1 Einleitung

Autor: Jonas Picker

Hier wird der Entwurf des Bibliotheksmanagementsystems BiBi beschrieben. Das Dokument bezieht sich auf das Lastenheft von Christian Bachmaier und Armin Größlinger und stellt eine technische Vertiefung unseres Pflichtenhefts dar, welches im Folgenden wiederholt referenziert wird.

2 Systemarchitektur

Autor: Ivan Charviakou

2.1 Design-Patterns: Schichtenunabhängigkeit

MVC: Als grundlegendes Design-Pattern sorgt MVC dafür, dass die Model-, Präsentations-, und Steuerungskomponente voneinander weitgehend entkoppelt sind. Dabei implementiert das JSF-Framework per Default Großteile der Präsentations- und Steuerungskomponenten, aber überlässt die Modelkomponente komplett dem Entwickler. Für die Präsentationskomponente werden nämlich vordefinierte UI-Komponente vorgesehen, während der eingebaute Faces-Servlet die Steuerungsaufgabe übernimmt. Durch selbst-definierte Tag-Libraries und Facelets lassen sich diese beiden Komponenten auch ggf. vervollständigen.

CDI: Die im JSF-Framework eingebaute CDI-Funktionalität ermöglicht eine direkte Kommunikation zwischen abhängigen Komponenten, die durch JSF verwaltet werden. Solche Beziehungen werden dann mit entsprechenden Annotationen, wie ‚@inject', gekennzeichnet. In der gegebenen Anwendung trägt CDI dadurch wesentlich zur Intraschichtenkommunikation bei.

DTOs: Ein Datentransferobjekt (DTO) enthält die Daten zu einer bestimmten Entität oder zu einem zusammenhängenden Ausschnitt aus verschiedenen Entitäten. In der gegebenen Anwendung entspricht das einem Medium, Exemplar, Nutzer, usw. Durch Übergabe von solchen Objekten an verschiedenen Modelschichten, können sie die Daten unabhängig voneinander anpassen oder evaluieren. Als Beispiel prüft im Model die Logikschicht das Format eines E-Mails, während die Sicherheitsschicht SQL-Injection-Attacken in der Eingabe ausschließt.

DAOs und ihre Fassaden: Zu jedem DTO existiert auch ein Datenzugriffsobjekt (DAO), das bei einer Übergabe eines DTOs die enthaltenen Daten speichert bzw. lädt. Damit mehrere Modelschichten diese Prozeduren eigens erweitern können, verwenden die DTOs in der gegebenen Anwendung den Fassade-Muster. Jede betroffene Schicht verwendet zur Erweiterung eines DAOs nämlich die gleiche Schnittstelle und ruft die Schnittstelle der nächsten Schicht auf. Als Beispiel gibt die Logikschicht ein Passwort im Klartext an und übergibt es an eine DAO-Instanz in der Sicherheitsschicht mittels eines DTOs. Diese Schicht berechnet dann aus dem Klartext einen Hash-Wert und leitet diesen Wert an die DAO-Instanz in der Datenzugriffsschicht weiter.

Event-Listener: Dadurch, dass JSF verschiedene Arten von Listenern bereitstellt, ist eine stärkere Entkopplung von Schichten möglich. Während Action-, Value-Change-, und Phasenlistener hauptsächlich eine Logikschicht ansprechen, kann beispielweise der eingebaute System-Event-Listener für die unabhängige Initialisierung der Modelschichten sorgen.

2.2 Design-Patterns: Fehlerbehandlung

Allgemeiner Testmodus: Es wird ein Testmodus eingeführt, der beim Applikationsstart als Parameter angegeben wird.

Exceptions: Jede Schicht in der gegebenen Anwendung definiert eigene Checked-Exceptions, die ggf. durch die Schichten hochgeworfen werden. Dabei wird eine solche Exception in jeder Zwischenschicht gefangen in eine eigens definierte Exception umgewandelt bis eine Schicht sie behandelt. Im Gegensatz werden Unchecked-Exceptions nicht behandelt und führen zum Absturz der Anwendung. Während für eine Präsentations- und Logikschicht oft eine GUI-Anzeige als geeignete Behandlung gilt, ist es für jede Schichten anders. Falls es beispielsweise durch eine Checked-Exception in der Datenzugriffsschicht festgestellt wird, dass die verwendete Datenbank nicht mehr erreichbar ist, wird stattdessen dynamisch auf lokale Applikationsdaten zugegriffen. Unter den DAOs in der Datenzugriffsschicht wird somit der Strategy-Muster angewendet.

Logging: Als Logging-Framework wird der im JDK vordefinierte Framework verwendet. Diese Funktionalität ist nur im Testmodus aktiv und es wird zu einer lokalen Log-Datei geschrieben.

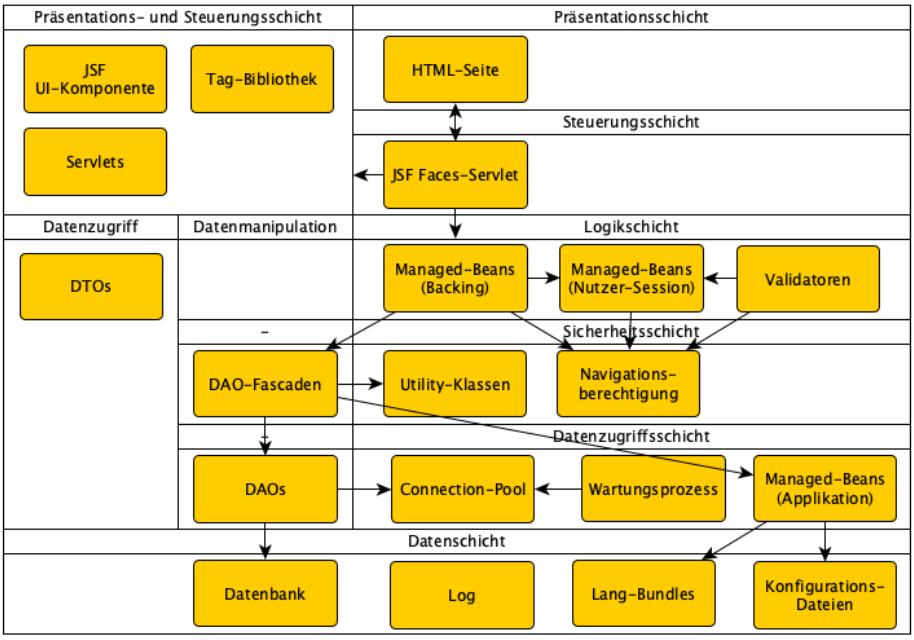
Validatoren: Das JSF-Framework gibt dem Entwickler die Möglichkeit, Daten auf Korrektheit zu prüfen, bevor sie dem Logikschicht bereitgestellt wird. Da aber komplexere semantische Fehler oft einen größeren Kontext und eine komplexere Behandlung benötigen, wird der Einsatz von diesen Validatoren hauptsächlich auf syntaktische Formattierfehler beschränkt.

2.3 Design-Patterns: Konkrete Features

Connection-Pool: Unter einem Pool bezeichnet man eine Sammlung an Objekten, die angefragt, zurückgegeben, und wiederverwendet werden. Dies vermeidet insbesondere die Erzeugung von solchen Objekten, was in Bezug auf Datenbankverbindungen als aufwendig gilt. Da es trivialerweise nur einen solchen Container geben sollte, wird ein Pool auch als Singleton verwendet. Für die gegebene Anwendung wird die HikariCP Implementierung einer solchen Connection-Pool genommen. Beim Testmodus wird die Anzahl an Verbindungen auf eins gesetzt.

Wartungsthread: In der gegebenen Anwendung kann es vorkommen, dass persistierte Daten oder Daten im Cache mit der Zeit ungültig werden. Du solchen Daten gehört beispielsweise die Gültigkeit von einem Passwortrücksetzungslink. Mit einem Wartungsprozess werden solche Inkonsistenzen erkannt und behoben.

2.4 Modeldiagramm



3 Klassendiagramm

Autor: Mohamad Najjar

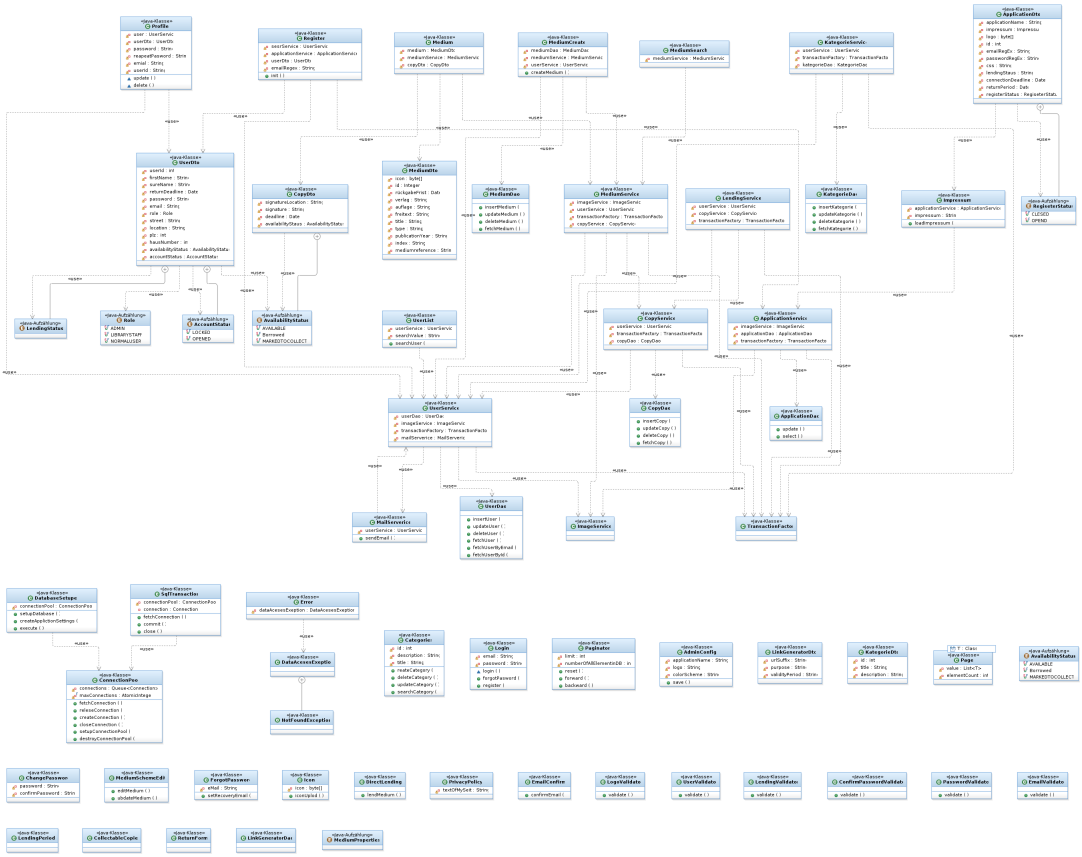


Abbildung 1: Klassendiagramm

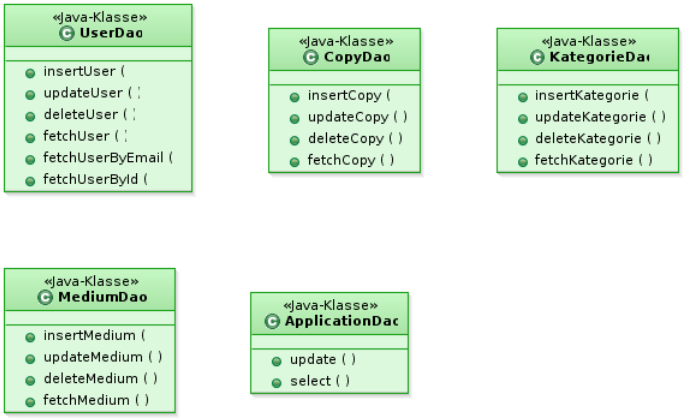
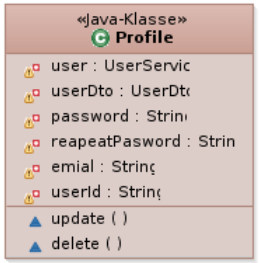
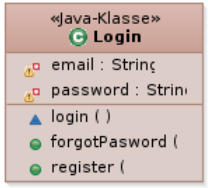
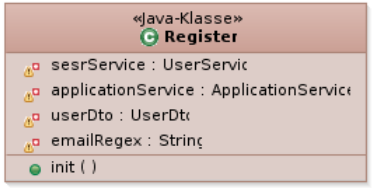


Abbildung 2: Klassendiagramm der Daos


«Java-Klasse»
Profile
user : UserServic
userDto : UserDt
password : Strin
reapeatPasword : Strin
emial : String
userId : String
update ( )
delete ( )


«Java-Klasse»
Login
email : String
password : Strin
login ( )
forgotPasword (
register (

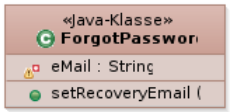

«Java-Klasse»
Register
sesrService : UserServic
applicationService : ApplicationServic
userDto : UserDt
emailRegex : String
init ( )


«Java-Klasse»
AdminConfig
applicationName : String
logo : String
colorScheme : Strin
save ( )


«Java-Klasse»
Medium
medium : MediumDt
mediumService : MediumServic
copyDto : CopyDto


«Java-Klasse»
UserList
userService : UserServic
searchValue : Strin
searchUser (


«Java-Klasse»
ChangePasswor
password : Strin
confirmPassword : Strin


«Java-Klasse»
ForgotPasswor
eMail : String
setRecoveryEmail (

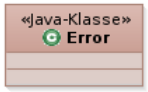

«Java-Klasse»
Impressum
applicationService : ApplicationServic
impressum : Strin
loadImpressum (


«Java-Klasse»
Categorie


«Java-Klasse»
CollectableC


«Java-Klasse»
DirectLending


«Java-Klasse»
EmailConfirm


«Java-Klasse»
Error


«Java-Klasse»
LendingPeriod


«Java-Klasse»
MediumSearch


«Java-Klasse»
ReturnForm


«Java-Klasse»
MediumSchemeEdit


«Java-Klasse»
PrivacyPolicy

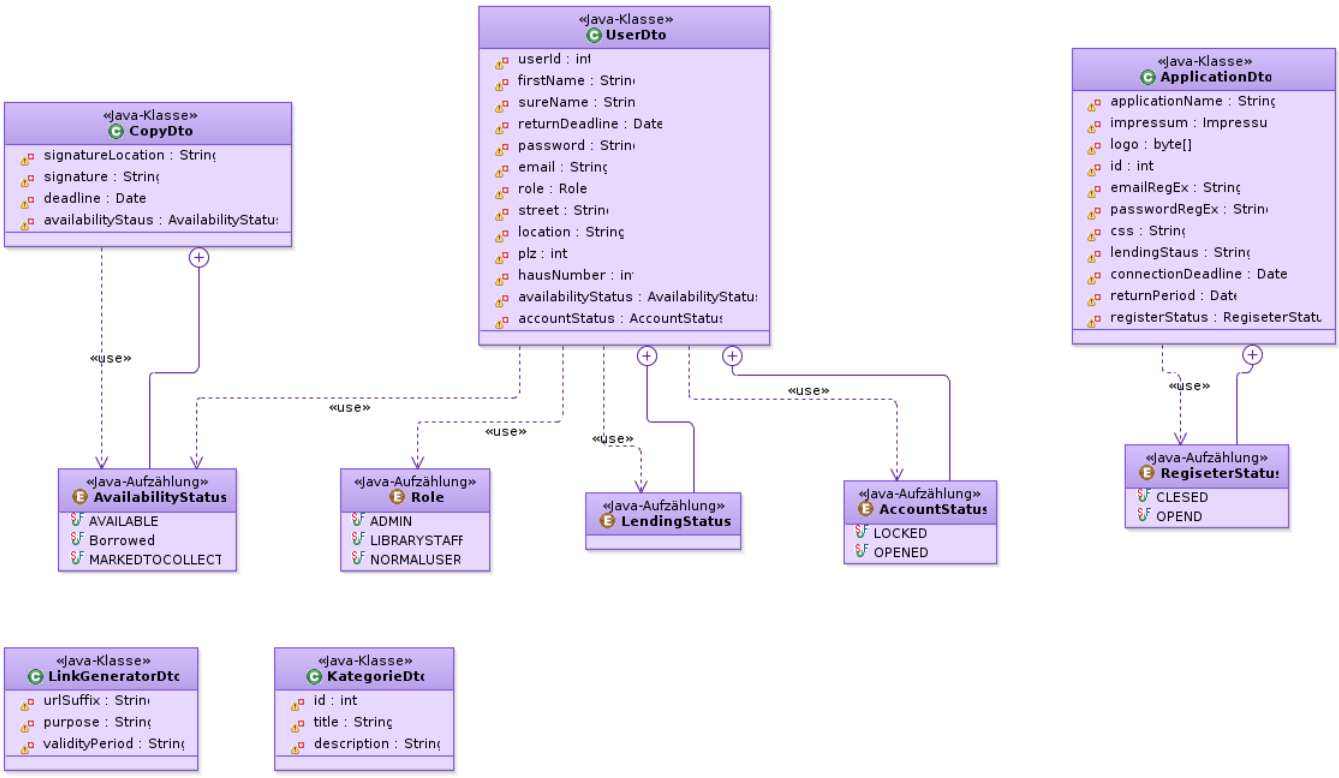


Abbildung 4: Klassendiagramm der Dtos

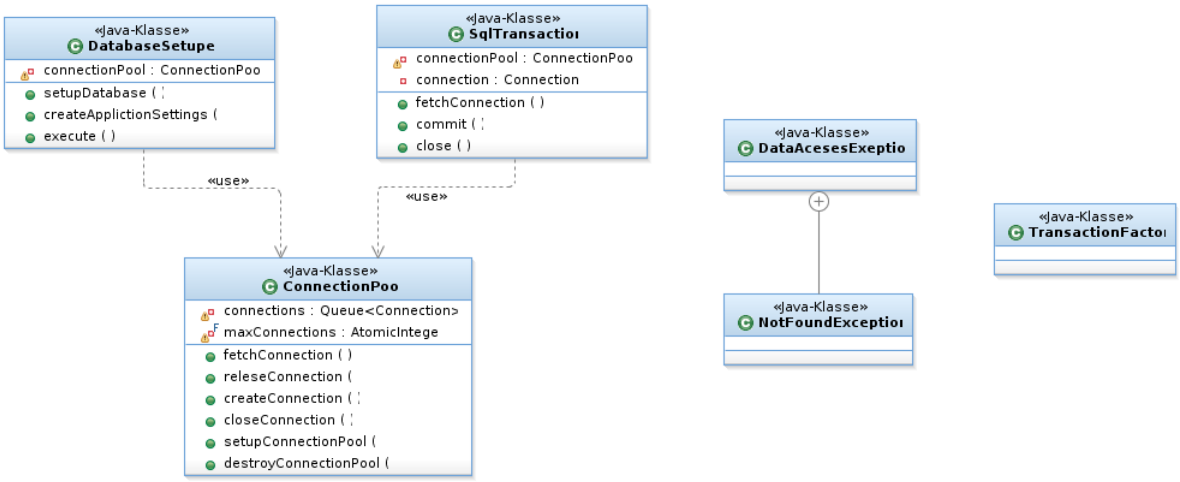


Abbildung 5: Klassendiagramm des Systems

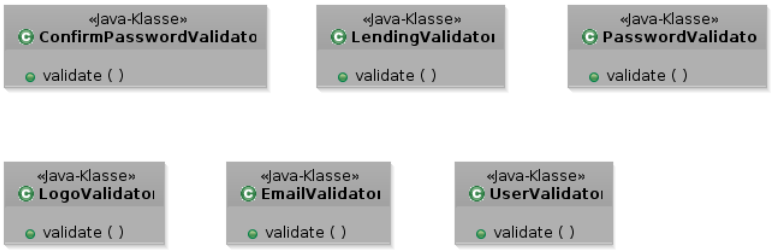


Abbildung 6: Klassendiagramm des Validator

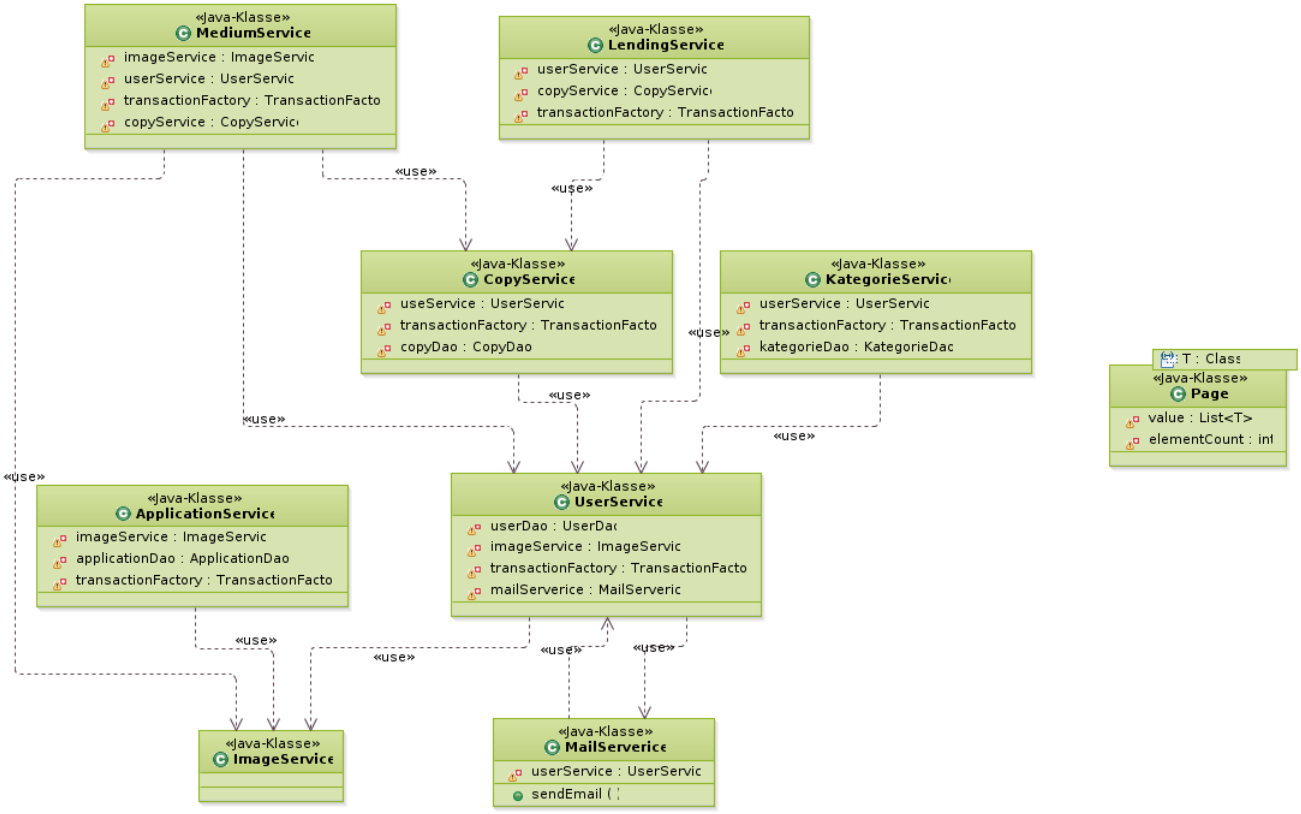


Abbildung 7: Klassendiagramm der Services

Java-Klasse	Beschreibung
Medium	Backing Bean für das Erstellen und Bearbeiten eines Medium.
AdminConfig	Backing Bean für die Einstellung der Webseite.
ChangePassword	Backing Bean für die Änderung des Passwortes.
Error	Backing Bean der Seite die Fehlermeldungen anzeigt.
Profile	Backing Bean für das Anzeigen und Bearbeiten des Profils eines Benutzers.
ForgotPassword	Backing Bean für die Seite, auf der man sich einen neues Passwort zusenden lassen kann, wenn man das altes Passwort vergessen hat.
Login	Backing Bean des Logins.
Register	Backing Bean für das Anlegen eines neuen Benutzers.
Icon	Backing Bean für das Hochladen und Bearbeiten eines Mediums und Logo des Systems.
Category	Backing Bean für die Kategorie, die als Liste angezeigt wird.
UserList	Backing Bean für die Seite der Benutzersuche.
CollectableCopies	Backing Bean der Seite, auf der alle Exemplare abzuholend makiert.
DirectLendin	Backing Bean für die Seite der Directausleihe.
EmailConfirm	Backing Bean für die Seite der Emailbestätigung.
LendingPeriod	Backing Bean für die Seite der Ausleihefrist.
MediumSearch	Backing Bean für die Seite der Mediumsuche.
ReturnForm	Backing Bean für die Seite der Rückgabe.
MediumEdit	Backing Bean für die Seite des Mediumedieren.
PrivacyPolicy	Backing Bean für die Seite der Datenschutzerklärung.

Java-Klasse	Beschreibung
UserDto	Enthält alle Daten des Profils eines Benutzers.
ApplicationDto	Enthält die Daten der Anwendung.
MediumDto	Enthält die Daten eines Mediums.
CopyDto	Enthält die Daten eines Exemplar.
CategoryDto	Enthält die Daten einer Kategorie.
LinkGeneratorDto	Enthält die Daten des Linkgenerators.

Java-Klasse	Beschreibung
MediumDao	Kontrolliert den Zugriff auf Mediumdaten in der Datenbank.
ApplicationDao	Kontrolliert den Zugriff auf die Einstellungen der Anwendung, die in der Datenbank gespeichert sind.
UserDao	Kontrolliert den Zugriff auf Benutzerdaten in der Datenbank.
CategoryDao	Kontrolliert den Zugriff auf Kategoriedaten in der Datenbank.
CopyDao	Kontrolliert den Zugriff auf die Exemplardaten in der Datenbank.

Java-Klasse	Beschreibung
EmailValidator	Prüft ob es sich eine gültige E-Mail-Adresse handelt.
PasswordValidator	Prüft beim Anlegen eines neuen Benutzerkontos oder beim Ändern des Passwortes, ob das Passwort den Mindestanforderungen entspricht.
ConfirmPasswordValidator	Prüft ob die Passwortbestätigung und Passwort übereinstimmen.
LogoValidator	Validiert ob eine Datei einem gängigen Bildformat entspricht und nicht zu groß oder auch zu klein ist.
UserValidator	Prüft ob eine Ausleihe deren Frist abgelaufen ist.
LendingValidator	Prüft ob eine Ausleihe deren Frist abgelaufen ist.

Java-Klasse	Beschreibung
ConnectionPool	Verwaltet die Verbindungen zur Datenbank.
DatabaseSetup	Erstellt das Datenbankschema und die Verbindung mit Datenbank.
SqlTransaction	Wird verwendet, um eine Sql-Transaction zu verarbeiten.
DataAcesesException	Wird geworfen, wenn es keine Verbindung mit Datenbank gibt.

Java-Klasse	Beschreibung
MediumService	Service für Aktionen zu Medien.
ApplicationService	Service für Aktionen zu der Anwendung.
UserService	Service für Aktionen zu dem Benutzer des System.
CategoryService	Service für Aktionen zu Kategorie.
CopyService	Service für Aktionen zu Exemplaren.
LendingService	Service für Aktionen zu der Ausleihe.

3.1 Paketdiagramm

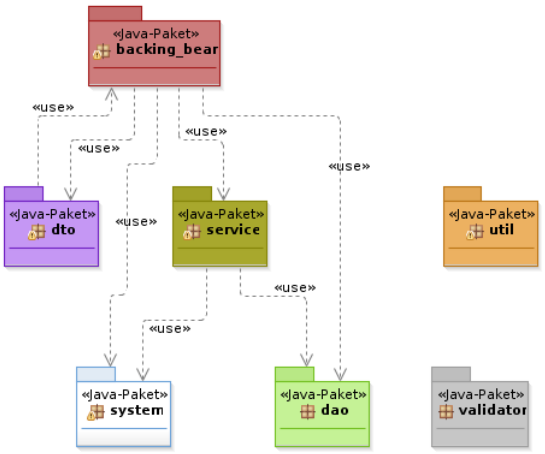


Abbildung 8: Paketdiagramm

4 JSF-Facelets

Autor: León Liehr

Nachfolgend verwenden wir die diese Abkürzungen:

abg. Nutz.	abgemeldeter Nutzer
ang. Nutz.	angemeldeter Nutzer
Bib.-Mit.	Bibliotheksmitarbeiter
Admin.	Administrator

4.1 Seitenvorlage (`template.xhtml`)

Typ	Beschreibung	Sichtbarkeit
Knopf[a]	zum Anzeigen der kontextsensitiven Hilfe	jeder
Hyperlink	zu der erweiterten Suche	jeder
Eingabefeld	für die Mediensuche	jeder
Hyperlink	zur Datenschutzerklärung	jeder
Hyperlink	zum Impressum	jeder
Hyperlink	zum Kontakt	jeder
Hyperlink	zu der Profilseite	ang. Nutz.
Hyperlink	zum Abmelden; zur Anmeldemaske	ang. Nutz.
Hyperlink	zur Anmeldemaske	abg. Nutz.
Hyperlink	zur Registrierungsseite	abg. Nutz.
Hyperlink	zu den abzuholenden Exemplaren	Bib.-Mit.
Hyperlink	zu der Medienrückgabe	Bib.-Mit.
Hyperlink	zu der Direktausleihe	Bib.-Mit.
Hyperlink	zu der Verwaltung	Admin.

[a]Dargestellt als Fragezeichenbildsymbol

4.2 Abzuholende Exemplare (`copies-ready-for-pickup.xhtml`)

Typ	Beschreibung	Sichtbarkeit
Tabelle	aller abzuholenden Exemplare	Bib.-Mit.
Knopf	nächste Seite der obigen Tabelle	Bib.-Mit.
Knopf	vorige Seite der obigen Tabelle	Bib.-Mit.

4.3 Abzuholende und ausgeliehene Exemplare (`my-copies.xhtml`)

Typ	Beschreibung	Sichtbarkeit
Tabelle	aller abzuholenden Exemplare	ang. Nutz.
Knopf	nächste Seite der obigen Tabelle	ang. Nutz.
Knopf	vorige Seite der obigen Tabelle	ang. Nutz.
Tabelle	aller ausgeliehenen Exemplare	ang. Nutz.
Knopf	nächste Seite der obigen Tabelle	ang. Nutz.
Knopf	vorige Seite der obigen Tabelle	ang. Nutz.

4.4 Anmeldemaske (`login.xhtml`)

Typ	Beschreibung	Sichtbarkeit
Eingabefeld	für die E-Mail-Adresse	jeder
Passworteingabefeld		jeder
Knopf	zum Anmelden	jeder
Knopf[a]	zum Passwortzurücksetzen	jeder

[a]Dargestellt als Hyperlink, um unauffälliger zu sein, da es eine zweitrangige Aktion ist

4.5 Datenschutzerklärung (`privacy-policy.xhtml`)

Typ	Beschreibung	Sichtbarkeit
Eingabefeld	für die Datenschutzerklärung	Admin.
Knopf	zum Speichern der Änderungen an der D.	Admin.

4.6 Direktausleihe (`direct-lending.xhtml`)

Typ	Beschreibung	Sichtbarkeit
Eingabefeld	für die E-Mail-Adresse des Ausleihendens	Bib.-Mit.
Eingabefeld[a]	für die Signatur eines auszuleihenden Exemplars	Bib.-Mit.
Knopf	zum Hinzufügen eines weiteren Signaturfelds	Bib.-Mit.
Knopf	zum Ausleihen	Bib.-Mit.

[a]Beim Laden der Seite fünfmal vorhanden. Durch das Pressen des relevanten Knopfs wird ein weiteres Feld angehängt

4.7 E-Mail-Bestätigung (`email-confirmation.xhtml`)

4.8 Fehlerseite (`error.xhtml`)

4.9 Impressum (`site-notice.xhtml`)

Typ	Beschreibung	Sichtbarkeit
Eingabefeld	für das Impressum	Admin.
Knopf	zum Speichern der Änderungen am Impressum	Admin.

4.10 Kategorienbearbeitung (`category-editor.xhtml`)

Typ	Beschreibung	Sichtbarkeit
Eingabefeld	für den Kategorienamen	Bib.-Mit.
Eingabefeld	für die Kategoriebeschreibung	Bib.-Mit.
Knopf	zum Speichern der Änderungen	Bib.-Mit.

4.11 Kategorierenstöberer (`category-browser.xhtml`)

Typ	Beschreibung	Sichtbarkeit
Eingabefeld	für den Suchterm der Kategoriensuche	jeder
Hyperlink	zur Kategorienbearbeitung (hier: Erstellung)	Bib.-Mit.
Knopf	zum Löschen einer Kategorie	Bib.-Mit.
Hyperlink	zur Medienerstellung[a]	Bib.-Mit.

[a] unter der aktuellen Kategorie

4.12 Kontakt (`contact.xhtml`)

4.13 Leihfristverstöße (`lending-period-violations.xhtml`)

Typ	Beschreibung	Sichtbarkeit
Tabelle	von Exemplaren und Nutzern	Admin.
Knopf	nächste Seite der obigen Tabelle	Admin.
Knopf	vorige Seite der obigen Tabelle	Admin.

4.14 Medienerstellung (`media-creation.xhtml`)

Typ	Beschreibung	Sichtbarkeit

4.15 Medienrückgabe (`return.xhtml`)

Typ	Beschreibung	Sichtbarkeit

4.16 Mediensuche (`media-search.xhtml`)

Typ	Beschreibung	Sichtbarkeit
Eingabefeld	für den Suchterm der freien Suche	jeder
Knopf	zur Durchführung der Suche	jeder
Drop-Down-Liste[a]	für den Suchoperator	jeder
Drop-Down-Liste[a]	für das Suchkriterium	jeder
Eingabefeld[a]	für den Suchterm der diff. Suche	jeder
Knopf	zum Hinzufügen eines weiteren Suchfelds	jeder
Tabelle	aller Suchergebnisse (falls vorhanden)	jeder
Knopf	nächste Seite der obigen Tabelle	jeder
Knopf	vorige Seite der obigen Tabelle	jeder

[a]Beim Laden der Seite dreimal vorhanden, jeweils einmal pro Dreiergruppe bestehend aus mit [a] markierten Elementen. Durch das Pressen des relevanten Knopfs wird eine weitere Dreiergruppe angehängt

4.17 Mediumsansicht (`medium.xhtml`)

Typ	Beschreibung	Sichtbarkeit
Hyperlink	zurück zur Trefferliste (Mediensuche)	jeder[a]
Eingabefeld[b]	für ein Medienattribut	Bib.-Mit.
Knopf	zum Speichern der Änderungen an den Medienattributen	Bib.-Mit.
Knopf	zur Bindung an die Abholung des Mediums	ang. Nutz.[c]
Zeitdauereingabefeld	für die Rückgabefrist	Bib.-Mit.
Knopf	zum Speichern der Änderungen an der Rückgabefrist	Bib.-Mit.
Hyperlink	zum Löschen des Mediums; zur zuvor aufgerufenen Seite	Bib.-Mit.
Tabelle	aller Exemplare	jeder
Knopf[d]	zum Löschen eines Exemplars	Bib.-Mit.
Knopf[d]	zum Stornieren einer Abholung	Bib.-Mit.
Hyperlink[d]	zur Direktausleihe	Bib.-Mit.
Knopf[d]	zur Bindung an die Abholung des Mediums	ang. Nutz.
Eingabefeld	für den Standort eines neuen Exemplars	Bib.-Mit.
Eingabefeld	für die Signatur eines neuen Exemplars	Bib.-Mit.
Knopf	zum Erstellen eines Exemplars	Bib.-Mit.

[a]Nur sichtbar, falls der Nutzer von der Suche kommt
[b]Existiert pro Medienattribut
[c]Nur, falls dem Nutzer erlaubt ist auszuleihen
[d]existiert pro Exemplar in der Tabelle

4.18 Mediumschemabearbeitung (`medium-schema-editor.xhtml`)

Typ	Beschreibung	Sichtbarkeit

4.19 Nutzersuche (`user-search.xhtml`)

Typ	Beschreibung	Sichtbarkeit
Eingabefeld	für den Suchterm	Admin.
Knopf	zur Durchführung der Suche	Admin.
Tabelle	aller Suchergebnisse (falls vorhanden)	Admin.
Knopf	nächste Seite der obigen Tabelle	Admin.
Knopf	vorige Seite der obigen Tabelle	Admin.

4.20 Passwortzurücksetzung (`password-reset.xhtml`)

Typ	Beschreibung	Sichtbarkeit
Passworteingabefeld	für das neue Passwort	jeder
Passworteingabefeld	zur Bestätigung	jeder
Knopf	zum Zurücksetzen des Passworts	jeder

4.21 Profilseite (`profile.xhtml`)

Typ	Beschreibung	Sichtbarkeit
Eingabefeld	für den Vornamen	ang. Nutz.
Eingabefeld	für den Nachnamen	ang. Nutz.
Passworteingabefeld		ang. Nutz.
Passworteingabefeld	zur Bestätigung	ang. Nutz.
Eingabefeld	für die E-Mail-Adresse	ang. Nutz.
Eingabefeld[a]	für die Adresse	ang. Nutz.
Drop-Down-Liste	für die Nutzerrolle	Admin.
Checkbox	für den Accountstatus (gesperrt / entsperrt)	Admin.
Checkbox	für den Ausleihstatus (gesperrt / entsperrt)	Admin.
Eingabefeld	für die Rückgabefrist	Admin.
Knopf	zum Speichern der Änderungen an den Benutzerdaten	ang. Nutz.
Hyperlink	zu den abzuholenden und ausgeliehenen Exemplaren	ang. Nutz.
Knopf	zum Schließen des Accounts; zur Anmeldemaske	ang. Nutz.
Knopf	zum Löschen des Nutzers; zur Verwaltungsseite	Admin.

[a] bzw. getrennt in Ort, PLZ, Straße, Hausnummer

4.22 Registrierungsseite (`registration.xhtml`)

Typ	Beschreibung	Sichtbarkeit
Eingabefeld	für den Vornamen	ang. Nutz.
Eingabefeld	für den Nachnamen	ang. Nutz.
Passworteingabefeld		ang. Nutz.
Passworteingabefeld	zur Bestätigung	ang. Nutz.
Eingabefeld	für die E-Mail-Adresse	ang. Nutz.
Eingabefeld[a]	für die Adresse	ang. Nutz.
Drop-Down-Liste	für die Nutzerrolle	Admin.
Knopf	zum Registrieren des Accounts	abg. Nutz./Admin.

[a] bzw. getrennt in Ort, PLZ, Straße, Hausnummer

4.23 Verwaltungsseite (`administration.xhtml`)

Typ	Beschreibung	Sichtbarkeit
Eingabefeld	für die Rückgabefrist	Admin.
Eingabefeld	für den Mahnungszeitpunkt	Admin.
Eingabefeld	für die Abholfrist	Admin.
Eingabefeld	für den Systemnamen	Admin.
Checkbox	für den Zugangsstatus (anonym oder nicht)	Admin.
Checkbox	für den Registrierungsstatus	Admin.
Eingabefeld	für den regulären Ausdruck valider E-Mail-Adressen	Admin.
Farbkreis[a]	für die erste Farbe des Systems	Admin.
Farbkreis[a]	für die zweite Farbe des Systems	Admin.
Datei-Upload	für das Logo des Systems	Admin.
Knopf	zum Speichern der Änderungen an den Systemeinstellungen	Admin.
Hyperlink	zu der Medienerstellung	Admin.
Hyperlink	zu der Registrierungsseite (fremden Nutzer erstellen)	Admin.
Hyperlink	zur Mediumsschemabearbeitung	Admin.
Hyperlink	zu den Leihfristverstößen	Admin.
Hyperlink	zu der Nutzersuche	Admin.
Eingabefeld	für den Suchterm der Nutzersuche	Admin.

[a] Alternativ ein einfaches Eingabefeld

5 Systemfunktionen

Autor: Jonas Picker

5.1 Technische Systemsicherheit

Kommunikationsverschlüsselung: Durch die CA-Zertifizierung des Servers wird eine TLS-Transportverschlüsselung bei der Kommunikation zwischen Klient und Server verwendet. Ist die Datenbank, wie im Pflichtenheft (Abschnitt 4.3 'Server') beschrieben, über SSL-VPN angebunden, ist die Kommunikation zwischen ihr und dem Server ebenfalls verschlüsselt.
Nutzerberechtigungen: Für die Überprüfung der Zugangsberechtigungen bei jeder HTTPS-Anfrage implementiert die Klasse !!!!!! das PhaseListener-Interface, welches zu implementierende Methoden zum Ausführen von Code vor oder nach jeder Phase des JSF-Life-Cycle vorgibt. In unserem Fall wird am frühestmöglichen Punkt (vor der Restore-View-Phase) geprüft, ob der Anfragesteller auch berechtigt ist, die vorhergesehene Antwort vom Server zu erhalten. Aus Redundanzgründen verwenden wir für Seiten, auf die mehrere Nutzerrollen Zugriff haben, die gleichen Facelets. In den jeweiligen Backing-Beans wird dann, durch Zugriff auf die von JSF getrackte Nutzer-Session der Klasse !!!!!!, die Rolle des Benutzers überprüft. Für jeden Seitenbesucher werden dann nur die rollenspezifischen Knöpfe und Anzeigen gerendert (siehe Abschnitt !!!JSF-Facelets!!!). Durch die inklusive Rollenhierarchie lassen sich so alle Funktionalitäten der Rollen im gleichen Facelet einbinden.
Session-Hijacking[1]**:** Um diesen Angriffsvektor zu schützen, wird der Identifikator der Nutzer-Session an kritischen Stellen (z.B. nach dem Login) manuell ausgetauscht.
Cross-Site-Scripting[2]**:** Das Escapen von HTML-Sonderzeichen wird von JSF bei allen nutzergenerierten Teilen der Anwendung unterstützt. Alle Elemente der JSF-Facelets, die nutzergenerierten In- und Output weitergeben, haben den impliziten Standartwert 'escape=true'. Da wir diesen in unserer Implementierung nie manuell auf 'false' setzen, ist XSS in dieser Applikation unmöglich.
SQL-Injection[3]**:** Durch das Konsequente Verwenden der 'Prepared Statements' in unserer !!!Datenzugriffsschicht!!! beugen wir SQL-Injections vor. Diese JDBC-Funktionalität trennt das eigentliche SQL-Statement von den nutzergenerierten Parametern und Verhindern so das Ausführen von maliziösem SQL-Code in der Datenbank.

5.2 Logging

Mit der Klasse !!!Logger!!! ist unser System mit einer eigenen Log-Funktion ausgestattet. Diese dient sowohl zum Debugging während der Entwicklung, als auch zum Protokollieren der Fehler und Abläufe im laufenden System. Durch Setzten der Variable 'LOG_CONSOLE: ' mit den Werten 'TRUE' oder 'FALSE' kann eingestellt in der Konfigurationsdatei[4] werden, ob die Meldungen auch in Echtzeit auf der Konsole ausgegeben oder nur in das Log-File geschrieben werden. Es gibt drei Log-Level, zwischen denen, durch Setzen der Variable 'LOG_LEVEL: ' mit einem der unten aufgeführten Werte, beim Systemstart umgeschaltet werden kann. Die folgenden Log-Stufen sind inklusiv, die Letzte schließt somit die oberen beiden mit ein.

[1] Das Stehlen einer validen Nutzer-Session um Zugriff auf den Account und seine Berechtigungen zu erhalten
[2] Auch XSS: das Einschleusen von browserinterpretierbarem HTML-Code auf ungesicherte Teile einer Website
[3] Einschleusen von SQL-Code in Formularfelder, um Zugriff auf die Datenbank zu erhalten
[4] config.properties im Ordner /WEB-INF (siehe Installationsanleitung)

'SEVERE': Diese Einstellung des Loggers protokolliert nur schwere Fehler, die unmittelbare Konsequenzen für den Anwendungsbetrieb haben. Um ein schnelles Volllaufen des Log-Files zu vermeiden, ist dies die Standarteinstellung.
'DETAILED': Fehler und fehlgeschlagene Prozeduren, die die Integrität der Anwendung nicht gefährden, werden zusätzlich protokolliert.
'DEVELOPMENT': Hierunter fallen sowohl Protokollierungen von erfolgreichen oder seltenen Abläufen, als auch sonstige nützliche Meldungen. Da das Log-File schnell sehr groß und unübersichtlich werden könnte, empfehlen wir, diese Option nur bei Problemen zu wählen.

5.3 Selbstständige Abläufe

Die Klasse !!!!!! ermöglicht dem System, selbstständig in einstellbaren Abständen Wartungsaufgaben durchzuführen. Alle unten aufgelisteten Aufgaben sind somit von der der aktuellen Systemzeit des Servers abhängig und werden darüberhinaus nur mit einer maximalen zeitlichen Unsicherheit des gewählten Intervalls durchgeführt. Die Einstellung wird beim Systemstart durch den gesetzten Wert der Variable 'SCAN_INTERVAL: ' bestimmt. Der von uns empfohlene Standartwert repräsentiert eine Minute, sollten Sie diesen verlängern wollen tragen Sie stattdessen eine andere positive ganze Zahl ein. Der Wartungsthread vergleicht die Operationsfrist der zur Abholung markierten Exemplare mit der aktuellen Systemzeit und setzt bei Überschreitung den Verfügbarkeitsstatus wieder auf 'verfügbar'. Bei Exemplaren mit dem Status 'ausgeliehen' wird, zusätzlich zur Frist, die Mahnungsversatzzeit beachtet und bei Bedarf die Klasse !!!!!! zum Versenden einer E-Mail angestoßen. Die abgelaufenen Tokens der Benutzer-Tabelle werden ebenfalls gelöscht. Zum Versenden der E-Mails wird das SMTP-Protokoll ohne Authentifikation verwendet. Dazu muss in der Konfigurationsdatei die Variable 'MAILSERVERHOST: ' mit der Adresse des verwendeten Mail-Servers und 'MAILSERVERPORT: ' mit dessen Port (Standart ist 25) befüllt werden. Außerdem muss die E-Mail-Adresse eines stellvertretenden Absenders für die Bibliothek in die Variable 'MAILSOURCE: ' eingetragen werden.

5.4 Datenbankverbindung

Wenn die Java Laufzeitumgebung des Systems plötzlich beendet wird und das System abstürzt, wird trotzdem mittels einer ShutdownHook noch das Schließen offener Datenbankverbindungen versucht. Außerdem wird durch das logische Zusammenfassen der voneinander abhängigen Datenbanktransaktionen in den Steuermethoden der !!!DAOs!!! sichergestellt, dass das ACID-Prinzip bestmöglich eingehalten und die Datenbank im Fehlerfall in einem konsistenten Zustand hinterlassen wird. Dies ist jedoch (z.B. bei einem Stromausfall) nicht immer möglich. Wie !!!hier!!! beschrieben, wird die Datenbank durch Connection Pooling geschont.

5.5 Starten und Stoppen der Anwendung

Start: Durch Implementierung des SystemEventListener-Interface werden beim Systemstart von der Klasse!!!!!! alle weiteren Aktionen aus der !!!Logikschicht!!! heraus angestoßen. Die Initialisierung des !!!Loggers!!! geschieht zuerst, danach wird die Konfigurationsdatei vom !!!!! eingelesen, das Log-Level und der gewählte Konsolenausgabemodus gesetzt und die Parameter für die E-Mail Funktionen der Klasse !!!!!! übergeben. Mit den restlichen Parametern wird eine !!!eigene Initialisierungsklasse!!! für Prozesse der !!!Datenzugriffsschicht!!! aufgerufen, in der zunächst die Datenbankverbin-

dung überprüft wird. Sollten die Verbindung fehlschlagen, wird, ähnlich zu !!!diesem Prozess!!!, eine Fehlermeldung ausgegeben/geloggt. Im Erfolgsfall wird dann überprüft, ob die benötigten Tabellenstrukturen bereits vorhanden sind. Falls nicht wird das manuelle Anlegen der Tabellen (Mediumsschema wird mit dem Standartattributsatz befüllt) über einen Konsoleninput mit 'Y' oder 'N' entschieden werden müssen. Bei existierenden Tabellen wird nun der !!!Connection-Pool!!! initialisiert und der !!!Wartungsthread!!! aktiviert um den Startprozess der !!!Datenzugriffsschicht!!! abzuschließen. Es wird jetzt das zuletzt ausgewählte Farbschema aus der Datenbank abgefragt, und als Parameter an die !!!obere Schicht!!! weitergeleitet, wo dann das ausgewählte Farbschema eingestellt wird. Danach ist das System betriebsbereit.
Stop: Beim planmäßigen Herunterfahren des Systems durch das Ausschalten des Tomcat auf dem Server (z.B. via shutdown.sh) reicht die Klasse !!!!!! eine Nachricht an die Klasse !!!!!!, von wo aus zunächst der !!!Wartungsthread!!! kontrolliert beendet und danach die noch offenen Datenbankverbindungen über den !!!Connection-Pool!!! geschlossen werden. Nach Durchreichen der Erfolgsmeldung nach oben, wird die Anwendung gestoppt. Nutzersessions überleben das Ausschalten des Servers nicht.

6 Datenfluss

Autor: Sergei Pravdin

Die Kommunikationen zwischen den Klassen und die Interaktionen des Systems werden durch den Sequenzdiagramme abgebildet. Um einen Datenfluss beispielhaft zu zeigen, werden die zwei Szenarien vorgelegt. Zuerst bucht ein angemeldeter Nutzer ein Medium-Exemplar erfolgreich zur Ausleihe. Im zweiten Szenario bucht ein angemeldeter Nutzer ein Medium-Exemplar erfolglos zur Ausleihe, weil die Verbindung mit der Datenbank fehlgeschlagen ist.

6.1 Interaktionen beim erfolgreichen Buchen eines Medium-Exemplars

Der Nutzer befindet sich auf der Mediensuche-Seite und wünscht das Buch 'Programmieren lernen' zu buchen. Im System existiert das Medium mit dem Titel 'Programmieren lernen' und der Signatur '17RE'. Ein Exemplar mit der Signatur '17RE (+1)' gehört zu dem genannten Medium. Das System ist so eingestellt, dass die angemeldeten Nutzer Zugriff auf den Medien haben.

6.1.1 Suchen nach dem Medium

Der Nutzer gibt 'Programmieren lernen' und '17RE' in die Suchfelder 'Titel' und 'Signatur' und klickt auf den Suchen-Button. Das Mediensuche-Bean kapselt die Sucheingabe ins Mediensuche-DTO ein. Das Mediensuche-DTO ruft die Methode 'getMedien' aus dem Mediensuche-DAO auf. Das Mediensuche-DAO ruft die Methode 'getConnection' aus dem Connection-Pool-Bean und bekommt eine Connection von dem zurück. Danach führt das Mediensuche-DAO eine selectSQL-Anfrage durch und gibt eine Liste der entsprechenden Medien dem Mediensuche-DTO zurück. Im nächsten Schritt leitet Mediensuche-DTO diese Liste der Medien dem Mediensuche-Bean weiter. Das Mediensuche-Bean ruft seine Methode 'updateMedien' auf und gibt dem Nutzer die aktualisierte Mediensuche-Seite zurück.

6.1.2 Navigation zum Medium

Der Nutzer klickt auf das angezeigte Medium 'Programmieren lernen'. Das Mediensuche-Bean ruft die Methode 'navigate' auf und gibt die Mediumsansicht-Seite zurück.

6.1.3 Buchen eines Exemplares

Der Nutzer klickt auf den Buchen-Button. Das Medium-Bean ruft die Methode checkStatus aus dem UserSession-Bean auf, um zu prüfen, ob der Nutzer Zugriff zum Buchen hat. Das UserSession-Bean gibt das positive Ergebnis dem Medium-Bean zurück. Das Medium-Bean kapselt das Buchen ins Medium-DTO ein und das Medium-DTO ruft die Methode book() aus dem Medium-DAO auf. Das Medium-DAO ruft die Methode 'getConnection' aus dem Connection-Pool-Bean und bekommt eine Connection von dem zurück. Danach führt das Medium-DAO eine updateSQL-Anfrage durch. Im nächsten Schritt gibt Medium-DTO dem Medium-Bean das Ergebnis der Operation zurück. Das Medium-Bean ruft seine Methode 'updateExamples' auf und gibt dem Nutzer die aktualisierte Medium-Seite zurück. Das Exemplar ist erfolgreich gebucht.

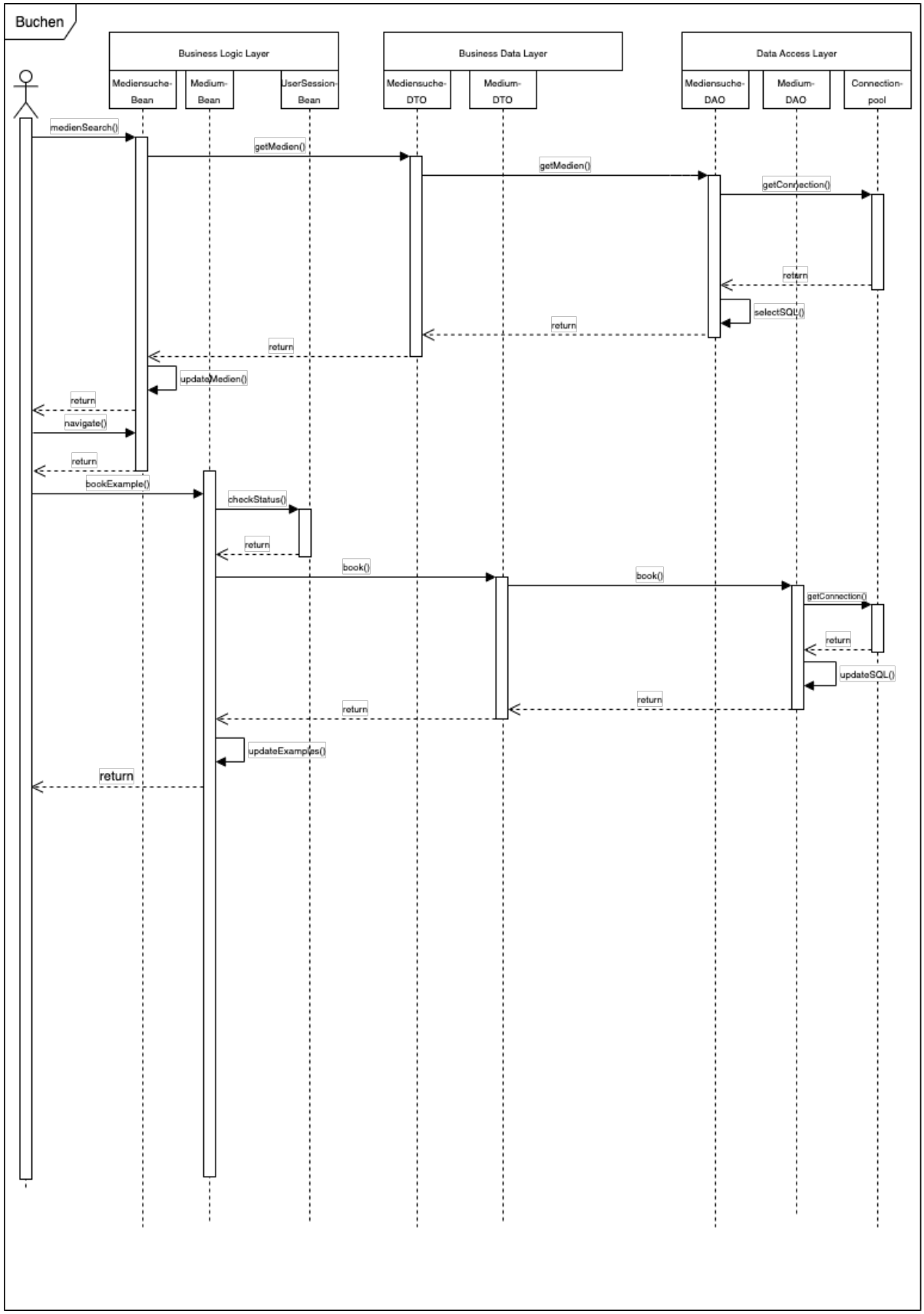


Abbildung 9: Interaktionen bei einem erfolgreichen Buchen eines Medium-Exemplars

7 ER-Modell

Autor: Jonas Picker

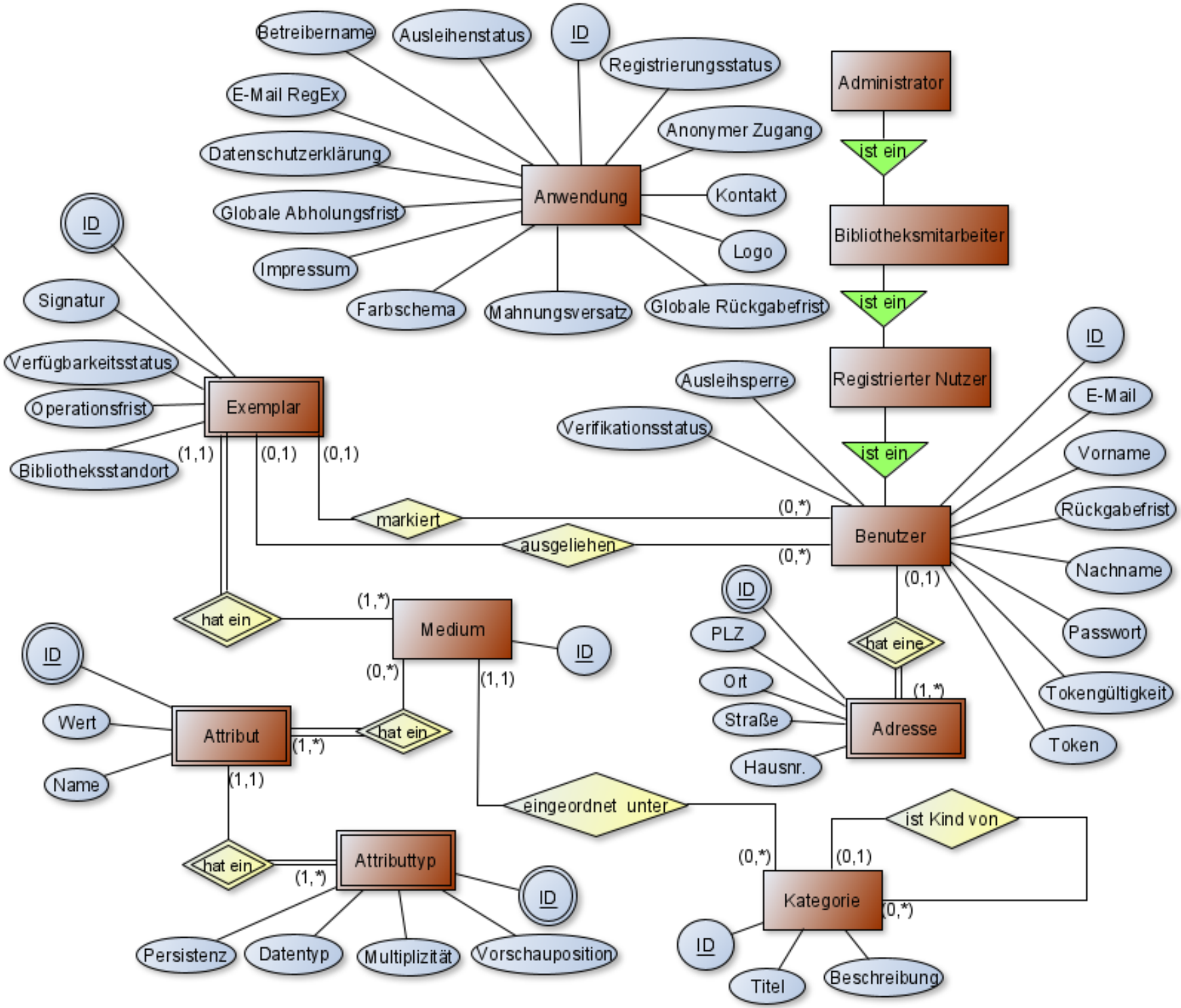


Abbildung 10: Entity-Relationship Diagramm

7.1 Legende

Beschreibung des ER-Diagramms:

Medium: Diese Entität modelliert die von der Bibliothek verwalteten Medien. Außer des Primärschlüssels besitzt sie mindestens ein zugehöriges Exemplar und eine variable Anzahl von Attributen. Der Standartattributsatz für Medien wird unten aufgeführt. Es können benutzerdefinierte Attribute hinzugefügt werden (PfHft. /F380/) und jedes der folgenden Attribute ist löschbar (PfHft. /F381/):

Titel	Erscheinungsjahr	Verlag
Medientyp	Version	Freitext
Autoren	Index (ISBN/ISSN)	Link auf elektronische Version

Attribut: Diese schwache Entität hält den Namen und Wert eines Medienattributs, ihr ist genau ein Attributtyp zugeordnet.
Attributtyp: Vom Attribut abhängige, schwache Entität. Hiermit werden dem Attribut zugehlrige Eigenschaften modelliert. 'Vorschauposition' bestimmt, ob und an welcher Stelle das Attribut in der Medienvorschau (z.B. in der Listenansicht der Suchergebnisse) angezeigt werden soll. 'Multiplizität' entscheidet, ob das zugehörige Attribut mehrfach pro Medium mit unterschiedlichen Werten vorkommen kann oder nicht. 'Persistenz' ist eine Markierung für Attribute, die nicht zum modifizierbaren Attributsatz gehören (z.B. die Rückgabefrist für Medien). 'Datentyp' beschreibt die im Attributwert gespeicherten Daten (z.B. 'String' für Textattribute oder 'Image' um Bilder zu speichern).
Kategorie: Die mit dieser Entität verbundenen Relationen ordnen jedem Medium genau eine Kategorie zu und modellieren die Kategoriehierarchie (PfHft. /W440/) durch die Selbstbeziehung. Es wird einen unlöschbaren Top-Knoten in der Hierarchie geben, zu dem alle Medien, die nie in eine Kategorie eingeteilt wurden oder deren Kategorie gelöscht wurde, gehören. Sollten alle Medien in benutzerdefinierten Kategorien stecken, hat der Top-Knoten keine zugeordneten Medien und nimmt somit nicht an der 'eingeordnet unter'-Relation teil.
Exemplar: Diese schwache Entität ist vom zugehörigen Medium abhängig. Ein bestimmtes Exemplar kann von genau einem Nutzer zur Abholung markiert oder ausgeliehen werden, diese Aktionen schließen sich gegenseitig aus (PfHft. /F310/) und ändern (genau wie eine Rückgabe) den Verfügbarkeitsstatus und die dazugehörige Operationsfrist dementsprechend. 'Signatur' und 'Bibliotheksstandort' halten bibliothekspezifische Kodierungen.
Benutzer: Ob ein Benutzer die Ausleihfunktion benutzen kann, wird durch das Attribut 'Ausleihsperre' modelliert. 'Verifizierungsstatus' zeigt hingegen an, ob der Nutzeraccount bereits den Verifizierungsprozess durchlaufen hat (PfHft. /W70/). Das Passwort wird in gehashter Form abgespeichert. Zur Passwortzurücksetzung und E-Mail-Verifikation wird pro Nutzer ein begrenzt gültiges, einzigartiges 'Token' verwendet, um den entsprechenden Link zu bauen. Die 'Rückgabefrist' für Ausleihen eines Benutzers wird ebenfalls modelliert.
Registrierter Nutzer: Diese Nutzer haben positiven 'Verifizierungsstatus'.
Bibliotheksmitarbeiter: Diese Entität modelliert die Rolle der Bibliotheksmitarbeiter.
Administrator: Diese Entität modelliert die Rolle der Administratoren. Die von der Benutzerentität ausgehende Hierarchie soll die inklusiven Rollen im System modellieren.
Adresse: Vom Benutzer abhängig. Enthält die Bestandteile einer Nutzeradresse als Attribute.
Anwendung: Hier werden die setzbaren globalen Variablen und Anwendungseinstellungen modelliert. 'Anonymer Zugang' bestimmt die Berechtigungen anonymer Nutzer beim Besuchen des

Webspaces (PfHft. /F10/), während 'Registrierungsstatus' eine offene (in diesem Fall gilt der 'E-Mail RegEx') oder geschlossene Registrierungsfunktion modelliert (PfHft. /F20/). 'Ausleihenstatus' steht für das Umschalten des Systems zur manuellen Freischaltung der Ausleihfunktion. Aus dem 'Mahungsversatz' ergibt sich der Zeitpunkt vor Ablauf einer Rückgabefrist, an dem eine automatische Benachrichtigung versendet wird (PfHft. /F240/). 'Farbschema' merkt sich das zuletzt ausgewählte Farbschema.