# UFO

**Ultimate Festival Organizer**

Stefan Kert

22. Januar 2016

## 1 Übersicht

### 1.1 Lösungsidee

Es sollte eine Weboberfläche entwickelt werden, mit welcher es auf möglichst einfache Art und Weise möglich ist, Informationen zu verschiedenen Veranstaltungen beim Pfasterspektakel in Linz zu bekommen. Dabei sollte dem Benutzer die Möglichkeit geborden werden, nach einzelnen Artisten, Spielorte oder Veranstaltungen zu suchen. Die Daten werden dabei von dem während des SWK Projektes entwickelten Webservice zur Verfügung gestellt werden. Dieser stellt die Daten im JSON Format über einen REST Service zur Verfügung.

### 1.2 Architektur

Die Anwendung ist mittels JSF umgesetzt und in 3 Teile gegliedert. Dabei gibt es einerseits die Datenzugriffsschicht, über welche die Daten vom Webservice konsumiert werden, bzw. dort hingeschickt werden. Diese Datenzugriffsschicht wird über eigene Beans angesteuert, welche schließlich die Daten für das Userinterface zur Verfügung stellen. In diesen Beans werden weiters die Aktionen implementiert, welche vom Benutzer getätigt werden können. Das Userinterface wird mit JSF Komponenten realisiert und als solches basiert es auf den Frameworks *PrimeFaces*, *PrimeFaces.Extensions* und *BootsFaces*. Diese sind vor allem hinsichtlich Aussehen und Usability sehr gut für den Benutzer zu verwenden und stellen daher einen modernen Lösungsansatz dar.

Der Zugriff der Beans auf die Datenzugriffsschicht wird mittels Dependency Injection realisiert. Die einzelnen Proxies, welche für den Zugriff auf den Webservice verwendet werden, sind mit der *@ApplicationScoped* Annotation gekennzeichnet und können somit als sogenannte *@ManagedProperty* in eine andere Klasse iniziiert werden.

Die Beans, welche wiederum als *@ManagedBean* gekennzeichnet sind, werden dabei mit dem Attribut *@ViewScoped* gekennzeichnet, sodass die Daten bei jedem Aufruf der Seite neu geladen werden und somit sichergestellt werden kann, dass diese aktuell sind. Es könnte hier auch noch eine Art Cachingmechanismus in der Datenzugriffsschicht implementiert werden, für diese Version der Implementierung wurde aber darauf verzichtet.

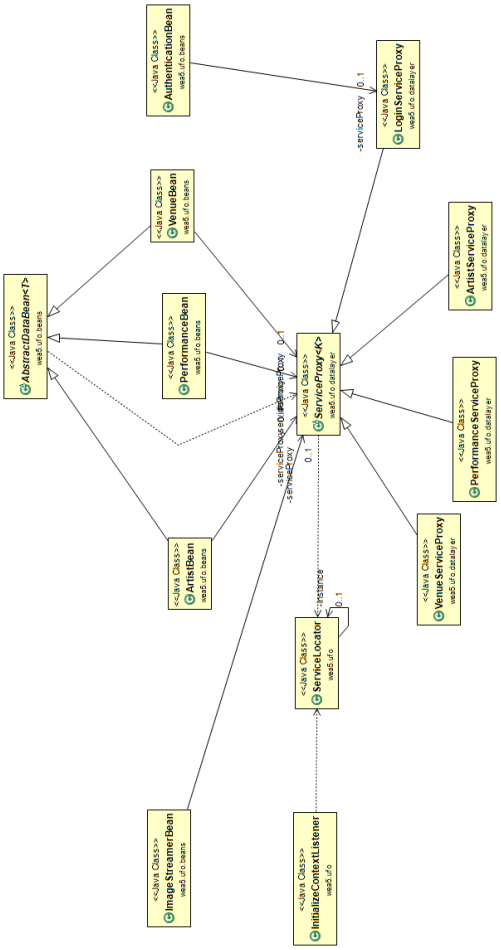


Abbildung 1: UML-Klassendiagramm

In der Grafik 1 wird das UML Klassendiagramm der wichtigsten Teile der Architektur dargestellt. Anhand dieser Grafik kann man gut erkennen wie die Abhängikeiten zwischen den einzelnen Klassen sind.

In der Grafik 1 werden weiters sämtliche Attribute und Methoden der einzelnen Klassen aufgelistet. In diesem Diagramm kann man sehr gut erkennen, dass die Klassen bis auf einige Attribute und Methoden sehr klein gehalten wurden und viele Methoden von den Basisklassen AbstractDataBean und dem ServiceProxy geerbt wurden und als solches verwendet werden.

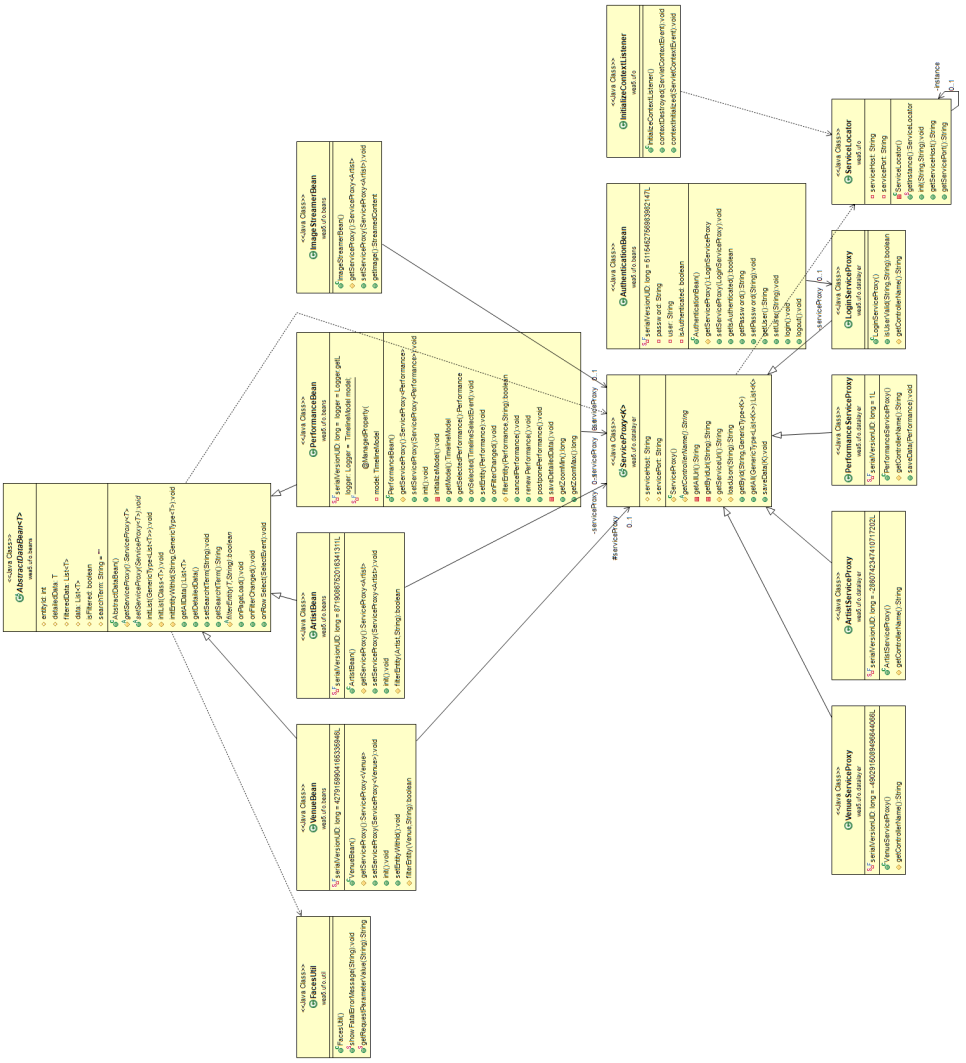

Abbildung 2: UML-Klassendiagramm

## 1.3 Frontend

Das Userinterface gliedert sich in Header und Content. Im Header sind die Links zu den einzelnen Seiten, sowie eine Suche die auf Elemente in der aktuell geöffneten Seite filtert, und die Möglichkeit zum Login für Benutzer.

Die Links führen zu 3 verschiedenen Seiten. Die Ansicht für Veranstaltungen, die Anischt für alle Künstler und die Ansicht für alle Spielorte. Dabei sind die Ansicht für Künstler und die Ansicht für Spielorte sehr ähnlich. Die einzelnen Elemente werden in einer übersichtlichen Tabelle dargestellt, diese Tabelle kann über das Suchfeld im Header gefiltert werden. Bei selektieren einer Zeile (Künstler, Spielort) kann auf diesen geklickt werden und es öffnet sich ein Detailfenster. Bei Künstlern wird dabei ein Fenster geöffnet, welches Informationen für diesen bzw. die Daten für die unterschiedlichen Veranstaltungen anzeigt. Beim Detailfenster für den Spielort wird der Name des Spielorts und eine Google Maps Karte dargestellt.

Ein weiteres sehr grundlegendes Element ist die Übersichtsseite für die verschiedenen Veranstaltungen. Diese Seite enthält ein *Timeline*-Control (PrimeFaces-Extensions), welches im Prinzip einen Kalender darstellt. Auf diesem sind die einzelnen Veranstaltungen gruppiert nach ihren Spielorten aufgelistet. Die Timeline kann verschoben werden, es kann vergrößert bzw. verkleinert werden. Bei Klick auf eine Veranstaltung öffnet sich ein Detailfenster mit näheren Informationen zu der Veranstaltung, zum Künstler und zum Spielort. Wie auch bei den anderen Seiten funktioniert können auch auf der Veranstaltungsseite die unterschiedlichen Veranstaltungen gefiltert werden. Dabei wird der im Suchfeld eingegeben Text mit den Namen der Künstler, dem Namen der Veranstaltung, dem Namen der Kategorie und dem Namen des Spielorts abgeglichen.

Im Details Dialog hat der Benutzer, falls er gültig authentifiziert wurde die Möglichkeit Veranstaltungen abzubrechen, zu verschieben oder wieder zu reaktivieren.

## 1.4 Konfiguration

Über die Kontextparameter *SERVICE_HOST* und *SERVICE_PORT* in der web.xml sind die Einstellungen für den Webservice konfigurierbar.

# 2 Anwendungsfälle

Im Folgenden werden zwei typische Anwendungsfälle der Webanwendung beschrieben. Zum bessren Verständnis werden diese mit Sequenzdiagrammen dargestellt. Die Abtrennung zwischen dem C# Webservice und dem JSF Teil wird durch einen roten Strich gekennzeichnet

## 2.1 Öffnen der Veranstaltungsansicht

Beim Öffnen der Veranstaltungsansicht (index.xhtml) werden alle Veranstaltungen vom Webserivce abgefragt. Hierzu werde alle Veranstaltungen mittels Get Request über den PerformanceProxy vom PerformanceDataController abgefragt und dieser übernimmt dann die Aufgabe, alle Vorstellungen aus der Datenbank zu lesen. Diese werden schließlich zurückgegeben und JSON serialisiert. Im PerformanceProxy werden diese wieder deserialisiert und danach weitergegeben. Die Performances werden gruppiert und in ein TimeModel gespeichert. Dieses wird din der View aktualisiert und der Benutzer kann diese durchsuchen.

### Sequenzdiagramm

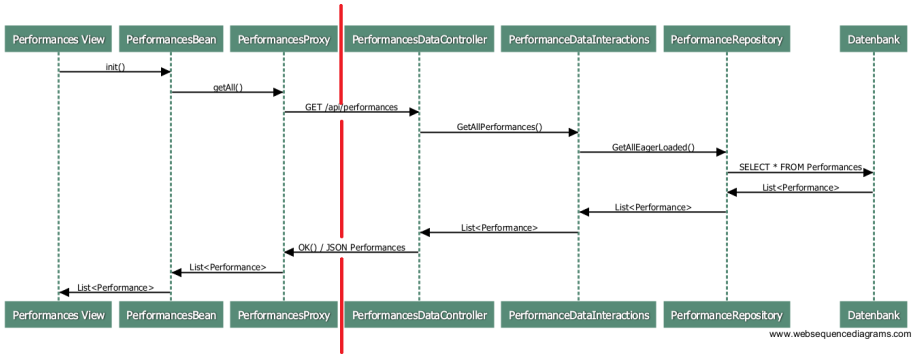


Abbildung 3: Sequenzdiagramm der JSF-Komponente beim initialisieren der Veranstaltungsansicht.

## 2.2 Abbrechen einer Veranstaltung

Beim Klicken des Abbrechenbuttons auf der Detailseite der Veranstaltungen wird die Methode cancelPerformance im PerformanceBean aufgerufen, welche das Canceled Flag der ausgewählten Veranstaltung auf true setzt. Anschließend wird diese Veranstaltung über den Proxy serialisiert und als JSON an den Webservice übertragen. Dort werden die JSON Daten wieder deserialisiert und schließlich in der Datenbank gespeichert. Danach wird bei Erfolg eine Ok() Meldung zurückgegeben und schließlich wird im PerformanceBean die TimeLine aktualisiert, was sich schließlich auf die Oberfläche auswirkt.

### Sequenzdiagramme

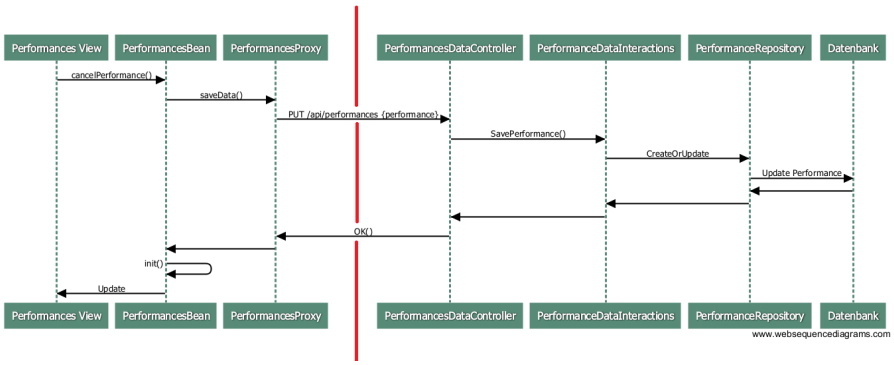


Abbildung 4: Sequenzdiagramm Abbrechen einer Veranstaltung.