**Aufgabenstellung**

**Web**

1. **Änderung Registrierung**
   1. User-Klasse: Passwortfeld + PasswordRetype-Feld hinzufügen
   2. Id-Feld löschen (falls vorhanden), da das eMail-Feld der Primary Key ist
   3. Db: EMail soll unique sein + primary key, PasswordRetype soll nicht in der Db aufscheinen (Fluent-API bzw. Annotations)
   4. Migrations + Kontrolle der Db-Tabelle
   5. Registration-View anpassen
   6. Registration()-Methode anpassen
   7. Passwort soll als Hash abgespeichert werden (siehe Internet: Klasse *PasswordHasher*)
   8. User registrieren lassen und Daten in der Db überprüfen
2. **Login**

* 1. komplett programmieren (View, Methoden, ...)
  2. bei Erfolg Umleitung z.B. auf die Startseite

1. **Userverwaltung**
   1. alle User der DB anzeigen
   2. neben jedem User jeweils einen Button fürs *Löschen* und *Ändern* erzeugen
   3. **Löschen** komplett programmieren + Überprüfung (DB)
   4. **Ändern** komplett programmieren (Hinweis: die Ändern-View ist identisch zur

Registration-View – verwende die Registration-View (keine Kopie)) + Überprüfung (DB)

(+ *ViewBag*)

1. **Sessionverwaltung**
   1. Theorie Sessions
   2. Session einbinden: im Controller
   3. Session einbinden: in einer View
   4. *Session-Erweiterungsmethode* für beliebige Objekte programmieren (standardmäßig kann in iner Session nur **int** und **string** verw. werden)
   5. Login: den eingeloggten User in einer Sessionvariable speichern (ohne Passwort)
   6. \_Layout: den Namen des eingeloggten Users anzeigen + Link *Logout* -> soll nur angezeigt werden, falls jemand eingeloggt ist
   7. Logout programmieren + zugehörige Sessionvariable löschen
   8. Die Links *Login* bzw. *Logout*sollten nur angezeigt werden, wenn es notwendig ist (nach dem **Login** soll der *Login*-Link nicht mehr angezeigt werden, aber der *Logout*-Link soll nun erscheinen; nach dem **Logout**, soll der *Login*-Link erscheinen und der *Logout*-Link soll verschwinden)
2. **Userverwaltung anpassen**
   1. Button *Ändern* (siehe 3.a + 3.b) entfernen
   2. Link *Userdaten ändern* in Navigation hinzufügen
   3. dieser Link soll nur angezeigt werden, wenn tatsächlich jemand eingeloggt ist
   4. wird dieser Link angeklickt, soll die Ändern-View mit den Daten des aktuell eingeloggten Users (*Session*) angezeigt werden
   5. *Ändern* programmieren/anpassen - falls notwendig + testen
3. **JavaScript**

Wir verwenden JavaScript für 2 spezielle Aufgaben. *Erstens* für die **Formularvalidierung** am **Client-PC** und *zweitens*, um **asynchrone JavaScript-Requests** (**AJAX-Requests**) zu senden (wir möchten im Hintergrund Daten (im JSON-Format) vom Server holen und diese Daten dann in die html-Seite mit Hilfe von JavaScript einbinden).

* 1. js-Datei und CSS-Datei *global einbinden*
  2. js-Datei und CSS-Datei nur für eine *spezielle View einbinden*
  3. js-Code nur ausführen, wenn die HTML-Seite komplett geladen ist
  4. **clientseitige** **Validierung**
     1. Vorteile/Nachteile gegenüber der serverseitigen Validierung
     2. führe eine **Überprüfung des Registrierungsformulars** clientseitig durch (verwende die gleichen Kriterien wie am Server) und sende das Formular nur an den Server, wenn alles richtig ist
     3. Ist eine serverseitige Validierung trotzdem sinnvoll?
  5. **AJAX**-**Requests**

Änderungen am Projekt: erzeuge ein weiteres Feld *Role* in der Userklasse. In ihr soll die Rolle des Users abgespeichert werden (mögliche Rollen: *RegisteredUser*, *Admin*). Ändere die Klasse User, erzeuge die Migration und übertrage sie an die DB + Überprüfung in der DB. Füge bei allen Usern (direkt in der DB) einen Wert für *Role* ein. Überprüfe, ob deine Anwendung weiterhin fehlerfrei funktioniert.

* + 1. Änderungen in der View *Alle User anzeigen*
       1. füge oberhalb der Tabelle ein Dropdown-Listenfeld hinzu
       2. wird diese Seite geladen, soll per AJAX-Request alle Rollen aus der DB ermittelt und in die Dropdown-Liste eingefügt werden
       3. füge zusätzlich den Eintrag „alle Rollen“ hinzu
       4. wird nun ein Eintrag in der Dropdown-Liste ausgewählt, sollen nur jene User per AJAX-Request aus der DB geladen werden, welche der ausgewählten Rolle entsprechen

1. Userrollen verwenden

Da bei jedem User die Userrolle (RegisteredUser, Admin) in der DB abgespeichert wurde, können wir das verwenden, um den Zugriff auf bestimmte Bereiche unserer Webanwendung einzuschränken.

* 1. Die Anzeige aller User, und somit auch das Löschen von Usern, soll nur für einen Admin möglich sein