

**MÉMOIRE DE FIN DE CYCLE**

**INFORMATIQUES OPTION GENIE LOGICIELLE (IGL)**

**CONCEPTION ET RÉALISATION D’UNE APPLICATION WEB DE GESTION :**

**CAS DE L’ENTREPRISE LASSIRE DECHETS SERVICES**

Présenté par :

OUATTARA BITCHERESSE ISSOUF

Période de stage :

Du 18 Juillet – 20 Octobre 2022-2023

**Encadrant académique : Maitre de stage :**

**Dr. ZOKORA ELVIS AGAH AGAH Rodigue**

Enseignant - chercheur



**MÉMOIRE DE FIN DE CYCLE**

**INFORMATIQUES OPTION GENIE LOGICIELLE (IGL)**

**CONCEPTION ET RÉALISATION D’UNE APPLICATION WEB DE GESTION :**

**CAS DE L’ENTREPRISE LASSIRE DECHETS SERVICES**

Présenté par :

OUATTARA BITCHERESSE ISSOUF

Période de stage :

Du 18 Juillet – 20 Octobre 2022-2023

**Encadrant académique : Maitre de stage :**

**Dr. ZOKORA ELVIS AGAH AGAH Rodigue**

Enseignant - chercheur

# **DÉDICACE**

Je dédie ce travail à mon père.

# **REMERCIEMENTS**

La réalisation de ce mémoire m’offre une opportunité pour exprimer ma profonde gratitude à tous ceux qui ont contribué de façon morale, intellectuelle, matérielle au bon déroulement de mon stage et à la finalisation de ce mémoire.

Sincères remerciements à l’ensemble du personnel d’UTA pour leur suivi de mes trois années de formation passé dans leur université.

Je tiens à remercier particulièrement :

-M. Le fondateur d’UTA

-M. ZOKORA mon maître de stage et à la fois mon encadreur qui m’a appris plusieurs choses

-M. Le directeur d’UTA

# **AVANT PROPOS**

Ce mémoire s’inscrit dans le cadre de l’obtention de la Licence en Informatique option Génie Logiciel (IGL) de l’université de technologie d’Abidjan (UTA). J’ai eu l'opportunité d'effectuer un stage au sein de l’entreprise LASSIRE DECHETS SERVICES, en tant que stagiaire en génie logiciel. Ce stage, d'une durée de 3 mois, a été une expérience enrichissante et formatrice qui m'a permis d'acquérir de nouvelles compétences et d'approfondir mes connaissances dans le domaine du génie logiciel.

Ce stage m'a donné l'occasion de mettre en pratique les concepts théoriques appris au cours de ma formation universitaire à l’Université de Technologie d’Abidjan. Cette expérience m'a permis de mieux comprendre les défis du monde professionnel et d'appréhender les méthodologies de développement de logiciels utilisées en entreprise.

Dans un environnement technologique en constante évolution, une gestion efficace permet d’optimiser l’utilisation des ressources, y compris le temps, le personnel et le budget, pour atteindre les objectifs du projet de manière rentable.

Cette recherche vise à apporter une contribution en mettant en lumière les possibilités offertes par la technologie dans le domaine de la gestion. Au regard des données exponentielles que l’entreprise LASSIRE DECHETS SERVICES gère, une application de gestion peut jouer un rôle crucial dans l’amélioration de la productivité et de l’efficacité.

# **SIGLE ET ABREVIATION**

**ANASUR :** Agence Nationale de la Salubrité Urbaine

**API :** Application programming interface

**CIAPOL :** Application Centre Ivoirien Antipollution

**CRUD :**Create,Read,Update, Delete

**DIND :** Déchets Industriels Non Dangereux

**DID :** Déchets Industriels Dangereux

**HTTP:** Hyper text transfer protocol

**JSON :** JavaScript object notation

**JWT:** JSON web token

**MCD** : model conceptuel de donnée

**MERISE** : Méthodes d’Etude et de Réalisation Informatique des Systèmes d’Entreprise

**MLD** : modèle logique de donnée

**MLDR** : modèle logique de données relationnelles

**SGBD** : système de gestion de base de données

**SQL**: *Structure Query Language*

**UML:** Unified Modeling Language

# **LISTE DES FIGUES**

**Figure 1 :** Structure de l’application

**Figure 2 :** Diagramme de Gant

**Figure 3 :** MCD (Modèle conceptuel de données)

**Figure 4 :** MPD (Modèle physique de données)

**Figure 5 :** page d’authentification

**Figure 6 :** page home de l’application

**Figure 7 :** page devis de l’application

**Figure 8 :** page creationdevis de l’application

**Figure 9** : schéma récapitulatif du déploiement.

# **SOMMAIRE**

[**REMERCIEMENTS** 5](#_Toc160014340)

[**AVANT PROPOS** 6](#_Toc160014341)

[**SIGLE ET ABREVIATION** 7](#_Toc160014342)

[**LISTE DES FIGUES** 8](#_Toc160014343)

[**SOMMAIRE** 9](#_Toc160014344)

[**INTRODUCTION** 10](#_Toc160014345)

[PREMIERE PARTIE : GENERALITE 11](#_Toc160014346)

[CHAPITRE 1 : PRESENTATION DU CADRE DE L’ETUDE 12](#_Toc160014347)

[DEUXIEME PARTIE : CONCEPTION 16](#_Toc160014350)

[CHAPITRE 2 **:** ÉTUDE CONCEPTUELLE 17](#_Toc160014351)

[TROISIENE PARTIE : REALISATION 30](#_Toc160014357)

[CHAPITRE 3 **:** REALISATION 31](#_Toc160014358)

[CHAPITRE 4 **:** DEPLOIEMENT 36](#_Toc160014361)

[CHAPITRE 5 **:** EVOLUTIONS ET PESPECTIVES 40](#_Toc160014365)

[CONCLUSION 41](#_Toc160014366)

# **INTRODUCTION**

Ces dernières années, le monde professionnel a subi une transformation rapide, largement due à la prolifération de l'informatique, une discipline axée sur le traitement automatique et logique des données. L'informatique est désormais omniprésente dans notre quotidien, introduisant de nouveaux concepts et des opportunités inédites.

Dans cet environnement en perpétuelle évolution, les entreprises sont confrontées à des défis de gestion de plus en plus complexes. Certaines d'entre elles ont dépassé le stade où les logiciels traditionnels, tels que Microsoft Excel, suffisaient à leurs besoins croissants. Elles se tournent donc vers des solutions logicielles plus sophistiquées capables de prendre en charge une multitude de paramètres, présentées comme la réponse à leurs nouveaux défis. C'est dans ce contexte que la gestion de client devient cruciale, servant d'outil essentiel pour améliorer continuellement la qualité des projets en réponse à une demande croissante de perfectionnement.

Face à ces enjeux, cette étude se concentrera sur la question suivante : comment la conception et la mise en œuvre d'une application web de gestion de client peuvent-elles constituer une solution efficace ?

Afin d'explorer en profondeur cette problématique et de comprendre comment la conception et la mise en œuvre d'une application web de gestion de projets peuvent offrir une solution efficace, notre analyse sera structurée en trois parties distinctes :

* Première parie : généralité
* Deuxième partie : conception
* Troisième partie : réalisation

# PREMIERE PARTIE : GENERALITE

## CHAPITRE 1 : PRESENTATION DU CADRE DE L’ETUDE

### STRUCTURE D’ACCUEIL

L’entreprise Créée en 1999, LASSIRE DECHETS SERVICES (LDS) est spécialisée dans la collecte, le ramassage, le recyclage et la valorisation des déchets domestiques et industriels ainsi que la collecte et traitement des filtres à huile. Le siège social et le garage sont situés au cœur d’Abidjan, dans la commune de Marcory. L’entreprise

travaille en collaboration avec le CIAPOL (CENTRE IVOIRIEN ANTIPOLLUTION) organisme agréé par la Côte d’Ivoire et nous procédons au ramassage des ordures selon leurs protocoles. Nous recevons toutes nos autorisations de mises en décharges par l’ANASUR (Agence Nationale de la Salubrité Urbaine).

La collecte et le ramassage des déchets publics. Au travers des marchés de la ville d’Abidjan, nous procédons à la collecte et au ramassage des déchets publics. A cet effet, nous disposons d’un type de véhicule pouvant effectuer le travail concerné. La collecte et le ramassage des déchets privés. A travers des contrats pour des usagers

privés, nous procédons à la collecte et au ramassage des déchets privés dans le district d’Abidjan en :

* DIND (déchets industriels non dangereux)
* DID (déchets industriels dangereux)

Nous opérons dans des domaines variés tel que les déchets de :

* L’agro-industrie
* La grande distribution
* De l’hôtellerie
* De la restauration des collectivités

### PROJET D’ETUDE

##### Problématique

L'entreprise de ramassage d'ordures et de déchets dispose d'une application desktop obsolète développée sous WinDev pour la gestion de ses clients, la création de devis, la transformation de ces devis en contrats, et la génération de bons d'enlèvements. Cette application a été efficace pendant des années, mais elle ne répond plus aux normes actuelles d'ergonomie, de modernité et d'accessibilité.

##### Objectifs du Projet

* Concevoir une nouvelle application web moderne et ergonomique.
* Assurer la transition des données de l'ancienne application vers la nouvelle.
* Ajouter des fonctionnalités améliorées pour optimiser la gestion des clients, des devis, des contrats et des bons d'enlèvements.
* Assurer la sécurité et l'accessibilité de l'application pour tous les utilisateurs.
* Amélioration de l'efficacité opérationnelle.
* Augmentation de la satisfaction cliente grâce à une meilleure gestion des contrats et des bons d'enlèvements.
* Réduction des coûts de maintenance de l'ancienne application.
* Réponse aux besoins actuels du marché en matière de convivialité et d'accessibilité.

##### Méthodologie de Travail

Pour atteindre les objectifs fixés et résoudre la problématique, nous avons suivi une méthodologie de travail rigoureuse qui a inclus plusieurs étapes essentielles :

* Analyse et Recherche

Dans cette première étape, nous avons effectué une analyse approfondie de l'application desktop existante pour identifier les problèmes et les lacunes. Cela comprenait l'examen de l'interface utilisateur, de la fonctionnalité, de la performance et de la sécurité. Nous avons également mené des recherches pour identifier les meilleures pratiques de développement web et les technologies modernes adaptées à ce projet.

* Conception de l'Application

Après avoir recueilli des informations grâce à l'analyse, nous avons élaboré une conception détaillée de la nouvelle application web. Cette conception a inclus la création de maquettes et de spécifications fonctionnelles pour définir l'interface utilisateur, les fonctionnalités et l'architecture globale. Nous avons également intégré les retours des parties prenantes et des utilisateurs pour m'assurer que la conception répondait à leurs besoins.

* Développement Front-End et Back-End

Le développement de la nouvelle application a été effectué en utilisant React.js pour la partie front-end et Flask (un framework Python) pour le back-end. Cette approche a été choisie pour ses avantages en termes de flexibilité, de performance et d'évolutivité. Nous avons écrit du code propre, optimisé et bien documenté pour assurer la robustesse de l'application. Nous avons également mis en place une base de données MySQL pour stocker les données de manière sécurisée

* Tests et Validation

Avant le déploiement, nous avons mené des tests rigoureux pour garantir que l'application fonctionnait correctement et répondait aux exigences fixées. Cela a inclus des tests d'unité, des tests d'intégration, des tests de performance et des

tests d'acceptation. Les retours des utilisateurs et des parties prenantes ont été pris en compte pour apporter des ajustements et des améliorations.

* Mise en Œuvre et Formation

Après la phase de tests, nous avons déployé la nouvelle application et nous avons assuré la migration des données de l'ancienne application vers la nouvelle. Nous avons également fourni une formation aux utilisateurs finaux pour s'assurer qu'ils maîtrisaient la nouvelle interface et les fonctionnalités.

* Suivi et Maintenance

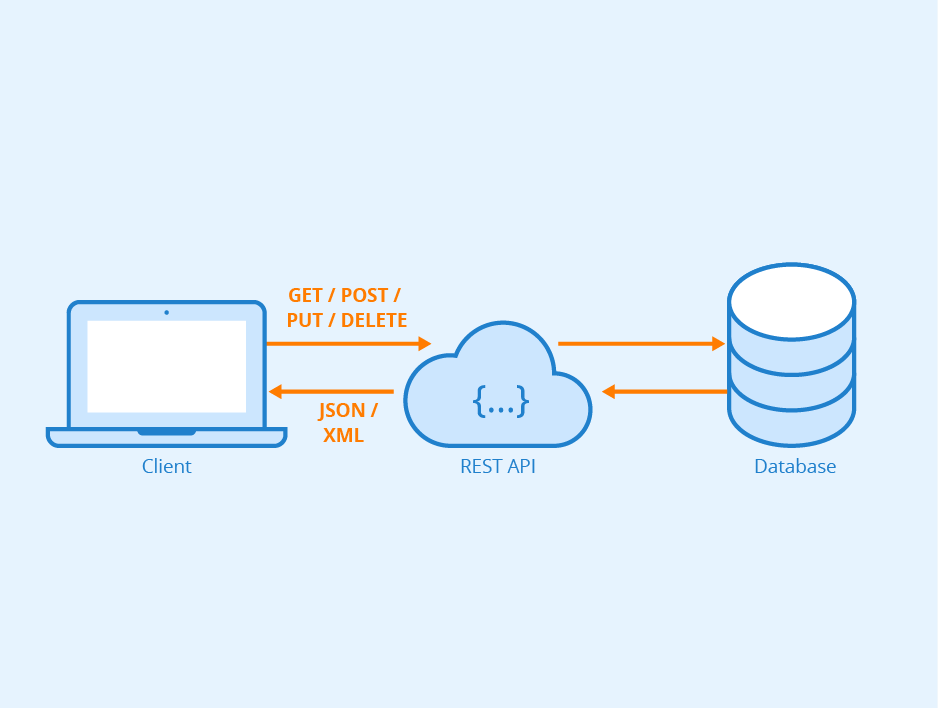
La mise en place de l'application n'est qu'une étape du projet. Nous avons établi un plan de suivi et de maintenance pour garantir que l'application reste

fonctionnelle, évolutive et sécurisée. Cela comprend la gestion des mises à jour, des correctifs de sécurité et des éventuelles améliorations futures.

# DEUXIEME PARTIE : CONCEPTION

## CHAPITRE 2 **:** ÉTUDE CONCEPTUELLE

### STRUCTURE DU PROJET



***Figure 1 :*** *Structure de l’application*

La conception de la structure du projet repose sur une approche modulaire, utilisant React.js pour le front-end et Flask pour le back-end, avec une base de données SQL. L'objectif est de créer une architecture robuste, permettant la communication sécurisée entre le front-end et le back-end via une API REST.

##### Front-End avec React.js

Le front-end de l'application est développé en utilisant React.js, une bibliothèque JavaScript moderne et performante. La structure du front-end est organisée de manière modulaire, avec des composants réutilisables, facilitant la maintenance et l'évolutivité de l'interface utilisateur.

* + **Composants Réactifs :** La conception modulaire du front-end permet l'utilisation de composants réactifs, facilitant la gestion des états et des interactions utilisateur. Cela garantit une expérience utilisateur fluide et dynamique.
  + **Gestion d'État :** Redux et Redux Toolkit sont intégrés pour la gestion centralisée de l'état de l'application, assurant une cohérence des données et des actions entre les différents composants.
  + **Styles avec CSS/SCSS :** Les styles sont gérés avec des feuilles de style en CSS/SCSS pour une présentation esthétique et une mise en page responsive.

##### Back-End avec Flask

Le back-end est construit avec Flask, un framework web léger pour Python. Il expose une API RESTful pour permettre au front-end d'accéder aux données de manière sécurisée.

* + **API RESTful :** Les fonctionnalités du back-end sont exposées via une API RESTful, utilisant des endpoints clairs pour accéder aux différentes ressources et opérations.
  + **Gestion des Routes :** Flask-Restful est utilisé pour la gestion des routes, facilitant la définition des points d'accès et des méthodes HTTP associées.
  + **Sécurité :** Les mécanismes de sécurité de Flask, tels que la protection CSRF et les bonnes pratiques en matière d'authentification, sont mis en œuvre pour garantir la sécurité des transactions entre le front-end et le back-end.

##### Base de Données SQL

Une base de données SQL, telle que MariaDB, est utilisée pour stocker de manière persistante les données de l'application.

* + **Schéma de Base de Données :** Le schéma de la base de données est conçu de manière à refléter de manière optimale la structure des données de l'application, assurant une gestion efficace et cohérente des informations.
  + **ORM avec SQLAlchemy :** SQLAlchemy est intégré comme ORM (Object- Relational Mapping) pour simplifier les interactions avec la base de données et garantir une abstraction efficace des requêtes SQL.

##### API REST pour Communication Front-Back

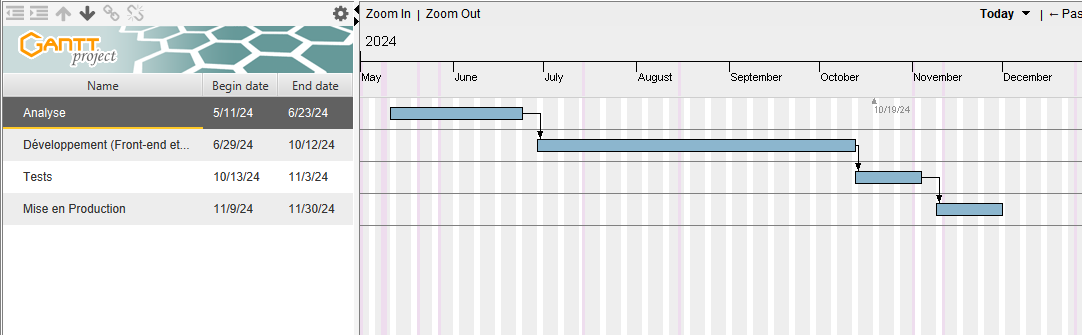
Pour permettre la communication entre le front-end et le back-end de manière sécurisée, une API REST est mise en place.

* + **Endpoints :** Des endpoints spécifiques sont définis pour chaque ressource, permettant au front-end d'effectuer des requêtes bien définies pour obtenir ou modifier des données.
  + **Authentification et Autorisation :** Des mécanismes d'authentification et

d'autorisation sont mis en œuvre pour garantir que seuls les utilisateurs autorisés peuvent accéder aux ressources et effectuer des opérations spécifiques.

En mettant en œuvre cette structure, le projet offre une séparation claire entre le front- end et le back-end, tout en assurant une communication fluide et sécurisée grâce à une API REST bien définie. La modularité, la gestion d'état centralisée, et la sécurité sont au cœur de la conception, garantissant ainsi la robustesse et l'évolutivité de l'application.

II.  PLANING DU PROJET METHODE DE MODELISATION



***Figure 2 :*** *Diagramme de Gant*

* + **Analyse :** cette phase va durer 2 semaines. Elle comprend la collecte des besoins, la rédaction de spécifications, la planification du projet, etc.
  + **Développement (Front-end et Back-end) :** cette phase va durer environ 6 semaines. Vous pouvez travailler sur le front-end et le back-end simultanément pour accélérer le développement.
  + **Tests :** cette phase va durer 2 semaines. Elle comprend la phase de test unitaire, de test d'intégration, de test de performance, etc.
  + **Mise en Production :** cette phase va durer 1 semaine pour assurer le déploiement de l'application et sa mise à disposition des utilisateurs.

### III. METHODE DE MODELISATION

###### Choix de Merise au lieu d'UML

Lors de la planification de ce projet de modernisation de l'application de gestion, le choix de la méthode de modélisation était un élément crucial pour la réussite du projet. Bien qu’UML soit une méthode de modélisation largement reconnue et utilisée, nous avons choisi d'opter pour Merise pour les raisons suivantes :

1. Adéquation aux Besoins du Projet

Merise est particulièrement adapte aux projets de développement de bases de données et de gestion de données. Dans notre cas, l’application de gestion impliquait la gestion des clients, des devis, des contrats et des bons d’enlèvements, tous basés sur une structure de données complexe. Merise offres des concepts et des outils spécifiques pour la modélisation des donnes, ce qui en faisait un choix naturel pour notre projet. La cohérence et la continuité du système préexistant ont exigé que nous adoptions également Merise pour maintenir l'harmonie et la facilité de transition au sein du projet. Cette contrainte a renforcé la décision d'utiliser Merise pour assurer la compatibilité et la continuité avec les structures existantes, garantissant ainsi une intégration harmonieuse et une évolutivité aisée du système.

##### Simplicités et Compréhension

Merise est réputé pour sa simplicité et sa facilité de compréhension. Cela a été particulièrement important dans le contexte de ce projet, car il impliquait plusieurs parties prenantes au sein de l'entreprise qui n'avaient pas nécessairement une expérience approfondie en matière de modélisation. La simplicité de Merise a permis une communication plus fluide et une meilleure compréhension des besoins et des spécifications du projet.

##### Focus sur la Gestion des Données

L'accent de Merise sur la gestion de données, y compris la modélisation des entités, des relations et des processus de l'entreprise, correspondait parfaitement à notre objectif de moderniser une application de gestion de clients, de devis, de contrats et de bons

d'enlèvements. Merise a fourni des outils pour une modélisation précise de ces données, ce qui était essentiel pour la refonte de l'application.

##### Historique des Réussite

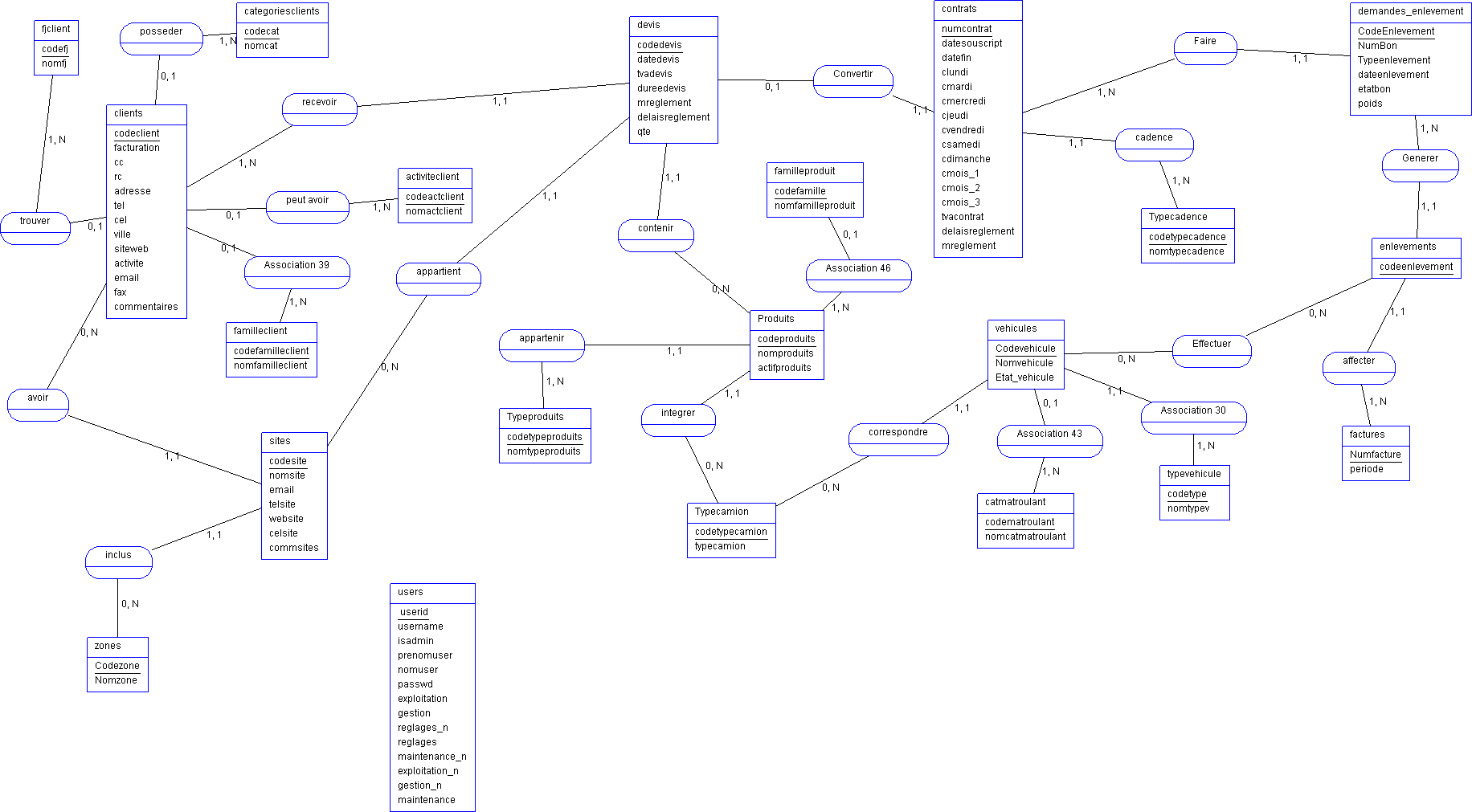
Merise a une longue histoire de succès dans le domaine de la gestion de données et des bases de données. De nombreux projets similaires ont été menés à bien en utilisant Merise comme méthode de modélisation. Cette fiabilité a renforcé notre confiance dans le choix de Merise pour notre projet.

En résumé, le choix de Merise au lieu d'UML pour ce projet de modernisation de l'application de gestion a été motivé par son adéquation aux besoins spécifiques du projet, sa simplicité, sa focalisation sur la gestion de données et son historique de réussite dans des projets similaires. Cette décision s'est avérée judicieuse et a contribué à la réussite globale du projet.

### MODELISATION DU SYSTEME

##### Modèle conceptuel des données

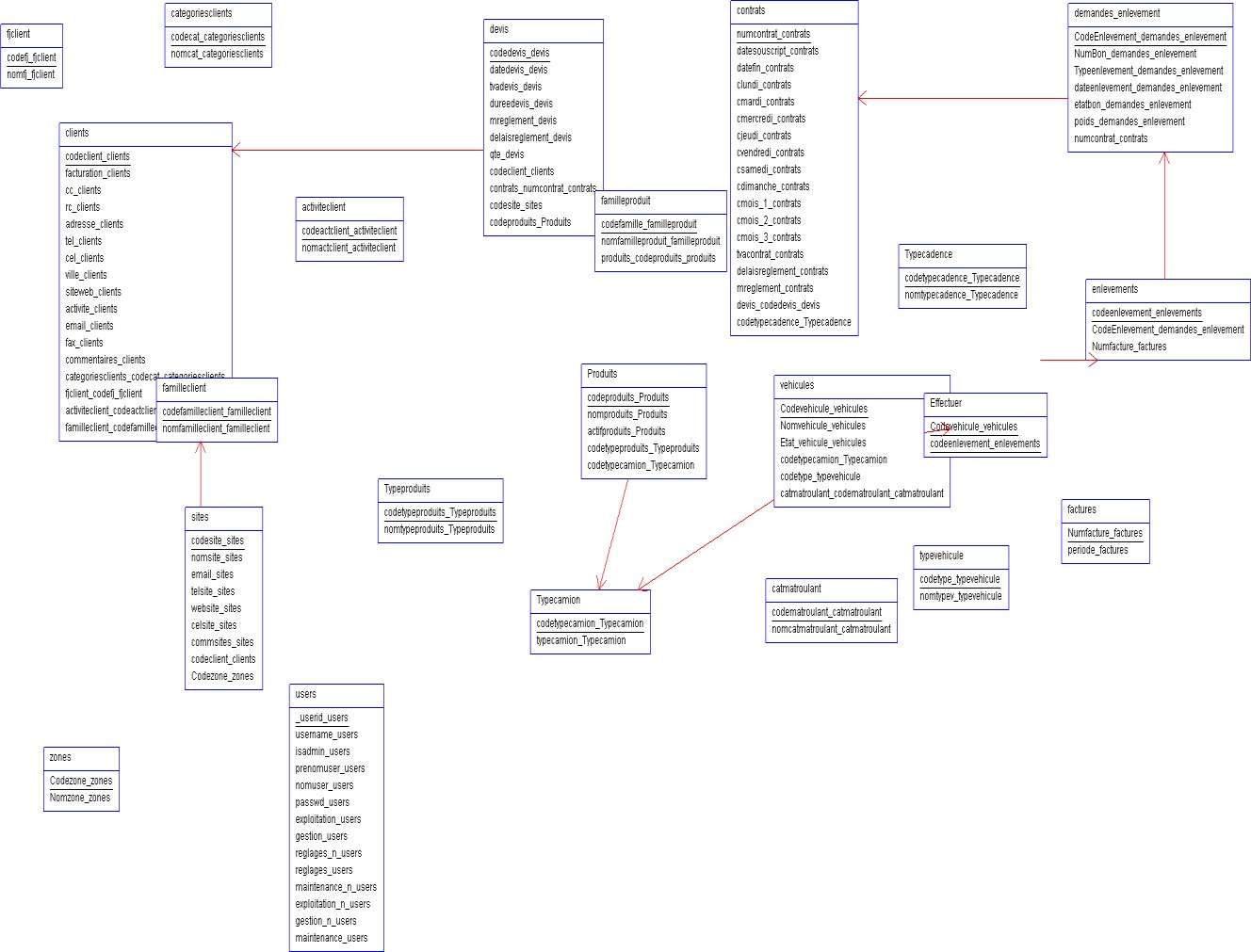
Le modèle conceptuel de données (MCD), a pour but d’exposer de façon formelle les données qui seront utilisées par le système informatique. Il s’agit donc d’une représentation des données facilement compréhensibles permettant de décrire le système d’informatique à l’aide d’entités et relation.



***Figure 3 :*** *MCD (Modèle conceptuel de données)*

##### Modèle physique des données

Le MDP représente la façon dont les données seront stockées physiquement dans la base de données. Il se situe après le MCD (Modèle Conceptuel de Données) qui représente les entités, les relations et les contraintes du système informatique, et avant le MLD (Modèle Logique de Données) qui se rapproche plus de la réalité physique en définissant les structures de données sans dépendre d’un système de gestion de bases de données spécifique.



***Figure 4 :*** *MPD (Modèle physique de données)*

##### Modèle logique de données relationnel

Le Modèle Logique de Données Relationnelles (MLDR) est une représentation intermédiaire entre le modèle conceptuel (MDC – Modèle Conceptuel de Données) et le modèle physique dans le processus de conception de bases de données.

Le MLDR décrit la structure des données de manière plus détaillée que le MCD, mais de manière plus abstraite que le modèle physique. Il se concentre sur la représentation des données à un niveau logique, en utilisant des entités, des attributs et des relations.

Contrairement au MCD qui utilise des concepts comme les entités, les relations et les cardinalités, le MDLR spécifie plus précisément les schémas de tables, les clés primaires et étrangères, ainsi que les contraintes d’intégrité.

**clients** (codeclient\_clients, facturation\_clients, cc\_clients, rc\_clients, adresse\_clients, tel\_clients, cel\_clients, ville\_clients, siteweb\_clients, activite\_clients, email\_clients,fax\_clients, commentaires\_clients, #categoriesclients\_codecat\_categoriesclients, #fjclient\_codefj\_fjclient, #activiteclient\_codeactclient\_activiteclient,#familleclient\_codefamilleclient\_familleclient)

**devis** (codedevis\_devis, datedevis\_devis, tvadevis\_devis, dureedevis\_devis, mreglement\_devis, delaisreglement\_devis, qte\_devis, #codeclient\_clients, #contrats\_numcontrat\_contrats, #codesite\_sites, #codeproduits\_Produits)

**contrats** (numcontrat\_contrats, datesouscript\_contrats, datefin\_contrats, clundi\_contrats, cmardi\_contrats, cmercredi\_contrats, cjeudi\_contrats, cvendredi\_contrats, csamedi\_contrats, cdimanche\_contrats, cmois\_1\_contrats, cmois\_2\_contrats, cmois\_3\_contrats, tvacontrat\_contrats, delaisreglement\_contrats, mreglement\_contrats, #devis\_codedevis\_devis, #codetypecadence\_Typecadence)

**sites** (codesite\_sites, nomsite\_sites, email\_sites, telsite\_sites, website\_sites, celsite\_sites, commsites\_sites, #codeclient\_clients, #Codezone\_zones)

**Produits** (codeproduits\_Produits, nomproduits\_Produits, actifproduits\_Produits, #codetypeproduits\_Typeproduits, #codetypecamion\_Typecamion)

**vehicules** (Codevehicule\_vehicules, Nomvehicule\_vehicules, Etat\_vehicule\_vehicules, #codetypecamion\_Typecamion, #codetype\_typevehicule,

#catmatroulant\_codematroulant\_catmatroulant)

**demandes\_enlevement** (CodeEnlevement\_demandes\_enlevement, NumBon\_demandes\_enlevement, Typeenlevement\_demandes\_enlevement, dateenlevement\_demandes\_enlevement, etatbon\_demandes\_enlevement, poids\_demandes\_enlevement, #numcontrat\_contrats)

**zones** (Codezone\_zones, Nomzone\_zones)

**Typeproduits** (codetypeproduits\_Typeproduits, nomtypeproduits\_Typeproduits)

**Typecamion** (codetypecamion\_Typecamion, typecamion\_Typecamion)

**enlevements** (codeenlevement\_enlevements, #CodeEnlevement\_demandes\_enlevement, #Numfacture\_factures)

**factures** (Numfacture\_factures, periode\_factures)

**Typecadence** (codetypecadence\_Typecadence, nomtypecadence\_Typecadence)

**typevehicule** (codetype\_typevehicule, nomtypev\_typevehicule)

**categoriesclients** (codecat\_categoriesclients, nomcat\_categoriesclients)

**fjclient** (codefj\_fjclient, nomfj\_fjclient)

**activiteclient** (codeactclient\_activiteclient, nomactclient\_activiteclient)

**familleclient** (codefamilleclient\_familleclient, nomfamilleclient\_familleclient)

**catmatroulant** (codematroulant\_catmatroulant, nomcatmatroulant\_catmatroulant)

**familleproduit** (codefamille\_familleproduit, nomfamilleproduit\_familleproduit, #produits\_codeproduits\_produits)

**users** (\_userid\_users, username\_users, isadmin\_users, prenomuser\_users, nomuser\_users, passwd\_users, exploitation\_users, gestion\_users, reglages\_n\_users, reglages\_users, maintenance\_n\_users, exploitation\_n\_users, gestion\_n\_users, maintenance\_users)

**Effectuer** (Codevehicule\_vehicules, codeenlevement\_enlevements)

### IMPLEMENTATION D’UNE API REST AVEC FLASK

Dans le cadre de la modernisation de notre application de gestion, nous avons adopté une approche orientée API pour garantir la flexibilité et l'extensibilité du système. Pour cela, nous avons choisi d'utiliser Flask, un framework web Python, pour la mise en place d'une API REST.

###### Utilisation de Flask pour l'API REST

Flask nous a offert un environnement flexible et robuste pour développer une API

RESTful. En utilisant les fonctionnalités de routage de Flask, nous avons défini plusieurs endpoints correspondant aux différentes fonctionnalités de notre application. Par exemple, nous avons créé des routes pour la gestion des clients, des devis, des

contrats, et d'autres entités clés de notre système.

###### Interaction avec la Base de Données

L'API REST développée avec Flask interagit étroitement avec les modèles de données que nous avons conçus, notamment le Modèle Logique de Données Relationnelles (MLDR). Chaque endpoint est lié à des opérations spécifiques sur la base de données, permettant la création, la lecture, la mise à jour et la suppression (CRUD) des entités

telles que les clients, les devis, et les contrats.

###### Avantages de l'Approche API REST

L'adoption d'une architecture orientée API via Flask a facilité la séparation des préoccupations et la scalabilité de notre application. Cette approche a permis une meilleure réutilisation du code et une communication efficace entre les différentes parties de notre système, favorisant ainsi la modularité et la maintenance aisée de l'application.

### SECURITE

Outre l'utilisation de JWT (JSON Web Tokens), il existe d'autres mécanismes et protocoles qui peuvent être utilisés pour renforcer la sécurité des API REST tels que :

* + **OAuth 2.0 :** Un protocole d'autorisation standard qui permet à une application

tierce d'obtenir l'accès à des ressources au nom d'un utilisateur. Il définit des flux d'autorisation tels que le flux d'autorisation implicite, le flux d'autorisation par mot de passe, etc.

* + **OAuth 2.0 avec OpenID Connect :** Une extension d'OAuth 2.0 qui ajoute une couche d'authentification. Il fournit un moyen sécurisé d'authentifier les

utilisateurs via des fournisseurs d'identité tiers.

* + **Basic Authentication :** Un mécanisme simple où les informations d'identification (nom d'utilisateur et mot de passe) sont incluses dans les en-têtes de la requête.

Bien que simple, il est recommandé d'utiliser HTTPS pour sécuriser la transmission des informations d'identification.

* + **API Keys :** Des clés API sont générées pour chaque client autorisé, et ces clés sont incluses dans les requêtes. C'est une méthode simple, mais elle nécessite la gestion sécurisée des clés.

Le choix de JSON Web Tokens (JWT) pour renforcer la sécurité de nos API REST repose sur des avantages significatifs qui répondent aux besoins spécifiques de notre application.

Les tokens JWT sont largement utilisés pour renforcer la sécurité des API REST en fournissant les avantages suivants :

* + **Authentification des Requêtes :** Les tokens JWT authentifient les requêtes du client en prouvant l'identité et les autorisations de l'utilisateur, renforçant ainsi la sécurité de l'accès aux ressources.
  + **Éviter l'Exposition des Identifiants :** Les informations d'authentification sont cryptées dans les tokens JWT, évitant ainsi l'exposition des identifiants sensibles lors des échanges entre le client et le serveur.
  + **Gestion des Accès et des Permissions :** Les tokens JWT contiennent des informations sur les permissions de l'utilisateur, permettant au serveur de prendre des décisions d'autorisation informées.
  + **Durée de Validité et Renouvellement :** La durée de validité des tokens JWT renforce la sécurité en imposant une réauthentification périodique, tandis que certains systèmes permettent le renouvellement pour une expérience utilisateur continue.

En résumé, les tokens JWT agissent comme des outils de sécurité robustes, assurant l'authenticité, l'autorisation et la confidentialité des échanges entre le client et le serveur au sein d'une API REST. Leur utilisation est particulièrement répandue en raison de leur simplicité, de leur efficacité et de leur compatibilité avec divers scénarios d'application.

# TROISIENE PARTIE : REALISATION

## CHAPITRE 3 **:** REALISATION

### OUTILS ET ENVIRONNEMENT DE TRAVAIL

Pour la réalisation de ce projet de modernisation de l'application de gestion, une gamme d'outils et d'environnements de travail ont été soigneusement sélectionnés pour répondre aux besoins spécifiques de développement front-end et back-end. Ces outils ont été choisis en fonction de leur adaptabilité, de leur efficacité et de leur adéquation aux exigences du projet.

##### Front-End

###### React.js:

Le développement du frontend de l'application a été réalisé en utilisant React.js, une bibliothèque JavaScript largement reconnue pour sa flexibilité et sa performance. React.js a été choisi en raison de sa capacité à créer des interfaces utilisateur dynamiques, ce qui était essentiel pour moderniser l'expérience cliente de l'application.

###### Redux et Redux Toolkit :

Pour gérer l'état de l'application de manière efficace, Redux a été intégré au projet. Redux Toolkit a été utilisé pour simplifier la configuration de Redux en

fournissant une structure claire pour la gestion des actions et des reducers. Cette combinaison a permis de gérer efficacement les données au sein de l'application.

* + CSS et SCSS **** **:**

Le stylage de l'interface utilisateur a été effectué en utilisant CSS (Cascading

Style Sheets) pour garantir une présentation cohérente. SCSS (Sassy CSS) a été employé pour optimiser la gestion des styles en permettant la réutilisation de code et la structuration des feuilles de style.

##### Back-End

###### Flask:

Le développement du backend a été réalisé avec Flask, un Framework Python léger et flexible. Flask s'est avéré être un choix judicieux pour la création d'API REST, fournissant une structure claire pour la gestion des routes, des

****contrôleurs, et des réponses HTTP.

* + MySQL **:**

Le système de gestion de bases de données MySQL a été sélectionné pour stocker les données relatives aux clients, devis, contrats, bons d'enlèvements, et autres informations cruciales. La robustesse et la facilité d'utilisation de MySQL étaient particulièrement adaptées aux besoins du projet.

##### Environnement de Développement

###### Visual Studio Code (VS Code):

En tant qu'environnement de développement principal, Visual Studio Code a

offert une interface conviviale et des fonctionnalités essentielles. Les extensions telles que "ESLint" et "Prettier" ont permis de garantir la qualité du code et

d'améliorer la productivité du développement.

****

* + Git **:**

Git a été utilisé comme système de gestion de versions pour suivre l'évolution du code source, faciliter la collaboration entre les membres de l'équipe et permettre le suivi des modifications apportées au projet.

* + Slack**** **:**

Slack a été la plate-forme de communication et de collaboration en équipe,

favorisant la communication instantanée, la gestion des tâches et la coordination au sein de l'équipe de développement.

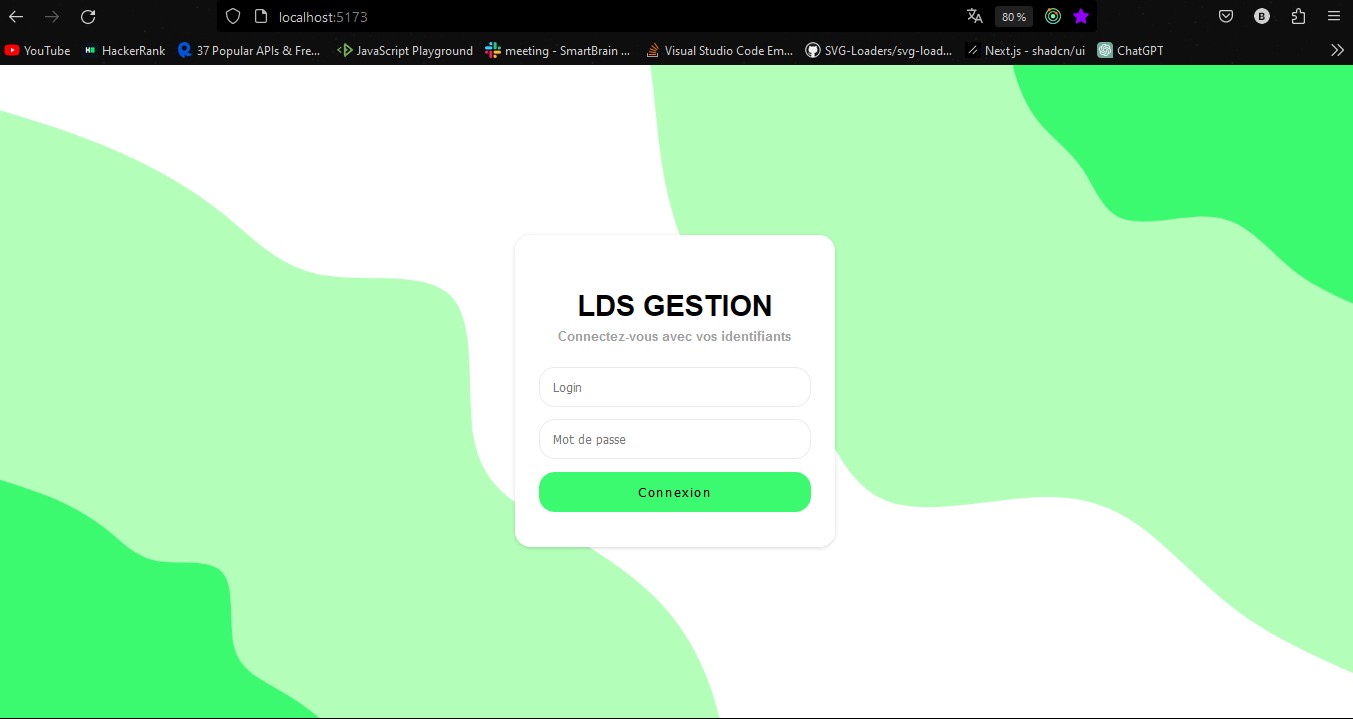
### PRESENTATION DE L’APPLICATION

##### Accès au site

Notre application a été déployée en local c’est-à-dire sur le même matériel d’environnement. A cet effet, pour y avoir accès nous entrerons dans la barre d’un navigateur http://localhost:5173/.

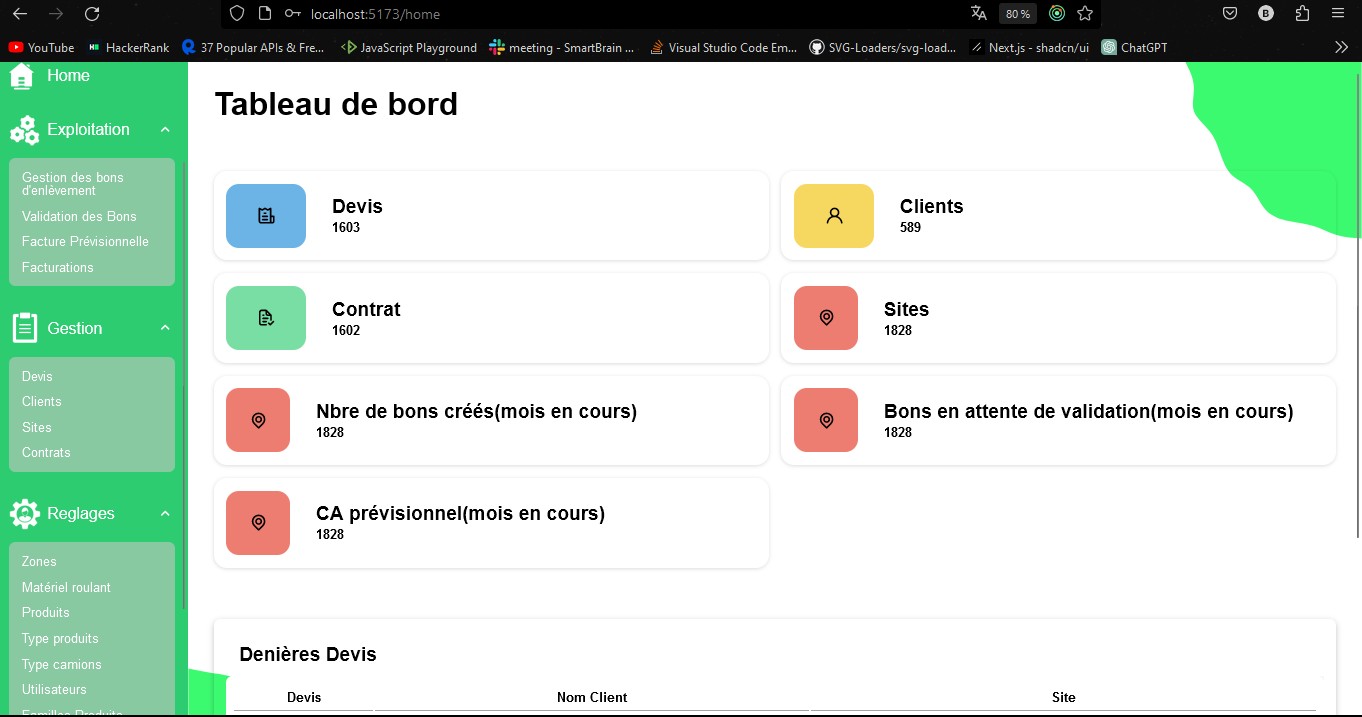
React js possède lui-même son serveur de développement local.

##### Gestion d’authentification et tableau de bord



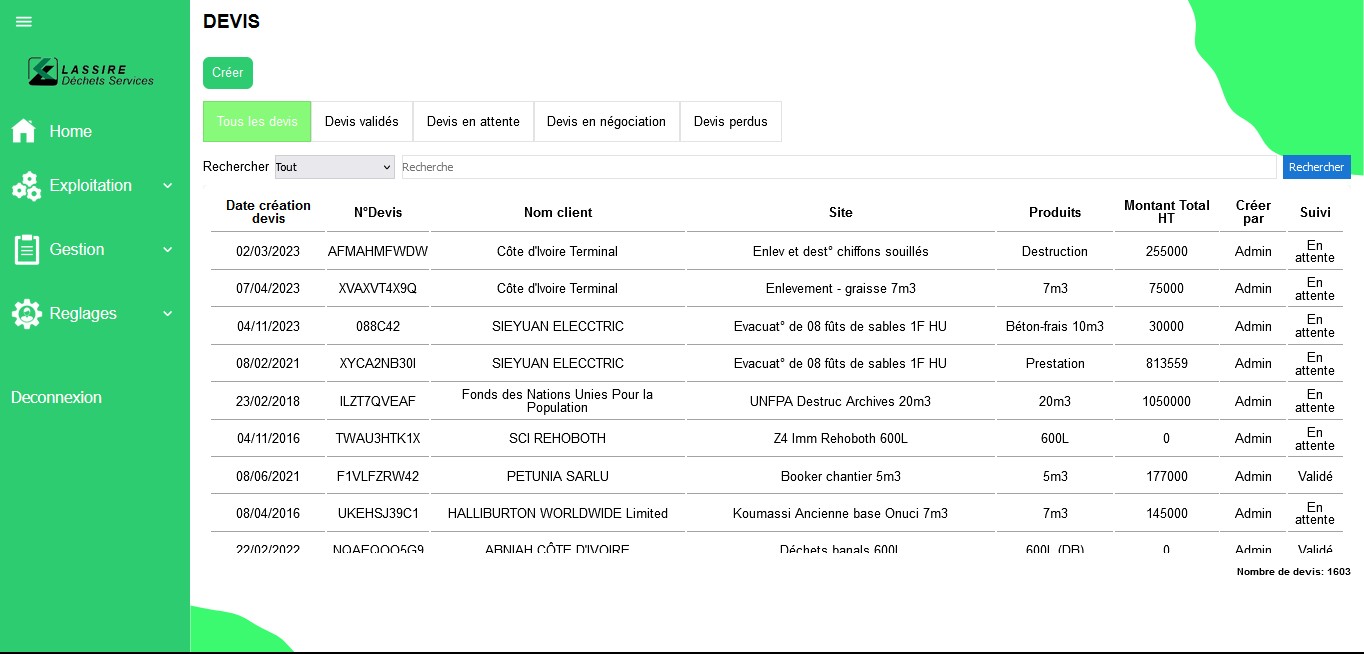
***Figure 5 :*** *page d’authentification*

Cette page est le portail d’accès au Dashboard de l’application.



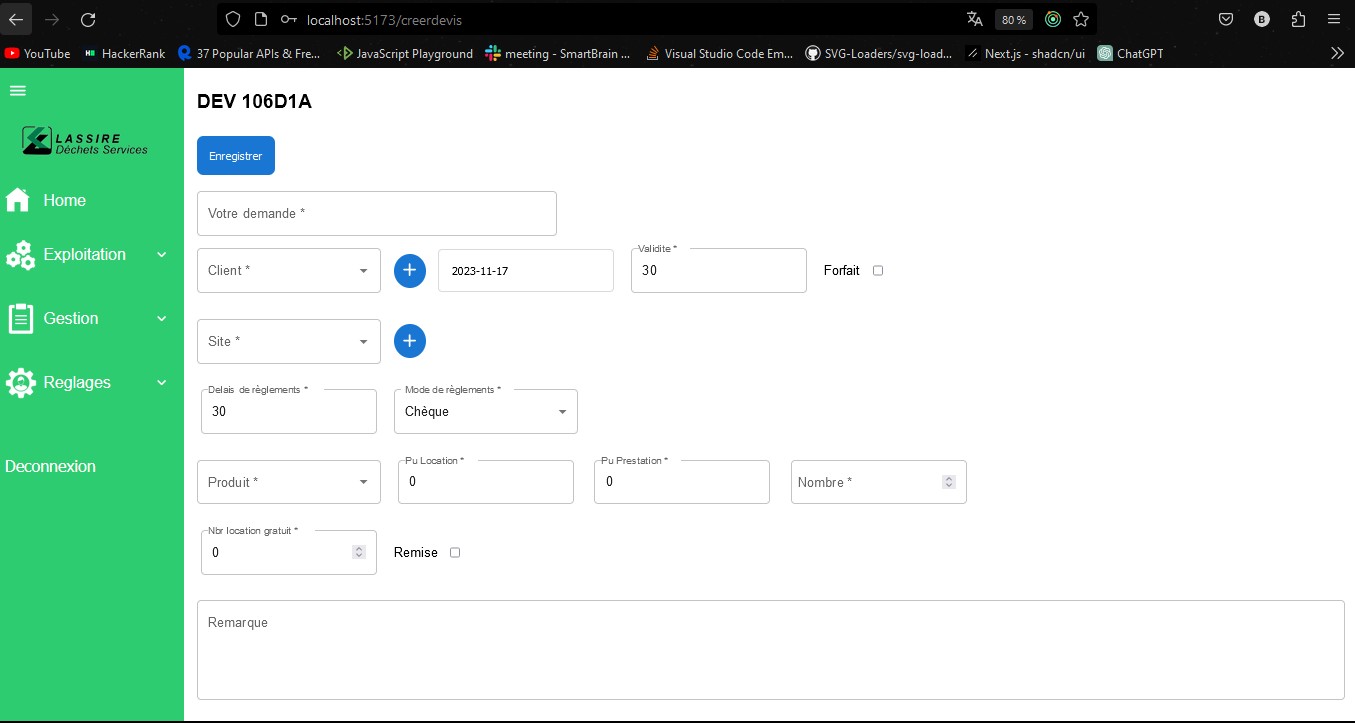
***Figure 6 :*** *page home de l’application*

Cette page est la page Home de l’application ici vous avez accès a toutes les fonctionnalités de l’application.



***Figure 7 :*** *page devis de l’application*

Cette page est la page *devis* de l’application ici vous avez accès à la liste de tous vos devis.



***Figure 8 :*** *page* creationdevis *de l’application*

Cette page est la page creationdevis de l’application ici vous pouvez créer des devis pour vos différents clients.

## CHAPITRE 4 **:** DEPLOIEMENT

La stratégie de déploiement de l'application repose sur une infrastructure organisée en cinq serveurs distincts, chacun ayant un rôle spécifique, et sur l'utilisation de

fonctionnalités de sécurité telles que des switches et des pares-feux pour gérer les communications et sécuriser les flux d'informations.

### REPARTITION DES SERVEURS

* + **Serveur Front-End :** Dédié à l'hébergement de l'interface utilisateur React.js, ce serveur fournit la plateforme visuelle de l'application.
  + **Serveur Back-End :** Hébergeant l'application Flask, ce serveur assure la logique métier et les interactions avec la base de données MariaDB.
  + **Serveur Base de Données :** Ce serveur gère l'ensemble des données stockées dans MariaDB, notamment les informations client, les devis, et les contrats.
  + **Serveur DNS :** Utilisé pour la résolution des noms de domaine en adresses IP, ce serveur permet l'accès à l'application par le biais de noms de domaine spécifiques.
  + **Serveur Proxy :** Nginx est utilisé comme serveur proxy pour servir

d'intermédiaire sécurisé entre les utilisateurs et le front-end, contrôlant l'accès à l'application.

### CONTROLE ET SECURITE DES FLUX D'INFORMATIONS

La gestion efficace des flux d'informations entre les serveurs est cruciale pour assurer la sécurité et la confidentialité des données échangées au sein de l'infrastructure. Pour cela, deux principaux éléments sont mis en place : l'utilisation de switches pour la communication entre les serveurs et la configuration de pare-feu pour chaque serveur.

###### Switches pour la Communication Inter-Serveurs :

Les switches jouent un rôle central dans l'acheminement et la régulation des communications entre les serveurs au sein de l'infrastructure. Chaque serveur est connecté à un switch central qui agit comme un hub pour le trafic de données. Les switches permettent de contrôler précisément la transmission des données entre les serveurs en acheminant les informations uniquement vers les destinations autorisées, limitant ainsi l'accès aux flux d'informations nécessaires à chaque interaction. Cette approche garantit que seuls les serveurs autorisés peuvent échanger des données, renforçant ainsi la sécurité globale du réseau.

###### Gestion des Pare-feu :

Des pares-feux sont configurés individuellement pour chaque serveur de l'infrastructure. Ces pares-feux agissent comme des barrières de sécurité en filtrant et en contrôlant strictement le trafic entrant et sortant de chaque serveur. Voici comment ils fonctionnent

:

* + **Filtrage des Flux Entrants :** Les pare-feu analysent et filtrent le trafic entrant, permettant uniquement les données conformes aux règles de sécurité établies à passer à travers les ports spécifiques autorisés. Cela empêche les accès non autorisés et les attaques potentielles sur le réseau.
  + **Contrôle des Flux Sortants :** De la même manière, les pares-feux surveillent et gèrent les données sortantes, s'assurant que seuls les flux d'informations

autorisés quittent le serveur. Cela limite la fuite de données sensibles ou la transmission involontaire d'informations non sécurisées.

* + **Gestion des Ports :** Les pare-feu permettent également de réguler strictement les ports utilisés pour les communications. Seuls les ports nécessaires au

fonctionnement des services spécifiques sont ouverts, réduisant ainsi les risques potentiels liés à l'ouverture de ports inutiles ou non sécurisés.

En combinant l'utilisation de switches pour contrôler les flux de données entre les serveurs et la mise en place de pare-feu pour chaque serveur individuel, l'infrastructure garantit une gestion sécurisée et contrôlée des échanges d'informations, réduisant ainsi les risques liés à d'éventuelles failles de sécurité ou à des accès non autorisés.

### SECURISATION DE L'ACCES UTILISATEUR

L'accès à l'application est sécurisé grâce à un processus de vérification des utilisateurs effectué par le serveur proxy. Avant de permettre l'accès au front-end de l'application, le serveur proxy met en place des mesures de contrôle d'accès pour garantir l'authenticité des demandes d'accès et la légitimité des utilisateurs.

###### Contrôle d'Accès Via Proxy :

Lorsqu'un utilisateur souhaite accéder à l'application, sa demande est d'abord dirigée vers le serveur proxy. Ce dernier agit comme un intermédiaire entre l'utilisateur et le

front-end de l'application. Avant de transférer la demande vers le front-end, le serveur proxy effectue plusieurs vérifications :

* + **Authentification de la Demande :** Le serveur proxy examine chaque demande d'accès entrante pour s'assurer qu'elle est conforme aux normes de sécurité

établies. Il vérifie si la demande est correctement formée et ne présente aucun signe de manipulation ou d'accès non autorisé.

* + **Validation via le Serveur DNS :** Le serveur proxy interroge ensuite le serveur DNS pour vérifier la validité du nom de domaine ou de l'adresse URL à laquelle l'utilisateur tente d'accéder. Cette étape vise à confirmer que l'adresse demandée est enregistrée et autorisée à être accessible.
  + **Vérification de l'Autorisation :** Une fois que le serveur proxy a validé la demande auprès du serveur DNS, il procède à une vérification supplémentaire pour confirmer que l'utilisateur est autorisé à accéder à cette partie spécifique de l'application.
  + **Filtrage des Requêtes Non Autorisées :** Si la demande échoue à l'une des étapes de vérification, le serveur proxy bloque l'accès et empêche la demande

d'atteindre le front-end. Cela garantit que seuls les utilisateurs autorisés peuvent accéder à l'application.

En consolidant ces étapes de vérification, le serveur proxy s'assure que seuls les utilisateurs légitimes, dont l'accès a été préalablement autorisé et validé, sont en

mesure d'atteindre et d'interagir avec le front-end de l'application. Cette stratégie de contrôle d'accès renforce la sécurité en établissant des barrières de vérification avant l'accès aux fonctionnalités de l'application, réduisant ainsi le risque d'accès non autorisé ou de manipulations malveillantes.

## CHAPITRE 5 **:** EVOLUTIONS ET PESPECTIVES

Apres le déploiement réussi de l’application modernisée, une phase cruciale an été celle de la collecte des retours et des feedbacks des utilisateurs. Ces retours ont été essentiels pour l’évaluer l’impact de l’application et orienter son évolution future. Nous avons mis en place des mécanismes de collecte de retours utilisateurs, comprenant des sondages, des sessions de feedback direct et des analyses de l’utilisation. Ces retours ont été analyses de manière approfondie, et plusieurs observations ont été relevées :

* + **Améliorations Ergonomiques** : Les utilisateurs ont exprimé une satisfaction concernant la nouvelle interface utilisateur, soulignant sa convivialité. Cependant, des suggestions pour des améliorations ergonomiques mineures ont été identifiées, notamment la simplification de certaines étapes du processus.
  + **Améliorations Fonctionnelles** : Plusieurs demande de fonctionnalités ont été recueillies.

Dans un futur proche nous penserons aussi à implanter des fonctionnalités comme :

###### **La géolocalisation des bacs et des coffres**

* + **La géolocalisation de camions**
  + **La gestion du garage**

# CONCLUSION

Le projet de modernisation de l’application de gestion des déchets a été une expérience enrichissante et formatrice. La transition vers une application web moderne a été réalisée avec succès, offrant à l’entreprise une plate-forme plus ergonomique et efficiente pour gérer ses opérations de ramassage des déchets.

Au cours de cette période, plusieurs jalons significatifs ont été atteints. Le déploiement réussi de l’application modernisée a permis de simplifier les processus de gestion des clients, des devis, des contrats et des bons. L’adoption de technologies telles que React js, Redux, Flask et MySql a permis de créer une interface utilisateur intuitive tout en garantissant une gestion robuste des données. Les défis rencontres, qu’il s’agisse de l’optimisation des performances ou de la prise en compte des retours utilisateurs, ont été abordés avec rigueur et créativité.

Ces obstacles ont été des opportunités pour acquérir de nouvelles compétences et renforcer notre expertise en développement web et en gestion de projet. Pour l’avenir, je suis convaincu que cette application modernisée continuera à jouer un rôle vital dans l’efficacité opérationnelle de l’entreprise. Les perspectives d’amélioration identifiées, telles que l’intégration avec d’autres systèmes et le développement de fonctionnalités avancées, ouvrent la voie à de nouvelles étapes pour optimiser davantage le processus.

Nous tenons à exprimer ma profonde gratitude envers toute l’équipe pour sa collaboration, son soutient et son encouragement tout au long de ce stage. Leur expertise et leur engagement ont été une source d’inspiration et ont grandement contribue à cette expérience enrichissante.

# BIBLIOGRAPHIE

* + - MEMOIRE CONSULTE
  1. Toure Lacina Junior. Conception et réalisation d’une application web de demandede documents : cas de la mairie de Treichville, Mémoire de fin de cycle pour

L’obtention de la licence en systèmes réseaux informatiques et télécommunications(SRIT), ESATIC, 2022, 55 pages.

* + - ***SITES CONSULTES***
  1. [https://chat.openai.com/](https://laravel.com/) Consulté le 01/10/2023 à 08H33
  2. [https://sweetalert2.github.io](https://sweetalert2.github.io/) Consulté le 18/10/2023 à 13H42
  3. <https://www.pexels.com/fr-fr/>Consulté le 09/11/2023 à 09H17

# TABLE DES MATIERES

[**DÉDICACE** 4](#_Toc160014339)

[**REMERCIEMENTS** 5](#_Toc160014340)

[**AVANT PROPOS** 6](#_Toc160014341)

[**SIGLE ET ABREVIATION** 7](#_Toc160014342)

[**LISTE DES FIGUES** 8](#_Toc160014343)

[**SOMMAIRE** 9](#_Toc160014344)

[**INTRODUCTION** 10](#_Toc160014345)

[PREMIERE PARTIE : GENERALITE 11](#_Toc160014346)

[CHAPITRE 1 : PRESENTATION DU CADRE DE L’ETUDE 12](#_Toc160014347)

[I. STRUCTURE D’ACCUEIL 12](#_Toc160014348)

[II. PROJET D’ETUDE 13](#_Toc160014349)

[DEUXIEME PARTIE : CONCEPTION 16](#_Toc160014350)

[CHAPITRE 2 **:** ÉTUDE CONCEPTUELLE 17](#_Toc160014351)

[I. STRUCTURE DU PROJET 17](#_Toc160014352)

[III. METHODE DE MODELISATION 21](#_Toc160014353)

[IV. MODELISATION DU SYSTEME 22](#_Toc160014354)

[IV. IMPLEMENTATION D’UNE API REST AVEC FLASK 27](#_Toc160014355)

[V. SECURITE 28](#_Toc160014356)

[TROISIENE PARTIE : REALISATION 30](#_Toc160014357)

[CHAPITRE 3 **:** REALISATION 31](#_Toc160014358)

[I. OUTILS ET ENVIRONNEMENT DE TRAVAIL 31](#_Toc160014359)

[II. PRESENTATION DE L’APPLICATION 33](#_Toc160014360)

[CHAPITRE 4 **:** DEPLOIEMENT 36](#_Toc160014361)

[I. REPARTITION DES SERVEURS 36](#_Toc160014362)

[II. CONTROLE ET SECURITE DES FLUX D'INFORMATIONS 36](#_Toc160014363)

[III. SECURISATION DE L'ACCES UTILISATEUR 38](#_Toc160014364)

[CHAPITRE 5 **:** EVOLUTIONS ET PESPECTIVES 40](#_Toc160014365)

[CONCLUSION 41](#_Toc160014366)

[BIBLIOGRAPHIE 42](#_Toc160014367)

[TABLE DES MATIERES 43](#_Toc160014368)

[RESUMER 44](#_Toc160014369)

[SUMMARY 44](#_Toc160014370)

# RESUMER

Ce mémoire examine l'impact grandissant de la digitalisation des projets informatiques dans un environnement professionnel en perpétuelle évolution. Les entreprises, confrontées à des défis de gestion complexes, choisissent des solutions spécialisées telles que les applications web de gestion. L'étude met en lumière l'importance cruciale de la gestion en temps réel pour favoriser l'amélioration continue. En détaillant la structure d'accueil, en réalisant une analyse conceptuelle approfondie avec la modélisation du système, et en mettant en œuvre concrètement le projet, ce mémoire apporte une contribution significative à la compréhension de l'efficacité de ces solutions dans un contexte dynamique.

# SUMMARY

This thesis examines the growing impact of the digitization of IT projects in an ever-changing business environment. Faced with complex management challenges, companies are opting for specialized solutions such as web-based project management applications. The study highlights the crucial importance of real-time management in driving continuous improvement. By detailing the host structure, carrying out an in-depth conceptual analysis with system modeling, and concretely implementing the project, this dissertation makes a significant contribution to understanding the effectiveness of these solutions in a dynamic context.