

Hotel SOAP

MAZERAND Elliot, GARCIA PENA Loris

18 novembre 2023

Résumé

Ce rapport présente notre travail réalisé dans le cadre du TP portant sur le service web SOAP.

Table des matières

1	Introduction	3
2	Architecture Globale	4
2.1	Client-Agence-Hôtel	4
2.1.1	Communication Client-Agence	4
2.1.2	Communication Agence-Hôtel	4
2.1.3	Recherche et Réservation	5
3	La classe Hôtel	6
3.1	Architecture	6
3.2	Méthodes	7
3.2.1	Recherche de Chambres	7
3.2.2	Réservation d'Offre	9
3.3	Exceptions	9
4	Diagramme de Conception	11
5	Distribution des Hotels	12
5.1	Architecture	12
5.2	Services web	12
5.3	Web Services Description Language (WSDL) dans le Projet SOAP	13
5.4	Fonctionnement de WSDL	13
5.5	Publication des Hôtels	13
5.5.1	Exposition du Répertoire d'Hôtels	14
5.5.2	Services de Recherche de Chambres	15
5.5.3	Services de Réservations	15

6	Communication avec les Agence	17
6.1	Consommation	17
6.1.1	Génération des Classes Java avec <code>wsimport</code>	17
6.1.2	Intégration des Classes Générées dans le Projet	17
6.1.3	Création du Proxy pour Utiliser les Opérations du Service	17
6.2	Publication des Agences	18
6.2.1	Création d'Agences	18
6.2.2	Services web Recherche de Chambres	19
6.2.3	Service web Identification	20
6.2.4	Gestion des utilisateurs	20
7	Intégration des Images	21
8	CLI	22
9	Interface Graphique	22
10	Conclusion	28
11	Read-Me	29

1 Introduction

Nous avons développé une application permettant à des clients de rechercher et de réserver des chambres d'hôtels à partir de plusieurs agences via des services web. L'objectif principal de notre application est de fournir aux utilisateurs une interface conviviale, leur permettant de saisir divers critères de recherche tels que la ville de séjour, les dates d'arrivée et de départ, un intervalle de prix souhaité, une catégorie d'hôtel en termes d'étoiles, et le nombre de personnes à héberger. En retour, l'application leur présente une liste d'hôtels correspondant à leurs critères, avec des informations détaillées sur chaque option, notamment le nom de l'hôtel, l'adresse complète, le prix, le nombre d'étoiles et le nombre de lits proposés.

Au-delà de la sélection d'un hébergement, notre application offre également la possibilité à l'utilisateur de procéder à la réservation en fournissant des informations personnelles telles que le nom et prénom de la personne principale à héberger, ainsi que les détails de la carte de crédit pour le paiement. Nous allons développer notre application à partir des services web SOAP.

La technologie Simple Object Access Protocol (SOAP) se profile comme une infrastructure robuste, permettant à des applications distribuées de communiquer de manière transparente à travers des services web. En substance, SOAP offre une norme d'échange de messages, facilitant ainsi l'interaction entre des systèmes distribués. SOAP joue un rôle central en permettant à des objets de s'invoquer mutuellement à travers des machines virtuelles distinctes.

En d'autres termes, le protocole SOAP offre la possibilité d'établir une communication à distance entre des composants logiciels, en permettant l'invocation de méthodes sur des objets distribués appelés stubs. Cette capacité de collaboration à distance élimine les frontières virtuelles entre les objets locaux et distants, offrant une expérience d'interaction fluide.

2 Architecture Globale

L'architecture globale de notre application repose sur un modèle distribué qui facilite la communication entre différents acteurs, à savoir les clients, les agences de voyage, et les hôtels. Cette structure modulaire permet une interaction efficace tout en garantissant une extensibilité du système.

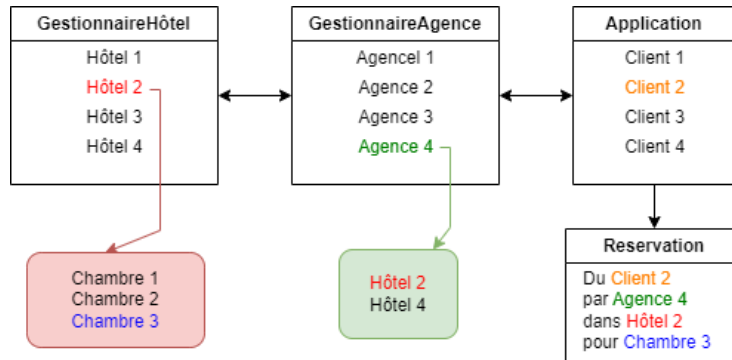


FIGURE 1 – Architecture application

Notre application s'articule donc au travers de trois grandes parties communicant, entre-elles, via SOAP.

2.1 Client-Agence-Hôtel

L'interaction entre les clients, les agences de voyage, et les hôtels s'effectue de la manière suivante :

2.1.1 Communication Client-Agence

Les clients interagissent avec les agences de voyage via une interface graphique permettant aux clients de rechercher des chambres d'hôtel en fonction de divers critères, de visualiser des offres, et de procéder à des réservations. La communication entre le client et l'agence se fait via des services web sécurisés.

2.1.2 Communication Agence-Hôtel

Les agences de voyage jouent le rôle d'intermédiaires entre les clients et les hôtels. Lorsqu'un client effectue une recherche ou une réservation, l'agence communique avec les hôtels pour obtenir des informations actualisées sur les disponibilités et les tarifs. Cette communication s'effectue également via des services web, garantissant une transmission fiable et sécurisée des données.

2.1.3 Recherche et Réservation

La recherche de chambres et la réservation se déroulent de manière transparente pour le client. L'agence utilise les services web des hôtels pour récupérer les informations en temps réel, assurant ainsi que les données affichées sont toujours à jour. Le client, lui, communique avec les services web de l'agence pour récupérer ces informations. Une fois qu'un client sélectionne une offre, l'agence communique avec l'hôtel correspondant pour finaliser la réservation.

En résumé, l'architecture distribuée de notre application favorise une communication fluide entre les clients, les agences de voyage, et les hôtels, tout en offrant la flexibilité nécessaire pour s'adapter à l'évolution des besoins du système.

3 La classe Hôtel

La première étape de ce projet réside dans la création d'hôtel. Nous avons initié le processus en implémentant une première version non distribuée de cette classe, établissant ainsi les fondations du système. Chaque hôtel est défini par un identifiant unique, un nombre d'étoiles, une adresse, une liste de chambres, et une liste de réservations.

L'objectif de cette entreprise est de permettre à un utilisateur de rechercher des chambres en fonction de critères spécifiques et d'obtenir une liste des chambres disponibles dans un hôtel répondant à ses exigences. L'utilisateur devrait également être en mesure de sélectionner une offre et de la réserver. Dans cette optique, nous avons mis en place une méthode dédiée à la recherche d'offres dans un hôtel, prenant en compte des critères tels que le nombre de lits, le prix minimum et maximum, ainsi que les dates d'arrivée et de départ. Cette méthode retourne une liste des offres correspondantes. De plus, nous avons développé une méthode pour effectuer une réservation en sélectionnant une offre disponible.

3.1 Architecture

Pour mettre en place la classe **Hotel**, nous avons créé plusieurs classes essentielles qui interagissent de manière cohérente pour former une structure robuste. Ces classes incluent **Adresse**, **Carte**, **Chambre**, **Client**, **Offre**, et **Reservation**. Chacune d'entre elles joue un rôle spécifique dans le fonctionnement global du système, contribuant ainsi à la création et à la gestion efficaces de l'entité **Hotel**.

- **Adresse** : Cette classe représente l'adresse physique d'un hôtel, permettant une identification précise de sa localisation.
- **Carte** : La classe **Carte** est responsable de la gestion des informations de carte, associées à des paiements pour des réservations.
- **Chambre** : Chaque instance de cette classe encapsule les détails d'une chambre spécifique dans un hôtel, comprenant des informations telles que le nombre de lits, le prix, et un numéro.
- **Client** : Représentant les utilisateurs du système, la classe **Client** gère les détails individuels tels que les informations personnelles et une carte de paiement.
- **Offre** : La classe **Offre** est conçue pour encapsuler les détails d'une offre spécifique dans un hôtel, fournissant des informations détaillées pour les utilisateurs lors de la recherche.
- **Réservation** : Responsable de la gestion des réservations, la classe **Reservation** enregistre les détails pertinents tels que les dates, le client concerné, et l'offre de chambres réservées.

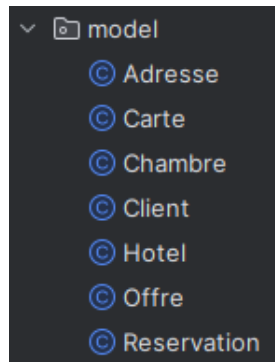


FIGURE 2 – classes

Un `Hôtel` contient donc une liste de Chambres, une liste de Réservations et une liste d’offre qui sera renvoyé par la fonction recherche de chambres.

```

1 public class Hotel {
2     private String identifiant;
3     private String nom;
4     private Adresse adresse;
5     private int nombreEtoiles;
6     private String imageHotel;
7     private ArrayList<Chambre> chambres;
8     private ArrayList<Reservation> reservations;
9     private ArrayList<Offre> offres;
10    ...
11 }

```

3.2 Méthodes

Notre classe `Hôtel` intègre des méthodes très importante pour la suite de l’application, notamment la recherche de chambres selon des critères définis par le client et la réservation d’une offre parmi celles renvoyées par la méthode de recherche. Ces fonctionnalités offrent à l’utilisateur la flexibilité nécessaire pour trouver des chambres répondant à ses exigences spécifiques, puis procéder à la réservation de l’offre choisie.

3.2.1 Recherche de Chambres

On commence par vérifier que l’hôtel correspond à la bonne ville et au bon nombre d’étoiles demandé par le client.

```

1     if (this.adresse.getVille().equals(ville) && this.
        nombreEtoiles == nombreEtoiles)

```

Si c’est le cas, on créer une `ArrayList` d’offre et une `ArrayList` de chambres disponible et on y ajoute toutes les chambres qui correspondent aux critères.

```

1 for (Chambre chambre : this.chambres) {
2     if (chambre.getPrix() >= prixMin && chambre.getPrix() <=
        prixMax
3         && chambre.getNombreLits() <= nombrePersonne) {
4         chambresDisponibles.add(chambre);
5     }
6 }

```

On supprime de la liste les chambres qui sont déjà réservé à la date saisie par le client :

```

1 List<Chambre> chambresARetirer = new ArrayList<>();
2
3 for (Reservation reservation : this.reservations) {
4     if (reservation.getDateArrivee().before(dateDepart) &&
        reservation.getDateDepart().after(dateArrivee)) {
5         chambresARetirer.addAll(reservation.getChambresReservees()
6     );
7 }
8
9 chambresDisponibles.removeIf(chambre -> chambresARetirer.stream().
    anyMatch(ch -> ch.getNumero() == chambre.getNumero()));

```

On regarde ensuite si il y a une chambre avec le nombre de lits qui correspond au nombre de personne à hébergé. Si c'est le cas, on ajoute cette chambre à notre liste d'offre :

```

1
2 for (Chambre chambre : chambresDisponibles) {
3     if (chambre.getNombreLits() == nombrePersonne) {
4         String indentifiant = "O" + new Date().getTime();
5         Offre offre = new Offre(...);
6         offres.add(offre);
7         this.addOffre(offre);
8         return offres;
9     }
10 }

```

Sinon, on regarde s'il existe des combinaisons de chambres qui peuvent permettre d'accueillir le bon nombre de personnes :

```

1     ArrayList<ArrayList<Chambre>> listeCombinaisonsChambres = new
        ArrayList<>();
2     Set<List<Integer>> combinaisonsDeLits = new HashSet<>();
3
4     chercherCombinaison(chambresDisponibles, nombrePersonne, new
        ArrayList<Chambre>(),
5     combinaisonsDeLits, listeCombinaisonsChambres);
6
7     Random random = new Random();
8     for (ArrayList<Chambre> combinaisonChambresDisponibles :
        listeCombinaisonsChambres) {
9         String indentifiant = "O" + random.nextInt(1000) + new Date
            ().getTime();
10        Offre offre = new Offre(indentifiant, nombrePersonne,
            combinaisonChambresDisponibles.stream().mapToDouble(Chambre::
                getPrix).sum(), dateArrivee, dateDepart,
            combinaisonChambresDisponibles, this.indentifiant);

```



```

11         offres.add(offre);
12         this.addOffre(offre);
13     }
14
15     return offres;
16 }
17 }
18 return null;

```

3.2.2 Réservation d'Offre

Cette fonction, nommée `reserverChambres`, a pour objectif de réaliser la réservation d'une offre de chambres spécifique dans un hôtel.

```

1 public Reservation reserverChambres(Offre offre ,
2 boolean petitDejeuner, String nomClient, String prenomClient,
   String email, String telephone, String nomCarte, String
   numeroCarte, String expirationCarte, String CCVCarte) throws
   DateNonValideException {
3     if (offre.getDateArrivee().after(offre.getDateDepart())) {
4         throw new DateNonValideException();
5     }
6     Carte carte = new Carte(...);
7     Client clientPrincipal = new Client(...);
8     String numero = "R" + new Date().getTime();
9     Reservation reservation = new Reservation(...);
10    this.addReservation(reservation);
11    clientPrincipal.addReservationToHistorique(reservation);
12    this.removeOffre(offre);
13    return reservation;
14 }

```

Ce code Java définit une méthode `reserverChambres` qui permet de réaliser la réservation d'une chambre en fonction d'une offre spécifique. La méthode effectue des vérifications, notamment la validité des dates, crée une carte de paiement, instancie un client principal, génère un numéro de réservation unique, crée une réservation, l'ajoute à l'hôtel, met à jour l'historique du client, retire l'offre réservée, et retourne l'objet de réservation créé. Si les dates d'arrivée et de départ sont invalides, une exception de type `DateNonValideException` est levée. En résumé, cette fonction effectue les étapes nécessaires pour créer et enregistrer une réservation dans un hôtel, en prenant en compte la validité des dates, les détails du client, et les informations associées à la carte de paiement. Elle encapsule plusieurs actions essentielles pour assurer le bon déroulement du processus de réservation.

3.3 Exceptions

Nous prenons en charge la gestion d'exceptions, telles que `DateNonValideException`, `HotelAlreadyExistsException`, et `HotelNotFoundException`. Ces exceptions sont conçues pour traiter respectivement les erreurs liées à des dates invalides dans le contexte de réservation, la création d'un hôtel qui existe déjà, et la recherche infructueuse d'un hôtel. Elles permettent une gestion précise des cas

exceptionnels et assurent une exécution contrôlée du programme dans ces situations spécifiques.

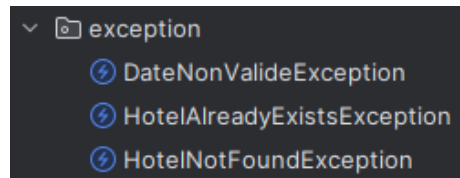


FIGURE 3 – dossier exception

4 Diagramme de Conception

Dans le cadre de la conception de mon application, nous avons élaboré un diagramme détaillé afin de visualiser l'architecture globale et les interactions entre les différents composants. Ce diagramme de conception a joué un rôle crucial dans la planification et l'organisation du développement de l'application.

Le diagramme met en lumière les différentes parties de l'application, notamment les composants modèles pour les entités clés telles que les hôtels, les agences, les utilisateurs et les offres.

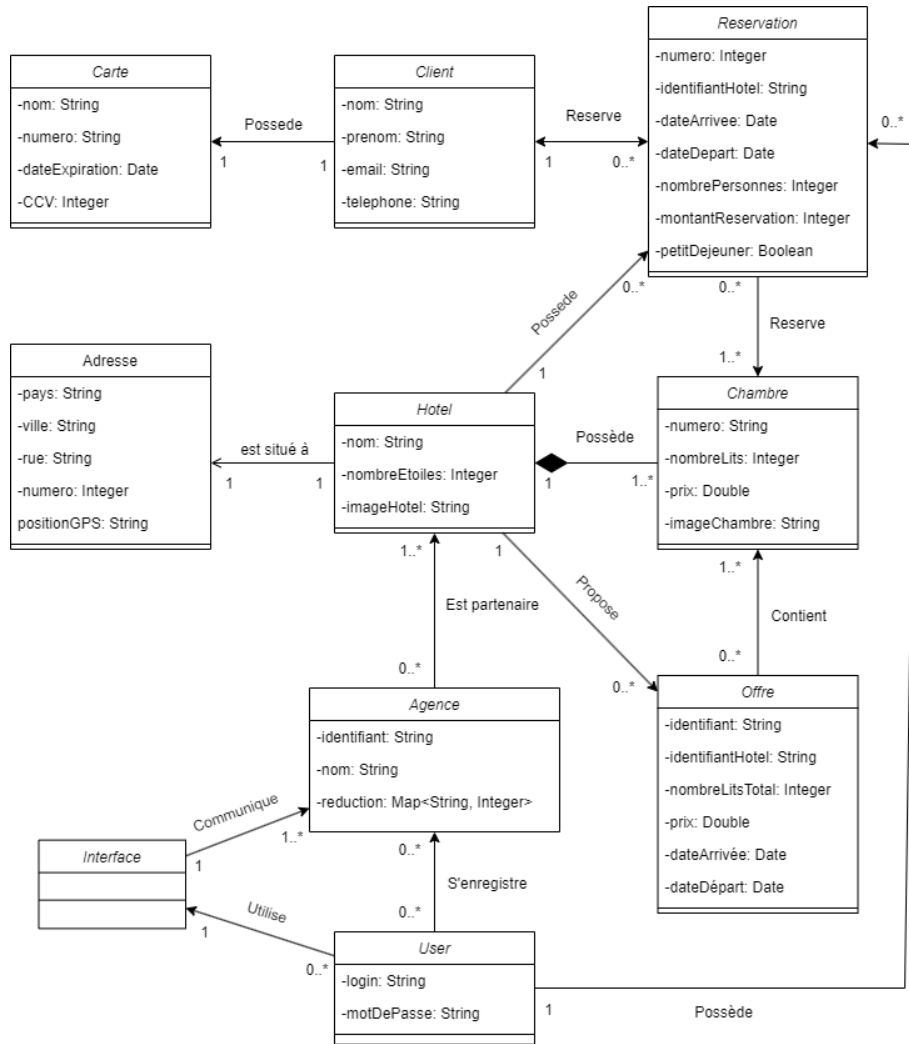


FIGURE 4 – Diagramme de conception

5 Distribution des Hotels

Dans un second temps, nous avons oeuvré à rendre l'application distributive, le but était donc de distribuer les hôtels et ses méthodes en utilisant le service SOAP.

5.1 Architecture

Pour cette étape de développement, la structure du projet hôtel évolue avec l'introduction de nouveaux composants, notamment un répertoire appelé **repository**, dédié à la création d'un répertoire d'hôtels. Les hôtels sont générés de manière aléatoire à l'aide de la classe **RandomDonneStockage**, ce qui inclut la création aléatoire de tous les éléments constituant un hôtel tels que le nombre d'étoiles, l'adresse, le pays, la ville, ainsi que les chambres qu'il contient. Cette approche permet de générer un nombre considérable d'hôtels différents, chacun étant attribué à un identifiant unique.

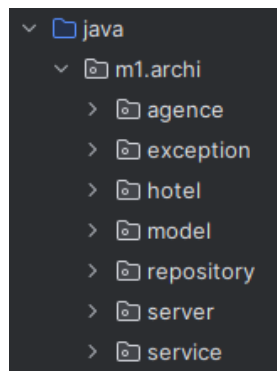


FIGURE 5 – Organisation du projet

Un nouveau composant, appelé **service**, est introduit pour créer les services web qui seront publiés par notre serveur d'hôtels. Enfin, la classe **HotelServicePublisher** dans le dossier **server** est chargée de publier l'ensemble des services web créés dans le répertoire **service**.

5.2 Services web

Cette fonctionnalité s'opère via le principe des services web : un service web SOAP est une technologie de communication facilitant l'échange de données structurées entre des systèmes informatiques distants. Chaque hôtel publie ses services web sur un serveur dédié, offrant aux clients la possibilité d'accéder à ces services via une communication distante.

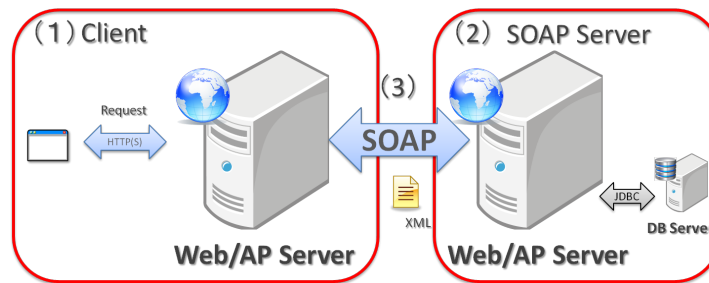


FIGURE 6 – Communication SOAP

5.3 Web Services Description Language (WSDL) dans le Projet SOAP

Dans le cadre de notre projet basé sur le protocole SOAP, le langage WSDL (Web Services Description Language) joue un rôle crucial en définissant l'interface du service web. En travaillant sur ce projet, il est essentiel de comprendre le fonctionnement et l'utilité de WSDL.

5.4 Fonctionnement de WSDL

WSDL agit comme un contrat formel entre les services web SOAP et les clients qui souhaitent les consommer. Il se présente sous la forme d'un document XML décrivant les opérations disponibles, les types de données associés, les protocoles de communication utilisés, les valeurs de retours et d'autres informations pertinentes.

Le fonctionnement de WSDL peut être compris en examinant ses principales composantes. La section `<types>` définit les types de données utilisés dans les messages, tandis que la section `<portType>` spécifie les opérations offertes par le service. L'enveloppe de la requête et de la réponse est détaillée dans la section `<message>`, et enfin, la section `<binding>` précise comment les opérations sont liées aux protocoles de communication, souvent HTTP dans le contexte SOAP.

En conclusion, WSDL agit comme un guide formel, facilitant la communication entre les parties prenantes du projet et contribuant à l'interopérabilité du service web.

5.5 Publication des Hôtels

Les serveurs d'hôtels constituent le premier composant essentiel de notre architecture distribuée. Dans cette conception, un serveur centralisé gère l'ensemble des hôtels, mais chaque hôtel dispose de son propre URL distinct qui contient son identifiant, et de ses services web dédiés permettant la recherche et la réservation. Cette approche permet une gestion centralisée tout en préservant l'individualité de chaque établissement. Chaque URL d'hôtel est unique, garantissant ainsi une identification claire lors des échanges avec les agences

partenaires via le protocole SOAP. La publication des services web pour notre projet comprend la mise à disposition du répertoire complet de nos hôtels. La publication se fait grâce au code suivant :

```
1
2     public static void main(String[] args) throws IOException {
3
4         String adresse = "http://localhost:8080/hotelservice/
5         identifiantHotels";
6         Endpoint.publish(adresse, new
7         HotelServiceIdentificationImpl());
8         System.err.println("Server ready");
9         System.out.println("Adresse du service des identifiants : "
10        + adresse);
11    }
```

5.5.1 Exposition du Répertoire d'Hôtels

Avant d'entrer dans les détails des méthodes spécifiques, il est impératif de publier le répertoire d'hôtels. Cela permet aux clients d'obtenir une vue d'ensemble de tous les établissements disponibles, chacun accompagné de ses caractéristiques, de ses chambres et de ses disponibilités. Chaque hôtel publie un service web d'identification comprenant plusieurs méthodes par exemple `getHotel()` permettant de récupérer l'hôtel associé à un ID ou encore `getIdentifiantHotels()` permettant d'obtenir la liste des ID de tous les hôtels.

```

1  public Hotel getHotel(String identifiant) throws
    HotelNotFoundException {
2      return this.hotelRepository.getHotel(identifiant);
3  }

```

(GetHotel)

```

1  public ArrayList<String> getIdentifiantHotels() {
2      return this.hotelRepository.getIdentifiantHotels();
3  }

```

(getIdentifiantHotels)

5.5.2 Services de Recherche de Chambres

Le code ci-dessous représente l'implémentation d'un service web de consultation d'hôtel. Ce service, défini par l'interface `HotelServiceConsultation`, est concrétisé par la classe `HotelServiceConsultationImpl`. Cette dernière initialise un objet `Hotel` lors de sa création.

```

1  @WebService(endpointInterface = "ml.archi.service.
    HotelServiceConsultation")
2  public class HotelServiceConsultationImpl implements
    HotelServiceConsultation {

```

La ligne annoté par `WebService` configure la classe pour qu'elle soit accessible en tant que service web SOAP via l'interface spécifiée, permettant aux clients distants d'invoker les méthodes définies dans cette interface.

Cette classe expose la méthode, `getChambreDisponibleCriteres` de la classe `Hôtel`, permettant de récupérer une liste d'offres de chambres disponibles selon divers critères.

```

1  public ArrayList<Offre> getChambreDisponibleCriteres(String ville
    ...) throws DateNonValideException {
2
3      return hotel.getChambreDisponibleCriteres(ville...);
4  }

```

5.5.3 Services de Réservations

La publication du services de réservation est également importante. La méthode associée à cette fonctionnalité doit être exposées de manière à permettre aux clients d'effectuer des réservations en fournissant les informations nécessaires telles que l'offre souhaité, les dates de séjour, et les détails du client, sa carte de paiement... De manière analogue au service web de rechercher d'offres, on créer une classe `HotelServiceReservationImpl` qui implémente l'interface `HotelServiceReservation` et définit la méthode `reserverChambres` qui utilise la méthode du même nom de la classe `Hôtel` visant à réaliser une réservation.

```

1  public Reservation reserverChambres(Offre offre, boolean
    petitDejeuner, String nomClient, String prenomClient, String
    email, String telephone, String nomCarte, String numeroCarte,
    String expirationCarte, String CCVCarte)
2  throws DateNonValideException {
3      return hotel.reserverChambres(...);
4  }

```

En conclusion, la publication des services web pour notre projet SOAP ne se limite pas à l'exposition de fonctionnalités de base. Elle englobe la mise à disposition du répertoire d'hôtels et des méthodes spécifiques de recherche de chambres et de réservations. Cette approche méticuleuse garantit une interaction efficace entre les clients et les agences.

6 Communication avec les Agence

Le processus de réservation d'hôtels s'opère à travers une agence de voyage, laquelle accède aux données des hôtels via leurs services web respectifs. Cette démarche vise à offrir une expérience personnalisée, où les prix proposés par les hôtels peuvent être adaptés en fonction des accords particuliers avec chaque agence. Il convient de noter que les agences peuvent établir des tarifs différents pour les mêmes produits ou chambres. Dans cette conception, avec une adaptation similaire à celle des serveurs d'hôtels, un serveur centralisé gère l'ensemble des agences partenaires.

Chaque agence dispose ainsi d'une URL propre, comportant son identifiant distinctif, offrant une individualité à ses services web. L'utilisation du protocole SOAP maintient la cohérence et la sécurité des échanges entre les hôtels et les agences. Cette section détaille les étapes de consommation des services web des hôtels par les agences, ainsi que la publication de nos services web pour permettre aux clients externes d'accéder à nos fonctionnalités en consommant les services web des agences.

6.1 Consommation

La consommation de services web implique l'utilisation, par une application, des fonctionnalités mises à disposition par des services distants. Cela inclut la génération de classes, leur intégration dans le projet, et la création de proxies pour faciliter l'appel et la gestion des opérations distantes.

6.1.1 Génération des Classes Java avec `wsimport`

Le processus de consommation des services web des hôtels débute par la génération des classes Java à partir de leurs descriptions WSDL (*Web Services Description Language*). L'utilisation de la commande `wsimport` simplifie cette tâche en automatisant la création des classes Java basées sur le WSDL fourni par les hôtels..

6.1.2 Intégration des Classes Générées dans le Projet

Une fois les classes Java générées, leur intégration dans notre projet est nécessaire. Cette étape implique l'incorporation de ces classes dans le package du projet des agences de notre application, Cette étape est essentielle, car lors de l'appel de nos services web par une agence, il est impératif de connaître la structure des hôtels ainsi que leurs méthodes. Ceci est nécessaire pour pouvoir utiliser efficacement le service web qui exploite ces fonctionnalités.

6.1.3 Création du Proxy pour Utiliser les Opérations du Service

La création de proxy est nécessaire pour utiliser les opérations fournies par les services web des hôtels. Ces proxy agissent comme des intermédiaires entre notre application et les services des hôtels, simplifiant ainsi l'appel et la gestion

des opérations distantes. Voici par exemple la création d'un proxy permettant d'utiliser le service de recherche d'offre de chambres :

```
1 URL url = new URL("http://localhost:8080/hotelservice/" +
    identifiantHotel + "/consultation");
2 HotelServiceConsultationImplService hotelServiceConsultation = new
    HotelServiceConsultationImplService(url);
3 HotelServiceConsultation proxy = hotelServiceConsultation.
    getHotelServiceConsultationImplPort();
```

Et son utilisation :

```
1 List<Offre> listeOffres = proxy.getChambreDisponibleCriteres
    (...);
```

6.2 Publication des Agences

Dans le cadre de notre système, les agences tirent parti des fonctionnalités de recherche et de réservation offertes par les hôtels. La publication de nos services web permet aux agences d'accéder à notre répertoire d'hôtels et d'offrir ces services à leurs clients.

De plus, nous assurons également la publication des services web des agences, englobant les fonctionnalités telles que la recherche et la réservation que nous avons récupérée dans les services web des hôtels, ainsi qu'un service web avec les mécanismes d'identification des agences, dans le même principe que celui des hôtels. Cette démarche offre une visibilité complète aux agences. Les services de recherche, de réservation et d'identification, sont mis à la disposition des clients finaux par le biais des agences. Une addition notable est l'introduction d'un nouveau service web permettant l'inscription et la connexion d'un utilisateur à une agence par le biais de la méthode "inscription" et "connexion" de la classe User.

En résumé, la mise en place d'une communication réussie avec les agences nécessite une consommation judicieuse des services web des hôtels, ainsi qu'une publication de nos propres services pour permettre une collaboration efficace avec les clients.

6.2.1 Création d'Agences

La section dédiée aux agences suit la même organisation que celle des hôtels : un composant **model** pour les entités (agences, utilisateurs, offres), un composant **repository** pour la création d'un répertoire d'agences, un composant **service** pour la définition des services web, un composant **exception** comprenant les classes pour la gestion des exceptions, ainsi qu'un package "hôtel" dans lequel sont regroupées les classes générées pour la consommation des services web des hôtels.

Il convient de noter que, dans un souci de variété et de réalisme, les agences sont créées aléatoirement au sein du système dans la même idée que la création des hôtels. De plus, elles sont associées de manière aléatoire à des listes d'hôtels partenaire. Cette approche vise à simuler un environnement dynamique

où différentes agences peuvent avoir des partenariats variés avec divers hôtels, contribuant ainsi à la diversité des offres présentées aux clients finaux.

6.2.2 Services web Recherche de Chambres

Les services web publiés par les agences utilisent les services web des hôtels pour faire leurs propres services web. Par exemple, pour le service web qui permettra à un client de rechercher une liste d'offre de chambres, utilise un proxy pour récupérer la liste des offres pour chacun de ses hôtels partenaire qui correspondent à la recherche du client. Le service web publié par l'agence renvoi ainsi une liste d'hôtel ayant chacun une liste d'offre.

Pour ce faire, on a créé une classe `Offres` qui a pour seul attribut une liste d'`Offre`. Cela permet d'avoir dans notre fonction une liste d'instance d'offres contenant chacune une liste d'offre.

Nous avons donc la fonction de recherche d'Offre dans une agence suivante :

```

1 public ArrayList<Offres> listeChoisOffresHotelCriteres(String login
    , String motDePasse, String ville,...) throws
    DateNonValideException, UserNotFoundException {
2     // essayer de se connecter
3     connectionUser(login, motDePasse);
4
5     ArrayList<Offres> listeChoisOffresHotel =
6     new ArrayList<Offres>();
7
8     // boucle qui recupere toutes les offres de chaque hotel
9     for (String identifiantHotel :
        mapIdentifiantsHotelsPartenairesReduction.keySet()) {
10         try {
11             // creation du proxy
12             URL url = new URL("http://localhost:8080/hotelservice/"
                + identifiantHotel + "/consultation");
13             HotelServiceConsultationImplService
                hotelServiceConsultation =
14                 new HotelServiceConsultationImplService(url);
15             HotelServiceConsultation proxy =
                hotelServiceConsultation.getHotelServiceConsultationImplPort();
16
17             ...
18
19             // cr ation des offres
20             Offres offres = new Offres();
21             ArrayList<Offre> listeOffres =
22                 (ArrayList<Offre>) proxy.getChambreDisponibleCriteres
                (...);
23
24             offres.setOffres(listeOffres);
25
26             listeChoisOffresHotel.add(offres);
27
28             ...
29         }
30         return listeChoisOffresHotel;
31     }
32 }
```

6.2.3 Service web Identification

Le service web d'identification, `AgenceServiceIdentificationImpl`, offre un accès à plusieurs méthodes cruciales pour le client, telles que `getListeAgence()` permettant d'obtenir la liste des agences et `getAgence(String identifiant)` qui prend en paramètre l'identifiant d'une agence et renvoie l'agence correspondante ou encore `getListeHotelsPartenaire()` qui retourne la liste des hôtels partenaire d'une agence.

6.2.4 Gestion des utilisateurs

Dans le cadre du système de gestion d'une agence, une classe `User` a été implémentée pour représenter les utilisateurs de l'agence. Chaque utilisateur possède un `login` et un `motDePasse`, nécessaires pour effectuer des opérations telles que la recherche ou la réservation de services. La classe `User` est définie comme suit :

```
1 public class User {
2     private String login;
3     private String motDePasse;
4     private List<Reservation> reservations;
5
6     public User(String login, String motDePasse) {...}
7 }
8
9 public class Agence {
10     ...
11     public User inscriptionUser(String login, String motDePasse)
12         throws UserAlreadyExistsException {...}
13     public User connectionUser(String login, String motDePasse)
14         throws UserNotFoundException {...}
15     public boolean addReservation(String login, String motDePasse,
16         Reservation reservation) throws UserNotFoundException {...}
17     ...
18 }
```

Chaque instance de la classe `User` maintient une liste de réservations associées à cet utilisateur. Cela permet de suivre les activités de réservation spécifiques de chaque utilisateur au sein de l'agence.

En synthèse, la mise en place d'un serveur d'agence a été réalisée pour gérer efficacement les différentes agences de notre application. Ces agences agissent comme des intermédiaires entre nos hôtels et nos clients, facilitant la consommation des services web et mettant en évidence l'architecture client/serveur sous-jacente.

7 Intégration des Images

Pour mettre en œuvre la possibilité de consulter les disponibilités par les agences partenaires et de retourner une image de la chambre au client, nous avons effectué des modifications notables au sein de notre application.

La classe "Chambre" a été étendue pour inclure la gestion de l'image associée à chaque chambre en ajoutant un élément image de type String à son constructeur.

```
1 public Chambre(..., String imageChambre) {
2     ...
3     this.imageChambre = imageChambre;
4 }
5
6 ...
7
8 public String getImageChambre() {
9     return this.imageChambre;
10 }
11
12 public void setImageChambre(String imageChambre) {
13     this.imageChambre = imageChambre;
14 }
```

Et la classe `HotelRepositoryImpl` a été ajustée pour générer aléatoirement ces images lors de la création des chambres. Pour faciliter l'envoi des images via SOAP, nous encodons l'image en utilisant la bibliothèque `Base64` :

```
1
2 int nombreChambres = RandomDonneStockage.randomNombreChambres();
3 for (int j = 1; j <= nombreChambres; j++) {
4     ...
5     File imageChambre = RandomDonneStockage.randomImageChambre(
6         nombreEtoiles);
7     byte[] imageBytes = readFileToByteArray(imageChambre);
8     String base64ImageChambre = Base64.encodeBase64String(
9         imageBytes);
10    Chambre chambre = new Chambre(..., base64ImageChambre);
11    ...
12 }
```

De plus, l'interface graphique du client a été adaptée pour récupérer et afficher ces images en tant qu'éléments visuels pour les utilisateurs finaux.

```
1 // Creez un JLabel pour afficher l'image
2 JLabel chambreImageLabel = new JLabel();
3 String base64Image = selectedChambre.getImageChambre();
4 byte[] imageBytes = Base64.getDecoder().decode(base64Image);
5 ImageIcon chambreImageIcon = new ImageIcon(imageBytes);
6 // Redimensionnez l'image pour harmoniser la taille
7 Image scaledImage = chambreImageIcon.getImage().getScaledInstance
8     (300, 200, Image.SCALE_SMOOTH);
9 chambreImageLabel.setIcon(chambreImageIcon);
```

Ces modifications constituent une amélioration significative de notre système, offrant aux agences partenaires la capacité de présenter des images de chambres

aux clients lors de la consultation des disponibilités. Cette évolution renforce l'expérience utilisateur en fournissant des informations visuelles, contribuant ainsi à une interaction plus riche et immersive dans le processus de réservation d'hôtels.

8 CLI

Une fois que toutes les fonctionnalités ont été mises en place, nous avons développé une interface en ligne de commande (CLI) pour tester l'ensemble des caractéristiques de l'application. Dans le script principal, nous commençons par récupérer la liste des agences en créant un proxy pour le service web d'identification. Cette interface permet de tester différentes fonctionnalités, notamment la création de compte, la recherche de chambres, la réservation d'offres, ainsi que des fonctions d'affichage, en utilisant les services web d'agence.

```
1 System.out.println("1. Afficher la liste des agences");
2 System.out.println("2. Afficher les d tails d'une agence");
3 System.out.println("3. Cr ation de compte pour une agence");
4 System.out.println("4. Afficher les d tails d'un hotel");
5 System.out.println("5. Afficher les reservations d'un hotel");
6 System.out.println("6. Recherche de chambres avec une agence");
7 System.out.println("7. R server des chambres avec une agence");
8 System.out.println("8.Lancer l'UI");
9 System.out.println("0. Quitter");
```

9 Interface Graphique

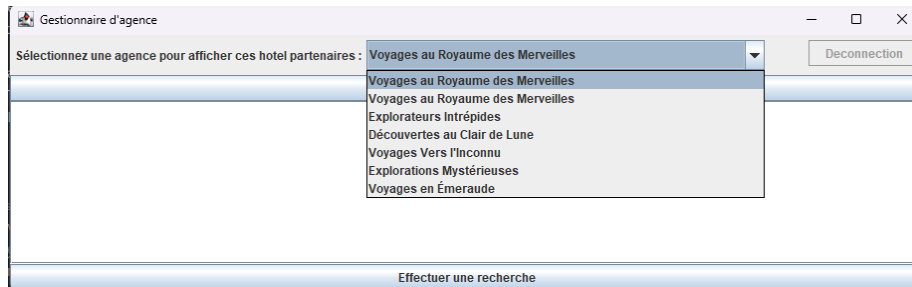
Dans le but d'offrir une expérience utilisateur plus visuelle et ergonomique, nous avons introduit une interface graphique. Cette interface permet aux utilisateurs d'explorer les fonctionnalités de manière plus interactive et d'interagir visuellement avec les informations.

Nous avons développé cette interface graphique intuitive en utilisant la bibliothèque **Swing**. Cette interface permet aux utilisateurs d'interagir avec les fonctionnalités offertes par l'application agence, simplifiant ainsi le processus de tests et garantissant une meilleure compréhension des différentes étapes.

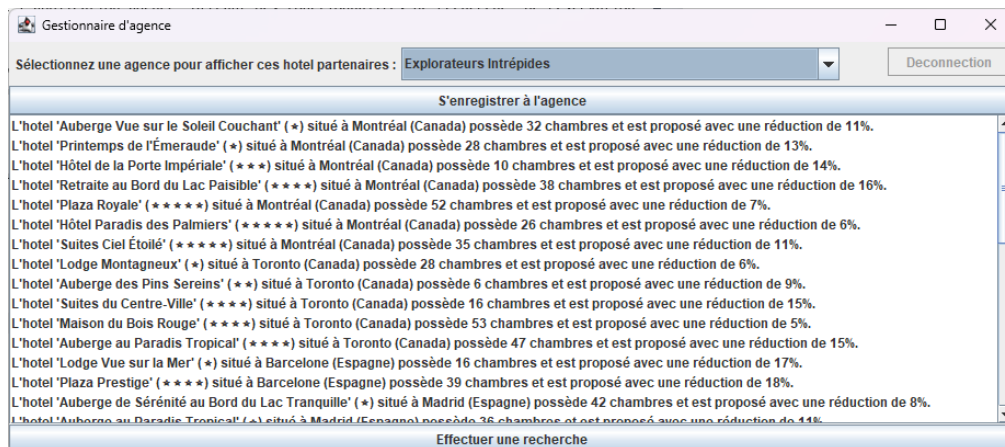
L'utilisation de Swing, une extension du toolkit AWT de Java, a permis de concevoir une interface graphique efficace et réactive. Cette approche a pour objectif d'optimiser la navigation des utilisateurs au sein de l'application agence, offrant des fonctionnalités de recherche, de réservation et de consultation des disponibilités de manière visuellement plus attrayante.

Cette nouvelle interface graphique renforce la convivialité de l'application agence, facilitant l'interaction avec les services web des hôtels et offrant une expérience utilisateur plus agréable. Elle constitue un ajout significatif pour simplifier les tests et améliorer la compréhension globale du fonctionnement de l'application côté agence.

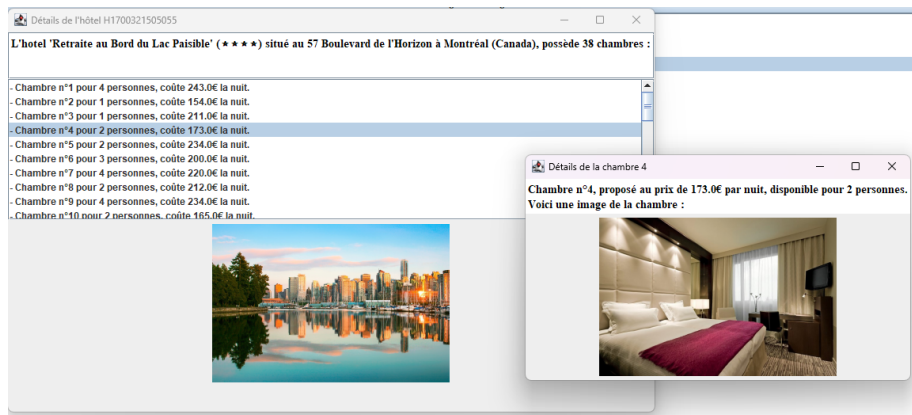
Il est possible de sélectionner une agence à partir du menu déroulant situé en haut de la fenêtre.



Après avoir sélectionné une agence, la liste complète de ses hôtels partenaires s'affiche à l'écran, accompagnée d'informations essentielles.



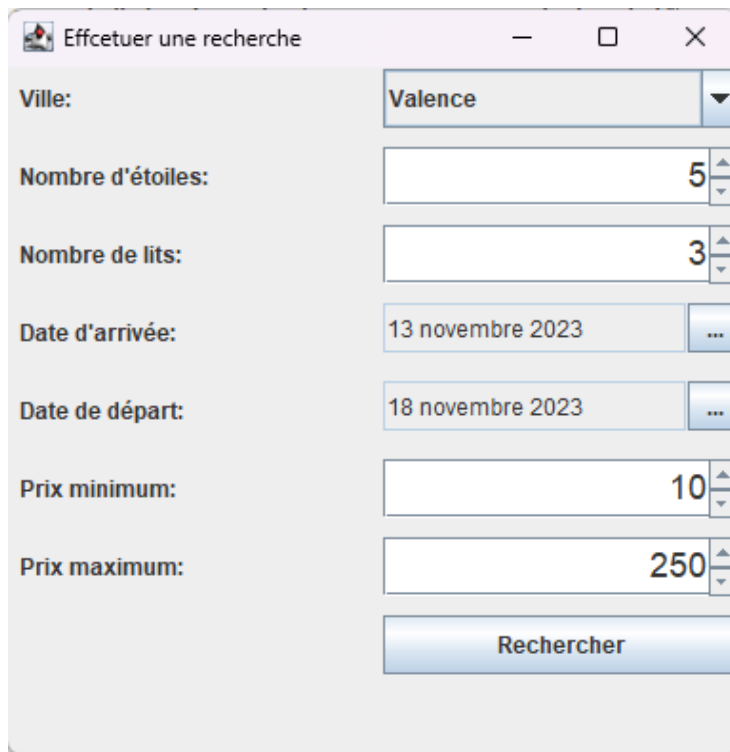
En cliquant sur un hôtel, on accède à sa liste de chambres, comprenant leur numéro, leur prix, leur capacité d'accueil, ainsi qu'une représentation visuelle de la ville de l'hôtel. De la même façon, en cliquant sur une chambre, on obtient ces informations détaillées ainsi que sa photo.



Pour effectuer une recherche ou une réservation, il est impératif de s'être connecté à une agence, pour cela, il faut cliquer sur le bouton situé en haut de la fenêtre. Depuis ce menu, il est possible de créer un nouveau compte dans une agence ou simplement de se connecter.

The image shows a login form window titled 'Connectez / Inscrivez vous :'. It has a close button (X) in the top right corner. The form contains two input fields: 'Nom d'utilisateur :' and 'Mot de passe :'. Below the input fields is a 'Valider' button.

Ensuite, il est possible de saisir nos critères de recherche et de lancer la recherche pour obtenir les offres disponibles.

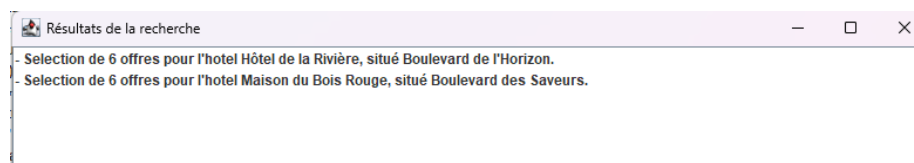


The screenshot shows a search window titled "Effctuer une recherche". It contains several input fields for search criteria:

- Ville:** A dropdown menu with "Valence" selected.
- Nombre d'étoiles:** A numeric input field with the value "5".
- Nombre de lits:** A numeric input field with the value "3".
- Date d'arrivée:** A date input field with the value "13 novembre 2023".
- Date de départ:** A date input field with the value "18 novembre 2023".
- Prix minimum:** A numeric input field with the value "10".
- Prix maximum:** A numeric input field with the value "250".

At the bottom of the form is a blue button labeled "Rechercher".

On obtient ainsi la liste des offres correspondantes regroupées par hôtels.



The screenshot shows a results window titled "Résultats de la recherche". It displays a list of search results:

- Selection de 6 offres pour l'hotel Hôtel de la Rivière, situé Boulevard de l'Horizon.
- Selection de 6 offres pour l'hotel Maison du Bois Rouge, situé Boulevard des Saveurs.

Après avoir cliqué sur une sélection d'offres, on a la liste des offres proposée par cet hôtel.

The screenshot shows a web application window titled "Détails de la selection". It displays a list of six hotel offers, each with a total price and a breakdown of room costs. Offer n°3 is highlighted in blue. A modal dialog titled "Confirmation de réservation" is open, asking "Voulez-vous réserver cette offre ?" with "Oui" and "Non" buttons.

Détails de la selection

Offre n°1 avec 2 chambres, proposée à 362.84€ :
- Chambre n°2 pour 2 personnes à 199.0€.
- Chambre n°8 pour 3 personnes à 187.0€.

Offre n°2 avec 2 chambres, proposée à 402.32€ :
- Chambre n°1 pour 1 personnes à 176.0€.
- Chambre n°5 pour 4 personnes à 252.0€.

Offre n°3 avec 3 chambres, proposée à 525.46€ :
- Chambre n°1 pour 1 personnes à 176.0€.
- Chambre n°3 pour 1 personnes à 196.0€.
- Chambre n°8 pour 3 personnes à 187.0€.

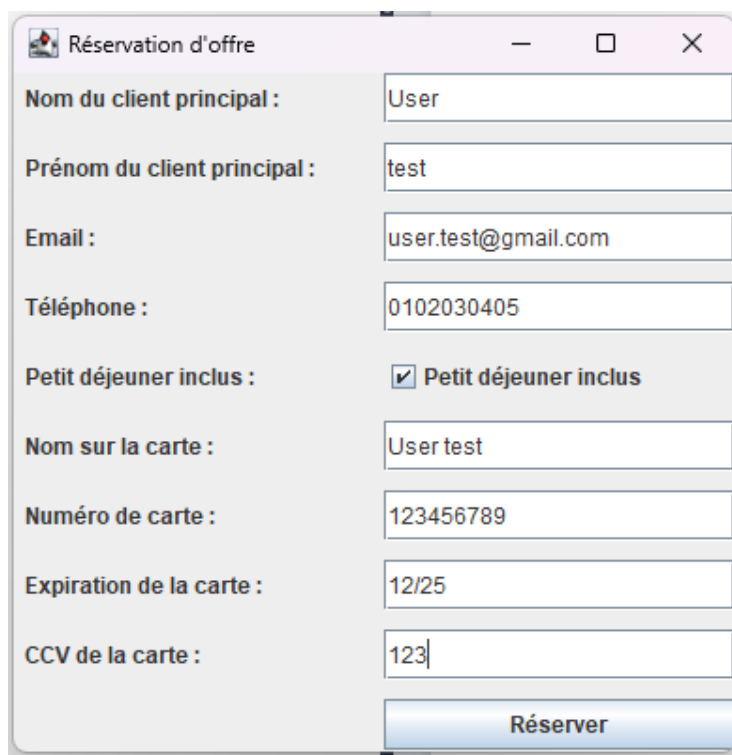
Offre n°4 avec 3 chambres, proposée à 570.58€ :
- Chambre n°1 pour 1 personnes à 176.0€.
- Chambre n°2 pour 2 personnes à 199.0€.
- Chambre n°4 pour 2 personnes à 232.0€.

Offre n°5 avec 4 chambres, proposée à 705.94€ :
- Chambre n°1 pour 1 personnes à 176.0€.
- Chambre n°3 pour 1 personnes à 196.0€.
- Chambre n°6 pour 1 personnes à 180.0€.
- Chambre n°2 pour 2 personnes à 199.0€.

Offre n°6 avec 5 chambres, proposée à 829.08€ :
- Chambre n°1 pour 1 personnes à 176.0€.
- Chambre n°3 pour 1 personnes à 196.0€.
- Chambre n°6 pour 1 personnes à 180.0€.
- Chambre n°11 pour 1 personnes à 184.0€.
- Chambre n°15 pour 1 personnes à 146.0€.

Confirmation de réservation
? Voulez-vous réserver cette offre ?
Oui Non

Une fenêtre s'affiche alors, permettant de saisir les informations nécessaires pour effectuer la réservation.

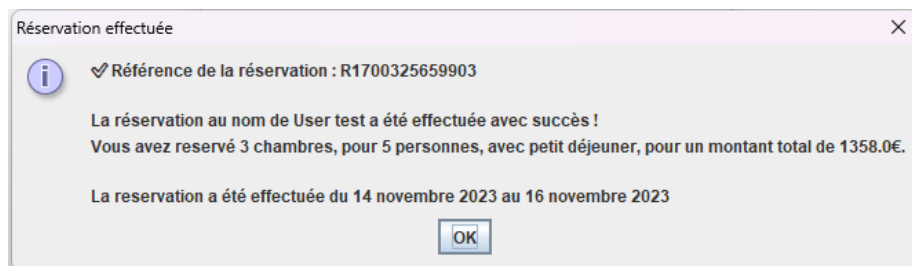


The screenshot shows a window titled "Réservation d'offre" with a standard Windows title bar (minimize, maximize, close buttons). The window contains a form with the following fields and values:

- Nom du client principal : User
- Prénom du client principal : test
- Email : user.test@gmail.com
- Téléphone : 0102030405
- Petit déjeuner inclus : ☒ Petit déjeuner inclus
- Nom sur la carte : User test
- Numéro de carte : 123456789
- Expiration de la carte : 12/25
- CCV de la carte : 123

At the bottom right of the form is a blue button labeled "Réserver".

Après avoir cliqué sur "Réserver", une nouvelle fenêtre s'affiche pour confirmer la réservation de l'offre sélectionnée, fournissant ainsi la référence de la réservation si toutes les informations fournies sont valides.



The screenshot shows a confirmation window titled "Réservation effectuée" with a close button (X) in the top right corner. The window contains the following information:

- An information icon (i) followed by a checkmark and the text: "Référence de la réservation : R1700325659903"
- A message: "La réservation au nom de User test a été effectuée avec succès !"
- Details: "Vous avez réservé 3 chambres, pour 5 personnes, avec petit déjeuner, pour un montant total de 1358.0€."
- Dates: "La reservation a été effectuée du 14 novembre 2023 au 16 novembre 2023"
- An "OK" button at the bottom center.

10 Conclusion

En conclusion, notre projet offre une solution complète pour la réservation d'hôtels en introduisant un système robuste basé sur l'architecture client/serveur. L'interaction entre les agences, les hôtels, et les clients est facilitée par l'utilisation efficace de services web SOAP. La mise en place d'une interface graphique utilisateur offre une expérience conviviale, permettant aux utilisateurs de naviguer facilement entre les agences et les hôtels partenaires, de consulter les offres disponibles, et de procéder à des réservations en toute simplicité. En intégrant des fonctionnalités telles que la génération automatique d'agences et leur association aléatoire à des hôtels partenaires, notre système simule un environnement dynamique et diversifié. En somme, notre application vise à fournir une plateforme flexible, offrant des services de réservation adaptés aux besoins variés des utilisateurs.

11 Read-Me

Pour exécuter et tester le projet, il faut suivre les étapes suivantes (avec l'IDE IntelliJ) :

1. Ouvrir le projet soaphotel et lancer le *main* depuis `HotelServicePublisher`.
2. Ouvrir le projet soapagence et lancer le *main* depuis `AgenceServicePublisher`.
3. Ouvrir le projet soapclient et lancer le *main* depuis `App`.
4. Interagir avec les différentes pages de notre interface graphique.