

## 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 Snowflake

---

Encadrant professionnel : **Monsieur Nassim JALLOUD**

Directeur adjoint du  
département Intégration  
et développement

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 méta-données de Snowflake

---

Encadrant professionnel : **Monsieur Nassim JALLOUD**

Directeur adjoint du  
département Intégration  
et développement

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 de fin d'études en vue de l'obtention du diplôme national d'ingénieur.

Encadrant professionnel, **Monsieur Nassim JALLOUD**

**Signature et cachet**

J'autorise l'étudiant à faire le dépôt de son rapport de stage de fin d'études en vue de l'obtention du diplôme national d'ingénieur.

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 chéri frère, Mouhamed,**

Même à des milliers de kilomètres, tu as toujours été une source de soutien. Ta présence, malgré la distance, m'a apporté réconfort et motivation pour surmonter les défis de ce projet. Chaque échange, chaque mot d'encouragement a compté plus que tu ne peux l'imaginer.

## **À ma chère belle-sœur, Nourchene,**

Ta gentillesse et ton soutien ont été des sources inestimables de réconfort tout au long de ce projet. Malgré la distance, tu as toujours su trouver les mots justes pour m'encourager et me motiver. Ton écoute et tes conseils m'ont aidé à traverser les moments difficiles.

## **À Mouhamed Moubarek,**

Tu été et tu resperas toujours ma source de motivation inépuisable. Chaque instant passé tu m'a donné la force et la détermination nécessaires pour mener à bien ce projet. Ton soutien indéfectible, ta patience et tes encouragements m'ont porté dans les moments les plus difficiles. Merci d'être toujours présent, de croire en moi et de partager cette aventure.

## **À mes meilleurs amis, Eya, Chaima, Moslem, Houssem et Abir,**

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.

**À mes chérs collègues,**

Merci pour votre soutien constant et votre camaraderie. Vos conseils avisés, votre patience et votre esprit d'équipe ont transformé chaque défi en une opportunité d'apprentissage. Travailler avec vous a été une expérience enrichissante et motivante.

**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 souhaite exprimer ma sincère reconnaissance à **Monsieur Nassim JALLOUD** pour son accompagnement précieux tout au long de ce projet. Votre expertise, vos conseils éclairés et votre disponibilité ont grandement contribué à la réussite de cette entreprise. Votre soutien constant et vos encouragements m'ont permis de surmonter les défis avec confiance et détermination. Merci pour votre engagement et votre investissement dans mon développement professionnel.

Je tiens à exprimer ma profonde gratitude envers **Monsieur Sahbi BAHROUN** pour sa guidance, son soutien et son dévouement tout au long de ce projet. Votre expertise, vos conseils éclairés et votre disponibilité ont été d'une aide précieuse pour mener à bien cette étude. Votre encadrement attentif et vos retours constructifs ont été essentiels dans mon parcours académique, et je vous en suis infiniment reconnaissant. Merci pour votre engagement et votre accompagnement tout au long de ce projet.

Je tiens à exprimer ma profonde gratitude **Monsieur Sofiene GHARBI** pour son soutien indéfectible tout au long de ce projet. Vos conseils avisés et votre encouragement constant ont été des éléments essentiels à la réussite de ce travail. Merci de m'avoir offert cette opportunité et de m'épanouir dans un environnement aussi stimulant. Votre confiance en mes capacités m'a motivé à donner le meilleur de moi-même chaque jour.

Je tiens à exprimer ma profonde gratitude envers **Madame Ameni SOUAI** qui m'a directement supervisé tout au long de ce projet. Votre guidance experte et votre soutien ont été d'une importance capitale pour moi. Votre capacité à identifier mes forces et mes faiblesses, ainsi que vos conseils avisés, m'ont permis de progresser et de m'épanouir dans mon travail. Merci infiniment pour votre dévouement et votre engagement envers mon succès.

Je souhaite exprimer ma sincère reconnaissance envers **Monsieur Ahmed BELHADJ KHELIFA**. Votre présence rassurante, vos encouragements et vos conseils ont été d'un grand soutien pour moi. Votre expertise et votre calme ont contribué à me permettre de donner le meilleur de moi-même. Merci pour votre soutien inestimable et votre présence précieuse à mes côtés.

Je suis reconnaissant envers les membres du jury qui vont lire attentivement mon rapport et évaluer la soutenance de mon projet. Leur expertise et leurs retours joueront un rôle crucial dans la validation de mon travail. Je les remercie pour leur investissement et leur contribution à cette étape importante de mon parcours académique.

Enfin, je n'oublie pas tous **mes enseignants de l'ISI** qui ont contribué à ma formation, durant mes 6 ans à l'ISI, qu'ils trouvent ici toute ma gratitude.

**Ons CHAHED**

# Table des matières

<b>Introduction générale</b>	<b>1</b>
<b>1 Cadre du projet</b>	<b>3</b>
Introduction . . . . .	4
1.1 Context général du projet . . . . .	4
1.2 Organisme d'accueil . . . . .	4
1.2.1 Présentation générale de Avaxia Group . . . . .	4
1.2.2 Les services de Avaxia Group . . . . .	4
1.3 Étude du projet . . . . .	5
1.3.1 Problématique . . . . .	6
1.3.2 Analyse et critique de l'existant . . . . .	6
1.3.3 Objectif du projet . . . . .	10
1.3.4 Solution proposée . . . . .	10
1.4 Choix méthodologique . . . . .	11
1.4.1 Différence entre Scrum et Kanban . . . . .	11
1.4.2 Justification de choix . . . . .	12
1.4.3 Le tableau Kanban . . . . .	12
Conclusion . . . . .	13
<b>2 Analyse et spécification des besoins</b>	<b>14</b>
Introduction . . . . .	15
2.1 Concepts de base . . . . .	15
2.1.1 La plateforme Snowflake . . . . .	15
2.1.2 Directed Acyclic Graph (DAG) . . . . .	15
2.1.3 Solution de Monitoring . . . . .	16
2.1.4 Analyse des Données . . . . .	16
2.1.5 L'ingénierie des données . . . . .	16
2.2 Capture des besoins . . . . .	17
2.2.1 Besoins fonctionnels . . . . .	17

---

2.2.2	Besoins non fonctionnels . . . . .	18
2.2.3	Diagramme de cas d'utilisation global . . . . .	19
2.3	Flux de travail . . . . .	20
2.4	Le backlog du produit . . . . .	22
	Conclusion . . . . .	24
<b>3</b>	<b>Architecture et conception</b>	<b>25</b>
	Introduction . . . . .	26
3.1	Étude architecturale . . . . .	26
3.1.1	Architecture logique . . . . .	26
3.1.2	Architecture physique . . . . .	31
3.2	Étude conceptuelle . . . . .	33
3.2.1	Conception globale . . . . .	33
3.2.2	Conception détaillée . . . . .	38
	Conclusion . . . . .	50
<b>4</b>	<b>Réalisation</b>	<b>51</b>
	Introduction . . . . .	52
4.1	Choix technologiques . . . . .	52
4.1.1	Frameworks et base de données . . . . .	52
4.1.2	Outils de communication . . . . .	53
4.1.3	Bibliothèques pour l'analyse et la manipulation des données . . . . .	53
4.1.4	Bibliothèques d'interaction avec le système d'exploitation . . . . .	54
4.1.5	Outils de test . . . . .	54
4.1.6	Outils de versionning . . . . .	55
4.1.7	Outils de documentation . . . . .	55
4.2	Réalisation . . . . .	55
4.2.1	Micro-service « ETL_Service » . . . . .	56
4.2.2	Micro-service « Authentification_Service » . . . . .	57
4.2.3	Micro-service « Warehouse_Monitoring » . . . . .	59
4.2.4	Micro-service « Query_Performance » . . . . .	63
4.2.5	Micro-service « Databases_Statistics » . . . . .	67

4.2.6	Micro-service « Access_Statistics » . . . . .	71
4.2.7	Micro-service « DAG_Monitoring » . . . . .	74
4.2.8	Micro-service « Notifications_Launcher» . . . . .	80
4.2.9	Personnalisation de l’interface utilisateur : thèmes clair et sombre . . . . .	84
Conclusion	. . . . .	86
<b>Conclusion générale</b>		<b>87</b>
<b>Bibliographie</b>		<b>89</b>

# Table des figures

1.1	Snowflake account usage dashboard . . . . .	7
1.2	Snowflake account usage dashboard . . . . .	8
1.3	Tableau de bord de «Google BigQuery» . . . . .	8
1.4	Tableau de bord de "Amazon Redshift" . . . . .	9
2.1	Diagramme de cas d'utilisation globale «Snowflake Monitoring Application» . . . . .	20
2.2	Processus métier du projet . . . . .	21
3.1	Architecture logique de «Snowflake Monitoring Application» . . . . .	27
3.2	Architecture physique de «Snowflake Monitoring Application» . . . . .	32
3.3	Diagramme de paquetages de «Snowflake Monitoring Application» . . . . .	34
3.4	le diagramme de classe des deux packages «authentification» et «Notification» . . . . .	35
3.5	le diagramme de classe du package «Warehouse_monitoring» . . . . .	35
3.6	le diagramme de classe du package «Dag&tasks» . . . . .	36
3.7	le diagramme de classe du package «Databases_statistics» . . . . .	36
3.8	le diagramme de classe du package «Users_access» . . . . .	37
3.9	le diagramme de classe du package «Query_performance» . . . . .	37
3.10	Diagramme d'activité du cas d'utilisation «Récupération des données » . . . . .	42
3.11	Diagrammme de séquence de cas d'utilisation «Visualiser DAG» . . . . .	44
3.12	Diagrammme de séquence de cas d'utilisation «Lister les détaills d'une tâche» . . . . .	46
3.13	Diagrammme de séquence de cas d'utilisation «Créer un compte» . . . . .	48
3.14	Diagrammme de séquence de cas d'utilisation «S'authentifier» . . . . .	49
4.1	Exécution de l'ETL . . . . .	56
4.2	La publication du message dans MQTT Broker . . . . .	57
4.3	Liste des APIs du microservice «Athentication_service» . . . . .	57
4.4	Interface de Création de compte «Sign up» . . . . .	58
4.5	Interface d'authentification «Login» . . . . .	58
4.6	«Logout» . . . . .	59
4.7	Liste des APIs du microservice «Warehouse_Monitoring» . . . . .	60

## Table des figures

---

4.8	Interface de la liste des entrepôts de données . . . . .	60
4.9	Tableau de bord du surveillance d l'entrepôts des données «Monitoring_warehouse» . . . . .	61
4.10	Tableau de bord du surveillance d l'entrepôts des données «Compute_WH» . . . . .	61
4.11	Changement du role de l'utilisateur de «ACCOUNT ADMIN» vers «ORGADMIN» . . . . .	62
4.12	Changement de seuil de credits vers «90\$» . . . . .	63
4.13	Liste des APIs du microservice «Warehouse_Monitoring» . . . . .	64
4.14	Interface de la liste des entrepôts de données . . . . .	65
4.15	Tableau de bord du surveillance des entrepôts des données . . . . .	65
4.16	Tableau de bord du surveillance des entrepôts des données . . . . .	66
4.17	Tableau de bord du surveillance de l'entrepôt des données «Monitoring_Warehouse» . . . . .	66
4.18	Tableau de bord du surveillance de l'entrepôt des données «Compute_WH» . . . . .	67
4.19	Liste des APIs du microservice« Databases_Statistics » . . . . .	68
4.20	La liste des bases de données . . . . .	69
4.21	Liste des schémas de base de données . . . . .	69
4.22	Liste des objets dans la base de données . . . . .	70
4.23	Tableaux de bord du surveillance des bases de données . . . . .	70
4.24	Liste des APIs du microservice « Access_Statistics » . . . . .	71
4.25	Historique des accées . . . . .	72
4.26	Tableau de bord du surveillance des accés de l'utilisateur «ONS CHAHED» . . . . .	73
4.27	Tableau de bord du surveillance des accés de l'utilisateur «SNOWFLAKE» . . . . .	73
4.28	Résultat du command «Show tasks» . . . . .	74
4.29	Etape 0 : état initial du DAG . . . . .	75
4.30	Lancement de l'exécution des tâches . . . . .	75
4.31	Etape 1 : Lancement des tâches . . . . .	76
4.32	Etape 3 : Execution de la racine . . . . .	76
4.33	Etape 4 : Exécution des sous-tâches de niveau I . . . . .	77
4.34	Etape 5 : Succes dans tous les branches du DAG «Load Roles» . . . . .	77
4.35	Message d'erreur de la tâche «Load test» . . . . .	78
4.36	Auto-suspend de la tâche . . . . .	78
4.37	DAG erronée . . . . .	79
4.38	La racine de Dag est suspendu automatiquement . . . . .	79

4.39 Détails de suspend de la racine . . . . .	80
4.40 Exécution de l'ETL . . . . .	80
4.41 Publication dans le topic «Mqtt/updates» . . . . .	81
4.42 Réception de la nouvelle notification de mise à jour dans «Snowflake Monitoring Application»	81
4.43 Consultation des notifications . . . . .	82
4.44 Modification de limites . . . . .	82
4.45 Publication dans le topic «Mqtt/credits» . . . . .	83
4.46 Récéption de la nouvelle notification . . . . .	83
4.47 Consultation des notifications . . . . .	84
4.48 Modification du thème . . . . .	84
4.49 Interfaces «Databases Dashboard » et «Database Objects List » en thème sombre . . . . .	85
4.50 Interfaces en «Queries Dashboard » et «Warehouses List » thème sombre . . . . .	86

# Liste des tableaux

1.1	Comparaison entre les méthodes de gestion du projet Scrum et Kanban . . . . .	11
2.1	Backlog de Produit . . . . .	24
3.1	description textuelle de cas d'utilisation «Lister les entrepôts de données» . . . . .	38
3.2	description textuelle de cas d'utilisation «Lister les requêtes SQL» . . . . .	39
3.3	description textuelle de cas d'utilisation «Lister l'historique des journaux d'accès au entrepôt des données» . . . . .	40
3.4	description textuelle de cas d'utilisation «Lister les bases de données, les schémas et les objets»	41

# Liste des abréviations

- **API** = Application Programming Interface
- **CPU** = Central Processing Unit
- **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
- **ISI** = Institut Supérieur de l’Informatique
- **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 monde où les données jouent un rôle crucial dans la prise de décision stratégique, la gestion efficace des données est devenue un impératif pour la compétitivité des entreprises. De plus avec l'avènement des plateformes de data warehousing cloud telque Snowflake, une plateforme de données en nuage reconnue pour sa flexibilité et ses performances, les entreprises nécessitent désormais d'outils puissants pour gérer, stocker et analyser leurs données à grande échelle. Ces systèmes de gestion de données performants permettent d'extraire des informations pertinentes et d'exploiter leur potentiel de manière optimale, ouvrant ainsi de nouvelles perspectives de croissance et d'innovation.

Cependant, exploiter les capacités de Snowflake requiert une surveillance constante et une optimisation proactive des opérations effectuées sur cette plateforme. C'est dans ce contexte que s'intègre notre sujet de fin d'études pour l'obtention du diplôme national d'ingénieur, effectué au sein de l'entreprise Avaxia Group, qui nous a accueilli dans un environnement créatif et sérieux et qui nous a confiée de concevoir et d'implémenter une plateforme analytique pour la visualisation des métadonnées de Snowflake son plateforme d'entrepôt de données principal.

Pour retracer l'acheminement chronologique de notre travail, le présent rapport a été subdivisé en quatre chapitres principaux, chacun abordant une étape essentielle du projet, depuis la définition du cadre et des besoins jusqu'à la réalisation technique et la mise en œuvre de la solution.

Le premier chapitre, **Cadre du projet**, pose les bases en introduisant le contexte général du projet, l'organisme d'accueil, et en analysant les enjeux auxquels nous faisons face. Il identifie également la problématique, propose une analyse critique de l'existant et définit les objectifs et solutions envisagées. Enfin, un choix méthodologique est justifié pour assurer une gestion de projet efficace.

Le deuxième chapitre, **Analyse et spécification des besoins**, se concentre sur la capture des besoins fonctionnels et non fonctionnels. Nous y détaillons les concepts clés liés à Snowflake, les solutions de monitoring et les techniques d'analyse de données. Un diagramme de cas d'utilisation global est présenté pour visualiser les interactions système-utilisateur, accompagné d'un flux de travail détaillant le processus de l'extraction à la visualisation des données. Le backlog du produit est également élaboré pour structurer et prioriser les tâches à accomplir.

Dans le troisième chapitre, **Architecture et conception**, nous développons l'architecture logicielle et matérielle du système proposé. L'étude architecturale aborde à la fois les aspects logiques et physiques, tandis que l'étude conceptuelle se focalise sur la conception globale et détaillée de la solution, garantissant une structure

## Introduction générale

robuste et scalable.

Enfin, Le quatrième et dernier chapitre, **Réalisation**, décrit le choix des technologies, les frameworks et outils utilisés pour le développement, ainsi que les étapes de réalisation des différents micro-services composant notre système de monitoring. Chaque micro-service est détaillé, illustrant la manière dont ils s'intègrent pour former une solution cohérente et efficace.

Nous clôturons notre rapport par une conclusion générale qui résume les réalisations essentielles de notre travail et propose des éventuelles perspectives.

# CADRE DU PROJET

---

## Plan

<b>Introduction</b>	4
1    Context général du projet	4
2    Organisme d'accueil	4
3    Étude du projet	5
4    Choix méthodologique	11
<b>Conclusion</b>	13

## Introduction

Ce chapitre introductif sera le départ pour la compréhension de notre projet. Nous commençerons la première partie par le cadre et le contexte de notre projet. Ensuite, nous allons étudier les solutions existantes afin de cibler les insuffisances et l'embauche de notre nouvelle solution.

### 1.1 Context général du projet

Ce travail s'inscrit dans le cadre d'un sujet de fin d'études en vue de l'obtention du diplôme national d'ingénieur au sein de L'**Institut Supérieur de l'Informatique (ISI)**. Durant un stage de quatres mois effectués chez **Avaxia group**, notre mission consiste à concevoir et implémenter une plateforme analytique pour les métadonnées de la Snowflake.

### 1.2 Organisme d'accueil

Dans cette section, nous allons présenter l'organisme d'accueil **Avaxia Group**.

#### 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 en Tunisie, au Japon et au Canada, et de futurs emplacements prévus 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 la problématique qui a menée à sa réalisation, 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 tel 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 ainsi que ses clients, se trouvent confronter à des problématiques particulières liées à l'utilisation de Snowflake, leur entrepôt de données principal, tel que :

- **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 Structured Query Language (**SQL**) complexes ou mal optimisées peuvent engendrer des goulets d'étranglement et des inefficacités dans l'utilisation des ressources de Snowflake, du coup elles nécessitent 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 les coûts tout en garantissant des performances optimales pour ses opérations ;

- **Surveillance et analyse des opérations :**

- 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 sont essentielles 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 de chaque solution existante, tout en mettant en lumière les lacunes qui ont conduit au développement de notre propre solution de monitoring des opérations Snowflake.

### 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. C'est pour cela, nous allons entamer l'analyse des outils existants utilisés pour surveiller les opérations sur les divers plateformes de l'analyse des données et le cloud data warehousing.

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

La figure 1.1 suivante illustre une partie de cette dashboard :

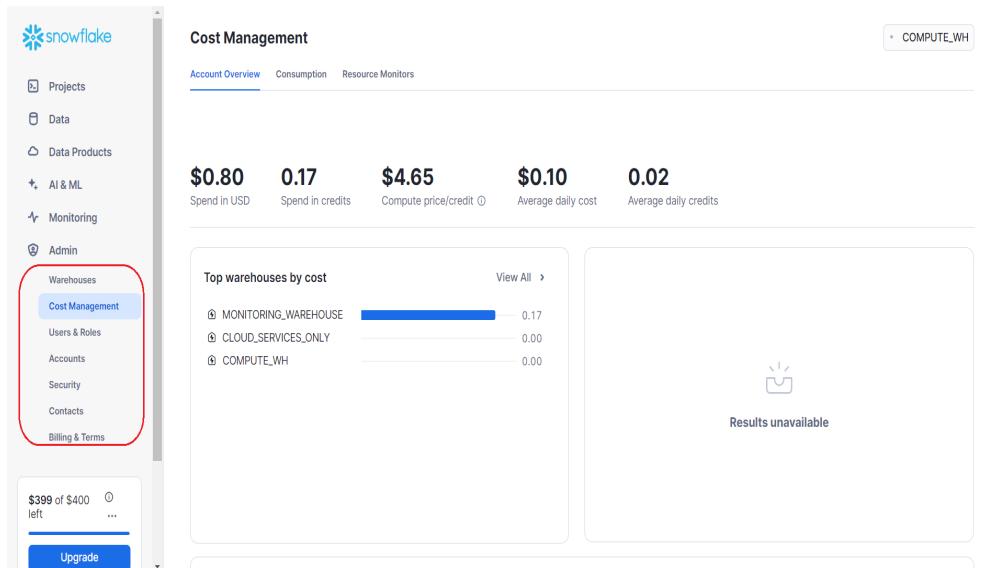


FIGURE 1.1 : Snowflake account usage dashboard

- **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 vue :

## Chapitre 1. Cadre du projet

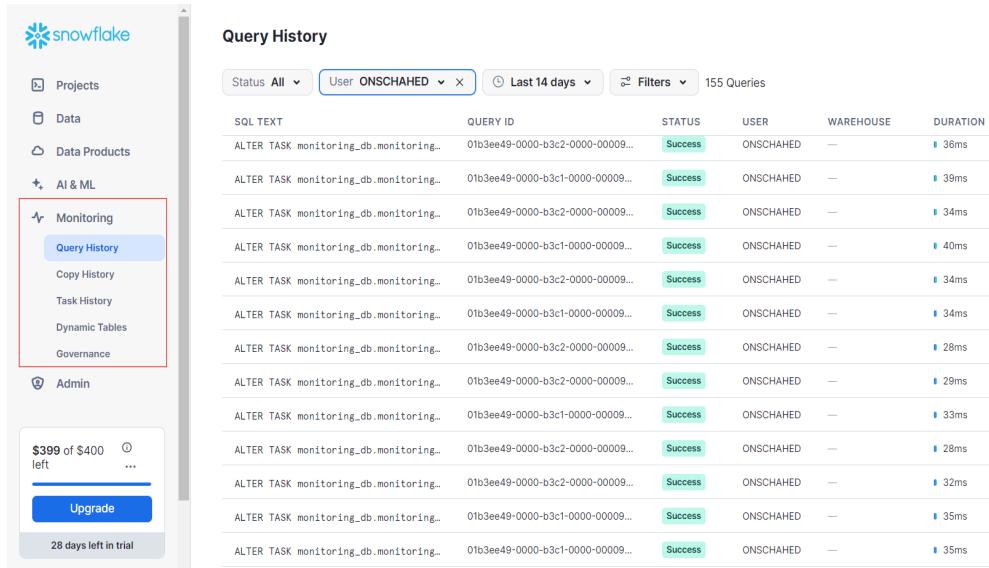


FIGURE 1.2 : Snowflake account usage dashboard

- **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 des grands ensembles de données ;

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

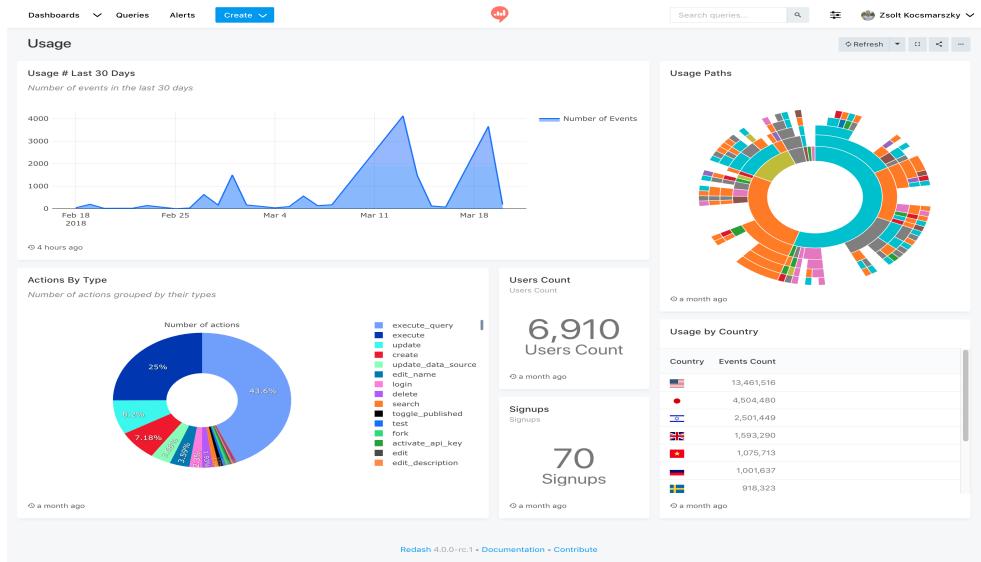


FIGURE 1.3 : Tableau de bord de «Google BigQuery»

- **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 RedShift :



**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 solutions 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 nos parties prenantes.

- Complexité des solutions :** les solutions 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 flux de travail si une certaine tache est échouée, les indicateurs disponibles sur snowflake ne peuvent pas identifier ou exactement le flux de travail est suspendu de façon à rendre les choses plus compliqués pour les utilisateurs de Snowflake de détecter les anomalies s'ils savent pas comment le manipuler technique.)
- 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 :** 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 Objectif du projet**

L'objectif principal de ce projet est de propulser les opérations Snowflake d'Avaxia Group vers de nouveaux sommets en termes de performance, de rentabilité et d'efficacité. Le développement de cette nouvelle plateforme de surveillance aidera l'équipe d'Avaxia à surmonter ces défis majeurs.

Grâce à cette initiative, Avaxia Group pourra optimiser ses processus, réduire les coûts opérationnels et améliorer la qualité des services offerts à ses clients. En effet, cette nouvelle plateforme permettra de détecter rapidement les problèmes de performance, d'anticiper les besoins en ressources et de garantir une gestion plus efficace des opérations.

### **1.3.4 Solution proposée**

Notre solution repose sur la conception et l'implémentation d'un système de surveillance et d'optimisation des opérations exhaustif. Cette solution se repartie en plusieurs composants clés, chacun ciblant des aspects spécifiques des problématiques identifiées :

- La mise en place des stratégies de monitoring en temps réel cela permettra d'identifier rapidement les problèmes de performance et de prendre des mesures correctives efficaces ;
- le développement d'outils d'analyse avancée des coûts qui aideront 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 ;
- 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 ;
- 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.

## 1.4 Choix méthodologique

Dans le but d'aboutir à une solution de qualité qui répond aux besoins exigés dans des temps et des coûts prévisibles, le choix du processus de développement convenable est une phase primordiale dans tout projet. Nous devons appliquer une méthode rigoureuse pour la conduite de notre projet.

### 1.4.1 Différence entre Scrum et Kanban

La table **1.1** résume une comparaison entre les méthodes agiles de gestion des projets **Scrum** et **Kanban** :

Point de comparaison	Scrum	Kanban
<b>Origine</b>	Développement logiciel	Fabrication Lean (Lean Manufacturing)
<b>Idéologie</b>	Apprendre de ses expériences, s'organiser et hiérarchiser, et réfléchir à ses réussites et ses échecs afin de s'améliorer en permanence	Utiliser des visuels pour améliorer le travail en cours
<b>Cadence</b>	Sprints réguliers et à durée déterminée (à savoir, deux semaines)	Flux continu
<b>Bonnes pratiques</b>	Planification du sprint, sprint, mêlée quotidienne (Daily Scrum), revue du sprint, rétrospective du sprint	Visualiser le flux de travail, limiter le travail en cours, gérer le flux, intégrer des boucles de rétroaction
<b>Rôles</b>	Product Owner, Scrum Master, équipe de développement	Aucun rôle requis
<b>Utilisation des tableaux</b>	Un tableau Scrum est nettoyé et recyclé après chaque sprint	Un tableau Kanban est utilisé tout au long du cycle de vie d'un projet
<b>Nombre des tâches</b>	Un tableau Scrum possède un nombre de tâches définies ainsi que des échéances strictes pour les effectuer	Les tableaux Kanban sont plus flexibles en termes de tâches et d'échéances. Les tâches peuvent être hiérarchisées à nouveau, réassignées ou mises à jour si besoin.

**TABLEAU 1.1** : Comparaison entre les méthodes de gestion du projet Scrum et Kanban

### 1.4.2 Justification de choix

Notre choix de Kanban comme étant pour la gestion de notre projet est le fruit d'une réflexion stratégique minutieuse. En effet, Kanban, qui tire son origine du terme japonais signifiant « tableau »[3], 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 favorise 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 claire de l'état d'avancement du projet, favorisant ainsi la collaboration et l'engagement de tous les membres.

### 1.4.3 Le tableau Kanban

Un tableau Kanban est un outil de gestion de projet Agile conçu pour aider à visualiser le travail, limiter le travail en cours et maximiser l'efficacité (ou le flux) [4].

Dans cette section, nous allons s'intéresser à la spécificités des tableaux Kanban et leurs types.

#### 1.4.3.1 Composants d'un tableau Kanban

Dans cette partie, nous allons découvrir les composants d'un tableau Kanban. En effet, Les tableaux Kanban utilisent des cartes, des colonnes, des couloirs et des limites de travail en cours pour permettre aux équipes de visualiser et de gérer efficacement leurs flux de travail. Examinons ces principaux éléments plus en détail :

- **Carte Kanban** : c'est une représentation visuelle des tâches. Chaque carte contient des informations importantes sur la tâche, telles que sa description, son statut d'avancement, les personnes impliquées, les échéances, etc[5] ;
- **Colonne Kanban** : chaque colonne une étape distincte du processus de travail. Les tâches traversent ces différentes colonnes au fur et à mesure de leur avancement, jusqu'à ce qu'elles soient complètement terminées. Cette organisation visuelle permet de suivre et de maîtriser le flux de travail de manière efficace [5] ;

- **Couloir Kanban** : sont des bandes horizontales pour séparer différents éléments tels que les activités, les équipes, les classes de service, etc. Ces bandes permettent d'organiser visuellement le tableau de manière plus granulaire et de mieux différencier les éléments qui le composent [5] ;
- **Limite de travail en cours** : elles limitent la quantité maximum de tâches dans les différentes étapes du flux de travail. Elles permettent de terminer des tâches plus rapidement en aidant l'équipe à ne concentrer que sur les tâches en cours [5] ;
- **Point d'engagement** : un point d'engagement est un point dans le processus de travail auquel une tâche est prête à être incorporée au système [5] ;
- **Point de livraison** : point du flux de travail auquel une tâche est considérée comme terminée [5].

#### 1.4.3.2 Types des tableaux Kanban

Les tableaux Kanban peuvent être divisés en plusieurs types :

- 1- **Un tableau Kanban physique** : est la forme la plus basique, les équipes utilisent des Post-its (représentant des tâches) et un tableau blanc (ou en liège). Les phases de travail sont représentées par des colonnes et les Post-its sont déplacés d'une étape à une autre [5].
- 2- **Un tableau Kanban numérique** : est une solution logicielle, ce qui le rend plus accessible que le tableau physique. Il peut fournir une visibilité sur la progression du travail depuis quasiment n'importe où, ce qui facilite la collaboration d'équipe. Certaines solutions numériques s'avèrent très souples, permettant aux responsables de suivre plusieurs flux de travail et d'organiser leur travail en différentes catégories [5].

Pour la gestion de notre projet, nous avons choisi « **Trello** » qui est une plateforme rapide et simple pour la création d'un tableau Kanban numérique.

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.

## Conclusion

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

---

# ANALYSE ET SPÉCIFICATION DES BESOINS

---

## Plan

<b>Introduction</b> . . . . .	<b>15</b>
1    Concepts de base . . . . .	15
2    Capture des besoins . . . . .	17
3    Flux de travail . . . . .	20
4    Le backlog du produit . . . . .	22
<b>Conclusion</b> . . . . .	<b>24</b>

## Introduction

La réussite de toute étude dépend de la qualité de la phase de démarrage. De ce fait, l'étape d'analyse des besoins constitue la base de départ de notre travail ainsi qu'une étape déterminante pour la suite. Dans ce chapitre, commencerons par la définition des concepts de bases que nous entamer tout au long du projet et nous évoquerons tout d'abord les besoins fonctionnels et non fonctionnels de notre solution. Ensuite Nous définirons clairement le flux de travail que nous allons élaborer tout au long de ce projet. Finalement, nous allons dresser le backlog produit qui contient toutes les fonctionnalités que nous allons utiliser comme étant la base essentielle des phases suivantes de planification et de développement.

### 2.1 Concepts de base

Dans cette section, nous définirons et expliquerons les principaux concepts et technologies essentiels à notre projet de monitoring des opérations sur Snowflake. La compréhension de ces concepts est primordiale pour saisir le fonctionnement et les objectifs de notre solution.

#### 2.1.1 La plateforme Snowflake

**Snowflake** est une plateforme de data warehousing cloud conçue pour le stockage et l'analyse des données à grande échelle. Elle offre des capacités de traitement massivement parallèles, permettant une scalabilité pratiquement illimitée. Snowflake se distingue par la séparation des couches de stockage et de calcul, offrant ainsi des avantages en termes de flexibilité, de performance et de coût [6].

##### Fonctionnalités clés de Snowflake :

- **Stockage des données** : utilisation de la compression et de la partition pour optimiser le stockage ;
- **Calcul élastique** : possibilité de redimensionner les ressources de calcul en fonction des besoins ;
- **Partage sécurisé des données** : facilite le partage des données entre différentes organisations tout en assurant la sécurité ;
- **Support SQL complet** : prise en charge des commandes SQL standards pour la manipulation des données.

#### 2.1.2 Directed Acyclic Graph (DAG)

Un **DAG** est une structure de données qui représente un ensemble de tâches et leurs dépendances sous forme de graphe dirigé sans cycles[7]. Dans le contexte des pipelines de données, un DAG est souvent utilisé pour

définir les étapes de traitement des données et leur ordre d'exécution.

**Utilisation des DAGs :**

- **Planification des tâches** : détermination de l'ordre dans lequel les tâches doivent être exécutées ;
- **Gestion des dépendances** : assure que les tâches dépendantes sont exécutées dans le bon ordre ;
- **Parallélisme** : permet l'exécution simultanée de tâches indépendantes.

### 2.1.3 Solution de Monitoring

Une **solution de monitoring** est un ensemble d'outils et de pratiques conçus pour surveiller les performances, la disponibilité et la santé des systèmes informatiques. Elle collecte, analyse et affiche des métriques et des journaux d'utilisation (logs) pour aider les administrateurs à détecter et résoudre les problèmes[8].

**Caractéristiques d'une solution de monitoring efficace :**

- **Collecte de métriques en temps réel** : surveillance continue des performances et des ressources ;
- **Alertes et notifications** : système d'alerte en cas d'anomalies ou de dépassement de seuils définis ;
- **Visualisation des données** : tableaux de bord interactifs pour l'analyse des métriques et des tendances ;
- **Reporting** : génération de rapports détaillés pour le suivi et l'analyse historique.

### 2.1.4 Analyse des Données

L'**analyse des données** est la science qui consiste à examiner les données pour en tirer des informations permettant de prendre des décisions ou d'approfondir les connaissances sur divers sujets [9]. Elle vise à comprendre et exploiter les données pour des prises de décision éclairées.

**Étapes de l'analyse des données :**

- **Collecte des données** : réunir les données pertinentes à partir de diverses sources ;
- **Préparation des données** : nettoyage et transformation des données brutes en un format utilisable ;
- **Modélisation** : utilisation de modèles statistiques et algorithmiques pour analyser les données ;
- **Interprétation et visualisation** : présentation des résultats sous forme de graphiques et de rapports compréhensibles.

### 2.1.5 L'ingénierie des données

L'**ingénierie des données** est la discipline consistant à concevoir, construire et maintenir des systèmes et des architectures pour collecter, stocker et analyser des données. Elle se concentre sur la création de l'infrastructure permettant l'exploitation des données par les analystes pour des prises de décision éclairées.

### Responsabilités de l'ingénierie des données :

- **Conception de pipelines de données** : définir et implémenter les processus de collecte, de transformation et de stockage des données ;
- **Gestion des bases de données** : assurer la performance, la sécurité et l'intégrité des bases de données ;
- **Optimisation des requêtes** : améliorer les requêtes pour une exécution plus rapide et efficace ;
- **Intégration de données** : combiner les données provenant de diverses sources pour une vue unifiée.

Cette section vise à établir une compréhension claire des concepts et des technologies essentiels utilisés dans notre projet. Cela permettra de mieux comprendre les choix de conception et les décisions techniques prises tout au long du développement de la solution de monitoring des opérations sur Snowflake.

## 2.2 Capture des besoins

La capture des besoins est une phase fondamentale dans la réalisation du projet puisque elle doit permettre aux utilisateurs finaux et au maître d'ouvrage de bien exprimer leurs besoins et de bien comprendre les fonctionnalités que le système va fournir.

### 2.2.1 Besoins fonctionnels

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

- **La collecte automatique des données de Snowflake** : Le système doit être capable de collecter automatiquement les données de Snowflake, y compris les requêtes exécutées et leurs informations, l'utilisation des ressources (CPU, mémoire, stockage), les bases de données et leurs objets (tâches, tables, tubes, etc.), les statistiques sur les entrepôts de données, etc. ;
- **La surveillance en temps réel des requêtes SQL** : Le système doit surveiller en temps réel les requêtes SQL exécutées sur Snowflake, enregistrant ces informations détaillées, les erreurs éventuelles en identifiant les requêtes lentes ou mal optimisées ;
- **La 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 des ressources ;
- **L'élaboration des 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 leurs DAGs** : Le système doit permettre le suivi des tâches et des DAGs exécutées dans Snowflake, enregistrant les étapes effectuées, tous leurs informations et les erreurs éventuelles ;
- **La 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 ;
- **La surveillance des utilisateurs et des accès** : Le système doit suivre l'activité des utilisateurs sur Snowflake, en enregistrant leurs requêtes exécutées, les objets 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'alertes.

### 2.2.2 Besoins non fonctionnels

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, 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**

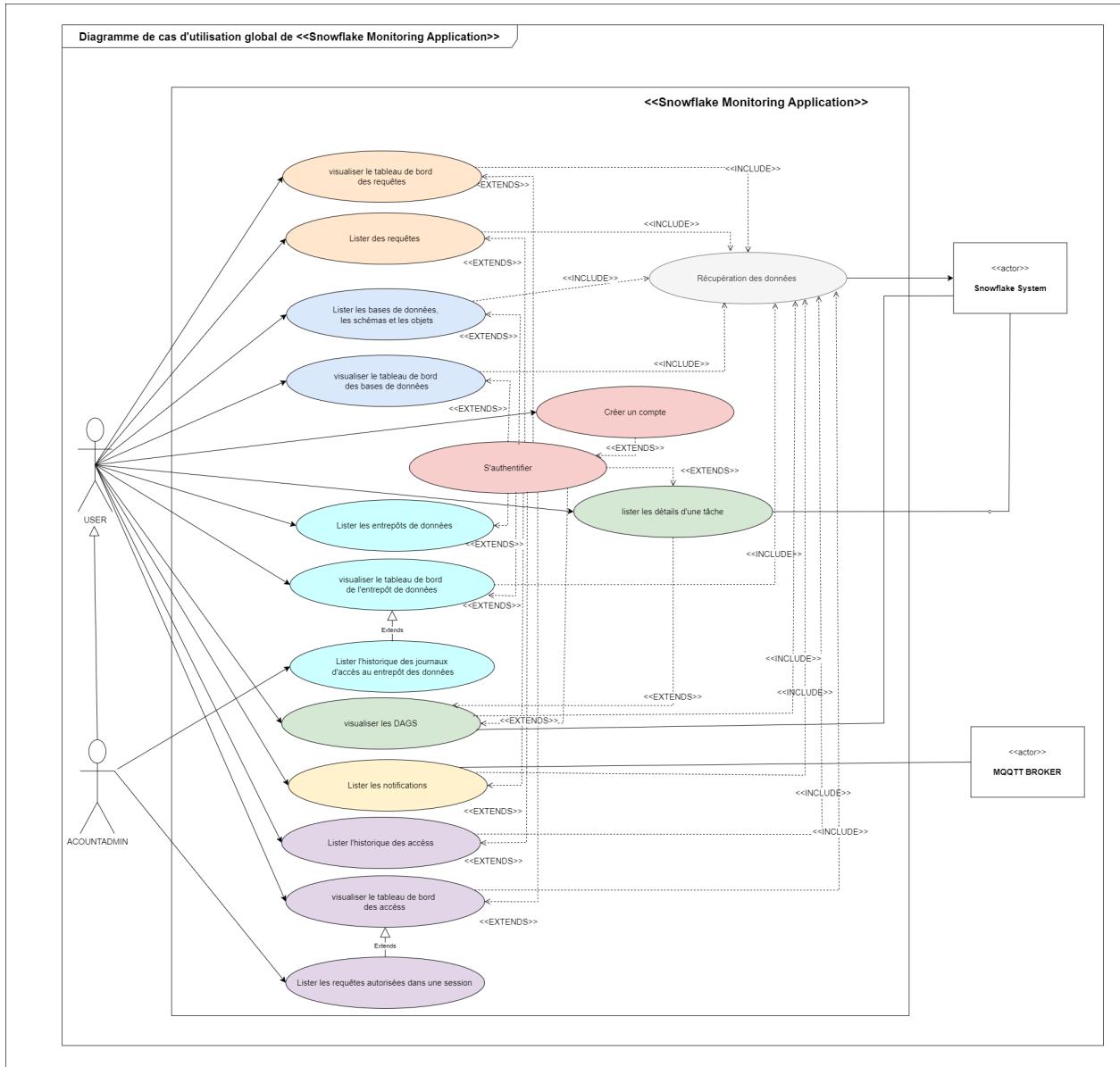
1. **Facilité d'utilisation** : le système doit être intuitif et convivial, avec une interface utilisateur bien conçue et une navigation facile ;
2. **Personnalisation** : le système doit permettre aux utilisateurs de personnaliser leur expérience, en configurant les préférences d'affichage et les paramètres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

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

### 2.2.3 Diagramme de cas d'utilisation global

Le diagramme de cas d'utilisation permet d'identifier les possibilités d'interaction entre le système et les acteurs (intervenants extérieurs au système), c'est-à-dire toutes les fonctionnalités que doit fournir le système. L'idée forte est de dire que l'utilisateur d'un système logiciel a un objectif quand il utilise le système [10].

La figure 2.1 qui suit représente le diagramme de cas d'utilisation global de notre système :



**FIGURE 2.1 :** Diagramme de cas d'utilisation globale «Snowflake Monitoring Application»

Le diagramme de cas d'utilisation global présenté ci-dessus offre une vue d'ensemble des interactions potentielles entre les utilisateurs et notre système. En identifiant clairement les objectifs des utilisateurs et les fonctionnalités requises, ce diagramme joue un rôle primordial dans la définition des exigences du système. En somme, ce diagramme est un outil essentiel pour garantir que le système final répond aux besoins des utilisateurs de manière efficace et cohérente.

### 2.3 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 2.2 suivante illustre notre processus métier :

## Le processus métier du projet



**FIGURE 2.2 : Processus métier du projet**

Maintenant, 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 consiste à recueillir des informations provenant des différentes vues du système Snowflake.

L'objectif principal est de collecter les données nécessaires à notre analyse. L'extraction de données est une phase primordiale dans l'ingénierie des données, car elle constitue le point de départ de tout projet d'analyse. Les techniques d'extraction comprennent des requêtes SQL avancées et l'automatisation des processus d'extraction pour garantir une collecte fiable et régulière ;

- **Transformation des données :**

L'objectif principal de cette étape est de nettoyer et structurer les données. Car, ces données brutes pouvaient contenir des erreurs, des doublons, des incohérences et/ou des valeurs manquantes. Nous allons appliquer des techniques de nettoyage, telles que l'imputation des valeurs manquantes, la déduplication et la normalisation des données. De plus, des processus de transformation ont été utilisés pour convertir les données dans des formats appropriés pour l'étape suivante ;

- **Chargement des données :**

Une fois les données sont nettoyées et préparées, elles sont acheminées vers une base de données. Le chargement vise à stocker les données de manière centralisée pour une accessibilité et une analyse ultérieures. Cette phase garantit que les données sont correctement indexées, partitionnées et optimisées pour des requêtes rapides et efficaces ;

- **Analyse des données :**

Cette phase consiste à explorer les données pour en extraire des informations significatives. L'analyse des données utilise des méthodes statistiques, des techniques de modélisation, des calculs de corrélation, etc., pour comprendre les tendances et les relations entre les données. Cette étape est essentielle pour transformer les données brutes en informations exploitables ;

- **Visualisation des données :**

L'analyse des données est souvent présentée de manière visuelle sous forme de tableaux de bord interactifs, de graphiques ou bien des diagrammes. En effet, 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 de manière concrète.

## 2.4 Le backlog du produit

Le backlog du 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.[11].

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 [12].

la table 2.1 suivante présente le backlog du produit de notre projet :

ID	User Story	Priorité	Estimation (points)
US1	En tant qu'administrateur, 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.) des bases de données pour identifier les goulets 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 surveillance 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 utilisateur, 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.	Haute	21
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 a la plateforme de monitoring.	Moyenne	5

US13	En tant qu'administrateur, je veux suivre en temps réelle l'exécution des tâches en temps réel avec une visualisation complète du DAG, ainsi que les différents états des tâches.	haute	13
US14	En tant qu'administrateur, je veux suivre l'historique des journaux d'accès au entrepôt des données.	haute	9
US15	En tant qu'administrateur, je veux paramétriser les visualisations, les alertes, les limites du coûts selon les besoins ou/et les rôles.	faible	9

**TABLEAU 2.1 : Backlog de Produit**

En conclusion, le backlog du produit joue un rôle essentiel dans notre projet en fournissant une vue d'ensemble claire et priorisée des fonctionnalités à développer et des tâches à accomplir. En utilisant la méthodologie Kanban, nous avons pu gérer efficacement ces éléments, ajuster nos priorités en temps réel et assurer une progression continue vers nos objectifs.

## Conclusion

Ce chapitre a établi les fondations de notre projet. Nous avons commencé par une introduction détaillée des concepts clés et des technologies qui sous-tendent notre solution, garantissant ainsi une compréhension solide de notre domaine d'application. Ensuite, nous avons effectué les besoins fonctionnels et non fonctionnels, et présenté le diagramme de cas d'utilisation global pour illustrer les interactions entre utilisateurs et système. Nous avons également détaillé le flux de travail, de l'extraction à la visualisation des données, et élaboré un backlog du produit structuré et priorisé, garantissant une progression continue du projet. En résumé, ce chapitre a formalisé les bases nécessaires pour le développement et la mise en œuvre de notre solution de monitoring, assurant ainsi une réponse cohérente aux besoins des utilisateurs. Dans le prochain chapitre, nous explorerons les aspects architecturaux et conceptuels de notre projet.

---

# ARCHITECTURE ET CONCEPTION

---

## Plan

<b>Introduction</b>	<b>26</b>
1    Étude architecturale	26
2    Étude conceptuelle	33
<b>Conclusion</b>	<b>50</b>

## Introduction

Le troisième chapitre explore l'architecture du projet et sa étude conceptuelle. Nous allons justifier notre choix architectural et détaillerons la conception qui nous a ménné 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.

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

#### 3.1.1 Architecture logique

L'architecture logique constitue le fondement sur lequel repose toute la structure de notre système. Dans cette section, nous plongeons au cœur de notre étude architecturale, explorant les choix et les stratégies qui sous-tendent la mise en œuvre de notre solution. À travers une approche méthodique, nous détaillons les composants clés, leurs interactions et leur organisation, offrant ainsi une vue d'ensemble claire et précise de notre architecture logique ainsi que les patrons de conception respecter dans cette dernière.

##### 3.1.1.1 Architecture logique adoptée

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.

Cette architecture met en lumière les différents composants fonctionnels et explique comment ces composantes interagissent pour fournir une expérience utilisateur fluide et des analyses précises[13]. Cette dernière est illustrée par la figure 3.1 suivante :

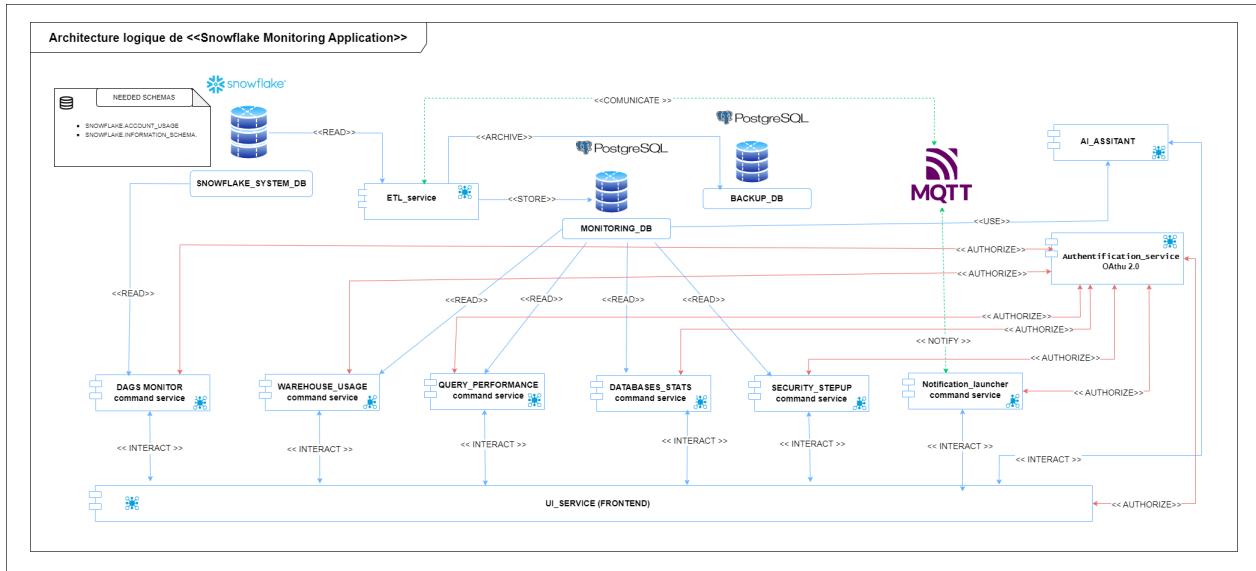


FIGURE 3.1 : Architecture logique de «Snowflake Monitoring Application»

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 toutes 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 utilisée pour l'archivage des données, 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 principales.

### 2. Couche applicative « micro-services » :

- **Etl\_Service** : service responsable de l'extraction, de la transformation et du chargement des données de la plateforme Snowflake vers la base de données de monitoring ;
- **DAG\_Monitoring** : service responsable du suivi et de la surveillance des flux de tâches programmées dans Snowflake en temps réel («DAG») ;
- **Warehouse\_Monitoring** : service chargé de la surveillance de l'utilisation, de la créditation et toutes les informations concernant les entrepôts de données Snowflake ;
- **Query\_Performance** : service d'analyse, collecte et traitement des performances des requêtes Snowflake ;

- **Databases\_Statistics** : service d'analyse, collecte et traitement des statistiques sur les bases de données Snowflake ;
- **Access\_Statistics** : service d'analyse, collecte et traitement des statistiques sur les access des utilisateurs des différents comptes et entrepôts de données Snowflake ;
- **Notifications\_Launcher** : service responsable de la gestion des notifications, il se charge de transmettre les notifications reçus 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** : service d'assistance IA intégré dans l'architecture pour apporter une dimension d'intelligence artificielle aux fonctionnalités de surveillance, d'analyse et d'optimisation des requêtes SQL.

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 et des résultats pertinentes.

### 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 APIs 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.

En conclusion, l'architecture logique adoptée, basée sur une approche micro-services, divise notre système en composants modulaires interagissant de manière fluide pour garantir la fonctionnalité globale de l'application. Cette approche assure une gestion efficace des données, une scalabilité optimale et une expérience utilisateur conviviale, constituant ainsi le fondement solide de notre solution de gestion et d'analyse de données.

### 3.1.1.2 Justification du choix de l'architecture micro-services

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 interagissent pour atteindre les objectifs fixés.

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. Car, chaque service fonctionne de manière indépendante, ce qui permet une adaptation aisée aux évolutions du système. Cette approche modulaire offre la possibilité d'ajouter de nouveaux services ou de mettre à l'échelle les services existants en fonction des besoins croissants de l'application ;
- **Modularité et maintenance :** la modularité est au cœur de l'architecture microservices, offrant un avantage significatif en termes de maintenance. En effet, chaque service peut être développé, testé et déployé de manière autonome, ce qui facilite la gestion des mises à jour et des correctifs, tout en isolant les composants fonctionnels du système, cette approche permet de limiter l'impact des changements sur l'ensemble de l'application. Assurant ainsi une maintenance plus efficace et moins sujette aux erreurs ;
- **Réutilisabilité et flexibilité technologique :** grâce à l'approche par microservices, notre projet bénéficie d'une grande réutilisabilité du code et d'une flexibilité technologique accrue. De fait, chaque service peut être développé en utilisant les technologies les mieux adaptées à ses besoins spécifiques, ce qui favorise l'adoption des meilleures pratiques et des outils les plus performants pour chaque composant. Car cette modularité permet également de réutiliser efficacement les services existants dans d'autres projets ou contextes, ce qui contribue à une meilleure efficacité du développement logiciel ;
- **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'adoption de l'architecture microservices se révèle être un choix judicieux pour notre projet, offrant une solution flexible, modulaire et résiliente. Elle nous permet de répondre efficacement aux besoins changeants de nos utilisateurs tout en garantissant des performances et une fiabilité optimales.

### 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[14].

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 model d'architecture qui sépare les deux parties de traitement (Écriture) et de réponse (Lecture) [15].

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[16].

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[17].

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

- **Le principe «de responsabilité unique (SRP)»** : ce principe dénonce que chaque qu'un 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[18].

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[19].

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.

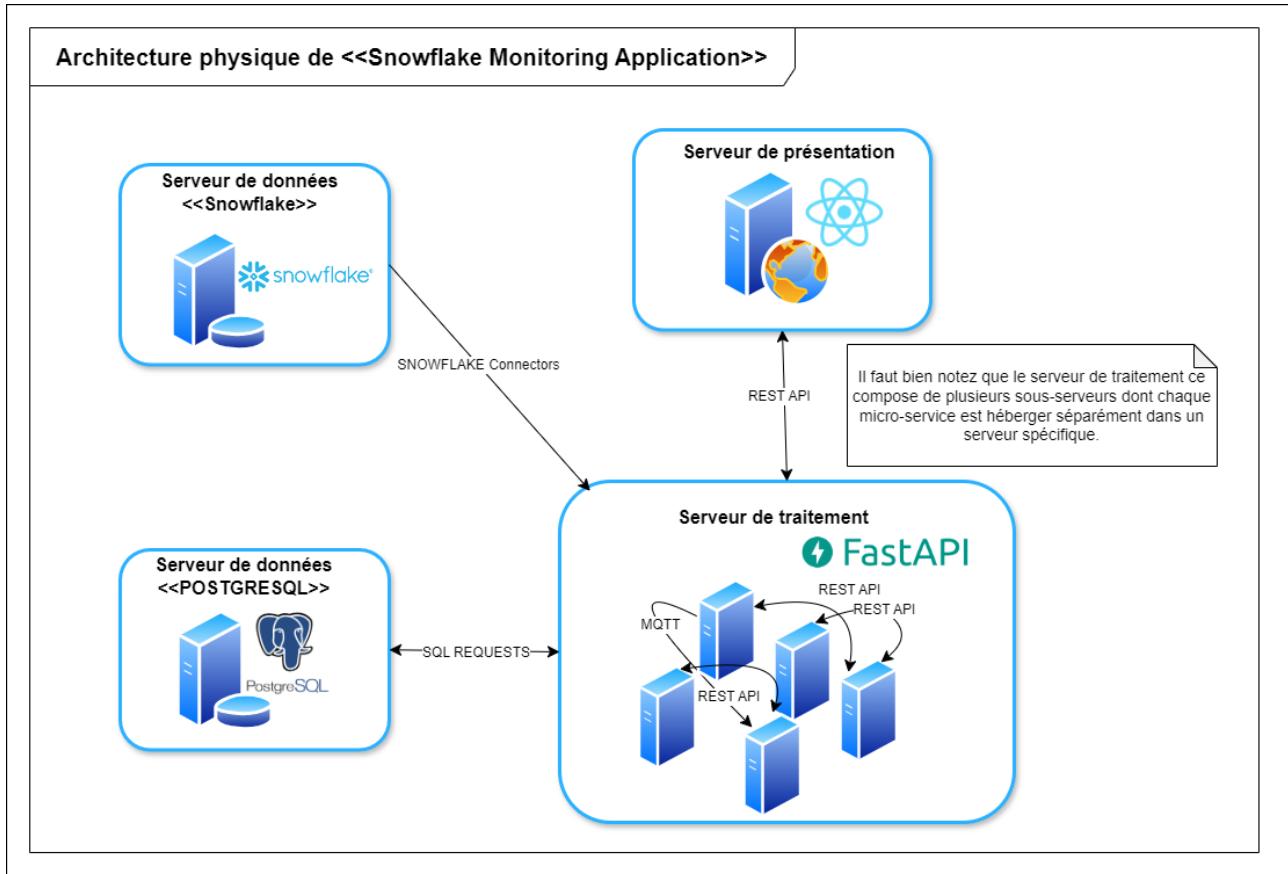
En conclusion, l'architecture logique adoptée, basée sur des microservices, offre une approche modulaire et flexible, permettant une gestion efficace des données et une évolutivité optimale de l'application. En intégrant des patrons de conception pertinents, nous garantissons une structure robuste et cohérente, répondant aux exigences spécifiques de notre projet de surveillance et d'analyse des données.

### 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[20].

Pour la mise en place de cette solution 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 cycliquement 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\_PERFORMANCE, 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.

Il est responsable de la présentation des données de monitoring aux utilisateurs via les tableaux de bord et les interfaces ;

- 4- **inter-communication** : cet tier représente les 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 la nature des données et les traitements qui circulent dans notre système.

### 3.2.1 Conception globale

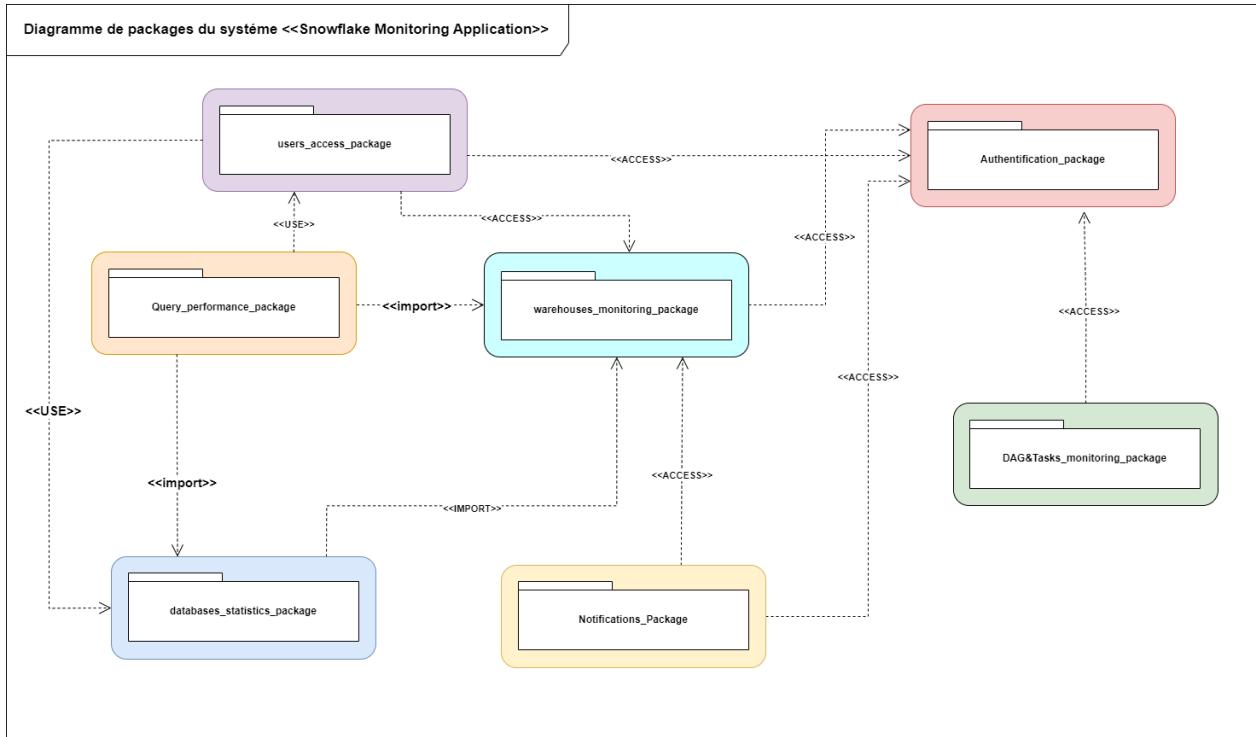
Dans cette section, nous allons illustrer les différents diagrammes modélisant la conception globale de notre solution.

#### 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[21].

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 :



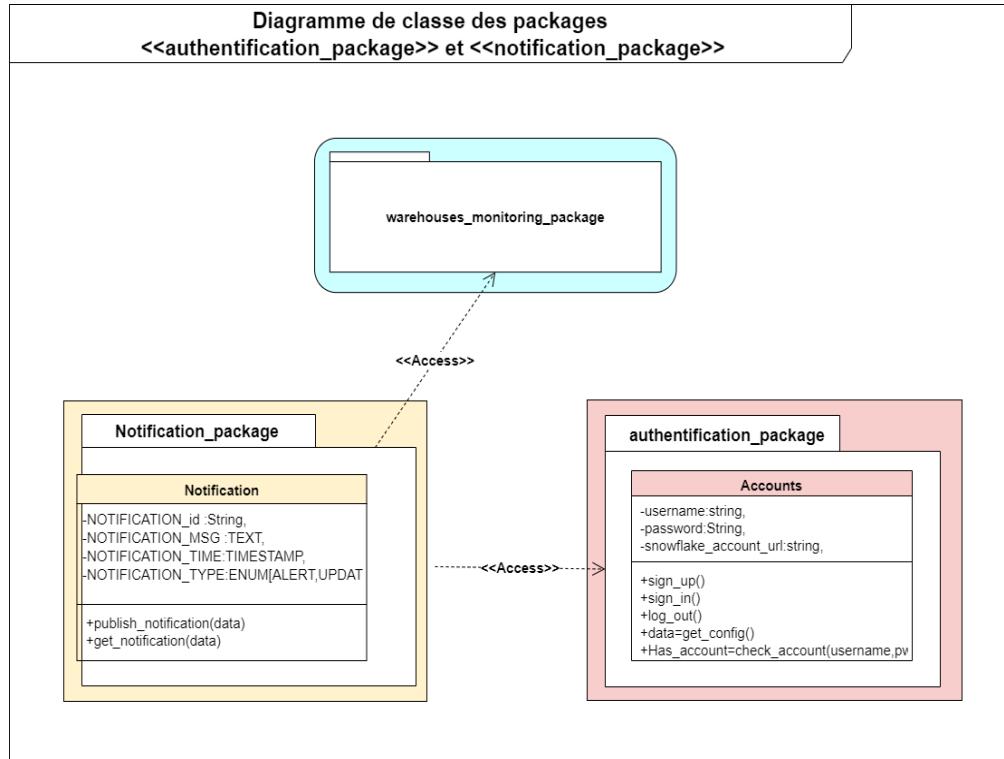
**FIGURE 3.3 : Diagramme de paquetages de «Snowflake Monitoring Application»**

### 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[22]. 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.

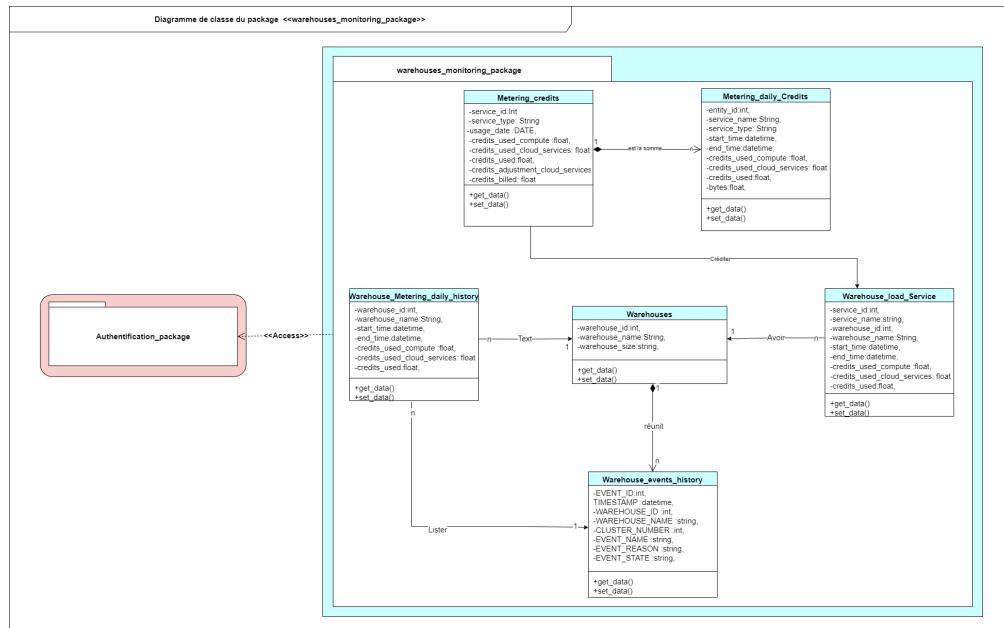
Dans cette partie, nous allons lister le diagrammes de classe de chaque package.

- Packages «Authentification» et «Notification» : la figure 3.4 qui suit représente le diagramme de classe des deux packages «authentification» et «Notification» :



**FIGURE 3.4 :** le diagramme de classe des deux packages «authentification» et «Notification»

- **Package «Warehouse\_monitoring» :** la figure 3.5 qui suit représente le diagramme de classe du package «Warehouse\_monitoring» :



**FIGURE 3.5 :** le diagramme de classe du package «Warehouse\_monitoring»

- Package «Dag&tasks» : la figure 3.6 qui suit représente le diagramme de classe du package «Dag&tasks» :

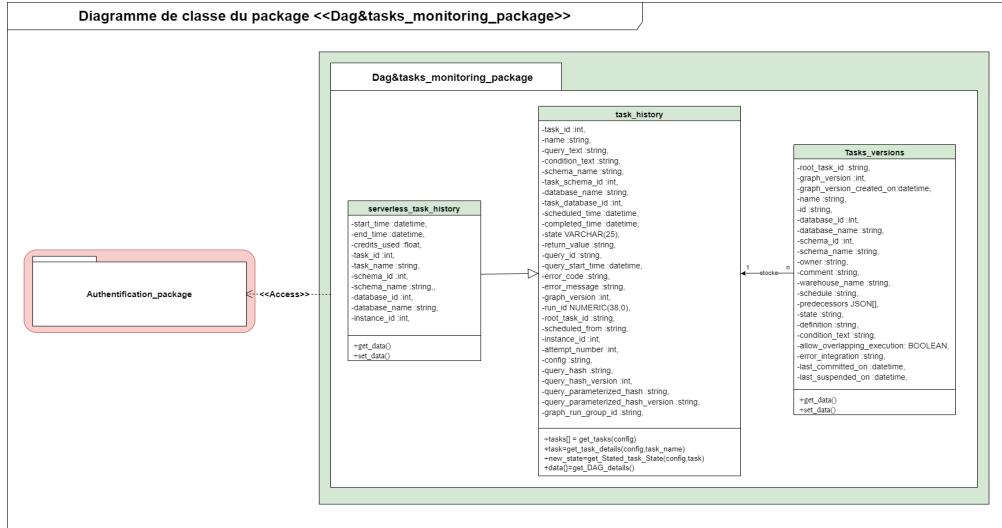


FIGURE 3.6 : le diagramme de classe du package «Dag&tasks»

- Package «Databases\_statistics» : la figure 3.7 qui suit représente le diagramme de classe du package «Databases\_statistics» :

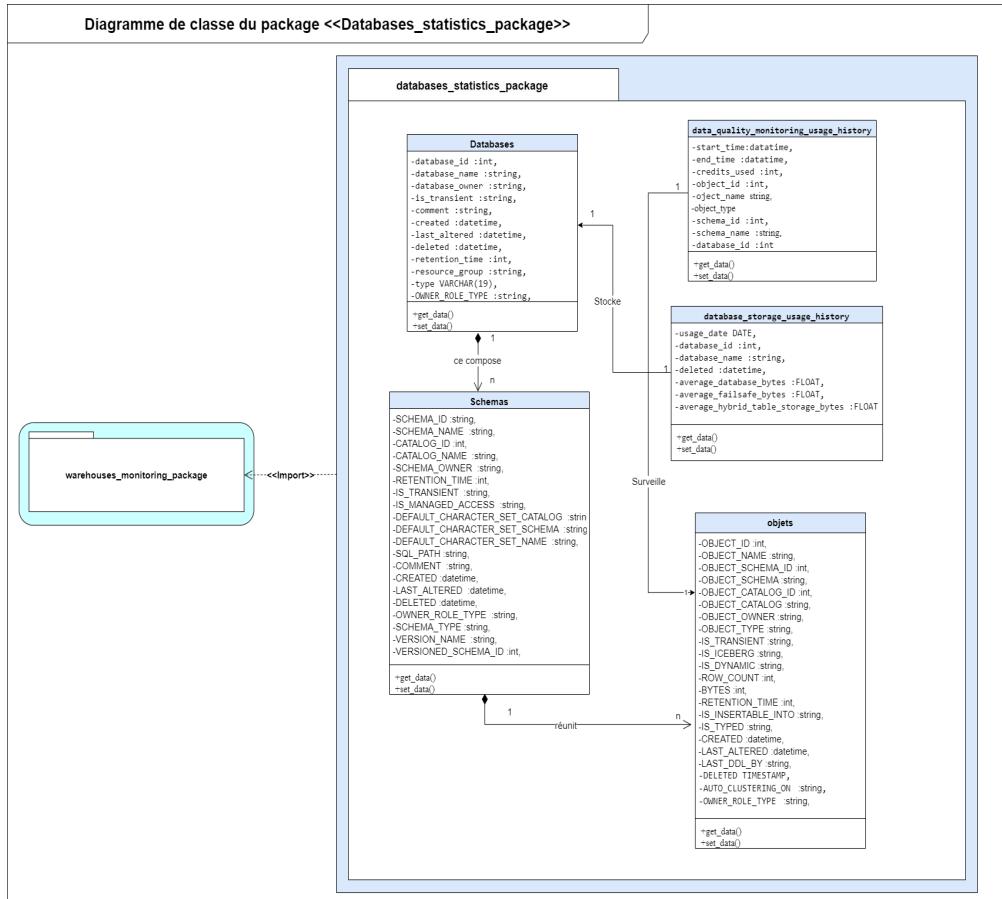


FIGURE 3.7 : le diagramme de classe du package «Databases\_statistics»

- Package «Users\_access» : la figure 3.8 qui suit représente le diagramme de classe du package «Users\_access» :

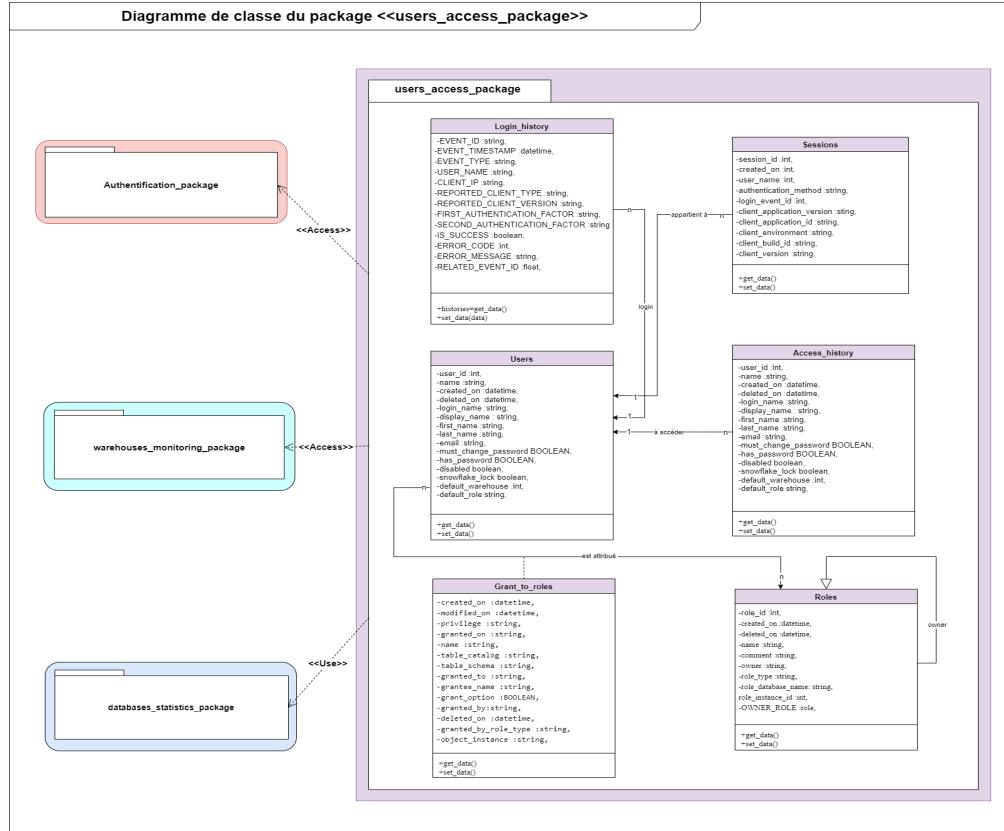


FIGURE 3.8 : le diagramme de classe du package «Users\_access»

- Package «Query\_performance» : la figure 3.9 qui suit représente le diagramme de classe du package «Query\_performance» :

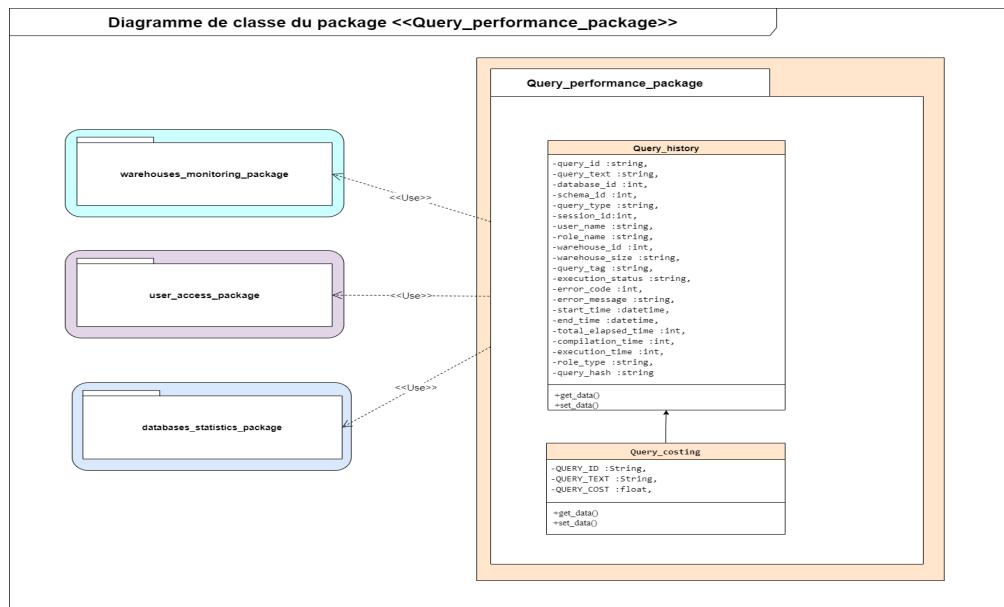


FIGURE 3.9 : le diagramme de classe du package «Query\_performance»

### 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[23].

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

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

<b>Titre</b>	Lister les requêtes SQL
<b>Résumé</b>	L'utilisateur souhaite de lister les requêtes SQL
<b>Acteurs</b>	Utilisateur
<b>Pré conditions</b>	L'utilisateur est authentifié et autorisé, via une token, à accéder à cette fonctionnalité
<b>Scénarios Nominaux</b>	<ol style="list-style-type: none"> <li>1. l'utilisateur sélectionne l'option «&lt;Queries&gt;»;</li> <li>2. le système récupère la liste des requêtes SQL exécutées par cet utilisateur de la base de données Monitoring_DB ;</li> <li>3. le système affiche les informations détaillées sur chaque requête (ID, type, status, etc.);</li> <li>4. l'utilisateur peut consulter les détails de chaque requête.</li> </ol>
<b>Scénario alternatif</b>	2.a. [«Aucune requête trouvée »] : le système retourne une liste vide en indiquant qu'il y aucune ligne
<b>Scénarios d'exceptions</b>	[«Token expirée»] : le système signale l'erreur et redirige l'utilisateur vers la page de «login».
<b>Post conditions</b>	l'utilisateur a accès à la liste des entrepôts de données et peut consulter leurs informations détaillées.

**TABLEAU 3.2 :** description textuelle de cas d'utilisation «Lister les requêtes SQL»

**Il faut mentionner que :** Si l'utilisateur est un **administrateur** le deuxième point de scénario nominal est modifiée. Car dans ce cas, le système va retourner la liste de tout les requêtes SQL exécutées par tout le monde.

**3. Description textuelle du cas d'utilisation «Lister l'historique des journaux d'accès au entrepôt des données» :**

<b>Titre</b>	Lister l'historique des journaux d'accès au entrepôt des données
<b>Résumé</b>	L'administrateur souhaite de lister l'historique des journaux d'accès à un entrepôt des données
<b>Acteurs</b>	Administrateur
<b>Pré conditions</b>	L'administrateur est authentifié et avoir le role autorisé, via une token, à accéder à cette fonctionnalité
<b>Scénarios Nominaux</b>	<ol style="list-style-type: none"> <li>1. l'administrateur sélectionne l'option «&lt;warehouse Monitoring» ;</li> <li>2. le système récupère les données nécessaires pour afficher le tableau de bord des entrepôts de donnés ;</li> <li>3. l'administrateur sélectionne un entrepôt parmis la liste des entrepôts ;</li> <li>4. le système renvoi les données spécifiques à cet entrepôt et liste ces journaux d'accées ainsi leurs détails dans le tableau de bord.</li> </ol>
<b>Scénario alternatif</b>	2.a. [«Aucun entrepôt trouvé »] : le système retourne un tableau de bord vide en indiquant qu'il y a pas de données à monitorer.
<b>Scénarios d'exceptions</b>	[«Token éxpirée»] : le système signale l'erreur et redirekte l'administrateur vers la page de «login».
<b>Post conditions</b>	L'administrateur a accès à la liste des journaux d'accès au entrepôt des données leurs informations détaillées.

**TABLEAU 3.3 :** description textuelle de cas d'utilisation «Lister l'historique des journaux d'accès au entrepôt des données»

**4. Description textuelle du cas d'utilisation «Lister les bases de données, les schémas et les objets» :**

<b>Titre</b>	Lister les bases de données et leurs objets
<b>Résumé</b>	L'utilisateur souhaite de lister les bases de données et leurs objets
<b>Acteurs</b>	utilisateur
<b>Pré conditions</b>	L'utilisateur est authentifié et autorisé, via une token, à accéder à cette fonctionnalité
<b>Scénarios Nominaux</b>	<ol style="list-style-type: none"> <li>1. l'utilisateur sélectionne l'option «Databases» ;</li> <li>2. le système récupère la liste des bases de données existants dans le compte Snowflake de l'utilisateur en affichant les détails de chaque base (Id, nom, date de création, etc.) ;</li> <li>3. l'utilisateur sélectionne une base de données de la liste ;</li> <li>4. le système récupère l'ID de cette base et lance une recherche dans la base de données Monitoring_DB des schémas appartenant à cette base ;</li> <li>5. le système affiche la liste des schémas de données de la base sélectionnée avec leurs détails ;</li> <li>6. l'utilisateur sélectionne un schéma d'après la liste des schémas pour lister les objets contenant dans cet schéma ;</li> <li>7. le système récupère l'ID de cet schéma et lance une recherche dans la base de données Monitoring_DB des objets appartenant à ce schéma de données ;</li> <li>8. le système affiche la liste des objets avec tout les détails.</li> </ol>
<b>Scénario alternatif</b>	2.a. [«Aucune base de données trouvée »] : le système retourne une liste des bases de données vide. 4.a. [«Aucun schéma trouvé »] : le système retourne une liste des schémas vide. 7.a. [«Aucun objet trouvé »] : le système retourne une liste des objets vide.
<b>Scénarios d'exceptions</b>	[«Token expirée»] : le système signale l'erreur et redirige l'utilisateur vers la page de «login».
<b>Post conditions</b>	L'utilisateur a accès aux listes des bases de données, des schémas et des objets ainsi que leurs informations détaillées.

**TABLEAU 3.4 :** description textuelle de cas d'utilisation «Lister les bases de données, les schémas et les objets»

### 3.2.2.2 Diagramme d'activité du cas d'utilisation «Récupération des données» :

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[24].

Le diagramme d'activité du cas d'utilisation ETL-service est représenté dans la figure 3.10 suivante :

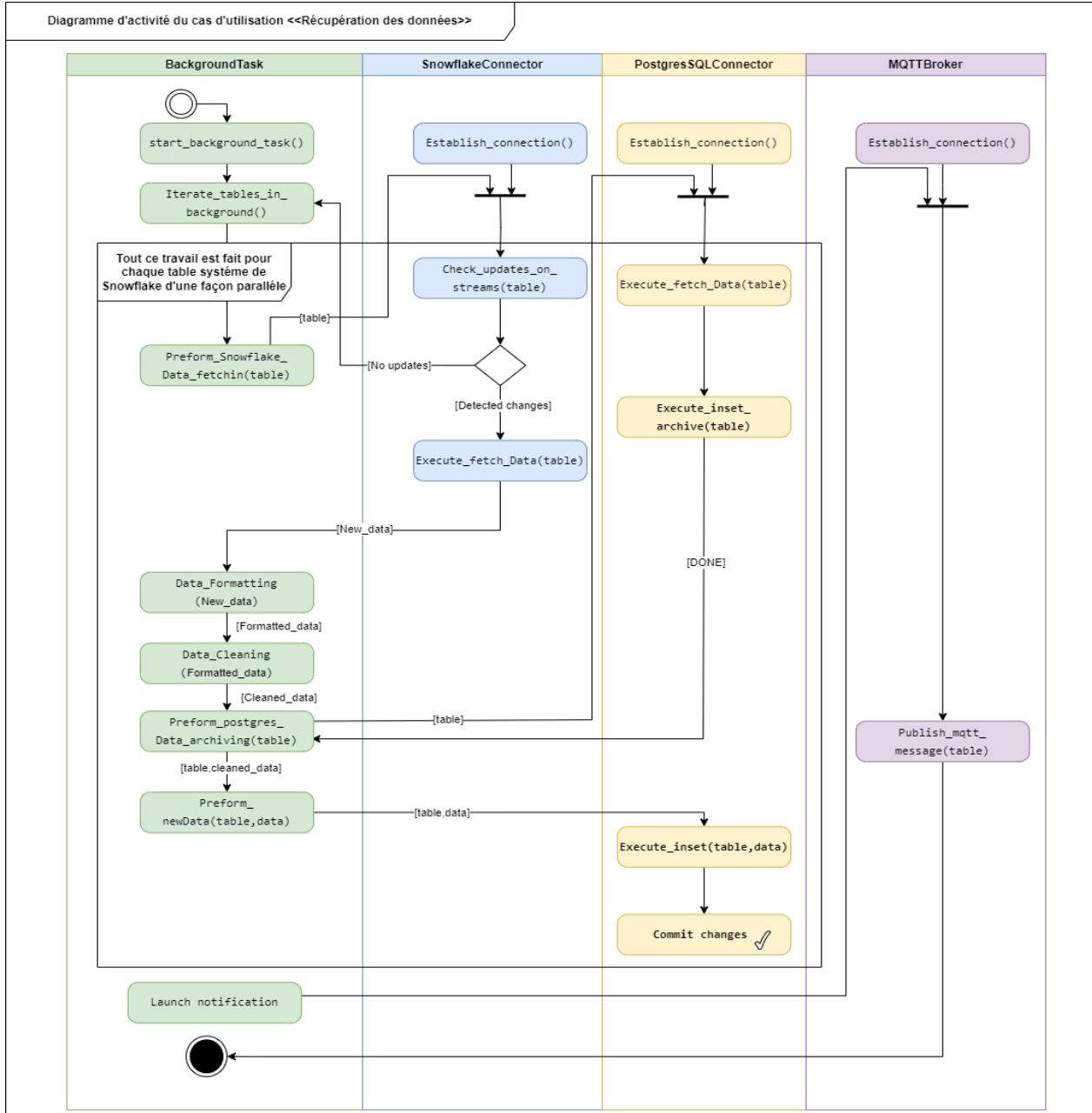


FIGURE 3.10 : Diagramme d'activité du cas d'utilisation «Récupération des données» »

Le traitement du cas d'utilisation «Récupération des données» commence par l'appel la méthode «`start_background_task()`», qui est une tâche programmée automatiquement (cron job) chaque 24 ou 12 heures selon la préférence de l'administrateur. 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 sa part connecte notre service avec le compte Snowflake dont nous souhaitons monitorer 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 a 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, la tâche 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.

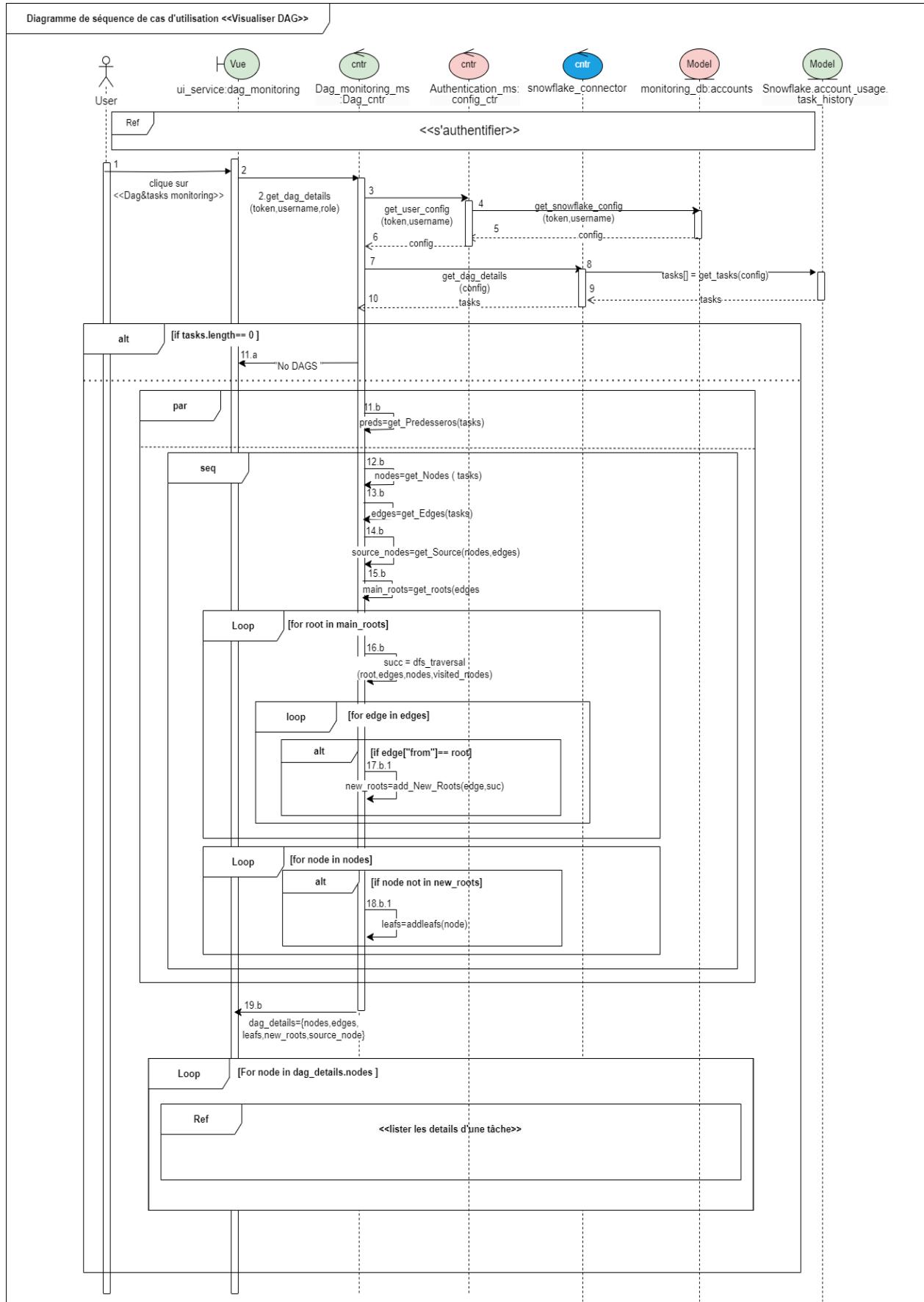
Enfin, le microservice ETL-service termine son traitement en attendant son prochain déclanchement.

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

Un diagramme de séquence est un type de diagramme d'interaction, car il décrit comment et dans quel ordre plusieurs objets fonctionnent ensemble [25].

Le diagramme de séquence de cas d'utilisation «Visualiser DAG» est représenté dans la figure 3.11 suivante :

## Chapitre 3. Architecture et conception



**FIGURE 3.11 : 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 se 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&tasks\_monitoring» pour récupérer les données spécifiques à chaque 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'autorisation.

Le contrôleur snowflake config\_ctrl du service d'authentification renvoie les configurations nécessaires au contrôleur principal.

Lorsque 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 longueur de la liste égal à 0];** 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\_transfersal) 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 ;

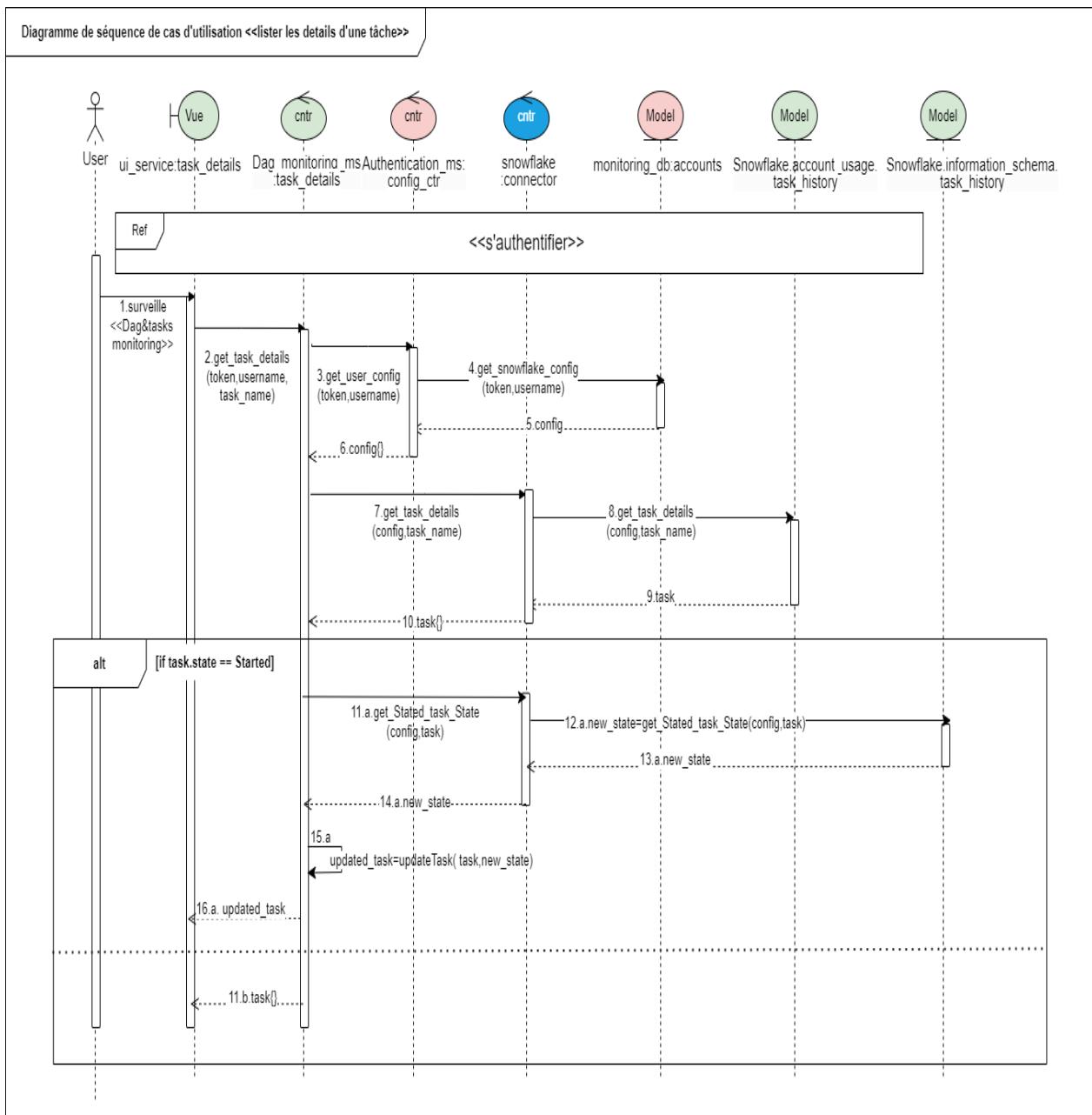
-Finalement, il renvoie toutes les informations traitées aux services前端 comme étant un objet JSON.

Enfin, le service frontend va lui même parcourre les noeuds de l'objet retourné, en dessinant le DAG finale en faisant appel au contrôleur Task\_details (cas d'utilisation «lister les details d'une tâche») inclus du même service pour afficher en details chaque noeud du DAG.

**NB :** Il faut bien préciser que le déclanchement de ce cas d'utilisation nécessite l'authentification de l'utilisateur et qu'il a les permissons et les rôles qui lui permet de visualiser les DAGs.

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

La figure 3.12 illustre le diagramme de séquence du cas d'utilisation «Lister les détaills d'une tâche» :



**FIGURE 3.12 :** Diagrammme de séquence de cas d'utilisation «Lister les détaills d'une tâche»

Pour Lister les détails 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 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 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 ;

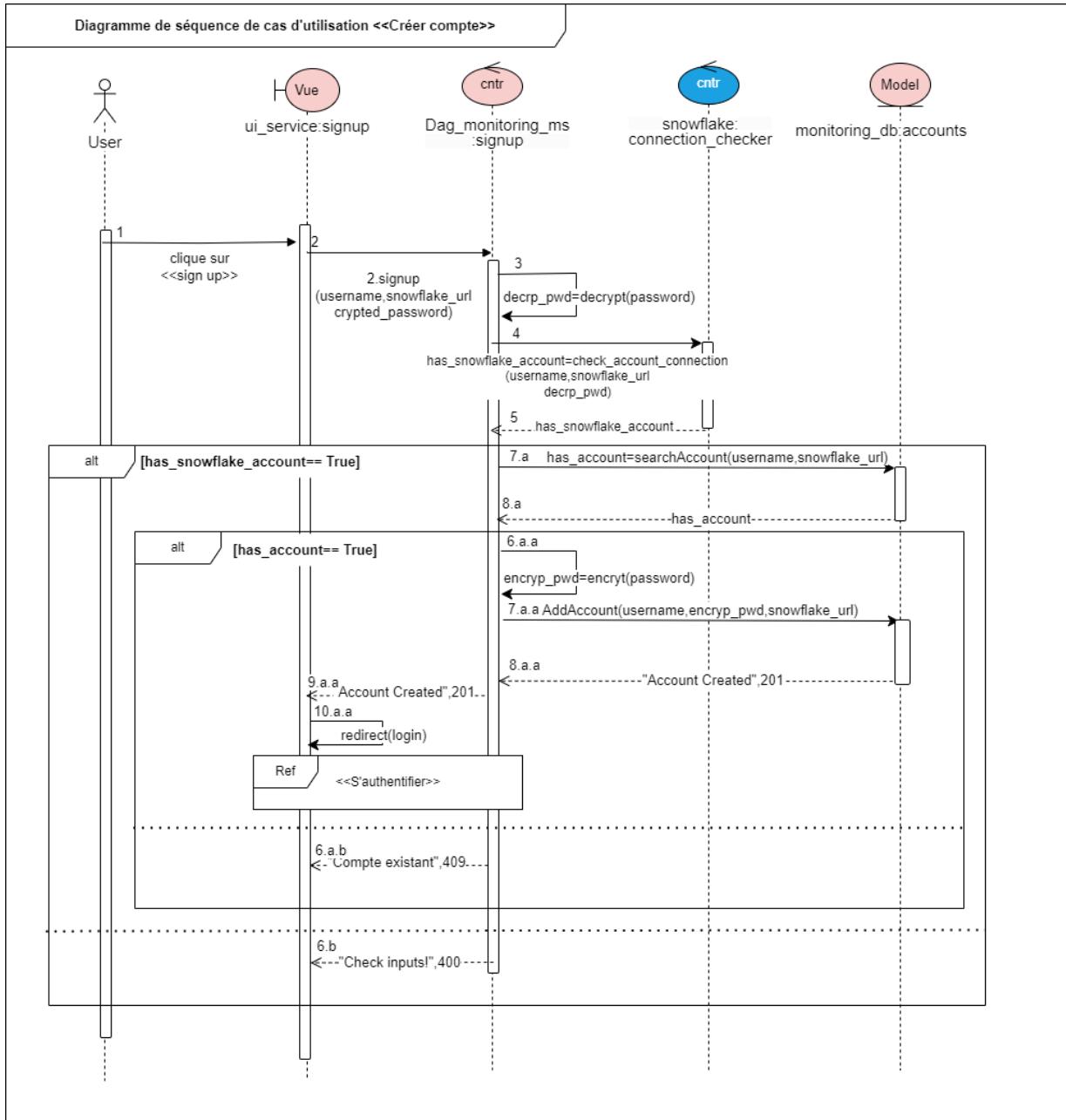
- **Si [L'état du tâche est «Started»] :**

- 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 a les informations sur les tâches commencées, pour récupérer l'état de cette tâche en temps réel 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 3.13 illustre le diagramme de séquence du cas d'utilisation «Créer un compte» :



**FIGURE 3.13 :** Diagramme de séquence de cas d'utilisation «Créer un compte»

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

En cliquant sur «signup», le contrôleur signup\_ctr 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'existence de ce compte dans Snowflake en retournant une réponse vers le contrôleur principal.

- [has\_snowflake\_account == True] :

-le contrôleur lance une recherche dans la table de monitoring\_DB accounts

-si [has\_account == True], 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.

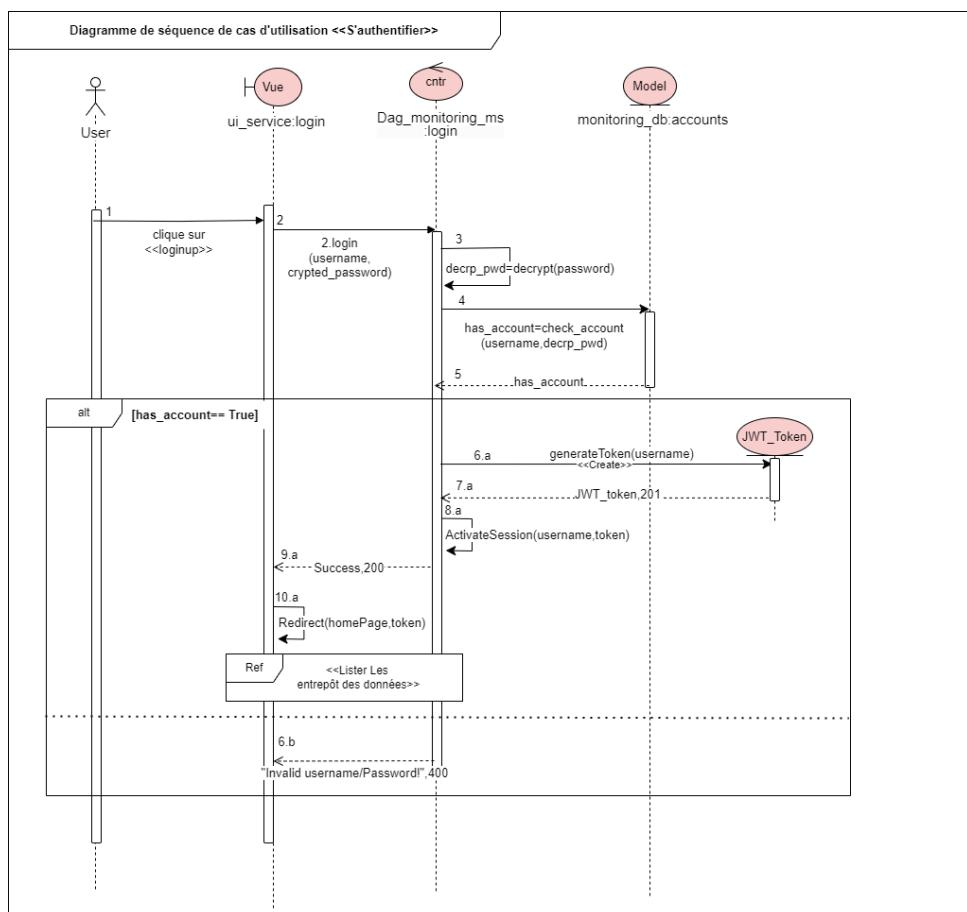
Enfin, il va renvoyer un message de succès au frontend qui va à son tour rediger l'utilisateur vers la page de l'authentification.

- [Sinon] :

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

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

La figure 3.14 illustre le diagramme de séquence du cas d'utilisation «S'authentifier» :



**FIGURE 3.14 :** Diagramme de séquence de cas d'utilisation «S'authentifier»

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

(nom d'utilisateur, mot de passe).

En cliquant sur «login», le contrôleur login\_ctr 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 ;

- **[Has\_account == 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 retournée 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 saisies sont invalides.

## Conclusion

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

# RÉALISATION

## Plan

<b>Introduction</b> . . . . .	<b>52</b>
1    Choix technologiques . . . . .	52
2    Réalisation . . . . .	55
<b>Conclusion</b> . . . . .	<b>86</b>

## Introduction

Le quatrième chapitre sera consacré à la mise en œuvre concrète du projet. Nous explorerons les choix techniques que nous avons adoptés et détaillerons le travail effectué pour donner vie à notre solution.

### 4.1 Choix technologiques

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

#### 4.1.1 Frameworks et base de données

- **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[26].

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.

- **ReactJS :**

React est une bibliothèque JavaScript destinée à la création d'interfaces utilisateur (UI) [27].

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.

- **PostgreSQL :**

PostgreSQL est un puissant système de gestion base de données relationnelle (SGBD) 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[28].

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.

#### **4.1.2 Outils de communication**

- **MQTT :**

Message Queuing Telemetry Transport (MQTT) est un protocole de messagerie basé sur des normes, ou un ensemble de règles, utilisé pour la communication de machine à machine [29].

Dans la réalisation de ce projet, nous avons opté pour MQTT comme étant un système de notification car il 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 :**

REpresentational State Transfer (REST) constitue un style architectural et un mode de communication fréquemment utilisé dans le développement de services Web [30].

Dans la réalisation de ce projet, nous avons opté pour RESTful API comme étant le mode de communiction 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.

#### **4.1.3 Bibliothèques pour l'analyse et la manipulation des données**

- **snowflake.connector :**

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.[31].

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 :**

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[32].

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 :**

Pandas est une bibliothèque de manipulation de données Python qui fournit des structures de données flexibles pour l'analyse des données[33].

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 :**

GMQTT est une bibliothèque de broker MQTT flexible et performante qui implémente entièrement le protocole MQTT V3.x et V5 en golang [34].

#### **4.1.4 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[35].

- **Scheduler :**

C'est une bibliothèque d'ordonnancement en python avec asyncio, threading ainsi que la prise en charge des fuseaux horaires. Elle permet de planifier 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 automatiser les tâches[36].

- **Requests :**

c'est un standard Python permettant d'effectuer des requêtes HTTP. Elle cache les subtilités des requêtes derrière une API simple, ce qui permet de concentrer sur l'interaction avec les services et la consommation de données dans l'application[37].

#### **4.1.5 Outils de test**

- **Postman :**

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 [38].

- **MQTTX :**

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[39].

#### **4.1.6 Outils de versionning**

- **Git :**

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)[40].

- **Github :**

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[41].

#### **4.1.7 Outils de documentation**

- **Swagger :**

Swagger est un outil spécialisé qui génère automatiquement la documentation des API RESTful des 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[42].

- **Latex :**

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[43].

- **Draw.io :**

Draw.io est une application en ligne gratuite, accessible via un navigateur web, 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 [44].

## **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 « ETL\_Service »

Le micro-service ETL\_Service est responsable de l'extraction, de la transformation et du chargement des données de Snowflake vers la base de données PostgreSQL de cette application.

Ce micro-service représente une tâche planifiée (cron job), il est programmée de s'exécuter quotidiennement (chaque 12 ou 24 heures par exemple) pour mettre à jour les données à moniter.

Dans le cas où la planification est atteinte, l'exécution du process ETL se commence par l'archivage des anciennes données pour garantir la disponibilité de la plateforme en cas d'échéance de la base de données principale. Ensuite, il va extraire les nouvelles données de chaque table système de Snowflake en faisant le nettoyage et la transformation sur ces derniers.

La figure 4.1 illustre les affichages retournés par le microservice :

```
>> New schedule has started by 2024-06-03 17:36:30.922162
----- ETL MICRO SERVICE HAS STARTED -----
>> Updating new data in progress ...
>> warehouse_load_history Archived successfully !
>> metering_history Archived successfully !
>> metering_daily_history Archived successfully !
>> warehouse_metering_history Archived successfully !
>> WAREHOUSE_EVENTS_HISTORY Archived successfully !
>> grants_to_roles Archived successfully !
>> roles Archived successfully !
>> users Archived successfully !
>> login_history Archived successfully !
>> sessions Archived successfully !
>> databases Archived successfully !
>> database_storage_usage_history Archived successfully !
>> SCHEMATA Archived successfully !
>> serverless_task_history Archived successfully !
>> tables Archived successfully !
>> data_quality_monitoring_usage_history Archived successfully !
>> query_history Archived successfully !
>> access_history Archived successfully !
>> OBJECT_DEPENDENCIES Archived successfully !
----- Tables Archived/ created in Postgres database -----

>> New data inserted successfully in warehouse_load_history !
>> New data inserted successfully in metering_history !
>> New data inserted successfully in metering_daily_history !
>> New data inserted successfully in warehouse_metering_history !
>> New data inserted successfully in WAREHOUSE_EVENTS_HISTORY !
>> New data inserted successfully in grants_to_roles !
>> New data inserted successfully in roles !
>> New data inserted successfully in users !
>> New data inserted successfully in login_history !
>> New data inserted successfully in sessions !
>> New data inserted successfully in databases !
>> New data inserted successfully in database_storage_usage_history !
>> New data inserted successfully in SCHEMATA !
>> New data inserted successfully in serverless_task_history !
>> New data inserted successfully in tables !
>> New data inserted successfully in data_quality_monitoring_usage_history !
>> New data inserted successfully in query_history !
>> New data inserted successfully in access_history !
>> New data inserted successfully in OBJECT_DEPENDENCIES !
----- Data updated in all system -----

>> Publishing notification in progress ...
Run time of job "run_etl_task (trigger: interval[0:02:00], next run at: 2024-06-03 17:40:30 WAT)" was missed by 0:00:37.508823
>> Notification published Successfully at 2024-06-03 17:39:18.422671

Time taken to process the request and return response is 167.50095295906067 sec
```

**FIGURE 4.1 : Exécution de l'ETL**

Enfin, il stocke les nouvelles données dans les tables appropriées et il publie un message dans le MQTT Broker pour notifier les utilisateurs du compte que les mises à jour sont faites avec succès comme l'indique la figure 4.2.

## Chapitre 4. Réalisation

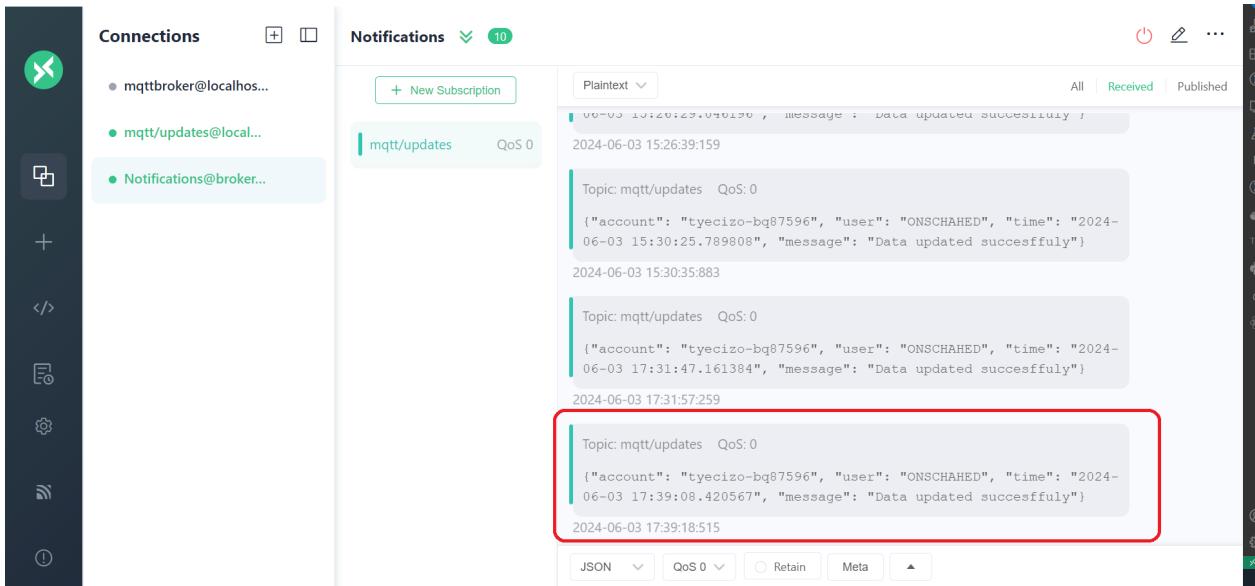


FIGURE 4.2 : La publication du message dans MQTT Broker

### 4.2.2 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 4.3 suivante :

The screenshot shows the FastAPI Swagger UI for the 'Authentification\_Service'. At the top, it says 'FastAPI 0.1.0 OAS 3.1 /openapi.json'. Below is a 'default' section with three POST methods: '/signup' (Signup), '/token' (Token), and '/logout' (Logout). Under 'Schemas', there are definitions for Body\_signup\_signup\_post, Body\_token\_token\_post, HTTPValidationError, and ValidationError, each with an 'Expand all' link.

FIGURE 4.3 : Liste des APIs du microservice «Athentification\_service»

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

L'interface de « Sign up» est représentée par la figure 4.4 suivante :

The screenshot shows the sign-up page for the Avaxia Snowflake Monitoring Application. At the top, there is a logo for 'Avaxia' with the subtitle 'Snowflake Monitoring Application'. Below the logo is a brief project description: 'Avaxia Snowflake operations Application aims to enable real-time data streaming, automated data updates, and seamless communication between different components. The goal is to enhance operational efficiency, provide actionable insights, and ensure data availability and integrity through scheduled tasks, proactive monitoring, and redundancy mechanisms. Ultimately, the project aims to empower stakeholders with timely, accurate information for informed decision-making.' A sub-header 'Sign up & Monitor your Snowflake account' is followed by a note: 'Be careful, you should sign up with your Snowflake account data.' The form fields include 'Username:' with the value 'ONSCHAHED', 'Password:' with a masked value, and 'Snowflake Account URL:' with the value 'https://tyecizo-bq87596.snowflakecomputing.com'. A large blue 'SIGN UP' button is at the bottom, and a link 'Already have an account? [Sign in](#)' is below it. To the right of the form is a large, semi-transparent watermark of a collage of various industrial and technological images.

**FIGURE 4.4 : Interface de Création de compte «Sign up»**

Afin de créer un compte, l'utilisateur accède à cette interface en remplissant un formulaire d'inscription avec les données spécifiques à son compte Snowflake (nom d'utilisateur, 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 4.5 suivante :

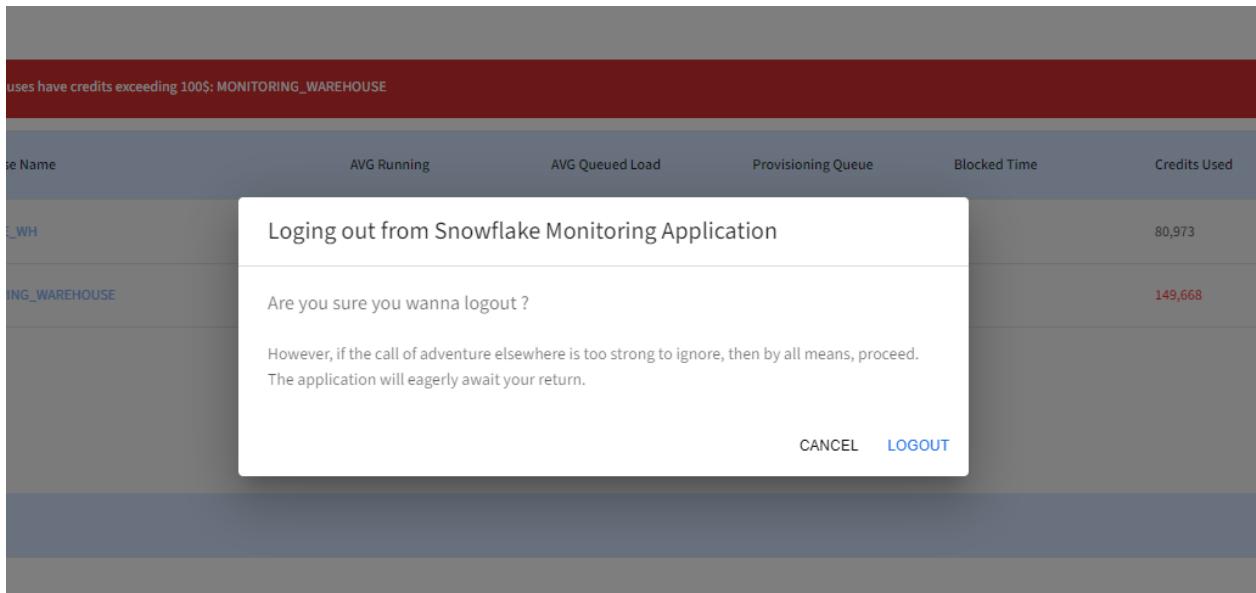
The screenshot shows the login page for the Avaxia Snowflake Monitoring Application. It features the same 'Avaxia' logo and project description as the sign-up page. The sub-header 'Sign in & Monitor your Snowflake account' and the note 'Be careful, you should sign in with your Snowflake account data.' are present. The form fields for 'Username:' (value 'ONSCHAHED') and 'Password:' (masked value) are identical to the sign-up page. A large blue 'SIGN IN' button is at the bottom, and a link 'Don't have an account? [Sign up](#)' is below it. The right side of the page contains the same large, semi-transparent watermark collage as figure 4.4.

**FIGURE 4.5 : Interface d'authentification «Login»**

Pour accéder a son compte, l'utilisateur remplit le formulaire avec son «nom d'utilisateur» et «mot de

pas». Une fois les données sont valides, le système attribut une token d'accées 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 4.6, il est redirigé vers l'interface de login automatiquement.



**FIGURE 4.6 : «Logout»**

s

En conclusion, ce microservice joue un rôle essentiel dans la sécurité et la fiabilité de l'application. En implémentant le protocole OAuth 2.0, il garantit une authentification et une autorisation robustes, permettant aux utilisateurs de se connecter en toute sécurité et d'accéder uniquement aux fonctionnalités et aux données auxquelles ils sont autorisés. Ce service agit comme un gardien de la confidentialité et de l'intégrité des données, assurant que seuls les utilisateurs légitimes puissent interagir avec le système.

#### 4.2.3 Micro-service « Warehouse\_Monitoring »

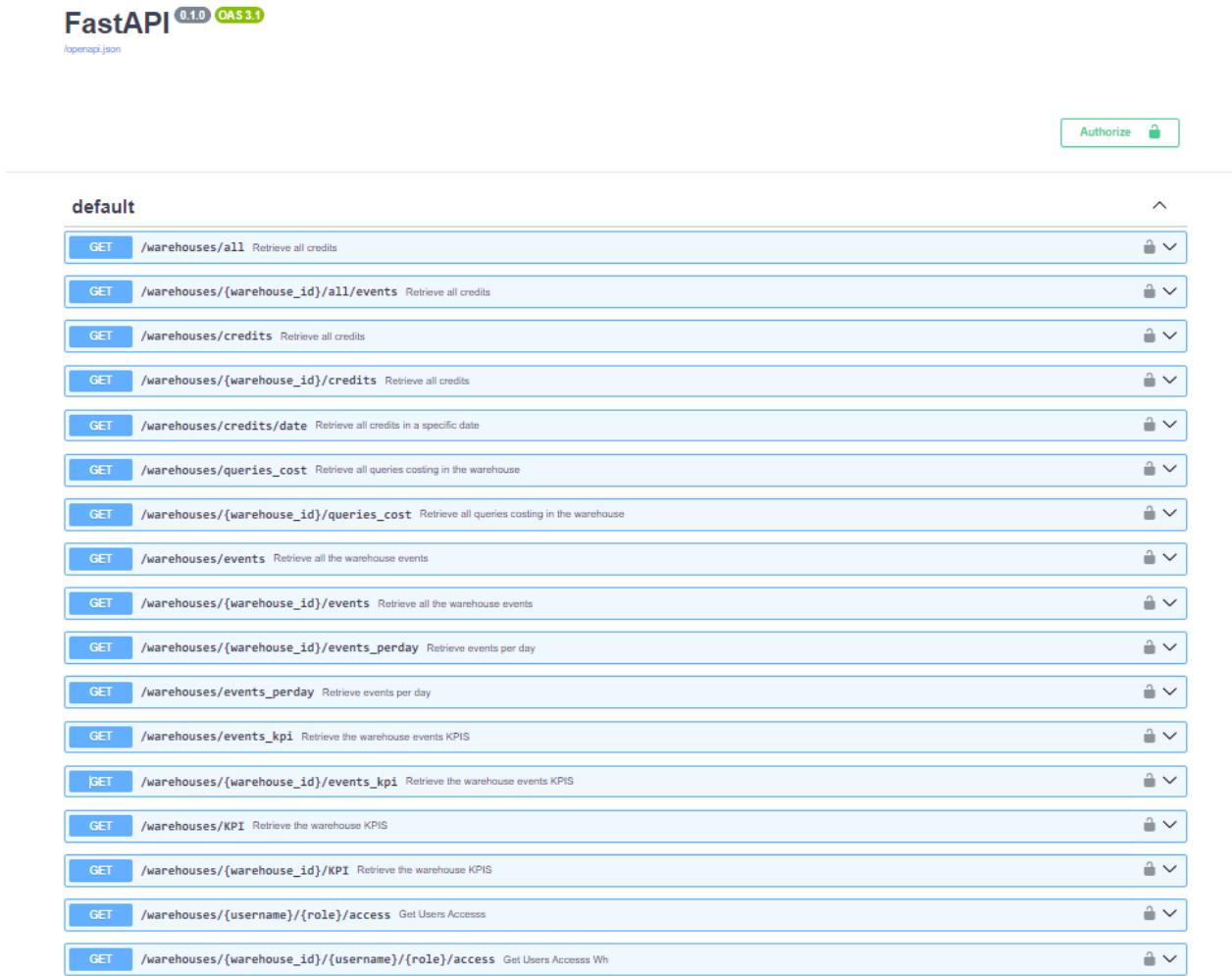
Le micro-service « Warehouse\_monitoring » est le service chargé de la surveillance de l'utilisation des entrepôts de données Snowflake.

- **La liste des APIs :**

La liste des APIs présents dans ce micro-service, documentée par Swagger, sont représentés par la figure 4.7 suivante :

## Chapitre 4. Réalisation

---



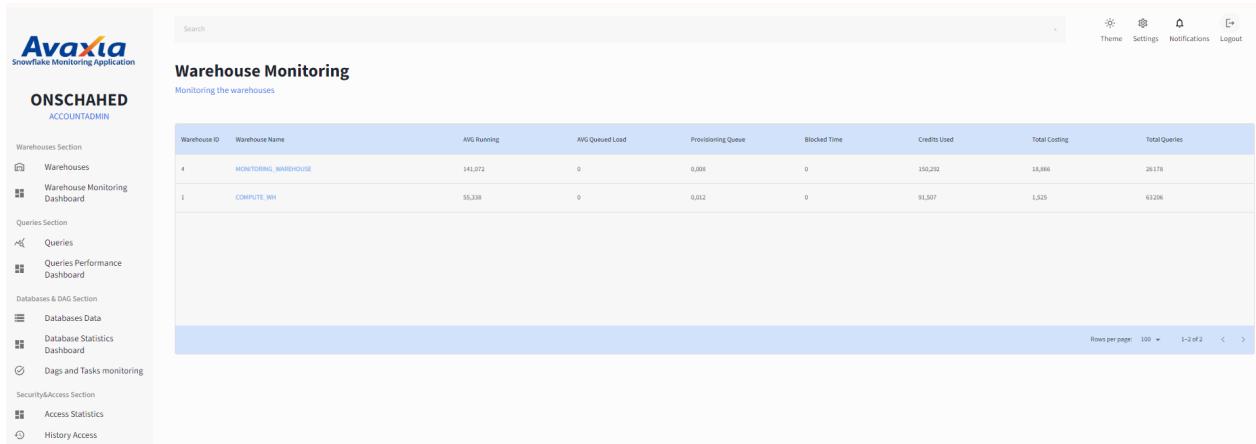
The screenshot shows the FastAPI documentation interface. At the top, it displays "FastAPI 0.1.0 OAS 3.1" and a link to "openapi.json". On the right side, there is a green "Authorize" button with a lock icon. Below the header, the word "default" is highlighted in blue. A list of API endpoints is shown, each with a "GET" method, a URL path, and a brief description. Most endpoints have a lock icon and a dropdown arrow to their right. The endpoints listed are:

- GET /warehouses/all Retrieve all credits
- GET /warehouses/{warehouse\_id}/all/events Retrieve all credits
- GET /warehouses/credits Retrieve all credits
- GET /warehouses/{warehouse\_id}/credits Retrieve all credits
- GET /warehouses/credits/date Retrieve all credits in a specific date
- GET /warehouses/queries\_cost Retrieve all queries costing in the warehouse
- GET /warehouses/{warehouse\_id}/queries\_cost Retrieve all queries costing in the warehouse
- GET /warehouses/events Retrieve all the warehouse events
- GET /warehouses/{warehouse\_id}/events Retrieve all the warehouse events
- GET /warehouses/{warehouse\_id}/events\_perday Retrieve events per day
- GET /warehouses/events\_perday Retrieve events per day
- GET /warehouses/events\_kpi Retrieve the warehouse events KPIs
- GET /warehouses/{warehouse\_id}/events\_kpi Retrieve the warehouse events KPIs
- GET /warehouses/KPI Retrieve the warehouse KPIs
- GET /warehouses/{warehouse\_id}/KPI Retrieve the warehouse KPIs
- GET /warehouses/{username}/{role}/access Get Users Accessss
- GET /warehouses/{warehouse\_id}/{username}/{role}/access Get Users Accessss Wh

**FIGURE 4.7 : 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 4.8 suivante :



The screenshot shows the Avaxia Snowflake Monitoring Application interface. The left sidebar has sections for "Warehouses Section" (Warehouses, Warehouse Monitoring Dashboard), "Queries Section" (Queries, Queries Performance Dashboard), "Databases & DAG Section" (Databases Data, Database Statistics Dashboard, Dags and Tasks monitoring), and "Security&Access Section" (Access Statistics, History Access). The main content area is titled "Warehouse Monitoring" and "Monitoring the warehouses". It shows a table with two rows of data:

Warehouse ID	Warehouse Name	Avg Running	Avg Queued Load	Provisioning Queue	Blocked Time	Credits Used	Total Costing	Total Queries
4	MONITORING_WAREHOUSE	141,072	0	0,008	0	150,212	18,866	26179
1	COMPUTE_WH	55,338	0	0,012	0	91,507	1,525	61206

At the bottom right of the table, there are buttons for "Rows per page: 100" and "1-2 of 2". The top right of the interface includes "Search", "Theme", "Settings", "Notifications", and "Logout" buttons.

**FIGURE 4.8 : Interface de la liste des entrepôts de données**

## Chapitre 4. Réalisation

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 4.9 et 4.10 :

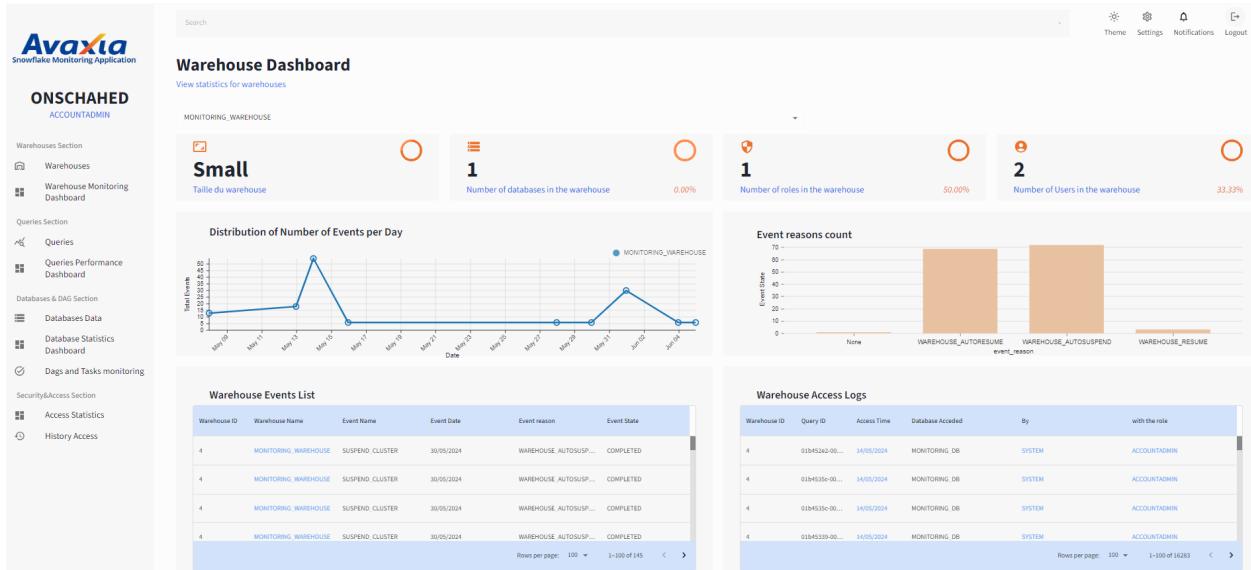


FIGURE 4.9 : Tableau de bord du surveillance d l'entrepôts des données «Monitoring\_warehouse»

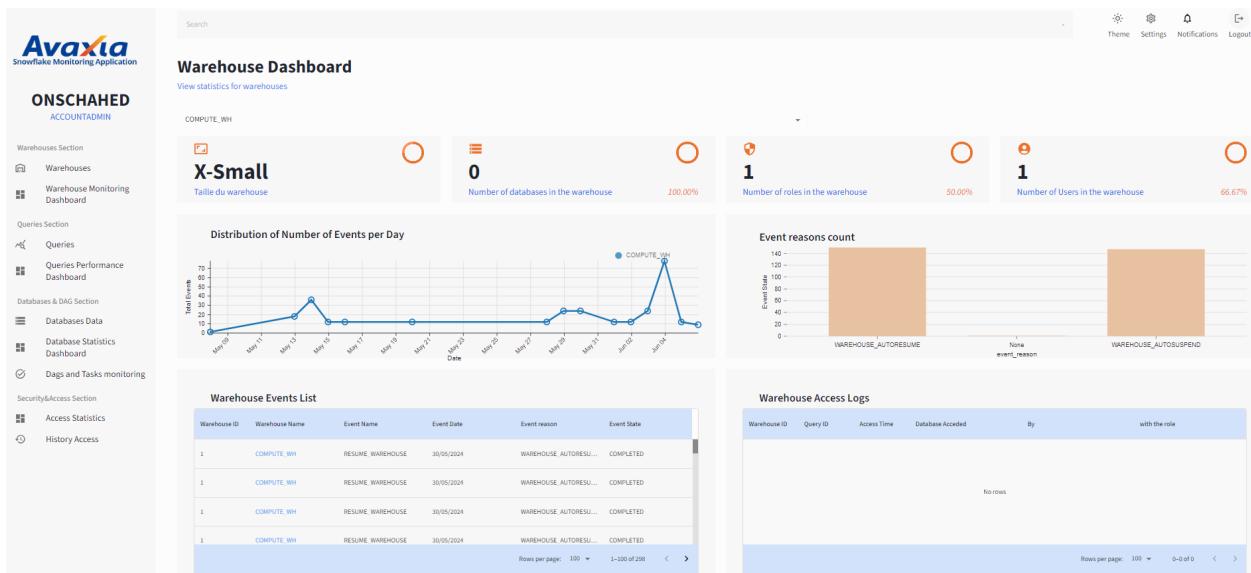


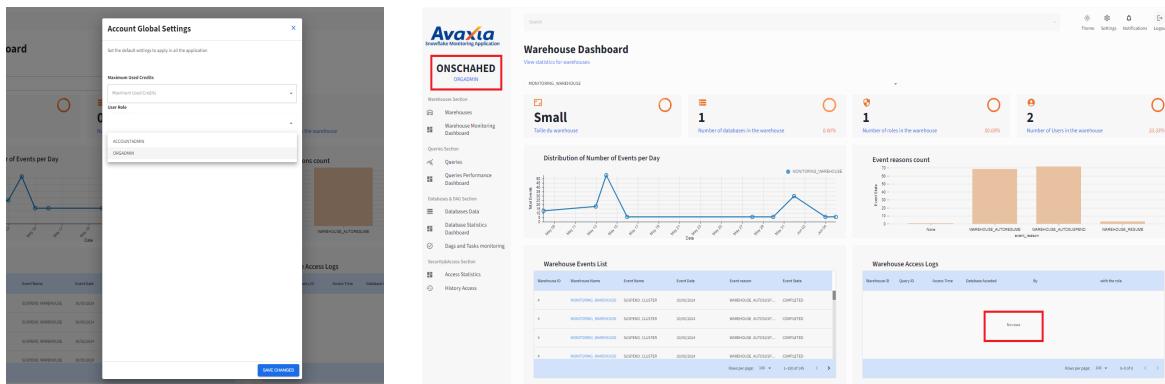
FIGURE 4.10 : Tableau de bord du surveillance d l'entrepôts des données «Compute\_WH»

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 événements effectuées dans l'entrepôt ainsi que le nombre de rôles et d'utilisateurs.

- **Changement des variables globaux**

Le changement des paramètres globaux peut反映er les vues de ce micro service, une fenêtre s'affiche lorsqu'on clique sur le bouton «Settings» du topbar. Selon les besoins de l'utilisateur/administrateur les changements qui peuvent être appliqués :

- **Changement du rôle d'utilisateur/Administrateur :**



**FIGURE 4.11 :** Changement du role de l'utilisateur de «ACCOUNT ADMIN» vers «ORGADMIN»

Lorsque l'utilisateur modifie les paramètres globaux, tel que 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	AWS Pending	AWS Queued Load	Provisioning Queue	Blocked Time	Credits Used	Total Costing	Total Queries
4	MONITORING_WAREHOUSE	135.20	0	0.000	0	123,800	728,200	1,027,200
1	COMPANY_WHL	32,727	0	0.004	0	40,750	33,200	236,237

**FIGURE 4.12 :** 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 utilisés de l'un des entrepôts de données à dépacer cette seuil pour informer l'utilisateur de cette malveillance.

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

#### 4.2.4 Micro-service « Query\_Performance »

Le micro-service « Query\_Performance » 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 4.13 suivante :

## Chapitre 4. Réalisation

---

The screenshot shows the FastAPI documentation interface for the 'Warehouse\_Monitoring' service. At the top, there is a header with the FastAPI logo, version 0.1.0, and OAS 3.1. Below the header, there is a navigation bar with a 'Authentique' button and a lock icon. The main content area is titled 'default' and contains a list of API endpoints. Each endpoint is represented by a blue box with a 'Get' method, a URL path, and a brief description. To the right of each endpoint is a dropdown arrow icon. The endpoints listed include:

- 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.

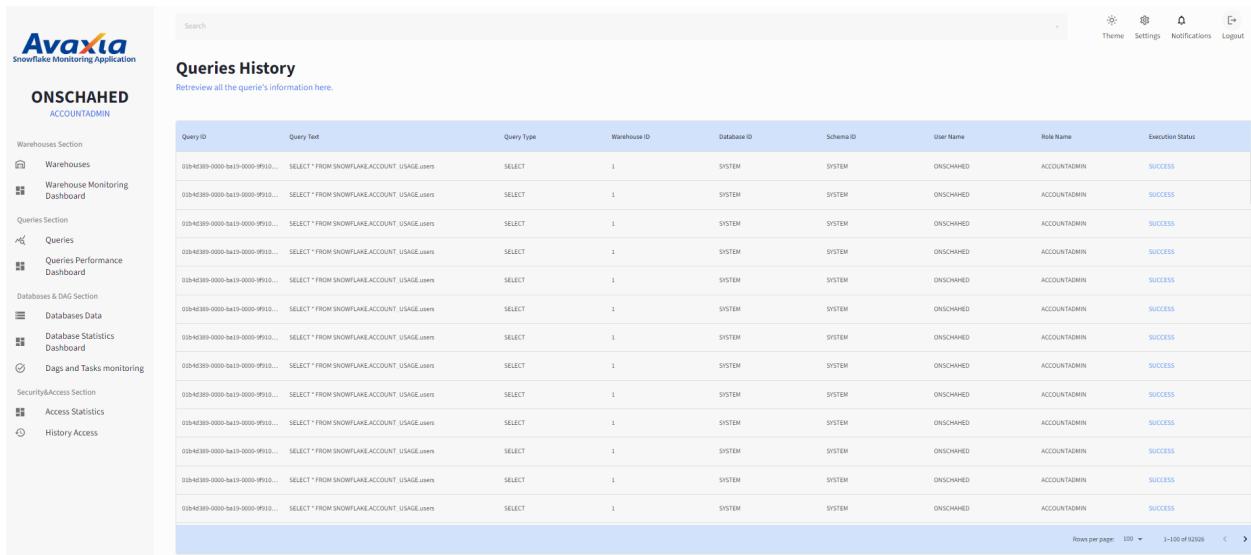
---

**FIGURE 4.13 :** Liste des APIs du microservice «Warehouse\_Monitoring»

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

L'interface de la liste des entrepôts de données est représentée par la figure 4.14 suivante :

## Chapitre 4. Réalisation



**FIGURE 4.14 :** Interface de la liste des entrepôts de données

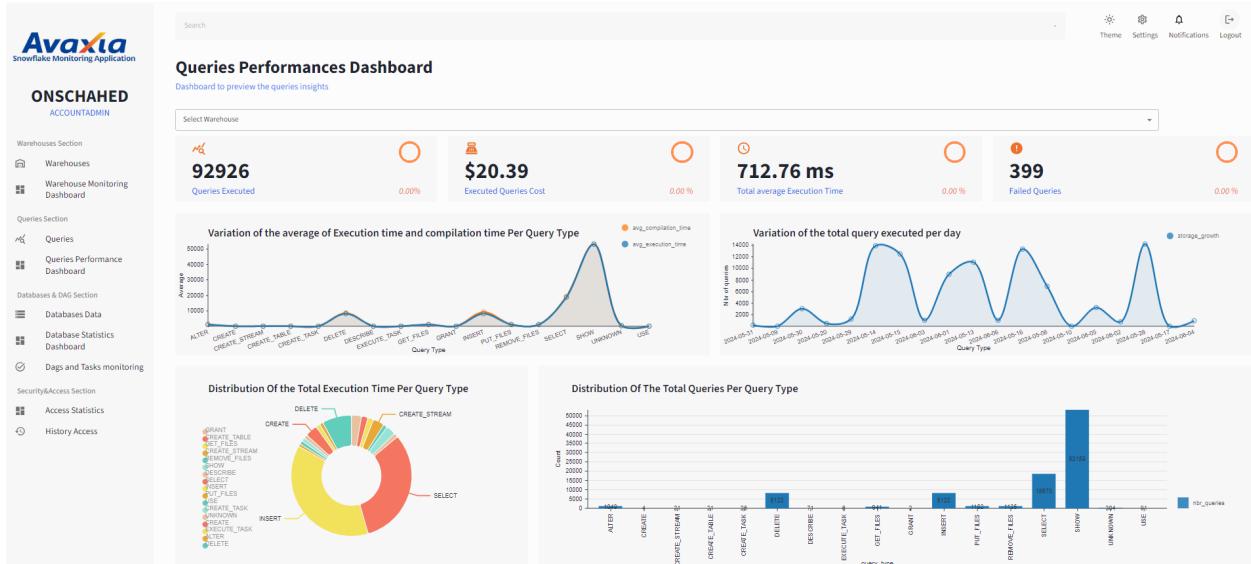
Cette vue représente l'historique des requêtes dans l'application de surveillance Snowflake. Elle fournit un aperçu détaillé de chaque requête exécutée.

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 4.15 illustre le tableau de bord global des requêtes SQL du compte Snowflake :

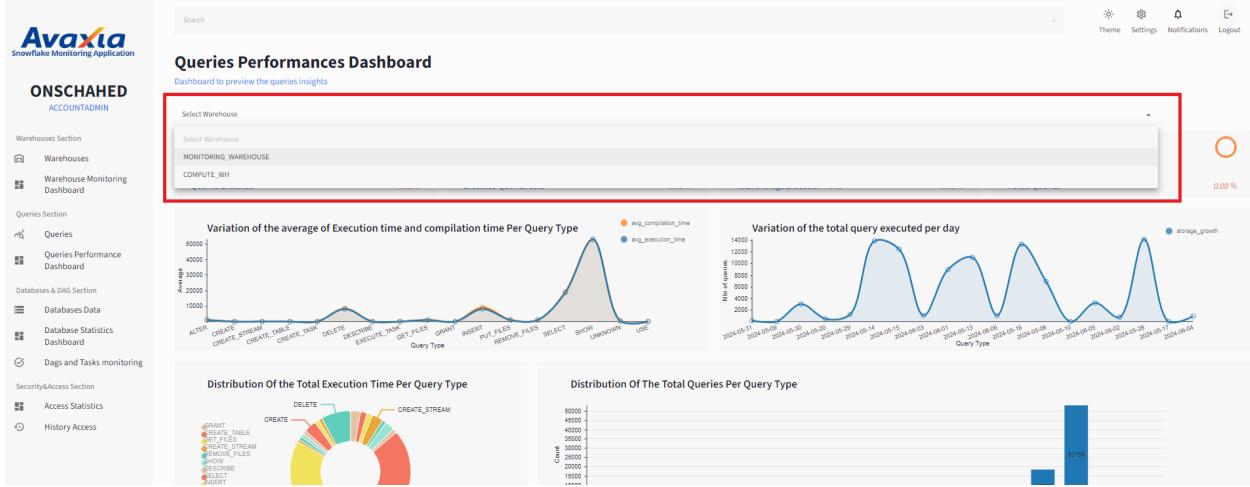


**FIGURE 4.15 :** Tableau de bord du surveillance des entrepôts des données

- **Selection de l'entrepôts des données appriori :**

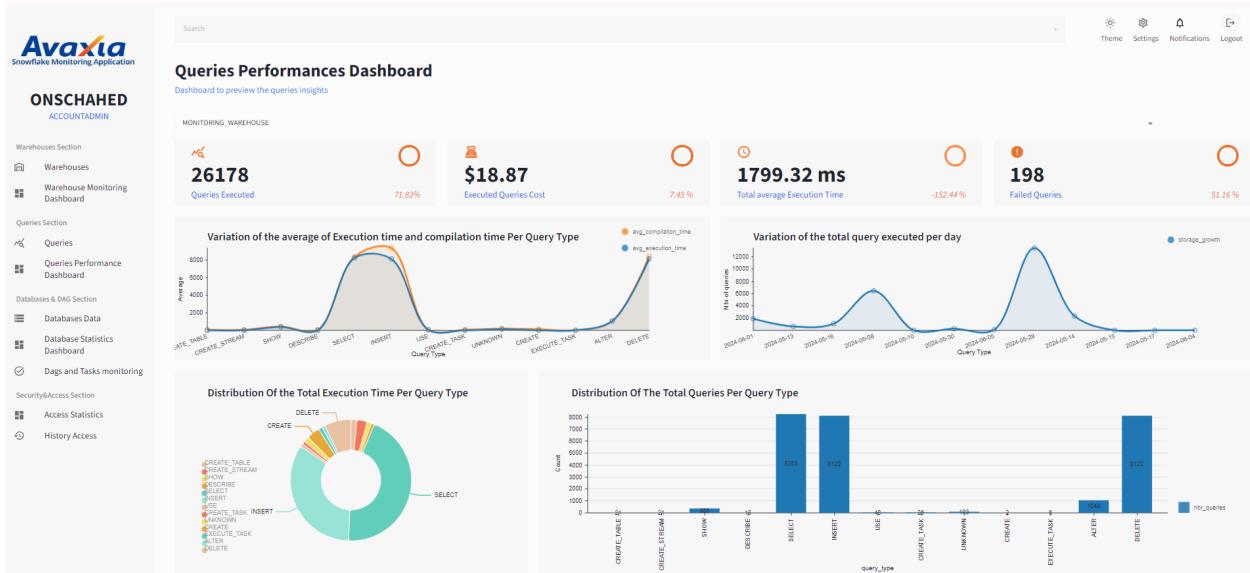
## Chapitre 4. Réalisation

Une fois l'utilisateur sélectionne 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 4.16, les données spécifie à cet entrepôt vont être affichées.



**FIGURE 4.16 :** Tableau de bord du surveillance des entrepôts des données

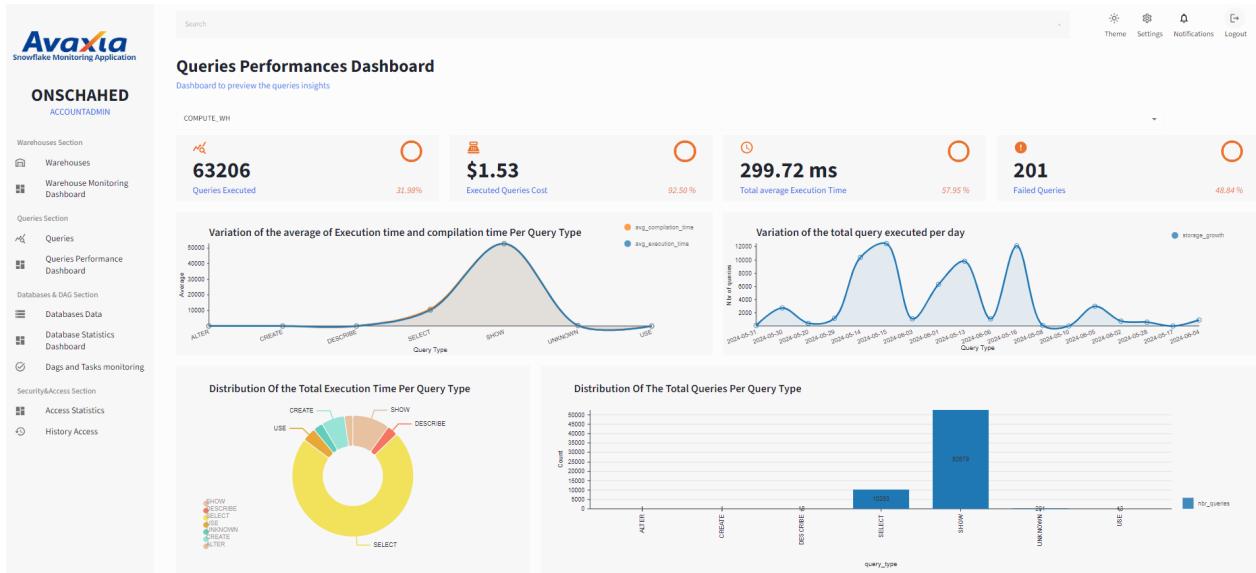
La figure 4.17 illustre le tableau de bord de l'entrepôt «Monitoring\_Warehouse» :



**FIGURE 4.17 :** Tableau de bord du surveillance de l'entrepôt des données «Monitoring\_Warehouse»

La figure 4.18 illustre le tableau de bord de l'entrepôt «Compute\_WH» :

## Chapitre 4. Réalisation



**FIGURE 4.18 :** 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.

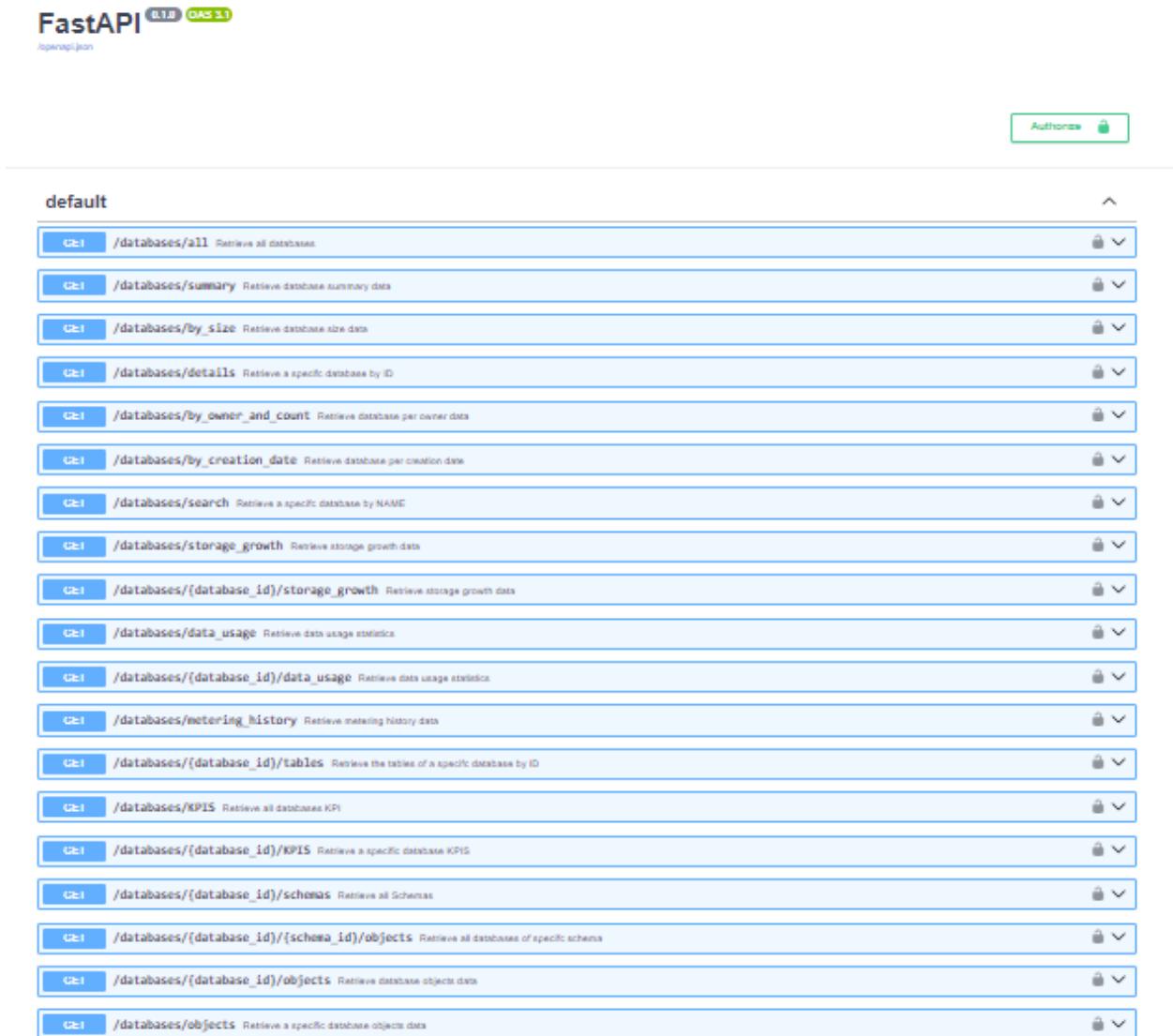
Le service « Query\_Performance » constitue un outil essentiel pour les équipes en charge de l'exploitation et de l'optimisation de l'infrastructure Snowflake. Il offre une visibilité complète sur les activités de requêtes, permettant de prendre des décisions éclairées afin d'améliorer la fiabilité,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

### 4.2.5 Micro-service « Databases\_Statistics »

Le micro-service « Databases\_Statistics » est le service chargé de la surveillance des base 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 4.19 suivante :



Method	Path	Description	Lock
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 KPI	🔒
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 4.19 : Liste des APIs du microservice « Databases\_Statistics »

- **La liste des bases de données et ses différents schémas :**

L'interface de la liste des bases de données et ses différents schémas est représentée par la figure 4.20 suivante :

## 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 type="checkbox"/>	7	MONITORING_DB	ACCOUNTADMIN	2024-05-08 08:45:07	2024-05-08 08:45:07	STANDARD

Rows per page: 100 ▾ 1-3 of 3 < >

**List of Schemas**

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

Rows per page: 100 ▾ 0-0 of 0 < >

**List of Objects**

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

Rows per page: 100 ▾ 0-0 of 0 < >

**FIGURE 4.20 :** 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 sélectionne une base de données, la liste des schémas s'affiche comme l'indique la figure 4.21 suivante :

**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 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

Rows per page: 100 ▾ 1-2 of 2 < >

**List of Objects**

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

Rows per page: 100 ▾ 0-0 of 0 < >

**FIGURE 4.21 :** Liste des schémas de base de données

Ensuite, pour que nous puissions lister les objets présents et créées dans cette base de données, l'utilisateur sélectionne le schéma apprivoisé pour lister ces objets.

La figure 4.22 illustre 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_SF	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_HIST...	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_SCHE...	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

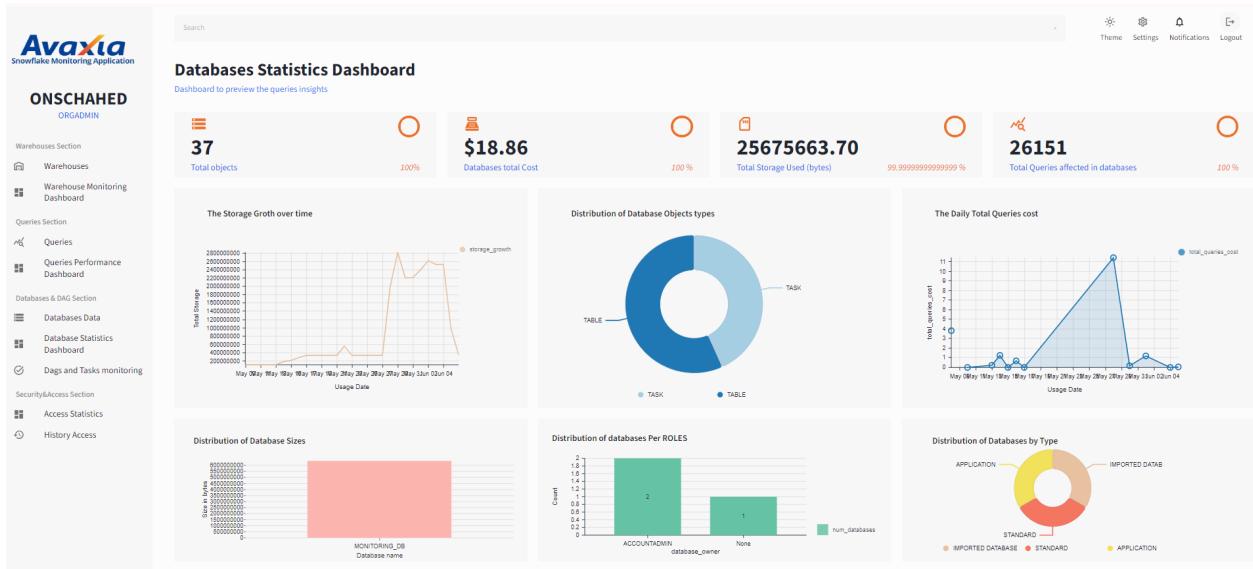
Rows per page: 100 ▾ 1-37 of 37 < >

**FIGURE 4.22 : 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 4.23 représente le tableau de bord du surveillance des bases de données :



**FIGURE 4.23 : 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.

#### 4.2.6 Micro-service « Access\_Statistics »

Le micro-service « Access\_Statistics » est le service chargé de la surveillance des accès aux comptes Snowflake.

- **La liste des APIs :**

La liste des APIs présents dans ce micro-service, documentée par Swagger, sont représentés par la figure 4.24 suivante :

The screenshot shows the FastAPI Swagger UI interface. At the top, it displays "FastAPI 0.1.0 OAS3.1" and a link to "/openapi.json". On the right side, there is a green "Authorize" button with a lock icon. Below the header, the word "default" is displayed in bold. A list of API endpoints is shown in a table format:

Method	Endpoint	Description	Lock
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 4.24 : Liste des APIs du microservice « Access\_Statistics »

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

L'interface de l'historique d'accés est représentée par la figure 4.25 suivante :

Access ID	Access Time	Event Type	Username	Access IP Address	Authentication Factor
175445119228062	2024-05-16 08:42:26	LOGIN	ONSCHAHED	41.224.4.121	PASSWORD
175445119191126	2024-05-15 13:21:08	LOGIN	ONSCHAHED	195.90.188.51	PASSWORD
175445119228130	2024-05-16 08:42:22	LOGIN	ONSCHAHED	41.224.4.121	PASSWORD
175445119191130	2024-05-15 13:21:21	LOGIN	ONSCHAHED	195.90.188.51	PASSWORD
175445119233390	2024-05-16 07:44:41	LOGIN	ONSCHAHED	41.224.4.121	PASSWORD
175445119191134	2024-05-15 13:21:36	LOGIN	ONSCHAHED	195.90.188.51	PASSWORD
175445119171124	2024-05-15 09:59:27	LOGIN	ONSCHAHED	197.31.232.164	PASSWORD
175445119233374	2024-05-16 07:44:25	LOGIN	ONSCHAHED	41.224.4.121	PASSWORD
175445119180902	2024-05-15 10:59:45	LOGIN	ONSCHAHED	197.31.232.164	PASSWORD
175445119221978	2024-05-16 07:27:36	LOGIN	ONSCHAHED	41.224.4.121	PASSWORD
175445119180898	2024-05-15 10:59:40	LOGIN	ONSCHAHED	197.31.232.164	PASSWORD
175445119232454	2024-05-16 07:35:52	LOGIN	ONSCHAHED	41.224.4.121	PASSWORD
175445119168326	2024-05-15 10:59:05	LOGIN	ONSCHAHED	197.31.232.164	PASSWORD

**FIGURE 4.25 : Historique des accés**

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 4.26 et 4.27 représentent les tableaux de bord du surveillance des accés de l'utilisateur «ONS CHAHED» et l'utilisateur système «Snowflake» :

## Chapitre 4. Réalisation

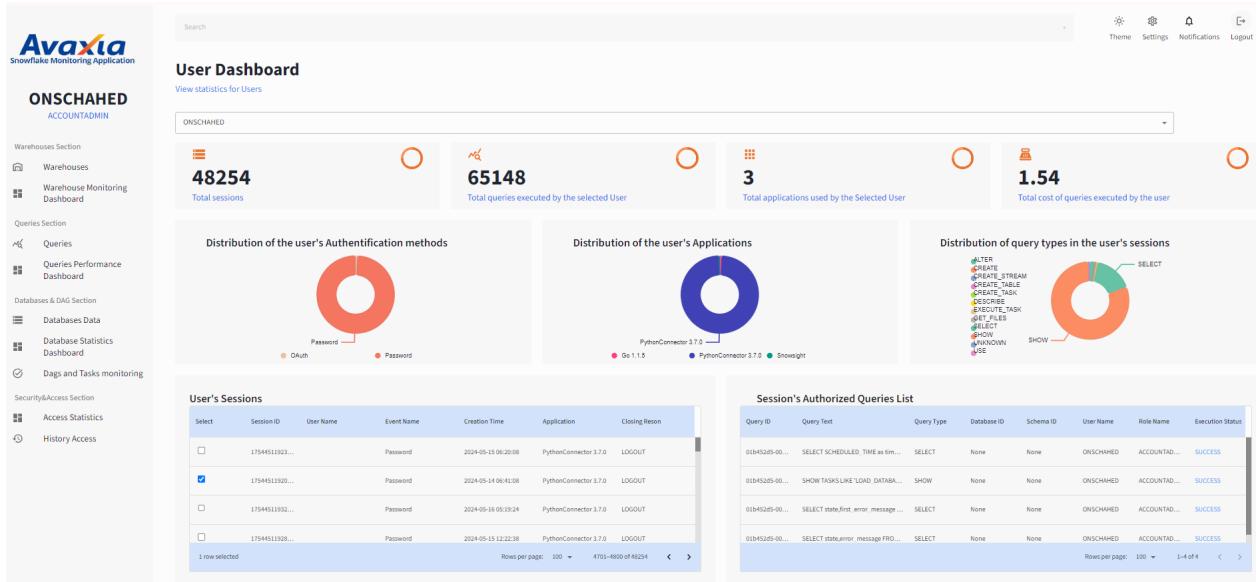


FIGURE 4.26 : Tableau de bord du surveillance des accès de l'utilisateur «ONS CHAHED»

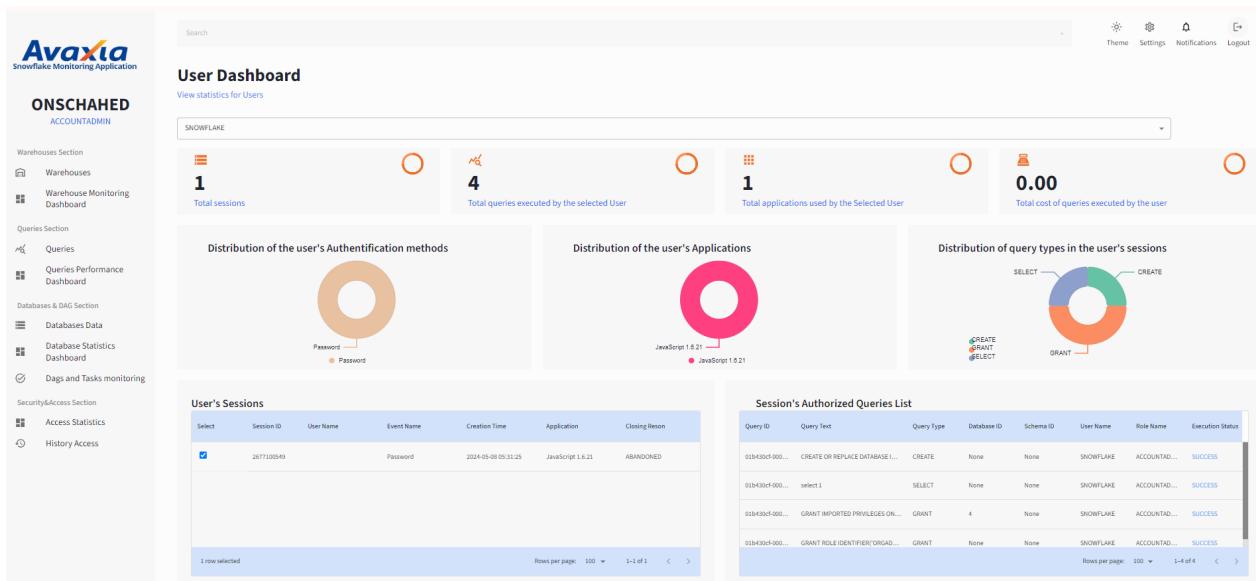


FIGURE 4.27 : Tableau de bord du surveillance des accès de l'utilisateur «SNOWFLAKE»

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.

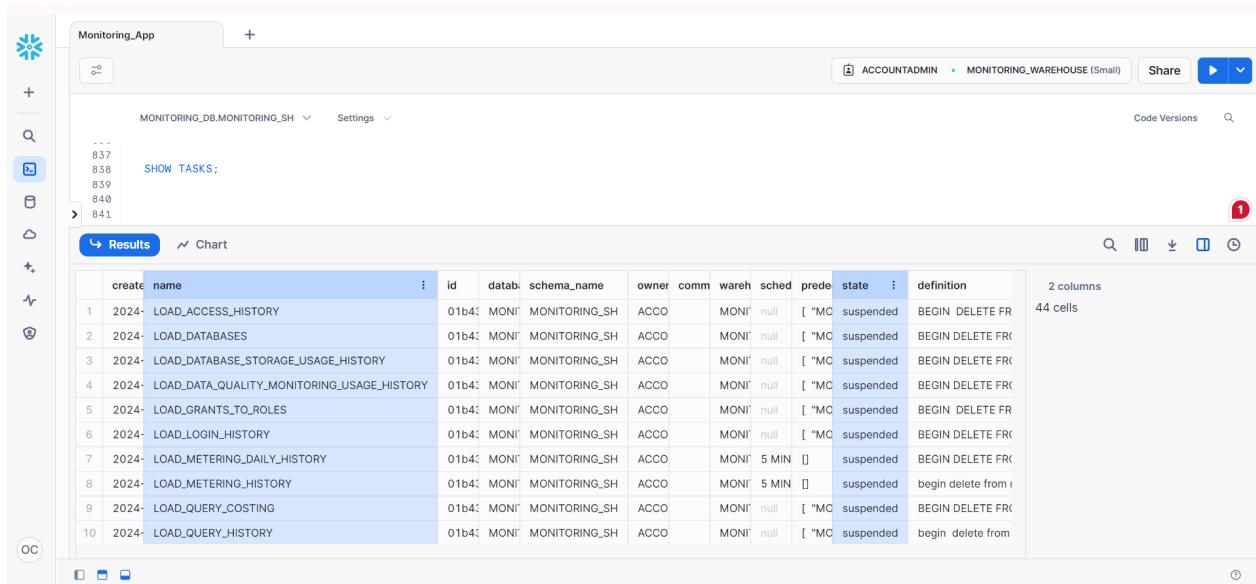
Ce microservice de surveillance des accès est bien plus qu'un simple registre. C'est un levier puissant pour les équipes en charge de la gouvernance et de l'exploitation des systèmes de données, leur permettant de prendre des décisions éclairées et de maintenir la fiabilité et la résilience de l'environnement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

#### 4.2.7 Micro-service « DAG\_Monitoring »

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

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

Ces tâches là sont déjà créées dans notre compte Snowflake de test, la figure 4.28 illustre la liste des tâches dont nous allons montrer dans snowflake, qui apparaîtra en exécutant la commande « Show TASKS » :

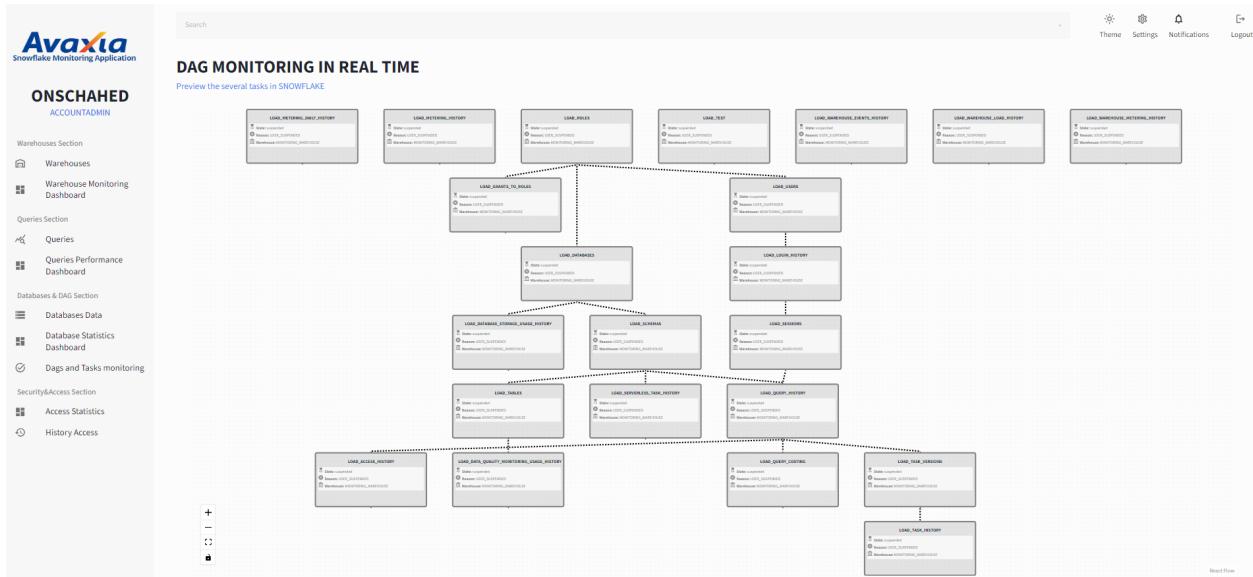


	create	name	id	dbatb	schema_name	owner	comm	wareh	sched	prede	state	definition	2 columns
1	2024-	LOAD_ACCESS_HISTORY	01b4c	MONI	MONITORING_SH	ACCO	MONI	null	[ "MC	suspended	BEGIN DELETE FR	44 cells	
2	2024-	LOAD_DATABASES	01b4c	MONI	MONITORING_SH	ACCO	MONI	null	[ "MC	suspended	BEGIN DELETE FR		
3	2024-	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 4.28 : Résultat du command «Show tasks»

Simultanément, dans notre application, les tâches sont illustrés par la figure 4.31 dans le même état «SUSPENDED» que dans snowflake :

## Chapitre 4. Réalisation



**FIGURE 4.29 :** Etape 0 : état initial du DAG

Une fois nous résumons l'exécution des tâches dans snowflake avec les commandes de la figure 4.30 :

```

MONITORING_DB.MONITORING_SH Settings
Code Versions
792 ----- Activate/SUSPEND all tasks -----
793
794
795 ALTER TASK monitoring_db.monitoring_sh.load_metering_daily_history RESUME;
796 ALTER TASK monitoring_db.monitoring_sh.load_databases RESUME;
797 ALTER TASK monitoring_db.monitoring_sh.load_database_storage_usage_history RESUME;
798 ALTER TASK monitoring_db.monitoring_sh.load_data_quality_monitoring_usage_history RESUME;
799 ALTER TASK monitoring_db.monitoring_sh.load_metering_history resume;
800 ALTER TASK monitoring_db.monitoring_sh.load_query_history RESUME;
801 ALTER TASK monitoring_db.monitoring_sh.load_grants_to_roles RESUME;
802 ALTER TASK monitoring_db.monitoring_sh.load_serverless_task_history RESUME;
803 ALTER TASK monitoring_db.monitoring_sh.load_sessions RESUME;
804 ALTER TASK monitoring_db.monitoring_sh.load_task_history RESUME;
805 ALTER TASK monitoring_db.monitoring_sh.load_task_versions RESUME;
806 ALTER TASK monitoring_db.monitoring_sh.load_users RESUME;
807 ALTER TASK monitoring_db.monitoring_sh.load_warehouse_load_history RESUME;
808 ALTER TASK monitoring_db.monitoring_sh.load_warehouse_metering_history RESUME;
809 ALTER TASK monitoring_db.monitoring_sh.load_access_history RESUME;
810 ALTER TASK monitoring_db.monitoring_sh.load_tables RESUME;
811 ALTER TASK monitoring_db.monitoring_sh.load_schemas RESUME;
812 ALTER TASK monitoring_db.monitoring_sh.load_login_history RESUME;
813 ALTER TASK monitoring_db.monitoring_sh.load_query_costing RESUME;
814 ALTER TASK monitoring_db.monitoring_sh.load_warehouse_events_history RESUME;
815 ALTER TASK monitoring_db.monitoring_sh.load_roles RESUME;
816 ALTER TASK monitoring_db.monitoring_sh.load_test RESUME;]

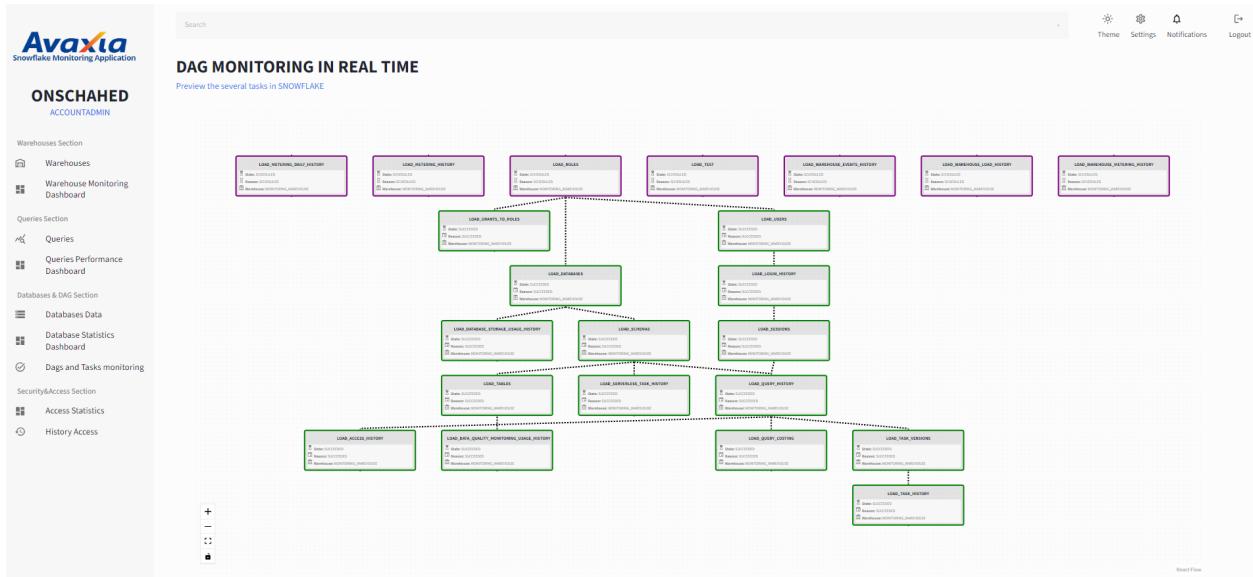
```

**FIGURE 4.30 :** Lancement de l'exécution des tâches

La visualisation du DAG commence à s'animer comme l'indique la figure 4.31 suivante telque :

- Les tâches qui représentent les racines de DAG doivent être toujours programmées 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.

## Chapitre 4. Réalisation



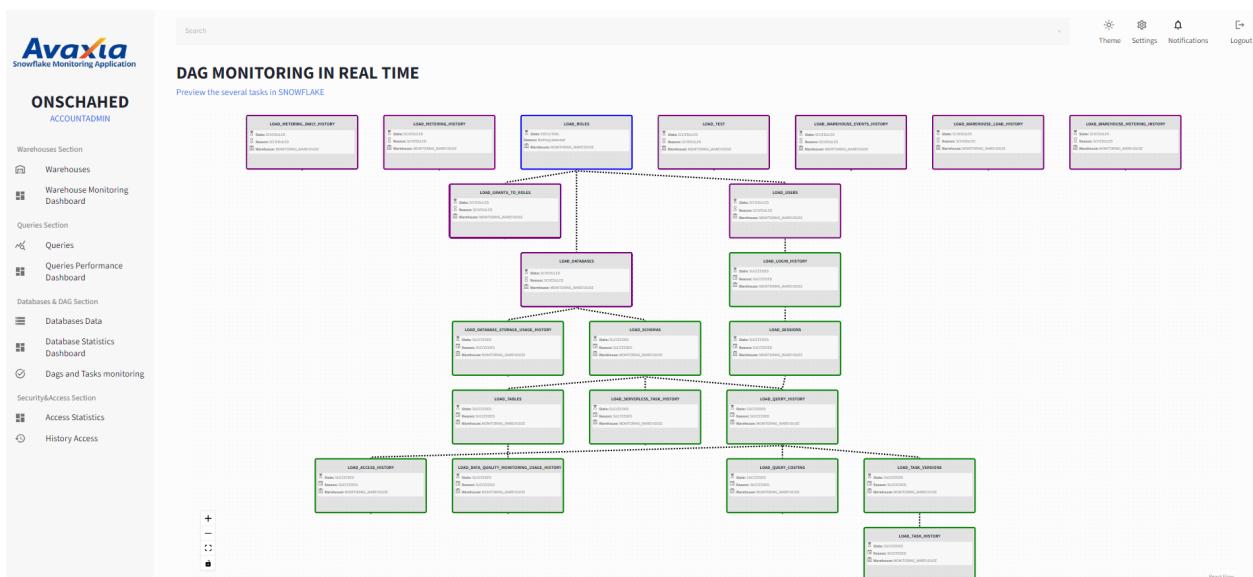
**FIGURE 4.31 : Etape 1 : Lancement des tâches**

Maintenant, nous allons superviser le DAG du tâche «Load roles».

Initialement, cette tâche est **programmée** chaque 5 minutes, elle est colorée en « **violet** » et tous ces sous-tâches sont colorées en « **vert** » pour indiquer que leurs dernier état était un succès.

Après 5 minutes, l'état de la tâche racine se modifie de **programmée** («Scheduled») vers **en exécution** («Executing») et sa couleur devenue « **bleu** » .

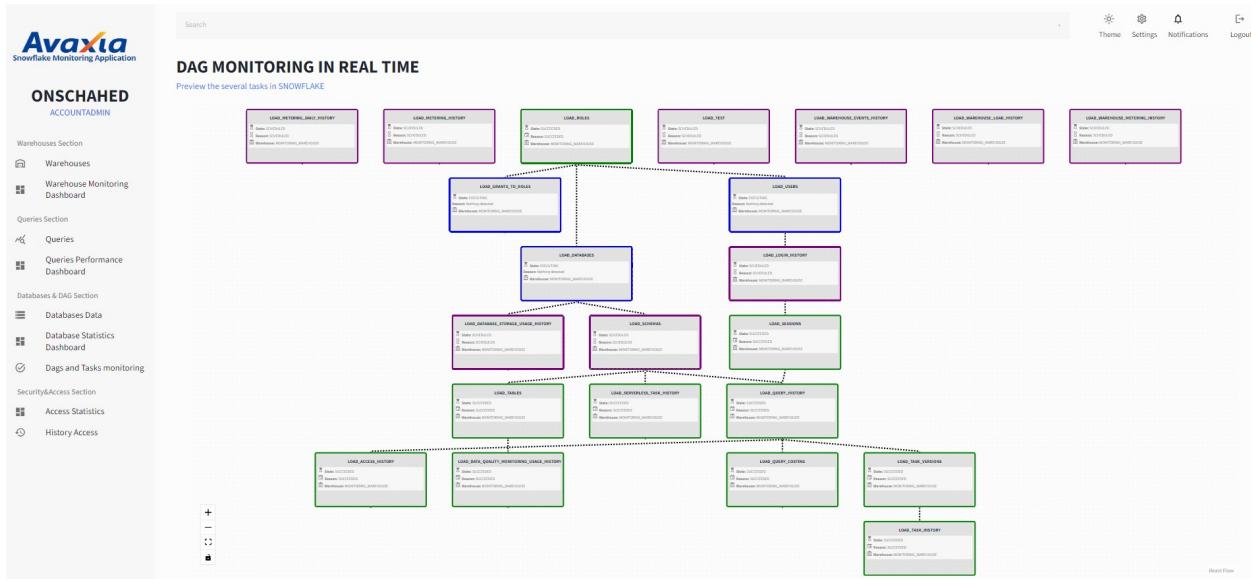
Parallélement, les sous-tâches du premier niveau, «**Load\_grant\_to\_roles**», «**Load users**» et «**Load databases**» deviennent **programmées** mais pour quelques instants car une fois la tâche supérieure complété son exécution ces tâches s'exécutent directement. La figure 4.32 illustre ce qui précéde :



**FIGURE 4.32 : Etape 3 : Execution de la racine**

## Chapitre 4. Réalisation

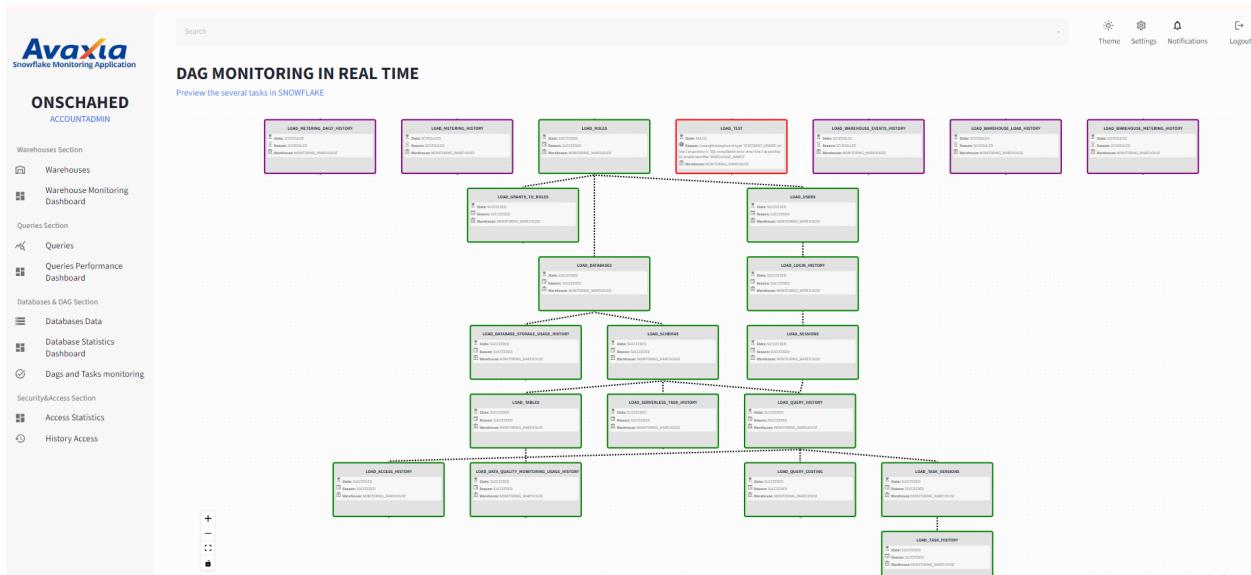
Une fois la tâche « Load roles » est **exécutée avec succès**, ses sous-tâches s'exécutent à leurs tour aussi comme l'indique la figure 4.33



**FIGURE 4.33 :** Etape 4 : Exécution des sous-tâches de niveau I

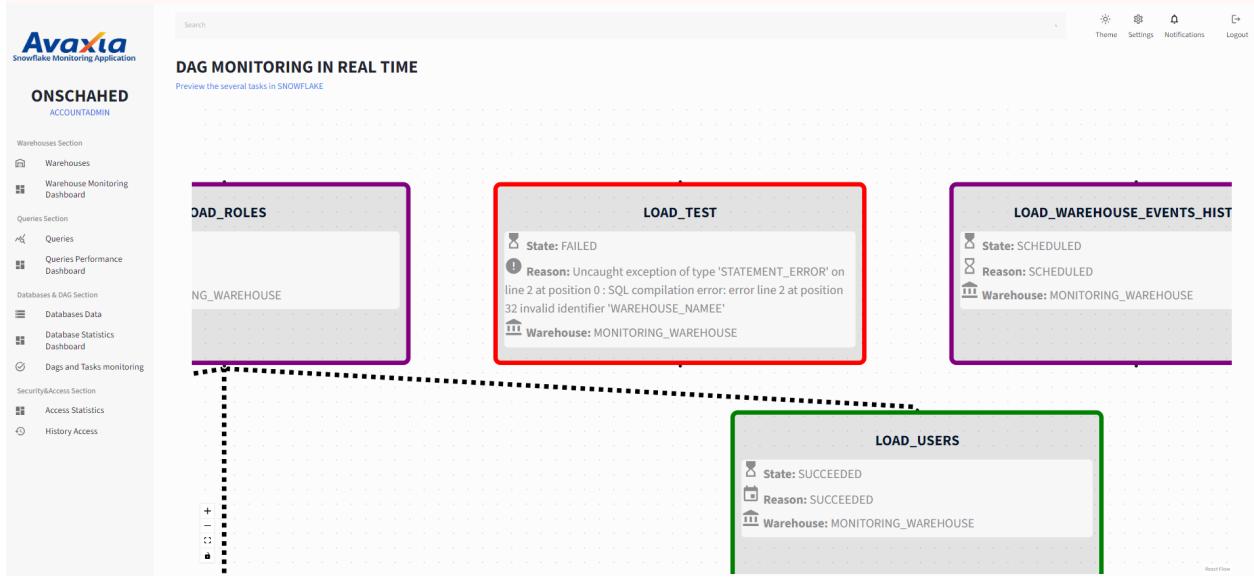
Ce cycle se répète d'un niveau à un autre jusqu'à avoir un succès par tout dans le DAG comme l'indique la figure 4.34 suivante.

À cet instant là, l'état du DAG revient à l'étape 1 du cycle si aucune malveillance est produite, sinon la tâche erronée va être colorée en «rouge» et en affichant le message d'erreur (présent dans la figure 4.35), ce qui le cas de la tâche «load test» dans la même figure.



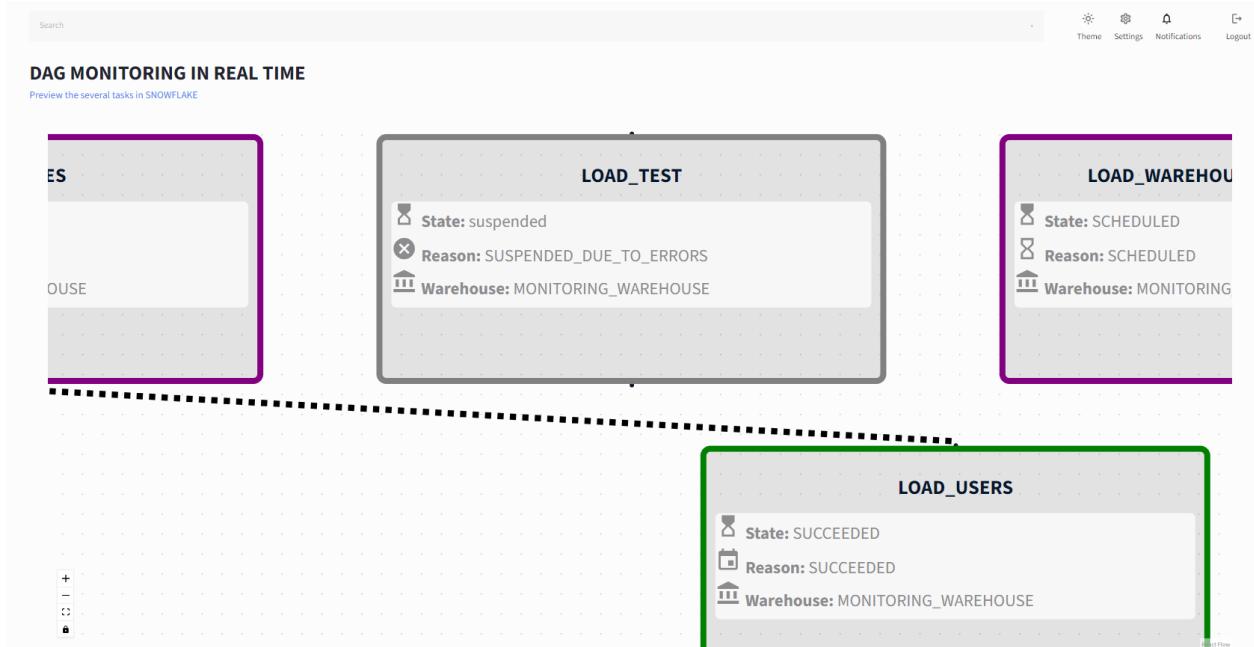
**FIGURE 4.34 :** Etape 5 : Succes dans tous les branches du DAG «Load Roles»

## Chapitre 4. Réalisation



**FIGURE 4.35 :** Message d’erreur de la tâche «Load test»

Il faut noter que, si l’erreur d’une tâche persiste, Snowflake va suspendu cette tâche pour ne pas affectée les tâches en relations avec elle. Bien évidement, si cette tâche erronée est présente dans une DAG, la tâche racine va être auto-suspendu directement ce qui va empêcher l’execution de tout les autres sous-tâches [45]. la figure 4.36 montre l’auto-suspend de la tâche «load test» après exactement 10 tentative d’essai (un paramètre par défaut dans Sonwflake fixé à 10) qui ont donné des erreurs.

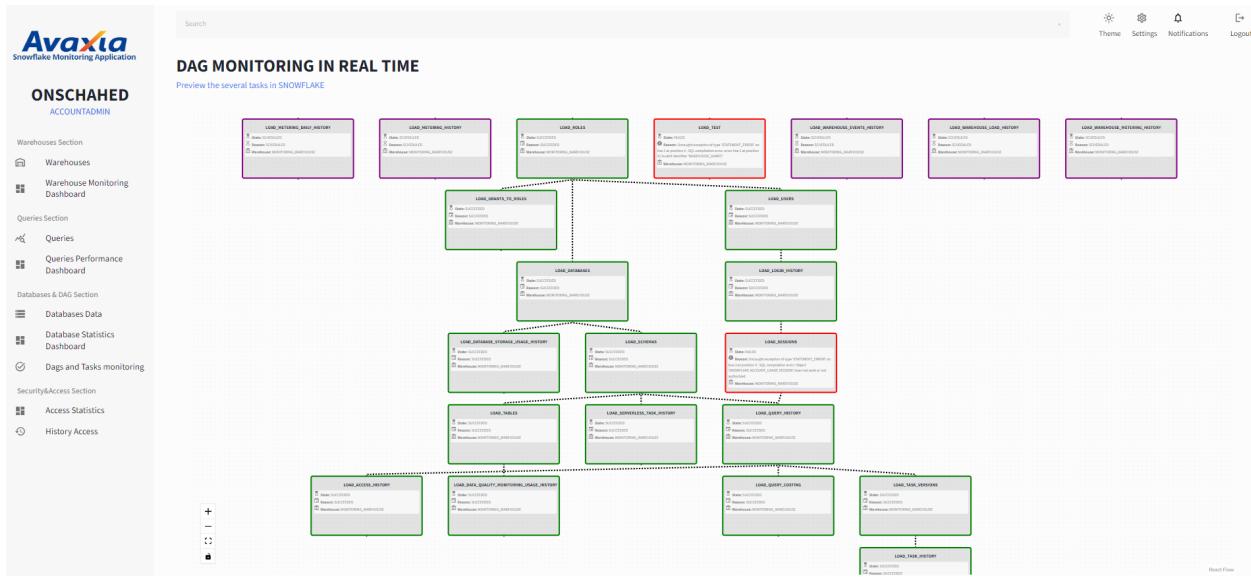


**FIGURE 4.36 :** Auto-suspend de la tâche

Nous allons prendre l’ exemple si la tâche «Load schemas» est erronée : La figure 4.37 illustre l’état du DAG

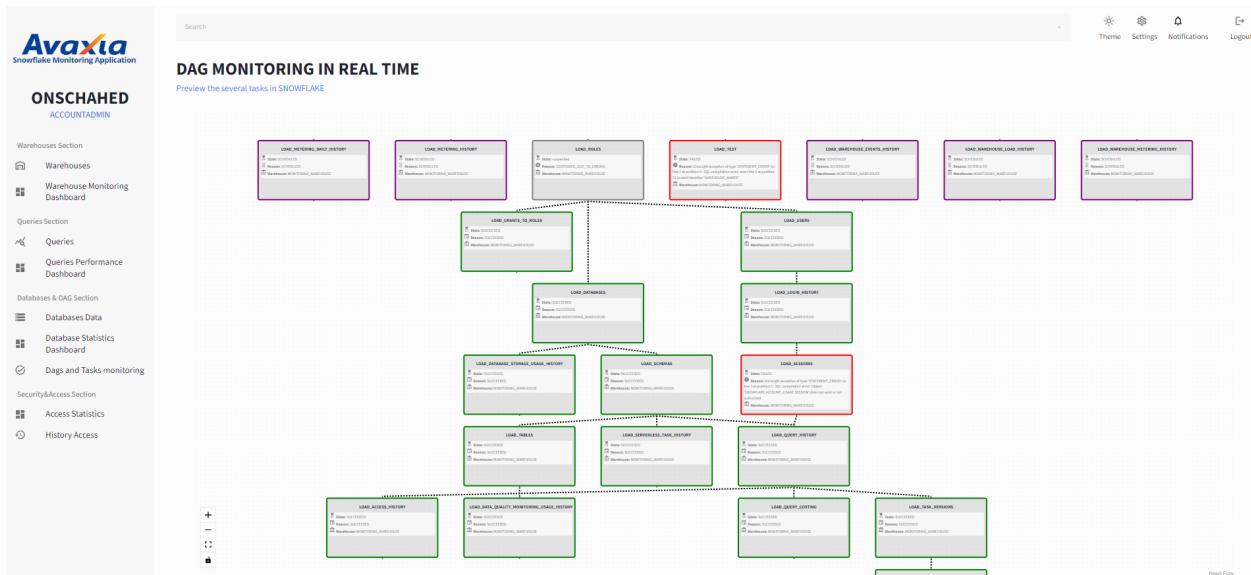
## Chapitre 4. Réalisation

à cet instant.



**FIGURE 4.37 : DAG erronée**

Après 10 tentatives, la racine du DAG est suspendu automatiquement à cause de cette tâche ce qui est illustré par les figures 4.38 et 4.39. De plus, les autres tâches vont conserver leurs derniers états comme l'indique la figure 4.38



**FIGURE 4.38 : La racine de Dag est suspendu automatiquement**

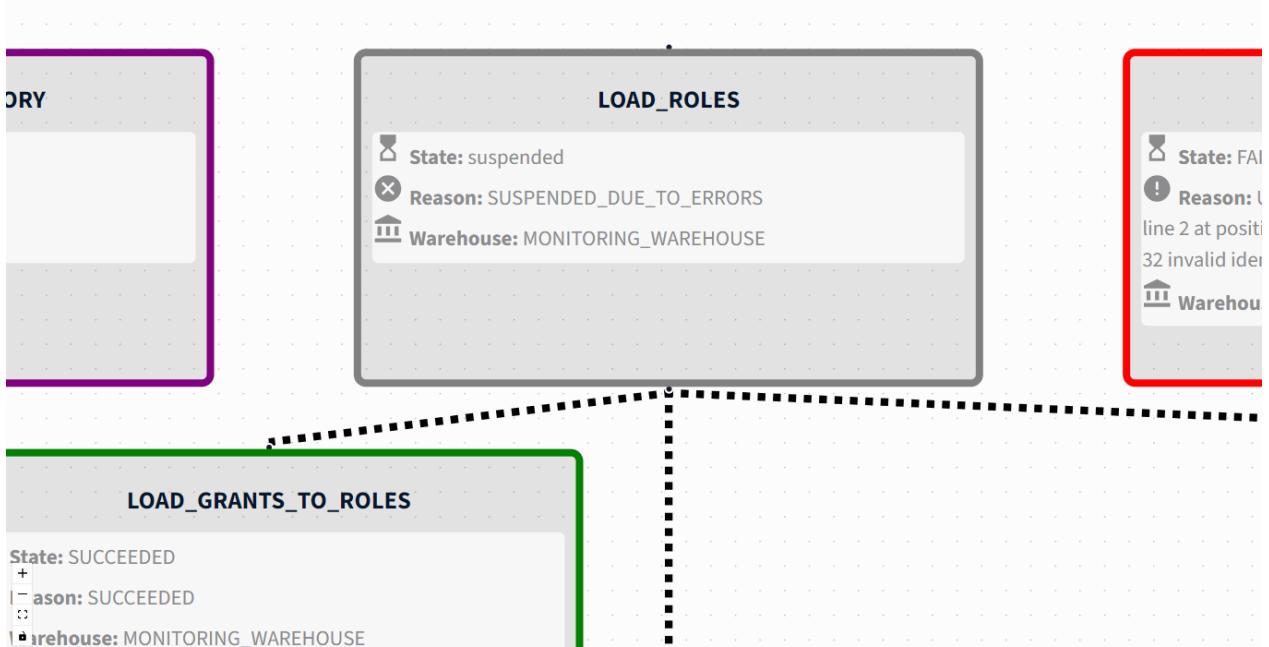


FIGURE 4.39 : Détails de suspend de la racine

En conclusion, ce microservice offre une vue d’ensemble détaillée et interconnectée du suivi des Tâches et des DAGs au sein du compte Snowflake. Cette approche intégrée donne aux administrateurs une compréhension globale de l’environnement, facilitant la détection rapide des problèmes, le diagnostic des causes et la mise en place de mesures correctives pour assurer la fiabilité et l’efficacité des données qui circule dans les entrepôts de données ainsi que l’environnement Snowflake.

#### 4.2.8 Micro-service «Notifications\_Launcher»

Ce micro-service est responsable de l’envoi des notifications aux utilisateurs concernés selon l’événement qui peut apparaître.

- Si les mise à jours sont effectuées :

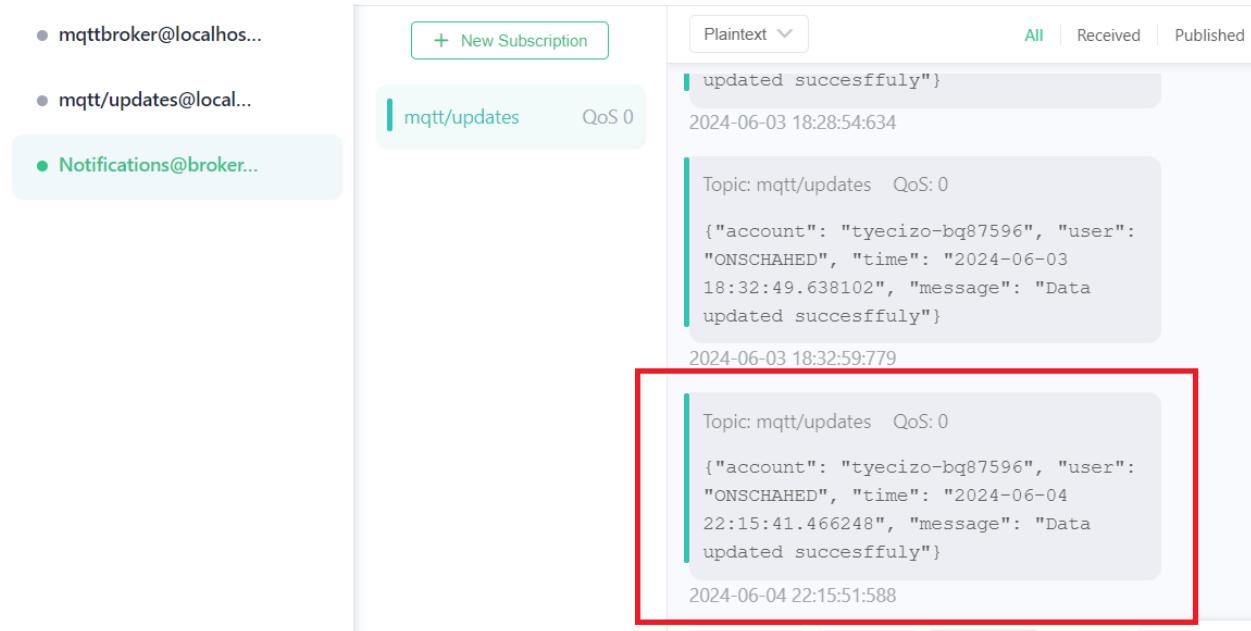
Dans le cas où les mise à jours sont effectuées pour un compte, et comme nous avons déjà préciser dans le microservice d’authentification, une fois le process ETL termine sont exécution il lance une notification à MQTT Broker, la figure 4.40 illustre une autre fois cette partie :

```
>> New data inserted successfully in access_history !
>> New data inserted successfully in OBJECT_DEPENDENCIES !
----- Data updated in all system -----

>> Publishing notification in progress ...
[LOST HEARTBEAT FOR 360 SECONDS, GOING TO CLOSE CONNECTION]
[TRYING WRITE TO CLOSED SOCKET]
Run time of job "run_etl_task (trigger: interval[0:02:00], next run at: 2024-06-04 22:16:00 WAT)" was missed by 0:01:40.555559
>> Notification published Successfully at 2024-06-04 22:15:51.469254
```

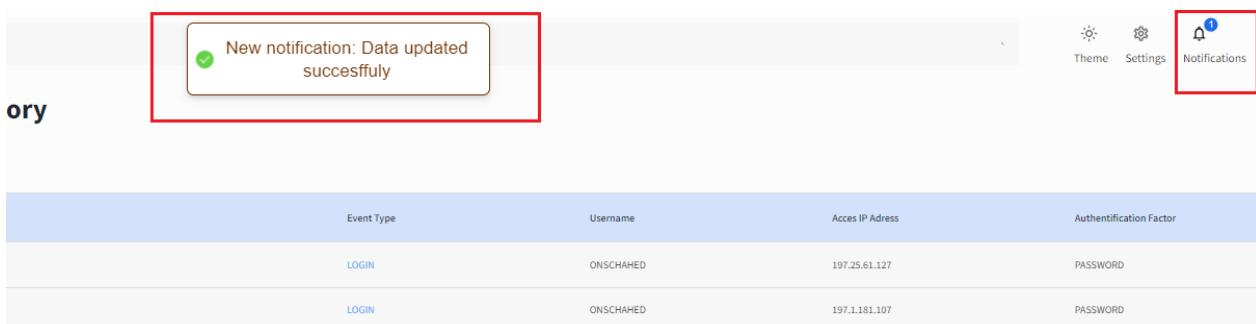
FIGURE 4.40 : Exécution de l’ETL

la publication de cette nouvelle notification dans MQTT est illustrée par la figure 4.41 suivante :



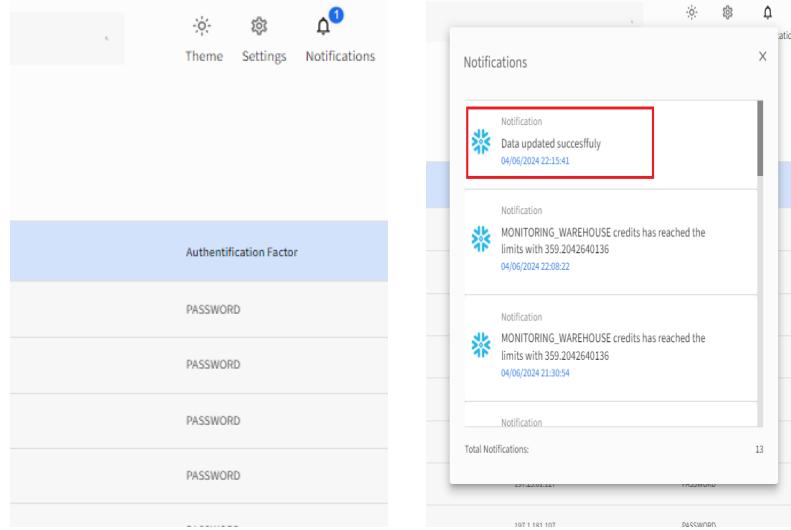
**FIGURE 4.41 :** Publication dans le topic «Mqtt/updates»

Simultanément, dans notre application, une popup s'affiche pour notifier les utilisateurs connectés à ce compte qu'un mise à jour a été effectuée. La figure 4.42 illustre la réception de la nouvelle notification :



**FIGURE 4.42 :** Réception de la nouvelle notification de mise à jour dans «Snowflake Monitoring Application»

une fois nous cliquons sur l'icône de notification, la liste des notifications pour cet utilisateur s'affiche, comme l'indique la figure 4.43 :



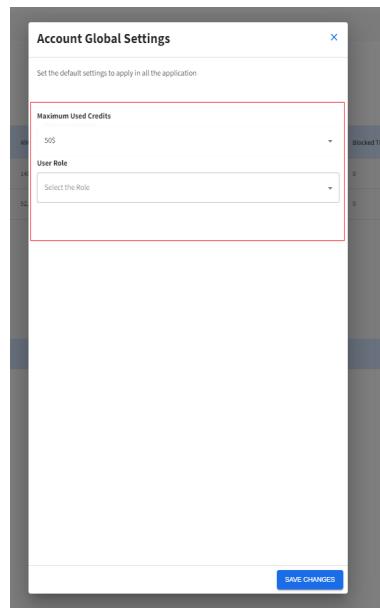
**FIGURE 4.43 : Consultation des notifications**

- **Si les limites de consommation sont atteintes :**

Dans le cas où les limites de consommation sont atteintes pour un des entrepôts de données, une notification est lancée.

Nous prenons l'exemple de la modification de limites à 50\$ dans les paramètres globaux et nous allons surveiller ce qui va se passer dans notre application.

La figure 4.44 illustre la modification des paramètres pour surveiller le lancement des notifications :

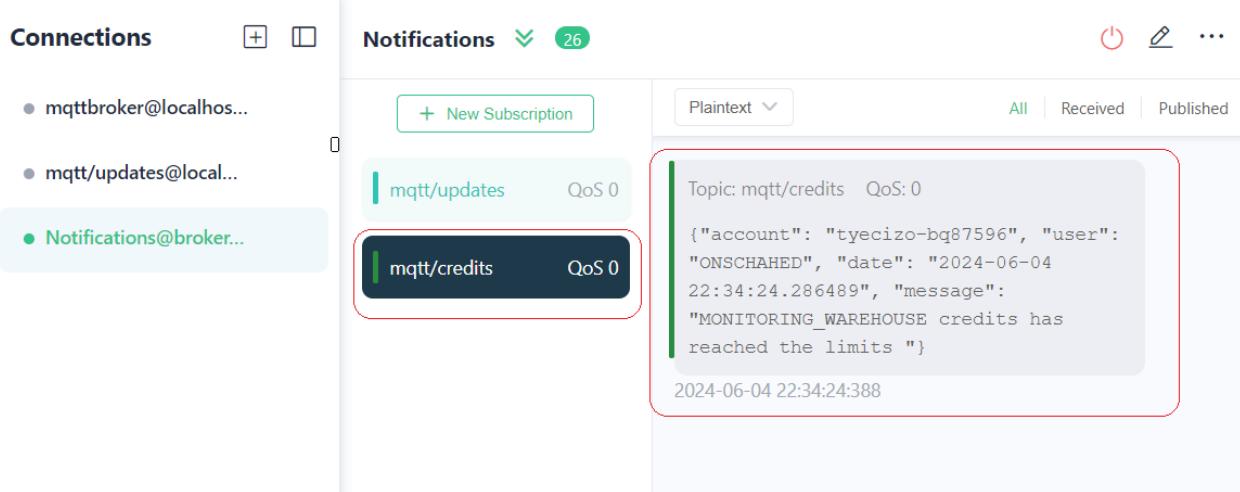


**FIGURE 4.44 : Modification de limites**

Une fois la modification effectuée, nous allons simuler une publication dans le topic «mqtt/credits»

comme l'indique la figure 4.45.

Cette fois ci, la publication dans MQTT est faite par le frontend de ce microservice.



**FIGURE 4.45 :** Publication dans le topic «Mqtt/credits»

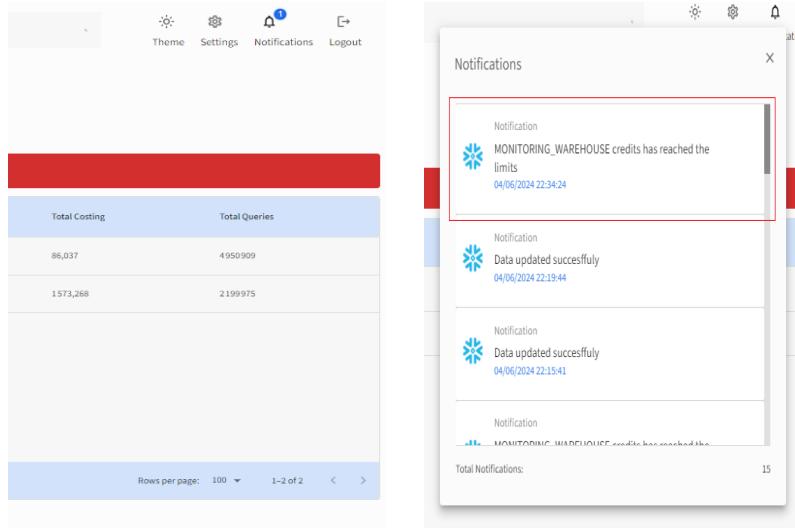
Parallélement, Une notification est reçue dans l'application et une popup s'affiche pour indiquer la réception de cette dernière. La figure 4.46 illustre la réception de la notification :



**FIGURE 4.46 :** Récéption de la nouvelle notification

Une fois nous cliquons sur l'icône de notification, la liste s'affiche avec la nouvelle notification.

La figure 4.47 représente la nouvelle liste des notifications :



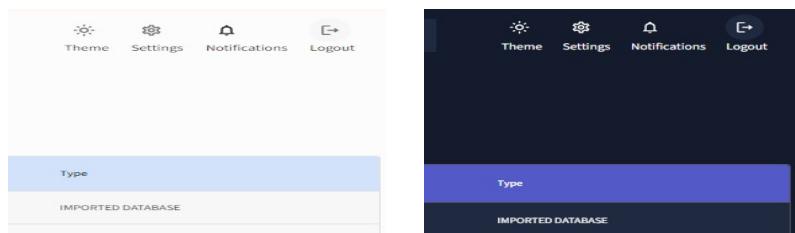
**FIGURE 4.47 :** Consultation des notifications

En conclusion, ce microservice joue un rôle essentiel dans la chaîne de valeur de l'application. En agissant comme une passerelle entre le service ETL et les utilisateurs finaux, il garantit que les informations critiques sont rapidement et efficacement communiquées.

#### 4.2.9 Personnalisation de l'interface utilisateur : thèmes clair et sombre

Dans ce projet, nous avons intégré des fonctionnalités de personnalisation pour améliorer l'expérience utilisateur, notamment le développement de thèmes clair et sombre.

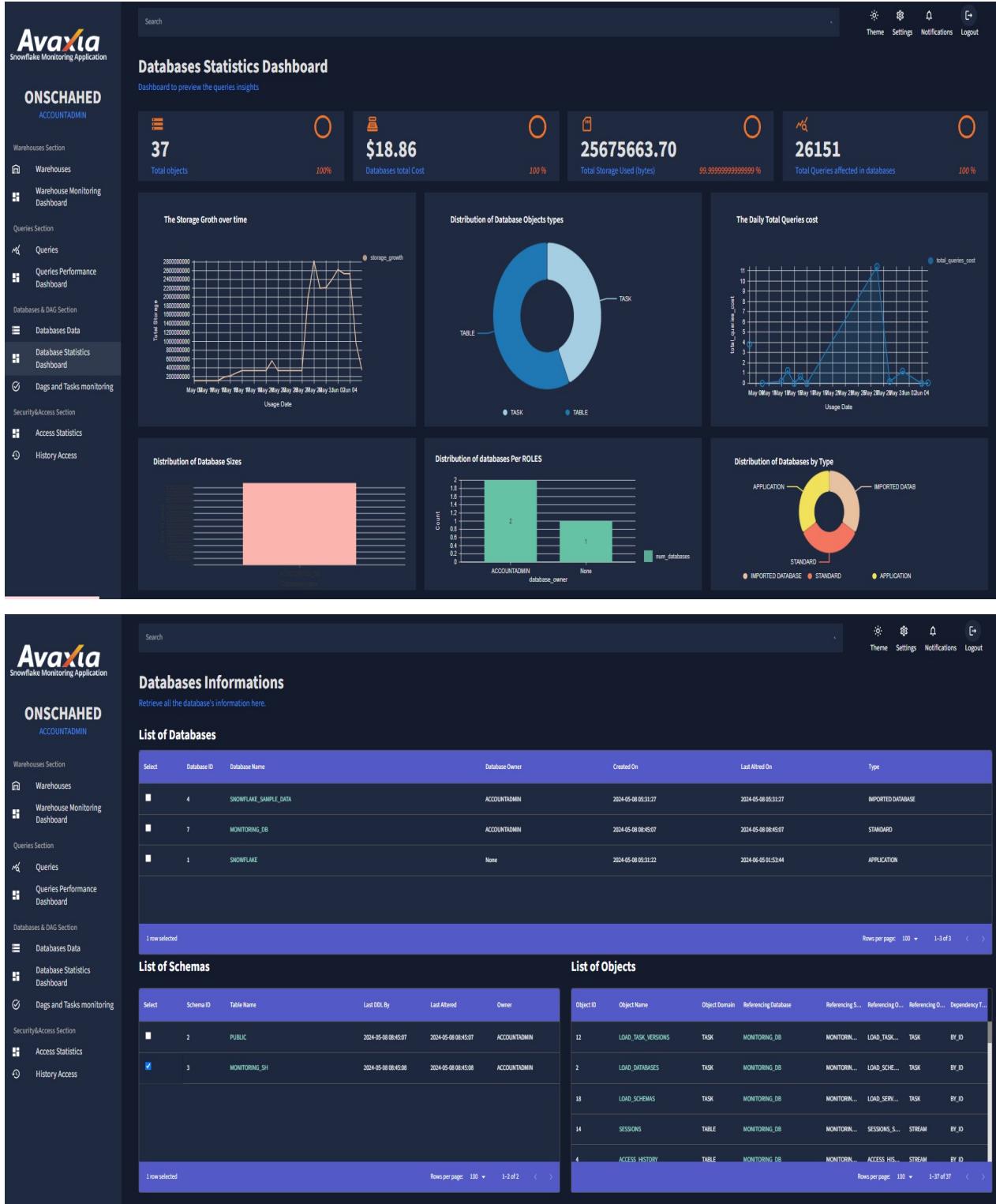
Cette double option permet aux utilisateurs de choisir entre une interface lumineuse et une interface sombre, en fonction de leurs préférences ou des conditions de luminosité ambiante. Le thème clair offre une visualisation nette et claire idéale pour les environnements bien éclairés, tandis que le thème sombre réduit la fatigue oculaire lors d'une utilisation prolongée dans des conditions de faible luminosité. Cette flexibilité assure une meilleure ergonomie et accessibilité de notre application, répondant ainsi aux attentes variées de notre base d'utilisateurs diversifiée. La figure 4.48 illustre l'option «theme» qui nous permet de modifier le thème :



**FIGURE 4.48 :** Modification du thème

## Chapitre 4. Réalisation

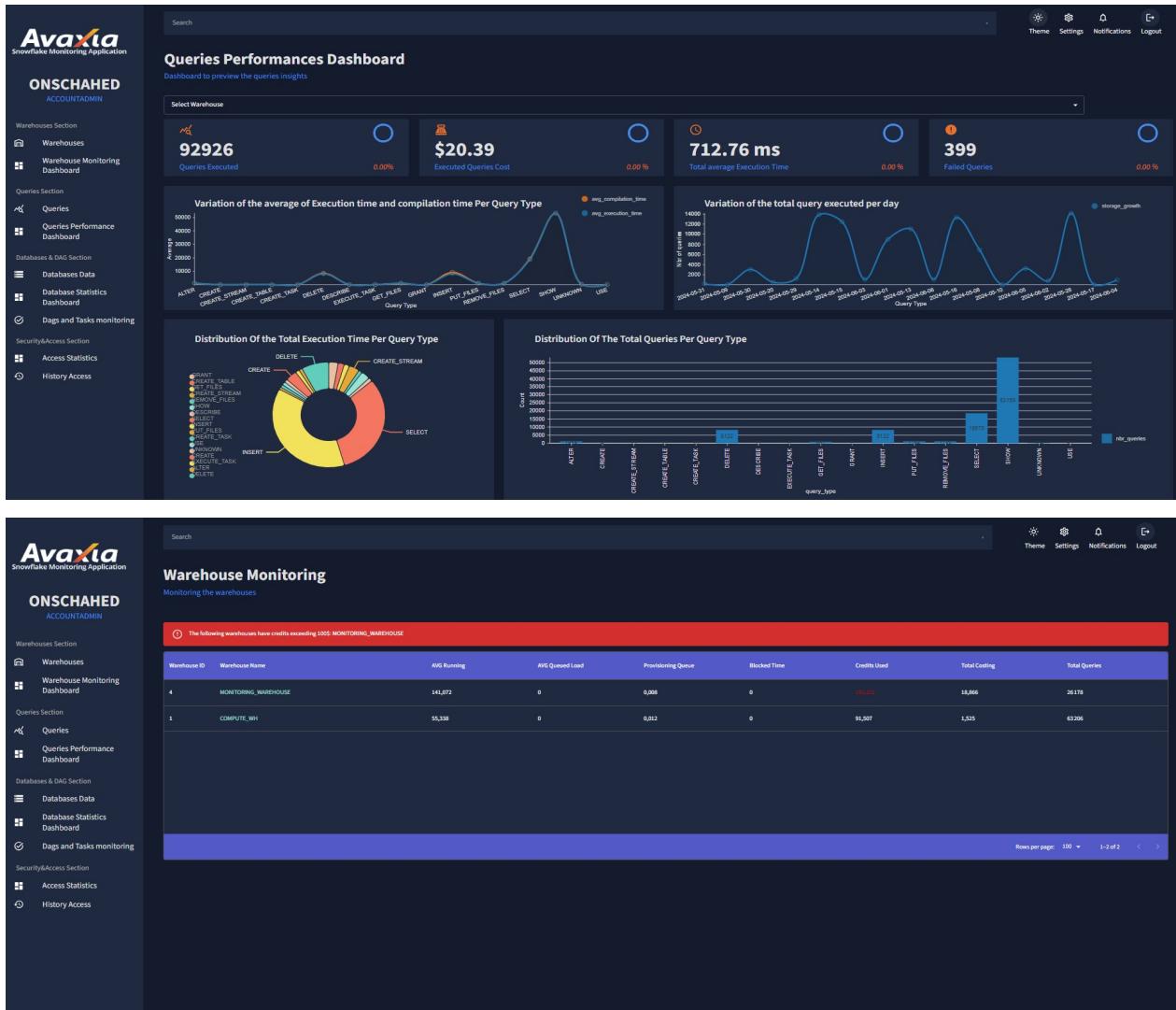
La figure 4.49 suivante illustrent les interfaces de «Databases Dashboard » et «Database Objects List » en mode sombre :



**FIGURE 4.49 :** Interfaces «Databases Dashboard » et «Database Objects List » en thème sombre

## Chapitre 4. Réalisation

La figure 4.50 suivante illustrent les interfaces de «Queries Dashboard » et «Warehouses List » en mode sombre :



**FIGURE 4.50 :** Interfaces en «Queries Dashboard » et «Warehouses List » thème sombre

## Conclusion

Ce chapitre résume le travail accompli pour donner forme à notre projet. Les choix techniques que nous avons adoptés et détaillés nous ont permis de concrétiser notre vision. De plus, nous avons pu présenter la réalisation de ce projet en illustrant chaque partie avec des captures explicatives, montrant ainsi le cheminement de notre démarche.

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

# Conclusion générale

À travers ce projet réalisé au sein d'Avaxia Group, nous avons pu constater la nécessité de disposer de systèmes performants pour gérer, stocker et analyser les données à grande échelle. La plateforme analytique conçue représente une avancée significative dans cette direction, offrant à l'entreprise les outils nécessaires pour extraire des informations pertinentes et exploiter son potentiel de manière optimale.

En retracant les différentes étapes du projet à travers les chapitres dédiés au cadre, à l'analyse des besoins, à l'architecture et à la réalisation, nous avons mis en évidence l'importance de chaque phase dans la réussite de l'ensemble du projet. De la définition des objectifs à la conception détaillée en passant par le choix des technologies et le développement des micro-services, chaque étape a contribué à la création d'une solution cohérente et fonctionnelle.

Ce projet nous a permis de mettre en pratique nos compétences techniques et méthodologiques, tout en nous confrontant à des défis réels du monde professionnel. Travailler au sein d'Avaxia Group nous a également donné l'opportunité de collaborer avec des professionnels expérimentés et d'évoluer dans un environnement stimulant et enrichissant.

Après avoir mené à bien ce projet, plusieurs perspectives d'amélioration et d'extension se dégagent, ouvrant la voie à des nouvelles opportunités et à une optimisation continue de la plateforme analytique conçue. Telque, l'intégration d'un assistant AI pour l'optimisation des requêtes SQL représente une avancée majeure dans l'amélioration de l'expérience utilisateur.

De plus, l'ajout d'un portail utilisateur spécifique aux besoins définis permettra une expérience plus intuitive et personnalisée. Les utilisateurs pourront ainsi accéder facilement aux fonctionnalités pertinentes et adaptées à leurs responsabilités, favorisant ainsi l'adoption et l'utilisation de la plateforme.

En outre, l'intégration de l'authentification avec le Single Sign-On (SSO) renforcera la sécurité et simplifiera la gestion des accès à la plateforme. Cette fonctionnalité permettra aux utilisateurs de se connecter en toute sécurité en utilisant leurs compte Snowflake ou Azure existants directement, améliorant ainsi la sécurité globale de la solution.

Enfin, le déploiement de la solution à grande échelle est essentiel pour répondre aux besoins croissants de l'entreprise. Cette étape comprendra non seulement la mise en production de la plateforme dans un environnement opérationnel, mais aussi son intégration avec d'autres systèmes et outils utilisés par l'entreprise, garantissant ainsi une performance optimale même en cas de forte demande.

En conclusion, ce projet a permis de poser les bases d'une plateforme analytique robuste et évolutive

## Conclusion générale

---

pour Avaxia Group. Les perspectives envisagées offrent des améliorations significatives qui renforceront encore d'avantage la capacité de l'entreprise à gérer et à exploiter ses données de manière optimale. Nous sommes convaincus que ces évolutions contribueront à maintenir Avaxia Group à la pointe de l'innovation technologique et à soutenir sa croissance future.

# Bibliographie

- [1] *Presentation Avaxia*, <https://www.avaxiagroup.com/fr/>, [En ligne ; Accès le 17-fev-2024], 2023.
- [2] *Services Avaxia*, <https://www.avaxiagroup.com/fr/services>, [En ligne ; Accès le 17-fev-2024], 2023.
- [3] *Origine Kanban*, <https://www.glowbl.com/blog/methode-kanban/>, [En ligne ; Accès le 20-fev-2023], 2017.
- [4] M. REHKOPF, *La méthodologie Kanban*, <https://www.atlassian.com/fr/agile/kanban/>, [En ligne ; Accès le 23-fev-2023], 2017.
- [5] P. NAYDENOV, *Les spécificités des tableaux Kanban*, <https://businessmap.io/fr/ressources/debuter-avec/tableau-kanban>, [En ligne ; Accès le 23-fev-2023], 2017.
- [6] M. JUMELLE, *Plateforme Snowflake*, <https://blent.ai/blog/a/snowflake-tout-savoir>, [En ligne ; Accès le 10-Mars-2024], 2024.
- [7] snowflake COMMUNITY, *DAG*, <https://docs.snowflake.com/fr/user-guide/>, [En ligne ; Accès le 10-Mars-2024], 2024.
- [8] OCINEO, *Solution de monitoring*, <https://ocineo.com/quel-est-l-importance-du-monitoring-informatique-pour-les-entreprises>, [En ligne ; Accès le 10-Mars-2024], 2024.
- [9] QUESTIONPRO, *Analyse des données*, <https://www.questionpro.com/blog/fr/question-ce-que-l-analyse-de-donnees/>, [En ligne ; Accès le 10-Mars-2024], 2024.
- [10] LUCIDCHART, *Diagramme de cas d'utilisation*, <https://www.lucidchart.com/pages/fr/diagramme-de-cas-d-utilisation-uml>, [En ligne ; Accès le 09-Avril-2024], 2024.
- [11] NUTCACHE, *backlog produit*, <https://www.nutcache.com/fr/blog/question-ce-quun-backlog-scrum/>, [En ligne ; Accès le 14-Mars-2024], 2019.
- [12] J. PAQUET, *difference backlog Kanban /backlog SCRUM*, <https://blog.myagilepartner.fr/index.php/2019/10/16/backlog-kanban/>, [En ligne ; Accès le 20-fev-2023], 2017.

## Bibliographie

---

- [13] M. TOUHTOUH, *Architecture Logique des systèmes*, <https://www.softfluent.fr/blog/architecture-logicielle-pour-application>, [En ligne ; Accès le 07-Avril-2024], 2023.
- [14] D. ZHART, *Patrons de conception*, <https://refactoring.guru/fr/design-patterns/what-is-pattern>, [En ligne ; Accès le 07-Avril-2024], 2023.
- [15] G. KHERIJI, *CQRS*, <https://www.invivoo.com/le-pattern-cqrs/>, [En ligne ; Accès le 07-Avril-2024], 2023.
- [16] A. SCHEVTS, *Patron façade*, <https://refactoring.guru/fr/design-patterns/façade>, [En ligne ; Accès le 07-Avril-2024], 2023.
- [17] A. SCHEVTS, *Patron observer*, <https://refactoring.guru/fr/design-patterns/observer>, [En ligne ; Accès le 07-Avril-2024], 2023.
- [18] *Single Responsability Principal*, <https://www.ionos.fr/digitalguide/sites-internet/developpement-web/quest-ce-que-le-factory-pattern/>, [En ligne ; Accès le 07-Avril-2024], 2022.
- [19] GEEKSFORGEEKS, *Data Access Object*, <https://www.geeksforgeeks.org/data-access-object-pattern/>, [En ligne ; Accès le 07-Avril-2024], 2024.
- [20] REDHAT, *architecture physique*, <https://www.redhat.com/fr/topics/cloud-native-apps/what-is-an-application-architecture>, [En ligne ; Accès le 08-Avril-2024], 2023.
- [21] LUCIDCHART, *Diagramme de package*, <https://www.lucidchart.com/pages/fr/diagramme-package-uml>, [En ligne ; Accès le 08-Avril-2024], 2024.
- [22] LUCIDCHART, *Diagramme de package*, <https://www.lucidchart.com/pages/fr/diagramme-de-classes-uml>, [En ligne ; Accès le 08-Avril-2024], 2024.
- [23] 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 09-Avril-2024], 2021.
- [24] IBM, *Diagramme d'activité*, <https://www.ibm.com/docs/fr/dmrt/9.5?topic=diagrams-activity>, [En ligne ; Accès le 08-Avril-2024], 2024.

## Bibliographie

---

- [25] LUCIDCHART, *Diagramme de séquence*, <https://www.lucidchart.com/pages/fr/diagramme-de-sequence-uml>, [En ligne ; Accès le 08-Avril-2024], 2024.
- [26] F. COMMUNITY, *Fastapi*, <https://fastapi.tiangolo.com/fr/>, [En ligne ; Accès le 28-Mai-2024], 2024.
- [27] R. COMMUNITY, *React*, <https://fr.legacy.reactjs.org/>, [En ligne ; Accès le 28-Mai-2024], 2024.
- [28] P. COMMUNITY, *PostgreSQL*, <https://www.postgresql.org/>, [En ligne ; Accès le 28-Mai-2024], 2024.
- [29] M. COMMUNITY, *MQTT*, <https://mqtt.org/>, [En ligne ; Accès le 28-Mai-2024], 2024.
- [30] T. TARGET, *REST*, <https://www.lemagit.fr/definition/REST>, [En ligne ; Accès le 28-Mai-2024], 2024.
- [31] D. VARRAZZO, *Python Snowflake Connector*, <https://docs.snowflake.com/en/developer-guide/python-connector/python-connector>, [En ligne ; Accès le 28-Mai-2024], 2024.
- [32] D. VARRAZZO, *PSYCOPG*, <https://pypi.org/project/psycopg2/>, [En ligne ; Accès le 28-Mai-2024], 2024.
- [33] *pandas*, <https://pandas.pydata.org/>, [En ligne ; Accès le 28-Mai-2024], 2024.
- [34] D. VARRAZZO, *GMQTT*, <https://github.com/wialon/gmqtt>, [En ligne ; Accès le 28-Mai-2024], 2024.
- [35] *OS Module*, <https://docs.python.org/3/library/os.html>, [En ligne ; Accès le 28-Mai-2024], 2024.
- [36] DIGONIO, *Scheduler*, <https://pypi.org/project/scheduler/>, [En ligne ; Accès le 28-Mai-2024], 2024.
- [37] *requests*, <https://pypi.org/project/requests/>, [En ligne ; Accès le 28-Mai-2024], 2024.
- [38] R. BENBRAHIM, *Postman*, <https://welovedevs.com/fr/articles/postman>, [En ligne ; Accès le 28-Mai-2024], 2021.
- [39] *MQTTX*, <https://mqttx.app/docs>, [En ligne ; Accès le 28-Mai-2024], 2024.
- [40] *Git*, <https://www.atlassian.com/fr/git/tutorials/what-is-git>, [En ligne ; Accès le 28-Mai-2024], 2024.

## Bibliographie

---

- [41] R. KASSEL, *Github*, <https://datascientest.com/github-tout-savoir>, [En ligne ; Accès le 28-Mai-2024], 2021.
- [42] *Swagger*, <https://appmaster.io/university/fr/tutorials/endpoints/quest-ce-que-swagger-et-comment-lutiliser>, [En ligne ; Accès le 28-Mai-2024], 2024.
- [43] *Latex*, <https://dms.umontreal.ca/wiki/index.php/LaTeX>, [En ligne ; Accès le 28-Mai-2024], 2024.
- [44] *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 28-Mai-2024], 2024.
- [45] *Auto suspend des tâches Snowflake*, <https://community.snowflake.com/s/article/Why-are-tasks-suspended-automatically>, [En ligne ; Accès le 31-Mai-2024], 2024.

# ملخص

هذا العمل هو جزء من مشروع ختم الدروس الهندسية في المعهد العالي للإعلامية للسنة الدراسية ٢٠٢٣-٢٠٢٤. لقد أجرينا تدريباً مع شركة Avaxia Group في نهاية الدراسة بهدف تصميم وتنفيذ منصة تحليلية لتصور البيانات الوصفية لم Snowflake. يستخدم هذا الحل بنية الخدمات المصغرة لتوفير المرونة والنمطية وقابلية التوسيع، دمج خدمات التحليل والمراقبة في الوقت الفعلي لتحسين الأداء وإدارة البيانات.

**كلمات مفاتيح :** تحليل البيانات, Kanban, ReactJS, PostgreSQL, FastAPI, الخدمات المصغرة

## Résumé

Ce travail s'inscrit dans le cadre de l'accomplissement de notre Projet de Fin d'Études d'ingénieur à l'Institut Supérieur d'Informatique pour l'année universitaire 2023-2024. Nous avons effectué ce stage au sein du Avaxia Group, qui ayant comme objectif de concevoir et implémenter une plateforme analytique pour la visualisation des métadonnées de Snowflake. Cette solution utilise une architecture microservices pour offrir une flexibilité, modularité et évolutivité, intégrant des services d'analyse et de monitoring en temps réel pour optimiser les performances et la gestion des données.

**Mots clés :** Analyse de données, FastAPI, PostgreSQL, Snowflake, ReactJS, Kanban, Microservices.

## Abstract

This work is part of the completion of our Final Engineering Project at the higher institute of computer science for the academic year 2023-2024. We carried out an end-of-study internship with the Avaxia Group with the aim of designing and implementing an analytical platform for the visualisation of Snowflake metadata. This solution uses a microservices architecture to offer flexibility, modularity and scalability, integrating real-time analysis and monitoring services to optimise performance and data management.

**Keywords :** Data analysis, FastAPI, PostgreSQL, Snowflake, ReactJS, Kanban, Microservices.