**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. Integration **WebAPI** (RestfuService, MicroService) – es ist problemlos möglich ein *ASP.Net Core MVC Projekt* um einen *WebAPI-Teil* zu erweitern (dafür muss kein neues Projekt erzeugt werden – ASP.Net Core MVC ist dafür vorbereitet)

Programmiere z.B. **6.a.i.** und zum testen gleich **6.b.i**.

* 1. **serverseitig**: füge deinem Projekt einen WebAPI-Controller (erzeuge einen Unterordner **API** unterhalb des Ordners *Controllers*) mit folgenden Methoden hinzu
     1. einen User zurückliefern
     2. alle User zurückliefern
     3. Userdaten eines Users ändern
     4. einen User löschen
     5. einen User hinzufügen
  2. **clientseitig**: erzeuge ein zusätzliches Projekt (z.B. vom Typ *Console*) und greife auf die WebAPI (aus **a.**) zu und (Klasse **HttpClient** verwenden)
     1. ermittle einen User + Ausgabe
     2. ermittle alle User + Ausgabe
     3. ändere die Daten eines Users
     4. lösche einen User
     5. einen User hinzufügen