SEP – Wintersemester 2010/2011

Praktomat - light

Lastenheft

C. Bachmaier, G. Hölbling, M. Matzeder, A. Paler

1 Zielbestimmung

Es soll ein webbasiertes System zur Qualitätskontrolle im Programmierpraktikum entwickelt werden. Das zu entwickelnde System *Praktomat - light* soll zum Bewerten von Programmen bzw. deren Sourcecode verwendet werden können. Dabei soll es möglich sein, verschiedene Programmier-Aufgaben zu stellen, welche dann durch Studenten gelöst und über die Weboberfläche im System eingereicht werden können. Desweiteren soll das System eine Bewertung dieser Einreichungen anbieten. Ein Muss für so ein System ist, dass es robust ist und zuverlässig arbeitet. Nachstehend finden sie ein ihnen bekanntes Beispiel für ein solches System:

- Praktomat `https://praktomat.fim.uni-passau.de/praktomat/`

Nicht angemeldete Benutzer können nur allgemeine Informationen über das System wie Kurzbeschreibung, Versionsnummer, Registrierungsformular, eine Liste aller laufenden Lehrveranstaltungen etc. sehen, jedoch keine Informationen zu den Aufgabenstellungen. Benutzer des Systems sollen nach der Registrierung und dem anschließenden Anmelden die Möglichkeit haben ihre persönlichen Daten zu ändern und anzupassen. Desweiteren haben sie die Möglichkeit Lösungen einzureichen. Im System sind zwei wesentliche Rollen zu unterscheiden, der Student und der Testierer:

- Registrierter Benutzer: Unter registrierten Benutzern werden primär Studenten verstanden die sich zu einer angebotenen Veranstaltung anmelden.
- Testierer: Der Testierer betreut und administriert die Lehrveranstaltung. Neben dem Anlegen der Veranstaltung, von Aufgaben und Checks kommt ihm auch die Aufgabe des Testierens und Bewertens der einreichten Lösungen zu.

Das Einsehen dieser Lösungen obliegt einzig und allein dem Einreichenden selbst und dem Testierer.

2 Produkteinsatz

Zielgruppe sind Studenten die begleitend zu einer Vorlesung über Programmierung selbständig Übungsaufgaben lösen sollen bzw. müssen.

3 Grundprinzip

Das System soll für die Verwaltung von mehreren Veranstaltungen konzipiert sein wobei die Datenhaltung (Datenbank) zwischen den einzelnen Veranstaltungen getrennt vorgenommen werden soll. Eine Veranstaltung (z.B. Prog 2) gliedert sich jeweils in mehrere Aufgabenstellungen die von, zur Veranstaltung registrierten Benutzern bearbeitet werden können.

Testierer pflegen das System, indem sie Aufgaben erstellen. Dazu müssen Datum und Zeit angegeben werden, ab wann eine Aufgabe für registrierte Benutzer veröffentlicht und damit sichtbar ist und bearbeitet werden kann. Desweiteren muss festgelegt werden, wann eine Aufgabe *empfohlen* und *endgültig* eingereicht werden muss. Hierbei wird unter *empfohlen* der im Ablauf der Veranstaltung als sinnvoll angesehene Abgabetermin und als *endgültig* der letztmögliche Termin festgelegt.

Sobald eine Aufgabenstellung für registrierte Benutzer sichtbar wird (siehe 3.1), können Lösungen eingereicht werden.

Registrierte Benutzer können ihre Lösungen bis zum Zeitpunkt der für die endgültige Abgabe festgelegt wurde einreichen. Nach diesem Zeitpunkt stehen die Lösungen dem Testierer zum Korrigieren und zum Testieren zur Verfügung. Eine wichtige Nebenbedingung stellt die Anonymität (persönliche Daten werden nicht angezeigt) der registrierten Benutzer dar, die über den gesamten Korrekturzyklus der Aufgabe bis zur endgültigen Bewertung gewahrt bleiben muss.

3.1 Das Anlegen von Aufgabenstellungen

Eine Aufgabenstellung wird in HTML (evtl. mit Bildern) eingegeben, wobei die einzelnen Fristen (Veröffentlichung, empfohlene Abgabe, endgültige Abgabe) ebenfalls erfasst werden müssen. Jede Aufgabenstellung hat eine Menge von *Checkern*, die beim Einreichen einer Lösung automatisch durchgeführt werden jedoch auch manuell gestartet werden können (siehe 3.3).

Zu jeder Aufgabenstellung muss vom Testierer angegeben werden können, welche Checker in welcher Reihenfolge auf die eingereichten Lösungen zu dieser Aufgabenstellung durchgeführt werden sollen. Für die einzelnen Checker sollte es in Abhängigkeit ihres Typs (siehe 3.3) auch für den registrierten Benutzer die Möglichkeit geben sich das Bestehen bzw. Nicht-bestehen eines Checks anzeigen zu lassen.

Das Anlegen von Aufgabenstellungen kann nur durch den Testierer (Administrator) vorgenommen werden, da diese Tätigkeit entsprechende Verwaltungsrechte im System benötigt.

3.2 Das Einreichen

Jeder Student muss sich registrieren, um das System nutzen zu können. Nach der Registrierung hat er die Möglichkeit sich am System anzumelden und für bereits sichtbare Aufgabenstellungen Lösungen einzureichen. Nach einem Einreichungsversuch meldet das System, welche (öffentlichen und erweitert) Checks erfolgreich (Grün bzw. Ërfolgreich") bzw. welche nicht bestanden wurden (Rot bzw. Nicht Bestanden"). Um eine Einreichung erfolgreich abzuschließen ist das Bestehen aller Checks die als *Voraussetzung zum Bestehen* markiert sind erforderlich. Erst nach dem erfolgreichen Einreichen wird die Lösung im System als solche gespeichert.

3.3 Die Checker

Checker sind Verfahren, die überprüfen, ob ein Ereignis eintritt, oder fehlschlägt. Es können verschiedene Checker beim Einreichen einer Lösung aufgerufen werden, um z.B. zu überprüfen, ob eine Datei hochgeladen werden konnte. Es soll ein Checker vorhanden sein, der den Code der eingereichten Lösung auf die maximale Zeilenbreite überprüft. Um eine Beeinflussung der Testierer zu vermeiden, soll überprüft werden, ob sich Hinweise auf persönliche Daten im Code befinden. Ebenfalls muss überprüft werden können, ob eine eingereichte Lösung kompiliert oder nicht. Weitere denkbare Checker (z.B. Überprüfung der Ausgaben eines Testfalls) sollen einfach und flexibel in das System integrierbar sein. Als Optionen für einen Check sind zumindest folgende Varianten vorzusehen:

- Voraussetzung zum Bestehen: Bei dieser Option kann bestimmt werden ob das bestehen des Checks Voraussetzung für ein erfolgreiches Einreichen ist.
- Ergebnis ersichtlich: Bei dieser Option kann bestimmt werden ob die Ergebnisse eines Checks dem einreichenden Benutzer mitgeteilt werden.

Checker 3.3 sollen auch manuell vom Testierer gestartet werden können.

3.4 Das Testieren

Wenn die Frist für die Einreichung abgelaufen ist (der Zeitpunkt der endgültigen Abgabe wurde überschritten) steht die jeweils letzte gültige Einreichung jedes Studenten zum Testieren und Bewerten zur Verfügung. Den Testierern werden zu korrigierende

Einreichungen jeweils zufällig zugewiesen. Um die Arbeit zu erleichtern soll es aber auch möglich sein ausgewählte (oder alle) Einreichungen als Archivdatei aus dem System herunter zu laden. Die Ergebnisse der bei der Einreichung durchgeführten Checks sollen dem Testierer übersichtlich dargestellt werden. Um eine Korrektur zu ermöglichen soll der Testierer im Webinterface folgende Möglichkeiten haben: - Quellcode der Einreichung korrigieren - Kommentare zum Quellcode hinzufügen Neben der kommentierten / korrigierten Fassung der Einreichung ist natürlich auch die Originalfassung im System zu erhalten. Zusätzlich soll der Testierer die Möglichkeit haben Kommentare zum Testat abzugeben. Hierbei können sowohl interne (nur für den Testierer ersichtlich) als auch für den einreichenden Nutzer ersichtliche Kommentare angegeben werden.

Die Bewertung einer Einreichung soll jeweils in ganzen Noten von 1 bis 5 für die Kriterien *Verständlichkeit* und *Funktionalität*, wobei sich Verständlichkeit in *Layout* und *Struktur* unterteilt, erfolgen. Dabei bleibt, bis zum endgültigen Abschluss der Bewertung, der einreichende Benutzer gegenüber dem Testierer anonym. Das Testat und die Bewertung bleiben bis zum endgültigen Freigabe durch den Testierer geheim gehalten und erst nach der Freigabe dem einreichenden Benutzer per Email mitgeteilt. Der Testierer hat die Möglichkeit in einer übersichtlichen Darstellung (Bewertungsstatistik) alle bisherigen Testate zur Veranstaltung einzusehen. Dem registrierten Benutzer steht auch eine Bewertungsstatistik für seine eigenen Lösungen zur Verfügung.

4 Produktfunktionen

4.1 Funktionen für nicht registrierte Benutzer

- Download eines Disclaimers zur Datenspeicherung
- Zugriff auf die Hilfe-Seiten und Startseite
- Anzeigen einer Liste aller verfügbaren Veranstaltungen
- Jeder Benutzer kann sich zu einer Veranstaltung registrieren. Dabei muss seine E-Mail-Adresse verifiziert werden (z. B. durch Zusendung eines Aktivierungscodes)

4.2 Funktionen für registrierte Benutzer

- Bearbeiten des eigenen Benutzerkontos
- Aufgabenstellungen nach Veröffentlichung einsehen
- Lösungen nach Veröffentlichung und vor endgültiger Abgabe einreichen
- Anzeige der (sichtbaren) Ergebnisse der Checks nach dem Einreichen
- Testate nach Freigabe einsehen
- Bewertungsstatistik für eigenen Lösungen einsehen

4.2.1 Anmeldung

- Nach der Anmeldung sieht jeder Nutzer eine personalisierte Einstiegsseite
- Ein registrierter Benutzer kann seine Daten editieren.

4.2.2 Benachrichtigung

Sämtliche Kommunikation von *Praktomat - light* mit seinen Benutzern erfolgt per E-Mail:

- Nach der Registrierung werden die Nutzerdaten bestätigt
- Benachrichtigung über Änderungen der Benutzerdaten
- Benachrichtigung bei erfolgreichem Einreichen einer Lösung. Hierbei wird der abgegebene Quellcode und die Ergebnisse der Checks der Email angefügt.
- Mitteilung über verfügbare Testate
- Mitteilungen von Administratoren

4.3 Funktionen für Testierer

- Erstellen, Bearbeiten, Löschen von *Aufgabenstellungen*
  - Jede Aufgabenstellung hat eine Beschreibung die in HTML inkl. Bilder eingegeben werden kann
  - Bei jeder Aufgabenstellung gibt es Zeitvorgaben: *Veröffentlichung*, *empfohlene Abgabe*, *verbindliche Abgabe*
  - Zu jeder Aufgabe können beliebig viele *Checker* hinzugefügt werden
  - An Checker müssen zumindest folgende angeboten werden:
    - Speichern von hochgeladenen Dateien
    - Prüfung der maximalen Zeilenbreite
    - Anonymitätsprüfung (ist eine persönliche Information eines registrierten Benutzers im Dokument enthalten)
    - Kompilieren
  - Die Reihenfolge der Testausführung kann soweit sinnvoll variiert werden
  - Angabe bei jedem Test
    - Wann er ausgeführt wird (schon bei der Abgabe oder erst manuell durch einen Testierer)
    - Ob er zum erfolgreichen Einreichen bestanden werden muss
    - Ob der registrierte Benutzer das Ergebnis erfährt
- (Nochmaliges) Ausführen aller der Aufgabe zugeordneten Checker

- Anzeigen von Notenübersicht und Bewertungsstatistik
- Anzeigen, Erstellen, Speichern, Bearbeiten, Versenden und Löschen von Testaten
  - Nach verbindlicher Abgabe
  - Abgabe wird dem Testierer zugelost (anonymes Testieren)
  - Anzeige der Ergebnisse der Checks
  - Checks können einzeln oder komplett (nochmal) ausgeführt werden
  - Download des Quellcodes als eine Archivdatei oder in separaten Dateien
  - Einfügen von Kommentaren in den abgegebenen Quellcode (Kennzeichnung zur Unterscheidung mit Quellcode notwendig)
  - Benotung durch zwei ganzen Noten 1 – 5 Funktionalität und Verständlichkeit, wobei sich Verständlichkeit aus Layout und Struktur zusammensetzt
  - Freikommentarfelder intern und für einreichenden Benutzer
  - Jeder Testierer sieht nur seine Testate
  - Testate werden auf explizite Anforderung *gemeinsam freigegeben*

4.4 Verwaltungsfunktionen

- Konfiguration des Erscheinungsbildes des *Praktomat - light* (Logo, Icons, Beschriftungen, . . .)
- Anlegen, Bearbeiten, Löschen von *Veranstaltungen*
  - Jede Veranstaltung wird in einer separaten Datenbank **unabhängig** von anderen gespeichert
  - Es gibt Fristen für die Anmeldung und den Zugang zu jeder Veranstaltung
  - Download aller bisher zu einer Aufgabenstellung eingereichten Lösungen (nur neuste Versionen) in einer Archivdatei
  - Download aller (bisherigen) Abgaben (alle Abgaben) einer Veranstaltung in einer Archivdatei
- Anlegen, Sperren, Löschen und Bearbeiten von Benutzerkonten (auch *Testierer* Konten)
  - Benutzerkonten werden pro Veranstaltung separat geführt
- Nachrichten an alle Teilnehmer einer Veranstaltung versenden

4.5 Allgemein

4.5.1 Anzeige

- Einfache Navigation auf der Seite

- Impressum bzw. Kontaktinformationen
- Allgemeinen Informationen zum System

4.5.2 Implementierung

- Einfaches und standardisiertes Nachrüsten von weiteren Checkern
- Jeder Checker muss in eigenem Thread ausgeführt werden, da er potentiell lange dauern kann
  - Benutzer wird über Fortschritt kontinuierlich informiert
  - Browsersession muss für den Ausführungszeitraum der Checks erhalten bleiben

5 Produktleistungen und Qualitätsanforderungen

5.1 Allgemein

Für die persistente Speicherung der Daten soll eine Datenbank (PostgreSQL) verwendet werden, die auf den Infosun-Rechnern zur Verfügung gestellt wird. Als Referenzplattform dienen die Rechner im CIP-Pool.

Besondere Beachtung soll der Einfachheit der Bedienung des Systems geschenkt werden. Dabei sollen nicht nur Daten leicht auffindbar und gut lesbar sein, sondern auch leicht einzugeben (Stichwort: *Benutzerfreundlichkeit*).

Bei Fehleingaben in ein HTML-Formular und der darauffolgenden Korrektur sollen die zuvor eingetragenen Felder nicht erneut einzugeben sein, sondern schon vorbesetzt sein. Außerdem sollte die Überprüfung der Eingaben nicht nach der ersten fehlerhaften Eingabe abbrechen, sondern alle Eingaben überprüfen und eine akkumulierte Fehlermeldung an den Benutzer zurückgeben. Alle Daten sind sensibel, dürfen also nur über eine SSL-Verbindung verschickt werden. Das System sollte mit steigender Last skalieren.

6 Ergänzungen

Die Benutzung des Systems sollte mit allen gängigen WWW-Browsern möglich sein. Wir raten allerdings davon ab, für jeden Browsertyp unterschiedlichen HTML-Code zu generieren. Der HTML-Code soll logisches Markup darstellen und nicht dazu missbraucht werden, eine bestimmte graphische Darstellung zu erzwingen. Deswegen dürfen Features wie Frames gar nicht und Javascript nur nach Absprache eingesetzt werden. Der HTML-Code muss HTML-konform sein und z.B. durch `http://validator.w3.org/` validierbar sein. Die Verwendung von Cascading Stylesheets (CSS) wird dringend angeraten.

Die Sessionverwaltung darf die Verwendung von Cookies nicht erzwingen.

Die Servlets müssen ein Log über alle Fehler führen, um das Debugging der Anwendung zu vereinfachen. Achten Sie darauf, dass die Fehlerbeschreibungen detailliert genug sind, um auf einen Fehler bzw. dessen Ursache schließen zu können.