Ausgangssituation

Grundsätzliche Beschreibung

Zur Vermittlung von Ferienwohnungen ist ein zentrales System zur Abwicklung des Reservierungsprozesses zu entwickeln (`Backend`). Das Backend muss mit dem `Spring Framework` umgesetzt werden. Die zu verwendenden Bibliotheken sind dem Projekt bereits hinzugefügt.

Detailbeschreibung

Kunden können Ferienwohnungen (z.B. “Ferienwohnung Relax”, “Ferienwohnung Romantik”, “Ferienwohnung Maturareise”, …), die jeweils unterschiedlichen Kategorien (z.B. “Komfort-Wohnungen”, “Romantik-Wohnungen”, “Panorama-Wohnungen”, “Party-Wohnungen” …) zugewiesen sind, für einen gewissen Zeitraum mieten. Zum Mieten einer Ferienwohnung muss der gewünschte An- sowie Abreisetermin bekanntgegeben werden. Jeder Kunde kann im selben Zeitraum (oder an sich überschneidenden Zeiträumen) mehrere Ferienwohnungen gleichzeitig mieten. Jede Ferienwohnung ist zu jedem Zeitpunkt maximal einem Kunden zugeordnet!

> ***Beispiel:*** *Herr Mustermann möchte zum Feiern seiner erfolgreich abgelegten Matura eine Ferienwohnung für sich und seine Klassenkamerad:innen buchen. In Summe benötigt er für 21 Personen Schlafplätze. Er bucht eine Ferienwohnung für sechs Personen und drei Ferienwohnungen für je fünf Personen. Die Anreise erfolgt am 08. Juni 2023 und die Abreise am 15.06.2023 (7 Tage).*

Die entstehenden Kosten setzen sich aus den Tagessätzen der gebuchten Ferienwohnungen sowie den gebuchten Versorgungspaketen zusammen. Es gibt grundsätzlich vier unterschiedliche Versorgungspakete (Preis je Tag):

- “OHNE Versorgung” (kein Aufpreis)
- “Frühstück” (Aufpreis 11€)
- “Halbpension” (Aufpreis 25€)
- “Vollpension” (Aufpreis 40€)

Folgende Punkte müssen beachtet werden:

- Die Kosten je Ferienwohnung sind variabel und für jede Ferienwohnung getrennt zu speichern!
- Diese vier obenstehenden Versorgungspakete sind einstweilen geplant. Damit Preisänderungen sowie das Hinzufügen von weiteren Paketen auch nachträglich möglich ist, sollen diese Pakete in der Datenbank gespeichert werden.

> ***Beispiel:*** *Die von Herrn Mustermann gebuchten Unterkünfte für fünf Personen kosten 150€ je Tag. Die Unterkunft für sechs Personen schlägt sich mit 170€ je Tag zu Buche. Zehn Personen nehmen das “Vollpension” - Versorgungspaket. Weitere zehn Personen nehmen das Versorgungspaket “Halbpension”. Für eine Person wird das “Frühstück” - Versorgungspaket gebucht. Der Endbetrag den Herr Mustermanna also bezahlen muss ist: (170 € * 1 Wohnung * 7 Tage) + (150 € * 3 Wohnungen * 7 Tage) + (10 Personen * 40 € Vollpension * 7 Tage) + (10 Personen * 25 € Halbpension * 7 Tage) + (1 Person * 11 € Frühstück * 7 Tage) = 8967 €*

Aufgabenstellung:

Als Ausgangspunkt für Ihre Umsetzung kann die Vorlage vom gegebenen GitHub - Account heruntergeladen werden.
Für die Speicherung der Daten wird eine MariaDB - Datenbank verwendet. Das Frontend wurde bereits umgesetzt und kann zum Testen der Funktionalität Ihres Backends verwendet werden. Die Zugangsdaten für den GitHub - Account, für die MariaDB - Datenbank sowie Infos zur Verwendung des gegebenen Frontends entnehmen Sie dem beigelegten Zettel.

Ergänzen Sie für den gegebenen Sachverhalt das bereitgestellte GitHub - Projekt. Arbeiten Sie mit dem Spring - Framework und den vorgegebenen Abhängigkeiten (siehe `pom.xml` - Datei).

Ferienwohnung (`Apartment - Entity`)

Eine Ferienwohnung wird durch ihren Namen, die maximale Bewohneranzahl sowie den Grundpreis je Tag beschrieben. Jede Ferienwohnung ist genau einer Kategorie zugewiesen. Eine Ferienwohnung kann mehrmals gebucht werden.

Buchung (`Booking - Entity`)

Eine Buchung ist einem Kunden zugeordnet. Die Buchung gilt von einem Start- bis zu einem Enddatum. Mit einer Buchung können mehrere Ferienwohnungen reserviert werden. Zudem können einer Buchung mehrere Pakete mit einer gewissen Anzahl zugeordnet sein.

Paket (`SupplementaryPackage - Entity`)

Ein Paket besitzt einen Namen sowie einen Preis. Ein Paket kann bei mehreren Buchungen zugeordnet sein.

Zu Erledigende Punkte

1. Dokumentieren Sie den Verlauf Ihrer Arbeit, indem Sie Ihre Änderungen regelmäßig im gegebenen GitHub - Repository veröffentlichen.
2. Richten Sie die Spring Anwendung für Kommunikation mit der zur Verfügung gestellten Datenbank ein.
3. Erstellen Sie ein ER - Diagramm mit draw.io (https://app.diagrams.net). In diesem Diagramm halten Sie alle Entitäten (auch die vorgegebenen), deren Eigenschaften (ohne Datentypen) sowie die Beziehungen zueinander fest. Fügen Sie auch das Diagramm dem Repository hinzu.
4. Setzen Sie die obenstehenden Entitäten, deren Eigenschaften und Beziehungen zueinander in Form von Java Klassen im vorgegebenen GitHub - Projekt um.
    - Die Klassen sollen über entsprechende Repositories mit der Datenbank kommunizieren.
    - Verwenden Sie für ***alle*** Primärschlüssel ausschließlich numerische, selbsterhöhende Werte (`INT, AUTO INCREMENT`).
    - Sollten Many-to-Many Beziehungen notwendig sein, überlegen Sie sich genau, ob eine sogenannte “Zwischen-Entity” notwendig ist.
    - Verwenden Sie für die Erstellung der Getter- und Settermethoden die Funktionalitäten der Lombok - Bibliothek.
    - Erweitern Sie die `Booking` Klasse um die Methode `public double calculateTotalPrice()`, welche den Gesamtpreis der Buchung berechnet und zurückgibt.
5. Setzen Sie die fehlenden Entitäten in Form von Relationen in der Datenbank um.

6. Ergänzen Sie die gegebenen JUnit - Testklassen und um den Test der `public double calculateTotalPrice()` - Methode der `Booking` Klasse.
7. Erweitern Sie die vorhandene GitHub - Action, sodass der JUnit - Test automatisch mit jedem `push` ausgeführt wird.
8. Erstellen Sie eine REST Schnittstelle, in der Sie folgende Endpunkte umsetzen:
   - GET Request, URL: `/apartment/available/{start date}/{end date}`
     Beispiel: `/apartment/available/2023-06-08/2023-06-15`
     Liefert eine Liste aller im gesamten Zeitraum verfügbaren Ferienwohnungen.
   - POST Request, URL: `/booking/create`
     POST - Body:

```
{
    customer id: 1,
    apartments: [
        {
            apartment_id: 1,
            date_from: '2023-06-08',
            date_to: '2023-06-15'
        },
        {
            apartment_id: 2,
            date_from: '2023-06-08',
            date_to: '2023-06-15'
        },
        {
            apartment_id: 3,
            date_from: '2023-06-08',
            date_to: '2023-06-15'
        },
        {
            apartment_id: 4,
            date_from: '2023-06-08',
            date_to: '2023-06-15'
        }
    ],
    packages: [
        {
            package_id: 4,
            amount: 10,
            date_from: '2023-06-08',
            date_to: '2023-06-15'
        },
        {
            package_id: 3,
            amount: 10,
            date_from: '2023-06-08',
            date_to: '2023-06-15'
        },
        {
            package_id: 2,
            amount: 1,
            date_from: '2023-06-08',
            date_to: '2023-06-15'
        }
    ]
}
```

Theoriefragen:

1. Erklären Sie, was man unter einer „Race – Condition“ versteht und wie diese verhindert werden kann.

2. Was versteht man unter dem „Observable – Pattern“ (auch als „Publisher – Subscriber – Pattern“ bekannt)?