Einzelprüfbericht	**Prüfgegenstand:** Sopra-projekt Team 17/23 **Inspektor:** Julius Friedrich **Überprüfungsaufwand in Minuten: 225**	**Anzahl der gefundenen Auffälligkeiten nach ihrer Kritikalität:** u=unbedeutend (Auffälligkeit kann bearbeitet werden) b=bedeutend(Auffälligkeit muss bearbeitet werden) k=kritisch(wie b,verletzt aber essentielle Projektziele) o= Optimierungspotential(kann später bearbeitet werden) f= Frage (wird in Sitzung geklärt und ggf. eingestuft)

Lfd. Nr.	Position im Prüfgegenstand	Beschreibung der Auffälligkeit	Checklisten Referenz	Kritikalität
1	DB-Manager	Beschränkte Anzahl von Connections (wenn nicht geschlossen, irgendwann alle gelockt), dann keine weiteren Connections mehr möglich => Deadlock	done	k
2	DB-Manager	Prepared Statements teilweise nicht geschlossen	done	k
3	DB-Manager	Beim Erstellen von Veranstaltungen in der Datenbank war „autocommit“ nicht auf false gesetzt => wenn mehrere Daten hintereinander in die Datenbank geschrieben werden, haben diese nicht als Transaktion funktioniert (kein Rollback, wenn Fehler auftritt). Führt zu inkonsistentem System.	done	k
4	DB-Manager	SQLException muss geworfen werden, wenn Resultsets leer sind.	done	k
5	DB-Manager	Beim Löschen von Benutzern wird die Connection nicht geschlossen.	done	u
6	DB-Manager	Die Veranstaltung kann beim Lesen des Wrappers null returnen, dies muss abgefragt werden und je nach dem weitergegeben werden oder direkt eine Exception werfen.	done	k
7	DB-Manager	Die Klasse BenachrNeuerKommentar muss groß geschrieben werden, wenn sie als Rückgabewert der Methode leseBenachrNeuerKommentar() verwendet wird (Java Konvention)	done	k
8	DB-Manager	Beim Löschen von Benutzern wird die Connection nicht geschlossen.	done	u
9	DB-Manager	Beim Schreiben von Benachrichtigungen wird keine SQL-Exception abgefangen (try-catch-Block fehlt)	done	b
10	DB-Manager	Roleback fehlt beim Bearbeiten von Veranstaltungen, falls eine DbUniqueConstraintException geworfen wurde (kann zu Inkonsistenzen in der Datenbank führen).	done	k
11	DB-Manager	Die Methode leseVorgänger(), bzw. leseNachfolger() sind relativ komplex, da hier Neo4j-Statements verwendet werden und nicht jeder Java-Programmierer Neo4j beherrscht. Hier wäre eine ausführlichere Kommentation wünschenswert.	done	u
12	DB-Manager	Die Suchfunktion liefert mit der Levenshtein-SQL-Funktion oft nicht erwartete Ergebnisse, dies liegt hauptsächlich an der Levenshtein-Funktion selbst. Man könnte diese später noch selbst optimieren oder eine andere/bessere Funktion implementieren.	done	o
13	DB-Manager	Beim Schreiben einer Veranstaltung fehlt das Abfangen einer mögichen DbFalseLoginDataException. Diese muss abgefangen werden, wenn das übergebene Passwort nicht mit dem aus der Datenbank übereinstimmt.	done	k

14	DB-Manager	Beim Bearbeiten der Benutzerdaten in der Datenbank wurde die Emailadresse vergessen. Diese muss aus dem Benutzerobjekt ausgelesen werden und in das PreparedStatement eingebaut werden.	done	b
15	DB-Manager	Wenn das Profilbild eines Benutzers gelöscht wird, muss auch in die Datenbank geschrieben werden, dass nun das „default.png“ verwendet werden soll. Ist dies nicht der Fall wird nach dem Löschen des Bildes bspw. beim Anschauen der Profilseite dieses Nutzers kein Bild angezeigt („Image not found“-Icon).	done	u
16	DB-Manager	Beim Bearbeiten einer Veranstaltung werden alle Moderatoren zuerst aus der Datenbank gelöscht und dann neu eingefügt (Die schon vorhandenen werden zwar nicht erneute benachrichtigt, aber performanter für die Datenbank wäre es, erst zu prüfen und dann nur die neuen einzutragen).	done	o
17	DB-Manager	Die Methode getConnection() muss synchronized sein, da der ConnectionPool eventuell nicht immer thread-safe ist. Es wäre unvorteilhaft, wenn die gleiche Connection aus dem ConnectionPool vergeben wird.	done	k
18	DB-Manager	Die Passwörter werden in der Datenbank als Klartext abgespeichert. Dies ist sicherheitsrelevant sehr kritisch. Sollte irgendjemand ungewollt Zugriff auf die Datenbank bekommen, sieht er alle Passwort-Felder aller Benutzer in der Datenbank. In jedem Fall kann zumindest der Administrator des Systems in bspw. PhpMyAdmin alle Passwort-Felder einsehen. Um dies zu Verhindern wäre die Verwendung einer Crypting-Bibliothek sinvoll (JBCrypt). Diese generiert zu jedem Passwort einen Salt und anschließend wird das Passwort zusammen mit dem Salt in einem SHA1 gehashed. Dieses Vorgehen versichert, dass man selbst mit dem Datenbankeintrag des Passwort-Feldes das Passwort nicht herausfinden kann.	done	b
19	DB-Manager	Wenn ein Administrator einen Benutzer bearbeitet muss in der Methode auch geprüft werden ob der Aufrufende auch Administrator ist. Ansonsten kann jede Person mit weniger Rechten auch Benutzerdaten verändern.	done	b
20	DB-Manager	Beim Löschen von Karteikarten in der Methode loescheKarteikarte() wird zwar rekursiv gelöscht, aber die Lücke im Strukturbaum der Neo4j-Datenbank wird nicht geschlossen. Die beiden Karteikarten, das heißt die Übergeordnete und die Untergeordnete, falls sie denn existiert, müssen miteinander verbunden werden.	done	k
21	DB-Manager	Beim Schreiben von Karteikarten wird teilweise nicht überprüft, ob ein Vater existiert oder nicht. Das kann zu NullpointerExceptions führen. Eine mögliche Lösung ist es, die ID der Vaterkarteikarte auf -1 zu setzen und dies abzufragen.	done	k