Projektdokumentation

„Planung und Entwicklung einer Java-Applikation zur Verwaltung eines webbasierten Veranstaltungskalenders“

Projektarbeit von:	Torsten Oehl Birkenfelder Straße 23 26160 Bad Zwischenahn
Prüfungsbetrieb:	Kommunale Datenverarbeitung Oldenburg (KDO) Elsässer Str. 66 26121 Oldenburg
Ausbildungsberuf:	Fachinformatiker-Anwendungsentwicklung
Prüfungszeit:	Sommer 2007

Inhaltsverzeichnis

Vorwort

Das Projekt wird in dem Ausbildungsbetrieb, der Kommunalen Datenverarbeitung Oldenburg (KDO), durchgeführt und betreut. Es wird in der Abteilung „32: Neu- und Weiterentwicklung“ in Zusammenarbeit mit Servicebereich „2: Vertrieb und Kundenmanagement“ entwickelt.
Die KDO ist ein kommunaler Zweckverband und fungiert als IT-Dienstleister für viele niedersächsische Kommunen. Neben der Entwicklung von Softwarelösungen und technischer Beratung, bietet die KDO auch viele Veranstaltungen und Seminare an.
Diese Projektdokumentation beschreibt zunächst das Projekt im Allgemeinen, geht dann auf die wirtschaftlichen Aspekte ein und erklärt anschließend die einzelnen Phasen der Entwicklung. Abschließend wird noch der Projekterfolg deutlich gemacht.

1 Projektbeschreibung

1.1 Ausgangssituation

Im vergangenen Jahr ist das Gebäude der KDO durch einen Anbau um fast das Doppelte gewachsen. Neben vielen neuen Büroräumen sind nun auch mehrere Schulungsräume vorhanden, wodurch sich das Schulungsangebot der KDO stark erweitert hat.
Für Seminare bzw. Schulungen können sich Kunden über die Internetseite der KDO anmelden. Für andere Veranstaltungen erfolgt die Anmeldung noch immer über den Postweg. Durch die Vielzahl an neuen Schulungsangeboten sind beide Anmeldeverfahren ungeeignet. Das bestehende Anmeldeprogramm verfügt über eine stark eingeschränkte Funktionalität und ist nicht mehr zeitgemäß zu bedienen. Die Bearbeitung von Anmeldungen per Post ist zu kostspielig und zeitaufwändig.
Der Vertrieb der KDO hat vorgeschlagen, ein neues Programm zu entwickeln, da es auf dem Markt keine Standardsoftware gibt, die alle Anforderungen der KDO abdecken kann. Mit dem neuen Programm sollen sich sämtliche Arten von Veranstaltungen verwalten lassen. Das Pflichtenheft wurde vorab in einem Gespräch mit dem Vertrieb erstellt. Außerdem wurde ein Termin im März vereinbart, an dem eine kleine Präsentation des Entwicklungsstandes gehalten werden soll.

1.2 Projektziel

Ziel des Projekts ist es, eine Anwendung für den Vertrieb der KDO zu entwickeln, mit dem alle Anmeldungen von Kunden zu KDO-Veranstaltungen über das Internet abgewickelt werden können. Die Verwaltung der Datenbestände, wie z.B. eingegangene Anmeldungen, Angaben zu Veranstaltungen, soll über ein zentrales Programm erfolgen.
Die neue Applikation löst eine alte ASP-Anwendung[1] ab, die bisher eingesetzt wird, um Seminare zu verwalten. Mit der neuen Lösung sollen allerdings auch andere Veranstaltungsarten gepflegt werden können, zum Beispiel Messen, Präsentationen oder Foren. Bei einigen Veranstaltungsarten spielen zusätzliche Daten eine Rolle. Zum Beispiel ist bei Seminaren ein Dozent und ein Themengebiet anzugeben, und bei Messen kann sich ein Kunde zusätzlich für verschiedene Vorträge anmelden.
Für die Datenhaltung wird eine neue Datenbank installiert, auf die später das Administrationsprogramm und auch das Anmeldeprogramm für Kunden zugreifen sollen.
Das Programm liegt zentral auf einem Server, wodurch der Aufwand bei Installationen auf Arbeitsplatz-PCs und bei evtl. Programmupdates minimal gehalten wird.

[1] Active Server Pages; Skriptsprache für dynamische Internetseiten von Microsoft

Das Administrationsprogramm soll von Mitarbeitern des Vertriebs und von Dozenten von Seminaren verwendet werden. Verschiedene Berechtigungsstufen erlauben es außerdem, das Programm bei Interessenten im Haus zu installieren, die nur Auskunftsrecht haben, d.h. keine Daten ändern können. Dies kann sich z.B. bei der Vorbereitung von Seminaren als nützlich erweisen. Zur Auswertung der Anmeldungen ist eine Export-Funktion vorgesehen, die verschiedene Listen als CSV-Datei[2] exportieren soll. Diese können dann mit einem gängigen Tabellenkalkulationsprogramm ausgewertet werden.
Für die Authentifizierung der Anwender wird beim Programmstart nach einem Benutzernamen und einem Passwort gefragt. Benutzer können von Mitarbeitern mit administrativen Rechten (z.B. der Vertrieb) angelegt und verwaltet werden.

1.3 Abgrenzungen

Das Web-Frontend, das die Anmeldungen für Kunden ermöglicht, wird parallel von einem anderen Mitarbeiter entwickelt. Das Administrationsprogramm und die Webapplikation hätten sich nicht in dem vorgegebenen Projektzeitraum von 70 Stunden entwickeln lassen. Daher beschränkt sich dieses Projekt auf das Administrationsprogramm.

1.4 Erweiterungen

Als Erweiterungen sind Suchfunktionen denkbar, um etwa gezielt nach Anmeldungen suchen zu können. Außerdem könnte der CSV-Export durch eine Excel-Schnittstelle abgelöst werden. Dadurch ließen sich die exportierten Listen besser gestalten, etwa durch angepasste Spaltenbreiten oder farbliche Untermalung. Eine Druckausgabe mit JasperReports[3] ist ebenfalls denkbar.

1.5 Projektumfeld

Auftraggeber des Projekts ist die Kommunale Datenverarbeitung Oldenburg (KDO). Da es sich um ein internes Projekt handelt, findet die Abwicklung hauptsächlich über die verschiedenen Abteilungen der KDO statt.
Da zeitnah zum Projektabschluss eine große Veranstaltung der KDO stattfindet, kann das Programm bereits kurz nach der Fertigstellung im Echteinsatz getestet werden.

1.6 Projektschnittstellen

Um das Projektziel zu erfüllen, ist eine enge Zusammenarbeit mit den Mitarbeitern der KDO notwendig. Ansprechpartnerin des Vertriebs ist Frau Schneider. Sie wird die Teilergebnisse begutachten und bewerten. Den Piloteinsatz für das Projekt werden Frau Schneider und Herr Neumann, der Bereichsleiter des Geschäftsbereichs „2: Anwendungs-Management“, durchführen, da sie verantwortlich für eine große Veranstaltung der KDO im April sind.
Eine weitere Schnittstelle ist Herr Block, der die Webanwendung entwickelt, mit der sich die Kunden später an das System anmelden können. Mit ihm muss z.B. die Datenbankstruktur geklärt werden.

Projektschnittstellen		
Frau Schneider	KDO	Ansprechpartnerin des Vertriebs, testet die Anwendung
Herr Neumann	KDO	Test der Anwendung im Echteinsatz
Herr Block	KDO	Entwicklung der Webanwendung

Tabelle 1: Projektschnittstellen

[2] Comma Separated Values; Dateiformat, das sich durch den einfachen Aufbau leicht auswerten lässt
[3] Tool zur Erstellung von Berichten unter Java

2 Projektplanung

2.1 Zeitplanung der Projektphasen

Innerhalb des Projektzeitraums vom 28.02. bis zum 10.05.2007 finden die Arbeiten an dem Projekt hauptsächlich in den ersten zwei Märzwochen statt. Die tägliche Arbeitszeit daran beträgt rund fünf Stunden.
Die Projektdurchführung erfolgt in mehreren Projektphasen, deren Teilaufgaben und die jeweilige Zeiteinschätzung der folgenden Tabelle zu entnehmen sind.

Projektphasen	Angefallene Aufgaben	Stundenzahl
Analysephase	Analyse des Ist-Zustandes Definition des Soll-Konzeptes Erstellen eines Fachkonzeptes	8
Designphase	Erstellen eines Datenflussplans Erstellen von Klassendiagrammen (UML) Erstellen eines DV-Konzeptes	9
Realisierungsphase	Erstellen der MySQL-Datenbank Programmierung der Anwendung	35
Testphase	Testen der Anwendung Ergebnisse dokumentieren	7
Auslieferung	Einführung und Übergabe	3
Dokumentation	Erstellen einer Benutzerdokumentation Erstellen der Projektdokumentation	8
Projektdauer insgesamt		**70**

Tabelle 2: Projektphasen

2.2 Ressourcenplanung

Für die Planung und Entwicklung der Anwendung wird ein Arbeitsplatzcomputer mit dem Betriebssystem Windows XP eingesetzt. Außerdem wird das Programm unter einem Gentoo-Linux-System getestet. Grafische Darstellungen, wie z.B. UML-Diagramme[4], werden mit Microsoft Visio 2003 und Concept Draw VI angefertigt. Für die Entwicklung des Administrationsprogramms wird die Entwicklungsumgebung Eclipse (Version 3.2) mit dem Zusatzprogramm Window Builder Pro verwendet.

2.3 Kostenplanung

Die Gesamtkosten des Projekts errechnen sich aus dem Stundensatz, verteilt auf die gesamte Projektdauer. Für die Kostenplanung wird der übliche Stundensatz der Kommunalen Datenverarbeitung Oldenburg für Entwicklungsarbeiten herangezogen. Der normale Stundensatz hierfür beträgt 90 €. Dieser Wert kann allerdings nicht verwendet werden, da er sich auf Standardprojekte bezieht, also schon kalkulatorische Kosten beinhaltet. Da es sich um eine interne Anwendung handelt wird der Stundensatz nur mit 48 € kalkuliert. Multipliziert mit der angestrebten Projektdauer von 70 Stunden ergeben sich somit Kosten in Höhe von 3360 €.

[4] Unified Modeling Language; Standard für die Modellierung von Software

2.4 Wirtschaftlichkeitsanalyse

2.4.1 Wahl der Programmiersprache

Der Grund der Neuentwicklung des Veranstaltungskalenders ist eine vereinfachte Bedienung, eine leichte Erweiterbarkeit, ein höherer Funktionsumfang. Außerdem soll die neue Anwendung zukunftssicher sein. Der erste Schritt ist es daher herauszufinden, welche Programmiersprache diesen Anforderungen am ehesten entspricht.

Programmiersprache	Anforderungen				
	Preis	Installation	Bedienung	Zukunft	Erweiterbarkeit
ASP	++	++	-	--	-
PHP	++	++	-	+	+
Java	++	-	++	++	++

Tabelle 3: Entscheidungsmatrix

ASP und PHP scheiden für die Entwicklung aus, da Browserprogramme die Anforderungen für die Administration nicht optimal erfüllen können. ASP ist außerdem veraltet.
Die Wahl fällt daher auf eine Java-Desktopanwendung, da es klare Vorteile in der Bedienungsfreundlichkeit und Erweiterbarkeit gibt. Die Webanwendung wird von Herrn Block in PHP entwickelt.

2.4.2 Kostenvergleichsrechnung

Um herauszufinden, ob sich eine neue Anwendung lohnt, wird nun die Erfassung der Anmeldungen per Post und die Entwicklung des neuen Programms verglichen.
Es wird davon ausgegangen, dass an 256 Tagen im Jahr jeweils eine Schulung stattfindet. Davon gehen 30 % der Anmeldungen per Post ein. Bei durchschnittlichen 15 Teilnehmern pro Schulung sind das insgesamt 1152 Anmeldungen, die bearbeitet werden müssen. Dazu kommen noch drei jährliche Großveranstaltungen der KDO, bei denen jeweils ca. 200 Anmeldungen per Post eingehen. Pro Jahr müssen also ca. 1752 Briefe bearbeitet werden. Diese werden, wenn möglich, per E-Mail beantwortet, durchschnittlich 945 Anmeldungen müssen jedoch per Post bestätigt werden. Es wird angenommen, dass die Bearbeitung sämtlicher Anmeldungen jährlich ca. 1300 € kostet. Dieser Wert beinhaltet Kosten für Briefe (Briefmarken, Umschläge usw.), sowie einen geschätzten Stundenlohn.
Das neue Programm wickelt die Anmeldebestätigung generell automatisch per E-Mail ab und durch die Export-Funktion können auch Teilnehmerlisten automatisch erstellt werden. Die jährlichen Kosten können dadurch auf 150 € gesenkt werden.
Ausgehend von einer Laufzeit von fünf Jahren beträgt die Gesamtkostendifferenz 2390 €.

	Alt	Neu
Jahr 1	1300 €	3510 €
Jahr 2	1300 €	150 €
Jahr 3	1300 €	150 €
Jahr 4	1300 €	150 €
Jahr 5	1300 €	150 €
Summe	6500 €	4110 €
Gesamtkostendifferenz		**2390 €**

Tabelle 4: Kostenvergleichsrechnung

2.4.3 Amortisationsdauer

Die Amortisationsdauer ist die Zeit, nach der die Entwicklungskosten des neuen Programms durch die jährlichen Ersparnisse, die es ermöglicht, refinanziert wird. Es gilt folgende Formel:

Amortisationsdauer = Entwicklungskosten / jährliche Ersparnisse

Hier betragen die Entwicklungskosten 3360 € und die jährlichen Ersparnisse 1150 € (1300 € für das Altverfahren abzüglich 150 € für das Neue).
Daraus errechnet sich die Amortisationsdauer von 2,9 Jahren. Die Entwicklung der Kosten wird in Abb. 1 deutlich gemacht. Es wird davon ausgegangen, dass das neue Verfahren ab Mitte 2007 eingesetzt wird.

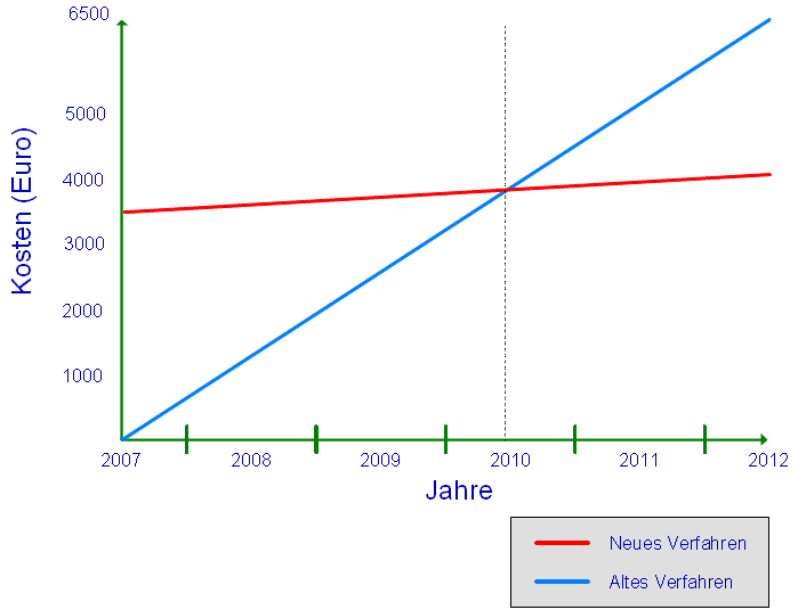


Abbildung 1: Amortisationsdauer

2.4.4 Nutzwertanalyse

Mit Hilfe einer Nutzwertanalyse soll geprüft werden, ob das geplante Verfahren wirtschaftlicher ist als das bestehende.

Kriterium	Bewertung	Produkt vorher	Produkt nachher
Einarbeitungszeit	1	4	5
Bearbeitungszeit	3	7	9
Exportmöglichkeit	2	5	7
Statistiken	3	6	4
Erweiterbarkeit	2	3	9
Summe	**11**	**25**	**34**
Wirtschaftlichkeitskoeffizient		**2,27**	**3,09**

Tabelle 5: Nutzwertanalyse

Der Wirtschaftlichkeitskoeffizient macht deutlich, dass sich die zu erfüllenden Aufgaben mit der neuen Lösung deutlich besser erfüllen lassen, als mit der bisherigen.

3 Fachkonzept

In dem Fachkonzept stehen alle Anforderungen, die der Anwender hinsichtlich der zu erfüllenden Aufgaben an den neuen Veranstaltungskalender hat. Das Fachkonzept befindet sich im „Anhang A: Fachkonzept“.

4 Datenverarbeitungskonzept

Im Datenverarbeitungskonzept ist der Umgang mit den Datenbeständen, die im Fachkonzept beschrieben sind, erläutert. Dazu gehören z.B. der Systemaufbau, der Aufbau der Anwendung und Datenbankmodelle. Das Datenverarbeitungskonzept befindet sich im „Anhang B: Datenverarbeitungskonzept“.

5 Projektdurchführung

5.1 Phasenmodell

Die Durchführung des Projekts wird in fünf Phasen eingeteilt, die nacheinander abgearbeitet werden. Die einzelnen Phasen müssen sehr gründlich abgeschlossen werden, damit Rücksprünge auf bereits abgeschlossene Phasen vermieden werden können.

5.2 Analysephase

Am 07.12.2006 fand eine Bereichssitzung der Abteilung „32: Neu- und Weiterentwicklung“ statt. Hier erwähnte Herr Franke, Bereichsleiter und Ausbilder, das erste Mal den Wunsch des Vertriebs, einen neuen Veranstaltungskalender zu entwickeln.
Es lag bereits eine grobe Liste mit Anforderungen vor, die in einem Soll-Ist-Vergleich mit dem derzeitigen Veranstaltungskalender für die Anmeldung für Seminare gegenübergestellt wurden. Dabei hat sich herausgestellt, dass die Wünsche für eine Neuentwicklung gerechtfertigt waren.
In einer Vorbesprechung, an der Frau Beier und Frau Schneider vom Vertrieb teilnahmen, wurde gemeinsam ein Pflichtenheft erstellt. Um die speziellen Anforderungen von Seminaren zu erfahren, wurde in einem weiteren Gespräch noch die Seminarleiterin Frau Rother hinzugezogen.
Aus den gesammelten Informationen wurde das Fachkonzept entwickelt, in dem sich eine Ist-Analyse zur aktuellen Situation und ein Sollkonzept mit den fachlichen Anforderungen befinden. Da zum Projektbeginn ein ausführliches Pflichtenheft vorlag und die Besprechungen keine größeren Reibungspunkte ergaben, ließ sich die Analysephase, schneller als geplant, in sieben Stunden abschließen.

5.3 Designphase

Da der Veranstaltungskalender aus einer PHP-Anwendung für den Kunden und einer Java-Anwendung für die Veranstalter besteht, muss ein Datenbanksystem gewählt werden, auf das beide Techniken aufsetzen können. Die Wahl fällt auf MySQL[5], da es sich für PHP bewährt hat und auch Java leicht mit einer MySQL-Datenbank in Verbindung treten kann. Zur Planung der Datenbank werden ein ERM[6] und anschließend ein Datenstruktur-Modell erstellt.

[5] Freies, relationales Datenbankmanagementsystem (DBMS)
[6] Entity Relationship Model; Darstellungsmethode einer relationalen Datenbank

Für die Entwicklung der Java-Applikation steht ein Framework[7] zur Verfügung, das bereits in einigen kleineren Anwendungen der KDO verwendet wurde. Dieses stellt bereits einen Grundaufbau des Designs, ein einfaches Benutzersystem und Methoden zur Datenbankverbindung zur Verfügung.
Der optische Aufbau der vorliegenden Anwendung ist in Abb. 2 zu sehen.
Um den Aufbau des Frameworks zu verstehen, werden UML-Klassendiagramme erstellt, die die Struktur und wichtige Klassen zeigen.
Die grafischen Benutzeroberflächen werden mit Java-Swing[8] erstellt. Als Hilfsprogramm kommt hierfür das Eclipse-Plugin Window Builder Pro zum Einsatz. Mit diesem Programm lassen sich GUIs[9] einfach gestalten. Der generierte Java-Code ist sehr sauber und übersichtlich.

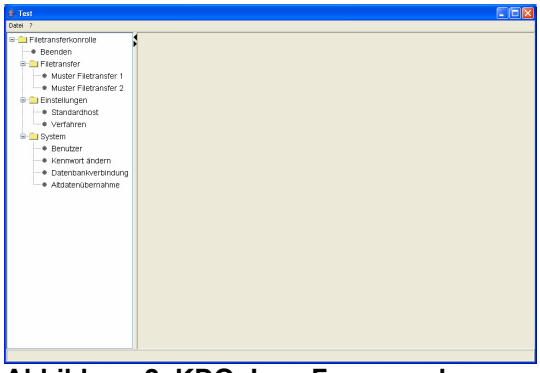


Abbildung 2: KDO-Java-Framework

Um das Layout zu planen, werden die GUIs vorab noch grob mit Microsoft Visio 2003 gezeichnet.
Dadurch kann der Aufbau der Oberflächen vor der Entwicklung mit den beteiligten Personen abgesprochen werden. Aufwändige nachträgliche Änderungen an den GUIs können dadurch verhindert werden.
Die Datenbankmodelle und UML-Diagramme wurden in einem Datenverarbeitungskonzept, das außerdem das Gesamtsystem beschreibt, dokumentiert. Da sich die Anwendung nach einem Framework richtet, in dessen Aufbau sich noch eingearbeitet werden musste, wurde die Designphase nach zehn Stunden abgeschlossen. Das ist eine Stunde länger als geplant.

5.4 Realisierungsphase

Für die Entwicklung wird das Model-View-Controller (MVC) Entwurfsmuster verwendet. Es beschreibt eine Lösung für die Struktur des späteren Programms. Das MVC-Prinzip sieht vor, dass Programmlogik, Datenhaltung und Darstellung voneinander getrennt werden.
Dadurch erlangt ein Programm, das nach diesem Muster programmiert wurde, eine sehr hohe Flexibilität, da man beispielsweise eine View austauschen kann, ohne dass sich die Programmfunktionen ändern.

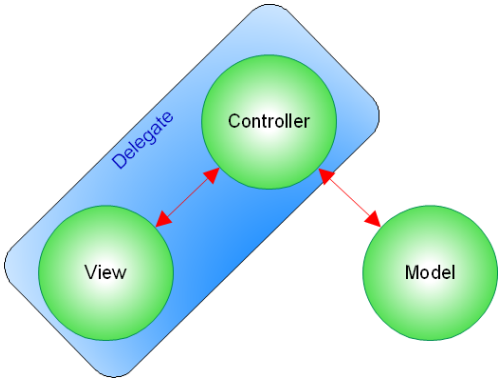


Abbildung 3: Model-View-Controller

Bei der Programmierung sehr kleiner Java-Swing-Anwendungen kombinieren manche Entwickler die View und den Controller zu einer einzigen Klasse[10], da die Entwicklung einer eigenen Controllerklasse zu zeitaufwändig wäre. Sind View und Controller in einer Klasse vereint so nennt man das einen „Delegate“.
In diesem Projekt wird nur ein Delegate verwendet und zwar die Klasse „ApplicationView“. Der Veranstaltungskalender setzt sich also aus Model-Klassen, View-Klassen und Controller-Klassen zusammen. Außerdem beinhaltet das Projekt Klassen zur Datenhaltung und Klassen, die Methoden bereitstellen, die für die Zielsetzung sinnvoll sind.

[7] Grundgerüst, das bereits einige Funktionen bereitstellt und architektonische Vorgaben macht
[8] Swing ist eine Programmbibliothek von Java zur Erstellung von Benutzeroberflächen
[9] Graphical User Interface; grafische Benutzeroberfläche
[10] Klassen sind die Bausteine einer Java-Anwendung. Sie beschreiben die Objekte, aus denen ein Programm besteht.

5.4.1 Aufgaben und Entwicklung der Views

Eine View ist für die Darstellung der Daten zuständig. Die Views werden so entwickelt, dass keinerlei Abhängigkeiten zu dem Controller oder dem Model bestehen.
Eine GUI muss in Swing einen sog. Layoutmanager zugewiesen bekommen. Dieser sorgt für die Anordnung der einzelnen Bedienelemente auf der GUI.
Für den Veranstaltungskalender wird JGoodies Form-Layout verwendet. Es beschreibt die View als Einteilung in Zeilen und Spalten.
Abbildung 4 zeigt die Einteilung anhand einer beispielhaften View. Die Maske auf der Abbildung besteht aus einem 4x4 großen Feld, in dem die Komponenten (Eingabefelder, Schaltflächen, usw.) angeordnet sind.
Die einzelnen Komponenten können sich dabei über beliebig viele Felder ausdehnen (horizontal und vertikal).

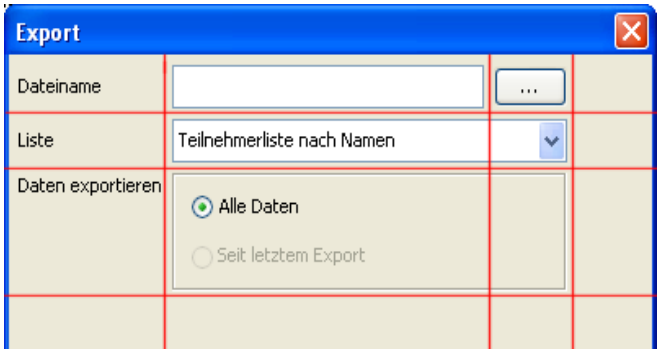


Abbildung 4: Eine View, eingeteilt in Zeilen/Spalten

Die Textfelder und Buttons müssen sich an die vorgegebenen Standards einer KDO-Java-Anwendung halten. Sie müssen eine bevorzugte Größe von 100x25 Bildpunkten besitzen, sich aber der Größe des Fensters anpassen. Dieses Verhalten wird durch das Form-Layout gewährleistet. Außerdem ist in jeder View ein Abstand zum rechten und unteren Fensterrand einzubauen. Das geschieht durch jeweils eine zusätzliche Zeile/Spalte (siehe Abb. 4).
Auch wenn die Views sozusagen der sichtbare Teil der Anwendung sind, ist deren Entwicklungszeit vergleichsweise gering. Für die Entwicklung der Views wurden insgesamt sechs Stunden benötigt.

5.4.2 Aufgaben und Entwicklung der Controller

Die Controller steuern den Programmablauf der Anwendung. Sie reagieren auf Eingaben in der View und leiten die Schritte zur Verarbeitung ein. Möchte der Benutzer beispielsweise seine Eingaben speichern, werden diese vom Controller auf Plausibilität geprüft. Ist dies der Fall, veranlasst der Controller das Model dazu, die Daten zu speichern. Zu jeder View gehört genau ein Controller. Controller verschiedener Views können auch untereinander agieren.
Der Hauptbestandteil der Realisierungsphase bestand in der Entwicklung der Controller. Es wurden insgesamt 23 Stunden für die Entwicklung der Controller benötigt.

5.4.3 Aufgaben und Entwicklung der Models

Ein Model beinhaltet die Daten, die in der View dargestellt bzw. bearbeitet werden sollen. Einige Entwickler programmieren Models so, dass es von der View „beobachtet“ wird und sich die View sofort den Änderungen des Models anpasst. In diesem Projekt stehen View und Model jedoch in keinerlei Zusammenhang zueinander. Soll eine View ein anderes Model darstellen, so führt der Weg immer zuerst über den Controller.
Die Models werden in dem Veranstaltungskalender in zwei Javaklassen aufgeteilt: Eine Klasse in der die benötigten Daten gespeichert sind und eine Klasse, um Daten aus der Datenbank zu holen bzw. in die Datenbank zu schreiben. Diese Aufteilung hat Vorteile in der Übersicht bei der Programmierung.
Die Klassen für die Datenhaltung konnten relativ schnell programmiert werden. Die Klassen für Datenbankzugriffe waren wegen SQL-Abfragen zeitaufwändiger. Die Entwicklung der Models dauerte acht Stunden.

5.4.4 Programmkomponenten

Um alle geforderten Aufgaben mit dem Programm zu erfüllen, werden insgesamt acht Programmkomponenten entwickelt. Eine Komponente besteht jeweils aus einem Model, einer View und einem Controller. Außerdem musste das Framework angepasst werden.

5.4.4.1 Framework anpassen

Das Framework stellt eine Programmoberfläche zur Verfügung, die auf der rechten Seite die Views vorsieht und auf der linken Seite ein Menü zur Navigation (siehe Abb. 2).
Die Menüstruktur wird so angepasst, dass sich jede einzelne Programmkomponente einfach finden lässt. Das Framework wird außerdem um einige Klassen ergänzt, mit denen die Plausibilitätsprüfung vereinfacht werden soll. Diese Klassen beinhalten u.a. Funktionen für die Prüfung von Datum und Uhrzeit. Auszüge aus diesen Klassen sind im „Anhang D: Quellcode u. Screenshots“ zu sehen.

5.4.4.2 Benutzer

Die Benutzer-Komponente ist bereits weitestgehend vom Framework vorgegeben. Es muss in die View nur noch eine Möglichkeit eingebaut werden, mit der festzulegen ist, ob ein Benutzer nur Auskunftsrecht hat. Diese Information muss dann auch im Controller, im Model und in der Datenbank angepasst werden.

5.4.4.3 Stammdaten

Vier der Komponenten sind zur Pflege der Stammdaten notwendig. Zu den Stammdaten gehören Dozenten, Seminarthemen, Kriterien (Zusatzoptionen für den Kunden) und Vorträge, wobei der Teil der Zusatzoptionen die komplexeste ist. Die Kriterien sollen später in der Webanwendung verschiedene Zusatzoptionen für den Kunden bieten. Beispiel: Wahl des Mittagessens.
Dafür wird zunächst ein Oberthema erstellt (Beispiel: Überschrift „Mittagessen“), dem dann die verschiedenen Kriterien (Beispiel: „Grünkohl“, „Schnitzel", usw.) zugewiesen werden. In der View wird das durch Listen und Schaltflächen zum Verschieben der Kriterien in ein beliebiges Thema realisiert.
Der Controller muss darauf achten, dass keine Kriterien gelöscht werden, die bereits von Kunden ausgewählt wurden.
In den übrigen Programmkomponenten für die Stammdaten sind vor allem Texteingaben zu managen.

5.4.4.4 Veranstaltungen

In der Komponente der Veranstaltungen werden sämtliche Daten zu den einzelnen Veranstaltungen bearbeitet. Hier finden sich außerdem eine Tabelle mit eingegangenen Anmeldungen zu der Veranstaltung und eine Tabelle, die Auskunft über die gewählten Kriterien der Kunden gibt. Durch die Fülle an Informationen muss der Controller hier sehr viele Plausibilitäten beachten.

5.4.4.5 Anmeldungen

Für die Bearbeitung der eingegangenen Anmeldungen werden in dieser Komponente Adressdaten und die vom Kunden gewählten Kriterien und Vorträge bearbeitet. Da die Kriterien und Vorträge dynamisch verwaltet werden, müssen auch die Eingabefelder dynamisch generiert werden.
Aufgrund der Tatsache, dass sich Anmeldungen immer auf eine bestimmte Veranstaltung beziehen, muss der Controller der Anmelde-Komponente mit dem Veranstaltungs-Controller zusammenarbeiten.

5.4.4.6 Export

Der Dialog für den Export der Anmeldelisten hat einen relativ kleinen Controller, da die View recht klein ist und wenige Ereignisse gesteuert werden müssen. Das Model dieser Komponente ist allerdings relativ umfangreich, da hier die Anmeldedaten ausgelesen und in eine CSV-Datei geschrieben werden müssen. Da es mehrere verschiedene Export-Listen gibt, steigert sich der Umfang des Models nochmals, da für die Listen unterschiedliche Daten verarbeitet werden müssen.

5.4.5 Ergebnis

Für die Realisierungsphase wurden insgesamt 37 Stunden benötigt; das sind zwei Stunden mehr als geplant. Der Unterschied kommt daher, dass sich einige Programmfunktionen schwerer umsetzen ließen als geplant. Das Benutzerhandbuch des Programms ist auszugsweise im „Anhang C: Auszüge aus dem Benutzerhandbuch“ zu finden.
Bildschirmfotos des fertigen Programms befinden sich unter „Anhang D: Quellcode u. Screenshots“.

5.5 Testphase

5.5.1 Statische Code-Analyse

Der erste Schritt des Testverfahrens ist es, den fertigen Quellcode nach Fehlern zu überprüfen. Dabei werden neben logischen Fehlern auch die Trennung von Model, View und Controller überprüft.

5.5.1.1 Model-Test

Es ist sichergestellt, dass von den Models nur plausible Daten in die Datenbank geschrieben werden. Die Models haben keinerlei Beziehung zu den dazugehörigen Views.

5.5.1.2 View-Test

Die Views (außer dem Delegate „ApplicationView“) enthalten keinerlei Programmlogik. Sie definieren lediglich grafische Komponenten. Die Views funktionieren völlig unabhängig vom Controller oder Model (siehe „Anhang D: Quellcode u. Screenshots“). Die Views werden zum Testen ohne Kenntnis von Model oder Controller gestartet. Die Views funktionieren, wie erwartet, aber enthalten keinerlei Funktionen.

5.5.1.3 Controller-Test

Sämtliche Controller enthalten die benötigten Methoden und funktionieren einwandfrei. Jeder Controller beinhaltet nur die Funktionen, die ein Controller haben muss. Andere Dinge wie z.B. Datenbankzugriffe erfolgen von keinem Controller.

5.5.1.4 Test der Programmlogik

Logikfehler, z.B. bei komplizierten Algorithmen, sind nicht vorhanden.

5.5.2 Komponententest

Die Softwarekomponenten werden im Blackbox-Testverfahren getestet. Dieses sieht vor, dass die Komponenten ohne Wissen über die dahinter stehende Programmlogik getestet werden. Der Blackbox-Test wird von Herrn Block durchgeführt.

Test	Erwartetes Ergebnis	Ergebnis
Fehlerhafte Eingaben in Textfelder (z.B. ungültiges Datum, Text als Postleitzahl usw.)	Durch die formatierten Textfelder sollten fehlerhafte Eingaben unmöglich sein. Falls das Format stimmt, aber die Eingabe trotzdem ungültig ist, erscheint eine Fehlermeldung.	Eingabe von fehlerhaften Formaten unmöglich, Fehlermeldung erscheint bei Logikfehlern.
Aufruf der Maske unter einem anderen Betriebssystem	Das Programm muss optisch unter jedem Betriebssystem gleich aussehen.	Das Programm sieht unter Linux-Systemen genau so aus wie unter Windows.
Verhalten der Masken beim Auseinanderziehen bzw. Verkleinern	Die Größen der Bedienelemente müssen sich der Fenstergröße anpassen.	Bedienelemente passen sich korrekt der Größe an.

Tabelle 6: Komponententest

5.5.3 Datenbanktest

Test	Erwartetes Ergebnis	Ergebnis
Arbeitsplatz-PC vom Netzwerk trennen während das Programm läuft	Das Programm muss eine entsprechende Fehlermeldung anzeigen, darf aber nicht abstürzen.	Das Programm läuft weiter und zeigt bei Datenzugriff die entsprechende Fehlermeldung an.
Ausführen fehlerhafter SQL-Anweisungen	Das Programm darf die SQL-Anweisungen nicht ausführen, sondern muss stattdessen eine Fehlermeldung anzeigen.	Es wird eine Fehlermeldung angezeigt und die Daten bleiben unverändert.
Fehlen der Datenbank	Es muss eine Eingabemaske erscheinen, in der die korrekte Datenbank gewählt werden soll.	Die Maske erscheint korrekt.

Tabelle 7: Datenbanktest

5.5.4 Ergebnis

Die Testphase konnte in den geplanten sieben Stunden abgeschlossen werden. Durch Vorüberlegungen in der Designphase traten keine nennenswerten Schwierigkeiten auf.

5.6 Auslieferung

Nachdem das Programm die Testphase erfolgreich bestanden hat, wird der Vertrieb der KDO über die Fertigstellung des Veranstaltungskalenders informiert.
Das Programm wird auf den Arbeitsplatz-PCs der Vertriebsmitarbeiter installiert. Danach wird Frau Schneider in die Bedienung des Programms eingeführt. Anschließend wird das Pflichtenheft noch einmal mit dem fertigen Produkt verglichen. Kurz darauf beginnt Frau Schneider selbstständig die Echtdaten für eine große Veranstaltung im April einzupflegen.
Dabei traten keine Komplikationen auf. Da außerdem alle Anforderungen erfüllt waren, konnte die Auslieferung in den geplanten drei Stunden abgeschlossen werden.

5.7 Dokumentation

Die Dokumentation des Projekts ist ein begleitender Prozess. Die Dokumentation wird während des gesamten Projektzeitraums beiläufig gepflegt und ergänzt. Die geplante Zeit wurde also hauptsächlich in die Erstellung der Benutzerdokumentation und das Erstellen von Anschauungsmaterial, wie z.B. Screenshots des fertigen Programms, investiert.
Da weniger abschließende Dokumentationen angefertigt werden mussten als geplant, ließ sich die Dokumentation in nur sechs Stunden abschließen.

6 Projekterfolg

6.1 Soll-Ist-Vergleich

Das Ergebnis des Projektes ist als durchweg positiv zu beschreiben. Phasen die länger dauerten als geplant, ließen sich durch Zeitgewinn in anderen Phasen ausgleichen, so dass die geplante Projektdauer von 70 Stunden eingehalten werden konnte.

		Stundenzahl	
Projektphasen	**Angefallene Aufgaben**	**Soll**	**Ist**
Analysephase	Analyse des Ist-Zustandes Definition des Soll-Konzeptes Erstellen eines Fachkonzeptes	8	7
Designphase	Erstellen von Aktivitätsdiagrammen (UML) Erstellen von Klassendiagrammen (UML) Erstellen eines DV-Konzeptes	9	10
Realisierungsphase	Erstellen der MySQL-Datenbank Programmierung der Anwendung	35	37
Testphase	Testen der Anwendung Erstellen einer Testdokumentation	7	7
Auslieferung	Einführung und Übergabe	3	3
Dokumentation	Erstellen der Projektdokumentation	8	6
Gesamt		**70**	**70**

Tabelle 8: Soll-Ist-Vergleich

6.2 Ausblick

Nachdem das Projekt bereits vom Vertrieb genutzt wird, soll es in Zukunft auch bei den Seminarleitern installiert werden.
Außer den in Punkt 1.4 erwähnten Erweiterungen, die in Zukunft evtl. noch eingebaut werden, entspricht das neue Projekt den Anforderungen der nächsten Jahre.

7 Änderungen gegenüber dem Projektantrag

Während der Projektentwicklung haben sich einige Änderungen gegenüber der ersten Planung im Projektantrag ergeben.
Das ursprünglich geplante Aktivitätsdiagramm wurde nicht erstellt, da sich diese Methode zur anschaulichen Darstellung des Projekts als ungeeignet herausstellte. Stattdessen wurde ein Datenflussplan erstellt.
Die Ergebnisse des Testverfahrens wurden innerhalb der Projektdokumentation erläutert, da die ursprünglich geplante Testdokumentation nur bei größeren Projekten Sinn macht.

8 Abbildungsverzeichnis

9 Anhang

Fachkonzept

Inhaltsverzeichnis

1 Einleitung

1.1 Problemstellung

Fast jeden Werktag findet in der KDO eine, oder oftmals auch mehrere, Schulungen statt. Der dadurch entstehende Verwaltungsaufwand soll weitestgehend minimiert werden. Anmeldungen werden derzeit mit einer webbasierten ASP-Anwendung verwaltet.
Ein Benutzersystem fehlt in dieser Anwendung, so dass nur wenige Leute Zugriff auf den Datenbestand haben. Durch das erweiterte Schulungsangebot der KDO entspricht das Programm nicht mehr den funktionellen Anforderungen. So müssen für bestimmte Veranstaltungen immer noch Anmeldungen per Post bearbeitet werden, was den Verwaltungsaufwand enorm erhöht.

1.2 Zielsetzung

Es soll eine neue Anwendung entwickelt werden, die neben einem Benutzersystem vor allem alle möglichen Arten von Veranstaltungen der KDO verwalten kann. Die Kunden müssen sich über das Internet anmelden können. Das Programm zur Administration muss leicht bedienbar und übersichtlich gestaltet werden. Außerdem soll das neue Programm Funktionen zum Export verschiedener Teilnehmerlisten beinhalten.

2 Ist-Analyse

2.1 Bestandsaufnahme

Die Administration der Veranstaltungen findet, genau wie die Anmeldungen der Kunden, über eine ASP-Anwendung statt. Der Administrator wird zunächst nach einem Benutzernamen und einem Passwort gefragt, die zuvor manuell in der Datenbank eingetragen werden müssen.
Danach gelangt der Anwender in ein Hauptmenü, von dem er zu den einzelnen Bearbeitungsoptionen navigieren kann.
Zunächst kann man Themen für Veranstaltungen bearbeiten. In der Liste der Themen kann außerdem festgelegt werden, ob die Themen sichtbar sein sollen oder nicht. Zusätzlich gibt es eine Funktion zum Löschen von Themen.
Der zweite Menüpunkt führt in die Bearbeitungsmaske für die Veranstaltungen. Hier muss zunächst das gewünschte Thema ausgewählt werden. Danach werden alle Veranstaltungen zu diesem Thema aufgelistet. Nun kann man eine neue Veranstaltung anlegen, bestehende bearbeiten oder löschen. Die Bearbeitung geschieht über eine übersichtliche Maske, die jedoch nur wenige Einstellungsmöglichkeiten bietet.
Die dritte Funktion ist für die Anmeldungen. Auch hier muss zunächst wieder das gewünschte Thema und anschließend die entsprechende Veranstaltung ausgewählt werden. Erst danach erscheint eine Liste mit eingegangenen Anmeldungen, die nun gelöscht oder bearbeitet werden können. Es gibt allerdings keine Funktion, um nachträglich einen neuen Datensatz hinzuzufügen.
Als letzter Menüpunkt sind Statistiken vorzufinden. Diese wurden allerdings nie zu Ende entwickelt. Beim Aufruf einer Statistik treten teilweise Fehlermeldungen auf oder es passiert gar nichts.
Das Programm für die Kundenanmeldungen ist ähnlich aufgebaut. Nach der Wahl des Seminarthemas gelangt der Kunde in eine Liste mit Veranstaltungen, zu der er sich nun anmelden kann.

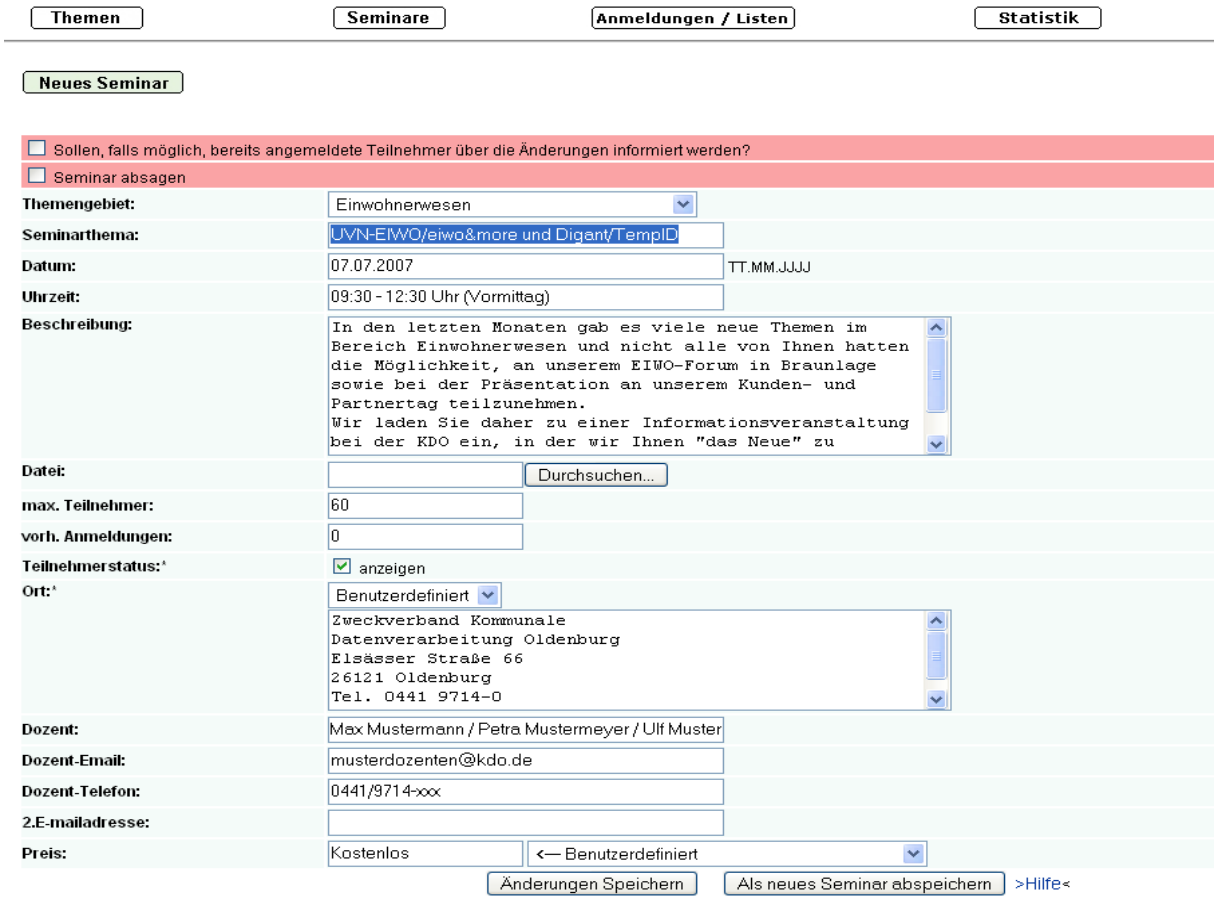


Abbildung 1: Bearbeitungsmaske

2.2 Schwachstellenanalyse

Die eingesetzte Technik (ASP) wird demnächst auf den KDO-Webservern nicht mehr betrieben. Eine Umstellung auf eine aktuelle Technik ist damit zwangsläufig notwendig.
Das gesamte Altverfahren ist relativ umständlich zu bedienen. Es sind stets mehrere Schritte notwendig, um eine gewünschte Bearbeitungsmaske zu erreichen. Außerdem sind die Bearbeitungsmöglichkeiten immer über mehrere Menüpunkte verteilt. Die Bedienung gestaltet sich dadurch für den Benutzer leicht unübersichtlich.
Die erfassten Daten enthalten viele Redundanzen (Beispiel: „Dozent“ in Abb. 1). Es gibt außer den Seminarthemen keine Möglichkeit, Stammdaten wie z.B. Dozenten zu erfassen. Dadurch entsteht unnötige Schreibarbeit und eine Integrität der Daten ist nicht gewährleistet.
Das Programm bietet zu wenig Einstellungsmöglichkeiten, um alle Veranstaltungen bearbeiten zu können. Für kommende Veranstaltungen sollen sich Kunden zu verschiedenen Vorträgen anmelden können. Außerdem sollen Kunden die Wahl zwischen mehreren frei wählbaren Zusatzoptionen haben. All diese Anforderungen erfüllt das Programm nicht. Zu Auswertungszwecken können des Weiteren keine Teilnehmerlisten exportiert werden. Veranstaltungsleiter möchten häufig gerne Auskunft über eingegangene Anmeldungen erhalten; das ist mit dem Altverfahren ebenfalls nicht möglich. Neue Benutzer müssen umständlich von Hand in die Datenbank eingearbeitet werden und haben dann auch Vollzugriff auf das gesamte System.
Da oftmals Anmeldungen nachträglich von Administratoren erfasst werden müssen, ist es außerdem sehr unpraktisch, dass man vom Administrationsbereich aus keine neuen Anmeldungen hinzufügen kann.

3 Sollkonzept

3.1 Funktionelle Anforderungen

Der Veranstaltungskalender ist in einer aktuellen Technologie neu zu entwickeln. Das Frontend für den Kunden muss per Browser aufrufbar sein, das Administrationsprogramm kann auch als Desktoplösung realisiert werden.
Es sollen Veranstaltungen der folgenden Arten verarbeitet werden können:

- ⇒ Schulungen/Seminare
- ⇒ Foren
- ⇒ Messen
- ⇒ Präsentationen

Für Schulungen und Präsentationen sind jeweils ein Dozent und ein Thema anzugeben. Für Foren und Messen müssen sich Kunden zu einzelnen Vorträgen anmelden können. Da Dozenten und Vorträge in der Regel öfter vorkommen, müssen sie als Stammdaten gepflegt werden können.
Für alle Veranstaltungsarten muss es möglich sein, neben Adressdaten auch Zusatzoptionen vom Kunden zu erfragen, zum Beispiel welches Mittagessen er bevorzugt. Diese Optionen müssen sich alle dynamisch im Administrationsprogramm verwalten lassen.
Durch eine Export-Funktion sollen sich drei Teilnehmerlisten erstellen lassen: eine nach Namen sortierte Teilnehmerliste, eine nach Behörde sortierte und eine Anforderungsliste, die die gewählten Zusatzoptionen der Kunden zeigt.
Alle Daten sollen übersichtlich in so wenigen Masken wie möglich untergebracht werden.

3.2 Qualitätsanforderungen

3.2.1 Export

Für den Export der Daten soll es möglich sein, nur Anmeldungen zu exportieren, die seit dem letzen Export dazugekommen sind. Die Listen müssen in einem frei wählbaren Verzeichnis gespeichert werden können.

3.2.2 Menü

Über das Menü des Administrationsprogramms muss sich die gewünschte Veranstaltung schnell finden lassen. Hierzu sind Filterfunktionen einzubauen. Beispielsweise sollen einem nur Veranstaltungen der Art „Präsentation“ angezeigt werden können.

3.2.3 Plausibilitätskontrollen

Bevor Benutzereingaben gespeichert werden, müssen diese auf Plausibilität geprüft werden. Es ist zum Beispiel darauf zu achten, dass alle Pflichtangaben korrekt ausgefüllt wurden und dass keine ungültigen Zeiträume ausgewählt werden. Bei Fehleingaben muss eine entsprechende Fehlermeldung erscheinen, die den Anwender auf die zu korrigierende Datenfelder aufmerksam macht.

3.2.4 Verlust einer Verbindung zur Datenbank

Schlägt die Verbindung zur Datenbank fehl, ist dem Anwender eine verständliche Fehlermeldung anzuzeigen.

3.3 Use-Case-Diagramm

Um alle Anwendungsfälle deutlich zu machen, wurden sie in einem Use-Case-Diagramm festgehalten.

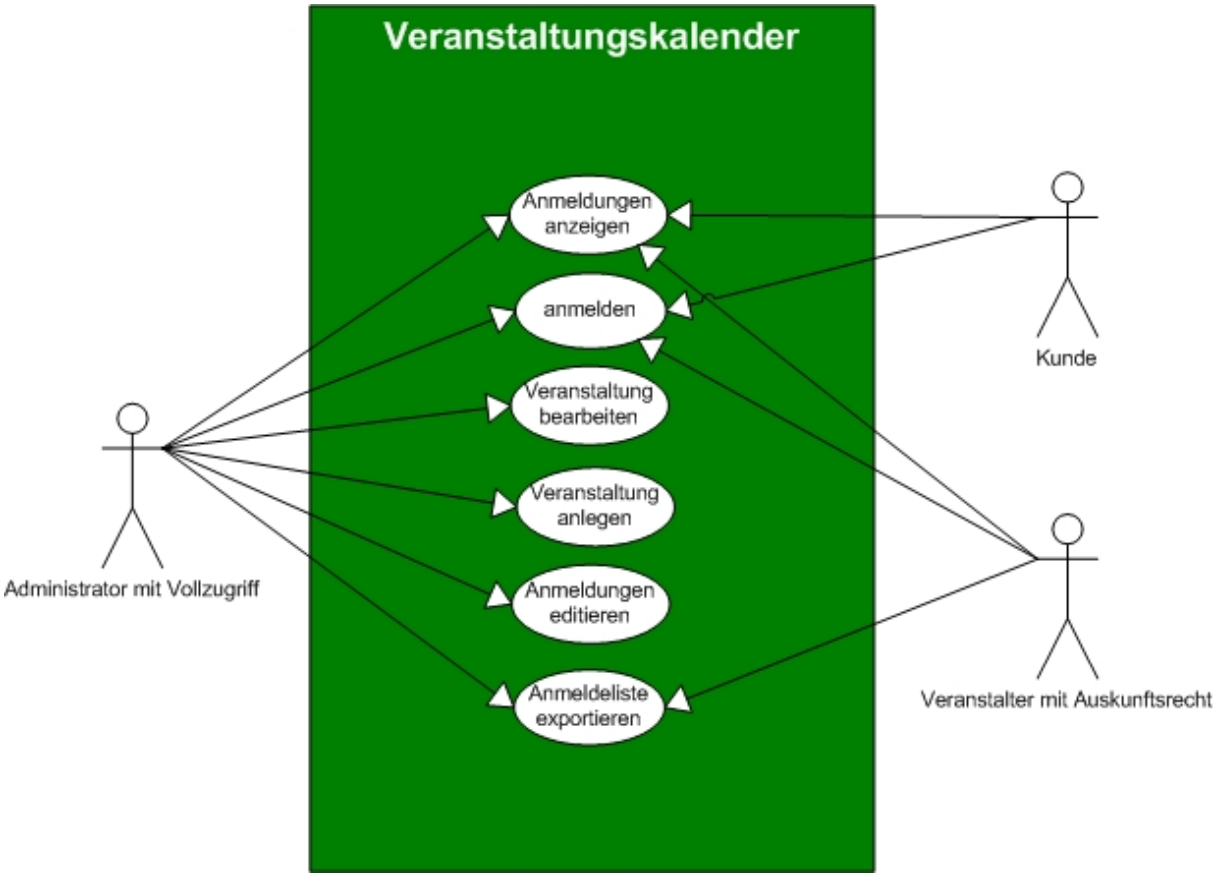


Abbildung 2: Use-Case-Diagramm

Datenverarbeitungskonzept

Inhaltsverzeichnis

1 Systembeschreibung

1.1 Systemumgebung

Der Teil des Veranstaltungskalenders, der im Internet für die Kunden sichtbar ist, wird auf dem bestehenden Webserver der KDO installiert. Das Administrationsprogramm wird als eine ausführbare JAR-Datei[1] auf dem KDO-Fileserver installiert, so dass es automatisch von jedem Arbeitsplatz-PC aus erreichbar ist.
Das MySQL-Datenbanksystem wird auf einem relativ neuen Datenbankserver installiert, der sowohl vom Fileserver als auch vom Webserver aus erreichbar ist. Die benötigten Tabellen werden in der Datenbank „veranstaltungskalender“ angelegt.

1.2 Systementwurf

Die Webapplikation ist für jeden Kunden mit einem aktuellen Browser erreichbar; spezielle Software wird nicht benötigt.
Für den Aufruf des Administrationsprogramms auf den Clients wird das JRE[2] ab der Version 1.4 benötigt. Dieses sorgt dafür, dass Java-Programme ausgeführt werden können. Da das JRE nicht zur Standardinstallation eines Windows-Betriebssystems gehört, muss es bei Bedarf noch nachinstalliert werden.
Die Java-Applikation kommuniziert über die JDBC-Schnittstelle[3] mit der Datenbank. JDBC regelt den Zugriff auf die Datenbank, gibt die SQL-Abfragen weiter und sorgt dafür, dass die Daten von Java weiterverarbeitet werden können.
Das vorgegebene Framework stellt bereits Klassen zur Abwicklung von Datenbankzugriffen zur Verfügung.
Durch das verwendete Model-View-Controller Entwurfsmuster liegt die Java-Anwendung in einer Drei-Schichten-Client/Server-Architektur vor. Datenhaltung, Darstellung und Programmlogik sind dadurch voneinander getrennt.

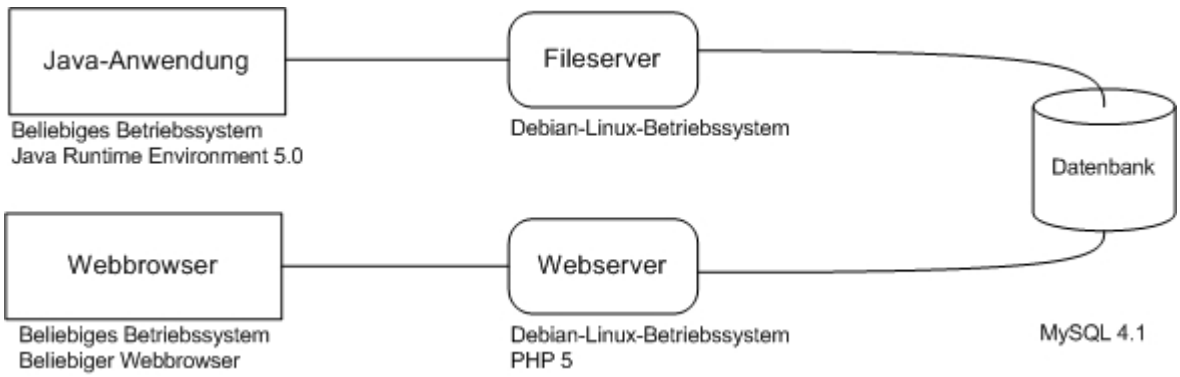


Abbildung 1: Systemaufbau

[1] Java Archive
[2] Java Runtime Environment
[3] Java Database Connectivity

2 Programmaufbau

2.1 Datenflussplan

Die grundsätzliche Verfahrensweise des Veranstaltungskalenders wird in einem Datenflussplan deutlich gemacht. Zunächst melden sich die Kunden über das Internet am System an. Die eingegangenen Anmeldungen können von den Administratoren in der Java-Anwendung bearbeitet werden. Benutzer mit Auskunftsrechten können über das Programm Einsicht in die Anmeldungen nehmen. Alle Benutzer haben zudem die Möglichkeit, Teilnehmerlisten als CSV-Dateien zu exportieren. Es gibt verschiedene Arten von Listen, zu denen neben Adressdaten auch die gewählten Anforderungen (Zusatzoptionen) gehören.
Die Datenbank wird regelmäßig nach dem Generationsprinzip gesichert.

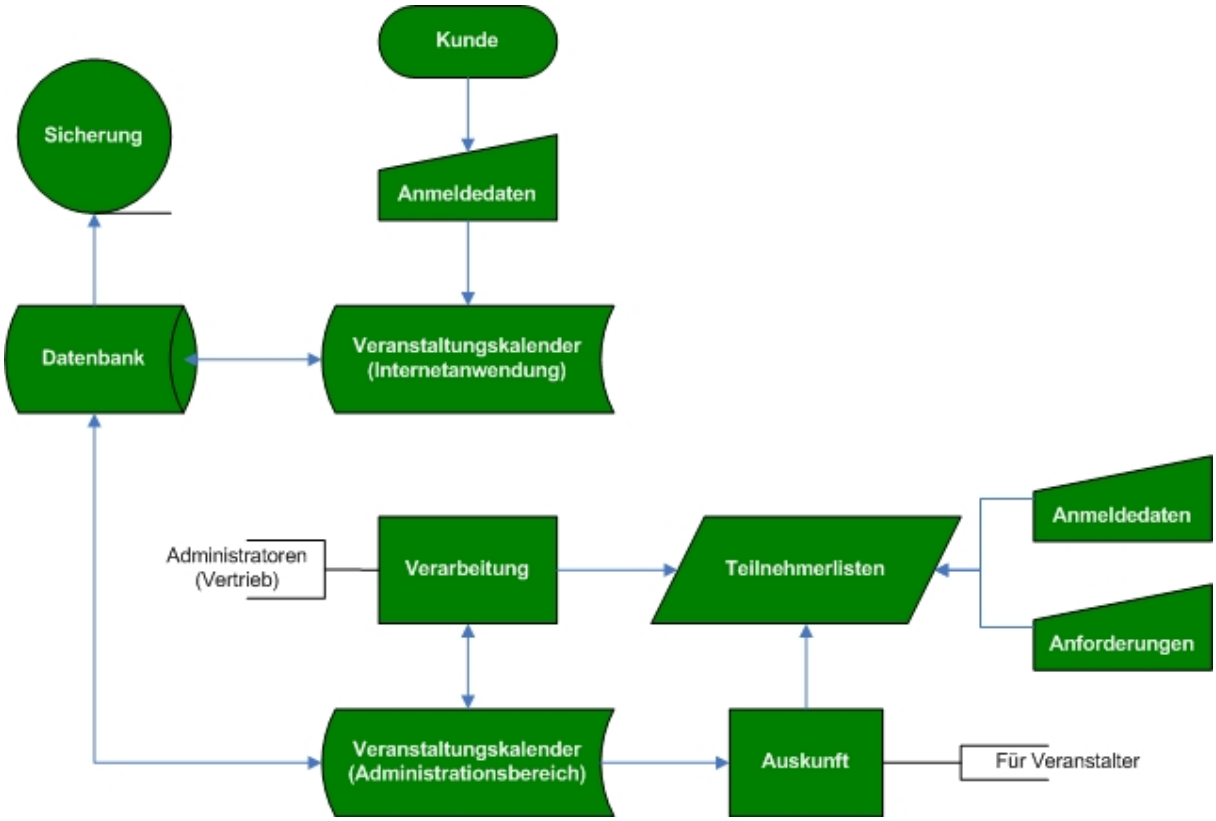


Abbildung 2: Datenflussplan

2.2 Systemaufbau als Blockdiagramm

Um den Aufbau des Java-Programms deutlich zu machen, wird der Programmaufbau zunächst als Blockdiagramm dargestellt.

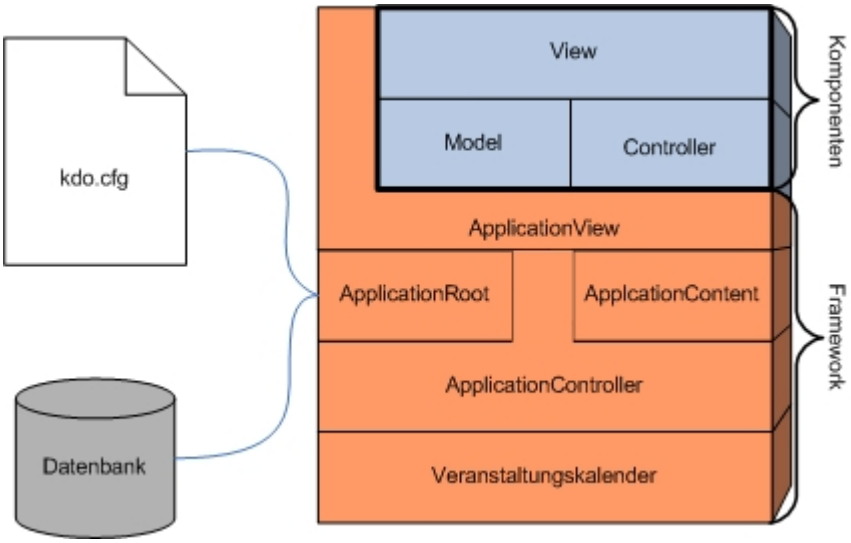


Abbildung 3: Blockdiagramm

Das Grundgerüst der Anwendung bildet ein Framework, das den grundsätzlichen Programmaufbau festlegt und steuert.
Die Klasse „Veranstaltungskalender“ ist die Basis des Programms. In der Klasse kann durch Variablen festgelegt werden, ob es sich um eine Entwicklungsversion oder eine fertige Version handelt.
Wenn es sich um eine Entwicklungsversion handelt, werden u.a. Fehlermeldungen anders angezeigt. Bei Datenbankfehlern wird z.B. auch das fehlerhafte SQL-Statement ausgegeben.
Weitere Funktionen sind in der Klasse „Veranstaltungskalender“ nicht enthalten. Beim Aufruf initialisiert sie lediglich noch die Klasse „ApplicationController“. Hier werden die bevorzugten Breiten von Schaltflächen und Textfeldern bestimmt und die Benutzeranmeldung an das System gesteuert. War die Anmeldung erfolgreich, initialisiert sie die Klassen „ApplicationRoot“, „ApplicationContent“ und „ApplicationView“.
Bei „ApplicationRoot“ handelt es sich um eine der wichtigsten Klassen des Programms. Denn hier werden Klassen initialisiert, die den Datenbankzugriff steuern. Die Verbindungsdaten werden aus der Datei kdo.cfg ausgelesen. Alle folgenden Klassen, die Datenbankzugriff haben sollen, müssen die „ApplicationRoot“-Klasse kennen.
Die Klasse „ApplicationContent“ beinhaltet Informationen über die Programmversion, den angemeldeten Benutzer und dessen Berechtigungen.
„ApplicationView“ ist die Klasse, die das Hauptfenster der Anwendung bereitstellt. Sie besteht aus einem linken Teil, in den ein Menü zur Navigation implementiert ist, und einem rechten Teil in den die einzelnen Programmkomponenten eingebettet werden.

2.3 Klassendiagramm

Das folgende Diagramm bildet die Klassen einer der Programmkomponenten ab. Als Beispiel ist die Komponente Seminarthema gewählt worden, da sie relativ wenige Daten beinhaltet.
Die Daten sind in Instanzen der Klasse „SeminarThema“ gespeichert. Wichtige Daten sind die Bezeichnung, also das Thema und die dazugehörige ID. Die Seminarthemen werden vom Model aus der Datenbank ausgelesen und in eine Liste geschrieben. Der Controller steuert die Verbindung zwischen dem Model und der View. Das Model kann beliebig viele Seminarthemen beinhalten. Sie werden in einer Arraylist verwaltet. Der Controller wird von der Klasse „KDOPanelController“ abgeleitet. Diese stellt die wichtigsten Funktionen für Controller zur Verfügung. Außer den Klassen „SeminarThema“ und „SeminarThemaModel“ startet der Controller außerdem die View. Im Fall der „SeminarThemaView“ handelt es sich dabei um eine Maske, die eine Toolbar, ein Textfeld und einen Button zum Speichern sowie einen zum Schließen der View besitzt. Da die Buttons „Speichern“ und „Schließen“ in vielen Views erscheinen, werden sie in ein separates Panel ausgelagert. Das erspart unnötigen Programmcode.

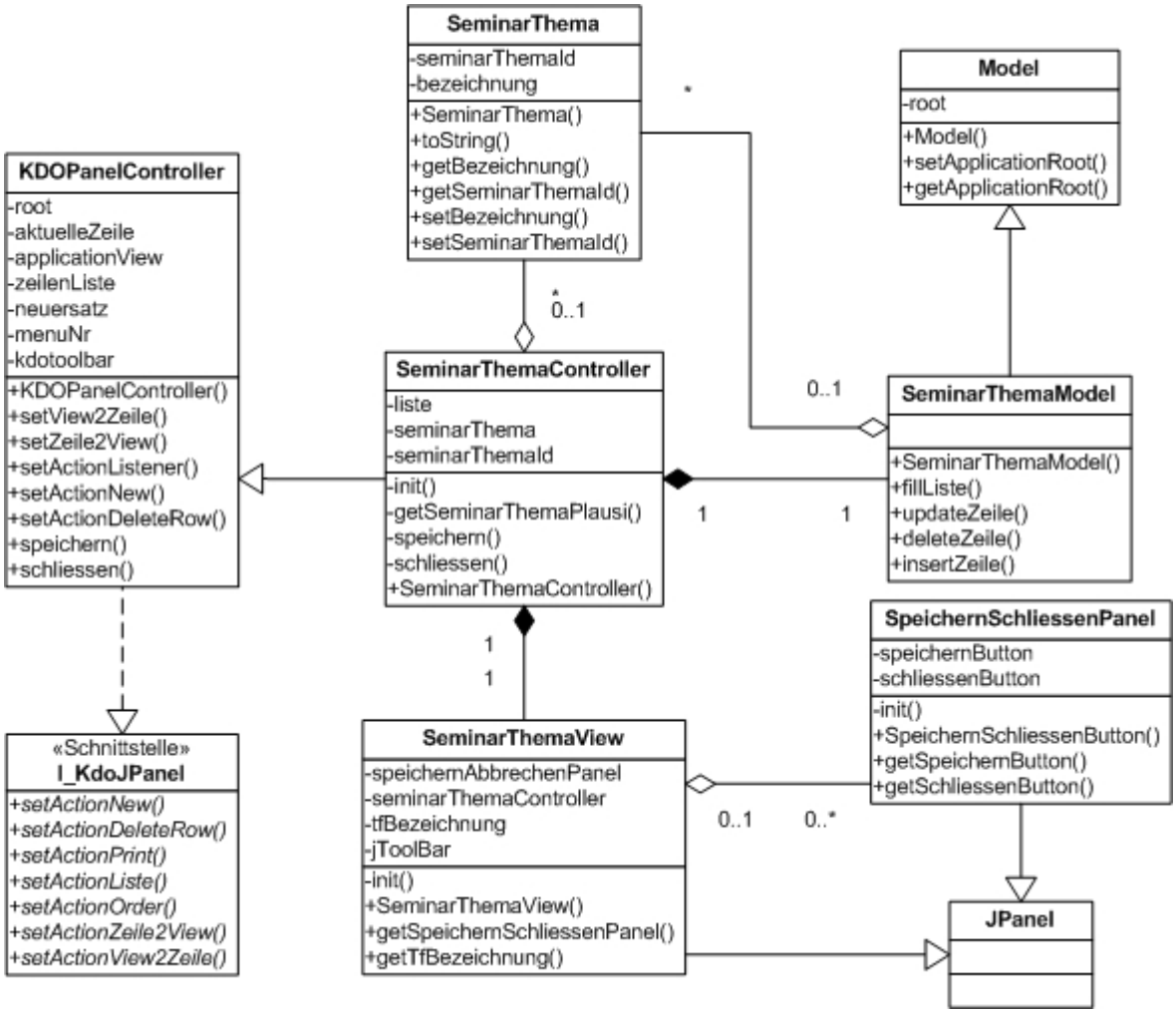


Abbildung 4: Klassendiagramm "Seminarthema"

2.4 Datenbank

2.4.1 ERM

Um die Datenbank zu planen, wird zunächst ein ERM erstellt, das die zu verarbeitenden Daten und ihre Beziehung zueinander darstellt. Das ERM ist allgemein beschrieben.

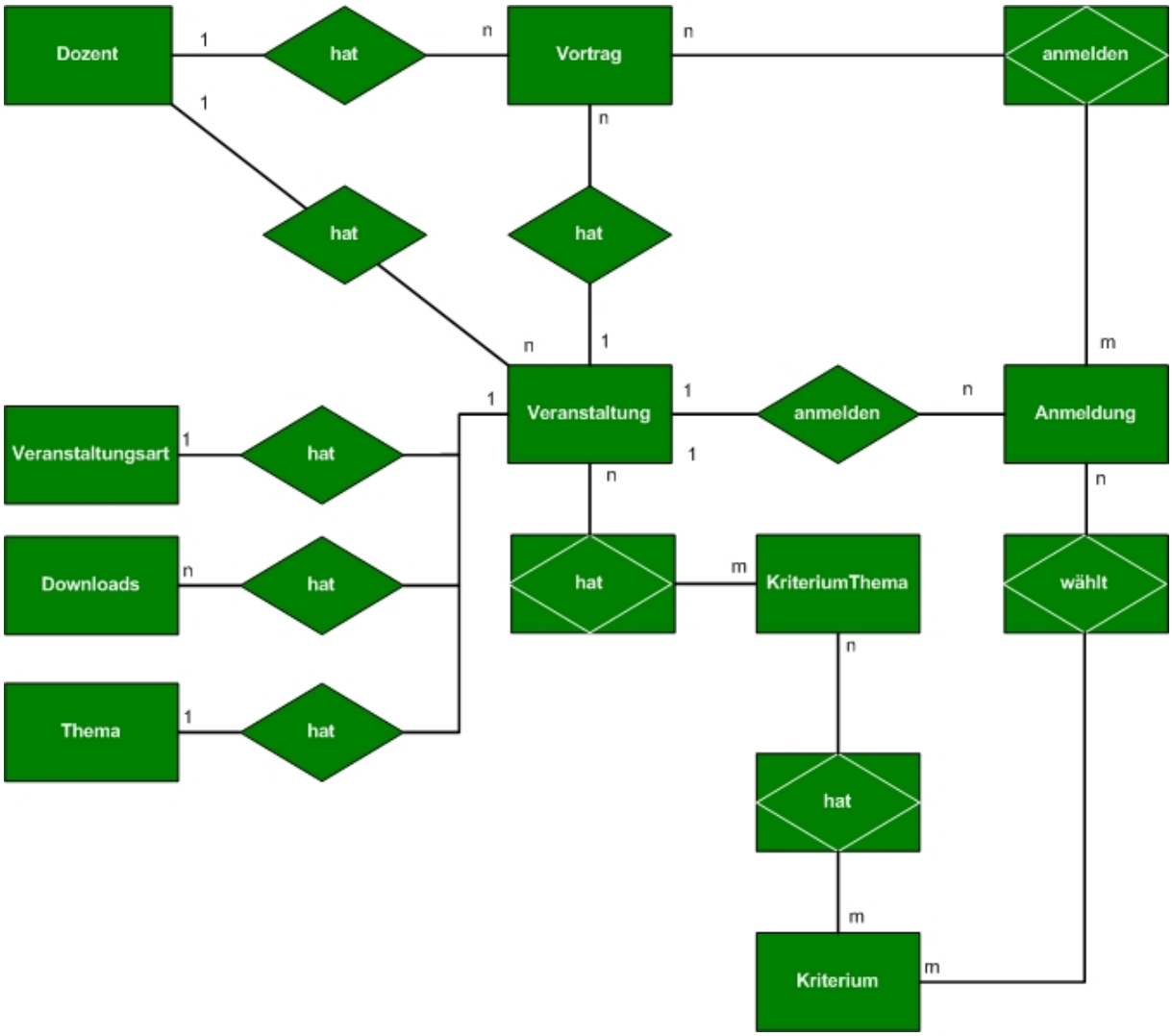


Abbildung 5: ERM

2.4.2 Datenbankstruktur

Das folgende Diagramm beschreibt den genauen Aufbau der Tabellen.

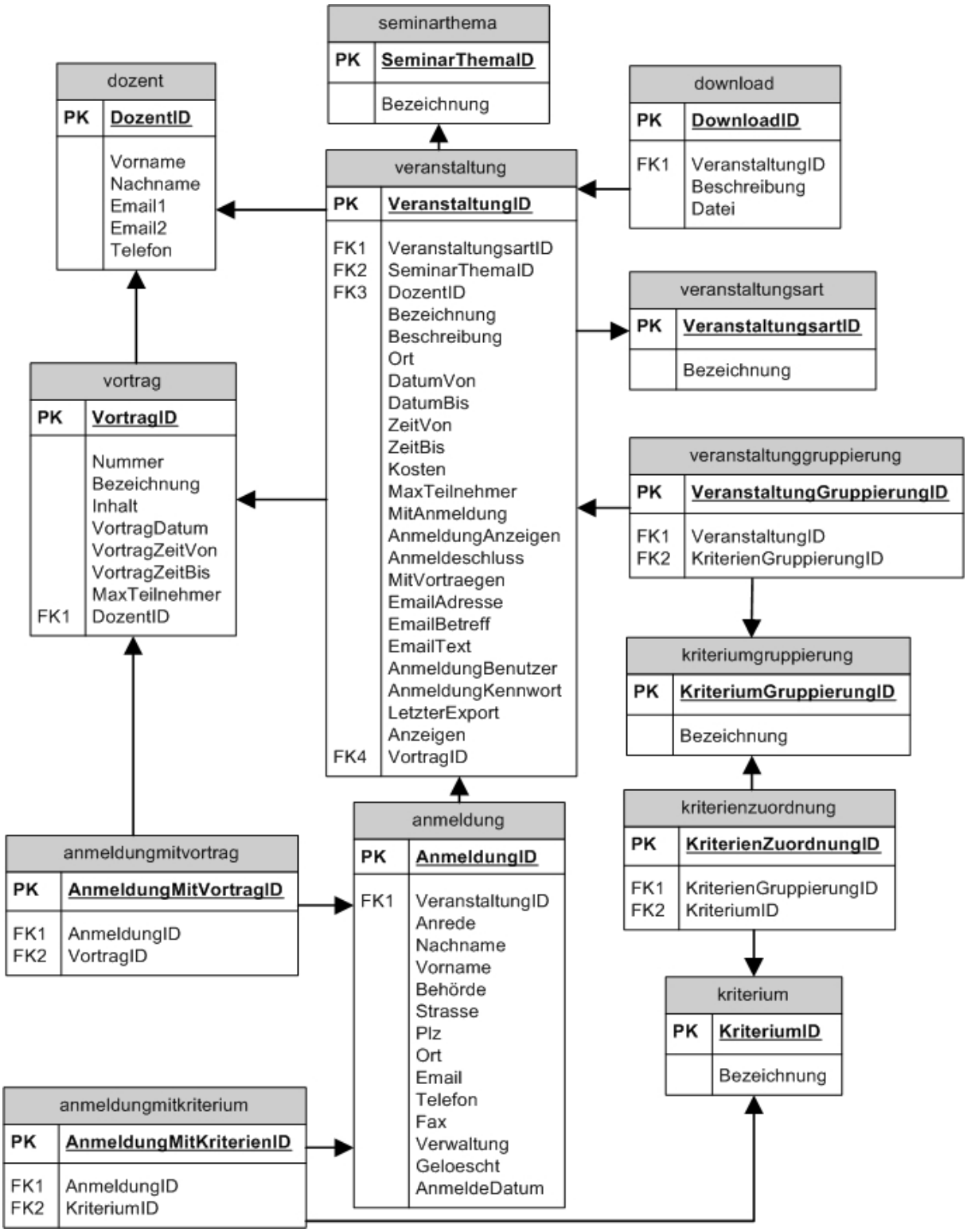


Abbildung 6: Datenbankstruktur

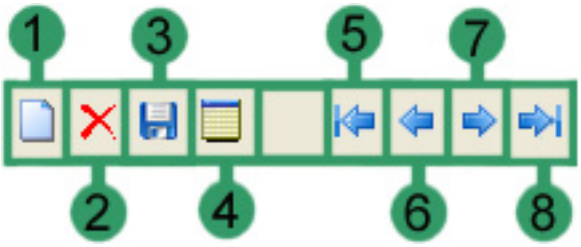
Auszüge aus dem Benutzerhandbuch

Inhaltsverzeichnis

1 Stammdaten

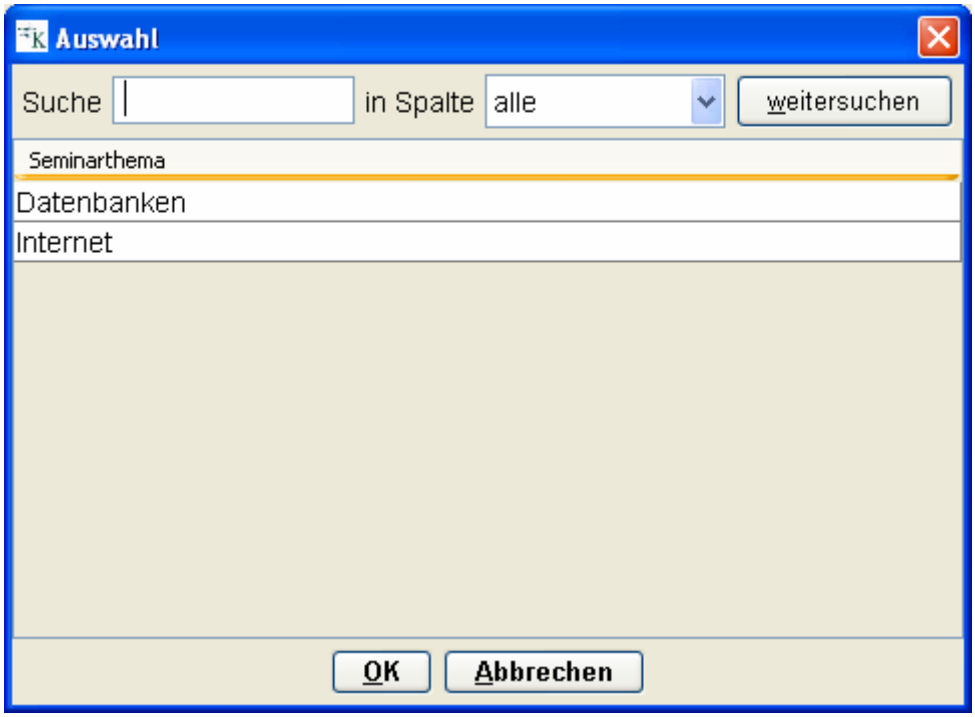
1.1 Dozenten, Seminarthemen und Vorträge

Diese drei Funktionen sind ähnlich aufgebaut. Sie werden alle über eine einheitliche Symbolleiste gesteuert.

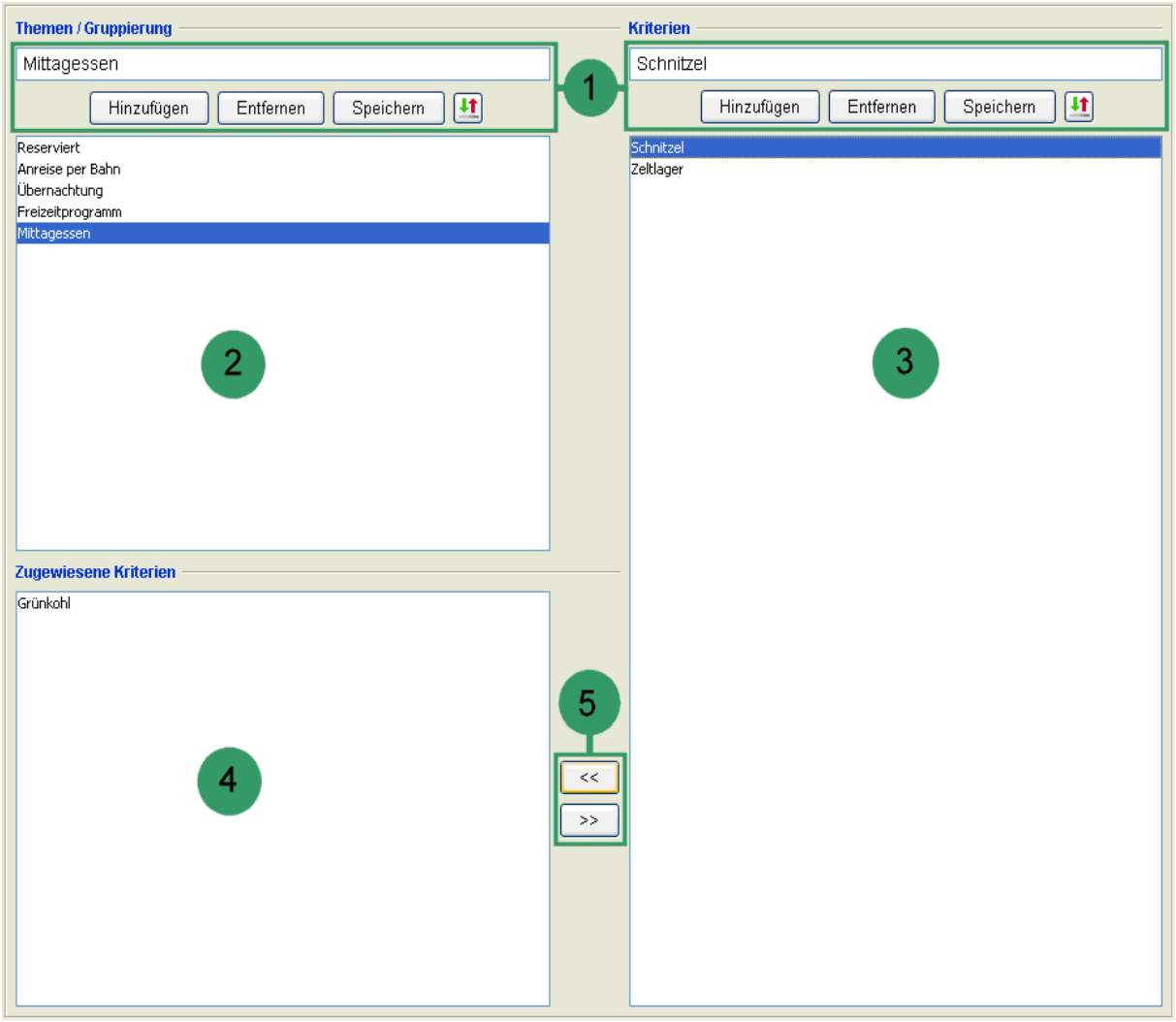


Punkt	Bedeutung
1	Neuen Datensatz anlegen
2	Datensatz löschen
3	Datensatz speichern
4	Liste anzeigen
5	Erster Datensatz
6	Vorheriger Datensatz
7	Nächster Datensatz
8	Letzter Datensatz

Alternativ kann ein Datensatz auch über die Schaltfläche Speichern gespeichert werden. Die folgende Abbildung zeigt anhand des Beispiels „Seminarthemen“ die Liste, die erscheint wenn in der Symbolleiste der Punkt 4 ausgewählt wurde. Hier können Sie nach einem Seminarthema suchen. Durch Auswahl der Schaltfläche OK wird das ausgewählte Seminarthema in der Bearbeitungsmaske angezeigt.

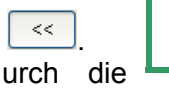


1.2 Kriterien



Punkt	Bedeutung
1	Themen/Kriterien bearbeiten
2	Liste aller Themen
3	Liste aller freien Kriterien
4	Liste aller zugewiesenen Kriterien
5	Buttons zum zuweisen/entfernen von Kriterien zu einem bestimmten Thema

Mit dieser Funktion können die Zusatz-Optionen für Kunden konfiguriert werden. In dem obigen Beispiel wurde dem Thema „Mittagessen“ bereits das Kriterium „Grünkohl“ zugewiesen. Das Kriterium „Zeltlager“ könnte z.B. dem Thema „Freizeitprogramm“ zugeordnet werden.

Dies geschieht über die Schaltfläche 

. Andersherum ist es möglich, durch die Schaltfläche 

ein Kriterium auch wieder zu entfernen. Allerdings nur, wenn noch keine Kunden dieses Kriterium gewählt haben.

Mit der Schaltfläche  können die Listen neu geordnet werden.

2 System

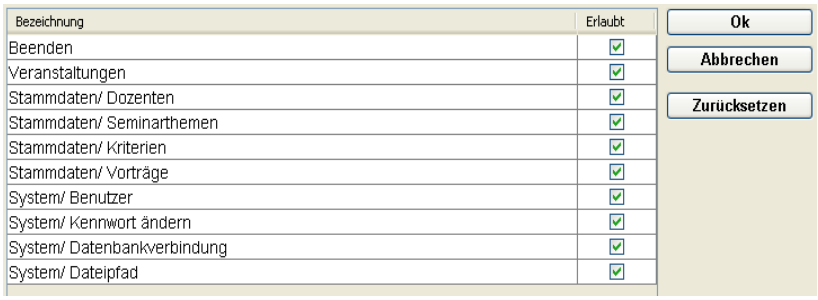
2.1 Benutzer

In dieser Maske können Sie, eine entsprechende Berechtigung vorausgesetzt, die Benutzer des Systems verwalten bzw. neu erfassen. Dies geschieht, wie bei den Stammdaten, über eine Symbolleiste.
Die Felder „Anmeldename“ und „Kennwort“ sind Pflichtfelder. Zusätzlich kann in dieser Programmfunktion ein Haken gesetzt werden, der dem ausgewählten Benutzer lediglich Auskunftsrecht erstattet. In diesem Fall kann der Benutzer keine Daten verändern, aber lesen und Anmeldelisten exportieren.

2.2 Zugriffsrechte

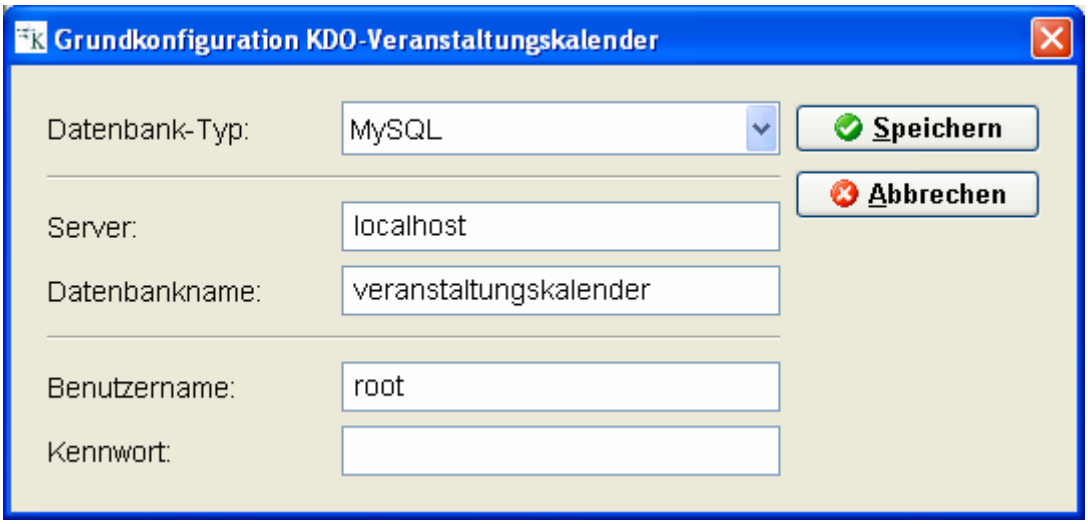
Über die Schaltfläche 

am unteren Bildschirmrand können die Menüpunkte festgelegt werden, die der Benutzer aufrufen darf.



2.3 Datenbankverbindung

Im Punkt Datenbankverbindung kann die verwendete Datenbank festgelegt werden. In der Regel müssen diese Einstellungen nicht verändert werden.
Ändern Sie diese Einstellungen bitte nur, wenn Sie diese Anweisung vom Administrator bekommen haben!



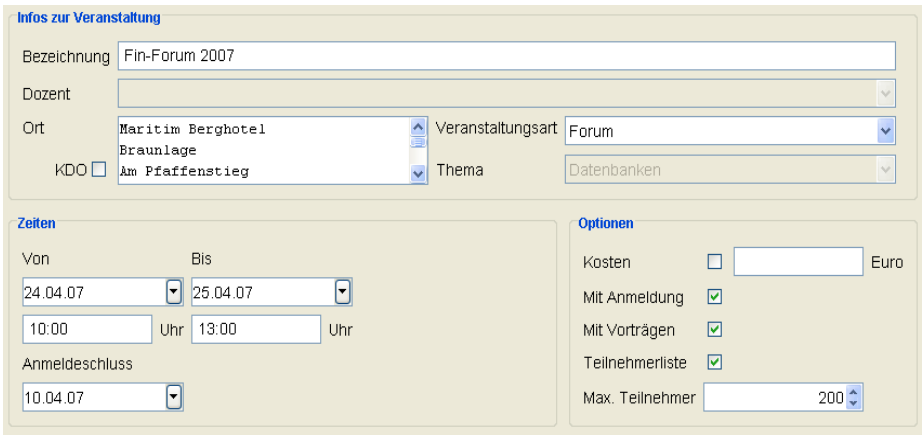
2.4 Dateipfad

Mit der Programmfunktion „Dateipfad“ können Sie den Standardpfad für die Export-Dateien festlegen. Dieser Pfad kann später noch geändert werden.

3 Veranstaltungen

3.1 Grundeinstellungen

Die Eingabemaske für die Grundeinstellungen bleibt immer sichtbar, egal welchen Reiter bei den Veranstaltungen Sie ausgewählt haben. Die Eingabefelder „Dozent“ und „Thema“ erscheinen nur, wenn als Veranstaltungsart „Seminar“ oder „Präsentation“ ausgewählt wurde. Über das Häkchen „KDO“ kann das Feld „Ort“ automatisch mit der Anschrift der KDO vorbelegt werden.



Quellcode u. Screenshots

Inhaltsverzeichnis

1 Quellcode

1.1 Klasse ThemenModel

```
package de.kdo.vk.daten;

import java.sql.ResultSet;
import java.sql.SQLException;
import de.kdo.cs.tools.CsError;
import de.kdo.vk.ApplicationRoot;
import de.kdo.vk.model.Model;
import de.kdo.vk.model.KriteriumThemenListModel;

/**
 * <p>Title: Veranstaltungskalender</p>
 * <p>Description: Themen-Model</p>
 * <p>Copyright: Copyright (c) 2007</p>
 * <p>Company: KDO</p>
 * @author Torsten Oehl
 * @version 1.0
 */
public class ThemenModel extends Model{
/**
 * Konstruktor
 * @param ApplicationRoot p0
 */
public ThemenModel(ApplicationRoot p0){
      super(p0);
}

/**
 * Methode zum Einfügen eines neuen Datensatzes
 * @param String id
 * @param String bezeichnung
 * @return boolean
 */
public boolean insertDaten(String id, String bezeichnung) {
      // Element in Datenbank schreiben
      String sql = "INSERT INTO kriteriumgruppierung
      (KriteriumGruppierungID,Bezeichnung) VALUES
      ('"+id+"','"+bezeichnung+"')";
      if(this.getRoot().getDatabaseAccess().setQueryStatment(sql)){
            return true;
      } else {
            return false;
      }
}

/**
 * Methode zum Updaten eines Datensatzes
 * @param String id
 * @param String bezeichnung
 * @return boolean
 */
```

```
public boolean updateDaten(String id, String bezeichnung) {
	// Element in Datenbank updaten
	String sql = "UPDATE kriteriumgruppierung SET
	Bezeichnung='"+bezeichnung+"' WHERE KriteriumGruppierungID='"+id+"'";
	if(this.getRoot().getDatabaseAccess().setQueryStatment(sql)){
		return true;
	} else {
		return false;
	}
}

/**
 * Methode zum Löschen eines neuen Datensatzes
 * @param String id
 * @param String bezeichnung
 * @return boolean
 */
public boolean deleteDaten(String id, String bezeichnung) {
	// Element aus der Datenbank löschen
	String sql = "DELETE FROM kriteriumgruppierung WHERE
	KriteriumGruppierungID='"+id+"'";
	if(this.getRoot().getDatabaseAccess().setQueryStatment(sql)){
		return true;
	} else {
		return false;
	}
}

/**
 * Methode zum Erstellen eines neuen ListModels
 * @return KriteriumThemenListModel
 */
public KriteriumThemenListModel createThemenListe(){
	KriteriumThemenListModel m = new KriteriumThemenListModel();
	ResultSet resultSet = null;

	// Datenbank-Abfrage
	resultSet = this.getRoot().getDatabaseAccess().getResultset("SELECT *
	FROM kriteriumgruppierung ORDER BY KriteriumGruppierungID DESC");

	// ListModel mit Daten füllen
	try {
		while(resultSet.next()){
			Thema k = new
			Thema(resultSet.getString("KriteriumGruppierungID"),
			resultSet.getString("Bezeichnung"));
			m.addElement(k);
		}
	} catch (SQLException e) {
		CsError.message("Fehler beim Auslesen der Daten!", e, true);
		System.exit(0);
	}
	return m;
}

}
```

1.2 Klasse Uhrzeit

```
package de.kdo.cs.tools;

import java.text.ParseException;

/**
 * <p>Title: Uhrzeit</p>
 * <p>Description: Uhrzeiten vergleichen</p>
 * <p>Copyright: Copyright (c) 2007</p>
 * <p>Company: KDO</p>
 * @author Torsten Oehl
 * @version 1.0
 */
public class Uhrzeit {

private int stunde;
private int minute;

/**
 * Konstruktor
 * @param String uhrzeit (##:##)
 */
public Uhrzeit(String uhrzeit) {
      try{
            this.stunde = Integer.parseInt(uhrzeit.substring(0, 2));
            this.minute = Integer.parseInt(uhrzeit.substring(3, 5));
      } catch(NumberFormatException e){
            this.stunde = 0;
            this.minute = 0;
      }
}

/**
 * Standard-Konstruktor
 */
public Uhrzeit(){
      this.stunde = 0;
      this.minute = 0;
}

/**
 * Getter für Stunde
 * @return int
 */
public int getStunde(){
      return this.stunde;
}

/**
 * Getter für Minute
 * @return int
 */
public int getMinute(){
      return this.minute;
}
```

```
/**
 * Setter für Stunde
 * @param String stunde
 * @throws ParseException
 */
public void setStunde(String stunde) throws ParseException {
	this.stunde = Integer.parseInt(stunde);
}

/**
 * Setter für Minute
 * @param String minute
 * @throws ParseException
 */
public void setMinute(String minute) throws ParseException {
	this.minute = Integer.parseInt(minute);
}

/**
 * Prüft, ob Zeit gültig ist
 * @return boolean
 */
public boolean istZeit() {
	if (this.stunde < 0 || this.stunde > 24)
		return false;
	if (this.minute < 0 || this.minute > 59)
		return false;
	return true;
}

/**
 * Prüft, ob Zeitraum gültig ist
 * @param Uhrzeit Anfangszeit
 * @param Uhrzeit Endzeit
 * @return boolean
 */
public static boolean istZeitraum(Uhrzeit anfang, Uhrzeit ende){
	// Wenn Format korrekt ist
	if(anfang.istZeit() && ende.istZeit()){
		// Wenn Anfangsstunde kleiner oder gleich Endstunde
		if(anfang.getStunde() <= ende.getStunde()){
			// Wenn Anfangsstunde gleich Endstunde
			if(anfang.getStunde() == ende.getStunde()){
				// Wenn Anfangsminute kleiner oder gleich Endminute
				if(anfang.getMinute() <= ende.getMinute()){
					return true;
				}
			} else {
				return true;
			}
		}
	}
	return false;
}

}
```

2 Screenshots

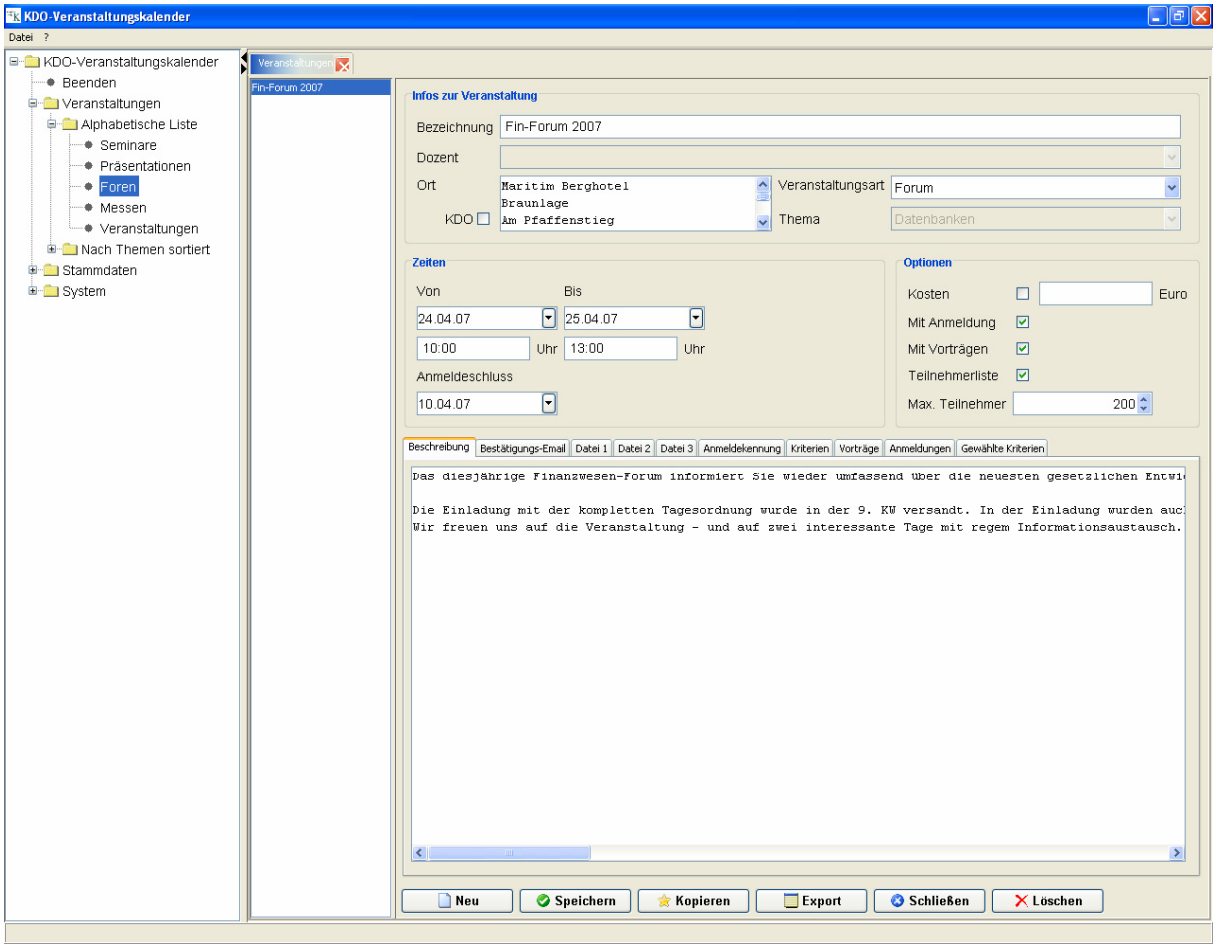


Abbildung 1: Java-Applikation (Bearbeitung von Veranstaltungen)

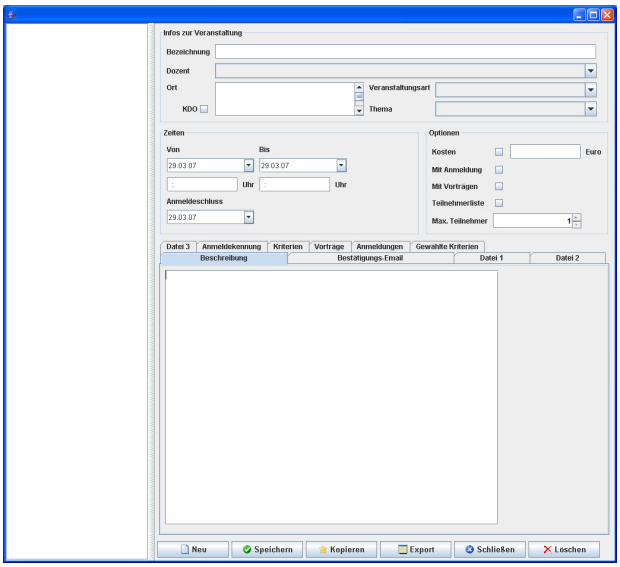


Abbildung 2: View gestartet ohne Kenntnis von Controller oder Model