Änderungen an **2.1 Entwurfsziele:**

**Robustheit, Zuverlässigkeit** → **priorisierte Kriterien**

Wartungskriterien:

Portierbarkeit → Das System soll auf Linux, Mac und Windows lauffähig sein. (zu klären: Problem der plattformspezifischen Pfadangaben [Aufgabenstellung Seite 7 oben]).

Leistungskriterien:

Antwortzeit → Es werden keine besonderen Anforderungen bzgl. der Antwortzeit gestellt, solange diese die Benutzerfreundlichkeit bzw. die Funktionalität nicht wesentlich einschränkt. Lediglich der Startvorgang der Anwendung nimmt eine gewisse Zeit in Anspruch, diese wird durch den Splashscreen überbrückt.

Zu **2.3 Verwendung existierender Softwarekomponenten**

Die Klasse *Rights*:

Die Klasse *Rights* deklariert sämtliche Rechte, die ein Userprofil haben kann. Diese können über die Methode *myRights* im korrespondierenden Userprofil abgefragt werden.

Das Interface *Hateable*:

Das Interface *Hateable* stellt Methoden zur Bewertung anderer Benutzer (außer dem Admin) sowie Medien bzw. Playlists zur Verfügung. Außerdem kann die Anzahl der positiven/ negativen Bewertungen sowie die Liste der positiven/ negativen Bewerter zurückgegeben werden.

Änderungen an **2.4 Management persistenter Daten:**

Begründung für Persistenzmechanismus: Die Verwendung einer Datenbank bietet einige für die vorliegende Anwendung entscheidende Vorteile gegenüber dem Konzept der Serialisierung.

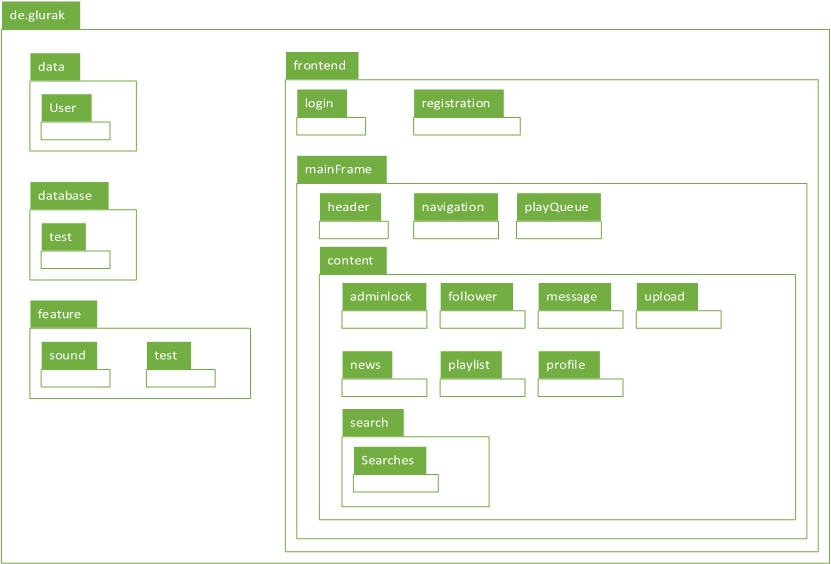
Diese sind vor allem:

1. Bessere Skalierung
2. Unterstützung komplexer Anfragen an Datenbestand

**2.5 Konfiguration/ Installation ?**

Die fertige Software wird als ZIP-Datei übergeben, die eine lauffähige JAR-Datei und die benötigten Ordnerstrukturen enthält.

Zu **2.2 Paketdiagramm**:



Zu **3.3 Dokumentation weiterer …**

1. Singleton:

Im Klassendiagramm in Figure 1 wurde das Singleton … .

3. Kompositum:

Der Ausschnitt des UML-Klassendiagramms in Figure 2 … .