DHBW
Duale Hochschule
Baden-Württemberg

DOKUMENTATION

JAVA EE-Projekt “Ticket Master”

Peter Fischer, Philipp Steigler, Jarno Wagner,
Roman Würtemberger

26.06.2019

DHBW Mosbach, INF16A/B

6. Semester

INHALTSVERZEICHNIS

PROJEKTPLAN MIT LEISTUNGSÜBERSICHT

Aufgabe	Bearbeiter	Fälligkeit
Aufsetzen des Projekts / Konfiguration	Alle	07.04.2019
Datenbankmodellierung	Jarno Wagner, Philipp Steigler	21.04.2019
Implementierung Frontend	Jarno Wagner, Philipp Steigler, Roman Württemberger	11.06. / 26.06.2019
Implementierung Backend	Jarno Wagner, Philipp Steigler	11.06. / 26.06.2019
Testing	Alle	Fortlaufend
Basic-Funktionalitäten	Alle	11.06.2019
Präsentation erstellen	Peter Fischer	11.06.2019
Erweiterungen (nice-to-haves)	Alle	26.06.2019
Dokumentation erstellen	Peter Fischer	26.06.2019

EINRICHTUNG UND KONFIGURATION

Die Anleitung für die Konfiguration befindet sich ebenfalls in der Datei `README.md` des Source-Codes – dort lassen sich die notwendigen Befehle besser zur Eingabe kopieren.

Datenbank-Konfiguration

Um dem Ticket-System den Zugriff auf die Datenbank zu erlauben, muss in MySQL ein neues Datenbankschema und ein passender Benutzer eingerichtet werden, welcher auf das Schema zugreift.

Hierfür sind folgende Schritte nötig:

1. MySQL Datenbank starten und mittels MySQL-Workbench auf diese verbinden
2. Neues Schema mit der Bezeichnung `tickets` einrichten
3. Benutzer mit folgenden Daten einrichten: Name = `inf16`, Passwort = `61fni@19`

Dabei ist es wichtig zu erwähnen, dass der Benutzer sämtliche Rechte für den Zugriff haben muss!

Wildfly-Konfiguration

Damit die Software bei der Ausführung des Wildfly-Application-Servers auf die Datenbank zugreifen kann, ist es erforderlich, anschließend einige Konfigurationen vorzunehmen:

1. Wildfly-Server temporär starten: `%WILDFLY-HOME%/bin/standalone.sh` *oder* `.bat`
2. JBOSS-CLI separat starten: `%WILDFLY-HOME%/bin/jboss-cli.sh` *oder* `.bat`
3. In der JBOSS-CLI über den Befehl `connect` auf den Wildfly-Server verbinden
4. Folgende drei Befehle hintereinander eingeben:

MySQL-Driver

```
/subsystem=datasources/jdbc-driver=mysql/:add( \
  driver-module-name=com.mysql, \
  driver-name=mysql, \
  driver-class-name=com.mysql.jdbc.Driver, \
  xa-datasource-class=com.mysql.jdbc.jdbc2.optional.MysqlXADataSource)
```

Datasource

```
/subsystem=datasources/data-source=ticketsDS/:add( \
  jndi-name=java:jboss/datasources/ticketsDS, \
  connection-url=jdbc:mysql://localhost:3306/tickets, \
  driver-name=mysql, \
  driver-class=com.mysql.jdbc.Driver, \
  enabled=true, \
  jta=true, \
  use-java-context=true, \
  user-name="inf16", \
  password="61fni@19", \
)
```

Security Subsystem

```
./subsystem=security/security-domain=ticketsDomain/:add
./subsystem=security/security-domain=ticketsDomain/authentication=classic:add( \
  login-modules=[{code=Database, flag=Required, module-options={ \
    dsJndiName="java:jboss/datasources/ticketsDS", \
    principalsQuery="SELECT password from User WHERE login_id=?", \
    rolesQuery="SELECT r.name, 'Roles' FROM Role r, User_Role ur, User u WHERE
r.id = ur.roles_id AND u.id = ur.User_id AND u.login_id = ?", \
    hashAlgorithm=SHA-256, \
    hashEncoding=base64, \
      hashUserPassword=true, \
      hashStorePassword=false, \
}}])
```

ANWENDERDOKUMENTATION

Das Ticketsystem soll Kunden ermöglichen, Tickets zu erstellen und den Bearbeitungsverlauf der eigenen und unternehmensweiten Tickets einzusehen. Bearbeiter sollen Tickets bearbeiten können, indem sie - ähnlich zu einem Blog - Kommentare verfassen können.

Login

Um das Ticket-System nutzen zu können ist es zunächst erforderlich, sich mit validen Nutzerdaten einzuloggen. Hierfür stehen eine Reihe von Testusern bereit, die jeweils unterschiedliche Rollen zur Benutzung sämtlicher Features verkörpern:

Login-ID	Passwort	Rolle(n)
root	toor	Admin, Bearbeiter, Kunde
admin	admin	Admin
editor1	mosbach	Bearbeiter
editor2	mosbach	Bearbeiter
editor3	mosbach	Bearbeiter
editor4	mosbach	Bearbeiter
customer1	mosbach	Kunde
customer2	mosbach	Kunde
customer3	mosbach	Kunde
customer4	mosbach	Kunde

Der Root-User ist ein besonderer User, der alle Seiten der Webanwendung sehen kann. Er ist vor allem zum testen der Anwendung gedacht und nicht für den Einsatz in einer Produktivumgebung.

Kunde

Ticket erstellen:
Um ein Ticket zu erstellen geht man über den Reiter “Neues Ticket”. Dort müssen ein Betreff und eine möglichst genaue Beschreibung des Problems hinterlegt werden. Jedes

neu erstellte Ticket wird unter dem Reiter “Meine Liste” hinzugefügt.

Ticketverlauf einsehen:
Über die Schaltfläche “Details” auf den Seiten “Meine Liste” oder “Mein Unternehmen” lassen sich selbst oder aus dem selben Unternehmen eröffnete Tickets einsehen. In der Detailansicht lassen sich allgemeine Informationen zum Ticket sowie alle verfassten Einträge einsehen.

Ticket kommentieren:
Selbst erstellte Tickets lassen sich kommentieren. Zum einen lassen sich so Ergänzungen zu Fehlerbeschreibungen abgeben und zum anderen kann man so auf einen Kommentar eines Bearbeiters reagieren. In der Detailansicht (vgl. Ticketverlauf einsehen) kann man über die Schaltfläche “Neuer Eintrag” einen neuen Kommentar verfassen.

Bearbeiter

Ticket nehmen/delegieren:
Unter dem Reiter “Alle Tickets” lassen sich mit einer Filterfunktion alle offenen Tickets anzeigen. Über die Schaltfläche “Details” wird die Detailansicht eines Tickets geöffnet. Neben allgemeinen Informationen zu dem jeweiligen Ticket, gibt es die zwei Schaltflächen “Delegieren” und “Nehmen”. Über diese kann man entweder das Ticket an einen anderen Bearbeiter delegieren oder es selbst in Bearbeitung nehmen.

Ticket öffnen:
Geschlossene Tickets (Reiter “Alle Tickets”) lassen sich wieder öffnen. Dies kann notwendig sein, wenn bspw. ein Problem erneut auftritt. In der Detailansicht lässt sich über die Schaltfläche “Öffnen” ein Ticket wieder in den Status offen setzen.

Ticket freigeben/schließen:
Tickets die einem Bearbeiter zugeordnet sind, lassen sich unter dem Reiter “Meine Liste” einsehen. In der Detailansicht zu einem Ticket finden sich sich u.a. die Schaltflächen “Freigeben” und “Schließen”. Über “Freigeben” kann ein Bearbeiter ein Ticket wieder abgeben. Dieses wird von der Seite “Meine Liste” entfernt und in den Status offen gesetzt. Sollte ein Problem gelöst sein, lässt sich ein Ticket schließen. Dabei wird das Ticket ebenfalls von “Meine Liste” entfernt und in den Status geschlossen gesetzt.

Ticket kommentieren:
Über die Schaltfläche “Neuer Eintrag” in der Detailansicht zu einem Ticket lässt sich ein Kommentar verfassen. Es lassen sich nur Tickets, die ein Bearbeiter unter “Meine Liste” hat, kommentieren.

Administrator

Ein Administrator kümmert sich um die Verwaltung der Benutzer. Über den Reiter “Administrator” --> “Benutzer verwalten” lassen sich alle Anwender einsehen. Über die Schaltfläche “Details” lassen sich die Detailinformationen zu einem Benutzer einsehen.

Benutzer löschen:
In der Detailansicht zu einem Benutzer gibt es die Schaltfläche “Löschen” über die sich der jeweilige Benutzer löschen lässt.

Neuen Benutzer anlegen:
Über den Reiter “Administrator” --> “Neuer Benutzer” lässt sich ein neuer Benutzer anlegen. Dazu müssen alle relevanten Daten sowie ein Passwort für den Anwender hinterlegt werden. Dieser wird per E-Mail über seine Zugangsdaten informiert und kann sein Passwort beim ersten Login ändern (siehe Mein Profil --> Passwort ändern).

FAQ

Jeder Benutzer hat die Möglichkeit unter dem Reiter “FAQ” das FAQ zu besuchen. Grundsätzlich ist es aber für Kunden gedacht. Dem FAQ neue Einträge hinzufügen oder es zu manipulieren ist nicht vorgesehen.

Mein Profil

Unter dem Reiter “Benutzer: Name” --> “Mein Profil” lassen sich Informationen zum eigenen Konto einsehen.

Passwort ändern:
Über die Schaltfläche “Passwort ändern” lässt sich das Passwort des eingeloggten Benutzers ändern. Bei erfolgreicher Änderung wird der Anwender automatisch ausgeloggt und muss sich neu authentifizieren.

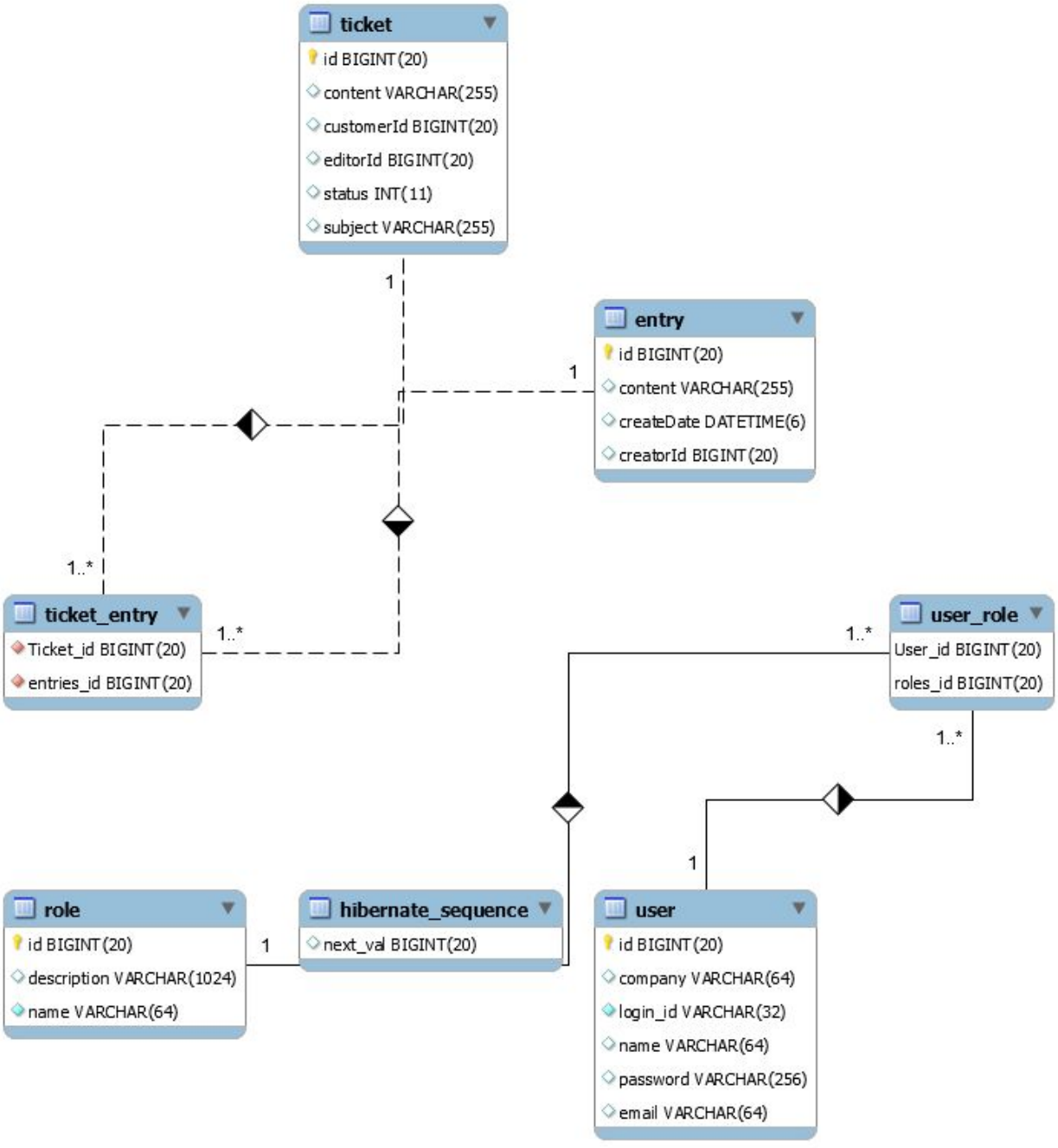
Profil löschen:
Über die Schaltfläche “Profil löschen” lässt sich ein Konto dauerhaft löschen. Die Wiederherstellung des Kontos ist nicht möglich, es müsste ein neues Konto eingerichtet werden.

Logout

Über den Reiter “Benutzer: Name” --> “Abmelden” kann sich ein Benutzer aus der aktuellen Sitzung ausloggen.

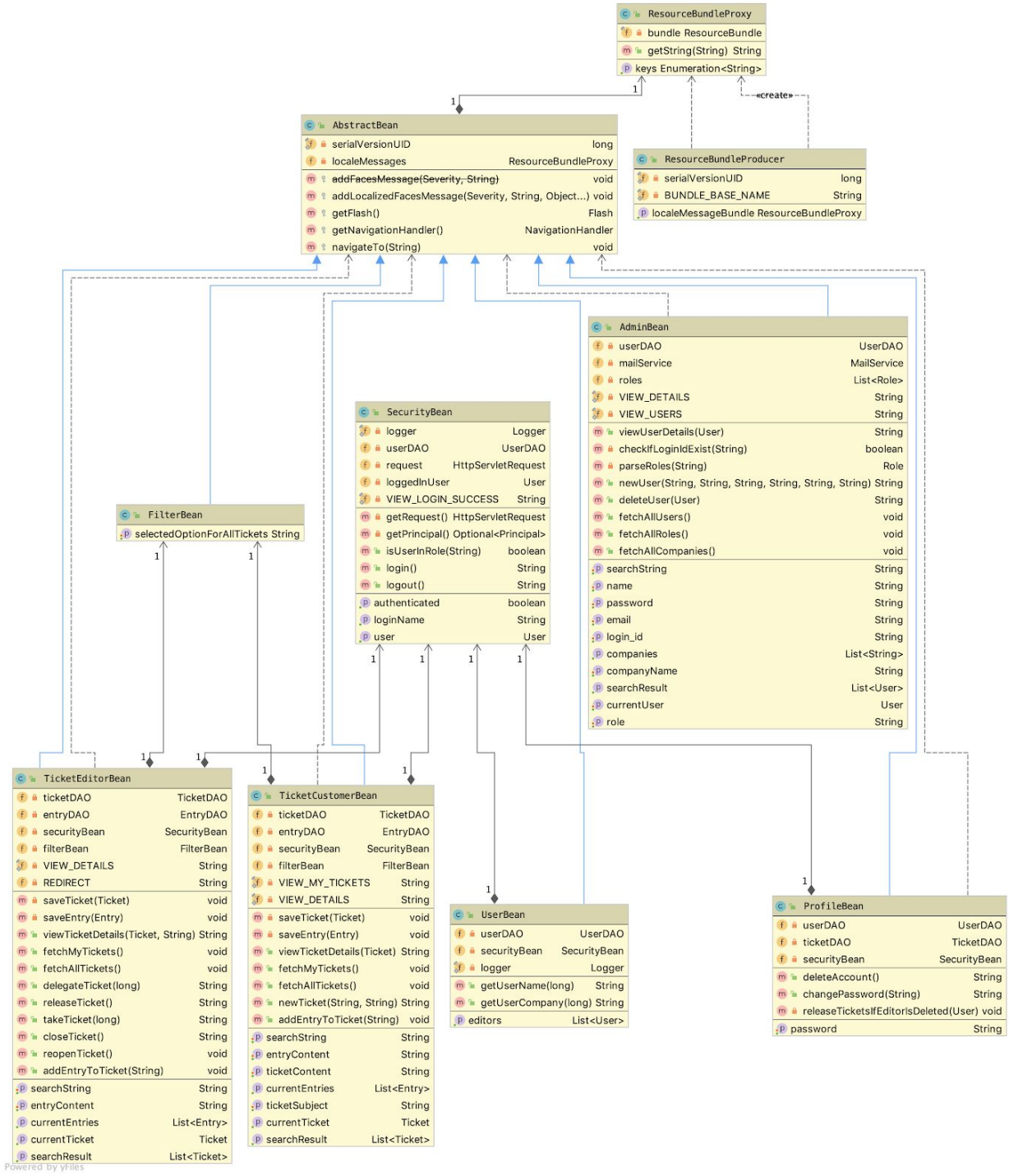
DESIGN

Datenbankmodellierung / Entity Relationship Model (ERM)

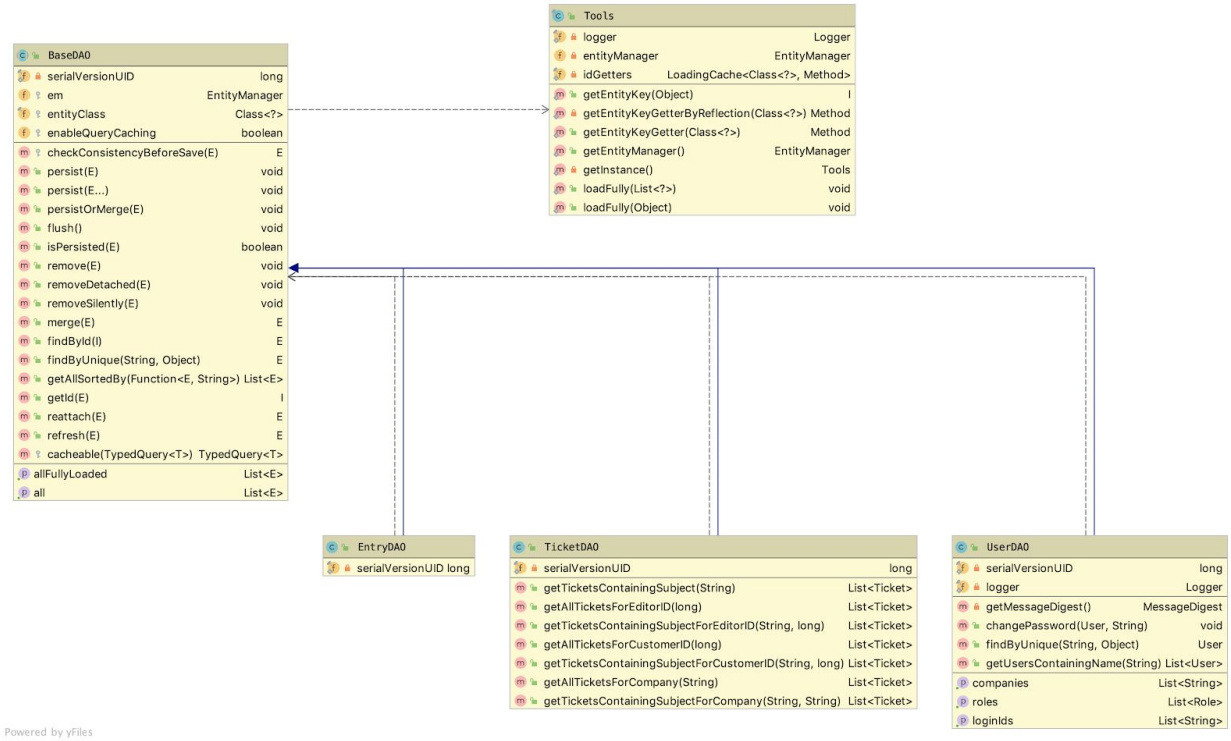


Java-Klassen

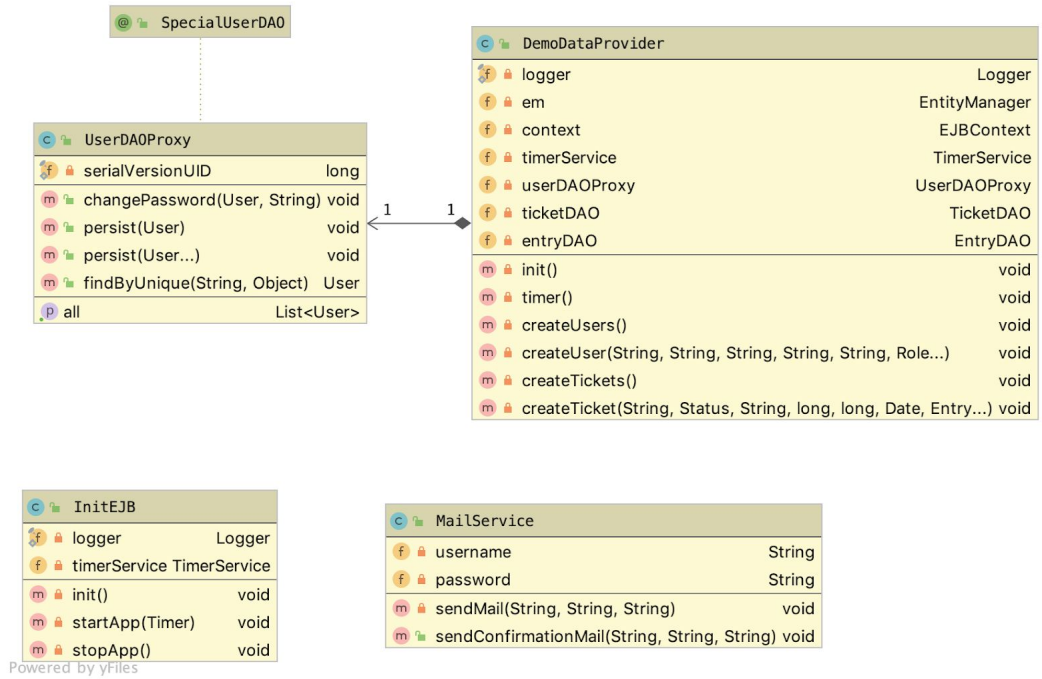
Package Beans



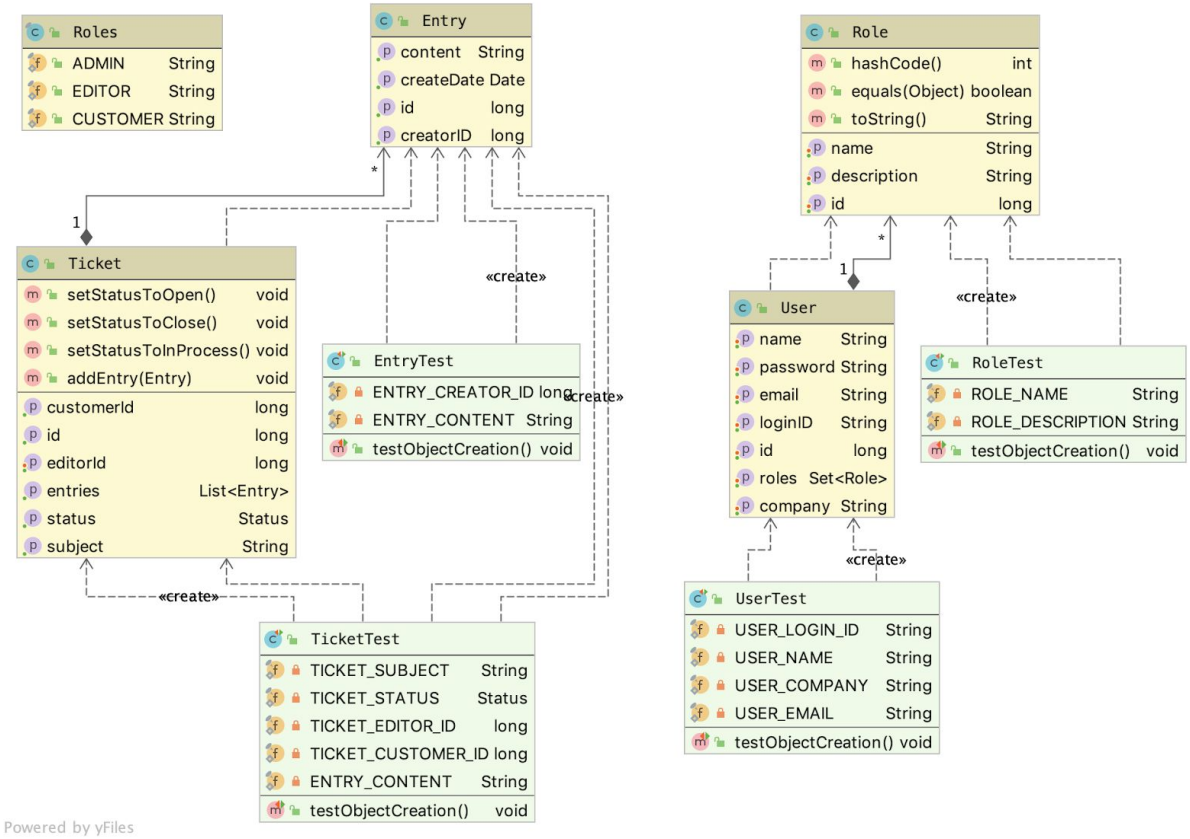
Package Database



Package EJB

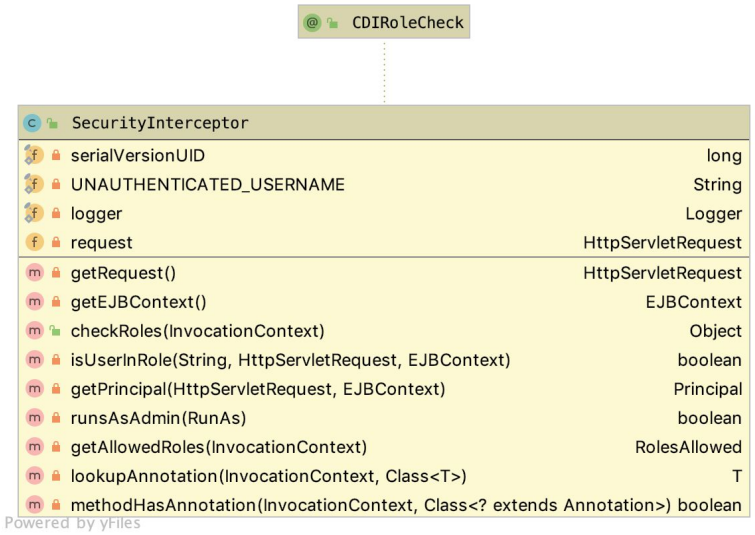


Package Model



Powered by yFiles

Package Security



Powered by yFiles

TECHNISCHE DOKUMENTATION

Aufbau der Applikation

Das Ticketsystem besteht aus einer 3-Schichten-Architektur. Es gibt eine Persistenz-, Anwendungs- und Darstellungsschicht.

Darstellungsschicht

Zum Aufbau der Darstellungsschicht wurden Java Server Faces (JSF), vor allem Primefaces, verwendet. Dabei erfolgt eine Aufteilung zwischen allgemeinen Seiten und benutzerspezifischen Seiten.

Folgende Packages werden von der Darstellungsschicht verwendet:

Package	Beschreibung
`view`	Spezielle Helfer-Klassen für Operationen im Frontend
`webapp`	Enthält die xhtml-Seiten, benötigte Ressourcen und Templates für die Darstellung im Frontend
`beans`	Filtern bei bedarf die Daten der Tickets und User und leiten diese an die Listen im Template weiter.

Das Package *view* bietet Hilfsfunktionen für die Darstellung im Frontend. In der Klasse *GeneralView* gibt es bspw. die Methode *formatDate* die den Zeitstempel aus der Datenbank in ein schöneres Format konvertiert. Die Klassen *EditorView* und *CustomerView* beinhalten je eine Funktion zum rendern von bestimmten Buttons für die Detailansicht eines Tickets. So wird z.B. für einen Bearbeiter nur der Button “Neuer Eintrag” gerendert, wenn dieser das Ticket in Bearbeitung hat. Weiterhin

Im Package *webapp/pages* finden sich in den Unterordnern *customer*, *editor* und *admin* die entsprechenden xhtml-Seiten für die unterschiedlichen Rollen. In diesen Befindet sich die Templates des Contents, der auf der Seite angezeigt wird. Die Seiten rufen Bspw. funktionen im backend auf, von denen sie Listen mit Ticket Objekten bekommen. Sie iterieren durch diese Listen und zeigen die Eigenschaften der Einzelnen Objekte an. Im Order *webapp/templates* befindet sich die basic.xhtml Seite. Diese definiert das Grundgerüst jeder Seite. Es wird dort unter anderem die Navigationsleiste, der Content der Seite und der Footer eingebunden. Der Unterordner *internals* beinhaltet mit *my-profile* und *help* die Seiten, die für jeden Benutzer verfügbar sind.

Das Package *Beans* beinhaltet die Klassen *TicketCustomerBean*, *TicketEditorBean*, UserBean und *FilterBean*. Diese Klassen stellen Methoden zur verfügung, diese filtern unter anderem die Listen von tickets, die aus der Persistenzschicht geladen werden nach den eingegebenen Suchbegriffen und den gewählten Optionen Offen, Geschlossen und in Bearbeitung. Die .xhtml Dateien bekommen die gefilterten Listen übergeben und zeigen diese an.

Die Zuordnung, welche Rolle auf welche Seiten zugreifen darf, erfolgt in der Datei *web.xml*. Dazu werden Security Constraints (Richtlinien) verwendet. So hat beispielsweise ein Customer nur Zugriff auf die Webseiten, die sich in im Unterverzeichnis *webapp/pages/editor* befinden (vgl. Abbildung).

```
<security-constraint>
    <display-name>Customer-Constraint</display-name>
    <web-resource-collection>
        <web-resource-name>Customer-Pages</web-resource-name>
        <url-pattern>/pages/customer/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>admin</role-name>
        <role-name>customer</role-name>
    </auth-constraint>
</security-constraint>
```

Die Rolle des Admins besitzt dabei laut Richtlinien Zugriff auf die Seiten der Bearbeiter und Kunden. Dies liegt jedoch daran, dass auf diese Weise der Superuser, welcher alle Rollen besitzt, für Testzwecke im Betrieb auf diese Seiten zugreifen kann. Der reguläre Administrator hingegen sieht in der Navigationsleiste nur seine eigenen Seiten, die sich unter *webapp/pages/admin* befinden (Über die URL wäre ein Zugriff auf Editor- und Customer-Pages dennoch möglich).

Anwendungsschicht

Für den Business Layer wird ein Mix aus Enterprise Java Beans und CDI-Beans verwendet, um die Daten zu verarbeiten. Nachfolgend sind die jeweiligen Packages und deren Aufgabe im Business Layer aufgeschlüsselt:

Package	Beschreibung
`beans`	Kernstück der Businesslogik; in diesem Package sind alle Klassen die zum Umgang mit Tickets, Entries und Usern im Backend notwendig sind.
`ejb`	Package für Enterprise Java Beans; mit EJBs werden im Ticket-System die Testdaten erzeugt und eingespeist. Auch der Mail-Dienst befindet sich hier.
`security`	Package zur Verwaltung einer Browser-Sitzung; hier werden Login/Logout sowie die Kontrolle der Rollen abgehandelt.

Im Package *ejb* findet sich zum einen die Klasse *DemoDataProvider*. Diese legt beim Starten der Anwendung Testdaten an, sofern noch keine in der Datenbank vorhanden sind. Ein weiteres Enterprise Java Bean stellt die Klasse *MailService* dar. Dieser nutzt die JavaMail API in Kombination mit einem Gmail Konto. Gmail als SMTP Server zu verwenden hat den Grund, dass nicht auf jedem Entwicklungsrechner lokal ein SMTP Server konfiguriert werden muss, bzw. ein zentraler Dienst bereitgestellt werden muss. Der Dienst wird bislang nur beim Anlegen von neuen Anwendern verwendet. Der vergebene Benutzername sowie das Passwort werden an die mit hinterlegte E-Mail Adresse versand. Entsprechend wurde dafür die Funktion *sendConfirmationMail* geschrieben, welche *sendMail* aufruft. Für die Authentifizierung gegenüber Gmail werden die Zugangsdaten ebenfalls in der Klasse *MailService* gespeichert (**Hinweis: das Passwort befindet sich aktuell nicht im Quelltext. Zum Testen der Funktionalität muss das Passwort "dhbwticketmaster" in der Klasse *ejb/MailService* eingetragen werden! Die Anwendung läuft aber auch ohne, es wird nur keine E-Mail versand**). Für Testzwecke hat sich dies als einfachste Konfigurationsmöglichkeit erwiesen, sollte aber für einen Produktiveinsatz geändert werden. (Um die Funktion zu testen, wird empfohlen als Administrator einen neuen Benutzer mit einer E-Mail - auf die der Tester Zugriff hat - anzulegen.)

Persistenzschicht

Zur Speicherung und Verwaltung von u.a. Tickets und Benutzern wird eine MySQL-Datenbank verwendet. Für die objektrelationale Abbildung wird die Java Persistence API in Kombination mit dem Framework Hibernate benutzt.

Für die Persistenzschicht gibt es zwei Packages, welche folgende Aufgaben übernehmen:

Package	Beschreibung
`database`	Package für Datenbankoperationen; hier befinden sich alle DAOs zum lesen aus und schreiben in die Datenbank.
`model`	Package zur Modellierung aller Datenbank-Entitäten.

Im Package *database* findet sich u.a. die Klasse *TicketDAO*. In dieser befinden sich Funktionen um Tickets nach bestimmten Attributen (Betreff, Bearbeiter ID, Kunden ID, ...) zu durchsuchen und entsprechende Treffer zurückzuliefern. Dazu werden über den EntityManager Anfragen an die Datenbank geschickt.

Die DAO-Klassen (Data-Access-Objects) erweitern die *BaseDAO* Klasse. Diese stellt neben dem EntityManager grundlegende Funktionen zum Speichern und Entfernen von Objekten aus der Datenbank.

Im Package *model* werden die Entitäten des Datenbankmodells in Java abgebildet. Für jede Entität wird eine Klasse erstellt. Diese enthalten einen Konstruktor zur Erstellung eines Objektes sowie Getter und Setter Methoden um auf dessen Attribute zugreifen zu können.