Architektur- und Design

2013-06-14

Projekt “Wartelistenverwaltung planbarer Operationen”

Autoren: Taylor Peer, Michael Wagner, David Stöckl

Version 1.0

Inhalt

Das folgende Dokument gibt Aufschluss über den technischen Aufbau der Applikation “Wartelistenverwaltung planbarer Operationen”.

1.1 Beschreibung der Komponenten

Die Aufbau der Applikation ist grundsätzlich wie folgt in Schichten aufzuteilen:

Daten-Layer
Die Datenbank als unterste Schicht der Applikation ist ein klar abgetrennter Bereich, auf den die Objekt des Business Layers ausschließlich über DAOs (Data Access Objects) zugreifen. Diese Datenbank ist allein für die sämtliche Datenhaltung der Applikation zuständig. Konkret wurde MongoDB eingesetzt, um die dort integrierte GEO-Suche nutzen zu können (=Anforderung der Applikation).

Domain-Layer
Der Domain Layer ist eine abgetrennte Schicht in der ausschließlich die Domain-Objekte definiert sind. Die Domain-Objekte und nur diese werden über den Aufruf von DAO-Methoden in die Datenbank gespeichert bzw. aus der Datenbank ausgelesen. Die DAOs und deren konkrete Implementierung werden jedoch im Domain-Layer verwaltet, da jede Komponente des Domain-Layers eigene Methoden hat.

Application-Layer
Der Application Layer selbst ist grundsätzlich in 3 Komponenten unterteilt (die jeweils eine eigene DAO Implementierung für den Zugriff auf den Datenbestand haben):

UI-Komponente	Diese Komponente erfüllt folgende Aufgaben: 1. REST-Schnittstelle/Kapselung nach “außen” ins öffentliche Web: Request-Handling & Mapping 2. Request-Processing & Views Generierung: Generiert je nach Aufruf die korrekte Seite und die entsprechende View (JSP) 3. Notifications-API (JSON): Liest Notifications für eingeloggte Clients aus der DB aus und stellt diese als JSON zur Verfügung 4. Weiterleitung von Reservierungsanfragen an die Allocator Komponente(n) (RabbitMQ) 5. Verarbeitung / Speicherung administrativer Daten
Allocator-Komponente	● Diese Komponente ist für den speziell aufwändigen Teil der Reservierungsfindung zuständig. ● Da ein Ausfälle der Komponente denkbar sind, wird diese Komponente auf zwei logisch unabhängigen Servern ausgeführt, so dass im Falle eines Ausfalls Reservierungsanfragen weiterhin bearbeitet werden können. ● Außerdem leitet diese Komponente Erfolgs/Misserfolgs-

	Nachrichten an die Messenger-Komponente weiter.
Messenger-Komponente	Diese Komponente ist für die Generierung Benutzer-spezifischer Nachrichten und deren Persistierung in der Datenbank zuständig. Sie wird ausschließlich vom Allocator mittels Messaging (RabbitMQ) angesprochen.

View Layer
Die UI-Komponente des Application Layers ist gleichzeitig für die Generierung der korrekten Views (HTML,CSS) für das Web zuständig. Dieser Teil der UI-Komponente entspricht dem View Layer.

1.2 Beschreibung der Schnittstellen

Wichtig: Für eine genauere Beschreibung der UI-Schnittstelle siehe das beiliegende Dokument HTTP-API.

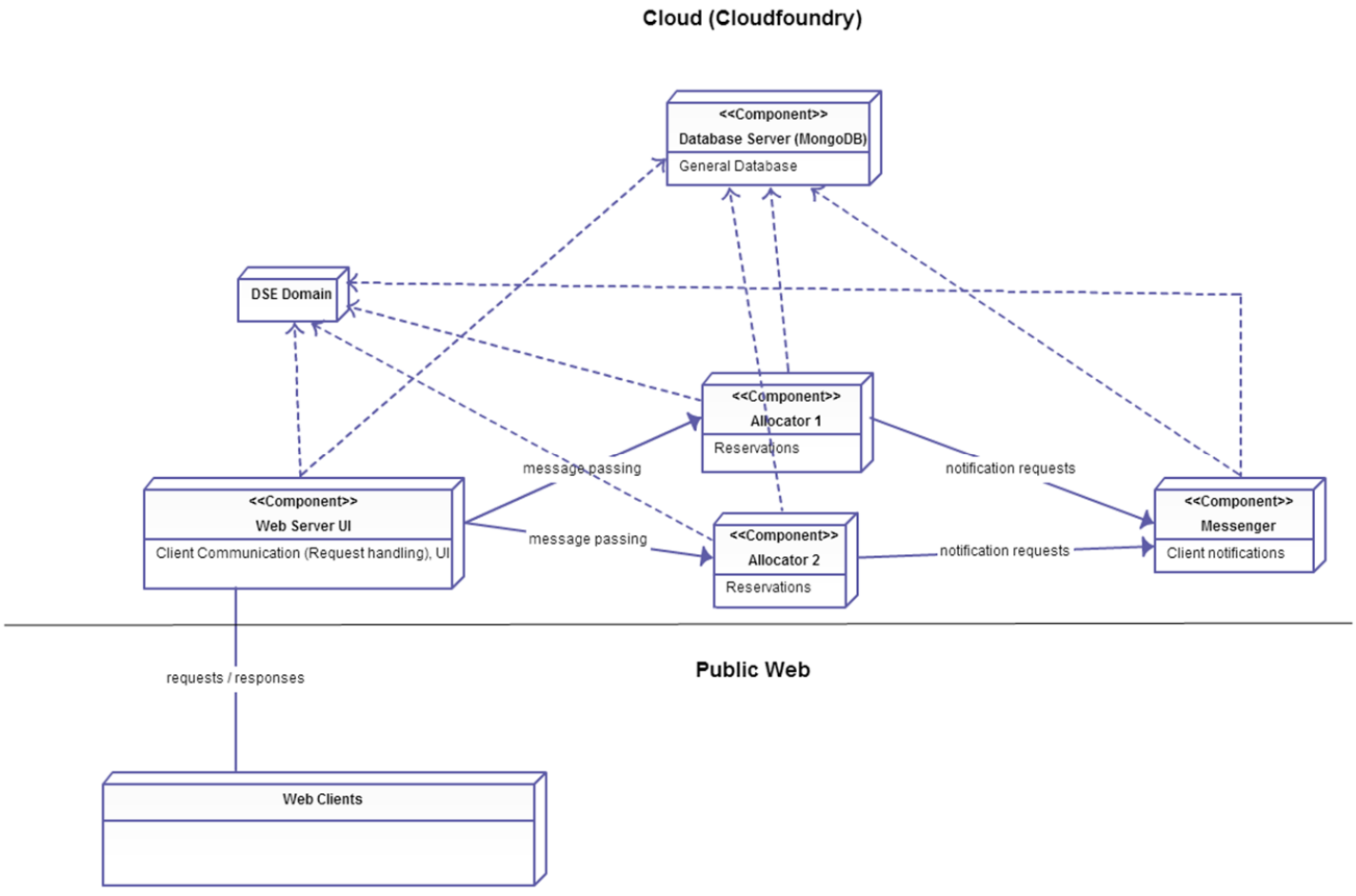
- Datenbank
    - Schnittstelle für Application Layer
        - benötigt Domain Layer
        - Zugriff über DAOs als Design-Empfehlung
- UI-Komponente:
    - Schnittstelle für Webclients (vor allem Browser)
        - für die Öffentlichkeit
            - OP-Slots + benutzerdef. Suche
        - für Patienten
            - Persönliche OP-Slot Liste + benutzerdef. Suche
        - für Doktoren
            - Persönliche OP-Slot Liste
            - Reservation-Request Interface
            - Liste aller Patienten
        - für Krankenhäuser
            - Liste der OP-Slots des Krankenhauses + benutzerdef. Suche
        - für Administratoren
            - Anlegen von Doktoren, Patienten und Krankenhäusern und deren jeweilige Anzeige
    - Schnittstelle für Notifications zur Client-spezifischen Verwertung (JSON-Schnittstelle)
- Allocator-Komponente
    - definierte Schnittstelle für Reservierungsanfragen
- Messenger-Komponente

- definierte Schnittstelle für Notification-Requests

Hinweis: Für eine genauere Definition der Schnittstellen ist das Dokument “HTTP-API” heranzuziehen.

1.3 Deployment of the application

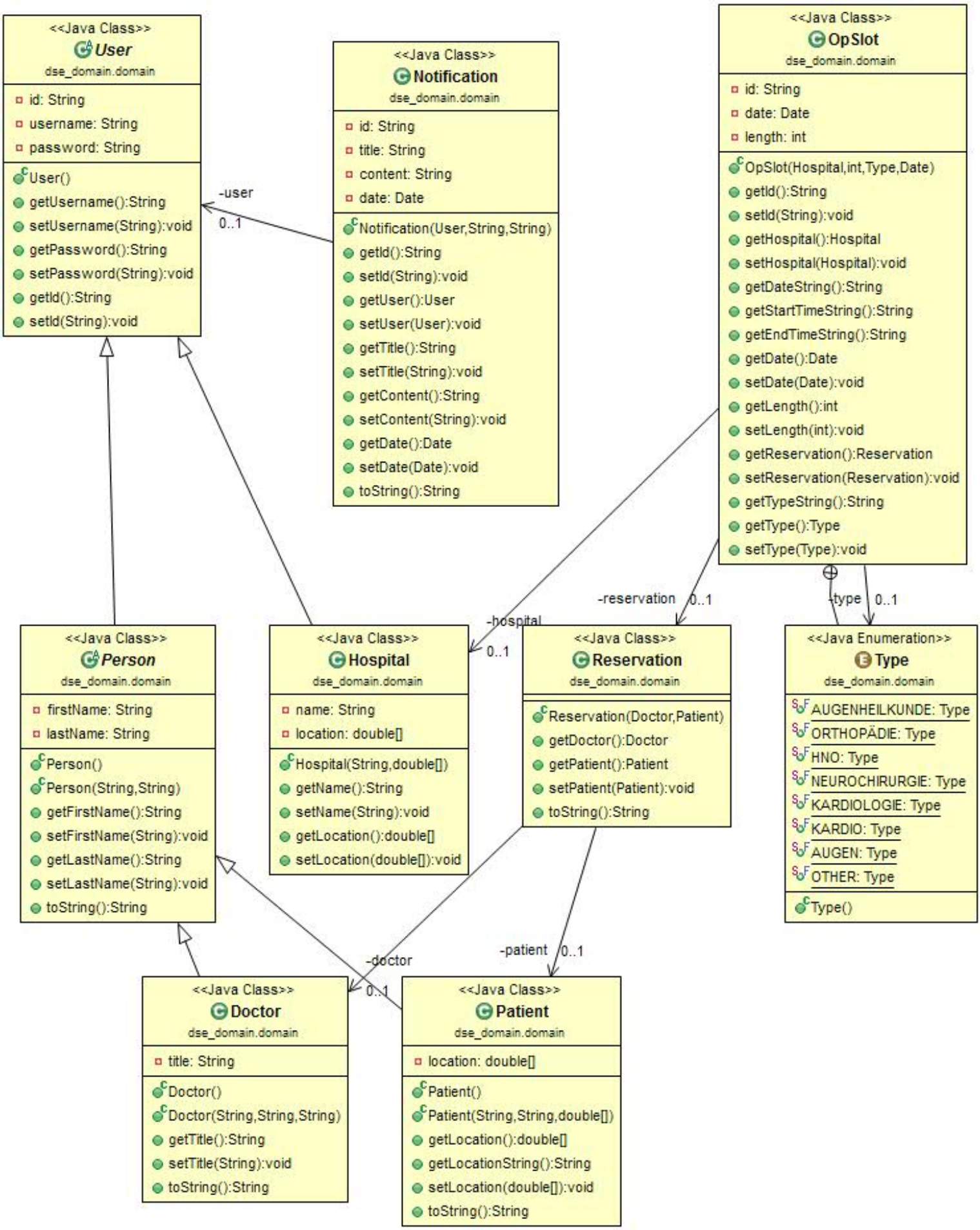
Das folgende Deployment-Diagram zeigt wie das Deployment funktioniert und wie die einzelnen Komponenten der Applikation getrennt gezeigt werden können (Deployment-Diagram).



Hierbei gilt:
gestrichelte Pfeile … Abhängigkeiten (bei der Datenbank auch Datenverkehr)
durchgehende Pfeile … Kommunikation (Messaging)
normale Verbindung … REST-Anfragen / Datenverkehr

1.4 Logical View

Das folgende Class-Diagramm zeigt den Aufbau des Domain Models, das sowohl die Basis für die Speicherung in den Aufbau der Datenbank als auch die Grundlage für die Funktionalität des Application Layers ist.


<<Java Class>>
User
dse_domain.domain
id: String
username: String
password: String
User()
getUsername():String
setUsername(String):void
getPassword():String
setPassword(String):void
getId():String
setId(String):void
<<Java Class>>
Notification
dse_domain.domain
id: String
title: String
content: String
date: Date
Notification(User,String,String)
getId():String
setId(String):void
getUser():User
setUser(User):void
getTitle():String
setTitle(String):void
getContent():String
setContent(String):void
getDate():Date
setDate(Date):void
toString():String
-user
0..1
<<Java Class>>
OpSlot
dse_domain.domain
id: String
date: Date
length: int
OpSlot(Hospital,int,Type,Date)
getId():String
setId(String):void
getHospital():Hospital
setHospital(Hospital):void
getDateString():String
getStartTimeString():String
getEndTimeString():String
getDate():Date
setDate(Date):void
getLength():int
setLength(int):void
getReservation():Reservation
setReservation(Reservation):void
getTypeString():String
getType():Type
setType(Type):void
-reservation
0..1
-type
0..1
-hospital
0..1
<<Java Class>>
Person
dse_domain.domain
firstName: String
lastName: String
Person()
Person(String,String)
getFirstName():String
setFirstName(String):void
getLastName():String
setLastName(String):void
toString():String
<<Java Class>>
Hospital
dse_domain.domain
name: String
location: double[]
Hospital(String,double[])
getName():String
setName(String):void
getLocation():double[]
setLocation(double[]):void
<<Java Class>>
Reservation
dse_domain.domain
Reservation(Doctor,Patient)
getDoctor():Doctor
getPatient():Patient
setPatient(Patient):void
toString():String
<<Java Enumeration>>
Type
dse_domain.domain
AUGENHEILKUNDE: Type
ORTHOPÄDIE: Type
HNO: Type
NEUROCHIRURGIE: Type
KARDIOLOGIE: Type
KARDIO: Type
AUGEN: Type
OTHER: Type
Type()
-doctor
0..1
-patient
0..1
<<Java Class>>
Doctor
dse_domain.domain
title: String
Doctor()
Doctor(String,String,String)
getTitle():String
setTitle(String):void
toString():String
<<Java Class>>
Patient
dse_domain.domain
location: double[]
Patient()
Patient(String,String,double[])
getLocation():double[]
getLocationString():String
setLocation(double[]):void
toString():String

Das folgende Sequenz-Diagramm zeigt den Ablauf für das Eintragen und Auslesen von Reservierungen.