

RÉPUBLIQUE TUNISIENNE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

Faculté des Sciences Economiques et de Gestion de Nabeul



Rapport de Projet

Master Professionnel : Ingénierie des Systèmes d'informations
et des Connaissance (ISIC)

Plateforme d'analyse des accidents routiers aux États-Unis

US Accidents Analytics Platform

Elaboré par :

Fida Sassi
Sarra Sghaier

Année Universitaire : 2025 - 2026

Table des matières

Liste des figures	v
Liste des tableaux	vi
Introduction générale	1
1 Présentation générale du projet	2
1.1 Introduction du projet	2
1.2 Présentation du dataset	2
1.3 Problématique	3
1.4 Objectifs du projet	3
1.5 Questions métier	3
1.6 Solution proposée	4
1.7 Choix des années d'étude	4
1.8 Conclusion	5
2 Analyse et conception du système	6
2.1 Introduction	6
2.2 Analyse des besoins	6
2.2.1 Besoins fonctionnels	6
2.2.2 Besoins non fonctionnels	7
2.3 Identification des acteurs	7
2.4 Diagramme de cas d'utilisation	8

2.5	Architecture générale	8
2.5.1	Introduction	8
2.5.2	Présentation de l'architecture	9
2.6	Conception du Datamart	10
2.6.1	Présentation du schéma étoile	10
2.6.2	Description des tables du schéma étoile	11
2.7	Pipeline ETL	13
2.7.1	Extraction et analyse des données	13
2.7.2	Chargement des données brutes (Staging)	13
2.7.3	Transformation et nettoyage des données	14
2.7.4	Construction du datamart	14
2.7.5	Gestion des jobs ETL	14
2.7.6	Sécurité et contrôle d'accès	15
2.8	Diagrammes de séquence	15
2.8.1	Inscription et vérification email	15
2.8.2	Connexion utilisateur	17
2.8.3	Diagramme de séquence de Pipeline ETL	17
2.9	Conclusion	19
3	Réalisation et Implémentation du Système	20
3.1	Introduction	20
3.2	Réalisation et Implémentation du Système	20
3.2.1	Environnement de Développement	20
3.3	Implémentation du Backend	21
3.3.1	API REST	21
3.3.2	Détails techniques des APIs Flask	22
3.3.3	Authentification et Sécurité	24
3.3.4	Gestion des erreurs et performances	24
3.4	Implémentation du Frontend	24
3.5	Implémentation du Module de Machine Learning	25

3.5.1	Objectif du Modèle	25
3.5.2	Préparation des données	25
3.5.3	Choix du modèle	26
3.5.4	Entraînement du modèle	26
3.5.5	Évaluation des performances	28
3.5.6	Sauvegarde et intégration du modèle	28
3.5.7	Intégration dans la plateforme	28
3.6	Conclusion	28
4	Tests, Résultats et Validation du Système	30
4.1	Introduction	30
4.2	Interfaces dans la plateforme	30
4.2.1	Page création de compte	30
4.2.2	Réception de code après la creation de compte	31
4.2.3	Page connexion	31
4.2.4	Mot de passe est oublié	31
4.2.5	Réinitialisation de mot de passe	32
4.2.6	Page Dashboard	32
4.2.7	Page Pipeline ETL	34
4.2.8	Page prédiction	35
4.3	Résultats de module de Machine Learning	35
4.3.1	Synthèse des résultats	38
4.3.2	Limites du projet	38
4.4	Perspectives d'évolution	38
4.5	Conclusion	38

Table des figures

2.1	Les principales fonctionnalités offertes par la plateforme	8
2.2	Architecture générale de la plateforme	9
2.3	Diagramme de classes du système	11
2.4	Structure de la table dim_location	12
2.5	Structure de la table dim_time	12
2.6	Structure de la table dim_weather	12
2.7	Structure de la table etl_jobs	13
2.8	Structure de la table users	13
2.9	Diagramme de séquence d'inscription et vérification email	16
2.10	Diagramme de séquence de connexion utilisateur	17
2.11	Diagramme de séquence du pipeline ETL	18
3.1	Test de l'endpoint /auth/login via Postman	23
3.2	Test de l'endpoint /api/stats/overview via Postman	23
4.1	Page creation compte	30
4.2	Réception code après création de compte	31
4.3	Page connexion	31
4.4	Mot de passe oublié	32
4.5	Réinitialisation de mot de passe	32
4.6	Page Dashboard partie -1-	33
4.7	Page Dashboard partie -2-	33

4.8	Page Dashboard partie -3-	34
4.9	Page Pipeline ETL partie -1-	34
4.10	Page Pipeline ETL partie -2-	35
4.11	Page prediction	35
4.12	Distribution des classes	36

Liste des tableaux

2.1	Identification des acteurs du système	7
2.2	Architecture en couches de la plateforme	9
2.3	Description des tables du schéma étoile	12
3.1	Technologies utilisées dans le projet	20
3.2	Hyperparamètres du modèle Random Forest	26
4.1	Distribution des classes de gravité des accidents	36
4.2	Variables les plus importantes du modèle Random Forest	37
4.3	Performance du modèle Random Forest	37

Introduction générale

Dans un monde de plus en plus interconnecté, les données occupent une place essentielle dans les processus décisionnels de nombreux secteurs tels que le transport, la santé et l'industrie. Avec l'émergence du Big Data, les organisations disposent aujourd'hui d'un volume considérable de données générées quotidiennement, offrant de nouvelles opportunités d'analyse et d'exploitation des informations.

Dans ce contexte, les données relatives aux accidents de la route représentent une source d'information importante pour l'amélioration de la sécurité routière. Cependant, ces données se caractérisent par leur volume élevé, leur diversité ainsi que leur complexité, ce qui rend leur traitement et leur interprétation difficiles sans l'utilisation d'outils adaptés.

Le présent projet s'inscrit dans cette perspective et a pour objectif la conception et le développement d'une plateforme web intelligente permettant d'analyser les données d'accidents routiers, de visualiser les tendances et de prédire la gravité des accidents selon différents facteurs.

La solution proposée repose sur l'utilisation de technologies modernes telles que Flask pour le développement du backend, React pour le frontend, ainsi que des techniques de traitement de données et d'apprentissage automatique.

Ce rapport est organisé comme suit :

- Le premier chapitre présente le contexte général du projet, le dataset utilisé, la problématique ainsi que les objectifs visés.
- Le deuxième chapitre est consacré à l'analyse des besoins et à la conception du système.
- Le troisième chapitre décrit la réalisation technique de la plateforme.
- Le quatrième chapitre présente les tests effectués ainsi que les résultats obtenus.
- Enfin, une conclusion générale résume les principales contributions du projet et propose quelques perspectives d'amélioration

Présentation générale du projet

1.1 Introduction du projet

Ce chapitre vise à présenter le contexte général du projet ainsi que le jeu de données utilisé dans le cadre de cette étude.

Dans ce travail, nous nous intéressons à l'analyse des accidents routiers aux États-Unis, un domaine crucial pour la sécurité publique en raison de leur fréquence et de leur gravité.

L'étude est basée sur le dataset US Accidents (2016–2023) disponible sur Kaggle, qui regroupe un grand volume de données d'accidents routiers. Ce dataset contient diverses informations telles que la localisation, les conditions météorologiques, les horaires ainsi que d'autres facteurs environnementaux pouvant influencer la gravité des accidents.

Ainsi, ce chapitre permet de comprendre le contexte du projet et la nature des données utilisées avant de passer aux étapes d'analyse et de traitement.

1.2 Présentation du dataset

Dans cette partie, nous introduisons le jeu de données utilisé dans le cadre de ce projet ainsi que ses principales caractéristiques. Il s'agit du **dataset US Accidents**, disponible sur la plateforme Kaggle, qui couvre la période **de 2016 à 2023**. Ce jeu de données contient un volume important d'enregistrements d'accidents routiers aux **États-Unis**, incluant des informations variées telles que **les conditions météorologiques, la localisation géographique, les données temporelles** ainsi que d'autres facteurs environnementaux liés aux accidents.

1.3 Problématique

Il existe une abondance de données relatives aux accidents de la route, mais elles ne sont pas directement utilisables dans leur format brut. Effectivement, la grande quantité et la variabilité de ces données rendent leur analyse complexe et compliquent leur interprétation.

Par ailleurs, ces données ne fournissent pas une perspective nette sur les tendances, les zones à risque ou les éléments affectant la sévérité des accidents sans un traitement et une étude adéquats.

On peut donc poser la question de ce projet de la manière suivante :

Comment exploiter efficacement des données massives et hétérogènes relatives aux accidents de la route afin de produire des informations pertinentes facilitant l'analyse et la prise de décision ?

1.4 Objectifs du projet

Objectif général

L'objectif principal de ce projet est de concevoir une plateforme web intelligente dédiée à l'analyse des données d'accidents routiers afin de faciliter l'exploration des données et l'aide à la décision.

Objectifs spécifiques :

- Collecter et traiter les données relatives aux accidents routiers.
- Analyser les facteurs influençant la gravité des accidents.
- Identifier les tendances et les zones à risque.
- Visualiser les indicateurs clés à travers des tableaux de bord interactifs.
- Mettre en œuvre un modèle prédictif basé sur le machine learning.

1.5 Questions métier

Lors de l'examen des données relatives aux accidents de la route, plusieurs interrogations professionnelles ont été établies pour guider l'analyse et mieux appréhender les éléments qui affectent la sévérité des accidents :

- À quels moments de la journée les accidents atteignent-ils leur gravité maximale ?
- Dans quelles zones des États-Unis le taux d'accidents est-il le plus haut ?
- Les accidents sont-ils plus courants et plus sérieux pendant les fins de semaine ou en semaine ?

- Quel est l'impact des conditions de température sur la sévérité des accidents ?
- Les accidents de forte sévérité prennent-ils généralement plus de temps à être résolus que les accidents moins graves ?

1.6 Solution proposée

Pour faire face à la question soulevée, il est suggéré d'utiliser une plateforme web intelligente pour regrouper et gérer les données relatives aux accidents de la route.

Cette solution facilite la conversion de données brutes en informations utiles via un système intégré qui combine des caractéristiques d'analyse, de visualisation et d'assistance à la prise de décision.

Elle présente donc une perspective complète des accidents de la route et simplifie leur compréhension grâce à une interface interactive.

1.7 Choix des années d'étude

Le dataset utilisé dans ce projet couvre plusieurs années d'accidents routiers aux États-Unis. Afin de garantir la qualité des données et la stabilité des analyses, une sélection des années les plus pertinentes a été effectuée.

Le choix des années retenues repose sur plusieurs critères :

- d'obtenir des données plus homogènes pour l'apprentissage du modèle ;
- d'éviter certaines périodes atypiques pouvant fausser les analyses ;
- d'améliorer les performances du traitement ETL et du modèle de machine learning ;
- de réduire le temps d'entraînement et la consommation mémoire.

Certaines années présentent en effet des comportements particuliers susceptibles d'influencer fortement les données d'accidents. Par exemple, durant la période de la pandémie de COVID-19, les conditions de circulation ont été fortement modifiées à cause des confinements, des restrictions de déplacement et des changements dans le trafic routier.

Ces événements exceptionnels peuvent entraîner des variations importantes dans le nombre d'accidents, les conditions de circulation ainsi que les comportements observés dans les données.

Ainsi, le choix des années d'étude permet de travailler sur des données plus représentatives des conditions normales de circulation, tout en garantissant une meilleure stabilité des analyses statistiques et des prédictions générées par le modèle.

1.8 Conclusion

Ce chapitre a fourni une présentation du contexte global du projet ainsi que du jeu de données exploité dans le cadre de cette recherche. Il a aussi souligné le problème concernant l'utilisation des données relatives aux accidents de la route, ainsi que les buts que vise ce projet.

Pour finir, nous avons exposé la solution complète suggérée ainsi que les questions professionnelles clés qui orienteront l'analyse des données.

Le prochain chapitre se focalisera sur l'étude et la création du système établi.

Analyse et conception du système

2.1 Introduction

Ce chapitre présente l'analyse des besoins ainsi que la conception générale du système proposé. Il décrit les fonctionnalités attendues, les contraintes non fonctionnelles ainsi que les différents modèles UML et l'architecture technique adoptée pour la réalisation de la plateforme.

2.2 Analyse des besoins

2.2.1 Besoins fonctionnels

Le système doit permettre :

Gestion des données

- Importation des fichiers CSV.
- Sélection de l'année d'importation.
- Ajout progressif des données sans suppression des données existantes.

Pipeline ETL

- Chargement des données brutes.
- Nettoyage des données.
- Construction du datamart analytique.
- Suivi et traçabilité des traitements.

Tableau de bord analytique

- Visualisation des KPIs.
- Filtrage selon plusieurs critères.
- Cartographie des zones à risque.

Authentification

- Inscription des utilisateurs.

- Validation par email.
- Authentification sécurisée via JWT.

Module prédictif

- Prédiction de la gravité des accidents.
- Utilisation des techniques de machine learning.

2.2.2 Besoins non fonctionnels

Performance Le système doit gérer un volume important de données. Le chargement des données doit être optimisé grâce au traitement par chunks. Le pipeline ETL doit assurer une exécution fluide des traitements.

Sécurité Les accès doivent être sécurisés via JWT. Les mots de passe doivent être chiffrés. Les comptes utilisateurs doivent être validés avant activation.

Fiabilité Le système doit permettre l'ajout progressif des données. Un mécanisme de suppression des doublons doit être mis en place.

Maintenabilité L'architecture doit être modulaire. Le backend doit être organisé en modules indépendants. Le frontend doit utiliser des composants réutilisables.

2.3 Identification des acteurs

Dans cette section, nous présentons les différents acteurs intervenant dans le système ainsi que leurs rôles principaux au sein de la plateforme.

TABLEAU 2.1 – Identification des acteurs du système

Acteur	Rôle
Utilisateur	Consulte les tableaux de bord interactifs ,effectue des prédictions de la gravité des accidents , supervise le pipeline ETL et contrôle les traitements des données.
Système ETL	Assure l'extraction, le nettoyage, la transformation et le chargement des données.
Modèle de Machine Learning	Génère les prédictions de la sévérité des accidents à partir des données du datamart.

2.4 Diagramme de cas d'utilisation

Le diagramme suivant présente les principales fonctionnalités offertes par la plateforme ainsi que les différentes interactions de l'utilisateur avec le système.

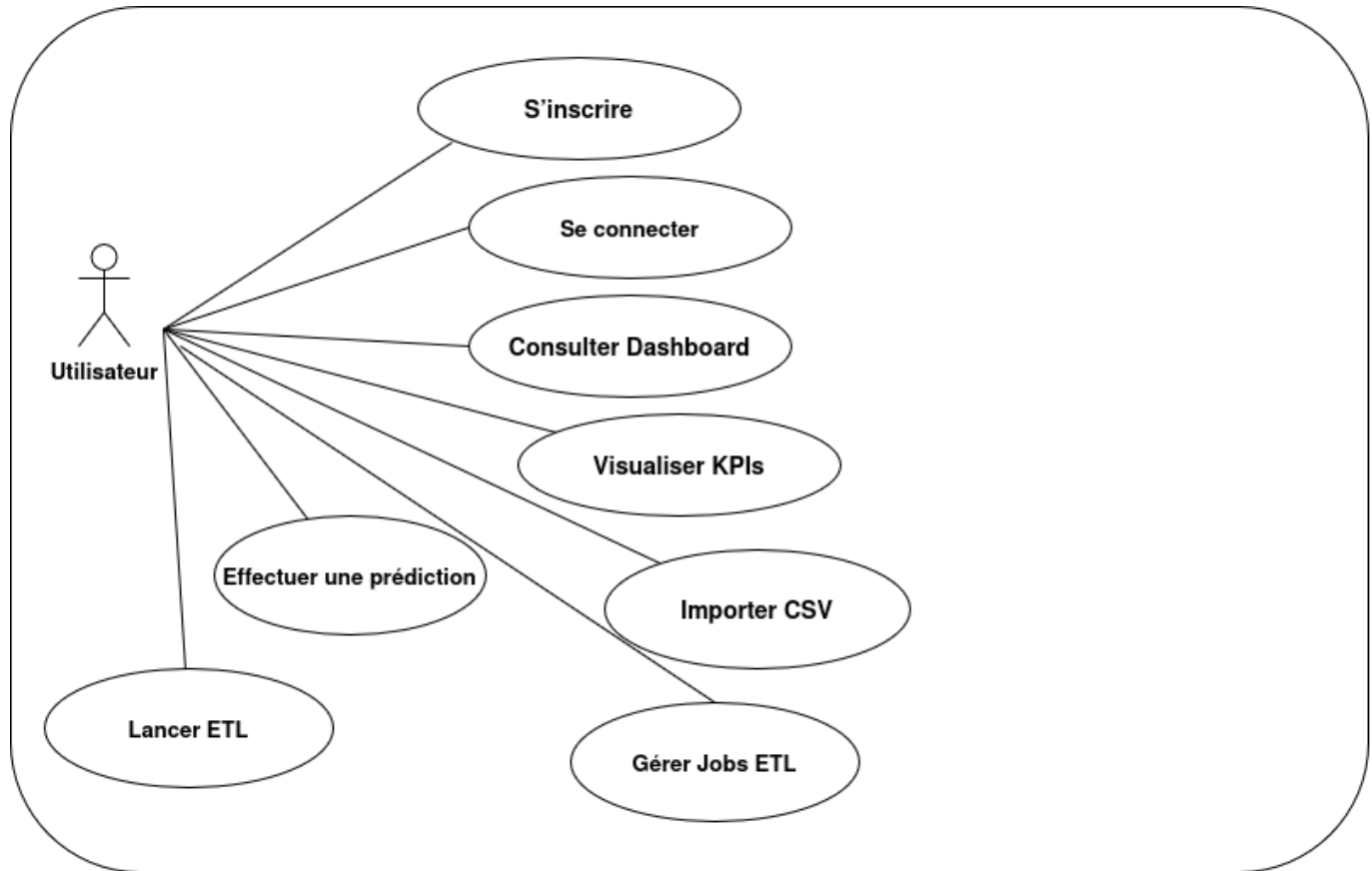


FIGURE 2.1 – Les principales fonctionnalités offertes par la plateforme

L'utilisateur peut accéder aux fonctionnalités de la plateforme après authentification, notamment l'inscription et la connexion. Il peut ensuite effectuer des prédictions de la gravité des accidents, consulter le dashboard interactif, visualiser les KPI, importer des fichiers CSV, gérer les jobs ainsi que lancer le pipeline ETL.

2.5 Architecture générale

2.5.1 Introduction

La plateforme proposée repose sur une architecture modulaire organisée en plusieurs couches permettant de séparer les responsabilités du système et de faciliter la maintenance ainsi que l'évolution de l'application.

2.5.2 Présentation de l'architecture

TABLEAU 2.2 – Architecture en couches de la plateforme

Couche	Technologies	Responsabilités
Présentation	React, TypeScript, CSS , Vite	Création de l'interface utilisateur, affichage des visualisations et gestion des interactions utilisateur.
API REST	Flask	Gestion des endpoints API, authentification des utilisateurs et communication entre les différentes couches du système.
ETL & Métier	Python, Pandas, SQLAlchemy	Importation des fichiers CSV, nettoyage, transformation, chargement des données et gestion des traitements ETL.
Machine Learning	Random Forest	Entraînement du modèle et prédiction de la gravité des accidents.
Base de données	PostgreSQL	Stockage des utilisateurs, des données brutes, des données nettoyées, du datamart et de l'historique des jobs ETL.

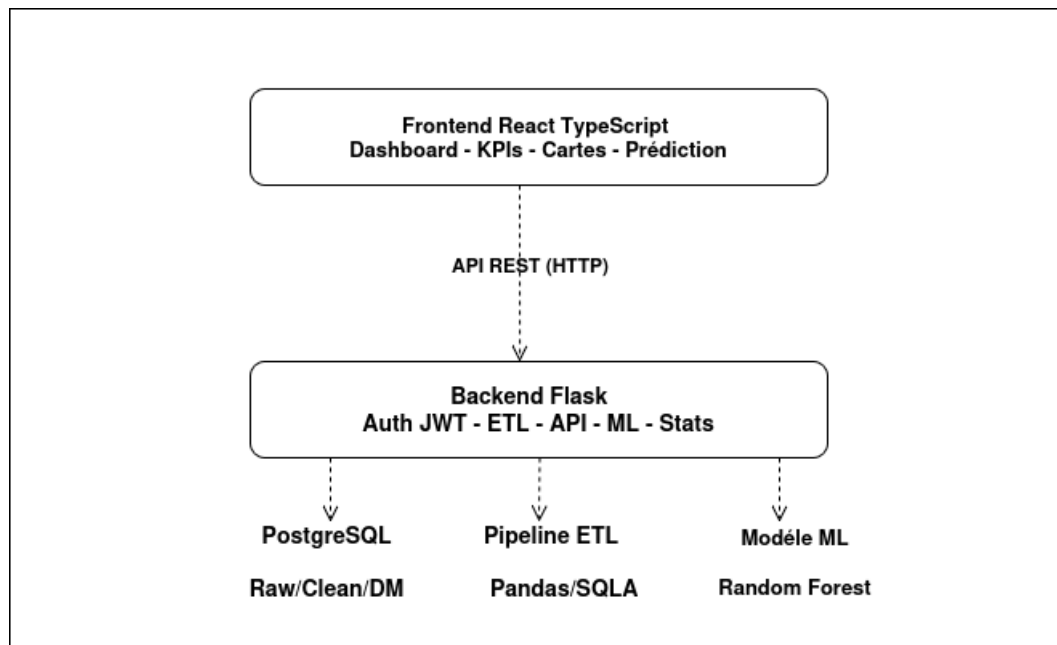


FIGURE 2.2 – Architecture générale de la plateforme

La structure de la plateforme est fondée sur une organisation en plusieurs niveaux qui permet de distinguer les responsabilités du système.

La couche frontend, développée avec React, TypeScript (TSX) et Vite, fournit une interface

utilisateur interactive pour les tableaux de bord, les visualisations et les fonctionnalités de prédiction. L'utilisation de TypeScript améliore la maintenabilité et la détection des erreurs grâce au typage statique, tandis que Vite permet un démarrage rapide du projet, un rechargement instantané et de meilleures performances de développement.

L'API REST, développée en utilisant Flask, permet de gérer les utilisateurs, d'assurer une authentification sécurisée via JWT, de contrôler le pipeline ETL et d'interagir avec le modèle d'apprentissage machine.

Le pipeline ETL, conçu à l'aide de Python et Pandas, est responsable de l'importation, du nettoyage, de la transformation et de l'intégration des données dans le datamart.

Le module de machine learning utilise un modèle Random Forest pour prédire la gravité des accidents à partir des données préparées.

PostgreSQL assure le stockage des données brutes, des données nettoyées, du datamart ainsi que des informations liées aux utilisateurs et aux traitements ETL.

Enfin, Dans le cadre du développement du projet, plusieurs outils ont également été utilisés :

- **GitHub** : outil de collaboration utilisé pour la gestion de version du code source et le travail en équipe, permettant de suivre les modifications et de synchroniser le développement entre les différents membres du projet ;
- **Postman** : test des API REST ;
- **Overleaf** : rédaction du rapport en LaTeX.

2.6 Conception du Datamart

2.6.1 Présentation du schéma étoile

Le datamart de la plateforme est conçu selon une architecture en schéma étoile (Star Schema), optimisée pour les analyses décisionnelles et les requêtes OLAP.

Le modèle est centré autour d'une table de faits représentant les accidents routiers et de plusieurs tables de dimensions permettant l'analyse multidimensionnelle des données.

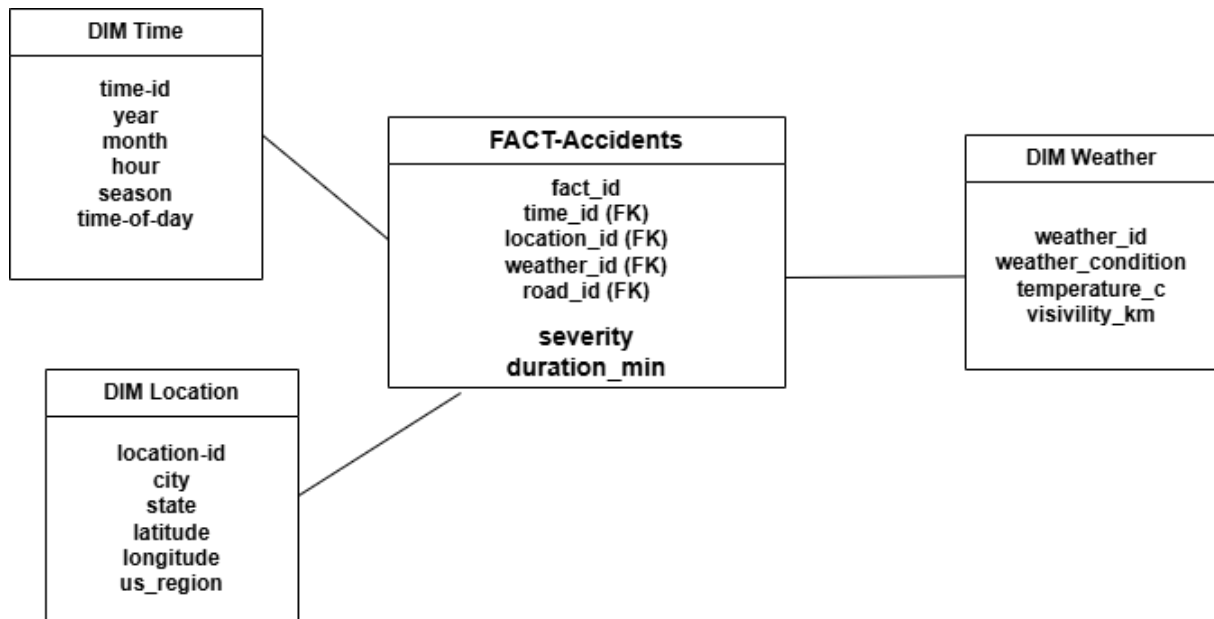


FIGURE 2.3 – Diagramme de classes du système

Le datamart de la plateforme est structuré selon une architecture en schéma étoile, centrée autour de la table de faits *FactAccident*. Cette table contient les mesures principales liées aux accidents routiers ainsi que les clés étrangères permettant de la connecter aux différentes dimensions.

Les dimensions permettent d’analyser les accidents selon plusieurs axes : le temps, la localisation géographique, les conditions météorologiques et les caractéristiques de la route.

Ce modèle est optimisé pour les analyses décisionnelles et les requêtes OLAP, facilitant ainsi l’exploration multidimensionnelle des données.

Relations entre tables

La base de données PostgreSQL implémente un schéma en étoile avec les relations suivantes :

- La table de faits `fact_accident` est reliée à la table `dim_time` par la clé étrangère `time_id`
- La table de faits `fact_accident` est reliée à la table `dim_location` par la clé étrangère `location_id`
- La table de faits `fact_accident` est reliée à la table `dim_weather` par la clé étrangère `weather_id`

2.6.2 Description des tables du schéma étoile

Le datamart de la plateforme est organisé selon un modèle en schéma étoile composé d’une table de faits centrale et de plusieurs tables de dimensions permettant l’analyse multidimensionnelle des données d’accidents routiers.

TABLEAU 2.3 – Description des tables du schéma étoile

Table	Type	Description
FactAccident	Table de faits	Contient les mesures principales liées aux accidents routiers, notamment la gravité, la durée et les clés étrangères vers les différentes dimensions.
DimTime	Dimension	Regroupe les informations temporelles telles que l'année, le mois, l'heure, la saison et le moment de la journée.
DimLocation	Dimension	Contient les informations géographiques des accidents comme la ville, l'état, la région, la latitude et la longitude.
DimWeather	Dimension	Stocke les conditions météorologiques associées aux accidents, notamment la température, la visibilité et les conditions climatiques.

Exemple de données stockées

dim_location :

	location_id [PK] integer ↗	city character varying (100) ↗	state character varying (50) ↗	latitude double precision ↗	longitude double precision ↗	us_region character varying (30) ↗
1	1	Sun City West	AZ	33.6633	-112.361	Ouest
2	2	Miami	FL	25.8039	-80.2059	Sud
3	3	Camarillo	CA	34.2156	-119.0136	Ouest

FIGURE 2.4 – Structure de la table dim_location

dim_time :

	time_id [PK] integer ↗	year integer ↗	month integer ↗	day integer ↗	hour integer ↗	day_of_week integer ↗	week_of integer ↗	season character varying (10) ↗	time_of_day character varying (15) ↗	is_weekend boolean ↗
1	1	2022	8	1	16	1	[null]	Été	Après-midi	false
2	2	2022	4	7	20	4	[null]	Printemps	Soir	false
3	3	2022	8	9	23	2	[null]	Été	Nuit	false

FIGURE 2.5 – Structure de la table dim_time

dim_weather :

	weather_id [PK] integer ↗	weather_condition character varying (100) ↗	temperature_c double precision ↗	visibility_km double precision ↗	temp_bucket character varying (20) ↗	visibility_bucket character varying (20) ↗
1	1	Mostly Cloudy	5	10	Froid	Bonne
2	2	Snow / Windy	-7	0	Glacial	Faible
3	3	Snow	-1	1	Glacial	Faible

FIGURE 2.6 – Structure de la table dim_weather

etl_jobs :

	id [PK] integer	name character varying (255)	job_type character varying (50)	status character varying (50)	schedule character varying (255)	last_run_at timestamp without time zone	created_at timestamp without time zone
1	53	load-raw	manuel	success	[null]	2026-05-13 11:32:02.893124	2026-05-13 11:32:02.893124
2	54	build-clean	manuel	success	[null]	2026-05-13 11:40:52.66561	2026-05-13 11:40:52.66561
3	64	build-datamart-2022	manuel	success	[null]	[null]	2026-05-13 11:40:52.66561

FIGURE 2.7 – Structure de la table etl_jobs

users :

	id [PK] integer	email character varying (255)	password_hash character varying (255)	created_at timestamp without time zone
1	25	fidaasassi@gmail.com	script:32768:8:1\$u2GUrImG...	2026-04-17 19:26:20.11244
2	31	fida.sassi.isic@gmail.c...	script:32768:8:1\$Ju4aJrpM...	2026-05-13 21:49:29.199196

FIGURE 2.8 – Structure de la table users

2.7 Pipeline ETL

Le module ETL représente le noyau du système de gestion des données intégré à la plateforme. Conçu avec Flask et Python, cet outil facilite la gestion complète du cycle de vie des données, depuis leur collecte jusqu'à leur conversion en un datamart utilisable pour l'analyse et la visualisation.

Le processus est organisé en différentes phases consécutives assurant la qualité, l'homogénéité et la traçabilité des données.

2.7.1 Extraction et analyse des données

La phase initiale du processus implique l'importation des fichiers CSV renfermant les informations sur les accidents de la route.

Pour assurer l'efficacité du système, les données sont analysées par segments (chunks), ce qui permet de gérer des volumes importants sans surcharger la mémoire.

On procède ensuite à une analyse initiale, comprenant notamment :

L'examen de la structure des données, l'analyse de la répartition temporelle (par an) et la détection des premières irrégularités potentielles.

2.7.2 Chargement des données brutes (Staging)

Les données récupérées sont par la suite conservées dans une table de préparation nommée AccidentRaw.

Cette phase représente l'aspect brut du système et autorise :

- Le nettoyage basique des catégories de données.
- L'élimination des répétitions.

- Le chargement progressif des données récentes pour prévenir les redondances.

Chaque instance du pipeline est documentée en tant que tâche ETL, garantissant ainsi un suivi intégral des opérations réalisées.

2.7.3 Transformation et nettoyage des données

Cette étape vise à rendre les données exploitables pour l'analyse.

Les opérations effectuées incluent :

- la conversion des types de données (dates, valeurs numériques), la normalisation des unités (par exemple : °F vers °C, miles vers kilomètres),
- l'enrichissement des données avec des variables dérivées (saison, moment de la journée, niveau de gravité),
- ainsi que la suppression des valeurs incohérentes ou invalides.

Le résultat de cette phase est stocké dans la table **AccidentClean**.

2.7.4 Construction du datamart

Les données épurées sont par la suite employées pour nourrir un datamart analytique organisé suivant un modèle en étoile (star schema).

Ce modèle comprend les éléments suivants :

- Une table de faits : **FactAccident**, qui illustre les incidents d'accidents, plusieurs tables de dimensions :
- la dimension temps,
- la dimension localisation,
- la dimension météo et la dimension route.

Ce modèle est affiné pour les demandes analytiques et facilite une utilisation performante dans les tableaux de bord de décision.

2.7.5 Gestion des jobs ETL

Chaque exécution du pipeline ETL est suivie via un système de gestion des jobs permettant le monitoring du processus.

Les informations enregistrées incluent :

- le statut du job (en cours, succès, échec),
- le nombre d'enregistrements traités,
- le nombre d'inserts et d'enregistrements ignorés,
- la durée d'exécution,
- ainsi que l'historique complet des traitements.

Ce mécanisme assure une meilleure supervision du pipeline et facilite le diagnostic en cas d'erreur.

2.7.6 Sécurité et contrôle d'accès

L'ensemble des opérations ETL est sécurisé afin de garantir l'intégrité du système.

Les principales mesures mises en place sont :

- la protection des endpoints via authentification JWT,
- la gestion centralisée des erreurs,
- ainsi que des mécanismes de rollback permettant de garantir la cohérence des données en cas d'échec d'un traitement.

2.8 Diagrammes de séquence

2.8.1 Inscription et vérification email

Le diagramme de séquence suivant illustre le processus d'inscription d'un utilisateur ainsi que la vérification de son adresse email. Il montre les différentes interactions entre l'utilisateur, l'interface frontend, l'API Flask et le système d'envoi d'emails afin de garantir une authentification sécurisée.

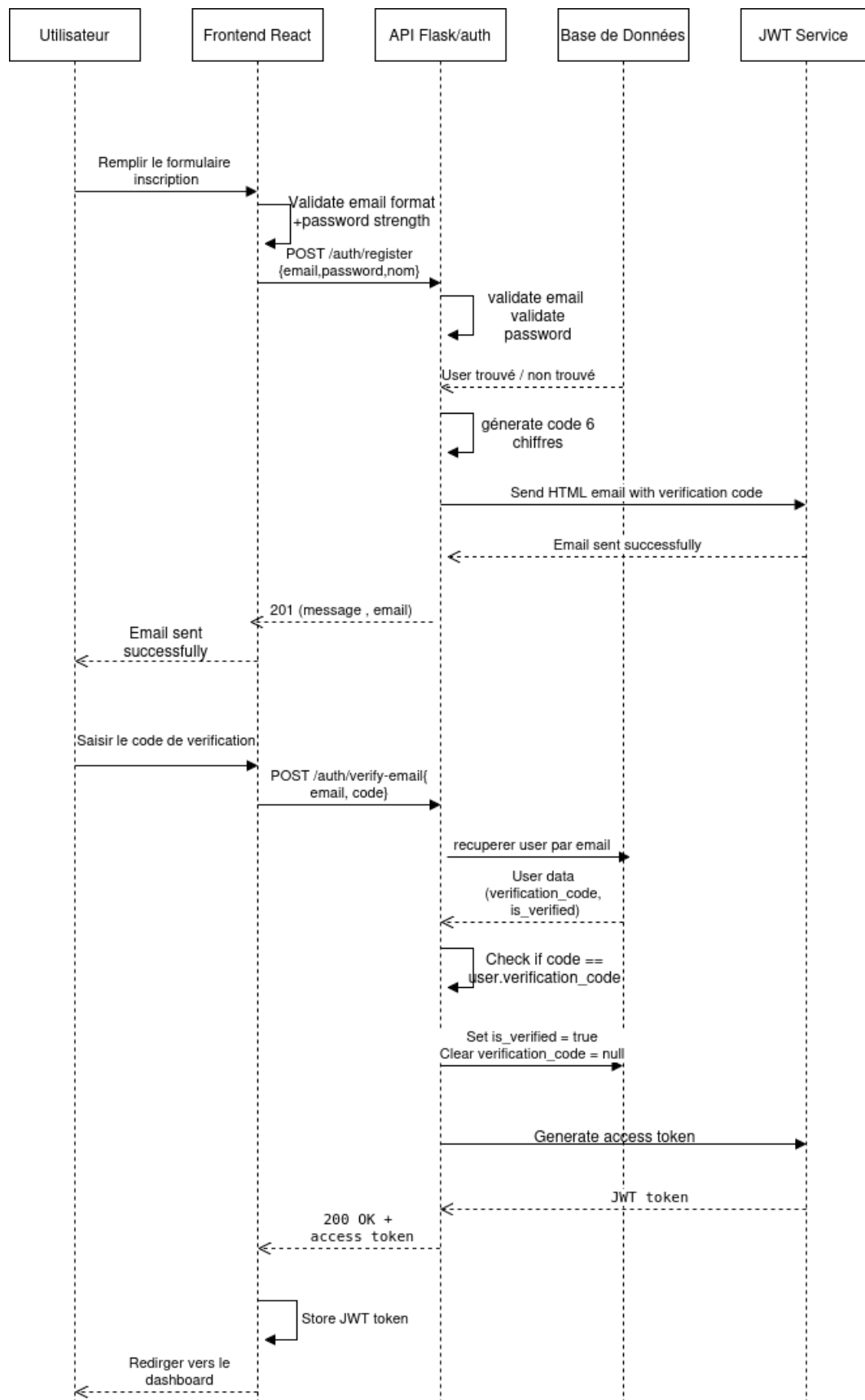


FIGURE 2.9 – Diagramme de séquence d’inscription et vérification email

2.8.2 Connexion utilisateur

Le diagramme de séquence suivant présente le processus de connexion utilisateur au sein de la plateforme. Il décrit les échanges entre l'utilisateur, l'interface frontend et l'API Flask lors de la vérification des identifiants et de la génération du token JWT permettant l'accès sécurisé au système.

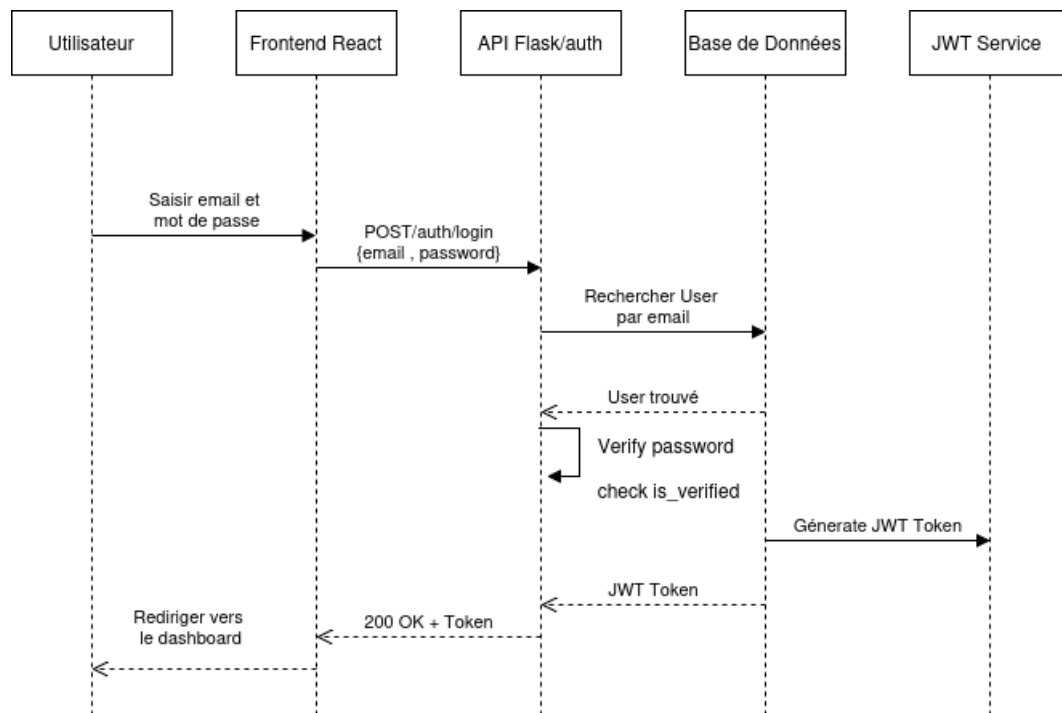


FIGURE 2.10 – Diagramme de séquence de connexion utilisateur

2.8.3 Diagramme de séquence de Pipeline ETL

Le diagramme de séquence suivant illustre le déroulement complet du pipeline ETL au sein de la plateforme. Il présente les différentes étapes d'extraction, de transformation et de chargement des données ainsi que les interactions entre les composants responsables du traitement, du stockage et du suivi des jobs ETL.

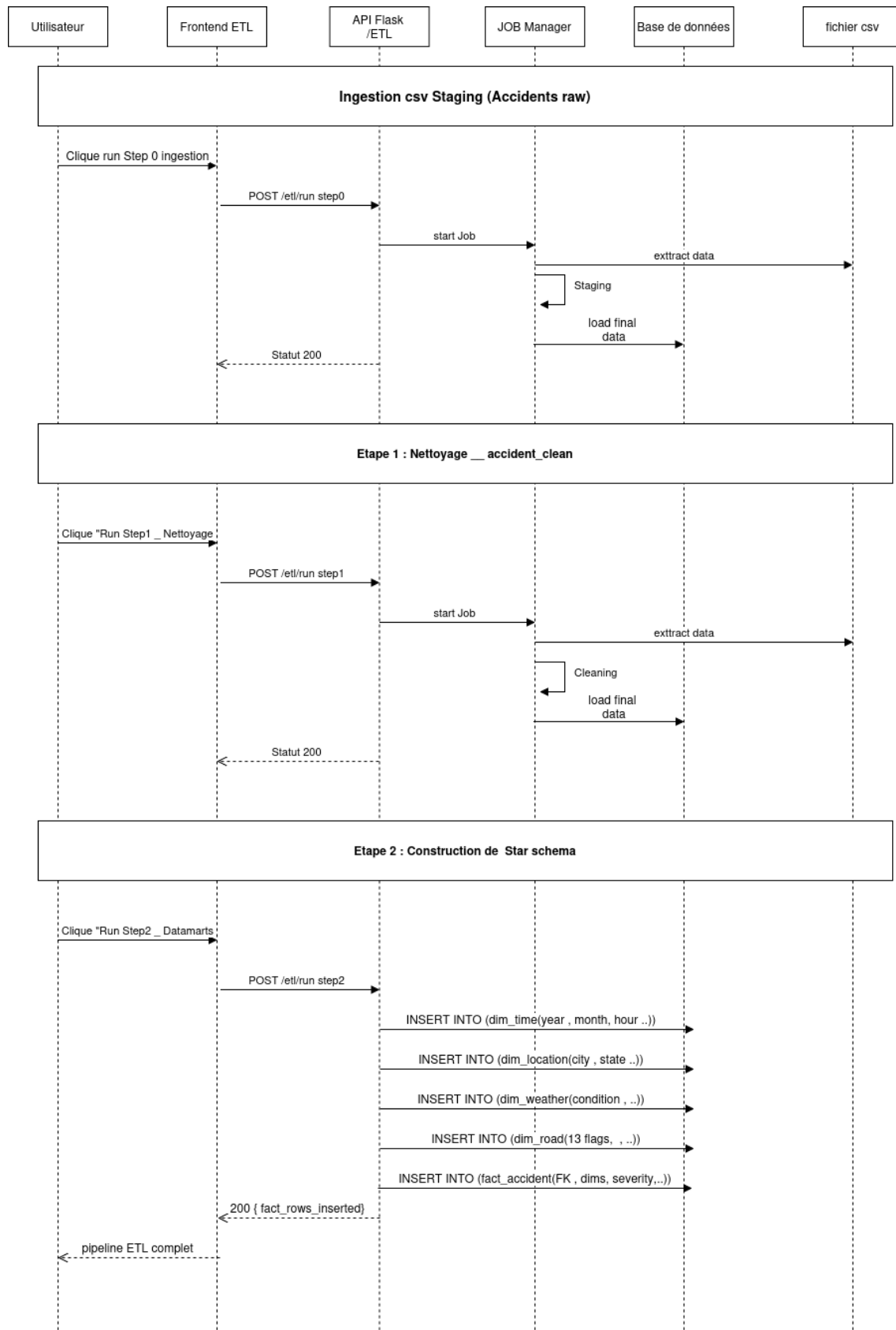


FIGURE 2.11 – Diagramme de séquence du pipeline ETL

2.9 Conclusion

Dans ce segment, nous avons mis en avant l'évaluation des exigences et la conception globale de la plateforme dédiée à l'analyse des accidents de la route. Nous avons aussi examiné la structure du système, l'élaboration du datamart selon le format étoilé ainsi que le processus ETL qui permet la manipulation des données.

De plus, des diagrammes de séquence ont été élaborés pour représenter les interactions clés du système, y compris les processus d'authentification et de traitement ETL.

Cette phase de conception constitue une base importante pour la réalisation technique de la plateforme, qui sera présentée dans le chapitre suivant.

Réalisation et Implémentation du Système

3.1 Introduction

Ce chapitre présente la réalisation technique de la plateforme d'analyse des accidents routiers. Il décrit les principales technologies utilisées ainsi que l'implémentation des différents modules du système, notamment le backend, le frontend, le pipeline ETL et le module de machine learning.

La plateforme repose sur une architecture fullstack intégrant un backend développé avec Flask, un frontend développé avec React et une base de données PostgreSQL pour le stockage des données.

3.2 Réalisation et Implémentation du Système

3.2.1 Environnement de Développement

TABLEAU 3.1 – Technologies utilisées dans le projet

Technologie	Utilisation
Python	Développement backend et ETL
Flask	API REST
React + TypeScript	Développement frontend
PostgreSQL	Base de données
Pandas	Traitement des données
Scikit-learn	Machine Learning
JWT	Authentification sécurisée

3.3 Implémentation du Backend

Le backend représente le cœur central de l'application. Il est conçu avec Flask et structuré en modules pour faciliter la maintenance et l'évolution du système.

Plusieurs fonctionnalités sont assurées par le backend :

- gestion des utilisateurs ;
- authentification sécurisée ;
- gestion des traitements ETL ;
- accès aux données analytiques ;
- communication avec le modèle de machine learning.

L'API REST facilite l'interaction entre le frontend React et les divers services au sein du système.

3.3.1 API REST

Route	Méthode	Description
/api/auth/login	POST	Authentification – Vérifie les identifiants d'un utilisateur et retourne un token JWT en cas de succès.
/api/stats/overview	GET (JWT)	Retourne un résumé global des accidents : nombre total, gravité moyenne, années couvertes et distribution des sévérités.
/api/stats/by-month	GET (JWT)	Analyse mensuelle des accidents : nombre d'accidents et gravité moyenne par mois.
/api/stats/by-year	GET (JWT)	Analyse temporelle annuelle (YoY) avec le nombre d'accidents, gravité moyenne et durée moyenne.
/api/stats/by-severity	GET (JWT)	Distribution des accidents selon les niveaux de gravité (nombre et pourcentage).
/api/stats/by-state	GET (JWT)	Classement des accidents par état avec calcul de la gravité moyenne par région.
/api/stats/map-points	GET (JWT)	Données géographiques pour la carte : statistiques par état et par ville (latitude, longitude, intensité des accidents).
/api/stats/by-weather	GET (JWT)	Analyse des accidents selon les conditions météorologiques (nombre, pourcentage et gravité moyenne).
/api/stats/by-hour	GET (JWT)	Heatmap des accidents selon l'heure et le jour de la semaine (identification des pics horaires).
/api/stats/by-env-bucket	GET (JWT)	Analyse des accidents selon les conditions environnementales : température et visibilité.
/api/stats/filter-options	GET (JWT)	Retourne les valeurs disponibles pour les filtres (années, états, mois, sévérités) utilisées dans le dashboard.

Organisation technique des APIs Flask

3.3.2 Détails techniques des APIs Flask

Organisation du backend

US-ACCIDENTS-ETL-DASHBOARD/

```
backend/  
  app/  
    __init__.py  
    models.py  
    config.py  
      # Contient :  
      # - URI de connexion PostgreSQL  
      # - Paramètres JWT  
      # - Configuration du serveur mail  
  
    routes/  
      auth.py  
        # Authentification :  
        # register, login, verify-email  
  
      stats.py  
        # KPIs et statistiques dashboard  
        # overview, by-month, by-year, etc.  
  
      etl.py  
        # Étapes ETL :  
        # analyze-csv, load-raw, build-clean  
  
      predict.py  
        # API de prédiction de sévérité  
  
      datamart.py  
        # Construction du DataMart  
        # ETL Step 3 : build-datamart  
  
    scripts/  
      train_model.py  
        # Entraînement du modèle Random Forest  
  
wsgi.py : le point d'entrée principal de l'application Flask  
  # il charge les variables d'environnement depuis le fichier .env  
  # initialise l'application puis démarre le serveur sur le port configuré.
```

Formats des réponses JSON

Toutes les routes API retournent des réponses au format JSON. Voici les exemples extraits directement du code :

1. Authentification - POST /auth/login (fichier auth.py)



FIGURE 3.1 – Test de l'endpoint `/auth/login` via Postman

3. Vue d'ensemble - GET /api/stats/overview (fichier stats.py)

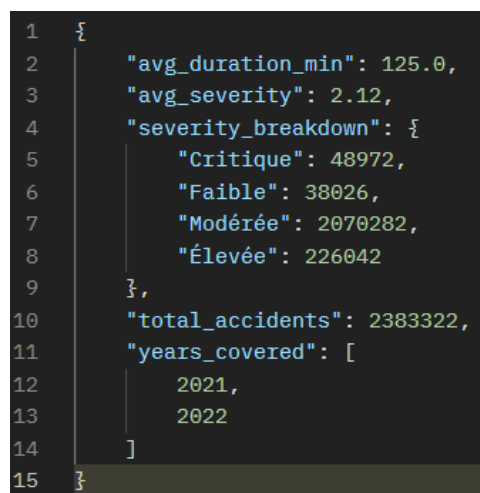


FIGURE 3.2 – Test de l'endpoint `/api/stats/overview` via Postman

8. Exemples d'erreurs possibles

Exemples d'erreurs lors de la connexion

Réponse (401 Non autorisé) :

```
{
  "message": "Identifiants invalides"
}
```

Réponse (400 Requête invalide) :

```
{
  "message": "Email déjà enregistré"
}
```

3.3.3 Authentification et Sécurité

L'accès aux diverses fonctionnalités de la plateforme est protégé par un système d'authentification basé sur JWT.

L'outil d'authentification offre la possibilité de :

- l'inscription des utilisateurs ;
- la vérification des emails ;
- la connexion sécurisée ;
- la réinitialisation des mots de passe.

Les mots de passe sont hachés avant leur stockage dans la base de données afin de garantir un niveau élevé de sécurité.

En outre, l'accès aux endpoints sensibles de l'API est strictement contrôlé : toute requête non accompagnée d'un token JWT valide est automatiquement rejetée. Cela empêche l'accès non autorisé aux ressources protégées et garantit que seules les utilisateurs authentifiés peuvent interagir avec les fonctionnalités sécurisées du système.

3.3.4 Gestion des erreurs et performances

Gestion des erreurs ETL :

- Traitement par lots
- Gestion des lignes invalides
- Contrôle des doublons
- Jobs ETL traçables
- Rollback transactionnel

Performances du système

Plusieurs optimisations ont été implémentées pour garantir des performances acceptables :

Optimisations base de données :

- **Schéma en étoile** : Accélération des requêtes d'agrégation (COUNT, AVG, GROUP BY)
- **Indexation PostgreSQL** : Index automatique sur clés primaires et étrangères
- **Jointures rapides** : Liens optimisés entre `fact_accident` et les dimensions

3.4 Implémentation du Frontend

Le frontend de la plateforme est développé avec React et TypeScript afin de fournir une interface utilisateur moderne et interactive.

L'application frontend contient plusieurs pages principales :

- page de connexion ;
- page d'inscription ;
- tableau de bord analytique ;
- page d'importation des fichiers CSV (interface ETL permettant le chargement et le traitement des données) ;
- page de prédiction des accidents.

Le dashboard permet d'afficher plusieurs indicateurs et graphiques interactifs facilitant l'analyse des accidents routiers.

3.5 Implémentation du Module de Machine Learning

3.5.1 Objectif du Modèle

Le module de machine learning vise à :

- prédire le niveau de gravité d'un accident routier (classes 1 à 4) ;
- exploiter les données issues du datamart en schéma en étoile ;
- fournir des résultats en temps réel via l'API Flask ;

Les variables utilisées proviennent principalement des dimensions :

- dimension temporelle (heure, jour, saison, etc.) ;
- dimension géographique (état, région) ;
- dimension météorologique (température, visibilité, conditions météo) ;
- dimension routière (infrastructure et signalisation).

3.5.2 Préparation des données

Les données d'apprentissage sont extraites directement du datamart structuré en schéma en étoile. Une phase de préparation est appliquée afin de garantir leur qualité et leur compatibilité avec les algorithmes de machine learning.

Cette étape comprend :

- la sélection des variables pertinentes issues des dimensions ;
- la gestion des valeurs manquantes (imputation) ;
- la transformation des variables catégorielles en variables numériques exploitables par le modèle, en combinant le one-hot encoding (pour les variables nominales à faible cardinalité), le label encoding (pour certaines variables structurées), ainsi qu'une réduction de cardinalité afin de regrouper les catégories rares sous une classe "Other" ;

Après le prétraitement, les données ont été séparées en deux ensembles :

- un ensemble d'apprentissage (train set) utilisé pour entraîner le modèle ;
- un ensemble de test (test set) utilisé pour évaluer les performances sur des données non vues durant l'apprentissage.

Cette séparation permet de mesurer la capacité de généralisation du modèle et d'éviter le sur-apprentissage.

Cette étape permet d'obtenir un dataset optimisé pour l'apprentissage automatique.

3.5.3 Choix du modèle

Pour la tâche de classification, un algorithme de type Random Forest a été retenu. Ce choix repose sur plusieurs avantages :

- capacité à gérer des données hétérogènes ;
- bonne robustesse face au bruit ;
- performance stable sur des datasets volumineux ;
- limitation du surapprentissage grâce à l'agrégation de plusieurs arbres de décision.

Ce modèle est particulièrement adapté aux problèmes de classification multi-classes comme la prédiction de la gravité des accidents.

3.5.4 Entraînement du modèle

Le processus d'entraînement suit les étapes suivantes :

- séparation des données en ensembles d'entraînement et de test ;
- entraînement d'un premier modèle sur l'ensemble complet des features ;
- analyse de l'importance des variables ;
- sélection des features couvrant 95% de l'importance cumulée ;
- réentraînement du modèle sur l'ensemble réduit ;
- évaluation sur l'ensemble de test.

Hyperparamètres du modèle Random Forest

TABLEAU 3.2 – Hyperparamètres du modèle Random Forest

Hyperparamètre	Valeur	Rôle
n_estimators	300	Nombre d'arbres dans la forêt
max_depth	18	Profondeur maximale des arbres
max_features	sqrt	Sélection des variables à chaque split
class_weight	balanced	Gestion du déséquilibre des classes
oob_score	True	Validation interne du modèle

Calibration des seuils de décision

Le dataset utilisé présente un fort déséquilibre entre les différentes classes de gravité. La majorité des accidents appartient à la classe *Severity 2*, tandis que les classes critiques, notamment *Severity 4*, sont beaucoup plus rares.

Dans une approche classique de classification, le modèle sélectionne uniquement la classe ayant la probabilité la plus élevée. Cette stratégie favorise fortement la classe majoritaire et réduit la capacité du système à détecter les accidents graves.

Afin d'améliorer la prédiction des classes minoritaires, une stratégie de calibration des seuils probabilistes a été mise en place. Le système applique des seuils spécifiques sur les probabilités générées par le modèle Random Forest.

La règle de décision utilisée est la suivante :

$$P(Severity = 1) > t_1 \rightarrow Classe\ 1$$

$$P(Severity = 4) > t_4 \rightarrow Classe\ 4$$

$$P(Severity = 3) > t_3 \rightarrow Classe\ 3$$

$$Sinon \rightarrow Classe\ 2$$

Dans cette stratégie, la classe *Severity 2* représente la classe par défaut. Ainsi, même si la probabilité de la classe 2 est supérieure aux autres, le système peut prédire une classe critique lorsque sa probabilité dépasse un seuil défini.

Par exemple, si :

$$P(Severity = 4) = 0.25$$

et que le seuil calibré est :

$$t_4 = 0.20$$

alors :

$$0.25 > 0.20$$

Le système prédit donc la classe *Severity 4*, même si la classe *Severity 2* possède une probabilité plus élevée.

Cette approche permet d'améliorer la détection des accidents critiques, d'augmenter le rappel des classes rares et d'obtenir de meilleures performances sur les métriques adaptées aux datasets déséquilibrés, telles que le *Balanced Accuracy* et le *Macro F1-score*.

Une attention particulière est accordée à la gestion du déséquilibre des classes afin d'éviter une prédiction biaisée vers les classes majoritaires.

3.5.5 Évaluation des performances

Le modèle est évalué à l'aide de plusieurs indicateurs permettant d'analyser sa performance globale ainsi que sa capacité à bien classer chaque catégorie.

Les principales métriques utilisées sont :

- **Accuracy globale** : mesure la proportion de prédictions correctes sur l'ensemble des données de test.
- **Balanced Accuracy** : permet de prendre en compte le déséquilibre entre les classes.
- **F1-score** : combine la précision et le rappel pour offrir une mesure équilibrée des performances.

Ces indicateurs montrent que le modèle est capable de fournir des prédictions cohérentes et exploitables pour l'analyse des accidents.

3.5.6 Sauvegarde et intégration du modèle

Une fois entraîné, le modèle est sauvegardé sous forme de fichier sérialisé (.pkl). Ce fichier contient :

- le modèle Random Forest entraîné ;
- la liste des features sélectionnées ;
- les paramètres d'entraînement ;
- les métriques de performance.

Le modèle est ensuite intégré dans le backend Flask, où il est chargé lors de l'exécution de l'API de prédiction.

Ainsi, lorsqu'un utilisateur saisit des informations via l'interface React, celles-ci sont envoyées au backend, traitées par le modèle, puis une prédiction de la gravité de l'accident est retournée en temps réel.

3.5.7 Intégration dans la plateforme

Le module de machine learning est directement intégré à l'architecture globale du système. Il communique avec :

- le pipeline ETL pour récupérer les données préparées ;
- le backend Flask pour exposer les prédictions ;
- le frontend React pour l'affichage des résultats.

Cette intégration permet de transformer la plateforme en un système complet d'analyse et de prédiction des accidents routiers.

3.6 Conclusion

Ce chapitre a introduit le module d'apprentissage automatique inclus dans la plateforme. Il a facilité l'explication de la préparation des données, de la sélection du modèle, du

processus d'apprentissage et de l'évaluation des performances.

L'incorporation du modèle Random Forest au sein du système général enrichit la plateforme en ajoutant une fonctionnalité de prévision, qui vient compléter les capacités d'analyse fournies par le datamart et le tableau de bord interactif.

Tests, Résultats et Validation du Système

4.1 Introduction

Ce chapitre présente les résultats obtenus après l'implémentation de la plateforme d'analyse des accidents routiers. Il met en évidence les performances du système dans ses différents modules, notamment le pipeline ETL, le dashboard analytique et le module de machine learning.

4.2 Interfaces dans la plateforme

4.2.1 Page création de compte

Cette page montre la procédure de création d'un compte dans la plateforme.

The screenshot shows a mobile-style interface for account creation. At the top, there's a header with a blue icon and the text 'Plateforme ETL & Dashboard Accidents Routiers'. Below this is a section titled 'Créer un compte' with a subtitle 'rejoignez-nous pour accéder au dashboard'. The form contains four input fields: 'NOM' with the value 'test', 'PRÉNOM' with the value 'tes', 'EMAIL' with the value 'test@gmail.com', and 'MOT DE PASSE' which is masked with asterisks. A blue button labeled 'Créer un compte' is positioned below the password field. At the very bottom, there is a link that says 'Vous avez déjà un compte ? Se connecter >'.

FIGURE 4.1 – Page creation compte

4.2.2 Réception de code après la creation de compte

Après le remplissage de formulaire , un code sera envoyé par email pour vérifier son email.

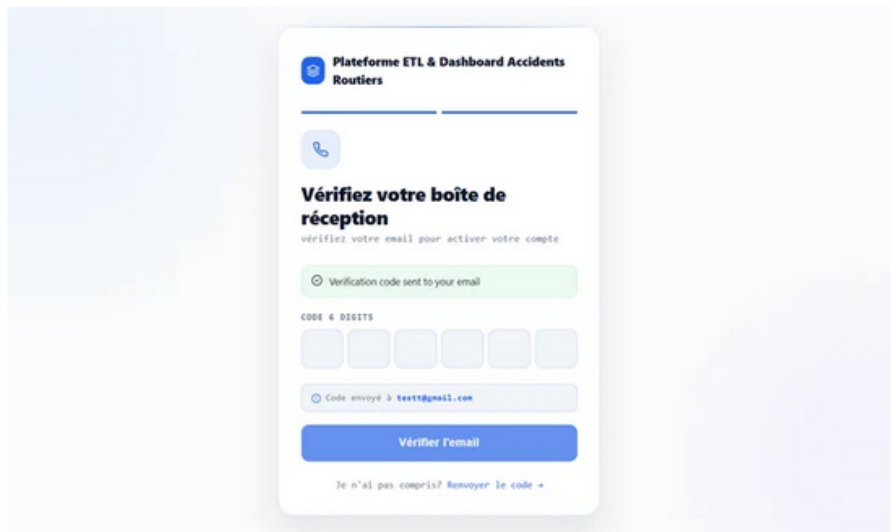


FIGURE 4.2 – Réception code après création de compte

4.2.3 Page connexion

Cette page montre la connexion de l'utilisateur à la plateforme .

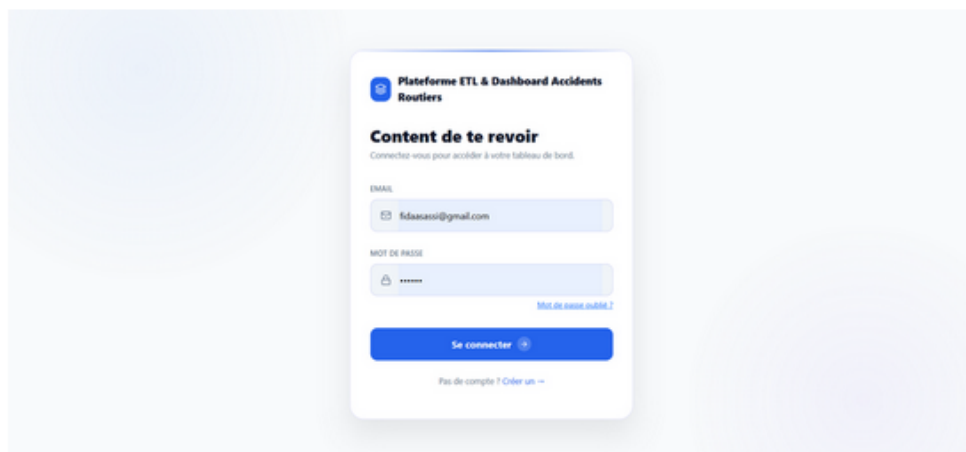


FIGURE 4.3 – Page connexion

4.2.4 Mot de passe est oublié

Si l'utilisateur a oublié son mot de passe , il devra donner son email pour obtenir un code de renititalisation de mot de passe .



FIGURE 4.4 – Mot de passe oublié

4.2.5 Rénitialisation de mot de passe

Un code sera envoyé par mail pour renitialiser le mot de passe

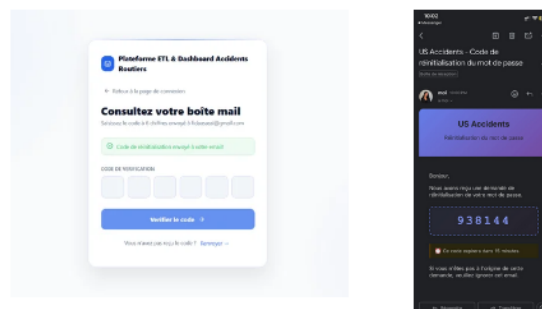


FIGURE 4.5 – Rénitialisation de mot de passe

4.2.6 Page Dashboard

Le tableau de board présente les principaux indicateurs de la plateforme , notamment nombre total d'accidents , répartition de la gravité ainsi que l'évolution des accidents selon les années . Il permet également d'afficher les tendances sur plusieurs années d'analyse . En haut de l'interface des filtres dynamiques et un bouton de rafraichissement permettant de mettre à jour les données en temps réel.

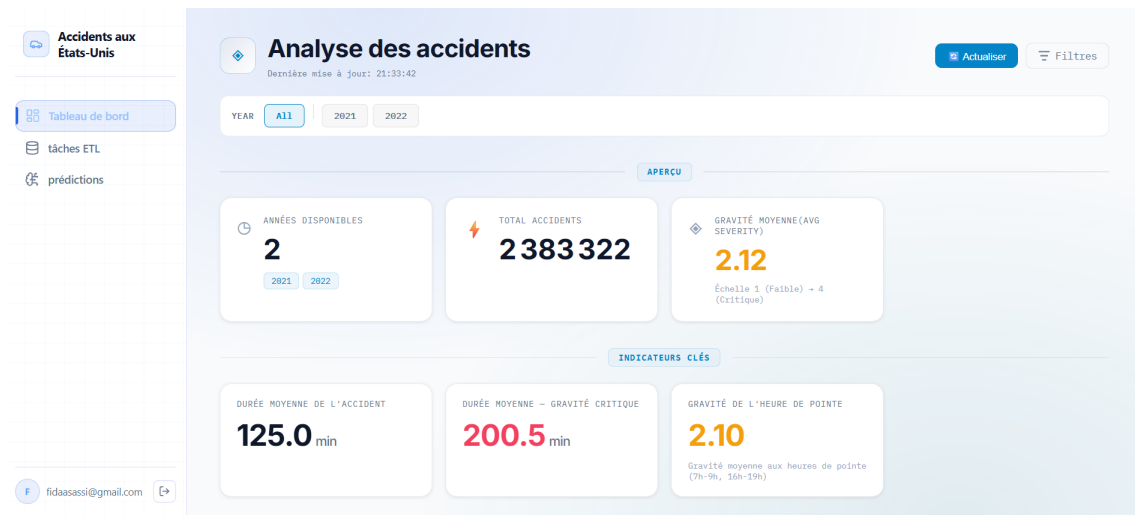


FIGURE 4.6 – Page Dashboard partie -1-

Le tableau de board présente les données sous forme de courbe et de diagrammes afin de visualiser les tendances des accidents. Il inclut également un classement des 10 états les plus touchés permettant d'identifier les zones à haut risques.

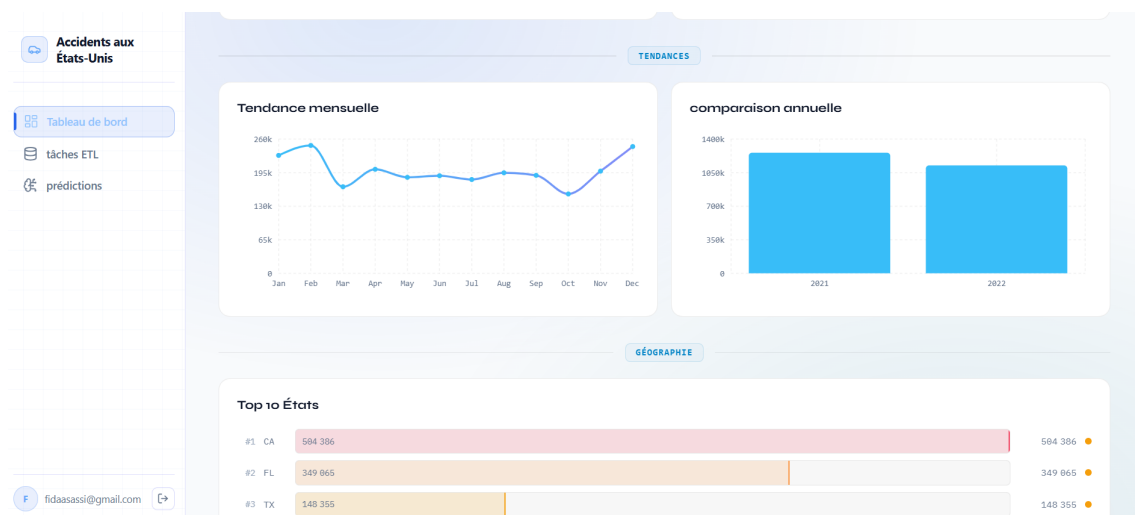


FIGURE 4.7 – Page Dashboard partie -2-

Cette partie montre les zones à risques d'accidents à travers une carte géographique.

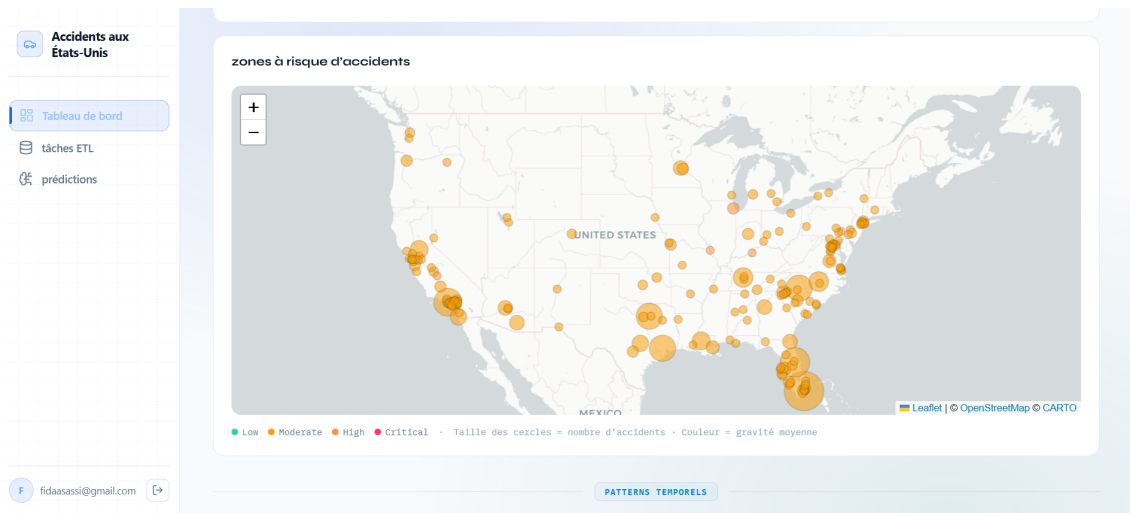


FIGURE 4.8 – Page Dashboard partie -3-

4.2.7 Page Pipeline ETL

Cette interface permet de superviser en temps réel l'évolution des données à travers les différentes étapes de pipeline ETL ainsi que l'historique des traitements.

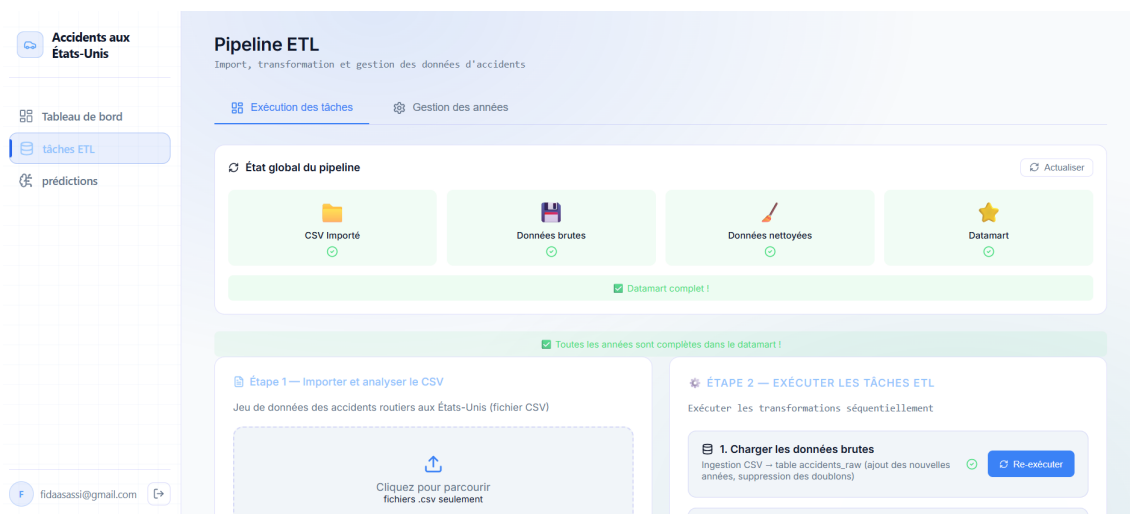


FIGURE 4.9 – Page Pipeline ETL partie -1-

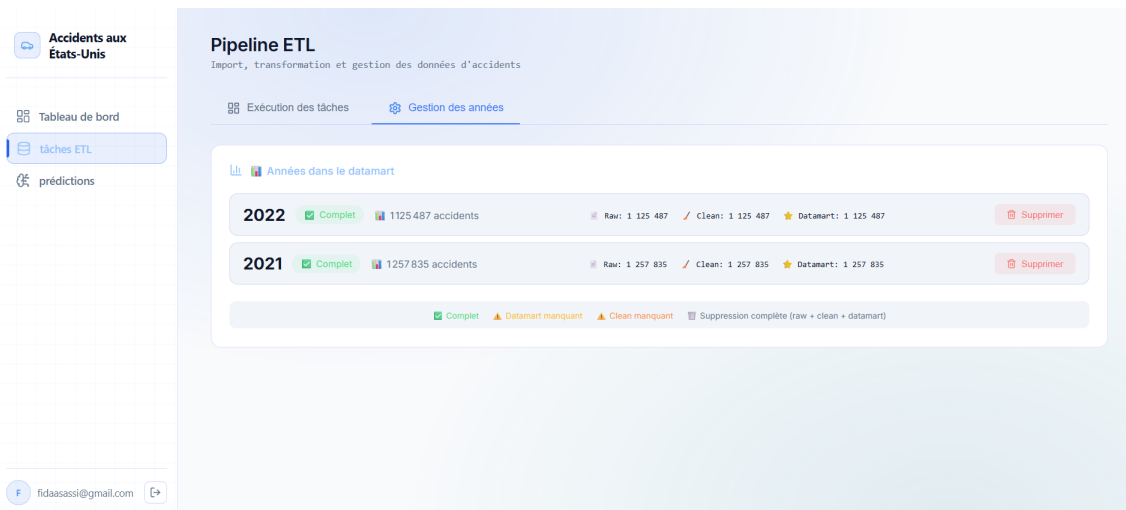


FIGURE 4.10 – Page Pipeline ETL partie -2-

4.2.8 Page prédiction

D'après cette interface, l'utilisateur peut prédire la gravité des accidents routiers. Il choisit les paramètres qu'il veut tester et le résultat s'affiche après.

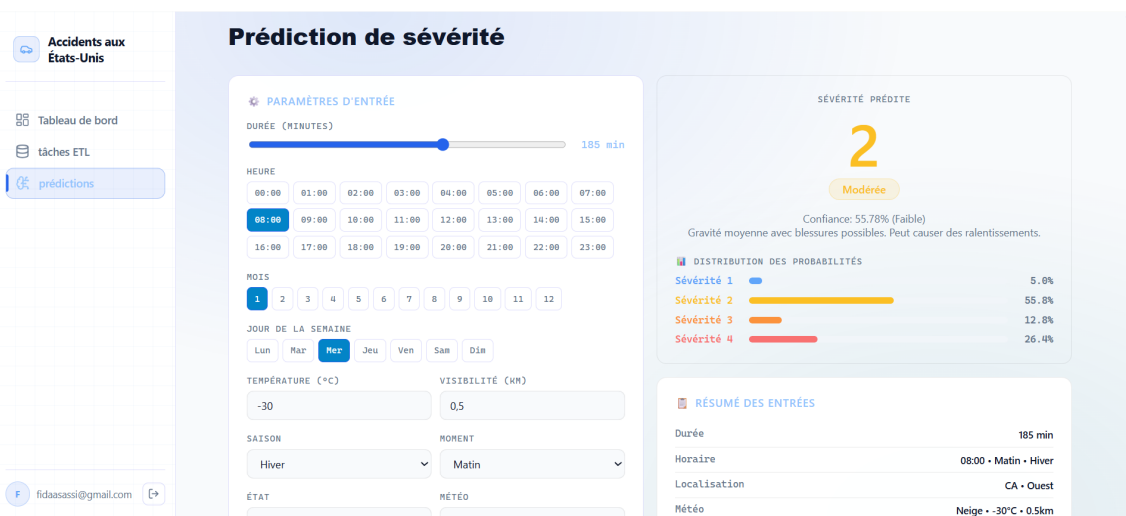


FIGURE 4.11 – Page prediction

4.3 Résultats de module de Machine Learning

1. Présentation des données et déséquilibre des classes

Le modèle **Random Forest** a été entraîné sur un datamart en schéma étoile contenant **2 426 253 accidents routiers**. Les données présentent un fort déséquilibre des classes, avec une dominance importante de la classe **Moderate** représentant **87.1%** des accidents, tandis que les classes **Low** et **Critical** restent minoritaires.

Distribution des classes

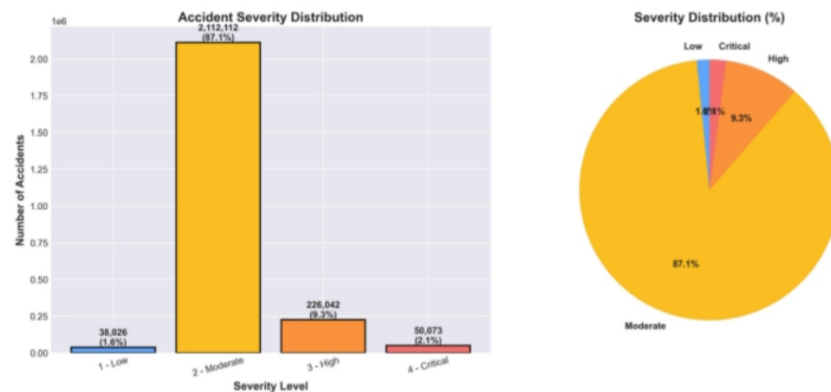


FIGURE 4.12 – Distribution des classes

TABLEAU 4.1 – Distribution des classes de gravité des accidents

Classe	Nombre	Pourcentage
Faible (1)	38 026	1.6%
Modérée (2)	2 070 282	86.9%
Élevée (3)	226 042	9.5%
Critique (4)	48 971	2.1%

Cette distribution justifie l'utilisation de :

- `class_weight='balanced'`
- la métrique **Balanced Accuracy**

afin d'éviter que le modèle favorise uniquement la classe majoritaire.

Prétraitement et réduction des variables

Après encodage et transformation des données, la matrice finale contenait **74 variables explicatives**.

Une étape de sélection basée sur l'importance des variables a permis :

- de conserver uniquement les variables représentant **95%** de l'information utile
- de réduire le nombre de variables de **74 à 32**
- soit une réduction de **52.9%**

Cette optimisation permet de réduire :

- la complexité du modèle
- le temps d'entraînement
- le risque de sur-apprentissage

tout en améliorant la capacité de généralisation.

Variables les plus importantes

TABLEAU 4.2 – Variables les plus importantes du modèle Random Forest

Variable	Importance
duration_min	0.3731
hour	0.0791
month	0.0678
temperature_c	0.0523
season_Été	0.0381

Les résultats montrent que :

- la **durée de l'accident (duration_min)** est la variable la plus influente
- les facteurs temporels jouent un rôle important

Performance finale du modèle

TABLEAU 4.3 – Performance du modèle Random Forest

Métrique	Valeur
Accuracy	61.25%
Balanced Accuracy	71.52%
OOB Score final	0.6126

Interprétation des métriques

Accuracy : L'accuracy mesure le pourcentage global de prédictions correctes. Le modèle atteint **61.25%**, ce qui reste correct compte tenu :

- du déséquilibre important des données
- du nombre élevé d'observations
- de la complexité des facteurs influençant les accidents

Balanced Accuracy : La Balanced Accuracy calcule la moyenne des taux de reconnaissance de chaque classe. Le score obtenu **71.52%** montre que le modèle :

- reconnaît correctement les classes minoritaires
- détecte les accidents graves et critiques
- ne se limite pas à la classe majoritaire

Conclusion

Les résultats obtenus démontrent que le modèle **Random Forest** est capable d'identifier efficacement les niveaux de gravité des accidents routiers à partir des données du datamart.

La sélection des variables basée sur l'importance des caractéristiques a permis :

- de réduire significativement la complexité du modèle
- d'améliorer ses performances
- et d'obtenir une meilleure capacité de généralisation

Le modèle final atteint une **Balanced Accuracy de 71.52%**, ce qui constitue une performance satisfaisante pour un problème réel fortement déséquilibré.

4.3.1 Synthèse des résultats

Le système global a été validé à travers différents tests couvrant l'ensemble des modules de la plateforme.

Les résultats obtenus montrent que le pipeline **ETL** fonctionne correctement, permettant la transformation efficace des données brutes en un **datamart exploitable**.

Le tableau de bord interactif assure une visualisation claire des indicateurs clés, tandis que le module de machine learning fournit des prédictions cohérentes et exploitables pour l'analyse de la gravité des accidents

4.3.2 Limites du projet

- **Performance** : Temps de chargement élevé du tableau de bord dû à la volumétrie des données (2,4M d'accidents). Construction lente du datamart et premier appel API de prédiction ralenti par le chargement du modèle.
- **Modèle ML** : Fort déséquilibre des classes (87% Moderate, 2,1% Critical) limitant la fiabilité sur les sévérités rares. Modèle non généralisable hors États-Unis. Absence de features importantes (vitesse limite, densité de trafic).

4.4 Perspectives d'évolution

Plusieurs améliorations peuvent être envisagées afin d'enrichir la plateforme.

Il serait intéressant d'intégrer des modèles de machine learning plus avancés afin d'améliorer la précision des prédictions.

De plus, l'automatisation complète du pipeline ETL via des outils d'orchestration tels qu'**Apache Airflow** pourrait être envisagée.

Enfin, l'ajout de visualisations géospatiales avancées ainsi que des fonctionnalités de monitoring en temps réel permettrait de renforcer les capacités analytiques du système.

4.5 Conclusion

Ce chapitre a présenté les tests et les résultats obtenus après l'implémentation de la plateforme. L'ensemble des modules a été validé avec succès, notamment le pipeline ETL,

le dashboard et le modèle de machine learning. Les résultats confirment la fiabilité et la cohérence du système dans son ensemble.

Conclusion générale

Ce projet a permis de concevoir et de développer une plateforme complète d'analyse et de prédiction des accidents routiers reposant sur une architecture de données intégrée. Le système commence par l'ingestion de données brutes issues de fichiers CSV, stockées dans une base PostgreSQL sous forme de données brutes (raw), puis traitées à travers un pipeline ETL structuré.

Ce pipeline assure le nettoyage, la normalisation et l'enrichissement des données, ainsi que leur organisation dans un datamart en schéma étoile composé d'une table de faits et de plusieurs dimensions (temps, localisation, météo et route), offrant ainsi une base optimisée pour l'analyse.

Une API REST développée avec Flask garantit ensuite la gestion des utilisateurs et l'authentification sécurisée via JWT, tout en prenant en charge les opérations d'inscription, de connexion et de récupération de mot de passe.

Sur cette base de données structurée, un module de machine learning a été intégré afin de prédire la gravité des accidents. Ce module repose sur un modèle Random Forest entraîné sur les données du datamart et évalué à l'aide de plusieurs métriques (accuracy, balanced accuracy, precision, recall et F1-score), assurant ainsi la fiabilité des prédictions.

Enfin, l'ensemble du système permet de transformer des données brutes en une plateforme intelligente d'aide à la décision, combinant analyse descriptive et prédictive dans le domaine de la sécurité routière.