- /token Token
- /logout Logout

Below the API list is a "Schemas" section containing definitions for error and validation models:

- Body_signup_signup_post > Expand all object
- Body_token_token_post > Expand all object
- HTTPValidationError > Expand all object
- ValidationError > Expand all object

FIGURE 4.16 : Liste des APIs du microservice « Athentification_service »

- **Interface de Création de compte «Sign up» :**

L'interface de « Sign up» est représentée par la figure 4.17 suivante :

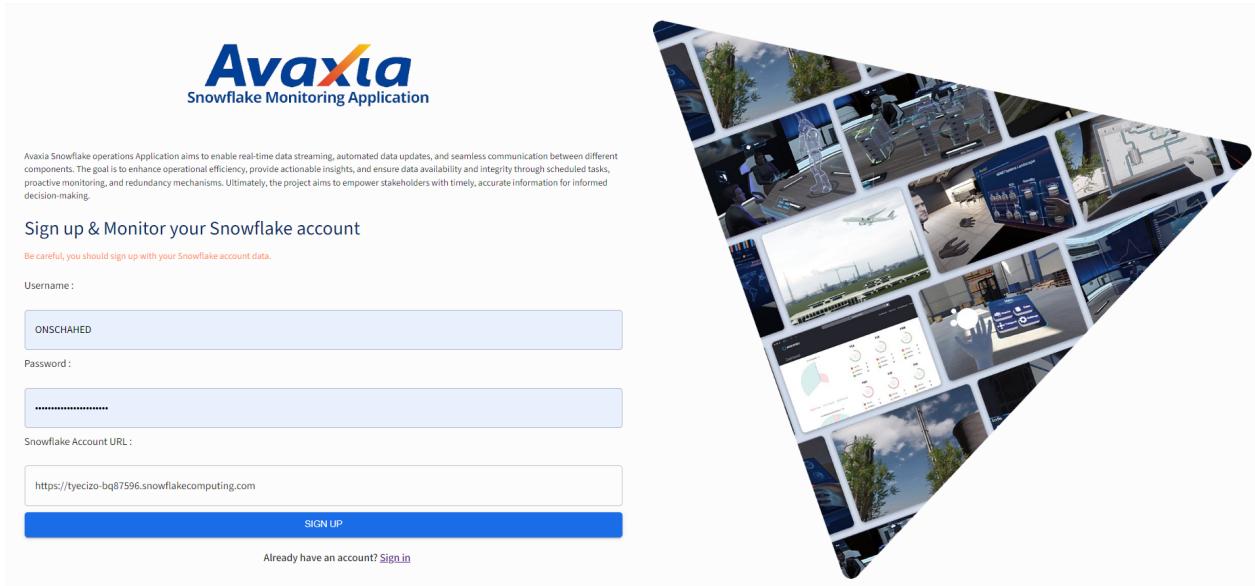


FIGURE 4.17 : Interface de Création de compte «Sign up»

Afin de créer un compte, l'utilisateur accède à cette interface en remplittant un formulaire d'inscription avec les données spécifiques à son compte Snowflake (username,mot de passe,l'url de connexion de son compte snowflake). Une fois l'utilisateur clique sur le bouton «SINGUP», si les données sont valides : le système redirige l'utilisateur vers la page d'authentification. Sinon, il relance cette interface.

- **Interface d'authentification «Login» :**

L'interface de «Login» est représentée par la figure 4.18 suivante :

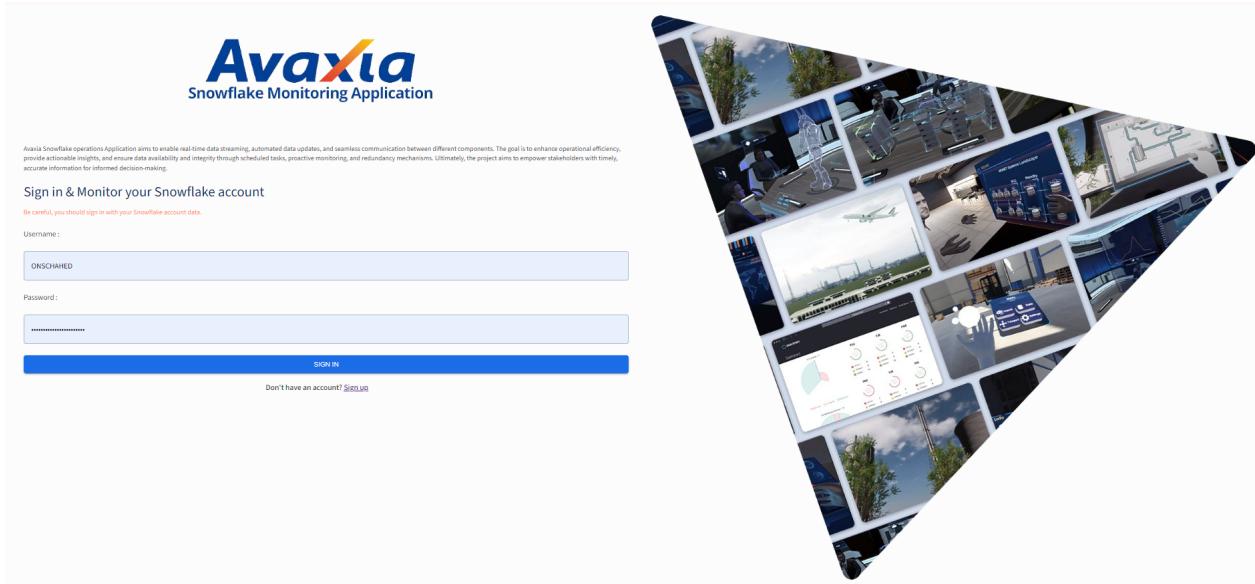


FIGURE 4.18 : Interface d'authentification «Login»

Pour accéder à son compte, l'utilisateur remplit le formulaire avec son «username» et «mot de passe». Une fois les données sont valides, le système attribut une token d'accès à cet utilisateur et le redirige vers la page d'accueil. Sinon, il relance la page de «login» en indiquant l'erreur. Si l'utilisateur est authentifié et clique sur le bouton «Logout», indiqué dans la figure 4.19, il est redirigé vers l'interface de login automatiquement.

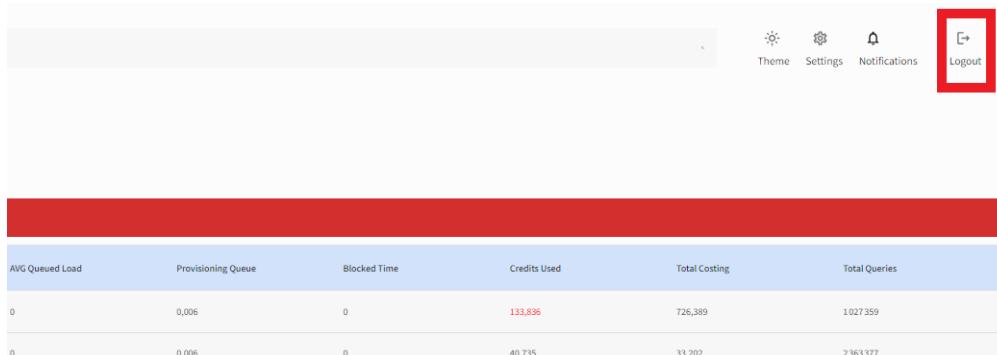


FIGURE 4.19 : «Logout»

4.2.2 Micro-service « Warehouse_Monitoring »

Le micro-service « Monitoring_Service » est le service chargé de la surveillance de l'utilisation des entrepôts de données Snowflake.

- **La liste des APIs :**

La liste des APIs présents dans ce micro-service, documentée par Swagger, sont représentés par la figure 4.20 suivante :

The screenshot shows the FastAPI documentation interface for a version 0.1.0 API. At the top, there's a header with the FastAPI logo, version 0.1.0, and OAS 3.1. Below the header, there's a link to 'openapi.json' and a green 'Authorize' button with a lock icon. The main content area is titled 'default'. It lists numerous GET requests for various endpoints, each with a brief description. The endpoints include '/warehouses/all', '/warehouses/{warehouse_id}/all/events', '/warehouses/credits', '/warehouses/{warehouse_id}/credits', '/warehouses/credits/date', '/warehouses/queries_cost', '/warehouses/{warehouse_id}/queries_cost', '/warehouses/events', '/warehouses/{warehouse_id}/events', '/warehouses/{warehouse_id}/events_perday', '/warehouses/events_perday', '/warehouses/events_kpi', '/warehouses/{warehouse_id}/events_kpi', '/warehouses/KPI', '/warehouses/{warehouse_id}/KPI', '/warehouses/{username}/{role}/access', and '/warehouses/{warehouse_id}/{username}/{role}/access'. Each endpoint entry has a lock icon and a dropdown arrow to its right.

FIGURE 4.20 : Liste des APIs du microservice «Warehouse_Monitoring»

- **Interface de la liste des entrepôts de données :**

L'interface de la liste des entrepôts de données est représentée par la figure 4.21 suivante :

Chapitre 4. Réalisation

Warehouse ID	Warehouse Name	Avg Running	Avg Queued Load	Provisioning Queue	Blocked Time	Credits Used	Total Costing	Total Queries
4	MONITORING_WAREHOUSE	130,28	0	0,006	0	133,836	726,389	1027359
1	COMPUTE_WH	32,727	0	0,006	0	40,735	33,202	2363377

FIGURE 4.21 : Interface de la liste des entrepôts de données

Le tableau présente des informations détaillées sur les différents entrepôts surveillés, notamment leur ID, leur nom, leurs métriques de performance (débit moyen, charge moyenne, file d'attente de provisionnement, temps bloqué, crédits utilisés, coût total) ainsi que le nombre total de requêtes.

- **Tableau de bord du surveillance des entrepôts des données**

Les deux tableaux de bord du surveillance des entrepôts des données «Monitoring_warehouse» et «Compute_WH» sont représentées par les deux figures 4.22 et 4.23 :

Warehouse ID	Warehouse Name	Event Name	Event Date	Event reason	Event State
4	MONITORING_WAREHOUSE	RESUME_WAREHOUSE	16/05/2024	WAREHOUSE_AUTORESU...	STARTED
4	MONITORING_WAREHOUSE	RESUME_WAREHOUSE	16/05/2024	WAREHOUSE_AUTORESU...	STARTED
4	MONITORING_WAREHOUSE	RESUME_WAREHOUSE	16/05/2024	WAREHOUSE_AUTORESU...	STARTED
4	MONITORING_WAREHOUSE	RESUME_WAREHOUSE	16/05/2024	WAREHOUSE_AUTORESU...	STARTED

Warehouse ID	Query ID	Access Time	Database Accessed	By	With the role
4	D1D4b388-00...	28/05/2024	MONITORING_DB	SYSTEM	ACCOUNTADMIN
4	D1D4c292-00...	28/05/2024	MONITORING_DB	SYSTEM	ACCOUNTADMIN
4	D1D4e37b-00...	28/05/2024	MONITORING_DB	SYSTEM	ACCOUNTADMIN
4	D1D4f445-00...	28/05/2024	MONITORING_DB	SYSTEM	ACCOUNTADMIN

FIGURE 4.22 : Tableau de bord du surveillance d l'entrepôts des données «Monitoring_warehouse»

Chapitre 4. Réalisation

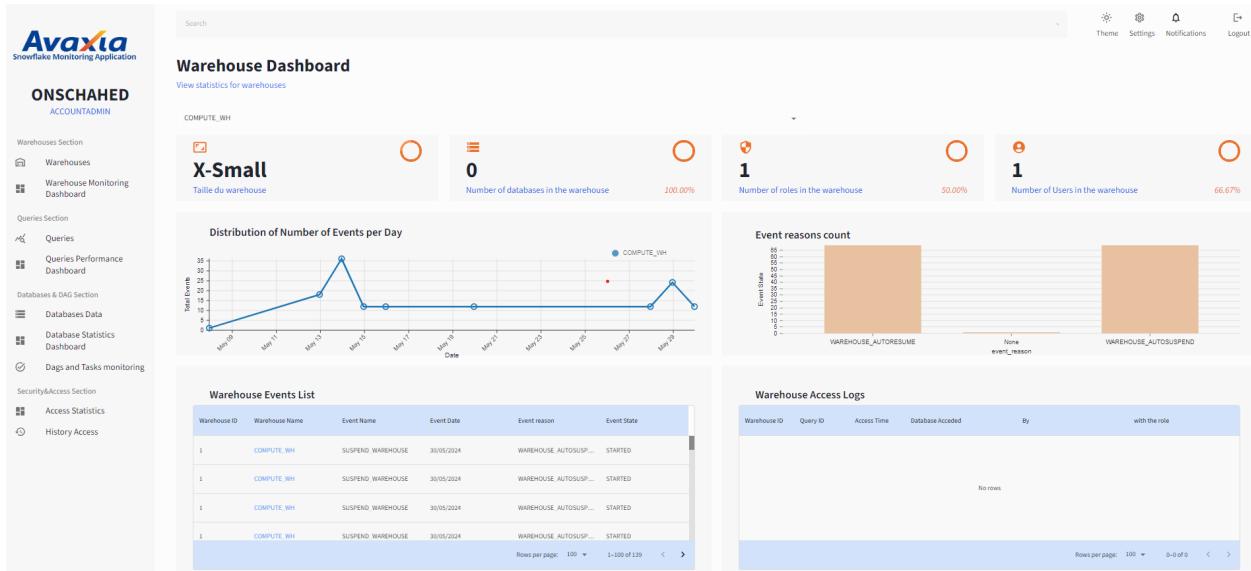


FIGURE 4.23 : Tableau de bord du surveillance d l'entrepôts des données «Compute_WH»

Ces tableaux de bord des entrepôts Snowflake offrent un aperçu complet des activités et de l'utilisation de l'infrastructure de données. Ils présentent des informations clés sur la taille, la composition et l'utilisation des entrepôts, notamment le nombre de bases de données, nombre des evenements effectuées dans l'entrepôt ainsi que le nombre de rôles et d'utilisateurs.

• Changement des variables globaux

Le changement des paramètres globaux peut reflecter les vues de ce micro service une fenêtre s'affiche lorsque on clique sur le bouton «Settings» du topbar. Selon les besoins de l'utilisateur/administrateur

- Changement du rôle d'utilisateur/Administrateur :

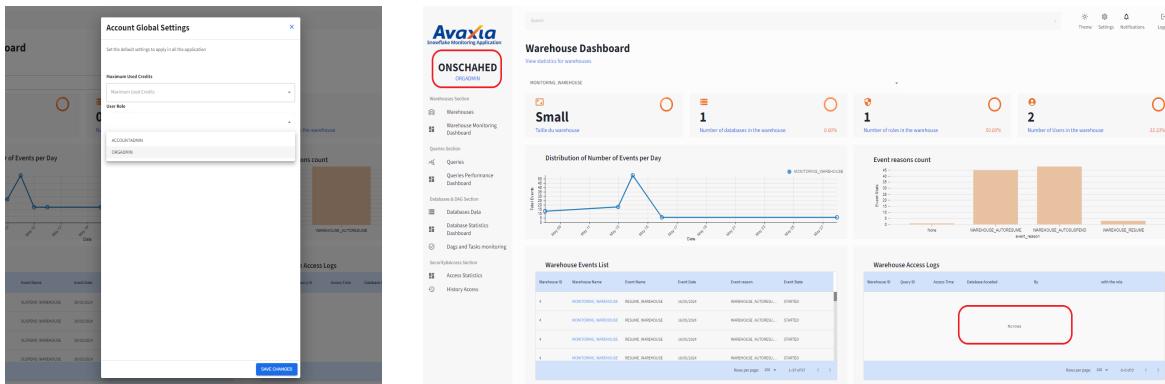


FIGURE 4.24 : Changement du role de l'utilisateur de «ACCOUNT ADMIN» vers «ORGADMIN»

Lorsque l'utilisateur modifit les paramètres globaux, telque le rôle de compte utilisateur, les journaux d'accès aux entrepôts ne sont plus affichés dans cette interface. Cela signifie

que les administrateurs doivent être conscients que certaines informations peuvent ne pas être visibles dans certaines configurations.

- **Changement du seuil maximum des crédits d'utilisation :**

Warehouse ID	Warehouse Name	Avg Running	Avg Queued Load	Provisioning Queue	Blocked Time	Credits Used	Total Costing	Total Queries
4	MONITORING_WAREHOUSE	130,28	0	0,00%	0	111,038	75,309	102739
1	COMPUTE_WH	32,727	0	0,00%	0	40,715	35,262	236337

FIGURE 4.25 : Changement de seuil de credits vers «90\$»

Lorsque l'utilisateur modifie les paramètres globaux, notamment le seuil de crédits maximum utilisés, une notification permanente devrait être affichée si les crédits utilisées de l'un des entrepôts de données dépasse ce seuil pour informer l'utilisateur de cette malveillance.

Dans l'ensemble, cet microservice offre une visibilité essentielle sur les performances et l'utilisation des entrepôts, facilitant la prise de décisions éclairées pour l'infrastructure de données.

4.2.3 Micro-service « Query_Performance »

Le micro-service « Monitoring_Service » est le service chargé de la surveillance des requêtes SQL effectuées au sein du compte Snowflake.

- **La liste des APIs :**

La liste des APIs présents dans ce micro-service, documentée par Swagger, sont représentés par la figure 4.26 suivante :



The screenshot shows the FastAPI OpenAPI UI for the 'default' API endpoint. The page has a header with the FastAPI logo and 'openapi.json'. On the right, there is a green button labeled 'Authentique' with a lock icon. The main content area is titled 'default' and contains a list of API endpoints. Each endpoint is shown in a blue-bordered box with a 'Get' method, a URL path, and a brief description. Most endpoints have a dropdown arrow icon to the right. The list includes:

- Get /queries/all: Retrieve all queries.
- Get /queries/all/{warehouse_id}: Retrieve all queries in a specific warehouse.
- Get /{warehouse_id}/queries/total/: Retrieve the count of queries.
- Get /queries/total: Retrieve the count of queries.
- Get /queries/per_type: Retrieve queries per type.
- Get /{warehouse_id}/queries/per_type: Retrieve queries per type.
- Get /{warehouse_id}/queries/execution_time: Retrieve the execution time KPIs.
- Get /queries/execution_time: Retrieve the execution time KPIs.
- Get /queries/failed: Retrieve the failed queries.
- Get /{warehouse_id}/queries/failed: Retrieve the failed queries.
- Get /queries/by_error: Retrieve queries per error.
- Get /queries/compilation_time: Retrieve the compilation time KPIs.
- Get /{warehouse_id}/queries/compilation_time: Retrieve the compilation time KPIs.
- Get /queries/per_day: Retrieve queries per day.
- Get /{warehouse_id}/queries/avg_execution_time_by_type: Retrieve the avg execution time per type of query.
- Get /queries/avg_execution_time_by_type: Retrieve the avg execution time per type of query.
- Get /queries/per_user: Retrieve queries per user.
- Get /queries/cost: Retrieve the costing data.
- Get /queries/date: Retrieve the queries on a specific date.
- Get /queries/type: Retrieve the queries of a specific type.
- Get /queries/user: Retrieve the queries of a specific user.
- Get /{warehouse_id}/queries/credits/: Retrieve the credits of a specific warehouse.
- Get /queries/credits/: Retrieve all the credits.
- Get /queries/{session}/{username}/all: All Queries Session.
- Get /queries/{username}/cost: All Queries Cost User.
- Get /queries/{username}/all_cost: All Queries Session Cost.
- Get /queries/{username}/types: All Queries Types Cost.

FIGURE 4.26 : Liste des APIs du microservice «Warehouse Monitoring»

- **Interface de historique des requêtes de l'utilisateur :**

L'interface de la liste des entrepôts de données est représentée par la figure 4.27 suivante :

Chapitre 4. Réalisation

FIGURE 4.27 : Interface de la liste des entrepôts de données

Cette vue représente l'historique des requêtes dans l'application de surveillance Snowflake. Elle fournit un aperçu détaillé de chaque requête exécutée.

Cet historique permet aux administrateurs de surveiller en détail les requêtes exécutées afin de pouvoir identifier les éventuels problèmes liés aux ces derniers dans le compte Snowflake en question.

- **Tableaux de bord du surveillance des Requêtes SQL**

- **Les informations générale sur la totalité des requêtes dans le compte de l'utilisateur :**

La figure 4.28 illustre le tableau de bord global des requêtes SQL du compte Snowflake :

Chapitre 4. Réalisation

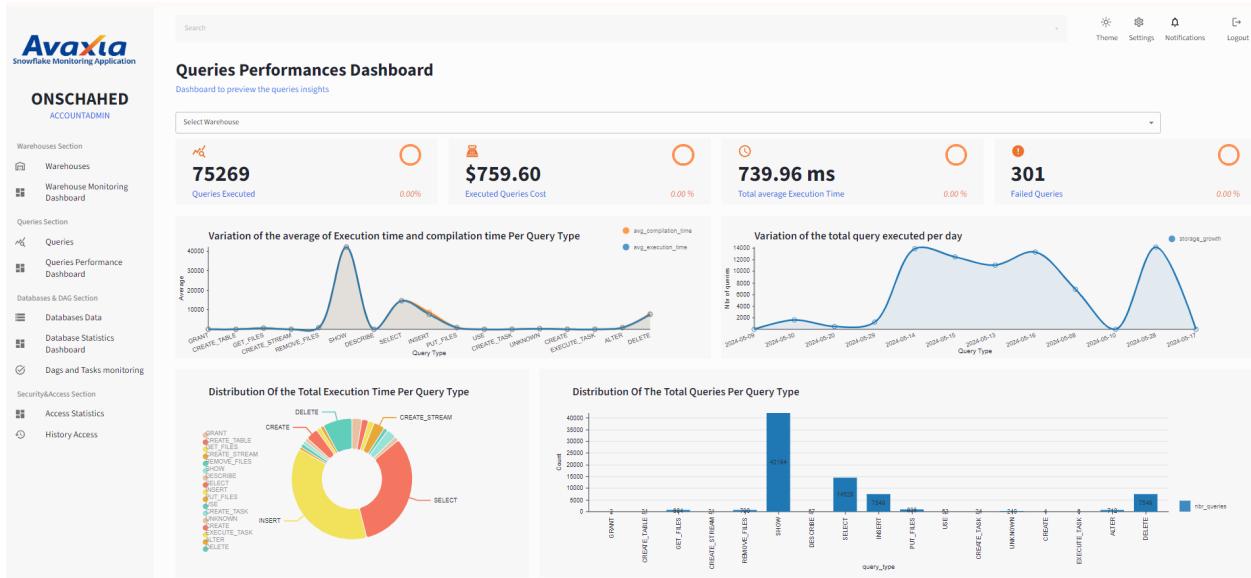


FIGURE 4.28 : Tableau de bord du surveillance des entrepôts des données

- Selection de l'entrepôts des données appriori :

Une fois l'utilisateur selectionne un entrepôt de données, depuis la liste de ces entrepôts, qui s'affiche en cliquant sur la liste déroulante «select warehouse» comme l'indique la figure 4.29, les données spécifique à cet entrepôt vont être affichées.

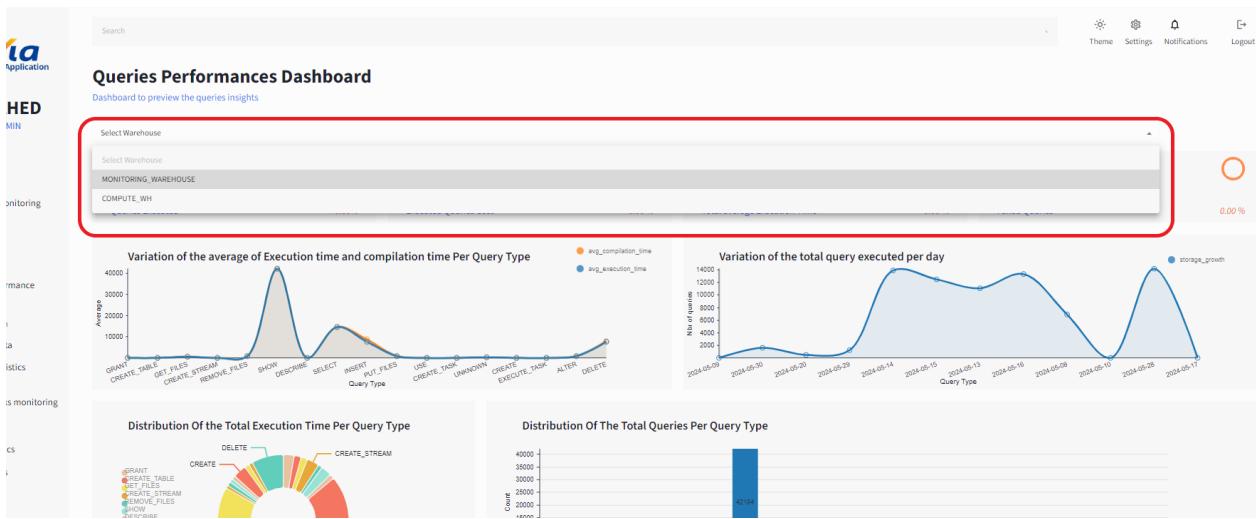


FIGURE 4.29 : Tableau de bord du surveillance des entrepôts des données

La figure 4.30 illustre le tableau de bord de l'entrepôt «Monitoring_Warehouse» :

Chapitre 4. Réalisation

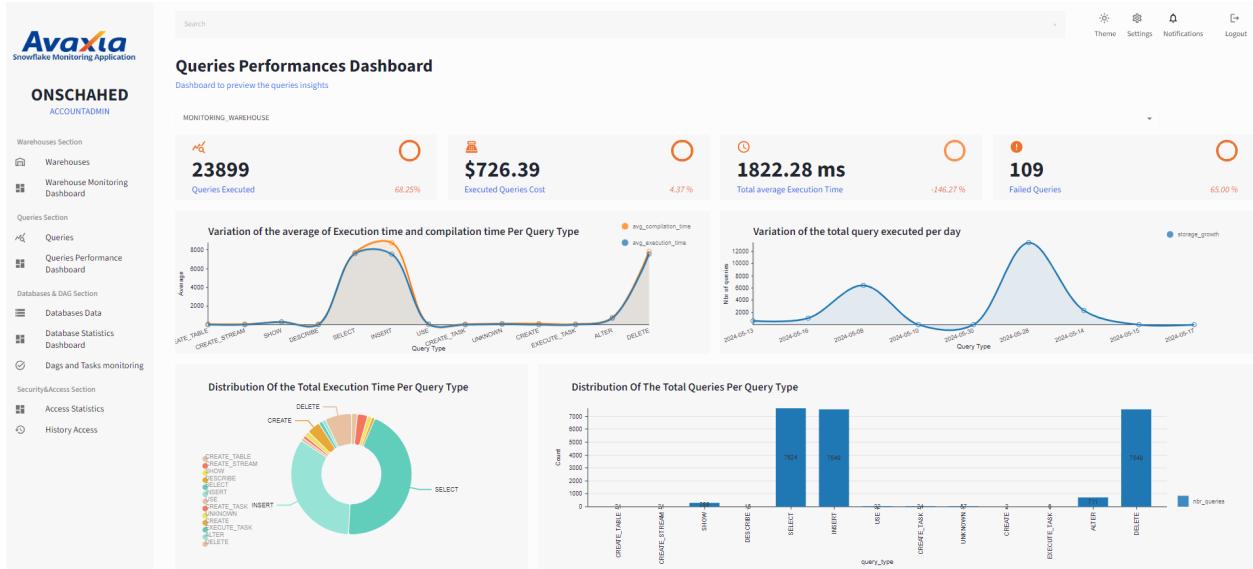


FIGURE 4.30 : Tableau de bord du surveillance de l'entrepôt des données «Monitoring_Warehouse»

La figure 4.31 illustre le tableau de bord de l'entrepôt «Compute_WH» :



FIGURE 4.31 : Tableau de bord du surveillance de l'entrepôt des données «Compute_WH»

ces tableaux de bord présentent des indicateurs clés sur les requêtes exécutées, tels que le nombre total de requêtes, le coût total, le temps d'exécution moyen et le taux de requêtes en échec. Cette vision synthétique permet aux administrateurs d'avoir une compréhension globale des performances.

Les graphiques complémentaires apportent une analyse approfondie des tendances par type de requête. Ils montrent la variation du temps d'exécution et de compilation, ainsi

que la répartition du nombre total de requêtes et du temps d'exécution par type de requête. Cette granularité permet d'identifier les domaines à optimiser pour améliorer l'efficacité du système.

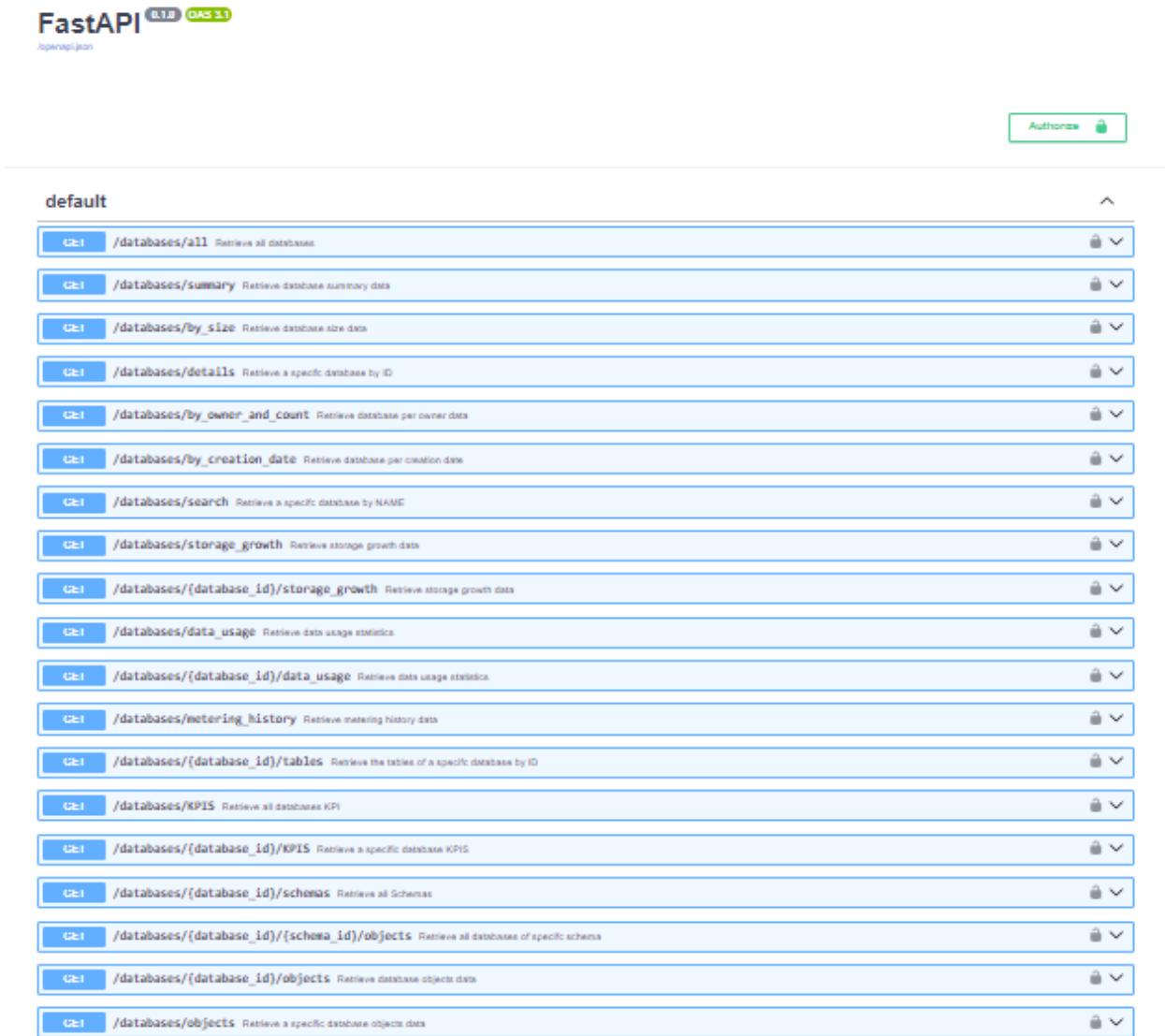
Le service « Query_Performance » constitue un outil essentiel pour les équipes en charge de l'exploitation et de l'optimisation de l'infrastructure Snowflake. Il offre une visibilité complète sur les activités de requêtes, permettant de prendre des décisions éclairées afin d'améliorer la fiabilité, les performances et la rentabilité globale du système.

4.2.4 Micro-service « Databases_Statistics »

Le micro-service « Databases_Statistics » est le service chargé de la surveillance des bases de données créées au sein du compte Snowflake.

- **La liste des APIs :**

La liste des APIs présents dans ce micro-service, documentée par Swagger, sont représentés par la figure **4.32** suivante :



Method	Path	Description	Auth
Get	/databases/all	Retrieve all databases	🔒
Get	/databases/summary	Retrieve database summary data	🔒
Get	/databases/by_size	Retrieve database size data	🔒
Get	/databases/details	Retrieve a specific database by ID	🔒
Get	/databases/by_owner_and_count	Retrieve database per owner data	🔒
Get	/databases/by_creation_date	Retrieve database per creation date	🔒
Get	/databases/search	Retrieve a specific database by NAME	🔒
Get	/databases/storage_growth	Retrieve storage growth data	🔒
Get	/databases/{database_id}/storage_growth	Retrieve storage growth data	🔒
Get	/databases/data_usage	Retrieve data usage statistics	🔒
Get	/databases/{database_id}/data_usage	Retrieve data usage statistics	🔒
Get	/databases/metering_history	Retrieve metering history data	🔒
Get	/databases/{database_id}/tables	Review the tables of a specific database by ID	🔒
Get	/databases/KPIs	Retrieve all databases KPI	🔒
Get	/databases/{database_id}/KPIs	Retrieve a specific database KPIs	🔒
Get	/databases/{database_id}/schemas	Retrieve all Schemas	🔒
Get	/databases/{database_id}/{schema_id}/objects	Retrieve all databases of specific schema	🔒
Get	/databases/{database_id}/objects	Retrieve database objects data	🔒
Get	/databases/objects	Retrieve a specific database object data	🔒

FIGURE 4.32 : Liste des APIs du microservice « Databases_ Statistics »

- La liste des bases de données et ces différents schémas :

L'interface de la liste des bases de données et ces différents schémas est représentée par la figure 4.33 suivante :

Chapitre 4. Réalisation

Select	Database ID	Database Name	Database Owner	Created On	Last Altered On	Type
<input type="checkbox"/>	4	SNOWFLAKE_SAMPLE_DATA	ACCOUNTADMIN	2024-05-08 05:31:27	2024-05-08 05:31:27	IMPORTED DATABASE
<input type="checkbox"/>	1	SNOWFLAKE	None	2024-05-08 05:31:22	2024-05-29 01:50:07	APPLICATION
<input type="checkbox"/>	7	MONITORING_DB	ACCOUNTADMIN	2024-05-08 08:45:07	2024-05-08 08:45:07	STANDARD

Select	Schema ID	Table Name	Last DDL By	Last Altered	Owner
No rows					

Select	Object ID	Object Name	Object Domain	Referencing Database	Referencing Sc...	Referencing O...	Referencing O...	Dependency T...
No rows								

FIGURE 4.33 : La liste des bases de données

La section "List of Databases" affiche les détails des différentes bases de données, y compris leur ID, leur nom, leur propriétaire et leurs dates de création et de dernière modification.

Une fois l'utilisateur selectionne une base de données, La liste des schémas d'affiche comme l'indique la figure 4.34 suivante :

Select	Schema ID	Table Name	Last DDL By	Last Altered	Owner
<input type="checkbox"/>	3	MONITORING_SH	2024-05-08 08:45:08	2024-05-08 08:45:08	ACCOUNTADMIN
<input type="checkbox"/>	2	PUBLIC	2024-05-08 08:45:07	2024-05-08 08:45:07	ACCOUNTADMIN

Select	Object ID	Object Name	Object Domain	Referencing Database	Referencing Sc...	Referencing O...	Referencing O...	Dependency T...
No rows								

FIGURE 4.34 : Liste des schémas de base de données

Ensuite, pour qu'on puisse lister les objets présents et créées dans cette base de données, l'utilisateur selectionne le schéma appriori pour lister ces objets.

La figure 4.36 ilusstre cette action :

Chapitre 4. Réalisation

Databases Informations

Retrieve all the database's information here.

List of Databases

Select	Database ID	Database Name	Database Owner	Created On	Last Altered On	Type
<input type="checkbox"/>	4	SNOWFLAKE_SAMPLE_DATA	ACCOUNTADMIN	2024-05-08 05:31:27	2024-05-08 05:31:27	IMPORTED DATABASE
<input type="checkbox"/>	1	SNOWFLAKE	None	2024-05-08 05:31:22	2024-05-29 01:50:07	APPLICATION
<input checked="" type="checkbox"/>	7	MONITORING_DB	ACCOUNTADMIN	2024-05-08 08:45:07	2024-05-08 08:45:07	STANDARD

1 row selected

Rows per page: 100 ▾ 1-3 of 3 < >

List of Schemas

Select	Schema ID	Table Name	Last DDL By	Last Altered	Owner
<input checked="" type="checkbox"/>	3	MONITORING_SCHEMA	2024-05-08 08:45:08	2024-05-08 08:45:08	ACCOUNTADMIN
<input type="checkbox"/>	2	PUBLIC	2024-05-08 08:45:07	2024-05-08 08:45:07	ACCOUNTADMIN

1 row selected

Rows per page: 100 ▾ 1-2 of 2 < >

List of Objects

Object ID	Object Name	Object Domain	Referencing Database	Referencing S...	Referencing O...	Referencing O... Dependency T...
8	WAREHOUSE, METERING, HISTORY	TABLE	MONITORING_DB	MONITORIN...	WAREHOUS...	STREAM BY ID
16	TABLES	TABLE	MONITORING_DB	MONITORIN...	TABLES STR...	STREAM BY ID
2	LOAD_DATABASES	TASK	MONITORING_DB	MONITORIN...	LOAD_SCHÉ...	TASK BY ID
4	ACCESS_HISTORY	TABLE	MONITORING_DB	MONITORIN...	ACCESS_HIS...	STREAM BY ID
12	LOAD_TASK VERSIONS	TASK	MONITORING_DB	MONITORIN...	LOAD_TASK...	TASK BY ID

Rows per page: 100 ▾ 1-37 of 37 < >

FIGURE 4.35 : Liste des objets dans la base de données

Ces informations permettent aux administrateurs de l'application de surveiller et de gérer son compte Snowflake, en ayant une vue d'ensemble des bases de données, des schémas et des objets utilisés dans le système de surveillance.

- **Tableaux de bord du surveillance des bases de données :**

La figure ?? représente le tableau de bord du surveillance des bases de données :

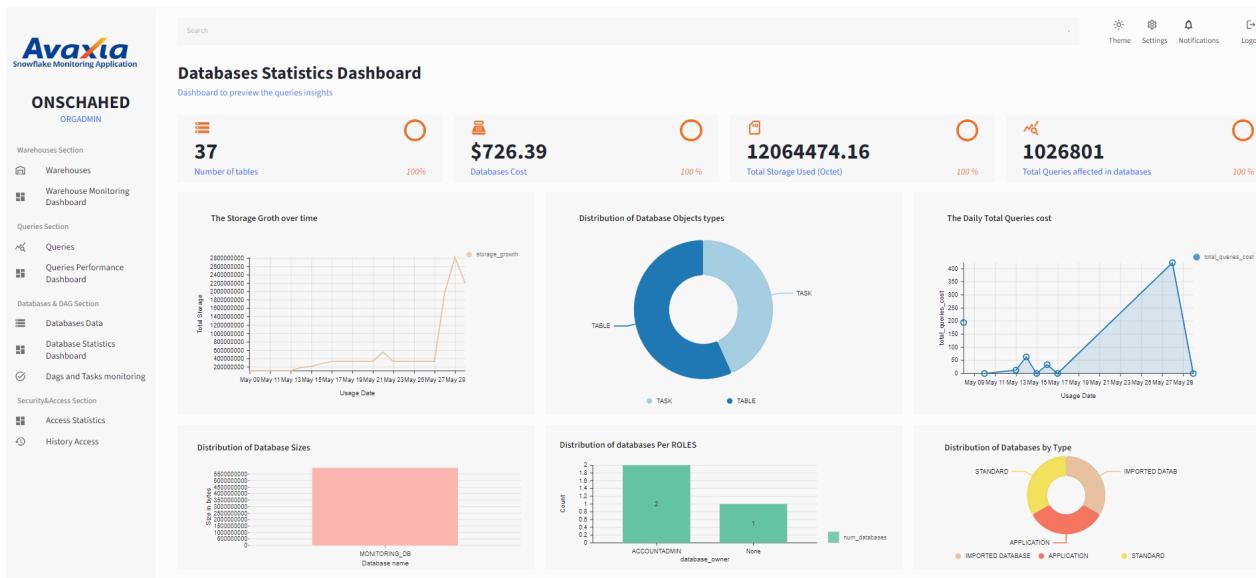


FIGURE 4.36 : Tableaux de bord du surveillance des bases de données

Cette interface de tableau de bord des statistiques des bases de données fournit une vue d'ensemble complète des principales métriques liées à l'utilisation et à la gestion des données dans l'environnement Snowflake. Elle présente des indicateurs clés tels que le nombre de tables,

le coût total des bases de données, le stockage total utilisé et le nombre total de requêtes affectées. Les graphiques détaillés complètent ces informations en montrant l'évolution de l'utilisation du stockage, la répartition des types d'objets de base de données, la distribution des bases de données par rôle et par type, ainsi que l'évolution du coût quotidien des requêtes.

Ce microservice offre aux administrateurs un outil complet et puissant pour surveiller les performances des bases de données, identifier les tendances clés et prendre des décisions éclairées afin d'optimiser l'utilisation et la gestion des coûts de leurs comptes.

4.2.5 Micro-service « Access_Statistics »

Le micro-service « Access_Statistics » est le service chargé de la surveillance des access aux comptes Snowflake.

- **La liste des APIs :**

La liste des APIs présents dans ce micro-service, documentée par Swagger, sont représentés par la figure 4.37 suivante :

The screenshot shows the FastAPI Swagger UI interface. At the top, it displays "FastAPI 0.1.0 OAS 3.1" and a link to "/openapi.json". On the right side, there is a green "Authorize" button with a lock icon. The main area is titled "default" and contains a list of API endpoints:

- GET /security/access_history Retrieve the access history data
- GET /security/login_history/{user_name} Retrieve the login history data
- GET /security/{user_name}/default Retrieve the roles of a user
- GET /security/{user_name}/user_roles Retrieve the roles of a user
- GET /security/role/{role_name} Retrieve the information of a role
- GET /security/dashboard/{role} Retrieve the security configuration data
- GET /security/last_access Retrieve the last access history data
- GET /security/all_users Retrieve all users
- GET /security/{username}/sessions Retrieve all user sessions
- GET /security/{username}/sessions/KPIS Retrieve all user sessions KPIS
- GET /security/{username}/sessions/authentications Retrieve all user authentications
- GET /security/{username}/sessions/applications_KPI Retrieve all user applications

FIGURE 4.37 : Liste des APIs du microservice « Access_Statistics »

- **L'historique des accès de l'utilisateur ONS CHAHED :**

L'interface de l'historique d'accès est représentée par la figure 4.38 suivante :

Access ID	Access Time	Event Type	Username	Access IP Address	Authentication Factor
175445113247718	2024-05-30 11:34:45	LOGIN	ONSCHAHED	197.1.181.107	PASSWORD
175445113247722	2024-05-30 11:34:50	LOGIN	ONSCHAHED	197.1.181.107	PASSWORD
175445113247726	2024-05-30 11:34:55	LOGIN	ONSCHAHED	197.1.181.107	PASSWORD
175445113247730	2024-05-30 11:34:58	LOGIN	ONSCHAHED	197.1.181.107	PASSWORD
175445113248534	2024-05-30 11:34:02	LOGIN	ONSCHAHED	197.1.181.107	PASSWORD
175445113248538	2024-05-30 11:34:08	LOGIN	ONSCHAHED	197.1.181.107	PASSWORD
175445113248542	2024-05-30 11:34:11	LOGIN	ONSCHAHED	197.1.181.107	PASSWORD
175445113248546	2024-05-30 11:34:12	LOGIN	ONSCHAHED	197.1.181.107	PASSWORD
175445113248550	2024-05-30 11:34:14	LOGIN	ONSCHAHED	197.1.181.107	PASSWORD
175445113248554	2024-05-30 11:34:15	LOGIN	ONSCHAHED	197.1.181.107	PASSWORD
175445113248558	2024-05-30 11:34:17	LOGIN	ONSCHAHED	197.1.181.107	PASSWORD
175445113248562	2024-05-30 11:34:18	LOGIN	ONSCHAHED	197.1.181.107	PASSWORD
175445113248566	2024-05-30 11:34:20	LOGIN	ONSCHAHED	197.1.181.107	PASSWORD

FIGURE 4.38 : Historique des accès

Cette interface d'historique des accès fournit une traçabilité complète des activités dans l'environnement Snowflake. Elle enregistre de manière détaillée chaque événement d'accès, capturant des informations clés telles que l'identité de l'utilisateur, l'heure d'accès et les méthodes d'authentification utilisées.

Ce journal d'activité exhaustif représente un outil de gouvernance et de surveillance essentiel pour les administrateurs. Il leur permet de suivre les mouvements des différents acteurs au sein du système, de détecter d'éventuelles activités suspectes et d'assurer la sécurité globale de l'infrastructure.

- **Tableaux de bord du surveillance des accécess des utilisateurs :**

Les figures 4.40 et ?? représentent les tableaux de bord du surveillance des accès de l'utilisateur «ONS CHAHED» et l'utilisateur système «Snowflake» :

Chapitre 4. Réalisation

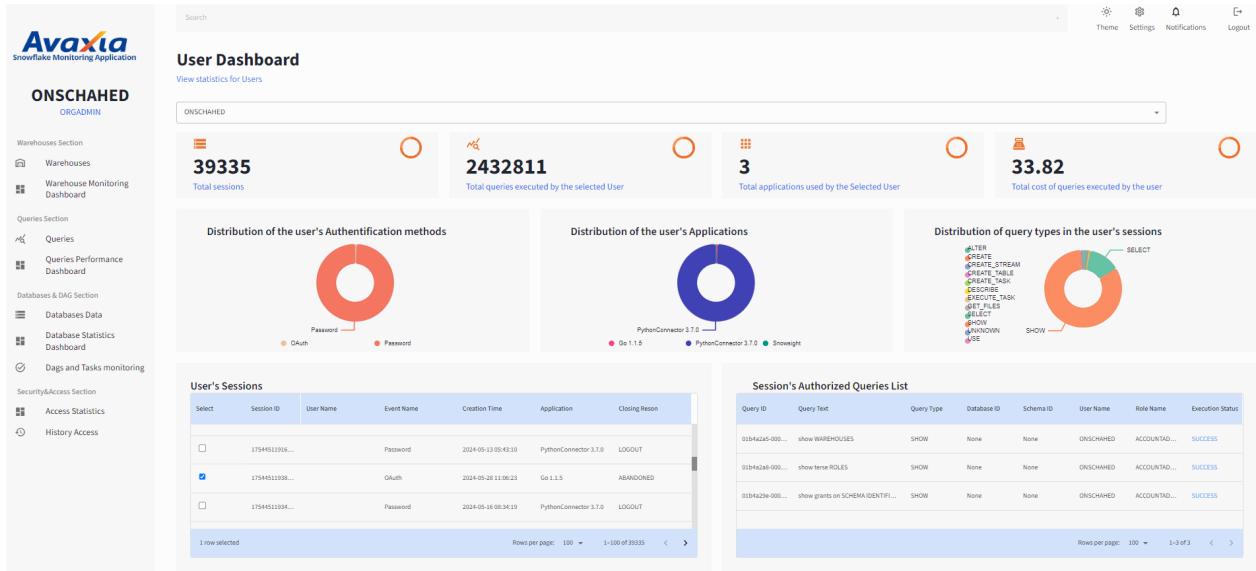


FIGURE 4.39 : Tableau de bord du surveillance des accééss de l'utilisateur «ONS CHAHED»

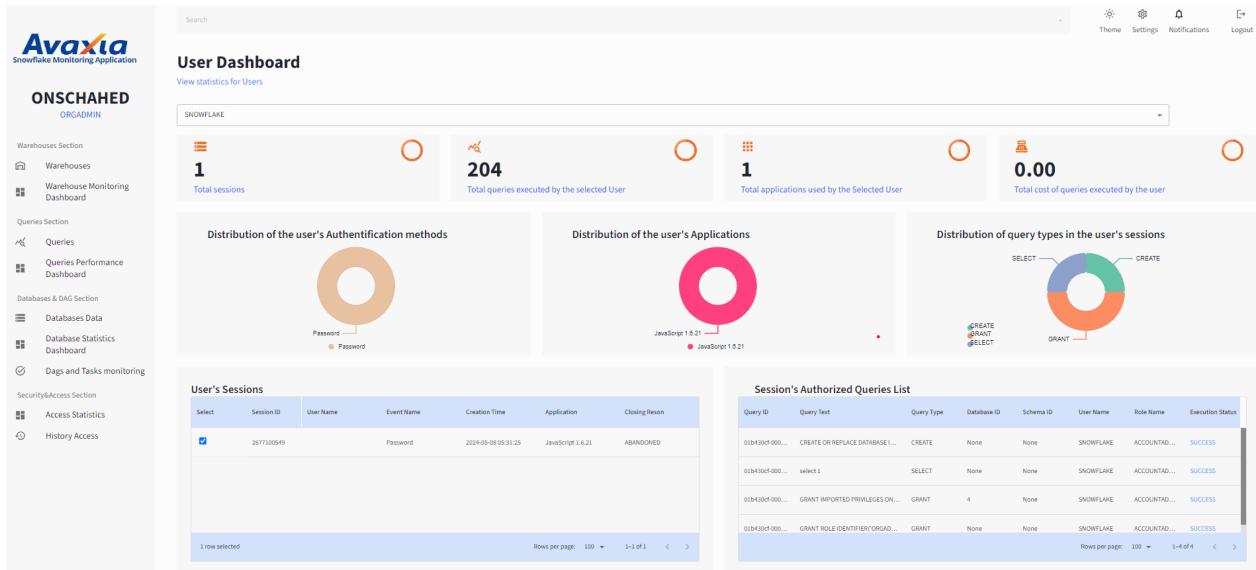


FIGURE 4.40 : Tableau de bord du surveillance des accééss de l'utilisateur «SNOWFLAKE»

Ces tableaux de bord offrent une visibilité complète sur l'activité des différents utilisateurs au sein de l'environnement Snowflake. Ils permettent de suivre des indicateurs clés tels que le nombre de sessions, de requêtes exécutées, d'applications utilisées et les coûts associés. Ces données quantitatives fournissent une image détaillée de l'utilisation du système par chaque utilisateur.

L'historique détaillé des sessions et des requêtes autorisées complète ce tableau, donnant aux administrateurs une traçabilité exhaustive des actions entreprises par chaque utilisateur. Cet outil de suivi leur permet de détecter les éventuelles anomalies et de s'assurer du respect des

politiques d'accès. Ils fournissent aux équipes opérationnelles les informations nécessaires pour prendre des décisions éclairées en matière de sécurité, de contrôle d'accès et d'utilisation des ressources.

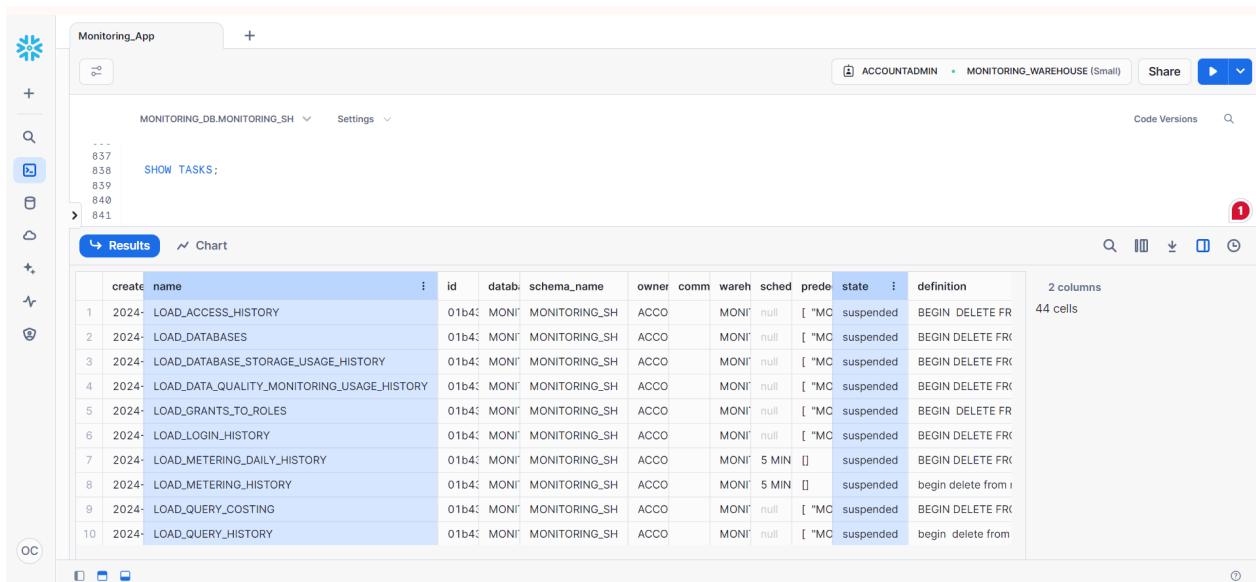
Ce microservice de surveillance des accès est bien plus qu'un simple registre. C'est un levier puissant pour les équipes en charge de la gouvernance et de l'exploitation des systèmes de données, leur permettant de prendre des décisions éclairées et de maintenir la fiabilité et la résilience de l'environnement Snowflake dans le temps.

4.2.6 Micro-service « DAG_Monitoring »

C'est le service responsable du suivi et de la surveillance en temps réelle des flux de tâches («DAG») exécutés et programmées dans Snowflake.

Dans cette section, nous allons surveiller l'exécution d'un ensemble des tâches en temps réelle.

Ces tâches sont déjà créées dans notre compte Snowflake de test. La figure 4.41 illustre la liste des tâches, dont nous allons montrer, dans snowflake qui apparaît en exécutant la commande « Show TASKS » :



	create	name	id	dbatb	schema_name	owner	comm	wareh	sched	prede	state	definition	2 columns
1	2024-	LOAD_ACCESS_HISTORY	01b4c	MONI	MONITORING_SH	ACCO	MONI	null	["MC	suspended	BEGIN DELETE FR	44 cells	
2	2024-	LOAD_DATABASES	01b4c	MONI	MONITORING_SH	ACCO	MONI	null	["MC	suspended	BEGIN DELETE FR		
3	2024-	LOAD_DATABASE_STORAGE_USAGE_HISTORY	01b4c	MONI	MONITORING_SH	ACCO	MONI	null	["MC	suspended	BEGIN DELETE FR		
4	2024-	LOAD_DATA_QUALITY_MONITORING_USAGE_HISTORY	01b4c	MONI	MONITORING_SH	ACCO	MONI	null	["MC	suspended	BEGIN DELETE FR		
5	2024-	LOAD_GRANTS_TO_ROLES	01b4c	MONI	MONITORING_SH	ACCO	MONI	null	["MC	suspended	BEGIN DELETE FR		
6	2024-	LOAD_LOGIN_HISTORY	01b4c	MONI	MONITORING_SH	ACCO	MONI	null	["MC	suspended	BEGIN DELETE FR		
7	2024-	LOAD_METERING_DAILY_HISTORY	01b4c	MONI	MONITORING_SH	ACCO	MONI	5 MIN	[]	suspended	BEGIN DELETE FR		
8	2024-	LOAD_METERING_HISTORY	01b4c	MONI	MONITORING_SH	ACCO	MONI	5 MIN	[]	suspended	begin delete from i		
9	2024-	LOAD_QUERY_COSTING	01b4c	MONI	MONITORING_SH	ACCO	MONI	null	["MC	suspended	BEGIN DELETE FR		
10	2024-	LOAD_QUERY_HISTORY	01b4c	MONI	MONITORING_SH	ACCO	MONI	null	["MC	suspended	begin delete from		

FIGURE 4.41 : Résultat du command «Show tasks»

Simultanément, dans notre application, les tâches sont illustrées par la figure 4.43 dans le même état «SUSPENDED» que dans snowflake :

Chapitre 4. Réalisation

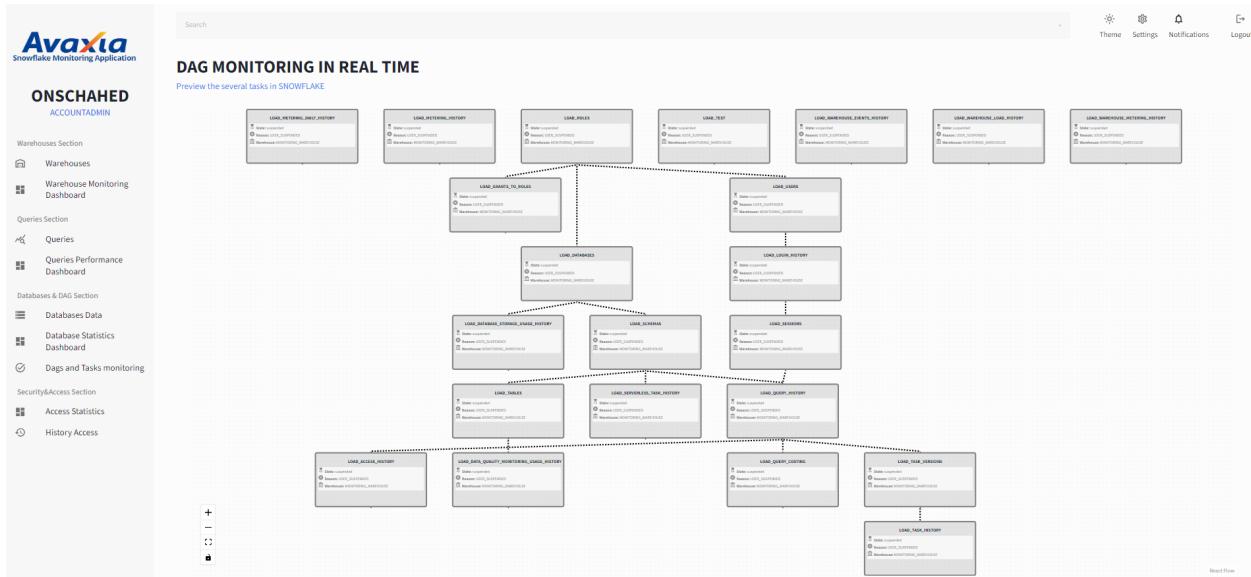


FIGURE 4.42 : Etat initial du DAG

Une fois nous résumons l'exécution des tâches dans snowflake, la visualisation du DAG commande à s'animer comme l'indique la figure 4.43 suivante :

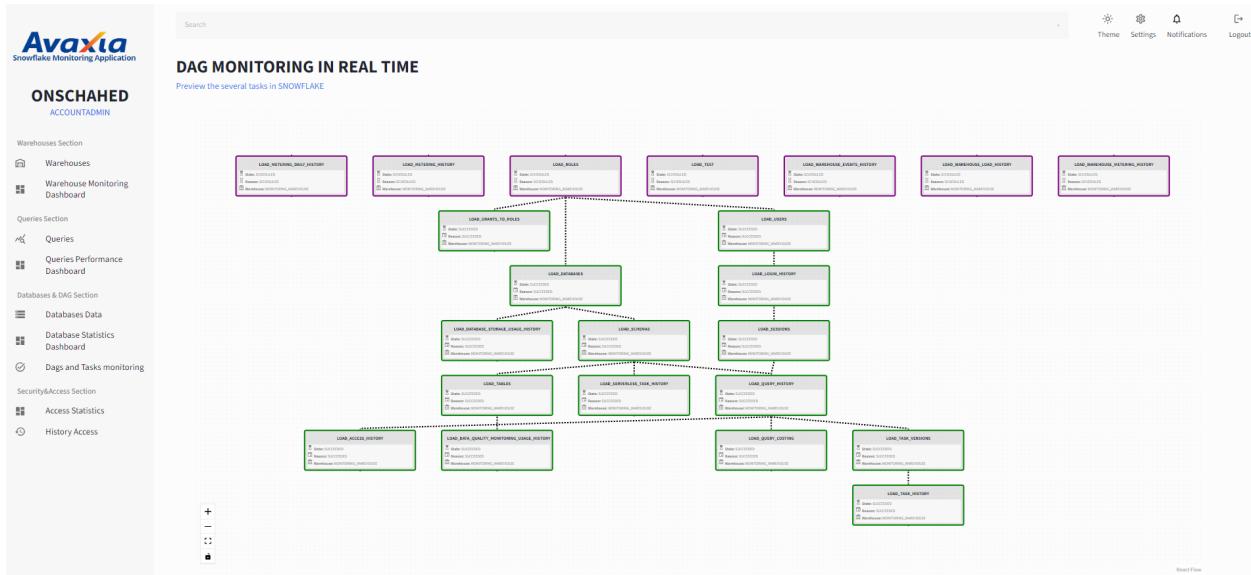


FIGURE 4.43 : Lancement des tâches

- Les tâches qui représentent les racines de DAG doivent être toujours programmées, ils s'affichent avec la couleur violet, pour qu'elles se lancent d'une façon séquentielle,
- Les sous-tâches suivent l'exécution de la tâche racine,
- L'état courant des sous tâches lorsque la racine est programmée préserve le dernier état après la dernière exécution,

<

Conclusion

Ce chapitre résume le travail accompli pour donner forme à notre projet. Les choix techniques et le workflow nous ont fourni un cadre solide pour concrétiser notre vision. Chaque étape de ce processus s'avère cruciale pour l'atteinte de nos objectifs.

Dans la conclusion générale à venir, nous réunirons l'ensemble de notre travail et soulignerons les perspectives pour l'avenir.

Conclusion générale

Dans un monde de plus en plus orienté vers les données, l'analyse des données joue un rôle crucial, particulièrement dans le domaine de l'informatique et de la technologie de l'information. Dans le cadre de notre projet au sein d'Avaxia Consulting, nous avons pu constater l'importance capitale de l'analyse des données pour obtenir des informations exploitables et prendre des décisions éclairées. Cette démarche ne se limite pas à une simple collecte de données, mais elle englobe une exploration approfondie, une interprétation et une transformation de ces données brutes en connaissances qui guident nos actions.

Au cœur de notre projet, l'analyse des données représente un catalyseur puissant, permettant d'optimiser les performances, de dégager des tendances significatives, et de cerner les domaines nécessitant des améliorations. Elle est le moteur qui propulse l'informatique vers des sommets de plus en plus élevés, contribuant ainsi au succès et à l'efficacité de notre équipe et, par extension, de l'ensemble du secteur des technologies de l'information.

Le premier chapitre de ce rapport a établi les bases de notre projet chez Avaxia Consulting. Nous avons présenté le contexte global du projet, mettant en lumière les objectifs que nous visons. De plus, nous avons donné un aperçu de l'organisme d'accueil, Avaxia Consulting, en expliquant ses services et son rôle dans le domaine de la technologie de l'information. Ce chapitre nous a fourni une solide fondation pour les étapes ultérieures du projet, en clarifiant la portée de notre travail et en soulignant notre ambition d'avoir un impact positif dans ce secteur en constante évolution.

Dans le deuxième chapitre, nous avons effectué une analyse minutieuse des besoins. Cela incluait une classification rigoureuse des besoins fonctionnels et non fonctionnels, ainsi qu'une définition claire du flux de travail et du backlog du produit. Ce chapitre a servi de fondement essentiel pour la phase ultérieure de planification et de développement.

Le troisième chapitre a été dédié à l'architecture et conception. Nous avons justifié notre choix d'architecture en couches et avons fourni des détails sur l'architecture logique et physique de notre solution.

Le quatrième chapitre s'est concentré sur la réalisation pratique du projet. Nous avons abordé en détail les choix techniques et le travail effectué pour donner vie à notre solution. De plus, nous avons décrit notre workflow en cinq étapes, de l'extraction à la visualisation des données. Chacune de ces étapes est cruciale pour la réussite du projet, et nous avons mis en place des mécanismes solides

Conclusion générale

pour garantir la qualité et la fiabilité de notre travail.

Le potentiel d'amélioration réside également dans les tableaux de bord que nous avons créés. Ils sont actuellement conçus pour fournir des informations essentielles, mais ils pourraient être étendus pour inclure des fonctionnalités plus avancées, telles que des analyses prédictives et des recommandations automatisées. Ces améliorations contribueraient à renforcer la capacité d'Avaxia Consulting à prendre des décisions stratégiques et à anticiper les tendances futures.

En résumé, bien que nous ayons rencontré des défis et des contraintes, ce projet représente une étape essentielle vers la réalisation de perspectives prometteuses pour Avaxia Consulting. Il souligne l'importance de l'analyse des données dans le secteur de la technologie de l'information et montre comment des améliorations continues peuvent renforcer la compétitivité et la réussite de l'entreprise. Notre engagement envers l'excellence et l'innovation nous encourage à explorer davantage ces perspectives pour offrir une valeur accrue à Avaxia Consulting.

Bibliographie

- [1] *Presentation Avaxia*, <https://www.avaxiagroup.com/fr/>, [En ligne ;Accès le 2-sep-2023], 2023.
- [2] *Services Avaxia*, <https://www.avaxiagroup.com/fr/services>, [En ligne ;Accès le 2-sep-2023], 2023.
- [3] NUTCACHE, *backlog produit*, <https://www.nutcache.com/fr/blog/quest-ce-qu'un-backlog-scrum/>, [En ligne ;Accès le 2-sep-2023], 2019.
- [4] J. PAQUET, *difference backlog Kanban /backlog SCRUM*, <https://blog.myagilepartner.fr/index.php/2019/10/16/backlog-kanban/>, [En ligne ;Accès le 19-sep-2023], 2017.
- [5] M. TOUHTOUH, *Architecture Logique des systèmes*, <https://www.softfluent.fr/blog/architecture-logicielle-pour-application>, [En ligne ;Accès le 21-Mars-2024], 2023.
- [6] D. ZHART, *Patrons de conception*, <https://refactoring.guru/fr/design-patterns/what-is-pattern>, [En ligne ;Accès le 21-Mars-2024], 2023.
- [7] G. KHERIJI, *CQRS*, <https://www.invivoo.com/le-pattern-cqrs/>, [En ligne ;Accès le 21-Mars-2024], 2023.
- [8] A. SCHEVTS, *Patron façade*, <https://refactoring.guru/fr/design-patterns/facade>, [En ligne ;Accès le 21-Mars-2024], 2023.
- [9] A. SCHEVTS, *Patron observer*, <https://refactoring.guru/fr/design-patterns/observer>, [En ligne ;Accès le 21-Mars-2024], 2023.
- [10] *Single Responsability Principal*, <https://www.ionos.fr/digitalguide/sites-internet/developpement-web/quest-ce-que-le-factory-pattern/>, [En ligne ;Accès le 21-Mars-2024], 2022.
- [11] GEEKSFORGEEKS, *Data Access Object*, <https://www.geeksforgeeks.org/data-access-object-pattern/>, [En ligne ;Accès le 21-Mars-2024], 2024.
- [12] REDHAT, *architecture physique*, <https://www.redhat.com/fr/topics/cloud-native-apps/what-is-an-application-architecture>, [En ligne ;Accès le 22-Mars-2024], 2023.
- [13] LUCIDCHART, *Diagramme de package*, <https://www.lucidchart.com/pages/fr/diagramme-package-uml>, [En ligne ;Accès le 22-Mars-2024], 2024.

Bibliographie

- [14] LUCIDCHART, *Diagramme de package*, <https://www.lucidchart.com/pages/fr/diagramme-de-classes-uml>, [En ligne ;Accès le 22-Mars-2024], 2024.
- [15] LUCIDCHART, *Diagramme de cas d'utilisation*, <https://www.lucidchart.com/pages/fr/diagramme-de-cas-dutilisation-uml>, [En ligne ;Accès le 23-Mars-2024], 2024.
- [16] laurent AUDIBERT, *description textuelle de diagramme de cas d'utilisation*, <https://laurent-audibert.developpez.com/Cours-UML/?page=diagramme-cas-utilisation>, [En ligne ;Accès le 23-Mars-2024], 2021.
- [17] IBM, *Diagramme d'activité*, <https://www.ibm.com/docs/fr/dmrt/9.5?topic=diagrams-activity>, [En ligne ;Accès le 22-Mars-2024], 2024.
- [18] F. COMMUNITY, *Fastapi*, <https://fastapi.tiangolo.com/fr/>, [En ligne ;Accès le 01-Mai-2024], 2024.
- [19] R. COMMUNITY, *React*, <https://fr.legacy.reactjs.org/>, [En ligne ;Accès le 01-Mai-2024], 2024.
- [20] P. COMMUNITY, *PostgresSQL*, <https://www.postgresql.org/>, [En ligne ;Accès le 01-Mai-2024], 2024.
- [21] M. COMMUNITY, *MQTT*, <https://mqtt.org/>, [En ligne ;Accès le 01-Mai-2024], 2024.
- [22] T. TARGET, *REST*, <https://www.lemagit.fr/definition/REST>, [En ligne ;Accès le 01-Mai-2024], 2024.
- [23] D. VARRAZZO, *Python Snowflake Connector*, <https://docs.snowflake.com/en/developer-guide/python-connector/python-connector>, [En ligne ;Accès le 01-Mai-2024], 2024.
- [24] D. VARRAZZO, *PSYCOPG*, <https://pypi.org/project/psycopg2/>, [En ligne ;Accès le 01-Mai-2024], 2024.
- [25] *pandas*, <https://pandas.pydata.org/>, [En ligne ;Accès le 01-Mai-2024], 2024.
- [26] D. VARRAZZO, *GMQTT*, <https://github.com/wialon/gmqtt>, [En ligne ;Accès le 01-Mai-2024], 2024.
- [27] *OS Module*, <https://docs.python.org/3/library/os.html>, [En ligne ;Accès le 01-Mai-2024], 2024.
- [28] DIGONIO, *Scheduler*, <https://pypi.org/project/scheduler/>, [En ligne ;Accès le 01-Mai-2024], 2024.

Bibliographie

- [29] *requests*, <https://pypi.org/project/requests/>, [En ligne ;Accès le 01-Mai-2024], 2024.
- [30] R. BENBRAHIM, *Postman*, <https://welovedevs.com/fr/articles/postman>, [En ligne ;Accès le 01-Mai-2024], 2021.
- [31] *MQTTX*, <https://mqtttx.app/docs>, [En ligne ;Accès le 01-Mai-2024], 2024.
- [32] *Git*, <https://www.atlassian.com/fr/git/tutorials/what-is-git>, [En ligne ;Accès le 01-Mai-2024], 2024.
- [33] R. KASSEL, *Github*, <https://datacientest.com/github-tout-savoir>, [En ligne ;Accès le 01-Mai-2024], 2021.
- [34] *Swagger*, <https://appmaster.io/university/fr/tutorials/endpoints/quest-ce-que-swagger-et-comment-lutiliser>, [En ligne ;Accès le 01-Mai-2024], 2024.
- [35] *Latex*, <https://dms.umontreal.ca/wiki/index.php/LaTeX>, [En ligne ;Accès le 01-Mai-2024], 2024.
- [36] *Draw.io*, <https://www.tice-education.fr/tous-les-articles-er-ressources/articles-internet/819-draw-io-un-outil-pour-dessiner-des-diagrammes-en-ligne>, [En ligne ;Accès le 01-Mai-2024], 2024.

ملخص

يوضح هذا التقرير مشروعًا داخل شركة Avaxia Consulting مركّزًا على تحليل أداء الفريق. يستكشف كل مرحلة من مراحل المشروع، بدءًا من استخراج البيانات إلى التحليل، مؤكّدًا على أهمية PostgreSQL في العملية. على الرغم من القيود الزمنية، نجح المشروع في تقديم آفاق للتحسين، بما في ذلك لوحتَان عمل متقدمة وفرص لدمج البيانات الإضافية. يؤكّد التقرير على الدور الحيوي لتحليل البيانات في قطاع تكنولوجيا المعلومات.

كلمات مفاتيح : تحليل البيانات، PostgreSQL، Python، لوحتَان العمل، أداء الفريق

Résumé

Ce rapport décrit un projet au sein d'Avaxia Consulting, axé sur l'analyse des performances de l'équipe. Il explore chaque phase du projet, de l'extraction des données à leur analyse, mettant en évidence l'importance de PostgreSQL dans le processus. Malgré des contraintes de temps, le projet a réussi à offrir des perspectives d'amélioration, notamment des tableaux de bord plus avancés et des possibilités d'intégration de données supplémentaires. Le rapport souligne le rôle essentiel de l'analyse des données dans le secteur de l'IT.

Mots clés : Analyse de données, Python, PostgreSQL, Snowflake, React, Tableaux de bord, Indicateur de Performance d'équipe.

Abstract

This report outlines a project within Avaxia Consulting, focusing on team performance analysis. It delves into each phase of the project, from data extraction to analysis, emphasizing the significance of PostgreSQL in the process. Despite time constraints, the project managed to offer improvement perspectives, including more advanced dashboards and opportunities for additional data integration. The report underscores the critical role of data analysis in the IT sector.

Keywords : Data analysis, Python, PostgreSQL, Dashboards, Key performance Indicators.