



كلية العلوم والتكنولوجيات بطنجة
+٥٢٤٣٦٠٩١ | +٣٥٥٥٠٩١ | +٣٥٤٣٤٧٤١ | EoT
Faculté des Sciences et Techniques de Tanger

FACULTÉ DES SCIENCES ET TECHNIQUES DE TANGER

RAPPORT DE PROJET DE FIN SEMESTRE DI LOGICIELS ET SYSTÈMES INTELLIGENTS

Assurance Véhicule ZOHS

Auteur:

Ziad BEN SAADA
Ouail LAAMIRI
Hajar SADIK
Samir ZIANI

Encadré par:
Mr. Lotfi EL ACHACK

Dec 2023 - Jan 2024

Contents

1	Introduction	6
1.1	Introduction :	6
1.2	Composition de l'équipe	6
1.3	Équipes divisées	7
1.3.1	Sadik Hajar avec Laamiri Ouail	7
1.3.2	Ben Saada Ziad avec Ziani Samir	7
1.4	Objectifs du Projet	7
1.5	Fonctionnalités Clés	7
1.6	Conclusion	8
2	Fondements Théoriques :	9
2.1	Introduction à l'Architecture Microservices	9
2.1.1	Principes Fondamentaux	9
2.1.2	Avantages Stratégiques	9
2.1.3	Défis et Solutions	10
2.1.4	Conclusion	10
2.2	Introduction à PWA, SSR et SPA : Transformant l'Expérience Utilisateur	10
2.2.1	Progressive Web Apps (PWA)	10
2.2.2	Rendu Côté Serveur (SSR)	11
2.2.3	Applications à Page Unique (SPA)	11
2.2.4	Conclusion	11
2.3	Introduction aux Containers, Docker,Jenkins,kafka et Kubernetes (K8S), : Révolutionner le Déploiement et l'Orchestration des Applications	12
2.3.1	Containers : Isolation Légère, Portabilité et Rapidité	12
2.3.2	Docker : Révolution dans la Conteneurisation	12
2.3.3	Kubernetes (K8S) : Orchestration d'Échelle et de Complexité	12
2.3.4	Intégration avec Kafka	13
2.3.5	Intégration avec Jenkins	13

2.3.6	Fondements Conceptuels	13
2.3.7	Conclusion Anticipée	13
2.4	La recommandation système et l'apprentissage machine	14
2.4.1	Filtrage collaboratif	14
2.4.2	Filtrage basé sur le contenu	14
2.4.3	Modèles probabilistes et statistiques	14
2.4.4	Évaluation des performances	15
2.4.5	Traitement du biais et de la froideur	15
2.4.6	Apprentissage en ligne et mise à l'échelle	15
2.4.7	Conclusion Anticipée	15
2.5	La technologie blockchain et les applications décentralisées	15
2.5.1	Blockchain	15
2.5.2	Contrats intelligents (Smart Contracts)	16
2.5.3	Décentralisation	16
2.5.4	Consensus	16
2.5.5	Ethereum et les plateformes de DApp	16
2.5.6	Conclusion Anticipée	17
3	Developpement web	18
3.1	Back-End : Micro Services	18
3.1.1	client-service	18
3.1.2	login-service	19
3.1.3	assurance-service	19
3.1.4	vehicule-service	19
3.1.5	contract-service	19
3.1.6	DriverLicense-service	19
3.1.7	Email-service	19
3.1.8	File-service	20
3.1.9	Link-service	20
3.1.10	service-discovery	20
3.1.11	server-getway	20
3.2	Front End:Capture d'ecan	20
4	La Systeme de recommandation d'assurance	26
4.1	Regroupement des données d'assurance	26
4.1.1	Nettoyage des données	27

4.1.2	Construction du modèle pour la recommandation	27
4.2	Classification Supervisée après Clustering	28
4.2.1	Traitement des Données Déséquilibrées : UNBALANSED DATA	28
4.2.2	Division des Données	29
4.2.3	Modèle de Stacking	30
4.2.4	Évaluation du Modèle	30
4.3	architecturer le projet en utilisant Mlops	32
4.3.1	Integration kafka	32
4.3.2	Pipeline Ml utilisant Kubeflow	32
4.3.3	Integration Airflow	36
5	Le Deveoppment BlockChain	38
5.1	Technologies utilisées :	38
5.1.1	Solidity :	38
5.1.2	Smart Contract :	38
5.1.3	Truffle:	38
5.1.4	Sepolia Network :	39
5.1.5	Rinkeby :	39
5.1.6	Etherscan :	39
5.1.7	Ganache:	39
5.1.8	Geth:	39
5.1.9	Metamask:	39
5.1.10	Express.js :	40
5.2	Clone the repository:	40
5.3	Accédez au dossier du projet:	40
5.4	Install dependencies:	40
5.5	Smart Contract Development:	40
5.5.1	Install Truffle globally (if not installed):	40
5.6	Ganache	40
5.6.1	Install Ganache (The local BlockChain)	40
5.7	Express-box	41
5.7.1	Téléchargez la boîte. Cela se charge également d'installer les dépendances nécessaires.	41
5.8	Truffle Development	41
5.8.1	Démarrez la console de développement Truffle en utilisant	41
5.9	Deployment	41

5.9.1	Déployez vos contrats intelligents sur un réseau spécifique (remplacez network.name par le réseau souhaité. Allez dans truffleconfig.js, décommentez le réseau et choisissez le nom du réseau)	41
5.10	Install Metamask	41
5.10.1	Go to https://metamask.io/download/ to install	41
5.10.2	Créez un compte dans Metamask	42
5.11	Run Express Server	43
5.11.1	Pour exécuter le serveur Express	43
5.12	Install Geth	43
5.12.1	Geth –version	43
5.12.2	Créez un nouveau compte	43
5.12.3	Exécutez Geth en mode développement	44
5.12.4	Exécutez Geth console JavaScript	44
5.12.5	Configurer le nœud Ethereum Sepolia	45
5.12.6	Connectez-vous au réseau Sepolia	45
5.12.7	Téléchargez le client de consensus	45
5.12.8	Créez un compte dans Sepolia	46
5.12.9	Téléchargez le bloc 1 de Sepolia	46
5.12.10	Assurez-vous de modifier "adresse du compte de Sepolia"	46
5.12.11	Configurer Rinkeby	46
5.13	Créez un compte sur Infura	47
5.13.1	Créez une clé API	47
5.13.2	Exécutez truffle migrate –network sepolia	47
5.13.3	rechercher l'adresse du contrat sur sepolia.etherscan	48
6	DevOps	49
6.1	Méthodologie DevOps	49
6.1.1	Explication du concept DevOps	49
6.1.2	Importance de DevOps dans le cycle de développement logiciel	50
6.1.3	Objectifs spécifiques de l'implémentation de pratiques DevOps dans le projet ZOHS	50
6.2	Outils et Technologies	51
6.2.1	Git	51
6.2.2	GitHub	51
6.2.3	Docker :	51
6.2.4	Kubernetes :	51
6.2.5	Kafka :	51

6.2.6 Jenkins :	51
6.3 Implémentation DevOps dans le projet	52
6.3.1 Méthodologies utilisées pour intégrer les pratiques DevOps dans le cycle de développement	52
7 Reference	54
7.1 Github Repository	54
7.2 DevOps	54
7.3 Developpment Web	55
7.4 Machine Learning	56
7.5 Blockchain	56

Introduction

1.1 Introduction :

Dans le contexte dynamique de l'industrie de l'assurance automobile, notre équipe s'est engagée dans un projet ambitieux visant à révolutionner l'expérience client tout en optimisant les processus internes. Sous la direction éclairée du Dr. Lotfi El Achack, notre équipe multidisciplinaire s'est rassemblée pour concevoir et développer une Progressive Web App (PWA) basée sur une architecture microservices.

Ce projet, au cœur de notre mission, vise à créer une plateforme moderne et réactive qui redéfinira les interactions entre les assureurs, les assurés et les intervenants dans le domaine de l'assurance automobile. Notre approche tire parti des dernières avancées technologiques, unifiant les compétences de nos experts en développement full stack, DevOps, blockchain et machine learning.

1.2 Composition de l'équipe

- *Ouail Laamiri* - Chef de projet et développeur full stack
- *Hajar Sadik* - Expert DevOps
- *Ziad Ben Saada* - Spécialiste Blockchain
- *Samir Ziani* - Professionnel du Machine Learning pour le développement d'un système de recommandation

Chaque membre de notre équipe apporte une expertise distinctive, créant une synergie essentielle pour atteindre les objectifs ambitieux définis pour ce projet novateur.

1.3 Équipes divisées

Après une évaluation approfondie de nos compétences et de nos objectifs, nous avons choisi de diviser notre équipe en deux groupes distincts pour maximiser notre efficacité et notre collaboration.

1.3.1 Sadik Hajar avec Laamiri Ouail

Dans cette équipe, Hajar et Ouail combinent leurs compétences en développement et en gestion de projet pour créer une dynamique de travail fluide. Leur collaboration étroite garantit une communication efficace et une résolution rapide des problèmes techniques.

1.3.2 Ben Saada Ziad avec Ziani Samir

Ziad et Samir forment une autre équipe où ils mettent en valeur leurs compétences respectives en blockchain et en machine learning. Leur collaboration promet d'apporter des solutions innovantes et efficaces à notre projet.

1.4 Objectifs du Projet

Le cœur de notre initiative réside dans la création d'une PWA intuitive et robuste, offrant une expérience utilisateur immersive et transparente. Cette application évolutive sera soutenue par une architecture microservices, permettant une adaptabilité continue aux évolutions du marché et des besoins de nos utilisateurs.

1.5 Fonctionnalités Clés

- **Système de Recommandation (ML)** : Samir Ziani dirige l'intégration d'un système de recommandation basé sur le machine learning, personnalisant les offres d'assurance en fonction des comportements et des besoins des utilisateurs.
- **Blockchain pour la Sécurité et la Transparence** : Ziad Ben Saada met en œuvre une infrastructure blockchain pour garantir la sécurité des données et la transparence des transactions, renforçant ainsi la confiance entre les parties prenantes.
- **Infrastructure et DevOps Optimisés** : Hajar Sadik supervise le développement et la gestion de l'infrastructure, assurant une disponibilité élevée, une résilience et une évolutivité pour répondre aux exigences croissantes de l'application.

- **Développement Full Stack et UX** : Ouail Laamiri orchestre le développement full stack, assurant une interface utilisateur intuitive et une expérience fluide, tout en intégrant les dernières tendances en matière de conception.

1.6 Conclusion

Ce projet représente bien plus qu'une simple mise à niveau technologique. Il incarne notre engagement envers l'innovation, l'efficacité opérationnelle et l'amélioration continue. À travers cette PWA basée sur des microservices, nous aspirons à redéfinir les normes de l'assurance automobile, créant une expérience holistique qui place nos utilisateurs au centre de notre démarche.

Fondements Théoriques :

2.1 Introduction à l'Architecture Microservices

Dans le paysage en constante évolution du développement logiciel, l'architecture microservices s'est imposée comme une approche novatrice et efficace pour la conception de systèmes distribués. Cette méthodologie de conception, caractérisée par la décomposition d'une application monolithique en composants indépendants et autonomes, offre une agilité et une flexibilité exceptionnelles dans le déploiement et la gestion des applications.

Cette transition d'une architecture monolithique vers une architecture microservices s'inscrit dans la quête constante d'optimisation des processus de développement, de déploiement et de maintenance des logiciels. Là où les architectures traditionnelles imposent des contraintes en termes de scalabilité, de maintenance et d'innovation, les microservices offrent une approche modulaire qui libère le potentiel de chaque composant du système.

2.1.1 Principes Fondamentaux

Au cœur de l'architecture microservices réside le principe fondamental de la séparation des responsabilités. Chaque microservice est conçu pour accomplir une fonction spécifique, favorisant ainsi une spécialisation qui facilite le développement, le test et la mise à jour indépendante de chaque composant. Cette approche permet également une gestion plus efficace des équipes de développement, chacune étant responsable d'un ou plusieurs microservices.

2.1.2 Avantages Stratégiques

L'adoption de l'architecture microservices offre une multitude d'avantages stratégiques. La scalabilité horizontale, la résilience accrue, la facilité de déploiement continu et la gestion simplifiée des technologies hétérogènes sont autant de bénéfices qui positionnent

cette méthodologie au cœur des pratiques de développement modernes.

2.1.3 Défis et Solutions

Cependant, avec ces avantages viennent des défis uniques, tels que la complexité de la gestion des communications entre microservices, le suivi des données distribuées et la garantie de la cohérence du système dans son ensemble. Ces défis exigent des stratégies et des outils adaptés pour garantir le bon fonctionnement du système global.

2.1.4 Conclusion

En conclusion, l'architecture microservices représente une évolution significative dans le domaine du développement logiciel, offrant une approche agile, modulaire et orientée vers le service. Ce changement de paradigme ouvre de nouvelles possibilités pour répondre aux exigences de performances, de maintenance et d'innovation dans un monde numérique en constante mutation. Ce rapport explorera en détail les tenants et les aboutissants de l'architecture microservices, mettant en lumière ses avantages, ses défis et les meilleures pratiques pour une mise en œuvre réussie.

2.2 Introduction à PWA, SSR et SPA : Transformant l'Expérience Utilisateur

Dans le paysage numérique contemporain, où la rapidité, l'accessibilité et l'expérience utilisateur sont au premier plan, différentes architectures et approches émergent pour répondre aux exigences toujours croissantes. Parmi celles-ci, les Progressive Web Apps (PWA), le rendu côté serveur (SSR) et les applications à page unique (SPA) se distinguent comme des paradigmes majeurs redéfinissant la façon dont nous concevons et livrons des applications web.

2.2.1 Progressive Web Apps (PWA)

Les Progressive Web Apps incarnent une vision avant-gardiste de l'expérience web. En fusionnant les meilleures pratiques du web et des applications mobiles, les PWA offrent une expérience utilisateur transparente, quel que soit le dispositif ou le navigateur utilisé. Ces applications se caractérisent par leur capacité à fonctionner hors ligne, leur installation simplifiée sans nécessité de passer par les stores d'applications, et leur engagement utilisateur accru grâce aux notifications push. Cette approche révolutionnaire élimine les

frontières entre le web et les applications natives, offrant ainsi une expérience utilisateur continue et immersive.

2.2.2 Rendu Côté Serveur (SSR)

Le rendu côté serveur, ou SSR, constitue une réponse aux défis posés par la performance et le référencement des applications web. En adoptant le rendu côté serveur, l'application génère les pages du côté serveur avant de les envoyer au client. Ceci permet d'améliorer la vitesse de chargement initial et d'optimiser le référencement, deux aspects cruciaux pour une expérience utilisateur fluide et une visibilité accrue sur les moteurs de recherche. L'approche SSR offre également une meilleure gestion de la sécurité, garantissant une protection renforcée des données sensibles.

2.2.3 Applications à Page Unique (SPA)

D'un autre côté, les applications à page unique, ou SPA, représentent une approche où une seule page web est chargée dynamiquement, et le contenu est mis à jour sans rechargement complet de la page. Cette architecture offre une expérience utilisateur interactive et rapide en tirant parti des capacités du JavaScript pour charger les données en arrière-plan. Les SPA fournissent une navigation fluide, réduisent les délais de chargement et améliorent la réactivité globale de l'application.

2.2.4 Conclusion

Ce rapport explorera en détail ces trois paradigmes - PWA, SSR et SPA - en analysant leurs avantages, leurs défis, et comment leur intégration judicieuse peut transformer l'expérience utilisateur. Nous examinerons les meilleures pratiques de mise en œuvre, les considérations de performance et les impacts sur le développement, afin de fournir une vision complète des choix architecturaux cruciaux dans le développement d'applications web modernes. En fin de compte, la fusion stratégique de ces approches peut créer des applications web puissantes, réactives et centrées sur l'utilisateur, répondant aux attentes croissantes des utilisateurs dans le monde numérique d'aujourd'hui.

2.3 Introduction aux Containers, Docker,Jenkins,kafka et Kubernetes (K8S), : Révolutionner le Déploiement et l'Orchestration des Applications

Dans le panorama toujours évolutif de l'infrastructure informatique, les concepts de conteneurs, Docker et Kubernetes (K8s) ont émergé comme des piliers essentiels, apportant une transformation significative dans la façon dont les applications sont déployées, gérées et mises à l'échelle. Cette section mettra en lumière les principes fondamentaux de ces technologies révolutionnaires, soulignant leur impact sur la modernisation des opérations informatiques.

2.3.1 Containers : Isolation Légère, Portabilité et Rapidité

Les containers ont introduit une innovation majeure en permettant l'isolation légère des applications et de leurs dépendances. Ils encapsulent tout ce dont une application a besoin pour s'exécuter de manière cohérente, indépendamment de l'environnement d'exécution. Cette approche favorise la portabilité des applications, simplifie le déploiement et garantit une exécution consistante dans divers contextes.

2.3.2 Docker : Révolution dans la Conteneurisation

Docker, en tant que plateforme de conteneurisation, a catalysé la popularisation des containers. Cette technologie offre une manière conviviale de créer, distribuer et exécuter des applications contenues. Grâce à Docker, les développeurs peuvent encapsuler leurs applications avec toutes leurs dépendances, assurant ainsi une exécution sans heurts sur n'importe quel système compatible Docker.

2.3.3 Kubernetes (K8S) : Orchestration d'Échelle et de Complexité

Kubernetes, souvent abrégé en K8s, est l'épine dorsale de l'orchestration des containers. Il fournit une plateforme puissante pour automatiser le déploiement, la mise à l'échelle et la gestion des applications conteneurisées. K8s simplifie la coordination des services, optimise l'utilisation des ressources et assure une haute disponibilité, permettant ainsi une gestion efficace des opérations à grande échelle.

2.3.4 Intégration avec Kafka

Pour compléter l'écosystème de déploiement et d'orchestration, l'intégration avec des outils tels que Kafka est cruciale. Kafka offre une plateforme de streaming en temps réel, permettant le transfert de données à grande échelle et offrant une architecture robuste pour la gestion des flux de données. Son intégration dans un environnement basé sur les conteneurs et Kubernetes permet d'assurer une communication efficace et en temps réel entre les différentes composantes de l'application.

2.3.5 Intégration avec Jenkins

Pour compléter l'écosystème de déploiement et d'orchestration, l'intégration avec des outils tels que Jenkins est cruciale. Jenkins facilite l'intégration continue et le déploiement continu (CI/CD), offrant ainsi une automatisation essentielle pour les processus de développement et de déploiement. Son intégration avec Docker et Kubernetes permet de rationaliser le processus de construction, de test et de déploiement des applications conteneurisées, garantissant ainsi une livraison rapide et fiable des logiciels.

2.3.6 Fondements Conceptuels

Cette section explorera les fondements conceptuels sous-jacents à ces technologies, y compris les espaces de noms, la virtualisation légère, les images Docker, et les concepts de pods et de services dans Kubernetes. La compréhension de ces principes est essentielle pour tirer pleinement parti des avantages qu'offrent les containers et les orchestrations comme Docker et Kubernetes.

2.3.7 Conclusion Anticipée

En synthèse, cette immersion dans les concepts des containers, Docker et Kubernetes jettera les bases nécessaires pour une utilisation éclairée de ces technologies. Comprendre comment ces outils fonctionnent ensemble permettra aux équipes informatiques de construire des infrastructures plus flexibles, évolutives et automatisées, répondant ainsi aux exigences changeantes du paysage informatique moderne.

2.4 La recommandation système et l'apprentissage machine

La recommandation système est un domaine de l'informatique qui vise à prédire les préférences ou les intérêts d'un utilisateur et à recommander des éléments pertinents en fonction de ces prédictions. Les systèmes de recommandation sont largement utilisés dans de nombreux domaines tels que le commerce électronique, les médias sociaux, la musique en streaming, les films et bien d'autres.

Les fondements théoriques des systèmes de recommandation reposent souvent sur des concepts issus du machine learning, en particulier sur les méthodes de filtrage collaboratif et de filtrage basé sur le contenu.

Voici quelques-uns des principaux concepts théoriques utilisés dans les systèmes de recommandation :

2.4.1 Filtrage collaboratif

Cette approche repose sur l'idée que les utilisateurs similaires ont tendance à apprécier des éléments similaires. Il existe deux types principaux de filtrage collaboratif : basé sur les utilisateurs et basé sur les éléments. Dans le filtrage collaboratif basé sur les utilisateurs, les recommandations sont faites en fonction des préférences des utilisateurs similaires. Dans le filtrage collaboratif basé sur les éléments, les recommandations sont faites en fonction de la similarité entre les éléments.

2.4.2 Filtrage basé sur le contenu

Cette approche recommande des éléments similaires à ceux que l'utilisateur a aimés dans le passé, en se basant sur les caractéristiques des éléments et les préférences déclarées de l'utilisateur. Par exemple, dans le cas des films, cela pourrait signifier recommander des films similaires en fonction du genre, des acteurs, du réalisateur, etc.

2.4.3 Modèles probabilistes et statistiques

Les modèles probabilistes et statistiques sont utilisés pour modéliser les préférences des utilisateurs et prédire les éléments qu'ils sont susceptibles d'apprécier. Les méthodes telles que les modèles de facteurs latents (comme la factorisation de matrices) sont couramment utilisées pour ce faire.

2.4.4 Évaluation des performances

Pour évaluer l'efficacité des systèmes de recommandation, diverses mesures sont utilisées, telles que la précision, le rappel, la F-mesure, la courbe ROC, etc.

2.4.5 Traitement du biais et de la froideur

Le biais se réfère à la tendance des systèmes de recommandation à recommander les éléments populaires plutôt que ceux qui pourraient être plus pertinents pour l'utilisateur. La froideur fait référence au défi de recommander des éléments pour lesquels il y a peu ou pas d'historique disponible.

2.4.6 Apprentissage en ligne et mise à l'échelle

Les systèmes de recommandation doivent souvent gérer de grandes quantités de données et s'adapter rapidement aux préférences changeantes des utilisateurs. L'apprentissage en ligne et les techniques de mise à l'échelle sont donc essentiels.

2.4.7 Conclusion Anticipée

En résumé, les systèmes de recommandation reposent sur un ensemble de principes théoriques issus du machine learning et de la statistique, adaptés aux besoins spécifiques de la recommandation d'éléments aux utilisateurs.

2.5 La technologie blockchain et les applications décentralisées

La technologie blockchain et les applications décentralisées (DApps) reposent sur un ensemble de principes théoriques fondamentaux qui les rendent uniques et puissants. Voici certains de ces concepts théoriques :

2.5.1 Blockchain

- **Structure de données distribuée** : La blockchain est une structure de données distribuée et décentralisée qui enregistre de manière sécurisée les transactions entre participants d'un réseau.
- **Consensus distribué** : Les blockchains utilisent des mécanismes de consensus distribué, tels que la preuve de travail (PoW), la preuve d'enjeu (PoS), etc., pour

parvenir à un accord sur l'état du réseau et sur les transactions valides.

- **Cryptographie** : La cryptographie est utilisée pour sécuriser les transactions et les données stockées sur la blockchain, assurant l'intégrité et la confidentialité des informations.

2.5.2 Contrats intelligents (Smart Contracts)

- **Automatisation des transactions** : Les contrats intelligents sont des programmes auto-exécutables qui automatisent l'exécution de contrats et d'accords entre les parties, sans nécessiter d'intermédiaires.
- **Turing-complétude** : Les contrats intelligents sont souvent turing-complets, ce qui signifie qu'ils peuvent effectuer une large gamme de fonctions et de calculs.

2.5.3 Décentralisation

- **Absence d'autorité centrale** : Les systèmes décentralisés comme les blockchains et les DApps fonctionnent sans autorité centrale, ce qui signifie que les décisions sont prises de manière distribuée par les participants du réseau.
- **Résilience et censure résistante** : La décentralisation rend les systèmes plus résilients aux pannes et aux attaques, car il n'y a pas de point unique de défaillance ou de contrôle.

2.5.4 Consensus

- **Mécanismes de consensus** : Les blockchains utilisent divers mécanismes de consensus pour parvenir à un accord sur l'état du réseau, garantissant que toutes les parties sont d'accord sur les transactions valides.
- **Sécurité et immuabilité** : Le consensus garantit la sécurité et l'immuabilité des données en empêchant la modification non autorisée des enregistrements existants sur la blockchain.

2.5.5 Ethereum et les plateformes de DApp

- **Plateformes de développement** : Ethereum est l'une des plateformes les plus populaires pour le développement de DApps, offrant un environnement de développement robuste pour la création de contrats intelligents et d'applications décentralisées.

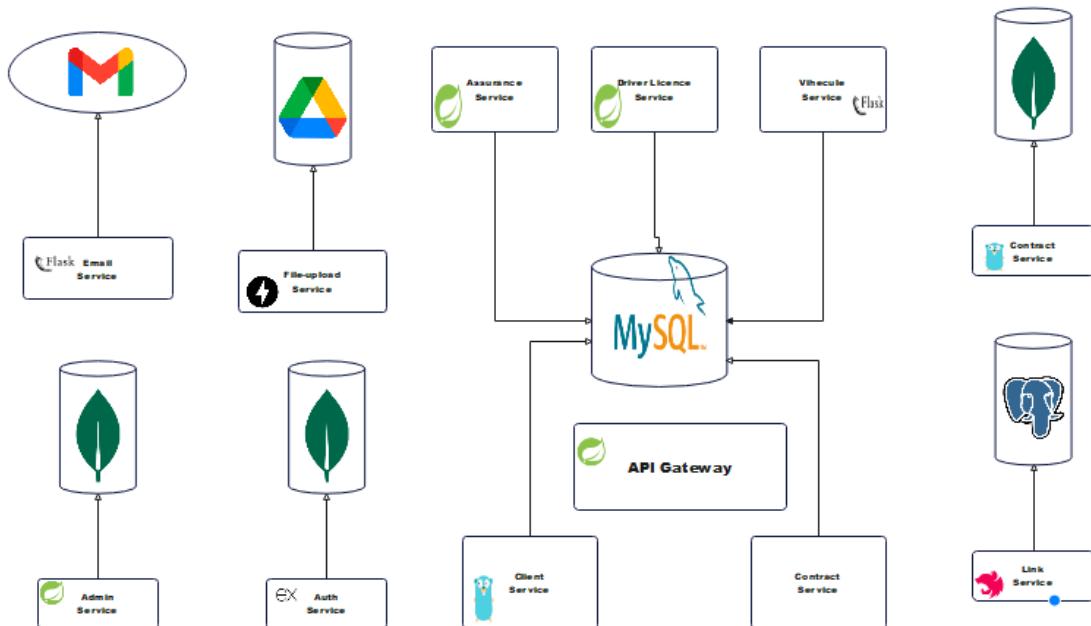
- **Interopérabilité et intégration** : Les DApps peuvent interagir entre elles et avec des systèmes externes grâce à des standards et des protocoles d'interopérabilité.

2.5.6 Conclusion Anticipée

Ensemble, ces concepts théoriques forment la base de la technologie blockchain et des applications décentralisées, ouvrant la voie à de nouveaux modèles économiques, à une gouvernance distribuée et à une confiance accrue dans les transactions numériques.

Développement web

3.1 Back-End : Micro Services



3.1.1 client-service

Client-Service est une interface de programmation d'application (API) permettant de gérer l'ajout, la mise à jour, la suppression et la récupération des clients de la société ZOHS.

<https://github.com/ZOHSGroup/Client-Service>

3.1.2 login-service

Auth-Service est une interface de programmation d'application (API) chargée de gérer le processus de connexion et d'inscription.

<https://github.com/ZOHSGroup/Login-SignUp-Service>

3.1.3 assurance-service

Assurance-Service est une interface de programmation d'application permettant d'ajouter, de mettre à jour, de supprimer et de récupérer les assurances de la société ZOHS.

<https://github.com/ZOHSGroup/Assurance-Service>

3.1.4 vehicule-service

Vehicule-Service est une interface de programmation d'application (API) permettant d'ajouter, de supprimer, de mettre à jour et de récupérer des véhicules.

<https://github.com/ZOHSGroup/Vehicle-Service>

3.1.5 contract-service

Contract-Service est une interface de programmation d'application (API) permettant de gérer l'ajout, la mise à jour, la suppression et la récupération des contrats de la société ZOHS dans une base de données NoSQL.

<https://github.com/ZOHSGroup/Contract-Service>

3.1.6 DriverLicense-service

DriverLicense-Service est une interface de programmation d'application permettant d'ajouter, de mettre à jour, de supprimer et de récupérer les permis de conduire de la société ZOHS.

<https://github.com/ZOHSGroup/DriverLicense-Service>

3.1.7 Email-service

Email-Service est une interface de programmation d'application (API) permettant d'envoyer des e-mails aux clients de la compagnie d'assurance ZOHS.

<https://github.com/ZOHSGroup/Email-Service>

3.1.8 File-service

File-Service est une interface de programmation d'application (API) permettant de télécharger des fichiers sur Google Drive de la compagnie d'assurance ZOHS.

<https://github.com/ZOHSGroupe/File-Service>

3.1.9 Link-service

Link-Service est une interface de programmation d'application permettant d'ajouter, de mettre à jour, de supprimer et de récupérer les liens de la société ZOHS.

<https://github.com/ZOHSGroupe/Link-Service>

3.1.10 service-discovery

un service de découverte est un composant essentiel dans les architectures distribuées modernes, offrant une manière dynamique et flexible de gérer les services et les communications entre eux.

https://github.com/ZOHSGroupe/service_discovery

3.1.11 server-getway

un serveur de passerelle API joue un rôle crucial dans la construction d'architectures distribuées, en offrant une interface unifiée, sécurisée et optimisée pour l'accès aux services internes. Il simplifie la gestion des requêtes, améliore la sécurité, la scalabilité et les performances, tout en offrant des fonctionnalités avancées telles que l'authentification, l'autorisation, la répartition de charge et la surveillance.

https://github.com/ZOHSGroupe/server_getway

3.2 Front End:Capture d'eCAN

<https://github.com/ZOHSGroupe/Front-End-ZOHS>

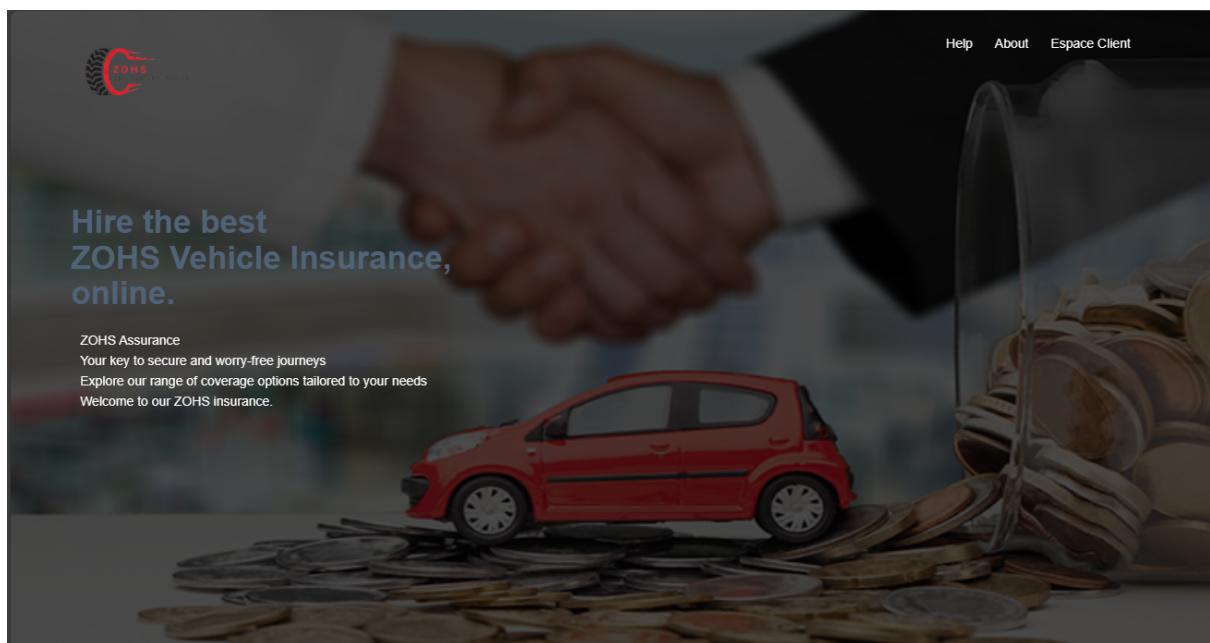


Figure 3.1: La page home

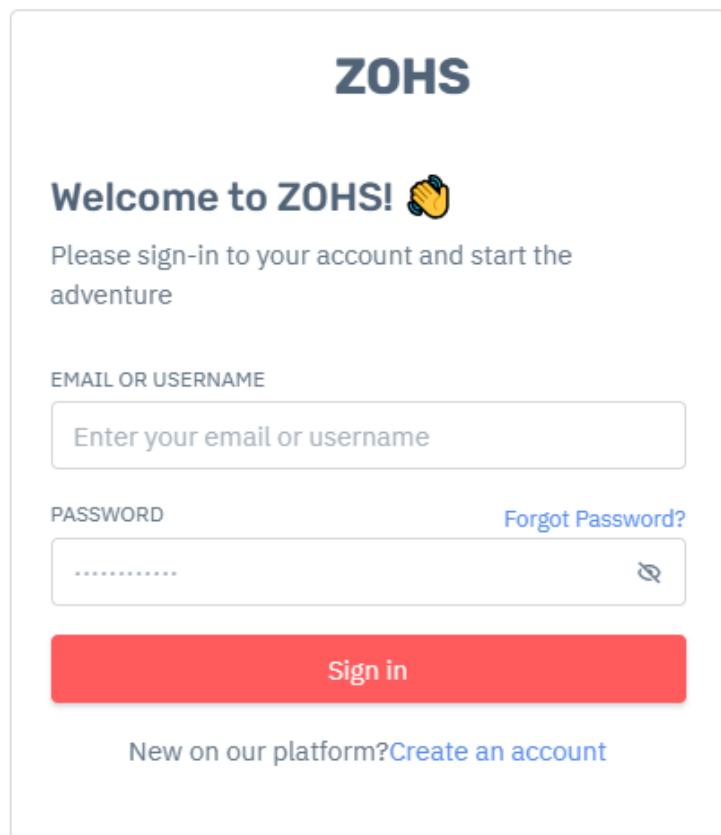
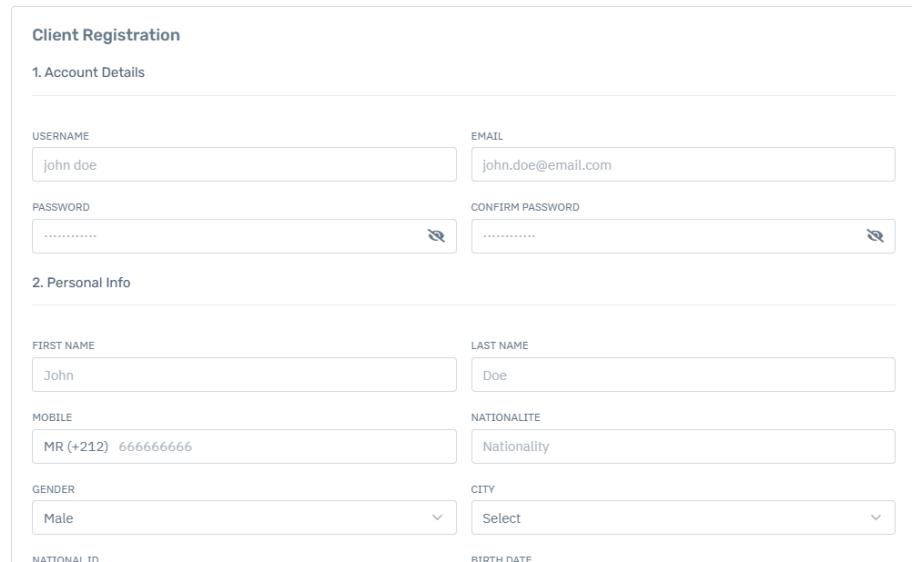


Figure 3.2: La page Signin



The image shows a client registration form titled "Client Registration". It is divided into two main sections: "1. Account Details" and "2. Personal Info".

1. Account Details:

- USERNAME: john doe
- EMAIL: john.doe@email.com
- PASSWORD: (with eye icon)
- CONFIRM PASSWORD: (with eye icon)

2. Personal Info:

- FIRST NAME: John
- LAST NAME: Doe
- MOBILE: MR (+212) 666666666
- NATIONALITE: Nationality
- GENDER: Male
- CITY: Select
- NATIONAL ID: (empty)
- BIRTH DATE: (empty)

Figure 3.3: La page Signin

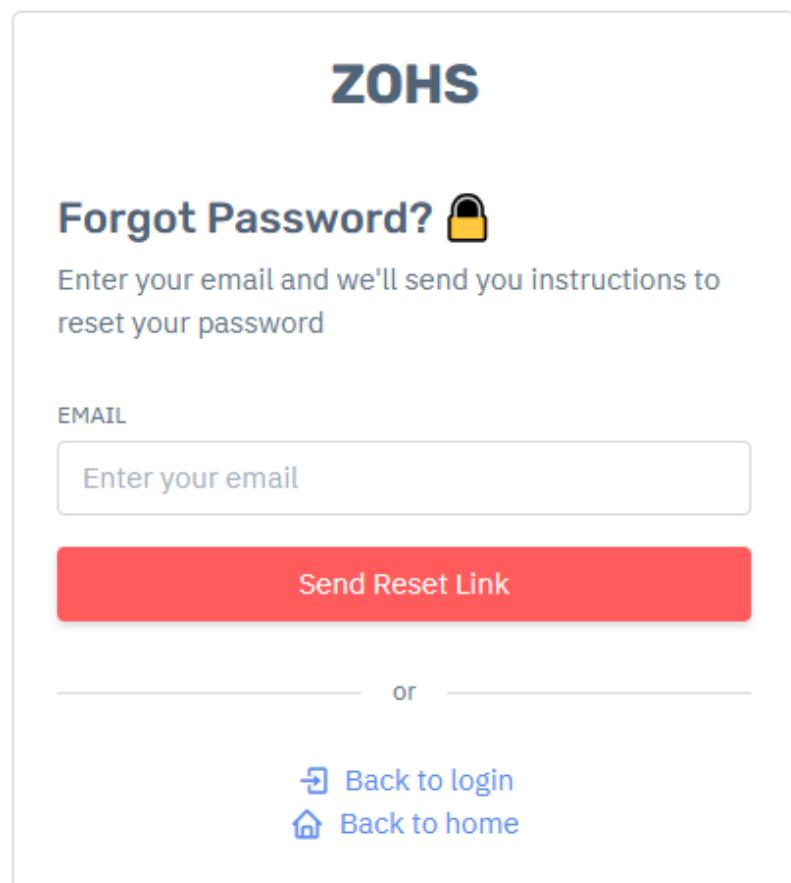


Figure 3.4: La page forgot password

The screenshot shows a web application interface. On the left is a sidebar menu with the following items:

- Home
- Notifications
- Help
- About
- Person Information
- Vehicle
- Driver License
- Assurance

On the right, there is a search bar with placeholder text "search...". Below it is a table titled "Light Table head". The table has columns: TYPE, LICENSE NUMBER, ISSUE DATE, EXPIRATION DATE, CREATION DATE, STATUS, and ACTIONS. It contains two rows of data:

TYPE	LICENSE NUMBER	ISSUE DATE	EXPIRATION DATE	CREATION DATE	STATUS	ACTIONS
CarB	ABC123	1640991600000	1711929600000	1706137200000	ACTIVE	⋮
CarA	ABC123	1640991600000	1711929600000	1706137200000	PENDING	⋮

Figure 3.5: La page forgot password

La Systeme de recommandation d'assurance



Il semble que vous introduisiez les trois phases de développement pour un projet de système de recommandation utilisant l'apprentissage automatique, qui sont :

1. Regroupement des données d'assurance
2. Construction du modèle pour la recommandation
3. Architecture du projet en utilisant MLOps

4.1 Regroupement des données d'assurance

Dans cette phase , mon objectif était **de réaliser une classification non supervisée des données d'assurance**. Les étapes clés comprenaient *le nettoyage des données* , l'élimination des duplicitas et des valeurs manquantes, suivi *d'une analyse de clustering à l'aide de l'algorithme k-means*.

4.1.1 Nettoyage des données

- **Suppression des duplicitas** : Une première étape cruciale pour assurer la qualité des données a été la suppression des entrées dupliquées, garantissant ainsi l'intégrité des informations.
- **Suppression des valeurs manquantes** : Les données ont été préalablement traitées pour éliminer les valeurs manquantes, évitant ainsi tout impact négatif sur le processus de clustering.

4.1.2 Construction du modèle pour la recommandation

- **Choix de l'algorithme k-means** : Nous avons opté pour l'algorithme k-means en raison de sa simplicité et de son efficacité dans la classification non supervisée.

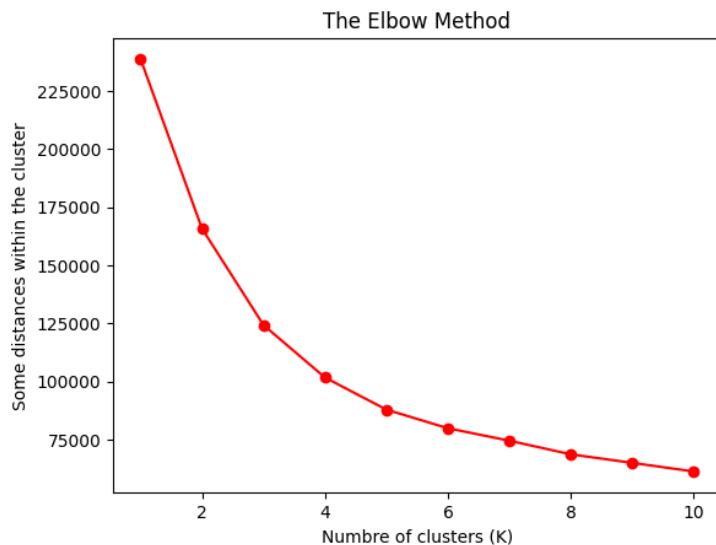


Figure 4.1: WCSS choisir le numéro K

- **Analyse des résultats** : Après l'application de l'algorithme k-means, nous avons obtenu trois clusters distincts, chacun représentant un type spécifique d'assurance. Ces clusters ont été étiquetés comme suit :
 - **Cluster 1** : Assurance Responsabilité Civile .
 - **Cluster 2** : Assurance avec Dommages .
 - **Cluster 3** : Assurance Tous Risques .

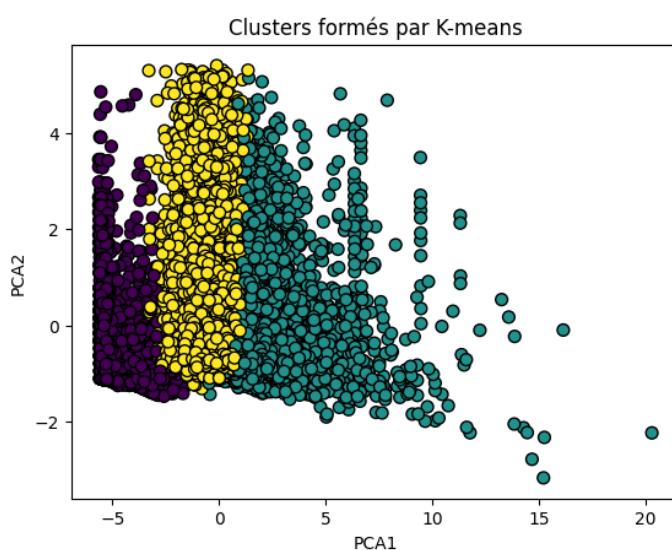


Figure 4.2: Visualisation des points de données après le clustering

Interprétation des Clusters

- **Cluster 1 (Assurance Responsabilité Civile)** : Ce cluster peut être associé à des produits d'assurance responsabilité civile, mettant l'accent sur la couverture des dommages causés à des tiers.
- **Cluster 2 (Assurance avec Dommages)** : Il semble représenter des produits d'assurance couvrant des dommages spécifiques, probablement liés à des biens ou des possessions.
- **Cluster 3 (Assurance Tous Risques)** : Ce cluster peut être interprété comme regroupant des produits d'assurance offrant une couverture complète, y compris les dommages aux biens et la responsabilité civile.

4.2 Classification Supervisée après Clustering

Dans cette phase , mon objectif était de réaliser une classification non supervisée des données d'assurance.

4.2.1 Traitement des Données Déséquilibrées : UNBALANSED DATA

Afin de remédier au déséquilibre des données généré par le clustering, nous avons équilibré le nombre d'instances pour chacun des trois clusters. Cela a été réalisé en sous-échantillonnant

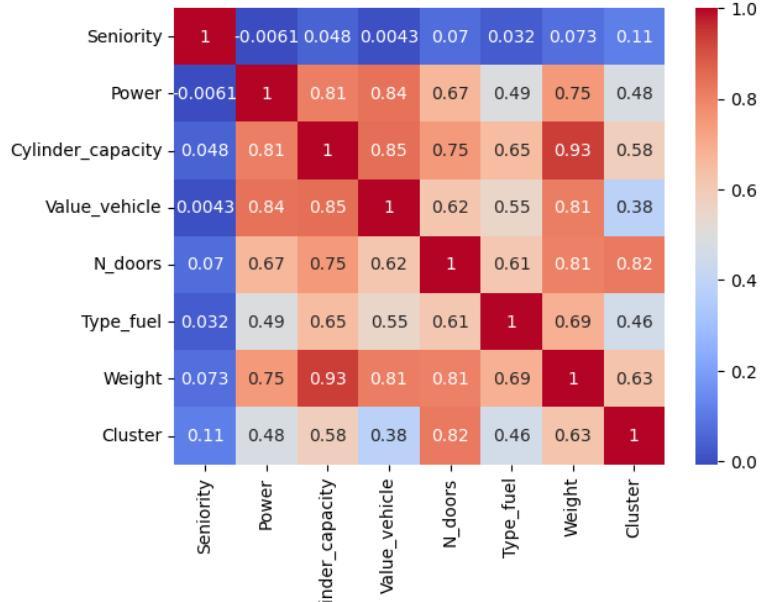


Figure 4.3: visualisation de la corrélation des caractéristiques

les clusters majoritaires pour correspondre à la taille du cluster minoritaire.

```
... # Créer une liste pour stocker les DataFrames équilibrés
balanced_dfs = []

# Boucler sur chaque cluster
for cluster_label, group in insurance.groupby('Cluster'):
    # Sous-échantillonner le cluster pour atteindre le nombre minimum d'observations
    balanced_cluster = resample(group, replace=False, n_samples=min_observation_count, random_state=42)

    # Ajouter le cluster équilibré à la liste
    balanced_dfs.append(balanced_cluster)

# Concaténer les DataFrames équilibrés pour créer un DataFrame équilibré
balanced_insurance = pd.concat(balanced_dfs, ignore_index=True)

# Afficher la distribution des clusters dans le nouveau DataFrame
print(balanced_insurance['Cluster'].value_counts())
```

Cluster	count
0	2671
1	2671
2	2671

Figure 4.4: Apré Traitement des Données Déséquilibrées

4.2.2 Division des Données

Les données équilibrées ont ensuite été divisées en ensembles d’entraînement (X_train, y_train) et de test (X_test, y_test) pour permettre l’évaluation du modèle.

4.2.3 Modèle de Stacking

Nous avons utilisé un modèle de stacking qui combine les prédictions de plusieurs modèles de base. Les modèles de base inclus étaient :

1. **AdaBoostClassifier**
2. **GaussianNB**
3. **Support Vector Machine (SVC)**
4. **K-Nearest Neighbors (KNeighborsClassifier)**
5. **RandomForestClassifier**

Le modèle final a été construit en utilisant Logistic Regression comme meta-classifieur pour agréger les prédictions des modèles de base.

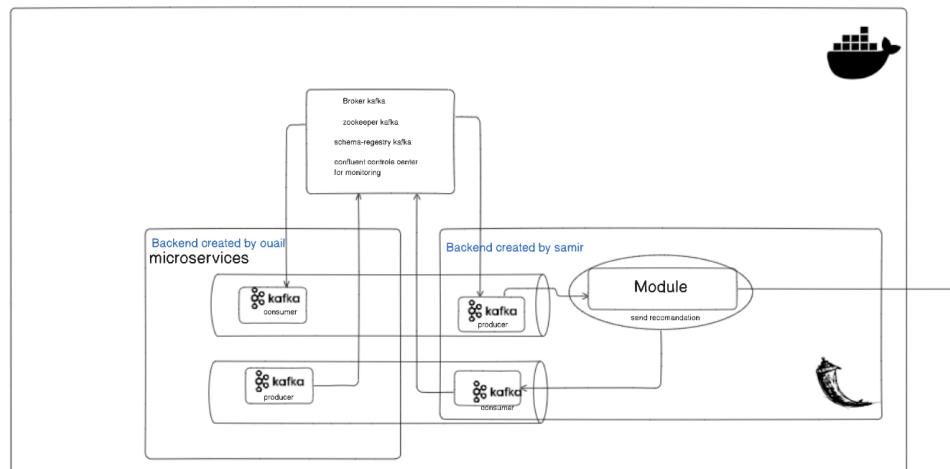
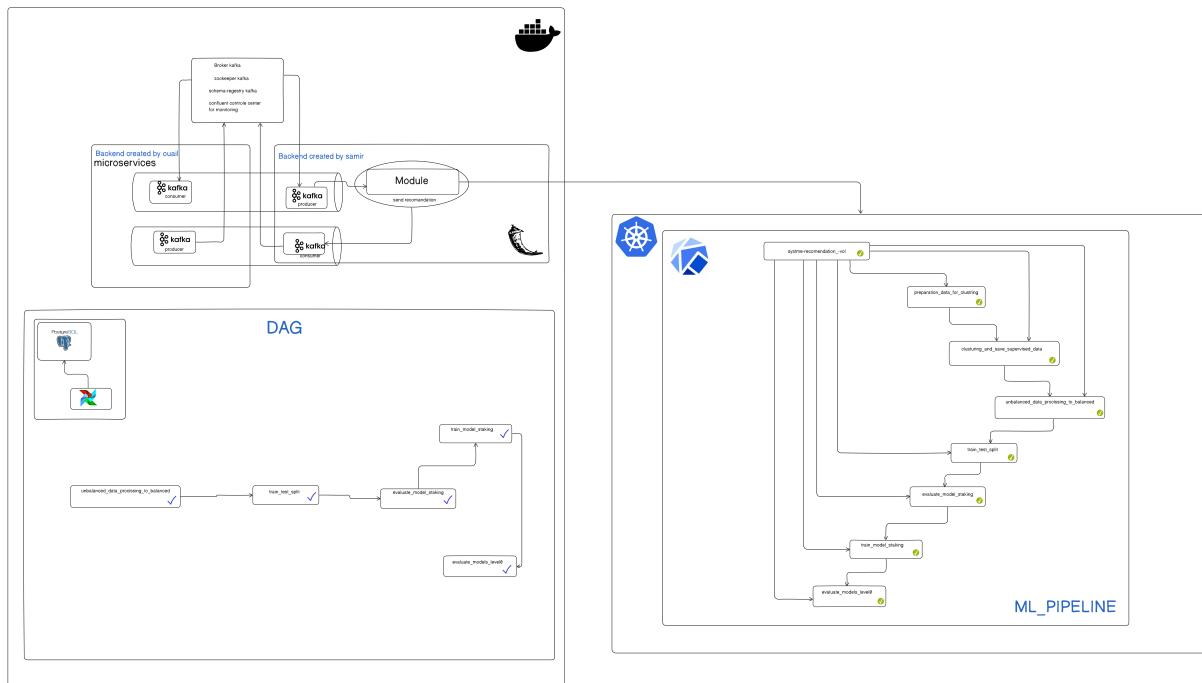
4.2.4 Évaluation du Modèle

Le modèle de stacking a montré des performances remarquables lors de l'évaluation sur l'ensemble de test.

Métrique	Score
Précision (Accuracy)	98.79%
Précision (Precision)	98.81%
Rappel (Recall)	98.79%
F1 Score	98.79%

interprétation :

- Le modèle a bien généralisé sur toutes les classes, avec une précision élevée, indiquant une capacité à classifier correctement les instances de chaque cluster.
- Aucune instance n'a été mal classée pour le Cluster 1, démontrant une excellente performance sur cette catégorie.
- Le faible nombre d'erreurs pour les Clusters 2 et 3 souligne la robustesse du modèle face à des données équilibrées.



4.3 architecturer le projet en utilisant Mlops

4.3.1 Integration kafka

Le diagramme montre que les microservices communiquent entre eux et avec un "Module de machine learning" central utilisant Kafka, qui est une plate-forme de streaming distribuée souvent utilisée pour créer des pipelines de données en temps réel et des applications de streaming. Concrètement, on peut voir :

Consommateur Kafka : composant qui consomme les messages d'un sujet Kafka.
Kafka Producer : composant qui produit des messages sur un sujet Kafka.
nous utilisant 4 conteneurs dans cette partie :

1. Apache Zookeeper (zookeeper) :

- **Description** : Gère la coordination distribuée pour Kafka.
- **Fonctionnement** : Fournit des services de coordination et de gestion de configuration nécessaires pour le bon fonctionnement de Kafka.

2. Apache Kafka Broker (broker) :

- **Description** : Courtier de messages Kafka.
- **Fonctionnement** : Stocke et gère les messages Kafka, expose les ports 9092 et 9101 pour la communication.

3. Confluent Schema Registry (schema-registry) :

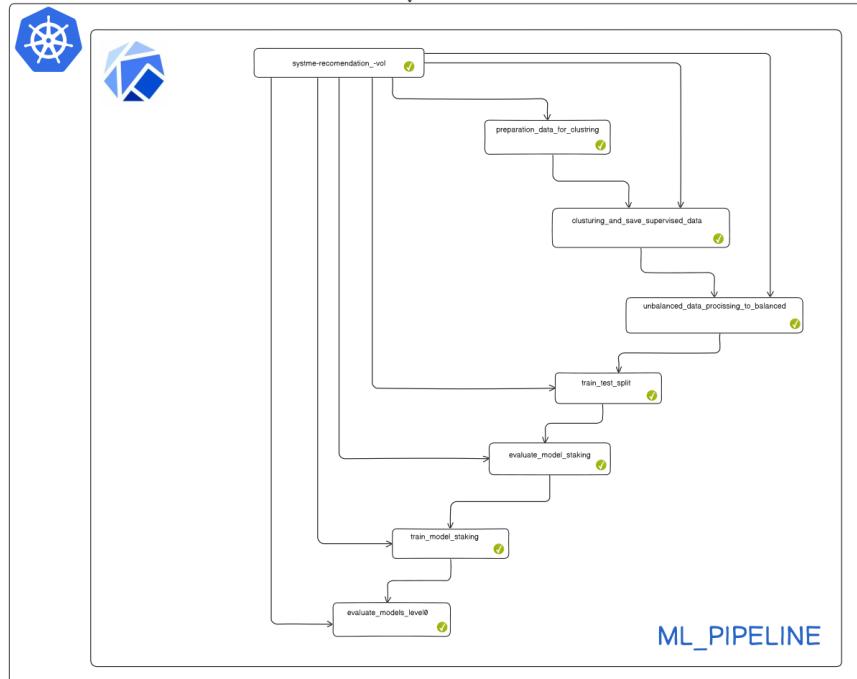
- **Description** : Gère les schémas Avro pour les sujets Kafka.
- **Fonctionnement** : Fournit un registre central pour stocker, récupérer et valider les schémas utilisés pour sérialiser/déserialiser les messages.

4. Confluent Control Center (control-center) :

- **Description** :: Interface utilisateur basée sur le web pour la surveillance et la gestion de Kafka.
- **Fonctionnement** : Permet aux utilisateurs de surveiller les clusters Kafka, de visualiser les données de production et d'effectuer des opérations de gestion.

4.3.2 Pipeline ML utilisant Kubeflow

Voici un bref résumé de ML_pipeline:



```

samir5636@samir5636: ~
[samir5636@samir5636: ~]
*serve-controller          active  1 kserve-controller      0.11/stable   435  10.152.183.230  no
*ubertflow-dashboard       active  1 kubeflow-dashboard  1.0/stable    454  10.152.183.40  no
*ubertflow-titles          active  1 kubeflow-titles   1.0/stable    371  10.152.183.40  no
*ubertflow-roles           active  1 kubeflow-roles   1.0/stable    187  10.152.183.144 no
*ubertflow-volumes         active  1 kubeflow-volumes 1.0/stable    260  10.152.183.143 no
*ubertflow-controller-operator active  1 ubertflow-controller-operator  cdk-1.0/stable  258  10.152.183.73  no
*ubertflow-nodo             active  1 ninfo               1.14/stable   127  10.152.183.240 no
*ubertflow-ldc-gatekeeper   active  1 ldc-gatekeeper     1.0/stable    201  10.152.183.240 no
*ubertflow-pvcviewer        active  1 pvcviewer-operator  1.0/stable    30   10.152.183.92  no
*ubertflow-tensorboard-controller active  1 tensorboard-controller  1.0/stable    579  10.152.183.233 no
*tensorboards-web-app       active  1 tensorboards-web-app  1.0/stable    245  10.152.183.233 no
*training-operator          active  1 training-operator   1.7/stable   330  10.152.183.224 no

*unit
*ubertflow-vphub0/*        active  idle   10.1.254.74
*argo-controller/*          active  idle   10.1.254.76
*dex-auth/*                 active  idle   10.1.254.76
*es/*                        active  idle   10.1.254.76
*istio-ingressgateway/*    active  idle   10.1.254.76  9090,9991/TCP
*istio-pilot/*              active  idle   10.1.254.77
*appflow-controller/*      active  idle   10.1.254.78
*appflow-viewer/*           active  idle   10.1.254.78
*uplayer-ul/*               active  idle   10.1.254.81
*atib-controller/*          active  idle   10.1.254.119  443,8088/TCP
*atib-db/*                  active  idle   10.1.254.88
*atib-dbo/*                 active  idle   10.1.254.88
*fq-dbo/*                  active  idle   10.1.254.88
*fp-dbo/*                  active  idle   10.1.254.92
*fp-profile-data-writer/*  active  idle   10.1.254.93
*fp-profile-persistence/*  active  idle   10.1.254.98
*fp-profile-controller/*  active  idle   10.1.254.99
*fp-profile-dbf/*           active  idle   10.1.254.99
*fp-ui/*                   active  idle   10.1.254.99
*p-vviewer/*                active  idle   10.1.254.99
*pv/*                       active  idle   10.1.254.99
*native-eventing/*         active  idle   10.1.254.99
*native-operator/*          active  idle   10.1.254.102
*native-logger/*            active  idle   10.1.254.103
*serve-controller/*          active  idle   10.1.254.103
*ubertflow-dashboard/*     active  idle   10.1.254.103
*ubertflow-titles/*         active  idle   10.1.254.103
*ubertflow-roles/*          active  idle   10.1.254.103
*ubertflow-volumes/*        active  idle   10.1.254.103  5000/TCP
*ninfo/*                   active  idle   10.1.254.121  9000-9081/TCP
*ldc/*                      active  idle   10.1.254.134
*ldc-gatekeeper/*           active  idle   10.1.254.134
*vcvviewer/*                active  idle   10.1.254.139
*atib-controller/*          active  idle   10.1.254.140
*atib-dbo/*                 active  idle   10.1.254.140
*tensorboard-controller/*  active  idle   10.1.254.187
*tensorboards-web-app/*    active  idle   10.1.254.188
*train/*                   active  idle   10.1.254.189
*trainoperator/*            active  idle   10.1.254.189
[samir5636@samir5636: ~]

```

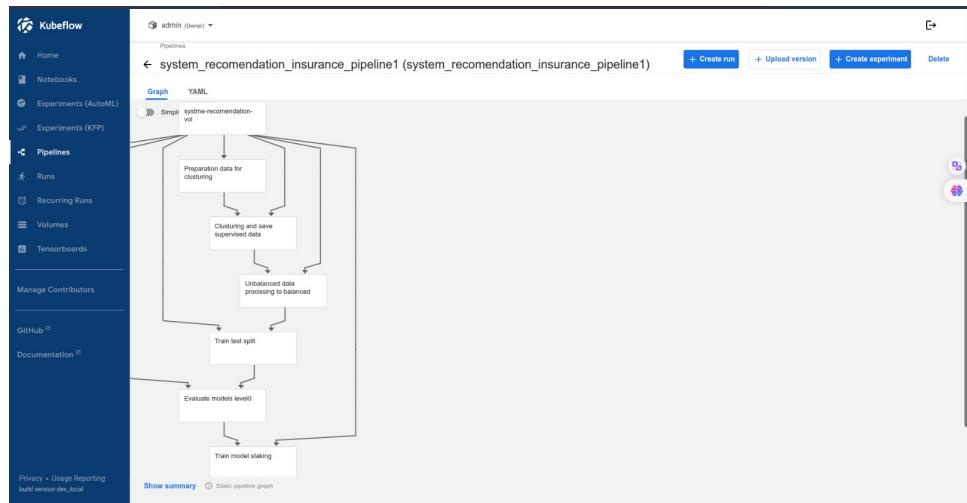
The screenshot shows the Kubeflow UI interface for managing ML pipelines. On the left, there are navigation menus for Home, Notebooks, Experiments (AutoML), Experiments (KFP), Pipelines, Runs, Recurring Runs, Volumes, Tensorboards, Manage Contributors, GitHub, and Documentation. The main area displays a pipeline named 'system_recomendation_insurance_pipeline1' (derived from 'system_recomendation_insurance_pipeline1'). The pipeline graph is shown above, and the detailed YAML configuration is displayed below.

```

graph TD
    A[system-recomendation_v0] --> B[preparation_data_for_clustering]
    B --> C[clustering_and_save_supervised_data]
    C --> D[unbalanced_data_processing_to_balanced]
    D --> E[train_test_split]
    E --> F[evaluate_model_staking]
    F --> G[train_model_staking]
    G --> H[evaluate_models_level0]
    H --> I[evaluate_model_staking]
    I --> J[train_model_staking]
    J --> K[evaluate_models_level0]
    K --> L[evaluate_model_staking]
    L --> M[evaluate_models_level0]
    M --> N[evaluate_model_staking]
    N --> O[evaluate_models_level0]
    O --> P[evaluate_model_staking]
    P --> Q[evaluate_models_level0]
    Q --> R[evaluate_model_staking]
    R --> S[evaluate_model_staking]
    S --> T[evaluate_model_staking]
    T --> U[evaluate_model_staking]
    U --> V[evaluate_model_staking]
    V --> W[evaluate_model_staking]
    W --> X[evaluate_model_staking]
    X --> Y[evaluate_model_staking]
    Y --> Z[evaluate_model_staking]
    Z --> AA[evaluate_model_staking]
    AA --> BB[evaluate_model_staking]
    BB --> CC[evaluate_models_level0]
    CC --> DD[evaluate_model_staking]
    DD --> EE[evaluate_model_staking]
    EE --> FF[evaluate_model_staking]
    FF --> GG[evaluate_model_staking]
    GG --> HH[evaluate_model_staking]
    HH --> II[evaluate_model_staking]
    II --> JJ[evaluate_model_staking]
    JJ --> KK[evaluate_models_level0]
    KK --> LL[evaluate_model_staking]
    LL --> MM[evaluate_models_level0]
    MM --> NN[evaluate_model_staking]
    NN --> OO[evaluate_model_staking]
    OO --> PP[evaluate_model_staking]
    PP --> QQ[evaluate_models_level0]
    QQ --> RR[evaluate_model_staking]
    RR --> SS[evaluate_model_staking]
    SS --> TT[evaluate_model_staking]
    TT --> UU[evaluate_model_staking]
    UU --> VV[evaluate_model_staking]
    VV --> WW[evaluate_model_staking]
    WW --> XX[evaluate_model_staking]
    XX --> YY[evaluate_model_staking]
    YY --> ZZ[evaluate_model_staking]
    ZZ --> AA

```

The YAML code defines the pipeline structure, including components like 'preparation_data_for_clustering', 'clustering_and_save_supervised_data', 'unbalanced_data_processing_to_balanced', 'train_test_split', 'evaluate_model_staking', 'train_model_staking', and 'evaluate_models_level0'. It also specifies parameters such as 'data_path', 'experiment', 'model', 'template', 'component', and 'script'. The code includes Python scripts for data processing and model evaluation, utilizing libraries like pandas, numpy, and scikit-learn.



Pipeline name	Description	Uploaded on
system_recommendation_insurance_pipeline1		1/26/2024, 12:42:27 AM

1. preparation_data_for_clustering:

- **Description** : Prépare les données pour le regroupement.
- **Sorties** : Données préparées pour le regroupement.

2. clustering_and_save_supervised_data:

- **Description** : Effectue un regroupement et sauvegarde les données supervisées.
- **Sorties** : Données regroupées et données supervisées.
- **Dépendances** : preparation_data_for_clustering

3. unbalanced_data_procissing_to_balanced:

- **Description** : Équilibre les données non équilibrées.
- **Sorties** : Données équilibrées.
- **Dépendances** : clustering_and_save_supervised_data

4. train_test_split:

- **Description** : Divise les données en ensembles d'entraînement et de test.
- **Sorties** : Ensembles d'entraînement et de test.
- **Dépendances** : unbalanced_data_procissing_to_balanced

5. evaluate_models_level0:

- **Description** : Évalue les modèles de niveau 0.
- **Sorties** : Résultats d'évaluation pour les modèles de niveau .
- **Dépendances** : train_test_split

6. train_model_stacking:

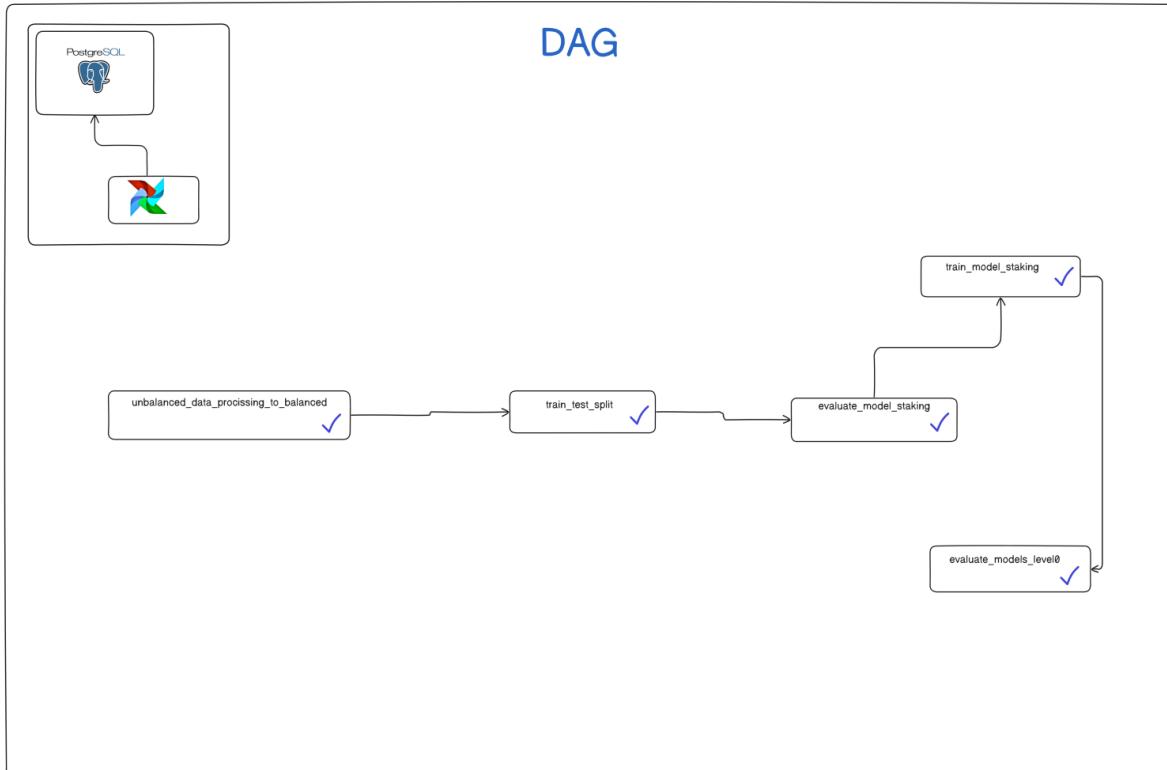
- **Description** : Entraîne un ensemble de modèles de stacking.
- **Sorties** : Ensemble de modèles de stacking entraînés.
- **Dépendances** : evaluate_models_level0

7. evaluate_model_stacking:

- **Description** : Évalue l'ensemble de modèles de stacking.
- **Sorties** : Résultats d'évaluation pour l'ensemble de modèles de stacking.
- **Dépendances** : train_model_stacking

4.3.3 Integration Airflow

Dans cette partie je utilise apache airflow pour automatise le retrenement du model chaque 1 mois .



- **Préparation des Données** : Cette tâche consiste à préparer les données avant de les utiliser pour l'entraînement du modèle. Elle peut inclure le chargement des données à partir de sources externes telles que des bases de données ou des fichiers, le nettoyage des données en éliminant les valeurs manquantes ou aberrantes, et la transformation des données en un format adapté à l'entraînement du modèle.
- **Entraînement du Modèle** La tâche d'entraînement du modèle exécute le code qui entraîne réellement le modèle de Machine Learning. Elle utilise les données préparées dans la tâche précédente pour ajuster les paramètres du modèle. Cette tâche peut inclure des algorithmes d'apprentissage supervisé ou non supervisé selon le type de modèle que vous utilisez.
- **Évaluation du Modèle** Une fois que le modèle est entraîné, il est important de l'évaluer pour comprendre ses performances. La tâche d'évaluation du modèle utilise souvent un ensemble de données de test distinct pour évaluer la capacité du modèle à généraliser sur de nouvelles données. Elle peut calculer différentes

métriques d'évaluation telles que l'exactitude, la précision, le rappel, ou d'autres métriques spécifiques au problème.

- **Réentraînement du Modèle (Tâche Mensuelle)** Cette tâche planifie le réentraînement périodique du modèle, comme vous l'avez mentionné, pour garantir que le modèle reste précis et adapté aux données récentes. Elle peut être déclenchée à des intervalles réguliers, par exemple chaque mois, et réexécute le pipeline complet de préparation des données, d'entraînement du modèle et d'évaluation.

Le Deveoppment BlockChain

participatif décentralisée utilisant la technologie de la blockchain, plus précisément la plateforme Ethereum. Cette application décentralisée (DApp) permettra aux utilisateurs de créer, financer et participer à des campagnes de financement participatif de manière sécurisée et transparente.

5.1 Technologies utilisées :

5.1.1 Solidity :

Solidity est un langage de programmation utilisé pour écrire des contrats intelligents sur des plateformes de blockchain telles qu’Ethereum. Il est conçu pour être compatible avec la machine virtuelle Ethereum (EVM).

5.1.2 Smart Contract :

Un smart contract est un protocole informatique destiné à faciliter, vérifier ou appliquer numériquement l’exécution ou la négociation d’un contrat, permettant ainsi la vérification, l’exécution ou la négociation automatique de contrats.

5.1.3 Truffle:

Truffle est un framework de développement Ethereum qui simplifie le processus de développement, de test et de déploiement de contrats intelligents. Il fournit une suite d’outils, notamment un environnement de développement, un gestionnaire de paquets, un compilateur et un débogueur.

5.1.4 Sepolia Network :

Il semble y avoir une faute de frappe dans votre liste. Je suppose que vous vouliez dire "Polkadot Network". Polkadot est une plateforme de blockchain qui vise à permettre l'interopérabilité entre différentes chaînes de blocs.

5.1.5 Rinkeby :

Rinkeby est un réseau de test Ethereum utilisé par les développeurs pour tester leurs contrats intelligents et leurs applications décentralisées (dApps) avant de les déployer sur le réseau principal Ethereum.

5.1.6 Etherscan :

Etherscan est un explorateur de blocs pour la blockchain Ethereum. Il permet aux utilisateurs de rechercher et de visualiser des informations sur les transactions, les adresses, les contrats intelligents et les blocs sur la blockchain Ethereum.

5.1.7 Ganache:

Ganache est un environnement de test personnel Ethereum qui permet aux développeurs de tester leurs contrats intelligents et leurs applications décentralisées localement. Il fournit une blockchain Ethereum locale pour le développement et le test.

5.1.8 Geth:

Geth est l'implémentation de référence du client Ethereum en Go. Il permet aux utilisateurs de se connecter au réseau Ethereum, de synchroniser avec la blockchain Ethereum et d'interagir avec les contrats intelligents.

5.1.9 Metamask:

Metamask est une extension de navigateur qui permet aux utilisateurs d'accéder à des applications décentralisées (dApps) basées sur Ethereum directement à partir de leur navigateur. Il agit comme un portefeuille Ethereum et fournit une interface utilisateur conviviale pour interagir avec la blockchain Ethereum.

5.1.10 Express.js :

Express.js est un framework de développement Web pour Node.js. Il est utilisé pour créer des applications Web et des API en fournissant une infrastructure robuste pour la gestion des routes, des requêtes et des réponses HTTP.

5.2 Clone the repository:

```
# clone repository from github
$ git clone https://github.com/ZOHSGroupe/
    ZOHR_BlockChain_Security.git\
```

5.3 Accédez au dossier du projet:

```
# go to ZOHR_BlockChain_Security directory
$ cd ZOHR_BlockChain_Security
```

5.4 Install dependencies:

```
# install requirements
$ npm install
```

5.5 Smart Contract Development:

5.5.1 Install Truffle globally (if not installed):

```
# install truffle globally
$ npm install -g truffle
```

5.6 Ganache

5.6.1 Install Ganache (The local BlockChain)

```
$ npm install -g ganache-cli
```

5.7 Express-box

- 5.7.1 Téléchargez la boîte. Cela se charge également d'installer les dépendances nécessaires.

```
$ truffle unbox arvindkalra/express-box
```

5.8 Truffle Development

- 5.8.1 Démarrez la console de développement Truffle en utilisant

```
$ truffle develop
```

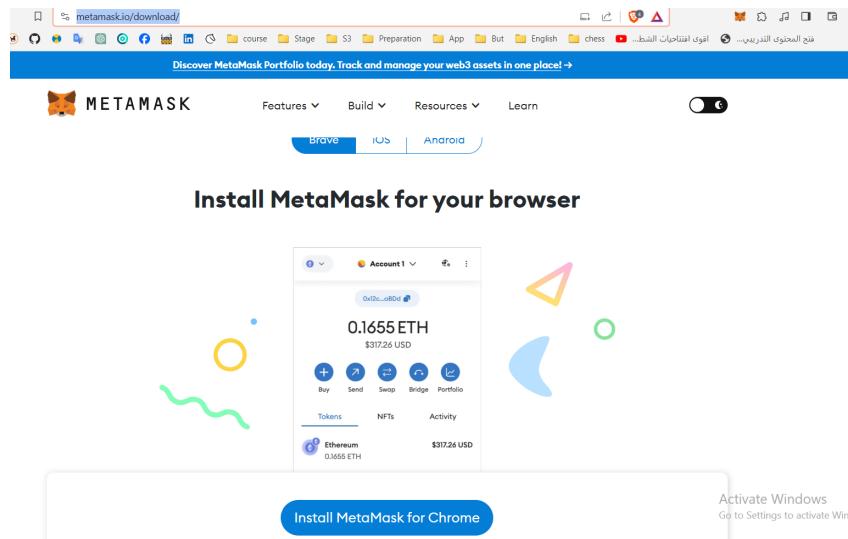
5.9 Deployment

- 5.9.1 Déployez vos contrats intelligents sur un réseau spécifique (remplacez network_name par le réseau souhaité. Allez dans truffleconfig.js, décommentez le réseau et choisissez le nom du réseau)

```
$ truffle migrate --network sepolia
```

5.10 Install Metamask

- 5.10.1 Go to <https://metamask.io/download/> to install



5.10.2 Créez un compte dans Metamask

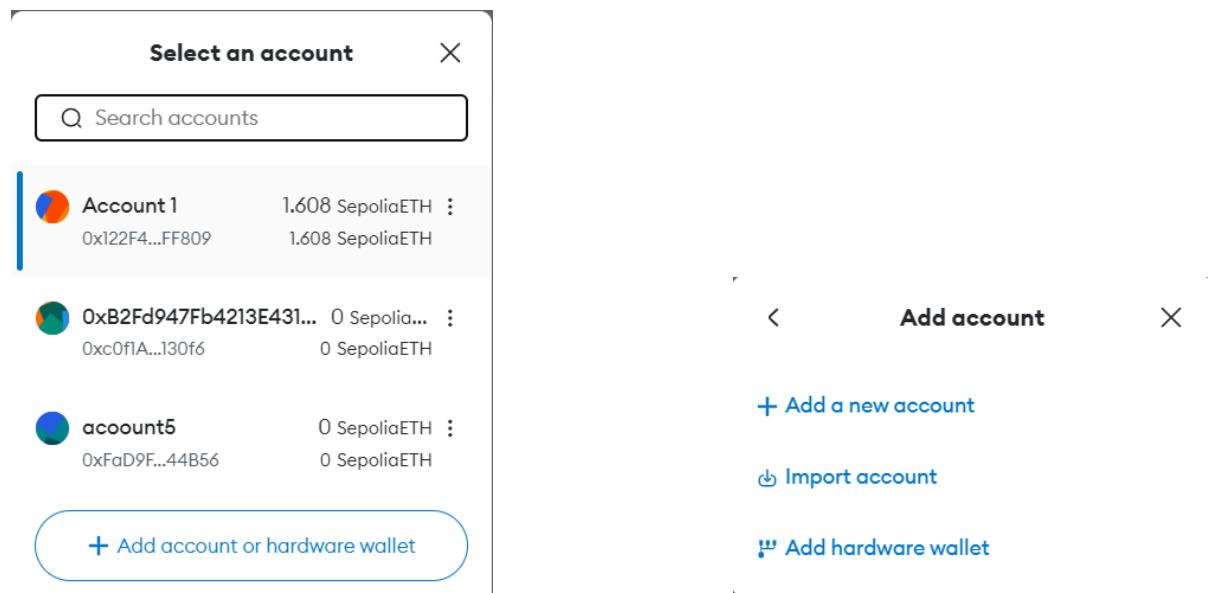


Figure 5.1: Créez un compte dans Metamask

5.11 Run Express Server

5.11.1 Pour exécuter le serveur Express

```
# start application
$ npm start
$ truffle migrate --network sepolia
```

Ouvrez le navigateur : <http://localhost:3000/>

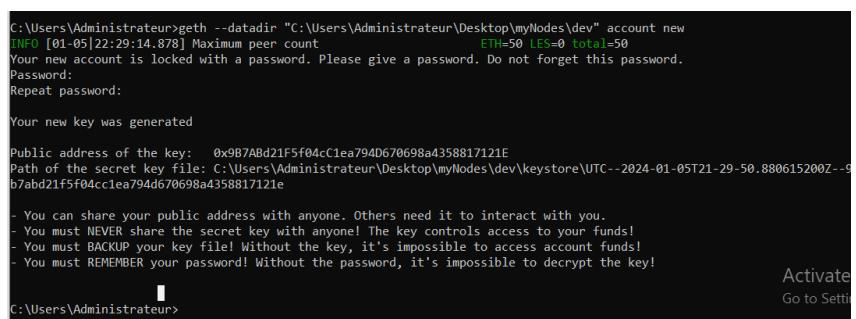
5.12 Install Geth

<https://geth.ethereum.org/downloads>

5.12.1 Geth --version

```
# get version of geth
$ geth --version
```

5.12.2 Créez un nouveau compte

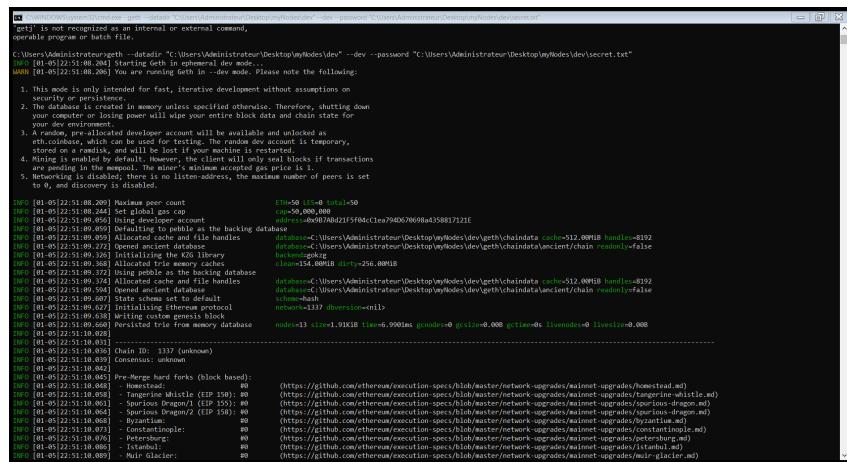


The screenshot shows a terminal window with the following text output:

```
C:\Users\Administrateur>geth --datadir "C:\Users\Administrateur\Desktop\myNodes\dev" account new
INFO [01-05|22:29:14.878] Maximum peer count                                     ETH=50  LES=0  total=50
Your new account is locked with a password. Please give a password. Do not forget this password.
Password:
Repeat password:
Your new key was generated
Public address of the key: 0x9B7Abd21F5f04cC1ea794D670698a4358817121E
Path of the secret key file: C:\Users\Administrateur\Desktop\myNodes\dev\keystore\UTC--2024-01-05T21-29-50.880615200Z--9
b7abd21f5f04cc1ea794d670698a4358817121E
- You can share your public address with anyone. Others need it to interact with you.
- You must NEVER share the secret key with anyone! The key controls access to your funds!
- You must BACKUP your key file! Without the key, it's impossible to access account funds!
- You must REMEMBER your password! Without the password, it's impossible to decrypt the key!
```

At the bottom right of the terminal window, there are two small buttons: "Activate" and "Go to Settings".

5.12.3 Exécutez Geth en mode développement



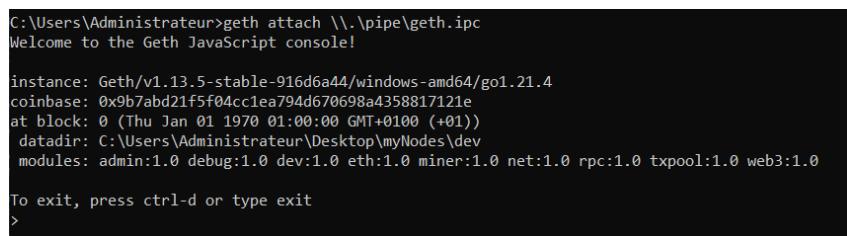
```
[1] geth is not recognized as an internal or external command,
         or operable program or batch file.

C:\Users\Administrateur>geth -datadir "C:\Users\Administrateur\Desktop\myNodes\dev" --dev --password "C:\Users\Administrateur\Desktop\myNodes\dev\secret.txt"

[INFO] [01-05-22:51:08.204] Starting Geth in ephemeral dev mode...
[WARN] [01-05-22:51:08.206] You are running Geth in ...-dev mode. Please note the following:
1. This mode is only intended for fast, iterative development without assumptions on
   every part of the system.
2. The database is created in memory unless specified otherwise. Therefore, shutting down
   your computer or losing power will wipe your entire block data and chain state for
   you.
3. A random, pre-allocated developer account will be available and unlocked as
   eth.accounts[0], which can be used for testing. The random dev account is temporary,
   temporary.
4. Mining is enabled by default. However, the client will only seal blocks if transactions
   are pending. Block rewards and miner's minimum accepted gas price is 0.
5. Networking is disabled; there is no listen-address, the maximum number of peers is set
   to 0, and discovery is disabled.

[INFO] [01-05-22:51:08.209] Maximum peer count:      50
[INFO] [01-05-22:51:08.264] Set global gas cap:    cap=50,000,000
[INFO] [01-05-22:51:08.265] Set global gas account: 0x0000000000000000000000000000000000000000
[INFO] [01-05-22:51:09.459] Defaulting to pebble as the backing database
[INFO] [01-05-22:51:09.460] Using C:\Users\Administrateur\Desktop\myNodes\dev\geth\chaindata as the chain directory
[INFO] [01-05-22:51:09.461] Opened ancient database
[INFO] [01-05-22:51:09.462] Initializing the X64 library
[INFO] [01-05-22:51:09.463] Backend: go-ethereum
[INFO] [01-05-22:51:09.464] Client version: v1.13.5-stable-916d6a44/windows-amd64/go1.21.4
[INFO] [01-05-22:51:09.465] Backend: go-ethereum
[INFO] [01-05-22:51:09.466] Using pebble as the backing database
[INFO] [01-05-22:51:09.467] Allocated cache and file handles
[INFO] [01-05-22:51:09.468] Cache handles: 1024
[INFO] [01-05-22:51:09.469] File handles: 1024
[INFO] [01-05-22:51:09.470] State schema set to default
[INFO] [01-05-22:51:09.471] Cache schema: <none>
[INFO] [01-05-22:51:09.472] Backend schema: <none>
[INFO] [01-05-22:51:09.473] Writing custom consensus block
[INFO] [01-05-22:51:09.480] Permitted trie from memory database
[INFO] [01-05-22:51:09.481] Cache size: 1.91KiB time=6.990ms pending=0 entries=0,000 getmisses=0, livelocks=0,000
[INFO] [01-05-22:51:09.482] Chain ID: 1337 (unknown)
[INFO] [01-05-22:51:09.483] Configuration loaded
[INFO] [01-05-22:51:09.484] Pre-Merge hard forks (block based):
[INFO] [01-05-22:51:09.485] - Homestead:          @0          (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/homestead.md)
[INFO] [01-05-22:51:09.486] - Tangerine Whistle (EIP 150) @0          (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/tangerine-whistle.md)
[INFO] [01-05-22:51:09.487] - Spurious Dragon (EIP 158) @0          (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/spurious-dragon.md)
[INFO] [01-05-22:51:09.488] - Byzantium (EIP 105) @0          (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/byzantium.md)
[INFO] [01-05-22:51:09.489] - Constantinople:          @0          (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/constantinople.md)
[INFO] [01-05-22:51:09.490] - Petersburg:          @0          (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/petersburg.md)
[INFO] [01-05-22:51:09.491] - London:          @0          (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/london.md)
[INFO] [01-05-22:51:09.492] - Muir Glacier:          @0          (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/muir-gacier.md)
```

5.12.4 Exécutez Geth console JavaScript



```
C:\Users\Administrateur>geth attach \\.\pipe\geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.13.5-stable-916d6a44/windows-amd64/go1.21.4
coinbase: 0xb7abd21f5f04cc1ea794d670698a4358817121e
at block: 0 (Thu Jan 01 1970 01:00:00 GMT+100 (+01))
datadir: C:\Users\Administrateur\Desktop\myNodes\dev
modules: admin:1.0 debug:1.0 dev:1.0 eth:1.0 miner:1.0 net:1.0 rpc:1.0 txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
>
```

5.12.5 Configurer le nœud Ethereum Sepolia

5.12.6 Connectez-vous au réseau Sepolia

```
C:\Users\Administrateur>Geth attach \\.\pipe\geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.13.5-stable-916d6a44/windows-amd64/go1.21.4
at block: 0 (Sun Oct 03 2021 14:24:41 GMT+0100 (+01))
datadir: C:\Users\Administrateur\Desktop\myNodes\depolia
modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 miner:1.0 net:1.0 rpc:1.0 txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
>
```

5.12.7 Téléchargez le client de consensus

<https://geth.ethereum.org/docs/getting-started/consensus-clients>
<https://docs.prylabs.network/docs/install/install-with-script>

```
C:\Users\Administrateur\Desktop\myNodes\depolia\consensus>curl https://raw.githubusercontent.com/prysmaticlabs/prysm/master/prysm.bat --output prysm.bat
% Total    % Received   % Xferd  Average Speed   Time     Time      Time  Current
                                         Dload  Upload   Total   Spent    Left  Speed
100  8743  100  8743    0      0  10831      0  --:--:--  --:--:--  10860

C:\Users\Administrateur\Desktop\myNodes\depolia\consensus>reg add HKCU\Console /v VirtualTerminalLevel /t REG_DWORD /d 1
The operation completed successfully.
```

```
C:\Users\Administrateur>eth --datadir "C:\Users\Administrateur\Desktop\myNodes\depolia" --sepolia account new
[...]
Your new account is locked with a password. Please give a password. Do not forget this password.
Password:
Repeat password:
Your new key was generated
Public address of the key: 0x82Fd947Fb4213E431005D729813949e39b3d5802
Path of the secret key file: C:\Users\Administrateur\Desktop\myNodes\depolia\keystore\UTC--2024-01-06T00:30:49_2978664007--b2fd947Fb4213E431005D729813949e39b3d5802
- You can share your public address with anyone. Others need it to interact with you.
- You must NEVER share the secret key with anyone! The key controls access to your funds!
- You must BACKUP your key file! Without the key, it's impossible to access account funds!
- You must REMEMBER your password! Without the password, it's impossible to decrypt the key!
[...]
C:\Users\Administrateur>
```

5.12.8 Créez un compte dans Sepolia

5.12.9 Téléchargez le bloc 1 de Sepolia

Téléchargez l'état initial de genesis de Sepolia depuis Github dans votre répertoire de consensus. Ensuite, utilisez la commande suivante pour démarrer un noeud de balise qui se connecte à votre noeud d'exécution local en remplaçant `CHEMIN_VERS_LE_FICHIER_IPC` par le chemin vers le fichier IPC que le client d'exécution a créé pour vous lors de l'étape précédente : Téléchargez l'état initial de genesis de Sepolia depuis Github dans votre répertoire de consensus.

5.12.10 Assurez-vous de modifier "adresse du compte de Sepolia"

```
geth --datadir "C:\Users\Administrateur\Desktop\myNodes\depolia" --sepolia --http -http.api eth,net,web3,admin
```

```
[...]
Starting Prysm beacon-chain - execution endpoint=http://localhost:8551 --sepolia --suggested-fee-recipient=0xb2Fd947Fb4213E431005D729813949e39b3d5802 --jet-secret="C:\Users\Administrateur\Desktop\myNodes\depolia\jetsecret" --genesis-state=genesis.ssz --checkpoint-sync-url=https://sepolia.beaconstate.info --genesis-beacon-api-url=https://sepolia.beaconstate.info
Latest prysm release is v4.1.1.
Using prysm version v4.1.1.
[...]
Verifying binary authenticity with SHA256 hash.
Starting Prysm beacon-chain - execution endpoint=http://localhost:8551 --sepolia --suggested-fee-recipient=0xb2Fd947Fb4213E431005D729813949e39b3d5802 --jet-secret="C:\Users\Administrateur\Desktop\myNodes\depolia\jetsecret" --genesis-state=genesis.ssz --checkpoint-sync-url=https://sepolia.beaconstate.info --genesis-beacon-api-url=https://sepolia.beaconstate.info
[...]
Prismatic Labs Terms of Use
By downloading, accessing or using the Prysm implementation ("Prysm"), you (referred herein as "you" or the "user") certify that you have read and agreed to the terms and conditions below.
TERMS AND CONDITIONS: https://github.com/prysmaticlabs/prysm/blob/master/TERMS_OF_SERVICE.md
Type "accept" to accept this terms and conditions [accept/decline]:>(0) ([92mdefault[0m: decline):
accept
[0m[2024-01-06 01:41:19] [0m+12m INFO+0m Finished reading MAT secret from C:\Users\Administrateur\Desktop\myNodes\depolia\geth\jetsecret
[0m[2024-01-06 01:41:19] [0m+13m WARNING+0m [0m flags:[0m Running on the Sepolia Beacon Chain Testnet
[0m[2024-01-06 01:41:19] [0m+13m WARNING+0m Fee recipient 0xb2Fd947Fb4213E431005D729813949e39b3d5802 will be used as the fee recipient. We recommend using a checksummed address (checksum) to prevent spelling mistakes in your fee recipient Ethereum address. The checksummed address is 0x82Fd947Fb4213E431005D729813949e39b3d5802.
[0m[2024-01-06 01:41:19] [0m+12m INFO+0m [0m node:<0m Default fee recipient is set to 0xb2Fd947Fb4213E431005D729813949e39b3d5802, recipient may be overwritten from validator client and persist in db. Default fee recipient is now 0xb2Fd947Fb4213E431005D729813949e39b3d5802.
[0m[2024-01-06 01:41:19] [0m+12m INFO+0m [0m node:<0m Checking [0m database-path=[0mC:\Users\Administrateur\AppData\Local\ETH2\beaconchain\data
[0m[2024-01-06 01:41:19] [0m+12m INFO+0m [0m db:<0m Opening Bolt DB at C:\Users\Administrateur\AppData\Local\ETH2\beaconchain\data
[0m[2024-01-06 01:41:19] [0m+12m INFO+0m Downloaded checkpoint sync state and blocks. +1296block root=0xa99fb6018c7d157eb2d8fc329c950a5f6da1135151400bc5082c7b2ad29199d +13mblock .last+0m+406390d +12m
[0m[2024-01-06 01:41:26] [0m+12m INFO+0m Downloaded checkpoint sync state and blocks. +1296block root=0xa99fb6018c7d157eb2d8fc329c950a5f6da1135151400bc5082c7b2ad29199d +13mblock .last+0m+406390d +12m
[...]
```

5.12.11 Configurer Rinkeby

```
geth --rpc --rpcauth="personal,eth,net,web3,net" --ipcpath
"~/Library/Ethereum/geth.ipc"
```

```
(C) Microsoft Corporation. All rights reserved.  
C:\Users\ADMINI~1\AppData\Local\Temp\cmd-111111\geth\rpcapi\geth\rpcapi.go:11  
Incorrect Usage: flag provided but not defined: -rpc  
  
NAME:  
  geth - the go-ethereum command line interface  
  
USAGE:  
  geth [global options] command [command options] [arguments...]  
  
VERSION:  
  1.13.5-stable-916d8a44  
  
COMMANDS:  
  account          Manage accounts  
  attach           Start an interactive JavaScript environment (connect to node)  
  console          Start an interactive JavaScript environment  
  ls               Low level database operations  
  dump             Dump a specific block from storage  
  dumpconfig      Dump configuration for the current EVM, VM, and storage  
  dumpgenesis     Dumps genesis block JSON configuration to stdout  
  export           Export blockchain into file  
  export-primitives Export blockchain into a stream into an RLP stream  
  import           Import a blockchain file  
  import-raw       Import a blockchain file or raw blocks from an RLP stream  
  init             Bootstrap and initialize a new genesis block  
  j2c              (DEPRECATED) Execute the specified Javascript files  
  resnode          Remove blockchain and state databases  
  show-deprecated-flags Show flags that have been deprecated  
  snapshot         Create a full database backup  
  verkle           A set of experimental verkle tree management commands  
  version          Prints the current Go version  
  version-check   Checks (online) for known Geth security vulnerabilities  
  web3             Web3 API interface  
  help, h          Shows a list of commands or Help for one command  
  
GLOBAL OPTIONS:  
  --allow-insecure-unlock (default: false) (GETH_ALLOW_INSECURE_UNLOCK)  
    Allow insecure account unlocking when account-related RPCs are exposed by http  
  --keystore-path= (default: .keystores) (GETH_KEYSTORES)  
    Directory for the keystore (default = inside the datadir)  
  --lightweight (default: false) (GETH_LIGHTWEIGHT)  
    Reduce key-derivation RAM & CPU usage at some expense of KDF strength
```

5.13 Créez un compte sur Infura

5.13.1 Créez une clé API

```
require('babel/register');
require('dotenv').config()
const HDWalletProvider = require("@truffle/hdwallet-provider")

const private_keys = [
  process.env.PRIVATE_KEY_0, // Assuming these are already strings
  process.env.PRIVATE_KEY_1,
];

module.exports = {
  networks: {
    ZiadNetwerk: {
      host: '127.0.0.1',
      port: 7545,
      network_id: '*',
    },
    sepolia: {
      provider: () => new HDWalletProvider({
        privateKeys: private_keys,
        provider: `https://sepolia.infura.io/v2/${process.env.INFURA_ID}&${process.env.INFURA_SECRET}`,
        numberofAddress: 2
      }),
      network_id: 11155111, // replace with the actual network id
      gas: 5000000,
      gasPrice: 123699748701,
      [b],
    },
    compilers: {
      solc: {
        version: "0.5.16", // Change this to the desired Solidity version
        settings: {
          optimizer: {
            enabled: true,
            runs: 200,
          },
        },
      },
    },
  }
};
```

5.13.2 Exécutez truffle migrate –network sepolia

```

  Testing
    * Contract: Metadata --> Library: ConvertLib
      Deploying "Metadata"
      + block number: 500073
      + block timestamp: 1505500000000
      + balance: 0.000000000000000
      + gas limit: 0.000000000000000
      + gas price: 0.00000000 wei
      + total cost: 0.000000000000000

      + total cost: 0.000000000000000 eth

  Deploy: AssuranceContext

  Deploying "AssuranceContext"
    + block number: 500073
    + block timestamp: 1505500000000
    + account: 0x1234567890123456789012345678901234567890
    + balance: 0.000000000000000
    + gas limit: 0.000000000000000
    + gas price: 0.00000000 wei
    + total cost: 0.000000000000000

    + total cost: 0.000000000000000 eth

  Summary
  + Total deployments: 4
  + Total cost: 0.000000000000000 eth

  Starting application...
  + Network name: "sepolia"
  + Chain ID: 10
  + Block gas limit: 30000000 (0x1D3800)

```

```

laptop: ~ % truffle migrate --network sepolia
Deploying 'Reputation'
+ Block 0: Sealed: 3
+ Block 1: Sealed: 4
+ Block 2: Sealed: 5
+ Block 3: Sealed: 6
+ Block 4: Sealed: 7
+ Block 5: Sealed: 8
+ Block 6: Sealed: 9
+ Block 7: Sealed: 10
+ Block 8: Sealed: 11
+ Block 9: Sealed: 12
+ Block 10: Sealed: 13
+ Block 11: Sealed: 14
+ Block 12: Sealed: 15
+ Block 13: Sealed: 16
+ Block 14: Sealed: 17
+ Block 15: Sealed: 18
+ Block 16: Sealed: 19
+ Saving artifacts to ./artifacts
+ Saving artifacts
+ Total cost: 0.0000000000000000 ETH

laptop: ~ % truffle migrate --network sepolia
Deploying 'Convertible'
+ Block 0: Sealed: 10
+ Block 1: Sealed: 11
+ Block 2: Sealed: 12
+ Block 3: Sealed: 13
+ Block 4: Sealed: 14
+ Block 5: Sealed: 15
+ Block 6: Sealed: 16
+ Block 7: Sealed: 17
+ Block 8: Sealed: 18
+ Block 9: Sealed: 19
+ Block 10: Sealed: 20
+ Block 11: Sealed: 21
+ Block 12: Sealed: 22
+ Block 13: Sealed: 23
+ Block 14: Sealed: 24
+ Block 15: Sealed: 25
+ Block 16: Sealed: 26
+ Block 17: Sealed: 27
+ Block 18: Sealed: 28
+ Block 19: Sealed: 29
+ Block 20: Sealed: 30
+ Block 21: Sealed: 31
+ Block 22: Sealed: 32
+ Block 23: Sealed: 33
+ Block 24: Sealed: 34
+ Block 25: Sealed: 35
+ Block 26: Sealed: 36
+ Block 27: Sealed: 37
+ Block 28: Sealed: 38
+ Block 29: Sealed: 39
+ Block 30: Sealed: 40
+ Saving artifacts to ./artifacts
+ Saving artifacts
+ Total cost: 0.0000000000000000 ETH

laptop: ~ % truffle migrate --network sepolia
Deploying 'Multichain'
+ Block 0: Sealed: 41
+ Block 1: Sealed: 42
+ Block 2: Sealed: 43
+ Block 3: Sealed: 44
+ Block 4: Sealed: 45
+ Block 5: Sealed: 46
+ Block 6: Sealed: 47
+ Block 7: Sealed: 48
+ Block 8: Sealed: 49
+ Block 9: Sealed: 50
+ Block 10: Sealed: 51
+ Block 11: Sealed: 52
+ Block 12: Sealed: 53
+ Block 13: Sealed: 54
+ Block 14: Sealed: 55
+ Block 15: Sealed: 56
+ Block 16: Sealed: 57
+ Block 17: Sealed: 58
+ Block 18: Sealed: 59
+ Block 19: Sealed: 60
+ Block 20: Sealed: 61
+ Block 21: Sealed: 62
+ Block 22: Sealed: 63
+ Block 23: Sealed: 64
+ Block 24: Sealed: 65
+ Block 25: Sealed: 66
+ Block 26: Sealed: 67
+ Block 27: Sealed: 68
+ Block 28: Sealed: 69
+ Block 29: Sealed: 70
+ Block 30: Sealed: 71
+ Saving artifacts to ./artifacts
+ Saving artifacts
+ Total cost: 0.0000000000000000 ETH

laptop: ~ % truffle migrate --network sepolia
Deploying 'MultichainInfo'
+ Block 0: Sealed: 72
+ Block 1: Sealed: 73
+ Block 2: Sealed: 74
+ Block 3: Sealed: 75
+ Block 4: Sealed: 76
+ Block 5: Sealed: 77
+ Block 6: Sealed: 78
+ Block 7: Sealed: 79
+ Block 8: Sealed: 80
+ Block 9: Sealed: 81
+ Block 10: Sealed: 82
+ Block 11: Sealed: 83
+ Block 12: Sealed: 84
+ Block 13: Sealed: 85
+ Block 14: Sealed: 86
+ Block 15: Sealed: 87
+ Block 16: Sealed: 88
+ Block 17: Sealed: 89
+ Block 18: Sealed: 90
+ Block 19: Sealed: 91
+ Block 20: Sealed: 92
+ Block 21: Sealed: 93
+ Block 22: Sealed: 94
+ Block 23: Sealed: 95
+ Block 24: Sealed: 96
+ Block 25: Sealed: 97
+ Block 26: Sealed: 98
+ Block 27: Sealed: 99
+ Block 28: Sealed: 100
+ Block 29: Sealed: 101
+ Block 30: Sealed: 102
+ Saving artifacts to ./artifacts
+ Saving artifacts
+ Total cost: 0.0000000000000000 ETH

laptop: ~ % truffle migrate --network sepolia
Deploying 'Multichain'
+ Block 0: Sealed: 103
+ Block 1: Sealed: 104
+ Block 2: Sealed: 105
+ Block 3: Sealed: 106
+ Block 4: Sealed: 107
+ Block 5: Sealed: 108
+ Block 6: Sealed: 109
+ Block 7: Sealed: 110
+ Block 8: Sealed: 111
+ Block 9: Sealed: 112
+ Block 10: Sealed: 113
+ Block 11: Sealed: 114
+ Block 12: Sealed: 115
+ Block 13: Sealed: 116
+ Block 14: Sealed: 117
+ Block 15: Sealed: 118
+ Block 16: Sealed: 119
+ Block 17: Sealed: 120
+ Block 18: Sealed: 121
+ Block 19: Sealed: 122
+ Block 20: Sealed: 123
+ Block 21: Sealed: 124
+ Block 22: Sealed: 125
+ Block 23: Sealed: 126
+ Block 24: Sealed: 127
+ Block 25: Sealed: 128
+ Block 26: Sealed: 129
+ Block 27: Sealed: 130
+ Block 28: Sealed: 131
+ Block 29: Sealed: 132
+ Block 30: Sealed: 133
+ Saving artifacts to ./artifacts
+ Saving artifacts
+ Total cost: 0.0000000000000000 ETH

```

5.13.3 rechercher l'adresse du contrat sur sepolia.etherscan

The screenshot shows the Etherscan interface for the Multichain contract. The transaction details are as follows:

Transaction Hash	Method	Block	Age	From	To	Value	Tx Fee
0x54a6873c47513b98...	0x60806040	5108793	2 mins ago	0x122F48..E2eFF809	Contract Creation	0 ETH	0.08335944

DevOps

The screenshot shows a digital workspace interface. On the left, there is a sidebar titled "Members" with four entries:

- Ouail Laamiri (LaamiriOuail)
- SadikHajar (SadikHajar)
- SamirZiani (samir5636)
- Ziad Ben Saada (ziadbensaada)

On the right, there is a section titled "Select all" with two projects listed:

- Machine-Learning & Blockchain**: "this team for develope a recommendation system and a blockchain". It includes icons for a brain, a dollar sign, and a gear.
- Micro-service & DevOps**: "this team for create all micro-service and use devOps tool for contalirazion ...". It includes icons for a person, a gear, and a globe.

6.1 Méthodologie DevOps

6.1.1 Explication du concept DevOps

Le concept DevOps est une approche de développement logiciel qui vise à combler le fossé entre les équipes de développement (Dev) et les équipes d'exploitation (Ops). Traditionnellement, ces deux équipes opéraient de manière séparée, ce qui pouvait entraîner des retards, des erreurs de communication et des inefficacités dans le processus de livraison logicielle. DevOps promeut une collaboration étroite entre ces équipes tout au long du cycle de vie du développement, de la livraison et de la maintenance des logiciels.

Dans le cadre du projet ZOHS d'assurance automobile, l'adoption de pratiques DevOps est essentielle pour plusieurs raisons :

6.1.2 Importance de DevOps dans le cycle de développement logiciel

- **Amélioration de la qualité du logiciel :** En intégrant la phase de test dès les premières étapes du développement, les erreurs et les bogues sont détectés plus tôt, ce qui permet d'améliorer la qualité globale du logiciel.
- **Livraisons plus rapides :** L'automatisation des processus de construction, de test et de déploiement permet de réduire les délais entre le développement d'une fonctionnalité et sa mise en production, ce qui permet à l'agence ZOHS de répondre plus rapidement aux besoins changeants du marché.
- **Stabilité opérationnelle :** En adoptant des pratiques DevOps telles que la surveillance continue et la gestion proactive des incidents, ZOHS peut garantir une stabilité opérationnelle optimale pour son application d'assurance automobile, minimisant ainsi les temps d'arrêt et les impacts négatifs sur les utilisateurs.
- **Scalabilité et flexibilité :** Grâce à l'automatisation de l'infrastructure avec des outils comme Kubernetes, l'agence ZOHS peut facilement adapter ses ressources informatiques à la demande, garantissant ainsi une expérience utilisateur optimale même en cas de pic de trafic.

6.1.3 Objectifs spécifiques de l'implémentation de pratiques DevOps dans le projet ZOHS

- **Automatisation des processus :** Automatiser les processus de construction, de test et de déploiement pour accélérer la livraison des fonctionnalités tout en garantissant la fiabilité du logiciel.
- **Collaboration et communication accrues :** Favoriser une culture de collaboration entre les équipes de développement, d'exploitation et de qualité pour garantir une livraison fluide et sans friction des nouvelles fonctionnalités.
- **Surveillance et gestion des performances :** Mettre en place des outils de surveillance et de gestion des performances pour assurer la disponibilité, la fiabilité et la performance optimale de l'application d'assurance automobile de ZOHS.
- **Réduction des délais de déploiement :** Réduire les délais entre le développement et la mise en production en automatisant les processus de déploiement et en adoptant des pratiques de déploiement continu.

6.2 Outils et Technologies

6.2.1 Git

Système de contrôle de version distribué utilisé pour la gestion du code source. Git permet aux développeurs de suivre les modifications apportées au code, de travailler sur des branches de manière collaborative et de fusionner les modifications en toute sécurité.

6.2.2 GitHub

Plateforme de développement collaboratif basée sur Git pour l'hébergement de code source et la gestion de projet. GitHub offre des fonctionnalités telles que le suivi des problèmes, la gestion des versions, le contrôle d'accès et la collaboration sociale pour les équipes de développement.

6.2.3 Docker :

Plateforme de conteneurisation utilisée pour encapsuler et distribuer des applications avec toutes leurs dépendances. Docker permet aux développeurs de créer des environnements d'exécution isolés et portables, ce qui facilite le déploiement et la gestion des applications dans différents environnements.

6.2.4 Kubernetes :

Système open-source de gestion d'orchestration de conteneurs pour automatiser le déploiement, la mise à l'échelle et la gestion des applications conteneurisées. Kubernetes fournit des fonctionnalités telles que la découverte de services, la gestion des ressources et la tolérance aux pannes pour garantir la disponibilité et la fiabilité des applications.

6.2.5 Kafka :

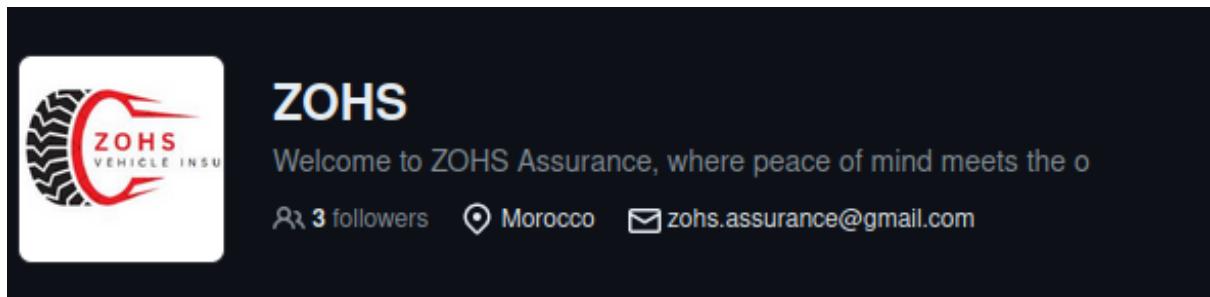
Plateforme de diffusion de messages distribuée utilisée pour la gestion des flux de données en temps réel. Kafka offre une architecture distribuée et hautement évolutive pour la diffusion de messages entre les applications, ce qui permet de gérer efficacement les flux de données à grande échelle.

6.2.6 Jenkins :

Outil d'intégration continue et de déploiement continu (CI/CD) utilisé pour automatiser le processus de construction, de test et de déploiement des applications. Jenkins permet

aux développeurs de définir des pipelines d'intégration continue pour automatiser les tâches répétitives et garantir la qualité du code à chaque étape du cycle de développement.

6.3 Implémentation DevOps dans le projet



6.3.1 Méthodologies utilisées pour intégrer les pratiques DevOps dans le cycle de développement

- **Flux de travail Git :** Organisation de branches, gestion des versions, processus de fusion, etc. Git est utilisé comme système de contrôle de version pour gérer efficacement le code source du projet. Les pratiques de gestion des branches telles que la création de branches de fonctionnalités, de branches de version et de branches de correctifs sont utilisées pour organiser le flux de travail et faciliter la collaboration entre les membres de l'équipe de développement.
- **Utilisation de Docker :** Docker est utilisé pour la conteneurisation des applications et la gestion des environnements de développement, de test et de production. Les conteneurs Docker offrent un moyen cohérent et portable de distribuer des applications avec toutes leurs dépendances, ce qui facilite le déploiement et la gestion des environnements logiciels.
- **Déploiement et gestion des conteneurs avec Kubernetes :** Kubernetes est utilisé pour automatiser le déploiement, la mise à l'échelle et la gestion des applications conteneurisées. Kubernetes offre des fonctionnalités telles que la découverte de services, la répartition de la charge, la gestion des ressources et la tolérance aux pannes, ce qui garantit la scalabilité et la haute disponibilité de l'application.
- **Utilisation de Kafka :** Kafka est utilisé pour la gestion des événements et la communication entre les différents composants de l'application. Kafka offre une architecture de diffusion de messages distribuée et hautement évolutive, ce qui permet

de gérer efficacement les flux de données en temps réel entre les différents modules de l'application.

- **Intégration de Jenkins :** Jenkins est utilisé pour automatiser les tests, les builds et les déploiements continus. Des pipelines d'intégration continue sont définis dans Jenkins pour automatiser les tâches répétitives telles que la compilation du code source, l'exécution des tests unitaires et fonctionnels, la génération de rapports de test et le déploiement des artefacts vers les environnements de développement, de test et de production.

Reference

7.1 Github Repository

- Frontend : <https://github.com/ZOHSGroupe/Front-End-ZOHS>
- Blockchain : https://github.com/ZOHSGroupe/BlockChain_Security
- Machine Learning : <https://github.com/ZOHSGroupe/ML-system-recommendation>
- Backend :
 - Client Service : <https://github.com/ZOHSGroupe/Client-Service>
 - Assurance Service : <https://github.com/ZOHSGroupe/Login-SignUp-Service>
 - Vehicule Service : <https://github.com/ZOHSGroupe/Vihecule-Service>
 - Contract Service : <https://github.com/ZOHSGroupe/Contract-Service>
 - Driver License Service : <https://github.com/ZOHSGroupe/DriverLicense-Service>
 - Email Service : <https://github.com/ZOHSGroupe/Email-Service>
 - File Service : <https://github.com/ZOHSGroupe/File-Service>
 - Link Service : <https://github.com/ZOHSGroupe/Link-Service>
 - Discovery Service : https://github.com/ZOHSGroupe/service_discovery
 - Gateway Service : https://github.com/ZOHSGroupe/server_getway

7.2 DevOps

- Docker : <https://www.docker.com/>
- jenkins : <https://www.jenkins.io/>
- Git : <https://git-scm.com/>

- Github : <https://github.com/>
- Kubernets : <https://kubernetes.io/>
- Minikube : <https://minikube.sigs.k8s.io/docs/>
- Kubeflow : <https://www.kubeflow.org/>
- Helm : <https://helm.sh/>

7.3 Developpment Web

- Python : <https://www.python.org/>
- TypeScript : <https://www.typescriptlang.org/>
- JavaScript : <https://developer.mozilla.org/fr/docs/Learn/JavaScript>
- Java : <https://docs.oracle.com/en/java/>
- GoLang : <https://go.dev/>
- Apache Kafka : <https://kafka.apache.org/>
- flask : <https://flask.palletsprojects.com/en/3.0.x/>
- Fast API : <https://fastapi.tiangolo.com/>
- Gin : <https://gin-gonic.com/>
- Spring Boot : <https://spring.io/projects/spring-boot/>
- Express JS : <https://expressjs.com/>
- Nest JS : <https://nestjs.com/>
- TypeORM : <https://typeorm.io/>
- Gorm : <https://gorm.io/>
- SQLAlchemy : <https://www.sqlalchemy.org/>
- Hibernate : <https://hibernate.org/>
- Angular JS : <https://angular.dev/>
- MySQL : <https://www.mysql.com/>

- PostgreSql : <https://www.postgresql.org/>
- MongoDB : <https://www.mongodb.com/>

7.4 Machine Learning

- Python : <https://www.python.org/>
- Kubeflow : <https://www.kubeflow.org/>
- Airflow : <https://airflow.apache.org/>
- Apache Kafka : <https://kafka.apache.org/>
- Flask : <https://flask.palletsprojects.com/en/3.0.x/>
- PostgreSql : <https://www.postgresql.org/>
- Sklearn : <https://scikit-learn.org/stable/index.html>
- CassandraDB : https://cassandra.apache.org/_/index.html
- Apache Spark : <https://spark.apache.org/>

7.5 Blockchain

- Rinkeby : <https://blockexplorer.one/ethereum/rinkeby>
- Infura : <https://www.infura.io/>
- Truffle : <https://trufflesuite.com/>
- Sepolia : <https://sepoliafaucet.com/>
- Geth : <https://geth.ethereum.org/>
- Express JS : <https://expressjs.com/>