Erstellung einer JEE Applikation zur Veranstaltungsverwaltung

WI44/14

Jan Raupach

Ann-Katrin Kottig

# Inhaltsverzeichnis

[1 1](#_Toc478590449)

[2 Kurzbeschreibung 1](#_Toc478590450)

[3 Umsetzung der User-Stories 1](#_Toc478590451)

[3.1 Anlegen einer neuen Veranstaltung (Manager) 1](#_Toc478590452)

[3.2 Veröffentlichen einer Veranstaltung (Manager) 1](#_Toc478590453)

[3.3 Bearbeiten einer nicht veröffentlichten Veranstaltung (Manager) 1](#_Toc478590454)

[3.4 Suchen nach einer Veranstaltung 1](#_Toc478590455)

[3.5 Anzeigen von Details zu einer Veranstaltung 2](#_Toc478590456)

[3.6 Erhalten eines eindeutigen Reservierungscodes bei der Reservierung 2](#_Toc478590457)

[3.7 Anzeigen einer Liste aller Reservierungen (Manager) 2](#_Toc478590458)

[3.8 Anzeige noch verfügbarer Tickets 2](#_Toc478590459)

[4 Sprint-Backlog 2](#_Toc478590460)

[5 Datenmodell 3](#_Toc478590461)

[6 Installationshinweise 3](#_Toc478590462)

# Installationshinweise

Für dieses Projekt wurde neben den angegebenen Frameworks (z.B. JSF) das Persistenz-Framework Hibernate, sowie Hibernate Search verwendet. Damit Hibernate in Kombination mit dem Glassfish 4 funktioniert, muss die Datei jboss-logging.jar die in unserem Paket angehängt wurde, in den Ordner „[Pfad zum glassfish Ordner]/glassfish/modules“ kopiert werden und die Orginaldatei überschrieben werden. Um die MySQL-Datenbank einzurichten, wurde die Datei „createDatabase.sql“ erstellt. Dieses SQL-Statement wurde bei uns mit der MySQL-Workbench, über die Import-Funktion eingeladen um die Datenbank zu erstellen. Damit sich Hibernate mit der Datenbank verbinden kann, muss sie auf dem Port 3306 lauschen, ein Benutzer mit dem Namen „veranstaltung“ und dem Passwort „veranstaltung“ existieren und die nötigen Berechtigungen eingerichtet sein.

Die korrekte Lauffähigkeit der Anwendung kann von uns nur im Browser „Google Chrome“ gewährleistet werden. Der Firefox weist zum Beispiel Probleme mit dem verwendeten Datepicker auf.

# Kurzbeschreibung

In der Webanwendung zur Veranstaltungsverwaltung ist es einem Anwender möglich alle veröffentlichten Veranstaltungen einzusehen und nach ihnen zu suchen. Der Anwender kann sich zudem als einfacher Nutzer oder als Manager zu registrieren. Nach einer Registrierung ist eine Anmeldung mit Benutzername und Passwort möglich. Ein angemeldeter Anwender kann Tickets zu einer Veranstaltung reservieren und seine Reservierungen in einer Übersicht ansehen. Ein angemeldeter Manager kann neue Veranstaltungen erstellen und seine eigenen noch nicht veröffentlichten Veranstaltungen bearbeiten. Desweiteren kann er eine Übersichten zu seinen eigenen Veranstaltungen sowie zu alle Reservierungen seiner Veranstaltungen ansehen.

Die Webanwendung ist nach dem Model-View-Controller-Prinzip aufgebaut. Die Models sind hierbei die Entitäten und die Services (DAOs). Sie regeln die Geschäftslogik. Die XHTML-Seiten und die Beans bilden die Views der Anwendung ab. In ihnen ist die Struktur der Oberfläche hinterlegt. Als Controller besitzt die Webanwendung verschiedene Converter und Validator, sowie die Klasse Security, in welcher beispielsweise der angemeldete Benutzer für die aktuelle Session hinterlegt wird. Über diese Controller kann der Programmfluss der Anwendung gesteuert werden.

# Umsetzung der User-Stories

## Anlegen einer neuen Veranstaltung (Manager)

|  |  |  |
| --- | --- | --- |
| Komponente/Klasse | Aufgabe | Zugehörigkeit |
| Security | In ihr ist der angemeldete Benutzer gespeichert. Über die Klasse Security wird abgefragt, ob der Benutzer ein Manager ist | Controller, Session-übergreifend |
| Models (Ticket, Veranstaltung, Nutzer) | Die Models bilden die notwendigen Tabellen der Datenbank ab. (s. ER-Model) | Models |
| DAOs (TicketDAO, VeranstaltungDAO) | Über die DAOs werden die Tabellen der Datenbank mit den Daten befüllt | Services |
| MinimaleAnzahlTickets-Validator | Ein Validator, über die die Anzahl der Tickets überprüft wird (>0) | Validatoren |
| DateValidator | Ein Validator, über den das Datum der Veranstaltung überprüft wird (darf nicht vor dem aktuellen Datum liegen) | Validatoren |
| VeranstaltungRequest | Eine Bean, über die die Veranstaltungsdaten der Oberfläche in die Services übergeben werden | View, Beans |
| neue\_veranstaltung.xhtml | Die xhtml-Seite, in welcher die Oberflächenkomponenten enthalten sind | View |

Schritte zur Anwendung der Story:

1. Login/Registrierung des Benutzers als Manager
2. Im Menü auf "Neue Veranstaltung" klicken
3. Die geforderten Werte (Titel, Beschreibung, Datum, Uhrzeit, Ort, Anzahl der Tickets, Ticketpreis) eingeben.
4. Auswahl "Veröffentlichen" (optional)
5. Klicken auf "Speichern".

## Veröffentlichen einer Veranstaltung (Manager)

für Variante 1: siehe 4.1

|  |  |  |
| --- | --- | --- |
| Komponente/Klasse | Aufgabe | Zugehörigkeit |
| Securtity | In ihr ist der angemeldete Benutzer gespeichert. Über die Klasse Security wird abgefragt, ob der Benutzer ein Manager ist | Controller, Session-übergreifend |
| Models (Nutzer, Ticket, Veranstaltung) | Die Models bilden die notwendigen Tabellen der Datenbank ab. (s. ER-Model) | Models |
| DAOs (TicketDAO, VeranstaltungDAO) | Über die DAOs werden die Tabellen der Datenbank mit den Daten befüllt | Services |
| MinimaleAnzahlTickets-Validator | Ein Validator, über die die Anzahl der Tickets überprüft wird (>0) | Validatoren |
| DateValidator | Ein Validator, über den das Datum der Veranstaltung überprüft wird (darf nicht vor dem aktuellen Datum liegen) | Validatoren |
| DetailVeranstaltungRequest | Eine Bean, über die die Veranstaltungsdaten der Oberfläche in die Services übergeben werden | Views, Beans |
| EventConverter | Ein Converter, der anhand einer ID die zugehörige Veranstaltung zurückgibt | Converter |
| detail\_veranstaltung.xhtml | Die xhtml-Seite, in welcher die Oberflächenkomponenten enthalten sind | View |

Schritte zur Anwendung der Story:

Variante 1: Neue Veranstaltung

1. Login/Registrierung des Benutzers als Manager
2. Im Menü auf "Neue Veranstaltung" klicken
3. Die geforderten Werte (Titel, Beschreibung, Datum, Uhrzeit, Ort, Anzahl der Tickets, Ticketpreis) eingeben.
4. Auswahl "Veröffentlichen"
5. Klicken auf "Speichern"

Variante 2: Bereits erstellte Veranstaltung

1. Login/Registrierung des Benutzers als Manager
2. Im Menü auf "Veranstaltung suchen" klicken
3. Klick auf Titel der noch nicht veröffentlichten Veranstaltung in "Meine Veranstaltungen"
4. Klicken auf "Bearbeiten"
5. Auswahl "Veröffentlichen"
6. Klicken auf "Speichern"

## Bearbeiten einer nicht veröffentlichten Veranstaltung (Manager)

Das Bearbeiten und Veröffentlichen einer Veranstaltung wurde in der Webanwendung zusammengefasst. Aus diesem Grund wird an dieser Stelle darauf verzichtet, dieselben Klassen ein weiteres Mal zu erläutern.

Schritte zur Anwendung der Story:

1. Login/Registrierung des Benutzers als Manager
2. Im Menü auf "Veranstaltung suchen" klicken
3. Klick auf Titel der noch nicht veröffentlichten Veranstaltung in "Meine Veranstaltungen"
4. Klicken auf "Bearbeiten"
5. Die geforderten Werte (Titel, Beschreibung, Datum, Uhrzeit, Ort, Anzahl der Tickets, Ticketpreis) eingeben
6. Auswahl "Veröffentlichen" (optional)
7. Klicken auf "Speichern"

## Suchen nach einer Veranstaltung

|  |  |  |
| --- | --- | --- |
| Komponente/Klasse | Aufgabe | Zugehörigkeit |
| Veranstaltung | Das Model der Veranstaltung, welches eine Tabellle der Datenbank repräsentiert | Models |
| VeranstaltungDAO | DAO, über welches auf die Daten der Tabelle "veranstaltung" der Datenbank zugegriffen werden kann | Services |
| DateValidator | Ein Validator, über den das Datum der Suche überprüft wird (darf nicht vor dem aktuellen Datum liegen) | Validatoren |
| AnzahlTicketsReservieren-Validator | Ein Validator, über den die Anzahl zu reservierender Tickets | Validatoren |
| VeranstaltungSuchen-Request | Eine Bean, über die die Veranstaltungsdaten der Datenbank an die Oberfläche übergeben werden | Views, Bean |
| veranstaltung\_suchen.xhtml | Die xhtml-Seite, in welcher die Oberflächenkomponenten enthalten sind | Views |

Schritte zur Anwendung der Story:

1. Klicken auf "Veranstaltung suchen"
2. Eingabe eines Suchtextes
3. Einschränkung der Suche über die Felder "von", "bis" und "Anzahl verfügbarer Tickets" (optional)

## Anzeigen von Details zu einer Veranstaltung

|  |  |  |
| --- | --- | --- |
| Komponente/Klasse | Aufgabe | Zugehörigkeit |
| Securtity | In ihr ist der angemeldete Benutzer gespeichert. Über die Klasse Security wird abgefragt, ob der Benutzer ein Manager ist. Nur ein Manager kann ein Event bearbeiten (wenn es von ihm erstellt, aber noch nicht veröffentlicht wurde). Nur ein angemeldeter Benutzer kann Tickets reservieren. | Controller, Session-übergreifend |
| Veranstaltung | Das Model der Veranstaltung, welches eine Tabellle der Datenbank repräsentiert | Models |
| VeranstaltungDAO | DAO, über welches auf die Daten der Tabelle "veranstaltung" der Datenbank zugegriffen werden kann | Services |
| DetailVeranstaltungRequest | Eine Bean, in welcher die Daten der ausgewählten Veranstaltung an die Oberfläche übergeben werden. | Views, Beans |
| detail\_veranstaltung.xhtml | Die xhtml-Seite, in welcher die Oberflächenkomponenten enthalten sind | Views |

Schritte zur Anwendung der Story:

1. Klick auf Titel der Veranstaltung in "Meine Veranstaltungen", "Veranstaltung suchen" oder "Home"

## Erhalten eines eindeutigen Reservierungscodes bei der Reservierung

|  |  |  |
| --- | --- | --- |
| Komponente/Klasse | Aufgabe | Zugehörigkeit |
| Security | In ihr ist der angemeldete Benutzer gespeichert. Über die Klasse Security wird abgefragt, ob der Benutzer ein Manager ist | Controller, Session-übergreifend |
| Models (Nutzer, Veranstaltung, Ticket, Reservierung) | Die Models bilden die notwendigen Tabellen der Datenbank ab. (s. ER-Model) | Models |
| DAOs (VeranstaltungDAO, TicketDAO, ReservierungDAO) | Über die DAOs werden die Tabellen der Datenbank mit den Daten befüllt | Services |
| AnzahlTicketsReservierenValidator | Ein Validator, über den die Anzahl zu reservierender Tickets | Validatoren |
| ReservierungenRequest | Eine Bean, überwelche die Daten aller Reservierungen eines Benutzers an die Oberfläche übergeben werden | Views, Beans |
| meine\_reservierungen.xhtml | Die xhtml-Seite, in welcher die Oberflächenkomponenten enthalten sind | Views |
| DetailVeranstaltungRequest | Eine Bean, in welcher die Daten der ausgewählten Veranstaltung an die Oberfläche übergeben werden. Zusätzlich kann eine bestimmte Anzahl Tickets reserviert werden. | Views, Beans |
| detail\_veranstaltung.xhtml | Die xhtml-Seite, in welcher die Oberflächenkomponenten enthalten sind | Views |
| VeranstaltungSuchenRequest | Eine Bean, über die die gesuchten Veranstaltungsdaten der Datenbank an die Oberfläche übergeben werden. Zusätzlich kann eine bestimmte Anzahl Tickets für jede Veranstaltung reserviert werden. | Views, Bean |
| veranstaltung\_suchen.xhtml | Die xhtml-Seite, in welcher die Oberflächenkomponenten enthalten sind | Views |

Schritte zur Anwendung der Story:

Variante 1:

1. Login/Registrierung des Benutzers
2. Suchen der Veranstaltung in "Veranstaltung suchen"
3. Eingabe der Anzahl Tickets
4. Klicken auf "Reservieren"

Variante 2:

1. Login/Registrierung des Benutzers
2. Klick auf Titel der Veranstaltung in "Meine Veranstaltungen", "Veranstaltung suchen" oder "Home" (Veranstaltung muss veröffentlicht sein)
3. Eingabe der Anzahl Tickets
4. Klicken auf "Reservieren"

## Anzeigen einer Liste aller Reservierungen (Manager)

|  |  |  |
| --- | --- | --- |
| Komponente/Klasse | Aufgabe | Zugehörigkeit |
| Security | In ihr ist der angemeldete Benutzer gespeichert. Über die Klasse Security wird abgefragt, ob der Benutzer ein Manager ist | Controller, Session-übergreifend |
| Models (Nutzer, Veranstaltung, Ticket, Reservierung) | Die Models bilden die notwendigen Tabellen der Datenbank ab. (s. ER-Model) | Models |
| DAOs (VeranstaltungDAO, TicketDAO, ReservierungDAO) | Über die DAOs werden die Tabellen der Datenbank mit den Daten befüllt | Services |
| ManagerRequest | Eine Bean, in welcher die Daten der Veranstaltungen des Managers und die Daten der Reservierungen an die Oberfläche übergeben werden. | Views, Beans |
| manager.xhtml | Die xhtml-Seite, in welcher die Oberflächenkomponenten enthalten sind | Views |

Schritte zur Anwendung der Story:

1. Login/Registrierung des Benutzers als Manager
2. Im Menü auf "Meine Veranstaltungen" klicken
3. Auf "Reservierungen" klicken

## Anzeige noch verfügbarer Tickets

|  |  |  |
| --- | --- | --- |
| Komponente/Klasse | Aufgabe | Zugehörigkeit |
| Models (Ticket, Veranstaltung) | Die Models bilden die notwendigen Tabellen der Datenbank ab. (s. ER-Model) | Models |
| VeranstaltungDAO | Über die DAOs werden die Tabellen der Datenbank mit den Daten befüllt | Services |
| DetailVeranstaltungRequest | Eine Bean, in welcher die Daten der ausgewählten Veranstaltung an die Oberfläche übergeben werden. | Views, Beans |
| Detail\_veranstaltung.xhtml | Die xhtml-Seite, in welcher die Oberflächenkomponenten enthalten sind | Views |
| HomeRequest | Eine Bean, in welcher die Daten der fünf neuesten Veranstaltungen, sowie die aller Veranstaltungen in Form von Listen an die Oberfläche übergeben werden | Views, Beans |
| home.xhtml | Die xhtml-Seite, in welcher die Oberflächenkomponenten enthalten sind | Views |
| ManagerRequest | Eine Bean, in welcher die Daten der Veranstaltungen des Managers und die Daten der Reservierungen an die Oberfläche übergeben werden. | Views, Beans |
| manager.xhtml | Die xhtml-Seite, in welcher die Oberflächenkomponenten enthalten sind | Views |
| VeranstaltungSuchenRequest | Eine Bean, über die die gesuchten Veranstaltungsdaten der Datenbank an die Oberfläche übergeben werden. | Views, Bean |
| veranstaltung\_suchen.xhtml | Die xhtml-Seite, in welcher die Oberflächenkomponenten enthalten sind | Views |

Schritte zur Anwendung der Story:

1. Enthalten in den Listen der "Home"- "Veranstaltung suchen"- und "Meine Veranstaltungen"-Seiten
2. Enthalten in der Detail-Ansicht einer Veranstaltung

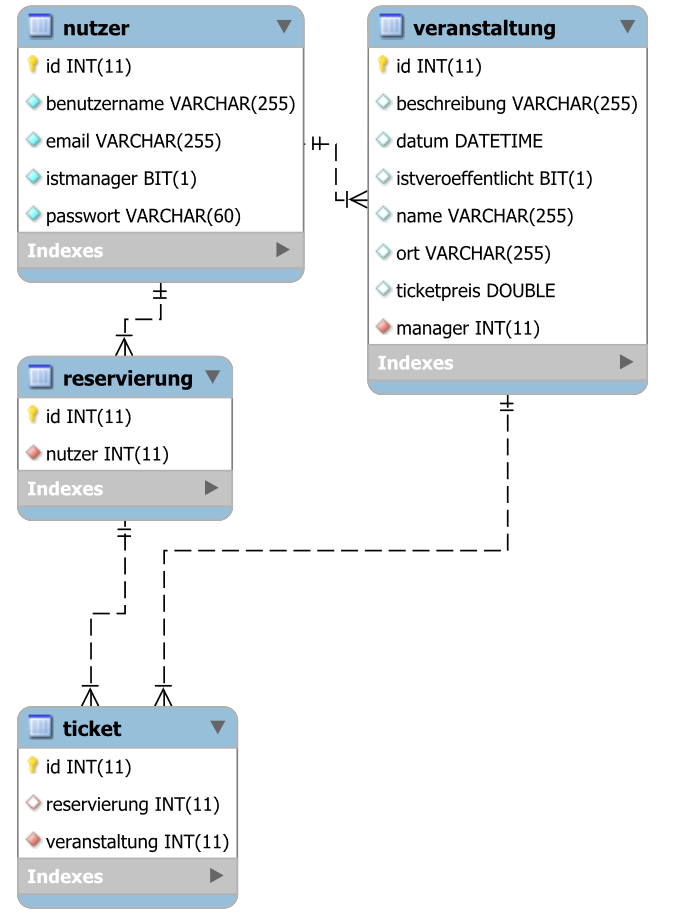
# Sprint-Backlog

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Beschreibung | Story | Entwickler |
| Hibernate Einbindung | Einbindung von Hibernate, Erstellen der notwendigen persistence.xml | 1-8 | Jan Raupach |
| Erstellen der Entities | Entwicklung der Models mit den passenden Annotations | 1-8 | Jan Raupach |
| Erstellen der DAOs | Entwicklung der DAOs (NutzerDAO, TicketDAO, ReservierungDAO, VeranstaltungDAO), Reagieren auf Anforderungen aus den User-Stories | 1-8 | Jan Raupach |
| Entwicklung des Logins und Logouts | Entwicklung der XHTML-Seite, Entwicklung des UserServices, Entwicklung des UserConverters, Entwicklung der Bean, Entwicklung der Klasse Security | 1-8 | Ann-Katrin Kottig |
| Entwicklung der Registrierung | Entwicklung der XHTML-Seite, Entwicklung der Bean | 1-8 | Ann-Katrin Kottig |
| Entwicklung der Startseite | Entwicklung der XHTML-Seite, Entwicklung der Bean | - (z.T. 5; 8) | Ann-Katrin Kottig |
| Entwicklung der "Veranstaltung-Detail"-Seite | Entwicklung der XHTML-Seite,  Entwicklung der Bean, Entwicklung des VeranstaltungConverters | 2; 3; 5; 8 | Ann-Katrin Kottig |
| Entwicklung der "Veranstaltung suchen"-Seite | Entwicklung der XHTML-Seite,  Entwicklung der Bean, Einbindung des Frameworks Hibernate Search | 4; 8 | Jan Raupach |
| Entwicklung der "neue Veranstaltung"-Seite | Entwicklung der XHTML-Seite,  Entwicklung der Bean | 1; 2 | Jan Raupach |
| Entwicklung der "Meine Veranstaltungen"-Seite | Entwicklung der XHTML-Seite,  Entwicklung der Bean | 7 | Ann-Katrin Kottig |
| Entwicklung der "Meine Reservierungen"-Seite | Entwicklung der XHTML-Seite,  Entwicklung der Bean | - | Ann-Katrin Kottig |
| Messages einbinden | Einfügen von FacesMessages für jede Seite mit Input-Feldern | 1-8 | Ann-Katrin Kottig  Jan Raupach |
| Validatoren einbinden | Entwicklung und Einbindung der Validatoren | 1-8 | Jan Raupach |
| Testen | Testen aller User-Stories, Korrektur von Fehlern | 1-8 | Ann-Katrin Kottig  Jan Raupach |

Hinweis: Die Nummern der User-Stories entsprechen den Kapitelnummern des Vorangegangenen Kapitels.

# Datenmodell

## ER-Diagramm



## Klassendiagramm