

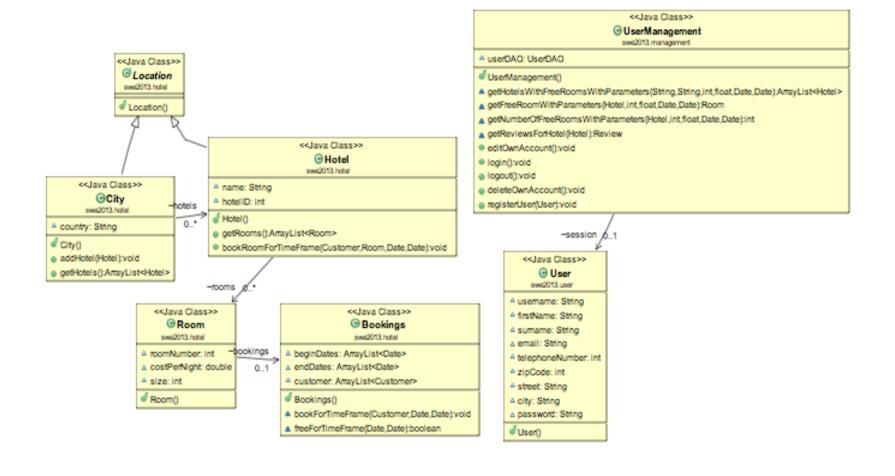
Grasshopper – Designmodell 2.0

Mit einem Sprung zum besten Urlaub

Anreiter, Simon Kocman, Andreas Moser, Victoria

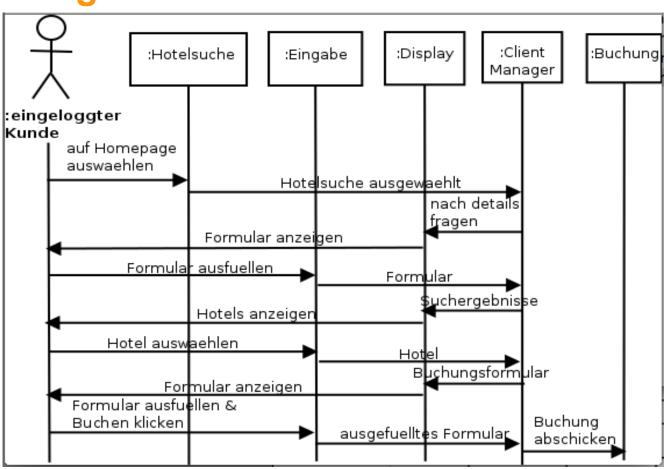


Buchung



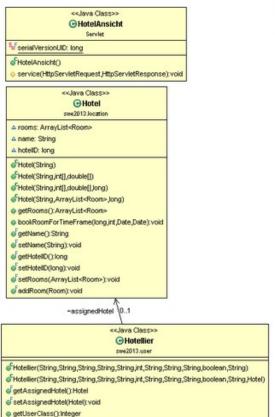


Buchung





Hotellieransicht



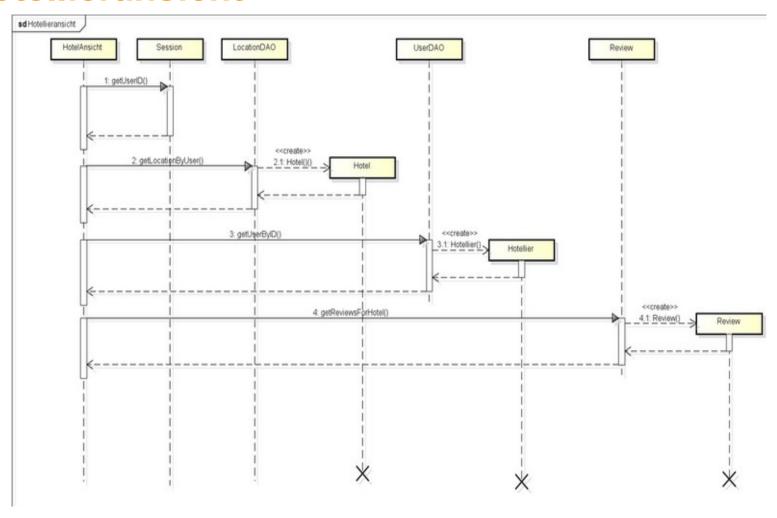
swe2013.dao ASInsert_City_SQL: String △Squery_City_begin: String AS delete_City_Query: String △Sassign_TA_Query: String △ScityName: String a^ScityCountry: String △ScityReviewText: String △ScityStars: String Δ^ScityTAID: String aScityOrder: String[] a insert_Hotel_SQL: String a guery_Hotel String aSquery_Hotel_Owner: String ASquery_Hotel_City: String △Sdelete_Hotel_Query: String △Supdate_Hotel: String △ShotelD: String △ShotelName: String aShotelCityName: String △ShotelCountryName: String AShotelOwner: String △SqueryFreeRooms: String △SqueryFreeHotels: String AShotelOrder: String[] SqlLocationDAO() saveHotel(Hotel,CityJong):long deleteHotel(long): void updateHotel(HotelJong): void saveCity(City):void deleteCity(City); void getHotelByID(long):Hotel getHotelbyOwner(long):Hotel getHotelsFromCity(String,String):ArrayList<Hotel> getCityByAssignedTA(long):City getTAForCity(String,String):Long assignTA(String,String,long);void freeRoomsForTimeFrame(long,Date,Date,int,int):long() m*toSGLDate(Date):Date hotelsWithFreeRooms(Date,Date,int,int):long[]

<<Java Class>>





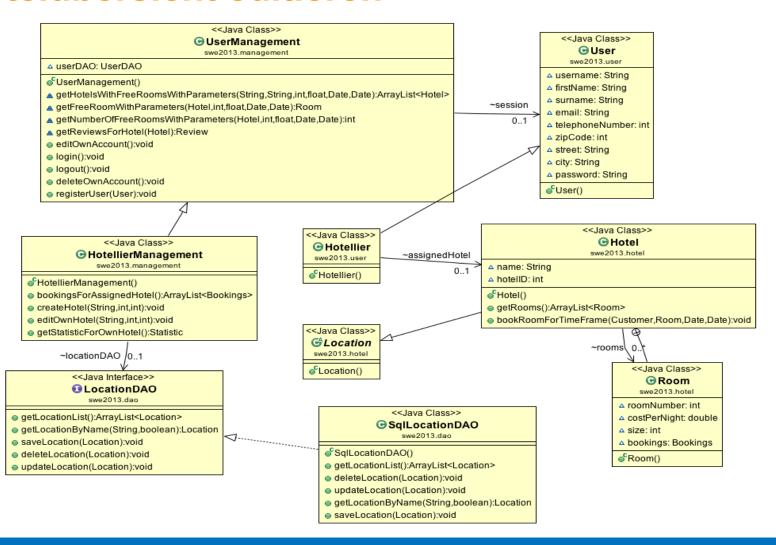
Hotellieransicht





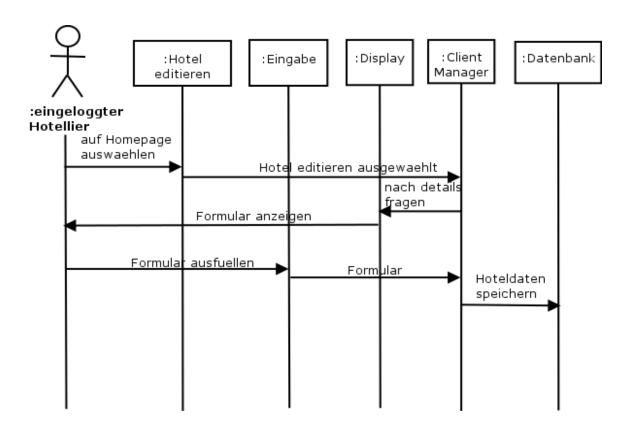


Hotelübersicht editieren





Hotelübersicht editieren



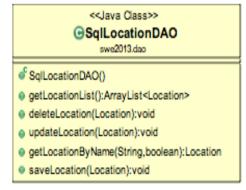


Hoteldaten editieren



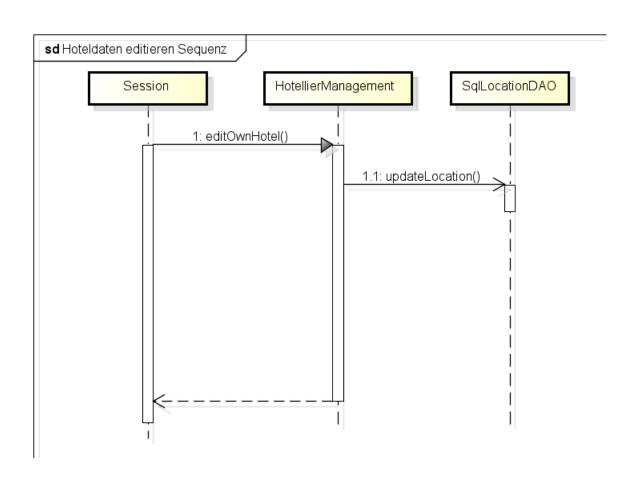








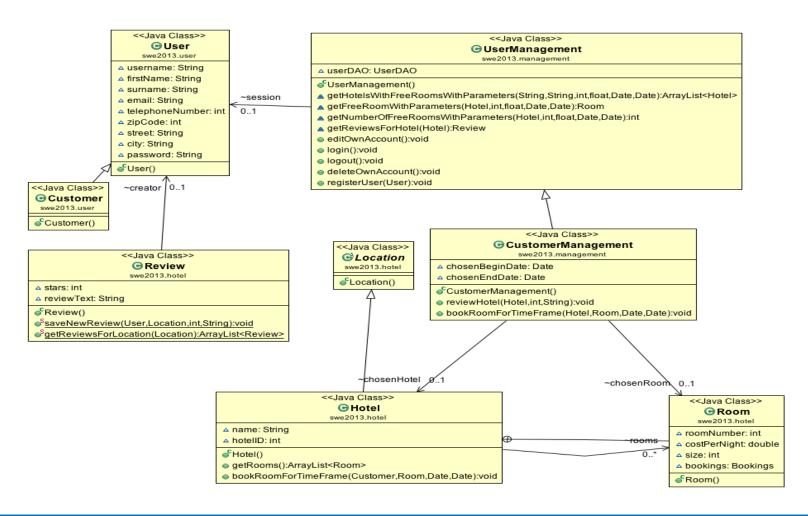
Hoteldaten editieren





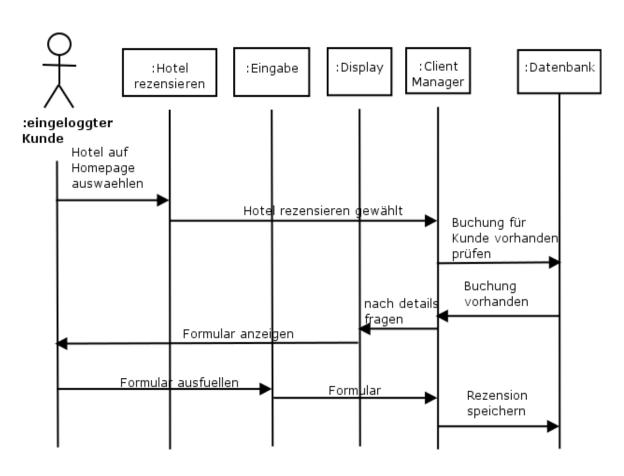


Rezenssion schreiben



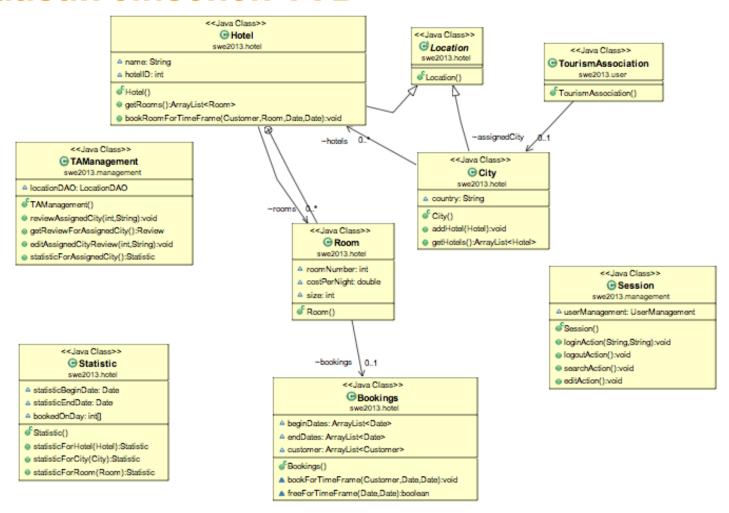


Rezension schreiben



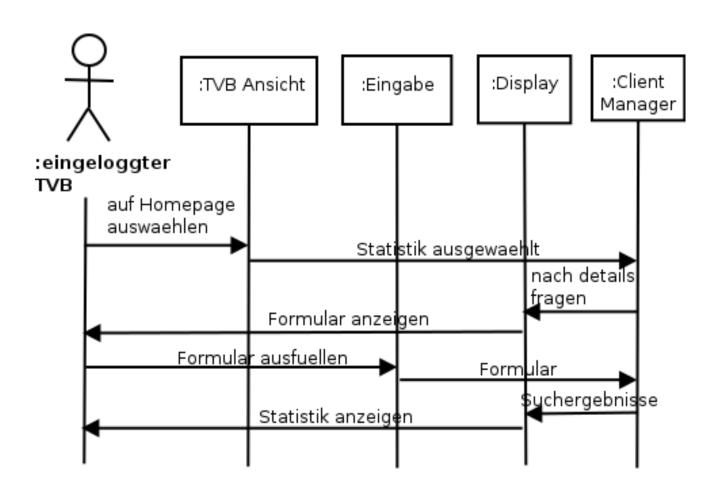


Statistik einsehen TVB





Statistik einsehen TVB



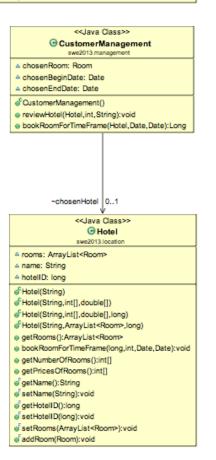


<<Java Class>>

Meine

Buchungen

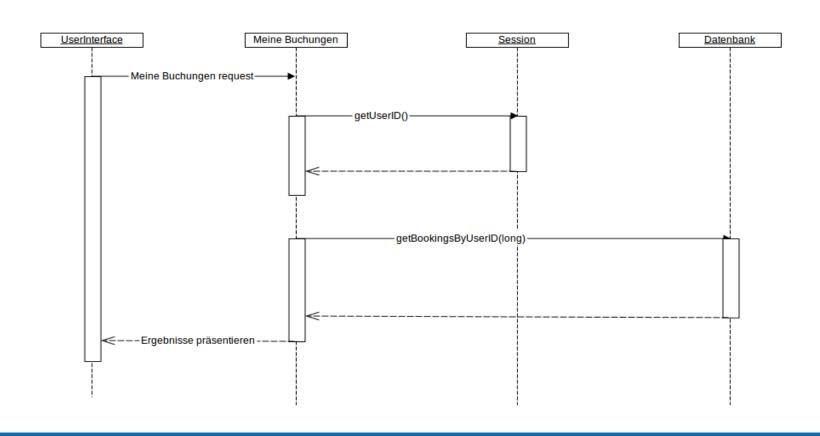
- & Customer(String, String, String, String, String, int, String, String, String, boolean, String)
- getUserClass():Integer
- getassignedLocationID():Integer



swe2013.dao △Sinsert City SQL: String △Squery City begin: String △Sdelete City Query: String △Sassign TA Query: String △^ScityName: String △ScityCountry: String △ScityReviewText: String △ScityStars: String △ScityTAID: String ^{a5}cityOrder: String[] △Sinsert Hotel SQL: String △Squery Hotel: String △Squery Hotel Owner: String △Squery Hotel City: String △Sdelete Hotel Query: String △Supdate Hotel: String △ShoteIID: String △ShotelName: String △ShotelCityName: String AShotelCountryName: String △ShotelOwner: String △SqueryFreeRooms: String △SqueryFreeHotels: String △ gueryFreeHotelsSummary: String △⁵summaryOrder: String[] 45hotelOrder: String[] SqlLocationDAO() saveHotel(Hotel, City, long):long ø deleteHotel(long):void updateHotel(Hotel,long):void saveCity(City):void ø deleteCity(City):void getHotelBvID(long):Hotel getHotelbyOwner(long):Hotel getHotelsFromCity(String,String):ArrayList<Hotel> getCityByAssignedTA(long):City getTAForCity(String,String):Long assignTA(String,String,long):void freeRoomsForTimeFrame(long, Date, Date, int, int):long[] hotelsWithFreeRooms(Date, Date, int, int);long[] freeHotelsSummary(String,String,Date,Date,int,int):String[][



Meine Buchungen





<<Java Class>>

@ Review

swe2013.location

△Sinsert Review Hotel; String

△Ssql hotelreviewText: String

△Ssql city reviewText: String

△Sql cityReviewCreator: String

△ShotelReviewOrder: String[]

△ScityReviewOrder: String[]

△ creatorUsername: String

getReviewText():String

setReviewText(String):void

a reviewCity(long,String,int);void

øreviewHotel(long,long,String,int):void

getReviewsForHotel(long):ArrayList<Review>

§ getReviewsForCity(String,String):Review

<u>ii reviewFromResultSet(Object[]):Review</u>

△Ssql hotelReviewCreator: String

△Sreview City: String

△ query_Hotel: String

^{∆S}guery City: String

△Ssql_stars: String

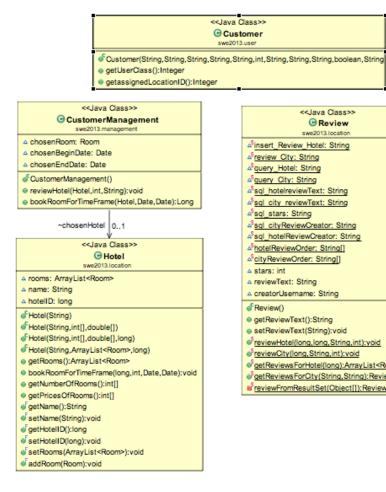
△ reviewText: String

△ stars: int

^cReview()

Meine

Rezensionen

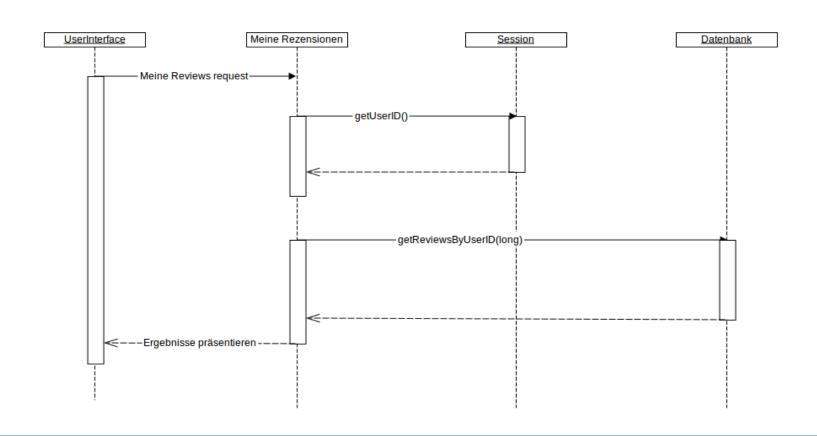


⊕ SalLocationDAO swe2013 dan △Sinsert City SQL: String △Squery_City_begin: String △Sdelete City Query: String △Sassign_TA_Query: String ^{∆S}citvName: String △ScityCountry: String △ScityReviewText: String △ScityStars: String △ScityTAID: String 45cityOrder: String[△Sinsert Hotel SQL: String △Squery_Hotel: String △Squery Hotel Owner: String △Squery_Hotel_City: String △Sdelete Hotel Query: String △Supdate Hotel: String △ShoteIID: String 45hotelName: String △ShotelCityName: String △ShotelCountryName: String △ShotelOwner: String △SqueryFreeRooms: String △SqueryFreeHotels: String △SqueryFreeHotelsSummary: String △SsummaryOrder: String[] 45hotelOrder: String[] SqlLocationDAO() saveHotel(Hotel, City, long):long deleteHotel(long):void updateHotel(Hotel,long):void saveCity(City):void deleteCity(City):void getHotelByID(long):Hotel getHotelbyOwner(long):Hotel getHotelsFromCity(String,String):ArrayList<Hotel> getCityByAssignedTA(long):City getTAForCity(String,String):Long assignTA(String,String,long):void freeRoomsForTimeFrame(long, Date, Date, int, int):long[] in toSQLDate(Date):Date hotelsWithFreeRooms(Date,Date,int,int):long[] freeHotelsSummary(String,String,Date,Date,int,int):String()[

<<Java Class>>

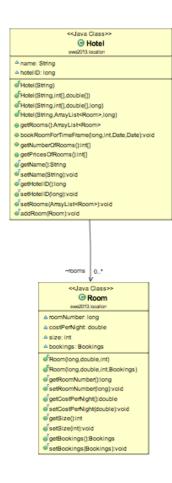


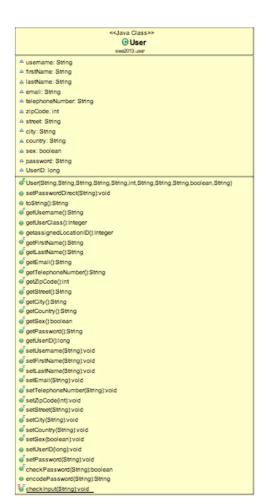
Meine Rezensionen





Hotelsuche



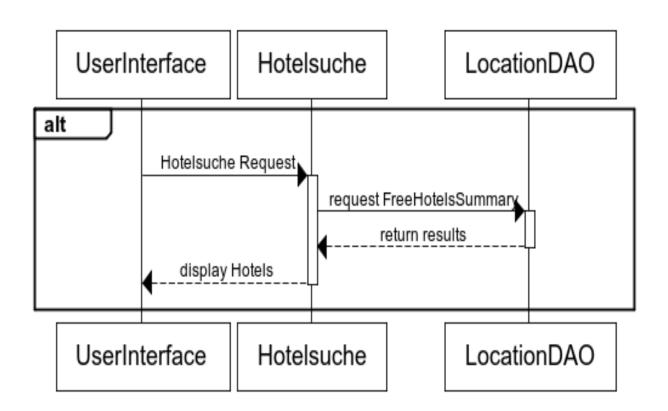




celava Classoo



Hotelsuche





·LocationDAO

- speichert & lest Locations
- zu den Locations gehören Städte und Hotels
- ist das Interface welches sich um die Speicherung von Location kümmert

·SQLLocationDAO

- Implementiert Location DAO und speichert die Daten in eine SQL Datenbank

·UserDAO

- Interface welches alle user daten speichert und lest

·SQLUserDAO

- Implementiert UserDAO und speichert die Daten in eine SQL Datenbank



·Bookings

- Buchungen für einen bestimmten Raum
 - Es wird angegeben wer die Buchung durchführt und das Datum von-bis
 - Buchungen werden nach Datum sortiert und in der SQL Datenbank gespeichert
 - Jeder Raum hat eine ArrayList wo die Buchungen für sich gespeichert werden

·City

- extends Location
 - Jede Stadt hat ein Land zu welchem es gehört
 - Hotels welche in einer Stadt sind werden in einem Array gespeichert



·Hotel

- extends Location
 - Jedes Hotel hat Räume welche in einem ArrayList gespeichert werden

·Location

- Abstrakte Überklasse von Hotel und City

·Review

- Objektwertig
 - Die Reviews werden direkt in die SQL Datenbank gespeichert
 - Man kann für jede Location, Hotel oder Stadt kann man alle Reviews suchen
 - Beim speichern kann jeder User zu einer Location immer nur ein Review schreiben
 - Sollte ein User zu einer Location ein erneutes Review schreiben, wird das vorherige überschrieben
 - Bei Kunden wird für das Hotelreview vorher überprüft, ob eine Buchung



·Statistik

- Statistik hat 3 Instanzvariablen: Beginn- EndDatum der Statistik und ein Array
 - Im Array wird gespeichert, wieviele Buchungen an dem Tag stattgefunden haben
 - Statistiken werden jeweils für bestimmte Tage erstellt
 - Die Hotelstatistik wird aus den Statistiken der Räume dieses Hotels erstellt
 - Die Stadtstatistiken werden aus den Statistiken der Hotels welche sich in dieser Stadt befinden erstellt

·HotellierManagement

·CustomerManagement

extends UserManagement



·Session

- Gibt die Pages an den Browser weiter

·TAManagement

- extends UserManagement

·UserManagement

- Sind die Funktionen die alle User ausführen können

·Customer, Hotellier, TA

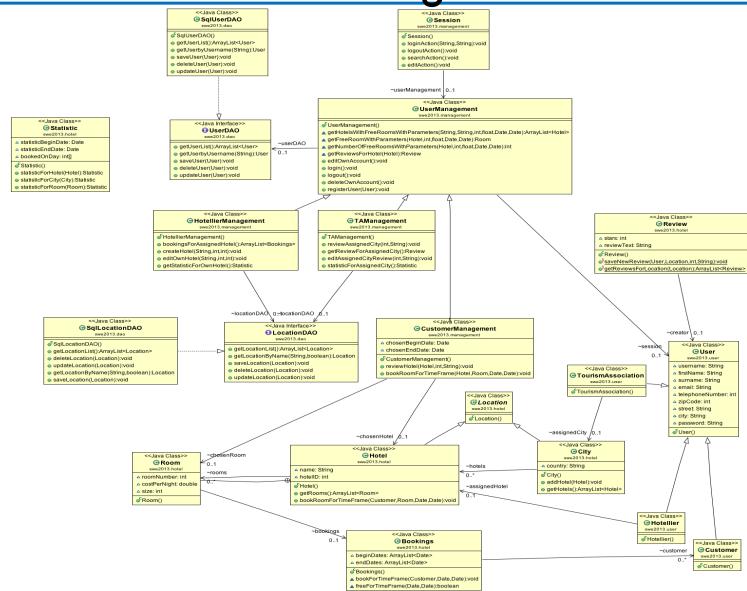
- extends User

·User

- public abstract class

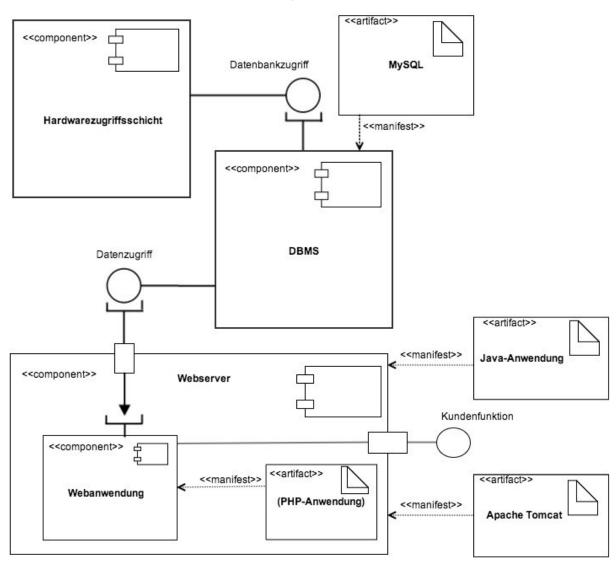
Übersichtsklassendiagramm





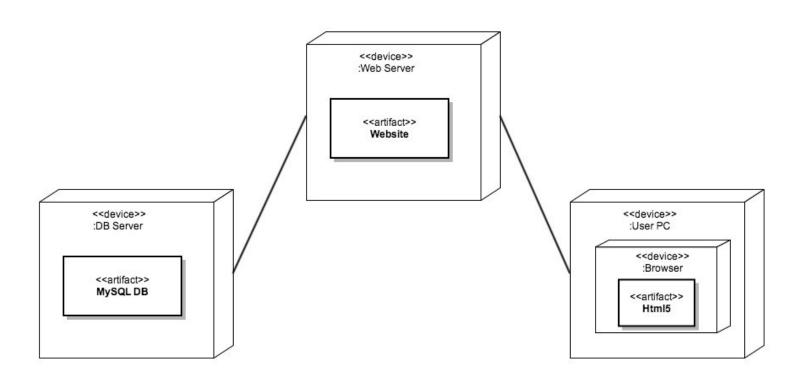


Komponentendiagramm









Datenspeicherung



User, Hotel und Buchungsdaten werden im Rahmen unserer Aufgabenstellung persistent mittels MySQL in einer Datenbank gespeichert. Die nötige Infrastruktur wird durch den ZID zur Verfügung gestellt (http://zid.univie.ac.at/mysql/).

Programmintern werden die Zugriffe auf die SQL Datenbank im Rahmen der Klassen SqlUserDAO und SqlLocationDAO mittels der *Java Database Connectivity (JDBC) API* Version 3.0 realisiert.



·Fragen? Anmerkungen?

Vielen Dank für Eure Aufmerksamkeit!