ClubAdministration (Auth)

Lehrziele

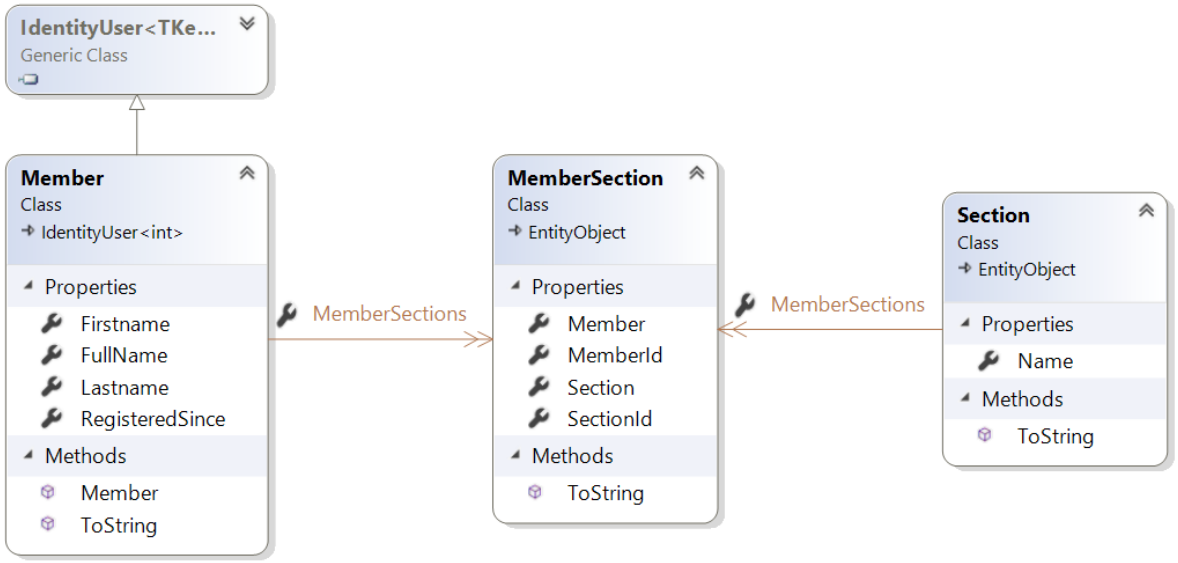
- ASP.NET Core Razor Pages
- ASP.NET Core WebApi
- Entity Framework Core
- Microsoft Identity

Es ist eine einfache Mitgliederverwaltung für einen Verein mit mehreren Sektionen zu erstellen. Ein Mitglied (Member) kann in mehreren Sektionen (Section) angemeldet sein. Die Zuordnung wird über die assoziative Entität MemberSection verwaltet.

Die Funktionalität ist bereits zum Großteil vorhanden, muss allerdings um Sicherheitsaspekte erweitert werden.

Core

Die Entitätsklassen sind bereits angelegt.

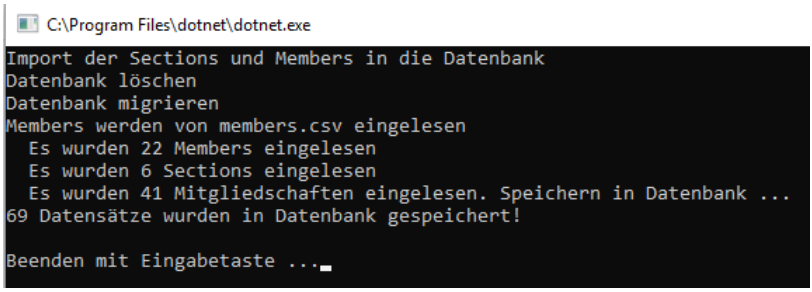


TODOs

1. Erweitern Sie Sie die Klasse „Member“ sodass diese per Microsoft Identity zum Login verwendet werden kann.
2. Im Corelayer sind die Contracts für die Repositories bedarfsgerecht zu erweitern (siehe Razor Pages bzw. Web API).

Import / Persistence

Es werden die 22 Mitglieder mit ihren 41 Mitgliedschaften in 6 Sektionen aus der Datei members.csv importiert.



Die ConnectionStrings wurden in den relevanten Projekten schon in der appSettings.json festgelegt.

TODOs:

1. Adaptieren Sie die Klasse „ApplicationDbContext“ sodass diese von Microsoft Identity verwendet werden kann.
2. Erstellen Sie die Migration „InitialMigration“
   a. StartUp Project: „ClubAdministration. Persistence”
   b. Default Project: „ClubAdministration.Persistence“
3. Wenden Sie die Migration in der Datenbank an.
4. Berechnen Sie in der ImportConsole die Username-Eigenschaft der importierten Members im Format vorname.nachname@htl.at. Entfernen Sie alle vorhandenen Umlaute mit der Hilfsmethode `MyString.ConvertUmlaute(..)` und speichern sie den Username in Kleinbuchstaben ab.
5. Die Repositories müssen noch mit den fehlenden Methoden (siehe Razor Pages/Web API) erweitert werden.

ASP.NET Core Razor Pages (ClubAdministration.Web)

Benutzer-Initialisierung

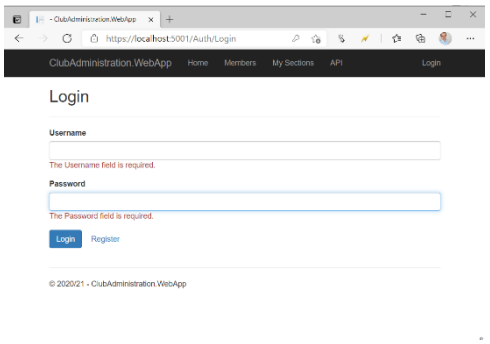
TODOs

1. Prüfen Sie beim Programmstart, ob bereits ein Benutzer mit dem Username admin@htl.at existiert.
2. Wenn nein => Initialisierung durchführen
   a. Benutzer admin@htl.at mit dem Passwort „Admin12345!“ erstellen.
   b. Rolle „Admin“ erstellen und den Admin-Benutzer die neue Rolle zuweisen.

Page „Login“

TODOs

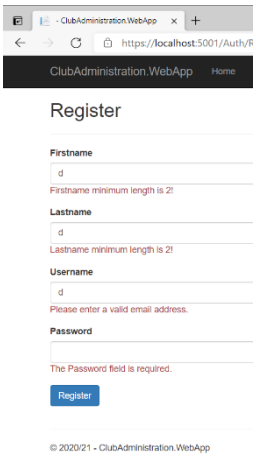
1. Implementieren Sie die Login-Logik
2. Im Fehlerfall sollen keine sensiblen Daten preisgegeben werden -> Halten Sie die Fehlermeldung so allgemein wie möglich: „Login failed“.

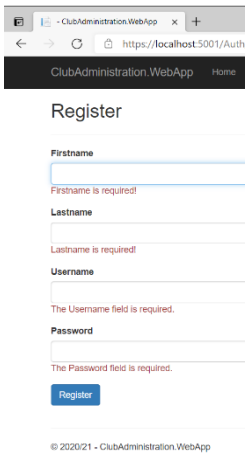


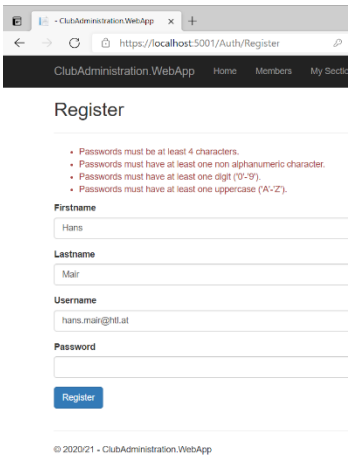
Page „Register“

TODOs

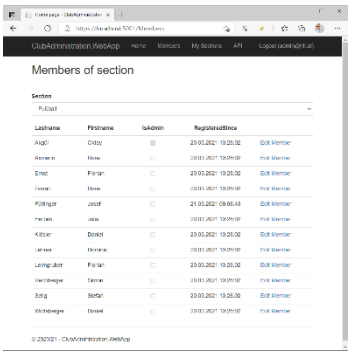
1. Erstellen Sie die Register-Razor Page lt. Screenshot.
2. Implementieren Sie die Register-Logik.
    a. Im Erfolgsfall soll anschließend auf die Login-Page weitergeleitet werden.
3. Sichern Sie die Register-Page mit den dargestellten Validierungen ab.







Page „Members/Index“



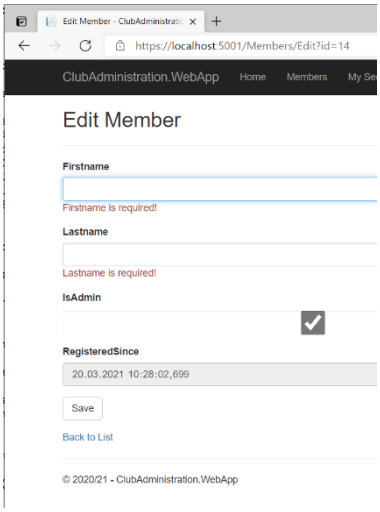
TODOs:

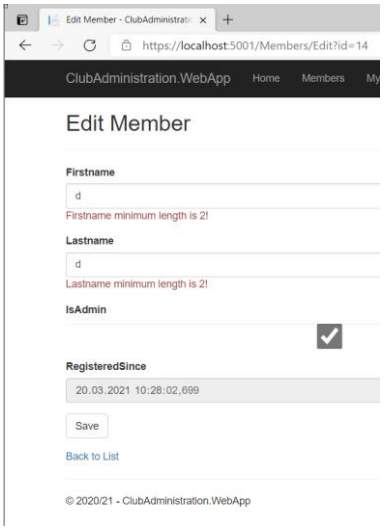
1. Nur Benutzer mit der Rolle „Admin“ dürfen diese Seite aufrufen!
2. Berechne die „IsAdmin“-Eigenschaft

Page „Members/Edit“

TODOs:

1. Nur Benutzer mit der Rolle „Admin“ dürfen diese Seite aufrufen!
2. Implementieren Sie das Laden der Member-Daten.
3. Implementieren Sie das Speichern der Member-Daten (inkl. der „Admin“-Rollenmitgliedschaft). Nach erfolgreicher Bearbeitung des Mitgliedes ist wieder die Übersichtsseite (Members/Index) aufzurufen.
4. Implementieren Sie die im Screenshot dargestellten Validierungen!



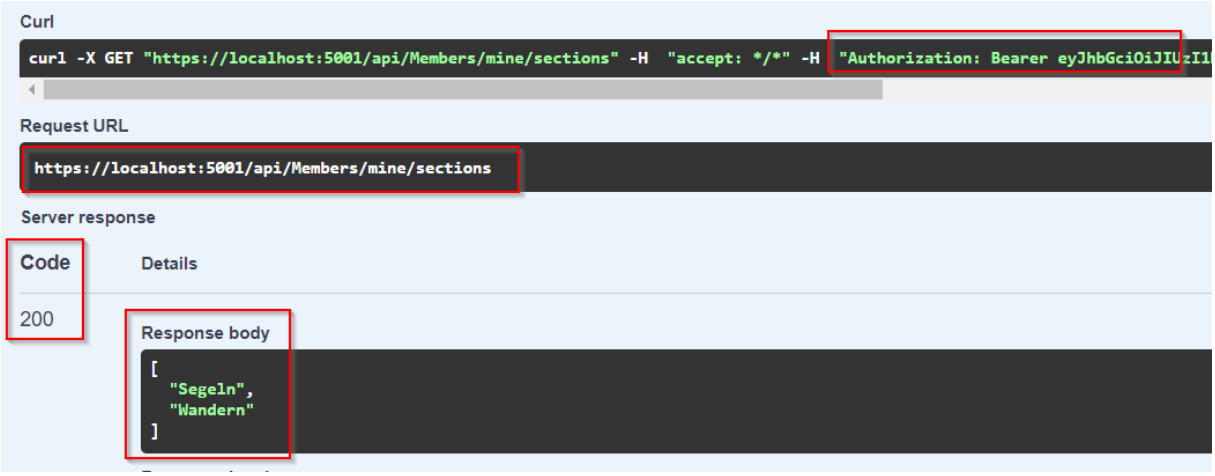


5. <u>Spezialistenaufgabe:</u> Der geänderte Name (Firstname und Lastname) darf nicht mit bereits in der Datenbank befindlichen anderen Mitgliedern übereinstimmen.

ASP.NET WebApi (ClubAdministration.Web)

Route „/api/members/mine/sections“

- Liefert die Namen der Sektionen des aktuell angemeldeten Mitglieds zurück
- Sortiert nach Sektionsnamen aufsteigend



TODOs:

1. Diese Route darf nur von angemeldeten Benutzern aufgerufen werden!
2. Implementieren Sie die Action (inkl. des Datenzugriffs über die Repositories)
3. Geben Sie die die zurückgelieferten ResponseTypen an.
   Hinweis: `[ProducesResponseType(StatusCodes.StatusXXX)]`!

Route „/api/members/lastname/firstname/sections“

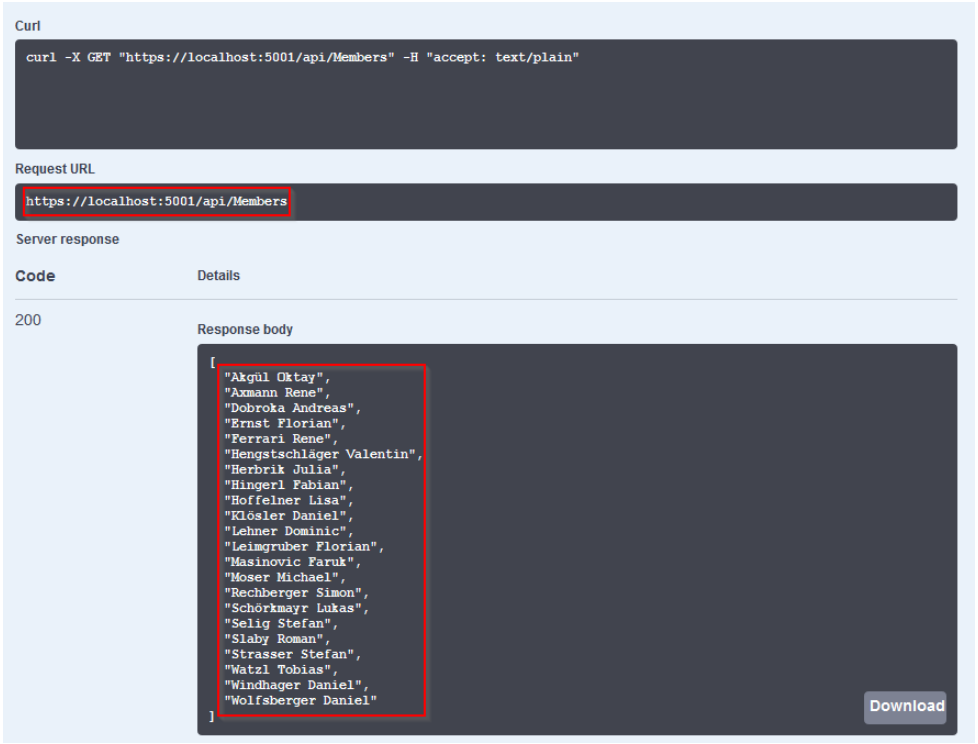
- Liefert die Namen der Sektionen des Mitglieds zurück
- Sortiert nach Sektionsnamen aufsteigend
- Gibt es das Mitglied nicht, wird NotFound zurückgegeben
- Gibt es das Mitglied, hat es aber keine Zuordnungen, wird ein leeres Array geliefert

TODOs:

1. Nur Benutzer mit der Rolle „Admin“ dürfen diese Route aufrufen.
2. Geben Sie die die zurückgelieferten ResponseTypen an.
   Hinweis: `[ProducesResponseType(StatusCodes.StatusXXX)]`!

Route „/api/members“

- Die Namen der Mitglieder sind alphabetisch sortiert auszugeben
- Der Rückgabetyp ist ein String-Array

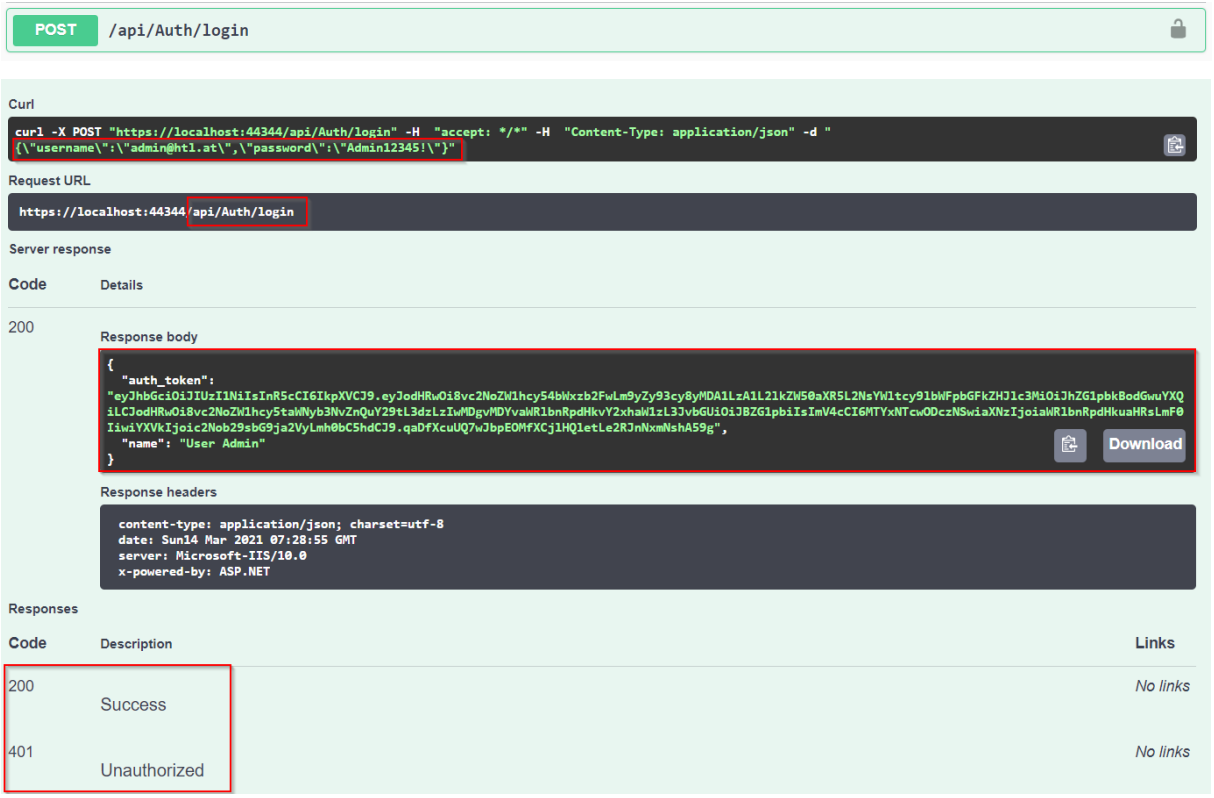


TODOs:

1. Nur Benutzer mit der Rolle „Admin“ dürfen diese Route aufrufen.
2. Geben Sie die die zurückgelieferten ResponseTypen an.
   Hinweis: `[ProducesResponseType(StatusCodes.StatusXXX)]`!

Route „/api/auth/login“

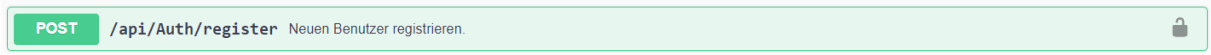
- Ermöglicht es einen Benutzer sich im API-Controller anzumelden
- Übergabe (username und password)



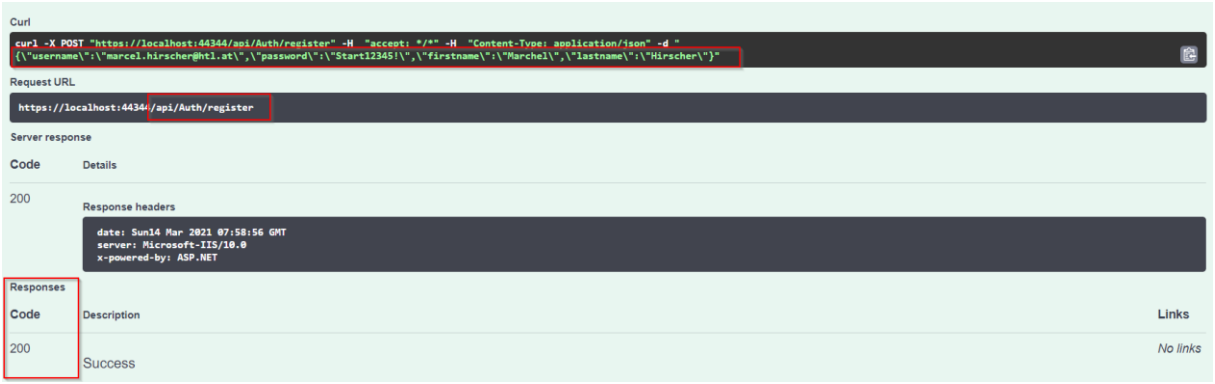
TODOs:

1. Implementieren Sie die Action
2. Geben Sie die die zurückgelieferten ResponseTypen an.
   Hinweis: `[ProducesResponseType(StatusCodes.StatusXXX)]`!

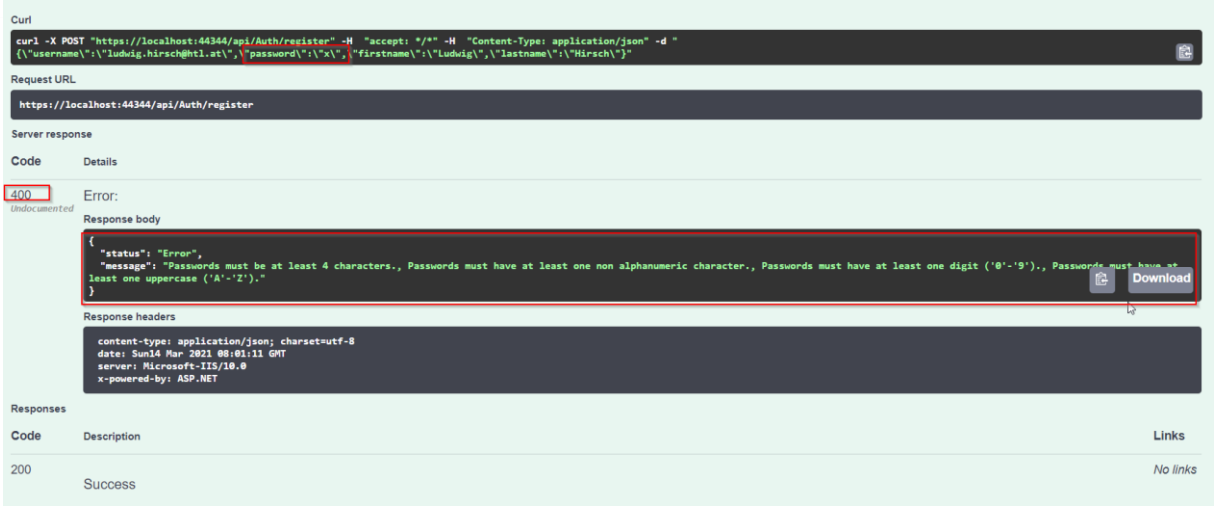
Route „/api/auth/register“



- Ermöglicht die Registrierung eines neuen Benutzers
- Alle Properties (Username, Firstname, Lastname und Password) sind verpflichtend!



Das Passwort muss folgenden Regeln entsprechen (identisch zur Register-Razor Page):



TODOs:

1. Implementieren Sie die Action
2. Geben Sie die die zurückgelieferten ResponseTypen an.
   Hinweis: `[ProducesResponseType(StatusCodes.StatusXXX)]`!