Variablen Dokumentation

Person

Variablenname	Datentyp	Atribut	Bescreibung
persnId	int	Primärschlußel	Identifizierungsnummer
firstName	String	Not *Null*	Vorname
lastName	String	Not *Null*	Familienname
adress	String	/	Adresse
phoneNumber	String	/	Telefon oder Handy Nummer
eMail	String	Not *Null*	E-Mail Adresse

Administrator / Employee

Variablenname	Datentyp	Atribut	Bescreibung
position	String	Not Null	Stelle/Position im Unternehmern
salary	String	Not Null	Gehalt Brutto
bonus	boolean	true/false	Prämie
admin	boolean	true/false	Recht als Administrator

Customer

Variablenname	Datentyp	Atribut	Bescreibung
NumberOfCustomers	int	Not Null	Anzahl an zusätzliche Kunden
checkIn	LocalDate	Not Null	CheckIn Datum
checkOut	LocalDate	Not Null	CheckOut Datum

Reservation

Variablenname	Datentyp	Atribut	Bescreibung
reservationId	int	Primerschlüßel	Buchungsnummer
contactInfo	String	Not Null	E-Mail vom Kunden
occupant	int	Not Null	Anzahl an Kunde in der Buchung
maxNumberOfResidents	int	Not Null	Maximale Anzahl an Schlafplätzen in der Buchung
isComplete	Boolean	Not Null	Nur „true“ wenn der Kunde buchen darf
checkIn	LocalDate	Not Null	Check In Datum (Vom Kunden)
checkOut	LocalDate	Not Null	Check Out Datum (Vom Kunden)
reservationsDate	LocalDate	Not Null	Buchungsdatum
finalPrice	double	Not Null	Gesamt Preis der Buchung

Room

SingleRoom, DoubleRoom, FamilyRoom, Suite vererben von room

Variablenname	Datentyp	Atribut	Bescreibung
roomId	int	Primärschlußel	Zimmer Identifizierungs-Nummer
isAvailable	boolean	true/false	Zimmer Verfügbarkeit
typeOfBed	Enum	Not Null	Art an Betten
pricePerNight	double	Not Null	Preis pro Nacht
breakfast	boolean	true/false	Ob der Kunde zusätliche Frühhstuck möchte
roomNumber	Int	Not Null	Zimmernummer
maxNumberOfResidents	Int	Not Null	Anzahl an Schlafplätzen pro Zimmer
extra	String	/	Wünsche des Kunden

Methode Dokumentation

Format

Methodename(parameter) -> Bescreibung der Methode

Localvariablen werden in Grün geschrieben

Returns werden in Blau geschrieben

ADMINISTRATORS

post → fügt einen/e Administrator/in in die Datenbank hinzu.

get (int id) → holt einen/e Administrator/in von der Datenbank.

return administrator

getAllAdministrators → Erzeugt eine List mit allen Administratoren.

return List<Administrator>

put → Aktualisiert die Daten von einem/er Administrator/in.

delete (int id) → Löscht einen/e Administrator/in aus der Datenbank.

EMPLOYEES

post → fügt einen/e Mitarbeiter/in in die Datenbank hinzu.

get (int id) → holt einen/e Mitarbeiter/in züruck.

return employee

getAllEmployees → Erzstellt eine List mit allen Mitarbeiter.

return List<Employee>

put → Aktualisiert die Daten von einem/er Mitarbeiter/in (id erforderlich).

Delete (int id) → Löscht einen/e Mitarbeiter/in aus der Datenbank.

getEmployeeSalary → Zeigt das Gehalt von den Mitarbeiter.

return employee.getSalary()

getEmployeeKeyInformation → erstellt eine Liste mit den gewünschten Informationen der Mitarbeiter mit Hilfe eines DTOs.

Return List<Employee>

convertToEmployeeKeyInformation → speichert die gewünschte Information in einem DTO und schickt diesen DTO zu *getEmployeeKeyInformation*.

COSTUMER

post → fügt einen Kunde in die Datenbank hinzu.

get (int id) → Holt ein Kunde aus der Datenbank-

return employee

getAllCostumers → schickt eine List von allen Kunden zurück.

return List<Employee>

put → aktualisiert die Daten eines Kunden (id erförderlich).

delete (int id) → Löscht einen Kunde aus der Datenbank.

getCostumerKeyInformation → Erstellt eine Liste mit den gewünschten Informationen der Kunden mit Hilfe eines DTOs.

return List<Employee>

convertToCostumerKeyInformation → Speichert die gewünschte Information in einem DTO und schickt diesen DTO zurück.

return employeeInfo

RESERVATION

postWithCustomerInfo → erstellt neue Buchung mit den Information von Kunde (Kunden Id erforderlich)

getById (reservationId) → Holt eine Buchung aus der Datenbank

return reservation;

getAllReservation → erzeugt eine Liste aller Buchungen.

return List<Reservation>

getSummary (reservationId)→ erstellt eine detaillierte Übersicht einer Buchung

return reservationSummary;

getAllReservationsStatus → erzeugt eine Liste von Buchungen mit den gewünschten Daten.

convertToReservationStatus → Der Preis von jeder Buchung wird um 10% erhöht, falls Hochseason ist und der gesamte Preis der Buchung wird angepasst.

return reservationStatus;

updateReservation (double roomPrice, int numberOfResidents)→ Der Status der Buchung wird mit den neuen Daten überschrieben

long durationOfStay; Tage zwichen checkIn und checkOut.

calculateReturnPrice → Beim Stornieren wird der Preis nach den Stornierungsrichtlinien berechnte und zurückgeschickt

double returnPrice; Beim Stornieren das Geld, das zurück bezahlt wird.

LocalDate firstDate; spielt die Rolle vom Variablen checkIn.

LocalDate secondDate; spielt die Rolle vom Variablen LocalDate(now())

LocalDate thirdDate; spielt die Romme vom Variablen reservationDate.

duration; Tage zwischen checkin und LocalDate(now())

duration2; Tage zwichen checkIn und reservationDate

duration3; Tage LocalDate(now()) und reservationDate

return returnPrice;

daysBetweenTwoDates → Berechnet die Tage zwischen 2 Daten

LocalDate firstDate;

LocalDate secondDate;

cancelReservation (reservationId) → storniert eine Buchung und mit Hilfe der Methode *calculateReturnPrice* wird das Geld, das zurückgezahlt muss, berechnet.

return calculateReturnPrice;

cancerRoomFromReservation(reservationId , roomId) → storiniert ein Zimmer von einer Buchung und mit Hilfe der Methode *calculateReturnPrice* wird der Preis von der Buchung abgezogen.

Double priceToBeReturn;

return priceToBeReturn;

delete(reservationId)→ löscht eine Buchung aus der Datenbank

SingleRoom / DoubleRoom / FamilyRoom / Suite → Room

post → Fügt ein Zimmer in die Datenbank hinzu

get(int id) → Schickt ein Zimmer zurückt

getAllAvailableRooms(int kundeId) → Erzeugt eine Liste von allen verfügbaren Zimmern. Die Verfügbarkeit wird von den LocalDate checkIn und checkOut bestimmt

return List<Room>

getRoomStatus →Erzeugt eine Liste mit dem Status aller Zimmer.

return List<Room>

convertToRoomStatus → speichert die gewünschte Information in einem DTO und schickt diese zu *getRoomStatus*

return roomStatus

bindRoomToReservation (int reservationId, int roomId)→ Verknüpft ein Zimmer mit einer Buchung

put → Aktualisiert ein Zimmer in der Datenbank

delete(int roomId) → Löscht ein Zimmer aus der Datenbank

DTOS

Liste von DTOs, die in diesem Projekt verwendet werden.

CostumerInfo:

- ***String firstName;***
- ***String lastName;***
- ***LocalDate checkIn;***
- ***LocalDate checkOut;***

EmployeeInfo:

- ***String firstName;***
- ***String lastName;***
- ***String position;***
- ***double salary;***
- ***boolean admin;***

ReservationStatus:

- ***double finalPrice;***
- ***boolean isComplete;***
- ***LocalDate checkIn;***
- ***LocalDate checkOut;***

ReservationSummary:

- ***Int reservationId;***
- ***LocalDate reservationsDate;***
- ***LocalDate checkIn;***
- ***LocalDate checkOut;***
- ***String customerInfo;***
- ***Set singleRooms;***
- ***Set doubleRooms;***
- ***Set familyRooms;***
- ***Set suites;***
- ***double finalPrice;***

RoomStatus:

- ***boolean isAvailable;***
- ***int roomNumber;***
- ***double price;***