

COMPTE-RENDU DE TP REST

Conception de la solution :



















3 Modules :

- HotelREST est une pur API REST dans le sens où elle fournit des services mais n'en consomme pas. Elle fournit pour chaque entité certaines opérations CRUD ainsi que des requêtes pour avoir la liste des offres disponibles par exemple.


Structure :





Liste des endpoint :


HotelREST		
 /booking [GET]	HTTP Server Spring MVC Controllers	RoomController
 /booking [POST]	HTTP Server Spring MVC Controllers	RoomController
 /booking/{id} [GET]	HTTP Server Spring MVC Controllers	RoomController
 /customer [GET]	HTTP Server Spring MVC Controllers	RoomController
 /customer [POST]	HTTP Server Spring MVC Controllers	RoomController
 /customer/signup/{name}{surname}{card} [POST]	HTTP Server Spring MVC Controllers	RoomController
 /customer/{id} [GET]	HTTP Server Spring MVC Controllers	RoomController
 /hotel [GET]	HTTP Server Spring MVC Controllers	RoomController
 /hotel/{id} [GET]	HTTP Server Spring MVC Controllers	RoomController
 /hotel/{id}/partner [GET]	HTTP Server Spring MVC Controllers	RoomController
 /hotel/{id}/room [GET]	HTTP Server Spring MVC Controllers	RoomController
 /hotel/{id}/{agency}{password}-from={start},to={end},minPrice={min},maxPrice={max},nbBeds={nbBeds} [GET]	HTTP Server Spring MVC Controllers	HotelController
 /partner [GET]	HTTP Server Spring MVC Controllers	RoomController
 /partner/{id} [GET]	HTTP Server Spring MVC Controllers	RoomController
 /room [GET]	HTTP Server Spring MVC Controllers	RoomController
 /room/{id} [GET]	HTTP Server Spring MVC Controllers	RoomController
 /room/{id}/booking [GET]	HTTP Server Spring MVC Controllers	RoomController
 /room/{roomId}/customer={customerId}/start={start}/end={end} [POST]	HTTP Server Spring MVC Controllers	RoomController


Vue sur la BDD :


 jdbc:h2:mem:hotel


 BOOKING


 CUSTOMER


 HOTEL


 PARTNER

 ROOM

 INFORMATION_SCHEMA

 Sequences

 Users

 H2 1.4.200 (2019-10-14)

RunRun SelectedAuto completeClearSQL statement:

SELECT * FROM HOTEL

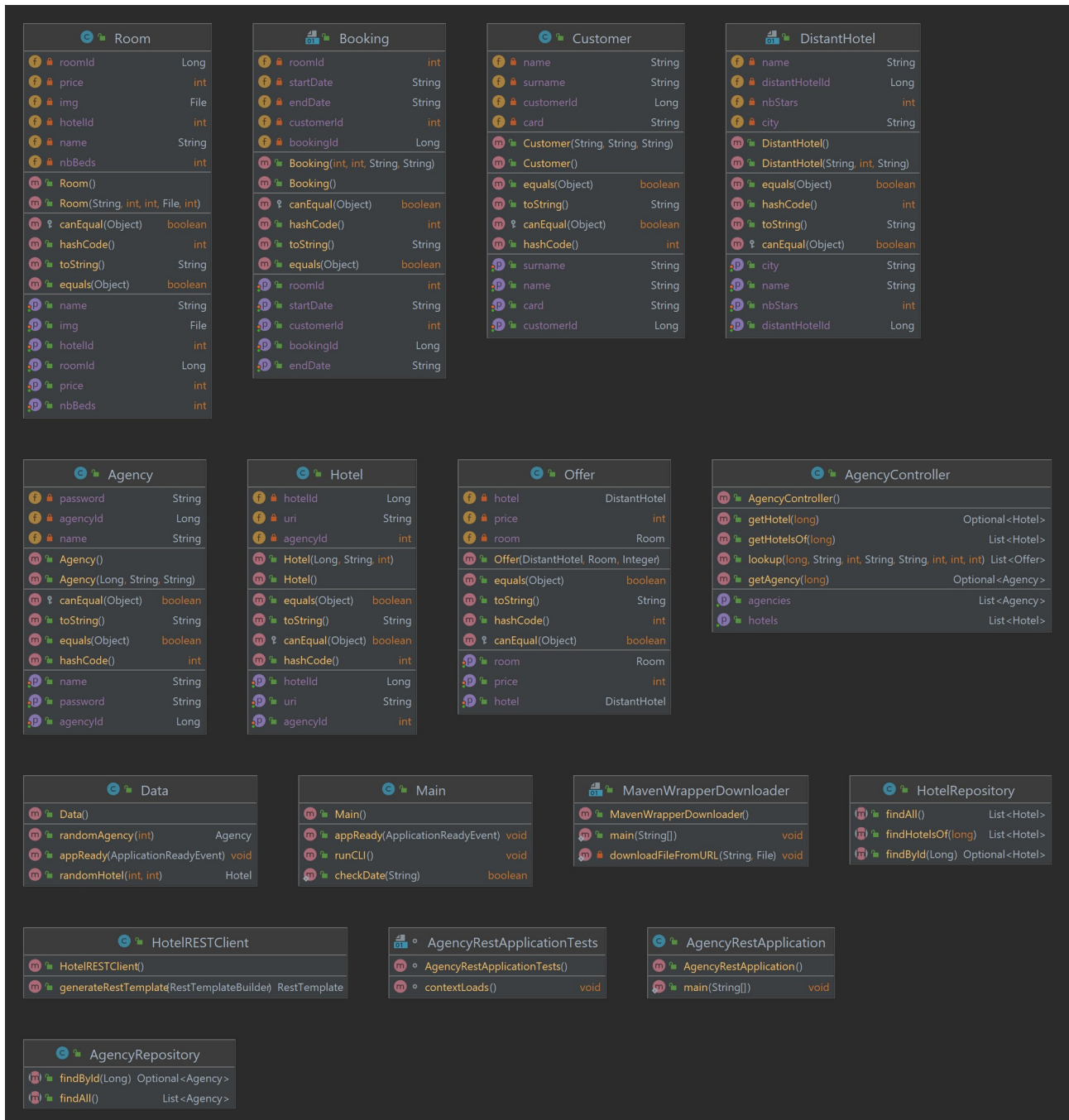
SELECT * FROM HOTEL;

HOTEL_ID	CITY	NAME	NB_STARS
1	Lyon	Hotel 1	4
2	Lyon	Hotel 2	3
3	Nice	Hotel 3	4
4	Nice	Hotel 4	2
5	Nice	Hotel 5	2
6	Montpellier	Hotel 6	4
7	Montpellier	Hotel 7	5
8	Paris	Hotel 8	3
9	Nice	Hotel 9	2
10	Paris	Hotel 10	3
11	Montpellier	Hotel 11	1
12	Paris	Hotel 12	3
13	Marseille	Hotel 13	5
14	Marseille	Hotel 14	5
15	Marseille	Hotel 15	3
16	Paris	Hotel 16	1
17	Montpellier	Hotel 17	3
18	Lyon	Hotel 18	3
19	Nice	Hotel 19	4
20	Montpellier	Hotel 20	4

(20 rows, 1 ms)

- AgencyREST est une API REST qui consomme les services de HotelREST afin d'offrir ses propres services. L'agence peut demander aux hôtels leurs chambres disponibles avec des prix préférentiels si ils sont partenaires afin de proposer au client utilisant sa CLI ou le module Trivago de réserver des chambres.

Structure :



Liste des endpoint :

AgencyREST	
/agency [GET] HTTP Server Spring MVC Controllers	AgencyController
/agency/{id} [GET] HTTP Server Spring MVC Controllers	AgencyController
/agency/{id}/hotel [GET] HTTP Server Spring MVC Controllers	AgencyController
/agency/{id}/lookup/city={city},nbStars={nbStars},from={start},to={end},minPrice={min},maxPrice={max},nbBeds={nbBeds} [GET] HTTP Server Spring MVC Controllers	AgencyController
/hotel [GET] HTTP Server Spring MVC Controllers	AgencyController
/hotel/{id} [GET] HTTP Server Spring MVC Controllers	AgencyController

Vue sur la BDD :

jdbc:h2:mem:agency

+

AGENCY

+

HOTEL

+

INFORMATION_SCHEMA

+

Sequences

+

Users

i

H2 1.4.200 (2019-10-14)

Run

Run Selected

Auto complete

Clear

SQL statement:

SELECT * FROM HOTEL

SELECT * FROM HOTEL;

HOTEL_ID ▼	AGENCY_ID	URI
1	1	http://localhost:8080/hotel/1
2	1	http://localhost:8080/hotel/2
3	1	http://localhost:8080/hotel/3
4	1	http://localhost:8080/hotel/4
5	1	http://localhost:8080/hotel/5
6	1	http://localhost:8080/hotel/6
7	1	http://localhost:8080/hotel/7
8	1	http://localhost:8080/hotel/8
9	1	http://localhost:8080/hotel/9
10	1	http://localhost:8080/hotel/10
11	1	http://localhost:8080/hotel/11
12	1	http://localhost:8080/hotel/12
13	1	http://localhost:8080/hotel/13
14	1	http://localhost:8080/hotel/14
15	1	http://localhost:8080/hotel/15
16	1	http://localhost:8080/hotel/16
17	1	http://localhost:8080/hotel/17
18	1	http://localhost:8080/hotel/18
19	1	http://localhost:8080/hotel/19
20	1	http://localhost:8080/hotel/20
21	2	http://localhost:8080/hotel/1
22	2	http://localhost:8080/hotel/2

- Module Trivago : module qui consomme les services de AgencyREST pour comparer les offres de toutes les agences.

Room <ul style="list-style-type: none"> roomId: Long img: File hotelId: int price: int nbBeds: int name: String Room(String, int, int, File, int) Room() hashCode(): int toString(): String equals(Object): boolean canEqual(Object): boolean name: String img: File price: int nbBeds: int hotelId: int roomId: Long 	Booking <ul style="list-style-type: none"> customerId: int startDate: String roomId: int bookingId: Long endDate: String Booking() Booking(int, int, String, String) canEqual(Object): boolean equals(Object): boolean toString(): String hashCode(): int startDate: String bookingId: Long customerId: int roomId: int endDate: String 	Customer <ul style="list-style-type: none"> customerId: Long surname: String name: String card: String Customer() Customer(String, String, String) hashCode(): int canEqual(Object): boolean toString(): String equals(Object): boolean surname: String name: String card: String customerId: Long 	DistantHotel <ul style="list-style-type: none"> distantHotelId: Long name: String nbStars: int city: String DistantHotel() DistantHotel(String, int, String) equals(Object): boolean canEqual(Object): boolean hashCode(): int toString(): String city: String name: String nbStars: int distantHotelId: Long
Offer <ul style="list-style-type: none"> price: int hotel: DistantHotel room: Room distantAgency: DistantAgency Offer(DistantHotel, Room, Integer) toString(): String hashCode(): int equals(Object): boolean room: Room distantAgency: DistantAgency hotel: DistantHotel price: int 	DistantAgency <ul style="list-style-type: none"> agencyId: Long name: String password: String DistantAgency() DistantAgency(String, String) hashCode(): int canEqual(Object): boolean equals(Object): boolean toString(): String name: String password: String agencyId: Long 	Agency <ul style="list-style-type: none"> agencyId: Long uri: String Agency(Long, String) Agency() canEqual(Object): boolean hashCode(): int equals(Object): boolean toString(): String uri: String agencyId: Long 	Main <ul style="list-style-type: none"> Main() runCLI(String[]): void checkDate(String): boolean appReady(ApplicationReadyEvent): void
MavenWrapperDownloader <ul style="list-style-type: none"> MavenWrapperDownloader() main(String[]): void downloadFileFromURL(String, File): void 	AgencyRepository <ul style="list-style-type: none"> findById(Long): Optional<Agency> findAll(): List<Agency> findByUri(String): Optional<Agency> 	AgencyRESTClient <ul style="list-style-type: none"> AgencyRESTClient() generateRestTemplate(RestTemplateBuilder): RestTemplate 	
TrivagoApplicationTests <ul style="list-style-type: none"> TrivagoApplicationTests() contextLoads(): void 	TrivagoApplication <ul style="list-style-type: none"> TrivagoApplication() main(String[]): void 		

La BDD AgencyRepository sert seulement à ajouter les agences auxquelles on veut se connecter et simplifie l'implémentation grâce aux méthodes fournies par Spring Data JPA.