Gruppe: gr-06-2 Projektleiter: Krabbes, Anja
Verantwortliche: Quan Nguyen Anh, Shengjie Zhang

Entwurfsbeschreibung

1. Allgemeines
2. Produktübersicht elatePortal
3. Grundsätzliche Struktur- und Entwurfsprinzipien für das Gesamtsystem
4. Grundsätzliche Struktur- und Entwurfsprinzipien der einzelnen Pakete

1. Allgemeines

ElatePortal ist ein Webanwendung, die unter JSR 168 entwickelt wird und auf Jetspeed 2 Portal basiert.
Es wird JDK 1.5.0 oder höhere Version, Tomcat benötigt. Mehr Deteils befinden sich unter www.elateportal.de und unter Installation/Prerequisites.

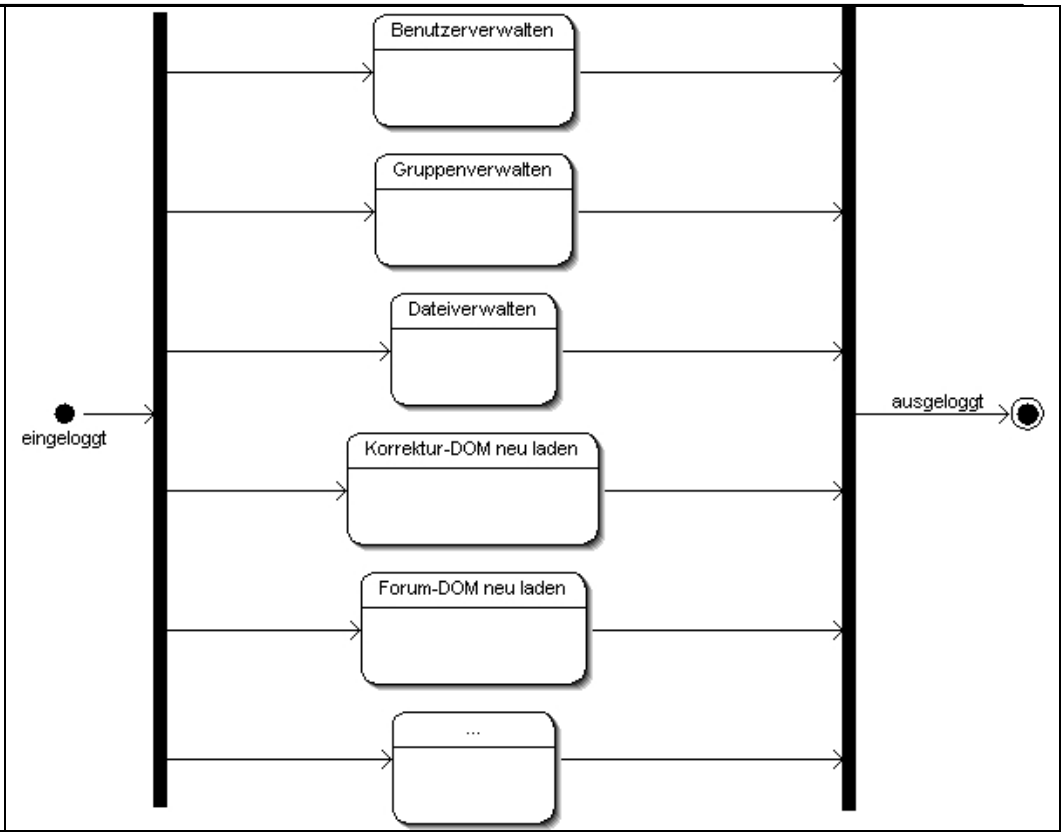
2. Produktübersicht elatePortal

elatePortal bietet die Intergrationen von einer großen Anzahl von Veranstaltungen und „eTesting“. Und die Oberfläche ist sehr einfach für Anwender (zum Beispiel : Dozenten, Studenten, Tutoren...) zu bedienen. ElatePortal kann leicht erweitert und skaliert.

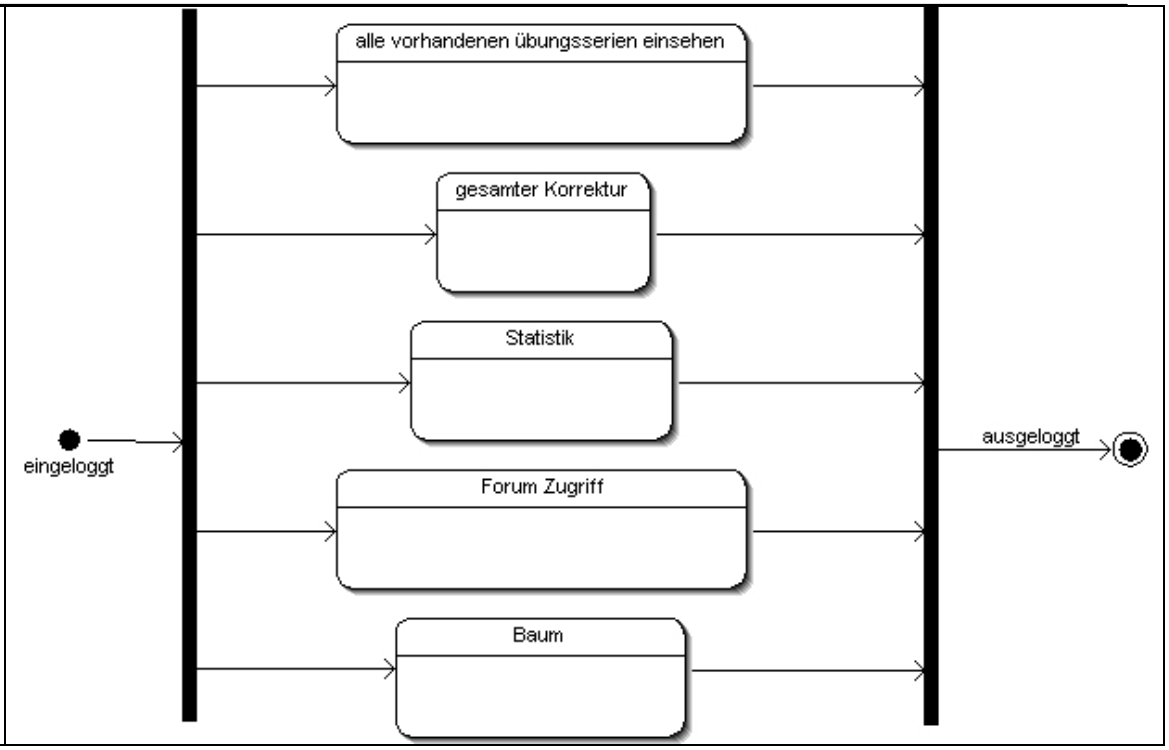
Hier sind die Funktion, die für die entsprechenden Rollen bestimmt:

Administrator	Tutor	Dozent	Student
- Benutzerverwaltung - Gruppenverwaltung - Dateiverwaltung - Konfiguration - Korrektur-DOM neu laden - Forum-DOM neu laden - Prefs-DOM neu laden - Serien-DOM neu laden - Identität eines bestimmten Benutzers annehmen	- alle vorhandenen übungsserien einsehen - gesamter Korrektur - Baum - Statistik - Forum Zugriff	- Benutzerverwaltung - Gruppenverwaltung - Übungsserien verwalten - Materialien zur Vorlesung veröffentlichen - Klausuren verwalten - Veranstaltung konfigurieren - News konfigurieren - Korrektur Statistik - Korrektur-Baum - System Konfiguration - Dateiverwaltung - Zugriff auf Forum	- Einschreibung zu einer bestimmten übungsgruppe - Download von übungsserien - Upload der Loesung zur übungsserie - eigenen Punktestand einsehen - Klausureinschreibung - Materialien zur Vorlesung herunterladen - Zugriff auf Forum

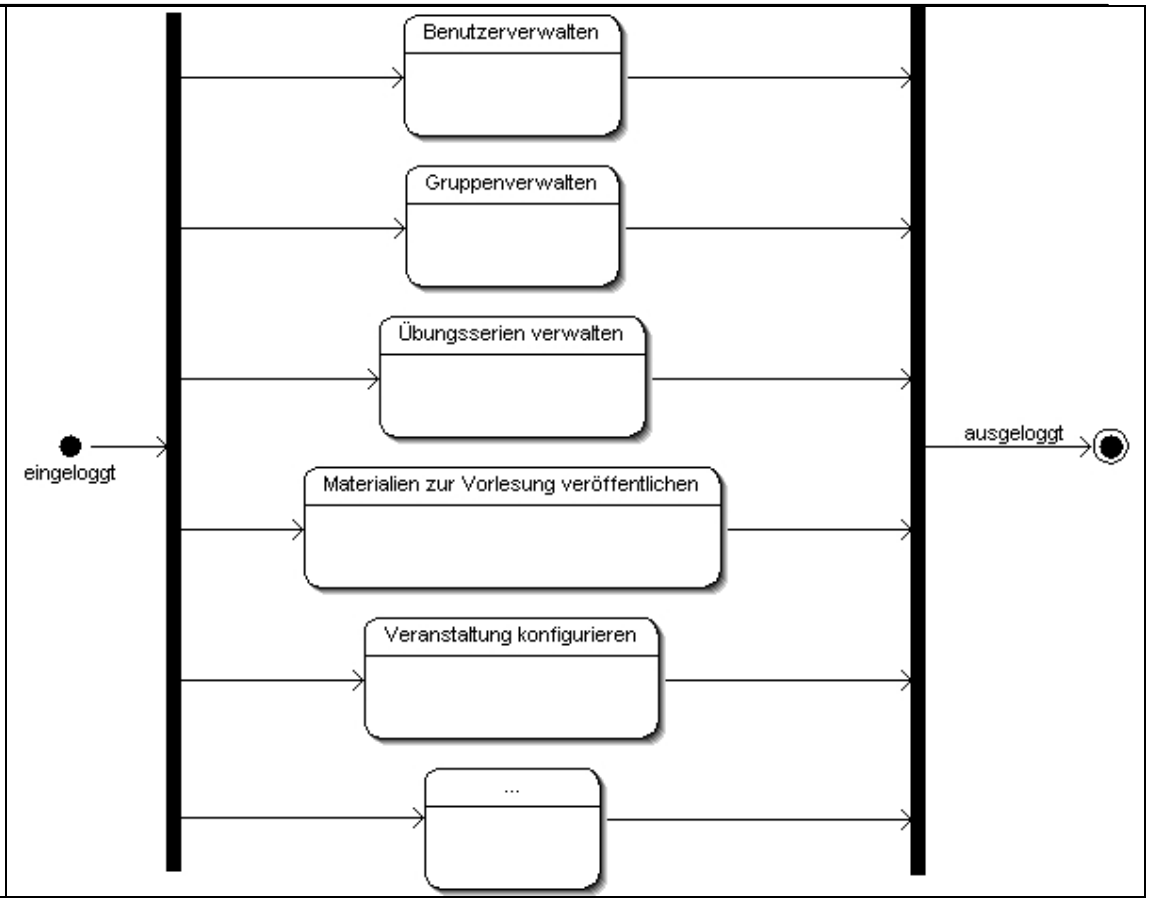
Administrator

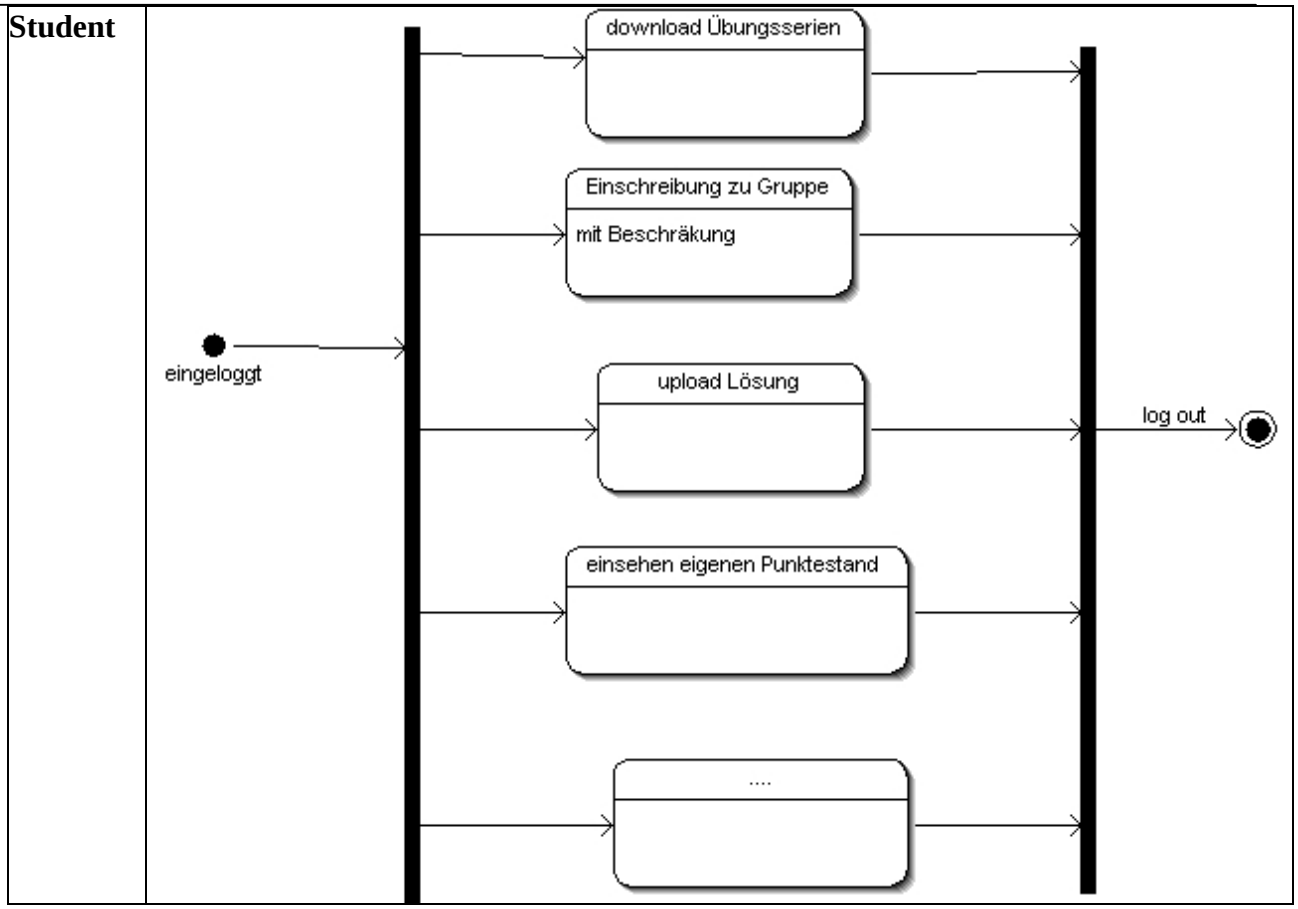


Tutor


alle vorhandenen übungsserien einsehen
gesamter Korrektur
Statistik
eingeloggt
ausgeloggt
Forum Zugriff
Baum

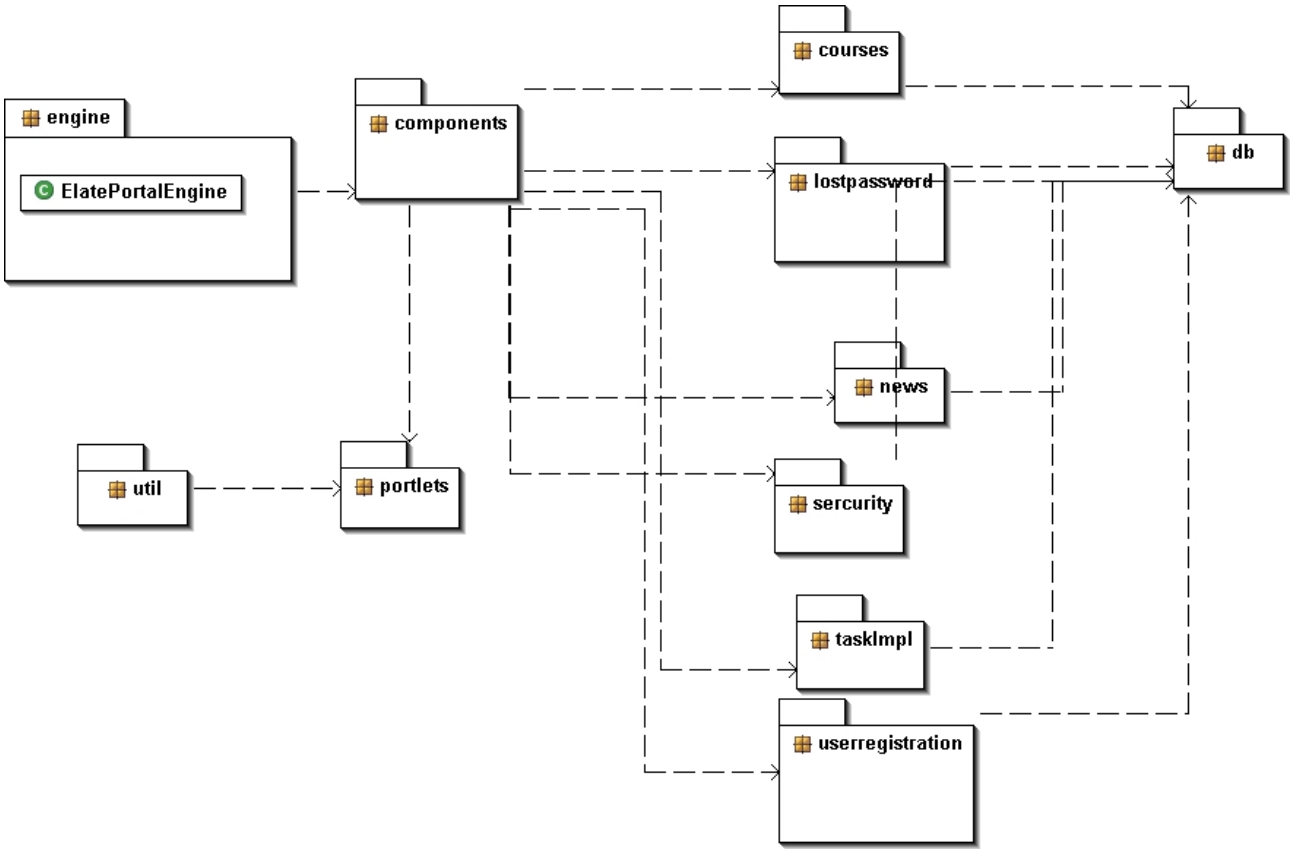
Dozent


Benutzerverwalten
Gruppenverwalten
Übungsserien verwalten
eingeloggt
ausgeloggt
Materialien zur Vorlesung veröffentlichen
Veranstaltung konfigurieren
...


Student
download Übungsserien
Einschreibung zu Gruppe
mit Beschräkung
eingeloggt
upload Lösung
log out
einsehen eigenen Punktestand
....

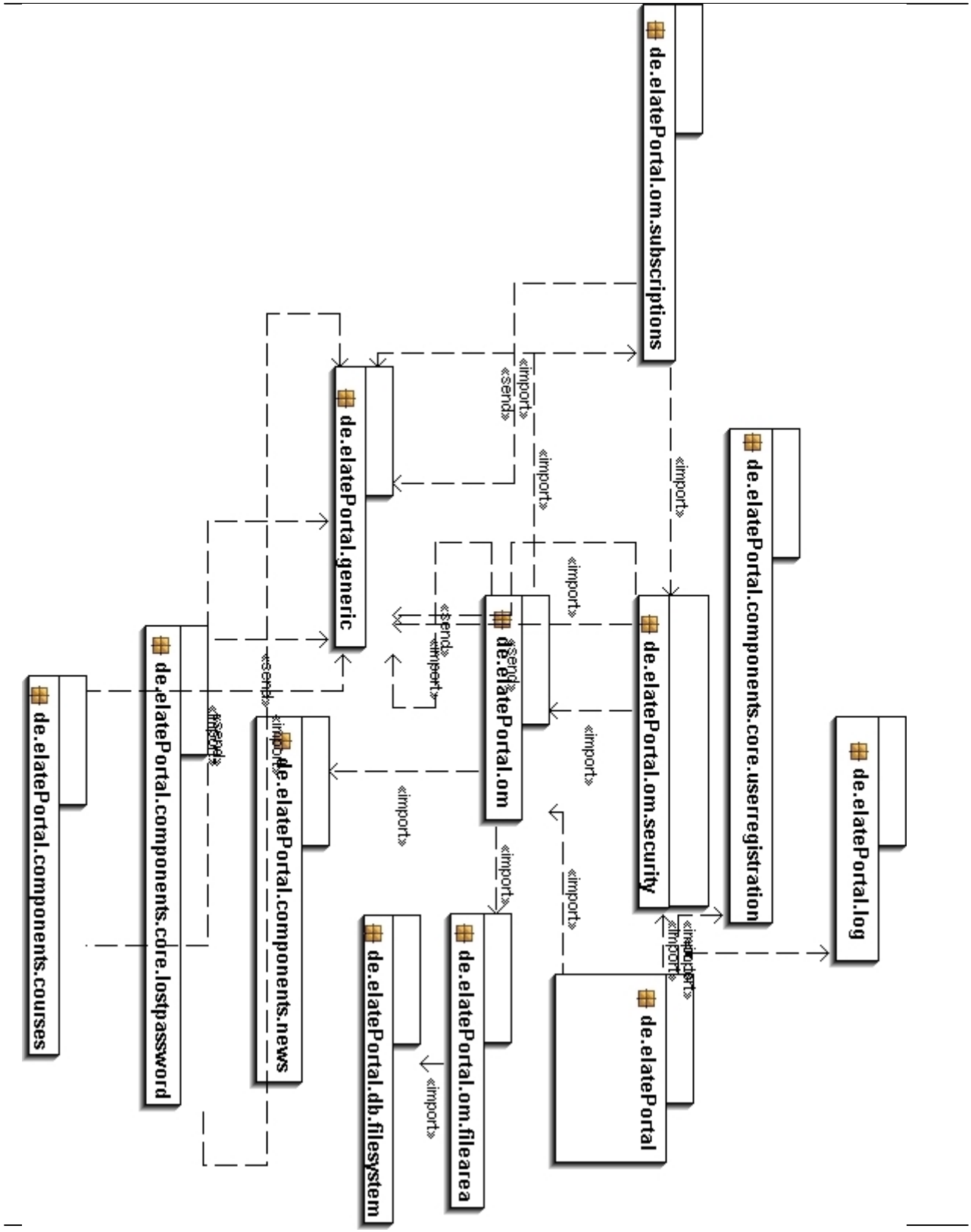
3. Grundsätzliche Struktur- und Entwurfsprinzipien für das Gesamtsystem

Packetstruktur elatePA:



Wie erwähnt wurde, dass das engine-Packet die Aufgabe hat, sodass es die Basiskomponente aufruft. Und components-Packet enthält die Basiskomponente, die die Daten von Benutzer verwaltet. Das db-Packet enthält die Klassen, die die Daten von Komponente auf Datenbank speichert. Das portlets-Packet enthält die Klassen, die die Daten von Komponente zur Repräsentation auf Webseite bringt.

Paketdiagramm api:


de.elatePortal.om.subscriptions
de.elatePortal.components.core.userregistration
de.elatePortal.log
de.elatePortal.generic
de.elatePortal.om.security
de.elatePortal.om
de.elatePortal
de.elatePortal.components.news
de.elatePortal.om.filearea
de.elatePortal.db.filesystem
de.elatePortal.components.core.lostpassword
de.elatePortal.components.courses
«import»
«send»

Die de.elatePortal Package verantwortlich für Aufruf die alle Servie der EletaPortal, und die andere Komponent Packet besitzt die alle grundlegende Methode, die die EletaProtalservice wieder benutzt.

4. Grundsätzliche Struktur- und Entwurfsprinzipien der einzelnen Pakete

Designschwerpunkte elatePA:

Anfang mit ElatePortal.java:
Am Anfang wird immer die Klasse **ElatePortal** aufgerufen. Diese Klasse eine Methode, die eine Instanz von der Klasse **ElatePortalEngine** in dem Packet **engine** erzeugt.

Code:

```
public static void createEngine( ElatePortalConfigManager configManager ) throws
ElatePortalException{
 engine = new ElatePortalEngine( configManager );
}
```

Nachdem die Klasse ElatePortalEngine initialisiert, erstellt seine Instanz die Basiskomponenten, die auf Spring Plattform basiert.

Code:

```
protected void init( ElatePortalConfigManager configManager ) throws
ElatePortalException{
        this.configManager = configManager;
        // instantiate the elatePortal File Repository Manager
        fileRepositoryManager = new FileRepositoryManagerImpl( configManager );
        // quick fix, see issues.txt (memory leak on redeployment)
        if( adminThreads == null )
                adminThreads = new AdminThreads();
        // we need repositoryPath in Spring XML Bean Definitions,
        // which is set in ElatePortalConfigManager
        try {
                System.out.println( configManager.getRealPath( "/WEB-
INF/conf/elatePortalSpring.xml" ) );
//              XmlBeanFactory beanFactory = new XmlBeanFactory(
//                      new FileSystemResource(
//                              configManager.getRealPath( "WEB-
INF/conf/elatePortalSpring.xml") ) );
                // start spring components
                // TODO use WebApplicationContext instead
//              GenericApplicationContext a = new GenericApplicationContext();
                FileSystemXmlApplicationContext appcontext=new
FileSystemXmlApplicationContext(
                        new String[]{ "/" + configManager.getRealPath( "/WEB-
INF/conf/elatePortalSpring.xml" ),
                                "/" + configManager.getRealPath(
"/WEB-INF/conf/taskAPISpring.xml" )}, false );
```

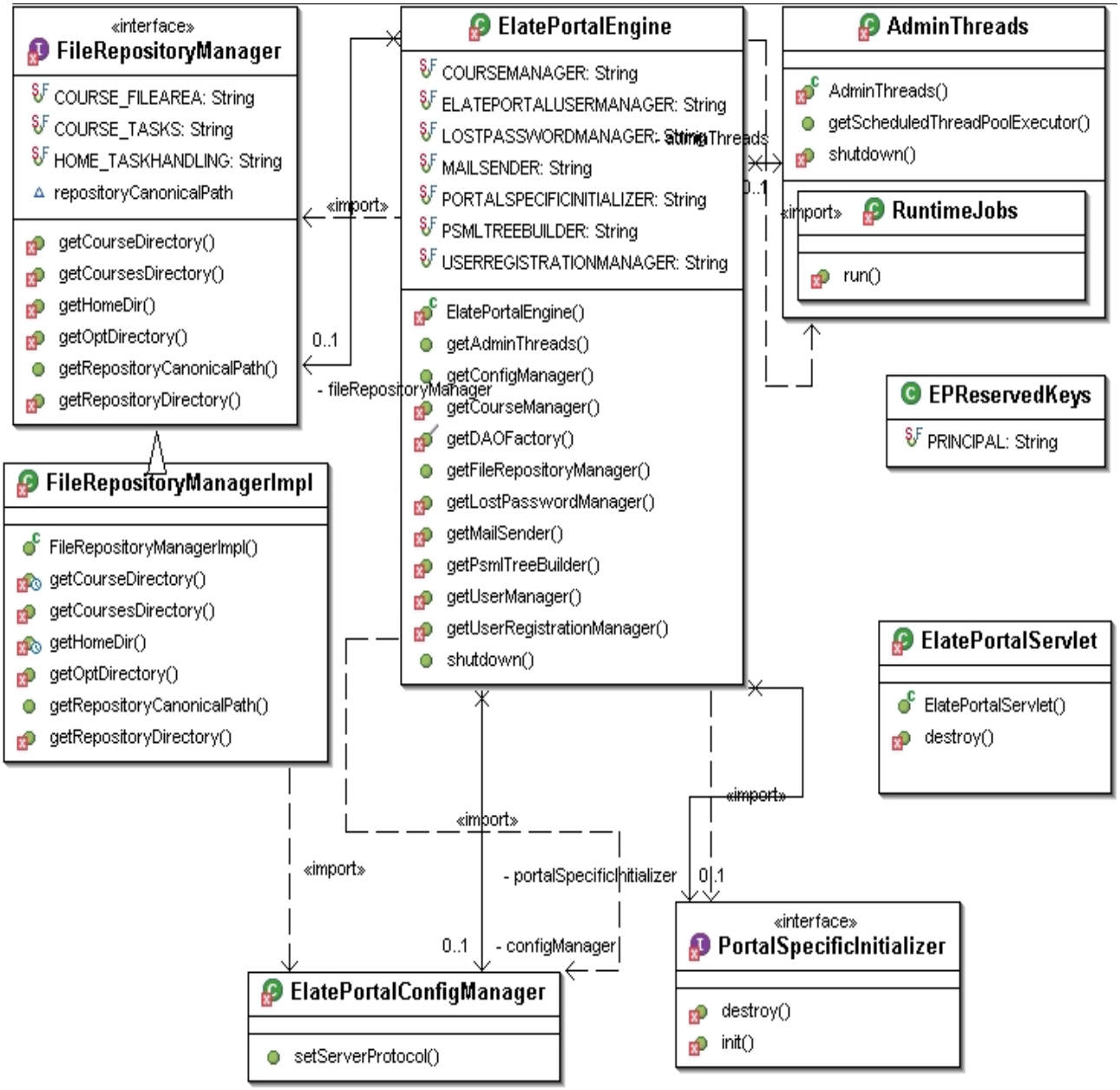
```
                    appcontext.refresh();
//                  WebApplicationContext appcontext =
WebApplicationContextUtils.getWebApplicationContext( configManager.getServletContext()
);
                    // init OJB, not needed when using an applicationcontext
                    //appcontext.getBean( "ojbConfigurer" );

                    // instantiate DAOFactory, TODO: contract first design
                    dAOFactory = new DAOFactory(appcontext);

                    // instantiate initializer, which is specific to the underlying portal
                    portalSpecificInitializer = (PortalSpecificInitializer)
                                                        appcontext.getBean(
PORTALSPECIFICINITIALIZER );
                    // instantiate the course manager
                    courseManager = (CourseManager)
                    appcontext.getBean( COURSEMANAGER );
                    // instantiate UserRegistrationManager
                    userRegistrationManager = (UserRegistrationManager)
                    appcontext.getBean( USERREGISTRATIONMANAGER );
                    // instantiate MailSender (from Spring)
                    mailSender = (MailSender)appcontext.getBean( MAILSENDER );
                    userManager = (EpUserManager)appcontext.getBean(
ELATEPORTALUSERMANAGER );
                    lostPasswordManager = (LostPasswordManager)
                                        appcontext.getBean( LOSTPASSWORDMANAGER );
                    psmlTreeBuilder = (PsmlTreeBuilder)
                            appcontext.getBean( PSMLTREEBUILDER );
                }
```

Und von diesen Basiskomponente werden die anderen komplexeren Komponenten aufgerufen. Und hier ist Klassendiagramm:

Gruppe: gr-06-2 Projektleiter: Krabbes, Anja
Verantwortliche: Quan Nguyen Anh, Shengjie Zhang



Desigenschwerepunkte api:

Am Anfang wird immer die Klasse **EletaPortalservice** aufgerufen. Diese Klasse erzeugt die Instance von Klassen UserRegistrationManager(Paket de.elatePortal.components.core.userregistration), Logger(Paket de.elatePortal.log), CourseManager(Paket de.elatePortal.om), EpUserManager (Paket de.elatePortal.om.security).

Quellcode:

```
public void setElatePortalUserManager(
                        EpUserManager elatePortalUserManager) {
                this.elatePortalUserManager = elatePortalUserManager;
        }
public EpUserManager getElatePortalUserManager() {
                return elatePortalUserManager;
        }
public void setRequestLogger(Logger requestLogger) {
                this.requestLogger = requestLogger;
        }
public Logger getRequestLogger() {
                return requestLogger;
        }
public void setUserRegistrationManager(
                        UserRegistrationManager userRegistrationManager) {
                this.userRegistrationManager = userRegistrationManager;
        }
public UserRegistrationManager getUserRegistrationManager() {
                return userRegistrationManager;
        }
public void setCourseManager(CourseManager courseManager) {
                this.courseManager = courseManager;
        }
public CourseManager getCourseManager() {
                return courseManager;
        }
```

Es wird auch ein Instance von Klasse EletaPortalservice erzeugt.

EpUserManager ist einen Interface.

public boolean authenticate(String login, String password)

Diese Methode Authenticates users. Es wird login und Password überprüft. Es ist true, wenn login und password akzeptiert, bei anderen Falls falsch.

Logger ist ein Interface.
public boolean isDebugEnabled()
debug logging currently enabled?
Call this method to prevent having to perform expensive operations
(for example, <code>String</code> concatenation) when the log level is more than debug.
public boolean isErrorEnabled()
Is error logging currently enabled?
Call this method to prevent having to perform expensive operations (for example, <code>String</code> concatenation) when the log level is more than error.
Es gibt auch Analog Methode, um die andere Form von Loggin zu prüfen.
public void trace(Object message)
Log a message with trace log level
Analog gibts es auch Methode, um die anderen Falls zu Logging.
UserRegistrationManager ist ein Interface.
Bei diesem Interface sind alle Methode zu Informationüberprüfen, um User zu regiestieren.
public void doAddSubmittedUserToPendingUsers(UserRegistrationContext urcontext) throws UserRegistrationException, MailingException;
public UserRegistrationContext doValidateAndInvoke(String id) throws UserRegistrationException
ublic void checkLoginFieldConstraint(String login) throws UserRegistrationException;

public void checkNameFieldConstraint(String name) throws UserRegistrationException;

public void checkSurnameFieldConstraint(String surname) throws UserRegistrationException;

public void checkMailFieldConstraint(String mail) throws UserRegistrationExcept
UserRegistrationContext ist eine Klasse.
Sie definiert die kontex der Registration von Users. Es beinhaltet loggin,surname,name,Email,IP,Date,TAN etc..
CourseManager ist ein Interface.
public List<de.elatePortal.om.Course> getActiveCourses();
public List<de.elatePortal.om.Course> getCourses();
Listet alle der Vorgänge.
public void addUserToCourse(String login, String courseId);
Fügt ein User(login, id) zur Vorgängen ein.

public List<de.elatePortal.om.Course> getCoursesForUser(String login)
Listet alle Vorgänge für ein User.
EpUser ist ein Interface, definiert alle Eigenschaften für ein User, die regiestieren möchte.
public static final String ATTRIBUTE_Surname = "user.name.given";
public static final String ATTRIBUTE_Name = "user.name.family";
public static final String ATTRIBUTE_EMail = "user.home-info.online.email";

Klassendiagramme: