wis

Projet « Développer les composants d'accès aux données »

PRÉSENTATION DU CONTEXTE DU CAHIER DES CHARGES

1. Présentation de l'entreprise à l'origine du besoin

L'entreprise EasyLoc' est une entreprise Lyonnaise de location de voitures à des particuliers. Elle a connu une croissance importante ces dernières années. Sa directrice veut ouvrir une nouvelle agence à Nice et a donc entrepris de numériser les données de son entreprise. Elle a pris conseil sur une ingénieure en SGBD qui lui a conseillé une répartition en base de données SQL et NoSQL.
En l'attente d'équipes de développement supplémentaires, elle a besoin d'une bibliothèque de gestion de base de données qui sera utilisée par les équipes Frontend et Backend pour accéder aux données de ses agences pour la gestion des locations, des véhicules et des paiements.

1.1. Contexte et besoin

Vous devez écrire et documenter une bibliothèque d'accès aux données des agences (décrites plus loin). Cette bibliothèque s'interfacera avec les différents outils développés dans le futur et devra donc être :

- ✓ documentée pour faciliter le travail des développeurs/développeuses des autres applications ;
- ✓ sécurisée ;
- ✓ fournie avec un jeu de test unitaires pour pouvoir refactoriser et maintenir la bibliothèque sans risques ;
- ✓ évolutive sans un coût technologique trop élevé.

1.2. Architecture matérielle

L'architecture choisie par la directrice de EasyLoc' est la suivante :

- ✓ 1 serveur applicatif sur lequel tournerons les outils de gestions des contrats de location et de véhicules.
- ✓ 1 serveur SQL Srv contenant les tables Contract et Billing (voir plus loin).
- ✓ 1 serveur MongoDB contenants les Customers et les Vehicles (voir plus loin).

1.3. Architecture des bases de données

Tables de la base de données SQL Srv

Table **Contract** : contient les données des contrats de locations.
Champs :
- id (INT - clé unique du contrat)
- vehicle_uid (CHAR(255) - uid du Vehicle associé au contrat)
- customer_uid (CHAR(255) - uid du Customer associé au contrat)
- sign_datetime (DATETIME - Date + heure de signature du contrat)
- loc_begin_datetime (DATETIME - Date + heure de début de la location)
- loc_end_ datetime (DATETIME - Date + heure de fin de la location)
- returning_datetime (DATETIME - Date + heure de rendu du véhicule)
- price (MONEY - Prix facturé pour le contrat)

Table **Billing** : Contient les données de payement des contrats. Payement en plusieur fois possible.
Champs :
- ID (INT - clé unique du payement)
- Contract_id (INT - clé unique du contrat concerné par le payement)
- Amount (MONEY - Montant payé)

Schémas de documents dans la base MongoDB

Table **Customer** : Contient les données clients
- uid (UUID - Identifiant unique du document)
- first_name (CHAR(255) - Nom)
- second_name (CHAR(255) - Prénom)
- address (CHAR(255) - Adresse complète)
- permit_number (CHAR(255) -numéro de permis)

Table **Vehicle** : Contient les données associées à un véhicule
- uid (UUID - Identifiant unique du document)
- licence_plate (CHAR(255) - Immatriculation du véhicule)
- informations (TEXT - Notes sur le véhicule, par exemple dégradations)
- km (INT - Kilométrage du véhicule)

2. Expression de la demande – réalisation attendue de l'apprenant

2.1. Préambule et Remarques générales

Les différents besoins exprimés ci-dessous devront être comblés en prenant garde à garder une architecture de code évolutive : la question « que se passerait-il si on ajoutait une nouvelle table ou un nouveau SGBD ?» devra rester en tête lors de la conception de la bibliothèque. On utilisera pour cela le paradigme objet.

On sera particulièrement attentif à la sécurité du code source (notamment injections SQL, à la validation des données d'entrée...).

Il est important que le code d'une bibliothèque qui va être utilisé par plusieurs applications soit testé et documenté. Pour chaque cas d'usage, il conviendra d'écrire des jeux de tests unitaires fonctionnels et de sécurité.

Lors du développement, on utilisera un outil de gestion du code source (type Git) afin de garder trace des contributions et de permettre les rollbacks.

Il pourra être utile, dans le cadre de la présentation du projet, de réaliser un outil en ligne de commande basique pour présenter les différentes fonctionnalités de la bibliothèque réalisée.

2.2. Cahier des charges

wis

Partie 1 : base de donnée SQL Srv

Étant donné un couple utilisateur/mot de passe, établir une connexion sécurisée au SGBD.

Table **Contract**

- Pouvoir créer la table Contract si elle n'existe pas.
- Pouvoir accéder à un contrat en particulier à partir de sa clé unique.
- Pouvoir créer un nouveau contrat à la date actuelle, à une date autre.
- Pouvoir supprimer/modifier un contrat existant.

Table **Billing**

- Pouvoir créer la table Billing si elle n'existe pas.
- Pouvoir accéder à un payement en particulier à partir de sa clé unique.
- Créer/Modifier/supprimer des données payement.

Partie 2 : Recoupement d'informations SQL

- Pouvoir lister tous les contrats associées à un uid de Customer.
- Pouvoir lister les locations en cours associées à un uid de Customer.
- Pouvoir lister toutes les locations en retard (une location est dite « en retard » si returning_datetime est postérieur à loc_end_datetime de plus d'1 heure).
- Pouvoir lister tous les payements associés à une location.
- Pouvoir vérifier qu'une location a été intégralement payée.
- Pouvoir lister toutes les locations impayées.
- Pouvoir compter le nombre de retard entre deux dates données.
- Pouvoir compter le nombre de retard moyens par Customer.
- Pouvoir lister tous les contrats où un certain véhicule a été utilisé.
- Pouvoir avoir la moyenne du temps de retard par véhicule.
- Pouvoir récupérer tous les contrats regroupés par véhicules ou par client/cliente.

Partie 3 : Base NoSQL

Étant donné un couple utilisateur/mot de passe, établir une connexion sécurisée à une instance MongoDB.

- Pouvoir créer/modifier/supprimer un document Customer et Vehicle.
- Pouvoir rechercher un Customer à partir de son nom+prénom.
- Pouvoir rechercher un véhicule à partir de son numéro d'immatriculation.
- Pouvoir compter les véhicules ayant plus (respectivement moins) d'un certain kilométrage.

Extensions possibles

✓ Gestion de différents types de SGBD

- La directrice de **EasyLoc'** décide d'utiliser une base PostgreSQL à coté de sa base SQL Srv, il faut modifier le code pour prendre en compte des SGBD différents tout en gardant la même interface.
- Même question en ajoutant une nouvelle base NoSQL (type ElasticSearch).

✓ Gestion du sharding.

En raison du grand nombre de paiement, du temps de traitement des données de la base SQL Billing, il est envisagé de répartir les données sur plusieurs bases situées sur des machines distinctes. Proposer une solution et une implémentation pour le faire.

2.3. Livrables

- ✓ Bibliothèque de gestion des bases de données.
- ✓ Documentation exhaustive de votre bibliothèque.
- ✓ Rapport présentant vos choix en terme d'architecture de code.
- ✓ Tests unitaires pour toutes les parties du cahier des charges.

-- Fin du document --