Sure, let's go through this data flow again with the new requirement:

1. **Receiving HotelSearchRequest**: The server receives a request from a client. You would use the HotelSearchRequest model for this stage.

json

{

"hotelIds": ["MCLONMAR", "MCLONGHM"],

"checkInDate": "2023-11-20",

"checkOutDate": "2023-11-23",

"roomQuantity": 1

}

1. **Generating Date Combinations**: The server generates all possible date combinations between the check-in and check-out dates. You would use the DateCombination model for this stage.

json

[

["2023-11-20", "2023-11-21"],

["2023-11-20", "2023-11-22"],

["2023-11-20", "2023-11-23"],

["2023-11-21", "2023-11-22"],

["2023-11-21", "2023-11-23"],

["2023-11-22", "2023-11-23"]

]

1. **Iterating Over Combinations & SearchAmadeusHotels Execution**: For each date combination, the server makes a request to the Amadeus API. This stage doesn't directly involve a model, but the HotelSearchRequest model is used to construct the API URL.
2. **Processing API Response**: The server receives the response from the Amadeus API for each request. For this stage, you would use the HotelOffersResponse model to deserialize the API response.

json

{

"data": [

{

"type": "hotel-offers",

"hotel": {

"type": "hotel",

"hotelId": "MCLONGHM",

"name": "JW Marriott Grosvenor House London",

// Additional hotel details...

},

"available": true,

"offers": [

{

"id": "ZSFJXY7N3W",

"checkInDate": "2023-11-20",

"checkOutDate": "2023-11-21",

// Additional offer details...

}

]

},

// Additional hotel-offers...

]

}

1. **Grouping Hotel Offers**: At this stage, you take each HotelOffersResponse, create HotelOffer instances for each hotel, and group them by the hotelId.

json

{

"MCLONGHM": [

{

"hotel": {

"hotelId": "MCLONGHM",

"name": "JW Marriott Grosvenor House London",

// Other hotel details...

},

"offers": [

{

// Offer details...

},

// More offers...

]

},

// More HotelOffer objects for MCLONGHM...

],

"MCLONMAR": [

// Array of HotelOffer objects for MCLONMAR...

],

// More groups of HotelOffer objects...

}

1. **Generating CombinationOffers and Sorting**: For each hotel group, you generate all combinations of offers that cover the full date range. Each of these becomes a CombinationOffers object. After generating the combinations, you sort them by the total cost, ascending.

json

{

"combinations": [

{

"dates": ["2023-11-20", "2023-11-23"],

"offers": [ /\* offers from the response \*/ ],

"totalPrice": "1000.00"

},

// More combinations...

]

}

1. **Response Sending**: Once all date combinations have been processed and sorted by the total cost, the server sends the final HotelSearchResponse to the client.

json

{

"combinations": [

{

"dates": ["2023-11-20", "2023-11-23"],

"offers": [ /\* cheapest offers for these dates \*/ ],

"totalPrice": "1000.00"

},

{

"dates": ["2023-11-20", "2023-11-22"],

"offers": [ /\* next cheapest offers for these dates \*/ ],

"totalPrice": "1200.00"

},

// More combinations...

]

}

In this modified approach, after each hotel's offers are grouped together and combinations are created within these groups, the combinations are sorted by their total cost. This ensures that the client receives the cheapest combinations of offers first.

1. **Receiving HotelSearchRequest**: The server receives a request from a client. You would use the HotelSearchRequest model for this stage.

json

{

"hotelIds": ["MCLONMAR", "MCLONGHM"],

"checkInDate": "2023-11-20",

"checkOutDate": "2023-11-23",

"roomQuantity": 1

}

1. **Generating Date Combinations**: The server generates all possible date combinations within the range between the check-in and check-out dates, with each combination containing two dates (a start and an end date). You would use the DateCombination model for this stage.

json

[

["2023-11-20", "2023-11-21"],

["2023-11-20", "2023-11-22"],

["2023-11-20", "2023-11-23"],

["2023-11-21", "2023-11-22"],

["2023-11-21", "2023-11-23"],

["2023-11-22", "2023-11-23"]

]

1. **Iterating Over Combinations & SearchAmadeusHotels Execution**: For each date combination, the server makes a request to the Amadeus API. This stage doesn't directly involve a model, but the HotelSearchRequest model is used to construct the API URL.
2. **Processing API Response**: The server receives the response from the Amadeus API for each request. For this stage, you would use the HotelOffersResponse model to deserialize the API response.

json

{

"data": [

{

"type": "hotel-offers",

"hotel": {

"type": "hotel",

"hotelId": "MCLONGHM",

"name": "JW Marriott Grosvenor House London",

// Additional hotel details...

},

"available": true,

"offers": [

{

"id": "ZSFJXY7N3W",

"checkInDate": "2023-11-20",

"checkOutDate": "2023-11-21",

// Additional offer details...

}

]

},

// Additional hotel-offers...

]

}

1. **Grouping Hotel Offers**: The server takes each HotelOffersResponse, creates HotelOffer instances for each hotel, and groups them by the hotelId.

json

{

"MCLONGHM": [

{

"hotel": {

"hotelId": "MCLONGHM",

"name": "JW Marriott Grosvenor House London",

// Other hotel details...

},

"offers": [

{

// Offer details...

},

// More offers...

]

},

// More HotelOffer objects for MCLONGHM...

],

"MCLONMAR": [

// Array of HotelOffer objects for MCLONMAR...

],

// More groups of HotelOffer objects...

}

1. **Generating CombinationOffers**: For each hotel group, the server generates all combinations of offers that cover the entire date range provided by the user. Each of these becomes a CombinationOffers object.
2. **Sorting CombinationOffers**: After generating the combinations, the server sorts them by their total cost, ascending.

json

{

"combinations": [

{

"dates": ["2023-11-20", "2023-11-23"],

"offers": [

{

"id": "OFFER1",

"checkInDate": "2023-11-20",

"checkOutDate": "2023-11-22",

"price": "500.00"

// other details...

},

{

"id": "OFFER2",

"checkInDate": "2023-11-22",

"checkOutDate": "2023-11-23",

"price": "200.00"

// other details...

}

],

"totalPrice": "700.00"

},

// More combinations...

]

}

1. **Response Sending**: Once all date combinations have been processed and sorted by total cost, the server sends the final HotelSearchResponse to the client. This response includes the sorted combinations of offers that cover the entire date range, sorted by total cost. The client receives the combinations with the cheapest offers first.

Based on the data you've provided, below are the model classes and the data flow that would suit your requirements:

**1. Models:**

a. DateCombination Model: This model is used to hold the combinations of check-in and check-out dates.

csharp

public class DateCombination

{

public DateTime CheckInDate { get; set; }

public DateTime CheckOutDate { get; set; }

}

b. Hotel Model: This model is used to hold the hotel information.

csharp

public class Hotel

{

public string HotelId { get; set; }

public string Name { get; set; }

// Other hotel details...

}

c. HotelOffer Model: This model is used to hold each offer from the hotel.

csharp

public class HotelOffer

{

public string Id { get; set; }

public DateTime CheckInDate { get; set; }

public DateTime CheckOutDate { get; set; }

public decimal Price { get; set; }

// Other offer details...

}

d. CombinationOffer Model: This model is used to hold combinations of hotel offers that cover the entire date range.

csharp

public class CombinationOffer

{

public List<DateTime> Dates { get; set; }

public List<HotelOffer> Offers { get; set; }

public decimal TotalPrice { get; set; }

}

e. HotelSearchResponse Model: This model is used to hold the response that will be sent back to the client.

csharp

public class HotelSearchResponse

{

public List<CombinationOffer> Combinations { get; set; }

}

**2. Data Flow:**

a. Server receives the HotelSearchRequest from the client. b. Server generates all possible DateCombinations within the range between the check-in and check-out dates. c. For each DateCombination, the server makes a request to the Amadeus API using the HotelSearchRequest to construct the API URL. d. Server receives the response from the Amadeus API for each request and uses the HotelOffersResponse model to deserialize the API response. e. Server takes each HotelOffersResponse, creates HotelOffer instances for each offer, and groups them by the hotelId. f. For each hotel group, the server generates all combinations of offers (creating a CombinationOffer for each) that cover the entire date range provided by the client. g. After generating the combinations, the server sorts them by their total cost, in ascending order. h. Once all date combinations have been processed and sorted by total cost, the server creates a HotelSearchResponse containing the sorted combinations of offers, and sends this to the client. The client receives the combinations with the cheapest offers first.

{ "data": [ { "type": "hotel-offers", "hotel": { "type": "hotel", "hotelId": "MCLONGHM", "chainCode": "MC", "dupeId": "700031300", "name": "JW Marriott Grosvenor House London", "cityCode": "LON", "latitude": 51.50988, "longitude": -0.15509 }, "available": true, "offers": [ { "id": "0APLIQQYNR", "checkInDate": "2023-11-20", "checkOutDate": "2023-11-23", "rateCode": "RAC", "rateFamilyEstimated": { "code": "PRO", "type": "P" }, "room": { "type": "AP7", "typeEstimated": { "category": "EXECUTIVE\_ROOM", "beds": 1, "bedType": "DOUBLE" }, "description": { "text": "Prepay Non-refundable Non-changeable, prepay in full\nExecutive King Room, Executive Lounge Access,\n1 King, 35sqm/377sqft-40sqm/430sqft, Wireless", "lang": "EN" } }, "guests": { "adults": 1 }, "price": { "currency": "GBP", "base": "2164.00", "total": "2164.00", "variations": { "average": { "base": "721.33" }, "changes": [ { "startDate": "2023-11-20", "endDate": "2023-11-21", "total": "700.00" }, { "startDate": "2023-11-21", "endDate": "2023-11-22", "total": "740.00" }, { "startDate": "2023-11-22", "endDate": "2023-11-23", "total": "724.00" } ] } }, "policies": { "cancellations": [ { "description": { "text": "NON-REFUNDABLE RATE" }, "type": "FULL\_STAY" } ], "paymentType": "deposit" }, "self": "https://test.api.amadeus.com/v3/shopping/hotel-offers/0APLIQQYNR?lang=ENG" } ], "self": "https://test.api.amadeus.com/v3/shopping/hotel-offers?hotelIds=MCLONGHM&adults=1&boardType=ROOM\_ONLY&checkInDate=2023-11-20&checkOutDate=2023-11-23&countryOfResidence=US&currency=USD&lang=ENG&paymentPolicy=NONE&roomQuantity=1" }, { "type": "hotel-offers", "hotel": { "type": "hotel", "hotelId": "MCLONMAR", "chainCode": "MC", "dupeId": "700009923", "name": "LONDON MARRIOTT MARBLE ARCH", "cityCode": "LON", "latitude": 51.51587, "longitude": -0.16413 }, "available": true, "offers": [ { "id": "58EMCG8T3R", "checkInDate": "2023-11-20", "checkOutDate": "2023-11-23", "rateCode": "S9R", "rateFamilyEstimated": { "code": "SRS", "type": "C" }, "room": { "type": "XMI", "typeEstimated": { "category": "DELUXE\_ROOM", "beds": 1, "bedType": "KING" }, "description": { "text": "Marriott Senior Discount, 62 years and older valid ID required\nDeluxe Room, 1 King, 22sqm/237sqft,\nLiving/sitting area, Wireless internet, for a", "lang": "EN" } }, "guests": { "adults": 1 }, "price": { "currency": "GBP", "base": "1173.00", "total": "1173.00", "variations": { "average": { "base": "391.00" }, "changes": [ { "startDate": "2023-11-20", "endDate": "2023-11-21", "total": "375.00" }, { "startDate": "2023-11-21", "endDate": "2023-11-23", "total": "399.00" } ] } }, "policies": { "cancellations": [ { "amount": "375.00", "deadline": "2023-11-19T23:59:00+00:00" } ], "paymentType": "guarantee" }, "self": "https://test.api.amadeus.com/v3/shopping/hotel-offers/58EMCG8T3R?lang=ENG" } ], "self": "https://test.api.amadeus.com/v3/shopping/hotel-offers?hotelIds=MCLONMAR&adults=1&boardType=ROOM\_ONLY&checkInDate=2023-11-20&checkOutDate=2023-11-23&countryOfResidence=US&currency=USD&lang=ENG&paymentPolicy=NONE&roomQuantity=1" } ], "dictionaries": { "currencyConversionLookupRates": { "GBP": { "rate": "1.289787", "target": "USD", "targetDecimalPlaces": 2 } } } }