

ClubAdministration (Auth)

Lehrziele

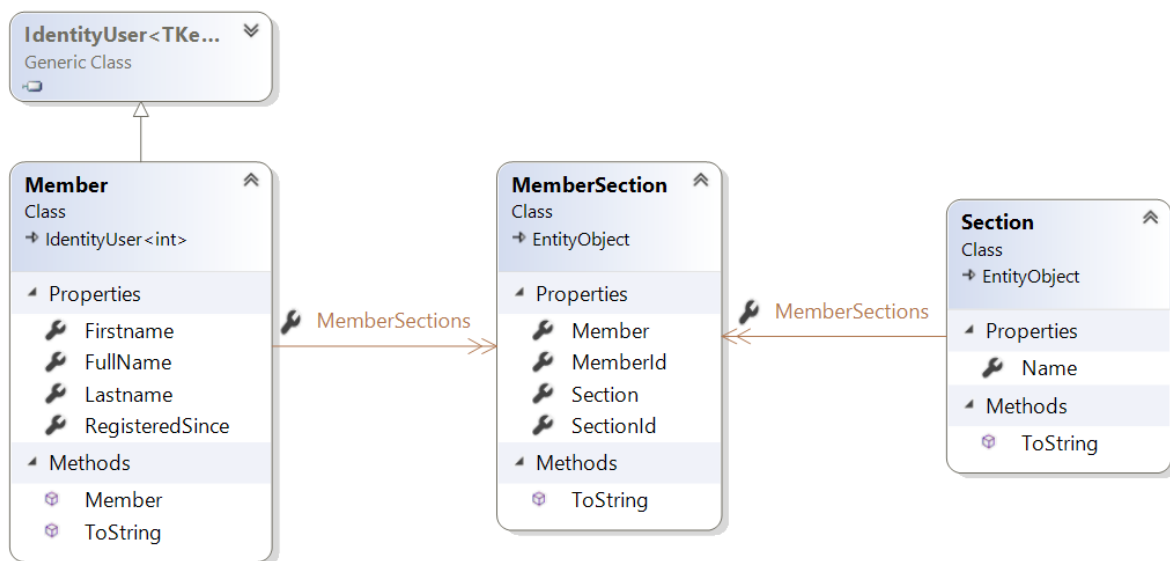
- ASP.NET Core Razor Pages
- ASP.NET Core WebApi
- Entity Framework Core
- Microsoft Identity

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 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.

```
C:\Program Files\dotnet\dotnet.exe
Import der Sections und Members in die Datenbank
Datenbank löschen
Datenbank migrieren
Members werden von members.csv eingelesen
  Es wurden 22 Members eingelesen
  Es wurden 6 Sections eingelesen
  Es wurden 41 Mitgliedschaften eingelesen. Speichern in Datenbank ...
69 Datensätze wurden in Datenbank gespeichert!

Beenden mit Eingabetaste ...
```

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.

The first screenshot shows the Register page with empty input fields for Firstname, Lastname, Username, and Password. The second screenshot shows validation errors: 'Firstname is required!', 'Lastname is required!', 'The Username field is required.', and 'The Password field is required.'. The third screenshot shows the page after successful registration, with a message: 'Passwords must be at least 4 characters.', 'Passwords must have at least one non alphanumeric character.', 'Passwords must have at least one digit (0-9)', and 'Passwords must have at least one uppercase (A-Z)'. The input fields now contain 'Hans', 'Main', 'hans.main@mail.at', and a password. A 'Register' button is visible at the bottom of each form.

Page „Members/Index“

TODOs:

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

The screenshot shows the 'Members of section' page. It features a table with columns: Lastname, Firstname, IsAdmin, and RegisteredSince. The table lists several members, including 'Angli', 'Aron', 'Eric', 'Loren', 'Pottner', 'Tamas', 'Krisz', 'Loren', 'Loren', 'Loren', and 'Loren'. Each row has an 'Edit Member' link. The 'IsAdmin' column shows a checkbox for each member.

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

The first screenshot shows the 'Edit Member' page with validation errors: 'Firstname is required!' and 'Lastname is required!'. The second screenshot shows the page after successful saving, with a message: 'Firstname minimum length is 2!' and 'Lastname minimum length is 2!'. The input fields now contain 'd' for both Firstname and Lastname. A 'Save' button is visible at the bottom of each form.

ASP.NET WebApi (ClubAdministration.Web)

Route „/api/members/mine/sections“

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

The screenshot shows a REST client interface with the following details:

- Curl:** `curl -X GET "https://localhost:5001/api/Members/mine/sections" -H "accept: */*" -H "Authorization: Bearer eyJhbGciOiJIUzI1I...`
- Request URL:** `https://localhost:5001/api/Members/mine/sections`
- Server response:**
 - Code:** 200
 - Response body:** `["Segeln", "Wandern"]`

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 zurückgelieferten ResponseTypen an.
Hinweis: `[ProducesResponseType(HttpStatusCode.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

The screenshot shows a REST client interface with the following details:

- Request URL:** `https://localhost:5001/api/Members/Ernst/Florian/sections`
- Server response:**
 - Code:** 200
 - Response body:** `["Fußball", "Tennis", "Wandern"]`

TODOs:

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

Route „/api/members“

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

The screenshot shows a REST client interface with the following sections:

- Curl:** `curl -X GET "https://localhost:5001/api/Members" -H "accept: text/plain"`
- Request URL:** `https://localhost:5001/api/Members`
- Server response:**
 - Code:** 200
 - Details:** Response body

The response body contains a JSON array of member names, which is highlighted with a red box:

```
[  
  "Akçul Oktay",  
  "Ammann Rene",  
  "Dobroka Andreas",  
  "Ernst Florian",  
  "Ferrari Rene",  
  "Hengstschläger Valentin",  
  "Herbrik Julia",  
  "Hingerl Fabian",  
  "Hoffelner Lisa",  
  "Klösl Daniel",  
  "Lehner Dominic",  
  "Leimgruber Florian",  
  "Masinovic Faruk",  
  "Moser Michael",  
  "Rechberger Simon",  
  "Schörkmaier Lukas",  
  "Selig Stefan",  
  "Slaby Roman",  
  "Strasser Stefan",  
  "Watzl Tobias",  
  "Windhager Daniel",  
  "Wolfsberger Daniel"  
]
```

A "Download" button is visible at the bottom right of the response body.

TODOs:

1. Nur Benutzer mit der Rolle „Admin“ dürfen diese Route aufrufen.
2. Geben Sie 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)

POST

/api/Auth/login

Curl

```
curl -X POST "https://localhost:44344/api/Auth/Login" -H "accept: */*" -H "Content-Type: application/json" -d "{ \"username\": \"admin@htl.at\", \"password\": \"Admin12345!\" }"
```

Request URL

```
https://localhost:44344/api/Auth/Login
```

Server response

Code	Details
200	<div>Response body<pre>{ "auth_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZWlncy54bWxzbnB2FmLm9yZyY3cy8yMDAxMTA1LTI1LjE1MDE0aXRSLDVsYWltcy9lbWVpbGFkZHJ1c3MiOiJhZGIpbk8odGwuYXQ1Cj0dHRwOi8vc2NoZWlncy5taHMyb3NvZnQuY29tLTgzLnIiwiaWF0IjoiMTcwODczNSwiaXNjaWRlbnRpdkluaHRSLmF0IiwiaXYkiOiJoaic2NoZ29sbG9ja2VyLmh0bCShdCJ9.qaDFxcuUq7wJbpEOmfXCjlHQletLe2K3nNxmnshAS9g", "name": "User Admin" }</pre></div> <div>Response headers<pre>content-type: application/json; charset=utf-8 date: Sun14 Mar 2021 07:28:55 GMT server: Microsoft-IIS/10.0 x-powered-by: ASP.NET</pre></div>

Responses

Code	Description	Links
200	Success	No links
401	Unauthorized	No links

TODOs:

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

Route „/api/auth/register“

POST
/api/Auth/register Neuen Benutzer registrieren.
🔒

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

Curl

```
curl -X POST "https://localhost:4434/api/Auth/register" -H "accept: */*" -H "Content-Type: application/json" -d '{
  \"username\": \"marcel.hirscher@htl.at\",
  \"password\": \"Start12345!@\",
  \"firstname\": \"Marcel\",
  \"lastname\": \"Hirscher\"
}'
```

Request URL

```
https://localhost:4434/api/Auth/register
```

Server response

Code	Details
200	<p>Response headers</p> <pre>date: Sun, 14 Mar 2021 07:58:56 GMT server: Microsoft-IIS/10.0 x-powered-by: ASP.NET</pre>

Responses

Code	Description	Links
200	Success	No links

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

Curl

```
curl -X POST "https://localhost:44344/api/Auth/register" -H "accept: */*" -H "Content-Type: application/json" -d '{"username":"ludwig.hirsch@htl.at","password":"x","firstname":"Ludwig","lastname":"Hirsch"}'
```

Request URL

https://localhost:44344/api/Auth/register

Server response

Code	Details
400	Error: Response body { "status": "Error", "message": "Passwords must be at least 4 characters., Passwords must have at least one non alphanumeric character., Passwords must have at least one digit ('0'-'9')., Passwords must have at least one uppercase ('A'-'Z')." }
Response headers content-type: application/json; charset=utf-8 date: Sun14 Mar 2021 08:01:11 GMT server: Microsoft-IIS/10.0 x-powered-by: ASP.NET	

Responses

Code	Description	Links
200	Success	No links

TODOs:

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