Vanessa Pham
206
Parcours C

Avril 2023

RAPPORT ARCHITECTURE LOGICIEL

Réservation/emprunt/retour médiathèque

Université de Paris

SOMMAIRE

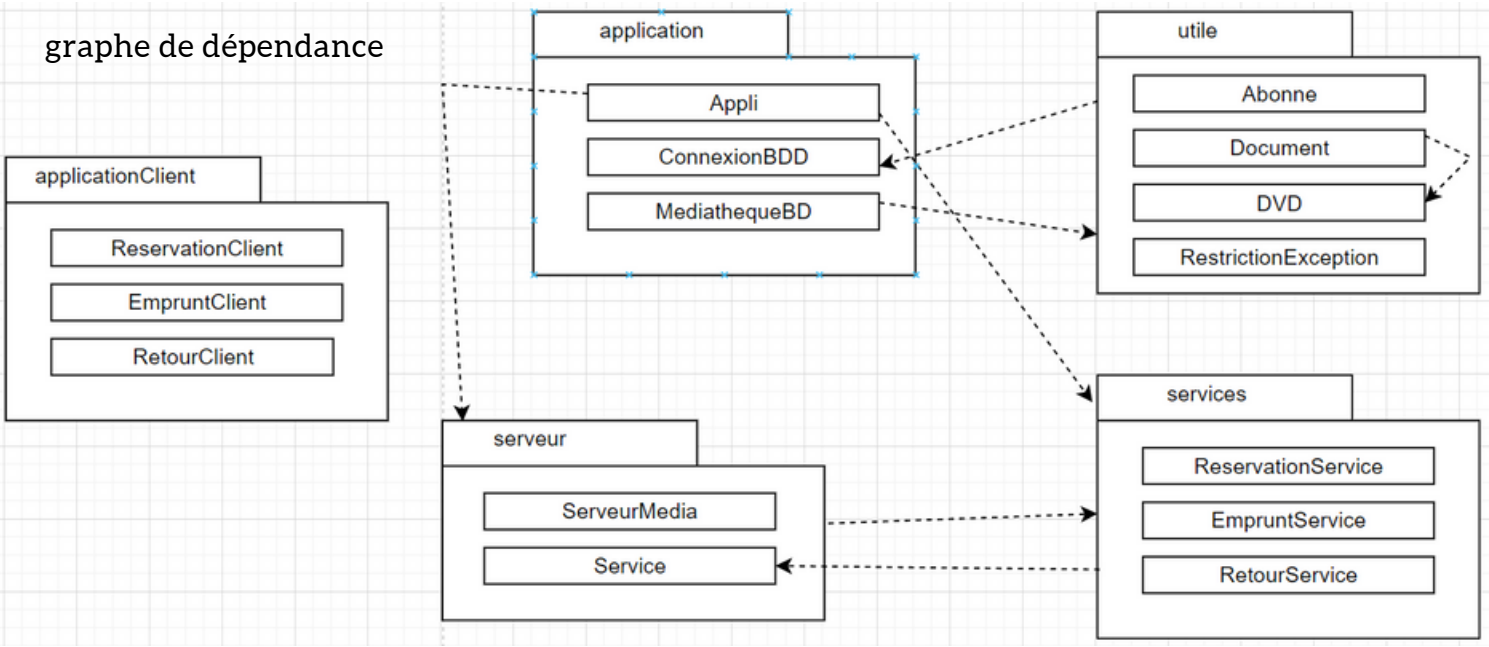
DESCRIPTION DU PROJET

Ce projet consiste à réaliser une application en Java, qui gère les réservations, emprunts, retours de documents pour une médiathèque. Cette gestion doit se faire à l'aide de la mise en place de Serveurs, Services, et de Sockets. Les 3 opérations correspondront à une application cliente chacune se connectant à un port défini et échangeant les données avec le service concerné. Les demandes de réservation sur le port 3000, les demandes d'emprunt sur le port 4000 et les retours sur le port 5000.
Il faudra également réaliser une connexion avec une base de données que nous créerons nous même, seules 2 tables sont nécessaires avec un schéma relationnel permettant de savoir qui a emprunté quoi.

COTÉ SERVEUR

structuration de code, découplage, code stable, graphe de dépendance, inter-package, mise en œuvre-utilisation de bibliothèques logicielles

graphe de dépendance



Le package serveur encapsule la Classe ServeurMedia et la classe Service, ces classes sont les bases de ce projet. La classe ServeurMedia permet de créer les 3 serveurs pour les 3 opérations proposées. En découplant le serveur d'un seule service, à l'aide de la technique suivante : créer un attribut de type Service à la classe ServeurMedia, puis de la passer en paramètre dans le constructeur. On peut alors créer des serveurs avec seulement le service qui change.

Le package serveur est un package qu'on aurait pu export en .jar, mais pour facilité la lecture je les ai laisser en .java, ce qui résulte d'un paramètre Service, avec Service étant une interface à la place d'un Class<? extends Service> . J'ai également décider de faire passer un paramètre de type MediathequeBD dans le constructeur des serveurs pour afin de créer une base de données pour notre médiathèque commune pour tous les serveurs.

Les services qui sont appelés dans le constructeur des serveurs implémentent l'interface Service et comme leur nom l'indique elles correspondent aux 3 opérations ReservationService, EmpruntService, RetourService. Ces services sont dans le package services et dans ces classes, sont implémentés les procédures (Threads) à lancer respectivement. C'est-à-dire que lorsque qu'on veut effectuer une opération, il faut appeler le serveur qui attend sur un port donné, afin de lancer le service, le thread à effectuer qui lui correspond.

Dans le package application en plus du main qui va créer les serveurs, il y a une classe qui a la fonction de faire la connexion entre le code java et la base de données (SGBD : Access), à l'aide des outils java.sql.Connection, java.sql.DriverManager etc. Et une classe MediathequeBD qui donnera la structure des traitements et stockages des données.

Le package utile réuni les objets du monde réel qui seront utilisés et avec des fonctions qui leurs est propre

ECHANGES CLIENT/SERVEUR

: normalisé, bttp1.0 ou 2.0, bibliothèque de cryptage

Dans le package applicationClient, j'ai séparé les clients qui souhaitent utiliser différents, j'ai donc crée 3 classes de client, ReservationClient, EmpruntClient, RetourClient. ReservationClient n'attend que 2 entrées claviers (Emprunt aussi) car ce qui intéresse le service c'est le numero de l'abonné et le numéro du DVD, mais retour lui n'attend qu'une entrée clavier, le numéro du DVD, pour différencier le nombre d'entrée exacte j'ai donc séparé en 3 clients, les clients qui appellent le port et serveur correspondant.
Les outils principalement utilisés sont les : BufferedReader et PrintWriter ,
BufferedReader sin = new BufferedReader (new InputStreamReader(socket.getInputStream ( )));
PrintWriter sout = new PrintWriter (socket.getOutputStream ( ), true);

LIEN ENTRE L'APPLICATION ET LA BASE DE DONNÉES

(rapport modèle objet-modèle relationnel, chargement/sauvegarde au début/à la fin/en temps réel ?

Dans le package c'est la classe ConnexionBDD, possède une méthode getConnexion() qui effectue la 1ere connexion avec la base de données, traiter sous Access. À l'aide des outils java.sql.Connection, java.sql.DriverManager, j'ajuste les liens et le chemin jusqu'à la base de donnée. Ensuite à la chaque fois qu'il classe à besoin de ce lier et d'utiliser la base de données elle doit créer une instance Connection con = ConnexionBDD.getConnexion(), pour ce connecter et pourvoir effectuer tous les traitement SQL qu'elle souhaite. En partant de cette connexion, on peut faire des SELECT, UPDATE dans le code Java et apporter des modifications sur la base de données qui a été initier dans la classe ConnexionBDD,
Lors du lancement des serveurs dans le main Appl, la base est chargée à part pour ensuite être appeler en paramètre dans les serveurs pour que tous les servurs et donc les services puissent travailler sur une même base de données.

NumAbonne	NomAbonne	DateNaiss
0	LOLA	01/01/2020
1	VANESSA	01/01/2000
2	ALICE	01/01/2020
3	BOB	02/01/2000
4	OLI	03/01/2000
5	FARA	04/01/2000

2 Tables :
Abonne(NumAbonne, NomAbonne, DateNaiss)
DVD(numDVD, titre, adulte, numEmprunt, numReserveur)

numDVD	titre	adulte	numEmprunt	numReserveur
1	Avatar	☐	1	
2	Titanic	☐		2
3	Intouchables	☐		
4	Matrix	☐		
5	Shining	☑	3	
6	Carrie	☑		4
7	Frankenstein	☑		
8	Eraserhead	☑		

Commentaires : J'ai fait le choix de créer une table DVD et non document dans le cadre du projet, cependant pour une meilleur maintenance évolutive on aurait pu faire une table Document avec un attribut "type" de document pour différencier les différents documents possibles (livres, dvd, cd)

CONCURRENCE

: identification des ressources partagées et sections critiques, gestion de la concurrence

La classe DVD qui implémente l'interface Document permet à la classe MediathequeBD de récuperer et stocker les DVD de la base de données dans ses méthodes (dans notre cas nous avons une liste "public List<Document> tousLesDvdDisponibles()" qui peut être appeler par plusieurs thread/services en même tant. Si elles utilisent la ressource partagée qu'est cette liste de DVD, ce points d'accès à la base de donnée, on peut remarquer des points critiques, c'est-à-dire de concurrence de priorité, ou on doit définir les droits sur qui commence sa tâche et si il peut la finir. Pour rendre le plus thread Safe possible cette concurrence, ma solution a été de rajouter des synchronized à chaque fois qu'une classe (Service) et sa méthode utilise la ressource partagées afin d'empêcher un autre service de toucher la ressource en parallèles en même temps.

MAINTENANCE ÉVOLUTIVE

Quels efforts sont nécessaires pour ajouter des livres, cds ?
Il faudra crée d'autre classe qui implémente l'interface Document et écrire, les spécifications propres à chaque type de document. Et faire attention aux attributs lorsqu'on instancie des objets Document car ils m'ont pas tous les mêmes attributs (DVD a l'attribut "adulte" qui lui est propre). Il faudrait aussi rajouter un attribut "type" qui sera l'indicateur le plus important pour différencier les documents.

Pour faire passer votre application en application web au lieu de serversocket/socket ?
On peut utiliser les outils : de servlet, c'est une classe Java qui permet de créer dynamiquement des données au sein d'un serveur HTTP. Ces données sont souvent au format HTML.
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;
(information trouvé avec des recherches personnelles mais je ne sais pas encore les appliquer)

Pour ajouter un nouveau service au port 6000 ? qu'est-ce qui reste tel quel dans chacun des cas ?
Il faut créer une classe qui correspondra au nouveau service. La classe ServeurMedia ne change pas, lorsqu'on souhaite créer le serveur sur le port 6000 pour le nouveau service on utilise le constructeur de ServeurMedia et on passe en paramètre le nouveau service qui devra implémenter l'interface Service.
(ou modifier et faire passer la classe avec Class< ? extends Service>.)

MISE EN PLACE DES BRETTESOFT

NON TRAITER

BILAN

Réussite : Le projet a été très formateur sur la compréhension des Serveurs, Services, Socket. J'ai pu tester et chercher par moi-même les notions et concepts traiter en cours et en TPs. J'ai appris à coder des serveurs découplés, à faire le liens entre une base de données et le java, traiter la concurrence et les synchronized et plus encore.

Point à améliorer : l'algorithmie derrière les services et le découpage peut être améliorer pour être plus évolutif. De plus, il manque certains points qui étaient indiqués dans l'énoncé qui ne sont pas développer comme les messages de Restriction pour les cas ou « ce DVD est déjà emprunté », « ce livre est réservé jusqu'à 12h25 ». Il n'y a que celle de la limite d'âge.
Il manque également la fonctionnalité qui démarre un timer afin d'encadrer la situation ou 'l'abonné doit passer à la médiathèque dans les 2h suivant l'emprunt (sinon la réservation est annulée)".
Il n'y a pas non plus les Certifications BretteSoft pour compléter le projet.

Et que finalement corriger le fait que le code ne debugge pas.