

2014

Autor/en:	Kodras, Özsoy, Scholz, Vogt
Klasse:	4AHIT
Letzte Änderung:	09.05.2014
Version:	1.3

Protokoll

Scheduler

Protokollierung der einfachen Scheduling-Software mit allen benötigten Funktionen für ein benutzerfreundliches und einfach bedienbares Terminvereinbarungssystem.

Inhaltsverzeichnis

Inhaltsverzeichnis.....	1
Aufgabenstellung.....	3
Terminvereinbarungssystem.....	3
Benutzer.....	3
Organisator.....	3
Teilnehmer	3
Notifications.....	3
Events.....	4
Aufgabenstellung.....	4
Vorgehensweise.....	4
Termine:.....	5
Bewertungskriterien:.....	5
Arbeitsaufteilung mit Aufwandschätzung.....	6
Endzeitaufteilung	7
Designüberlegung	8
Applikationsaufteilung.....	8
Klassendiagramm.....	8
Requirementsanalyse (Use-Case-Diagram).....	9
Statemachine-Diagram (Zustandsdiagramm).....	10
GUI Mockup.....	11
Technologiebeschreibung.....	15
Hibernate.....	15
Allgemein.....	15
Funktionalität.....	15
Java Persistence API	17
Entities.....	17
Primary-Keys in Entities	17
Managing Entities	17
Mapping Annotations.....	18
Vaadin.....	18
Glassfish-Server	18
MySQL	18

Arbeitsdurchführung.....	19
Testbericht.....	20
Quellenangaben.....	21

Aufgabenstellung

Terminvereinbarungssystem

update 25.04.14

Es soll ein kollaboratives Terminvereinbarungssystem (ähnlich wie Doodle) erstellt werden, in dem sich Benutzer koordinieren können. Folgende Anforderungen sollen dabei erfüllt werden:

Benutzer

- Neue Benutzer können sich registrieren
- Existierende Benutzer können sich einloggen
- Man kann nach registrierten Benutzern im System suchen (über ihren Namen).
- Ein Benutzer kann gleichzeitig ein Organisator und Teilnehmer sein.
- Jeder Benutzer kann sich die Events, die er organisiert, oder Events, an denen er teilnimmt, anzeigen lassen.

Organisator

- ist ein Benutzer, der Events mit Namen und mehreren Termin- und Zeitvorschlägen erstellt und die Einladungen an einige registrierte Benutzer schickt
- darf den Namen, die Termine und Zeiten eines Events ändern, aber nur bevor sich einer der Benutzer zu dem Event angemeldet hat
- darf neue Benutzer zu seinen Events zusätzlich einladen
- darf eingeladene Benutzer wieder löschen, bevor sich diese zu dem Event angemeldet haben
- darf die Events jederzeit löschen
- darf zu seinen Events Kommentare posten
- darf Kommentare zu seinen Events löschen (auch die von anderen Benutzern)
- Nachdem sich alle Benutzer zu einer Einladung angemeldet haben, darf der Organisator einen fixen Termin festlegen.

Teilnehmer

- wählt aus den vorgeschlagenen Terminen und Zeiten eines Events (eine Checkbox pro Zeitvorschlag reicht)
- darf seine Wahl ändern, bis ein fixer Termin existiert
- darf Kommentare zu Events, **zu denen er eingeladen ist**, posten

Notifications

- Ein Teilnehmer wird über jede neue/editierte/gelöschte Eventeinladung notifiziert.
- Weiteres wird ein Teilnehmer notifiziert, sobald ein fixer Termin für ein Event festgelegt wird.
- Ein Organisator wird notifiziert, sobald **alle Teilnehmer einen Termin gewählt haben**.
- Wenn ein Benutzer zur Zeit einer Notification offline ist, darf diese nicht verloren gehen. Der Benutzer bekommt alle seine versäumten Notifications, sobald er online kommt.

Events

Es kann zwei Arten von Events geben:

1. Events, bei denen sich die Teilnehmer auf (möglichst) einen Termin einigen sollen (Standardfall). Der Organisator legt letztendlich einen fixen Termin fest.
2. Events, bei denen jeweils nur ein Teilnehmer pro Termin erlaubt ist (z.B. für Elternsprechtage). Der Organisator muss jede Teilnehmer/Termin-Kombination fixieren.

Aufgabenstellung

Entwickeln Sie ein GUI-Programm, welches das Terminvereinbarungssystem realisiert. Bei der Abgabe müssen Sie die Aufgabe auf mindestens drei Rechnern (mit mehreren gleichzeitig gestarteten Clients) präsentieren.

Beim Starten des Programms müssen der gewünschte Benutzername und die Netzwerkadresse des Servers angegeben werden (kein Passwort erforderlich). Die Registrierung kann automatisch bei der ersten Anmeldung erfolgen.

Achten Sie bei der Implementierung auf die transaktionale Sicherheit. Überlegen Sie sich Situationen, in denen z.B. ein Benutzer versucht, eine Terminwahl zu einem in der Zwischenzeit gelöschten Event zu realisieren. Ihr Programm sollte auf solche und ähnliche Situationen entsprechend reagieren.

Beachten Sie bei der Implementierung, dass die Kommentare in derselben Reihenfolge aufgelistet werden müssen, in der diese von den einzelnen Benutzern abgeschickt wurden. Sie müssen sich auch Gedanken über die Persistenz der Informationen machen. Wenn die Serverinstanz herunterfährt, muss der gesamte Inhalt dauerhaft abgelegt worden sein.

Es reicht ein einfaches, aber funktionales GUI. Sie **dürfen (sollten!)** dafür Frameworks einsetzen.

Vorgehensweise

Es sind die Meta-Regeln zu beachten. Dabei ist zu beachten, dass nur durch eine obligatorische Design-Review durch die unterrichtenden Lehrkräfte, das gewählte Design (realisiert und vorgestellt mittels UML Diagrammen) verwendet werden darf. Nachträgliche Änderungen müssen durch einen Change-Request genehmigt werden. Diese müssen in eine Feature/Requirements Liste münden, die z.B. durch User-Stories definiert werden können. Zu bedenken sind auch nicht-funktionale Anforderungen an das System, wie z.B. die Anzeigegeschwindigkeit der ersten Termine und Kommentare.

Des Weiteren sind Programmier-Teams verpflichtend. Diese sind durch eine/n Tester/in und eine/n Programmierer/in definiert. Angenommene Tasks der einzelnen Stories werden gleichzeitig(!) vom Tester und Programmierer behandelt, wobei der Tester die Anforderungen in z.B. Unit-Tests und der Programmierer in den entsprechenden Codeteilen implementiert. Dabei soll sichergestellt sein, dass sofort geeignete Testfälle den gerade eben implementierten Code auf dessen Funktionstüchtigkeit überprüfen.

Integrations- und Systemtests sind verpflichtend. Dabei sind in diesem Fall auch automatisierte GUI-Tests zu verwirklichen. Der Testbericht im Protokoll muss auch eine kontinuierliche Verbesserung der zu erzielenden Story Points ersichtlich machen.

Termine:

8.5.2014 Ads: Design Review - Deadline für Design der Applikation!

4.6.2014 23:55: Deadline Abgabe (fix - keine Verlängerung möglich!)

5.6.2014: Abnahme Interview (15min / Gruppe)

Dazwischen müssen die Teamleader wöchentlich einen Review Termin selbstständig wahrnehmen.

Bewertungskriterien:

- Dokumentation
- Design
- Implementierung
- Funktionalität
- Tests/System

Arbeitsaufteilung mit Aufwandschätzung

Zuständige Person(en)	Task	Beschreibung	Geschätzte Zeit in h
Kodras	GUI-Mockups	GUI-Mockup für das Client Programm (am besten mit https://moqups.com/), soll alle wichtigen Funktionen enthalten. 3 Pros und Cons	4
Vogt	Use-Case-Diagram	Requirementsanalyse der Angabe anhand von USE-Case-Diagrammen und der dazugehörigen Beschreibung (erweiterte Punkteliste der Angabe)	3
Özsoy	Statemachine-Diagram	Erstellung eines Zustandsdiagrammes für SigmaScheduler in ASTAH und zusätzliches Exportieren des ASTAH-Files als Bilddatei	2
Özsoy	Dokumentation	Erstellung, Design, Zusammenführung aller Grafiken und Erweiterung/Ergänzung der Dokumentation unter Beachtung der Meta-Regeln	2
Scholz	Applikationsaufteilung	Aufteilung der Anwendung in Applikationen (Client/Server) und den verwendeten Technologien	3
Scholz	Teammanagement	Aufteilung und Beschreibung der Arbeitspakete für die Teammitglieder	1
Kodras, Vogt	UML-Export	Exportieren des ASTAH-UML-Diagramms als JAVA-Dateien	½
Özsoy	Technologiebeschreibung	Technologiebeschreibung von Hibernate, Vaadin, Glassfish-Server und MySQL	2
Kodras, Vogt	Hibernate-Mapping	Mappen der zu persistierenden Klassen mit Testing	4

Geschätzter Gesamtzeitaufwand

Person	Zeitaufwand in h
Kodras	8 ½
Özsoy	6
Scholz	4
Vogt	7 ½
Summe:	24

Endzeitaufteilung

Zuständige Person(en)	Task	Geschätzte Zeit in h	Tatsächliche Zeit in h	Kommentar
Kodras	GUI-Mockups	4	3	Erwartete Zeit zum Einarbeiten in das Mock-Up-Tool war zu hoch (Das Tool ist sehr leicht zu bedienen und zu verstehen)
Vogt	Use-Case-Diagram	3	4	Einlesen in die UML-Use-Case-Spezifikation
Özsoy	Statemachine-Diagram	2	5	Einlesen in die UML-Statemachine-Spezifikation; Änderungen/Verbesserungen nach der Erstellung des Diagramms
Özsoy	Dokumentation	2	4	Verbesserung, dynamische Gestaltung und Updaten der Dokumentation
Scholz	Applikations-aufteilung	3	5	Verzögerung, da 1. Designentwurf mit Client-/Server-Applikation verworfen wurde
Scholz	Teammanagement	1	1	/
Kodras, Vogt	UML-Export	$\frac{1}{2}$	$\frac{1}{4}$	Keine Probleme beim ASTAH-Export
Özsoy	Technologiebeschreibung	2		
Kodras, Vogt	Hibernate-Mapping	4		

Tatsächlicher Gesamtzeitaufwand

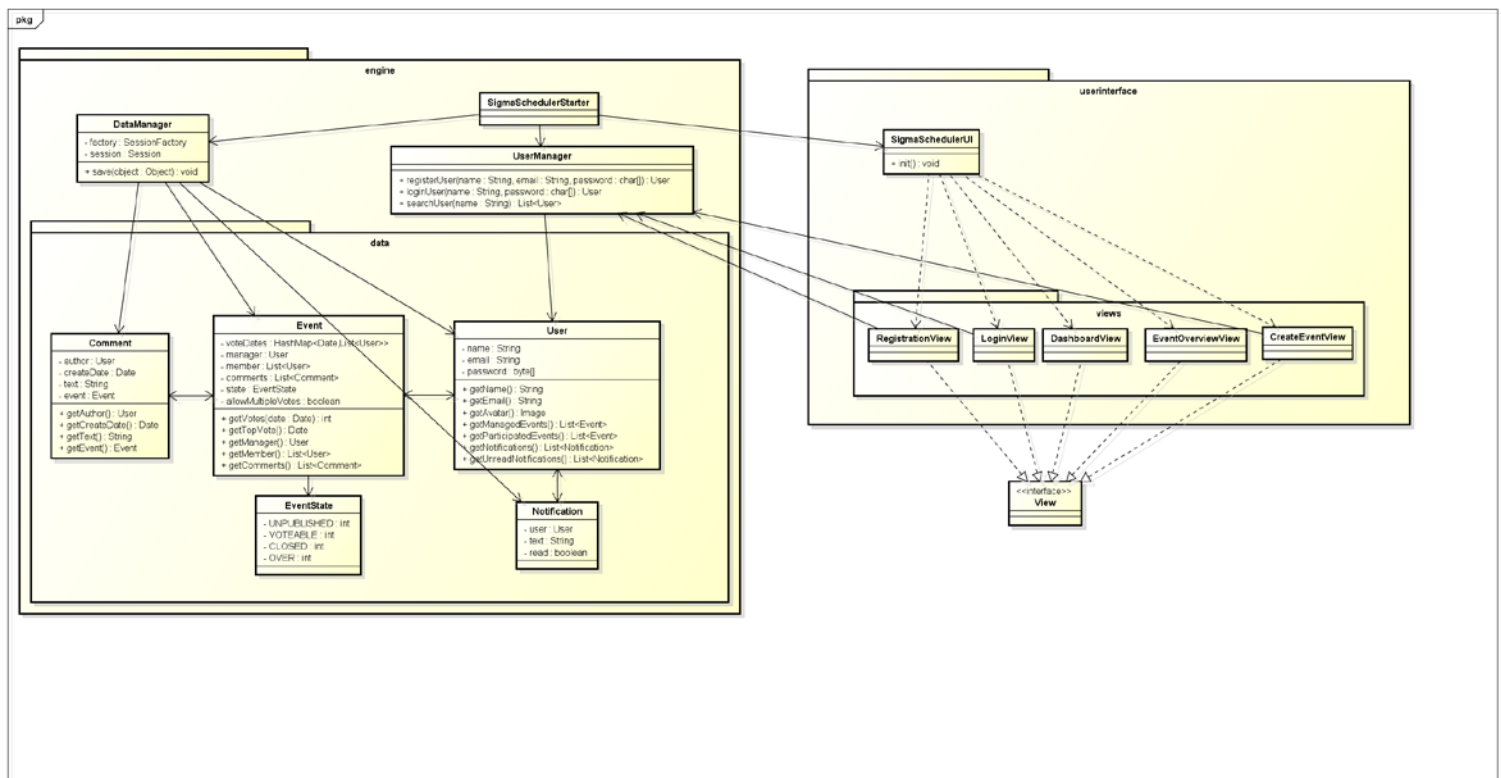
Person	Zeitaufwand in h
Kodras	3 $\frac{1}{4}$
Özsoy	9
Scholz	6
Vogt	4 $\frac{1}{4}$
Summe:	22 $\frac{1}{2}$

Designüberlegung

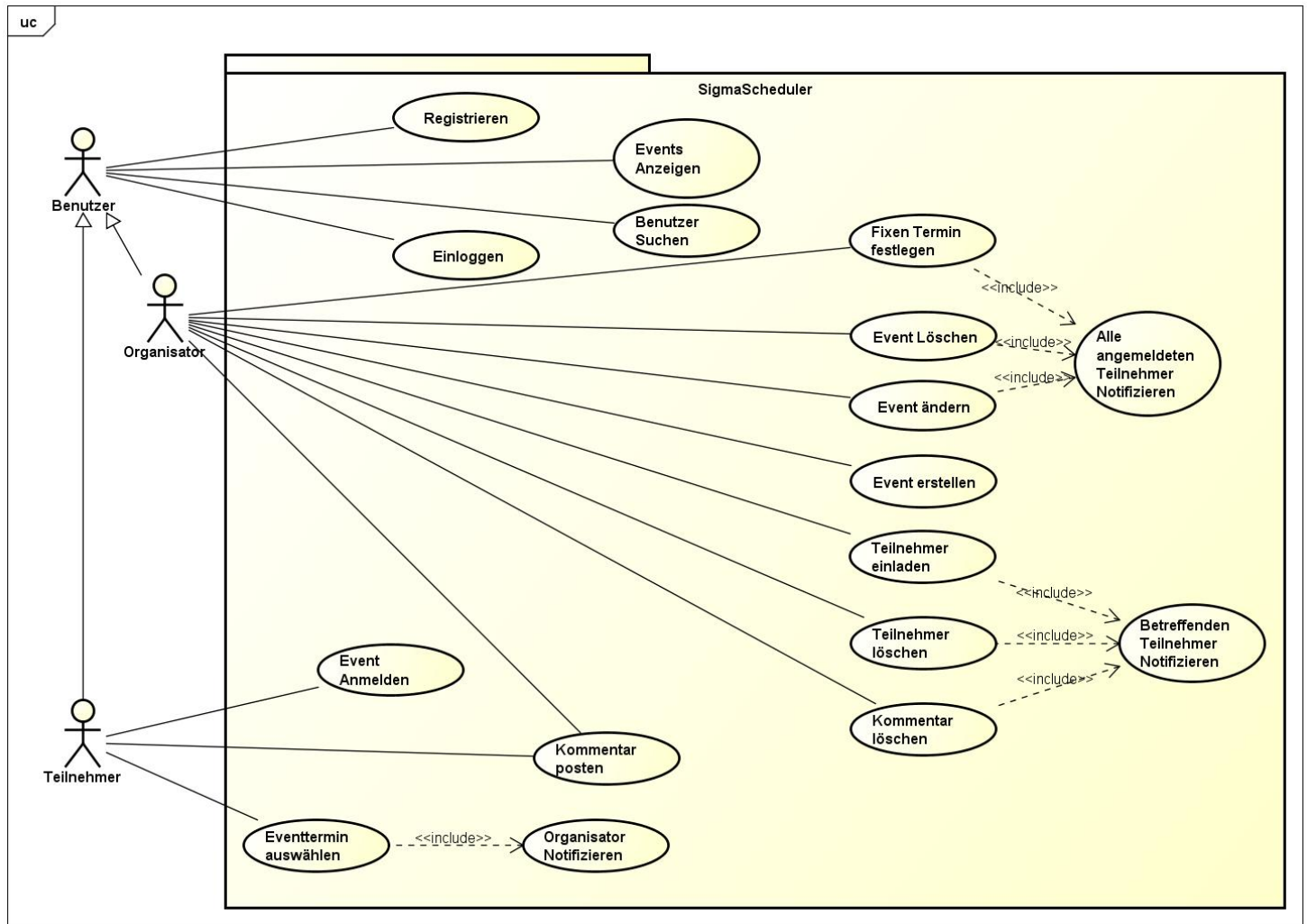
Applikationsaufteilung

Unser erster Designentwurf, war eine Aufteilung in eine Client und eine Server Applikation. Die Persistierung der Daten sollte der Server mit Hibernate lösen und die Kommunikation zwischen Client und Server mittels RMI realisiert werden. Das Client Programm wäre in dieser Variante eine einfache Java Applikation mit Swing GUI gewesen. Die Serverapplikation hätte den Datenbankzugriff mittels Hibernate implementiert gehabt. Nach einigen Überlegungen, sind wir im Team zu dem Entschluss gekommen, dass die Anwendung besser als Web Anwendung umgesetzt werden sollte. Gründe dafür sind vor allem die leichtere sowie Betriebssystem unabhängige Bedienung jedes Benutzers und der Entfall der Client-Server Kommunikation. Der neue Designvorschlag, sieht eine einzige Server Application vor (Java EE), welche mittels Vaadin Framework das grafische Interface im Browser realisiert, und die Daten wie schon davor mit Hibernate persistiert.

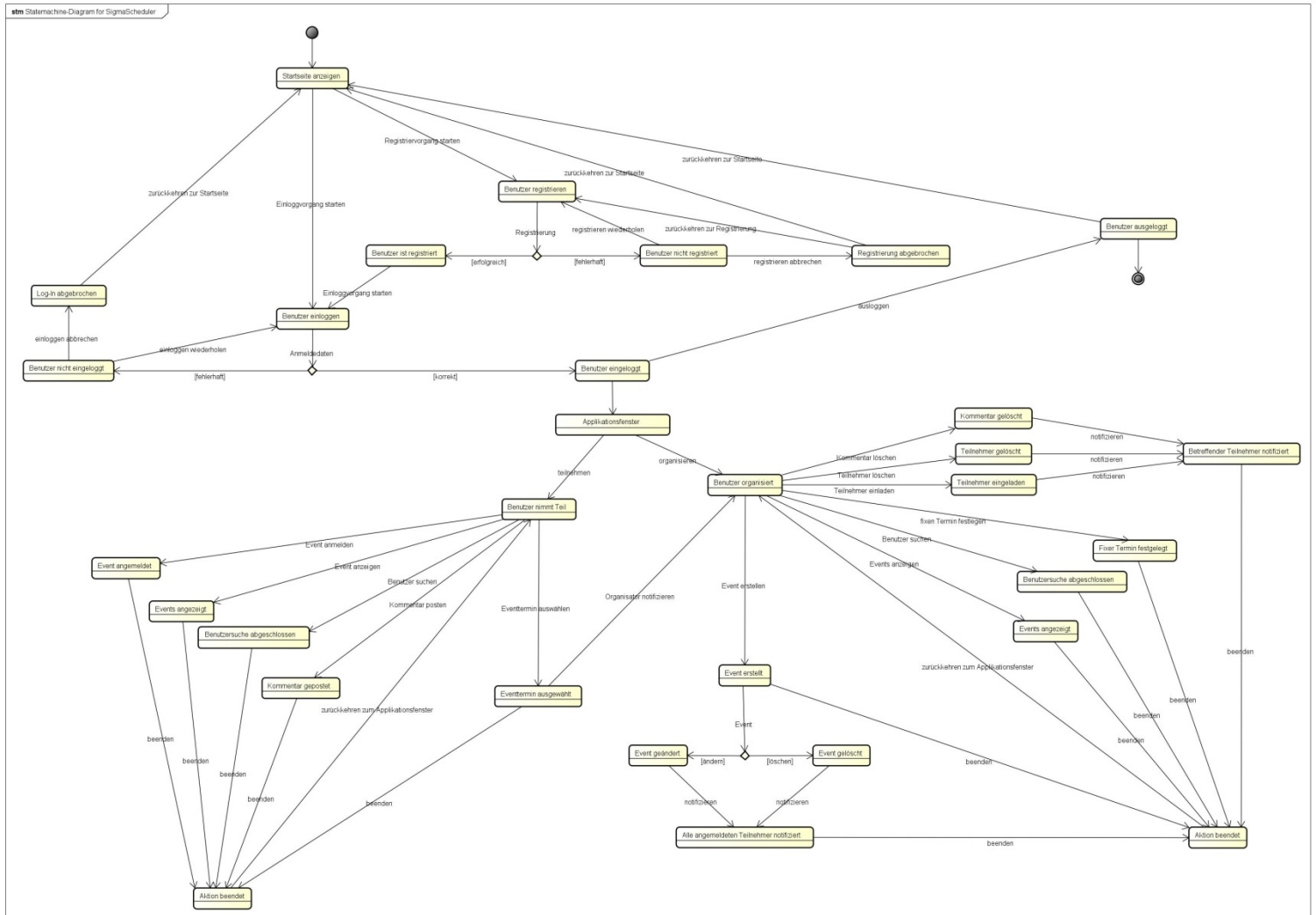
Klassendiagramm



Requirementsanalyse (Use-Case-Diagram)



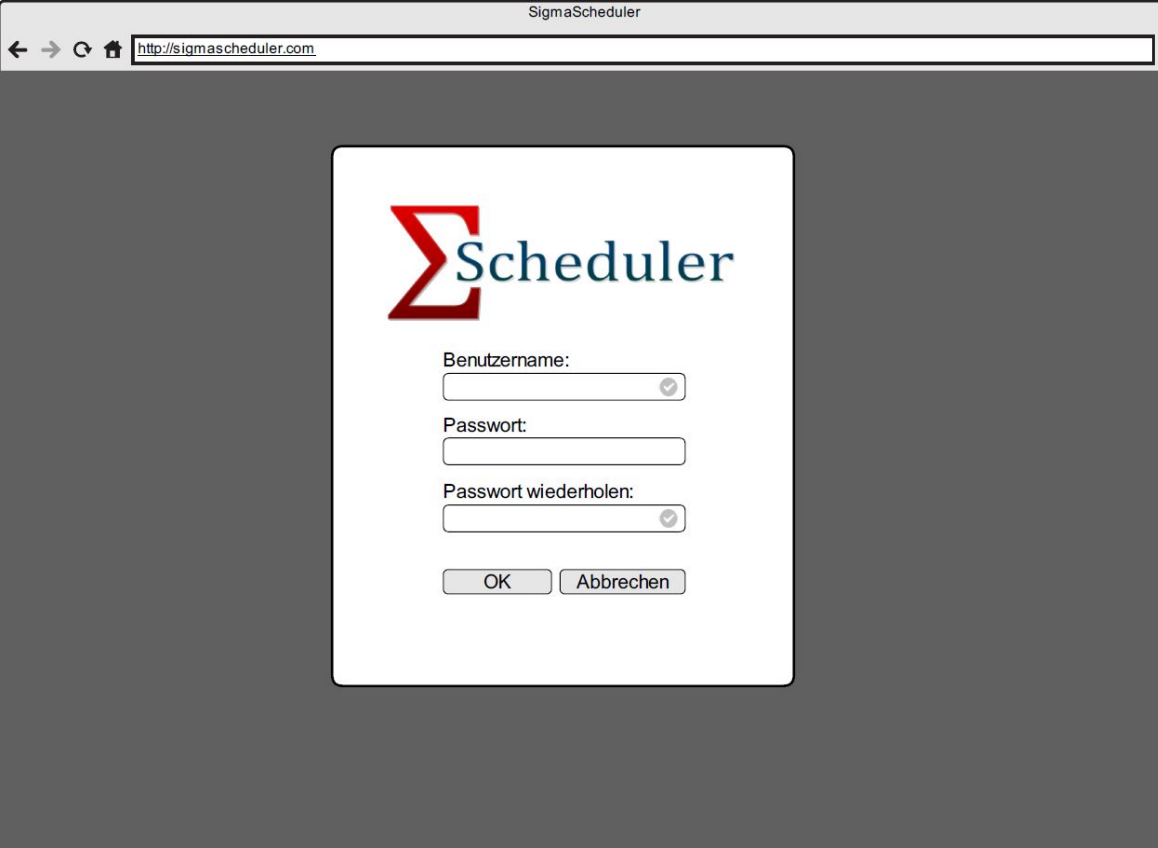
Statemachine-Diagramm (Zustandsdiagramm)



GUI Mockup




A web browser window titled "SigmaScheduler" with the address bar showing "http://sigmascheduler.com". The main content area has a dark gray background. In the center is a white rectangular box containing the login form. The form features the "SigmaScheduler" logo at the top, followed by labels "Benutzername:" and "Passwort:" each with a corresponding text input field. At the bottom of the form are two buttons: "Login" and "Registrieren".



A web browser window titled "SigmaScheduler" with the address bar showing "http://sigmascheduler.com". The main content area has a dark gray background. In the center is a white rectangular box containing the registration form. The form features the "SigmaScheduler" logo at the top, followed by labels "Benutzername:", "Passwort:", and "Passwort wiederholen:" each with a corresponding text input field. The "Benutzername:" and "Passwort wiederholen:" fields have a small checkmark icon on the right. At the bottom of the form are two buttons: "OK" and "Abbrechen".

SigmaScheduler


← → ↻ 🏠
http://sigmascheduler.com



Startseite

Meine Events


Event erstellen



Max Mustermann

⚙️

✖️



Max Mustermann

May 09, 2014

Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Anstehende Events:


Heute (09.05.2014):
8:00 [SYT-Wiederholung](#)
9:50 [Design-Abnahme](#)

Sonntag (11.05.2014):
3:00 [Wiener Schulball - Afterparty](#)
20:00 [Grillfest](#)

Samstag (17.05.2014):
20:00 [Grillfest](#)

SigmaScheduler

← → ↻ 🏠
http://sigmascheduler.com



Startseite

Meine Events

Event erstellen

Alle Events

Meine Events

Eingeladene Events

May 09, 2012

Su	Mo	Tu	We	Th	Fr	Sa
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

SigmaScheduler

← → ↻ 🏠 <http://sigmascheduler.com>

Σ Scheduler

🔍 Search

Startseite

Meine Events

Event erstellen

Alle Events | Meine Events | Eingeladene Events

May 09, 2012

Su	Mo	Tu	We	Th	Fr	Sa
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

SigmaScheduler

← → ↻ 🏠 <http://sigmascheduler.com>

Σ Scheduler

🔍 Search

Startseite

Meine Events

Event erstellen

Alle Events | Meine Events | Eingeladene Events

May 09, 2012

Su	Mo	Tu	We	Th	Fr	Sa
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

14 | 21

Technologiebeschreibung

Hibernate

Allgemein

Hibernate ist ein Open-Source-Persistenz- und ORM-Framework für Java. Für .NET ist eine portierte Version namens NHibernate verfügbar. [4]

Die Hauptaufgabe von Hibernate ist ORM(=Object-Relational Mapping). Dies ermöglicht es, gewöhnliche Objekte mit Attributen und Methoden (im Java-Bereich POJOs genannt) in relationalen Datenbanken zu speichern und aus entsprechenden Datensätzen wiederum Objekte zu erzeugen. Beziehungen zwischen Objekten werden auf entsprechende Datenbank-Relationen abgebildet. [4]

Darüber hinaus bietet Hibernate Mechanismen zur Kompatibilität mit verschiedenen Datenbanken. Die zum Datenbankzugriff erforderlichen SQL-Statements werden nicht explizit in SQL programmiert, sondern von Hibernate in Abhängigkeit vom SQL-Dialekt der verwendeten Datenbank generiert. [4]

Anwendungsseitig kann Hibernate in Java-Applikationen und Servlet-Engines benutzt werden oder in einen Applikationsserver integriert werden. [4]

Funktionalität

Zur Abfrage der persistierten Objekte gibt es die Möglichkeit über die SQL-ähnliche Abfragesprache Hibernate Query Language (HQL), mittels SQL-Statements oder objektorientiert mittels der Hibernate Criteria-API. Die Abfragen werden je nach verwendeter Datenbank mittels JDBC in den entsprechenden SQL-Dialekt übersetzt. Hibernate bietet für alle aktuellen relationalen Datenbanksysteme entsprechende Einstellungen, neue SQL-Dialekte können vom Benutzer selbst hinzugefügt werden. [4]

Objektrelationale Abbildung:

Hibernate ermöglicht eine transparente Persistenz von Plain Old Java Objects (POJOs). Einzig ein parameterloser Konstruktor muss gegeben sein. Da Hibernate mittels Reflection auf die Attribute der Klassen zugreift, sind keine öffentlichen Zugriffsfunktionen notwendig. [4]

Das Mapping von Java-Klassen auf Datenbanktabellen wird mittels XML-Datei(Mapping File) oder mit Java-Annotation durchgeführt. Bei der Verwendung von XML-Dateien können mittels der Hibernate Tools sowohl die passenden Java-Klassen als auch die Datenbanktabellen generiert werden, bei der Verwendung von Annotations ist letzteres auch möglich. [4]

Hibernate unterstützt Objektreferenzen (1:1-Relationen) und (typisierte) Java Collections (1:N- und N:M-Relationen), sowie auch reflexive Beziehungen auf die eigene Klasse. Hibernate kann auch so konfiguriert werden, dass Operationen wie Speichern oder Löschen auch über Relationen kaskadieren und somit die referentielle Integrität gewahrt bleibt. [4]

Hibernate unterstützt alle drei Arten der objektrelationalen Abbildung von Vererbungsbeziehungen (Tabelle pro Vererbungshierarchie, Tabelle pro Unterklasse und Tabelle pro konkrete Klasse), darüber hinaus auch Impliziter Polymorphismus als Spezialform von Tabelle pro konkrete Klasse. [4]

Ebenso ermöglicht Hibernate eine wahlfreie Abbildung von Java-Typen auf die von der Datenbank unterstützten SQL-Typen. Damit wird z. B. ermöglicht, Java-Enums analog zu normalen Integer-Properties abzubilden, oder einzelne Properties auf mehrere Tabellenspalten zu verteilen. [4]

Die wohl wichtigsten Hibernate Klassen sind SessionFactory, Session und Transaction. SessionFactory lädt die Konfiguration und die Mappings, und wird normalerweise nur einmal pro Anwendung erzeugt. Session ist das Bindeglied zwischen der Java-Applikation und den Hibernate-Diensten, und bietet Methoden für Insert-, Update-, Delete- und Query-Operationen. Transaction bildet JDBC- und JTA-Transaktionen ab. Geschachtelte Transaktionen werden nicht unterstützt. [4]

Weitere Funktionalitäten:

Neben der Kernfunktionalität umfasst Hibernate noch folgende Unterprojekte:

- ***Hibernate Annotations***
OR-Mapping mittels Annotations
- ***Hibernate EntityManager***
Umsetzung der Schnittstelle Java Persistence API
- ***Hibernate Shards***
Zugriff auf horizontal partitionierte Datenbanktabellen
- ***Hibernate Validator***
Definition von Daten-Integritäts- und Validierungsregeln in JavaBean-Klassen mittels Annotations
- ***Hibernate Search***
Transparente Erstellung eines Volltextindexes und Volltextsuche mit Lucene
- ***Hibernate Tools***
Entwicklungstools für Eclipse und Ant

- **NHibernate**

Hibernate für das .NET Framework. Die aktuelle Version 3.3.3 ist im August 2013 erschienen. NHibernate ist mehr als eine reine C# Portierung, sondern nutzt die zusätzlichen Möglichkeiten von C# wie z.B. Properties. Zu NHibernate gibt es bereits eine Reihe von Unterprojekten. So kann mit Fluent NHibernate die Erstellung der xml-Dateien für das Mapping vermieden werden. Das Automapping erlaubt es, auf eine erneute Aufzählung einzelner Attribute zu verzichten. Für die Konfiguration macht Fluent NHibernate intensiven Gebrauch von Lambda-Ausdrücken. [4]

Java Persistence API

Dient zum Mappen von Relationalen Daten in Java Applikationen. [2]

4 Bereiche:

- Java Persistence API
- Java query language
- Java Persistence Criteria API
- Object/relational mapping metadata [2]

Entities

Die Entitys representieren Tabellen in einer relationalen Datenbank. [2]

Entity instance = Datensatz / Spalte in einer Tabelle [2]

Voraussetzungen für Entity-Klassen:

- Klasse muss „javax.persistence.Entity“ beinhalten
- Klasse muss mind. einen leeren Konstruktor haben
- Klasse darf nicht als „final“ deklariert werden

Primary-Keys in Entities

Jede Entity hat einen Unique Object identifier(=Primary-Key). [2]

Managing Entities

Wird verwendet zur Verwaltung der Entity Insances. [3]

- Createn und Removen von Entity Instances

- Finden von Entites mittels Primary Key
- allows queries to be run on entities

Mapping Annotations

2 Arten:

- Logical Mapping --> describing the object model, the association between two entities etc.
- Physical Mapping --> describing the physical schema, tables, columns, indexes, etc. [3]

Grundlegende Annotationen: [3][5]

- @Entity - Deklariert die Klasse als Entity --> (Klasse wird als Tabelle gemapped)
- @Table - Zum setzen von: Name, Schema, Unique Constraints und Catalog der Tabelle
- @Id - Zur Identifikation der Tabelle (In der Art Primary-Key)
- @Column - Zum setzen der Eigenschaften einer Spalte (z.B.: Name, Length, Nullable, Unique)
- @GeneratedValue - Zum Festlegen der Strategie zur Erzeugung des Primary-Keys
- Annotations für Beziehungen zwischen Tabellen:
 - @OneToOne
 - @OneToMany
 - @ManyToOne
 - @ManyToMany

Vaadin

Glassfish-Server

MySQL

Arbeitsdurchführung

Testbericht

Quellenangaben

-
- | | |
|------------|--|
| [1] | Titel: Moqups;
Autor: moqups.com;
Online/Quelle: https://moqups.com/ ;
geändert am: /
entnommen am: 08.05.2014; |
| [2] | Titel: The Java EE 6 Tutorial;
Autor: Oracle and/or ist affiliates;
Online/Quelle: http://docs.oracle.com/javaee/6/tutorial/doc/bnbpy.html ;
geändert am: /
entnommen am: 09.05.2014; |
| [3] | Titel: Hiberate Annotations;
Autor: Red Hat Inc. And the various authors;
Online/Quelle: http://docs.jboss.org/hibernate/stable/annotations/reference/en/html/ ;
geändert am: /
entnommen am: 09.05.2014; |
| [4] | Titel: Hibernate Framework;
Autor: Wikipedia;
Online/Quelle: http://de.wikipedia.org/wiki/Hibernate_(Framework) ;
geändert am: /
entnommen am: 09.05.2014; |
| [5] | Titel: Important annotations used in Hibernate;
Autor: Sujith Mohan;
Online/Quelle: http://sujithforu.blogspot.co.at/2013/05/important-annotations-used-in-hibernate.html ;
geändert am: /
entnommen am: 09.05.2014; |
| [6] | Titel:
Autor:
Online/Quelle:
geändert am:
entnommen am: 09.05.2014; |
-