Library Kata "Benutzeranmeldung"

Entwickeln Sie eine Bibliothek, mit der Benutzeranmeldungen bei Websites verwaltet werden können.

# Anforderungen

Neue Benutzer registrieren sich zunächst. Sie müssen mindestens ihre Email-Adresse angeben. Wenn in Passwort gewählt wurde, generiert die Registrierung eines, das der Benutzer später ändern kann. Sollte dieselbe E-Mail bereits registriert sein, wird die erneute Regstrierung abgelehnt

Wer sich registriert, bekommt eine Registrierungsemail geschickt mit einem Link zur Bestätigung, in dem eine Registrierungsnummer enthalten ist. Erst wenn die Registrierung mit dieser Nummer bestätigt ist, ist der Benutzer permanent im System.

Unbestätigte Registrierungen sollten ein Verfallsdatum bekommen und danach automatisch gelöscht werden. Ob eine Registrierung noch gültig ist (noch nicht abgelaufen) muss überprüft werden können.

Die Anmeldung erfolgt mit Email-Adresse oder Nickname und Passwort nach Bestätigung. Ist sie erfolgreich, liefert sie ein Token zurück. Das kann später immer wieder bei Bedarf zur Prüfung vorgelegt werden, ob Anfragen von einem Client gültig sind. Zu diesem Zweck sollte das Token so beschaffen sein, dass Clients sie nicht fälschen können. Für den Rest der Welt ist das Token opaque. Der Token ist eine bestimmet Zeit gültig, danach ist eine neue Anmeldung notwendig.

Wer sein Passwort vergessen hat, kann eine Zurücksetzung beantragen. Es wird dann eine Nachricht an die E-Mail-Adresse gesendet, in der ein Link steht, über den man ein neues Passwort anfordern kann. Wird der angeklickt, wird ein Passwort generiert und an die beantragende Email-Adresse versandt. Der Link mit der Reset-ID ist nur für eine gewisse Zeit gültig. Die Gültigkeit muss überprüft werden können.

Benutzer können ihre Anmeldedaten verändern. Sie greifen darauf über ihr Token zu, den sie bei der Anmeldung bekommen. Anschliessend werden die Benutzerdaten über eine interne Id identifiziert; Benutzer sind mithin Entitäten, deren Daten bis auf die Id veränderbar sind. Neben den bereits erwähnten Daten, sollen auch noch der letzte Login, die letzte Aktualisierung und das Registrierungsdatum des Benutzers in der Entität vorhanden sein.

## Konfigurierbarkeit

* Die Daten für das Versenden der E-Mails (SMTP-Server, Kontodaten, Absenderadresse etc.) sollen frei mittels Konfigurationsdatei gewählt werden können.
* Die Gültigkeitsdauer eines Tokens soll ebenfalls frei gewählt werden können und in einer Konfigurationsdatei abgelegt werden.
* Die Gültigkeitsdauer einer Registrierung soll ebenfalls frei gewählt werden können und in einer Konfigurationsdatei abgelegt werden.
* Alle Konfigurationen sollen in einer Datei abgelegt werden. Siehe dazu <https://docs.oracle.com/javase/tutorial/essential/environment/properties.html>

# UML Sequenzdiagramm

## Use-Case "Registrierung"



## Use-Case "Anmeldung"



## Use-Case "Passwort zurücksetzen"



# Schnittstellen

Entnehmen Sie die notwendigen zu erstellenden Klassen (Eigenschaften und Methoden/Funktionen) aus der obigen Beschreibung und aus den entsprechenden UML-Sequenzdiagrammen. Die zur Bibliothek gehörenden Klassen sind in Schweinchen Rosa eingefärbt. Zusätzliche (Hilfs-) Klassen sind möglich, müssen jedoch ebenfalls einzeln getestet werden.

## Testing

Alle Klassen der Schnittstelle / Bibliothek sollen einzeln und automatisiert getestet werden. Dazu wird JUnit eingesetzt. Um das Versenden von E-Mail zu testen, kann auf Test-SMTP-Server zurückgegriffen werden: [GreenMail](http://www.icegreen.com/greenmail/) (Empfehlung, aktuell und einfach), [Dumbster](http://quintanasoft.com/dumbster/) oder [Wiser](https://github.com/voodoodyne/subethasmtp/blob/master/Wiser.md) oder bieten mögliche Lösungen dazu.

### Maven Dependencies

#### GreenMail

<dependency>

<groupId>com.icegreen</groupId>

<artifactId>greenmail</artifactId>

<version>1.5.1</version>

</dependency>

#### Dumbster

<dependency>

<groupId>dumbster</groupId>

<artifactId>dumbster</artifactId>

<version>1.6</version>

</dependency>

#### Wiser

<dependency>

<groupId>org.subethamail</groupId>

<artifactId>subethasmtp</artifactId>

<version>3.1.7</version>

</dependency>

# Nicht funktionale Anforderungen

## Team

Ein Zweier-Team entwickelt eine eigenständige Lösung.

## Tools & Technologien

### Programmiersprache

Die Lösung muss mittels Java (JDK 1.8 oder höher) entwickelt werden. Es müssen die Sun/Oracle Code Conventions eingehalten werden. (siehe <http://www.oracle.com/technetwork/java/javase/documentation/codeconvtoc-136057.html>, Ausnahmen, die max. Zeilenlänge ist 160 Zeichen)

### Versionskontrolle

Damit das Team zusammen arbeiten kann wird eine Versionskontrolle benötigt. Dazu wird **SubVersion** eingesetzt. Als Provider wird [SourceForge](https://sourceforge.net/) verwendet. Der Verlauf der Arbeiten (commits) sollen ersichtlich sein.

### Build- und Konfigurationsmanagement

Als Konfigurationsmanagement wird **Maven** eingesetzt. Alle Maven Files sollen in der Versionsverwaltung abgelegt werden (jedoch keine Settings der verwendeten Entwicklungsumgebung)

Als Dokumentation (Minimum) sollen im pom.xml die entsprechenden Entwickler aufgeführt werden. Ebenso soll die URL des Projekts in SourceForge im POM eingetragen werden. Es muss sichergestellt werden, dass Anhand der URL sämtliche Daten des Projekts (Source-Code, commits etc.) ersichtlich sind. Dazu muss dem SourceForge-User 'randeres' Leserechte (Gruppe 'Member') auf das Projekt erteilt werden. Andere (nicht-) anonyme User sollen das Projekt nicht einsehen können.

# Abgabe

Es soll im Moodle ein ZIP-File abgegeben werden in welchem das Maven-Projekt abgelegt ist.

# Bewertung & Benotung

Die Lösung wird bewertet und benotet. Damit eine faire Benotung stattfinden kann muss das Team dem Dozenten mitteilen in welchem Verhältnis jeder einzelne zur Lösung beigetragen hat. Beispielsweise hat jeder zu gleichen Teilen zur Lösung beigetragen: 100:100. Hat der das eine Teammitglied mehr dazu beigetragen, kann das Verhältnis z.B. 110:90 sein. Die Note wird entsprechend angepasst: Erhält die Arbeit eine Note 5 so erhält der eine Student die Note 5.5 und der Andere 4.5 (bei einem angenommenen Verhältnis von 110:90)