Abschlussbericht/ Pflichtenheft

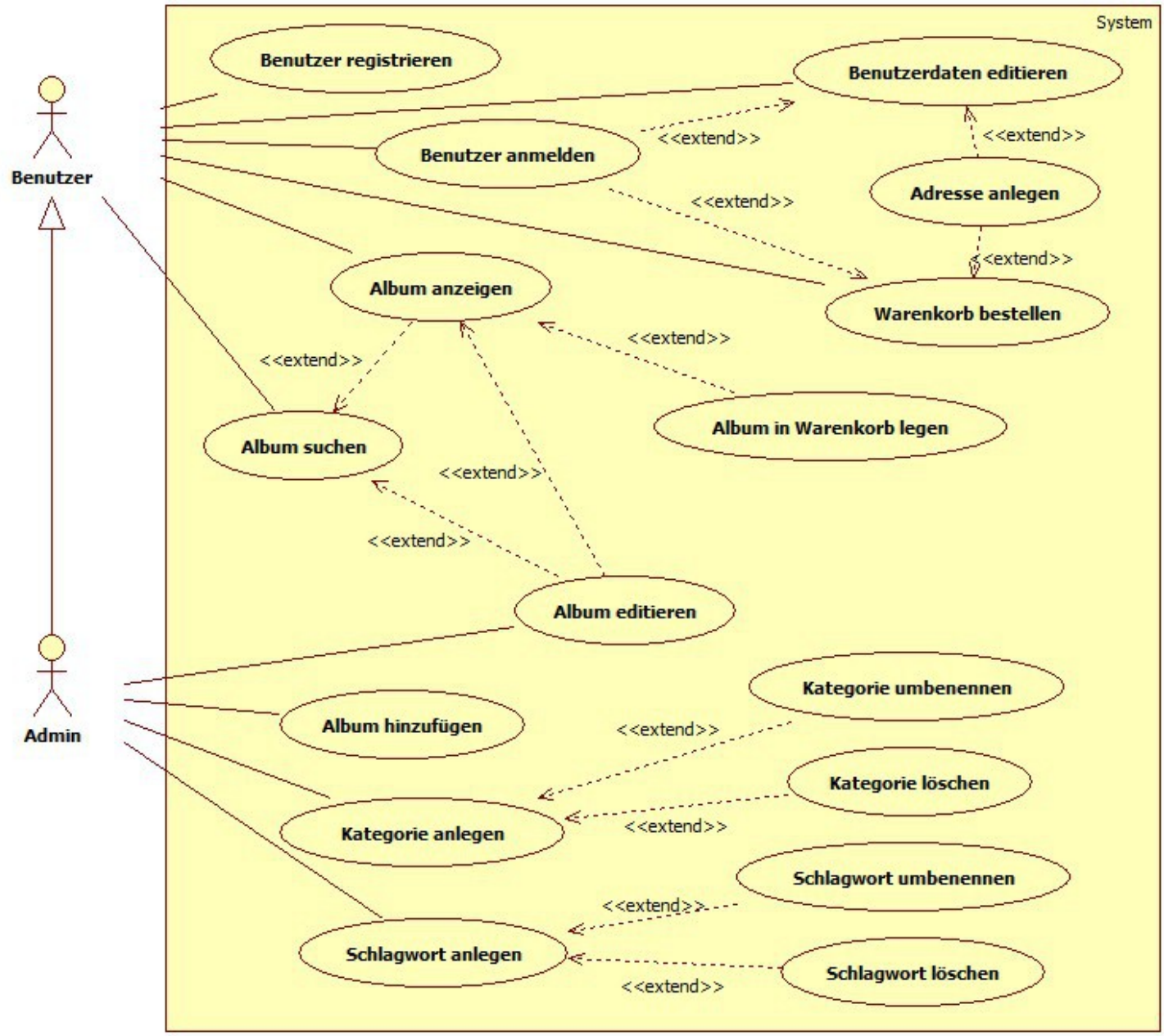
Projekt: MP3-Webshop

Gruppe B (ZVAFZWECTS)

Mitglieder:
~~***Christoph Antes***~~
Andreas Baur
Markus Henn
Jochen Pätzold
~~***Christian Zöllner***~~

Inhaltsverzeichnis

1. Use-Cases


System
Benutzer registrieren
Benutzerdaten editieren
Benutzer anmelden
Adresse anlegen
Album anzeigen
Warenkorb bestellen
Album suchen
Album in Warenkorb legen
Album editieren
Kategorie umbenennen
Album hinzufügen
Kategorie löschen
Kategorie anlegen
Schlagwort umbenennen
Schlagwort anlegen
Schlagwort löschen
<<extend>>
Benutzer
Admin

1.1. Allgemeine Use-Cases

Diese Use-Cases stehen allen Benutzern des Systems zur Verfügung.

1.1.1. Use-Case „Benutzer anmelden“

Use Case	Benutzer anmelden
Vorbedingung	Benutzer ist nicht angemeldet, befindet sich auf der “Login”-Seite
Nachbedingung Erfolg	Benutzer angemeldet
Nachbedingung Fehlschlag	Benutzer nicht angemeldet
Akteure	Benutzer
Auslösendes Ereignis	Aktion des Benutzers
Beschreibung	1. Benutzer gibt E-Mail und Passwort ein 2. Benutzer bestätigt seine Eingaben, System liefert Rückmeldung
Erweiterung	a. (zu 1): Benutzer wählt “neuer Benutzer/registrieren”, Use-Case “Benutzer registrieren”
Alternativen	A. (zu 1-2): Abbruch

1.1.2. Use-Case „Benutzer registrieren“

Use Case	Benutzer registrieren
Vorbedingung	Benutzer ist nicht angemeldet, Benutzer befindet sich auf der “registrieren”-Seite
Nachbedingung Erfolg	Benutzer angemeldet, Benutzerdaten im System gespeichert
Nachbedingung Fehlschlag	Benutzer nicht angemeldet, Benutzerdaten verworfen
Akteure	Benutzer
Auslösendes Ereignis	Aktion des Benutzers
Beschreibung	1. Benutzer gibt Benutzerdaten ein 2. Benutzer bestätigt seine Eingaben, System speichert Benutzerdaten und meldet Benutzer an
Erweiterung	---
Alternativen	A. Falls schon ein Benutzer mit der E-Mail-Adresse existiert, Fehleranzeige; weiter mit 1 B. (zu 1-2): Abbruch

1.1.3. Use-Case „Benutzerdaten editieren“

Use Case	Benutzerdaten editieren
Vorbedingung	Benutzer ist angemeldet, Benutzer befindet sich auf der “Benutzerdaten editieren”-Seite
Nachbedingung Erfolg	Benutzerdaten im System aktualisiert
Nachbedingung Fehlschlag	Benutzerdaten unverändert
Akteure	Benutzer
Auslösendes Ereignis	Aktion des Benutzers
Beschreibung	1. Benutzer editiert Benutzerdaten 2. Benutzer bestätigt seine Eingaben, System speichert Benutzerdaten
Erweiterung	a. (zu 1): Benutzer wählt “Adresse hinzufügen”, Use-Case “Adresse anlegen”
Alternativen	A. (zu 2): Sind Benutzerdaten ungültig, zeigt System Fehlermeldung, weiter mit 1. B. (zu 1-2): Abbruch

1.1.4. Use-Case „Adresse anlegen“

Use Case	Adresse anlegen
Vorbedingung	Benutzer ist angemeldet
Nachbedingung Erfolg	Adresse ist im System hinterlegt
Nachbedingung Fehlschlag	Adresse verworfen
Akteure	Benutzer
Auslösendes Ereignis	Aktion des Benutzers
Beschreibung	1. Benutzer gibt Adressdaten ein 2. Benutzer bestätigt die Adresse, System speichert Adresse
Erweiterung	---
Alternativen	A. (zu 2): Ist Adresse ungültig, zeigt System Fehlermeldung, weiter mit 1. B. (zu 1-2): Abbruch

1.1.5. Use-Case „Album anzeigen“

Use Case	Album anzeigen
Vorbedingung	Benutzer befindet sich auf der “Album anzeigen”-Seite
Nachbedingung Erfolg	Album wird angezeigt
Nachbedingung Fehlschlag	---
Akteure	Benutzer
Auslösendes Ereignis	Aktion des Benutzers
Beschreibung	1. Album wird angezeigt
Erweiterung	a. (zu 1): Akteur startet Wiedergabe eines Tracks, System startet Wiedergabe des Tracks b. (zu 1): Use-Case “Album in Warenkorb legen” c. (zu 1): Falls Akteur angemeldeter Admin, Use-Case “Album editieren”
Alternativen	A. (zu 1): Abbruch

1.1.6. Use-Case „Album suchen“

Use Case	Album suchen
Vorbedingung	Benutzer befindet sich auf der “Album suchen”-Seite
Nachbedingung Erfolg	Suchergebnisse werden angezeigt
Nachbedingung Fehlschlag	---
Akteure	Benutzer
Auslösendes Ereignis	Aktion des Benutzers
Beschreibung	1. Suchkriterien eingeben 2. Suchkriterien abschicken, System zeigt Suchergebnisse an
Erweiterung	a. (zu 2): User wählt ein Suchergebnis aus, Use-Case “Album anzeigen” b. (zu 2): Falls Benutzer angemeldeter Admin, Use-Case “Album editieren”
Alternativen	A. (zu 1-2): Abbruch

1.1.7. Use-Case „Album in Warenkorb legen“

Use Case	Album in Warenkorb legen
Vorbedingung	Album wird angezeigt
Nachbedingung Erfolg	Album in angegebener Anzahl im Warenkorb hinterlegt
Nachbedingung Fehlschlag	Album nicht im Warenkorb hinterlegt
Akteure	Benutzer
Auslösendes Ereignis	Aktion des Benutzers
Beschreibung	1. Benutzer wählt Stückzahl Bestellungen für angezeigtes Album 2. Benutzer bestätigt Bestellung, System legt Album in ausgewählter Stückzahl in Warenkorb und aktualisiert Warenkorb-Darstellung
Erweiterung	---
Alternativen	A. (zu 2): Ist das Album nicht mehr in entsprechender Stückzahl vorhanden, wird eine Fehlermeldung angezeigt; weiter mit 1 B. (zu 1-2): Abbruch

1.1.8. Use-Case „Warenkorb bestellen“

Use Case	Warenkorb bestellen
Vorbedingung	Warenkorb ist nicht leer
Nachbedingung Erfolg	Warenkorb ist leer, Bestellung erfolgt, Stückzahlen des Lagerbestandes der Ware im Korb aktualisiert
Nachbedingung Fehlschlag	Warenkorb unverändert, keine Bestellung getätigt, Lagerbestand der Waren unverändert
Akteure	Benutzer
Auslösendes Ereignis	Aktion des Benutzers
Beschreibung	1. Benutzer wählt Rechnungs- und Lieferadresse 2. Benutzer bestätigt Bestelldaten, System speichert Bestellung, leert Warenkorb und aktualisiert Lagerstand (Stückzahl) der bestellten Waren
Erweiterung	a. (zu 1): Falls der Benutzer nicht angemeldet ist, Use Case “Benutzer anmelden”, weiter mit 1. b. (zu 1): Benutzer ändert Bestellmengen von Warenkorb-Elementen, System aktualisiert Bestellmenge, weiter mit 1.

	c. (zu 1): Falls keine Adresse angelegt, Use-Case “Adresse anlegen”, weiter mit 1.
Alternativen	A. (zu 1-2): Abbruch

1.2. Admin Use-Cases

1.2.1. Use-Case „Kategorie anlegen“

Use Case	Kategorie anlegen
Vorbedingung	Benutzer ist als Admin angemeldet und befindet sich auf der “Kategorien verwalten”-Seite
Nachbedingung Erfolg	Kategorie ist angelegt
Nachbedingung Fehlschlag	Kategorien unverändert
Akteure	Admin
Auslösendes Ereignis	Aktion des Benutzers
Beschreibung	1. System zeigt Liste bestehender Kategorien 2. Nutzer gibt einen Kategorie-Namen ein 3. Nutzer bestätigt seine Eingabe, System liefert Rückmeldung
Erweiterung	a. (zu 1): Benutzer wählt eine Kategorie und “umbenennen”, Use-Case “Kategorie umbenennen” b. (zu 1): Use-Case “Kategorie löschen”
Alternativen	A. (zu 3): Falls Kategorie schon vorhanden, zeigt System Fehlermeldung, weiter mit 2. B. (zu 1-3): Abbruch

1.2.2. Use-Case „Kategorie umbenennen“

Use Case	Kategorie umbenennen
Vorbedingung	Benutzer ist als Admin angemeldet und befindet sich auf der “Kategorie umbenennen”-Seite
Nachbedingung Erfolg	Kategorie ist umbenannt
Nachbedingung Fehlschlag	Kategorie unverändert
Akteure	Admin
Auslösendes Ereignis	Aktion des Benutzers
Beschreibung	1. Nutzer gibt einen neuen Kategorie-Namen ein 2. Nutzer bestätigt seine Eingabe, System liefert Rückmeldung
Erweiterung	---
Alternativen	A. (zu 2): Falls schon eine Kategorie mit dem

	angegebenen Namen vorhanden, zeigt System Fehlermeldung, weiter mit 1. B. (zu 1-2): Abbruch

1.2.3. Use-Case „Kategorie löschen“

Use Case	Kategorie löschen
Vorbedingung	Benutzer ist als Admin angemeldet und befindet sich auf der “Kategorien verwalten”-Seite
Nachbedingung Erfolg	Kategorie ist gelöscht
Nachbedingung Fehlschlag	Kategorie besteht weiterhin
Akteure	Admin
Auslösendes Ereignis	Aktion des Benutzers
Beschreibung	1. Nutzer wählt Kategorie 2. Nutzer bestätigt Auswahl, System löscht Kategorie, liefert Rückmeldung
Erweiterung	---
Alternativen	A. (zu 1-2): Abbruch

1.2.4. Use-Case „Schlagwort anlegen“

Use Case	Schlagwort anlegen
Vorbedingung	Benutzer ist als Admin angemeldet und befindet sich auf der “Schlagworte verwalten”-Seite
Nachbedingung Erfolg	Schlagwort ist angelegt
Nachbedingung Fehlschlag	Schlagwort unverändert
Akteure	Admin
Auslösendes Ereignis	Aktion des Benutzers
Beschreibung	1. System zeigt Liste bestehender Schlagworte 2. Nutzer gibt einen Schlagwort-Namen ein 3. Nutzer bestätigt seine Eingabe, System liefert Rückmeldung
Erweiterung	a. (zu 1): Benutzer wählt ein Schlagwort und “umbenennen”, Use-Case “Schlagwort umbenennen” b. (zu 1): Use-Case “Schlagwort löschen”
Alternativen	A. (zu 3): Falls Schlagwort schon vorhanden, zeigt System Fehlermeldung, weiter mit 2.

	B. (zu 1-3): Abbruch

1.2.5. Use-Case „Schlagwort umbenennen“

Use Case	Schlagwort umbenennen
Vorbedingung	Benutzer ist als Admin angemeldet und befindet sich auf der “Schlagwort umbenennen”-Seite
Nachbedingung Erfolg	Schlagwort ist umbenannt
Nachbedingung Fehlschlag	Schlagwort unverändert
Akteure	Admin
Auslösendes Ereignis	Aktion des Benutzers
Beschreibung	1. Nutzer gibt einen neuen Schlagwort-Namen ein 2. Nutzer bestätigt seine Eingabe, System liefert Rückmeldung
Erweiterung	---
Alternativen	A. (zu 2): Falls schon ein Schlagwort mit dem angegebenen Namen vorhanden, zeigt System Fehlermeldung, weiter mit 1. B. (zu 1-2): Abbruch

1.2.6. Use-Case „Schlagwort löschen“

Use Case	Schlagwort löschen
Vorbedingung	Benutzer ist als Admin angemeldet und befindet sich auf der “Schlagworte verwalten”-Seite
Nachbedingung Erfolg	Schlagwort ist gelöscht
Nachbedingung Fehlschlag	Schlagwort besteht weiterhin
Akteure	Admin
Auslösendes Ereignis	Aktion des Benutzers
Beschreibung	1. Nutzer wählt Schlagwort 2. Nutzer bestätigt Auswahl, System löscht Kategorie, liefert Rückmeldung
Erweiterung	---
Alternativen	A. (zu 1-2): Abbruch

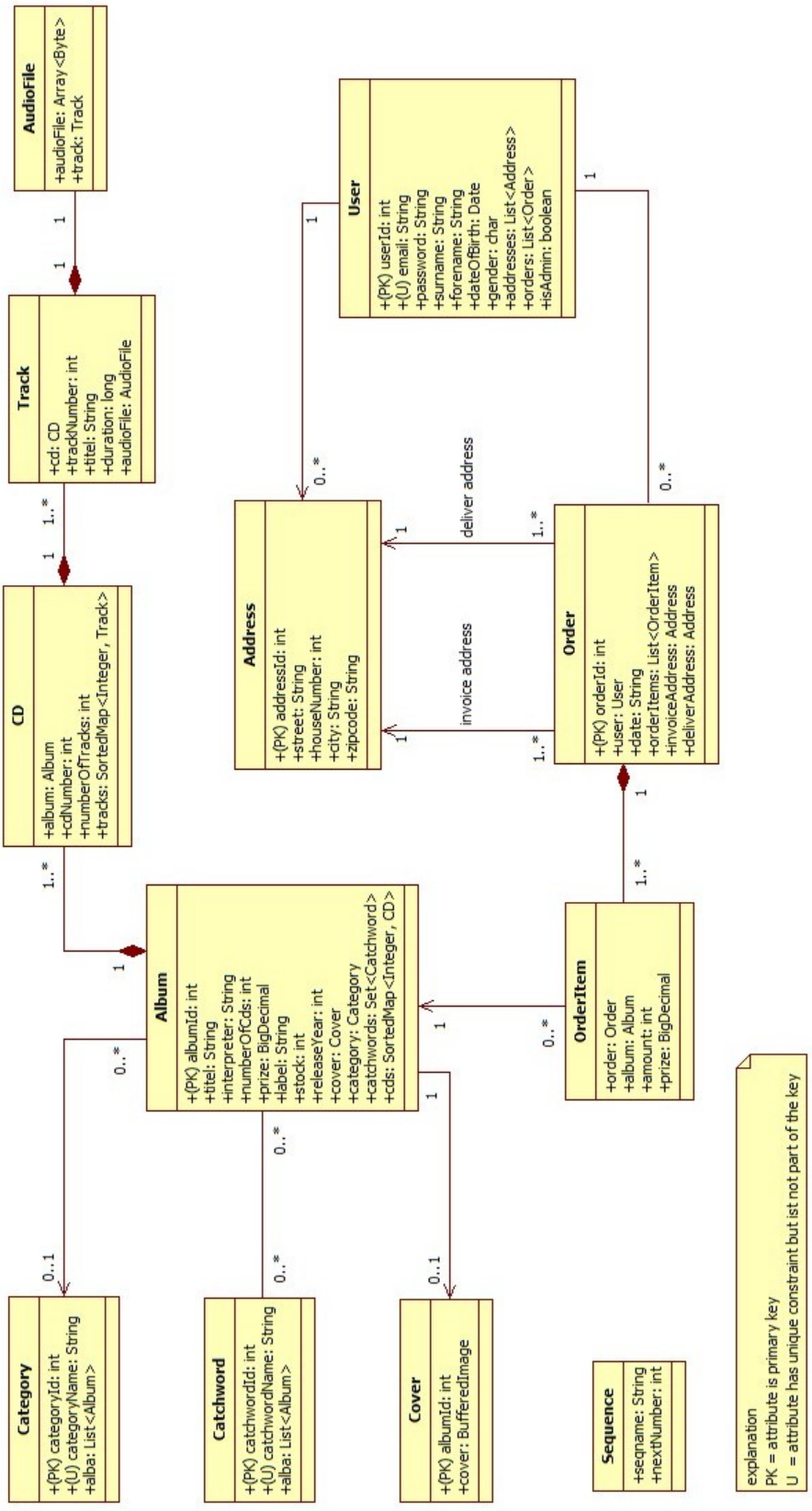
1.2.7. Use-Case „Album hinzufügen“

Use Case	Album hinzufügen
Vorbedingung	Benutzer als Admin angemeldet und befindet sich auf der “Album hinzufügen”-Seite
Nachbedingung Erfolg	Album mit allen Daten hinzugefügt
Nachbedingung Fehlschlag	Album nicht hinzugefügt, Daten verworfen
Akteure	Admin
Auslösendes Ereignis	Aktion des Benutzers
Beschreibung	1. Admin gibt Album-Stammdaten an, wählt Kategorien, Schlagworte, Cover und Tracks aus 2. Admin bestätigt Hinzufügen, System speichert Daten und komplettiert Sie, wenn möglich
Erweiterung	---
Alternativen	A. (zu 2.): Falls nach automatischer Komplettierung Albumname oder Interpret fehlen, Fehlermeldung, weiter mit 1. B. (zu 1-2): Abbruch

1.2.8. Use-Case „Album editieren“

Use Case	Album editieren
Vorbedingung	Benutzer ist als Admin angemeldet und befindet sich auf der “Album editieren”-Seite
Nachbedingung Erfolg	Album editiert
Nachbedingung Fehlschlag	Album nicht editiert
Akteure	Admin
Auslösendes Ereignis	Aktion des Benutzers
Beschreibung	1. Admin editiert Album-Stammdaten, Kategorien, Schlagworte oder Cover 2. Admin bestätigt Änderungen, System liefert Rückmeldung
Erweiterung	---
Alternativen	A. (zu 2): Falls Albumname oder Interpret fehlen, Fehlermeldung, weiter mit 1. B. (zu 1-2): Abbruch

2. Datenbank-Design

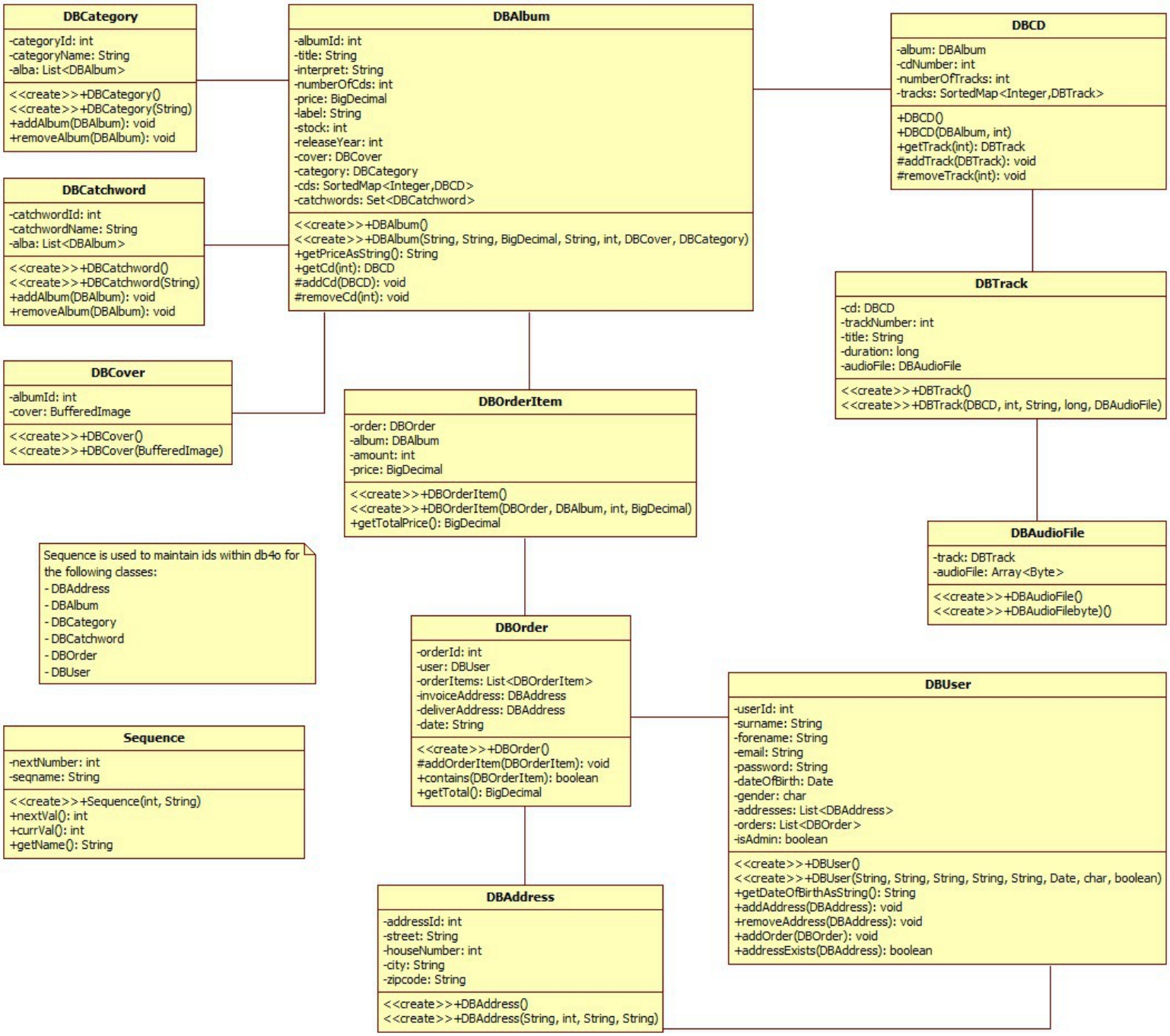


3. Klassen-Design

Das Design ist grob in DTO-, DAO-, DAO-Exception-, Servlet-, Adapter-, Util-, db4o-Connection-Pooling- und db4o-Server-Klassen eingeteilt.

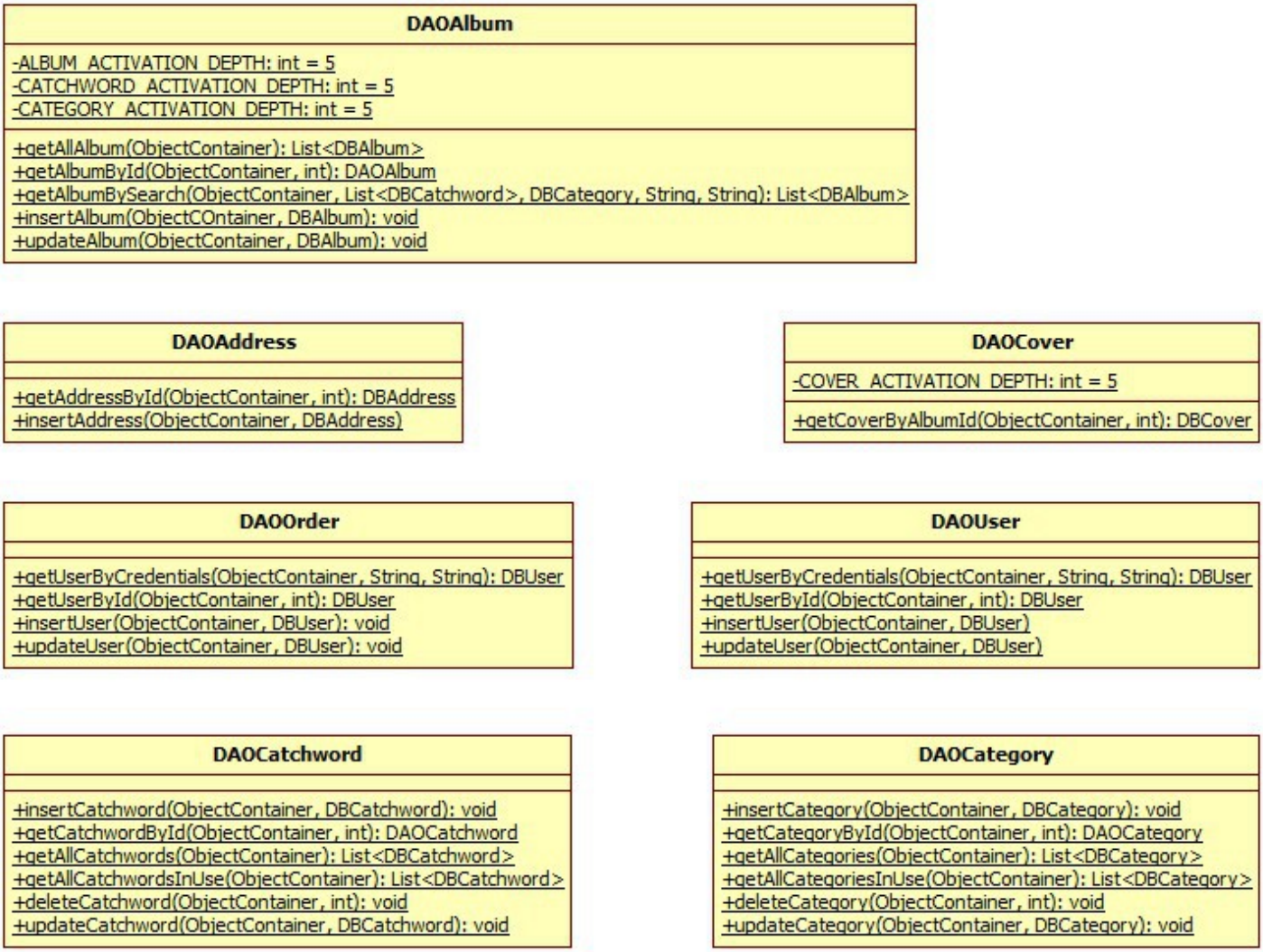
3.1. DTO-Klassen

Auf die Nennung reiner Getter- und Setter-Methoden wurde im folgenden Diagramm verzichtet. Die Art der Assoziationen sind genau wie die Multiplizitäten dem Datenbank-Design zu entnehmen.



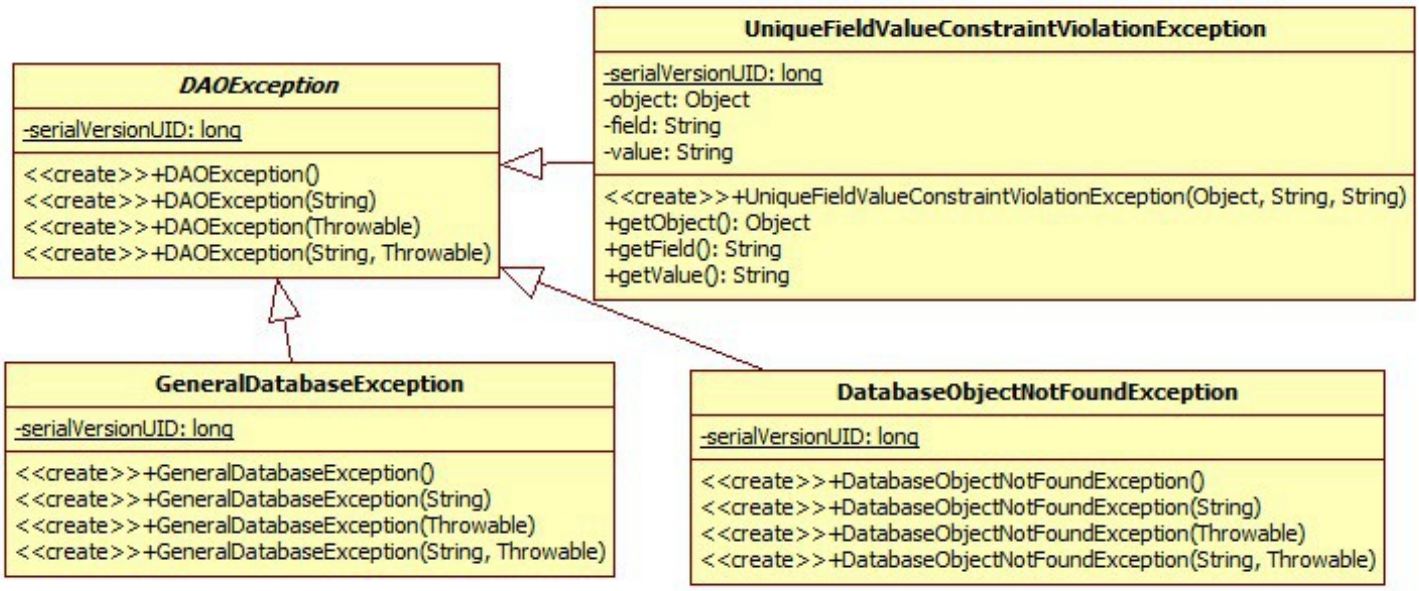
3.2. DAO-Klassen

Die DAO-Klassen kapseln den Zugriff auf die Datenquelle.



3.3. DAO-Exception-Klassen

Beim Zugriff auf die Datenquelle können verschiedenste Probleme auftreten. Diese werden in Form der folgenden Exceptions gemeldet.

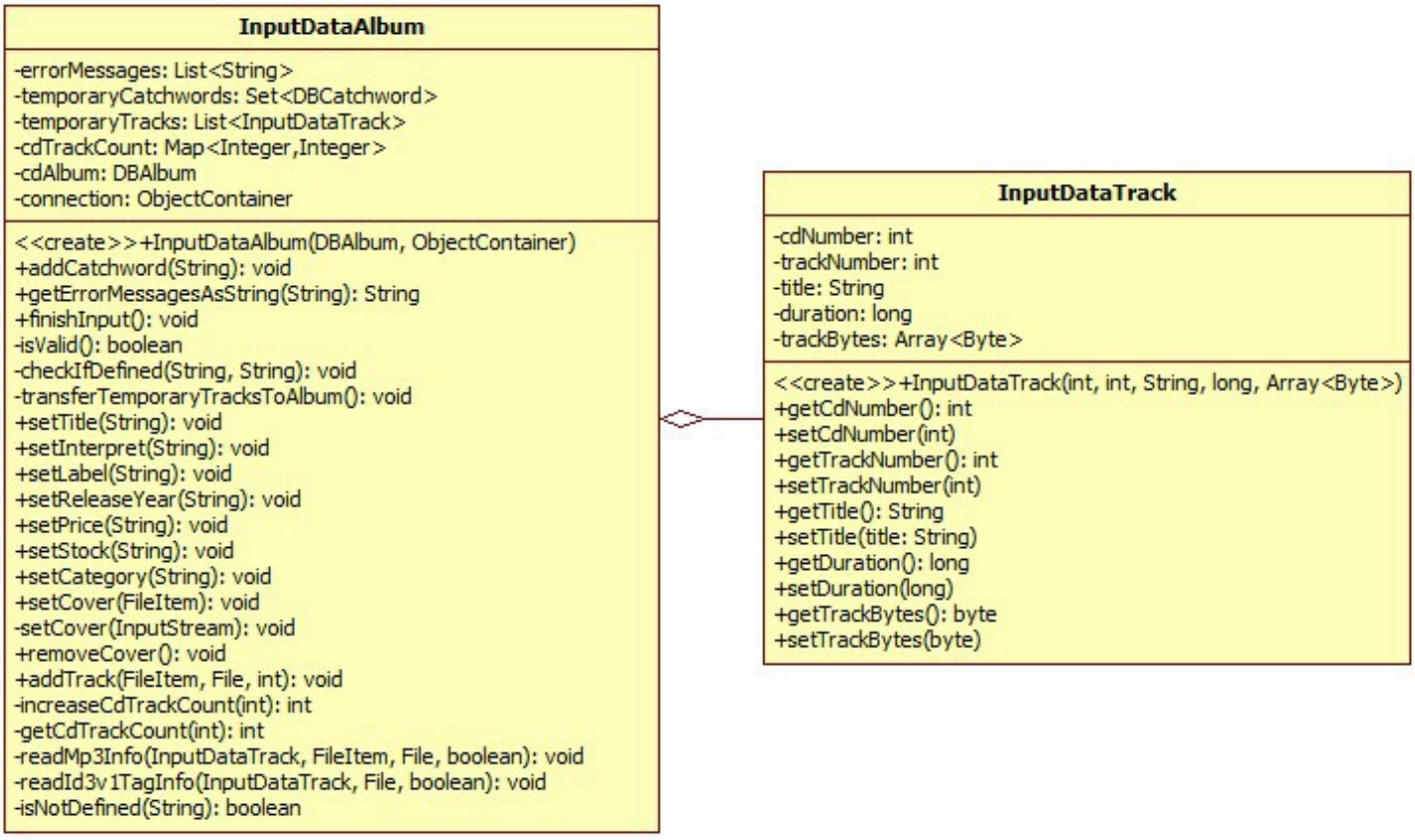


3.4. Servlet-Klassen

Alle Servlet-Klassen erben vom abstrakten GeneralServlet. Über die abstrakten doGet- und doPost-Methoden mit zusätzlichem ObjectContainer-Parameter wird den abgeleiteten Servlets die Datenbank-Verbindung zur Verfügung gestellt.

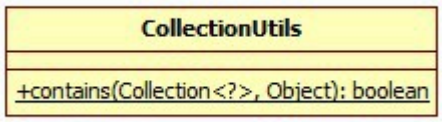
3.5. Adapter-Klassen

Für das Hinzufügen/Editieren von Alben bestehen Adapter-Klassen. Sie sorgen für die detaillierte Eingabevalidierung und kümmern sich um die Umwandlung der erhaltenen Daten in die in den DTO-Klassen verwendeten Formate.



3.6. Util-Klassen

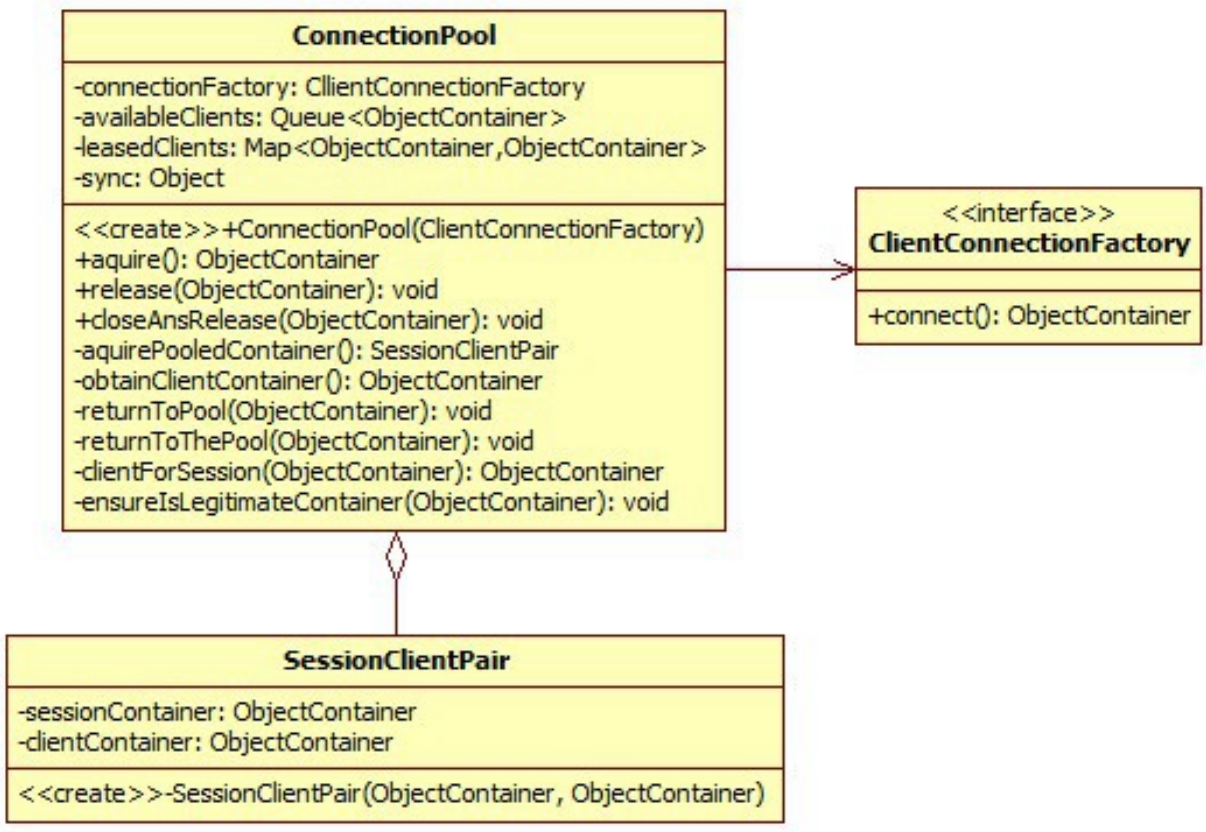
Zum Vergleichen der Inhalte zweier Listen wurde eine CollectionUtils-Klasse angelegt. Die contains-Methode wird über eine JSTL tag library (Web-INF/custom-functions.tld) in der editalbum.jsp zum Vorselektieren bestehender Schlagworte verwendet.



Desweiteren existiert eine Whirlpool-Klasse, die den gleichnamigen Hash-Algorithmus umsetzt. Sie wird für die Passwort-Verschlüsselung verwendet. Die Klasse wurde unverändert aus einem vorherigen Projekt übernommen und wird deswegen hier nicht weiter betrachtet.

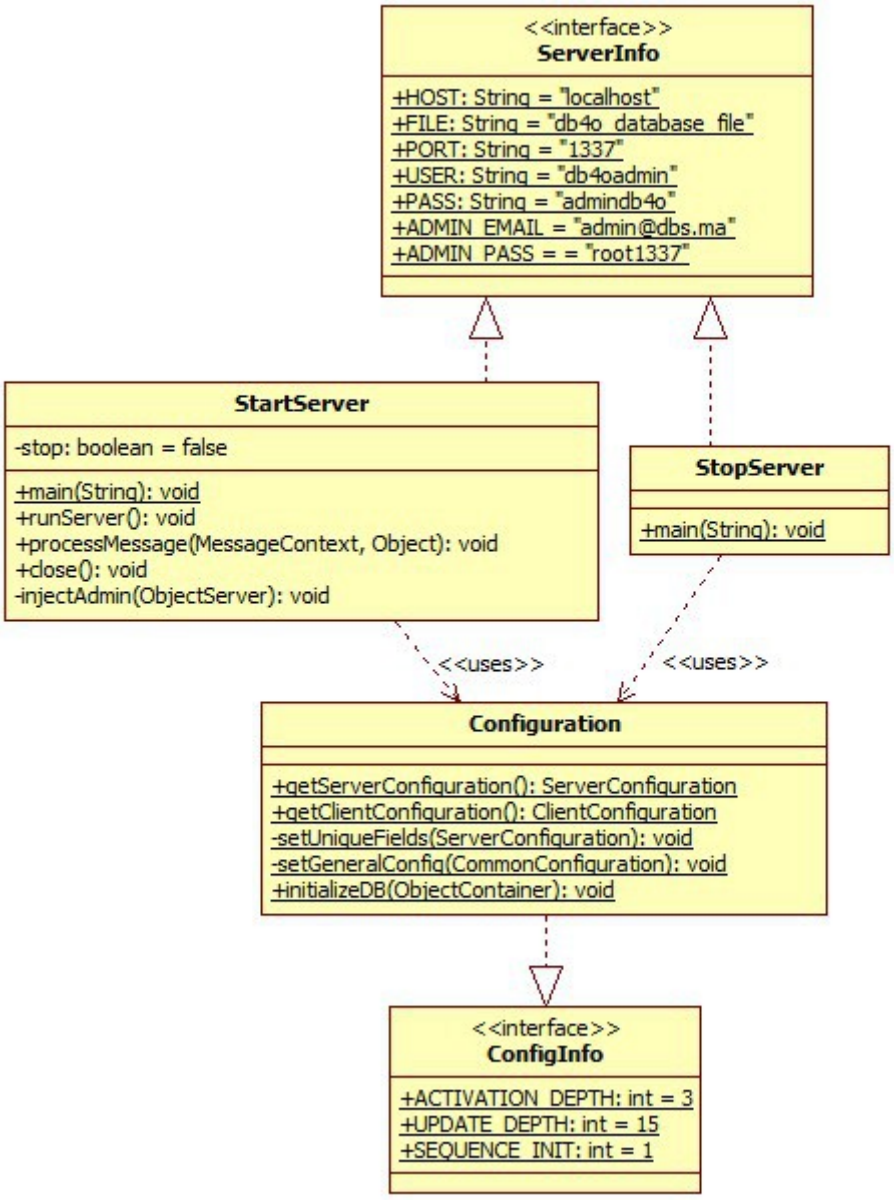
3.7. Db4o-Connection-Pooling-Klassen

Die Verbindung zur db4o-Datenbank wird via Connection Pooling durchgeführt. Die Klassen wurden unverändert aus der db4o-Referenz-Dokumentation entnommen: http://community.versant.com/documentation/reference/db4o-8.1/java/reference/Content/CodeExamples/clientserver/pooling/Example-clientserver-pooling-java.zip.



3.8. Db4o-Server-Klassen

Unabhängig von der eigentlichen Webapplikation bestehen zwei Anwendungen, die zum Starten (StartServer) bzw. Stoppen (StopServer) eines eigenständigen db4o-Servers verwendet werden können.



4. Testen

Da der Fokus des Projektes auf der Vollständigkeit der Funktionalitäten und der Funktionsfähigkeit der Anwendung lag wurde auf einen ausführlichen Testbericht verzichtet. Die DAO-Klassen sowie die DTO-Klassen wurden mittels JUnit-Tests getestet. Die restlichen Klassen (Servlets, Server, Adapter, etc.) und die *.jsps wurden während der Implementierung ständig auf ihre Funktionalität und Funktionsfähigkeit hin getestet.

5. Zugangsdaten

Von der Server-Anwendung wird ein Admin-Account für die Web-Anwendung angelegt. Die Zugangsdaten sind:

E-Mail: admin@dbs.ma
Passwort: root1337

6. Aufwandsmatrix

	Andreas Baur	Markus Henn	Jochen Pätzold	Team
Analyse und Design				
Use-Cases	-	7,00	-	**7,00**
Klassendesign/ Datenbankdesign	7,50	1,50	7,50	**16,50**
Pflichtenheft	0,25	4,00	-	**4,25**
Gesamt	**7,75**	**12,50**	**7,50**	**27,75**
Implementierung				
db4o (server, config, pool)	4,50	-	2,00	**6,50**
DAO-Klassen	31,50	2,50	15,00	**49,00**
DTO-Klassen	10,25	1,00	8,50	**19,75**
Adapter-Klassen (Eingabevalidierung, MP3-Tags, ...)	-	10,75	-	**10,75**
Servlets	15,00	31,00	5,00	**51,00**
JSTL/HTML/JavaScript	4,00	24,00	1,00	**29,00**
Gesamtt	**65,25**	**69,25**	**31,50**	**166,00**
Test und Wartung				
JUnit-Tests	6,00	-	8,50	**14,50**
Servereinrichtung	2,00	-	2,00	**4,00**
Gesamt	**8,00**	**0,00**	**10,50**	**18,50**
Projektmanagement				
Meeting	10,00	10,00	10,00	**30,00**
Präsentation	13,50	12,00	13,50	**39,00**
Dokumentation	1,50	2,25	3,00	**6,75**
Gesamt	**25,00**	**24,25**	**26,50**	**75,75**
Gesamt	**106,00**	**106,00**	**76,00**	**288,00**
Plan nach ECTS (8/3 * 30 Std.)	78,00	78,00	78,00	**234,00**

7. Probleme

7.1. DB4O

7.1.1. Maximale Datenbank-Größe

Standardmäßig können db4o-Datenbanken eine maximale Größe von 2 GB annehmen. Per Konfiguration kann man diese Größe auf maximal 254 GB hochschrauben. Wird die maximale Größe erreicht, kann (theoretisch) noch lesend auf die Datenbank zugegriffen werden, aber nicht mehr schreibend.

Problem

Nach Erreichen der maximalen Größe wurde schon beim Aufruf der Methode grantAccess(String username, String password) des ObjektServer-Objekts die Exception “DatabaseMaximumSizeReachedException” geworfen. In der API-Dokumentation von db4o ist dieser mögliche Exception-Wurf nicht dokumentiert. Ohne Zugriffsrechte kann allerdings auch nicht lesend auf die Datenbank zugegriffen werden. Die Datenbank konnte nicht mehr verwendet werden.

Problemlösung

Eine echte Lösung des Problems wurde nicht gefunden. Wir haben die Test-Datenbank gelöscht und eine neue angefangen. Als Workaround kann die maximale Größe der Datenbank mit folgendem Befehl konfiguriert werden:

```
Db4o.configure().blockSize(newBlockSize);
```

Die neue Block-Größe darf zwischen 1 (entspricht 2 GB) und 127 (entspricht 254 GB) gewählt werden.
Da der verfügbare Speicherplatz auf unserem Zielsystem allerdings weniger als 2 GB beträgt und die Datenbank diese Marke somit nicht erreichen kann, haben wir keine Änderungen vorgenommen.

7.1.2. update depth, activation depth

Um das Laden von Objekten zu optimieren wird mit Hilfe der Activation Depth (im Folgenden AD genannt) eine Tiefe angegeben, mit der die Objekte aus der Datenbank geladen werden. Analog gilt dies für die Update Depth (im Folgenden UD genannt). Diese Angaben werden in der Configuration für den ObjectContainer angegeben.
Die db4o behandelt jedes Objekt als eigene Ebene. D.h.: Verwendet man in einem Objekt “Album” eine Liste mit Referenzen auf “CD”, so kann man mit einer AD von 1 nicht auf die CDs zugreifen. Auch eine AD von 2 ist nicht ausreichend, da “Album” schon als Ebene 1 gezählt wird und die Liste ebenfalls eine Ebene ausmacht. Erst mit einer AD von 3 kann man auf die CDs zugreifen.

Probleme

1. Verwendet man Album-Objekt eine Referenz auf ein Cover, kann man nicht mit einer AD von 2 auf das Cover zugreifen.
2. Aufgrund des unterschiedlichen Verwendung der Albumobjekte sind verschiedene ADs vonnöten, die jedoch nicht in der Configuration angegeben werden können.

Problemlösung

1. Das Cover-Objekt enthält selbst wiederum eine Objektreferenz auf ein BufferedImage und das BufferedImage enthält ebenfalls Referenzen auf verschiedene Objekte (zB. Colormodel). Erst mit einer AD von 7 konnten die Cover richtig gelesen und angezeigt werden.
2. Die AD der Configuration des ObjectContainers wurde auf 3 gesetzt. Beim Zugriff auf dieses Cover kann allerdings nur auf die Referenz des Objektes zugegriffen werden. Daher wird das Objekt beim Laden zusätzlich mit der Stufe 5 aktiviert. Des Weiteren konnte mit der AD von 3 beim Anzeigen von Alben keine Trackinformationen gelesen werden. Um dies zu ermöglichen wird beim Album anzeigen das Objekt mit Stufe 5 aktiviert.

Offene Fragen

Obwohl die Cover durch die Aktivierung korrekt angezeigt werden, treten des Öfteren Exceptions auf, die jedoch weder für den Benutzer sichtbar werden, noch die Anwendung in irgendeiner Art und Weise beeinträchtigen. Wird das Coverobjekt mit einer höheren Stufe aktiviert ändert sich das Verhalten erst bei einer Stufe von 11. Dieses Verhalten kann nicht detailliert nachvollzogen werden. Erst eine ausgiebige Analyse der verwendeten Objekte kann hier einen Aufschluss liefern.

7.1.3. UniqueFieldValueConstraints

Problem

Werden die UniqueFieldValueConstraints sowohl in der Server Configuration als auch in der Client Configuration angegeben führt dies zu einer Exception:
"Db4objects.Db4o.Constraints.UniqueFieldValueConstraint should be configured on the server."

Problemlösung

Keine UniqueFieldValueConstraints in der Client Configuration angeben.
(siehe: http://tracker.db4o.com/browse/DOC-661)

7.1.4. Objektidentifizierung

Das Aktualisieren eines Datenbank-Objekts wird mit demselben Befehl angestoßen wie das Einfügen eines neuen Objektes (store()).

Problem

Wir wollten ein aus der Datenbank stammendes Objekt, das zwischenzeitlich in der Session gespeichert wurde, aktualisieren und wieder in der Datenbank ablegen. Db4o hat das Objekt allerdings nicht wiedererkannt und wollte ein neues Objekt mit den Werten anlegen. Durch

die zwischenzeitliche Speicherung des Objekts in der Session scheint die von d4o zur Identifizierung verwendete Eigenschaft zerstört worden zu sein.

Problemlösung

Das in der Session gespeicherte Objekt wird nicht direkt editiert, sondern erst neu aus der Datenbank ausgelesen. Damit ist das Abspeichern dieses Objektes wieder ohne weiteres möglich.

7.2. Zeichenkodierung

Für alle JSP- bzw. HTML-Dokumente haben wir UTF-8 als Character Encoding festgelegt.

7.2.1. Apache FileUpload

Wir haben uns entschieden den Upload der Tracks und Cover über dasselbe Formular zu ermöglichen wie die Eingabe der Album-Stammdaten. Da die Parameter eines mit enctype="multipart/form-data" übergebenen Formulars allerdings nicht wie sonst im HttpServletRequest-Objekt über die Methode getParameter(String parameterName) abgreifbar sind, verwenden wir für das Auslesen der (textuellen) Stammdaten auch die FileUpload-Bibliothek von Apache.

Problem

Die eigentlich bereits in UTF-8 übergebenen Parameter werden von der Bibliothek ohne weitere Konfiguration erneut in UTF-8 kodiert (als wenn die Parameter in ANSI-Kodierung vorliegen würden).

Beispiel

aus dem Wert “Test” wird “TÃ¤st”.

Problemlösung

Die Zeichenkodierung muss an 3 Stellen gesetzt werden, um eine korrekte Kodierung der ausgelesenen Daten zu gewährleisten.

```
String charEncoding = “UTF-8”;
request.setCharacterEncoding(charEncoding);
FileItemFactory factory = new DiskFileItemFactory();
ServletFileUpload upload = new ServletFileUpload(factory);
upload.setHeaderEncoding(charEncoding);
List<?> parameters = upload.parseRequest(request);
for (Iterator<?> iter = parameters.iterator(); iter.hasNext();) {
        FileItem element = (FileItem) iter.next();
        String fieldName = element.getFieldName();
        // …
        String fieldValue = element.getString(charEncoding);
        // …
}
```

7.2.2. Apache Tomcat

Auch im Apache Tomcat selbst muss bei Verwendung von UTF-8 als Zeichenkodierung nachkonfiguriert werden.

Problem

Werden UTF-8 kodierte Formulardaten per HTTP-Methode “get”, also in der URL übertragen, so werden die übermittelten Parameter bei Verwendung der Request-Methode getParameter(String parameterName) (ohne vorherige Anpassung der server.xml) erneut in UTF-8 kodiert. Es besteht in diesem Fall also dasselbe Problem wie bei Verwendung der Apache FileUpload-Bibliothek (siehe oben).

Problemlösung

Editieren der server.xml.
Finden der Connector-Definition, z.B.:

```
<Connector connectionTimeout="20000" port="8080" protocol="HTTP/1.1"
redirectPort="8443" />
```

Hinzufügen des Attributs URIEncoding mit dem Wert “UTF-8”:

```
<Connector connectionTimeout="20000" port="8080" protocol="HTTP/1.1"
redirectPort="8443" URIEncoding="UTF-8" />
```

7.3. Optimierungsgedanken

7.3.1. DB4o

Öfter benötigte Datenbank-Objekt-Collections könnten in der Session zwischengespeichert werden. Überprüfung auf Änderungen in den Collections kann man z.B. mittels Hashwert realisieren. Hierfür müsste dann allerdings eine eigene Klasse, die die Hashwerte verwaltet, geschrieben und gelesen werden. Bei sehr großen Collections ist dies durchaus sinnvoll, da der Hashwert nur bei einer Änderung in der Collection erzeugt werden muss und Änderungen selten sind (siehe Category, Catchword, …) [zumindest im laufenden Betrieb]. Dies reduziert die Dauer eines Datenbankzugriffes z.B. beim Suchen nach Alben, wenn die Categories und Catchwords aus der DB geladen werden. In diesem Fall kann einfach auf die Session zugegriffen werden.