mpuls WASKA - Systemdokumentation

Web-Applikations-Server für Kompetenzagenturen

MITWIRKENDE

	TITEL :	*REFERENCE* :
	mpuls WASKA - Systemdokumentation	

AKTION	*NAME*	*DATUM*	*UNTERSCHRIFT*
VERFASST DURCH	Torsten Irlaender	4. September 2008	

VERSIONSGESCHICHTE

NUMMER	DATUM	BESCHREIBUNG	NAME

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Zusammenfassung

Dieses Dokument beschreibt die Systemarchitektur eines mpuls WASKA-Servers und soll ein grundsätzliches Verständnis für das Zusammenwirken der verschiedenen Komponenten von mpuls WASKA ermöglichen.

1 Einführung

mpuls ist ein Web-Applikations-System zur Führung einer computergestützten Fallaktendokumentation basierend auf Freie Software Komponenten. WASKA (Web-Applikations-Server für Kompetenzagenturen) ist implementiert eine Fallakte für das Case-Management (zunächst für Kompetenzagenturen) auf dieser Basis.

Dieses Dokument beschreibt die Architektur von mpuls WASKA. Dabei werden sowohl die einzelnen Komponenten vorgestellt, als auch deren Zusammenspiel erklärt.

2 Grundlegende Komponenten des Systems

mpuls WASKA kann zur Darstellung des Systems in vier Bereiche gegliedert werden:

- Nutzer
- Web-Anwendung
- Datenbank-Management-System (DBMS)
- Optional: Auswerteserver

Ein wichtiges Merkmal ist die strikte Trennung zwischen Applikationslogik in dem Applikationsserver und der Datenhaltung auf dem Datenbankserver.

Der Benutzer kann sich mit Hilfe eines Web-Browser (Browser) bei der Web-Anwendung anmelden. Neben Anmeldekennung und Passwort benötigt der Benutzer dazu ein Zertifikat, das auch zur Identifikation der Kompetenzagentur dient. Die gesamte Kommunikation erfolgt verschlüsselt über SSL (mindestens 128 bit Verschlüsselung). Ist die Anmeldung erfolgreich, bekommt der Nutzer von der Web-Anwendung, und dem als Datenspeicher dienenden Datenbank-Management-System (DBMS) Rechte zugesichert, die es ihm ermöglichen, über den Browser Daten im System abzulegen, zu verändern, auszulesen oder zu löschen.

Die Web-Anwendung beinhaltet die gesamte Anwendungslogik. Der Server verarbeitet Anfragen der Nutzer, koordiniert Abfragen an den Datenbankserver und liefert eine Ergebnisseite an den Nutzer aus.

Das Datenbanksystem hält die Daten der Kompetenzagenturen. Die Daten mehrerer Kompetenzagenturen können von einem Datenbanksystem verwaltet werden. In diesem Fall wird innerhalb des Datenbanksystems für jede Kompetenzagentur eine eigene Datenbank erstellt. Auf welcher der Datenbanken ein Benutzer arbeitet, wird über einen im Zertifikat hinterlegten Wert festgelegt.

Der vierte Bereich beschreibt den optionalen Auswerteserver. Eine Einbindung eines solchen Auswerteserver ist für den Grundbetrieb nicht notwendig, soll aber zur Vollständigkeit erwähnt werden. Ein Auswerteserver ist ein unter einer speziellen Konfiguration laufender mpuls WASKA-Server. Der Aufbau ist dabei identisch zu einem normalen mpuls WASKA-Server. Er unterscheidet sich in seiner Benutzeroberfläche und der dem Nutzer zur Verfügung stehenden Funktionen: Er dient dazu, die Daten verschiedener Agenturen in anonymisierter Form entgegenzunehmen und an zentraler Stelle verfügbar zu machen. Die Daten der einzelnen Kompetenzagenturen werden anonymisiert in regelmäßigen Intervallen über eine SSH-Verbindung auf dem Auswerteserver geladen.

3 Webanwendung

Die Web-Anwendung lässt sich wiederum in vier Teile gliedern:

- Apache Webserver
- Pylons Application Framework
- Optional: PDF Import
- Optional: memcache Dienst

PDF-Importer und Memcache sind optional und nicht zwingend für den Betrieb von mpuls WASKA notwendig sind. Ein Verzicht auf dem PDF-Importer bedeutet allerdings, dass kein Import von PDF-Fallakten möglich ist. Im folgendem werden die einzelnen Komponenten der Web-Applikation vorgestellt.

3.1 Apache Webserver

Als Webserver wurde der Apache Webserver gewählt. Dies hat mehrere Gründe. Der Apache Webserver ist der weltweit am häufigsten eingesetzte Web-Server[1]. Die Qualität und Robustheit des Apache Webservers hat sich in der Praxis bewährt. Weiter existiert um den Webserver herum eine aktive Entwicklergemeinde, die sich um die Pflege und Erweiterung der Software kümmert. Ein weiterer wichtiger Grund ist die Flexibilität des Server. Durch seinen modularen Aufbau lässt sich der Server durch Einbinden weiterer Module einfach in seiner Funktionalität erweitern. Von mpuls WASKA werden insbesondere folgende Module genutzt:

- *mod_ssl:* Die verschlüsselte Übertragung der Daten ist eine der zentralen Anforderung an WASKA. Das mod_ssl Modul stellt die Verschlüsselung der Inhalte zwischen der Applikation und dem Benutzer sicher. Es dient weiter zur Überprüfung der Authenzität der Kommunikationspartner anhand der X.509 Zertifikate und stellt die Informationen aus den Zertifikaten für die Web-Anwendung bereit.
- *mod_wsgi:* Das mod_wsgi[2] Modul dient zur Einbindung des Pylons Webframework über die in PEP333[3] standarderisierte WSGI (Web Server Gateway Interface) Schnittstelle in den Apache Server. Gegenüber der alternativen Möglichkeit einer Anbindung über mod_python[4] ermöglicht mod_wsgi eine bessere Performanz und höhere Portabilität und Flexibilisierung der Anwendung.

Durch die Möglichkeit über die mod_wsgi Schnittstelle die Pylons Webanwendung einzubinden, und diese über die umfangreichen Möglichkeiten des mod_ssl Moduls über eine gesichtere Verbindung zur Verfügung zu stellen, führten zur Wahl des Apache Webserver als Webserver für mpuls WASKA.

Durch den modularen Ansatz wird es möglich, einen Teil der Aufgaben durch Apache selbst zu übernehmen. So übernimmt aus Gründen der Performance der Apache das Servieren von statischen Inhalten innerhalb von WASKA.

[1] http://news.netcraft.com/archives/web_server_survey.html
[2] http://code.google.com/p/modwsgi/
[3] http://www.python.org/dev/peps/pep-0333/
[4] http://www.modpython.org/

3.1.1 Konfiguration

WASKA wird innerhalb eines virtuellen Hosts betrieben. Folgende Dateien sind für die Konfiguration des Apache Webserver von Relevanz. Für weitere Informationen zu den Dateien konsultieren Sie bitte die Installationsanleitung bzw. die Dokumentation des Apache Webserver, in der auf die einzelnen Parameter näher eingegangen wird.

- */etc/apache2/httpd.conf:* Konfiguration grundlegender Einstellungen des Apache Server. Damit der Webserver ausschliesslich gesicherte Verbindungen auf Port 443 entgegen nimmt, muss der Port in der Konfigurationsdatei angepasst werden:

```
#Listen 80
Listen 443
```

- */etc/apache2/sites-available/default.conf:* Seitenspzifische Einstellungen zu dem virtuellen Host (hier der default), in dem WASKA betrieben wird. Hier finden sich neben der Einbindung des Pylons Webframework vor allem Anweisungen, die die SSL-Konfiguration betreffen:

```
WSGIScriptAlias / /usr/local/wsgi/scripts/waska.wsgi

SSLEngine On
SSLCipherSuite HIGH:MEDIUM
SSLCertificateFile    /etc/apache2/ssl/server.crt
SSLCertificateKeyFile /etc/apache2/ssl/server.key
SSLCACertificateFile  /etc/apache2/ssl/cacert.pem
SSLVerifyClient       require
SSLVerifyDepth        3
SSLOptions            StdEnvVars

<Location />
  SSLRequire %{SSL_CLIENT_S_DN_OU} eq "WASKA" \
  and %{SSL_CLIENT_VERIFY} eq "SUCCESS"
</Location>
```

3.2 Pylons

Pylons[5] ist ein in Python implementiertes Webframework. Für die Verwendung in mpuls WASKA wurde Pylons aus einer mehreren Gründen gewählt. Pylons ist eins der ersten Frameworks welches den WSGI Standard umsetzt. Es lässt sich in einer Reihe von Umgebungen einsetzten. Pylons lässt sich sowohl in einer Linux als auch Windowsumgebung betreiben. Dabei ist eigenständig lauffähig, oder kann über WSGI, FastCGI, SCGI, CGI, mod_python und weitere in andere Server integriert werden. Dies gibt einen großen Freiheitsgrad in der Wahl der Zielplattform. Pylons verfügt über einen starken modularen Ansatz. Der Großteil der Funktionalität von Pylons basiert auf eigenständigen Komponenten, die bei Bedarf hinzugefügt, entfernt, oder neu geschrieben werden können. Auf diese Weise lässt sich der Funktionsumfang genau auf die Anforderungen abstimmen indem nur die Komponenten eingebunden werden, die nötig sind.

3.2.1 Komponenten von Pylons

Die WSGI Architektur von Pylons ermöglicht den bereits genannten starken modularen Ansatz. Nachstehend finden Sie eine Liste einiger der wichtigsten Komponenten von Pylons.

- **Python Paste** Python Paste[6] ist eine Sammlung von Werzeugen zur Entwicklung von WSGI Webanwendungen. Paste ist als eine der Hauptkomponenten die Grundlage des Pylons Webframework und stellt dem Namen nach eine Art Klebstoff zwischen den anderen Komponenten von Pylons.
- **Mako** Mako[7] ist die Standard Templating Bibliothek, innerhalb von Pylons.

[5] http://www.pylonshq.com
[6] http://pythonpaste.org/
[7] http://www.makotemplates.org/

- **Routes** Routes[8] dient zur Abbildung von URLS auf bestimmte Aktionen innerhalb der Webanwendung. Dabei wird die URL nach einem festgelegten Muster in einzelene Fragmente zerlegt. Abhängig von der Bedeutung dieser Fragmente können diese verschiedenen Funktionsaufrufen und den dazugehörigen Parametern zugeordnet werden.
- **Beaker** Beaker[9] stellt die Sessionverwaltung innerhalb von Pylons zur Verfügung.
- **Formencode** Formencode[10] ist eine hilfreiche Bibliothek zur Verarbeitung von Formulardaten. Formencode bietet zum einen hauptsächlich Möglichkeiten zur Überprüfung von Formulardaten, und Behandlung von Fehlerfällen.

Pylons besteht aus weiteren hier nicht aufgeführten Komponenten. Weitere Information hierzu finden Sie in der Dokumentation. Viele der Komponenten lassen einfach gegen Alternativen auszutauschen. Diese Flexibilität zeichnet Pylons im Gegensatz zu anderen Webframeworks wie Turbogears[11] oder Django[12]

3.2.2 Prinzip der Anwendungslogik

Pylons ist die eigentliche Kernkomponente von mpuls WASKA. Hier ist ein Großteil der Anwendungslogik umgesetzt. Um besser verstehen zu können wie der Informationsfluss innerhalb von mpuls WASKA ist, soll kurz das Prinzip einer Model-View-Controller (MVC) Architektur erklärt werden. Pylons setzt in seiner Architektur auf die Trennung zwischen dem Datenmodel (Model), der Representation der Daten (View), und der webspezifischen Anwendungslogik (Controller). Zum besseren Verständnis soll einmal ein typischer Anfrage innerhalb der Webanwendung verfolgt werden. Bei der Anfrage handelt es sich um das Übersenden von Formulardaten. Die Betrachtung ist hier stark vereinfacht.

Nach dem Absenden der Formulardaten erreicht die Anfrage das Webframework. Hier wird zunächst mit Hilfe von *Routes* die URL zerlegt und der aufzurufende Controller ermittelt. Die Formulardaten werden an den Controller zur weiteren Verarbeitung weitergegeben. Der Controller hat nun die Aufgabe die Anfrage richtig zu bearbeiten und im Anschluss eine Antwort zu generieren, die an den Nutzer gesendet werden kann. Die Bearbeitung der Anfrage beinhaltet in der Regel das Überprüfen von Zugriffsrechten, die Verwaltung von Sessiondaten (Beaker), die Überprüfung von Formulardaten (Formencode), sowie das Laden, Bearbeiten und Anlegen von Objekten aus dem Model, bis am Ende eine Ergebnisseite für den Nutzer generiert wird.

Der Controller wird zunächst mit Hilfe von *Formencode* die übersendeten Formulardaten validieren. Da Formulardaten immer als Zeichenkette übersendet werden, findet bei der Validierung findet zugleich auch eine Konvertierung der Daten in das richtige Format statt. Ist die Validierung erfolgreich werden die Daten an das Model übergeben.

Das Model ist idealerweise unabhängig von der Anwendungslogik umgibt und bietet für diese eine definierte Schnittstelle für den Zugriff auf die Daten. Das Model kümmert sich nun um die weitere Verarbeitung der Daten (z.b. das Speichern der Daten in der Datenbank). Aus Sicht der umgebenden Anwendung ist das Model eine Blackbox.

Nachdem die Daten in dem Model verarbeitet wurden, sorgt der Controller für die Erstellung einer Antwort. Falls Sie z.B eine Suchanfrage über die in mpuls WAKSA gespeicherten Fallakten gestartet haben, stößt der Controller nun die Generierung einer grafische Darstellung des Suchergebnis in Form einer HTML-Seite an. Im MVC-Ansatz ist dies die Aufgabe der View. Die Template Bibliothek *Mako* dient zur Erstellung von HTML Seiten. Die generierte Webseite wird dann als Antwort gesendet.

3.2.3 Organisation der Dateien

Pylons gibt eine bestimmte Verzeichnisstruktur vor, in der sich auch die in [?] erwähnten Komponenten des MVC-Ansatzes wiederfinden. Die Tabelle Örganisation der Dateien"(vgl. Tabelle 1) gibt eine Übersicht der wichtigsten Dateien und Verzeichnisse innerhalb des Pylons Webframework. Weiterführende Informationen zu dem Aufbau und der Funktionsweise entnehmen Sie bitte der Pylons Dokumentation.

3.2.4 Konfiguration

Datenbankverbindung: Für den Betrieb von WASKA muss im Wesentlichen die Verbindung zur Datenbank konfiguriert werden. Dies wird in der Datei production_wsgi.ini vorgenommen:

[8] http://routes.groovie.org/
[9] http://beaker.groovie.org/
[10] http://formencode.org/
[11] http://www.turbogears.org
[12] http://www.djangoproject.com

Verzeichnis/Datei	Inhalt
data/sessions	Enthält die Daten der Cookie-basierten Sessions. Dieses Verzeichnis ist nur dann relevant, wenn nicht memcached verwendet wird. Bei Verwendung von memcached wird die dort eigene Speichermethode verwendet.
data/templates	Enthält bereits gerenderte HTML Dateien, die aus dem Templates erstellt wurden.
development.ini bzw. production_wsgi.ini	Die zentrale Konfigurationsdatei der WASKA Anwendung. Für den produktiven Betrieb ist die Datei production_wsgi.ini zu verwenden.
waskaweb/config/	Weitere Konfigurationsdateien. Hier werden z.B. die Mappings zwischen URLs und Controller-Aktionen konfiguriert.
waskaweb/controllers/	Das Verzeichnis enthält die Controllerklassen der Anwendung. Sämtliche Aktionen des Nutzers rufen zunächst einen Controller auf. Dieser beinhaltet dann einen Teil der Applikationslogik, bzw. delegiert die Anfrage an tiefer gelegene Klassen der Anwendung und gibt letztendlich das Ergebnis der Anfrage an den Nutzer zurück.
waskaweb/i18n/	Ein Großteil der Beschriftungen und Dialoge sind internationalisiert. Innerhalb dieses Verzeichnisses finden sich die nötigen Dateien, um eine Internationalisierung auf andere Sprachen durchzuführen, bzw. Anpassungen an bestehenden Texten vorzunehmen.
waskaweb/lib/	Enthält Bibliotheken, die an verschiedenen Stellen der Anwendung immer wieder gebraucht werden.
waskaweb/model/	Enthält das Modell der Anwendung und definieren einen Großteil der Applikationslogik. Weiter stellt das Model die Verbindung zu der Datenbank her. Die Klassen stellen Schnittstellen zur Verfügung, über die vom Controller aus auf die in der Datenbank gespeicherten Daten zugegriffen werden kann.
waskaweb/public/	Dieses Verzeichnis enthält die statischen Daten der WAKSA-Webanwendung wie CSS-Dateien und Images bzw. Icons.
waskaweb/templates/	Die Mako-Templates, die zur Generierung von Webseiten und Dialogen innerhalb von WASKA genutzt werden, sind hier gespeichert.
waskaweb/formed/	Enthält den FormEd Formular-Baum, der die Grundlage für das Formular in der Webanwendung darstellt. Weiter finden sich hier Dateien die eng mit dem Formular in Verbindung stehen wie z.B die Hilfedateien.

Tabelle 1: Organisation der Dateien

```
db_host = 127.0.0.1 # IP-Adresse des Datenbank-Servers
db_port = 5432
```

Betrieb als Auswerteserver: Um den WASKA Server als Auswerteserver zu betreiben ist lediglich ein Parameter in der Konfigurationsdatei zu ändern.

Nach einem Neustart wird WASKA als Auswerteserver betrieben. Weitere Informationen zum Auswerteserver finden Sie im Abschnitt [?]

```
evaluation_server = 1 # 1 Auswerteserver, 0 normaler Betrieb
```

Sessionverwaltung: Einbinden von memcache: Sessiondaten werden in einer Standardinstallation von Pylons im Hauptspeicher gehalten. Alternativ können die Sessiondaten (wie bei mpuls WASKA) von *memcached* verwaltet werden. Die entsprechende Konfiguration findet sich ebenfalls in der production_wsgi.ini :

```
#beaker.session.type = memory # Sessiondaten werden im Hauptspeicher gehalten
beaker.session.type = ext:memcached # Verwendung von memcache
beaker.session.url = 127.0.0.1 # URL auf der der memcache läuft.
```

Es empfiehlt sich bei einem Neustart des WASKA Servers auch den memcached neu zu starten, um so zu garantieren dass keine alten Sessiondaten mehr vorhanden sind. Dies kann insbesondere bei dem Einspielen von neuen Versionen von WASKA zu Problemen führen. Weitere Informationen zu dem memcached finden Sie im Abschnitt [?].

3.3 Memcache

Memcached ist ein hoch performantes, verteiltes Cachesystem in dem Speicherobjekte für einen schnellen Zugriff vorgehalten werden können. Im Rahmen von WASKA wird memcached dazu verwendet die Sessiondaten zu speichern.

Die Verwendung von memcached ist optional. Alternativ können die Sessiondaten auch durch die Beaker-Komponente des Pylons Webframework im Hauptspeicher verwaltet werden.

3.3.1 Konfiguration

Die Konfiguration des memcached wird in der Datei /etc/memcached.conf vorgenommen. Hier können Sie festlegen wieviel Hauptspeicher memcached maximal für sich beanspruchen darf und von wem der memcached abgefragt werden darf. Für den Produktivbetrieb haben sich dort folgende Werte als sinnvoll herausgestellt:

```
# Log memcached's output to /var/log/memcached
logfile /var/log/memcached

# Start with a cap of 64 megs of memory. It's reasonable, and
# the daemon default Note that the daemon will grow to this
# size, but does not start out holding this much memory
-m 64

# Specify which IP address to listen on. The default is to
# listen on all IP addresses.  This parameter is one of the
# only security measures that memcached has, so make sure it's
# listening on a firewalled interface.
# -l 12.34.56.78
-l 127.0.0.1 # memcache auf dem gleichen Rechner wie die
             # Pylonsanwendung
```

3.4 PDF-Importer

Eine Besonderheit an WASKA ist, das die Fallakten zunächst in drei verschiedenen Versionen als PDF-Dokumente (mit Daten in XFA-Elementen) geführt wurden. Alle drei Versionen können ausgelesen und in die Web-Applikation importiert werden.

Der PDF-Importer ist ein eigenständiger in Java implementierter Server, der der Extraktion der in den PDF-Dateien enthaltenen XFA-Daten dient. Dazu wird die Bibliothek XYZ benutzt.

Hierbei gibt das Pylons-Framework über eine Verbindung die PDF-Fallakte an den PDF-Importer. Dieser extrahiert aus der PDF-Datei die darin enthaltenen XFA-Daten und gibt diese wieder an den Pylons-Framework zurück. Die XFA-Daten können nun durch das Pylons-Framework dazu genutzt werden eine neue Fallakte zu erstellen, sie mit Werten auszufüllen und zu speichern.

3.5 Sicherheit der Webanwendung

Die Sicherheit der Webanwendung wird durch mehrere Maßnahmen sichergestellt.

3.5.1 Verwendung von Zertifikaten

Um mit WASKA kommunizieren zu können, ist es zwinged notwendig im Besitz eines gültigen Zertifikats für WASKA zu sein. Ohne ein gültiges Zertifikat wird eine Anfrage an den WASKA Server mit einer entsprechenden Fehlermeldung quitiert. Unbefugte Personen werden somit schon auf der Ebene des Webserves abgewiesen. Das Zertifikat sichert zudem die Authenzität der beiden Kommunikationspartner. Die Zertifikate werden dabei durch eine zentrale Stelle in Koordination mit der Intevation vergeben.

3.5.2 Verschlüsselte Übertragung

Während der Kommunikation mit dem WASKA Server werden die Daten nur verschlüsselt überragen und so verhindert, das Passworte z.B mitgelesen werden können.

3.5.3 Sicherung der Sessiondaten

Um sich nicht bei jeder Anfrage an den WASKA Server neu anmelden zu müssen, wird die Information über eine erfolgreich durchgeführte Anmeldung innerhalb einer Session gespeichert. Ein mögliches Angriffsszenario ist die Übernahme dieser Session durch eine unbefugte Person.

4 Datenbanksystem

[?] Hier wird PostgreSQL 8.2 eingesetzt, welches von Hause aus eine hohe Sicherheit, Stabilität und Performanz mitbringt. Durch die Verwendung von Datenbank-Rollen kann der einschränkende Zugriff auf die verwalteten Daten effizient und sicher gewährleistet werden. Der Datenaustausch mit dem DBMS kann SSL-verschlüsselt durchgeführt werden.

ANMERKUNG
Die Benennung der Felder in der Datenbank wurden automatisch generiert. Die Namen wurden dabei zu einem Großteil aus der PDF Versionen der Fallakten übernommen.

4.1 Datenbankschema

Abbildung Abbildung 1 zeigt eine vereinfachte Übersicht des Datenbankschemas. Das Schema zeigt die wichtigsten Tabellen und die referenziellen Beziehungen zwischen diesen Tabellen.

Nicht zu sehen sind Views, Funktionen oder Referenztabellen der Datenbank. Diese dienen entweder zur Etablierung des Rechtesystems oder kapseln komplexere Abfragen, und stellen somit Hilfsviews bzw. Hilfsfunktionen dar. Für das grundsätzliche

Verständnis der Organisation der Datenbank ist eine genaue Vorstellung der Funktionen und Views allerdings nicht nötig. Diese finden sich in [?].

Von zentraler Bedeutung sind die Tabellen *master_tbl* sowie alle *rg_$name_tbl* Tabellen. Zusammen speichern diese Tabellen die Daten der Fallakte. Die rg* Tabellen speichern die Daten sogenannter *Repeat-Groups* und referenzieren die master_tbl über den Fremdschlüssel master_id. Ein Großteil der Daten in diesen Tabellen werden als numerische Werte gespeichert. Insbesondere in der master_tbl kommt dies häufig vor. Die numerischen Werte innerhalb der master_tbl referenzieren die tatsächlichen Werte innerhalb der Referenztabellen.

Bei der Verwendung von solchen numerischen Werten innerhalb von Auswahlfeldern hat sich datenbankseitig folgende Konvention etabliert:

Typ	**Wert**
Ja	1
Nein	0
Keine Angabe	-1
Sonstiges	-2

Tabelle 2: Konvention zur Vergabe von Indexwerten in Auswahlfeldern

Repeatgroups dienen dazu bestimmte Daten mehrfach zu einer Fallakte zu speichern. So gibt es z.B. Repeatgroups für die verschiedenen Unterstützungsangebote, von denen es pro Fallakte mehrere geben kann. Weiter Informationen zu den Repeatgroups finden Sie in Abschnitt [?].

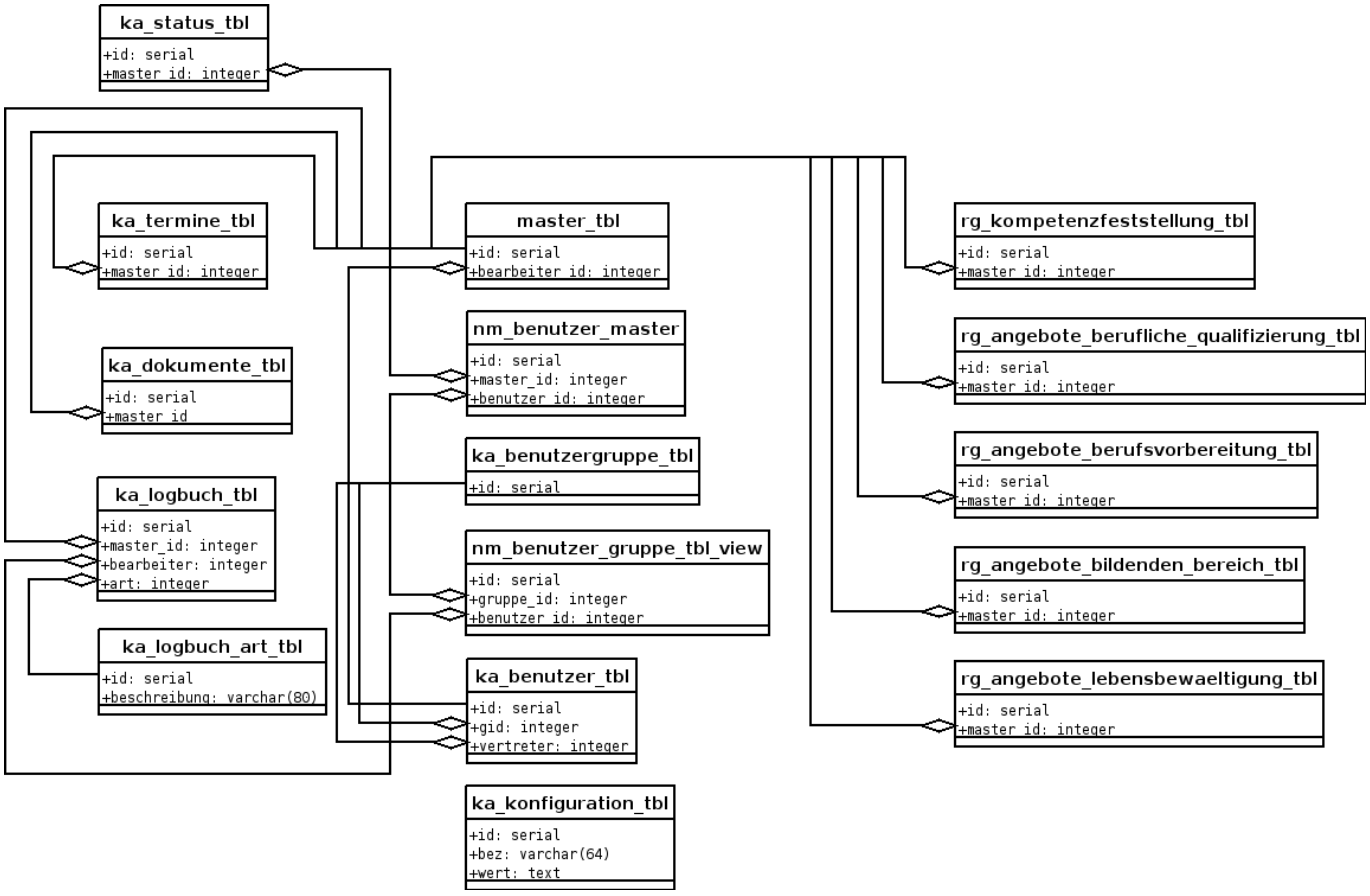

Abbildung 1: Schema der Datenbank

4.2 Kurzbeschreibung der Tabellen

Es werden nun in kurzer Form die Bedeutung der in der Abbildung dargestellten Tabellen erläutert.

4.2.1 Tabellen zur Fallakte

master_tbl: In der Tabelle master_tbl werden die Hauptdaten einer Fallakte gespeichert. Innerhalb der Tabelle wird neben den eigentlich Daten der Fallakten auch festgehalten welcher Mitarbeiter der Hauptbearbeiter der Fallakte ist. Die Struktur dieser Tabelle und der dazu gehörigen Repeatgroups basieren auf dem FormEd Formularbaum, der die Grundlage für das verwaltete Formular in WASKA bildet. Weitere Informationen zu dem FormEd finden Sie in [?]

ka_status_tbl: In dieser Tabelle werden Daten über den SZustand"der Fallakte gespeichert. Zur Zeit ist dies ein Zeitstempel der letzen Veränderung an der Fallakte, sowie die Information in welchem Zustand sich die Fallakte aus verwaltungstechnischer Sicht befindet (bearbeitbar, schwebend gelöscht, schwebend anonymisiert, usw.). Weitere Informationen zu den Zuständen der Fallakte finden sich in [?]

Es folgenden nun die verschiedenen Repeatgroups

rg_kompetenzfestellung_tbl: Tabelle zur Speicherung der Kompetenzfestellungen eines Jugendlichen

rg_angebote_berufliche_qualifizierung_tbl: Tabelle zur Speicherung der Unterstützungsangebote im Bereich der beruflichen Qualifizierung

rg_angebote_berufvorbereitung_tbl: Tabelle zur Speicherung der Unterstützungsangebote im Bereich der Berufsvorbereitung

rg_angebote_bildenden_bereich_tbl: Tabelle zur Speicherung der Unterstützungsangebote im bildenden Bereich

rg_angebote_lebensbewaeltigung_tbl: Tabelle zur Speicherung der Unterstützungsangebote im Bereich der Lebensbewältigung

4.2.2 Benutzerverwaltung

ka_benutzer_tbl: Diese Tabelle enthält die Benutzer des Systems. Neben den Namen und Loginnamen werden auch weitere Informationen wie die Raumnummer, Telefon oder die Standardvertretung (Benutzergruppe) gespeichert.

nm_benutzer_master_tbl: Diese Transitionstabelle dient zur Etablierung der Vertretungsregelung und bildet eine N:M Beziehung zwischen Fallakten und Benutzergruppen.

ka_benutzergruppe_tbl: Die verschiedenen Benutzergruppen im System werden in dieser Tabelle gespeichert.

nm_benutzer_gruppe_tbl: Die Zugehörigkeit der einzelnen Benutzer in den jeweiligen Gruppen wird über diese Transitionstabelle festgelegt. Nutzer und Gruppen stehen in einem N:M Verhältnis zueinander.

4.2.3 Sonstige Funktionen

ka_konfiguration_tbl: Agenturweite Einstellungen werden in dieser Tabelle gespeichert. Die Werte sind als Schlüssel-Wert Paare gespeichert und lassen sich auf diese Weise leicht erweitern. Zu den gespeicherten Werten gehört die maximal zulässige Speicherdauer, die verschiedenen Einverständniserklärungen oder das Förderkennzeichen der Kompetenzagentur.

ka_termine_tbl: Diese Tabelle ist wesentlicher Bestandteil der Terminverwaltung innerhalb von WASKA und speichert sowohl die agenturweiten als auch die fallspezifischen Termine. Die Unterscheidung, ob einer Termin agenturweit oder fallspezifisch ist, wird durch die Existenz eines Fremdschlüsseleintrags auf die master_tbl determiniert. Weitere Informationen zu der Terminfunktion finden Sie unter [?]

ka_dokumente_tbl: Diese Tabelle ist wesentlicher Bestandteil der Dokumentenverwaltung innerhalb von WASKA und speichert sowohl die agenturweiten als auch die fallspezifischen Dokumente. Die Unterscheidung, ob einer Dokument agenturweit oder fallspezifisch ist, wird durch die Existenz eines Fremdschlüsseleintrags auf die master_tbl determiniert.

ka_logbuch_tbl: Die Einträge des Falltagebuchs werden in dieser Tabelle gespeichert. Die jeweilige Fallakte wird über den Fremdschlüssel master_id referenziert. Jeder Logbucheintrag ist einer bestimmten Kategorie zugeordnet, die in der Tabelle ka_logbuch_art_tbl festgelegt ist.

ka_logbuch_art_tbl: Die verschiedenen Kategorien eines Logbucheintrags werden in dieser Tabelle gespeichert.

4.3 Rechtesystem

Das Rechtesystem der Datenbank gewährleistet als letzte Instanz das der Zugriff auf die Daten nur für befugte Personen möglich ist. Dies wird im wesentlichen dadurch erreicht, dass es für jeden Nutzer der WASKA Anwendung auch einen eigenen PostgreSQL-Nutzer gibt.

Neben den verschiedenen Benutzern gibt es den Ansatz lesende und schreibende Operationen nicht direkt auf den Tabellen zuzulassen, sondern hierfür Views, bzw. Funktionen zu nutzen. Die jeweiligen Views und Funktionen lassen sich dann über das Rollensystem der PostgreSQL Datenbank festgelegte Privilegien nur von befugten Benutzern aufrufen, bzw. abfragen.

4.3.1 Benutzer und Gruppenrollen

Wie bereits erwähnt wird jeder Benutzer in WASKA auf einen PostgreSQL-Nutzer abgebildet. Jeder dieser Nutzer erbt die Befugnisse einer der drei verfügbaren Gruppenrollen ein und erhält auf diese Weise kontrollierten Zugriff auf die Funktionen und Views der Datenbank. WASKA unterscheidet zwischen folgenden drei Rollen:

- **ka_$dname_admin_ka** Die Rollen für Administratoren. Administratoren haben nur eingeschränkten Zugriff auf die Daten der Fallakten und übernehmen hauptsächlich Aufgaben der Benutzerverwaltung.
- **ka_$dname_cm_ka** Die Rolle für Case-Manager. Die wesentlichen Aufgabe eines Case-Managers ist das Anlegen und Bearbeiten von Fallakten.
- **ka_$dname_pb_ka** Diese Rolle wird für Nutzer des Auswerteservers genutzt. Um Auswertungen durchführen zu können, haben diese Nutzer lesenden Zugriff auf alle Fallakten, die im System gespeichert sind.

Da Nutzer und Rollen auf dem Datenbankserver clusterweit gespeichert werden, bedarf es einem einheitlichen Bennenungsschema, um die Nutzer der verschiedenen Datenbanken unterscheidbar zu halten. So werden Nutzer innerhalb von WASKA nach folgendem Schema benannt: *ka_$dbname_$login*. Wobei $dbname der Name der Datenbank ist (zb. 01jk0815) und $login der vom Nutzer gewählten Anmeldekennung entspricht.

4.3.2 Views und Funktionen

Schon Eingangs wurde erwähnt, dass kein direkter Zugriff auf die Tabellen der Datenbank möglich ist, sondern für das Anlegen, Löschen oder Bearbeiten Views und Funktionen erstellt wurden. Über das Zuweisen von Zugriffs bzw. Ausführrechten an eine der oben genannten Gruppenrollen lässt sich ein fein granulares Rechtesystem erstellen.

Der größte Anteil der Funktionen dienen dem Anlegen und Löschen von Datensätzen. Der Grund hierfür liegt in einer elementaren Designentscheidung, die der Sicherheit der Anwendung dient:

- Das Anlegen und Löschen von Daten in der Datenbank ist nur über entsprechende Funktionen möglich. Für keine der Views im System existieren Regel für das Anlegen oder Entfernen von Datensätzen.
- Das Bearbeiten von Datensätzen wird über die Update-Regeln der entsprechenden View definiert.

Der Einsatz von definierten Funktionen für den Zugriff auf die Daten sorgt für zusätzliche Sicherheit und wird auch vom IT-Grundschutz empfohlen[13] Sie sorgen dafür, dass der Benutzer nur in einer vorgegebenen Form Daten manipulieren kann. Die benötigten Parameter der Funktion sind vorgegeben und können überprüft werden. Innerhalb der Funktionen wird direkt auf den Tabellen der Datenbank gearbeitet. Funktion bieten die Möglichkeit komplexe Aufgaben, wie das Anlegen eines neuen Benutzers für WASKA, in einer Funktion zu kapseln und diese über eine definierte Schnittstelle bestimmten Benutzern zur Verfügung zu stellen. Damit die Funktionen direkt auf den Tabellen der Datenbank arbeiten können, laufen viele der Funktionen im Kontext des Datenbankbeseitzer (SECURITY DEFINER), und damit mit den höchst möglichen Privilegien.

Durch Beschränkung des Zugriffs auf die Daten über Views, lässt auf Ebene der Tabellenspalten festlegen, auf welche Daten der Benutzer Zugriff haben soll. Dies wäre bei direktem Zugriff auf die Tabellen nicht möglich. Desweiteren bieten Views die Möglichkeit, komplexe Abfragen unter einer einer einheitlichen Schnittstelle zur Verfügung zu stellen und den Zugriff auf diese nur für bestimmte Nutzer zu erlauben.

[13] Siehe hierzu Punkt M2.129 aus dem Leitfaden des IT Grundschutz des Bundesamt für Sicherheit (BSI). http://http://www.bsi.bund.de/gshb/deutsch/m/m02129.htm.

5 Konzepte in der Webanwendung

5.1 Benutzer und Benutzergruppen

Neben den bereits in [?] erwähnten Datenbanknutzern existieren als direktes Gegenstück die applikationsseitigen Benutzer und Benutzergruppen. Diese Benutzer und Gruppen werden primär dafür verwendet Zuordnungen zwischen Benutzern und Fallakten herzustellen, und bilden so die Grundlage für die in [?] beschriebenen Zugriffsregelungen.

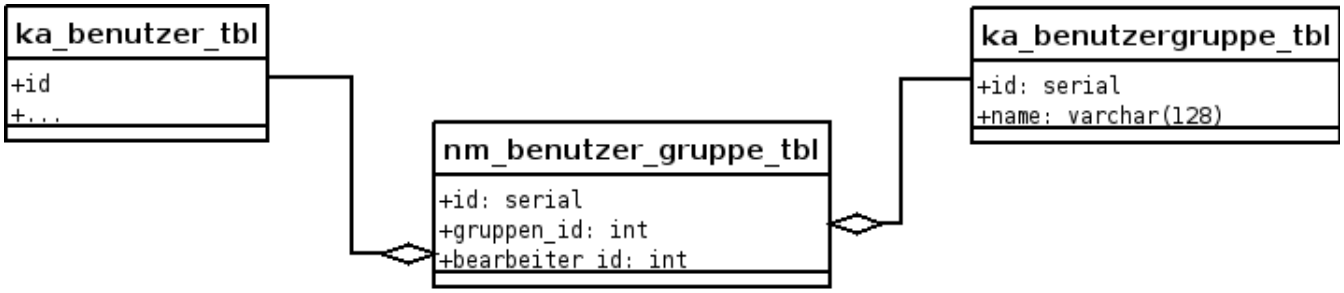

Abbildung 2: Benutzer und Benutzergruppen

Abbildung 2 zeigt wie Benutzer und Benutzergruppen organisiert sind. Benutzer und Benutzergruppen stehen in einem NM-Verhältnis zueinander, welches über die Tabelle *nm_benutzer_gruppen* hergestellt wird.

5.1.1 Benutzer

Jeder Nutzer hat ein entsprechendes Benutzerkonto innerhalb von WASKA. Dieses Benutzerkonto dient zum einen für eine Zuordnung von Personen zu Fallakten, sowie, über die zu einem Nutzer gespeicherten Information, als Vorlage für bestimmte Felder in der Fallakte. So wird bei dem Anlegen einer neuen Fallakte automatisch der Name des Benutzers als zuständige Person in die Fallakte eingetragen. Neben den persönlichen Informationen des Nutzer enthält das Benutzerkonto auch Angaben über die gewählte Standardvertretung, oder die Einstellung ob sich ein Benutzer an der Anmeldung anmelden darf oder nicht (Benutzer ist aktiviert).

5.1.2 Benutzergruppen

Benutzergruppen sind die wesentliche Grundlage zur Realisation der Vertretungsregelungen. WASKA kennt zwei Arten von Benutzergruppen:

- **Automatisch erstelle Gruppen** Bei dem Anlegen eines neuen Benutzers wird für diesen automatisch eine eigene Benutzergruppe angelegt. Diese Gruppen haben zwei Besonderheiten: Sie enthalten nur den neu angelegten Nutzer als Mitglied, und weiter sind sie zu durch keinen Benutzer nachträglich zu bearbeiten. Es können somit keine weiteren Nutzer hinzugefügt werden.

  Die Notwendigkeit für diese besondere Form der Benutzergruppen ergibt sich aus der Art wie Vertretungsregelungen umgesetzt sind. Diese werden ausschließlich über Benutzergruppen realisiert. Um Zugriffsrechte auch auf Ebene einzelner Benutzer zuweisen zu können, wurde diese Form der Benutzergruppen eingeführt.

  Automatisch angelegte Benutzergruppen werden nach dem Schema *$Nachname, $Vorname ($Anmeldekennung)* angelegt und stehen unter dieser Bezeichnung zur Auswahl zur Verfügung.

- **Durch Benutzer erstelle Gruppen** Die Administration hat die Möglichkeit im Rahmen der Benutzerverwaltung neue Benutzergruppen anzulegen, und diesen Benutzern zuzuordnen. Der Name der Benutzergruppe kann dabei frei gewählt werden, darf aber nicht mit bereits existierenden Gruppennamen kollidieren.

Benutzergruppen haben als einziges Attribut den Namen der Benutzergruppe.

5.2 Löschen und Anonymisieren von Akten

Das Löschen und Anonymisieren von Fallakten wurde in einem zweistufigen Modell realisiert. Hierbei müssen die Case-Manager zunächst eine Fallakte zur Löschung bzw. zur Anonymisierung freigeben. Erst dann kann die Administration eine Fallakte endgültig Löschen. Grundlage für die Realisation dieses Workflows sind die verschiedenen Zustände einer Fallakte. Siehe hierzu [?].

Bei der Freigabe einer Fallakte zum Löschen wechselt diese in den Zustand *Schwebend anonymisiert*. Analog verhält es sich bei der Freigabe zu Anonymisierung. Schwebend anonymisierte Fallakten werden für den Case-Manager aus der Übersicht ausgeblendet und stehen für eine Bearbeitung nicht mehr zur Verfügung. Die Administration hat nun die Möglichkeit die Fallakte endgültig zu Löschen, oder Wiederherzustellen. Bei der Wiederherstellung wird die Fallakte wieder in den Zustand *Offen* versetzt und steht danach wieder dem Case-Manager für weitere Bearbeitungen zur Verfügung.

Das zweistufige Modell zum Löschen und Anonymisieren wurde vor allen zum Schutz vor Datenverlust implementiert. Fallakten werden zum Teil über Monate hinweg gepflegt und daher sollte die Gefahr versehentlich Daten zu löschen minimiert werden.

5.3 Fallaktenübersicht / Suche

Die Fallaktenübersicht ist eine der zentralen Stellen innerhalb der Webanwendung. Sie gibt sowohl der Administration als auch dem Case-Managern eine Übersicht über die im System gespeicherten Fälle, und liefert schon auf den ersten Blick einige Informationen. Mit der Suche über die Fallakten kommt ein weiterer zentraler Baustein in WASKA hinzu, der es dem Benutzer ermöglicht die Menge der Fälle unter der Angabe von bestimmten Suchkriterien einzuschränken.

Die Fallaktenübersicht lässt den Benutzer bestimmte Aktionen für eine Auswahl von Fallakten durchführen. Handelt es sich um einen Auswerteserver, so lassen sich die Aktionen für die ausgewählten Agenturen durchführen. Um solche Operationen durchführen zu können, wurde über die Metapher eines Aktenbündels das Konzept der *CaseBundle* bzw. der *AgencyBundle* eingeführt. CaseBundles bzw. AgencyBundles werden durch die Auswahl in der Fallaktenübersicht zusammengestellt. Diese Konzept soll nun kurz erläutert werden.

5.3.1 Fallaktenbündel (CaseBundle)

CaseBundles fassen mehrere Fallakten zu einem Objekt zusammen. Dies kann dann genutzt werden, um Aktionen über die Menge der in ihnen enthaltenen Fallakten durchzuführen. Hierfür bieten CaseBundles eine Schnittstelle die genutzt werden kann. CaseBundles enthalten keine vollständigen Fallakten sondern nur deren IDs.

5.3.2 Agenturbündel (AgencyBundle)

Analog zu den CaseBundles existieren auf dem Auswerteserver sogenannte AgencyBundles. Um nun Fallakten agenturübergreifend auszuwählen, enthalten AgencyBundles ihrerseits CaseBundles. Diese CaseBundles enthalten wiederum sämtliche Fallakten einer Agentur. Ebenso wie die CaseBundles bieten auch die AgencyBundles eine Schnittstelle für typische Operationen wie Löschen oder Auswerten.

5.3.3 Suche

Die Fallaktensuche bietet dem Benutzer die Möglichkeit die Menge der angezeigten Fallakten nach bestimmten Kriterien einzuschränken. Auf diese Weise lassen sich bequem die oben genannten gebündelten Aktionen auf eine Submenge von Fallakten anwenden.

Die Eingabe von Suchkriterien lässt sich über zwei Wege erreichen. Zum einen bietet die Weboberfläche entsprechende Formularelemente, die eine bequeme Eingabe erlauben. Zum anderen lassen sich die Kriterien auch direkt in die Suchmaske eingeben. Der Suchstring enthält dann kodiert die verschiedenen Kriterien. Ein typischer Suchstring hat folgende Syntax:

```
suchbegriff;option1:wert1;option2:wert2;...;optionN-1:wertN-1;optionN:wertN
```

Bei der Eingabe über die Weboberfläche werden die übersendeten Parameter zunächst geparst und ein Suchstring zusammen gesetzt. Mit Hilfe eines solchen Suchstring ist es möglich, die Suche mit vordefinierten Optionen zu starten, ohne das der Benutzer

eine Eingabe tätigen muss. Die Suche gibt wider Erwarten kein CaseBundle zurück, sondern aus Gründen der Performance eine Liste von abgespeckten Fallakten. Diese Fallakten enthalten einige Informationen, die nötig sind um die Übersicht der Fallakte rendern zu können.

```
# Holen des aktuell angemeldeten Nutzer
user = session.get('USER_AUTHORIZED')
# Erstellen der Fallaktenübersicht
overview = CaseOverview()
# Suche eigene offene Fälle
case_ids = overview.search('state:1;state:2;own:%s' % user.id)
```

Derzeit gibt es keine Möglichkeit individuelle logische Verknüpfungen der Optionen zu setzen. Diese sind momentan noch fest durch die Implementation vorgegeben. In den meisten Fällen handelt es sich um UND-Verknüpfungen. Ausnahmen hiervon gibt es für einzelne Optionen. Im folgenden sollen die verschiedenen möglichen Suchoptionen beschrieben werden.

- **Name, Vorname, Kundennummer** Über das Textfeld in der Suche können Fallakten nach den genannten Suchbegriff gesucht werden. Mehrer Suchbegriffe werden durch Leerzeichen voneinander getrennt. Die einzelnen Begriff sind mit einem logischen ODER verknüpft.
- **Fallakten-Status** Eine Suche nach dem Status der Fallakte ist über die Option *state* möglich. Die entsprechenden Werte entnehmen Sie bitte [?]. Bei Angabe von mehreren Statis werden diese miteinander ODER verknüpft.
- **Case-Management-Status** Um die Suche nach dem Status des Case-Managements einzuschränken, verwenden Sie die Option *cm_state*. Mögliche Werte sind *before* für Beratungskunden, *active* für aktuell laufendes Case-Management und *finished* für ein abgeschlossenes Case-Management. Eine Angabe von mehreren Statis ist nicht möglich. Für die Suche zählt der zuletzt angegebene Status.
- **Eigene Fallakten / Vertretung anzeigen** Über die Option *own* und *standin* lässt sich angeben ob eigene bzw. Vertretungsfälle angezeigt werden sollen. Der jeweilige Wert der beiden Optionen ist die derzeitige Benutzer ID. Die Optionen sind miteinander ODER verknüpft.
- **Fallakten eines bestimmten Editors anzeigen** Diese Option steht nur für die Administration zur Verfügung. Über die Option *editor*, und als Wert die Benutzer ID des gewünschten Editors, lassen sich die Fallakten des gewählten Benutzers anzeigen. Es ist nicht möglich mehrere Optionen anzugeben. Für die Suche ist der letzte angegebene Editor relevant.
- **Geschlecht** Eine Suche nach dem Geschlecht des Jugendlichen ist über die Option *gender* möglich. Die entsprechenden Werte sind: Keine Angabe (-1), Männlich (1), Weiblich (0), Intersexuell (2). Es ist nicht möglich mehrere Angaben miteinander zu verknüpfen. Für die Suche zählt die letzte angegebene Option.
- **Zeitraum** In der Suche lässt sich der Zeitraum angeben, mit dem sich die Lebensdauer der Fallakte schneiden muss, um angezeigt zu werden. Die Lebensdauer definiert sich ohne weitere Vorgaben über das Datum des Erstgespräch bis zur Beendigung des Case-Management. Andere Kriterien lassen sich aber vorgeben. Zur Definition des Zeitraums können in WASKA vier verschiedene Optionen verwendet werden:

  Datumsangaben werden grundsätzlich in folgender Form angegeben: *JJJJ-MM-TT*, wobei J für Jahr, M für Monat und T für den Tag steht.

  – interval_start_date: Das Startdatum
  – interval_start_field: Definition des Feldes in der Datenbank, welches zur Überprüfung des Startdatum genutzt werden soll. Der Vorgabewert ist das Feld zu Speicherung der Erstgesprächsdatum.
  – interval_end_date: Das Enddatum
  – interval_end_field: Definition des Feldes in der Datenbank, welches zur Überprüfung des Enddatum genutzt werden soll. Der Vorgabewert ist das Feld zu Speicherung der Beendigung des Case-Management.

  Wird kein Zeitraum angegeben, findet die Suche ohne jegliche Einschränkung über alle Fallakten statt.
- **Filiale** Die Option *branch* schränkt die Suche auf Fallakten einer bestimmten Zweigstelle der Kompetenzagentur ein. Der eingetragene Wert wird mit dem Wert für die Filiale, der beim Anlegen eines neuen Benutzers angegeben wurde, verglichen. Eine Angabe von mehreren Filialen ist nicht möglich. Es zählt die letzte angegebene Filiale für die Suche.

- **Förderkennzeichen** Die Option *fkz* schränkt die Suche auf Fallakten mit einem bestimmten Förderkennzeichen ein. Eine Angebe von mehreren Förderkennzeichen ist nicht möglich. Es zählt das letzte angegebene Förderkennzeichen für die Suche.
- **Inkonsistente Fallakten** Die Option *bad* schränkt die Suche auf Fallakten mit inkonsistenten Angaben ein. Die Option *bad* kann dabei verschiedene Werte annehmen, von denen jeder einen bestimmten Fall einer Inkonsistenz bezeichnet. Die verschiedenen Werte werden nun beschrieben:
  - Fehlende Angabe zum Datum des Erstgespräch: Wert 1
  - Datum des Erstgespräch liegt in der Zukunft: Wert 2
  - Beendigungsdatum des CM liegt vor dem Datum des Erstgespräch: Wert 3
  - Beendigungsdatum des CM liegt in der Zukunft: Wert 4
  - CM ist beendet, aber keine Beendigungsdatum angegeben: Wert 5
  - CM ist nicht beendet, aber Beendigungsdatum angegeben: Wert 6

  Mehrfachangaben der Optionen werden in der Suche ODER verknüpft.

Neben den oben genannten Suchoptionen gibt es noch zwei weitere Optionen, die nicht dazu dienen die Suche auf bestimmte Fallakten einzuschränken, sondern die Sortierung des Suchergebnis betreffen.

- sort_field: Der Wert dieser Option beschreibt das Datenbankfeld (tatsächlicher Name in der Datenbank), welches für eine Sortierung verwendet wird. Der Vorgabewert ist der Nachname des CM-Kunden.
- sort_order: Der Wert dieser Option beschreibt die Sortierreihenfolge. Die möglichen Werte sind *desc* für absteigend und *asc* für aufsteigend. Der Vorgabewert ist absteigend.

5.4 Zustände einer Fallakte

Fallakten können innerhalb von WASKA verschiedenen Zustände annehmen. Diese Zuständen sind für die Verwaltung der Fallakten von Bedeutung und dienen der Etablierung eines bestimmten Workflows. Waska unterscheidet in der Applikation zwischen fünf verschiedenen Zuständen, wie sie in Tabelle 3 dargestellt sind. Wichtig ist hier zu erwähnen, dass die Zustände *Offen* und

Typ	**Wert**
Offen	1
Geschlossen (aktuell ungenutzt)	2
Schwebend gelöscht	3
Schwebend anonymisiert	4
Anonymisiert	5

Tabelle 3: Zustände und zugehörige Indexwerte

Geschlossen nicht im Sinne des Case-Management zu verstehen sind, sondern ein rein verwaltungstechnischer Zustand ist. Offen und Geschlossen sind hier eher als Metapher für den aktuellen Bearbeitungszustand einer Fallakte zu sehen.

ANMERKUNG
In der aktuellen Version 1.0.3 von WASKA wird der Zustand *Geschlossen* nicht genutzt bzw. ist gleichbedeutend mit den Zustand *Offen*

Zustände werden innerhalb von WASKA dazu genutzt abhängig vom Zustand der Fallakte nur ausgewählte Aktionen für die Fallakte zuzulassen. So ist es z.B. nur möglich eine Fallakte zu Löschen, wenn diese zuvor in dem Zustand SSchwebend gelöscht"gewesen ist. Weitere Informationen hierzu finden sich in [?].

Die Fallakten können nicht beliebig zwischen den Zuständen wechseln. Abbildung 3 zeigt die möglichen Übergänge zwischen den verfügbaren Zuständen

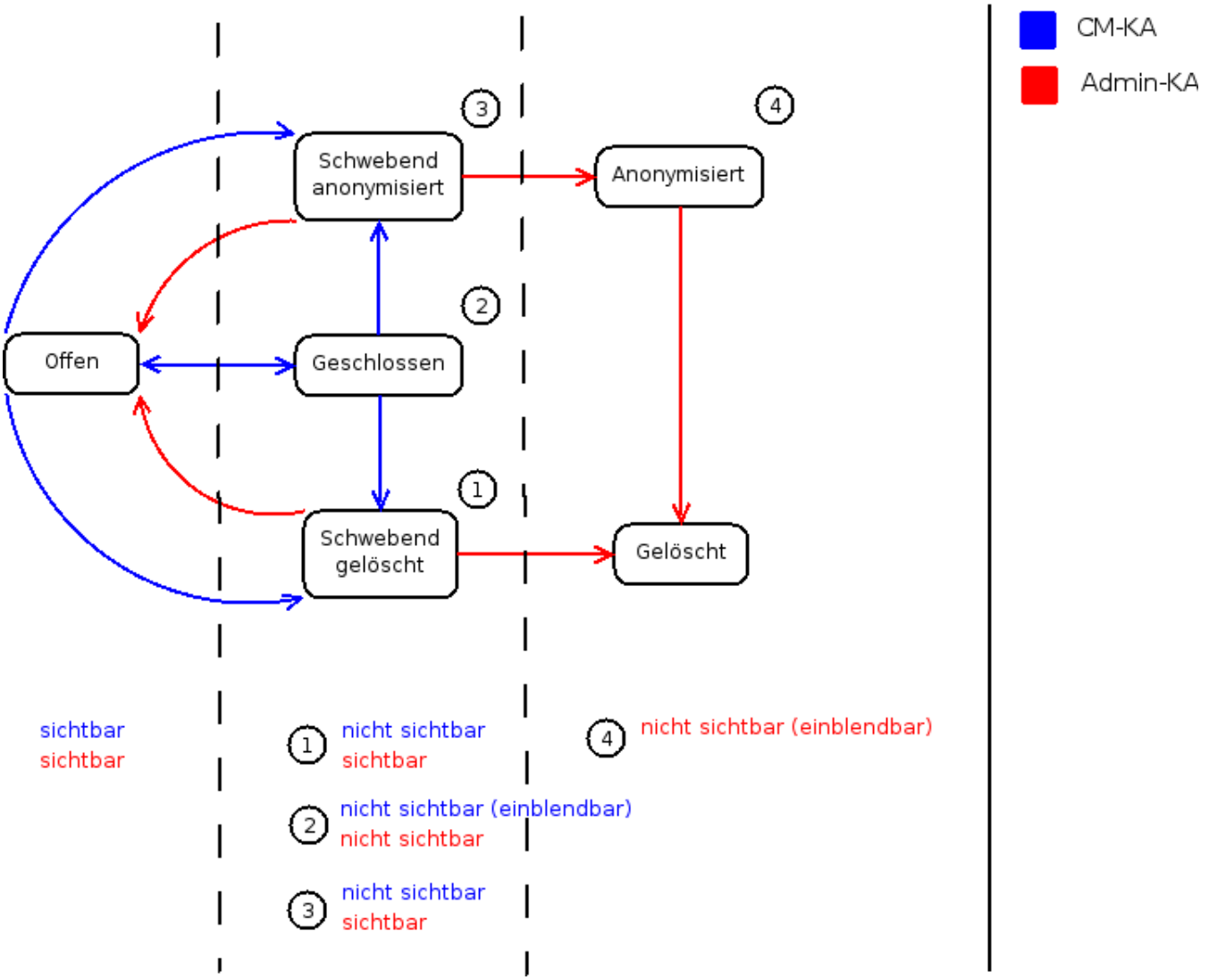

Abbildung 3: Übergänge zwischen den Zuständen der Fallakte

In Pfeile zeigen die möglichen Wechsel der Zustände. Die Farbe der Pfeile gibt an ob der Wechsel durch die Administration oder dem Case-Manager durchgeführt werden kann. Zusätzlich wird noch die Sichtbarkeit der Fallakte in der Fallaktenübersicht angegeben.

5.5 Zugriffsregelungen

Bei dem Zugriff auf die Fallakte wird zwischen den verschiedenen Rollen unterschieden. Administratoren können grundsätzlich auf alle Fallakten eine Kompetenzagentur lesend zugreifen. Eine Bearbeitung der Fallakte ist aber in keinem Fall möglich. Das Löschen der Fallakte vgl. [?] stellt hier eine Ausnahme da. Case-Manager haben zunächst nur Zugriff auf ihre eigenen Fallakten. Zugriff auf weitere Fallakten kann über die Einrichtung von Vertretungsregelungen realisiert werden.

5.5.1 Hauptbearbeiter

Jede Fallakte hat einen Case-Manager als Hauptbearbeiter. Dies ist der Besitzer der Fallakte. In der Regel ist das der Benutzer, der die Fallakte initial angelegt hat. In der *master_tbl* ist dieser Nutzer in dem Feld *bearbeiter_id* angegeben, welches auf den entsprechenden Nutzer in der Tabelle *ka_benutzer_tbl* verweist. Siehe Abbildung 1. Ohne weitere Einstellungen hat nur dieser Hauptbearbeiter Zugriff auf die Fallakte. Alle anderen Case-Manager können diese Fallakte weder bearbeiten noch in ihrer Fallaktenübersicht sehen.

Der Hauptbearbeiter hat im Vergleich zu Vertretern besondere Rechte bei dem Zugriff auf die Fallakte. So ist er neben der Administration die einzige Person, die die Vertretungsregelung ändern kann.

5.5.2 Vertreter

Um auch anderen Benutzern Zugriff auf die Fallakten zu ermöglichen, können der Hauptbearbeiter oder die Administration für die jeweilige Fallakte Vertretungen definieren. Vertretungsregelungen werden über Benutzergruppen definiert. Siehe hierzu [?].

Jeder Fallakte können eine beliebige Anzahl von Benutzern, bzw. Benutzergruppen zugewiesen werden. Die Benutzer bzw. die Mitglieder der Benutzergruppe haben dann ebenfalls schreibenden Zugriff auf die Fallakte. Die Abbildung 4 zeigt die Strutur, die der Vertretungsregelung zu Grunde liegt.

Abbildung 4: Vertretungregelung

Die Tabelle *nm_master_benuter_tbl* stellt eine 1:N Beziehung zwischen einer Fallakte und den Benutzergruppen her. Die Mitgliedschaft der einzelnen Benutzer in den Gruppen wird über die Tabelle *nm_benutzer_gruppe_tbl* hergestellt.

Zur Umsetzung, dass tatsächlich nur befugte Nutzer Zugriff auf die Fallakten erhalten, werden datenbankseitig Views erstellt, die ein entsprechendes Regelwerk beinhalten:

```
CREATE OR REPLACE VIEW master_tbl_view
AS SELECT master_tbl.* FROM master_tbl
WHERE
  master_tbl.bearbeiter_id = getuserid()
  OR master_tbl.id IN (
    SELECT nm.master_id from nm_benutzer_master_tbl nm
    WHERE nm.benutzer_id IN (
      SELECT nm2.gruppe_id from nm_benutzer_gruppe_tbl nm2
      WHERE nm2.benutzer_id = getuserid()
    )
  )
  OR pg_has_role('admin_ka','MEMBER');
```

Die Fallakte soll in drei Fällen sichtbar werden. Dies wird durch die Oder-Verknüpfung der Bedingungen innerhalb der WHERE-Klausel erreicht. Die erste Bedingung sichert, dass der Hauptbearbeiter Zugriff auf die Fallakte erhält. Die dritte Bedingung legt fest, dass Nutzer in der Rolle des Administrators ebenfalls Zugriff bekommen. Die zweite Bedingung betrifft die eigentliche

Vertretungsregelung. Diese besteht aus zwei ineinander geschachtelten Abfragen. Die innere liefert die Menge der Benutzergruppen, in der der aktuelle Nutzer Mitglied ist. Die äussere Abfrage gibt die Menge der Fallakten zurück, auf die die zuvor ermittelten Gruppen Zugriff haben. Ist die aktuelle Fallakte in dieser Menge enthalten, so erhält der Benutzer lesenden Zugriff auf die Fallakte.

Die hier erläuterte Abfrage ist von zentraler Bedeutung und wird in dieser Form auch an anderen Stellen wiederverwendet, um so z.B. auch den Schreibzugriff, der über die entsprechenden Regeln zu der View definiert ist, zu reglementieren.

6 Auswerteserver

Der Auswerteserver dient im Rahmen eines zentralen Monitorings dazu Auswertungen über alle Fallakten der verschiedenen Kompetentagenturen durchführen zu können. Hierzu übertragen die Agenturen die Fallakten in anonymisierter Form in regelmäßigen Abständen zu dem Auswerteserver. Die Codebasis des Auswerteservers ist identisch zu der eines normalen WASKA Servers, jedoch unterscheidet sich der Server in der zur Verfügung stehenden Funktionalität. Mit Hilfe einer Konfigurationsvariablen kann der Modus, in dem der Server betrieben werden soll, festgelegt werden. Nähere Informationen zur Konfiguration des Servers als Auswerteserver entnehmen Sie bitte der Installationsanleitung.

6.1 Anonymisierung der Daten

Aus Gründen des Datenschutzes dürfen die Fallakten nur in anonymisierter Form auf den Auswerteserver übertragen werden. Die Anonymisierung findet während des Auslesens der Fallakten auf dem Server der Kompetenzagenturen statt. Das Anonymisieren wird dabei datenbankseitig durch eine eigene View zum anonymisierten Auslesen der Daten realisiert. Diese View sorgt dafür, dass personenbezogene Daten nicht mit exportiert werden. Hierzu werden die Daten durch die View:

- Vollständig gelöscht: Hierzu gehören die persönlichen Angaben des Jugendlichen sowie Freitextfelder, da in diesen nicht sichergestellt werden kann, dass sich keine personenbezogenen Daten darin befinden.
- Gekürzt: Die Postleitzahl, als personenbezogenes Datum, wird zur Wahrung des Datenschutzes auf drei Stellen gekürzt, um auch weiterhin Auswertungen über die Herkunft des Jugendlichen durchführen zu können.
- Generalisiert: Das Geburtsdatum der Jugendlichen wird auf den 01.01 des jeweiligen Geburtsjahr gesetzt. Auf diese Weise kann kein Rückschluss mehr auf den Jugendlichen gezogen werde, es bleiben aber weiterhin Auswertungen zur Altersverteilung möglich.

Eine genau Beschreibung der stattfindenen Anonymisierung entnehmen Sie bitte den Erläuterungen zur Anonymisierung[14].

6.2 Übertragung der Daten

Die Übertragung der Daten von den Kompetenzagenturen auf den Auswerteserver findet in regelmäßigen Abständen statt. Der Auswerteserver ist bei der Übertragung passiv, d.h er nimmt nur die Daten entgegen, die im zugesendet werden. Das Übermitteln der Fallakten findet grundsätzlich in drei Schritten statt:

1. Export der Daten: Die Daten der einzelnen Kompetenzagenturen werden anonymisiert als XML Datei exportiert. Dabei wird pro Kompetentzagentur eine XML Datei angelegt die sämtliche Fallakten der Kompetenzagentur enthält. Die Dateien folgen in der Bennenung dem Schema *$DBNAME-$DATUM.xml*, um Sie auf der Seite des Auswerteserver einer bestimmten Kompetenzagentur zuordnen zu können. Die XML-Dateien werden vor der Übertragung zusätzlich komprimiert.
2. Übertragen der Daten: Die Übertragung der komprimierten XML Dateien geschieht über eine SSH-Verbindung. Die SSH-Schlüssel des Nutzers wurden auf dem Auswerteserver hinterlegt, um eine Passworteingabe überflüssig zu machen. Die Dateien werden jeweils direkt nach dem Export der Daten einer Kompetenzagentur übertragen.

[14] https://waska-anwender.intevation.de/DownloadBereich

3. Import der Daten: Der Import findet mit einem Tag Versatz zu dem Export auf dem Auswerteserver statt. Bei dem Import wird über die hochgeladenen Dateien iteriert. Jede Datei wird zunächst entpackt und dann in die Datenbank importiert. Bei dem Import wird anhand des Namens der Kompetentzagantur im Dateinamen die Förderkennziffer jeder einzelnen Akte überschrieben. So kann gewährleistet werden, dass die Fallakten später in den Auswertungen einer Kompetenzagentur zugeordnet werden können. Nach dem Import werden die Daten gelöscht.

Die Schritte werden sowohl auf Seiten der Kompetenzagentur als auf Seiten des Auswerteserver durch cron-jobs auf den Datenbanksystemen angestoßen. Das Ausführen und der Erfolg der Aktionen werden in einem Logfile gespeichert. Für das Ausführen der cron-jobs werden eigene Benutzer angelegt, die nur so priviligiert sind wie es für das Ausführen der cronjobs erforderlich ist.

6.3 Unterschiede zu dem normalen Betrieb

Auch wenn die Codebases des Auswerteservers identisch zu dem eines WASKA-Servers im Normalbetriebi ist, so unterscheidet sich dieser in der zur Verfügung stehenden Funktionalität. Der Umfang der Funktionen wurde auf die Auswertung und die Benutzerverwaltung beschränkt. Weiter stehen eine Auswahl von Exportmöglichkeiten zur Verfügung, die im Rahmen der Auswertung sinnvoll sind. Funtionen des Casemanagemend stehen nicht zur Verfügung. Es ist weder möglich einzelne Fallakten anzusehen, noch zu bearbeiten. Um zu gewährleisten, dass Nutzer des Auswerterteservers nur die Aktionen ausführen können für die sie priviligiert sind, wurde die Rolle *pb_ka* eingeführt.