Softwarepraktikum

WWU Münster
Institut für Informatik

Prof. Dr. Markus Müller-Olm
Alexander Wenner
Benedikt Nordhoff
Sebastian Kenter

WS 13/14 Aufgabenstellung 17.02.2014

Hauptaufgabe

Entwickeln Sie in Gruppenarbeit einen Prototyp einer Software für eine Social-Media-Plattform à la *Spotify* und *last.fm*.

Eine erste Beschreibung des Anwendungsgebietes finden Sie unten. Wenden Sie dabei den in der Vorlesung vorgestellten, objektorientierten Softwarekonstruktionsprozess an. Liefern Sie Ihre Zwischen- und Endergebnisse in dem in der Vorlesung bekannt gegebenen Umfang rechtzeitig zu den festgelegten Terminen bei Ihren Betreuern ab. Stellen Sie die von Ihnen entwickelte Software und ihr Design in einer Präsentation (ca. 25 Minuten) vor.

Beschreibung des Anwendungsgebietes

Das System speichert verschiedene Musikstücke oder Videos zusammen mit Informationen wie Titel, Künstler/Band und Genre.

Jeder *Benutzer* hinterlegt in dem System persönliche Angaben wie Geburtsdatum, Herkunftsland und Geschlecht. Alle Benutzer können alle Medien abrufen, zu eigenen Wiedergabelisten zusammenfügen, abspielen und auch bewerten. Sie greifen auf die Medien zu, indem sie den Bestand an veröffentlichten Medien durchsuchen oder auf Empfehlungen zurückgreifen, die ihnen vom System beispielsweise aufgrund der von ihnen bereits abgespielten Medien, ihrer Bewertungen und ihrer persönlichen Daten gegeben werden. Für alle Benutzer soll die Bewertung eines Mediums durch andere Benutzer einsehbar sein.

Benutzer können zusätzlich als *Künstler* neue Medien hochladen. Es ist auch möglich, ein neues Medium zunächst als Ankündigung zu veröffentlichen, sodass alle Benutzer bereits Meta-Informationen sehen können und sich über die Neuankündigung austauschen können, bevor später der eigentliche Inhalt hochgeladen wird.

Des Weiteren können sich auch verschiedene Labels durch eigene Profile präsentieren. Benutzer können als *Label-Manager* Aktionen für das jeweilige Label durchführen. Labels können Künstler einladen und Künstler sich andererseits bei Labels bewerben. Hat die jeweils andere Seite diese Zuordnung akzeptiert, kann der Künstler Medien veröffentlichen, die diesem Label zugeordnet werden. Label-Manager können im Namen der ihrem Label zugeordneten Künstler neue Medien veröffentlichen.

Der *Betreiber* des Systems kann Genres anlegen und bearbeiten, in die die Medien einsortiert werden können, sowie beliebige Medien und andere Benutzer sperren.

Alle Benutzer können sich untereinander auf unterschiedliche Art und Weise über die von ihnen abgespielten und ggf. angebotenen Medien austauschen.

Anforderungen

Die zu entwickelnde Plattform muss nicht unbedingt die gesamte gewünschte Funktionalität realisieren. Es sollen aber mindestens die Grundfunktionen realisiert werden. Bitte sprechen Sie den genauen Umfang Ihrer Software zu gegebener Zeit mit Ihrem Betreuer ab.

Grundfunktionen

Ihre Software muss alle der folgenden Funktionen unterstützen:

- Unterstützung mindestens eines Medientyps (Musikstück, Video...).
- Die Identifikation der Benutzer gegenüber dem System soll, wo notwendig, mit Benutzername und Passwort erfolgen.
- Registrierung neuer Benutzer.
- Bearbeiten der persönlichen Daten im eigenen Benutzerprofil, wie Geschlecht, Geburtsdatum und Herkunftsort.
- Erstellung von Label-Profilen, denen man dann als Manager zugeordnet ist.
- Von Label-Managern durchgeführte Einladung von Künstlern bzw. von Künstlern durchgeführte Bewerbung bei Labels und Akzeptieren/Ablehnen der Einladung bzw. Bewerbung.
- Anlegen/Bearbeiten/Löschen von Medien und -Ankündigungen mit zugehörigen Meta-Informationen durch einen Künstler und den Manager seines Labels.
- Sperren von Medien und Benutzern durch den Betreiber.
- Erstellen und Bearbeiten einer beliebig tiefen Hierarchie von Genres und Untergenres durch den Betreiber, in welche die Medien einsortiert werden können.
- Durchsuchen des Medienbestands nach verschiedenen Kriterien, z.B. Genre/Untergenre und Teile des Namens von Medium, Künstler und/oder Label.
- Anlegen, Benennen und Bearbeiten von Wiedergabelisten bestehend aus verfügbaren Medien.
- Zusammenstellung einzelner als Künstler hochgeladener Medien zu Alben.
- Abspielen von einzelnen Medien und ganzen Wiedergabelisten/Alben.
- Markierungen von favorisierten Medien und Künstlern.
- Anzeigen von Empfehlungen von Medien für einen Benutzer anhand favorisierter Medien und Künstler sowie bereits abgespielter und ggf. hochgeladener Medien.
- Aufrufen von Profilen anderer Benutzer, die folgende Informationen enthalten:

- Persönliche Daten,
- favorisierte Medien und Künstler,
- bereits abgespielte und ggf. hochgeladene Medien,
- eigene Wiedergabelisten und ggf. Alben,
- ggf. eigene Fans.

Zusätzliche Funktionen

Zusätzlich kann Ihre Software weitere Funktionen unterstützen, die Ihnen sinnvoll erscheinen, zum Beispiel:

- Unterstützung weiterer Medientypen.
- Komplexere Meta-Informationen, mit denen Medien organisiert werden können, z.B. Markierung mit Tags.
- Möglichkeit, mehrere Künstler oder Medien miteinander in Beziehung zu setzen, z.B. für kooperierende Künstler oder Cover-Versionen von Liedern.
- Möglichkeit des Hinzufügens von Bildern zu Benutzerprofilen oder Wiedergabelisten/Alben, etwa Benutzerfotos oder Album-Cover.
- Following-Funktion, mit der Benutzer anderen Benutzern folgen können, um Meldungen über deren neue Medien zu erhalten.
- Komplexere Statistiken für Künstler und Label-Manager basierend auf Altersgruppen oder Herkunftsländer ihrer Fans und Follower.
- Bewertung von Medien auf einer mehrstufigen Skala zur Verbesserung von Statistiken und Empfehlungen.
- Einstellungsoptionen für die Sichtbarkeit des eigenen Profils (persönliche Daten, favorisierte Medien, Wiedergabelisten...) für andere Benutzer.
- Möglichkeit, Medien zu kommentieren. Diese Funktion kann auch zu medienbezogenen Diskussionsforen ausgebaut werden.
- Senden von Nachrichten an andere Benutzer über die Plattform.
- Erweiterte Empfehlung von Medien, bei der etwa Informationen verschiedener Benutzer anhand gemeinsamer Interessen oder übereinstimmender Angaben in den Benutzerprofilen kombiniert werden. Hier ist einiger Spielraum für Kreativität vorhanden.
- Funktion, die Benutzern andere Benutzer mit ähnlichen Vorlieben vorschlägt, damit diese in Kontakt treten können.
- Verwaltung von Events, an denen Künstler mitwirken.

- Möglichkeit für Künstler, Crowd-Funding-Kampagnen zu starten.
- Einschränkung der Abspielmöglichkeiten je nach Medium und Benutzer und z.B. Freischaltung nach Bezahlung.
- Integration von Shops von Künstlern für Merchandise-Artikel oder Konzertkarten, die den Fans bzw. Followern vorgeschlagen werden.
- Verwaltung von Werbespots, die von Werbetreibenden hochgeladen werden können und dann Benutzern mit entsprechenden Vorlieben zwischen der Wiedergabe einzelner Medien angezeigt werden. Auch dies könnte man noch weiter ausbauen, beispielsweise um eine Anpassung der Anzeigehäufigkeit von Werbespots an die Höhe der Bezahlung der Künstler, oder um eine Übersicht über die Werbeeinnahmen für den Betreiber.

Außerdem können Sie gerne weitere sinnvolle Funktionen realisieren, für die Sie Ideen haben.

Allgemeine Anforderungen

Neben den funktionalen Anforderungen gibt es weitere Anforderungen an das System:

- Verwenden Sie das in der Vorlesung vorgestellte Vorgehensmodell, und beachten Sie die dort festgelegten Richtlinien für Dokumentation, Testen, Implementierung etc.
- Geben Sie die geforderten Zwischenabgaben zu den festgesetzten Terminen ab.
- Ihre Software soll in Java (mindestens Java 5) geschrieben sein.
- Ihre Software soll eine graphische Benutzeroberfläche (GUI) haben.
- Die wichtigen Daten sollen zwischen dem Beenden und einem erneuten Start Ihrer Applikation persistent gehalten werden.
- Die Anwendung soll möglichst plattformunabhängig entworfen werden. Das Kompilieren, Deployment und vor allem Ausführen der JUnit-Tests und der Anwendung selbst soll *von Beginn an* von der Kommandozeile aus funktionieren. Nachdem ein erster Prototyp Ihrer Applikation erstellt wurde, sollte eine Datei `README` im Wurzelverzeichnis Ihres `git`-Repositorys beschreiben, welche Kommandos hierfür jeweils nötig sind. Testen Sie dies auf den Linux-Rechnern des Fachbereichs. Achten Sie dabei insbesondere darauf, dass (falls Sie kein Build-Tool mit automatischer Abhängigkeitsverwaltung wie *Apache Maven* verwenden) sich alle benötigten Bibliotheken (z.B. JUnit) im Repository befinden und Sie keine von der IDE bereitgestellten Bibliotheken verwenden, falls Sie Ihr Build-Skript mit Hilfe der IDE automatisch generieren.

  Beispielsweise könnten Sie ein *Apache Ant*-Skript mit den folgenden Targets erstellen:

  1. *compile*: Kompiliert die Anwendung und Tests (compile-ANT-Task).
  2. *test*: Führt alle Tests aus und hinterlegt die Ergebnisse im XML-Format (JUnit-ANT-Task).

3. *run*: Startet eine Instanz der Anwendung, die schon mit geeigneten Daten befüllt ist, um eine Begutachtung aller Funktionen zu vereinfachen.
4. *deploy*: Erstellt einen *dist*-Ordner, in dem eine frische Installation der Anwendung (*JAR*-Datei (jar-ANT-Task)), inklusive aller benötigten Bibliotheken und Konfigurationsdateien, installiert wird (copy-ANT-Task).

Sie können sich auch bezüglich dieser Richtlinien an der Hibernate-Beispielanwendung orientieren, die im Material-Ordner im BSCW verfügbar ist.

Einschränkungen

Im Rahmen eines vierwöchigen Softwarepraktikums kann man keine vollständig ausgereifte Software mit "allen Schikanen" entwickeln. Folgende Einschränkungen, die den Entwicklungsaufwand verringern, sollten für den zu entwickelnden Prototyp der Software gemacht werden. Falls Sie eine dieser Einschränkungen nicht machen wollen, halten Sie vorher bitte Rücksprache mit Ihren Betreuern.

- Die GUI soll ausschließlich mit Java-Swing entworfen werden. Die GUI soll dabei von Hand, d.h. ohne die Verwendung eines GUI-Editors, erstellt werden.
- Für die Realisierung der Persistenz empfehlen wir eine Datenbank und die Verwendung von Hibernate. Alternativ können Sie den Serialisierungsmechanismus von Java oder die XStream-Bibliothek verwenden.
- Die Anwendung soll als Einzelplatzanwendung entworfen werden, d.h. der Zugriff erfolgt von einem einzelnen Terminal. Die von der Software verwalteten Mediendateien können im lokalen Dateisystem gespeichert werden.

Links mit Hinweisen zu den genannten Technologien finden Sie auf der Webseite des Softwarepraktikums (*http://cs.uni-muenster.de/sev/teaching/ws1314/sopra/*).

Warm-Up-Aufgabe

Diese Aufgabe dient zum Aufwärmen der Java-Kenntnisse. Sie soll von jedem Teilnehmer eigenständig bearbeitet werden.

Implementieren Sie ein Adressbuch als Java-Anwendung. Die Anwendung soll Informationen zu Personen verwalten. Ein Eintrag besteht aus Vor- und Nachname, E-Mail-Adresse, Telefonnummer, Straße, Hausnummer, PLZ, Ort und Land einer Person. Felder können dabei ggf. leer gelassen werden. Die GUI soll alle Einträge in einer Liste (Tabelle) darstellen, und das Hinzufügen, Entfernen und Ändern von Einträgen erlauben. Die Einträge sollen beim Beenden der Anwendung natürlich nicht verloren gehen. Benutzen Sie einen Persistenzmechanismus, um die Daten zu speichern. Dabei genügt es, wenn die Daten beim Starten der Anwendung aus einer festen Datei geladen und beim Beenden der Anwendung in diese Datei zurückgeschrieben werden. Die GUI könnte zum Beispiel wie in Abbildung 1 dargestellt aussehen.

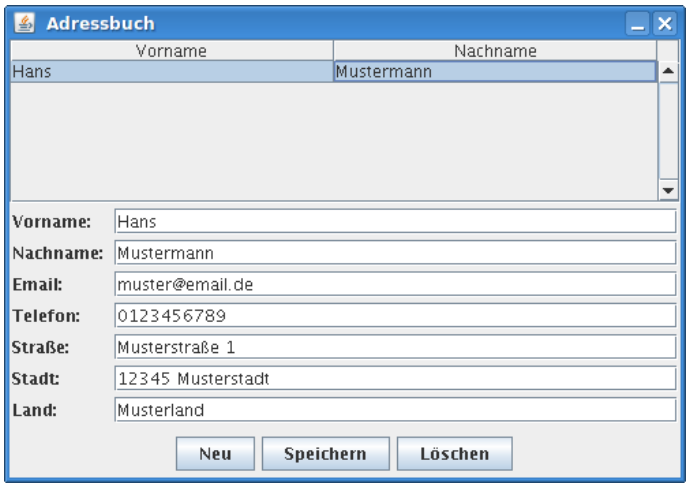

Abbildung 1: Beispiel für Lösung der Warm-Up-Aufgabe

Hinweise

Die GUI soll ohne die Verwendung eines GUI-Editors erstellt werden. Wenn Sie nicht genau wissen, wie Sie bei der Realisierung der GUI vorgehen sollen, beachten Sie die folgenden Anregungen:

- Leiten Sie die Klasse für Ihr Fenster von *JFrame* ab.
- Verwenden Sie *BorderLayout* als Layout-Manager für die *ContentPane* des Fensters.
- Ordnen Sie die Knöpfe in einem *JPanel* an, das Sie mit dem *BorderLayout.SOUTH*-Constraint zur *ContentPane* hinzufügen. Dieses *JPanel* verwendet am besten ein *FlowLayout*.
- Erzeugen Sie auf ähnliche Art und Weise ein *JPanel* mit geeignetem Layout für die Eingabefelder zum Ändern der Daten in der Mitte der *ContentPane* (*BorderLayout.CENTER*).
- Platzieren Sie die Tabelle in einer *JScrollPane* im oberen Bereich der *ContentPane* (*BorderLayout.NORTH*).
- Verwenden Sie *DefaultTableModel* als Basis für das Datenmodell Ihrer Tabelle.
- Um das Speichern beim Beenden der Anwendung zu realisieren, müssen Sie einen *WindowListener* implementieren. Über dessen *WindowClosed* bzw. *WindowClosing* Methoden bekommen Sie mit, wenn das Fenster geschlossen wird. Achten Sie darauf, dass Sie wirklich nur die Adressdaten serialisieren und nicht (ungewollt) auch Teile der Komponentenstruktur der Anwendung. Dies kann passieren, wenn Sie das Datenmodell der Tabelle serialisieren, und die Serialisierung rekursiv auch alle registrierten Listener und alles, was daran hängt, mitserialisiert.
- Vergessen Sie nicht, dem Fenster einen sinnvollen Titel und eine sinnvolle Anfangsgröße zu geben.

- Die Warm-Up-Aufgabe soll plattformunabhängig gelöst werden, d.h. das Programm soll mindestens auf Linux, Mac und Windows lauffähig sein. Das häufigste Problem in diesem Zusammenhang ist die Verwendung von plattformspezifischen Pfadangaben, z.B. *C:\TEMP\DATA.SER*. Eine portable Entsprechung würde stattdessen z.B. auf die System-Property *java.io.tmpdir* Bezug nehmen oder relativ zum Ausführungsort *./* Dateien ablegen.

Material

Das folgende Material kann bei der Bearbeitung der Warm-Up-Aufgabe hilfreich sein:

- Ein gutes Buch zur Java-Programmierung, z.B.:
  - *Handbuch der Java-Programmierung. Guido Krüger, Thomas Stark.* Eine HTML-Version kann unter *http://javabuch.de/download.html* heruntergeladen werden. Teil 6 (Kap. 36-39) beschäftigt sich mit Swing.
  - *Java ist auch eine Insel. Ullenboom.* Eine Online-Version ist verfügbar unter: *http://openbook.galileocomputing.de/javainsel/*. Kapitel 14 beschäftigt sich mit der GUI-Programmierung mit Swing.
- Java-Swing-Tutorial: *http://docs.oracle.com/javase/tutorial/uiswing*
  Insbesondere die Abschnitte: *Using Swing Components*, *Laying Out Components Within a Container* und *Writing Event Listeners*.
- Java-API Spezifikation: *http://docs.oracle.com/javase/7/docs/api/*

Zusätzliche Links finden Sie auf der Webseite des Softwarepraktikums.

Zeitplan

Mo, 24.02.2014, 17:00 Uhr	Abgabe des Analysedokuments (Pflichtenheft)
Di, 25.02.2014	Präsentation und Besprechung des Analysedokuments
Di, 25.02.2014, 17:00 Uhr	Abgabe der Warm-Up-Aufgabe
Mi, 05.03.2014, 17:00 Uhr	Erste Abgabe des Entwurfsdokuments und der Implementierung
Fr, 07.03.2014, ab 9:00 Uhr	Feedback-Runde durch die Betreuer
Di, 11.03.2014, 17:00 Uhr	Abgabe des überarbeiteten Entwurfsdokuments und der Implementierung, zudem Abgabe des Handbuchs
Mi, 12.03.2014	Vorbereitung der Abschlusspräsentation, Prüfungsgespräche und Einzelvorführungen in den Gruppen
Do, 13.03.2014	Abschlusspräsentation und Besuch der Abschlusspräsentationen anderer Gruppen
nach Absprache	Finale Feedback-Runde durch die Betreuer

Weitere Informationen zur Zeitplanung finden Sie auf den Vorlesungsfolien. Die genaue Zeitplanung der Einzelvorführungen und Abschlusspräsentationen wird rechtzeitig auf der Webseite des Softwarepraktikums bekannt gegeben.