

2016

Ultimate Festival Organizer

SWK-5 / WEA5

WOLFGANG LUMETSBERGER

Allgemeines:

Ultimate Festival Organizer (UFO), ist eine Applikation, welche im Rahmen eines Semesterprojekts in den Fächern SQK5 und WEA5 realisiert wird. Eine detaillierte Beschreibung von Anforderungen können der Angabe entnommen werden.

Systemaufbau:

Das System wird grob in folgende Teile unterteilt:

- UFO-Server
- UFO-Commander
- UFO-WebService
- UFO-Web

UFO-Server:

Ist die zentrale Komponente zur Verwaltung von Künstlern, Spielorten und Tagesprogrammen. Im Wesentlichen beinhaltet dieser also die Gesamtheit der Business-Logik. Auch die Datenbankzugriffe erfolgen über diese Schicht.

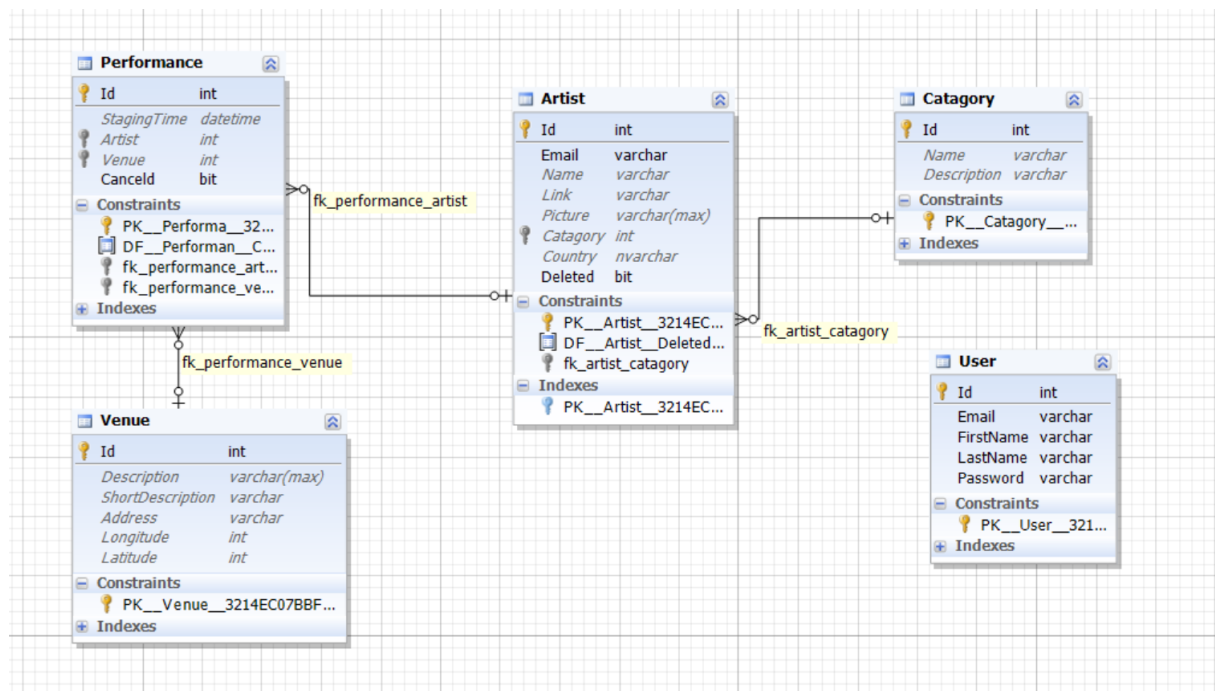
Der UFO-Server wurde in folgende Projekte unterteilt:

- UltimateFestivalOrganizer.DAL.Common
- UltimateFestivalOrganizer.DAL.SqlServer
- UltimateFestivalOrganizer.DAL.Test
- UltimateFestivalOrganizer.BusinessLogik

Weiters wird im UFO-Server die Datenbank selbst, welche eine SQL-Server-LocalDb (file) ist, verwaltet.

Datenbank:

Wie bereits erwähnt wird eine SQL-Server Local-DB verwendet. Das Datenbankschema wurde wie folgt erstellt:



Es wurden Surrogates zu jeder Entität hinzugefügt. Dies erleichtert später das Handling in der Applikation, und wir können uns Code sparen und Fehleranfälligkeit reduzieren. Die realen Keys werden nicht desto trotz als Unique Constraints abgebildet.

Datenzugriffsschicht:

Diese Schicht wird von 2 Projekten repräsentiert. DAL.Common, DAL.SqlServer.

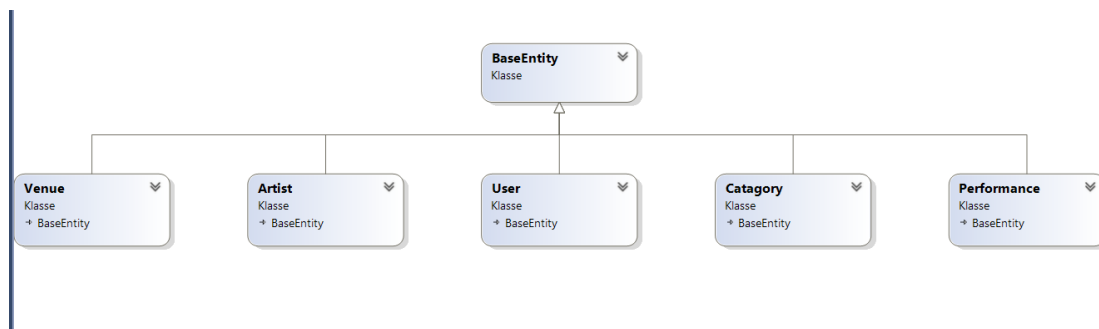
Im Projekt Common werden dabei die Entitäten verwaltet, sowie jeweils ein Interface in welchem definiert wird, welche Zugriffs-Methoden möglich sind. Im Projekt SqlServer wird diese Zugriffsschicht dann auch wirklich konkret für unsere Datenbank programmiert. Über eine Factory-Klasse und der app.conf kann dann festgelegt werden, welche Ausprägung für das Interface verwendet werden soll. Da wir eben nur den für einen SQL-Server programmiert haben, können wir auch nur diesen

verwenden. Würden wir zum Beispiel auf eine Oracle-DB wechseln, könnten wir in einem neuen Projekt die DAO Interfaces nun ausprogrammieren und dieses dann in der AppConfig konfigurieren.

POCOS:

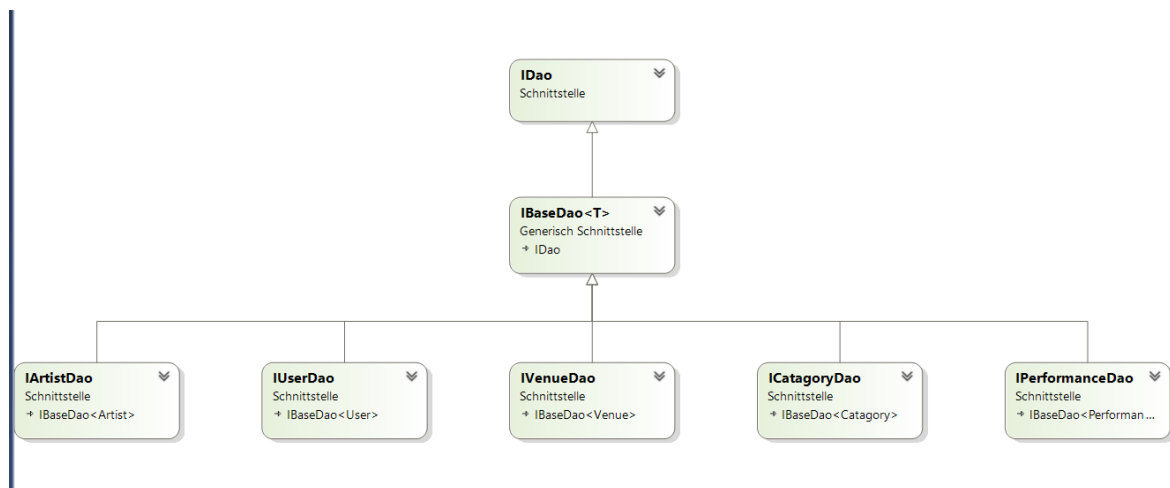
Für jede Entität wurde ein POCO erstellt. Um später mit einem DAO dynamisch Queries erzeugen zu können, gelten folgende Konventionen für die POCOS:

- Jedes POCO leitet von der Basis-Klasse BaseEntity ab.
- Jedes POCO enthält als KlassenAttribut [Table] , in welchem der Name für die Datenbanktabelle angegeben werden kann.

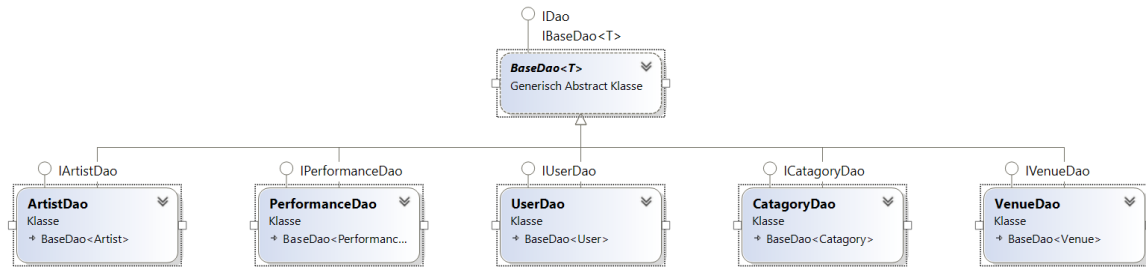


DAOS:

Im Package Common werden die Interfaces für die DAOs deklariert. Es gilt, das jedes erstellte DAO muss vom Interface IDao und IBaseDao ableiten. Später kann in der Implementierung dann Basis Funktionalität die immer benötigt wird, schon im BaseDao implementiert werden.



DAOS (Implementierung im DAL.SqlServer):



Die Klasse BaseDao implementiert alle benötigten Grundfunktionalitäten wie findByKey, delete, update, save ...

Diese Funktionalitäten, müssen in den einzelnen DAOS also nicht mehr ausprogrammiert werden. In der Schnittstelle DAO ist lediglich so programmiert, dass sie jeweils einfach die Funktionalität von IBaseDao aufruft. Es wird jedoch benötigt, um ein Mappen von Foreignkeys erfolgreich durchführen zu können. Beispiel:

```

public IList<T> findAll () { DbCommand command = database .CreateCommand( this .
GetDefaultSelect ()); using (IDataReader reader = database . ExecuteReader(command)) { return
ConvertResultToList ( reader ); } } object IDao . findAll () { return findAll (); }
  
```

Tests:

Um die DAL-Schicht und den Server auf Korrektheit zu überprüfen, wurden Unit-Tests und Integration-Tests geschrieben. Es wurden folgende Ergebnisse erzielt:

The screenshot shows the Visual Studio interface with the Test Explorer on the left and the Code Coverage window at the bottom. The Test Explorer displays a list of tests grouped by category, all of which passed successfully. The Code Coverage window shows the coverage percentage for the project hierarchy.

Test Explorer Results:

- ArtistDaoTest (6)**
 - TestFindAll: 289 ms
 - TestFindByld: 91 ms
 - TestFindByldAndCheckFetchCatagory: 85 ms
 - TestFindByUniqueProperty: 90 ms
 - TestInsert: 100 ms
 - TestUpdate: 100 ms
- CatagoryDaoTest (4)**
 - TestFindAll: 104 ms
 - TestFindByld: 90 ms
 - TestInsert: 91 ms
 - TestUpdate: 96 ms
- PerformanceDaoTest (4)**
 - TestFindAll: 109 ms
 - TestFindByld: 113 ms
 - TestInsert: 141 ms
 - TestUpdate: 98 ms
- UserDaoTest (5)**
 - TestFindAll: 82 ms
 - TestFindByld: 80 ms
 - TestFindByUniqueProperty: 85 ms
 - TestInsert: 85 ms
 - TestUpdate: 83 ms
- VenueDaoTest (4)**
 - TestFindAll: 101 ms
 - TestFindByld: 113 ms
 - TestInsert: 142 ms
 - TestUpdate: 107 ms

Zusammenfassung
Letzter Testlauf Erfolgreiche (Gesamte Laufzeit 0:00:18)
 23 Tests Erfolgreiche

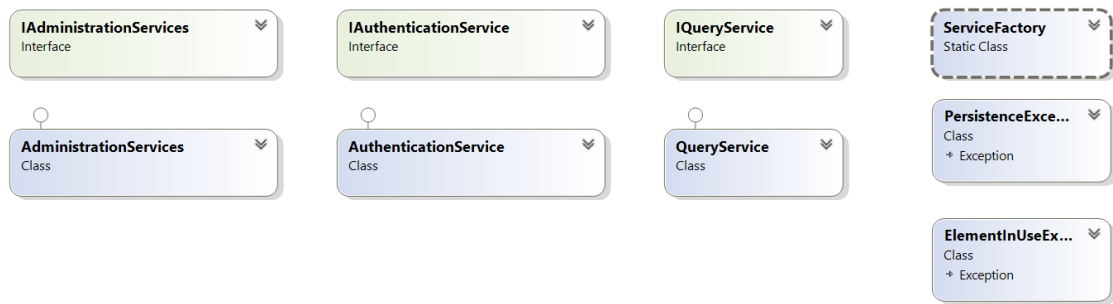
Codeabdeckungsergebnisse

Hierarchie	Nicht abgedeckt (Blöcke)	Nicht abgedeckt (% Blöcke)	Abgedeckt (Blöcke)	Abgedeckt (% Blöcke)
Wolfgang_WOLFGANG-PC 2015-11-20 14:...	57	7,07 %	749	92,93 %
ultimatefestivalorganizer.dal.c...	13	8,02 %	149	91,98 %
ultimatefestivalorganizer.dal.s...	44	12,61 %	305	87,39 %
ultimatefestivalorganizer.dal.t...	0	0,00 %	295	100,00 %

Business-Logik:

Diese ist im Projekt UltimateFestivalOrganizer.BusinessLogik implementiert. Sie hat nur eine Abhängigkeit auf das Projekt DAL.Common, nicht jedoch auf DAL.SqlServer.

In dieser Schicht wird das Abbilden von Speicher, Bearbeiten, holen und validieren von Daten realisiert. Es werden dazu 3 Interfaces erstellt. IAdministrationService, IAuthenticationService, und IQueryService. Im QueryService werden alle jene datenbankzugriffe durchgeführt, welche nur aus der Datenbank lesen. Die Authentifizierung von einem Benutzer erfolgt im AuthenticationService. Das Administration Service hat in der Implementierung eine Abhängigkeit auf die beiden anderen Services. Es wird vom Commander in den ViewModels verwendet, weshalb ich es als sinnvoll empfand dort Funktionalitäten zusammenzufassen.



UFO-Commander:

Der Commander wird durch eine WPF-Applikation repräsentiert, und wird dazu verwendet die Administration durchführen zu können. Die Kommunikation erfolgt dabei nicht über das Webservice, sondern es kann direkt auf die Komponenten des Servers zugegriffen werden.

Dieses Projekt hat eine Abhängigkeit zum Common, und zur BusinessLogik. In der MainView wird ein neues AdministrationServicer erzeugt, welches dann an alle anderen ViewModels per Konstruktor übergeben wird. Schöner wäre es noch, wenn man die Objekte über Dependency Injection lösen würde.

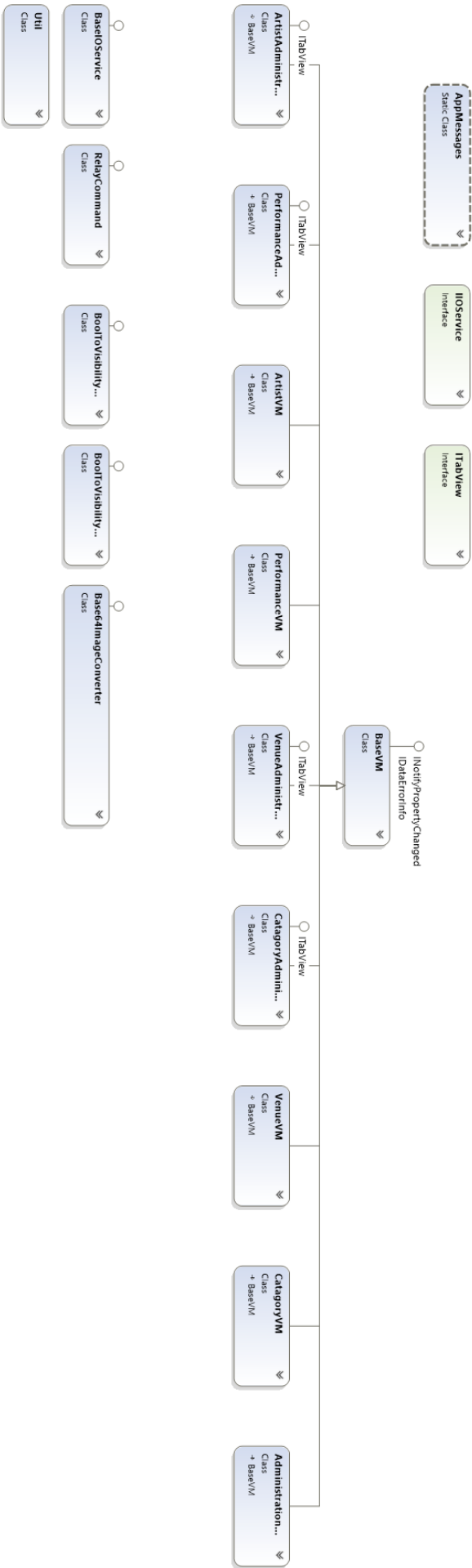
Die Kommunikation zwischen den ViewModels funktioniert über Messaging. Diese Methode liefert das MVVM-Light Package, welches aus diesem Grund hinzugefügt wurde. In einer Klasse AppMessages befinden sich alle möglichen Messages, sowie Hilfsmethoden zum Registrieren um auf bestimmte Messages zu hören, oder um eine Message zu feuern.

Zur Validierung wird das Framework MVVM-Validation eingesetzt. Dazu kann man im BaseModel IDataErrorInfo implementieren, danach werden Fehler korrekt dargestellt.

Es gibt in meiner Implementierung bis auf ein einmaliges Setzen des DataContexts im MainView, sowie ein Rendern von einer Message Box keine Code behind.

Der FileDialog wird über ein Service angesprochen.

Das Senden von Mails kann in der App.config konfiguriert werden. Damit die Mails nicht direkt versendet werden, sondern einfach in einen Ordner gespeichert werden. Dies wurde so konfiguriert, um das Testen zu beschleunigen.



*Benutzeroberfläche:**Login:*

Gültige Logindaten um sich einzuloggen sind admin@test.at / admin. Gibt man ungültige Daten ein, bleibt man am gleichen Screen und ein Hinweis wird angezeigt.



UserId:	<input type="text" value="admin@test.at"/>
Password:	<input type="password" value="••••"/>
<input type="button" value="LOGIN"/>	

Artists:

Hier können Artisten bearbeitet, gelöscht oder hinzugefügt werden. Es können nur jene Artists gelöscht werden, welche keiner Aufführung zugeordnet sind.

ADMINISTRATIONWINDOW

Artists Catagories Venues Performances

+

📁

🗑️

Klassik

Larry Page

sandra riley

audrey hughes

roberto medina

reginald daniels

dddddd

ffffff

asdfasdfsdfas

Catagory 3

Marry Page

gilbert phillips

ramona mendoza

tyler lewis

jonathan owens

adam bell

brett robinson

Catagory 2

TO DELETE

beverley turner

alexander gray

ralph sullivan

Name:

Larry Page

Email:

l.page@gmx.com

Link:


asdf

Country:

Catagory:

Klassik

Vorschau-Bild:



Catagories:

Hier können Kategorien verwaltet werden. Wird eine hinzugefügt oder gelöscht, ist diese sofort bei einem Wechsel auf eine andere Ansicht zum Beispiel Artist verfügbar.

ADMINISTRATIONWINDOW

Artists Catagories Venues Performances

+

Klassik

Sport

Category 1

Category 2

Category 3

Category 4

Category 5

Category 6

Category 7

Category 8

Category 9

Category 10

Category 1

Category 2

Category 3

Category 4

Category 5

Category 6

Category 7

Category 8

Category 9

Name:

Description:

Klassik

Klassik

Venues:

Hier können die Spielstätten verwaltet werden. Die Koordinaten z.B. 14,3000 müssen *1000000 gerechnet werden, und sind ohne kommastelle einzugeben.

ADMINISTRATIONWINDOW

Artists Catagories Venues Performances

This is Random Stage No 0

This is Random Stage No 2

This is Random Stage No 1

This is Random Stage No 1

This is Random Stage No 1

This is Random Stage No 1

This is Random Stage No 1

This is Random Stage No 1

This is Random Stage No 1

This is Random Stage No 1

This is Random Stage No 1

This is Random Stage No 1

This is Random Stage No 1

This is Random Stage No 1

This is Random Stage No 1

This is Random Stage No 1

This is Random Stage No 1

This is Random Stage No 1

This is Random Stage No 1

This is Random Stage No 1

This is Random Stage No 1

Description:

Short Description:

Adress:

Longitude:

Latitude:

This is Random Stage No 0

Stage1

asdf

1

1

Performances:

Hier kann das Tages-Programm erstellt und Validiert werden. Um einen neuen Auftritt hinzuzufügen, kann einfach im Dropdown zur richtigen Bühne und richtigen Uhrzeit ausgewählt werden. Es wird dabei sofort auf Validität geprüft. Speichern erfolgt über das Speichern Symbol. Das Häckchen dient dazu, das gesamte Tagesprogramm noch einmal zu validieren. Mit dem Mail-Symbol werden all jene Künstler benachrichtigt, welche im Tagesprogramm vorkommen. Einem Artist alleine kann ein Email gesendet werden, indem auf das Mail-Symbol bei seinem Auftritt gedrückt wird.

ADMINISTRATIONWINDOW

Artists Catagories Venues Performances

☒ ☐ ☒ 20.12.2015 ☐

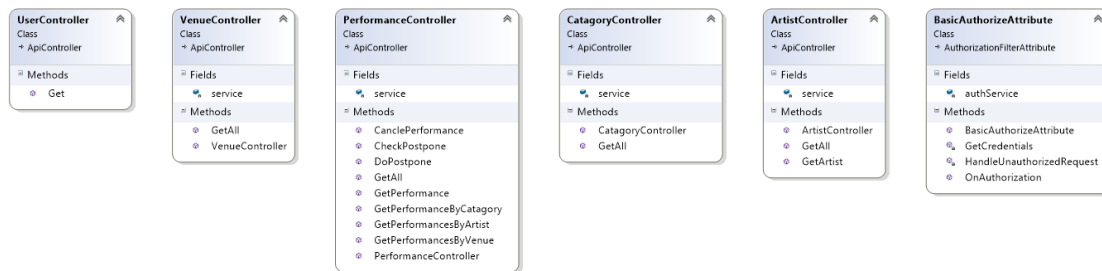
Venues	14-15	15-16	16-17	17-18	18-19
This is Random Stage No 0	<div>ARG Klassik ddddd</div> <div>ddddd</div> <div><input type="checkbox"/> <input type="checkbox"/></div>	<div>Klassik asdfasdfas</div> <div>asdfasdfas</div> <div><input type="checkbox"/> <input type="checkbox"/></div>	<div></div> <div></div> <div><input type="checkbox"/> <input type="checkbox"/></div>	<div></div> <div></div> <div><input type="checkbox"/> <input type="checkbox"/></div>	<div></div> <div></div> <div><input type="checkbox"/> <input type="checkbox"/></div>
This is Random Stage No 2	<div>AUT Catagory 1 isaac sims</div> <div>isaac sims</div> <div><input type="checkbox"/> <input type="checkbox"/></div>	<div>AUT Catagory 5 ernest moore</div> <div>ernest moore</div> <div><input type="checkbox"/> <input checked="" type="checkbox"/></div>	<div>AUT Klassik sandra riley</div> <div>sandra riley</div> <div><input type="checkbox"/> <input type="checkbox"/></div>	<div>Klassik Larry Page</div> <div>Larry Page</div> <div><input type="checkbox"/> <input type="checkbox"/></div>	<div></div> <div></div> <div><input type="checkbox"/> <input type="checkbox"/></div>
This is Random Stage No 1	<div></div> <div></div> <div><input type="checkbox"/> <input type="checkbox"/></div>	<div>AUT Catagory 6 alicia knight</div> <div>alicia knight</div> <div><input type="checkbox"/> <input checked="" type="checkbox"/></div>	<div>AUT Catagory 1 leon carroll</div> <div>leon carroll</div> <div><input type="checkbox"/> <input type="checkbox"/></div>	<div>AUT Klassik reginald daniels</div> <div>reginald daniels</div> <div><input type="checkbox"/> <input type="checkbox"/></div>	<div></div> <div></div> <div><input type="checkbox"/> <input type="checkbox"/></div>
This is Random Stage No 1	<div></div> <div></div> <div><input type="checkbox"/> <input type="checkbox"/></div>	<div></div> <div></div> <div><input type="checkbox"/> <input type="checkbox"/></div>	<div></div> <div></div> <div><input type="checkbox"/> <input type="checkbox"/></div>	<div></div> <div></div> <div><input type="checkbox"/> <input type="checkbox"/></div>	<div></div> <div></div> <div><input type="checkbox"/> <input type="checkbox"/></div>

UFO-WebService:

Es wird eine Rest-Schnittstelle zur Verfügung gestellt, welche es erlaubt, das Tages-Programm, sowie Künstler und weitere Informationen abzufragen. Weiters könne Performances hier Abgesagt, beziehungsweise verschoben werden, unter der Voraussetzung, dass sich der User mit dem Request erfolgreich authentifiziert.

Dieses Projekt hat die gleichen Abhängigkeiten wie der Commander (Common und BusinessLogik). Mit diesem Rest-Webservice werden alle benötigten Funktion für das Projekt Web zur Verfügung gestellt.

Für die Authentifizierung wird BasicAuthentication verwendet. Dazu wurde ein Attribut verwendet, welches an den Controller Methoden angebracht wird, bei welchem es nötig ist, einen Authentifizierten User zu kennen. Im Controller PerformanceController wird dies beispielsweise bei den Methoden für verschieben und absagen benötigt.



Das Webservice Liefert dabei jeweils immer eine JSON-Response.

UFO-Web:

Stellt eine JSF-Applikation dar, die es dem User ermöglicht ein Tages-Programm, sowie Künstler und deren Informationen abzufragen. Die Kommunikation dieser Applikation mit dem Server erfolgt über das Webservice.

Konfiguration:

Um sich nicht mit diversen Problemen von Projekt und Entwicklungs-umgebungen totschlagen zu müssen, wurde das Projekt als Gradle-Projekt erstellt. Dazu wurden sowohl die Abhängigkeiten in diesem korrekt verwaltet, als auch das korrekte erstellen eines Eclipse-Projektes.

Im Verzeichnis kann mittels gradlew cleanEclipse Eclipse ein neues Eclipse-Projekt für dieses Projekt erstellt werden. Öffnet man dieses dann, sind alle Abhängigkeiten und Facets wie JSF usw. korrekt vorhanden.

Als Server wird ein TomcateEE verwendet, da dieser JSF sowie auch ApacheOpenBeans Libraries korrekt enthält, um Dependency Injection und eben JSF korrekt ausführen zu können.

Ich habe mich entschieden keine Tag-Library wie Icefaces oder Primefaces zu verwenden, sondern die Seite mit Plain-JSF 2.2 zu gestalten.

Die Entscheidung wurde deshalb getroffen, da ich bei meinem derzeitigen Arbeitgeber mit genau diesen zwei Tag-Libraries beschäftigt bin, und ich vertraut bin, wie mühevoll es ist, wenn ein Button nur ein wenig anders aussehen sollte und man eine solche Library benutzt.

Weiters wollte ich herausfinden, wie gut das neue Konzept von JSF 2.2 mit Passthrough Attributen, und Action Tags auf normalen HTML5-Elementen funktioniert.

Um auch ohne TagLibraries, gutes Design in die Applikation zu bekommen, habe ich mich entschieden Twitter-Bootstrap zu verwenden.

Zur Verschönerung des Java-Codes wurde das Projekt Lombok hinzugefügt, welche erlaubt Getter und Setter anhand einer Annotation über der Variable zu definieren.

Interfaces wurden nicht korrekt nach Java-konventionen erstellt, da diese meiner Meinung nach keinen Sinn machen, und mehr verwirren als sie gut machen.

Seitengestaltung:

Es wurde ein BasisTemplate erstellt, auf welchem immer ein Navigations-Teil und ein eigentlicher Content zu sehen ist.

Auf einer Einstiegsseite wird man freundlich begrüßt, und auf zwei weiteren Seiten kann man jeweils Artisten, sowie das Programm ansehen und durchsuchen.

Architektur:

Services:

Mithilfe von Services kann mit dem UFO-WebServer kommuniziert werden. Zu jedem Service gibt es ein Interface, und jedes Service wird einfach per Dependency-Injection angelegt. Dies kann in der Bean z.B. wie folgt erfolgen:

```
@Inject @Named("QueryService")
IQueryService queryService;
```


In einer FactoryMethode wird festgelegt, welches QueryService erzeugt werden soll. Diese Klasse sieht wie folgt aus:

```
@ApplicationScoped
public class ServiceFactory {

    @Inject
    private RestClient client;

    @Produces @Named("QueryService")
    public IQueryService getQueryService() {
        return new QueryService(this.client);
    }

    @Produces @Named("AuthenticationService")
    public IAuthenticationService getAuthenticationService() {
        return new AuthenticationService(this.client);
    }

    @Produces @Named("AdministrationService")
    public IAdministrationService getAdministrationService() {
        return new AdministrationService(this.client);
    }

}
```

Im RestClient findet die eigentliche Kommunikation statt. Über HttpURLConnection wird dabei auf den Server gegangen, und mittels Gson das Json beidseitig geparsed. Ist eine Authentifizierung nötig, so wird diese ebenfalls bei jeder Anfrage beigefügt.

Controller/Beans:

Die Applikation besteht aus 3Beans. Jeweils eine Bean pro View. Die Beans sind ViewScoped designed. Nur die UserBean ist Session-Scoped. Wird Beispielsweise eine Aktion durchgeführt, welche nicht ohne authentifizierung durchgeführt werden kann, so wird dieser Teil in der View auch nur dargestellt, wenn ein User authentifiziert ist. Wird die Aktion dann durchgeführt, kann auf die LoginDaten für den Service auf die Bean zurückgegriffen werden.

Util:

Es wurde eine Utility-Klasse erstellt, um zum Beispiel einfacher eine FacesMessage erzeugen zu können.

Klassen-Diagramm:

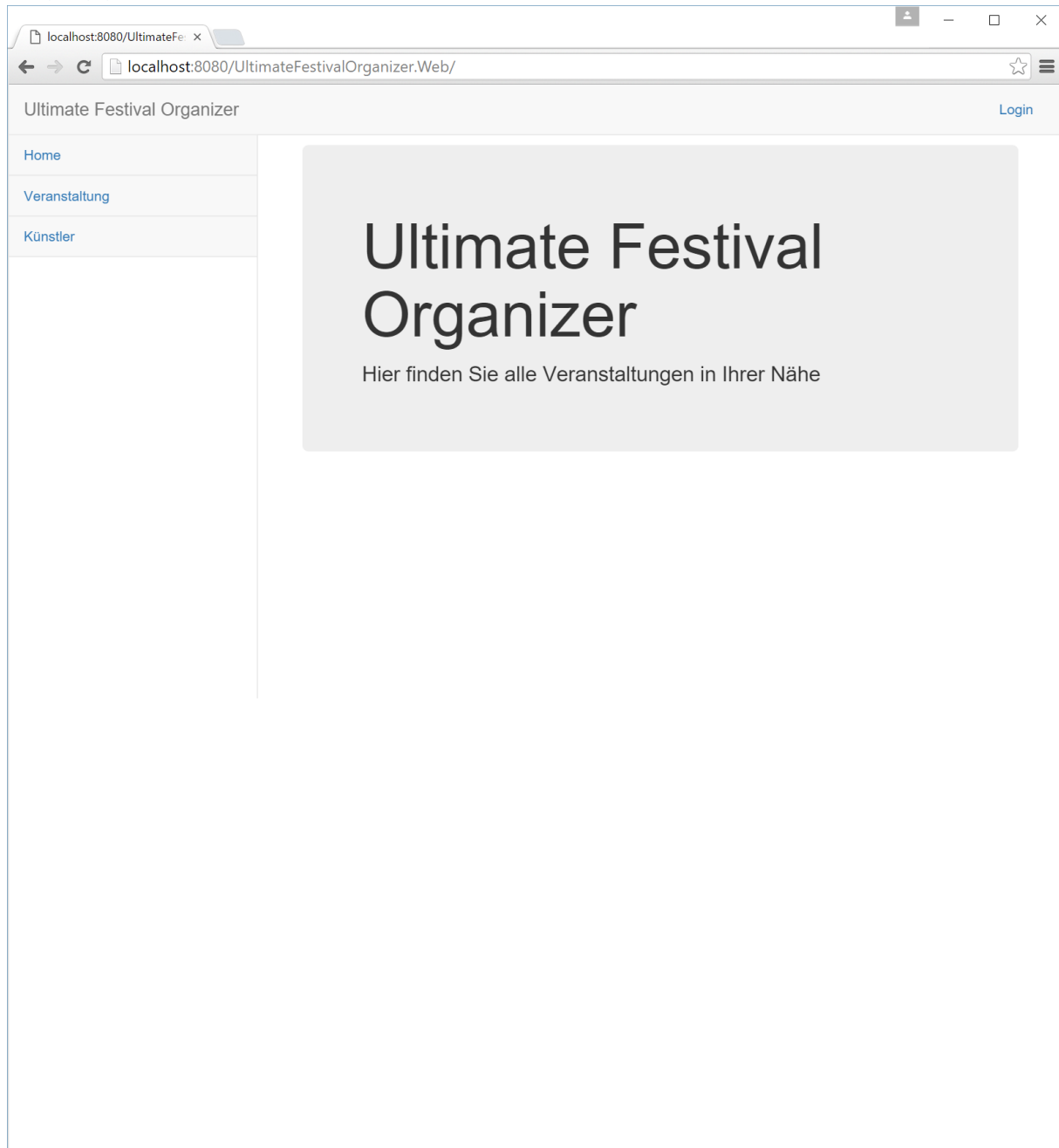
TODO

Design und Anwendung der Applikation:

Anwendungsfalldiagramm:

TODO:

Ausgangspunkt:



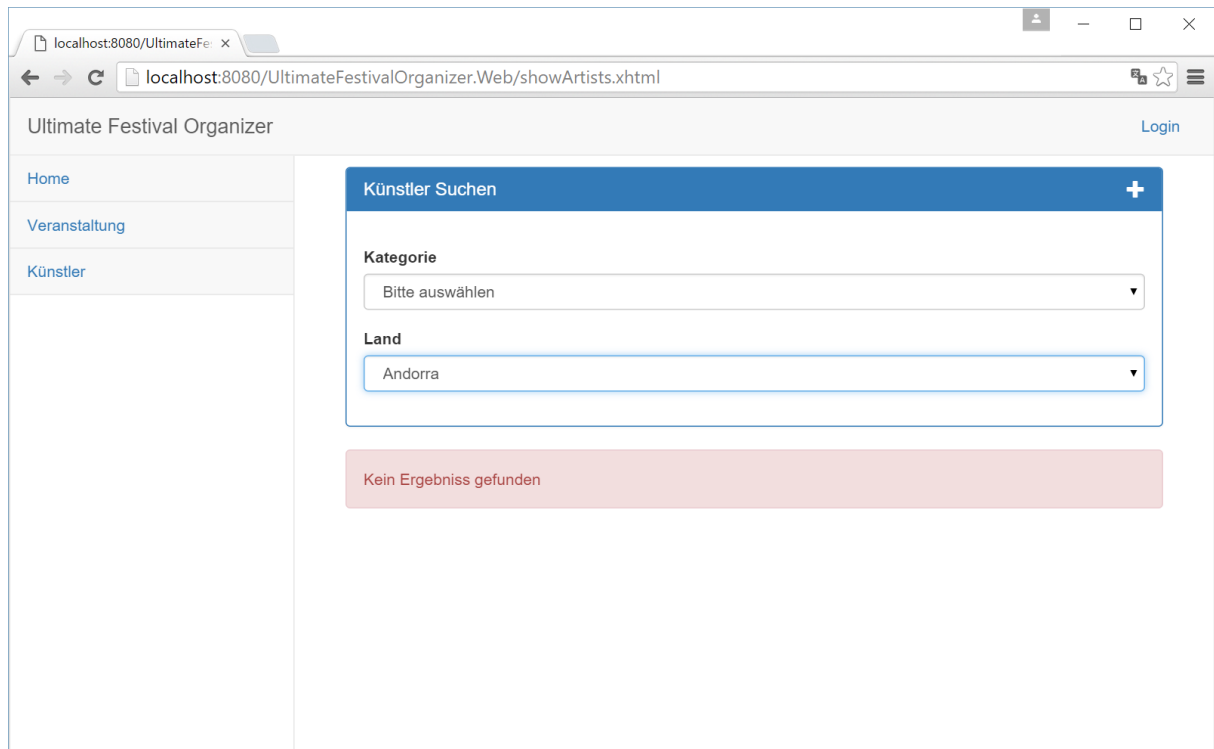
Künstler:

The screenshot shows a web browser window with the URL `localhost:8080/UltimateFestivalOrganizer.Web/showArtists.xhtml`. The page title is "Ultimate Festival Organizer". On the left is a navigation menu with links for "Home", "Veranstaltung", and "Künstler". The main content area is titled "Künstler Suchen" and features two search filters: "Kategorie" and "Land", both with dropdown menus set to "Bitte auswählen". Below the filters, a green banner states "Folgende Künstler wurden gefunden". There are six artist cards displayed in a 2x3 grid. Each card includes a profile picture, the artist's name, email, category, category description, and country. At the bottom of each card are two buttons: "zur Homepage" and "Veranstaltungen".

Name	Email	Kategorie	Land
margot meyer	margot.meyer@example.com	Catagory 5	Österreich
ambre dufour	ambre.dufour@example.com	Catagory 6	Österreich
ambre lemoine	ambre.lemoine@example.com	Catagory 8	Österreich
elouan riviére	elouan.riviere@example.com	Catagory 7	
capucine nicolas	capucine.nicolas@example.com	Catagory 9	
fabio barbier	fabio.barbier@example.com	Catagory 9	

Über die Suche können Künstler nach Kategorie und Land gefiltert werden. Dies geschieht dann per AJAX, und sofort nach Auswahl wird die Ergebnissliste aktualisiert. Ist kein Ergebniss vorhanden, wird dies ebenfalls angezeigt.

Mit dem Button Veranstaltungen kommt man direkt zu den Auftritten des Künstlers.



Veranstaltung:

Auf der Seite Veranstaltung ist als BasisFilter eingestellt, das alle Veranstaltungen des heutigen Tages angezeigt werden. Indem man die Suche ausklappt, kann man jedoch die Sucheinschränkungen verändern. Kommt man über einen Künstler, so wird nicht auf den Tag sondern auf den Künstler eingeschränkt.

The screenshot shows a web browser window with the address bar displaying `localhost:8080/UltimateFestivalOrganizer.Web/showPerformances.xhtml`. The application title is "Ultimate Festival Organizer" with a "Login" link in the top right. A left sidebar contains navigation links: "Home", "Veranstaltung", and "Künstler". The main content area features a blue header "Veranstaltung Suchen" with a "+" icon. Below this, a green banner states "Folgende Veranstaltungen wurden gefunden". A vertical line with circular icons separates the search area from a list of performance cards. Each card includes the artist's name, date and time, category, location, and buttons for "Künstler" and "Details".

Artist	Date & Time	Category	Location
fabio barbier	2016-01-23 14:00	Category 9	This is Random Stage No 6
nils roche	2016-01-23 14:00	Category 2	This is Random Stage No 7
eden gautier	2016-01-23 14:00	Category 4	This is Random Stage No 8
ambre dufour	2016-01-23 15:00	Category 6	This is Random Stage No 3
capucine nicolas	2016-01-23 15:00		

The screenshot shows a web browser window with the URL `localhost:8080/UltimateFestivalOrganizer.Web/showPerformances.xhtml`. The application is titled "Ultimate Festival Organizer" and has a navigation menu with links for "Home", "Veranstaltung", and "Künstler". A "Login" link is also present in the top right.

The main content area features a search form titled "Veranstaltung Suchen". The form includes the following fields:

- Datum:** A date input field showing "2016-01-23".
- Ort:** A dropdown menu showing "This is Random Stage No 10".
- Kategorie:** A dropdown menu showing "Bitte auswählen".
- Künstler:** A dropdown menu showing "Bitte auswählen".

A blue "Suchen" button is located below the search fields.

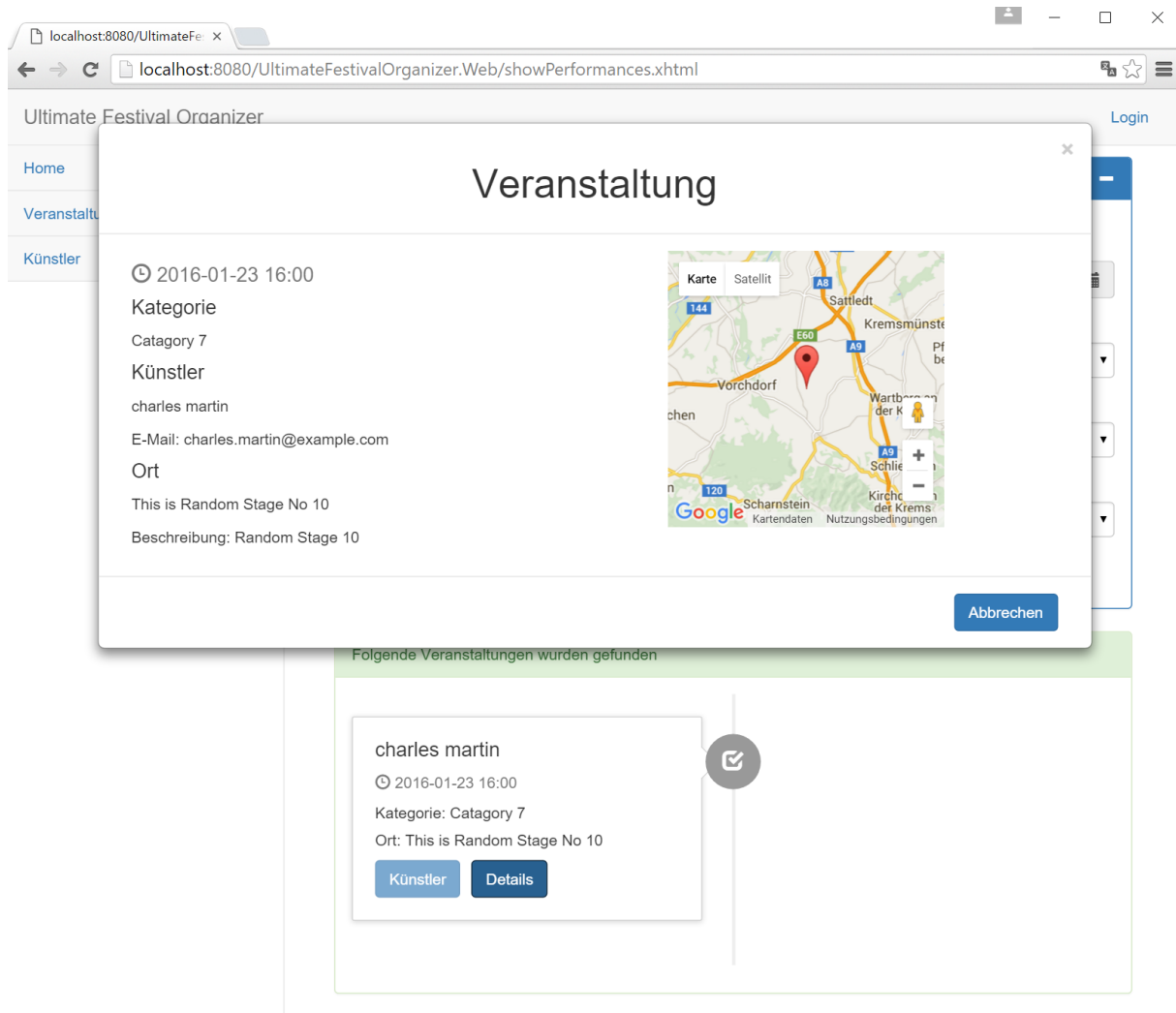
Below the search form, a green banner indicates "Folgende Veranstaltungen wurden gefunden". A single result is displayed in a card format:

- Artist:** charles martin
- Date:** 2016-01-23 16:00
- Category:** Catagory 7
- Location:** Ort: This is Random Stage No 10

At the bottom of the result card are two buttons: "Künstler" and "Details". A circular icon with a checkmark is visible to the right of the result card.

Die DropDowns aktualisieren sofort das Suchergebniss. Wird das Datum verändert, muss der Button Suchen verwendet werden.

Mit dem Button Künstler in der ErgebnissListe kommt man direkt zur Hompeage des Künstlers. Ist keine hinterlegt oder bekannt, ist dieser Button ausgegraut. Details öffnet einen Dialog in welchem mehr Details zur Veranstaltung bekannt werden.

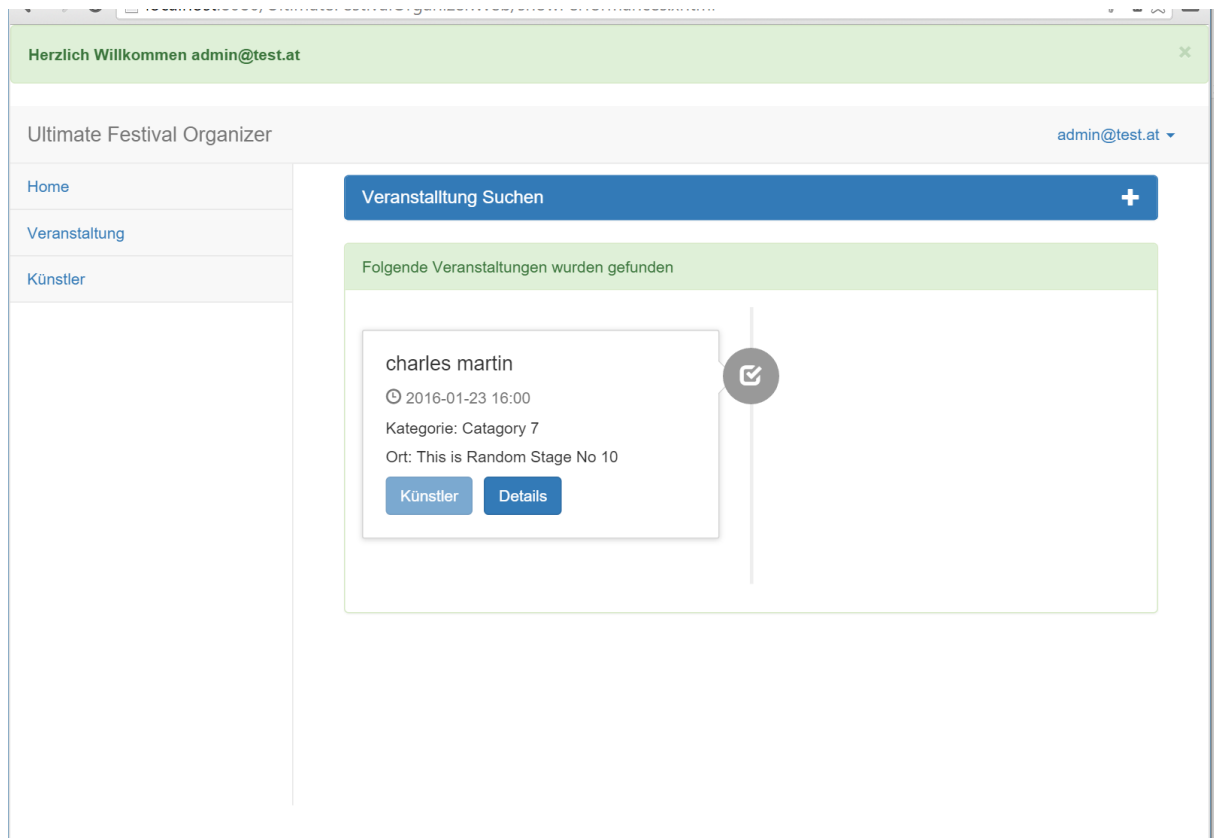


Ist ein Administrator im System eingeloggt, kann er hier die Veranstaltung Absagen oder Verschieben.

Login:

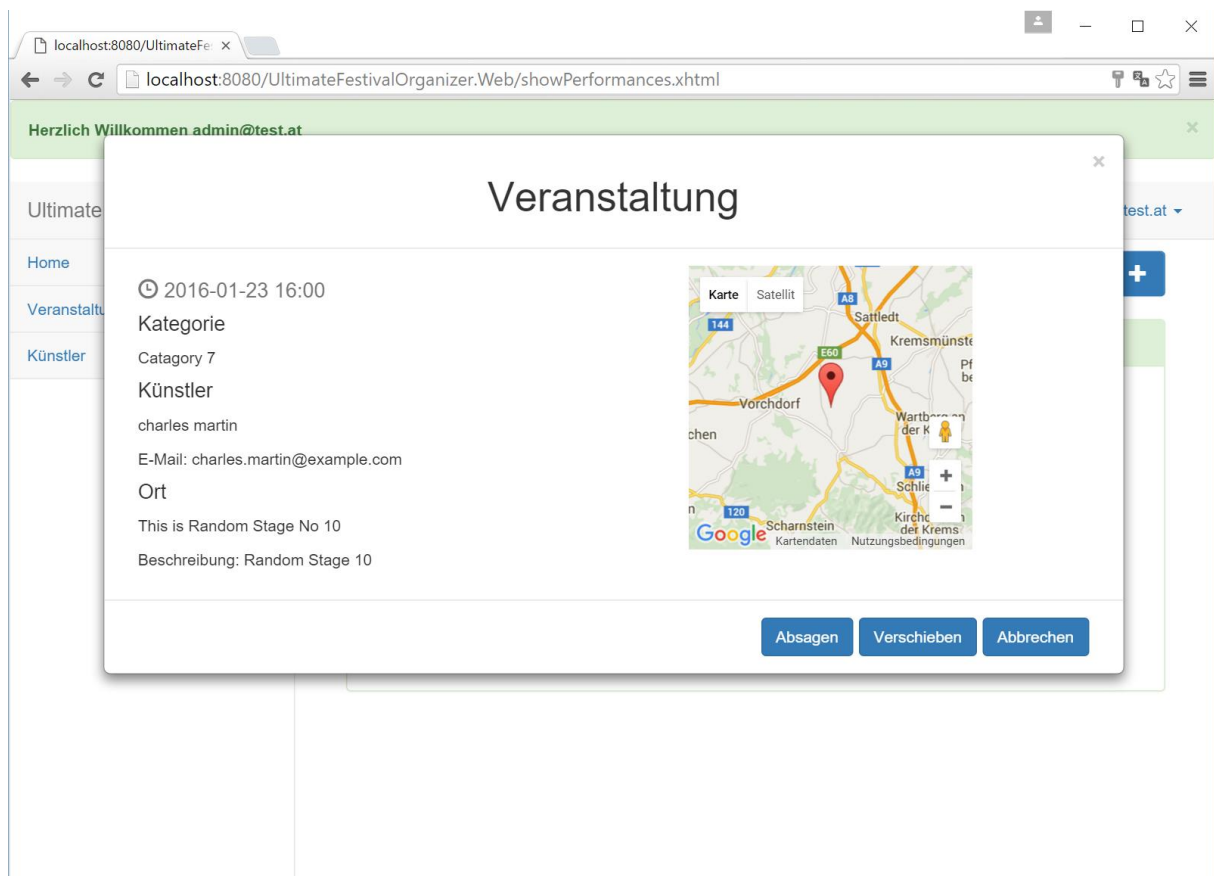
The screenshot shows a web browser window with the address bar displaying 'localhost:8080/UltimateFestivalOrganizer.Web/showPerformances.xhtml'. The page title is 'Ultimate Festival Organizer'. A sidebar on the left contains links for 'Home', 'Veranstaltung', and 'Künstler'. A 'Login' link is visible in the top right corner. A modal dialog titled 'Login' is centered on the screen. It contains two input fields labeled 'Email' and 'Password', a blue 'Anmelden' button, and a grey 'Cancel' button. The modal has a close button (X) in the top right corner.

Ein Login mit falsche Daten, führt zu einer Anzeige, einer Fehlermeldung, dass man falsche Daten eingeben hat.

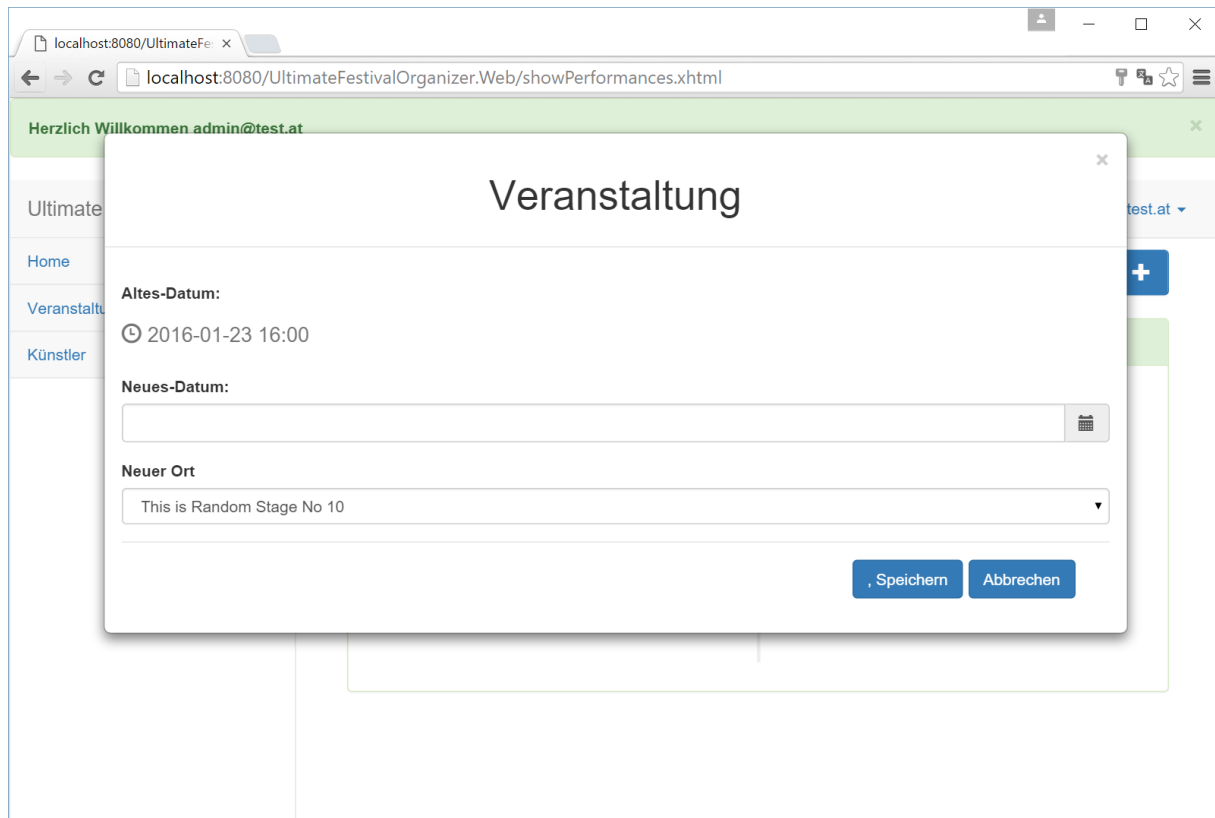


Veranstaltung verschieben:

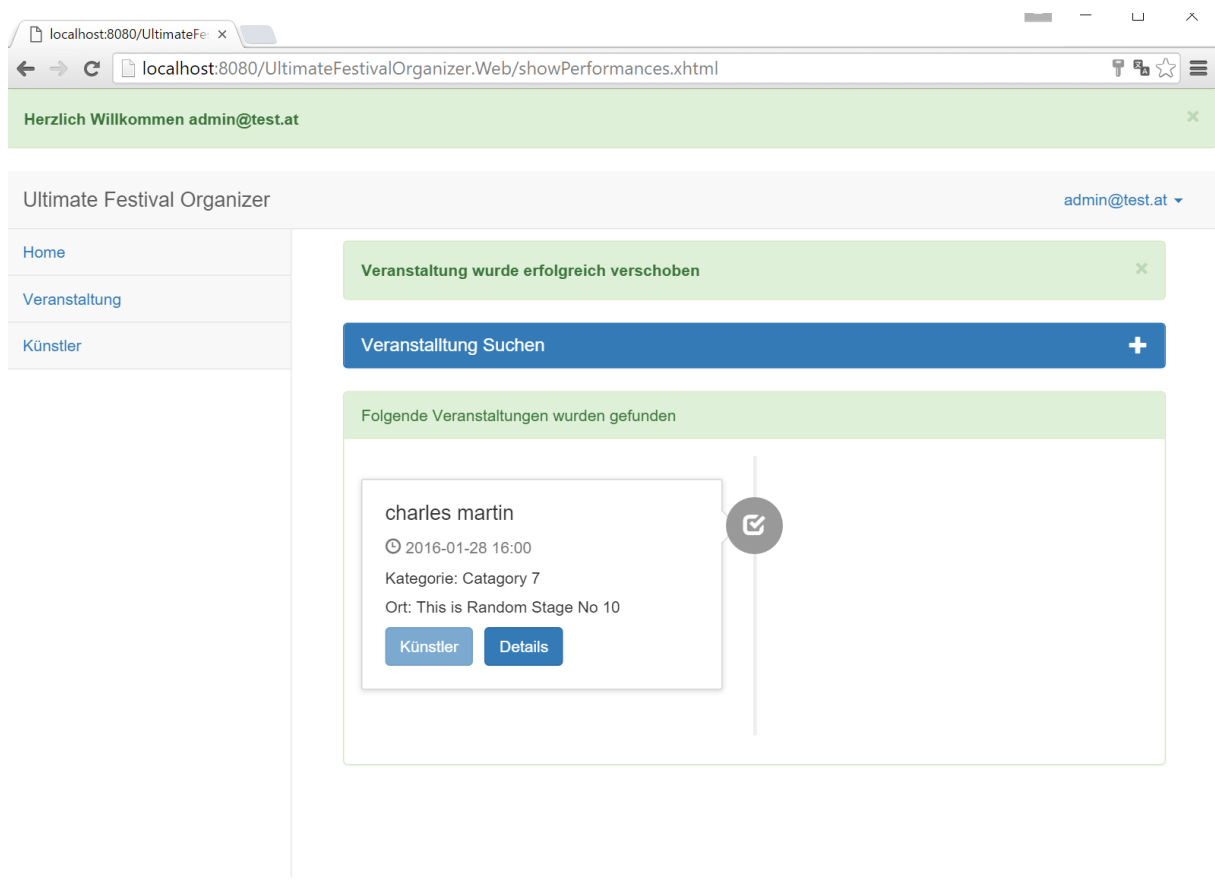
Dazu einfach auf der Veranstaltung nach dem Login wieder auf Details gehen.



Hier kann man nun Verschieben wählen:



Es kann nun ein neuer Ort, sowie eine Neue Zeit ausgewählt werden. Kommt es zu einem Konflikt mit einer anderen Veranstaltung, wird der Termin nicht verschoben, und eine Meldung wird ersichtlich. Kommt es zu keinen, wird der Termin verschoben, und der Benutzer bekommt ebenfalls eine Meldung.



Es wird auch gleich nach dem neuen verschobenen Termin gesucht.

Termin Absagen.

Wir gehen in den Detail-Dialog wie oben und wählen Absagen. Der Termin verschwindet aus der Ansicht, und es kommt eine Meldung das der Termin storniert wurde.

Logout:

