PROJET REST

Etienne

Martin-Chantereau

Université de
MONTPELLIER

Master Informatique
Parcours ICO

Framework

Spring

Spring-DATA-JPA

LANGUAGE

JAVA

SQL

SOMMAIRE

[redacted]

QUESTION 1 : ALGORITHME CONSULTATION :

```
    @GetMapping(uri+"/hotel/consultation/id_agence={identifiantAgence}/password_agence=
      {passwordAgence}/date_arrivee={dateArrivee}/date_depart={dateDepart}"
            +"/nbPersonne={nbPersonne}/pays={pays}/ville={ville}/etoiles={etoiles}/prixMin
={prixMin}/prixMax={prixMax}")
```

L'algorithme utilise la méthode hotelService.getHotelByPaysAndVilleAndEtoiles pour obtenir un objet Hotel en utilisant les paramètres pays, ville et etoiles afin d'exécuter une requête sql dérivée via soit l'objet est présent, soit vide.
Dans le cas de présence l'algorithme extrait et stocke le résultat de la requete dans la variable hotelPresent.
Ensuite, l'algorithme filtre la liste des agences partenaires de l'hôtel en utilisant la méthode stream() et la méthode filter(), en cherchant une agence avec l'identifiant spécifié (identifiantAgence) et un mot de passe correspondant (passwordAgence). Si une agence correspondante est trouvée, elle est renvoyée dans un objet Optional<AgencePartenaire>. Sinon, l'objet est vide (absent).

Si un objet AgencePartenaire est présent dans l'objet Optional<AgencePartenaire>, alors le coefficient de promotion de cette agence est extrait et stocké dans la variable coeffPromotion. Sinon, la valeur par défaut de coeffPromotion est définie sur 1.

Ensuite, l'algorithme utilise la méthode chambreService.findChambreByDisponbilitee pour obtenir une liste de chambres disponibles en utilisant les paramètres dateArrivee, dateDepart, coeffPromotion, prixMin, prixMax et nbPersonne.
Cette méthode vient exécuter la requête suivante se trouvant dans la class ChambreRepository :

```
@Query(value = "SELECT * FROM chambre WHERE chambre.id_chambre NOT IN "
                    + "(SELECT reservation.id_chambre FROM reservation WHERE
reservation.date_depart > :dateArrivee AND reservation.date_arrivee < :dateDepart)"
                    + "AND prix_chambre * :coeffPromotion BETWEEN :prixMin AND :prixMax
AND nombre_lits >= :nbPersonne",nativeQuery = true)
```

Pour chaque chambre de la liste obtenue, un nouvel objet Offre est créé en utilisant le coefficient de promotion (coeffPromotion) et la chambre, et est ajouté à la liste listOffreDispo.

Enfin la liste est retournée au client.

Si la liste est vide alors le client va afficher un message d'erreur sinon il va afficher l'ensemble des caractéristique de l'offre.

QUESTION 1 : Algorithme Reservation :

Le client a passée en paramètre l'ensemble des données nécessaire à une réservation, je ne voulais pas que l'agence est accès à tous le mapping de mes classes de l'hotel alors j'ai stocké tout les données dans une hashmap que j'envoie ensuite dans le body de la requete post HTPP.

Il utilise la méthode reservationService.findIfReservationIsPossible pour vérifier si une réservation est possible pour les dates spécifiées dans les paramètres reservation.getFirst("dateArrivee") et reservation.getFirst("dateDepart").

Cette méthode fait appel à la requête sql de la class ReservationRepository qui sélectionne tous les tuples de la table reservation où la date de départ est postérieure à la date spécifiée dans le paramètre dateArrivee et la date d'arrivée est antérieure à la date spécifiée dans le paramètre dateDepart.

```
@Query(value= "SELECT * FROM reservation WHERE date_depart > :dateArrivee AND
date_arrivee < :dateDepart", nativeQuery=true)
```

Cette méthode renvoie une liste vide si la réservation est possible, sinon elle renvoie une liste non vide.

S'il est possible de faire une réservation, l'algorithme crée un nouvel objet Reservation en utilisant les dates de réservation spécifiées dans les paramètres reservation.getFirst("dateArrivee") et reservation.getFirst("dateDepart"), ainsi que le nombre de lits spécifié dans reservation.getFirst("nbLitsResa").
Il crée également un nouvel objet Client en utilisant les prénom et nom spécifiés dans les paramètres reservation.getFirst("prenomClient") et reservation.getFirst("nomClient").

Ensuite, l'algorithme vérifie si la carte bancaire spécifiée dans les paramètres reservation.getFirst("numeroCarte"), reservation.getFirst("cardDateExpiration") et reservation.getFirst("cryptogramme") est déjà associée à un client existant en utilisant la méthode carteBancaireService.findCarteBancaireByNumeroANDcryptogrammeANDdateExpiration.

Cette méthode fait appel à une requête sql de CarteBancaireRepository :

```
@Query(value = "SELECT * FROM cartebancaire WHERE numero_carte =
:numeroCarte AND date_expiration= :dateExpiration AND cryptogramme =
:cryptogramme", nativeQuery = true)
```

Cas 1 :
Si la carte bancaire n'est pas trouvée, un nouvel objet CarteBancaire est créé en utilisant les détails de la carte bancaire spécifiés dans les paramètres, et est associé au client nouvellement créé.
La carte bancaire est enregistrée dans la base de données en utilisant la méthode carteBancaireService.createCarteBancaire.

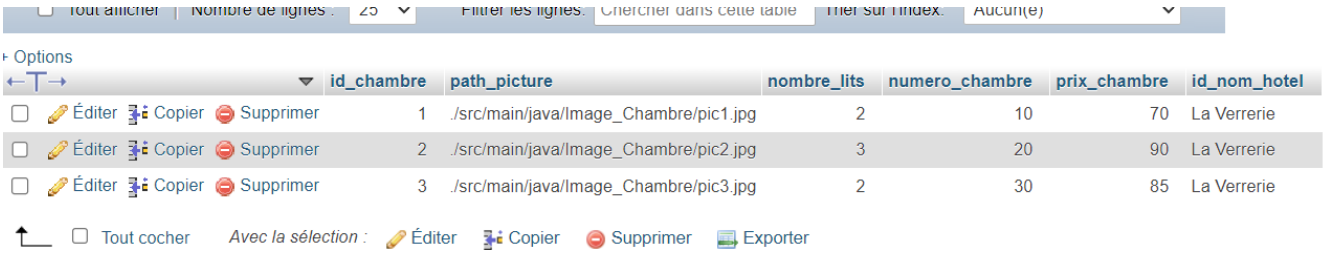
Cas 2 :
Si la carte bancaire est déjà associée à un client existant, alors le client nouvellement créé est associé à cette carte bancaire existante et est enregistré dans la base de données en utilisant la méthode clientService.createClient.

Ensuite, la réservation nouvellement créée est associée au client nouvellement créé ou existant et à la chambre spécifiée dans le paramètre reservation.getFirst("identifiantOffre"), qui est obtenue en utilisant la méthode chambreService.findChambreById. La réservation est enregistrée dans la base de données en utilisant la méthode reservationService.createReservation.

Puis on retourne une string qui renvoie soit les détails de la réservation soit une erreur si une réservation était déjà faite pour ces dates là.

QUESTION 2 : CONVERSION ET INTÉGRATION D'IMAGE

Pour que le client puisse recevoir une image de la chambre j'ai implémentée et modifiée ma solution la méthode réalisé pour le projet SOAP avec Christophe Houbron. Cela consiste à récupérer le path de la photo côté serveur et de la transformer en List de ByteCode.
De ce fait et partant du postulat qu'il ne faut pas enregistrer des jpegs dans une base de données pour une question de vitesse d'exécution j'ai décidé d'intégrer seulement dans les chambres le chemin d'accès (path) comme ceci :



Options

	id_chambre	path_picture	nombre_lits	numero_chambre	prix_chambre	id_nom_hotel
Éditer Copier Supprimer	1	./src/main/java/Image_Chambre/pic1.jpg	2	10	70	La Verrerie
Éditer Copier Supprimer	2	./src/main/java/Image_Chambre/pic2.jpg	3	20	90	La Verrerie
Éditer Copier Supprimer	3	./src/main/java/Image_Chambre/pic3.jpg	2	30	85	La Verrerie

Tout cocher Avec la sélection : Éditer Copier Supprimer Exporter

Ensuite pendant l'exécution de mon algorithme de consultation je peux convertir les paths en ByteCode pour transferable en XML il me suffit simplement de créer un attribut byteCode et une méthode de conversion du path dans ma classe Offre :

```
public class Offre extends Chambre {

	//Attributs
	….
	private byte[] picOfChamber;

	//Constructeur
	public Offre(float coeffPromotion, Chambre chambre) throws IOException {
		super(….)
		….
		setPicOfChamberByPathPicture(chambre.getPicOfChamberByPathPicture());
	}
	//Méthodes
	public void setPicOfChamberByPathPicture(String pathPicure) throws IOException {
		File fichier = new File(pathPicure);
		BufferedImage fileToByteCode = ImageIO.read(fichier);
		ByteArrayOutputStream ByteForXML = new ByteArrayOutputStream();
		ImageIO.write(fileToByteCode, "jpg", ByteForXML);
		this.picOfChamber = ByteForXML.toByteArray();
		}
```

[redacted]

Ainsi le corps de ma requête http get contiendra du byteCode :


```
{
    "disponibilitee": [
        "14-01-2023",
        "15-01-2023",
        "17-01-2023",
        "15-01-2024"
    ],
    "nomAgence": "Selectour",
    "adressehotel": "1 Rue de l'Égalité",
    "nomhotel": "La Verrerie",
    "etoileshotel": 3,
    "identifiant": 1,
    "numero": 10,
    "nbLits": 2,
    "prixChambreWithReduction": 69.0,
    "picOfChamber": "/9j/4AAQSkZJRgABAgAAAQABAAD/
        2wBDAAgGBgcGBQgHBwcJCQgKDBQNDAsLDBkSEw8UHRofHh0aHBwgJC4nICIsIxwcKDcpLDAxNDQ0Hyc5PTgyPC4zNDL/
        2wBDAQkJCQwLDBgNDRgyIRwhMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjL/wAARCAHCAqgDASIAAhEBAxEB/
        8QAHwAAAQUBAQEBAQEAAAAAAAAAAAECAwQFBgcICQoL/
        8QAtRAAAgEDAwIEAwUFBAQAAAF9AQIDAAQRBRIhMUEGE1FhByJxFDKBkaEII0KxwRVS0fAkM2JyggkKFhcYGRolJicoKSo0NTY3ODk6Q0RFRkdISUpTVFVWV1h
        ZWmNkZWZnaGlqc3R1dnd4eXqDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uHi4+Tl5ufo6erx8vP09fb3+Pn6/
        8QAHwEAAwEBAQEBAQEBAQAAAAAAAAECAwQFBgcICQoL/
        8QAtREAAgECBAQDBAcFBAQAAQJ3AAECAxEEBSExBhJBUQdhcRMiMoEIFEKRobHBCSMzUvAVYnLRChYkNOEl8RcYGRomJygpKjU2Nzg5OkNERUZHSElKU1RVVld
        YWVpjZGVmZ2hpanN0dXZ3eHl6goOEhYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3uLm6wsPExcbHyMnK0tPU1dbX2Nna4uPk5ebn6Onq8vP09fb3+Pn6/
        9oADAMBAAIRAxEAPwDLsfGms2DbHnF3EDjZcDJx/vdf511Nh480y5wl2ktnIerH50/Mc/
        pXmrffP1NJmvElSiz1o1JI9wt7iG7iEttPHPGejRvuFP59a8Rt7ma0mEttNJDIP4o22mumsPHmp22Fu0jvUHBLDY
```


QUESTION 3 : COMPARATEUR SERVICE WEB :

```
@GetMapping(uri+"/offre/ville={ville}/date_arrivee={dateArrivee}/date_depart={dateDepart}/nbPersonne={nbPersonne}/etoiles={etoiles}")
```

Ici c'est le comparateur qui sera le client et utilisateur final des données, transmise par l'agence, ce qui veut dire qu'il faut convertir l'agence comme un client/serveur. En effet le client va utilisée à la fois le webservice des hotels donc il sera client des hotels, et il sera serveur car il va projeter son propre serviceWeb.

Pour cela on modife les projets agences en leurs créeant une base de donnée :

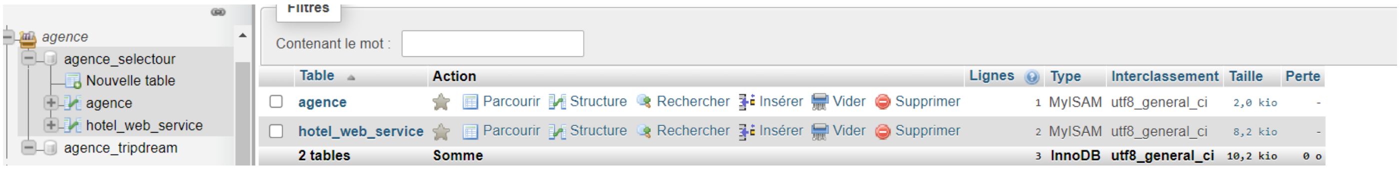


Table	Action							Lignes	Type	Interclassement	Taille	Perte
agence		Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	1	MyISAM	utf8_general_ci	2,0 kio	-
hotel_web_service		Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	2	MyISAM	utf8_general_ci	8,2 kio	-
2 tables	**Somme**							3	**InnoDB**	**utf8_general_ci**	10,2 kio	0 o

La table agence va contenir toute les informations d'une agence, et la table hotel_web_service l'url des webservices des différents hotels.

Pour faire simple on modifie le service web consultation de la question des hotels en leur retirant les paramètres de pays et du prix.

```
@GetMapping(uri+"/hotel/consultation/id_agence={identifiantAgence}/password_agence={passwordAgence}/date_arrivee={dateArrivee}/date_depart={dateDepart}"+"/nbPersonne={nbPersonne}/ville={ville}/etoiles={etoile
```

Ceci vient changer légèrement la requête de recherche d'hotel et des chambres comme suivant :

-Pour hotel :

```
Optional<Hotel> optHotel hotelService.getHotelByVilleAndEtoiles(ville, etoiles)
```

Pour la requête de recherche des disponibilités dans le repository des chambres :

```
"SELECT * FROM chambre WHERE chambre.id_chambre NOT IN "
+ "(SELECT reservation.id_chambre FROM reservation WHERE reservation.date_depart >
:dateArrivee AND reservation.date_arrivee < :dateDepart)"
                  + "AND nombre_lits >= :nbPersonne".
```

Voici la trace de l'appel des différents service web avec l'agence Selectour :

L'utilisateur va entrée toutes informations dans la CLI du Comparateur, qui va appeler le service web d'une agence avec l'url suivant (méthode GET) .

http://localhost:6666/agence/api/offre/ville=Gaillac/date_arrivee=2023-02-17/date_depart=2023-02-18/nbPersonne=2/etoiles=3

L'agence va ensuite récupérer l'identification de l'agence dans son webService (AgenceController):

```
Optional<Agence> optAgence = agenceService.getAgence();
            Agence agence = optAgence.get();
            ArrayList<Offre> listOffreDispo = new ArrayList<>();

            List<String> listHotelWebServiceURL = new ArrayList<>();
            listHotelWebServiceURL = hotelWebServiceURLService.getAllURL();
```

Puis pour chaque URL le programme va appeler le webService des hotels enrichie des données d'identifiant de l'agence et du motdepasse de celle-ci pour pouvoir calculer le prix avec réduction des chambres :

```
for (String hotelWebServiceURL : listHotelWebServiceURL) {
String uri =
hotelWebServiceURL+"consultation/id_agence="+agence.getIdentifiant()+"/password_agence=
"+agence.getPassword()+"/date_arrivee="+dateArrivee+"/date_depart="+dateDepart+"/nbPers
onne="+nbPersonne+"/ville="+ville+"/etoiles="+etoiles;
```

Exemple pour l'hotel de Gaillac :

http://localhost:8080/hotelservice/api/hotel/consultation/id_agence=1/password_agence=selectour/date_arrivee=2023-02-17/date_depart=2023-02-18/nbPersonne=2/ville=Gaillac/etoiles=3

Les données récupérées seront par exemples les suivantes :


```
    {
        "idChambre": 0,
        "numero": 10,
        "nbLits": 2,
        "prixChambre": 70.0,
        "listReservation": [],
        "disponibilitee": [
            "14-01-2023",
            "15-01-2023",
            "17-01-2023",
            "15-01-2024"
        ],
        "prixChambreWithReduction": 69.0,
        "identifiant": 1,
        "nomhotel": "La Verrerie",
        "adressehotel": "1 Rue de l'Égalité",
        "etoileshotel": 3,
        "picOfChamber": …. Volontairement Réduit pour l'exemple car les données sont longues.
    },
```


Il faut donc enrichir la réponse des webServices des hotels avec le nom de l'agence comme suivant :

```
if (Arrays.asList(listOffre).isEmpty()) {
                        System.out.println("Pour "+ hotelWebServiceURL+"nous n'avons
                        rien trouvé");
                        System.out.println();
                }else {
                        Arrays.asList(listOffre).forEach(System.out::println);
                        for (Offre offre : Arrays.asList(listOffre)) {
                              System.out.println(offre);
                              offre.setNomAgence(agence.getNom());
                              listOffreDispo.add(offre);
                        }
                }
```

Algorithme Consultation CLASS OFFRECONTROLLER :

```
@RestController
    @GetMapping(uri+"/hotel/consultation/id_agence={identifiantAgence}/password_agence={passwordAgence}/date_arrivee={dateArrivee}/date_depart={dateDepart}"
            +
"/nbPersonne={nbPersonne}/pays={pays}/ville={ville}/etoiles={etoiles}/prixMin={prixMin}/prixMax={prixMax}")
    public ArrayList<Offre> consultationOffre(@PathVariable int identifiantAgence,
@PathVariable String passwordAgence,
            @PathVariable int nbPersonne, @PathVariable String dateArrivee, @PathVariable String dateDepart,
            @PathVariable String pays, @PathVariable String ville, @PathVariable int etoiles,
            @PathVariable float prixMin, @PathVariable float prixMax) throws IOException{

        float coeffPromotion;
        ArrayList<Offre> listOffreDispo = new ArrayList<>();

        Optional<Hotel> optHotel = hotelService.getHotelByPaysAndVilleAndEtoiles(pays, ville, etoiles);
        if (optHotel.isPresent()) {

            Hotel hotelPresent = optHotel.get();

            Optional<AgencePartenaire> filterAgencePartenaire = hotelPresent.getAgencesPartenaire().stream()
                    .filter(agencePartenaire -> agencePartenaire.getIdAgence() == identifiantAgence
                    &&
agencePartenaire.getAgencePartenairePKID().getPasswordAgence().equals(passwordAgence))
                    .findFirst();
            if (filterAgencePartenaire.isPresent()) {
                coeffPromotion = filterAgencePartenaire.get().getCoeffPromotion();
            }else {
                coeffPromotion = 1;
            }

            List<Chambre> listChambreDispo = chambreService.findChambreByDisponbilitee(dateArrivee, dateDepart, coeffPromotion, prixMin, prixMax, nbPersonne);

            for (Chambre chambre : listChambreDispo) {
                Offre offre = new Offre(coeffPromotion, chambre);
                listOffreDispo.add(offre);
            }

        }
        System.out.println(listOffreDispo.toString());
        return listOffreDispo;
    }
```

Algorithme Reservation :

```
@PostMapping(uri+"/reservation")
    public String doReservation(@RequestBody MultiValueMap<String, String> reservation) {
            String response = "ok";
//      Test Agence
            if
(reservationService.findIfReservationIsPossible(reservation.getFirst("dateArrivee"),
                    reservation.getFirst("dateDepart")).isEmpty()) {

                Reservation reservationToSave = new
Reservation(reservation.getFirst("dateArrivee"),
                        reservation.getFirst("dateDepart"),
Integer.parseInt(reservation.getFirst("nbLitsResa")));
                Client clientToSave = new
Client(reservation.getFirst("prenomClient"),reservation.getFirst("nomClient"));

                if
(!carteBancaireService.findCarteBancaireByNumeroANDcryptogrammeANDdateExpiration(reservati
on.getFirst("numeroCarte"), reservation.getFirst("cardDateExpiration"),
                        Integer.parseInt(reservation.getFirst("cryptogramme"))).isPresent(
)) {

                    CarteBancaire cbToSave = new
CarteBancaire(reservation.getFirst("numeroCarte"),reservation.getFirst("cardDateExpiration
"),
                            Integer.parseInt(reservation.getFirst("cryptogramme")));

                    clientToSave.setCarteBancaire(cbToSave);
                    cbToSave.setClient(clientToSave);
                    carteBancaireService.createCarteBancaire(cbToSave);

                } else {
                    CarteBancaire cbAlreadyBind =
carteBancaireService.findCarteBancaireByNumeroANDcryptogrammeANDdateExpiration(
                                    reservation.getFirst("numeroCarte"),
reservation.getFirst("cardDateExpiration"),
                                    Integer.parseInt(reservation.getFirst("cryptogramme"))
).get();
                    clientToSave.setCarteBancaire(cbAlreadyBind);
                    clientService.createClient(clientToSave);
                }
                reservationToSave.setClient(clientToSave);
                reservationToSave.setChambre(chambreService
                        .findChambreById(Integer.parseInt(reservation.getFirst("identifian
tOffre")))).get());
                reservationService.createReservation(reservationToSave);
                response = reservationToSave.toString();
            } else {
                response = "Erreur l'offre est déjà réservée pour les dates demandées";
            }
            return response;
        }
```

Algorithme COMPARATEUR

Côté Agence

```
	@GetMapping(uri+"/offre/ville={ville}/date_arrivee={dateArrivee}/date_depart={dat
eDepart}"
			+ "/nbPersonne={nbPersonne}/etoiles={etoiles}")
	public ArrayList<Offre> consultationOffre(@PathVariable int nbPersonne,
@PathVariable String dateArrivee, @PathVariable String dateDepart,
			@PathVariable String ville, @PathVariable int etoiles) throws
IOException{

		//Je récupère mon agence
		Optional<Agence> optAgence = agenceService.getAgence();
		Agence agence = optAgence.get();
		ArrayList<Offre> listOffreDispo = new ArrayList<>();

		List<String> listHotelWebServiceURL = new ArrayList<>();
		listHotelWebServiceURL = hotelWebServiceURLService.getAllURL();

		for (String hotelWebServiceURL : listHotelWebServiceURL) {
			String uri =
hotelWebServiceURL+"consultation/id_agence="+agence.getIdentifiant()+"/password_agence=
"+agence.getPassword()+"/date_arrivee="+dateArrivee

	+"/date_depart="+dateDepart+"/nbPersonne="+nbPersonne+"/ville="+ville+"/etoiles="
+etoiles;

			Offre [] listOffre = proxy.getForObject(uri, Offre[].class);
			//J'ajoute le résultat à l'arraylist

			System.out.println("test");
			System.out.println(listOffre.toString());

			if (Arrays.asList(listOffre).isEmpty()) {
				System.out.println("Pour "+ hotelWebServiceURL+"nous n'avons
rien trouvé");
				System.out.println();
			}else {
				Arrays.asList(listOffre).forEach(System.out::println);
				for (Offre offre : Arrays.asList(listOffre)) {
					System.out.println(offre);
					offre.setNomAgence(agence.getNom());
					listOffreDispo.add(offre);
				}
			}
			//Pour chaque résultat je rajoute les informations manquantes.
		}
		return listOffreDispo;
	}
```

Côté Hotel les seuls changement notable par rapport à consultation sont les `paramètres de` recherche d'hotel :

```
Optional<Hotel> optHotel = hotelService.getHotelByVilleAndEtoiles(ville, etoiles);
```

Et de la disponibilitée des chambres :

```
List<Chambre> listChambreDispo = chambreService.findChambreByDisponbilitee(dateArrivee,
dateDepart, nbPersonne);
```

Resultat Question 1 Consultation :

NOTA BENE le nom des villes et pays des hotels est récupérée par l'appel d'un webService respectif pour chacune des informations.

```
Mercie d'entrer un URL de service Web valide
http://localhost:
0.Quit.
 Taper 1. Pour consulter les Offres
 Taper 2. Pour effectuer une réservation
1
CLI consultation :
Merci d'entrée un identifiant correct pour l'agence selectour
1
Merci d'entrée le mot de passe correct de l'agence selectour
selectour
Veuillez entrer une date d'arrivée (jj-mm-yyyy) après le : 15-01-2023
22-02-2023
Veuillez entrer une date de départ (jj-mm-yyyy) qui se situe après votre date d'arrivée le : 22-02-2023
23-02-2023
Merci d'entrée un nombre de personne qui doit être supérieur à 0
2
Veuillez entrée un pays parmi la liste proposée :
[France, Belgique]
France
Veuillez entrée une ville parmi la liste proposée :
[Gaillac]
Gaillac
Quelle catégorie d'hotel recherchez-vous (nombres d'étoiles) : 1, 2, 3, 4 ou 5 étoiles ?
3
Veuillez indiquer le prix minimum différent de 0 pour une intervalle de prix :
0
Veuillez indiquer le prix maximum pour un intervalle de prix :
sachant que vous avez indiqué le prix minimum est de 0.0
120

Pour l'hotel Progress désolé nous n'avons pas trouvée d'offre correspondant à vos critère de recherche

Offre [disponibilitee=[14-01-2023, 15-01-2023, 17-01-2023, 15-01-2024], nomHotel=La Verrerie, prixChambreWithReduction=69.0, identifiant=1, numero=10, nbLits=2, picO
Offre [disponibilitee=[15-01-2023, 15-01-2024], nomHotel=La Verrerie, prixChambreWithReduction=89.0, identifiant=2, numero=20, nbLits=3, picOfChamber=[B@64c781a9]
Offre [disponibilitee=[15-01-2023, 15-01-2024], nomHotel=La Verrerie, prixChambreWithReduction=84.0, identifiant=3, numero=30, nbLits=2, picOfChamber=[B@71ed560f]
```

Resultat Question 1 Reservation :

```
0.Quit.
 Taper 1. Pour consulter les Offres
 Taper 2. Pour effectuer une réservation
2
CLI Reservation :
Merci d'entrée un identifiant correct pour l'agence selectour
1
Veuillez entrée le login de l'agence
sel_log
Merci d'entrée le mot de passe correct de l'agence selectour
selectour
Veuillez entrée un nom d'hotel parmi la liste proposée :
[Progress, La Verrerie]
La Verrerie
Veuillez entrée un numéro d'offre parmi la liste proposée :
[1, 2, 3]
1
Merci d'entrée un prénom valide
Etienne
Merci d'entrée un nom de Famille valide
a
Veuillez entrer votre numéro à 16 chiffre de Carte Bancaire
123456781234588
Un numéro de Carte Bancaire doit être composé de 16 chiffres, merci de réessayer votre entrée
Veuillez entrer votre numéro à 16 chiffre de Carte Bancaire
1234567881234567
Merci de bien vouloir entrer le cryptogramme de votre carte Bancaire
222
Merci d'entrer la date d'expiration au format (mm/yy) de votre carte après le 01/23
02/25
Veuillez entrer une date d'arrivée (jj-mm-yyyy) après le : 15-01-2023
16-02-2023
Veuillez entrer une date de départ (jj-mm-yyyy) qui se situe après votre date d'arrivée le : 16-02-2023
17-02-2023
Merci d'entrée un nombre de personne qui doit être supérieur à 0
2
http://localhost:8080/hotelservice/api/reservation
Reservation [idReservation=2, dateArrivee=2023-02-16, dateDepart=2023-02-17, nbLitsResa=2]
```

Resultat Question 2 :

```
ambreWithReduction=70.0, picOfChamber=[B@25f15f50]
 picOfChamber=[B@52b46d52]
 picOfChamber=[B@7327a447]
ambreWithReduction=69.0, picOfChamber=[B@67022ea]
 picOfChamber=[B@2954f6ab]
 picOfChamber=[B@58fbd02e]
```

Resultat Question 3 :

```
CLI comparaison :
Veuillez entrée une ville parmi la liste proposée  :
[Bruxelles, Gaillac]
Gaillac
Veuillez entrer une date d'arrivée (jj-mm-yyyy) après le : 15-01-2023
22-02-2023
Veuillez entrer une date de départ (jj-mm-yyyy) qui se situe après votre date d'arrivée le : 22-02-2023
23-02-2023
Merci d'entrée un nombre de personne qui doit être supérieur à 0
2
Quelle catégorie d'hotel recherchez-vous (nombres d'étoiles) : 1, 2, 3, 4 ou 5 étoiles ?
3
Offre [disponibilitee=[14-01-2023, 15-01-2023, 17-01-2023, 15-01-2024], nomAgence=TripDream, adressehotel=1 Rue de l'Égalité, nomhotel=La Verrerie, etoileshotel=3, identi
Offre [disponibilitee=[15-01-2023, 15-01-2024], nomAgence=TripDream, adressehotel=1 Rue de l'Égalité, nomhotel=La Verrerie, etoileshotel=3, identifiant=2, numero=20, nbLi
Offre [disponibilitee=[15-01-2023, 15-01-2024], nomAgence=TripDream, adressehotel=1 Rue de l'Égalité, nomhotel=La Verrerie, etoileshotel=3, identifiant=3, numero=30, nbLi
Offre [disponibilitee=[14-01-2023, 15-01-2023, 17-01-2023, 15-01-2024], nomAgence=Selectour, adressehotel=1 Rue de l'Égalité, nomhotel=La Verrerie, etoileshotel=3, identi
Offre [disponibilitee=[15-01-2023, 15-01-2024], nomAgence=Selectour, adressehotel=1 Rue de l'Égalité, nomhotel=La Verrerie, etoileshotel=3, identifiant=2, numero=20, nbLi
Offre [disponibilitee=[15-01-2023, 15-01-2024], nomAgence=Selectour, adressehotel=1 Rue de l'Égalité, nomhotel=La Verrerie, etoileshotel=3, identifiant=3, numero=30, nbLi
```

```
?

Dream, adressehotel=1 Rue de l'Égalité, nomhotel=La Verrerie, etoileshotel=3, identifiant=1, numero=10, nbLits=2, prixChambreWithReduction=70.0, picOfChamber=[B@25f
e de l'Égalité, nomhotel=La Verrerie, etoileshotel=3, identifiant=2, numero=20, nbLits=3, prixChambreWithReduction=90.0, picOfChamber=[B@52b46d52]
e de l'Égalité, nomhotel=La Verrerie, etoileshotel=3, identifiant=3, numero=30, nbLits=2, prixChambreWithReduction=85.0, picOfChamber=[B@7327a447]
ctour, adressehotel=1 Rue de l'Égalité, nomhotel=La Verrerie, etoileshotel=3, identifiant=1, numero=10, nbLits=2, prixChambreWithReduction=69.0, picOfChamber=[B@670
e de l'Égalité, nomhotel=La Verrerie, etoileshotel=3, identifiant=2, numero=20, nbLits=3, prixChambreWithReduction=89.0, picOfChamber=[B@2954f6ab]
e de l'Égalité, nomhotel=La Verrerie, etoileshotel=3, identifiant=3, numero=30, nbLits=2, prixChambreWithReduction=84.0, picOfChamber=[B@58fbd02e]
```