Entwurf

Endgültige Abgabe

Team 2

SEP WS 2021/22

Betreuer:

Prof. Dr. Christian Bachmaier

Projektphase	**Leiter**
Pflichtenheft	Johann Schicho
Entwurf	Stefanie Gürster
Feinspezifikation	Johannes Garstenauer
Implementierung	Thomas Kirz
Validierung	Sebastian Vogt

12. November 2021

Inhaltsverzeichnis

1 Einleitung

Johann Schicho

In diesem Dokument ist der erste Entwurf der Anwendung niedergeschrieben. Dazu werden die einzelnen Komponenten genau spezifiziert und aufgelistet oder grafisch dargestellt.

In den Kapiteln wird dann die Klassen- und Datenbankstruktur genau dargestellt und die Systemfunktionen und Systemarchitektur definiert.

2 Systemarchitektur

Johannes Garstenauer

2.1 Schichten

Diese Anwendung folgt einer Schichtenarchitektur von vier Schichten. Dies bietet ein gutes Verhältnis zwischen dem Mehraufwand, welcher mit der Verwendung von zusätzlichen Schichten verbunden ist, und der dadurch gewonnen Modularität und Austauschbarkeit.

- Die *view*-Schicht enthält die Komponenten zur grafischen Darstellung.
- Die *control*-Schicht enthält die Komponenten zur Steuerung der grafischen Darstellung und Reaktion auf Nutzereingaben.
- Die *business*-Schicht enthält die Komponenten, welche die Anwendungslogik umsetzen.
- Die *persistence*-Schicht enthält die Komponenten zum Zugriff auf den Datenbankserver.

Diese Schichten folgen der *MVC-Architektur* wie in **Abbildung 1** dargestellt. Die Persistenz- und Businessschichten gehören hierbei zum *Model*. Die Kontrollschicht stellt den *Controller* und die *Faceletschicht* die *View* dar.

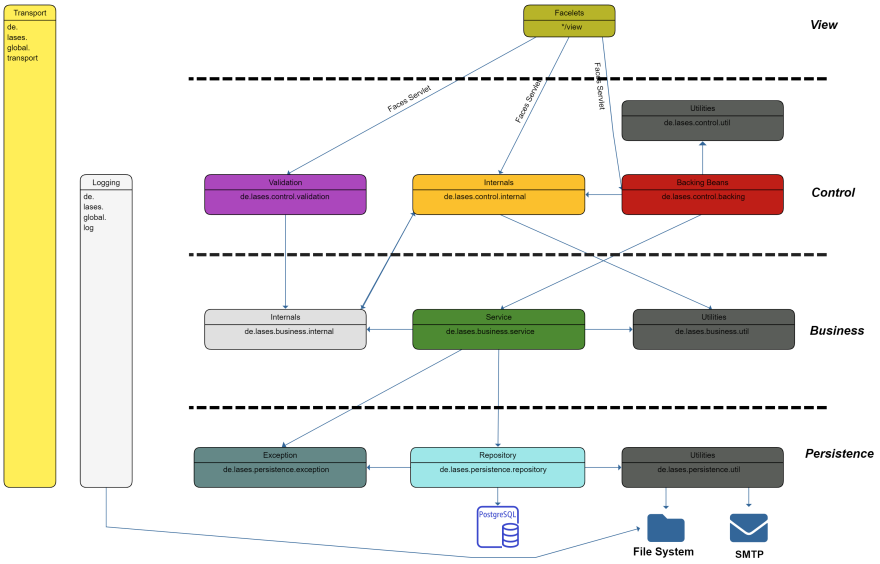


Abbildung 1: Das Schichtenmodell der LasEs-Anwendung.

2.2 Pakete

2.2.1 de.lases.control

Dieses Paket enthält alle Klassen der Kontrollschicht.

de.lases.control.internal enthält die Klassen, welche zur internen Verwaltung verwendet werden. Hierzugehört beispielsweise die Sessionverwaltung und das Generieren von *FacesMeessages*.

de.lases.control.validation enthält alle Klassen, die zur Validierung von Nutzereingaben in *Facelets* verwendet werden.

de.lases.control.backing enthält alle *Managed Beans* welche zur Darstellung der *Facelets* aus der *View* Verwendung finden. Zu jedem *Facelet* [facelet] existiert genau ein *Backing-Bean*, welches [facelet]Backing genannt wird.

de.lases.control.util enthält statische Hilfsklassen, welche Dienste zur Verfügung stellen, die von der Kontrolllogik selbst getrennt werden können.

2.2.2 de.lases.business

Dieses Paket enthält alle Klassen der Anwendungslogikschicht.

de.lases.business.service enthält alle Klassen, die den Hauptteil der Anwendungslogik umsetzen. Das sind also alle Dienste welche den *Backing Beans* zur Verfügung gestellt werden und auf die Repositories zugreifen.

de.lases.business.util enthält statische Hilfsklassen, welche Dienste zur Verfügung stellen, die von der Anwendungslogik selbst getrennt werden können.

de.lases.business.exception enthält die Exceptionklassen, die ein Fehlverhalten im System repräsentieren.

de.lases.business.internal enthält die Klassen, welche zur internen Verwaltung verwendet werden. Beispielsweise Dienste, welche periodische Aufräumaufarbeiten in der Datenbank anstoßen.

2.2.3 de.lases.persistence

Dieses Paket enthält alle Klassen der Persistenzschicht.

de.lases.persistence.repository enthält alle Klassen, die dem direkten Zugriff auf die Datenbank dienen.

de.lases.persistence.util enthält statische Hilfsklassen zur Datenbankverwaltung, wie Dienste zum Einlesen der Konfiguration.

de.lases.persistence.exception enthält die Exceptionklassen, welche ein Fehlverhalten im Umgang mit der Datenbank repräsentieren. Sie repräsentieren ebenfalls Fehler, die durch die Verwendung der *Repository*-Methoden entstehen. Somit können sie auch in der *Businessschicht* verwendet werden. Das bedeutet, dass sich die beiden Schichten die *Exception*-Klassen teilen.

2.2.4 de.lases.global

Dieses Paket enthält alle Klassen welche über mehrere Schichten hinweg verwendet werden.

de.lases.global.logging enthält Hilfsklassen für schichtenübergreifende Loggingfunktionen.

de.lases.global.transport enthält alle POJO-Klassen, die Daten in der Anwendung repräsentieren und zu deren Transport verwendet werden. Sie setzen keine Anwendungslogik um.

2.3 Fehlerbehandlung

Bei Verwendung der Anwendung können Fehler auftreten. Diese werden zur Nachvollziehbarkeit immer geloggt. Daraufhin wird auf verschiedene Weisen verfahren.

2.3.1 Geprüfte Ausnahmen

Geprüfte Ausnahmen repräsentieren Ausnahmesituationen, mit denen gerechnet werden muss und auf die reagiert wird. Treten solche in der *Persistenzschicht* auf, dann werden sie in eine semantisch passende Ausnahmeklasse mit Fehlermeldung gekapselt und

an den Aufrufer weiterpropagiert. Wenn in der *Business-* oder *Kontrollschicht* solch eine Ausnahmeklasse ankommt oder entsteht, so wird ein *Event* ausgelöst, welches eine beschreibende *UIMessage* enthält. Dieses wird vom *UIMessageGenerator* gefangen, woraufhin eine passende *FacesMessage* generiert wird. Diese werden dann von den zugehörigen *Facelets* gerendert. Der Vorteil dieses Ansatzes ist, dass die Logik zum Abfangen der Ausnahmen und zur Generierung der *FacesMessages* lokal an einer Stelle im System stattfindet.

2.3.2 Ungeprüfte Ausnahmen

Bei ungeprüften Ausnahmen handelt es sich um fatale Fehler, wie sie beispielsweise durch Programmierfehler entstehen können. Teilweise werden geprüfte Ausnahmen in ungeprüfte umgewandelt, wenn das System und der Nutzer bei deren Behandlung machtlos sind (z.B. keine Datenbankverbindung vorhanden). Ungeprüfte Ausnahmen gehören generell jeder Schicht an, da sie alle Schichten unbemerkt durchlaufen, bis sie vom anwendungsweiten *UncheckedExceptionHandler* aufgegriffen werden. Der Nutzer wird daraufhin auf die Fehlerseite weitergeleitet, wo ihm eine informative *FacesMessage* angezeigt wird.

2.4 Frameworks

2.4.1 Jakarta Server Faces

Jakarta Server Faces (JSF; früher JavaServer Faces) ist ein Framework-Standard zur Entwicklung von grafischen Benutzeroberflächen für Webanwendungen in Java, basierend auf Servlets und JSP-Technik. Es wird die Referenzimplementierung *Mojarra* verwendet.

2.4.2 Context and Dependency Injection

Die *Context and Dependency Injection* ist eine Technologie welche die Abhängigkeitsverwaltung zwischen Objekten vereinfacht. Im Sinne der *Inversion of Control* werden dem Entwickler Werkzeuge an die Hand gelegt, mithilfe derer er Objektreferenzierung tätigen kann, ohne diese Objekte vorher explizit zu erstellen. Diese referenzierten Objekte können in andere Objekte zur Laufzeit *injiziert* werden. Somit können Objekte in dieser Anwendung ohne Initialisierung oder umständlichen Transport der Referenzierung verwendet werden. Dies ermöglicht eine losere Kopplung zwischen Klassen und somit Erweiterbarkeit und Austauschbarkeit. Die *CDI* verwaltet in dieser Anwendung alle Objekte der *Backing-Beans* als *@RequestScoped* und alle *Services* als *@ApplicationScoped*. Weiterhin sind einige Klassen *@SessionScoped* wie beispielsweise die *SessionInformation*. Letzlich gibt es einige Klassen, welche *@ApplicationScoped* wie der *ConnectionPool* oder *PeriodicWorker*.Es wird die Referenzimplementierung *Weld* von *JBoss* verwendet.

2.5 Patterns

2.5.1 Model View Controller

Dieses Entwurfsmuster trennt die Klassen und Komponenten der Anwendung in drei verschieden Teile, welche jeweils eigene Aufgaben übernehmen:

- Die ***View-Komponenten*** übernehmen alle Aufgaben, welche mit der grafischen Darstellung der Anwendung zusammenhängen. In dieser Anwendung sind dies die *Facelets*.
- Die ***Controller-Komponenten*** übernehmen alle Aufgaben, welche die Interaktion zwischen der View und dem Model betrifft. Sie behandeln Nutzereingaben, Validation und Konvertierung etc. In dieser Anwendung befinden sie sich in den de.lases.control Paketen.
- Die ***Model-Komponenten*** beinhalten die Modelldaten, sowie jene Klassen welche sich mit deren Beschaffung und Verarbeitung beschäftigen. In dieser Anwendung befinden sie sich in den de.lases.business und de.lases.persistence Paketen.

Die Verwendung dieses Patterns bietet viele Vorteile hinsichtlich der Modularität und Erweiterbarkeit der Anwendung.

2.5.2 Singleton

Eine Klasse, welche nach dem Singleton-Entwurfsmuster implementiert wurde zeichnet sich dadurch aus, dass es nur genau eine Instanziierung von dieser Klasse geben kann und darf. Hierdurch werden vorrangig Ressourcen gespart, außerdem wird die Verwaltung der im Objekt gekapselten Funktionalität erleichtert. Dies ermöglicht ebenfalls einen leichten globalen oder paketweiten Zugriff auf jene Funktionalitäten. Ein Beispiel hierfür ist der Connection Pool. Er verwendet das *Singleton*-Pattern damit sichergestellt wird, dass eine genau festgelegte Anzahl an Datenbankverbindungen existiert und nicht mehr.

2.5.3 Data Transfer Object

Dieses Entwurfsmuster schreibt die Verwendung von *Data Transfer Objects* zur einheitlichen Darstellung der Entitäten in den Modelldaten und dem Transport von Daten vor. Diese sind POJOs, welche private Felder zur Repräsentation der Daten mit Getter- und Settermethoden haben. Der Vorteil der Verwendung dieses Entwurfsmusters liegt darin, dass eine einheitliche und übersichtliche Schnittstelle für Datenrepräsentation und Transport in der Anwendung besteht. Der interne Transport von Daten wird in diesem System größtenteils mithilfe von *DTOs* oder *Java Collections von DTOs* umgesetzt. Die zugehörigen Klassen lagern im de.lases.global.transport Paket.

2.5.4 Repository

Beim *Repository* Pattern wird für jedes Business-Objekt oder jede Entität eine Klasse definiert welche den Datenbankzugriff auf die zugehörigen Daten steuert. Weiterhin findet hier das Mapping auf die DTO-Objekte dieser Anwendung statt. In dieser Anwendung finden sich die *Repository-Pattern* Klassen im Paket de.lases.persistence.repository. Durch die mit der Verwendung dieses Patterns gewonnenen Übersichtlichkeit erleichtern wir uns die Implementation und ermöglichen eine leichte Erweiterbarkeit der Datenschemas.

2.5.5 Object Pool

Das *Object Pool* Erzeugungsmuster wird dazu verwendet, Objekte nach initialer Erzeugung vorzubehalten, da dies sinnvoller ist als sie bei jeder Verwendung neu zu erzeugen. Somit werden Ressourcen gespart. In dieser Anwendung findet dieses Entwurfsmuster beim *ConnectionPool* Verwendung und bietet den zusätzlichen Vorteil, dass eine feste Maximalanzahl an Datenbankverbindungen festgelegt werden kann.

2.5.6 Observer

Das *Observer* Verhaltensmuster beschreibt Methoden, welche auf Änderungen im Zustand eines Systems warten, darüber benachrichtigt werden und daraufhin handeln. In unserem System findet das beispielsweise beim *UIMessageGenerator* Verwendung, welcher eine *Event<UIMessage>*Events beobachtet und auf diese mit der Generierung von *FacesMessages* reagiert. Somit herrscht eine größere Entkoppelung und es gibt einen geringeren Programmieraufwand.

3 Klassendiagramm

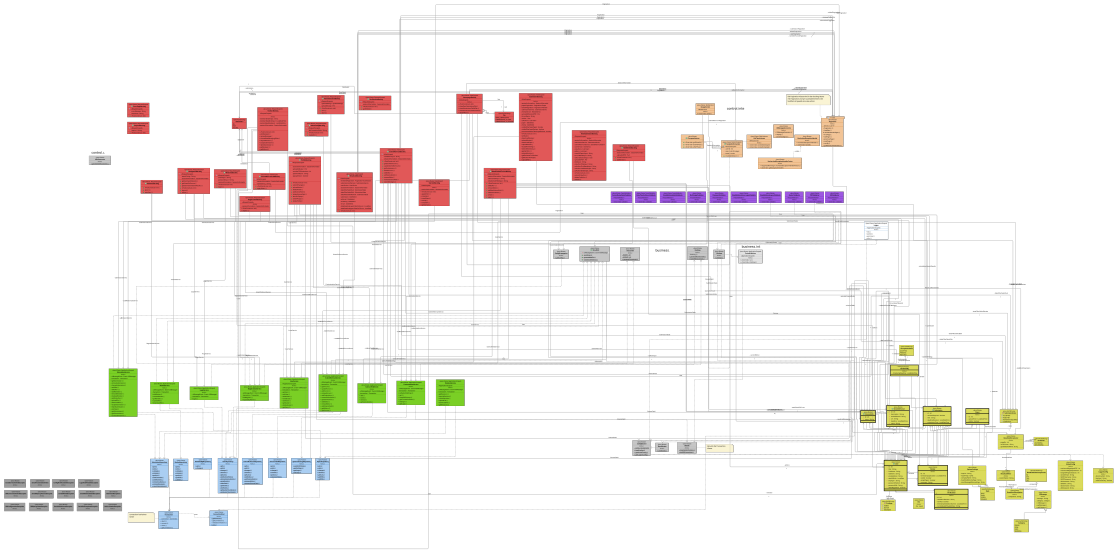
- RSA: Sebastian Vogt
- Control Schicht: Sebastian Vogt
- Business Schicht: Johannes Garstenauer
- Persistence Schicht: Sebastian Vogt
- Cross-Cutting Concerns: Sebastian Vogt

3.1 Klassendiagramm

Im folgenden Klassendiagramm wurden folgende Konventionen verwendet:

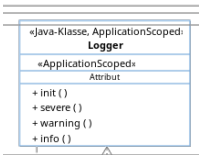
- Properties sind als private Attribute modelliert. Konstruktoren sind ausgespart.
- Referenzattribute auf andere Klassen des Diagramms sind als Pfeile dargestellt. Diese haben standardmäßig keine Getter und Setter, außer anders angegeben.
- Gestrichelte Pfeile sind *use-Pfeile*, egal ob diese mit «*use*» versehen sind oder nicht.

Weitere Informationen zur Interpretation des Diagramms finden sich in den jeweiligen Packages.



3.2 Klassenbeschreibungen

3.2.1 de.lases.global.logging



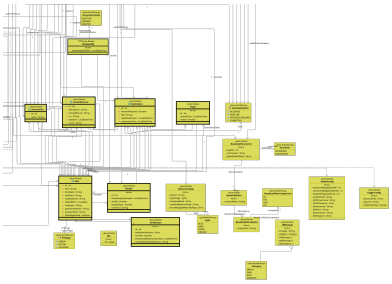
Klassenname	Beschreibung
Logging	This Logger allows logging in different log-levels.

3.2.2 de.lases.global.transport

Im folgenden Diagramm gilt für die Klassen mit dickem schwarzem Rand folgende Konvention:

- Die Zuordnungen zwischen den Klassen werden nicht durch Java Zeiger realisiert, sondern durch Speichern der Id des jeweiligen Objekts.
- Bei einer $x \to 1$ Beziehung (mit $x \in \{1, *\}$) merkt sich das Objekt auf der Linken Seite die Id des zugeordneten Objektes
- Bei einer $x \to *$ Beziehung merkt sich das Objekt auf der Linken Seite *keine* Liste der zugehörigen Ids. Zuordnungen sind in diese Richtung also nicht unmittelbar traversierbar.

Die Vollständigkeit der DTOs wurde gegen das ER-Diagramm gegengeprüft. natürlich gibt es aber auch DTOs, die nicht direkt aus dem ER-Diagramm kommen.

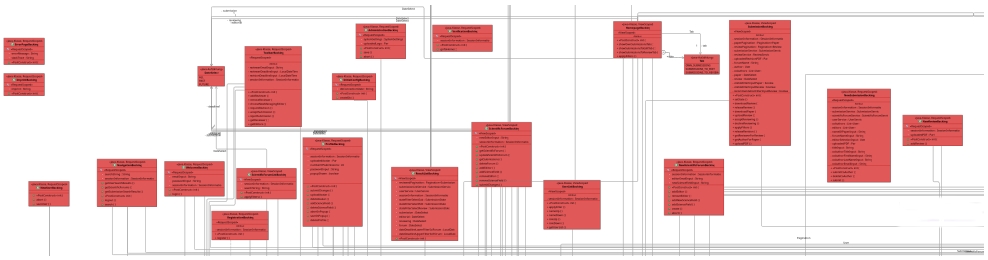


Klassenname	Beschreibung
Paper	This DTO represents a paper.
Privilege	This represents a user privilege.
Review	This DTO represents a review.
ScienceField	This DTO represents a field of Science.
ScientificForum	This DTO represents a forum
Style	This DTO represents a user interface style.
Submission	This DTO represents a submission.
SubmissionState	This DTO represents a submissions state.
SystemSettings	This DTO represents the system settings.
User	This DTO represents a user.
ResultListParameters	This class bundles parameters for requesting lists from the database. This includes page numner, sorting of the results and filtering of the results
SortOrder	A List can be sorted in ascending or descending order
ResultListFilter	A filter that can be applied on a column of a table
ResultListFilterOption	A single comparison that can be used to construct a ResultListFilter
ResultListFilterComparator	The three comparators equal to, less than or equal and greater than or equal
GlobalConfig	Holds all data that is read from the config file.
Verification	Bundles information about the verification process is somebody is newly registered or has just changed theirq email.
File	Encapsulates a file that is passed between the layers.
ReviewedBy	Stores Information about the relationship of a Submission and a User (the reviewer).
AcceptanceStatus	Holds the information wheather a requested reviewer has accepted or rejected the request or has yet to decide.
LoggerConfig	Holds configuration information about the logger.
UIMessage	This DTO holds the information necessary for constructing a *FacesMessage*. It is contained in events used for displaying a message to the user.

3.2.3 de.lases.control.backing

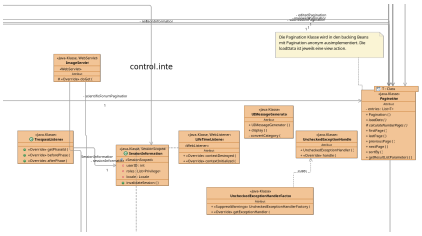
Im Diagramm dieses Pakets sind folgende Punkte zu beachten:

- Pfeile, die zur SessionInformation führen wurden aus Gründen der Übersichtlichkeit ausgespart und lediglich als Attribut dargestellt. Die SessionInformation hat natürlich trotzdem keine Getter und Setter.
- Alle backing Beans sind managed Beans.



Klassenname	Beschreibung
NewSubmissionBacking	Backing bean for the page for creating a new submission.
SubmissionBacking	Backing bean for the submission page.
NewReviewBacking	Backing bean for the page for creating adding a new review.
VerificationBacking	Backing bean for the verification page.
ToolbarBacking	Backing bean for the side toolbar.
NavigationBacking	Backing bean for the navigation bar.
WelcomeBacking	Backing bean for the welcome and login page.
RegistrationBacking	Backing bean for the registration page.
HomepageBacking	Backing bean for the homepage for logged in users.
NewUser	Backing bean for the page for adding a new user.
ScientificForumListBacking	Backing bean for the list of scientific forums.
UserListBacking	Backing bean for the list of users.
ResultListBacking	Backing bean for the search result page.
ScientificForumBacking	Backing bean for the scientific forum page.
NewScientificForum-Backing	Backing bean for the page for adding a new forum.
ProfileBacking	Backing bean for the profile page.
AdministrationBacking	Backing bean for the administration page.
FooterBacking	Backing bean for the footer template.

3.2.4 de.lases.control.internal



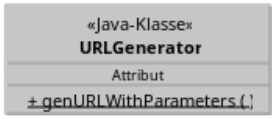
Klassenname	Beschreibung
TrespassListener	Manages user access to resources, most importantly webpages.
SessionInformation	Wraps data saved in the session.
LifeTimeListener	Takes care of system start and shutdown operations
ExceptionHandler	Handles all Exceptions that leave the business layer by either showing a faces message or an error page.
Pagination	Generic class to implement a paginated list of items of a certain type.
ImageServlet	Serves avatars and logos.
ImageServlet	Serves avatars and logos.
UIMessageGenerator	Observes all events concerning *UIMessage*s and creates *FacesMessage*s from them.
UncheckedExceptionHandler	Catches all unchecked exceptions and redirects the user to an error page.
UncheckedExceptionHandlerFactory	A factory for the *UncheckedExceptionHandler*.

3.2.5 de.lasses.control.validation

Klassenname	Beschreibung
AvatarValidation	Validates that a suitable avatar image has been uploaded.
PDFValidator	Validates that a suitable PDF file has been uploaded.
EmailAddressLayoutValidation	Validates that a given String is a valid E-Mail Address.
EmailAdressUnoccupiedValidator	Validates that a given email is not already in use within the system.
URLValidator	Validates that a given String is a valid URL.
PasswordValidator	Checks if a password meets the minimum requirements for a password.
NotEmptyValidator	Checks if a free text imput is empty.

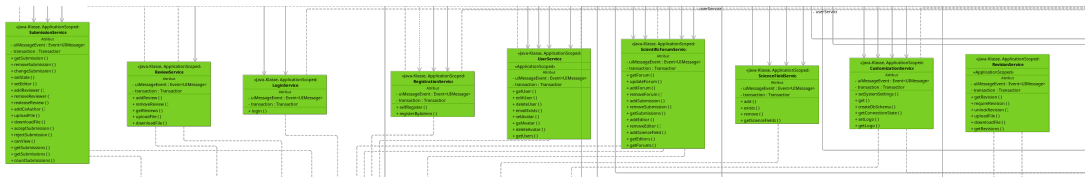
EmailAddressExistsValidator	Checks whether a given email address is a verified address in the system.
DateTimeInFutureValidator	Checks if a date input lies in the future.
ForumNameExistsValidator	Checks if a given input exists as a forum in the system.

3.2.6 de.lasses.control.util



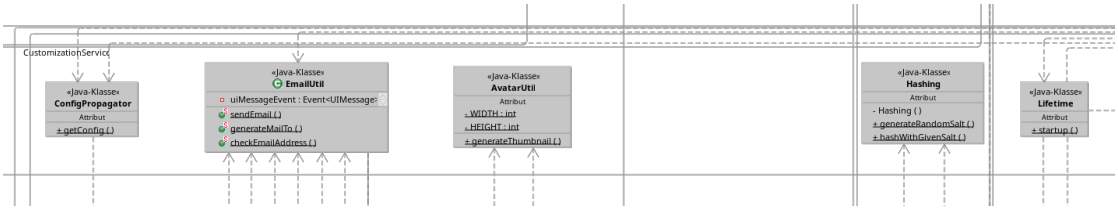
Klassenname	Beschreibung
URLGenerator	Generates URLs with URL params.

3.2.7 de.lases.business.service



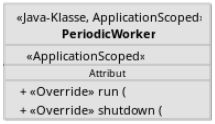
Klassenname	Beschreibung
CustomizationService	Provides methods for the manipulation of system settings.
ScienceFieldService	Provides methods for adding and removing scientific categories.
UserService	Provides methods for the manipulation and removal of users.
ScientificForumService	Provides methods for the manipulation and creation of scientific forums.
RegistrationService	Provides methods for the creation of users.
LoginService	Provides methods for the login and logout operations.
RevisionService	Provides methods for the creation and manipulation of revisions for submissions.
ReviewService	Provides methods for the delivery, creation and removal of reviews.
SubmissionService	Provides methods for the delivery, creation, removal and manipulation of submissions.

3.2.8 de.lases.business.util



Klassenname	Beschreibung
AvatarUtil	Provides support in the generation of a thumbnail from an image.
Hashing	Provides support in the hashing of passwords.
EmailUtil	Provides support in the sending of emails, creation of *mailto links* and checking of E-mail addresses.
ConfigPropagator	The purpose of this utility is to propagate the configuration data to the control layer.

3.2.9 de.lases.business.internal

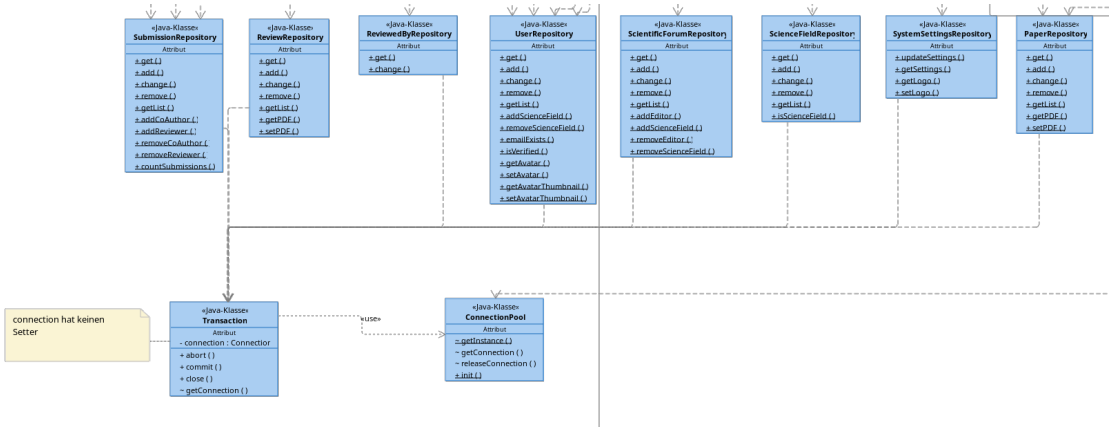


Klassenname	Beschreibung
PeriodicWorker	Takes care of periodical database cleanup jobs.

3.2.10 de.lases.persistence.repository

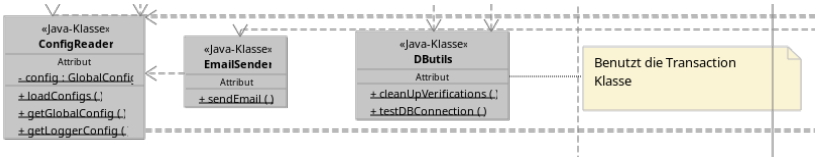
Im Folgenden Klassendiagramm gilt:

- Jedes Repository ist für genau eine Entität im ER-Diagramm bzw. für genau ein DTO zuständig.
- Die Methode *getList*() ist jeweils überladen. Falls man eine Liste von Objekten will, die mit einem bestimmten anderen Objekt *a* direkt einer Relationship stehen (siehe ER-Diagramm), dann kann man *a* an die Methode übergeben. Dies ist auch mit mehreren Relationships zugleich möglich.
- Spezifische *addObject*/*removeObject* Methoden sind von der jeweils vorhandenen *change* Methode insofern abzugrenzen, dass sie je Listen von zugeordneten Objekten bearbeiten. Die *change* Methode kann nur Attribute mit Kardinalität 1 bearbeiten. Eine Ausnahme hierbei sind natürlich die *get*/*set* Methoden für PDFs und Bilder.



Klassenname	Beschreibung
PaperRepository	This repository can get the list of papers for a submission or add new ones.
ReviewRepository	This repository can get the reviews for a certain submission and add new ones.
ScienceFieldRepository	This repository can get or add new fields of science from/to the database.
ScientificForumRepository	This repository can get a list of journals and conferences or add new ones.
SubmissionRepository	This repository allows CRUD operations on submissions.
SystemSettingsRepository	This repository can read or update the system settings.
UserRepository	This repository allows CRUD operations on users.
ReviewedByRepository	This respository allows to get and change information about the relationship of a submission and a reviewer.
ConnectionPool	Provides and manages connections to the database.

3.2.11 de.lases.persistence.util



Klassenname	Beschreibung
DButils	Provides functions for general database services.
EmailSender	Talks to the SMTP server.
ConfigReader	Reads the global configuration file from the disk.

3.2.12 de.lases.persistence.exception

Diese Exceptions werden ebenfalls von der *Businessschicht* verwendet.

Klassenname	Beschreibung
InvalidFieldsException	Hints at an invalid field in a dto given to the persistence layer.
InvalidQueryParamsException	Hints at invalid field in the ResulstListParameters object given to the persistence layer.
DBConnectionFailedException	Is thrown when the database connection cannot be established or times out.
DBConfigurationException	Is thrown when bad configuration of the database leads to a failure to establish a connection.
DBQueryFailedException	Is thrown when a query to the database fails and will probably fail again.
EmailServiceFailedException	Fires when the email service used by the application fails unrecoverably.
ConfigNotReadableException	Fires when the config file of the application can definitively not be read.
LogsNotWritableException	Is thrown when the logger cannot write log files.

Tabelle 12: Unchecked Exceptions

Klassenname	Beschreibung
EmailAddressExistsException	Is thrown when the repository tries to save an email-address to the databse that already exists in the database.
DataNotWrittenException	Is thrown when the writing of data fails but a retry has a high probability of succeeding, for example when a DataTruncationException occurs.
DataNotCompleteException	Is thrown when the reading of data yields an incomplete result but a retry has a high probability of succeeding, for example when a DataTruncation warning occurs.
NoPermissionException	Is thrown when a user does not have the correct role to complete a certain operation. This was most likely caused by a race condition where someone lost a role during completion of an operation.

NotFoundException	Is thrown when a queried object is not found in the database. In the common case this is caused by a race condition where an object was removed while being selected.
EmailTransmissionFailed	Fires whenever the transmission of an email (to a specific list of senders) fails but the email service of this system is intact.

Tabelle 13: Checked Exceptions

4 Facelets

Stefanie Gürster, Johann Schicho

Im Folgendem Abschnitt werden Abkürzungen für die verschiedenen Rollen eingeführt: **A** steht für Administrator, **AN** für einen anonymen Nutzer, **N** für einen angemeldeten Nutzer, **E** für einen Editor und **G** steht für einen Gutachter. Ist eine Funktion für alle Benutzerrollen außer dem anonymen Nutzer vorgesehen, so werden diese unter dem Begriff **Alle** zusammengefasst. Außerdem verfügt ein Administrator über alle Rechte der anderen Rollen und wird daher nur bei Schlüsselfunktionen genannt.

Labels werden in diesem Abschnitt nicht aufgelistet, sind aber angedacht. Diese sind jedoch in Form eines *outputLabels* oder eines Fontawesome-Icon vorgesehen. Des Weiteren wird *selectOneListbox* von Primefaces verwendet, um lange Dropdown-Menüs benutzerfreundlich zu gestalten.

Die Vollständigkeit der einzelnen Facelets wurde anhand der Backing Beans und dem ER-Diagramm gegengeprüft.

4.1 Templates

Johann Schicho

navigation.xhtml ist die Kopfzeile der Webanwendung. Diese bietet die Suchfunktion an und Links zu verschiedenen Listen und dem Profil.

ID	Typ	Beschreibung	Rechte
logo	graphicImage	Logo der Applikation	Alle
directTo-Home	link	Weiterleitung zur Homepage.	Alle
searchField	inputText	Suchleiste	Alle
search	commandButton	Suche ausführen.	Alle
userList-Link	link	Link zur Übersichtsseite aller Nutzer	E
forumList-Link	link	Link zur Übersichtsseite alle Journale und Konferenzen	Alle
logoutBut-ton	commandButton	Loggt den Nutzer aus dem System aus und leitet zur Loginseite weiter	Alle
profileLink	link	Link zur Profilübersicht	N

main.xhtml ist das Template, welches den Inhalt der Seiten zwischen Kopf- und Fußzeile einbettet.

ID	Typ	Beschreibung	Rechte
navigation-Bar	include	Kopfzeile	Alle
mainCon-tent	insert	Seiteninhalt	Alle
footerBar	include	Fußzeile	Alle

footer.xhtml ist die Fußzeile der Webanwendung und für alle sichtbar.

ID	Typ	Beschreibung	Rechte
imprint-Link	link	Link zur Seite des Impressums	Alle
language	outputText	Anzeige der eingestellten Sprache.	Alle

toolbar.xhtml ist die Seitenleiste für Editoren und Administratoren auf der Einreichungsübersicht. Hier wird die Einreichung verwaltet.

ID	Typ	Beschreibung	Rechte
addRe-viewerField	inputText	Eingabefeld zur Angabe einer E-Mail-Adresse	E
addDead-lineRe-viewer	inputText	Eingabefeld zur Angabe einer Deadline.	E
addRe-viewerBtn	commandButton	Knopf zum Hinzufügen des Gutachters	E
review-erTable	dataTable	Liste der Gutachter	E
removeRe-viewer	commandButton	Entferne einen bestimmten Gutachter	E
selectEdi-tor	selectOneMenu	Auswahländerung des verwaltenden Editors	E
saveEditor	commandButton	Speichern der Änderung	E
showEditor	outputText	Anzeige des aktuellen Editors.	E
revisi-onDeadline	inputText	Eingabe zur Angabe einer Deadline.	E
require-Revisi-onBtn	commandButton	Fordert den Einreicher auf, seine Einreichung zu überarbeiten	E
acceptSub-missionBtn	commandButton	Akzeptiere die Einreichung	E
rejectSub-missionBtn	commandButton	Lehne die Einreichung ab	E

pagination.xhtml ist ein Composite Component zur Paginierung von Listen.

Dazu wird folgendes Attribut übergeben:

- **pagination** Implementation der abstrakten Klasse *Pagination.java*.

ID	Typ	Beschreibung	Rechte
dataTable	dataTable	Hier werden die Spalten eingefügt. Durch Klicken auf die Spaltennamen werden die Einträge sortiert. Zusätzlich kann durch ein Textfeld die Liste gefiltert werden.	Alle
apply	commandButton	Mit diesem Knopf können gewählte Sortierung und Filterung angewandt werden.	Alle
navButtons	panelGrid	Hier befinden sich die Knöpfe zur Navigation zwischen den Seiten der Paginierungen.	Alle
firstPage	commandButton	Zurück zur ersten Seite.	Alle
prevPage	commandButton	Eine Seite zurück.	Alle
currentPage	outputText	Nummer der aktuellen Seite.	Alle
nextPage	commandButton	Eine Seite weiter.	Alle
lastPage	commandButton	Zur letzten Seite.	Alle

sortfiltercolumn.xhtml ist ein Composite Component, das eine sortierbare und filterbare Spalte einer Liste modelliert.

Dazu werden folgende Attribute übergeben:

- **pagination** Implementation der abstrakten Klasse *Pagination.java*.
- **columnNamei18n** Anzeigename im Spaltenkopf.
- **columnIdent** *String Identifier* der Spalte.
- **columnNo** Spaltennummer, Null indiziert.

ID	Typ	Beschreibung	Rechte
columnName	commandLink	Durch Klicken auf den Spaltennamen wird nach dieser Spalte aufsteigend, bzw. absteigend sortiert.	Alle
columnFilter	inputText	Hier kann ein Wert eingegeben werden, nach dem die Spalte dann gefiltert wird.	Alle

4.2 Seiten

Stefanie Gürster

Für jedes Facelet ist ein Bereich, für Anzeigen von Fehlern oder positiven Benachrich-

tigungen, vom Typ *messages* angedacht. Um Duplikation zu vermeiden, wird dies als Konvention für alle Facelets festgelegt.

Außerdem wird auf jeder Seite, welche eine Liste enthält, eine *viewAction* implementiert. Diese lädt die Daten aus der Datenbank bei der ersten Betrachtung der Seite.

4.2.1 Anonymer Nutzer

welcome.xhtml Auf der Login- bzw. Welcome-Seite wird das System vorgestellt. Zusätzlich gibt es ein Login-Formular zur Anmeldung im System. Für nicht registrierte Nutzer wird man über einen gegebenen Link zu *registration.xhtml* weitergeleitet.

ID	Typ	Beschreibung	Rechte
welcomeHeading	outputText	Überschrift der Welcomepage.	AN
welcomeText	outputText	Bewerben der Anwendung mithilfe einer Kurzbeschreibung.	AN
email	inputText	Textfeld für E-Mail Eingabe.	AN
password	inputSecret	Textfeld für Passwort Eingabe.	AN
login	commandButton	Ausführen des Login-Prozesses.	AN
register	link	Weiterleitung zur Registrierung.	AN

registration.xhtml Seite zur Registrierung anonymer Nutzer.

ID	Typ	Beschreibung	Rechte
title	inputText	Angabe eines Titels.	AN
firstName	inputText	Angabe des Vornamens.	AN
name	inputText	Angabe des Nachnamen.	AN
password	inputSecret	Angabe eines Passwortes.	AN
email	inputText	Angabe einer validen E-Mail.	AN
register	commandButton	Ausführen des Registrations-Prozesses.	AN
welcomepage	link	Weiterleitung zur Login Seite.	AN

verification.xhtml Wird bei erfolgreicher Verifikation der E-Mail angezeigt.

ID	Typ	Beschreibung	Rechte
successText	outputText	Text zur erfolgreichen Verifizierung der E-Mail	AN
goToHome	commandButton	Button zur Weiterleitung auf die Homepage.	AN

errorPage.xhtml Auf diese Seite wird navigiert, wenn auf eine nicht existierende URL oder auf eine URL zugegriffen wird, auf die man keine Zugriffsrechte hat.

ID	Typ	Beschreibung	Rechte
errorMessage	outputText	Anzeige einer Fehlermeldung.	Alle,AN
stackTrace	outputText	Stacktrace für den ”Productive Mode”.	Development

imprint.xhtml Das Impressum gibt die vom Administrator angegebenen Kontaktdaten des Betreibers wieder.

ID	Typ	Beschreibung	Rechte
imprintHeading	outputText	Überschrift der Ansicht	A
imprint	outputText	Impressum des Betreibers.	A

4.2.2 Angemeldeter Nutzer

homepage.xhtml Startseite die den Überblick über alle Einreichungen, aufgeteilt in Reiter, beinhaltet.

ID	Typ	Beschreibung	Rechte
yourSubmission	commandLink	Reiter zum Anzeigen der eigenen Einreichungen.	Alle
yourReviews	commandLink	Reiter zum Anzeigen der zu begutachtenden Einreichungen	G
editorialOverview	commandLink	Reiter zum Anzeigen aller Einreichungen, welche man editiert.	E
stateSelectEditor	selectOneMenu	Filtern des Status von Einreichungen	E
stateSelectReview	selectOneMenu	Filtern des Status von Einreichungen	G
stateSelect	selectOneMenu	Filtern des Status von Einreichungen	Alle
dateSelectEdit	selectOneMenu	Auswahl der Filtermöglichkeit eines Datums.	E
dateSelectReview	selectOneMenu	Auswahl der Filtermöglichkeit eines Datums.	G
dateSelect	selectOneMenu	Auswahl der Filtermöglichkeit eines Datums.	Alle
submissionPagination	pagination	Liste aller eignen Einreichungen.	Alle
reviewPagination	pagination	Liste aller Gutachten.	G
editorPagination	pagination	Liste aller Einreichungen für einen Editor.	E

Die einzelnen Tabellen *paperPagination*, *reviewPagination* und *editorPagination* sind nach den gleichen Punkten gegliedert: Titel des Papers, Datum, Deadline und Status der Einreichung und Name des zugehörigen Forums.

ID	Typ	Beschreibung	Rechte
title	Link	Titel des Papers und Weiterleitung zur Einreichung.	Alle
date	outputText	Datum der Einreichung.	Alle
deadline	outputText	Deadline.	Alle
state	outputText	Status der Einreichung.	Alle
forum	Link	Name des zugehörigen Forums und Weiterleitung zum Forum.	Alle

submission.xhtml ist die Übersichtsseite einer Abgabe. Hier werden alle mit der Einreichung verbundenen Aktivitäten abgebildet.

ID	Typ	Beschreibung	Rechte
toolBar	include	Seitenleiste mit Tools.	E
uploadRevision	inputFile	Upload einer Revision.	N
upload	commandButton	Ausführen des Uploads.	N
newReview	commandLink	Weiterleitung zur Einreichung eines neuen Gutachtens.	G
acceptReviewing	commandButton	Annahme der Rolle eines Gutachters für diese Einreichung.	G
declineReviewing	commandButton	Ablehnen der Rolle eines Gutachters für diese Einreichung.	G
title	outputText	Titel des Papers.	Alle
forum	Link	Name des Forums und Weiterleitung zur Forumsübersicht.	Alle
author	Link	Name des Autors und Weiterleitung zum Profil.	E
email	Link	E-Mail-Adressen der Co-Autoren und Mailto-Link.	E
dateSelect	selectOneMenu	Auswahl der Filtermöglichkeit eines Datums.	Alle
dateSelectReview	selectOneMenu	Auswahl der Filtermöglichkeit eines Datums.	Alle
reviewState	selectOneMenu	Filtern der Gutachten nach ihrem Status.	E,G
recommendation	selectOneMenu	Auswahl der Art von Empfehlungen.	Alle
paperPagination	pagination	Liste aller Versionen	Alle
reviewPagination	pagination	Liste aller/der eigenen Gutachten.	E,G

paperPagination enthält die Spalten: Version, Datum, Deadline, Sichtbarkeit, Status und Download. Die Elemente der Liste, der unterschiedlichen Versionen der Einrichtung, sieht wie folgt aus:

ID	Typ	Beschreibung	Rechte
version	outputText	Versionsnummer der Einreichung	Alle
date	outputText	Datum der Einreichung	Alle
deadline	outputText	Deadline der Revision	Alle
unlockRevision	commandButton	Freischalten einer Revision.	E
state	radioButton	Status der Einreichung	Alle
pdf	commandButton	Download der Einreichung	Alle

reviewPagination enthält die Spalten: Version, Gutachter, Datum, Deadline, Status, Empfehlung, Kommentar und Download. Ist ein Gutachten freigeschaltet, so erhält auch ein Nutzer Leserechte. Die Elemente der Liste von Gutachten sind folgende:

ID	Typ	Beschreibung	Rechte
version	outputText	Versionsnummer des Gutachtens.	E,G
reviewer	outputText	Namen des Gutachters.	E,G
date	outputText	Datum des Gutachtens.	E,G
deadline	outputText	Deadline des Gutachtens.	E,G
state	commandButton	Freigabe des Gutachtens	E
stateText	outputText	Freigabestatus des Gutachtens	G
recommendation	checkbox	Empfehlung eines Gutachters	E,G
comment	outputText	Kommentar des Gutachters.	E,G
pdf	commandButton	Download der Einreichung.	E,G

newSubmission.xhtml Hier können neue Paper eingereicht werden.

ID	Typ	Beschreibung	Rechte
submissionName	inputText	Name des abzugebenden Papers.	N
forumName	inputText	Name des Forums, bei welchem abgegeben wird.	N
editorSearch	selectOneMenu	Angabe eines Editors	N
pfdUpload	inputFile	Angabe der Abgabedatei.	N
titel	inputText	Angabe eines Titels.	N
coAuthorFirstName	inputText	Angabe des Vornamen eines Co-Autors.	N
coAuthorLastName	inputText	Angabe des Namen eines Co-Autors.	N
coAuthorEMail	inputText	Angabe der E-Mail eines Co-Autors.	N

submitCo-Author	commandButton	Ausgabe des Co-Autors in einer Liste.	N
coAuthor-List	outputText	Anzeige aller Co-Autoren in einer Liste.	N
deleteCo-Author	commandButton	Löschen des zugehörigen Co-Authors.	N
submit	commandButton	Ausführen des Abgabe-Prozesses.	N

resultList.xhtml zeigt alle Ergebnisse einer Suche an.

Wird nach den eigenen Einreichungen, nach denen, die man begutachtet oder nach Einreichungen in eigener editorialer Verantwortung gesucht, so erscheinen die Listen im gleichen Schema wie auch auf der Homepage. Ergibt die Suche eine Liste von User, so wird diese wie in der Userliste für den Administrator und dem Editor angezeigt. Ist das Ziel der Suche, wissenschaftliche Foren zu finden, so erhält man bei passender Eingabe eine Liste im selben Format wie diejenige auf der Seite der Forumsliste.

scientificForumList.xhtml Hier erhalten die Nutzer eine Übersicht über alle wissenschaftliche Foren.

ID	Typ	Beschreibung	Rechte
dateSelect	selectOneMenu	Auswahl der Filtermöglichkeit eines Datums.	Alle
forumsPagination	pagination	Liste aller Foren im System.	Alle

Die Liste aller Foren im System wird in folgende Bereiche unterteilt: Name des Forums und Deadline. Dabei beinhaltet jede Zeile dieser Liste folgende Elemente:

ID	Typ	Beschreibung	Rechte
name	link	Name des Forums.	Alle
deadline	outputText	Abgabefrist des jeweiligen Forums.	Alle

profile.xhtml Die Profilseite eines angemeldeten Nutzers ist nur vom Nutzer selbst und vom Administrator editierbar. Ein Editor besitzt nur Leserechte. Alle zuvor gespeicherten Angaben werden in den jeweiligen Feldern angezeigt.

ID	Typ	Beschreibung	Rechte
avatar	graphicImage	Aktueller Avatar.	N,E
newAvatar	inputFile	Hochladen eines neuen Avatars.	N,E
submitAvatar	commandButton	Lade neuen Avatar hoch.	N
deleteAvatar	commandButton	Löschen des Avatars.	N
role	outputText	Angabe der Rolle eines Nutzers	N,E
isAdmin	checkbox	Zuteilung der Rolle des Administrators.	A

title	inputText	Titel des Nutzenden.	N,E
firstName	inputText	Vorname des Nutzenden.	N,E
name	inputText	Nachname des Nutzenden.	N,E
password	inputSecret	Leeres Eingabefeld für ein neues Passwort.	N
email	inputText	E-Mail des Nutzenden.	N,E
submissionNumber	outputText	Anzahl der eigenen Einreichungen	N,E
employer	inputText	Angabe des Arbeitgebers	N,E
scienceField	outputText	Liste aller Fachgebiete	N,E
addScienceField	selectOneListbox	Spezialgebiete des Nutzers	N
deleteScienceField	commandButton	Löschen des ausgewählten Fachgebietes.	N
addFieldButton	commandButton	Füge Fachgebiet zur Liste hinzu.	N
dateOfBirth	inputText	Angabe des Geburtstages	N
save	commandButton	Daten werden persistiert.	N
delete	commandButton	Nutzer und alle damit verbundenen Daten werden gelöscht.	N

Sollte ein Administrator einem anderen Nutzer die Rolle des Administrators zuweisen oder entziehen, so erscheint beim Auslösen des Save-Buttons ein Popup-Dialog. In diesem Fenster wird der Administrator angewiesen, die Änderung mit seinem Passwort zu bestätigen.

ID	**Typ**	**Beschreibung**	**Rechte**
password	inputSecret	Angabe eines Passworts.	A
reference	outputText	Hinweis zur Veränderung der Adminrolle.	A
abort	commandButton	Abbruch der Änderung.	A
save	commandButton	Speichern aller Änderungen.	A

scientificForum.xhtml Die Ansicht eines Forums dient zur Ausgabe von Informationen über die jeweilige Konferenz oder das jeweilige Journal. Das Anzeigen und Filtern der Tabellen *editorialPagination*, *reviewPagination* und *submissionPagination* wird genauso gehandhabt, wie auf *homepage.xhtml*. Ausnahme dabei ist die Spalte Forum, welche hier nicht angezeigt wird.

ID	**Typ**	**Beschreibung**	**Rechte**
forumName	inputText	Name des Forums.	Alle
editor	outputText	Liste der verantwortliche Editoren.	Alle
newEditor	inputText	Angabe der E-Mail eines Editors.	E

addEditor	commandButton	Füge Editor zur Liste hinzu.	E
removeEditor	commandButton	Lösche ausgewählten Editor.	E
deadline	inputText	Deadline des Forums.	Alle
description	inputTextArea	Kurzbeschreibung des Forums.	Alle
url	outputLink	Link zur Konferenz oder zum Journal.	Alle
changeUrl	inputText	Angabe einer neuen URL.	E
reviewInstructions	inputTextArea	Anleitung für eine Begutachtung.	G,E
scienceField	outputText	Fachgebiet des Forums.	Alle
newScienceField	selectOneListbox	Hinzufügen eines neuen Fachgebietes.	E
addScienceField	commandButton	Speichern des neuen Fachgebietes.	E
deleteScienceField	commandButton	Löschen eines Fachgebietes.	E
save	commandLink	Speichere Veränderungen.	E
editorialPagination	pagination	Paginierte Liste aller Einreichungen, die der Editor verwaltet.	E
reviewPagination	pagination	Paginierte Liste aller Einreichungen, die der Gutachter bearbeitet.	E,G
submissionPagination	pagination	Paginierte Liste aller Einreichungen, die der Nutzer eingereicht hat.	Alle

4.2.3 Gutachter

newReview.xhtml Möchte ein Gutachter eines Papers seine Beurteilung abgeben, so ist dies auf dieser Seite möglich.

ID	**Typ**	**Beschreibung**	**Rechte**
versionNumber	outputText	Versionsnummer des eingereichten Papers, zu welchem der Gutachter ein Gutachten abgeben kann.	G
recommendation	checkbox	Empfehlung des Gutachters ein Paper zu akzeptieren.	G
comment	inputTextarea	Kommentar eines Gutachters	G
reviewPdf	inputFile	Einzureichendes Gutachten.	G
submitReview	commandButton	Gutachten einreichen.	G

4.2.4 Editor

userList.xhtml Editoren und Administratoren erhalten hier einen Überblick über alle Nutzer. Die Seiten sind paginiert.

ID	Typ	Beschreibung	Rechte
user-pagination	pagination	Die Tabelle enthält eine Paginierung.	E

Für die Benutzerliste sind die folgenden Spalten vorgesehen, wobei diese in die Bereiche: Nutzerrolle, Name, E-Mail-Adresse, Arbeitgeber und Fachgebiete aufgegliedert ist.

ID	Typ	Beschreibung	Rechte
role	outputText	Rolle des Nutzers	E
name	link	Name des Nutzers. Link mit URL-Parameter zur Weiterleitung auf die Profilseite des Nutzers.	E
email	link	Mailto-Link mit E-Mail-Adresse des Nutzers.	E
employer	outputText	Arbeitgeber des Nutzers.	E

4.2.5 Admin

initialConfig.xhtml Auf dieser Seite landet man beim ersten Systemstart. Sie ist nur für Administratoren zugänglich. Hier kann der Administrator die Datenbankschemata erstellen lassen.

ID	Typ	Beschreibung	Rechte
dbConnectionState	outputText	Information über die Verbindung mit der Datenbank	A
createDbBtn	commandButton	Erstellt die Datenbankschemata	A

newUser.xhtml Der Administrator kann hier einen neuen Nutzer anlegen.

ID	Typ	Beschreibung	Rechte
password	inputSecret	Angabe des Passworts.	A
firstName	inputText	Angabe des Vornamen.	A
name	inputText	Angabe des Nachnamen.	A
titel	inputText	Angabe eines Titels.	A
email	inputText	Angabe einer E-Mail-Adresse.	A
isAdmin	checkbox	Zuordnung einer Administratorenrolle.	A
abort	commandButton	Abbruch des Vorgangs: Nutzererstellen.	A
save	commandButton	Speichern des neuen Nutzers.	A

administration.xhtml Hier kann ein Administrator Konfigurationen vornehmen.

ID	Typ	Beschreibung	Rechte

selectTheme	selectOneMenu	Auswahl eines Farbthemas.	A
logo	graphicImage	Logo des Systems.	A
changeLogo	inputFile	Hochladen eines neuen Logos.	A
welcomeHeading	inputText	Angabe einer Überschrift auf der *welcomepage*	A
welcomeText	inputTextArea	Angabe eines Textes auf der *welcomepage*	A
institution	inputText	Angabe des Namens der Einrichtung.	A
imprint	inputTextArea	Angabe des Impressums der Einrichtung.	A
abort	commandButton	Abbruch des Änderungsvorgangs.	A
save	commandButton	Speichern der Änderungen.	A
newUser	link	Weiterleitung zu *newUser.xhtml*.	A
newScientificForum	link	Weiterleitung zu *newScientificForum.xhtml*.	A

newScientificForum.xhtml Auf der Seite zum Erstellen eines wissenschaftlichen Forums kann ein Administrator dessen wesentlichen Daten festlegen.

ID	**Typ**	**Beschreibung**	**Rechte**
forumName	inputText	Angabe des Name des Forums.	A
editors	outputText	Liste der Editoren	A
emailEditor	inputText	Angabe der E-Mail-Adresse eines Editor.	A
addEditor	commandButton	Hinzufügen eines Editors in eine Liste.	A
deleteEditor	commandButton	Löschen des zugehörigen Editors von der Liste.	A
deadline	inputText	Hinzufügen einer Deadline des Forums.	A
description	inputTextArea	Angabe einer Kurzbeschreibung.	A
url	inputText	Link zur Konferenz oder zum Journal.	A
reviewInstructions	inputTextArea	Angabe einer Anleitung für eine Begutachtung.	A
scienceFieldList	outputText	Liste der ausgewählten Fachgebiete.	A
scienceField	selectOneListbox	Auswahl von Fachgebieten.	A
newScienceField	inputText	Hinzufügen neuer Fachgebiete.	A

addSci-enceField	commandButton	Ausführen der Hinzufüge-Aktion.	A
save	commandButton	Speichern des neuen Forums.	A
abort	commandButton	Abbruch des Erstellungsprozesses.	A

5 Systemfunktionen

Thomas Kirz

5.1 Logging

Die Java Logging API wird benutzt, um bei bestimmten Ereignissen Meldungen in eine Logdatei zu schreiben. Dabei kann man verschiedene *Loglevel* einstellen, um nur Meldungen mit bestimmter Wichtigkeit zu speichern. Es gibt folgende Loglevel, wobei jedes Level jeweils die Nachrichten des vorherigen Levels beinhaltet:

`SEVERE` Sehr wichtige Ereignisse, die eine reguläre Benutzung verhindern und deren Auswirkungen auch für Nutzer verständlich sind.

`WARNING` Wichtige Ereignisse, die zu Problemen führen können und für Administratoren und Endnutzer relevant sind.

`INFO` Für den Administrator relevante Meldungen im normalen Betrieb. Diese sind auch fürs Entwickeln und Testen hilfreich.

Nachrichten können geloggt werden durch Aufruf der `severe()`,`warning()` oder `info()` Methoden in `de.lases.global.logging`.

5.2 Sicherheit

Für die häufigsten Sicherheitsschwachstellen bei Webanwendungen, die für LasEs relevant sind, wird im Folgenden erklärt, wie ausnutzende Angriffe verhindert werden.

Fehlerhafte Zugangskontrolle Um unerlaubten Zugriff auf Seiten, Informationen und Aktionen zu verhindern, werden Zugriffe auf nicht-öffentliche Ressourcen (das ist alles außer Start-, Registrierungs- und Verifikationsseite) standardmäßig zuerst abgelehnt. Die Klasse `TrespassListener` entscheidet dann, ob der Nutzer die nötigen Rechte hat und akzeptiert eine Anfrage nur dann, sonst wird auf eine Fehler-Seite weitergeleitet. Dazu wird über die angefragte URL das zugehörige Facelet bestimmt und geprüft, ob darauf mit der Rolle des Nutzers zugegriffen werden darf. Auf die Rolle kann der `TrespassListener` über die SessionInformation zugreifen. Der Zugriff hängt darüber hinaus auch von der Beziehung des Nutzers zur angefragten Ressource ab (ein normaler Nutzer kann z.B. nur sein *eigenes* Profil oder eine *eigene* Einreichung sehen). Deshalb prüft für die relevanten Seiten das jeweilige Backing Bean diese Beziehung und leitet ggf. auf eine Fehlerseite weiter.

Kryptographie & Vertraulichkeit sensibler Daten Alle Anfragen und Antworten werden mit dem HTTPS-Protokoll mit TLS-Verschlüsselung durchgeführt. Dafür wird der Tomcat-Server, der diese Technologie unterstützt, entsprechend konfiguriert. Passwörter werden mit der modernen und sicheren PBKDF2-Funktion gehasht.

Dabei wird für jedes Passwort ein eigener zufällig generierter Salt benutzt. Eingegebene Passwörter werden so früh wie möglich gehasht, damit die Klartextpasswörter eine minimale Lebenszeit auf dem Server haben. Für das Generieren eins Salts und das Hashing bietet die Klasse `de.lases.business.util` Methoden an. Sie verwendet die robuste `javax.crypto`-Bibliothek.

Injection & Cross-site-Scripting Da JSF components für die HTML-Ausgabe verwendet werden, bereinigt JSF alle Nutzereingaben und macht Cross-site-Scripting unmöglich.
Für SQL-Code werden `PreparedStatements` benutzt, um SQL-Injections zu verhindern.

Session Hijacking In der `web.xml`-Datei ist ein Timeout für Sessions konfigurierbar. Dieser wirkt Session Hijacking sowie unerwünschten physischen Zugriffen auf den Browser des Clients entgegen. JSF setzt für das Session-Cookie standardmäßig das `http-only`-Flag, damit kann darauf nicht mit JavaScript zugegriffen werden und Hijacking wird erschwert. Nach jeder Anmeldung wird die Session verworfen (mit `ExternalContext#invalidateSession`) und neu generiert. Nach einer Authentifizierung bleibt also keine alten Session bestehen und *Session-Fixation*-Attacken werden verhindert.

5.3 Asynchrone Threads

Für das periodische Aufräumen von nicht mehr benötigten Datenbankeinträgen (abgelaufene E-Mail-Verifikationslinks) gibt es einen eigenen Thread `PeriodicWorker`.

5.4 Systemstart und -stopp

Die Klasse `LifetimeListener` aus dem Paket `control.internal` implementiert die Methode `ServletContextListener#contextDestroyed`, welche aufgerufen wird, sobald das Servlet bereit ist.
Diese ruft dann die `startup()`-Methode in `business.internal.Lifetime` auf, welche das Logging initialisiert und die Konfigurationsdatei liest.
Außerdem wird der `PeriodicWorker`-Thread aus Kapitel 5.3 gestartet und im `ConnectionPool` `init()` aufgerufen, um die Datenbankverbindungen aufzubauen.

Der Server kann über den *shutdown port* heruntergefahren werden. Dann, aber auch wenn der Server unerwartet gestoppt wird, werden `ShutdownHooks` ausgeführt.
Einer wird von `Lifetime` bei der Ausführung von `init()` hinzugefügt zum Beenden des `PeriodicWorker`-Threads. In der `init()`-Methode von `ConnectionPool` wird einer hinzugefügt für das Schließen der Datenbankverbindungen. Damit wird ein geordneter Shutdown sichergestellt.

5.5 Lokalisierung

Ausgabestrings werden in `.properties`-Dateien gespeichert, getrennt für Deutsch und Englisch und diese bilden `ResourceBundles`. Die Bundles und die verfügbaren Sprachen werden in `faces-config.xml` registiert. Dadurch ist es möglich, von Facelets aus mit der Expression Language auf die Strings zuzugreifen und von den Backing Beans mit `Application#getResourceBundle(FacesContext c, String name)`.

Um Standard-JSF-Meldungen (z.B. bei vorgegebenen Validatoen) zu überschreiben, werden `MessageBundles` verwendet. Welche Sprache verwendet wird, hängt vom Header-Element `Accept-Language` der HTTP-Anfrage ab, falls die gewünschte Sprache nicht verfügbar ist, wird Englisch als Standardsprache verwendet. Die Sprache wird von JSF automatisch ausgewählt.

5.6 Transaktionsverwaltung

Die Klasse `Transaction` im Paket `persistence.repository` repräsentiert eine Datenbanktransaktion. Sie kann mit `startTransaction()` und `stopTransaction()` gestartet bzw. beendet werden. Alle Datenbankzugriffe dazwischen, bei denen der `persistence`-Schicht diese `Transaction` übergeben wurde, werden von der Datenbank als atomare Operation durchgeführt.

6 Datenfluss

Thomas Kirz

6.1 Hochladen einer Revision

Als Beispiel für einen Datenfluss wird ein in Abbildung 2 Sequenzdiagramm zu folgendem Hochladen betrachtet: Ein angemeldeter Nutzer ruft die Seite einer eigenen Einreichung auf, für die eine Revision angefordert wurde. Dort fügt er eine Revision, also ein neues Paper hinzu.

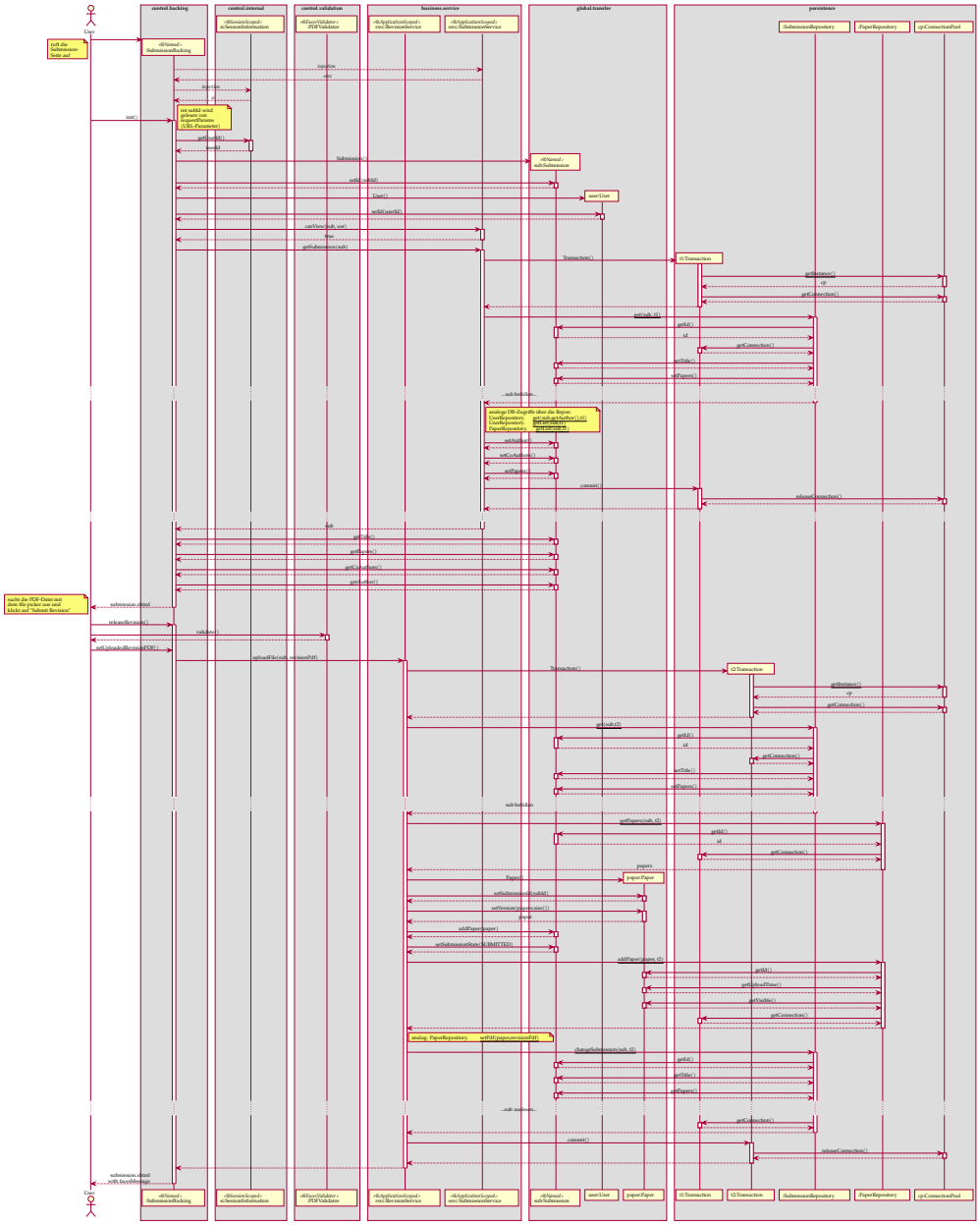


Abbildung 2: Sequenzdiagramm zur Ablehnung einer Einreichung

6.2 Ablehnung einer Einreichung

Abbildung 3 zeigt die Interaktionen zwischen den Klassen beim Ablehnen einer Einreichung, die Verbindung zum Datenbankserver schlägt aber fehl und die Einreichung kann nicht verändert werden. Hier ruft ein Editor die Seite einer Einreichung auf und klickt auf den Button zum Ablehnen. Beim Datenbankzugriff für die Aktualisierung des Status passiert ein Timeout. Da dies ein fataler Fehler ist, wird der Nutzer mithilfe des ExceptionHandlers auf eine Fehlerseite weitergeleitet.

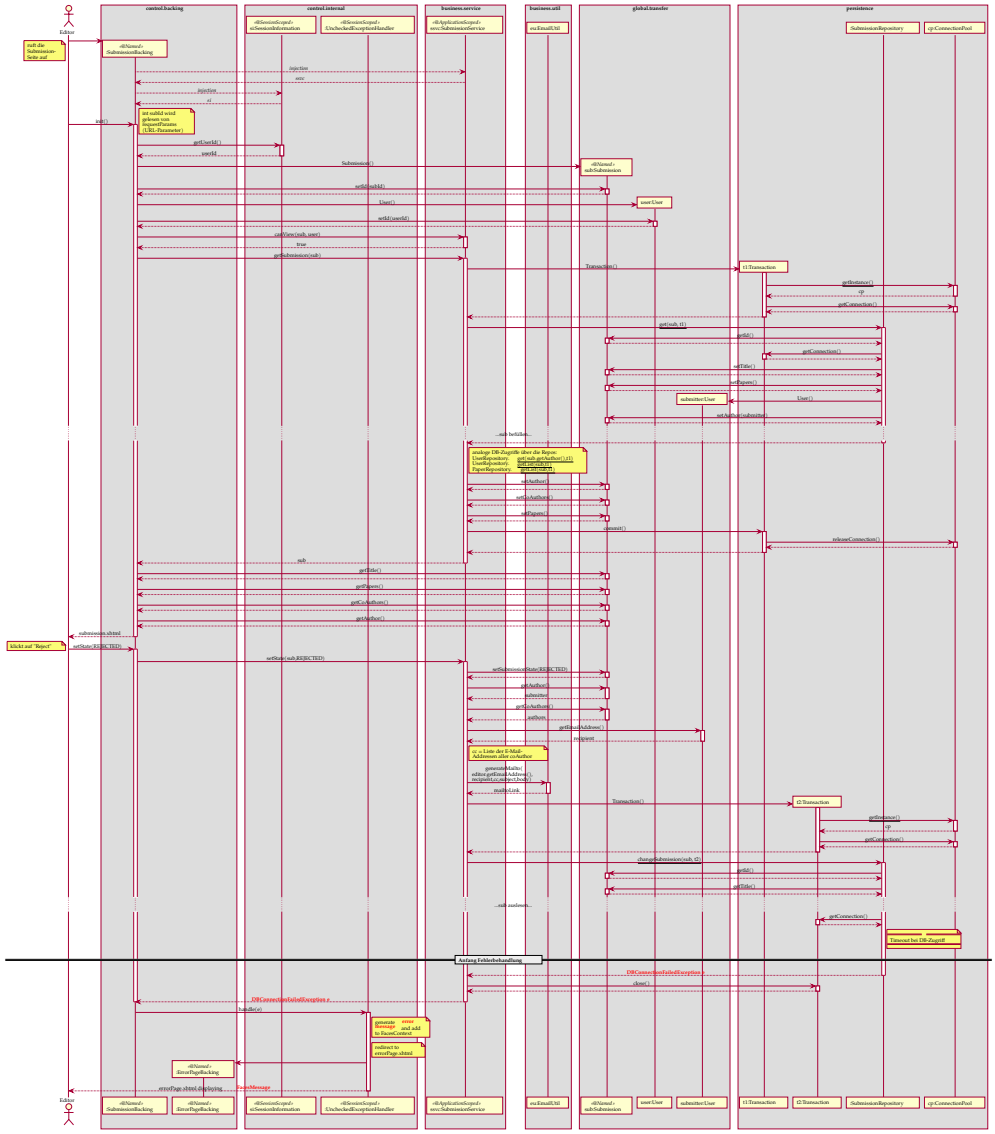


Abbildung 3: Sequenzdiagramm zur Ablehnung einer Einreichung

7 ER-Modell

Johann Schicho

Im Entity-Relationship-Diagramm in Abbildung 4 sieht man die gesamte persistente Datenstruktur von LasEs. Die Kardinalität der *Relationships* gibt die ganzzahligen Abhängigkeiten zwischen der einzelnen *Entities* an. Die *is-A* Spezialisierungen modellieren die unterschiedlichen Subtypen einer Entität. Ein im Diagramm doppelt umrandetes Entity ist ein *schwaches* Entity. Fette unterstrichene Attribute sind die Primärschlüssel einer Relation, kursive unterstrichene Attribute sind *Uniques* in der Relation.

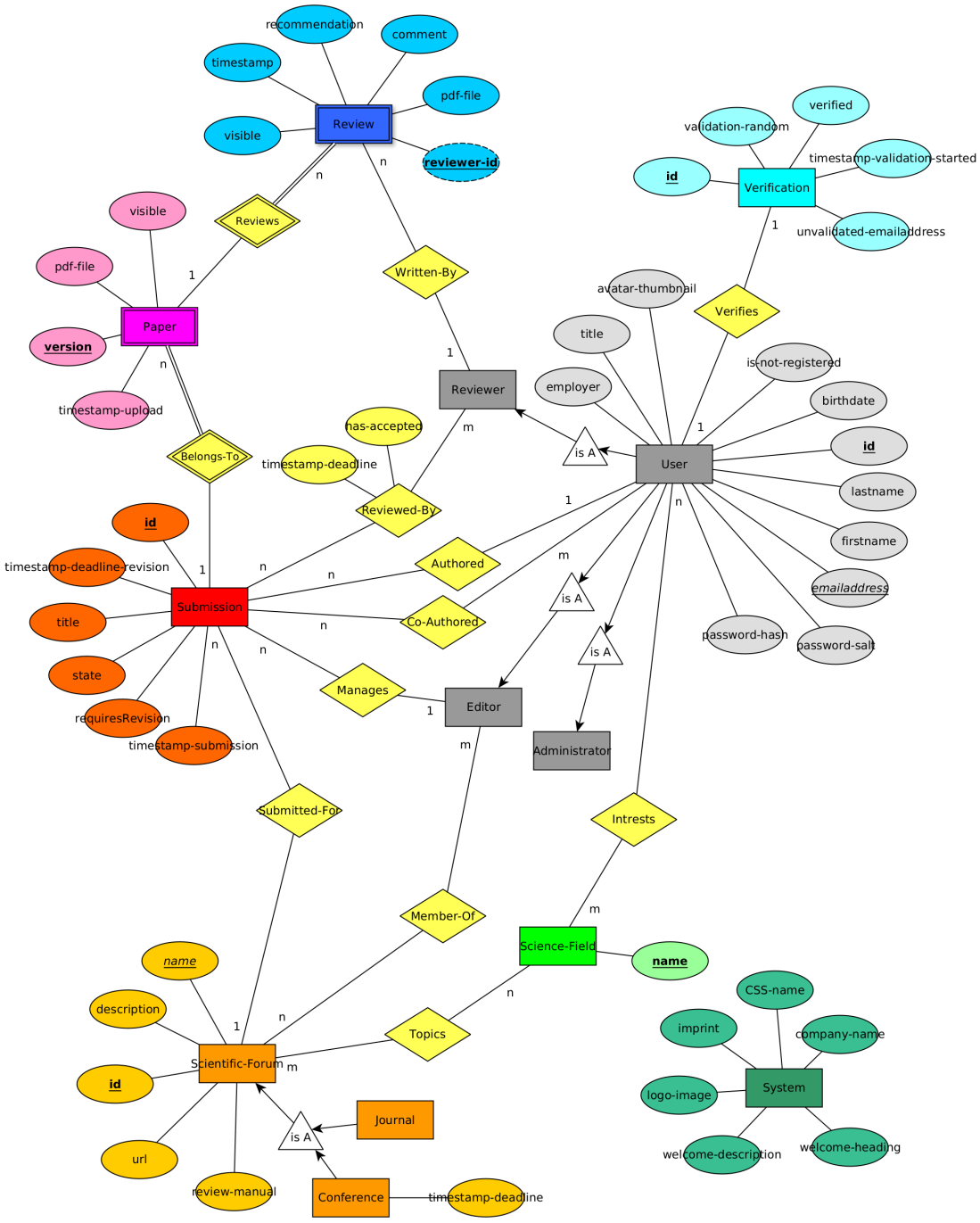


Abbildung 4: Übersicht über die Zusammenhänge zwischen den Entitys

7.1 Entities

In Abbildung 4 werden die einzelnen Entities nochmal im Detail beschrieben. Mit der Notation /DXXX/ wird sich auf die im Pflichtenheft aufgeführten Produktdaten bezogen.

User Ein Nutzer kann verschiedene Rollen wahrnehmen, diese werden durch die *is-A*-Spezifizierung modelliert. Die Attribute entsprechen den in /D010/ und /DW015/ angegebenen Daten.

Verification Da jede E-Mailadressenänderung eine Verifizierung benötigt, werden hier die Daten, zu welchen geändert wird, gespeichert. Das *validation-random* ist dabei der String, welcher im Verifizierungslink per E-Mail versendet wird.

Paper Ein Paper ist das in einem Einreichungsprozess hochgeladene Paper. Es ist modelliert als eine schwache Entität, abhängig von einer Einreichung, da ein Paper nicht im System existieren kann, ohne zu einem Einreichungsprozess zugeordnet zu werden. Es speichert die in /D020/ angegebenen Daten.

Submission Eine Einreichung speichert alle zu einem Einreichungsprozess gehörenden Daten, inklusive der Metadaten, wie den Einreicher und Ko-Autor, also alle in /D025/ und /DW021/ angegebenen Daten.

Review Ein Gutachten ist immer genau einem Paper zugeordnet, welches wiederum genau einer Einreichung zugeordnet ist. Es speichert die PDF als binäres Attribut in der Datenbank ab und alle zugehörigen Metadaten. Damit spiegelt es /D040/ wieder.

Scientific Forum Ein wissenschaftliches Forum besitzt die Referenz auf alle zu diesem Journal/Konferenz eingereichten Papers. Zusätzlich sind allgemeine Metadaten abgespeichert. Damit erfüllt es /D030/

Scientific Field Liste aller Fachgebiete bzw. Interessengebiete, die in LasEs verwendet werden. Diese werden als Fachgebieten der Foren und als Forschungsinteressen der Nutzer verwendet. Nach /D035/.

System Systemdaten speichern allgemeine Informationen über die Firma, die LasEs einsetzt. Sie sind in /D050/ spezifiziert.