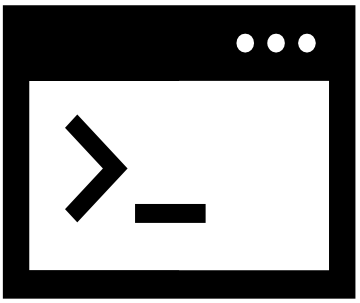
TP 2-Service web SOAP



- **Master Informatique - Sujet :** - **Service web SOAP**

- **Membres** :

- 21908889 : Adam DAIA **PARCOURS GL**
- 22014701 : Mohammed Dafaoui **PARCOURS GL**

2023/2024

Table des matières :

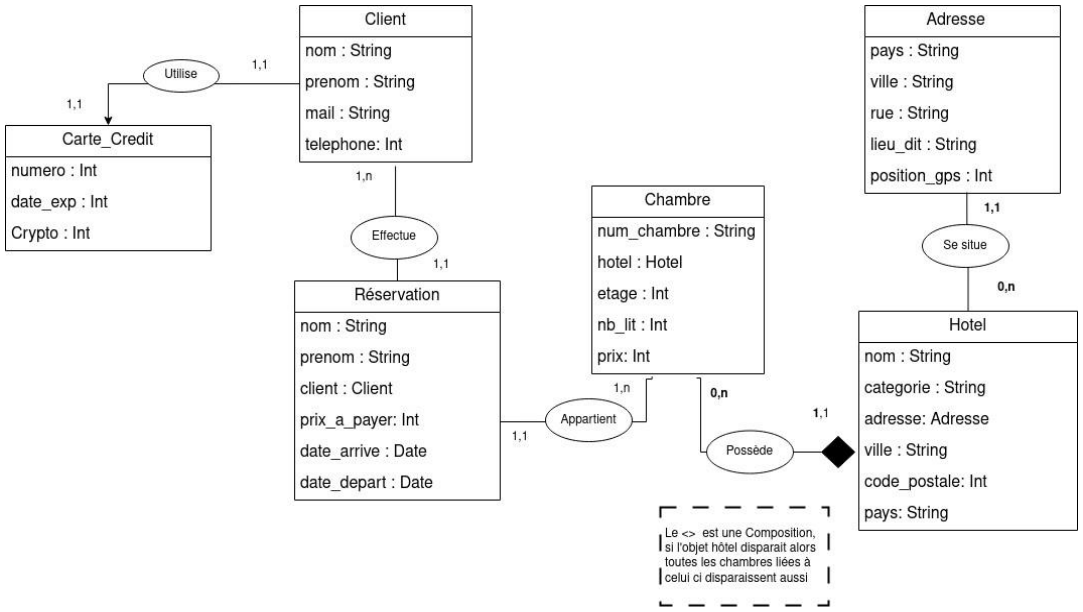
I/ Introduction

Nous avons développé une application de réservation d'hôtels en ligne qui simplifie le processus de réservation pour les utilisateurs. Cette application permet aux utilisateurs de saisir des informations relatives à leur séjour, telles que les dates d'arrivée et de départ, un intervalle de prix souhaité, ainsi que le nombre de personnes à héberger.

Pour effectuer la recherche d'hôtels et la réservation, notre application utilise des services web via le protocole SOAP. Grâce à cette intégration, les utilisateurs obtiennent une liste d'hôtels qui correspondent à leurs critères. Ils peuvent ensuite accéder à des informations plus détaillées sur chaque établissement avant de finaliser leur réservation. Cela garantit une expérience conviviale et efficace pour les utilisateurs tout au long du processus de réservation.

La première section de notre rapport se consacre à la version non distribuée de l'application. Dans cette partie, nous présentons une conception en UML de l'application qui n'implique pas la distribution, suivie de l'implémentation de cette conception sans intégration de la gestion de la persistance des données.

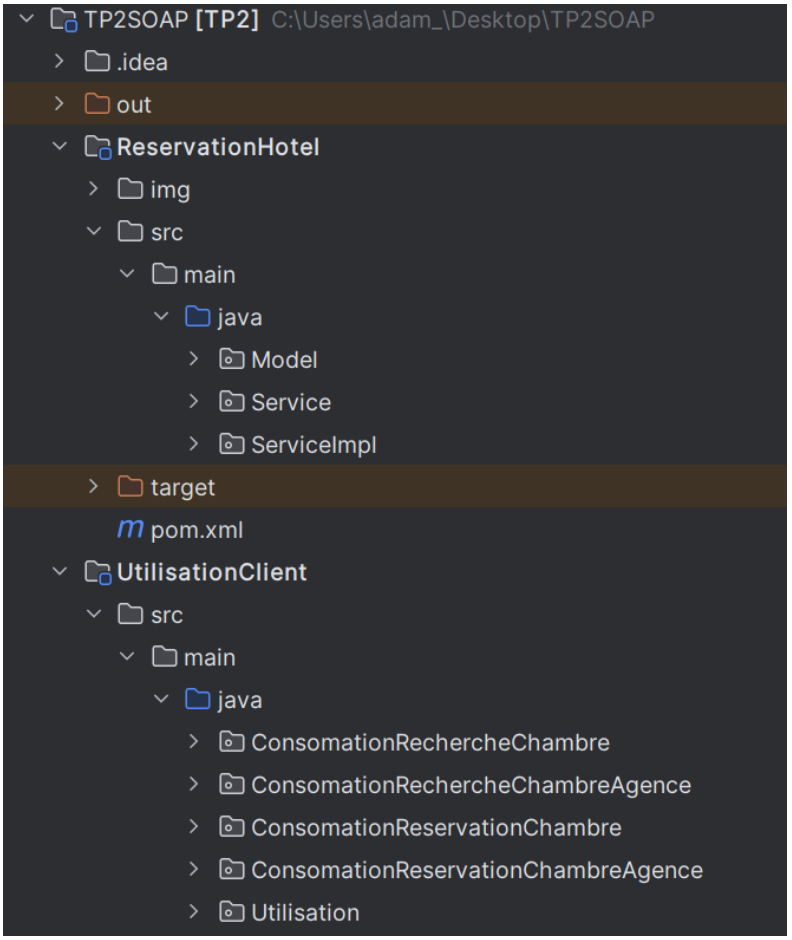
1) Conception en UML :



2) Implémentation de la conception UML

Nous avons réalisé l'implémentation des classes de la conception UML en définissant les attributs, les constructeurs, les méthodes getters et setters, ainsi que des méthodes spécifiques pour chaque classe. Cette démarche nous a permis de créer des instances des classes et d'effectuer diverses opérations sur ces instances, le tout sans nécessité de gérer la persistance des données via une base de données.

II/ Architecture du Projet



Le projet se divise en deux parties distinctes.
D'un côté, nous avons le côté serveur nommé "**ReservationHotel**". Cette partie comprend une section "Model" qui contient le point d'entrée du programme, généralement le '*main*' avec la méthode *publish* pour exposer les services web. En outre, cette section comprend les classes d'implémentation essentielles telles que *Hotel*, *Reservation*, *CarteCredit*, *Client*, etc., qui constituent les éléments clés du système. Ces classes d'implémentation sont responsables de la gestion des données et de la logique métier du côté serveur, assurant ainsi le bon fonctionnement de l'application.

D'un autre côté, nous avons l'implémentation côté client nommée "**UtilisationClient**," qui inclut les classes générées par wsimport ainsi que le client lui-même, permettant de consommer les services web fournis par le serveur. Cette architecture à deux niveaux assure une séparation claire des responsabilités entre le serveur et le client, facilitant ainsi le développement et la maintenance du projet.

III/ Explication des classes et interfaces

1/ Reservation Hotel :

Dans ce repertoire, nous avons organisé notre code en trois packages distincts :

"*Model*", "*Service*" et "*ServiceImpl*."

Ces packages jouent des rôles spécifiques dans le développement de notre application de réservation d'hôtels en ligne.

Package Model :

Dans le package Model, nous avons créé des classes pour modéliser les entités principales de notre système de réservation d'hôtels en ligne. Chacune de ces classes est conçue pour représenter des informations spécifiques, et elles partagent une structure commune qui comprend des attributs, des constructeurs pour instancier ces classes, et des méthodes getter pour accéder aux données.

1. **Classe Hotel** :

   La classe *Hotel* modélise un établissement d'hébergement. Elle contient une liste de chambres représentées par des instances de la classe *Chambre*. De plus, la classe *Hotel* propose une méthode *ajouterChambre* pour l'ajout de chambres à la liste et des méthodes getter pour accéder aux données de l'hôtel.

2. **Classe Client** :

   La classe *Client* représente un utilisateur de notre application. Elle contient une instance de la classe *CarteCredit* pour gérer les informations de carte de crédit du client, ainsi qu'une liste de réservations représentées par des instances de la classe *Reservation*. La classe *Client* propose une méthode *ajouterReservation* pour ajouter des réservations à sa liste et des méthodes getter pour accéder aux données du client.

3. **Classe Chambre** :

   La classe *Chambre* modélise une chambre dans un hôtel. Elle contient une référence à l'hôtel auquel elle appartient et propose des méthodes getter pour accéder aux informations de la chambre. De plus, elle inclut une méthode *setDisponible* pour gérer la disponibilité de la chambre.

4. **Classe CarteCredit** :

   Cette classe représente les informations d'une carte de crédit et est utilisée pour gérer ces données.

5. **Classe Adresse** :

   La classe *Adresse* modélise une adresse et offre des méthodes pour accéder à ces informations.

6. **Classe Reservation** :

   La classe *Reservation* contient une référence au client qui a effectué la réservation. Elle est utilisée pour modéliser une réservation de chambre d'hôtel et fournit des méthodes pour accéder aux détails de la réservation.

7. **Classe Offre** :

   La classe *Offre* représente une offre de chambre d'hôtel. Elle contient des informations sur la disponibilité et l'expiration de l'offre. De plus, elle propose une méthode i*sDisponible* pour vérifier la disponibilité de l'offre pour une période donnée.

8. **Classe Agence** :

   La classe *Agence* est utilisée pour modéliser les agences proposant des réservations d'hôtels. Elle comprend des méthodes getter et setter pour accéder et modifier les valeurs de ses attributs.

Ces classes du package Model forment la base de données métier de notre application de réservation d'hôtels en ligne, permettant de gérer les entités clés de manière organisée et efficace

Package Service :

Dans le package "Service," nous avons défini des interfaces pour décrire les services web que notre application doit fournir. Voici une description de ces interfaces :

1. **Interface RechercheChambreHotel :**

   Cette interface détermine un service web permettant la recherche de chambres d'hôtel. Elle expose une méthode nommée "**rechercherChambres**" qui requiert en paramètres un prix minimum, un prix maximum, et le nombre de lits. Grâce à cette interface, les clients peuvent effectuer des recherches pour trouver des chambres d'hôtel correspondant à leurs critères.

2. **Interface ReservationChambre :**

   L'interface *ReservationChambre* définit un service web dédié à la réservation de chambres d'hôtel. Elle inclut une méthode *reserverChambre* qui prend en entrée le numéro de la chambre, le nom et le prénom du client, ainsi que les dates d'arrivée et de départ. Cette interface permet aux utilisateurs de réserver des chambres d'hôtel en spécifiant les détails de leur séjour.

3. **Interface RechercheChambreAgence** :

   Cette interface, *RechercheChambreAgence*, est conçue pour la recherche de chambres d'hôtel en collaboration avec une agence de voyage. Elle propose une méthode nommée *rechercherChambresAgence* qui requiert des paramètres similaires à l'interface *RechercheChambreHotel*, à savoir un prix minimum, un prix maximum, et le nombre de lits. Cela permet aux utilisateurs de rechercher des chambres en partenariat avec une agence de voyage.

4. **Interface ReservationChambreAvecAgence :**

   L'interface *ReservationChambreAvecAgence* définit un service web pour la réservation de chambres d'hôtel en collaboration avec une agence de voyage. Elle expose une méthode nommée *reserverChambreAvecReduction*, qui prend en paramètres le nom de l'hôtel, le numéro de la chambre, le nom et le prénom du client, les dates d'arrivée et de départ, ainsi que le pourcentage de réduction accordé par l'agence de voyage. Cette interface permet aux utilisateurs de bénéficier de réductions spécifiques lors de la réservation de chambres en partenariat avec une agence de voyage.

Ces interfaces jouent un rôle central dans la définition des services web offerts par notre application, permettant aux utilisateurs de rechercher et de réserver des chambres d'hôtel, que ce soit directement ou en partenariat avec des agences de voyage. Les annotations telles que **@WebService**, **@WebMethod**, et **@WebParam** sont utilisées pour spécifier leur fonction en tant que services web.

Package ServiceImpl :

Dans le package *ServiceImpl*, nous avons fourni les implémentations des interfaces de services web définies dans le package *Service*. Voici une description des classes d'implémentation :

1. **Classe RechercheChambreHotelImpl** :

   La classe *RechercheChambreHotelImpl* implémente l'interface *RechercheChambreHotel*. Elle gère une liste d'instances de la classe *Chambre* représentant les chambres disponibles. Cette classe fournit une implémentation de la méthode *rechercherChambres* définie dans l'interface. Cette méthode prend en paramètres un prix minimum, un prix maximum, et le nombre de lits, et elle retourne une liste de chambres répondant aux critères de recherche spécifiés.

2. **Classe ReservationChambreImpl** :

   La classe *ReservationChambreImpl* implémente l'interface *ReservationChambre*. Elle gère à la fois une liste de chambres (*Chambre*) et une liste de clients (*Client*). Cette classe fournit une implémentation de la méthode *reserverChambre* définie dans l'interface. La méthode prend en paramètres le numéro de la chambre, le nom et le prénom du client, ainsi que les dates d'arrivée et de départ. En fonction de ces informations, elle retourne une chaîne de caractères indiquant si la réservation a été effectuée avec succès.

3. **Classe RechercheChambreAgenceImpl** :

   La classe *RechercheChambreAgenceImpl* est une implémentation de l'interface *RechercheChambreAgence*. Elle est chargée de la recherche de chambres d'hôtel en collaboration avec une agence de voyage, en utilisant les critères de recherche spécifiés.

4. **Classe ReservationChambreAvecAgenceImpl** :

   La classe *ReservationChambreAvecAgenceImpl* est une implémentation de l'interface *ReservationChambreAvecAgence*. Elle doit fournir une implémentation de la méthode *reserverChambreAvecReduction* définie dans l'interface. Cette méthode gère la réservation de chambres d'hôtel avec une réduction accordée par l'agence de voyage. Elle retourne une chaîne de caractères indiquant si la réservation a été effectuée avec succès.

Ces classes d'implémentation utilisent l'annotation **@WebService** pour indiquer qu'elles sont des services web. De plus, elles utilisent l'attribut **endpointInterface** pour spécifier l'interface de service web qu'elles implémentent. Ces implémentations jouent un rôle essentiel dans la mise en œuvre des fonctionnalités de notre application de réservation d'hôtels en ligne.

2/ Utilisation Client :

Dans ce répertoire, nous avons plusieurs packages, dont Utilisation qui contient les classes *myClient* et *myAgence*, ainsi que deux autres packages générés automatiquement par *wsimport* à partir du WSDL des services web. Les classes *myClient* et *myAgence* sont responsables de la consommation des services web de notre application de réservation d'hôtels en ligne.

Package "utilisation" :

Dans ce package, nous avons créé deux classes, *myClient* et *myAgence*, qui permettent d'interagir avec les services web de notre application pour rechercher et réserver des chambres d'hôtel.

Classe myClient :

La classe *myClient* est la classe principale de l'application côté client. Elle utilise les services web de recherche et de réservation de chambres d'hôtel pour effectuer des opérations de recherche et de réservation de chambres.

Pour les opérations de recherche, la classe instancie le service de recherche de chambres en utilisant l'URL du service web. Ensuite, elle obtient un proxy pour le service en utilisant la méthode *getRechercheChambreHotelImplPort*. La classe utilise ce proxy pour appeler la méthode *rechercherChambres*, en fournissant les critères de prix minimum, prix maximum et nombre de lits, et elle imprime les résultats sur la console.

Pour les opérations de réservation de chambres, la classe crée une instance du service de réservation de chambres en utilisant l'URL du service web, puis obtient un proxy pour le service à l'aide de la méthode *getReservationChambreImplPort*. Elle utilise ce proxy pour effectuer deux réservations de chambres avec différentes informations de client et imprime les résultats sur la console.

2. **Classe "myAgence"** :

La classe "myAgence" joue un rôle central en tant que gestionnaire d'hôtels et consommateur de services web pour notre application de réservation d'hôtels en ligne. Contrairement à "myClient," qui fonctionne comme un hôtel individuel, "myAgence" gère plusieurs hôtels et utilise les services web pour coordonner les opérations de recherche et de réservation sur une plus grande échelle.

Pour les opérations de recherche, "myAgence" crée des instances du service de recherche de chambres en utilisant l'URL du service web de chaque hôtel géré. Ensuite, elle obtient des proxies pour chaque service en utilisant la méthode appropriée. La classe "myAgence" peut ainsi coordonner des recherches de chambres auprès de multiples hôtels en fonction des critères spécifiques de chaque client.

Pour les opérations de réservation de chambres, "myAgence" crée des instances du service de réservation de chambres de chaque hôtel géré, en utilisant les URL correspondantes. Elle obtient ensuite des proxies. Cette approche permet à "myAgence" de gérer la réservation de chambres dans différents hôtels tout en profitant des réductions offertes par les agences de voyage.

Ces classes utilisent les classes générées automatiquement par "wsimport" à partir du WSDL des services web pour interagir avec les services web de l'application de réservation d'hôtels en ligne. "myClient" et "myAgence" permettent aux utilisateurs d'effectuer des opérations de recherche et de réservation de chambres, que ce soit directement ou en collaboration avec une agence de voyage pour bénéficier de réductions spécifiques.

IV/ Fonctionnalités clés

Service de Recherche de Chambres

Description : Le service de recherche de chambres permet aux utilisateurs de rechercher des chambres d'hôtel en fonction de divers critères, notamment le prix minimum, le prix maximum et le nombre de lits. Cette fonctionnalité simplifie la tâche des utilisateurs en leur permettant de trouver rapidement des chambres qui correspondent à leurs besoins et à leur budget.

Interface Utilisée : *RechercheChambreHotel*

Méthode :

```
List<Offre>  rechercherChambre(
                    float prixMinimum,
                    float prixMaximum,
                    int nbLit
                    )
```

Paramètres :

- prixMin : Prix minimum que l'agence est prête à payer par nuit.
- prixMax : Prix maximum que l'agence est prête à payer par nuit.
- nbLit : Nombre minimum de lits requis dans la chambre.

Fonctionnement :

L'utilisateur peut spécifier les critères de sa recherche, tels que le prixMin, prixMax ainsi que le nbLit. Le service vérifie les chambres et effectue la réservation. En cas de succès, le service retourne une liste d'offres, chaque offre étant caractérisée par un identifiant unique, une date de disponibilité, une date d'expiration, un prix et un nombre de lit. Cette fonctionnalité permet aux hôtels de proposer des offres de chambres d'hôtel correspondant aux besoins spécifiques de leurs clients, facilitant ainsi la planification de séjours à l'hôtel.

Service de Réservation de Chambres

Description : Le service de réservation de chambres permet aux utilisateurs de réserver une chambre d'hôtel en spécifiant des informations clés, notamment le numéro de la chambre, le nom, le prénom de la personne principale à héberger, ainsi que les dates d'arrivée et de départ. Cette fonctionnalité offre aux utilisateurs une manière pratique et efficace de garantir leur séjour dans l'hôtel de leur choix.

Interface Utilisée : *ReservationChambre*

Méthode : String reserverChambre(

int numChambre,

String nom,

String prénom,

String dateArrive,

String dateDepart)

Paramètres :

- numChambre : Numéro de la chambre à réserver.
- nom : Nom de la personne principale à héberger.
- prénom : Prénom de la personne principale à héberger.
- dateArrive : Date d'arrivée à l'hôtel.
- dateDepart : Date de départ de l'hôtel.

Fonctionnement :

L'utilisateur spécifie les détails de sa réservation, y compris le numéro de la chambre souhaitée, le nom et le prénom de la personne principale, ainsi que les dates d'arrivée et de départ. Le service vérifie la disponibilité de la chambre et effectue la réservation. En cas de succès, le service retourne une confirmation de réservation qui inclut le prix à payer et la référence de la réservation, si celle-ci est confirmée. Cette fonctionnalité simplifie grandement le processus de réservation de chambres d'hôtel, offrant aux utilisateurs un moyen direct de garantir leur séjour.

Service de Recherche de Chambres par Agences Partenaires

Description : Le service de recherche de chambres par agences partenaires offre aux agences partenaires la possibilité de rechercher des chambres d'hôtel au sein d'une liste d'hôtels partenaires en fonction de critères spécifiques, tels que le prix minimum, le prix maximum et le nombre de lits. Cette fonctionnalité permet aux agences de voyage de trouver rapidement des offres correspondant aux besoins de leurs clients, simplifiant ainsi la recherche de chambres.

Interface Utilisée : *RechercheChambreAgence*

Méthode : List<Offre> rechercherChambresAgence (

int prixMin,
int prixMax,
int nbLit)

Paramètres :

- prixMin : Prix minimum rechercher par le client par nuit.
- prixMax : Prix maximum rechercher par le client par nuit.
- nbLit : Nombre minimum de lits requis dans la chambre.

Fonctionnement :

Les agences partenaires utilisent ce service en fournissant les critères de recherche, notamment les prix minimum et maximum et le nombre de lits. En réponse, le service retourne une liste d'offres, chaque offre étant caractérisée par un identifiant unique, un type de chambre, une date de disponibilité, une date d'expiration et un prix. Cette fonctionnalité permet aux agences partenaires de proposer des offres de chambres d'hôtel correspondant aux besoins spécifiques de leurs clients, facilitant ainsi la planification de séjours à l'hôtel.

Service de Réservation de Chambres par Agences Partenaires

Description : Le service de réservation de chambres par agences partenaires permet aux agences partenaires de réserver une chambre d'hôtel en fournissant des informations spécifiques, notamment le nom de l'hôtel, le numéro de la chambre, le nom du client, le prénom du client, les dates d'arrivée et de départ, ainsi que le pourcentage de réduction. Cette fonctionnalité offre aux agences de voyage un moyen efficace de réserver des chambres d'hôtel pour leurs clients tout en bénéficiant de réductions spécifiques

Interface Utilisée : *ReservationChambreAvecAgence*

Méthode : String reserverChambreAvecReduction (

String nomClient,
String prenomClient,
int prixMin,
int prixMax,
int nbLit
double pourcentageReduction)

Paramètres :

- nomClient : Nom du client qui effectue la réservation.
- prenomClient : Prénom du client qui effectue la réservation.
- prixMin : Prix minimum rechercher par le client par nuit.
- prixMax : Prix maximum rechercher par le client par nuit.
- nbLit : Nombre minimum de lits requis dans la chambre.
- pourcentageReduction : Pourcentage de réduction accordé par l'agence de voyage.

Fonctionnement :

Les agences partenaires utilisent ce service en fournissant les informations de la personne principale ainsi que les critères rechercher. Le service vérifie la disponibilité de la chambre et applique la réduction spécifiée. En cas de succès, le service retourne une confirmation de réservation, comprenant la référence de la réservation. Cette fonctionnalité permet aux agences partenaires de réserver des chambres d'hôtel pour leurs clients, en profitant de réductions spécifiques, simplifiant ainsi la gestion des réservations de groupe.

Explication sur l'utilisation des fichiers générés par *wsimport* pour interagir avec les services web à partir du client :

Lorsque nous utilisons *wsimport* pour générer des fichiers à partir d'un fichier WSDL (Web Services Description Language), nous obtenons un ensemble de classes Java qui correspondent aux services web définis dans le WSDL. Ces classes Java sont essentielles pour communiquer avec les services web à partir de notre code Java, car elles agissent comme des proxies pour appeler les méthodes des services web, tout en masquant la complexité de la communication réseau.

Voici comment cela fonctionne en pratique, en utilisant un exemple :

1. **Génération des classes à partir du WSDL :** Nous avons un service web avec une méthode `rechercherChambres` qui prend trois paramètres (`prixMin`, `prixMax`, `nbLit`) et retourne une liste d'offres. Lorsque nous générons des fichiers à partir du WSDL de ce service web, nous obtenons une classe Java, par exemple, `RechercheChambreHotel`. Cette classe correspond au service web, et elle contient une méthode `rechercherChambres` avec les mêmes paramètres que la méthode du service web.

2. **Création d'une instance de la classe générée :** Dans notre classe Java cliente, comme `MyClient`, nous pouvons créer une instance de la classe générée, dans notre exemple `RechercheChambreHotel`. Cela se fait généralement en instanciant la classe générée à partir de son constructeur.

3. **Appel de la méthode du service web :** Après avoir créé une instance de la classe générée, nous pouvons appeler la méthode *rechercherChambres* comme si nous appelions une méthode locale. Nous lui passons les paramètres requis, tels que prixMin, prixMax, et nbLit, et la classe générée se charge de la communication avec le service web distant pour récupérer les résultats.

4. **Traitement des résultats :** Une fois que la méthode du service web est appelée avec succès, la classe générée renvoie les résultats, dans ce cas, une liste d'offres correspondantes aux critères de recherche. Nous pouvons ensuite traiter cette liste de résultats, par exemple, pour afficher les informations sur les chambres à l'utilisateur.

En résumé, les classes Java générées par wsimport simplifient grandement l'interaction avec les services web à partir de notre application cliente. Elles nous permettent d'appeler les méthodes des services web de manière transparente, comme si elles étaient des méthodes locales, en encapsulant toute la complexité liée à la communication avec le service distant. Cela rend l'intégration des services web dans notre application Java plus fluide et plus facile.

V/ Exemples d'utilisation :

Service web 1 :

Étape 1 : Création du service web

Nous avons développé un service web appelé "RechercheChambreHotelImpl", qui permet de rechercher des chambres d'hôtel en fonction de critères tels que le prix minimum, le prix maximum et le nombre de lits. Ce service a été implémenté en utilisant l'interface "RechercheChambreHotel".

Étape 2 : Publication du service web

Ce service web a été publié à une adresse spécifique, par exemple : **http://localhost:8080/service-recherche-chambre**, ce qui signifie qu'il est accessible via cette URL.

Étape 3 : Génération des classes cliente avec wsimport

Pour consommer ce service web, nous avons utilisé `la commande` **: wsimport -keep -p "ConsomationRechercheChambre" "http://localhost:8080/service-recherche-chambre".**

En spécifiant l'URL du service web, nous avons exécuté la commande `wsimport`, ce qui a généré le répertoire *''ConsomationRechercheChambre''* contenant des classes Java correspondant aux services web disponibles.

Étape 4 : Consommation du service web dans MyClient

Dans notre classe *MyClient*, nous avons créé une instance du service de recherche de chambres en utilisant l'URL du service web, par exemple :

```
URL urlRechercheChambre = new URL
        ("http://localhost:8080/service-recherche-chambre-kyriad?wsdl");

RechercheChambreHotelImplService serviceRechercheChambre = new
        RechercheChambreHotelImplService(urlRechercheChambre);

RechercheChambreHotel proxyRechercheChambre =
        serviceRechercheChambre.getRechercheChambreHotelImplPort();
```

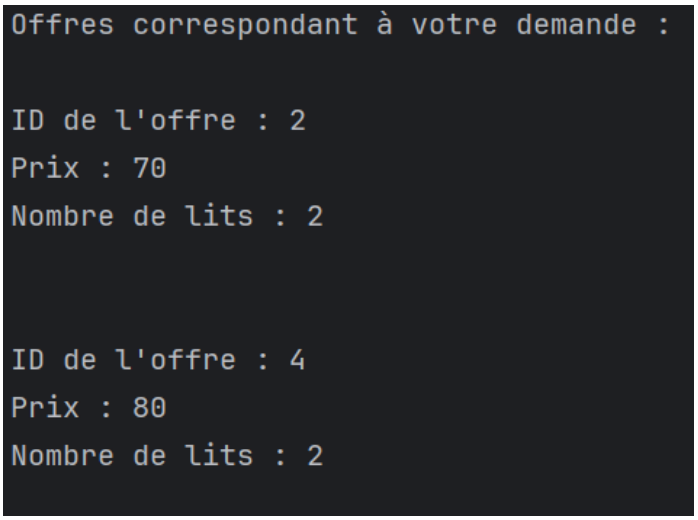
Étape 5 : Appel de la méthode du service web

En utilisant le proxy (`proxyRechercheChambre`), nous avons appelé la méthode *`rechercherChambres`* en passant les critères de recherche, par exemple :

```
List<Offre> offresTrouvees = proxyRechercheChambre.rechercherChambres(70, 100, 2);
```

Étape 6 : Traitement des résultats

Le service web de recherche de chambres s'avère être un outil efficace pour les utilisateurs cherchant des chambres d'hôtel répondant précisément à leurs critères de recherche. Les paramètres clés tels que le prix minimum, le prix maximum et le nombre de lits permettent aux utilisateurs de spécifier leurs préférences. Cette fonctionnalité offre une solution rapide et pratique pour identifier des chambres qui correspondent parfaitement à leurs besoins et respectent leur budget.



En particulier, dans l'hôtel Kyriad, une recherche a été effectuée avec des critères spécifiques : un prix compris entre 70 et 100 euros, et un nombre de lits minimal de 2. Les résultats de cette recherche ont montré que les chambres 2 et 4 de l'hôtel correspondent pleinement à ces critères. Cette démonstration concrète met en lumière l'efficacité du service en fournissant des résultats pertinents et en facilitant le processus de sélection pour les utilisateurs. Cette fonctionnalité simplifie ainsi de manière significative leur processus de recherche et de réservation, améliorant l'expérience utilisateur globale.

```
// Chambre
    Chambre chambreKyriad1 = new Chambre( numChambre: 1, hotelKyriad, étage: 0, nombreLit: 1, prix: 50);
    Chambre chambreKyriad2 = new Chambre( numChambre: 2, hotelKyriad, étage: 0, nombreLit: 2, prix: 70);
    Chambre chambreKyriad3 = new Chambre( numChambre: 3, hotelKyriad, étage: 4, nombreLit: 3, prix: 120);
    Chambre chambreKyriad4 = new Chambre( numChambre: 4, hotelKyriad, étage: 1, nombreLit: 2, prix: 80);
```

Service web 2 :

Étape 1 : Création du service web

Nous avons mis en place un service web appelé "ReservationChambreImpl", qui permet de réserver une chambre d'hôtel en spécifiant le numéro de la chambre, le nom et le prénom de la personne principale à héberger, ainsi que les dates d'arrivée et de départ. Ce service a été implémenté en utilisant l'interface "ReservationChambre".

Étape 2 : Publication du service web

Ce service web a été publié à une adresse spécifique, par exemple : **http://localhost:8080/service-reservation-chambre**, ce qui signifie qu'il est accessible via cette URL.

Étape 3 : Génération des classes cliente avec wsimport

Pour consommer ce service web nous avons utilisé `la commande` **: wsimport -keep -p "ConsomationRechercheChambre" " http://localhost:8080/service-reservation-chambre".**

En spécifiant l'URL du service web, nous avons exécuté la commande `wsimport`, ce qui a généré le répertoire "ConsomationRechercheChambre contenant des classes Java correspondant aux services web disponibles.

Étape 4 : Consommation du service web dans MyClient

Dans notre classe "MyClient", nous avons créé une instance du service de réservation de chambres en utilisant l'URL du service web, par exemple :

```
String serviceURL = "http://localhost:8080/service-reservation-chambre?wsdl";
ReservationChambreImplService serviceReservationChambre = new ReservationChambreImplService(new URL(serviceURL));
ReservationChambre proxyReservationChambre = serviceReservationChambre.getReservationChambreImplPort();
```

Étape 5 : Appel de la méthode du service web

En utilisant le proxy (`proxyReservationChambre`), nous avons appelé la méthode `reserverChambre` en passant les détails de la réservation, par exemple :

```
String resultatRES1 = proxyReservationChambre.reserverChambre(1, "AD", "AM", "01/01/2023", "07/01/2023");
String resultatRES2 = proxyReservationChambre.reserverChambre(1, "NAB", "DAF", "01/01/2023", "07/01/2023");
```

Étape 6 : Traitement des résultats

Le service web de réservation de chambres offre une expérience utilisateur complète en confirmant les réservations avec un résumé détaillé de la transaction. Lorsqu'une réservation est effectuée avec succès, le service fournit un numéro de réservation, le prix, ainsi que les dates d'arrivée et de départ. Cette confirmation claire assure à l'utilisateur une vue d'ensemble précise de sa réservation, renforçant ainsi la confiance dans le processus de réservation.

```
String resultatRES1 = proxyReservationChambre.reserverChambre(1,
"AD", "AM", "01/01/2023", "07/01/2023");

String resultatRES2 = proxyReservationChambre.reserverChambre(1,
"NAB", "DAF", "01/01/2023", "07/01/2023");
```

Cependant, dans le cas où la chambre demandée n'est pas disponible, le service informe immédiatement l'utilisateur de cette indisponibilité. Cette réponse transparente garantit que l'utilisateur est informé en temps réel de la situation et peut prendre des décisions éclairées. Un exemple concret de cette situation est la tentative de réservation d'une chambre qui vient d'être réservée par un autre utilisateur. Dans ce scénario, le service web indiquera explicitement que la chambre n'est plus disponible, offrant ainsi une gestion claire des disponibilités en temps réel.

```
Réservation confirmée
 Résumé de la reservation :
   Numéro de réservation : 1770477526
   Prix : 100
   Date d'arrivé : 01/01/2023
   Date départ : 07/01/2023

La chambre n'est pas disponible pour la réservation.
```

Service web 3 :

Étape 1 : Création du service web

Nous avons mis en place un service web appelé "RechercheChambreAgenceImpl" qui permet aux agences partenaires de rechercher des chambres d'hôtel en fonction de critères tels que le prix minimum, le prix maximum, et le nombre de lits. Ce service a été implémenté en utilisant l'interface "RechercheChambreAgence".

Étape 2 : Publication du service web

Ce service web a été publié à une adresse spécifique, par exemple : **http://localhost:8080/service-recherche-chambre-agence**, ce qui signifie qu'il est accessible via cette URL.

Étape 3 : Génération des classes cliente avec wsimport

Pour consommer ce service web nous avons utilisé `la commande` **: wsimport -keep -p "RechercheChambreAgenceImpl "http://localhost:8080/service-recherche-chambre-agence ".**

En spécifiant l'URL du service web, nous avons exécuté la commande *`wsimport`*, ce qui a généré le répertoire "RechercheChambreAgenceImpl" contenant des classes Java correspondant aux services web disponibles.

Étape 4 : Consommation du service web dans MyAgence

Dans notre classe "MyAgence", nous avons créé une instance du service de recherche de chambres par agences partenaires en utilisant l'URL du service web, par exemple :

```
URL urlRechercheChambreAgence =
                    new URL("http://localhost:8080/service-recherche-chambre-agence?wsdl");

RechercheChambreAgenceImplService serviceRechercheChambreAgence = new
          RechercheChambreAgenceImplService(urlRechercheChambreAgence);

RechercheChambreAgence proxyRechercheChambreAgence =
          serviceRechercheChambreAgence.getRechercheChambreAgenceImplPort();
```

Étape 5 : Appel de la méthode du service web

En utilisant le proxy (`proxyRechercheChambreAgence`), nous avons appelé la méthode `rechercherChambresAgence` en passant les critères de recherche, par exemple :

```
List<Chambre> chambresTrouvees = proxyRechercheChambreAgence.rechercherChambresAgence(50,
120, 2);
```

Étape 6 : Traitement des résultats

Pour le service web de recherche de chambres, notre objectif principal était d'obtenir une liste d'offres correspondant aux critères de recherche, notamment le prix minimum, le prix maximum et le nombre de lits. Heureusement, avec les ajustements et la résolution des problèmes initiaux liés à l'utilisation de wsimport, nous avons réussi à obtenir les résultats escomptés.

Concrètement, nous avons appelé le service web de recherche de chambres avec des critères spécifiques, à savoir un prix compris entre 70 et 100 euros et un nombre minimum de lits égal à 2. Voici comment nous avons effectué cet appel :

```
List<ConsomationRechercheChambreAgence.Offre>
offresTrouvees =
proxyRechercheChambreAgence.rechercherChambresAgence(70,
100, 2);
```

```
System.out.println("Offres correspondant à votre demande :");
for (Offre offre : offresTrouvees) {
    System.out.println("\nID de l'offre : " + offre.getIdOffre());
    System.out.println("Prix : " + offre.getPrix());
    System.out.println("Nombre de lits : " + offre.getNombreLits() + "\n");
}
```

Cette approche nous a efficacement permis d'obtenir une liste d'offres répondant aux critères spécifiés, notamment un prix compris entre 70 et 100 euros et un nombre minimum de lits égal à 2. Nous avons réussi à afficher ces offres avec leurs détails, tels que l'ID, le prix et le nombre de lits. Ainsi, le service web de recherche de chambres fonctionne désormais correctement, offrant une expérience utilisateur améliorée.

```
Démarrage du SW de Recherche de Chambre

Offres correspondant à votre demande :

ID de l'offre : 3
Prix : 80
Nombre de lits : 3


ID de l'offre : 4
Prix : 90
Nombre de lits : 3


ID de l'offre : 2
Prix : 70
Nombre de lits : 2


ID de l'offre : 4
Prix : 80
Nombre de lits : 2
```

Service web 4 :

Étape 1 : Création du service web "ReservationChambreAvecAgenceImpl"

Nous avons développé un service web nommé "ReservationChambreAvecAgenceImpl" qui permet aux agences partenaires de réserver des chambres d'hôtel avec une réduction accordée par l'agence. Ce service a été implémenté en utilisant l'interface "ReservationChambreAvecAgence".

Étape 2 : Publication du service web

Pour que le service web soit accessible, nous l'avons publié à une adresse spécifique. Par exemple, nous l'avons publié à l'adresse : **http://localhost:8080/service-reservation-chambre-agence.**

```
Endpoint.publish("http://localhost:8080/service-reservation-chambre-agence",
        new ReservationChambreAvecAgenceImpl(listeClients, listeHotels, AgenceTrivago));
```

Cela signifie que le service est maintenant disponible à cette adresse, prêt à être consommé.

Étape 3 : Génération des classes cliente avec wsimport

Pour consommer ce service web nous avons utilisé `la commande` : **wsimport -keep -p "ConsomationReservationChambreAgence" "http://localhost:8080/service-reservation-chambre-agence" .**

En spécifiant l'URL du service web, nous avons exécuté la commande `wsimport`, ce qui a généré le répertoire "`Consomation`ReservationChambreAgence" contenant des classes Java correspondant aux services web disponibles.

Étape 4 : Consommation du service web dans MyAgence

Dans notre classe "MyAgence", nous avons créé une instance du service de réservation de chambres avec une réduction en utilisant l'URL du service web, par exemple :

```
        URL urlReservationChambreAgence = new URL("http://localhost:8080/service-reservation-
        chambre-agence");

ReservationChambreAvecAgenceImplService service = new
            ReservationChambreAvecAgenceImplService(urlReservationChambreAgence);

ReservationChambreAvecAgence proxy = service.getReservationChambreAvecAgenceImplPort();
```

Étape 5: Appel de la méthode du service web

Avec le proxy `proxy`, nous avons appelé la méthode `reserverChambreAvecReduction`. Cette méthode permet de réserver une chambre avec les détails spécifiés, tels que le nom de l'hôtel, le numéro de chambre, les informations du client, les dates d'arrivée et de départ, et le pourcentage de réduction.

```
String Reservation1 = proxy.reserverChambreAvecReduction("DUPONT", "PIERRE", 100, 150, 2,  10.0);
String Reservation2 = proxy.reserverChambreAvecReduction("AD", "AM", 50, 80, 1,  5.0);
```

Étape 6 : Traitement des résultats

Le service web "ReservationChambreAvecAgenceImpl" suit un processus fluide pour garantir une expérience de réservation claire et transparente. Tout commence par l'authentification de l'agence, où l'agence doit fournir son nom et son mot de passe pour accéder au service.

```
Saisissez le nom de l'agence : Trivago
Saisissez le mot de passe de l'agence : trivago
```

Après avoir réussi l'authentification, le service web "ReservationChambreAvecAgenceImpl" offre à l'agence un aperçu des offres disponibles pour la réservation. Cette phase joue un rôle essentiel dans le processus de réservation, permettant à l'agence de visualiser les différentes options de chambres qui répondent aux critères spécifiés, tels que le prix et le nombre de lits.

```
Liste des offres disponibles :
Offre 1:
Numéro de l'offre : 2
Prix : 120
Nombre de lits : 2

Offre 2:
Numéro de l'offre : 3
Prix : 120
Nombre de lits : 3
```

Après avoir examiné les différentes offres disponibles, l'agence est sollicitée pour saisir le numéro de l'offre qui correspond à la chambre qu'elle souhaite réserver. Cette étape permet à l'agence de préciser son choix parmi les options présentées, assurant ainsi une personnalisation du processus de réservation en fonction des préférences spécifiques du client.

```
Saisissez le numéro de l'offre que vous souhaitez (1, 2, ...) : 2
```

Après cette étape, le service web génère dans myAgence un résumé complet de la transaction. Ce résumé inclut des détails essentiels tels que le numéro de réservation, le prix de la chambre, ainsi que les dates d'arrivée et de départ.

```
Réservation confirmée !
 Résumé de la réservation :
   Numéro de réservation : 1
   Client : DUPONT PIERRE
   Prix à payer : 108.0
   Offre sélectionnée : 3
```

En cas d'échec de la réservation, le service web réagit de manière proactive pour informer l'agence des problèmes rencontrés. Si l'authentification échoue, une notification indique que le nom ou le mot de passe est incorrect. Si l'offre sélectionnée est déjà réservée, le service indique que la chambre n'est plus disponible. De même, si le numéro d'offre saisi n'est pas valide ou si le client n'est pas reconnu dans la base de données de l'agence, des messages appropriés sont affichés pour clarifier la situation.

Cette approche globale garantit une communication transparente à chaque étape du processus de réservation, assurant ainsi que l'agence est informée de manière précise et en temps opportun, même en cas d'échec de la réservation.

Voici 2 exemples de ces cas d’erreurs :

```
Première réservation :

Numéro d'offre non valide. Veuillez choisir un numéro valide.

Deuxième réservation :

Nom d'agence ou mot de passe incorrect. Veuillez réessayer.
```

```
Deuxième réservation :

Client non trouvé.
```

VI. Conclusion

Dans le cadre de ce TP, nous avons entrepris la conception et la mise en œuvre d'une application de réservation d'hôtels en ligne. Cette application repose sur l'utilisation de services web SOAP, une approche de communication standardisée, pour permettre aux utilisateurs de rechercher des chambres d'hôtel et de procéder à leur réservation en ligne.

Nous avons découpé le TP plusieurs phases essentielles, notamment la conception de l'architecture de l'application, la modélisation des entités clés, la définition des services web nécessaires, ainsi que l'implémentation de ces services. Nous avons également généré des fichiers à partir du WSDL pour faciliter l'interaction avec les services web depuis le côté client.

L'application offre des fonctionnalités clés, telles que la recherche de chambres en fonction de divers critères, la réservation de chambres avec ou sans réductions proposées par des agences partenaires. Ces fonctionnalités permettent aux utilisateurs, y compris les agences de voyage, de trouver rapidement et efficacement des hébergements qui correspondent à leurs besoins spécifiques.

En conclusion, ce projet a mis en lumière l'importance des services web dans le domaine de la réservation d'hôtels en ligne. L'utilisation de normes telles que SOAP a permis de standardiser la communication entre les clients et les serveurs, facilitant ainsi l'accès aux services de réservation. L'application que nous avons développée illustre comment les services web peuvent simplifier le processus de réservation pour les utilisateurs tout en offrant aux agences de voyage un moyen efficace de gérer les réservations pour leurs clients.