Abschlussprojekt
DBAE-/WI-Praktikum
Wintersemester 2015/16

TvAddict

Dokumentation/ Handbuch

Teilnehmer:

Marcel Sievert
235186
Wirtschaftsinformatik, Semester 7

Daniel Weniger
234816
Wirtschaftsinformatik, Semester 7

Said Hassib Sadat
224960
IMIT, Semester 8

Benutzeroberfläche

1.Startseite

Startseite von TvAddict (Realisiert durch index.jsp), dient als Benutzeroberfläche und stellt die Buttons „ANMELDEN“ und „REGISTRATION“.

1.1 Registration

Ziel(Muss-Anforderung): Es gibt die Möglichkeit für den Benutzer sich zu registrieren und einen persönlichen Account zu erstellen.

Umsetzung: Über den Button 

gelangt man zur Registrierung.

Registrierungsseite von TvAddict (Realisiert durch registrieren.jsp), dient als Oberfläche zur Registrierung und stellt ein „Eingabeformular zur Registrierung“.

1.1.1 Eingabeformular zur Registrierung

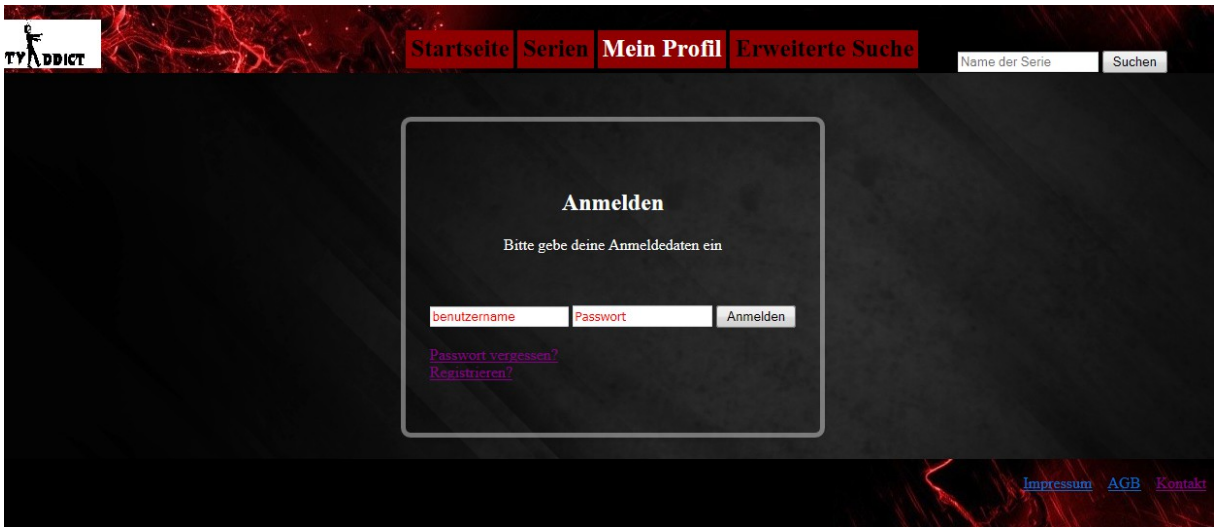
Das Eingabeformular zur Registrierung stellt die Felder Benutzername, Vorname, Nachname, E-Mail, Passwort und Passwort wiederholen. Über den Button „registrieren“ werden die Daten abgesendet und über das Servlet (Realisierung durch RegistrierenServlet.java) wird unter bestimmten Bedingungen (Benutzername noch nicht vergeben, Passwortwiederholung stimmt überein) ein neuer Benutzer in der Datenbank angelegt.[1]

1.2 Login

Ziel(Muss-Anforderung): Das System muss dem Benutzer die Möglichkeit geben sich in seinen Account einzuloggen und auszuloggen.

Umsetzung: Über den Button 

gelangt man zum Login.



Loginseite von TvAddict (Realisierung durch Anmelden.jsp), dient als Oberfläche zur Anmeldung und stellt ein Eingabeformular zum Login auf TvAddict.

1.2.1 Eingabeformular zum Login

Das Eingabeformular zum Login stellt die Felder Benutzername und Passwort. Über den Button „Anmelden“ werden die Daten abgesendet und über das Servlet (Realisierung durch AnmeldenServlet.java) wird mit Hilfe der Methode einer Helferklasse (Realisierung durch SearchObject.java, Methode: anmeldeAbfrage) abgefragt ob der Benutzer in der Datenbank vorhanden ist.[2]

1.2.2 Möglichkeit zum ausloggen

Das System teilt einem angemeldeten Benutzer auf den Seiten von TvAddict mit als welche Person der Benutzer eingeloggt ist und stellt einen Button zur Abmeldung.

2. Serien

Ziel(Muss-Anforderung): Das System gibt dem Benutzer ein Überblick über die vorhandenen Serien. Die Serien sind alphabetisch geordnet. Für jede Serie ist ein Link zu einer Detailsicht gegeben.

Umsetzung: Man erreicht die Serienübersicht über die Navigationsleiste, welche im oberen rechten Bildrand positioniert ist.

Serienübersicht von TvAddict (Realisierung durch serienUbersicht.jsp), dient als Oberfläche zur Betrachtung der vorhandenen Serien und bietet Links zu Detailansichten der Serien.

2.1 Serienübersicht

Name	Beschreibung	FSK	Detailansicht
Lucifer	Der Teufel nimmt sich Urlaub und hilft dem LAPD.	16	Serieninfo
Suits	Der geniale College-Abbrecher Mike Ross gibt sich als Anwalt aus.	12	Serieninfo
The Big Bang Theory	4 Nerds treffen auf die hübsche Blondine Penny.	12	Serieninfo
Blindspot	Eine unbekannte Frau ohne Erinnerung versucht mit den Tattoos auf ihrem Körper rauszufinden wer sie ist.	16	Serieninfo
Supernatural	2 Brüder gegen Dämonen und andere Monster.	16	Serieninfo
Limitless	Die Droge NZT erweitert Brian Sinclairs Bewusstsein und ist dem FBI damit eine große Hilfe.	16	Serieninfo

Auf der Seite Serien wird dem Benutzer in Tabellenform ein Überblick über alle in der Datenbank vorhandenen Serien bereitgestellt. In der Tabelle werden Name, Beschreibung, FSK und ein Link zur Detailsicht der jeweiligen Serie angezeigt.
Name, Beschreibung und FSK der Serie werden über ein Servlet mittels der Helferklasse SearchObject.java und der darin enthaltenen uebersichtSearch()-Methode[3] ermittelt und in einem ResultSet gespeichert. Nach der Konvertierung stehen die ermittelten Daten der Tabelle der serienUebersicht.jsp zur Verfügung.

3.Suchfunktionen

3.1. Schnelle Suche

Ziel(Muss-Anforderung): Das System muss den Benutzern eine Seriensuche zur Verfügung stellen. Serien können per Namen gesucht werden, wobei Wortteile eines Serienamens auch zu einem Treffer führen.

Umsetzung: Das System bietet dem Benutzer auf der Oberfläche von TvAddict ein Eingabefeld, in welches der Benutzer den Namen einer Serie oder Wortteile einer Serie eingeben kann. Über den Button „Suchen“ erfolgt dann die Suche nach Treffern in der Datenbank.

Die Suche verweist auf das „SchnelleSucheServlet“, in welchem der Parameter Name entgegengenommen, ein SearchObject angelegt und durch dieses mittels der simpleSearch()-Methode[4] der Helferklasse ein ResultSet der Serien gespeichert wird. Dieses wird dann noch konvertiert.

Die Ergebnisse der Suche werden dann dem Benutzer in Tabellenform zur Verfügung gestellt.

Name	Beschreibung	FSK	Detailansicht
Lucifer	Der Teufel nimmt sich Urlaub und hilft dem LAPD.	16	Serieninfo

Die Suchergebnisse werden mit Name, Beschreibung, FSK und einem Link zur Detailansicht ausgegeben.

3.2 Erweiterte Suche

Ziel(Soll-Anforderung): Mit einer erweiterten Suche können Serien auch nach Genre oder Altersbegrenzung gesucht werden.
Der Link zu der erweiterten Suchfunktion leitet auf die dafür vorgesehene Seite um.

Umsetzung: Man erreicht die Erweiterte Suche über die Navigationsleiste, welche im oberen rechten Bildrand positioniert ist.

Über den Link **Erweiterte Suche** wird die Erweiterte Suche aufgerufen, welche zusätzlich zum Eingabefeld für den Namen der Serie, ein Eingabefeld für die Altersfreigabe, sowie ein Auswahl des Genres bereitstellt.

4. Accountverwaltung

Ziel(Muss-Anforderung): Das System bietet dem Benutzer die Möglichkeit die persönlichen Daten zu ändern. Dies beinhaltet das Passwort, E-Mail sowie Lieblingsgenre.

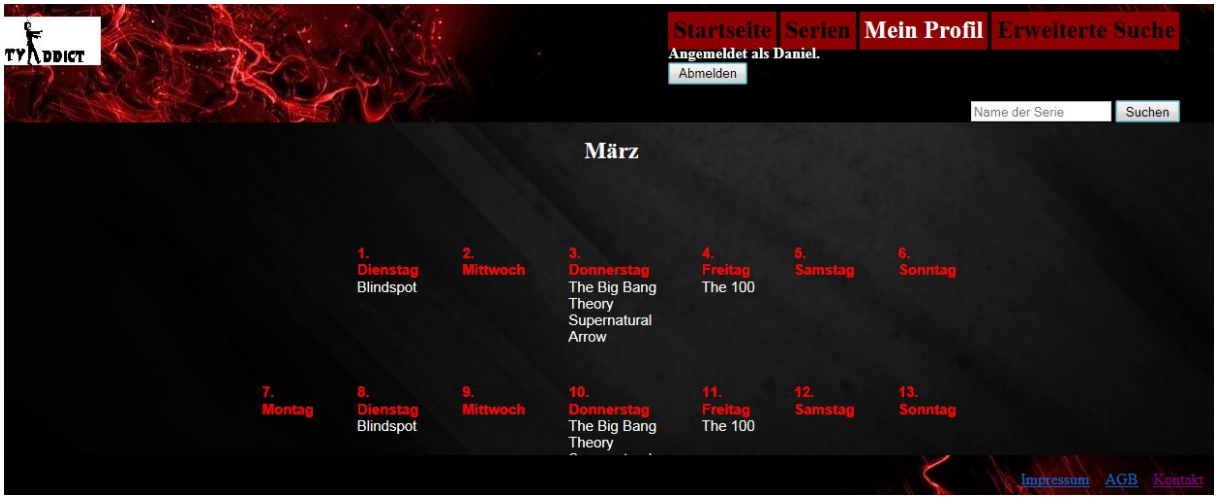
Umsetzung: Auf der Seite „Mein Profil“ bietet das System dem Benutzer die Möglichkeit sein Passwort und seine Emailadresse zu ändern.

Die Änderung des Passworts erfolgt über das PasswortAendernServlet[5] und die Änderung der Emailadresse erfolgt über das EmailBearbeitenServlet[6].
Lieblingsgenre: Wurde im Verlauf des Projekts beschlossen herauszunehmen, da es innerhalb des TvAddict-Systems keine weitere Funktionalität darstellt.

5. Sendeplan

Ziel(Muss-Anforderung): Das System muss den Benutzern die Möglichkeit bieten aus den vorhandenen Serien für sie interessante Serien zu markieren. Das System muss den Benutzern eine Art ''Sendeplan'' für ihre ausgewählten Fernsehserien bieten. Dabei wird ein Monatsüberblick angezeigt.

Umsetzung: Auf der Seite „Mein Profil“ bietet das System dem Benutzer eine Art ''Sendeplan'' für die vorhandenen Serien.



Dabei wird dem Benutzer ein Monatsüberblick angezeigt, welcher durch die getSendeplan()-Methode[7] der Helferklasse Sendeplan.java realisiert wird. Durch die Methode wird der Sendeplan eines Benutzers in einem Monatsüberblick erstellt. Dieser wird als String zur direkten Ausgabe in der meinProfil.jsp erstellt.

In der Detailansicht der jeweiligen Serie besteht die Möglichkeit die Serie zum Sendeplan hinzuzufügen oder die Serie aus dem Sendeplan zu nehmen.

6. Administratoren-Bereich

Ziel(Muss-Anforderung): Das System muss Administratoren eine Oberfläche bieten, welche es ihnen ermöglicht rollenspezifische Aktionen durchzuführen.
Das System muss Administratoren ermöglichen Benutzer zu bearbeiten. (Löschen/Editieren)
Das System muss Administratoren die Möglichkeit bieten Serien zu löschen, editieren oder neu anzulegen.

Umsetzung: Bei der Anmeldung wird geprüft ob es sich um einen Benutzer oder einen Mitarbeiter handelt. Hat man sich als Mitarbeiter angemeldet wird man von der anmeldeCheck.jsp auf die MitarbeiterFunktionen weitergeleitet.

Hier wird dem Mitarbeiter
… eine Benutzerübersicht

Benutzerübersicht

(Mitarbeiter ausgeschlossen)

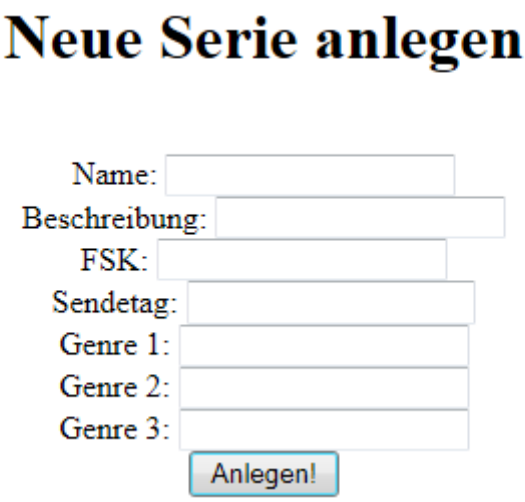
ID	Benutzername	Vorname	Nachname	Email	Verschl. PW	Bearbeiten	Löschen
7	TestBenutzer	Max	Mustermann	mm@uni-hildesheim.de	DNEP62krF6rwZGH65v+65w==	Bearbeiten	Löschen
10	regpwtest	reg	pwtest	pwtest@huhu.de	DNEP62krF6rwZGH65v+65w==	Bearbeiten	Löschen
5	MingLee	copy	pasterino	email@uni-hildesheim.de	DNEP62krF6rwZGH65v+65w==	Bearbeiten	Löschen

… eine Serienübersicht

Serienübersicht

ID	Name	Beschreibung	FSK	Sendetag	Genre 1	Genre 2	Genre 3	Bearbeiten	Löschen
1	Lucifer	Der Teufel nimmt sich Urlaub und hilft dem LAPD.	16	Montag	Fantasy	Comedy	Crime	Bearbeiten	Löschen
2	Suits	Der geniale College-Abbrecher Mike Ross gibt sich als Anwalt aus.	12	Donnerstag	Drama	Comedy		Bearbeiten	Löschen
4	Blindspot	Eine unbekannte Frau ohne Erinnerung versucht mit den Tattoos auf ihrem Körper rauszufinden wer sie ist.	16	Dienstag	Crime	Mystery	Action	Bearbeiten	Löschen
5	Supernatural	2 Brüder gegen Dämonen und andere Monster.	16	Donnerstag	Drama	Action	Fantasy	Bearbeiten	Löschen
6	Limitless	Die Droge NZT erweitert Brian Sinclairs Bewusstsein und ist dem FBI damit eine große Hilfe.	16	Mittwoch	Drama	Crime	Thriller	Bearbeiten	Löschen
7	The 100	97 Jahre nach einem Atomkrieg versuchen 100 Jugendliche herauszufinden, ob die Erde wieder bewohnbar ist.	16	Freitag	Science-Fiction	Action		Bearbeiten	Löschen
8	Arrow	Fünf Jahre auf einer einsamen Insel bringen Oliver Queen dazu in Starling City für Gerechtigkeit zu sorgen.	16	Donnerstag	Drama	Action		Bearbeiten	Löschen
3	The Big Bang Theory	4 Nerds treffen auf die hübsche Blondine Penny.	12	Donnerstag	Comedy	Fantasy		Bearbeiten	Löschen
10	TestSerie	Plot	18	Montag	Comedy	Fantasy	Crime	Bearbeiten	Löschen

… sowie ein Formular zur Serienerstellung



bereitgestellt.

Die Bearbeitung und Löschung von Benutzern und Serien erfolgt über die jeweiligen Servlets.

- BenutzerBearbeitenServlet[8]
- SerieBearbeitenServlet[9]
- BenutzerLoeschenServlet[10]
- SerieLoeschenServlet[11]

Welche auf die Helferklasse DatabaseEdit.java zurückgreifen, durch welche die Änderung bzw. Löschung der Daten von Benutzern und Serien realisiert wird.

Anhang

Ausschnitte des Quellcodes zur funktionalen Erläuterung:

[1] Neuen Benutzer anlegen:

```
if(pwCheck && (!usernameAlreadyExists)){
    new DatabaseEdit().addBenutzer(benutzername, vorname, nachname, email, pw);
}
```

[2] Anmeldungsabfrage:

```
String benutzername = request.getParameter("benutzername");
String passwort = request.getParameter("passwort");

SearchObject so = new SearchObject();
//Wenn abfrage = 1 dann gibt es die benutzer+passwort Kombination. Wenn 0, dann ist mindestens eins davon falsch (Gibt es nicht)
int abfrage = so.anmeldeAbfrage(benutzername, passwort);
```

[3] Ermittlung und Speicherung aller Serien zur Serienübersicht

```java
/**
 * Alle Serien in der Datenbank werden zur Übersicht in ein ResultSet gespeichert.
 *
 */
public ResultSet uebersichtSearch() {
    String sql = "SELECT serie_name, beschreibung, fsk FROM serie";
    try {
        PreparedStatement pstmt = con.prepareStatement(sql);
        result = pstmt.executeQuery();
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return result;
}
```

[4] simpleSearch()-Methode

```java
public ResultSet simpleSearch(String name) {
    //Basic SQL Statement
    String sql = "SELECT serie_name, beschreibung, fsk FROM serie WHERE TRUE";
    //Leerer where-String
    String where ="";
    //Boolean der je nach dem ob der Benutzer einen Begriff eingegeben hat auf true oder false gesetzt wird.
    boolean nameIsSet = true;
    //Abfrage um zu überprüfen ob der Benutzer nach einem Begriff gesucht hat
    if (!name.equals("") && name != null) {
        where += " AND serie.serie_name ilike ?";

    } else {
        //Wenn nichts eingegeben wurde, wird der boolean auf false gesetzt um die "?" Platzhalter nur zu setzen, wenn der
        //"where" String mit einer Bedingung erweitert wurde-
        nameIsSet = false;
    }

    //SQL Statement & Where Statement verbinden
    String completeSQL = sql + where;

    //Prepared Statement in ResultSet
    try {
        PreparedStatement pstmt = con.prepareStatement(completeSQL);
        //Platzhalter nur setzen, wenn etwas eingegeben wurde, sonst gibt es einen Fehler, da kein Parameter gesetzt werden müsste.
        if(nameIsSet){
            pstmt.setString(1, "%" + name + "%");
        }
        //Statement wird ausgeführt.
        result = pstmt.executeQuery();
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return result;
}
```

[5]PasswortAendernServlet doPost-Methode:

```java
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    //Die nötigen Informationen werden geholt und gespeichert.
    HttpSession session = request.getSession();
    Benutzer benutzer = (Benutzer) session.getAttribute("benutzer");
    String neuesPasswort = request.getParameter("neuesPasswortBenutzerEdit");
    String neuesPasswortWdh = request.getParameter("neuesPasswortWdhBenutzerEdit");

    //Fehlermeldung, die sich je nach Fehler verändert.
    String errorMessage = "";

    //Prüft, ob die beiden Passwort Inputs übereinstimmen und passt entsprechend die Fehlermeldung an.
    if(neuesPasswort.equals(neuesPasswortWdh)){
        String verschlPW = new Verschluesseln().pwVerschluesseln(neuesPasswort);
        new DatabaseEdit().meinProfilPasswortEdit(verschlPW, Integer.parseInt(benutzer.getUserID()));
        errorMessage = "Passwort erfolgreich geändert!";
    } else {
        errorMessage = "Passwort stimmt nicht überein!";
    }
    request.setAttribute("errorMessage", errorMessage);
    request.getRequestDispatcher("meinProfil.jsp").forward(request, response);
}
```

[6]EmailBearbeitenServlet doPost-Methode

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    //Die nötigen Informationen werden geholt und gespeichert.
    HttpSession session = request.getSession();
    Benutzer benutzer = (Benutzer) session.getAttribute("benutzer");
    String neueEmail = request.getParameter("neueEmailBenutzerEdit");
    String neueEmailWdh = request.getParameter("neueEmailWdhBenutzerEdit");
    boolean emailAlreadyExists = false;
    List<Benutzer> benutzerList = new SearchObject().benutzerSearch();
    //Fehlermeldung, die sich je nach Fehler verändert.
    String errorMessage = "";
    //Prüft ob email bereits benutzt wird.
    for (Benutzer user : benutzerList) {
        if(user.getEmail().equals(neueEmail)) {
            emailAlreadyExists = true;
            errorMessage = "Email bereits vergeben.";
        }
    }
    //Prüft, ob die beiden Email Inputs übereinstimmen und passt entsprechend die Fehlermeldung an.
    if(neueEmail.equals(neueEmailWdh) && (!emailAlreadyExists)){
        new DatabaseEdit().meinProfilEmailEdit(neueEmail, Integer.parseInt(benutzer.getUserID()));
        errorMessage = "Email erfolgreich geändert!";
    } else {
        if(!(errorMessage.equals("Email bereits vergeben."))) {
            errorMessage = "Email stimmt nicht überein!";
        }
    }
    request.setAttribute("errorMessage", errorMessage);
    request.getRequestDispatcher("meinProfil.jsp").forward(request, response);
}
```

[7]getSendeplan()-Methode

```
public String getSendeplan() {
    //
    sendeplan = "<h2>" + monthNames[aktuellerMonat] + "</h2><br><table id=\"sendeplan\"><tr style=\"vertical-align: top\">";
    int weekCounter = 1;
    //Schleife um für jeden Tag im Monat. Es wird dabei die Anzahl der Tage des aktuellen Monats berücksichtig.
    for (int tagInMonat = 1; tagInMonat <= maxDayMonth; tagInMonat++) {
        Calendar temp = Calendar.getInstance();
        temp.set(aktuellesJahr, aktuellerMonat, tagInMonat);

        //Differenz der DAY_OF_WEEK um die Anzahl leer-<td> zu bekommen
        int leerTdCount = temp.get(Calendar.DAY_OF_WEEK)-2;
        //Vor dem ersten Tag sollen alle Tage des letzten Monats der aktuellen Woche mit leer-<td> gesetzt werden um die Tabelle einheitlich zu gestalten
        while (tagInMonat == 1 && leerTdCount > 0) {
            sendeplan += "<td></td>";
            leerTdCount--;
        }
        //Tag und Wochentag Ausgabe
        sendeplan += "<td><label style=\"color: red\"><b>"+temp.get(Calendar.DAY_OF_MONTH) +". "+ wochenTage[temp.get(Calendar.DAY_OF_WEEK)-1]+"</b></label><br>";
        //Jede Serie, die den Sendetag am aktuellen Wochentag im Monat hat wird dem Tag hinzugefügt.
        for (Serie serie : serieList) {
            if(serie.getSendetag().equals(wochenTage[temp.get(Calendar.DAY_OF_WEEK)-1])) {
                sendeplan += serie.getName() + "<br>";
            }
        }
        sendeplan += "</td>";
        //Nach jedem Sonntag wird eine neue Woche gestartet.
        if(wochenTage[temp.get(Calendar.DAY_OF_WEEK)-1].equals("Sonntag")) {
            weekCounter++;
            //System.out.println("Woche: "+ weekCounter);
            sendeplan += "</tr><tr style=\"vertical-align: top\">";

        }
    }
    sendeplan += "</table>";
    return sendeplan;
}
```

[8]BenutzerBearbeitenServlet doPost-Methode:

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    // Neues Benutzer-Objekt erstellen
    Benutzer benutzer = new Benutzer();

    // Setzen der Benutzer-Objekt-Atrribute
    benutzer.setUserID(request.getParameter("userID"));
    benutzer.setBenutzername(request.getParameter("benutzername"));
    benutzer.setVorname(request.getParameter("vorname"));
    benutzer.setNachname(request.getParameter("nachname"));
    benutzer.setEmail(request.getParameter("email"));
    benutzer.setVerschluesseltesPW(request.getParameter("verschluesseltesPW"));

    // Attribut Benutzer wird gesetzt
    request.setAttribute("benutzer", benutzer);

    request.getRequestDispatcher("benutzerBearbeiten.jsp").forward(request, response);
}
```

[9]SerieBearbeitenServlet doPost-Methode:

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    Serie serie = new Serie();
    serie.setId(request.getParameter("id"));;
    serie.setName(request.getParameter("name"));
    serie.setBeschreibung(request.getParameter("beschreibung"));
    serie.setFsk(request.getParameter("fsk"));
    serie.setSendetag(request.getParameter("sendetag"));
    serie.setGenre1(request.getParameter("genre1"));
    serie.setGenre2(request.getParameter("genre2"));
    serie.setGenre3(request.getParameter("genre3"));


    request.setAttribute("serie", serie);
    request.getRequestDispatcher("serieBearbeiten.jsp").forward(request, response);
}
```

[10]BenutzerLoeschenServlet doPost-Methode:

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    // BenutzerID wird der deleteBenutzer-Methode übergeben
    String userID = request.getParameter("userID");
    new DatabaseEdit().deleteBenutzer(userID);

    request.getRequestDispatcher("MitarbeiterFunktionenServlet").forward(request, response);
}
```

[11]SerieLoeschenServlet doPost-Methode:

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    //SerieID wird der deleteSerie Methode übergeben um sie zu löschen.
    String id = request.getParameter("id");
    new DatabaseEdit().deleteSerie(id);

    request.getRequestDispatcher("MitarbeiterFunktionenServlet").forward(request, response);
}
```