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

3 Klassendiagramm

Autor: Mohamad Najjar

4 JSF-Dialoge

Autor: León Liehr

5 Systemfunktionen

Autor: Jonas Picker

5.1 Technische Systemsicherheit

Kommunikationserschlüsselung: Durch die CA-Zertifizierung des Servers wird eine TLS-Transportverschlüsselung bei der Kommunikation zwischen Klient und Server verwendet. Ist die Datenbank, wie im 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. Durch die inklusive Rollenhierarchie lassen sich so alle Funktionalitäten der Rollen im gleichen Facelet einbinden.
Session-Hijacking[1]**:** Es wird durch das manuelle Austauschen des Identifikators für die Nutzer-Session an kritischen Stellen dafür gesorgt, dass Session-Hijacking unmöglich wird.

[1] Das Stehlen einer validen Nutzer-Session um Zugriff auf den Account und seine Berechtigungen zu erhalten

Cross-Site-Scripting[2]**:** 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, greift das JSF-interne Escapen von HTML-Sonderzeichen bei allen nutzergenerierten Teilen der Anwendung.
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 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ist unser System mit einer eigenen Log-Funktion ausgestattet. Diese dient sowohl zum Debugging während der Entwicklung, alsauch zum Protokollieren der Fehler und Abläufe im laufenden System. Durch Setzten der Variable 'LOG_CONSOLE: ' mit den Werten 'TRUE' oder 'FALSE' kann eingestellt 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: ' in der Konfigurationsdatei mit einem der unten aufgeführten Werte beim Systemstart umgeschaltet werden kann. Die folgenden Kategorien sind inklusiv, die Letzte schließt somit die oberen beiden mit ein.
'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 (z.B. !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!wie hier) werden zusätzlich protokolliert.
'DEVELOPMENT': Hierunter fallen sowohl Protokollierungen von erfolgreichen oder seltenen Abläufen, alsauch 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.

[2]Einschleusen von browserinterpretierbarem HTML-Code auf ungesicherte Teile einer Website
[3]Einschleusen von SQL-Code in Formularfelder, um Zugriff auf die Datenbank zu erhalten

5.4 Datenbankverbindung

Wenn die Java Laufzeitumgebung des Systems plötzlich beendet wird und das System abstürzt, wird mittels einer ShutdownHook trotzdem noch das Schließen offener Datenbankverbindungen versucht. Außerdem wird durch das logische Kapseln der voneinander abhängigen Datenbanktransaktionen durch die Klasse !!!!!!!!!!!!!!!!!! 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.

5.5 Starten und Stoppen der Anwendung

Start: Durch Implementierung des SystemEventListener-Interface werden beim Systemstart von der Klasse !!!!!!!!!!!!!!!!!!!!!!!!!!! alle weiteren Aktionen angestoßen. Die Initialisierung des !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!Loggers geschieht zuerst, danach wird die Konfigurationsdatei eingelesen, das Log-Level gessetzt und mit den restlichen Parametern eine !!!!!!!!!!!!!!!!!!!!!!!!!!eigene Initialisierungsklasse für Prozesse der !!!!!!!!!!!!!!!!!!!!Buisnesslogik/Serviceschicht aufgerufen. Diese wiederum gibt die Initialisierung der !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!Datenzugriffsschicht an eine !!!!!!!!!!!!!!!!!!!!!!!!!!!!weitere Klasse ab, in der zunächst die Datenbankverbindung überprüft wird. Sollten die Verbindung fehlschlagen, wird !!!!!!!!!!!!!!!!!!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 !!!!!!!!!!!!!!!!!!!!!!!Default befüllt) über einen Konsoleninput mit 'Y' oder 'N' entschieden werden müssen. Bei existierenden Tabellen wird nun der !!!!!!!!!!!!!!!!!Connection-Pool initialisiert und der Startprozess der !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!Datenzugriffsschicht ist abgeschlossen. Es wird die !!!!!!!!!!!!!!!!!!!!darüberliegende Schicht benachrichtigt und von !!!!!!!!!!!!!!!!!!!!!!!!!!!dort aus der !!!!!!!!!!!!!!!!!!Wartungsthread aktiviert. Die nun folgende Erfolgsmeldung zurück an den !!!!!!!!!!!!!!!!!!!!!!!!!!!SystemEventListener gibt das momentan ausgewählte Farbschema nach oben durch, 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 der !!!!!!!!!!!!!!!!!!!!!!!!!!!SystemEventListener eine Nachricht an die Klasse!!!!!!!!!!!!!!!!!!, von wo aus zunächst der !!!!!!!!!!!!!!!!!!Wartungsthread beendet und danach die !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!Datenzugriffsschicht kontaktiert wird. Von !!!!!!!!!!!!!!!!!!!!!!!!!!!!dort aus werden zunächst die noch offenen Datenbankverbindungen über den !!!!!!!!!!!!!Connection-Pool geschlossen und, nach Durchreichen der Erfolgsmeldung nach oben, die Anwendung gestoppt. Nutzersessions überleben das Ausschalten des Servers nicht.

6 Datenfluss

Autor: Sergei Pravdin

7 ER-Modell

Autor: Jonas Picker

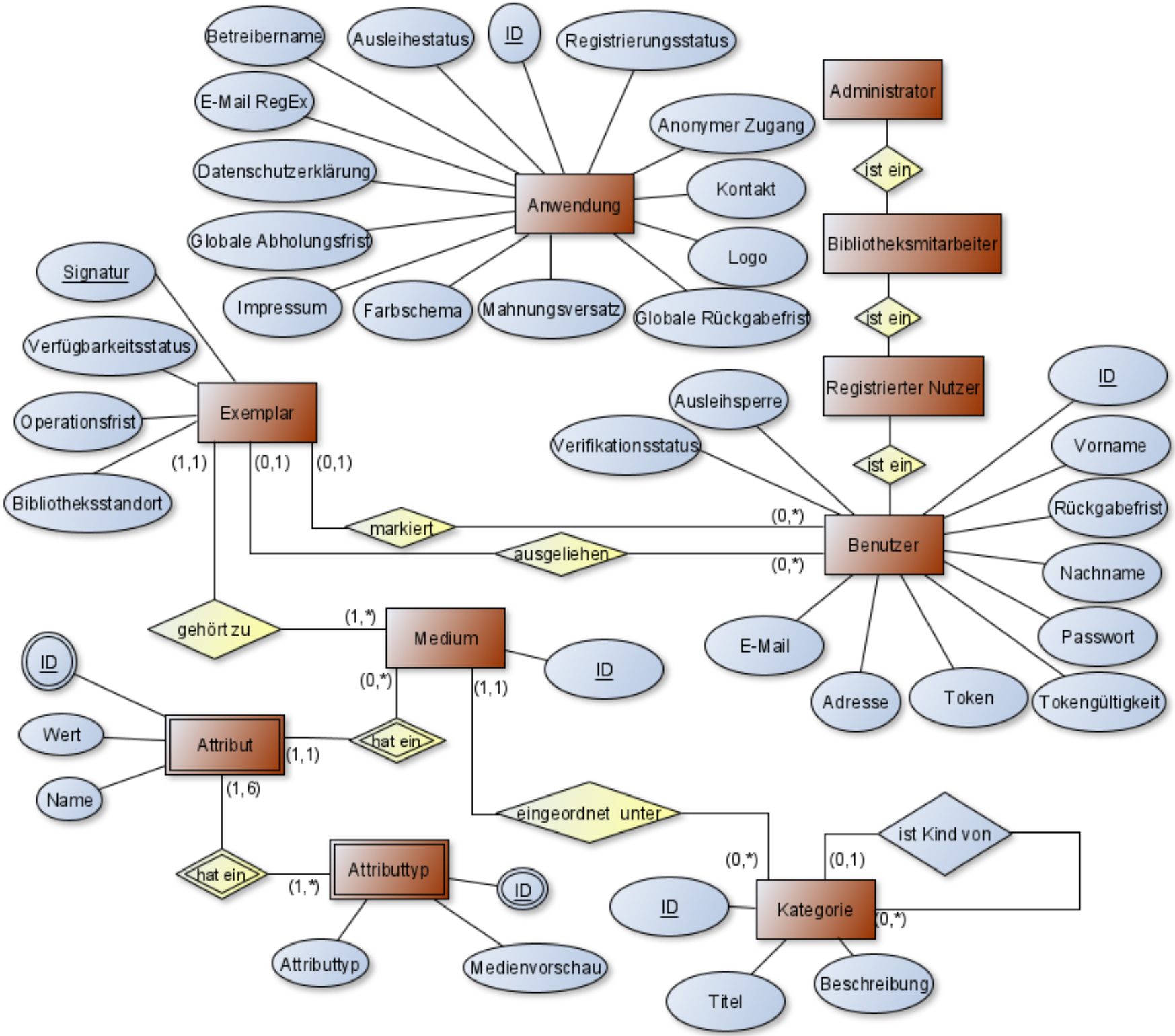

Abbildung 1: Entity-Relationship Diagramm

7.1 Legende

Jede Entität wird im Folgenden zusammen mit ihren nicht offensichtlichen Attributen und Relationen kurz beschrieben.
Medium: Diese Entität modelliert die in der Bibliothek gehaltenen Medien. Zusätzlich zum Primärschlüssel ist auch das Attribut 'Rückgabefrist' und 'Icon' nicht vom Administrator entfernbar, die restlichen Attribute (kursiv) stellen den modifizierbaren Standartsatz der Medienattribute dar (PfHft. /D020/). Erwähnenswert ist hier noch das als mehrwertig markierte, editierbare Attribut 'Autoren', diese Markierung kann auch zu benutzerdefinierten Attributen hinzugefügt werden und wird mit dem Anlegen einer neuen Entität modelliert (PfHft. /F380/).
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: Von jedem Medium muss mindestens ein Exemplar vorhanden sein. Auch kann ein bestimmtes Exemplar 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.
Benutzer: Ob ein Benutzer die Ausleihfunktion benutzen kann, wird durch das Attribut 'Ausleihstatus' modelliert. 'Accountstatus' zeigt hingegen an, ob der Nutzeraccount bereits den Verifizierungsprozess durchlaufen hat (PfHft. /W70/). Das Passwort wird in gehashter Form abgespeichert.
Linkgenerator: Diese Entität kapselt einen befristet gültigen URL-Suffix, aus dem der Verifizierungs- oder Passwortzurücksetzungslink (je nach Zweck) für genau einen Account erstellt wird.
Anwendung: Hier werden die setzbaren globalen Variablen und Anwendungseinstellungen gespeichert. Diese Tabelle hat nur einen einzigen Eintrag. 'Anonymer Zugang' modelliert die Berechtigungen anonymer Nutzer beim Besuchen des Webspaces (PfHft. /F10/), während 'Registrierungsstatus' eine nutzerfreundlichere Alternative als den RegEx zum Sperren der Registrierung bietet (PfHft. /F20/). 'Ausleihstatus' steht für das Umschalten des Systems zur manuellen Freischaltung der Ausleihfunktion für registrierter Nutzer. 'Look & Feel' speichert das momentan ausgewählte Farbschema.