

Projet Fin De Module:

Développement et réalisation d'une application microservices d'assurance basé sur la technologie de blockchain

Réalisé par :

Mohcine BOUDENJAL
Said FRIKH
Hind EL OUAHABI
Mohammed DAOUAN

Encadré par :

Pr. EL AACHAK Lotfi

Table de matière

Table de matière.....	2
Chapitre 1:Conception et Développement.....	5
Introduction:.....	5
1. Conception:.....	5
2. Technologies Utilisées :.....	6
2.1. Spring Boot :.....	6
2.2. Angular :.....	7
2.3. MongoDB :.....	7
2.4. Architecture Microservices :.....	8
2.5 Kafka et RestTemplate :.....	10
3. L'architecture Globale :.....	11
4. Structure de projet :.....	11
5. Les outils :.....	13
5.1.Swagger:.....	13
5.2.Eureka Server:.....	13
6. Les interfaces :.....	14
6.1. Partie Client :.....	14
6.2. Partie Administrateur :.....	18
6.2.1.Employee:.....	18
6.2.2.Admin:.....	20
Chapitre 2: La technologie Blockchain.....	22
Introduction:.....	22
1. Blockchain: Fonctionnement et Principes.....	22
2. Architecture Web Décentralisée:.....	23
3. Technologies Utilisées dans le Projet:.....	25
1. Ethereum:	25
2. Solidity:	25
3. Hardhat:	25
4. MetaMask:	26
5. Etherscan:	27
4. Structure du Projet avec Hardhat :.....	27
5. Processus de Transactions et Confirmation avec MetaMask:.....	28
❖ Préparation des Données:.....	29
❖ Création de la Transaction:.....	29
❖ Envoi de la Transaction:.....	29
❖ Confirmation avec MetaMask:.....	29
❖ Vérification via Etherscan:.....	30
6. Consultation de l'Historique des Transactions par les Utilisateurs:.....	30
Conclusion:.....	31
Chapitre 3:DevOps.....	32
Introduction:.....	32
Choix du cloud:.....	33
Les technologies DevOps mise en oeuvre:.....	34
Conception de l'infrastructure:.....	34

Utilisation de terraform:.....	35
CI / CD:.....	35
Pipeline Jenkins:.....	36
Serveur production:.....	38
Chapitre 4:Système de recommandation.....	40
1. Introduction.....	40
1.1. Contexte du Projet.....	40
1.2. Objectif Global.....	40
2. Technologies Utilisées.....	40
2.1 Beautiful Soup.....	40
2.2 Kafka.....	41
2.3 Scikit-Learn.....	41
2.4 Cassandra.....	41
2.5 Airflow.....	42
3. Web Scraping.....	42
3.1 Sources de Données.....	42
3.2 Processus de Scrapping.....	43
3.3 Assurance Qualité des Données.....	43
3.4 Exemple des données collectées via le web scraping.....	43
4. Modélisation du Système de Recommandation.....	44
4.1. Prétraitement des Données.....	44
4.2. Visualisation des Données.....	45
4.3. Entraînement du Modèle.....	47
4.3.1 Choix de l'Algorithme.....	47
4.3.2 collaborative filtering.....	48
4.3.4 Évaluation du Modèle.....	48
5. Communication entre Microservices avec Kafka: Mise en Place d'une Pipeline de Recommandation.....	49
1. Producer - Extraction des Features (features topic) :.....	49
2. Kafka Broker :.....	50
3. Consumer - Génération de Recommandations (recommendation topic) :.....	50
4. Consumer Final - Distribution des Recommandations :.....	50
6. Création d'un Pipeline Hebdomadaire avec Airflow.....	51
1. Tâche Cassandra - Gestion du Dataset :.....	51
2. Tâche Python - Prétraitement du Dataset :.....	52
3. Tâche Python - Entraînement du Modèle :.....	52
4. Tâche Python - Évaluation du Modèle :.....	52
5. Tâche Python - Sauvegarde du Modèle (Si Performant) :.....	52
6. Tâche Python - Élimination du Modèle (Sinon) :.....	52
Conclusion.....	53

Chapitre 1:Conception et Développement

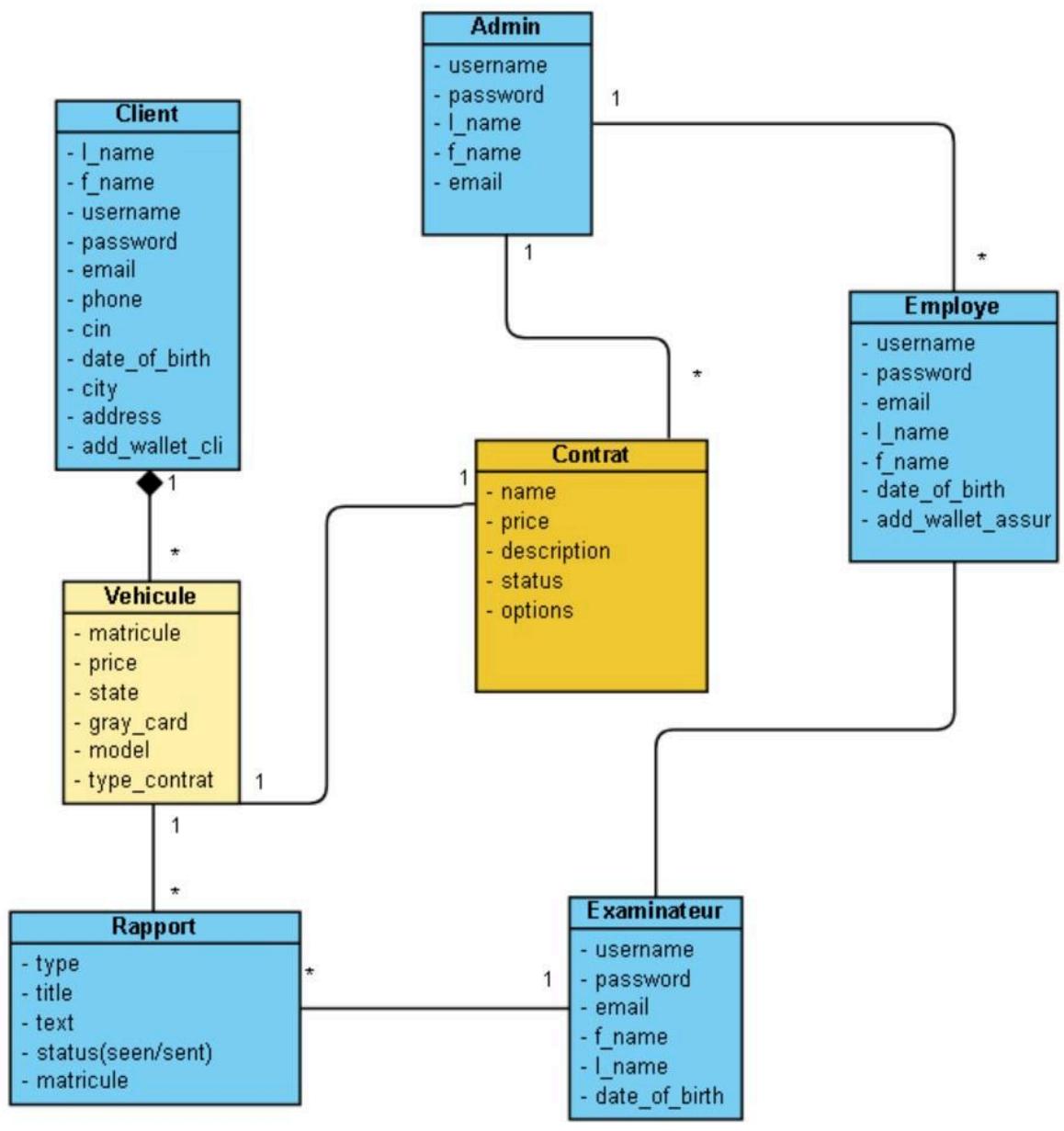
Introduction:

Le secteur de l'assurance automobile est en constante évolution, cherchant continuellement des moyens d'améliorer la gestion des risques et d'accroître l'efficacité des processus. Dans ce contexte, notre projet vise à concevoir et mettre en œuvre une application web intelligente et décentralisée d'assurance automobile en utilisant la technologie **blockchain** et une architecture de **microservices**, le tout intégré dans un processus **DevOps/MLOps**. Cette approche novatrice cherche à transformer fondamentalement la manière dont l'assurance automobile est gérée, en offrant une plus grande transparence, une sécurité renforcée et une agilité accrue.

1. Conception:

La conception du système est étayée par un diagramme de classe détaillé, offrant une vue globale des relations et des interactions entre les différents composants du système. Chaque entité, que ce soit les profils des employés, des clients ou des examinateurs, est soigneusement modélisée pour refléter de manière précise les besoins spécifiques de l'assurance automobile.

Le diagramme de classe illustre de manière exhaustive les relations et les attributs des différentes entités du système. Chaque classe représente une entité clé, telles que les profils des employés, des clients, des examinateurs, et détaille leurs propriétés et relations.



2. Technologies Utilisées :

2.1. Spring Boot :



Spring Boot, basé sur le framework Spring de Java, constitue la colonne vertébrale robuste de notre backend. Spring Boot simplifie le développement de microservices en offrant des solutions prêtes à l'emploi pour la gestion de la configuration, la sécurité, et la persistance des données. En utilisant le langage Java, Spring Boot assure une flexibilité, une évolutivité, et une maintenance facilitée tout au long du cycle de vie du projet.

2.2. Angular :



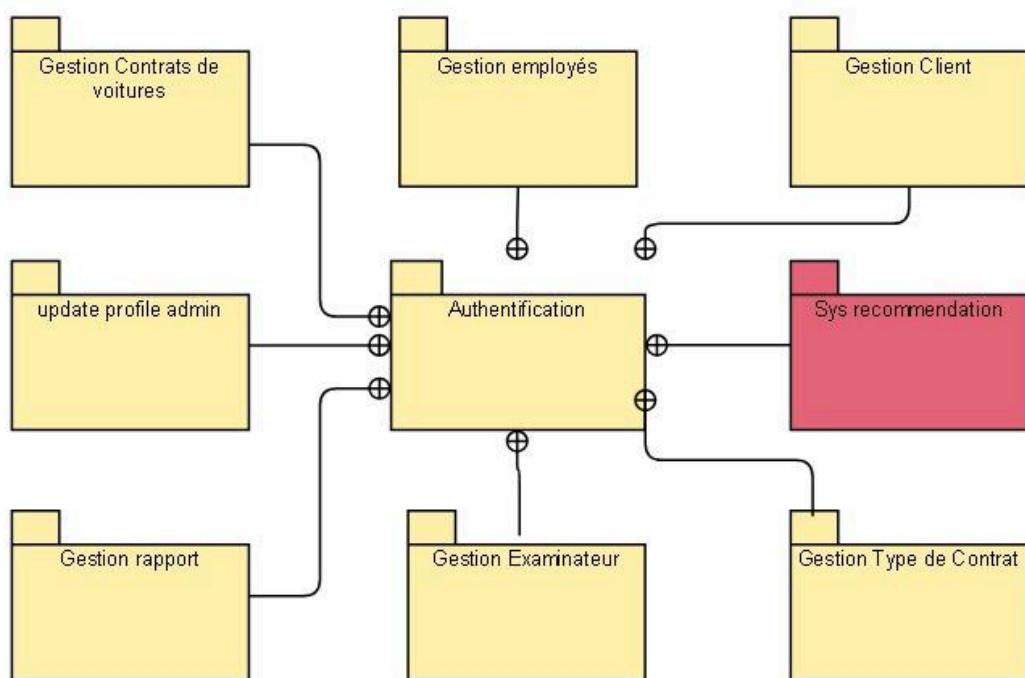
Angular, un framework développé par Google, est au cœur de notre frontend. En plus de TypeScript, Angular s'appuie sur des technologies complémentaires telles que Bootstrap, HTML, et CSS. Bootstrap facilite la conception d'interfaces utilisateur réactives et attrayantes, tandis qu'HTML et CSS permettent la structuration et la stylisation des pages web. Ensemble, ces technologies offrent une expérience utilisateur moderne et intuitive.

2.3. MongoDB :



MongoDB, une base de données NoSQL, est choisie pour le stockage des données, fournissant une flexibilité et une évolutivité nécessaires dans un environnement décentralisé. En utilisant un modèle de données orienté document, MongoDB facilite le stockage des informations détaillées sur les contrats, les clients, et autres entités du système. Sa capacité à s'intégrer naturellement avec les microservices dans une architecture décentralisée en fait un choix idéal pour notre projet.

2.4. Architecture Microservices :



L'infrastructure fondamentale de notre projet repose sur une architecture de microservices soigneusement conçue, où chaque composant joue un rôle essentiel dans la gestion efficace du processus d'assurance automobile. Ces microservices, tels que MS Authentification, MS Employee, MS Examinators, MS Clients, MS Profile Admin, MS Reports, MS Vehicule et MS Contracts, sont spécialisés pour répondre aux différentes facettes de notre système, garantissant ainsi une expérience utilisateur cohérente et sécurisée.

- **Microservice d'Authentification (MS Authentification) :**

Gère l'authentification des administrateurs, examinateurs, clients et employés. Fournit des fonctionnalités d'inscription pour les clients. Utilise WebSecurity Spring et JWT pour assurer la sécurité des connexions. Responsable du décodage des JSON Web Tokens (JWT) pour l'autorisation.

- **Microservice des Employés (MS Employee) :**

Gère la création, la mise à jour et la suppression des profils des employés.
Stocke des informations détaillées sur les employés

- **Microservice des Examinateurs (MS Examinators) :**

Gère la création, la mise à jour et la suppression des profils des examinateurs.
Stocke des informations spécifiques aux examinateurs

- **Microservice des Clients (MS Clients) :**

Gère la création, la mise à jour et la suppression des profils clients.
Stocke des informations liées aux clients

- **Microservice de Profil Administrateur (MS Profile Admin) :**

Permet la modification des informations relatives aux administrateurs.

- **Microservice des Rapports (MS Reports) :**

Gère la création, la mise à jour et la suppression des rapports d'incidents liés aux voitures.
Stocke des informations détaillées sur les incidents.

- **Microservice des Véhicules (MS Vehicle) :**

Gère la création, la mise à jour et la suppression des informations sur les véhicules.
Stocke des données sur les voitures, y compris les détails sur l'assurance.

- **Microservice des Contrats (MS Contracts) :**

Gère la création, la mise à jour et la suppression des types de contrats des voitures.
Active ou désactive les contrats.

- **Microservice d'Eureka :**

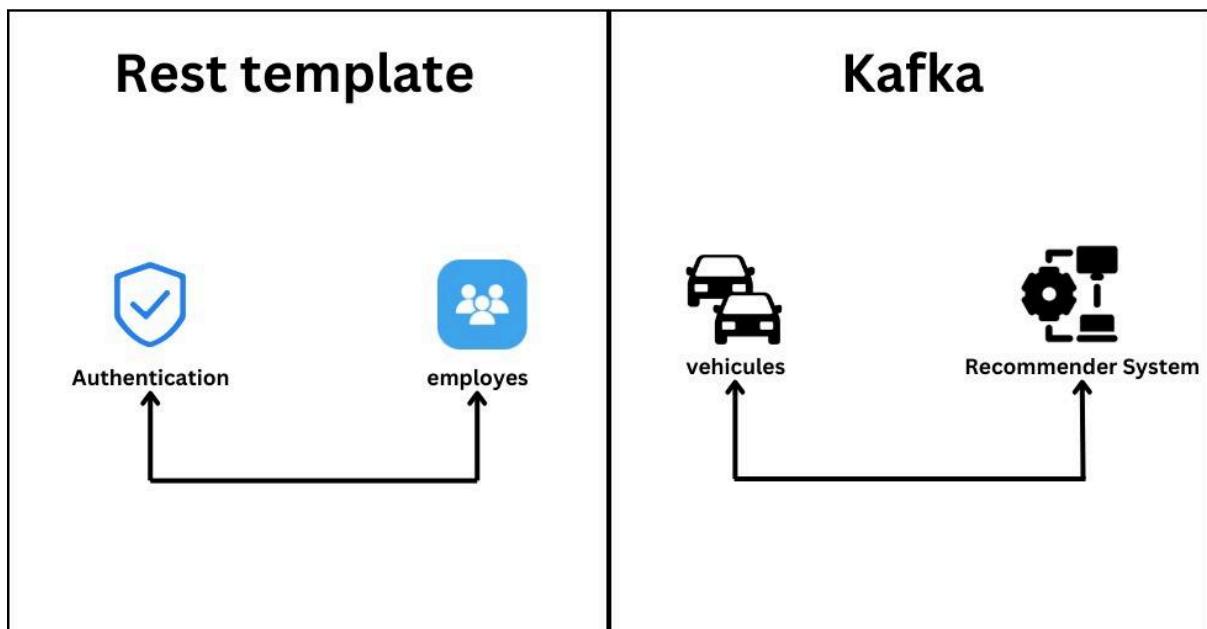
Le Microservice d'Eureka est notre guide dynamique pour la découverte de services au sein de l'architecture de microservices. Il facilite la localisation et l'interaction entre les différents services, assurant ainsi une communication fluide et une gestion dynamique des instances de microservices.

- **Microservice Gateway :**

Le Microservice Gateway agit comme la porte d'entrée intelligente de notre architecture. Il gère le routage des requêtes utilisateur vers les microservices appropriés, assurant une distribution efficace du trafic. Doté de capacités de filtrage, d'authentification, et d'autorisation, il garantit la sécurité et la gestion centralisée de l'accès aux différents services.

2.5 Kafka et RestTemplate :

Au cœur de notre architecture de microservices, la communication dynamique entre les composants est essentielle pour assurer une intégration transparente. Pour cette raison, nous avons intégré avec succès Apache Kafka et RestTemplate, deux outils puissants qui jouent un rôle clé dans la gestion des flux d'informations au sein de notre écosystème.



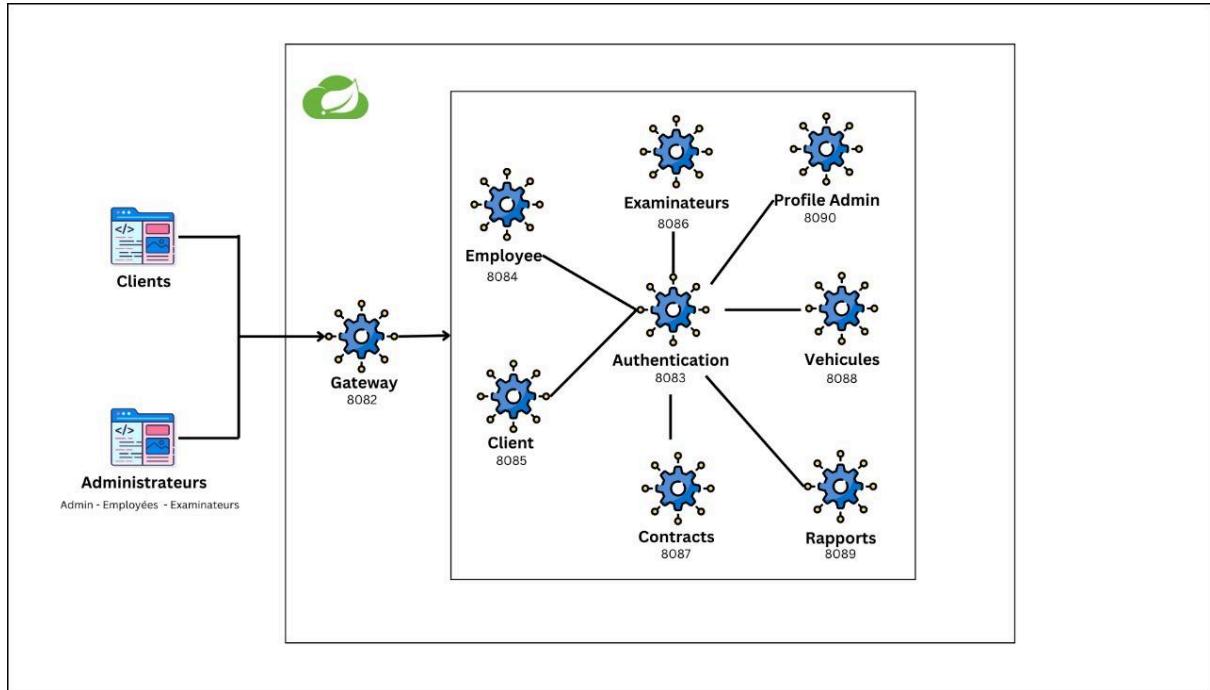
- **Kafka :**

Apache Kafka, notre hub d'événements, offre une infrastructure solide pour la transmission asynchrone des messages entre les microservices. Grâce à son modèle de publication-souscription, Kafka garantit une communication fiable et distribuée, permettant aux différents composants de réagir aux événements en temps réel. Cette architecture de messagerie joue un rôle central dans la création d'un écosystème de microservices réactif et évolutif.

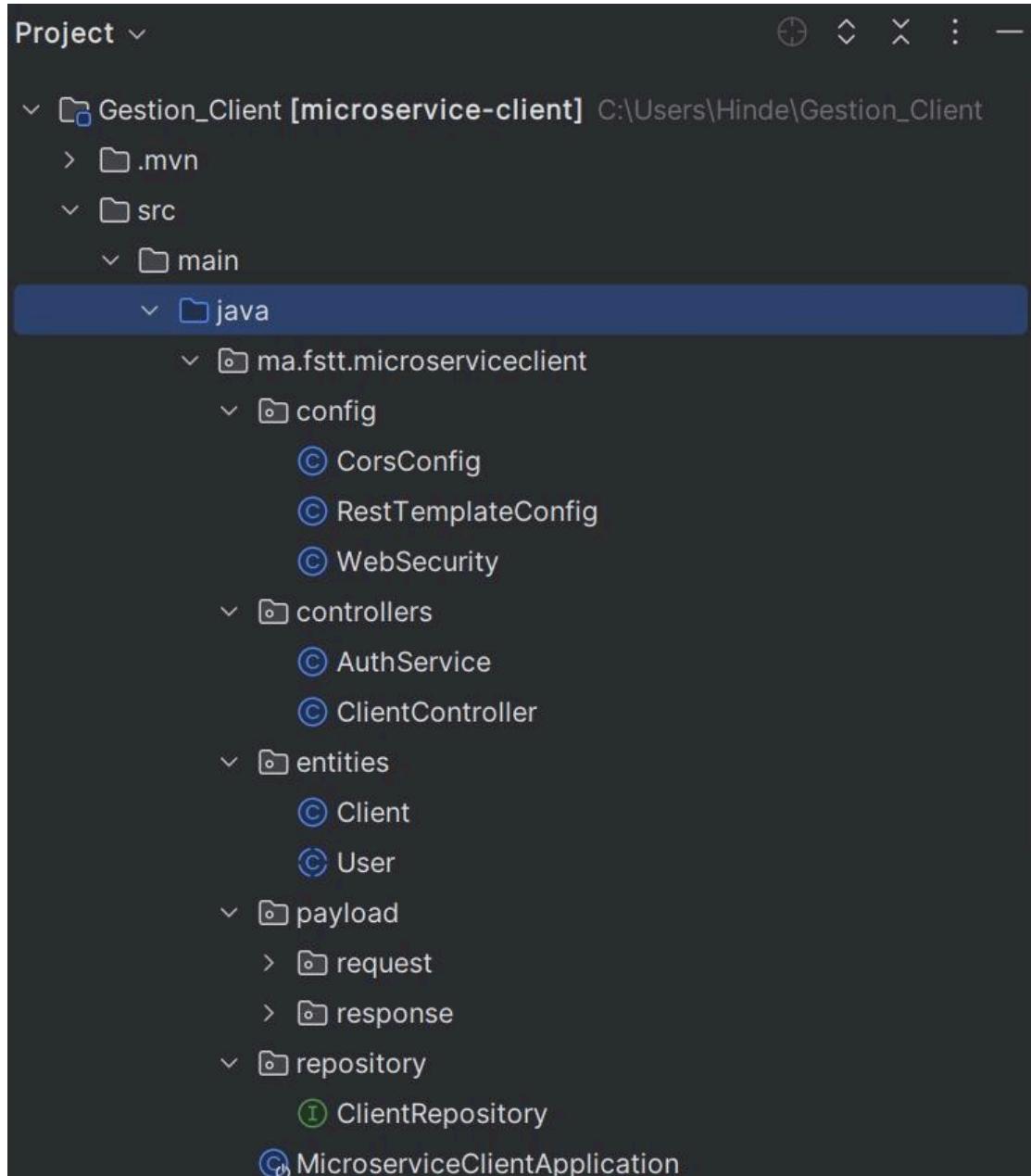
- **RestTemplate :**

RestTemplate est notre choix pour simplifier les appels HTTP entre les microservices. En s'appuyant sur le modèle REST, RestTemplate offre une interface pratique pour envoyer des requêtes HTTP et traiter les réponses. Cela garantit une communication efficace et structurée entre les différents composants, favorisant ainsi une collaboration harmonieuse au sein de notre architecture décentralisée.

3. L'architecture Globale :



4. Structure de projet :



La structure du projet est organisée en plusieurs packages comme suit :

- **Config** : Ce package est dédié à la configuration de l'application, y compris la gestion des CORS et d'autres configurations nécessaires.
- **Controllers** : Il regroupe les méthodes et la logique métier de l'application. Les contrôleurs sont responsables de traiter les requêtes et de fournir les réponses appropriées.
- **Entities** : Ce package définit les modèles de données de la base de données, spécifiant les entités et leurs attributs.
- **Repository** : Il contient les interfaces de repository pour MongoDB, décrivant les méthodes d'accès à la base de données.

- **Payload** : C'est un package composé de deux sous-packages, l'un pour les requêtes (request) et l'autre pour les réponses (response). Il est utilisé pour définir le format des requêtes reçues par l'application et les réponses renvoyées.

5. Les outils :

5.1. Swagger:

The screenshot shows the Swagger UI interface. At the top, it displays the 'Swagger' logo and the URL 'http://35.193.139.203:8087'. Below this, the title 'OpenAPI definition' is shown with a 'v0' and 'OAS3' badge. A 'Servers' dropdown menu is open, showing the URL 'http://35.193.139.203:8087 - Generated server url'. The main content area is titled 'contrat-controller' and lists several API endpoints:

- GET /api/contracts/{contractId}
- PUT /api/contracts/{contractId}
- DELETE /api/contracts/{contractId}
- PUT /api/contracts/state/{contractId}
- POST /api/contracts/add
- GET /api/contracts/getAll

Below the endpoints, there is a 'Parameters' section and a 'Try it out' button.

Swagger est un framework open-source qui simplifie le développement, la conception et la documentation des API REST. Il offre un ensemble d'outils permettant de spécifier, de tester et de documenter les API de manière standardisée.

5.2. Eureka Server:

The screenshot shows the Spring Eureka UI. At the top, it displays the 'spring Eureka' logo and the text 'HOME LAST 1000 SINCE STARTUP'.

System Status

Environment	test	Current time	2024-01-25T19:02:32 +0000
Data center	default	Uptime	00:34
		Lease expiration enabled	true
		Renews threshold	22
		Renews (last min)	48

DS Replicas

localhost

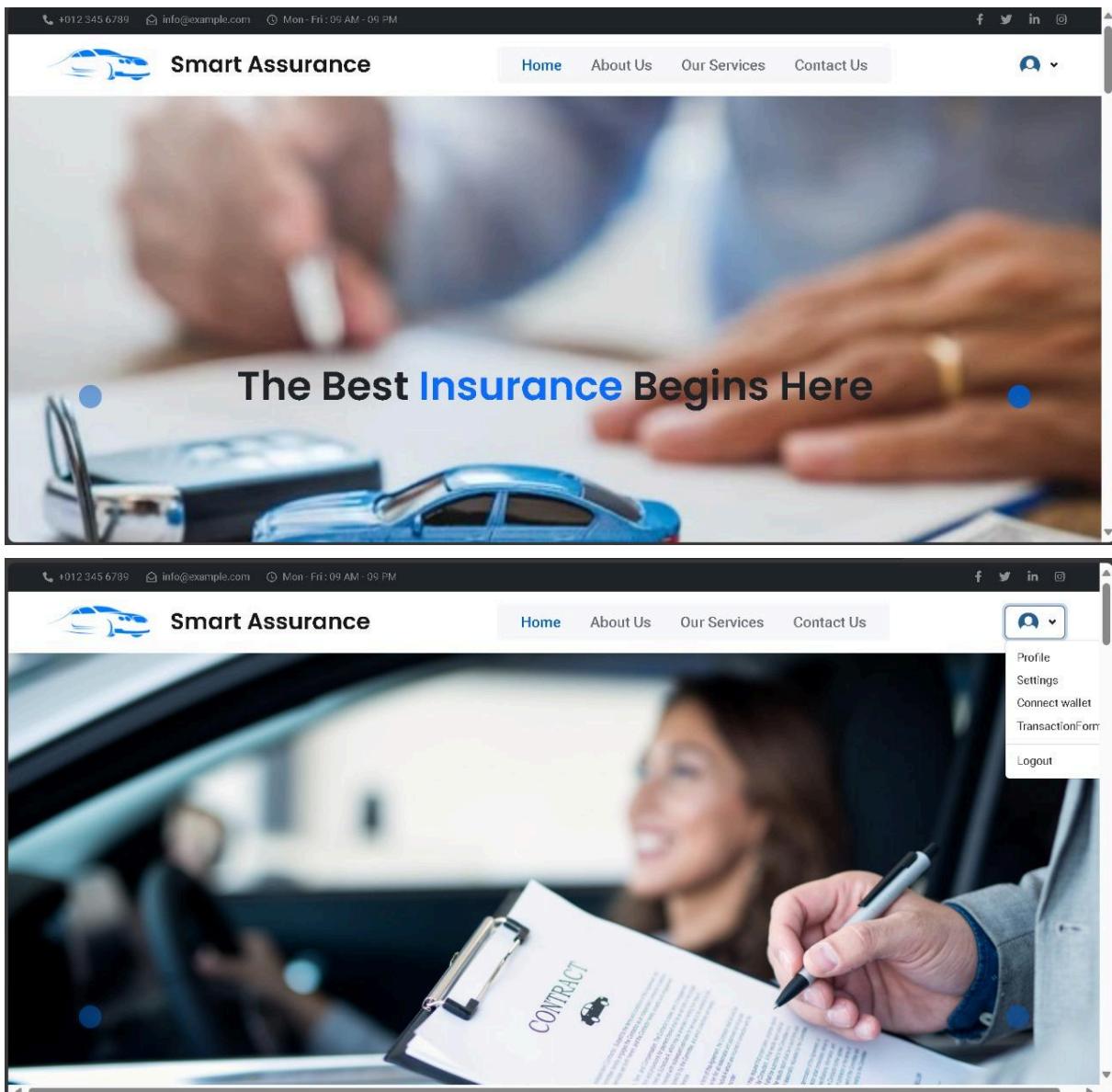
Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
API-GATEWAY-SERVICE	n/a (1)	(1)	UP (1) - a239a38fe233.api-gateway-service:8082
AUTHENTICATION-SERVICE	n/a (2)	(2)	UP (2) - 97d5c1412483.authentication-service:8083 , PC-MOHCINE.authentication-service:8083
CLIENT-SERVICE	n/a (1)	(1)	UP (1) - a305c4d209de.client-service:8085
CONTRACT-SERVICE	n/a (1)	(1)	UP (1) - d19f57459363.contract-service:8087
EMPLOYEE-SERVICE	n/a (1)	(1)	UP (1) - 4de9c0cbadbb.employee-service:8084
EXAMINATER-SERVICE	n/a (1)	(1)	UP (1) - b34e481550ae.examinater-service:8086

Eureka Server est une composante du framework Spring Cloud qui fournit des fonctionnalités de découverte de services dans un environnement de microservices.

6. Les interfaces :

6.1. Partie Client :



25
Years
Experience



Comprehensive Car Insurance Tailored for You

Ensure peace of mind on the road with our flexible and reliable car insurance plans. We've been providing top-notch insurance solutions for over 25 years.



Flexible Car Insurance Plans



Money Back Guarantee

Our insurance plans offer comprehensive coverage and come with a money-back guarantee. Drive with confidence, knowing you're protected against unexpected events on the road.



For Inquiries, Call Us: +012 345 6789

Few Reasons Why People Choosing Us!

At Smart Assurance, we strive to provide the best insurance solutions for our clients. Here are a few reasons why people choose our services:



Easy Process



Fast Delivery



Policy Controlling



Money Saving



Contact Us for Insurance Services

Have questions or need assistance? Feel free to reach out to us. We are here to help you with all your insurance needs.



Call Us: +012 345 6789

Your Name	Your Email
Your Mobile	Service Type
Message	
Send Message	

We Provide Professional Insurance Contracts

Jeune Conducteur

[Read More](#)

Luxe

[Read More](#)

Premium

[Read More](#)

Standard

[Read More](#)

Obligatoire

[Read More](#)

Insurance Solutions for Individuals and Organizations

Discover comprehensive insurance coverage tailored for both individuals and organizations. Our agency is committed to providing top-notch insurance services to ensure the security and well-being of our clients.

[Explore Our Services](#)

1500

Satisfied Clients

1000

Successful Insurance Projects

10

Awards for Excellence

50

Dedicated Team Members

What They Say About Our Insurance



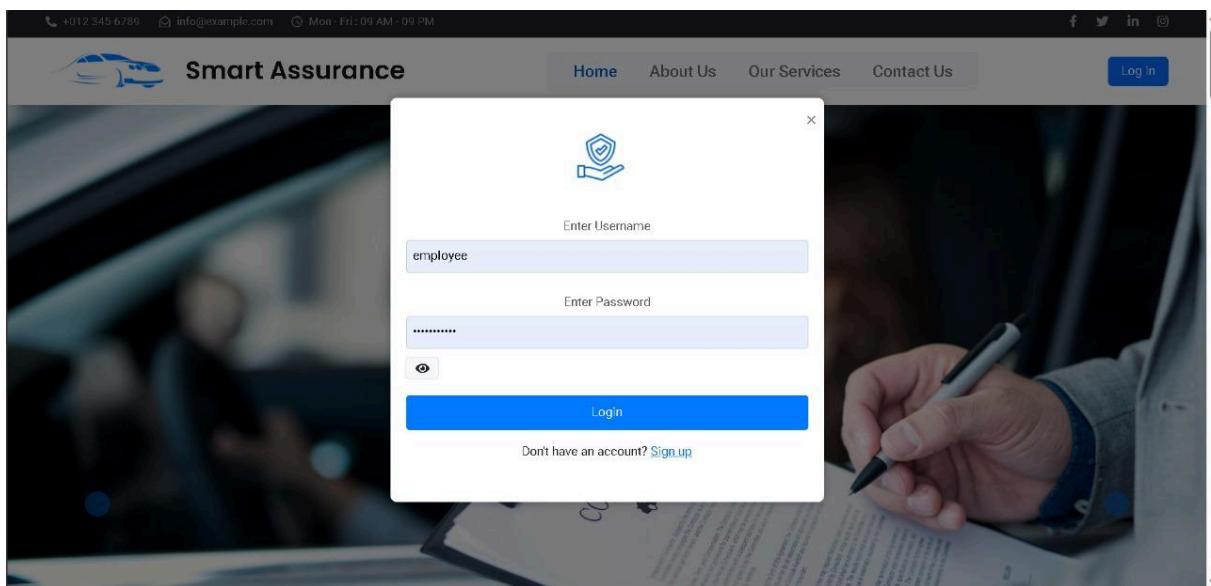
"Choosing this insurance was one of the best decisions for my family. The comprehensive coverage and personalized service have given us peace of mind. Thank you for your outstanding support!"

Jack

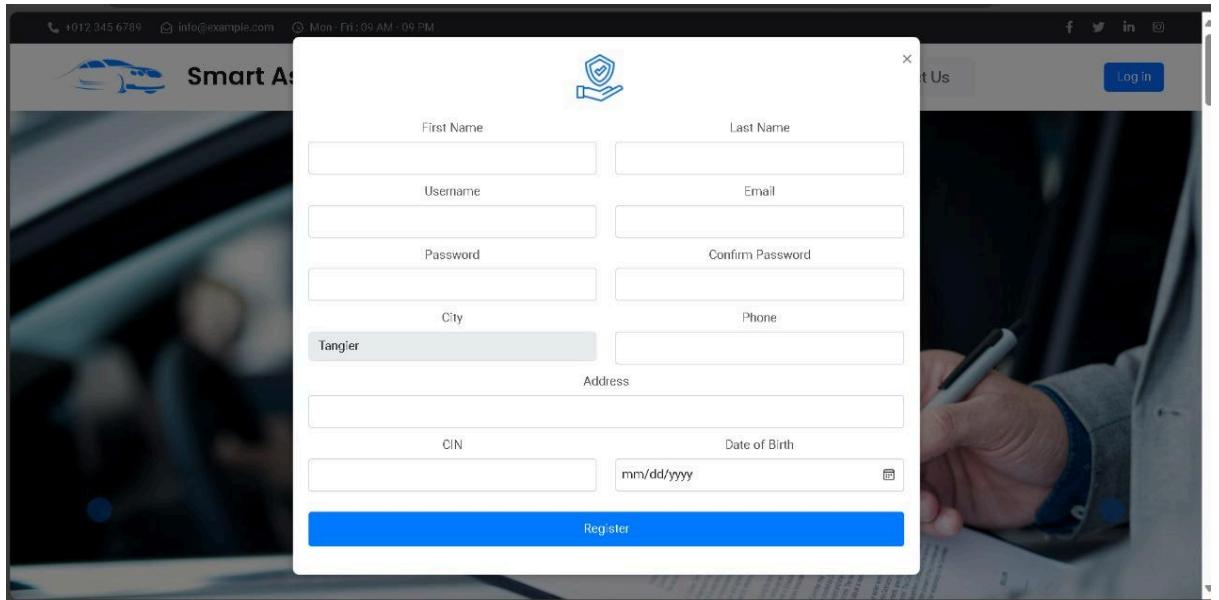
Home Maker



Page d'accueil ciblée



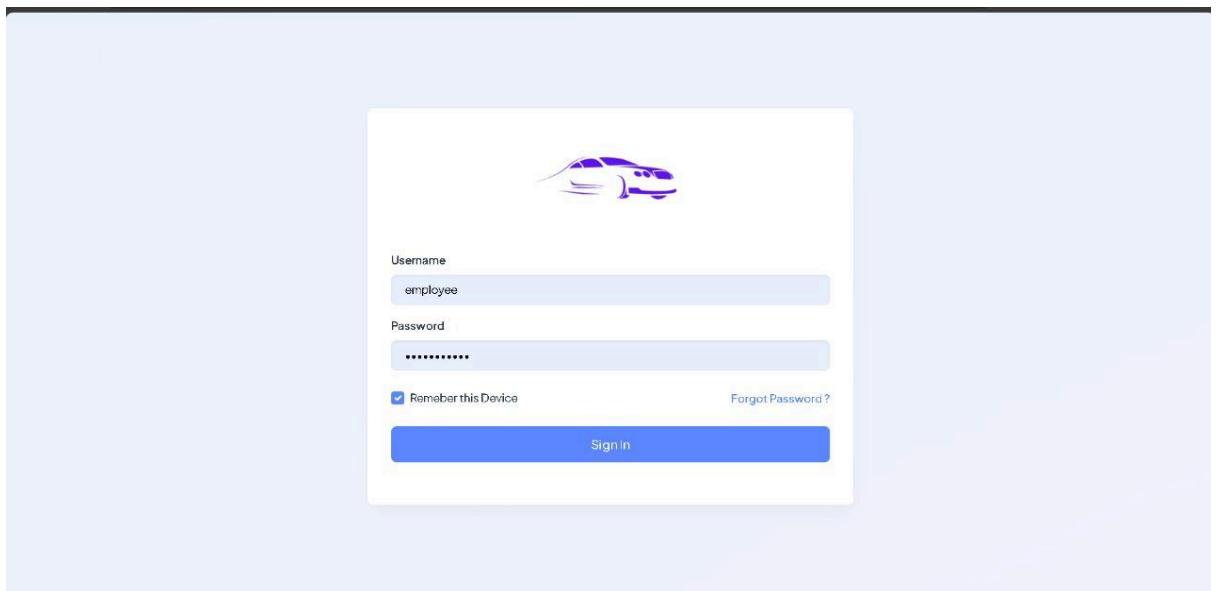
Log in



Register

6.2. Partie Administrateur :

6.2.1.Employee:



Log in

Clients management

CIN	Full Name	Email	Phone	City	Actions
CIN123	John Doe 123 Streettt	john.doe23@example.com	123456789A	Al Hoceima	
AB123	Hind BOUahabi 123 Rue Tanger	hind@gmail.com	0672817273	Tanger	
TEST123	Mohcine Boudjenjal 148 LOTELWAHIDA 2 DEROUA	mohcine@gmail.com	+21270168663	Tangier	

Gestion Client

Add New Client

Last Name	First Name
<input type="text"/>	<input type="text"/>
Email	Phone
<input type="text"/>	<input type="text"/>
CIN	Date of birth
<input type="text"/>	<input type="text"/>
Address	
<input type="text"/>	
City	Wallet
<input type="text"/>	<input type="text"/>

Add Client

Ajouter Client

Update Client

Last Name	First Name
<input type="text"/>	<input type="text"/>
Email	Phone
<input type="text"/>	<input type="text"/>
CIN	Date of birth
<input type="text"/>	<input type="text"/>
Address	
<input type="text"/>	
City	Wallet
<input type="text"/>	<input type="text"/>

Save

Modifier Client

CIN	Full Name	Email	Phone	City	Actions
EE1234	Saide Frikhe Hay Oulfa	saide@gmail.com	07332211333	Meknes	
WA123123	Examiner Examiner 123 rue	examiner@gmail.com	123123	Casa	

Gestion Examinateur

6.2.2.Admin:

CIN	Full Name	Email	Phone	City	Actions
CIN123	John Doe 123 Street	john.doe@example.com	123456789	City	
CIN123	John Doe 123 Street	john.doe@example.com	123456789	City	
CIN123	John Doe 123 Street	john.doe2@example.com	123456789	City	
CIN123	John Doe 123 Street	john.doe3@example.com	123456789	New York	
CIN123	John Doe 123 Streettt	john.doe23@example.com	123456789A	Al Hoceima	
1234567890	John Doe 123 Street	john@example.com	123456789	New York	
123456789012	John Doe 123 Street	john@example.com	123456789	New York	

Gestion employee

Contracts management

Ref	Name	Price	Options	Status	Actions
65b2f157e01830262b450f51	Jeune Conducteur	5000	Responsabilité civile Protection des passagers Dommage au véhicule Remplacement des clés	Active	⊕ ⊖ ✖
65b2f17be01830262b450f52	Luxe	10000	Toutes les caractéristiques disponibles	Active	⊕ ⊖ ✖
65b2f18be01830262b450f53	Premium	8000	Responsabilité civile Vol Incendie Bris de glace Dommage collision Valeur à neuf	Active	⊕ ⊖ ✖

Gestion des Contrats

Update admin profile

Last Name:

First Name:

Username:

Password:

Email:

Phone:

City:

Address:

Update Profile

Modifier profile

Client Address: 0xd52Dd9bA003F2AD

Get Transaction History

Recent Transactions for: 0xd52Dd9bA003F2AD5D9968b0b14fE3C17a43634Cc

Hash	From	To	DateTime	Amount
0xe64...3c6a9	0x87c9b02a...cd9bf	0xd52dd9ba003f2ad5d9968b0b14fe3c17a43634cc	25/01/2024 02:47:42	0.5 ETH
0x437...cc0f4	0xd52dd9ba...634cc	0xedd55ac4e340fda1d221965d0f72df6d99491ae	25/01/2024 03:08:28	0.2 ETH

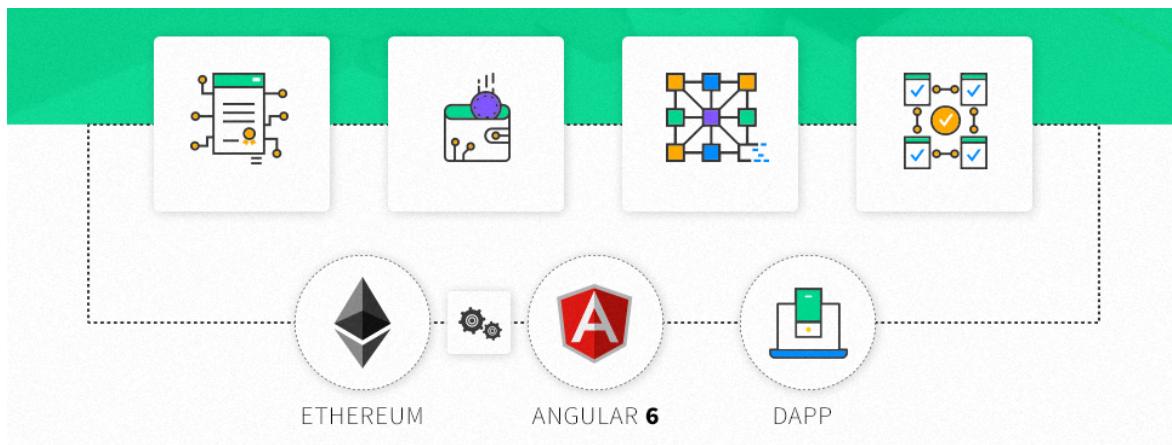
Transactions

Chapitre 2: La technologie Blockchain

Introduction:

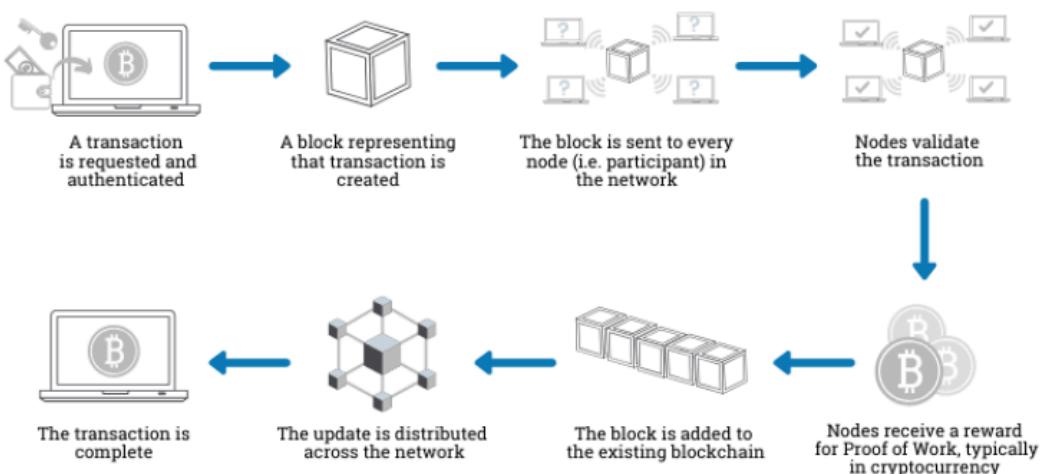
Le projet DAPP (Decentralized Application) élaboré avec Angular et Hardhat représente une application novatrice exploitant la technologie blockchain.

Cette section du rapport se concentre sur la blockchain elle-même, son fonctionnement, l'architecture web décentralisée, ainsi que les technologies spécifiques employées, notamment Ethereum, Solidity, Hardhat, MetaMask et Etherscan.

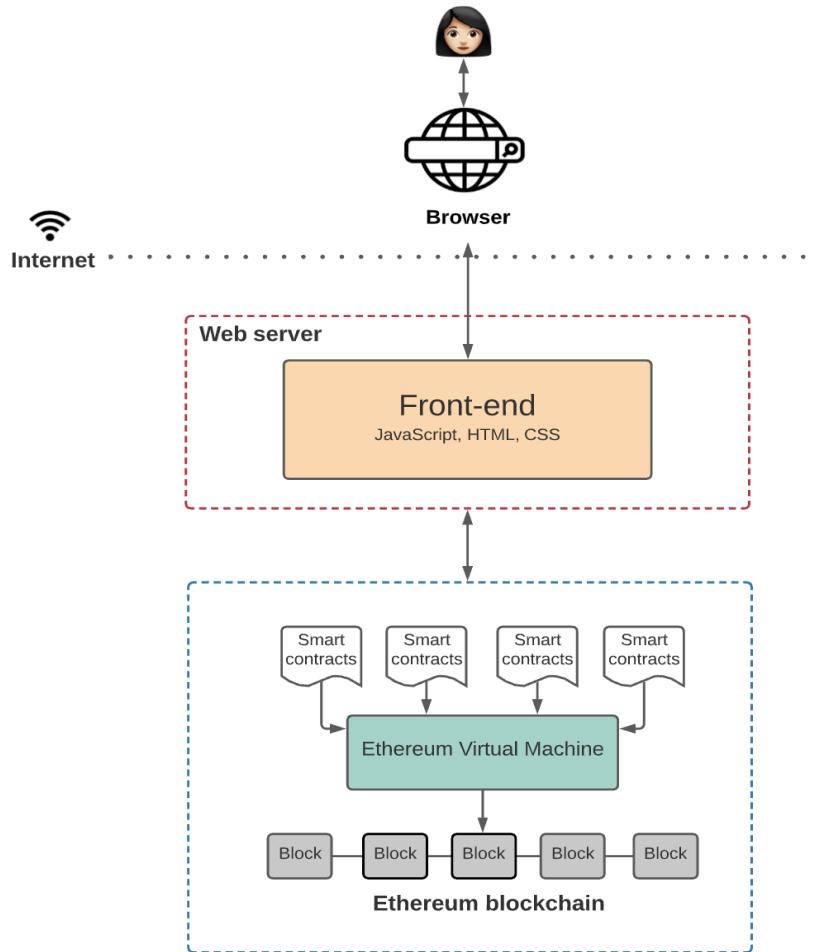


1. Blockchain: Fonctionnement et Principes

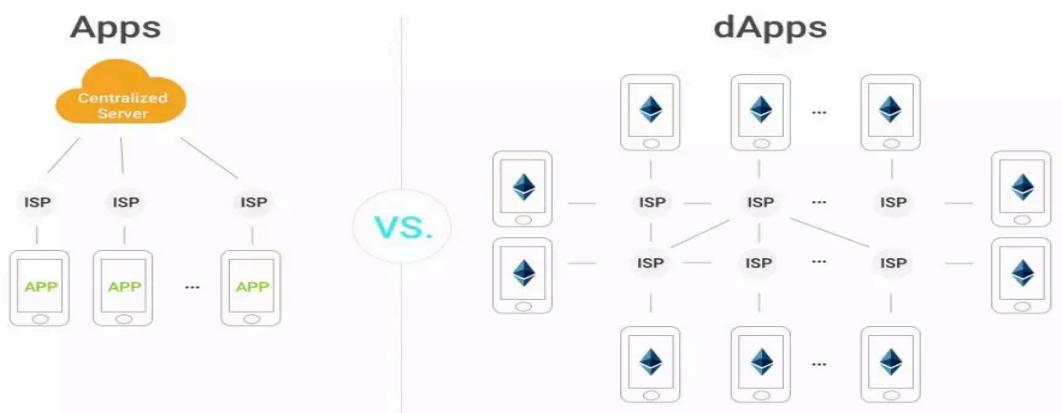
La blockchain, au cœur de ce projet, est un registre numérique décentralisé basé sur une architecture distribuée. Son fonctionnement repose sur le consensus, où chaque participant du réseau contribue à la validation des transactions et à la création de nouveaux blocs. La cryptographie assure la sécurité et l'intégrité des données stockées, tandis que des mécanismes de consensus tels que la preuve de travail ou la preuve d'enjeu garantissent un accord universel.



2. Architecture Web Décentralisée:



L'architecture web décentralisée (DWeb) transforme la manière dont les applications web sont construites. Elle délaisse la centralisation au profit d'une distribution des données sur la blockchain. Cette approche offre des avantages significatifs, notamment la résilience à la censure, la transparence accrue et une confiance sans nécessité de tiers de confiance. Dans le contexte de ce projet, les DApps fonctionnent sur un réseau de nœuds décentralisés, chacun stockant une copie du registre blockchain. Les utilisateurs interagissent avec ces applications via des interfaces utilisateur conviviales, souvent sous forme d'applications web.



3. Technologies Utilisées dans le Projet:

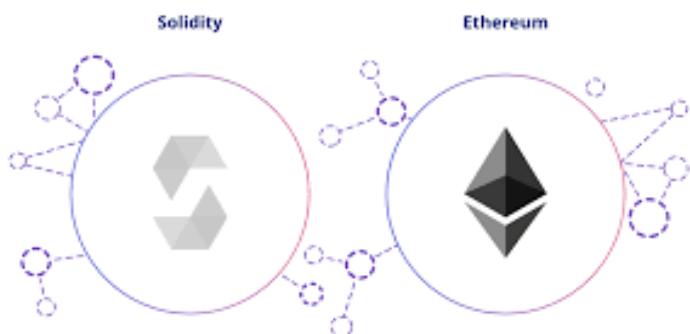
1. Ethereum:

En tant que plateforme blockchain, Ethereum permet le développement de contrats intelligents et d'applications décentralisées. Sa machine virtuelle (EVM) exécute du code décentralisé de manière sécurisée.



2. Solidity:

Langage de programmation spécifique à Ethereum, Solidity est utilisé pour écrire des contrats intelligents. Il offre des fonctionnalités puissantes pour la création de règles autonomes et d'applications décentralisées.



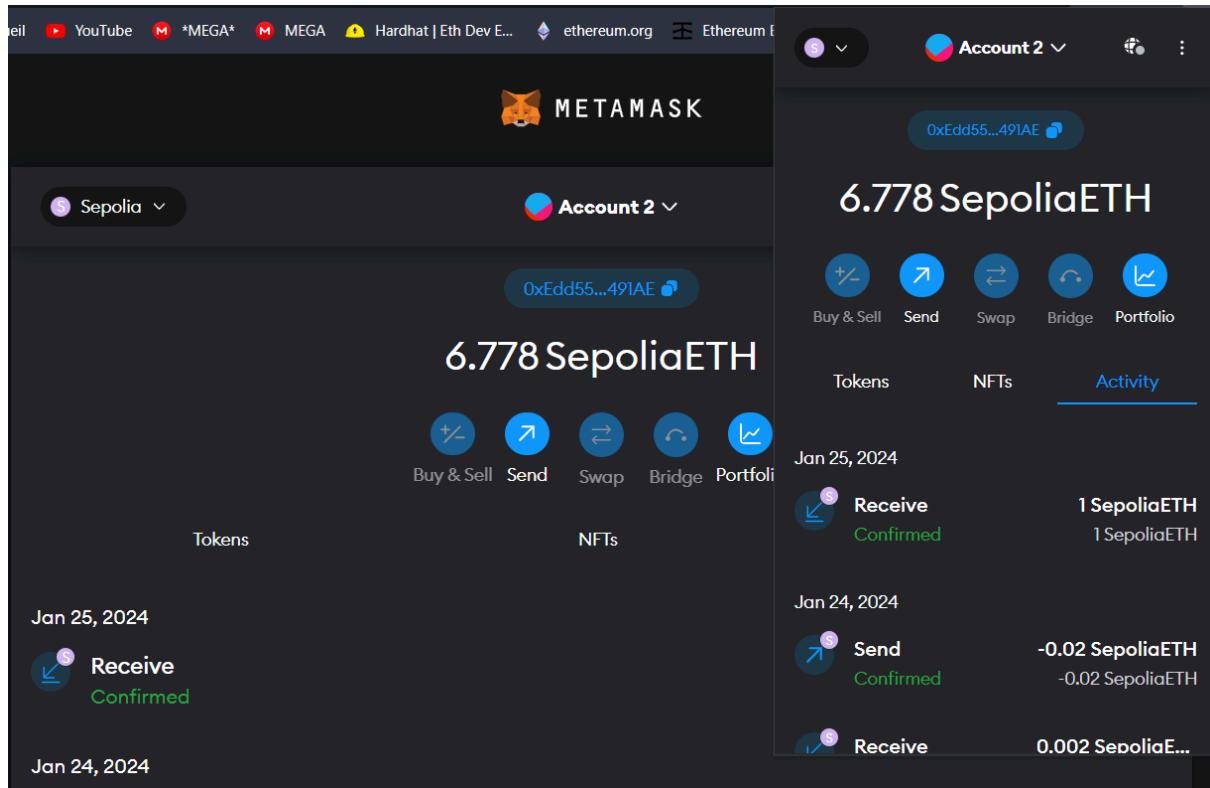
3. Hardhat:

Cet outil de développement pour Ethereum simplifie le processus de développement, de déploiement et de test de contrats intelligents. Il fournit un environnement robuste pour le développement.



4. MetaMask:

En tant qu'extension de navigateur, MetaMask permet aux utilisateurs d'accéder aux DApps directement depuis leur navigateur. Il agit comme un portefeuille numérique, simplifiant la gestion des clés privées.



5. Etherscan:

Explorateur de blocs Ethereum, Etherscan permet de vérifier les transactions, d'explorer les contrats intelligents et d'analyser l'état de la blockchain Ethereum.

The screenshot shows the Etherscan interface for the address 0xd52Dd9ba003F2AD5D9968b0b14fE3C17a43634Cc. The Overview section shows a balance of 0.299963925387167664 ETH. The Transactions section lists the latest 2 transactions from a total of 2, both being transfers. The first transfer is an OUT transaction to 0xedd55ac4e340fda1d1... with a value of 0.2 ETH and a fee of 0.000036074612. The second transfer is an IN transaction from 0x87c9b02a10ec2cb4dc... with a value of 0.5 ETH and a fee of 0.000020080678. An option to download CSV is available at the bottom right.

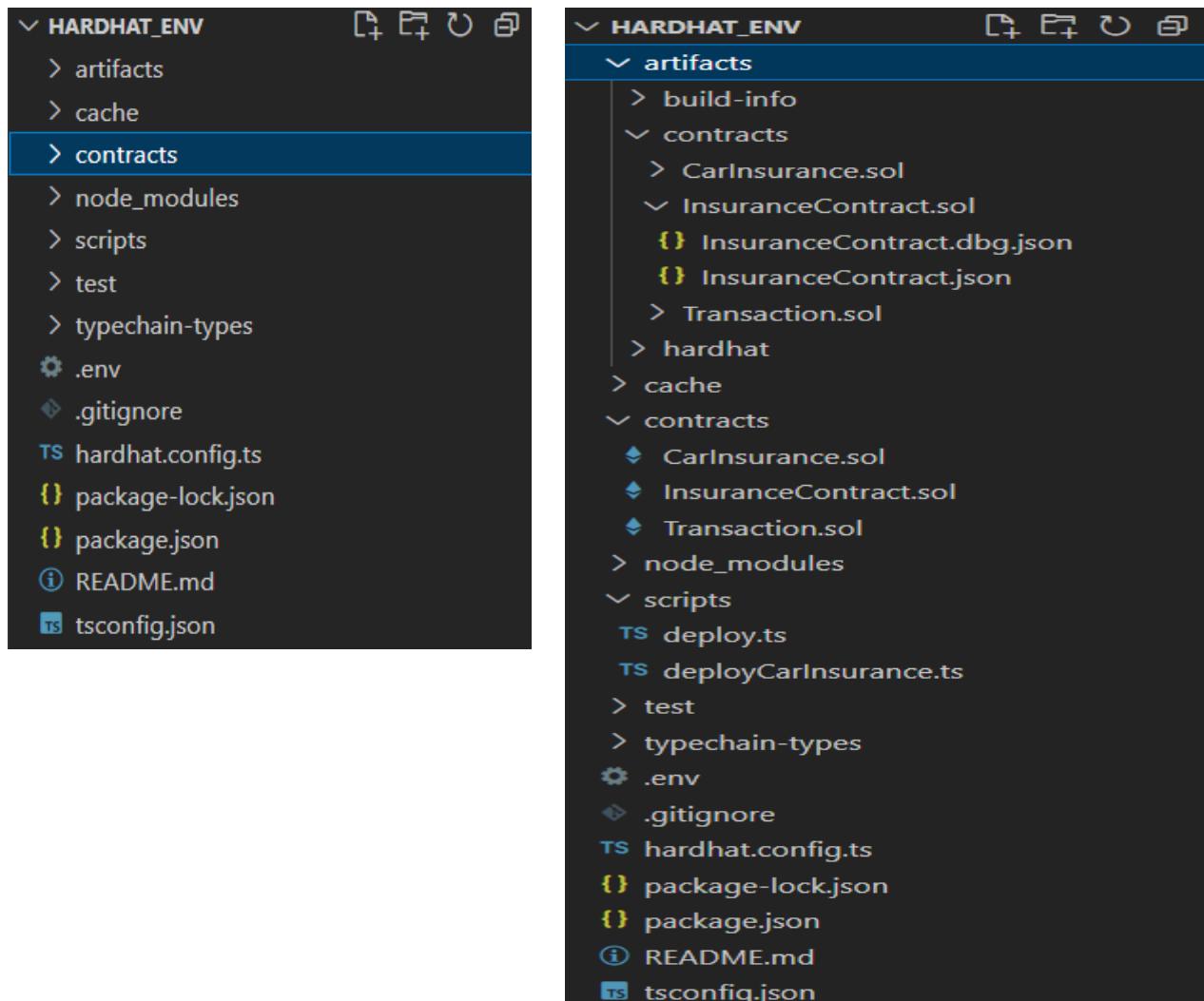
4. Structure du Projet avec Hardhat:

La structure du projet Hardhat a été soigneusement conçue pour faciliter le développement, le test et le déploiement des contrats intelligents. Elle se compose de plusieurs éléments clés, dont les fichiers de contrats Solidity, les artifacts stockant les ABI des contrats, ainsi que des scripts déployant ces contrats sur la blockchain.

Le répertoire principal du projet contient généralement un dossier "contracts" où les contrats intelligents sont définis en utilisant le langage Solidity. Ces fichiers ".sol" contiennent la logique métier de l'application décentralisée, définissant les règles et les interactions entre les différentes parties prenantes du réseau blockchain.

En parallèle, le répertoire "artifacts" joue un rôle essentiel en stockant les ABI (Application Binary Interface) générées à partir des contrats Solidity. Ces ABI fournissent une interface standardisée permettant aux applications frontales de communiquer avec les contrats intelligents sur la blockchain.

Le script de déploiement, généralement nommé "deploy.js", se trouve dans le répertoire "scripts". Ce script est responsable de déployer les contrats intelligents sur la blockchain. Il utilise l'API Hardhat pour interagir avec le réseau Ethereum, gérer les déploiements et mettre à jour les adresses des contrats dans l'application.

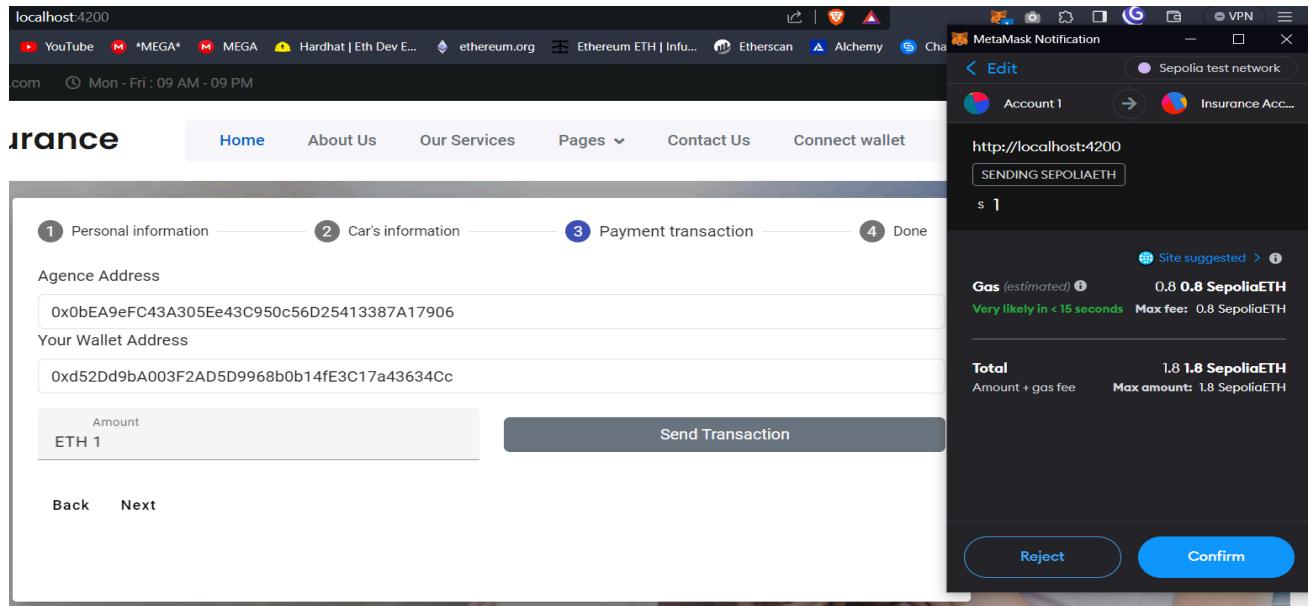


La configuration principale du projet, généralement nommée "hardhat.config.js", centralise les paramètres du projet, tels que la configuration des réseaux Ethereum, les plugins utilisés et d'autres réglages spécifiques à Hardhat.

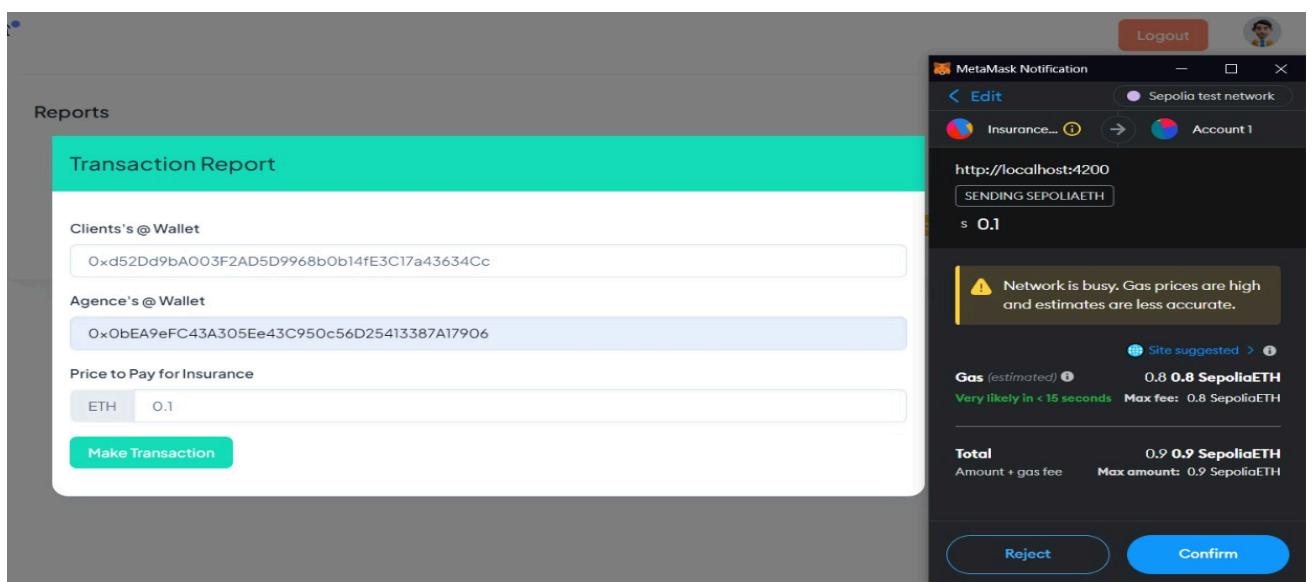
En résumé, la structure du projet Hardhat offre une organisation claire et efficace, permettant de créer, tester et déployer des contrats intelligents de manière cohérente et reproductible sur la blockchain Ethereum.

5. Processus de Transactions et Confirmation avec MetaMask:

Le processus de transactions dans une DApp développée avec Angular, Hardhat et Ethereum implique plusieurs étapes, de la préparation des données à la confirmation finale via MetaMask.



“From Client Account To Insurance Account”



“From Insurance Account To Client Account”

❖ Préparation des Données:

Lorsqu'un utilisateur souhaite effectuer une transaction, il fournit des informations essentielles telles que l'adresse du client, l'adresse de l'agence d'assurance, et le montant à envoyer. Ces données sont ensuite utilisées pour créer une transaction à l'aide des contrats intelligents déployés.

❖ Création de la Transaction:

Le script approprié dans le répertoire "scripts" du projet Hardhat est utilisé pour créer une transaction Ethereum. Ce script peut être appelé depuis l'application Angular, généralement via une interface utilisateur, en utilisant les bibliothèques Web3 ou ethers.js pour interagir avec la blockchain.

❖ Envoi de la Transaction:

Une fois la transaction créée, elle est signée avec la clé privée du client et envoyée à MetaMask pour confirmation. La transaction est ensuite diffusée sur le réseau Ethereum. Le montant spécifié est converti en wei si nécessaire, conformément aux normes de l'Ethereum.

❖ Confirmation avec MetaMask:

MetaMask, en tant qu'extension de navigateur, intervient à ce stade. L'utilisateur voit une fenêtre pop-up de MetaMask qui affiche les détails de la transaction, y compris le montant, les adresses des parties concernées, et les frais de transaction estimés.

- Confirmation des Détails:

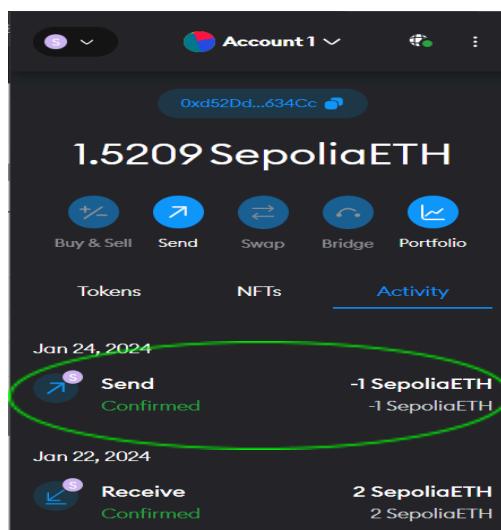
L'utilisateur examine les détails de la transaction et peut ajuster les frais de transaction s'il le souhaite. MetaMask fournit également une estimation du coût de la transaction en gaz.

- Approbation de la Transaction:

Une fois que l'utilisateur approuve la transaction, MetaMask demande à l'utilisateur de confirmer la transaction. La confirmation prouve que l'utilisateur est légitime et autorise l'exécution de la transaction.

- Envoi à la Blockchain:

Une fois confirmée, MetaMask envoie la transaction signée à la blockchain Ethereum. La transaction est ensuite ajoutée à la file d'attente des transactions en attente de confirmation (État: Pending => État: Confirmed).



❖ Vérification via Etherscan:

Les utilisateurs peuvent également vérifier la progression de la transaction sur des explorateurs de blocs tels que Etherscan. Une fois la transaction confirmée par le réseau Ethereum, les changements d'état, tels que le transfert de fonds entre les adresses, sont reflétés.

The screenshot shows the Etherscan interface for the Sepolia Testnet. At the top, there is a search bar labeled "Search by Address / Txn Hash / Block / Token". Below the search bar, the Etherscan logo and navigation links for Home, Blockchain, Tokens, NFTs, and Misc are visible. The main content area displays information for the address 0x0bEA9eFC43A305Ee43C950c56D25413387A17906. The "Address" section includes a QR code, the address itself, and a balance of 4.00254 ETH. Three tabs are present: "Overview", "More Info", and "Multichain Info", with "More Info" currently selected. It shows the last transaction sent (0x8d7e3e6a9ac5... from 10 hrs 35 mins ago) and the first transaction sent (0xd2efe12562d9c... from 3 days 18 hrs ago). The "Transactions" tab is active, showing a table of the latest 20 transactions. The table columns include Transaction Hash, Method, Block, Date Time (UTC), From, To, Value, and Txn Fee. The first transaction listed is a transfer from 0x0bEA9e...87A17906 to 0xEdd55a...D99491AE, with 1 ETH value and 0.00168 Txn Fee.

Transaction Hash	Method	Block	Date Time (UTC)	From	To	Value	Txn Fee
0x8d7e3e6a9ac55f501...	Transfer	5149197	2024-01-25 4:27:36	0x0bEA9e...87A17906	0xEdd55a...D99491AE	1 ETH	0.00168
0x8efbf6091a2802944...	Transfer	5146856	2024-01-24 18:20:36	0xd52D9...a43634Cc	0x0bEA9e...87A17906	1 ETH	0.00168
0x07ad7ed23bad5061...	Transfer	5142863	2024-01-24 2:32:12	0xEdd55a...D99491AE	0x0bEA9e...87A17906	0.02 ETH	0.00168

Ce processus assure une expérience transparente et sécurisée pour les utilisateurs, et MetaMask joue un rôle clé en facilitant la confirmation des détails de la transaction et en sécurisant l'autorisation de la transaction.

6. Consultation de l'Historique des Transactions par les Utilisateurs:

Pour offrir une visibilité transparente de l'historique des transactions, l'application Angular développée avec Hardhat inclut une fonctionnalité de consultation des transactions. Les utilisateurs, qu'il s'agisse de clients ou de l'agence d'assurance, ont la possibilité de consulter l'historique des transactions associées à une adresse Ethereum spécifique.

- ❖ Recherche par Adresse: L'utilisateur peut entrer une adresse Ethereum dans l'interface utilisateur de l'application pour obtenir un historique des transactions associées à cette adresse.
- ❖ Affichage des Transactions: L'application affiche les transactions passées, organisées par date ou par ordre chronologique inverse. Chaque transaction est présentée de manière claire avec plusieurs informations essentielles.

- Émetteur de la Transaction: L'adresse qui a initié la transaction est clairement indiquée. Il peut s'agir d'un client qui envoie des fonds ou de l'agence d'assurance effectuant une opération spécifique.
- Destinataire de la Transaction: L'adresse qui a reçu les fonds ou les avantages de la transaction est également affichée. Cette information permet de comprendre le flux des fonds ou des données dans l'écosystème de l'application.
- Montant de la transaction: Le montant en ethers de la transaction est présenté de manière lisible, permettant aux utilisateurs de comprendre les valeurs financières associées à chaque transaction.
- Date et Heure de la Transaction: Chaque transaction enregistrée dans l'historique est également associée à une date et une heure précises, fournissant un contexte temporel essentiel. L'horodatage de la transaction indique le moment exact où l'événement financier a été déclenché sur la blockchain Ethereum.
- Hash de la Transaction: Chaque transaction est associée à un identifiant unique appelé le hash de transaction. Ce hash permet aux utilisateurs de vérifier les détails spécifiques de la transaction sur des explorateurs de blocs tels que Etherscan.

Transaction History for address: 0xf39Fd6e51aad88F6F4ce6ab8827279cffFb92266

<p>FROM: 0x7e480e349a31f71f59b57cb30bd850b4d643d617</p> <p>To: 0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266</p> <p>Date/Time: 19/11/2023 06:06:12</p> <p>Amount: 0.1 ETH</p> <p>Hash: 0xd2e5369fd1cf17e2026a7e14517c13138544c513f617c6d5ff1a32d0dcc489d</p>	<p>FROM: 0x1a1e021a302c237453d3d45c7b82b19ceeb7e2e6</p> <p>To: 0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266</p> <p>Date/Time: 10/11/2023 03:42:22</p> <p>Amount: 1e-18 ETH</p> <p>Hash: 0x44606afe71d18f42f6e8193c864bd9c77101b9c187106e916017fe3e2e54d5e6</p>
<p>Client Address: 0xf39Fd6e51aad88F6F4ce6ab8827279cffFb92266</p> <p>Get Transaction History Voir tous</p>	

Cette fonctionnalité d'historique des transactions offre aux utilisateurs une compréhension détaillée des interactions financières passées, favorisant la transparence et la confiance au sein de l'écosystème décentralisé. Les informations fournies permettent à chaque partie prenante, qu'il s'agisse d'un client ou de l'agence d'assurance, de suivre et de comprendre les activités financières liées à leur adresse Ethereum.

Conclusion:

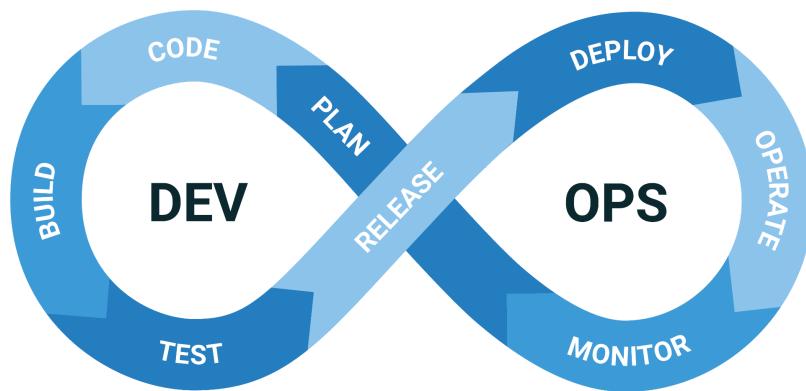
Le projet DAPP détaillé dans cette section tire parti des technologies blockchain pour créer une expérience utilisateur transparente et sécurisée. L'ensemble des technologies utilisées, d'Ethereum à Etherscan, démontre l'écosystème complexe mais puissant qui sous-tend ce type de développement. Cette initiative représente une avancée significative vers des applications décentralisées, offrant une alternative viable aux solutions centralisées traditionnelles.

Chapitre 3:DevOps

Introduction:

DevOps, une fusion des termes "Development" et "Operations", représente une approche culturelle et méthodologique visant à améliorer la collaboration entre les équipes de développement et d'exploitation.

Au-delà d'une simple série d'outils, DevOps encourage l'automatisation des processus, l'intégration continue, le déploiement continu, et la surveillance constante des performances. Ce mouvement favorise la réduction des silos organisationnels, la responsabilité partagée, et une culture d'amélioration continue, transformant ainsi la manière dont les logiciels sont développés, déployés et maintenus.



Choix du cloud:

DevOps et le cloud computing sont deux concepts étroitement liés qui, lorsqu'ils sont combinés, contribuent à transformer la manière dont les équipes développent, déplacent et gèrent des applications.

L'utilisation de cloud dans la philosophie DevOps permet:

- Une scalabilité permettant de répondre aux besoins changeants des applications.
- L'automatisation de l'infrastructure.
- Des outils CI/CD intégrés.
- Une flexibilité des environnements de développement.
- Le paiement à l'utilisation.



Google Cloud

Google Cloud Platform (GCP) propose souvent des offres promotionnelles telles que les 300 \$ de crédit gratuits valables sur une période de trois mois afin d'inciter de nouveaux utilisateurs à explorer et à adopter leur plateforme.

GCP propose aussi une documentation exhaustive et conviviale qui guide les utilisateurs à travers les différents services disponibles, explique les meilleures pratiques et fournit des exemples concrets.

Les technologies DevOps mise en oeuvre:

1. Jenkins

Jenkins est une plateforme open-source d'intégration continue et de déploiement continu (CI/CD). Il offre un environnement automatisé pour la construction, les tests et le déploiement d'applications, facilitant ainsi une livraison continue. Jenkins permet de créer des pipelines CI/CD pour automatiser le processus de développement logiciel. Il s'intègre avec de nombreux outils et frameworks, prend en charge l'exécution de scripts, et offre une interface utilisateur conviviale pour la configuration des jobs.

2. Terraform

Terraform est un outil d'Infrastructure as Code (IaC) développé par HashiCorp. Il permet de définir et de provisionner l'infrastructure de manière déclarative, facilitant la gestion et le déploiement cohérents des ressources cloud. Avec Terraform, les utilisateurs décrivent l'infrastructure souhaitée dans des fichiers de configuration. Ces fichiers sont ensuite utilisés pour créer, mettre à jour et supprimer des ressources cloud de manière prévisible. Terraform prend en charge plusieurs fournisseurs cloud, permettant une gestion unifiée de l'infrastructure sur des plateformes telles que AWS, Azure et GCP.

3. Docker

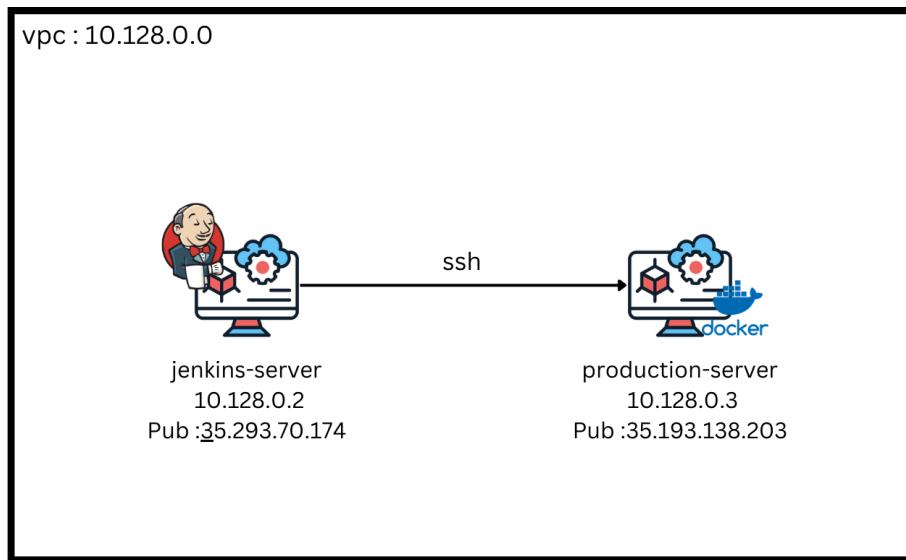
Docker est une plateforme de conteneurisation qui permet d'emballer une application et ses dépendances dans un conteneur léger et portable. Docker simplifie le déploiement en encapsulant une application et ses dépendances dans un conteneur. Ces conteneurs peuvent être exécutés de manière cohérente sur n'importe quel environnement compatible avec Docker. Cela facilite la création d'environnements isolés, accélère le déploiement des applications, et encourage l'adoption d'architectures basées sur des microservices.

4. Bash Scripting

Bash, ou Bourne Again SHell, est un interpréteur de commandes UNIX qui offre une interface en ligne de commande pour interagir avec le système d'exploitation.
Utilisation : Bash est utilisé pour automatiser des tâches en ligne de commande. Dans le contexte de DevOps, des scripts Bash sont souvent utilisés pour automatiser des processus, configurer des environnements, et exécuter des opérations système.

Conception de l'infrastructure:

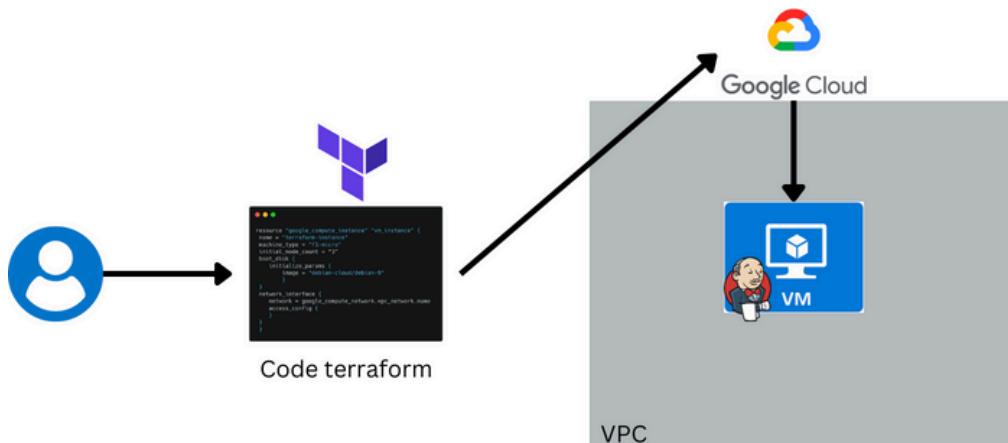
On a adopté une infrastructure simple contenant un réseau (vpc: virtual cloud platform):



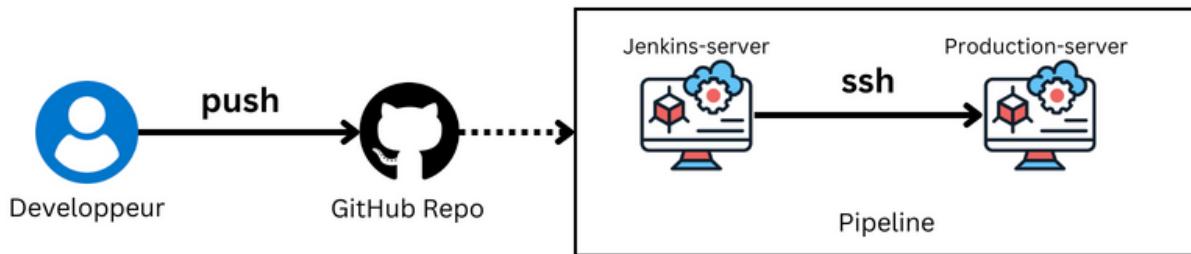
Utilisation de deux VM, une contient Jenkins et l'autre pour la production contient Docker, les deux sont liées par SSH pour échanger les informations et les fichiers.

Utilisation de terraform:

A travers un code HCL, j'ai défini les prérequis de mon Virtual Machine sous forme d'un code qui va s'exécuter pour créer une instance.



CI / CD:



La partie CI/CD (Continuous Integration/Continuous Deployment) dans le scénario décrit implique un processus automatisé qui déclenche la construction, les tests et le déploiement d'un microservice à chaque modification du code source.

1. Développeur Push dans GitHub :

Un développeur effectue des modifications dans le code source du microservice et pousse ces modifications vers le référentiel GitHub. Cette action déclenche une notification que le code source a été mis à jour.

2. Jenkins Trigger cette Push :

Jenkins est configuré pour surveiller le référentiel GitHub à la recherche de modifications. Lorsqu'un développeur pousse du code, Jenkins détecte cette modification grâce à un webhook.

3. Pipeline Jenkins pour Générer un Nouveau JAR :

Jenkins récupère le dernier code depuis GitHub.
Il compile le code source et exécute des tests unitaires.
Le nouveau JAR est généré avec succès.

4. Envoyer Jar via ssh vers Production-Server

Jenkins utilise scp ou un autre mécanisme pour transférer l'image Docker vers le serveur de production via SSH

5. Importation et Déploiement sur le Serveur de Production

Sur le serveur de production, Jenkins se connecte via SSH et utilise docker load pour charger l'image Docker.

Jenkins utilise Docker Compose pour déployer la nouvelle image. Cela peut inclure l'arrêt des conteneurs existants avec `docker-compose down` et le redémarrage avec la nouvelle image via `docker-compose up -d`

Pipeline Jenkins:

Un pipeline Jenkins est une séquence d'étapes automatisées qui permet de gérer et d'orchestrer le processus de développement, de test et de déploiement des logiciels de manière continue.

Tout d'abord il faut créer des pipelines pour chaque microservice, et lier chaque repository par son pipeline. Dans chaque pipeline faut mentionner le lien de repository et le code pipeline.(dans notre cas chaque repository contient un JenkinsFile)

.mvn/wrapper	type contracts management	5 days ago
src	delete docker file	2 days ago
.gitignore	type contracts management	5 days ago
Jenkinsfile	Update Jenkinsfile	2 days ago
mvnw	type contracts management	5 days ago
mvnw.cmd	type contracts management	5 days ago
pom.xml	Update pom.xml	2 days ago

Il faut ajouter un webhook pour chaque repository pour faire trigger sur le push

General

Webhooks / Manage webhook

Access

Collaborators and teams

Code and automation

- Branches
- Tags
- Rules
- Actions
- Webhooks** (selected)
- Pages
- Custom properties

Settings

Recent Deliveries

We'll send a **POST** request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL *

`http://35.239.70.174:8080/github-webhook/`

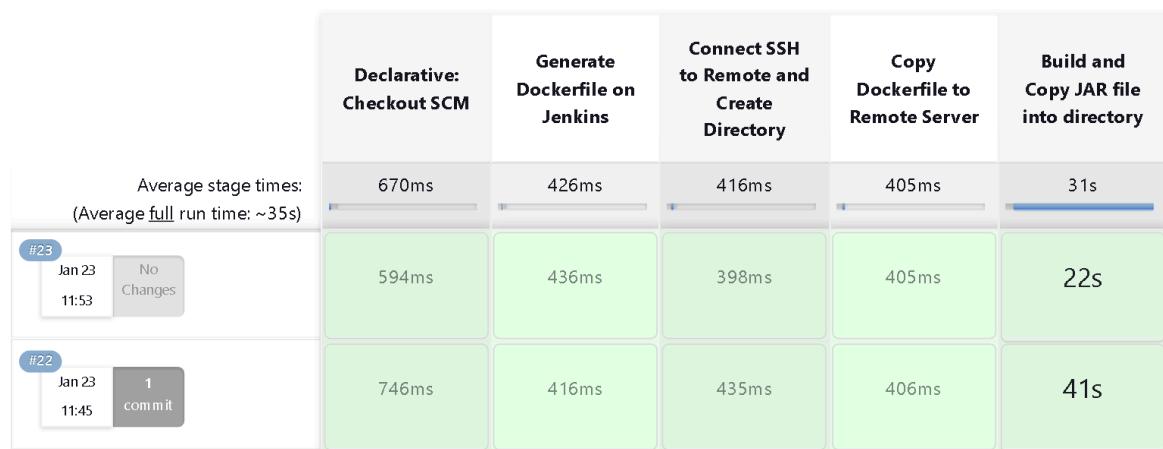
Content type

`application/json`

Secret

Un pipeline jenkins pour définir les différents stage afin de déployer dans le serveur backend

Stage View



Le pipeline est créé pour chaque microservice

S	W	Name	Last Success	Last Failure	Last Duration
		eureka-server	2 days 17 hr #23	3 days 13 hr #15	25 sec
		gateway	2 days 17 hr #7	N/A	47 sec
		microservice-admin	2 days 3 hr #7	2 days 3 hr #6	35 sec
		microservice-authentication	2 days 17 hr #42	2 days 18 hr #37	1 min 43 sec
		microservice-client	2 days 16 hr #6	3 days 6 hr #1	36 sec
		microservice-contact	2 days 5 hr #2	2 days 5 hr #1	32 sec
		microservice-employee	2 days 4 hr #3	2 days 5 hr #2	40 sec
		microservice-examinateur	2 days 3 hr #6	2 days 4 hr #5	51 sec
		microservice-report	10 hr #8	10 hr #7	37 sec
		microservice-vehicule	10 hr #6	10 hr #5	45 sec

Serveur production:

Après le process de build, les microservices sont regroupés dans le serveur de production chacun avec son dockerfile

```
mohcineboudjenal@production-server:~/smartassurance/prod
mohcineboudjenal@production-server ~/smartassurance/prod $ ls -l
total 64
-rw-r--r-- 1 mohcineboudjenal mohcineboudjenal 3753 Jan 24 01:00 docker-compose.yaml
drwxr-xr-x 2 mohcineboudjenal mohcineboudjenal 4096 Jan 23 08:19 eureka-server
drwxr-xr-x 2 mohcineboudjenal mohcineboudjenal 4096 Jan 23 08:19 gateway
drwxr-xr-x 2 mohcineboudjenal mohcineboudjenal 4096 Jan 23 23:19 microservice-Employee
drwxr-xr-x 2 mohcineboudjenal mohcineboudjenal 4096 Jan 24 00:43 microservice-admin
drwxr-xr-x 2 mohcineboudjenal mohcineboudjenal 4096 Jan 24 00:13 microservice-admin-profile
drwxr-xr-x 2 mohcineboudjenal mohcineboudjenal 4096 Jan 23 10:04 microservice-authentication
drwxr-xr-x 2 mohcineboudjenal mohcineboudjenal 4096 Jan 23 10:47 microservice-client
drwxr-xr-x 2 mohcineboudjenal mohcineboudjenal 4096 Jan 23 23:17 microservice-contact
drwxr-xr-x 2 mohcineboudjenal mohcineboudjenal 4096 Jan 23 23:20 microservice-contact
drwxr-xr-x 2 mohcineboudjenal mohcineboudjenal 4096 Jan 24 00:30 microservice-employee
drwxr-xr-x 2 mohcineboudjenal mohcineboudjenal 4096 Jan 24 00:26 microservice-examinater
drwxr-xr-x 2 mohcineboudjenal mohcineboudjenal 4096 Jan 24 00:37 microservice-examineur
drwxr-xr-x 2 mohcineboudjenal mohcineboudjenal 4096 Jan 24 00:02 microservice-profile-admin
drwxr-xr-x 2 mohcineboudjenal mohcineboudjenal 4096 Jan 23 23:18 microservice-report
drwxr-xr-x 2 mohcineboudjenal mohcineboudjenal 4096 Jan 24 00:56 microservice-vehicule
mohcineboudjenal@production-server ~/smartassurance/prod $
```

et un docker compose pour définir et faciliter la configuration des applications, chacun avec son port et ses dépendances

```
mhcineboudenjal@production-server:~/smartassurance/prod
mhcineboudenjal@production-server ~/smartassurance/prod $ cat docker-compose.yaml
version: '3.8'
services:
  eureka-server:
    image: smartassurance-eureka-server:0.0.1-SNAPSHOT
    build: eureka-server/
    ports:
      - 8761:8761
  gateway:
    image: smartassurance-gateway:0.0.1-SNAPSHOT
    build: gateway/
    depends_on:
      - eureka-server
    environment:
      SPRING_APPLICATION_JSON: '{"eureka":{"client":{"serviceUrl":{"defaultZone":"http://eureka-server:8761/eureka"}}}}'
    ports:
      - 8082:8082
  microservice-authentication:
    image: smartassurance-authentication:0.0.1-SNAPSHOT
    build: microservice-authentication/
    depends_on:
      - eureka-server
    environment:
      SPRING_APPLICATION_JSON: '{"eureka":{"client":{"serviceUrl":{"defaultZone":"http://eureka-server:8761/eureka"}}}}'
    ports:
      - 8083:8083
  microservice-client:
    image: smartassurance-client:0.0.1-SNAPSHOT
    build: microservice-client/
    depends_on:
      - eureka-server
      - microservice-authentication
    environment:
      SPRING_APPLICATION_JSON: '{"eureka":{"client":{"serviceUrl":{"defaultZone":"http://eureka-server:8761/eureka"}}, "auth": {"service": {"url": "http://microservice-authentication:8083/api/auth"}}}'
    ports:
      - 8085:8085
  microservice-employee:
    image: smartassurance-employee:0.0.1-SNAPSHOT
    build: microservice-employee/
    depends_on:
      - eureka-server
```

Chapitre 4:Système de recommandation

1. Introduction

L'évolution constante du secteur de l'assurance des véhicules soulève la nécessité d'adopter des approches innovantes pour mieux répondre aux besoins spécifiques des utilisateurs. Dans ce contexte, notre projet se concentre sur la conception d'un système de recommandation dédié aux contrats d'assurance automobile.

1.1. Contexte du Projet

Le paysage de l'assurance automobile est complexe, marqué par une multitude d'options et de critères spécifiques. Notre initiative vise à simplifier ce processus en introduisant un système de recommandation intelligent qui anticipe les exigences individuelles des utilisateurs, offrant ainsi une expérience personnalisée et efficiente.

1.2. Objectif Global

Notre objectif principal est de créer un système de recommandation agile, capable d'analyser les caractéristiques uniques de chaque utilisateur et de suggérer des contrats d'assurance adaptés à leurs besoins spécifiques.

2. Technologies Utilisées

La sélection judicieuse des technologies est cruciale pour garantir la robustesse, la flexibilité et la performance d'un système de recommandation dédié aux contrats d'assurance automobile. Chacune des technologies choisies a été intégrée dans le projet en raison de ses avantages spécifiques, contribuant ainsi à une architecture globale cohérente et efficace.

2.1 Beautiful Soup

Beautiful Soup a été choisi pour son excellente capacité à extraire des données à partir de sources web. Dans le cadre de notre projet, Beautiful Soup facilite le web scraping nécessaire à la collecte de données sur les différents contrats d'assurance automobile disponibles sur diverses plateformes en ligne. Sa souplesse et sa simplicité d'utilisation ont simplifié le processus d'acquisition de données, garantissant ainsi une base d'information exhaustive.

Beautifulsoup

2.2 Kafka

Kafka a été adopté comme système de messagerie pour la communication entre les microservices du système de recommandation. Sa capacité à gérer efficacement les flux de données en temps réel, sa scalabilité horizontale et sa tolérance aux pannes en font un choix idéal pour maintenir une communication fluide et fiable entre les différents composants du système.



2.3 Scikit-Learn

Scikit-Learn a été retenu pour son ensemble complet d'outils d'apprentissage automatique et de traitement des données. Pour la modélisation du système de recommandation, les fonctionnalités de prétraitement des données, de visualisation, d'entraînement des modèles et d'évaluation des performances de Scikit-Learn se sont avérées essentielles. Son architecture modulaire et sa facilité d'utilisation ont grandement accéléré le cycle de développement.



2.4 Cassandra

Cassandra a été privilégiée comme base de données NoSQL pour la gestion et la mise à jour du dataset. Sa capacité à gérer un grand volume de données avec une latence faible, ainsi que sa distribution horizontale, sont particulièrement adaptées aux besoins de stockage et de mise à jour dynamique des informations sur les contrats d'assurance automobile.



2.5 Airflow

Airflow a été intégré pour orchestrer et automatiser les différentes étapes du processus, formant ainsi un pipeline cohérent. Grâce à son interface utilisateur conviviale et à ses capacités d'automatisation robustes, Airflow simplifie la gestion des tâches planifiées, assurant une exécution fluide et fiable du pipeline du système de recommandation.



3. Web Scraping

La phase d'acquisition de données constitue une étape fondamentale dans la construction d'un système de recommandation pour les contrats d'assurance automobile. L'utilisation de Beautiful Soup pour le web scraping a permis d'extraire des données réelles et pertinentes à partir du site **moteur.ma**, en se concentrant particulièrement sur les neufs véhicules et les véhicules d'occasion.



3.1 Sources de Données

Le site **moteur.ma** a été sélectionné comme source principale de données en raison de sa riche variété d'informations sur les véhicules, leurs caractéristiques et les contrats d'assurance associés. Les neufs véhicules et les annonces de véhicules d'occasion ont été ciblés pour garantir une représentation exhaustive du marché.



3.2 Processus de Scrapping

Le processus de web scraping a été réalisé en plusieurs étapes pour assurer la collecte exhaustive et précise des données :

Identification des Pages Pertinentes : BeautifulSoup a été utilisé pour naviguer à travers les pages du site moteur.ma, en identifiant les sections dédiées aux neufs véhicules et aux véhicules d'occasion.

Extraction des Informations Clés : À l'aide de balises HTML spécifiques et de classes définies, BeautifulSoup a été employé pour extraire des informations telles que la marque, le modèle, l'année de fabrication, le type de carburant, et d'autres caractéristiques des véhicules.

Gestion de la Pagination : La pagination des résultats a été gérée pour collecter des données sur plusieurs pages du site, garantissant ainsi une représentation complète des neufs véhicules et des véhicules d'occasion.

Stockage des Données : Les données extraites ont été stockées dans des fichiers JSON, assurant une structure facilement exploitable pour les étapes ultérieures du projet.

3.3 Assurance Qualité des Données

La qualité des données est cruciale pour la fiabilité du système de recommandation. Pour assurer la qualité des données collectées via le web scraping, plusieurs mesures ont été mises en place :

3.4 Exemple des données collectées via le web scraping

```
{
    "marque": "abarth",
    "model": "595",
    "version": "abarth-595-595-turismo-165ch-2360",
    "Année de sortie du modèle": "2020 - 2024",
    "Carburant": "Essence",
    "Puissance fiscale": "8 cv",
    "Puissance réelle": "165 ch",
    "Boite de vitesses": "Manuelle",
    "price": "242403",
    "car_url": "https://www.moteur.ma/fr/neuf/fiche-technique-prix"
},
{
    "marque": "abarth",
    "model": "595",
    "version": "abarth-595-595-competizione-180-2361",
    "Année de sortie du modèle": "2020 - 2024",
    "Carburant": "Essence",
    "Puissance fiscale": "8 cv",
    "Puissance réelle": "180 ch",
    "Boite de vitesses": "Manuelle",
    "price": "271503",
    "car_url": "https://www.moteur.ma/fr/neuf/fiche-technique-prix"
},
```

4. Modélisation du Système de Recommandation

4.1. Prétraitement des Données

Le prétraitement des données constitue une étape essentielle dans la modélisation du système de recommandation dédié aux contrats d'assurance automobile. L'intégration des données collectées à partir du web scraping a nécessité une approche rigoureuse pour garantir la qualité, la cohérence, et la fiabilité du dataset utilisé pour l'entraînement des modèles.

4.1.1 Fusion des Datasets

Les données provenant de sources multiples ont été fusionnées en un seul dataset cohérent. Ceci a impliqué l'utilisation de la bibliothèque Pandas pour la gestion efficace des structures de données, permettant ainsi la consolidation des informations relatives aux neufs véhicules et aux véhicules d'occasion.

4.1.2 Élimination des Données Null et des Doubles

Le dataset a été scrupuleusement inspecté afin d'identifier et de supprimer les valeurs nulles. Les doubles, résultant potentiellement de la nature hétérogène des données collectées, ont également été éliminés. Cette étape, réalisée avec l'aide de Pandas, a contribué à garantir l'intégrité et la cohérence du dataset.

4.1.3 Normalisation des Valeurs

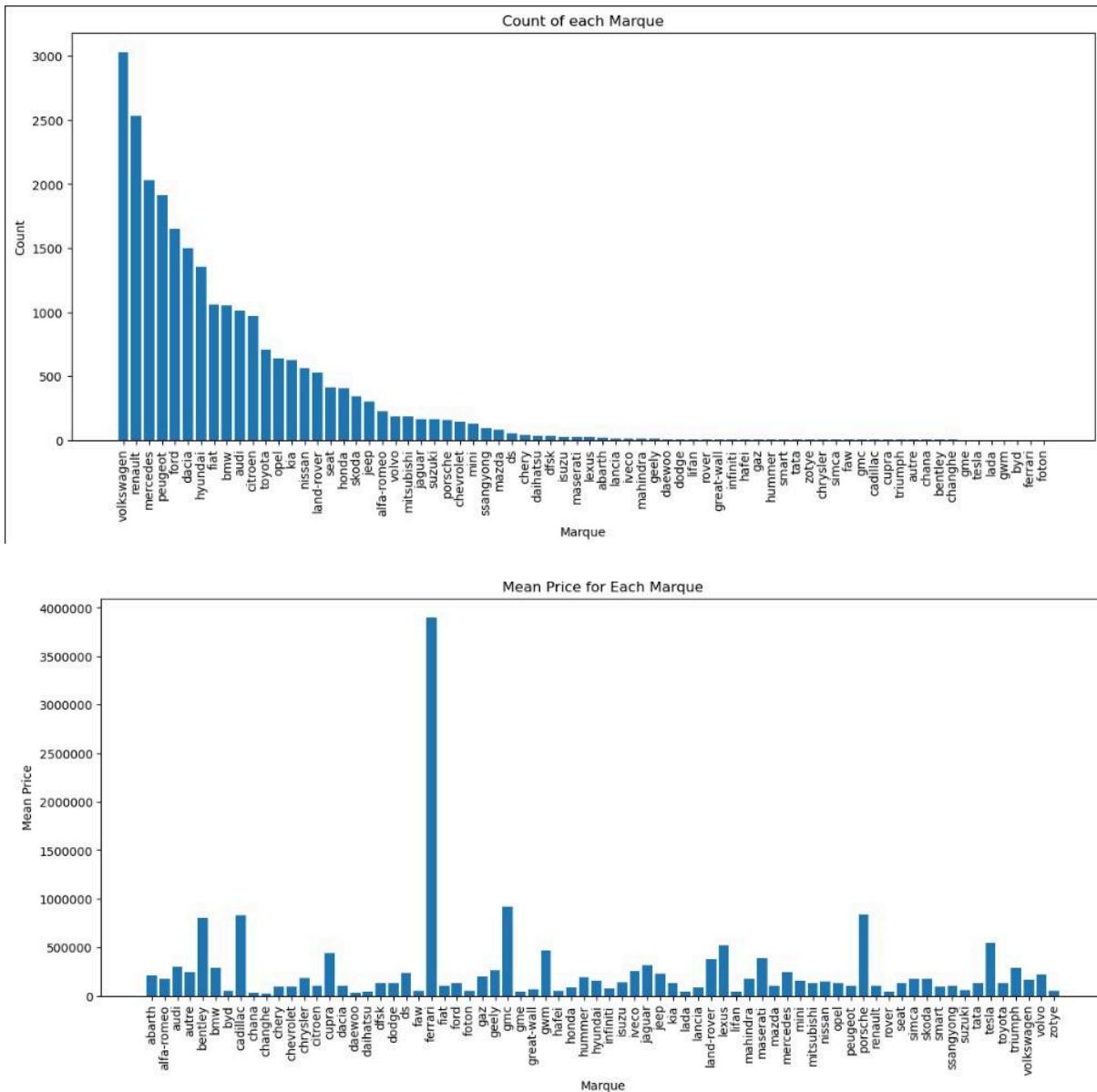
La normalisation des valeurs a été entreprise pour standardiser l'échelle des caractéristiques, garantissant ainsi que chaque attribut contribue de manière équitable à l'entraînement des modèles. La bibliothèque scikit-learn a été utilisée pour appliquer des techniques de normalisation telles que la mise à l'échelle StandardScaler.

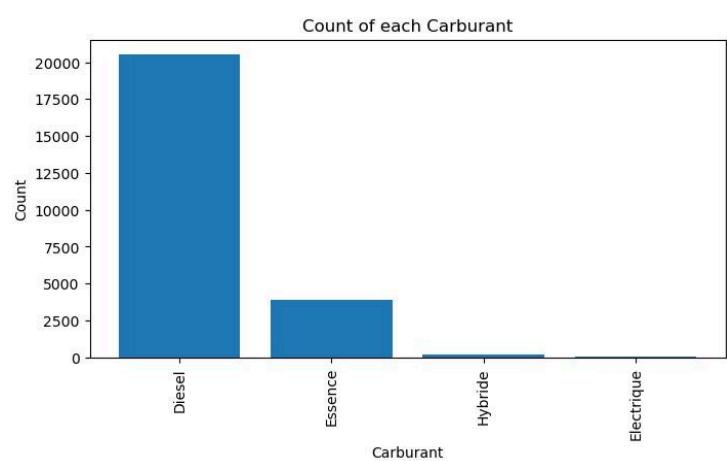
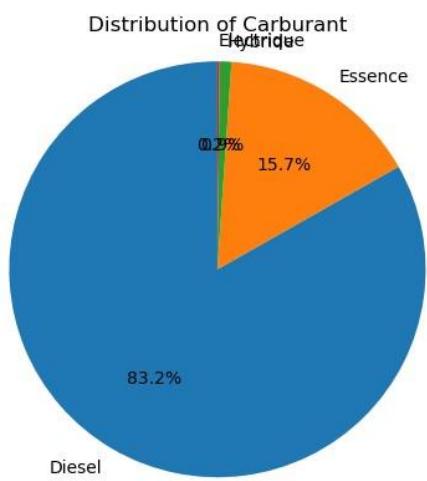
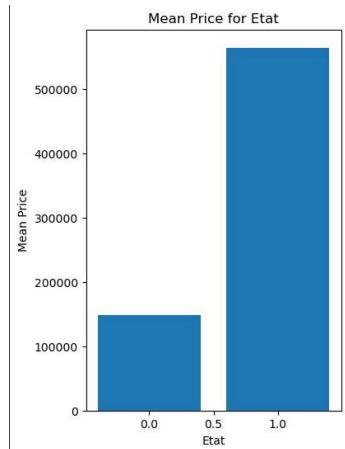
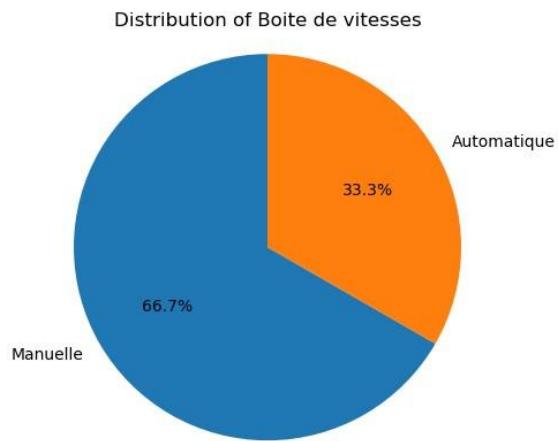
exemple de dataset

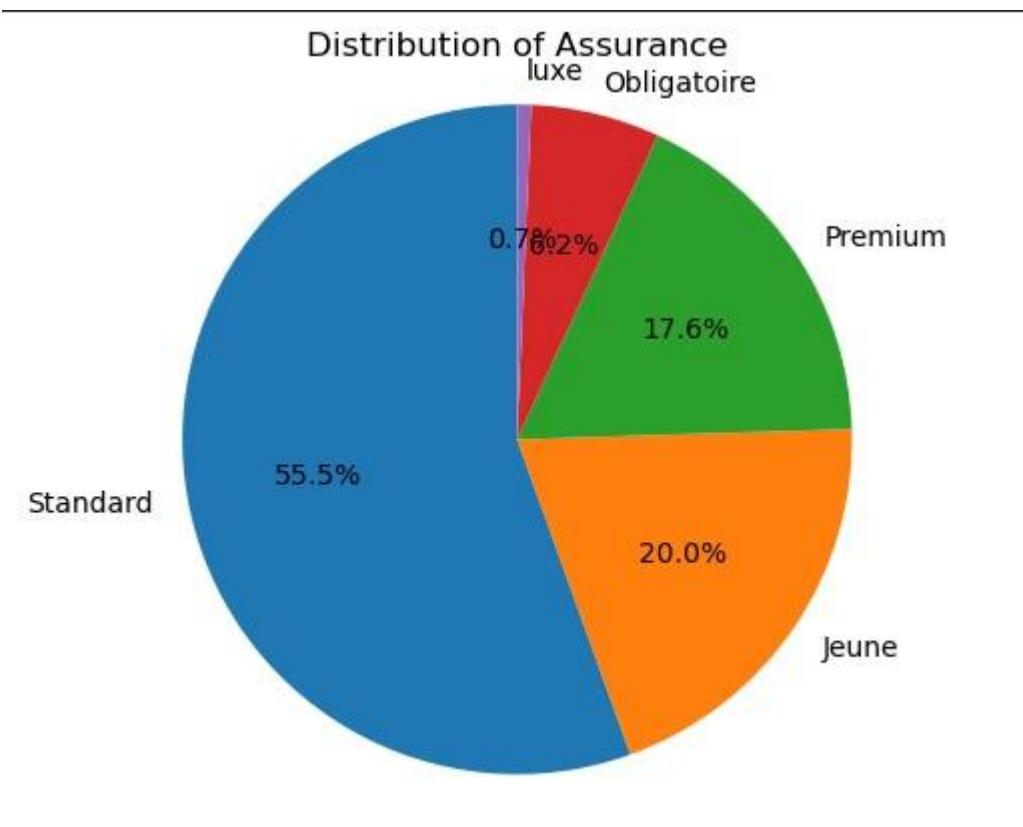
	age	marque	model	Puissance fiscale	Carburant	Année	Boite de vitesses	etat	assurance
0	18	land-rover	Range	12.0	Diesel	2017	Automatique	0	Jeune
1	46	land-rover	Range	12.0	Diesel	2016	Automatique	0	Premium
2	42	bmw	X5	8.0	Diesel	2016	Automatique	0	Premium
3	27	land-rover	Range	8.0	Diesel	2018	Automatique	0	Premium
4	24	mercedes	Classe	8.0	Diesel	2022	Automatique	0	Jeune

4.2. Visualisation des Données

La visualisation des données joue un rôle crucial dans la compréhension approfondie de la structure des informations collectées, permettant ainsi de découvrir des tendances, des corrélations, et d'autres insights pertinents. Dans cette section, nous explorons les différentes visualisations utilisées pour analyser le dataset des contrats d'assurance automobile après le processus de prétraitement.





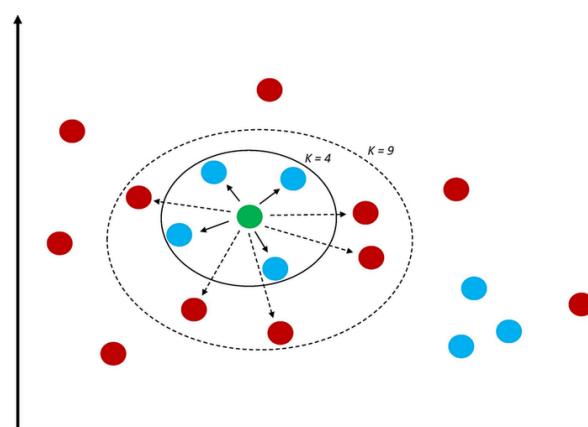


4.3. Entraînement du Modèle

Le processus d'entraînement du modèle de recommandation est une étape cruciale qui repose sur le choix approprié de l'algorithme. Dans notre cas, nous avons opté pour l'algorithme K-Nearest Neighbors (KNN) dans le cadre du filtrage collaboratif. Voici une explication détaillée de la méthodologie employée et des décisions prises pendant cette phase.

4.3.1 Choix de l'Algorithme

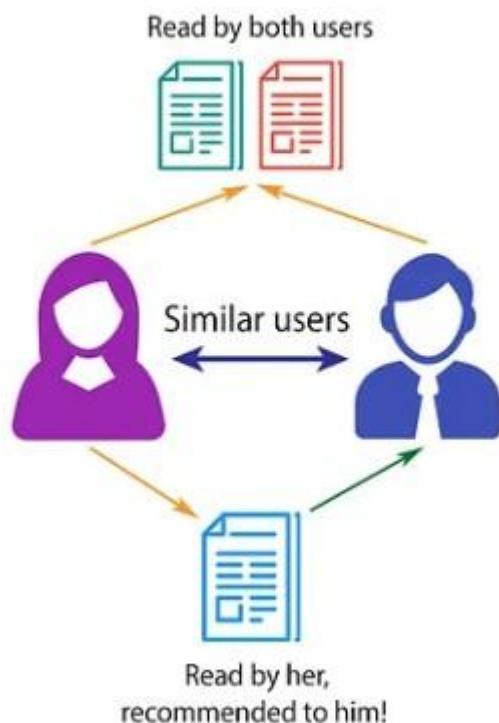
L'algorithme K-Nearest Neighbors (KNN) a été sélectionné en raison de sa simplicité conceptuelle et de sa flexibilité pour modéliser les relations entre les contrats d'assurance automobile. KNN fonctionne en identifiant les contrats similaires en se basant sur la proximité dans l'espace des caractéristiques, permettant ainsi de recommander des contrats en se basant sur les préférences d'utilisateurs similaires.



4.3.2 collaborative filtering

Le filtrage collaboratif user-item est une méthode de recommandation qui analyse les comportements passés des utilisateurs pour prédire leurs préférences. Dans ce cadre, la similarité entre utilisateurs est évaluée, permettant de recommander des items appréciés par des utilisateurs similaires. Cette approche repose sur la notion que des utilisateurs partageant des goûts communs peuvent influencer positivement les choix d'autres utilisateurs, offrant ainsi des recommandations personnalisées et pertinentes en se basant sur des connexions d'utilisateur à utilisateur.

COLLABORATIVE FILTERING



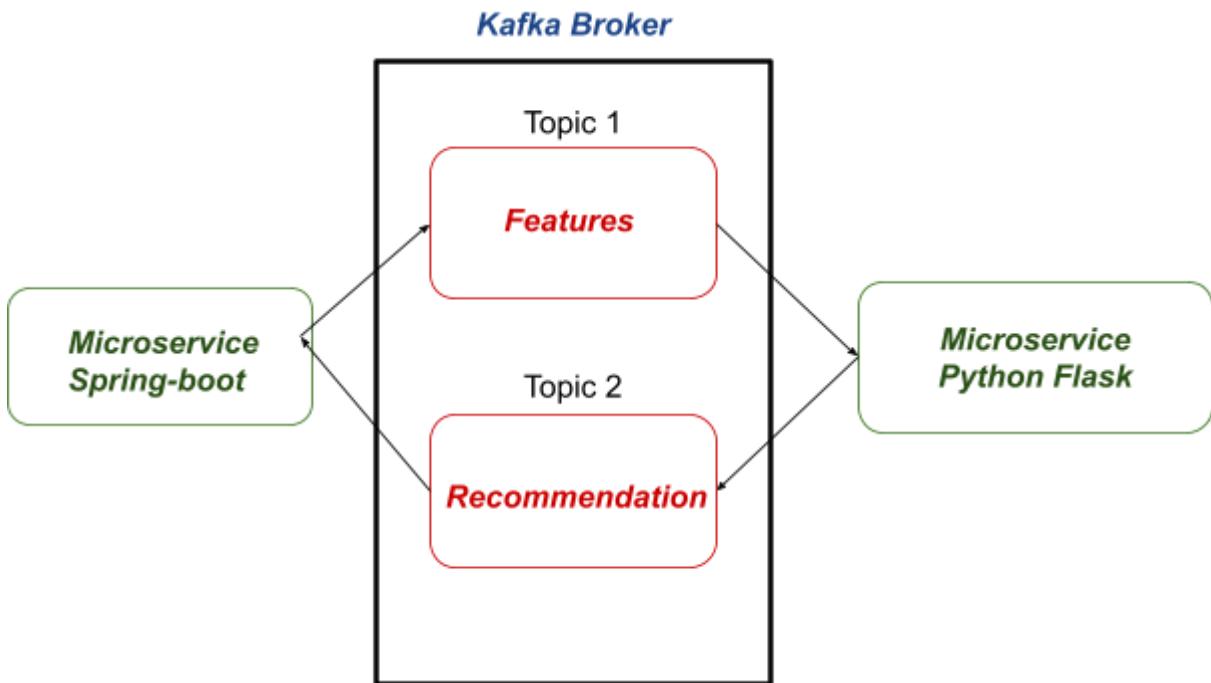
4.3.4 Évaluation du Modèle

Le modèle a été évalué en utilisant des métriques appropriées pour le filtrage collaboratif, telles que la précision, le rappel et la mesure F1. Les performances du modèle ont été analysées en comparant les recommandations générées par le modèle avec les préférences réelles des utilisateurs.

Metrics	Accuracy	Precision	recall	f1-score
Valeurs	93%	89%	85%	87%

5. Communication entre Microservices avec Kafka: Mise en Place d'une Pipeline de Recommandation

La mise en place d'une communication efficace entre microservices est cruciale pour le fonctionnement harmonieux du système de recommandation. Dans notre architecture, Kafka joue un rôle central en facilitant cette communication grâce à la création de deux topics distincts : "**features**" et "**recommendation**".



1. Producer - Extraction des Features (features topic) :

Source des Features : Les données caractéristiques des véhicules sont extraites à partir de diverses sources, puis agrégées.

Envoi vers Kafka : Un microservice producteur envoie ces features vers le topic "features" sur Kafka, assurant ainsi une transmission en temps réel.

2. Kafka Broker :

Gestion des Messages : Le broker Kafka sert de médiateur, gérant les messages provenant du producteur et les acheminant vers les consommateurs appropriés.

3. Consumer - Génération de Recommandations (recommendation topic) :

Réception des Features : Un microservice consommateur récupère les features du topic "features".

Calcul des Recommandations : En utilisant des algorithmes de recommandation, ce microservice génère des suggestions de contrats d'assurance automobile adaptés aux caractéristiques spécifiques des véhicules.

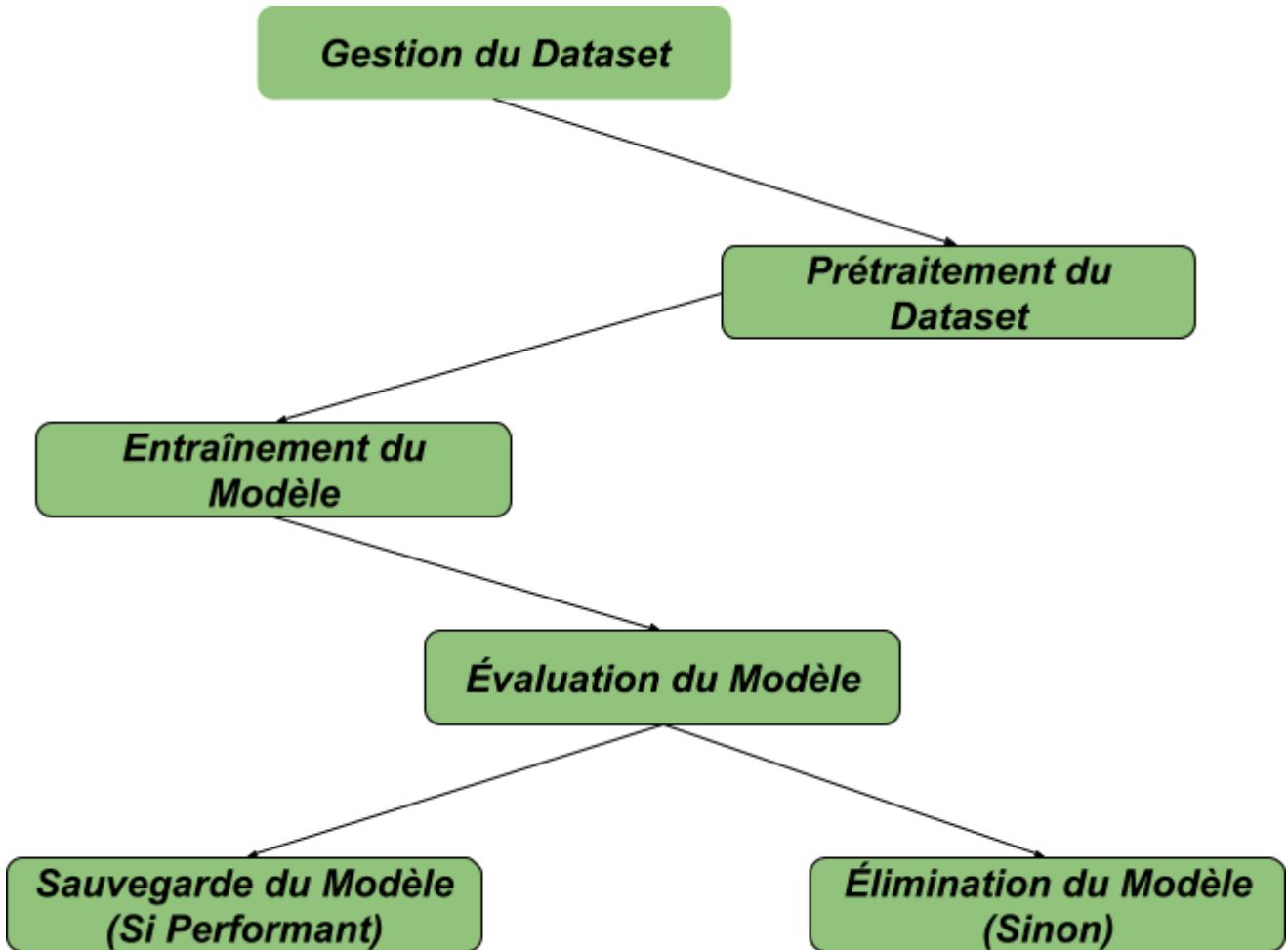
Envoi vers Kafka : Les recommandations sont ensuite publiées sur le topic "recommendation" de Kafka.

4. Consumer Final - Distribution des Recommandations :

Réception des Recommandations : Un dernier microservice consommateur récupère les recommandations depuis le topic "recommendation".

6. Création d'un Pipeline Hebdomadaire avec Airflow

La mise en place d'un pipeline automatisé avec Apache Airflow facilite le flux de travail du système de recommandation dédié aux contrats d'assurance automobile. Voici comment les différentes étapes sont orchestrées dans le pipeline, intégrant l'utilisation de Cassandra et les tâches de prétraitement, d'entraînement, d'évaluation, et de gestion des modèles avec Airflow.



1. Tâche Cassandra - Gestion du Dataset :

Mise à Jour du Dataset : Cassandra est utilisée pour gérer et mettre à jour le dataset des contrats d'assurance. Les avantages d'une base de données NoSQL comme Cassandra résident dans sa capacité à gérer un grand volume de données dynamiques avec une latence faible, offrant ainsi une solution scalable et performante.

2. Tâche Python - Prétraitement du Dataset :

Exécution du Script Python : Un script Python est exécuté pour effectuer le prétraitement du dataset. Cela inclut la fusion des données collectées via le web scraping, l'élimination des valeurs nulles ou dupliquées, la détection et la gestion des valeurs aberrantes, ainsi que la normalisation des données.

3. Tâche Python - Entraînement du Modèle :

Exécution du Script Python : Un autre script Python est lancé pour entraîner le modèle de recommandation. Cela implique l'utilisation de l'algorithme KNN avec les features mises à jour, extraites du dataset géré par Cassandra.

4. Tâche Python - Évaluation du Modèle :

Exécution du Script Python : Un troisième script Python est utilisé pour évaluer les performances du modèle. Cela permet de vérifier la précision des recommandations générées par le modèle entraîné.

5. Tâche Python - Sauvegarde du Modèle (Si Performant) :

Sauvegarde du Modèle : Si le modèle est jugé performant après l'évaluation, il est sauvegardé en tant que nouveau modèle en remplacement de l'ancien. Cela garantit que le système utilise toujours le modèle le plus optimal.

6. Tâche Python - Élimination du Modèle (Sinon) :

Élimination du Modèle : Si le modèle n'atteint pas les critères de performance définis, il est éliminé, et l'ancien modèle déjà en place est conservé pour assurer la continuité du service.

Conclusion

En conclusion, la fusion de microservices, de technologies telles que Kafka, Cassandra, et Airflow, ainsi que l'adoption d'un algorithme de filtrage collaboratif, offre un système de recommandation robuste pour les contrats d'assurance automobile. Cette architecture assure une communication en temps réel, une gestion efficace des données, et une automatisation régulière des tâches, garantissant ainsi des recommandations précises. Le filtrage collaboratif améliore la personnalisation des suggestions. La pipeline hebdomadaire, orchestrée par Airflow, assure une mise à jour régulière du modèle, une évaluation continue, et une gestion flexible du cycle de vie des recommandations. En somme, cette solution présente une approche agile et performante pour répondre aux besoins dynamiques du secteur de l'assurance automobile.