

Entwurf

TalentTauscher



Betreuer: Prof. Dr. Christian Bachmaier

Team 1

Phase	Verantwortlich
Pflichtenheft	Leon Föckersperger
Entwurf	Jakob Edmaier
Spezifikation	Mike Karl
Implementierung	Mike Karl
Validierung & Abschlusspräsentation	David Sturm

Version 1.0

7. Mai 2024

Inhaltsverzeichnis

1 Einleitung	3
2 Änderungen zum Pflichtenheft	3
3 Systemarchitektur	3
3.1 Schichtenmodell	3
3.1.1 model	3
3.1.2 view	3
3.1.3 controller	3
3.1.4 dto	3
3.1.5 logging	4
3.2 Paketstruktur	4
3.2.1 model	4
3.2.2 logging	5
3.2.3 dto	5
3.3 Design Patterns	6
3.4 Fehlerbehandlung	7
3.4.1 Nutzerfehler	7
3.4.2 Checked Exception	7
3.4.3 Unchecked Exception	7
4 Klassen	7
4.1 Konventionen	7
4.2 Klassendiagramm	8
4.3 Klassenbeschreibungen	9
4.3.1 dto	9
4.3.2 logging	10
4.3.3 model.backing.beans	10
4.3.4 model.backing.exceptions	11
4.3.5 model.backing.session	11
4.3.6 model.backing.util	11
4.3.7 model.backing.validation	12
4.3.8 model.business.exceptions	12
4.3.9 model.business.lifecycle	12
4.3.10 model.business.services	13
4.3.11 model.business.util	13
4.3.12 model.persistence.connection	13
4.3.13 model.persistence.dataaccess	14
4.3.14 model.persistence.exceptions	15
4.3.15 model.persistence.lifecycle	15
4.3.16 model.persistence.util	15
5 Facelets	15
5.1 Template	16
5.2 Custom Components	16
5.2.1 Header	16
5.2.2 Footer	16
5.2.3 Sidebar	16
5.2.4 Dynamic Data Table	17
5.2.5 Sortable Data Column	17
5.3 Seiten	17
5.3.1 Anonyme Nutzer	17
5.3.2 Angemeldete Nutzer	20
5.3.3 Administrator	22
6 Systemfunktionen	23
6.1 Lifecycle	23
6.1.1 Systemstart	23
6.1.2 Systemstop	24
6.2 Konfiguration	24
6.3 Logging	24
6.4 E-Mail	24
6.5 Zugriffskontrolle (TrespassListener)	25
6.6 Systemsicherheit	25

6.7	Mehrbenutzerbetrieb	25
7	Datenfluss	26
7.1	Erfolgreiches Bearbeiten einer Anzeige	26
7.2	Fehlgeschlagenes Bearbeiten einer Anzeige	27
8	ER-Modell	27

1 Einleitung

LEON FÖCKERSPERGER

Dieses Dokument enthält den ersten Entwurf der Anwendung, in dem die einzelnen Komponenten detailliert spezifiziert und entweder in Listenform dargestellt oder grafisch illustriert sind. Die Kapitel erörtern umfassend die Klassen- und Datenbankstruktur und definieren sowohl die Funktionen als auch die Architektur des Systems.

2 Änderungen zum Pflichtenheft

JAKOB EDMAIER

Aufgrund des Ausscheiden eines weiteren Teammitglieds können die im Folgenden aufgelisteten Aspekte der Anwendung aus dem Pflichtenheft nicht mehr umgesetzt werden. In Klammern sind jeweils die betroffenen Produktfunktionen angegeben.

- Es wird im System keine Anzeigekategorien geben. (**/F1025/, /F3040/, /F3050/, /F3060/**)
- Pro Anzeige kann nur noch maximal ein Bild hochgeladen werden. (**/F1030/, /F2030/, /F2050/**)
- Prozesse, die im Hintergrund periodisch ausgeführt werden müssten, entfallen. Dies betrifft das automatische Löschen von nicht verifizierten Benutzern aus der Datenbank nach einem bestimmten Zeitintervall sowie das Invalidieren von Verifizierungstokens (z.B. beim Zurücksetzen von Passwörtern). (**/F1065/, /F1080/**)
- Die Freitextsuche bei Anzeigen wird nicht umgesetzt. (**F1110W**)

3 Systemarchitektur

DAVID STURM

In diesem Kapitel wird die Architektur unseres Programmes eingegangen. Dazu wird zunächst das zugrunde liegende Schichtenmodell mittels Abbildung veranschaulicht und die einzelnen Schichten kurz erklärt. Im Anschluss werden die einzelnen Pakete und deren Funktion angegeben. Abschließend werden noch die verwendeten Designpatterns erklärt und auf die verwendete Fehlerbehandlung eingegangen.

3.1 Schichtenmodell

Wie Abbildung 1 veranschaulicht, wird das Programm gemäß dem Model-View-Controller Designpatterns (3.3) in drei Grundlegende Schichten unterteilt. Außerdem werden noch zwei weitere schichtenübergreifende Pakete verwendet.

3.1.1 model

Das Model enthält alle Klassen die der Geschäftslogik des Programms dienen. Des weiteren befinden sich hier jene Klassen die dem Zugriff auf das Speichersystem dienen. Das Model ist intern wiederum in drei weitere Schichten aufgeteilt.

backing Beinhaltet die Logik für die Darstellung der Daten, die Validierung und Verarbeitung von Nutzerinputen sowie das managen der Aktuellen Nutzersession.

business Enthält die Klassen, welche die darstellungsunabhängige Anwendungslogik bilden.

persistence Beinhaltet Klassen für den Zugriff auf den verwendeten Datenspeicher für die Daten. Die spezifische Art des Speichers, hier eine PostgreSQL Datenbank, wird dabei für die oberen Schichten vollständig abstrahiert.

3.1.2 view

In diesem Paket befinden sich die sogenannten Facelets. Diese dienen als Vorlage für die Darstellung von Benutzeroberflächen und ermöglichen die Erstellung einer wiederverwendbaren, komponentenbasierten Anzeige.

3.1.3 controller

Die Controller reagieren auf Ereignisse welche in der View ausgelöst werden. Diese Aufgabe wird von Jakarta Faces automatisch übernommen und erfordert keine weitere Klassen unsererseits.

3.1.4 dto

Hier befinden sich die so genannten Data-Transfer-Objets (2). Sie werden genutzt um Daten zwischen den verschiedenen Schichten der Programms zu übertragen.

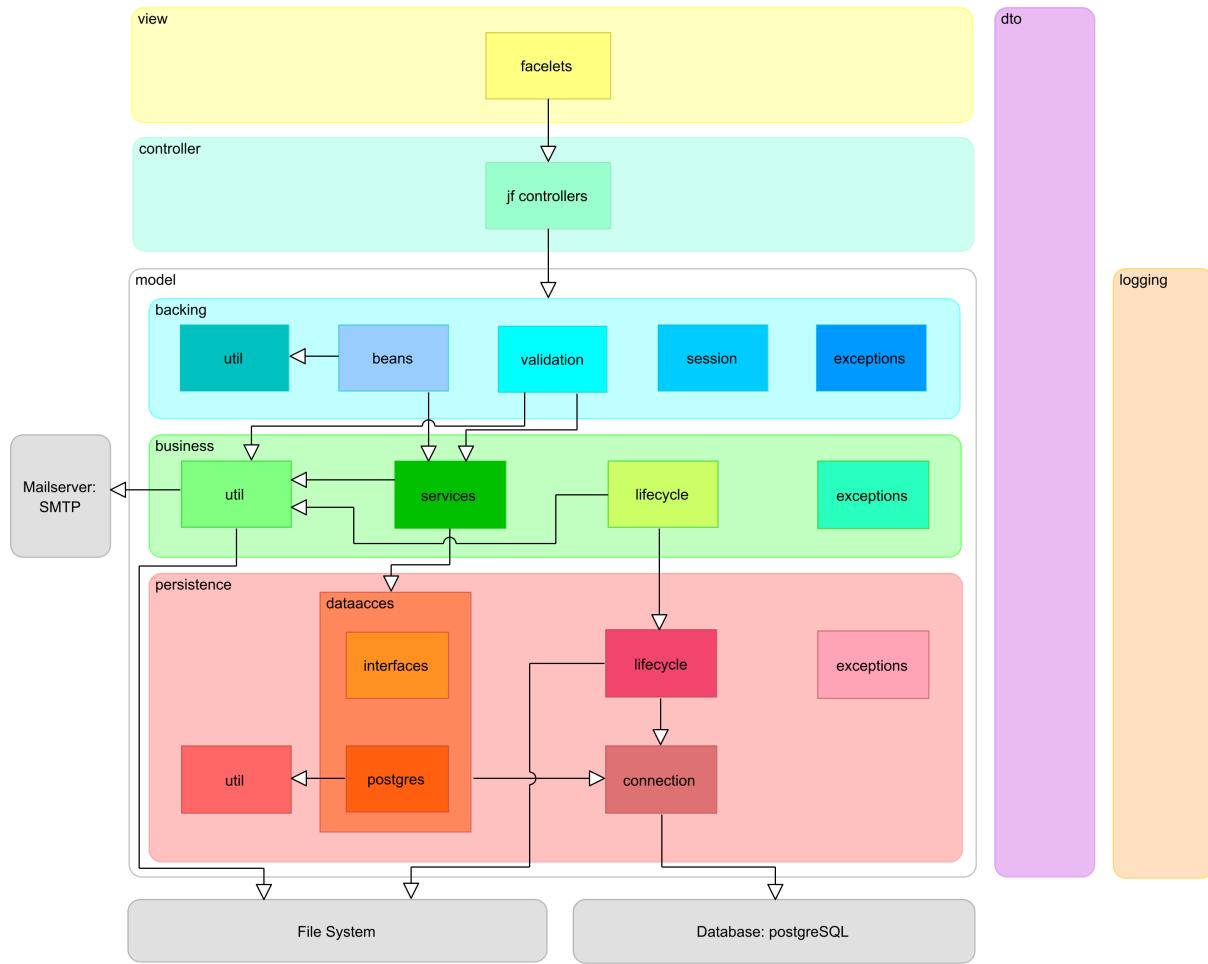


Abbildung 1: Schichtenmodell

3.1.5 logging

Hier befindet sich der Loggerproducer.

3.2 Paketstruktur

3.2.1 model

Package	Beschreibung
Backing	
<i>beans</i>	Enthält alle Backing beans welche für die Verarbeitung der Nutzeraktivitäten in dem Programm genutzt werden. Die Beans tragen hierbei den selben Namen des ihnen zugehörigen Facelets aus der View mit dem Postfix "Bean" ("facelet" \Rightarrow "faceletBean").
<i>validation</i>	Enthält alle Validatoren zur Überprüfung von Nutzereingaben in den Facelets. Des weiteren befindet sich hier der Passwordconverter, welcher für das Hashen des Nutzernamens dient.
<i>session</i>	Enthält Klasse für die aktuelle Sessionverwaltung des Nutzers.
<i>util</i>	Enthält Utilityklassen für das Laden von Nachrichten aus dem Resource Bundle, die Paginierung so wie den Tresspasslistener.

<i>exceptions</i>	Enthält den Exception-Handler. Dieser fängt und verarbeitet schwere Runtime Exceptions welche im Verlauf des Programms eventuell auftreten und reagiert dementsprechend. Ebenfalls liegt hier die UnauthorizedAccessException des Tresspasslisteners.
<i>business</i>	
<i>services</i>	Enthält per CDI verwaltete Serviceklassen welche die Interaktion mit den Data-Access-Objects der des persistence Pakets managen.
<i>lifecycle</i>	Enthält Klasse welche den Start und Stopp des Programms handhabt.
<i>util</i>	Enthält Utilityklassen für die Passwortverifikation und für das Senden von E-Mails.
<i>exceptions</i>	Enthält eigen definierte Exceptions für die E-Mail Funktionalität.
<i>persistence</i>	
<i>connection</i>	Enthält die Klasse für den Connection Pool. Diese stellt die Verbindung mit der Datenbank bereit.
<i>dataaccess</i>	Enthält eine Factory, welche Instanzen des Transaction Interfaces bereitstellt. Dieses repräsentiert einen Zugriff auf den Datenspeicher unabhängig von dessen genauen Typen.
<i>dataaccess.interfaces</i>	Hier liegen Interfaces welche den Zugriff auf den Datenspeicher repräsentieren. Dabei handelt es sich einerseits um das Interface Transaction, andererseits um alle genutzten Data-Access-Objects (kurz DAO) welche den Datenzugriff handhaben. Außerdem liegt hier die DAOFactory, welche ausimplementierte Instanzen der DAO's bereitstellt. Die DAO Klassen enden immer mit dem Postfix " <i>DAO</i> ".
<i>dataaccess.postgres</i>	Enthält für PostgreSQL spezifische Implementierungen der Interfaces des Pakets <i>dataaccess.interfaces</i> . Der Name der Klassen dieses Pakets hat immer den Präfix " <i>PostgreSQL</i> ".
<i>util</i>	Enthält Klasse für das Bauen von SQL-Anfragen aus für die Pagination.
<i>lifecycle</i>	Enthält Klassen zum initialen aufsetzen der Datenbank.
<i>exceptions</i>	Enthält eigen definierte Exceptions für Fehler die in Zusammenhang mit dem Speicherzugriff so wie dem auslesen der Config-Data auftreten.

3.2.2 logging

Enthält den Loggerproducer, welcher Instanzen des Loggers bereitstellt.

3.2.3 dto

Enthält alle Data-Transfer-Objects des Programms so wie einige Enums. Der Name der Klassen der DTO's endet mit "*DTO*".

3.3 Design Patterns

Für die Entwicklung des Programms greifen wird auf die hier folgend aufgeführten Designpatterns zurückgegriffen.

Pattern	Beschreibung
<i>Model-View-Controller</i>	Das MVC-Pattern ist das zugrunde liegende Entwurfsmuster für unsere Anwendung. Dieses unterteilt das Programm in drei Komponenten. Das Model enthält die Programmlogik so wie die Komponenten für den Datenzugriff. Die View sind Facelets welche die Nutzeroberfläche bilden. Der Controller befindet sich zwischen den beiden anderen. Er verarbeitet die Nutzerinteraktion. Diese Aufgabe wird von Jakarta Faces automatisch übernommen.
<i>Contexts and Dependency Injection</i>	Dies ist ein Framework welches die Verwaltung von Objektabhängigkeiten und -lebenszyklen erleichtert. Wir nutzen dies unter anderem zum Verwalten und Injizieren der Services in den Beans, für das Verwalten der Beans selber so wie der UserSession im Trespasslistener.
<i>Singleton Pattern</i>	Das Singleton Pattern stellt sicher, dass für eine Java Klasse nur eine einzige Instanz gleichzeitig im System existiert, diese dafür aber globalen Zugriff hat. Dies wird für die EmailConfig in der model.business Schicht sowie der DatabaseConfig in der model.persistence Schicht genutzt. Des Weiteren findet es Anwendung für den ConnectionPool welcher die Verbindung zur Datenbank verwaltet.
<i>Object Pool Pattern</i>	. Anstatt immer neue Instanzen zu erstellen und nach Gebrauch wieder zu löschen wird beim Object Pool pattern auf einen bestehenden Pool an bereits initialisierten Objekten zugegriffen und diese nach Verwendung an den Object Pool zurückgegriffen. Dieses Pattern findet wird beim ConnectionPool zum Verwalten der Connection Objekte verwendet, welche die Verbindungen zur Datenbank darstellen.
<i>Data-Access-Object</i>	Dieses Entwurfsmuster bietet eine Abstraktionsebene zwischen der Anwendungslogik und dem Datenspeicher. Jedes DAO stellt eine Schnittstelle bereit, über die die Anwendungslogik auf Daten zugreifen kann. Dies wird bewerkstelligt, indem von oben immer nur auf Datenspeicher unspezifische Interfaces zugegriffen wird.
<i>Data-Transfer-Object</i>	Data-Transfer-Objects werden verwendet, um Daten zwischen den verschiedenen Schichten des Programms zu übertragen. Es handelt sich hierbei um Klassen welche zusammenhängende Informationen gebündelt speichern, selbst aber keinerlei eigene Logik beinhalten.
<i>Factory Pattern</i>	Beim Factory Pattern wird die spezifische Instanzierung eines Objekts, z.B. eines Interfaces, an eine Fabrikklasse ausgelagert. Dadurch braucht der aufrufende nichts über die spezifische Implementierung wissen. Dies wird unter anderem bei dem Transaction Interface so wie den DAO Interfaces genutzt, wodurch die spezifische Art des Datenspeichers mit geringen Kosten ausgetauscht werden kann.

3.4 Fehlerbehandlung

Die Fehlerbehandlung ist ein zentraler Aspekt unserer Webapplikation, um die Stabilität und Benutzerfreundlichkeit sicherzustellen. Fehler während der Programmausführung werden systematisch erfasst, protokolliert (geloggt) und entsprechend ihres Typs behandelt.

3.4.1 Nutzerfehler

Nutzerfehler resultieren aus fehlerhaften Eingaben der Benutzer, wie beispielsweise ein ungültiges E-Mail-Format. Diese Fehler werden mithilfe von Validatoren identifiziert. Der Benutzer wird über eine klare und verständliche Fehlermeldung informiert, die durch das Frontend angezeigt wird. Diese Rückmeldung ermöglicht es dem Benutzer, die Eingabe zu korrigieren und verbessert die Interaktion mit der Anwendung.

3.4.2 Checked Exception

Checked Exceptions sind vorhersehbare Fehler, die während der Laufzeit auftreten können und vom System erwartet werden. Unsere Applikation behandelt diese Fehler proaktiv, indem sie direkt abgefangen und an Ort und Stelle behandelt oder zur weiteren Behandlung an eine übergeordnete Schicht weitergeleitet werden. Sollte es sich um einen Fehler handeln, welcher die Funktion des Programms signifikant beeinträchtigt wird der Nutzer stets durch eine FacesMessage informiert, welche klar und verständlich formuliert ist. Diese Nachrichten werden systematisch durch das zugehörige Facelet gerendert. Dies gewährleistet, dass der Nutzer auch bei signifikanten Störungen, wie beispielsweise einer Datenbankstörung, stets angemessen informiert wird und weiß, wie er auf das Problem reagieren sollte.

3.4.3 Unchecked Exception

Unchecked Exceptions sind kritische Fehler, die sich gravierend auf den Ablauf der Applikation auswirken und oft nicht vorhersehbar sind. Diese Fehler können durch alle Schichten der Anwendung propagieren, bevor sie vom globalen `AppExceptionHandler` erfasst werden. In solchen Fällen wird der Nutzer automatisch auf eine Fehlerseite umgeleitet, auf der ihm eine informative und hilfreiche Nachricht angezeigt wird, die das Problem erläutert und mögliche nächste Schritte aufzeigt.

4 Klassen

JAKOB EDMAIER

Das folgende Kapitel enthält das Klassendiagramm sowie eine kurze Beschreibung der vorgesehenen Klassen. Im Anhang findet sich eine Version des Klassendiagramms in einem größeren Format (Klassendiagramm.png).

4.1 Konventionen

Für jede DAO-Klasse in der `persistence`-Layer (z.B. `UserDAO`) gibt es eine entsprechende `@ApplicationScoped` Service-Klasse im `business`-Layer (z.B. `UserService`), die alle Zugriffe auf die DAOs für die `backing`-Layer abstrahiert. Die Klassen in den Schichten `backing` und `business` sind per CDI gemanaged (mit Ausnahme der Exceptions). Die Klassen der untersten Schicht `persistence` sind nicht gemanaged.

Klassen der `persistence`-Layer, die einen Logger benötigen, erstellen eine Instanz über eine Fabrik-Methode der Klasse `LoggerProducer`. Klassen der `backing` und `business`-Layer injecten bei Bedarf einen Logger und benutzen damit ebenfalls implizit den `LoggerProducer`. Diese use-Beziehungen sind im Klassendiagramm nicht eingezzeichnet.

4.2 Klassendiagramm

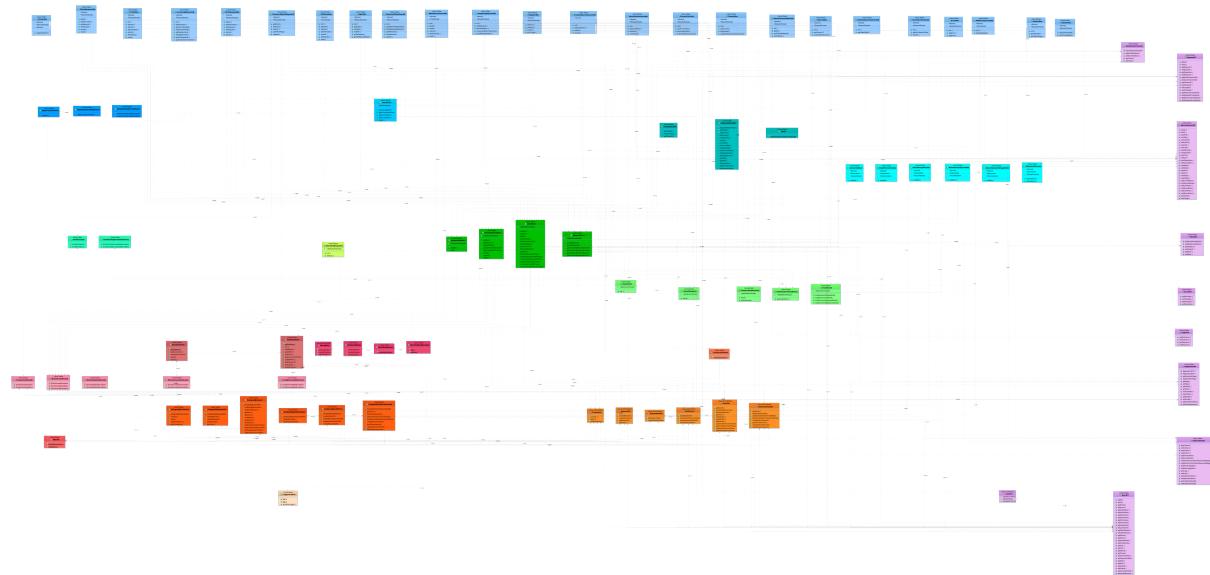


Abbildung 2: Klassendiagramm

4.3 Klassenbeschreibungen

In diesem Abschnitt wird kurz die Funktion der einzelnen Klassen erklärt. Die Auflistung ist nach Paketen sortiert, und innerhalb der Pakete alphabetisch. Die Beschreibungen sind in Englisch gehalten, um sie später ggf. als Javadoc-Kommentare wiederverwenden zu können.

4.3.1 dto

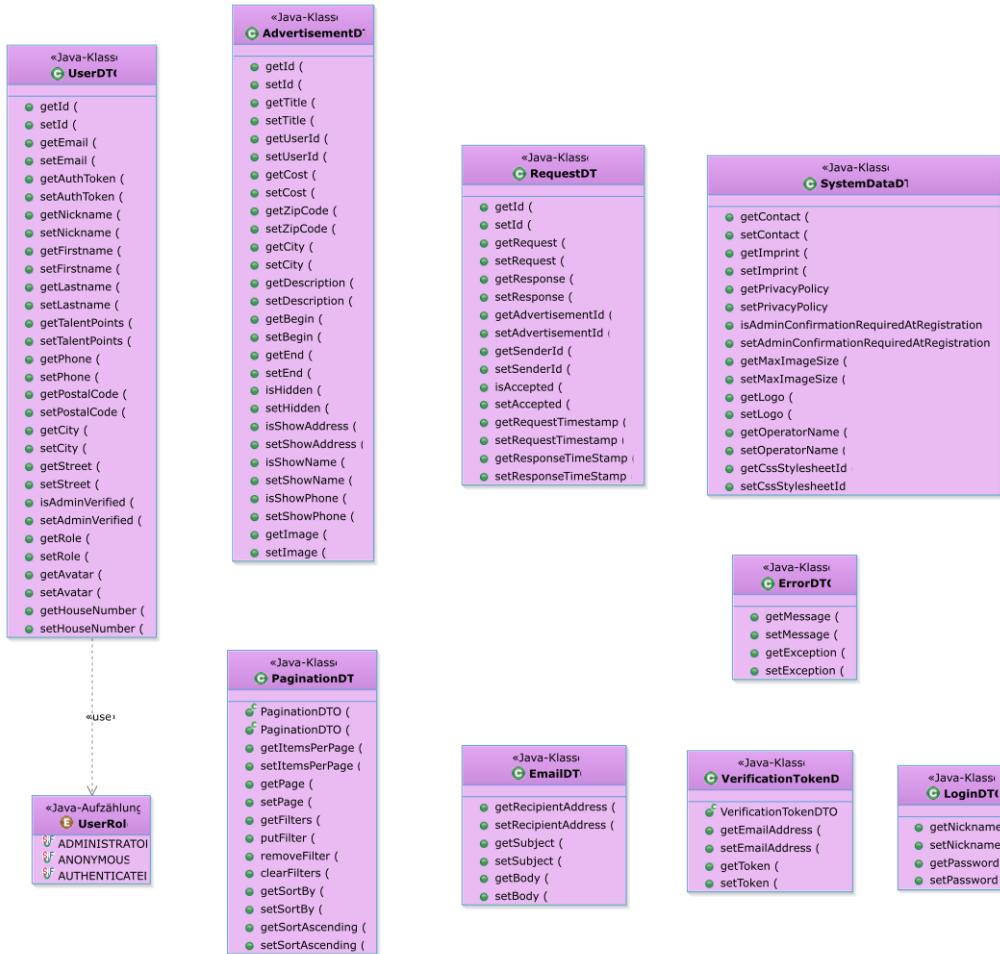


Abbildung 3: Klassendiagramm **dto**

Klasse	Beschreibung
<i>AdvertisementDTO</i>	Represents an advertisement.
<i>EmailDTO</i>	Represents an email with recipient, subject and message body.
<i>ErrorDTO</i>	Represents a fatal error.
<i>LoginDTO</i>	Represents the login data.
<i>PaginationDTO</i>	Represents the filters, sorting and pagination applied to a data collection.
<i>RequestDTO</i>	Represents a request and, if present, the associated response.
<i>SystemDataDTO</i>	Represents the system configuration data.
<i>UserDTO</i>	Represents a user.
<i>UserRole</i>	Represents a user role in the system (anonymous, authenticated or administrator).
<i>VerificationTokenDTO</i>	Represents a verification token used for password resetting and email confirmation.

4.3.2 logging



Abbildung 4: Klassendiagramm logging

Klasse	Beschreibung
<i>LoggerProducer</i>	Provides instances of the <i>Logger</i> class.

4.3.3 model.backing.beans

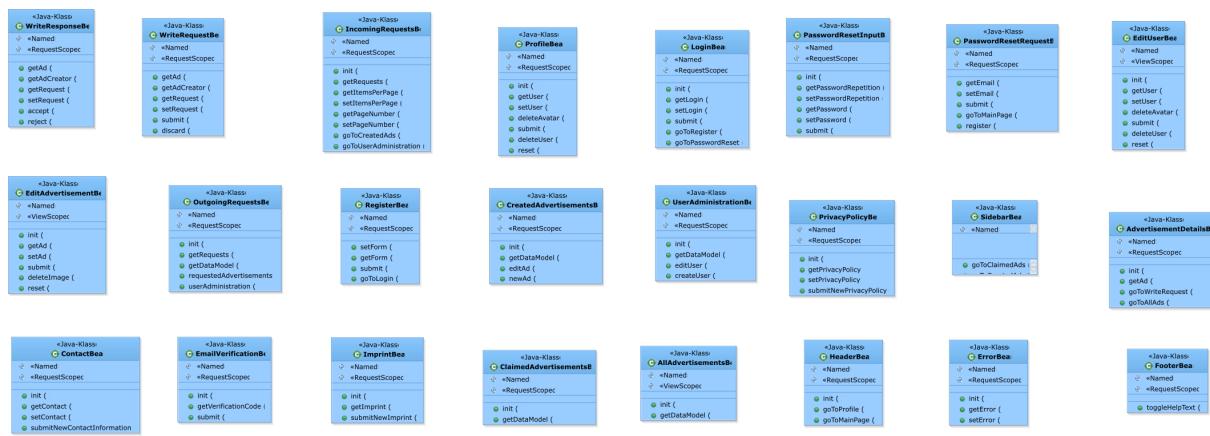


Abbildung 5: Klassendiagramm model.backing.beans

Klasse	Beschreibung
<i>AdvertisementDetailsBean</i>	Backing bean for the advertisement details screen.
<i>AllAdvertisementsBean</i>	Backing bean for the advertisement overview screen.
<i>ClaimedAdvertisementsBean</i>	Backing bean for the list of advertisements claimed by a user.
<i>ContactBean</i>	Backing bean for the contact information of the administrators.
<i>CreatedAdvertisementsBean</i>	Backing bean for the list of advertisements created by a user.
<i>EditAdvertisementBean</i>	Backing bean for the advertisement creation/editing screen.
<i>EditUserBean</i>	Backing bean for the user creation/editing screen.
<i>EmailVerificationBean</i>	Backing bean for the email verification landing page.
<i>ErrorBean</i>	Backing bean for the error page.
<i>FooterBean</i>	Backing bean for the footer.
<i>HeaderBean</i>	Backing bean for the header.
<i>ImprintBean</i>	Backing bean for the imprint page.
<i>IncomingRequestsBean</i>	Backing bean for the list of incoming requests.
<i>LoginBean</i>	Backing bean for the login page.
<i>OutgoingRequestsBean</i>	Backing bean for the list of outgoing requests.
<i>PasswordResetInputBean</i>	Backing bean for the reset password form.
<i>PasswordResetRequestBean</i>	Backing bean for the password forgotten page.
<i>PrivacyPolicyBean</i>	Backing bean for the privacy policy page.
<i>ProfileBean</i>	Backing bean for the user profile page.
<i>RegisterBean</i>	Backing bean for the register page.
<i>SidebarBean</i>	Backing bean for the sidebar.
<i>UserAdministrationBean</i>	Backing bean for the user administration page.
<i>WriteRequestBean</i>	Backing bean for the page for writing requests.
<i>WriteResponseBean</i>	Backing bean for the page for writing responses.

4.3.4 model.backing.exceptions

Abbildung 6: Klassendiagramm **model.backing.exceptions**

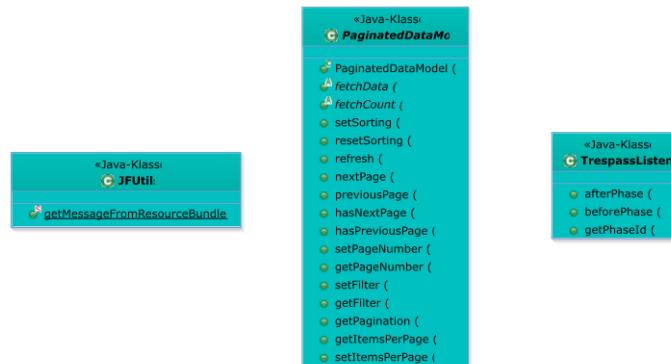
Klasse	Beschreibung
<i>AppExceptionHandler</i>	Responsible for handling fatal exceptions thrown in lower levels of the application and redirecting the user to a suitable error page.
<i>AppExceptionHandlerFactory</i>	Factory class for creating instances of the <i>AppExceptionHandler</i> class.
<i>UnauthorizedAccessException</i>	Checked exception thrown when a user tries to access a page without the necessary permissions.

4.3.5 model.backing.session

Abbildung 7: Klassendiagramm **model.backing.session**

Klasse	Beschreibung
<i>UserSession</i>	Manages the current user session.

4.3.6 model.backing.util

Abbildung 8: Klassendiagramm **model.backing.util**

Klasse	Beschreibung
<i>JFUtils</i>	Utility class for accessing strings from a resource bundle.
<i>PaginatedDataModel</i>	Responsible for loading and providing data displayed in a paginated data table.
<i>TrespassListener</i>	Responsible for checking the permissions when a user navigates to another page.

4.3.7 model.backing.validation

Abbildung 9: Klassendiagramm **model.backing.validation**

Klasse	Beschreibung
<i>AdvertisementImageValidator</i>	Ensures that an image for an advertisement has the required format and size.
<i>AvatarValidator</i>	Ensures that an avatar image has the required format and size.
<i>EmailFormatValidator</i>	Ensures that a string is in a valid email address format.
<i>EmailUniqueValidator</i>	Ensures that an email address is unique among all users.
<i>NicknameUniqueValidator</i>	Ensures that a nickname is unique among all users.
<i>PasswordConverter</i>	Converts a password to a hashed authentication token.

4.3.8 model.business.exceptions

Abbildung 10: Klassendiagramm **model.business.exception**

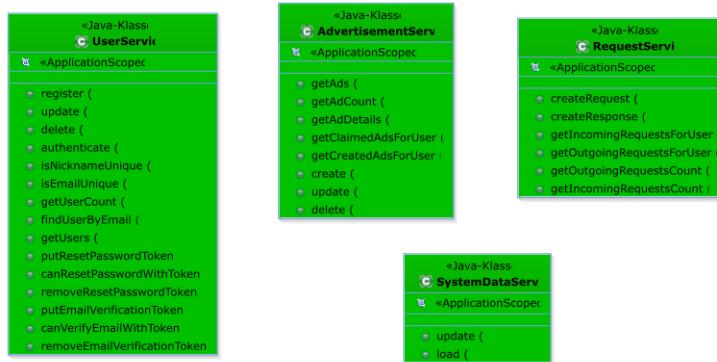
Klasse	Beschreibung
<i>EmailConfigUnreadableException</i>	Unchecked exception thrown when the email config file cannot be loaded.
<i>EmailException</i>	Checked exception thrown when an email cannot be sent.

4.3.9 model.business.lifecycle

Abbildung 11: Klassendiagramm **model.business.lifecycle**

Klasse	Beschreibung
<i>StartAndStopListener</i>	Responsible for managing the system startup and shutdown.

4.3.10 model.business.services

Abbildung 12: Klassendiagramm **model.business.services**

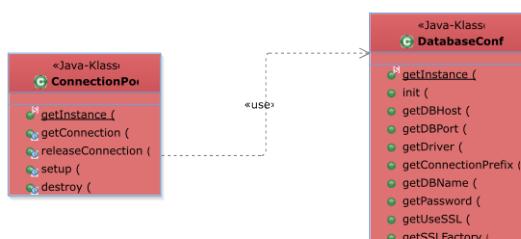
Klasse	Beschreibung
<i>AdvertisementService</i>	Manages interactions with <i>AdvertisementDAO</i> .
<i>RequestService</i>	Manages interactions with <i>RequestDAO</i> .
<i>SystemService</i>	Manages interactions with <i>SystemServiceDAO</i> .
<i>UserService</i>	Manages interactions with <i>UserDAO</i> .

4.3.11 model.business.util

Abbildung 13: Klassendiagramm **model.business.util**

Klasse	Beschreibung
<i>EmailBuilder</i>	Creates <i>EmailDTOs</i> for all situations, in which emails are sent by the application.
<i>EmailConfig</i>	Loads the email properties file and provides the configuration parameters through its getters.
<i>EmailDispatcher</i>	Utility class for sending emails.
<i>PasswordAuthentication</i>	Utility class for hashing and salting passwords.
<i>VerificationTokenGenerator</i>	Generates random verification tokens.

4.3.12 model.persistence.connection

Abbildung 14: Klassendiagramm **model.persistence.connection**

Klasse	Beschreibung
<i>ConnectionPool</i>	Manages the connections to the database.
<i>DatabaseConfig</i>	Loads the database properties file and provides the configuration parameters through its getters.

4.3.13 model.persistence.dataaccess

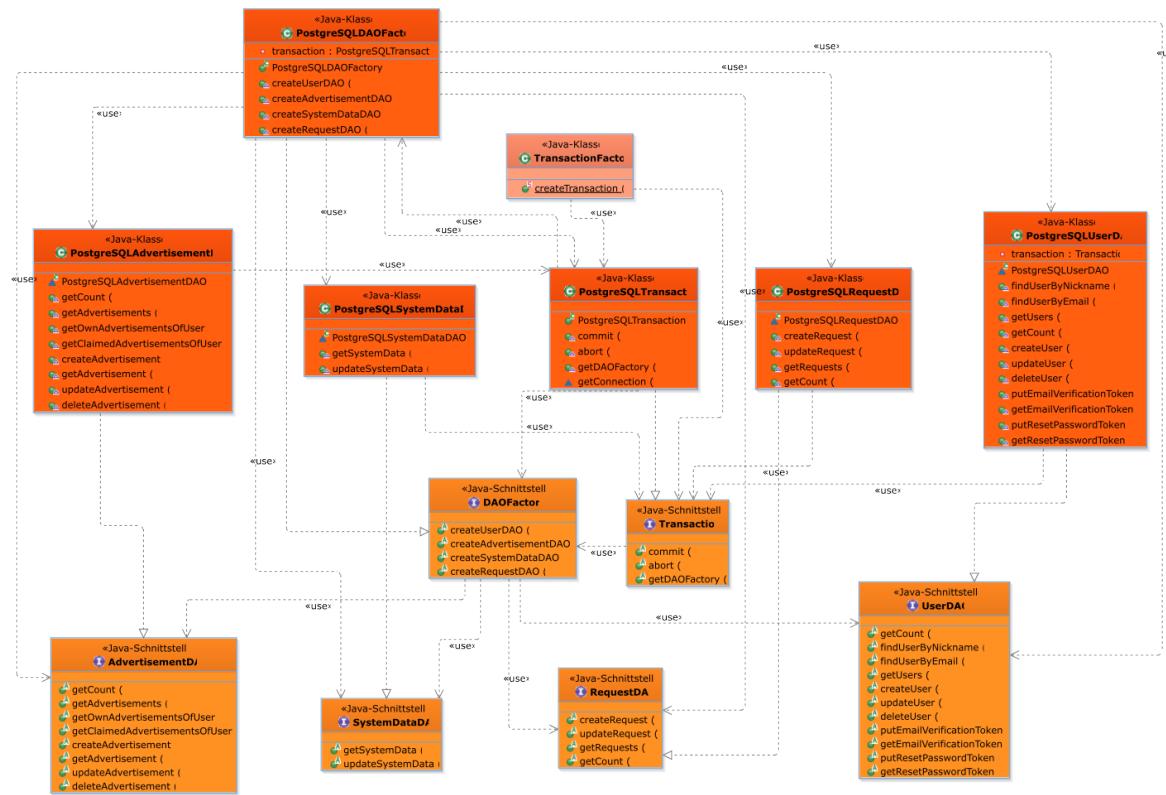
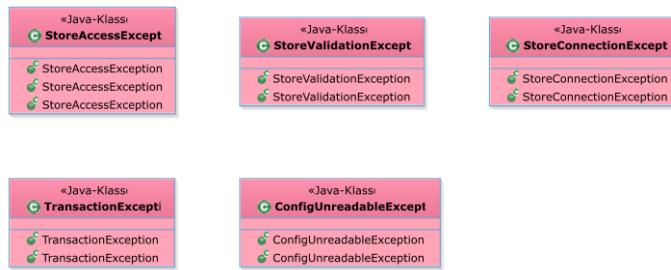


Abbildung 15: Klassendiagramm `model.persistence.dataaccess`

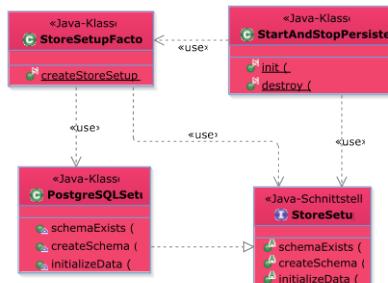
Klasse	Beschreibung
<i>TransactionFactory</i>	Factory class for creating instances of the <code>Transaction</code> interface.
dataaccess.interfaces	
<i>AdvertisementDAO</i>	Interface for data store accesses concerning advertisements.
<i>DAOFactory</i>	Interface for factory classes for creating DAO's.
<i>RequestDAO</i>	Interface for data store accesses concerning requests.
<i>SystemDataDAO</i>	Interface for data store accesses concerning the system data.
<i>Transaction</i>	Interface representing a data store transaction.
<i>UserDAO</i>	Interface for data store accesses concerning user management.
dataaccess.postgres	
<i>PostgreSQLAdvertisementDAO</i>	PostgreSQL implementation of the <code>AdvertisementDAO</code> interface.
<i>PostgreSQLDAOFactory</i>	PostgreSQL implementation of the <code>SystemDataDAO</code> interface.
<i>PostgreSQLRequestDAO</i>	PostgreSQL implementation of the <code>DAOFactory</code> interface.
<i>PostgreSQLSystemDataDAO</i>	PostgreSQL implementation of the <code>SystemDataDAO</code> interface.
<i>PostgreSQLTransaction</i>	PostgreSQL implementation of the <code>Transaction</code> interface.
<i>PostgreSQLUserDAO</i>	PostgreSQL implementation of the <code>UserDAO</code> interface.

4.3.14 model.persistence.exceptions

Abbildung 16: Klassendiagramm `model.persistence.exceptions`

Klasse	Beschreibung
<code>ConfigUnreadableException</code>	Unchecked exception thrown if the database config cannot be loaded.
<code>StoreAccessException</code>	Checked exception thrown when an error occurs during a data store access.
<code>StoreConnectionException</code>	Unchecked exception thrown if the application cannot connect to the data store.
<code>StoreValidationException</code>	Checked exception indicating an invalid input.
<code>TransactionException</code>	Checked exception thrown when a transaction commit/abort fails.

4.3.15 model.persistence.lifecycle

Abbildung 17: Klassendiagramm `model.persistence.lifecycle`

Klasse	Beschreibung
<code>PostgreSQLSetup</code>	PostgreSQL implementation of the <code>StoreSetup</code> interface.
<code>StoreSetup</code>	Interface for setting up a datastore with a given schema and initial data.
<code>StoreSetupFactory</code>	Static factory class for creating a <code>StoreSetup</code> instance.

4.3.16 model.persistence.util

Abbildung 18: Klassendiagramm `model.persistence.util`

Klasse	Beschreibung
<code>SQLUtil</code>	Utility class for building SQL queries from <code>PaginationDTO</code> objects.

5 Facelets

Dieser Abschnitt erörtert die Implementierung und Konfiguration von Facelets und zugehörigen Steuerelementen in der Anwendung. Es wird darauf hingewiesen, dass die Sichtbarkeit der Steuerelemente von der Benutzerrolle und kontextbezogenen Bedingungen abhängig ist.

Legende:

AN = Alle Nutzer

AY = Anonyme Nutzer

N = Authentifizierte Nutzer

A = Administratoren

5.1 Template

Das Template bildet das Grundgerüst für die Darstellung aller Seiten in der Anwendung.

skeleton.xhtml

Art	Name	Beschreibung	Sichtbarkeit
Header	header	Bietet navigationsrelevante Funktionen einschließlich Authentifizierungsoptionen.	AN
Footer	footer	Zugang zu organisatorischen Informationen wie Kontakt und Impressum.	AN
Sidebar	sidebar	Ermöglicht die Navigation zwischen den Hauptbereichen der Anwendung.	AN
Content	childContent	Dient als Container für den spezifischen Inhalt der verschiedenen Seiten.	AN
Message	message	Stellt systemweite Benachrichtigungen dar.	AN
TextArea	helpText	Bietet kontextabhängige Hilfstexte.	AN
Button	helpButton	Schaltet die Sichtbarkeit des Hilfstextes um.	AN

5.2 Custom Components

5.2.1 Header

Der Header wird konsistent über alle Seiten hinweg eingefügt und bietet Zugang zu Authentifizierungsdialogen.

header.xhtml

Art	Name	Beschreibung	Sichtbarkeit
Image	logo	Stellt das durch die Administration gewählte Logo dar.	AN
Button	userManagement	Verlinkt zur Seite der Benutzerverwaltung für Administratoren.	A
Button	login/logout	Bietet den Zugang zur Anmelde- bzw. Abmeldeseite je nach Nutzerstatus.	AY/N
Button	profile	Ermöglicht Zugriff auf die Profilseite für authentifizierte Nutzer.	N

5.2.2 Footer

Der Footer bildet den unteren Abschluss jeder Seite und enthält Links zu wichtigen organisatorischen Informationen.

footer.xhtml

Art	Name	Beschreibung	Sichtbarkeit
Link	privacy	Verlinkt zur Datenschutzerklärung.	AN
Link	contact	Verlinkt zu Kontaktinformationen.	AN
Link	imprint	Verlinkt zum Impressum.	AN

5.2.3 Sidebar

Die Sidebar ermöglicht eine effiziente Navigation durch die Hauptinhaltsbereiche und ist ein wesentliches Element der Benutzeroberfläche.

sidebar.xhtml

Art	Name	Beschreibung	Sichtbarkeit
Link	overview	Bietet Zugang zur Hauptübersichtsseite.	AN
Link	createdAds	Ermöglicht die Navigation zu selbst erstellten Anzeigen.	N
Link	requestedAds	Bietet Zugang zu den in Anspruch genommenen Anzeigen.	N

5.2.4 Dynamic Data Table

Die **DynamicDataTable**-Komponente bietet eine dynamische Anzeige und Interaktionsmöglichkeit mit Datenlisten, die direkt aus einer Datenbank geladen werden. Sie unterstützt das Blättern durch die Daten mittels Pagination, was das Durchsuchen großer Datenmengen erleichtert. Folgende Konfigurationen sind notwendig:

- Ein Bean, das die abstrakte Klasse `PaginatedDataModel` implementiert.
- Eine Vorgabe für die standardmäßige Sortierspalte.

dynamicDataTable.xhtml

Art	Name	Beschreibung	Sichtbarkeit
Table	table	Stellt Daten in definierten Spalten dar.	A
Dropdown	page	Ermöglicht die schnelle Navigation zu bestimmten Seiten.	A
Link	previous	Navigiert zur vorherigen Seite.	A
Link	next	Navigiert zur nächsten Seite.	A

5.2.5 Sortable Data Column

Die **SortableDataColumn** ermöglicht das individuelle Anpassen von Tabellenspalten, um das Sortieren und Filtern nach Benutzerdefinierten Kriterien wie Syntax oder Wildcard zu optimieren. Erforderliche Konfigurationen sind:

- Ein Bean, das die abstrakte Klasse `PaginatedDataModel` implementiert.
- Ein Anzeigename für den Spaltenkopf.
- Ein eindeutiger Identifier der Spalte (Spaltenname in SQL und Name des DTO-Properties).
- Die Spaltennummer, null-indiziert.

sortableDataColumn.xhtml

Art	Name	Beschreibung	Sichtbarkeit
TextArea	columnName	Bestimmt den Namen der Spalte.	A
InputFields	search	Ermöglicht das Filtern der Daten nach spezifischen Kriterien je Spalte.	A
Button	startSearch	Löst eine Anfrage für eine aktualisierte Datenliste aus.	A
Button	direction	Passt die Sortierkriterien und -reihenfolge an.	A

5.3 Seiten

DAVID STURM

Für alle Seiten wird skeleton.xhtml benutzt um das Layout der Seite aufzubauen. Da Header, Footer und Sidebar damit bereits eingebaut sind, sind folgende Bedienelemente also als Zusatz im Main Content zu betrachten. Die Seiten werden aufsteigend nach den benötigten Zugriffsrechten gruppiert, somit kann außer explizit jede Seite, außer explizit angegeben, ebenfalls von den Nutzern der folgenden Sektionen aufgerufen werden. Elemente der Art "Inputfield" sowie "InputFile" implizieren immer, dass es ebenfalls Elemente des Typs "Message" gibt welche gegebenenfalls Fehlermeldungen anzeigen. Die Namenskonvention für diese ist immer "Inputname" ⇒ "InputnameMessage" und sie besitzen die selbe Sichtbarkeit wie ihr zugehöriges Inputelement.

5.3.1 Anonyme Nutzer

Die folgenden Seiten können von allen Nutzern, somit insbesondere von anonymen Nutzern, aufgerufen werden.

Impressum Diese Seite ermöglicht dem Nutzer das Impressum einzusehen. Der Admin hat hier die Möglichkeit das Impressum zu bearbeiten.

imprint.xhtml

Art	Name	Beschreibung	Sichtbarkeit
TextArea	imprint	Zeigt das gesamte Impressum.	AN
Button	update	Ermöglicht Bearbeitung des Impressums.	A

Kontakt Diese Seite ermöglicht dem Nutzer die Kontaktdaten des Admins einzusehen. Der Admin hat hier die Möglichkeit die Daten zu bearbeiten.

contact.xhtml

Art	Name	Beschreibung	Sichtbarkeit
TextArea	contact	Zeigt alle Kontaktdaten.	AN
Button	update	Ermöglicht Bearbeitung der Kontaktdaten.	A

Datenschutz Diese Seite ermöglicht dem Nutzer den Datenschutzeintrag einzusehen. Der Admin hat hier die Möglichkeit den Datenschutzeintrag zu bearbeiten.

privacyPolicy.xhtml

Art	Name	Beschreibung	Sichtbarkeit
TextArea	privacyPolicy	Zeigt den gesamten Datenschutzeintrag.	AN
Button	update	Ermöglicht Bearbeitung des Datenschutzeintrags.	A

Login Die Login Seite bietet dem Nutzer die Möglichkeit seine Anmeldedaten einzugeben oder zur Registrieren Seite zu wechseln. Optional kann auch das Passwort zurückgesetzt werden. Diese Seite kann nur von anonymen Nutzern aufgerufen werden.

login.xhtml

Art	Name	Beschreibung	Sichtbarkeit
InputField	nick	Hier kann der Nutzernname oder die Email eingegeben werden.	AY
InputField	password	Hier kann das Passwort eingegeben werden.	AY
Link	register	Weiterleitung an die Registrieren Seite.	AY
Link	forgotPassword	Weiterleitung an die Passwort Rücksetzung Anfrage Seite.	AY
Button	submit	Versucht mit den gegebenen Daten anzumelden.	AY

Passwort Rücksetzung Anfrage Diese Seite ermöglicht dem Nutzer eine Email anzugeben, deren Account eine Passwort Rücksetzung benötigt. Diese Seite kann nur von anonymen Nutzern aufgerufen werden.

passwordResetRequest.xhtml

Art	Name	Beschreibung	Sichtbarkeit
InputField	email	Hier kann eine Email festgelegt werden.	AY
Button	submit	Versucht mit den gegebenen Daten eine Passwort Rücksetzung zu starten.	AY

Passwort Rücksetzung Passworteingabe Diese Seite ermöglicht dem Nutzer ein neues Passwort anzugeben, sofern eine Passwort Rücksetzung läuft. Diese Seite kann nur von anonymen Nutzern mittels einem per E-Mail erhaltenen Link aufgerufen werden.

passwordResetInput.xhtml

Art	Name	Beschreibung	Sichtbarkeit
InputField	password	Hier kann ein neues Passwort festgelegt werden.	AY
Message	passwordMessage	Hier werden Fehler angezeigt, die das neue Passwort betreffen.	AY
Button	submit	Versucht mit den gegebenen Daten die Passwort Rücksetzung abzuschließen.	AY

Registrieren Die Registrieren Seite bietet dem Nutzer die Möglichkeit sich zu registrieren oder zur Login Seite zu wechseln. Diese Seite kann nur von anonymen Nutzern aufgerufen werden.

register.xhtml

Art	Name	Beschreibung	Sichtbarkeit
InputField	username	Hier kann ein Nutzernname festgelegt werden.	AY
InputField	password	Hier kann ein Passwort festgelegt werden.	AY
InputField	email	Hier kann eine Email festgelegt werden.	AY
InputField	name	Hier kann der Vorname eingegeben werden.	AY
InputField	lastName	Hier kann der Nachname eingegeben werden.	AY
InputField	street	Hier kann die Straße der Adresse eingegeben werden.	AY
InputField	houseNumber	Hier kann die Hausnummer der Adresse eingegeben werden.	AY
InputField	addressAddition	Hier kann ein Zusatz zur Adresse eingegeben werden.	AY
InputField	city	Hier kann der Stadtnamen eingegeben werden.	AY
InputField	postalCode	Hier kann die Postleitzahl eingegeben werden.	AY
InputField	phone	Hier kann die Telefonnummer eingegeben werden.	AY
InputFile	avatar	Hier kann ein Profilbild hochgeladen werden,	AY
Link	login	Weiterleitung an die Login Seite.	AY
Button	submit	Versucht mit den gegebenen Daten zu registrieren.	AY

Email Verifikation Diese Seite ermöglicht dem Nutzer den Code aus der Registrierungsemail einzugeben, um die Registrierung abzuschließen. Das InputField wird dabei falls möglich mit einem Code aus den View Parametern vorbefüllt.

emailVerification.xhtml

Art	Name	Beschreibung	Sichtbarkeit
InputField	code	Hier kann der Verifikationscode eingegeben werden.	AY
Button	submit	Versucht mit dem gegebenen Code die Registrierung abzuschließen.	AY

Übersicht/Suche Diese Seite ermöglicht dem Nutzer alle Anzeigen einzusehen und zu durchsuchen.

allAdvertisements.xhtml

Art	Name	Beschreibung	Sichtbarkeit
DynamicDataTable	requestTable	Hier können alle Anzeigen aufgelistet und durchsucht werden.	AN
SortableDataColumn	image	Zeigt das Hauptbild der Anzeige.	AN
SortableDataColumn	title	Zeigt den Titel der Anzeige.	AN
SortableDataColumn	points	Zeigt die Kosten der Anzeige in Talentpunkten.	AN
SortableDataColumn	postalCode	Zeigt die Postleitzahl des Anzeigenerstellers.	AN

Anzeigen Details Diese Seite ermöglicht dem Nutzer eine Anzeigen genauer anzusehen. Von hier aus werden auch Anfragen getätigter.

advertisementDetails.xhtml

Art	Name	Beschreibung	Sichtbarkeit
Image	AdImage	Großansicht des Anzeigenbildes.	AN
Image	avatar	Profilbild des Anzeigenerstellers.	AN
Label	user	Vor und Nachname des Anzeigenerstellers.	AN
Label	title	Titel der Anzeige.	AN
Paragraph	description	Freitext der bei der Anzeigenerstellung verfasst wurde.	AN
Label	city	Stadt des Anzeigenerstellers.	AN
Label	postalCode	Postleitzahl des Anzeigenerstellers.	AN
Label	address	Straße, Hausnummer und gegebenenfalls Adresszusatz des Anzeigenerstellers.	N / A
Label	email	E-Mail-Adresse des Anzeigenerstellers.	N / A
Label	phonenumbers	Telefonnummer des Anzeigenerstellers.	N / A
Button	request	Durch Klicken wird die Anfrage schreiben Seite geladen.	N / A

5.3.2 Angemeldete Nutzer

Die folgenden Seiten können von angemeldeten Nutzern aufgerufen werden.

Mein Profil Auf dieser Seite befinden sich alle notwendigen Bedienelemente, um einen Benutzer zu bearbeiten oder zu löschen. Alle Felder sind zu Beginn mit den aktuellen Werten vorbeifüllt.

profile.xhtml

Art	Name	Beschreibung	Sichtbarkeit
Image	avatar	Aktuelles Profilbild des Nutzers.	N / A
InputField	username	Setzt den sichtbaren Nutzernamen.	N / A
InputField	password	Setzt das Passwort.	N / A
InputField	email	Setzt die Email-Adresse.	N / A
InputField	Name	Setzt den Vornamen des Nutzers.	N / A
InputField	lastName	Setzt den Nachnamen des Nutzers.	N / A
InputField	street	Setzt die Straße in der Adresse des Nutzers.	N / A
InputField	houseNumber	Setzt die Hausnummer in der Adresse des Nutzers.	N / A
InputField	addressAddition	Setzt den Zusatz in der Adresse des Nutzers.	N / A
InputField	city	Setzt die Stadt in der Adresse des Nutzers.	N / A
InputField	postalcode	Setzt die Postleitzahl in der Adresse des Nutzers.	N / A
InputField	phonenumbers	Setzt die Telefonnummer des Nutzers.	N / A
InputFile	NewAvatar	Setzt das Profilbild des Nutzers.	N / A
Button	deleteAvatar	Löscht falls gegeben das Profilbild.	N / A
Button	update	Speichert alle Änderungen der Daten.	N / A
Button	delete	Löscht den aktuellen Benutzer.	N / A

Anzeige erstellen/bearbeiten/löschen Auf dieser Seite befinden sich alle notwendigen Bedienelemente, um eine Anzeige zu erstellen, zu bearbeiten oder zu löschen. Sollte eine Anzeige erstellt werden, sind alle Felder zu Beginn leer. Sollte eine Anzeige bearbeitet werden, sind alle Felder zu Beginn mit den aktuellen Werten vorbeifüllt.

editAdvertisement.xhtml

Art	Name	Beschreibung	Sichtbarkeit
InputField	title	Setzt den Anzeigentitel.	N / A
InputFile	adImage	Setzt das Titelbild der Anzeige.	N / A
InputField	cost	Setzt die Kosten der Anzeige in Talentpunkten.	N / A
InputField	description	Setzt den Freitext der Anzeige.	N / A
InputField	city	Stadt der Anzeige.	AN
InputField	postalCode	Postleitzahl der Anzeige.	AN
InputField	street	Setzt die Straße in der Adresse der Anzeige.	N / A
InputField	houseNumber	Setzt die Hausnummer der Anzeige.	N / A
InputField	addressAddition	Setzt den Zusatz in der Adresse der Anzeige.	N / A
InputField	email	E-Mail-Adresse der Anzeige.	N / A
InputField	phonenumbers	Telefonnummer der Anzeige.	N / A
Checkbox	showPhone	Bestimmt ob die Telefonnummer des Erstellers sichtbar ist.	N / A
Checkbox	showAddress	Bestimmt ob die volle Adresse des Erstellers sichtbar ist.	N / A
Checkbox	showName	Bestimmt ob der echte Name des Erstellers sichtbar ist.	N / A
Checkbox	hidden	Bestimmt ob die Anzeige von Anderen gefunden werden kann.	N / A
Button	deleteImage	Löscht falls gegeben das Titelbild der Anzeige.	N / A
Button	update	Speichert alle Änderungen der Daten.	N / A
Button	delete	Löscht die aktuelle Anzeige.	N / A

Anfrage beantworten Diese Seite ermöglicht dem Nutzer auf eine Anfrage zu antworten.

writeResponse.xhtml

Art	Name	Beschreibung	Sichtbarkeit
InputField	answer	Hier kann ein Freitext verfasst werden.	N / A
Button	accept	Durch Klicken wird die Anfrage akzeptiert.	N / A
Button	reject	Durch Klicken wird die Anfrage abgelehnt.	N / A

Eingegangene Anfragen + Antworten Diese Seite ermöglicht dem Nutzer einkommende Anfragen und die gegebenen Antworten einzusehen.

incomingRequests.xhtml

Art	Name	Beschreibung	Sichtbarkeit
DynamicDataTable	requestTable	Hier können alle eingehenden Anfragen aufgelistet und durchsucht werden.	N / A
SortableDataColumn	user	Zeigt den Anfragensteller.	N / A
SortableDataColumn	advertisement	Zeigt die Anzeige die Angefragt wurde.	N / A
SortableDataColumn	request	Zeigt Anfang des Anfragetextes.	N / A
SortableDataColumn	answer	Zeigt Anfang des Antworttextes.	N / A
SortableDataColumn	writeResponse	Durch Klicken wird ein Form zum Antworten geladen.	N / A
Link	createdAdvertisements	Durch Klicken wird die Seite der erstellten Anzeigen geladen.	N / A
Link	userAdministration	Durch Klicken wird die Nutzerverwaltung geladen.	A

Erstellte Anzeigen Diese Seite ermöglicht dem Nutzer eigens erstellte Anzeigen einzusehen.

createdAdvertisements.xhtml

Art	Name	Beschreibung	Sichtbarkeit
DynamicDataTable	requestTable	Hier können alle erstellten Anzeigen aufgelistet und durchsucht werden.	N / A
SortableDataColumn	image	Zeigt das Hauptbild der Anzeige.	N / A
SortableDataColumn	title	Zeigt den Titel der Anzeige.	N / A
SortableDataColumn	points	Zeigt die Kosten der Anzeige in Talentpunkten.	N / A
SortableDataColumn	editAdvertisement[row]	Durch Klicken wird ein Form zum Bearbeiten geladen.	N / A
Button	new	Durch Klicken wird ein Form zum Erstellen einer neuen Anzeige geladen.	N / A
Link	incomingRequests	Durch Klicken wird die Seite der eingehenden Anfragen geladen.	N / A

Ausgehende Anfragen + Antworten Diese Seite ermöglicht dem Nutzer ausgehende Anfragen und die gegebenen Antworten einzusehen.

outgoingRequests.xhtml

Art	Name	Beschreibung	Sichtbarkeit
DynamicDataTable	requestTable	Hier können alle ausgehenden Anfragen aufgelistet und durchsucht werden.	N / A
SortableDataColumn	user	Zeigt den Anzeigenersteller.	N / A
SortableDataColumn	advertisement	Zeigt die Anzeige die Angefragt wurde.	N / A
SortableDataColumn	request	Zeigt Anfang des Anfragetextes.	N / A
SortableDataColumn	answer	Zeigt Anfang des Antworttextes.	N / A
Link	requestedAdvertisements	Durch Klicken wird die Seite der beanspruchten Anzeigen geladen.	N / A
Link	userAdministration	Durch Klicken wird die Nutzerverwaltung geladen.	A

Beanspruchte Anzeigen Diese Seite ermöglicht dem Nutzer eigens beanspruchte Anzeigen einzusehen.

claimedAdvertisements.xhtml

Art	Name	Beschreibung	Sichtbarkeit
DynamicDataTable	requestTable	Hier können alle beanspruchten Anzeigen aufgelistet und durchsucht werden.	N / A
SortableDataColumn	image	Zeigt das Hauptbild der Anzeige.	N / A
SortableDataColumn	title	Zeigt den Titel der Anzeige.	N / A
SortableDataColumn	points	Zeigt die Kosten der Anzeige in Talentpunkten.	N / A
Link	outgoingRequests	Durch Klicken wird die Seite der ausgehenden Anfragen geladen.	N / A

Anfrage schreiben Diese Seite ermöglicht dem Nutzer eine Anfrage für eine Anzeige zu schreiben.

writeRequest.xhtml

Art	Name	Beschreibung	Sichtbarkeit
Image	mainImage	Kleinansicht des Hauptbildes.	N / A
Image	avatar	Profilbild des Anzeigenerstellers.	N / A
Label	user	Vor und Nachname des Anzeigenerstellers.	N / A
Label	title	Titel der Anzeige.	N / A
TextInput	description	Freitext der der Anfrage angehängt wird.	N / A
Button	submit	Durch Klicken wird die Anfrage abgeschickt.	N / A

5.3.3 Administrator

Die folgenden Seiten können nur von einem Administrator aufgerufen werden.

Nutzerverwaltung Diese Seite ermöglicht dem Admin alle Nutzer einzusehen und zu bearbeiten.

userAdministration.xhtml

Art	Name	Beschreibung	Sichtbarkeit
DynamicDataTable	userTable	Ermöglicht die Einsicht und Bearbeitung der Nutzer.	A
SortableDataColumn	avatar	Profilbild des Nutzers.	A
SortableDataColumn	Name	Nutzername des Nutzers.	A
SortableDataColumn	points	Kontostand in Talentpunkten des Nutzers.	A
SortableDataColumn	AVerified	Verifizierungsstatus durch den Admin.	A
SortableDataColumn	edit	Lädt das Form zum Bearbeiten des Nutzers.	A
Button	new	Lädt das Form zum Erstellen eines Nutzers.	A

Nutzer erstellen/bearbeiten/löschen In diesem Form befinden sich alle notwendigen Bedienelemente, um einen Benutzer zu erstellen, zu bearbeiten oder zu löschen. Sollte ein Benutzer erstellt werden, sind alle Felder zu beginn leer. Sollte ein Nutzer bearbeitet werden, sind alle Felder zu beginn mit den aktuellen Werten vorbefüllt.

editUser.xhtml

Art	Name	Beschreibung	Sichtbarkeit
Image	avatar	Aktuelles Profilbild des Nutzers.	A
InputField	username	Setzt den sichtbaren Nutzernamen.	A
InputField	password	Setzt das Passwort.	A
InputField	email	Setzt die Email-Adresse.	A
InputField	Name	Setzt den Vornamen des Nutzers.	A
InputField	lastName	Setzt den Nachnamen des Nutzers.	A
InputField	street	Setzt die Straße in der Adresse des Nutzers.	A
InputField	houseNumber	Setzt die Hausnummer in der Adresse des Nutzers.	A
InputField	addressAddition	Setzt den Zusatz in der Adresse des Nutzers.	A
InputField	city	Setzt die Stadt in der Adresse des Nutzers.	A
InputField	postalcode	Setzt die Postleitzahl in der Adresse des Nutzers.	A
InputField	phonenumbers	Setzt die Telefonnummer des Nutzers.	A
InputField	skillPoints	Setzt die Talentpunkte des Nutzers.	A
InputFile	NewAvatar	Setzt das Profilbild des Nutzers.	A
Checkbox	Averifikation	Setzt den Verifizierungsstatus durch den Admin	A
Button	deleteAvatar	Löscht falls gegeben das Profilbild.	A
Button	update	Speichert alle Änderungen der Daten.	A
Button	delete	Löscht den aktuellen Benutzer.	A

6 Systemfunktionen

LEON FÖCKERSPERGER

In diesem Kapitel werden die zentralen Funktionalitäten des Systems detailliert beschrieben. Ziel ist es, einen Überblick zu geben, wie die einzelnen Funktionen zur Sicherstellung eines effizienten und störungsfreien Betriebs beitragen.

6.1 Lifecycle

6.1.1 Systemstart

Die Initialisierung der Anwendung erfolgt in der `business`-Layer, koordiniert durch die Methode `StartAndStopListener.init` im `lifecycle`-Paket. Diese Methode ist mit der Annotation `@Observes @Initialized(ApplicationScoped.class)` versehen, die den Jakarta EE-Container instruiert, sie zu starten, sobald die Anwendung betriebsbereit ist. Diese Annotation ermöglicht eine präzise Steuerung des Initialisierungszeitpunkts und garantiert, dass die Methode erst nach dem vollständigen Laden aller Abhängigkeiten und Konfigurationen ausgeführt wird. Dadurch wird eine optimale Vorbereitung der Anwendung sichergestellt. Vor der Durchführung der Initialisierungsprozesse in der `business`-Layer initiiert diese Methode die `StartAndStopPersistence.init`-Methode der `persistence`-Layer, welche die Initialisierungsprozesse ihrer Schicht übernimmt.

Systemstart-Aufgaben: Der Systemstart erfolgt in folgender festgelegter Reihenfolge:

Laden der Logging-Konfiguration (persistence-Layer): Die Konfigurationsdatei `logger_config.properties` wird durch `LoggerProducer.init` geladen. Anschließend wird sie an `java.util.logging.LogManager.readConfiguration` übergeben.

Laden der Datenbank-Konfiguration (persistence-Layer): Die Datei `db_config.properties` wird in der Methode `DatabaseConfig.init` verarbeitet, um die notwendigen Systemverbindungen zu etablieren.

Initialisierung der Datenbankverbindung (persistence-Layer): Der Datenbank-Connection-Pool wird durch `ConnectionPool.getInstance` auf Basis der durch `DatabaseConfig` bereitgestellten Verbindungsdaten eingerichtet.

Initialisierung des Datenbankschemas (persistence-Layer): Das Vorhandensein des Datenbankschemas wird durch `StoreSetup.schemaExists` überprüft und bei Bedarf durch `StoreSetup.createSchema` bzw. `StoreSetup.initializeData` neu erstellt.

Laden der SMTP-Konfigurationsdaten (business-Layer): Die Datei `mail_config.properties` wird in der Methode `EmailConfig.init` verarbeitet, um die Verbindung zum E-Mail-Server herzustellen.

Initialisierung des Shutdown-Hooks (business-Layer): Ein Shutdown-Hook wird in der `business-Layer` eingerichtet, der bei Auslösung die `destroy`-Methode der Klasse `StartAndStopPersistence` ausführt.

6.1.2 Systemstop

Der Anhaltungsprozess des Systems wird durch die Methode `StartAndStopListener.destroy` in der `backing-Layer` gesteuert, ausgelöst durch die Annotation `@Observes @Destroyed(ApplicationScoped.class)`. Dies leitet den Aufruf der `StartAndStopPersistence.destroy`-Methode in der `persistence-Layer` ein.

Systemstop-Aufgaben:

Schließen der Datenbankverbindung (persistence-Layer): Der Datenbank-Connection-Pool wird durch `ConnectionPool.destroy` abgebaut, um die Datenbankverbindungen sicher zu trennen.

6.2 Konfiguration

Die Systemkonfiguration kann durch Modifikation der `.properties` Dateien im `WEB-INF/config` Verzeichnis angepasst werden. Diese Dateien erlauben die Anpassung kritischer Systemparameter wie Verbindungsdaten zur PostgreSQL-Datenbank, Konfigurationen des SMTP-Mail-Servers und Einstellungen für das Logging. Wie im Abschnitt [6.1.1](#) beschrieben, lädt das System diese Konfigurationsdateien beim Startvorgang. Änderungen an den Konfigurationsdateien erfordern daher einen Neustart der Anwendung, um die aktualisierten Einstellungen zu übernehmen. Dies gewährleistet, dass das System stets mit den neuesten Konfigurationsdaten operiert und trägt zur Stabilität und Sicherheit der Anwendung bei.

6.3 Logging

Das Logging-Subsystem der Anwendung dient dazu, ein detailliertes Protokoll aller systemrelevanten Ereignisse zu erstellen. Besonders im Falle von Fehlern werden relevante Daten aufgezeichnet, um die Ursachenanalyse zu unterstützen.

Für das Logging verwendet die Anwendung das `java.util.logging` Framework. Die Konfiguration dieses Frameworks kann durch den Administrator angepasst werden, um unterschiedliche Protokollierungsstufen und Ausgabeformate zu unterstützen (siehe Abschnitt [6.2](#)). Um eine kontextsensitive Loggenerierung zu ermöglichen, verwendet die Anwendung die Methode `LoggerProducer.produceLogger`. Diese Methode fungiert als CDI-Producer und ermöglicht die Injektion von `Logger`-Instanzen, die automatisch den Namen der aufrufenden Klasse tragen. Da die Persistenzschicht der Anwendung frei von CDI (Contexts and Dependency Injection) bleibt, wird in diesem Bereich das `Logger`-Objekt manuell mittels `Logger.getLogger` erzeugt und zugewiesen; dies passiert in der Methode `LoggerProducer.get`. Dies stellt sicher, dass die Logik der Datenspeicherung klar von der Anwendungslogik getrennt ist und die Unabhängigkeit der Schichten gewahrt bleibt.

6.4 E-Mail

Das System implementiert eine robuste E-Mail-Funktionalität, um verschiedene Schlüsseloperationen zu unterstützen, wie die Verifizierung neuer Benutzerkonten und das Zurücksetzen von Passwörtern. Die Handhabung des E-Mail-Versands wird von der Klasse `EmailDispatcher` im `util`-Paket der `business-Layer` übernommen, welche die Jakarta Mail API verwendet.

Die `EmailDispatcher`-Klasse abstrahiert die Komplexität der direkten Nutzung der Jakarta Mail API und bietet eine vereinfachte Schnittstelle für andere Systemkomponenten. Dies ermöglicht es, E-Mail-bezogene Aktionen effizient zu integrieren und zu verwalten. Konfigurationsparameter, wie SMTP-Serverdetails und Authentifizierungsdaten, werden in den `.properties`-Dateien definiert und können über die Systemkonfiguration (siehe Abschnitt [6.2](#)) angepasst werden, um die Flexibilität und Sicherheit des E-Mail-Dienstes zu gewährleisten.

6.5 Zugriffskontrolle (TrespassListener)

Das System definiert mehrere Nutzerrollen zur Verwaltung der Zugriffsrechte auf seine Ressourcen:

- **Anonym** – Zugriffsbereich `anonymous`
- **Angemeldeter Nutzer** – Zugriffsbereich `authenticated`
- **Administrator** – Zugriffsbereich `admin`

Es wird eine Hierarchie der Zugriffsberechtigungen festgelegt: `anonymous ⊂ authenticated ⊂ admin`.

Die Zugriffskontrolle wird über zwei Ebenen verwaltet:

1. **Systemebene:** Die Klasse `TrespassListener`, die von `jakarta.faces.event.PhaseListener` erbt, überwacht den Zugang zu Systemseiten. Der Listener wird nach der Restore-View-Phase im JF-Request-Lifecycle durch den Rückgabewert `PhaseId.RESTORE_VIEW` von `TrespassListener.getPhaseId` aktiviert. In der Methode `TrespassListener.afterPhase` wird die aktuelle URL, abgerufen durch `UIViewRoot.getViewId`, gegen die berechtigten Zugriffsrechte des Nutzers, ermittelt durch `Session.getCurrentUserRole`, geprüft. Seiten-URLs sind typischerweise mit `/<Zugriffsbereich>/*.xhtml` kodiert, was eine direkte Zuordnung der URL zum Zugriffsbereich ermöglicht. Bei einer fehlenden Berechtigung wird entweder eine Weiterleitung zur 404-Fehlerseite oder zur Startseite über `jakarta.faces.context.FacesContext.NavigationHandler.handleNavigation` initiiert.
2. **UI-Komponentenebene:** Die `init`-Methode annotiert mit `@PostConstruct` in den Managed Beans bestimmt, welche UI-Elemente dem Nutzer angezeigt werden. Dies geschieht durch eine Überprüfung der Rolle des Nutzers innerhalb der aktuellen Session. Abhängig von den Zugriffsrechten werden entsprechende UI-Komponenten ein- oder ausgeblendet.

Diese strukturierte Herangehensweise an die Zugriffskontrolle sorgt für eine klare Trennung zwischen System- und UI-Ebene, was die Sicherheit erhöht und die Wartbarkeit der Zugriffslogik vereinfacht.

6.6 Systemsicherheit

Dieser Abschnitt behandelt präventive Maßnahmen gegen ausnutzende Angriffe, die auf gängige Sicherheits schwachstellen von Webanwendungen abzielen und für unser System relevant sind.

Kryptographie & Vertraulichkeit sensibler Daten Die Sicherstellung der Vertraulichkeit sensibler Daten wird durch ausschließlich verschlüsselte HTTPS-Verbindungen erreicht, konfiguriert in der `web.xml`-Datei der Anwendung. Passwörter werden unter Einsatz der SHA-512-Hashfunktion und einem einzigartigen, zufällig generierten Salt für jedes Passwort gehasht. Die Hashfunktion wird in der Methode `hash` der Klasse `PasswordAuthentication` angewendet, die während der Authentifizierung im `UserService` aufgerufen wird.

Injection & Cross-site-Scripting Um SQL-Injection-Angriffe zu verhindern, verwendet unser System Prepared Statements. Diese Methode trennt den SQL-Befehl von den Nutzereingaben, indem sie den SQL-Code vordefiniert und Daten erst zur Ausführungszeit einbindet. Dies verhindert die Ausführung von eingeschleustem Schadcode, da Eingabedaten nur als Parameter behandelt und nicht als Teil des SQL-Codes ausgeführt werden. Gegen Cross-Site-Scripting (XSS) schützt das System durch den Einsatz von Standard-JSF-Komponenten, die alle Benutzereingaben automatisch escapen. Diese Sicherheitsmaßnahme blockiert die Ausführung schädlicher HTML- oder JavaScript-Codes und erhöht somit die Sicherheit der Anwendung.

Session Hijacking In der `web.xml`-Datei kann ein Session-Timeout definiert werden, der als Schutzmechanismus gegen Session Hijacking dient. Außerdem wird in Jakarta Faces (JF) standardmäßig das `http-only`-Flag für Cookies aktiviert, was den Zugriff auf das Session-Cookie über JavaScript unterbindet. Nach jedem Login wird die vorhandene Session durch den Aufruf von `ExternalContext.invalidateSession` beendet und durch den Aufruf von `HttpServletRequest.getSession(true)` sofort eine neue Session gestartet, um Session-Fixation-Angriffe effektiv zu verhindern.

6.7 Mehrbenutzerbetrieb

Das System garantiert die Konsistenz des Systemzustands auch bei simultaner Nutzung durch mehrere Benutzer. Dies ist besonders wichtig, da der ConnectionPool die gleichzeitige Ausführung von Datenänderungen durch mehrere Benutzer ermöglicht.

Transaktionsmanagement Zur Vermeidung von Inkonsistenzen nutzt das System Transaktionen, die in der persistence-Layer durch das Interface `Transaction` verwaltet werden. Dies ermöglicht es, mehrere voneinander abhängige Operationen als eine einzige atomare Einheit auszuführen. Die Klasse `PostgreSQLTransaction`, die das `Transaction`-Interface implementiert, bezieht eine aktive Datenbankverbindung aus dem `ConnectionPool`. Standardmäßig ist das `autocommit`-Verhalten aller Datenbankverbindungen im `ConnectionPool` deaktiviert (`autocommit = false`). Nach Abschluss aller Operationen innerhalb der Transaktion wird die Methode `Transaction.commit` aufgerufen, um die Änderungen in der Datenbank zu bestätigen. Falls während der Transaktion Fehler auftreten, ermöglicht das Rollback-Verfahren durch `Transaction.rollback` das Zurücksetzen aller bis dahin durchgeföhrten Änderungen, wodurch die Datenintegrität gewahrt bleibt.

7 Datenfluss

DAVID STURM

7.1 Erfolgreiches Bearbeiten einer Anzeige

Hier folgend wird beispielhaft für den Datenfluss im Programm ein Sequenzdiagramm für das Bearbeiten einer Anzeige. Es wird angenommen, dass ein angemeldeter Nutzer eine bereits zuvor von ihm erstellte Anzeige aufruft um diese zu bearbeiten.

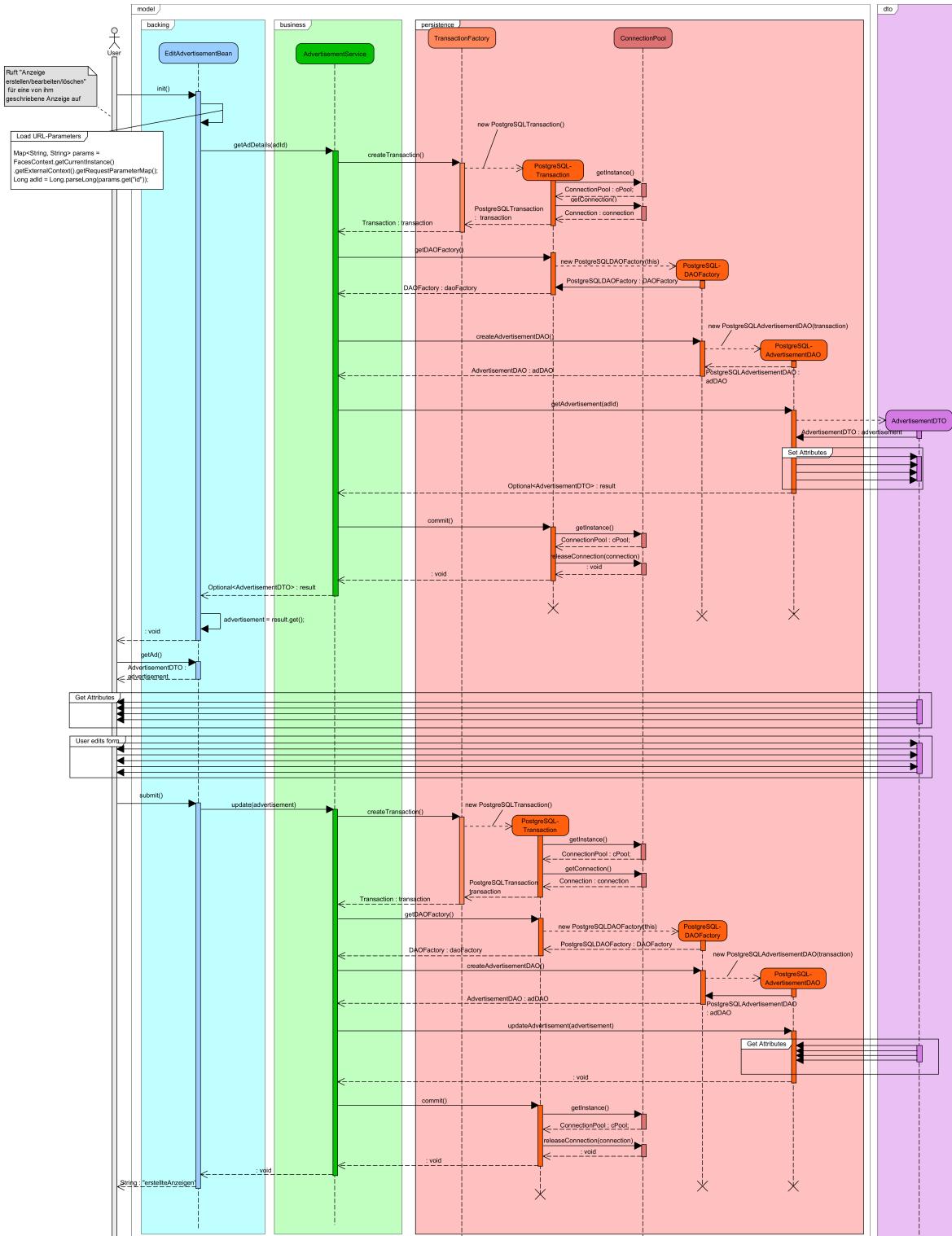


Abbildung 19: Flussdiagramm Erfolg

7.2 Fehlgeschlagenes Bearbeiten einer Anzeige

Hier folgend wird beispielhaft die Fehlerbehandlung bei einem auftretenden fehler in der Datenbankübertragung während des selben Programmflusses wie zuvor.

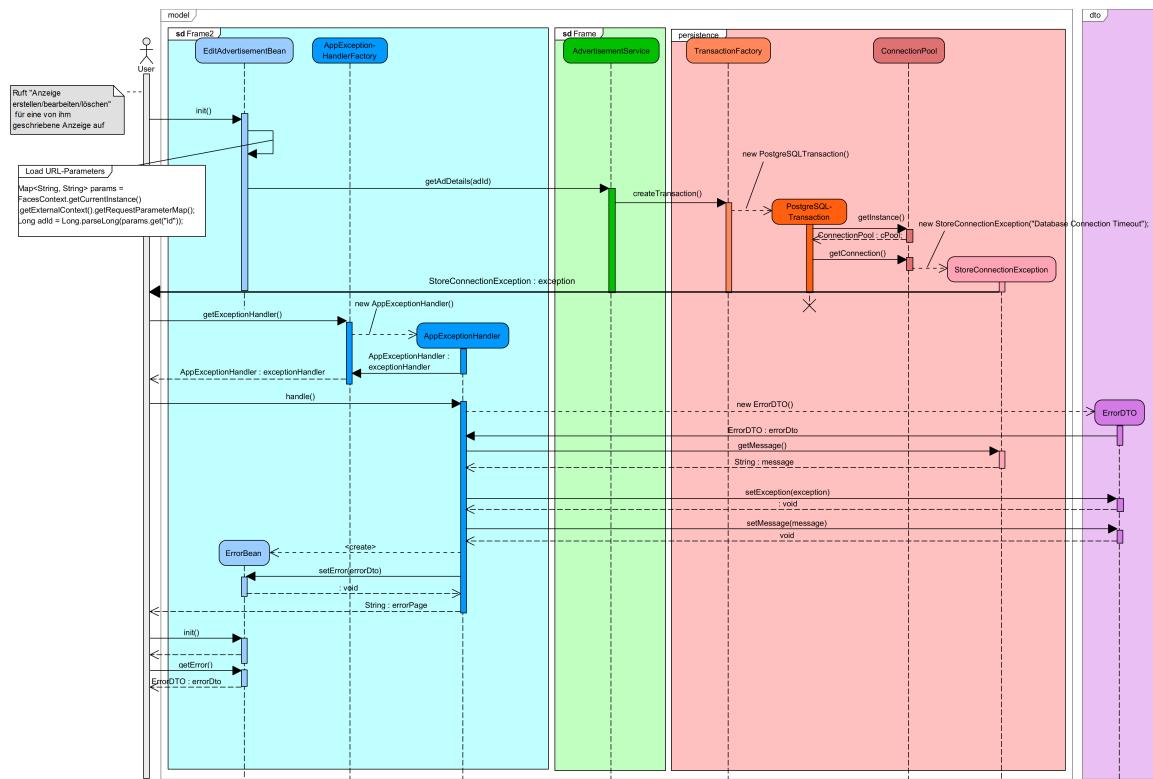


Abbildung 20: Flussdiagramm Misserfolg

8 ER-Modell

LEON FÖCKERSPERGER

Das **Entity-Relationship-Diagramm** (ERD), dargestellt in Abbildung 21, visualisiert die gesamte persistente Datenstruktur der Webapplikation *Talenttauscher*. Die *Kardinalität* der Beziehungen im Diagramm repräsentiert die ganzzahligen Abhängigkeiten zwischen den *Entitäten*. Des Weiteren werden durch *is-A*-Spezialisierungen die unterschiedlichen Subtypen einer *Entität* modelliert. Eine schwache *Entität* wird im Diagramm durch eine doppelte Umrandung dargestellt. *Primär Schlüssel* einer *Relation* sind durch fett und unterstrichene Attribute hervorgehoben, während *Fremdschlüssel*-Attribute kursive und unterstrichene Darstellung finden. Attribute, die *unique* sind, werden lediglich kursiv dargestellt.

- **User:**
 - Enthält Informationen über Benutzer wie ID, Name, Profilbild, Spitzname, Talentpunkte und mehr.
 - Hat Attribute für die E-Mail-Adresse und das Passwort (sicherheitshalber gespeichert als Hash und Salt), Hausnummer und zusätzliche Adressinformationen.
 - Es gibt zwei Arten von user: *administrator* und *registered users*.
- **System:**
 - Speichert allgemeine systembezogene Einstellungen und Konfigurationen, wie die maximal zulässige Dateigröße für Uploads, Datenschutzinformationen und Impressum.
- **Ad:**
 - Repräsentiert *Gebot Anzeigen*, die von registrierten Benutzern erstellt wurden.
 - Beinhaltet Informationen wie Titel, Kosten in Talentpunkten, Freitext für Details, Veröffentlichungsdatum und Erstellerdetails (Name, Spitzname, Adresse usw.).

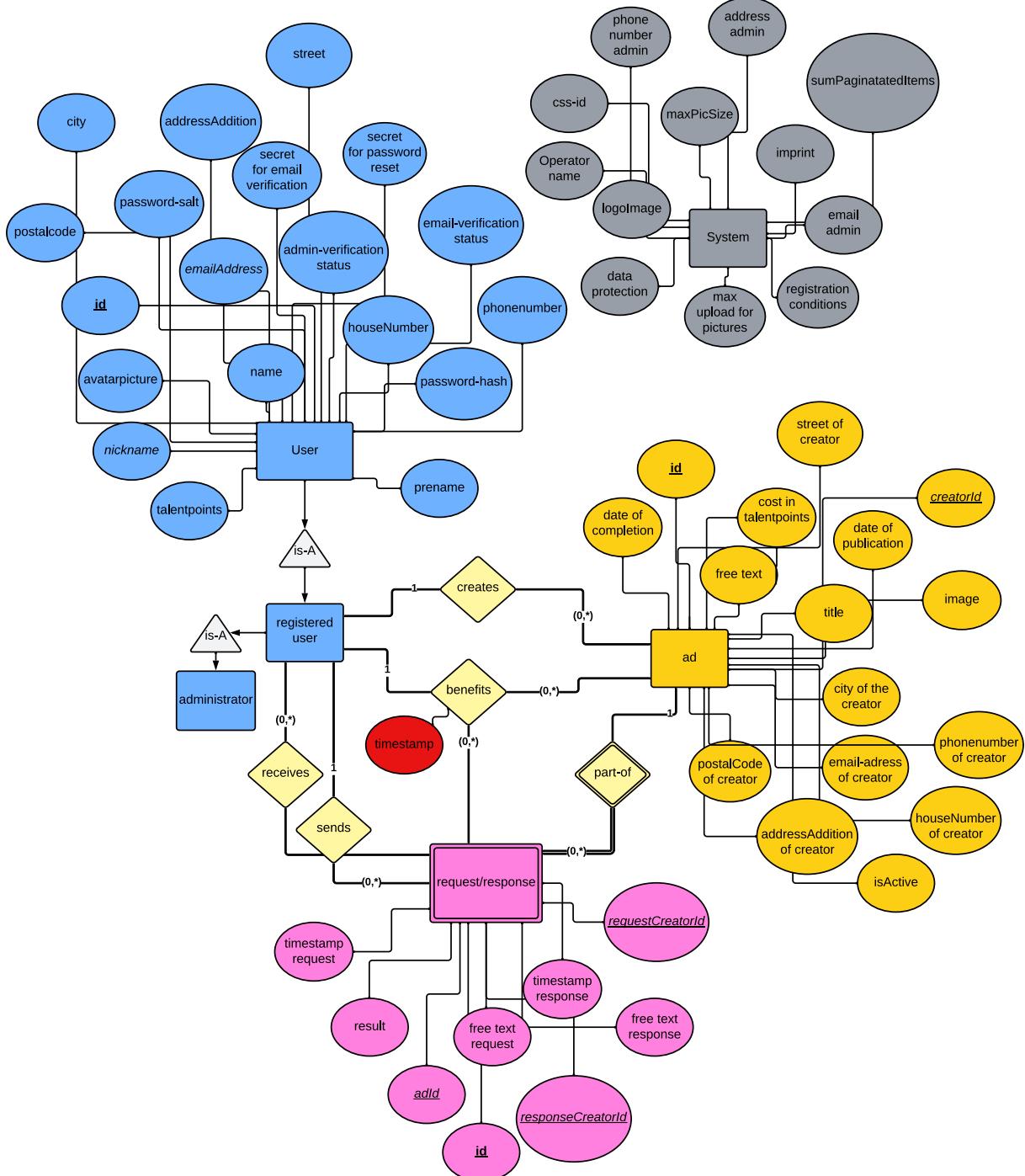


Abbildung 21: Entity-Relationship-Diagramm der Datenstruktur von Talenttauscher.

- **Anfrage/Antwort:**

- Verwaltet Anfragen und Antworten in einem Datensatz. Zeitstempel und Freitextfeld werden jeweils für Anfrage und Antwort gespeichert.
- Die Beziehung zum Anzeigenersteller, also dem Empfänger einer Anfrage, wird über die Beziehung zur Anzeige geregelt, da dort der Ersteller der Anzeige gespeichert wird. Die Verbindung zum Anfragenden wird über ein Fremdschlüsselattribut festgehalten.