Nom et Prénom: Bouaziz Imene
Formation: M1 GL

Architectures Distribuées
TP 3 REST

Lien vers le code et la demo:

Click here to visit code and demo:
https://drive.google.com/drive/folders/1GlYBu3XCdvAVT15xA6Wnd$_hFE6dtlbcj?usp = sharing$

Contents

1 Conception.

1.1 Diagramme de classe

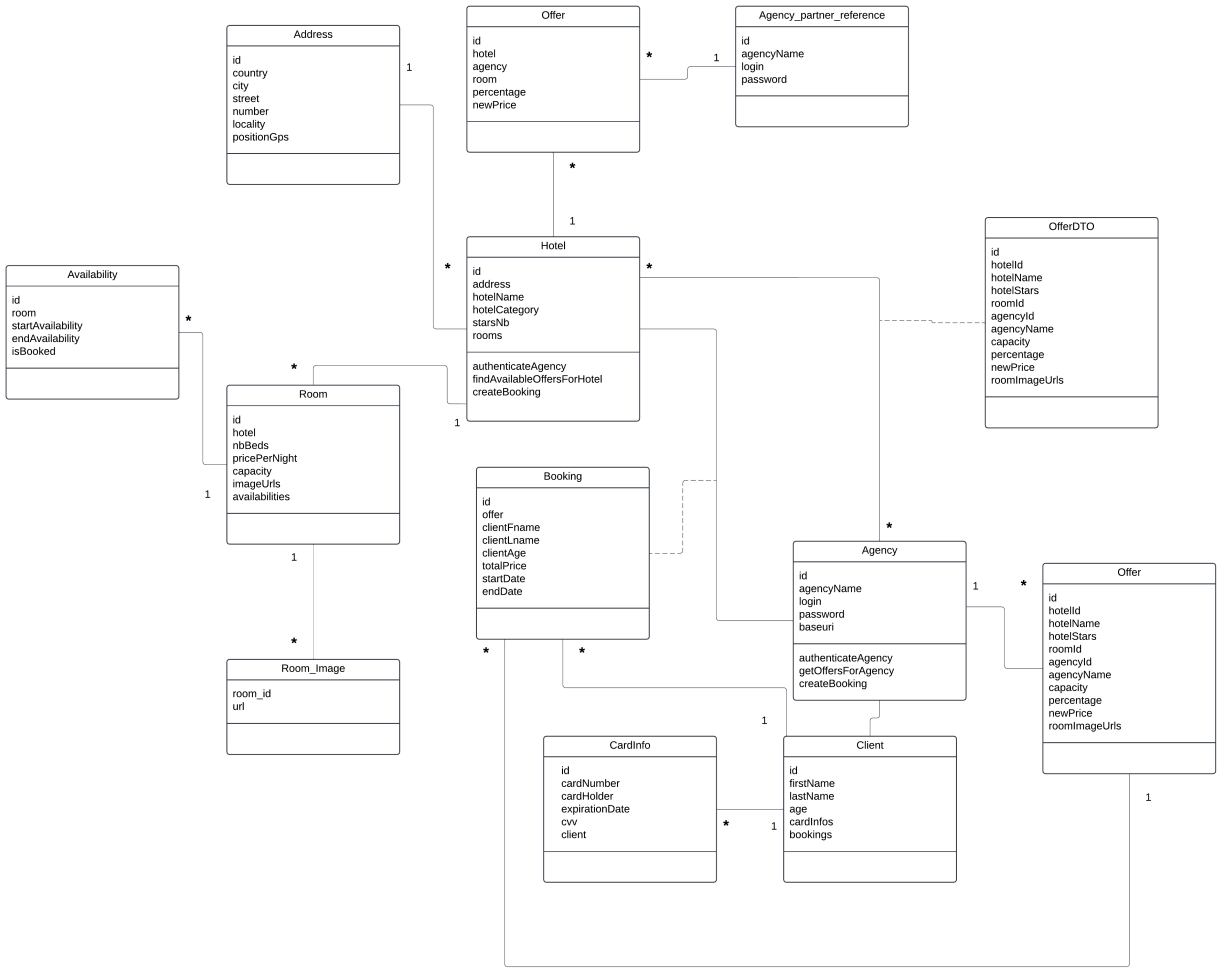

Figure 1: Class Diagram

Note: La table offre dans la bd agence contient les memes attributs que l'entité OfferDTO, et la table booking dans hotel_db et agency_db contient les memes attributs aussi.

1.2 Diagrammes D'activité

Diagramme basé sur le hotel AvailabilityController:

expliquant la recherche de disponibilité d'offres par hotel.

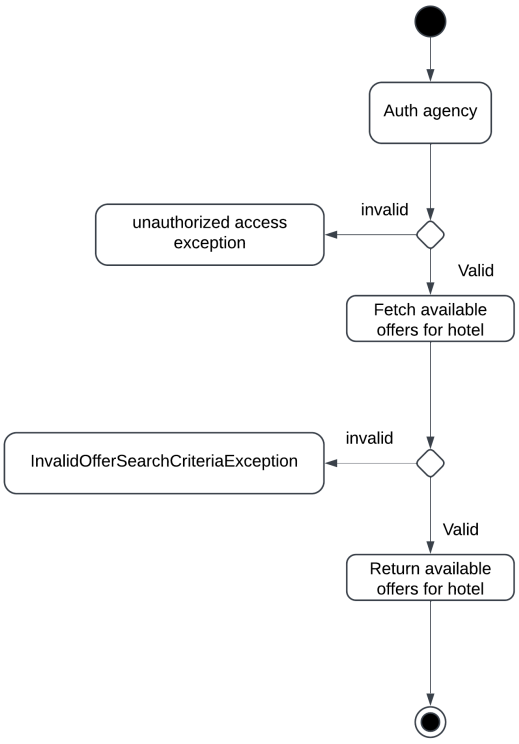

Figure 2: Hotel offer availability activity diagram

Diagramme basé sur le hotel BookingController:

expliquant les démarches de réservation hotel

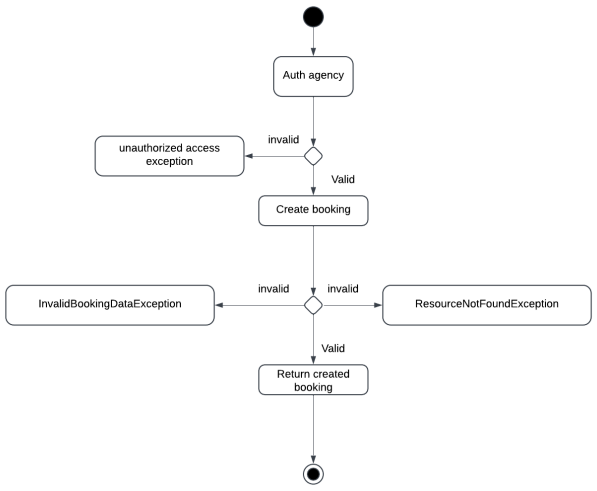

Figure 3: Hotel booking activity diagram

Diagramme basé sur l'agence AgencyController:

expliquant la recherche de disponibilité d'offres dans tous les hotels partenaires d'une agence.

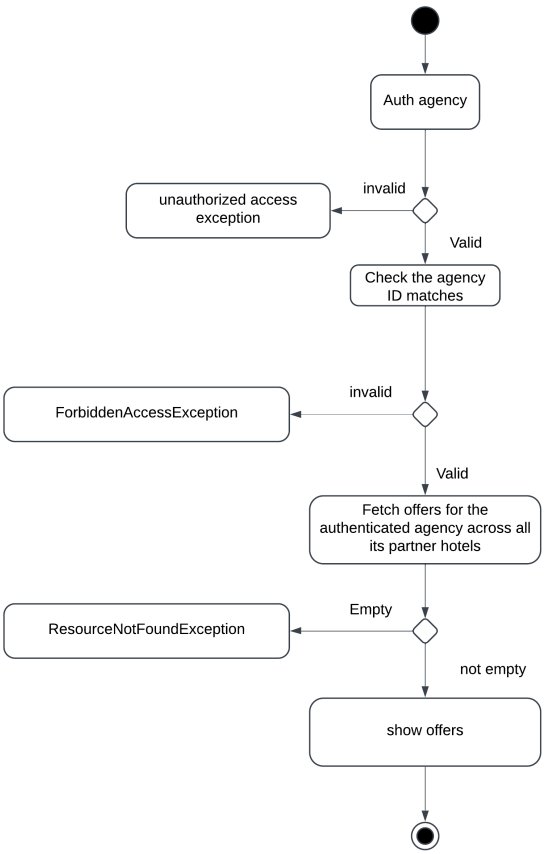

Figure 4: Agency offer availability activity diagram

Diagramme basé sur l'agence BookingController:

expliquant les démarches de réservation par une agence.

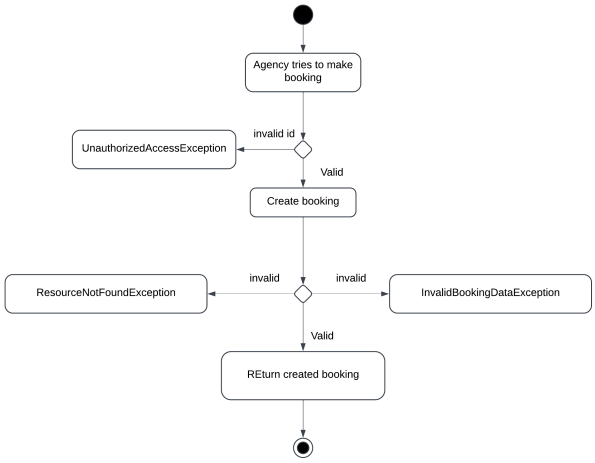

Figure 5: Agency booking activity diagram

2 Technologies utilisées

- **Backend:** Java / Spring boot
- **Base de données:** Mysql (phpmyadmin)
- **Postman:** pour les testes d'API
- **Frontend:** React JS
- **Bootstrap :** pour une interface utilisateur réactive et conviviale.
- **jsPDF :** pour générer dynamiquement des fichiers PDF.
- **Axios :** pour la communication entre l'IHM et les services des agences.

3 Structure de la Base de Données

On a 2 bases de données, une pour les agences soit agency_db et une pour les hotels soit hotel_db.

3.1 hotel_db

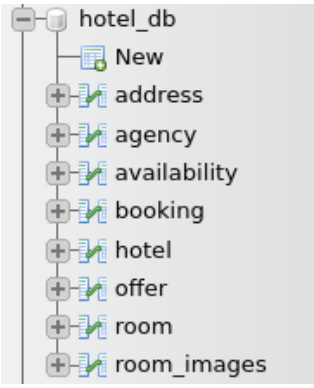

Figure 6: Structure BD Hotel

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)			No	None		AUTO_INCREMENT	Change Drop More
2	address_id	bigint(20)			Yes	NULL			Change Drop More
3	hotel_name	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More
4	hotel_category	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More
5	stars_nb	int(11)			Yes	NULL			Change Drop More

Figure 7: Structure table hotel

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)			No	None		AUTO_INCREMENT	Change Drop More
2	country	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More
3	city	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More
4	street	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More
5	number	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More
6	locality	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More
7	position_gps	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More

Figure 8: Structure table adresse

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)			No	None		AUTO_INCREMENT	Change Drop More
2	hotel_id	bigint(20)			Yes	NULL			Change Drop More
3	nb_beds	int(11)			Yes	NULL			Change Drop More
4	price_per_night	double			Yes	NULL			Change Drop More
5	capacity	int(11)			Yes	NULL			Change Drop More

Figure 9: Structure table chambre

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	room_id	bigint(20)			No	None			Change Drop More
2	image_url	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More

Figure 10: Structure table des images de chambres

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)			No	None		AUTO_INCREMENT	Change Drop More
2	room_id	bigint(20)			No	None			Change Drop More
3	start_availability	date			No	None			Change Drop More
4	end_availability	date			No	None			Change Drop More
5	is_booked	tinyint(1)			Yes	0			Change Drop More

Figure 11: Structure de table availability

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)			No	None		AUTO_INCREMENT	Change Drop More
2	agency_name	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More
3	login	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
4	password	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More

Figure 12: Structure de table agency

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)			No	None		AUTO_INCREMENT	Change Drop More
2	hotel_id	bigint(20)			Yes	NULL			Change Drop More
3	agency_id	bigint(20)			Yes	NULL			Change Drop More
4	room_id	bigint(20)			Yes	NULL			Change Drop More
5	percentage	double			Yes	NULL			Change Drop More
6	new_price	double			Yes	NULL			Change Drop More

Figure 13: Structure de table offer

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)			No	None		AUTO_INCREMENT	Change Drop More
2	offer_id	bigint(20)			Yes	NULL			Change Drop More
3	client_fname	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More
4	client_lname	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More
5	client_age	int(11)			Yes	NULL			Change Drop More
6	total_price	double			Yes	NULL			Change Drop More
7	start_date	date			No	None			Change Drop More
8	end_date	date			No	None			Change Drop More

Figure 14: Structure de table booking

3.2 agency_db

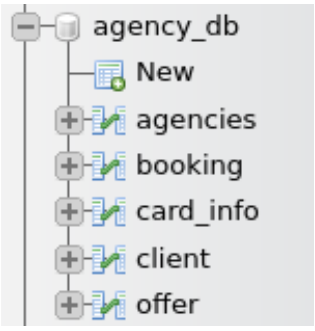

Figure 15: Structure de BD agence

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)			No	None		AUTO_INCREMENT	Change Drop More
2	name	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
3	login	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
4	password	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
5	base_uri	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More

Figure 16: Structure table agences

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)			No	None		AUTO_INCREMENT	Change Drop More
2	first_name	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
3	last_name	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
4	age	int(11)			No	None			Change Drop More

Figure 17: Structure table clients

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)			No	None		AUTO_INCREMENT	Change Drop More
2	card_number	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More
3	card_holder	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
4	expiration_date	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More
5	cvv	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More
6	client_id	bigint(20)			Yes	NULL			Change Drop More

Figure 18: Structure table des cartes bancaire

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)			No	None		AUTO_INCREMENT	Change Drop More
2	hotel_id	bigint(20)			No	None			Change Drop More
3	agency_id	bigint(20)			No	None			Change Drop More
4	room_id	bigint(20)			No	None			Change Drop More
5	percentage	double			No	None			Change Drop More
6	new_price	double			No	None			Change Drop More
7	hotel_name	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
8	hotel_stars	int(11)			No	None			Change Drop More
9	room_image_urls	varbinary(255)			Yes	NULL			Change Drop More
10	agency_name	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More

Figure 19: Structure table offre

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)			No	None		AUTO_INCREMENT	Change Drop More
2	offer_id	bigint(20)			No	None			Change Drop More
3	client_id	bigint(20)			No	None			Change Drop More
4	client_fname	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
5	client_lname	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
6	client_age	int(11)			No	None			Change Drop More
7	total_price	double			No	None			Change Drop More
8	start_date	date			No	None			Change Drop More
9	end_date	date			No	None			Change Drop More

Figure 20: Structure table booking

4 Architecture et Implémentation de l'Application des services de l'hotel

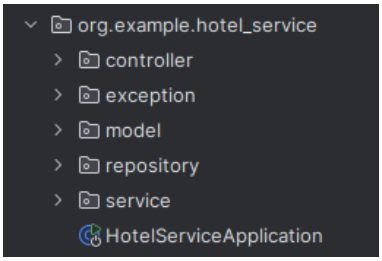

Figure 21: Architecture générale du serveur hotel

4.1 Model

pour la définition des entités selon ma base de données. contiennet les attributs, les constructeurs et les méthodes setters et getters.

Ici **Hotel** c'est la liste des hotels et **Agency** c'est une courte description de la liste des agences qui sont en partenariat avec les hotels, les deux sont liés par la table **offer**.

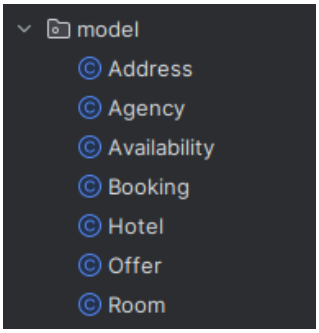

Figure 22: les models du serveur hotel

4.2 Repository

Mises dans le but de gérer les interactions entre le code de l'application et la base de données de manière organisée et découplée, dans ce cas par **JpaRepository**.

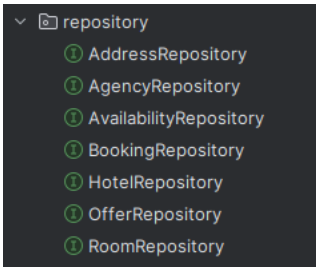

Figure 23: Repositories

- **AgencyRepository:** on peut trouver la méthode `findById`
- **AvailabilityRepository:** on peut trouver la méthode `findByRoomIdAndStartAvailabilityBeforeAndEndAvailabilityAfter`
- **OfferREpository:** on peut trouver la méthode `findByRoomId`

4.3 DTO

Le role du DTO (Data Transfer Object) est de servir d'objet intermédiaire pour le transfert des données dont celles de l'offre proposé par l'hotel aux agences et les détails de chambres, et communiquer avec un service externe, comme le serveur d'agence, en facilitant sérialisation/désérialisation.

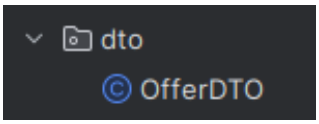

Figure 24: Offer DTO

4.4 Service

On a utilisé les services dans notre architecture pour respecter les bonnes pratiques du **modèle MVC** et garantir une séparation claire des responsabilités. Les contrôleurs (Controller) se concentrent uniquement sur la gestion des requêtes et réponses HTTP, tandis que la logique métier est déléguée aux services. Cela permet de garder les contrôleurs légers, tout en rendant la logique métier **réutilisable et facilement testable.**

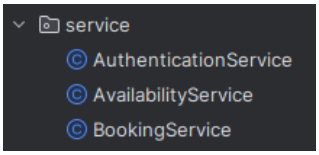

Figure 25: Services

1. **AuthenticationService**
   - `authenticateAgency` : Vérifie les informations de connexion d'une agence et renvoie l'agence authentifiée si les identifiants sont corrects.
2. **AvailabilityService**
   - `findAvailableOffersForHotel` : Recherche et renvoie les offres disponibles pour un hôtel spécifique selon les critères fournis (ville, catégorie, capacité, prix, etc.). La comparaison du *maxPrice* est effectuée avec le *newPrice*, qui est calculé en appliquant une remise sur le prix par nuit de la chambre.
3. **BookingService**
   - `createBooking` : Crée une réservation pour une offre donnée, vérifie les disponibilités, calcule le prix total, et met à jour les périodes de disponibilité associées.

4.5 Controller

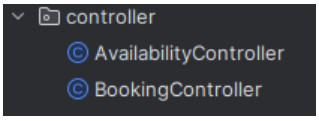

Figure 26: Controller

Dans cette API de service hôtelier, chaque hôtel est identifié de manière unique par son ID. Les opérations liées à chaque hôtel, comme la récupération des offres disponibles ou la création de réservations, sont effectuées en fonction de cet ID, garantissant ainsi que les données de chaque hôtel sont manipulées de façon distincte.

Les contrôleurs suivants gèrent les opérations de disponibilité et de réservation pour chaque hôtel, en se basant sur cet ID unique:

1. **AvailabilityController**
   - `getAvailableOffersForHotel` : Cette méthode permet de récupérer les offres disponibles pour un hôtel spécifique, en fonction des critères et en utilisant la méthode definie dans **AvailabilityService**. Le processus inclut l'authentification de l'agence avec le login et le mot de passe, puis recherche des offres et retourne celles qui correspondent aux critères pour cet hotel seulement.
2. **BookingController**
   - `createBooking` : Cette méthode crée une réservation pour un hôtel spécifique en utilisant la méthode du service. L'authentification de l'agence est également effectuée avant la création de la réservation.

4.6 Exceptions

Dans cette API de service hôtelier, la gestion des exceptions permet de traiter les erreurs de manière claire et spécifique. Lorsqu'une opération échoue, l'API renvoie des messages d'erreur détaillés qui expliquent la nature du problème.

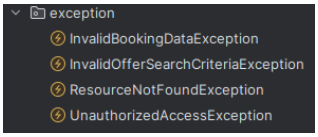

Figure 27: Exceptions

Les principales exceptions traitées dans l'API incluent :

1. **UnauthorizedAccessException** : Cette exception est lancée lorsqu'une tentative d'accès est effectuée avec des identifiants d'agence invalides. Elle assure que seules les agences autorisées peuvent accéder aux données de l'hotel.
2. **InvalidOfferSearchCriteriaException** : Levée lorsque les critères de recherche des offres sont invalides ou ne correspondent à aucune offre disponible.

3. **InvalidBookingDataException** : Cette exception est déclenchée lorsqu'un booking contient des données invalides, comme des informations incorrectes sur le client ou les dates.

4. **ResourceNotFoundException** : Levée lorsque l'élément recherché (par exemple, une offre ou un booking) n'est pas trouvé dans le système.

5 Architecture et Implémentation de l'Application de l'agence client

Les agences partenaires consomment les services proposés par les hotels et proposent elles memes des services de recherche d'offres d'hotels partenaires et de booking selon une agence specifique.

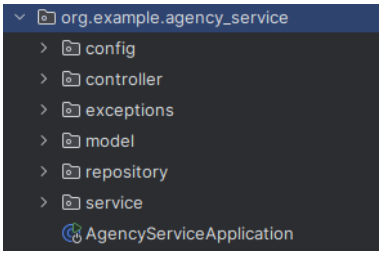

Figure 28: Structure générale serveur agence

5.1 Config

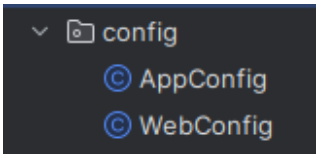

Figure 29: Config

- **AppConfig** : Configure un bean `RestTemplate` pour la communication HTTP avec des services externes.
- **WebConfig** : Configure les règles CORS pour permettre les requêtes depuis des origines spécifiques, par exemple `localhost:3000`.

5.2 Model

L'offre dans ce cas contient les memes données partagés par le offerDTO dans coté serveur.

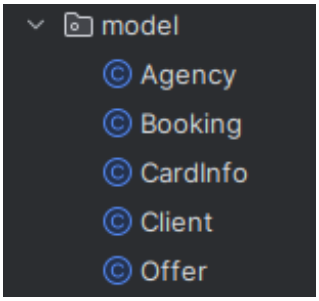

Figure 30: Model

5.3 Repository

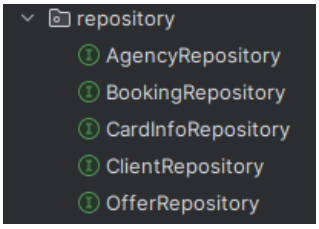

Figure 31: Repository

- **AgencyRepository:** on peut trouver les méthodes `findByLoginAndPassword` et `findById`.
- **CardInfoRepository:** on peut trouver la méthode `findByCardNumber`.

5.4 Service

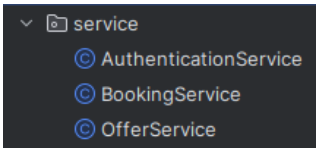

Figure 32: Service

1. **AuthenticationService**
   - `authenticateAgency` : Authentifie une agence en vérifiant ses identifiants (login et mot de passe) via la base de données des agences.
2. **OfferService**
   - `getOffersForAgency` : Recherche les offres disponibles pour une agence en interrogeant les services des hotels via des appels HTTP. Les offres respectant les critères fournis (ville, étoiles, catégorie, capacité, prix, etc.) sont sauvegardées dans la base de données locale.
3. **BookingService**

- `createBooking` : Crée une réservation pour une offre spécifique en vérifiant les disponibilités via les services des hotels, en calculant le prix total en fonction du nombre de nuits, et en sauvegardant les informations de réservation dans la base de données locale et celle de l'hotel via POST HTTP request. Elle inclut également une gestion des informations de paiement (cartes bancaires) ou les détails de la carte bancaire ne seront sauvegardés que s'ils n'existent pas déjà dans la base de données.

5.5 Controller

Dans cette API de service d'agence, chaque agence est identifiée de manière unique par son ID. Les opérations telles que la récupération des offres disponibles et la création de réservations sont effectuées en fonction de cet ID, garantissant une gestion isolée des données spécifiques à chaque agence.

Les contrôleurs suivants gèrent les opérations liées aux offres et aux réservations pour une agence donnée, en s'appuyant sur cet ID unique :

1. **AgencyController**
   - `getOffersForAgency` : Cette méthode permet de récupérer les offres disponibles pour une agence spécifique en fonction des critères donnés, en utilisant la méthode définie dans **OfferService**. Le processus inclut :
     - l'authentification de l'agence avec le login et le mot de passe via **AuthenticationService**,
     - la vérification que l'ID fourni correspond bien à celui de l'agence authentifiée,
     - la recherche des offres correspondant aux critères spécifiés (ville, étoiles, catégorie, capacité, etc.),
     - et la récupération des offres valides ou une erreur si aucune offre ne correspond.
2. **BookingController**
   - `createBooking` : Cette méthode permet de créer une réservation pour une offre spécifique, en utilisant les services de **BookingService**. Elle inclut :
     - la validation des informations du client (nom, prénom, âge, etc.),
     - la vérification des détails de la carte bancaire pour le paiement,
     - et l'enregistrement de la réservation en associant l'offre, l'agence et les informations du client.

5.6 Exception

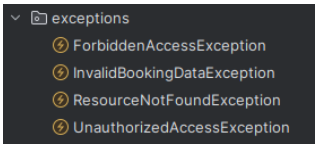

Figure 33: Exceptions

Les principales exceptions traitées dans l'API incluent :

1. **UnauthorizedAccessException** : Cette exception est levée lorsqu'une tentative d'accès est effectuée avec des identifiants d'agence invalides. Elle garantit que seules les agences authentifiées peuvent accéder aux données des offres et des réservations.
2. **ForbiddenAccessException** : Levée lorsque l'ID de l'agence fourni ne correspond pas à celui de l'agence authentifiée. Cela empêche l'accès non autorisé aux ressources d'une autre agence.
3. **ResourceNotFoundException** : Déclenchée lorsque l'élément recherché, tel qu'une offre ou une réservation, est introuvable dans le système. Par exemple, si aucune offre ne correspond aux critères de recherche ou si l'identifiant de réservation est invalide.
4. **InvalidBookingDataException** : Levée lorsqu'une tentative de création de réservation contient des données invalides. Par exemple, des informations incorrectes sur le client, des dates non valides, ou des données bancaires erronées.
5. **ServerErrorException** : Utilisée pour gérer des erreurs inattendues ou des problèmes internes du serveur. Cela inclut des exceptions non prévues ou des échecs lors du traitement des requêtes.

6 Le comparateur

L'interface utilisateur (IHM) du comparateur permet de :

- Interagir avec les services des agences pour consulter et comparer les offres.
- Personnaliser les critères de recherche selon les besoins des utilisateurs (ville, catégorie, capacité, etc.).
- Consulter les offres disponibles en fonction des critères saisis.
- Effectuer des réservations avec les détails du client et du paiement.
- Générer et télécharger un reçu de réservation en PDF.

6.1 Détails des fonctionnalités

6.1.1 Recherche d'offres

- Sélectionnez une agence ou interrogez toutes les agences disponibles.
- Définissez les critères de recherche : ville, nombre d'étoiles, catégorie, capacité, dates, et prix maximum.
- Obtenir les offres disponibles correspondant aux critères.

6.1.2 Formulaire de réservation

- Renseignez les informations du client : ID, prénom, nom, âge.
- Sélectionnez une offre et indiquez les dates de début et de fin.
- Ajoutez les détails de paiement : numéro de carte, titulaire, date d'expiration et CVV.
- Soumettez le formulaire pour créer une réservation.

6.1.3 Téléchargement de reçu en PDF

- Un fichier PDF est automatiquement généré après la réservation.
- Le fichier inclut les informations essentielles : ID de l'offre, ID client, nom, âge, dates, etc.
- Téléchargez le reçu pour confirmation ou usage ultérieur.

7 Notes et Perspectives d'Amélioration

- Améliorer l'interface homme-machine (IHM) du comparateur en y intégrant un système de **routage** pour une navigation plus fluide et intuitive.
- Développer un **backend** ainsi qu'une **API** dédiée pour le comparateur, afin de faciliter les intégrations et améliorer les performances.
- Ajouter un **QR code** au fichier de réservation pour simplifier l'accès et le partage des informations.
- Enrichir la **fenêtre de confirmation de réservation** ainsi que le fichier associé avec davantage de données pertinentes pour l'utilisateur.