

## SchoolLocker Auth

### Lehrziele

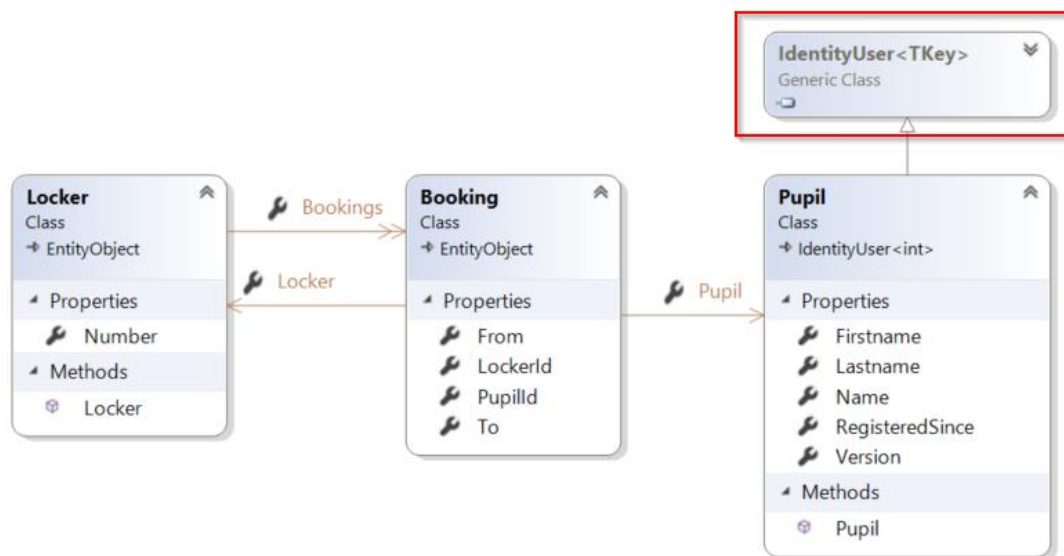
- ASP.NET Core Razor Pages
- ASP.NET Core Web API
- Entity Framework Core
- Unit of Work / Repositories
- Validierung
- Authentication / Authorization
- Microsoft Identity

Es ist eine einfache Verwaltung von Buchungen für die Spinde in der Schule zu erstellen. Im Zentrum der Anwendung steht die Buchung (Booking) eines Spindes (Locker) durch einen Schüler (Pupil). Ein Schüler kann mehrere Spinde anmieten. Ein Spind kann mehrfach, aber natürlich nicht gleichzeitig gebucht werden. Es ist möglich, bei der Buchung das Ende-Datum offen zu lassen.

Die Use Cases sind zu einem Großteil schon umgesetzt. Nun soll die Applikation noch um Authentication und Authorization auf Basis von Microsoft Identity ergänzt werden.

### Core

Die Grundstruktur der Entitätsklassen ist bereits angelegt.

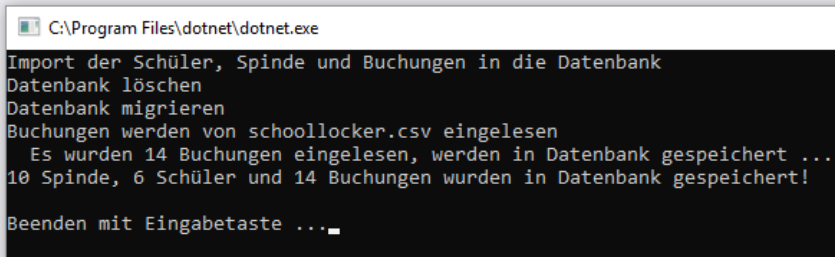


Bereiten Sie die Klasse *Pupil* für die Verwendung in Microsoft Identity vor (von *IdentityUser<int>* ableiten + Initialisierungscode).

Im Corelayer sind die Contracts für die Repositories bedarfsgerecht zu erweitern

## Import / Persistence

Es werden die 10 Spinde mit 14 Buchungen für 6 Schüler aus der Datei schoollocker.csv importiert.



```
C:\Program Files\dotnet\dotnet.exe
Import der Schüler, Spinde und Buchungen in die Datenbank
Datenbank löschen
Datenbank migrieren
Buchungen werden von schoollocker.csv eingelesen
  Es wurden 14 Buchungen eingelesen, werden in Datenbank gespeichert ...
10 Spinde, 6 Schüler und 14 Buchungen wurden in Datenbank gespeichert!
Beenden mit Eingabetaste ...
```

Erstellen Sie mithilfe der NuGet Package Manager Console eine EF Migration und wenden Sie diese in der Datenbank an:

- SchoolLocker.Web als StartupProject
- SchoolLocker.Persistence als DefaultProject
- *Add-Migration InitialMigration*
- *Update-Database*
- Kontrolle, ob DB korrekt angelegt wurde
- Die Daten über die ImportConsole importieren.

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

Die Repositories müssen noch mit den fehlenden Methoden erweitert werden.

## ASP.NET Core Razor Pages (SchoolLocker.Web) - Benutzerverwaltung

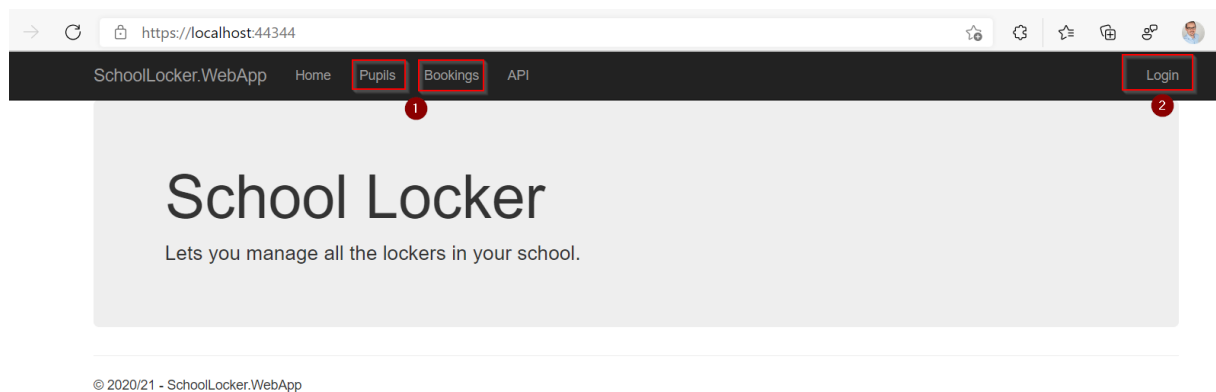
Konfigurieren Sie Microsoft Identity laut Foliensatz.

Prüfen Sie beim Programmstart ob folgender Benutzer / Rolle bereits existiert, ansonsten diese erstellen:

- Rolle: Admin
- Benutzer
  - o Username: [admin@htl.at](mailto:admin@htl.at)
  - o Passwort: Admin12345!
  - o Rolle: Admin

Erweitern Sie die bestehende Implementierung um folgende Use Cases.

### Auth\Login



Beim Klick auf die Pages (1) bzw. direkt auf den Login-Link (2) soll dem Benutzer eine Anmeldemaske präsentiert werden:

Verwenden Sie für den Username / Password die notwendigen Validierungsattribute!

Wenn der Username bzw. das Passwort korrekt sind, so wird das entsprechende Authentication Cookie per Microsoft Identity erzeugt (Aufruf `SignInManager.SignInAsync(..)`) und ein LocalRedirect

durchgeführt:

```
string redirectTo =
    !string.IsNullOrEmpty(Request.Query["ReturnUrl"])
    ? Request.Query["ReturnUrl"]
    : "/Index";

return LocalRedirect(redirectTo);
```

Durch Klick auf „Register“ wird dem Benutzer das Registrierungsformular präsentiert.

## Auth\Register

Neue Benutzer können über dieses Formular erstellt werden und sollen initial KEINE Admin-Rolle besitzen!

Implementieren Sie folgende Validierungsregeln für neue Benutzer (inkl. Passwortregeln in Microsoft Identity):

Register

Firstname  
The Firstname field is required.

Lastname  
The Lastname field is required.

Username  
The Username field is required.

Password  
The Password field is required.

Register

© 2020/21 - SchoolLocker.WebApp

Username

franz.huber

Please enter a valid email address.

## Register

- 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').

Firstname

Franz

Lastname

Huber

Username

franz.huber@htl.at

Password

Register

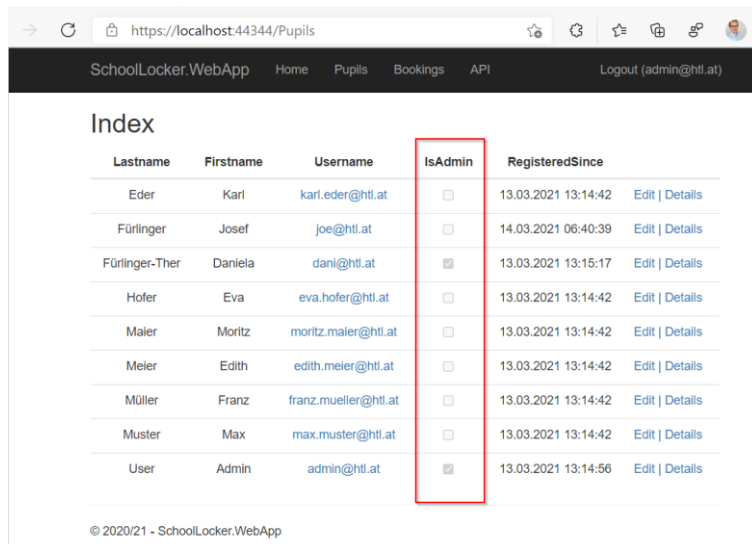
Wenn die Validierungen erfolgreich waren so soll ein neuer *Pupil*-Datensatz per *UserManager* erstellt werden und anschließend ein Redirect auf die *Login*-Seite durchgeführt werden.

## Auth\Logout

Implementieren Sie die notwendige Logik zum Abmelden in der Datei `\Auth\Logout.cshtml.cs`.

## Pupils\Index

**Achtung:** Diese Seite darf nur für angemeldete Benutzer mit der Rolle „Admin“ zugänglich sein.



Index

Lastname	Firstname	Username	IsAdmin	RegisteredSince	
Eder	Karl	karl.eder@htl.at	<input type="checkbox"/>	13.03.2021 13:14:42	<a href="#">Edit</a>   <a href="#">Details</a>
Fürlinger	Josef	joe@htl.at	<input type="checkbox"/>	14.03.2021 06:40:39	<a href="#">Edit</a>   <a href="#">Details</a>
Fürlinger-Ther	Daniela	dani@htl.at	<input checked="" type="checkbox"/>	13.03.2021 13:15:17	<a href="#">Edit</a>   <a href="#">Details</a>
Hofer	Eva	eva.hofer@htl.at	<input type="checkbox"/>	13.03.2021 13:14:42	<a href="#">Edit</a>   <a href="#">Details</a>
Maier	Moritz	moritz.maier@htl.at	<input type="checkbox"/>	13.03.2021 13:14:42	<a href="#">Edit</a>   <a href="#">Details</a>
Meier	Edith	edith.meier@htl.at	<input type="checkbox"/>	13.03.2021 13:14:42	<a href="#">Edit</a>   <a href="#">Details</a>
Müller	Franz	franz.mueller@htl.at	<input type="checkbox"/>	13.03.2021 13:14:42	<a href="#">Edit</a>   <a href="#">Details</a>
Muster	Max	max.muster@htl.at	<input type="checkbox"/>	13.03.2021 13:14:42	<a href="#">Edit</a>   <a href="#">Details</a>
User	Admin	admin@htl.at	<input checked="" type="checkbox"/>	13.03.2021 13:14:56	<a href="#">Edit</a>   <a href="#">Details</a>

© 2020/21 - SchoolLocker.WebApp

Stellen Sie die Liste aller vorhandenen Benutzer (Schüler) im System dar! Sortierung nach dem Nachnamen absteigend.

Sollte der Benutzer die „Admin“ Rolle zugeordnet haben so soll das Flag „IsAdmin“ *true* sein. Verwenden Sie zum Auslesen der Rollen eines Benutzers die Methode *GetRoles(user)* der Klasse *UserManager<Pupil>*.

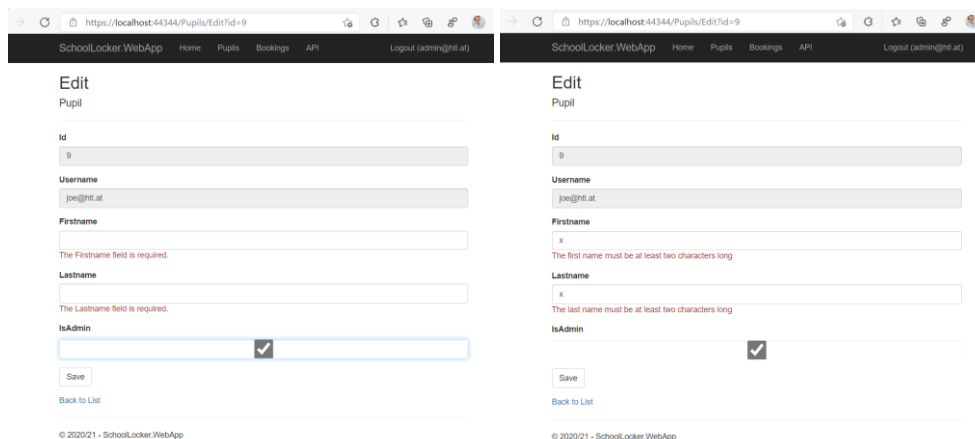
Bei Klick auf *Edit* wird die Maske zum Adaptieren der Benutzerdaten angezeigt.

## Pupils\Edit

Achtung: Diese Seite darf nur für angemeldete Benutzer mit der Rolle „Admin“ zugänglich sein.

Hier können die Daten eines Schülers geändert werden. Wobei die Id und der Username schreibgeschützt zur Info dargestellt werden sollen.

Implementieren Sie folgende Validierungen:



Two screenshots of the 'Edit Pupil' form. The left screenshot shows the form with validation messages: 'The Firstname field is required.' and 'The Lastname field is required.' The right screenshot shows the form with validation messages: 'The first name must be at least two characters long' and 'The last name must be at least two characters long'.

Es soll per Checkbox festgelegt werden können, ob der Benutzer die Admin-Rolle besitzt.

Bei Klick auf „Save“ kommt der Benutzer wieder in die Übersichtsseite der Schüler.

## ASP.NET Core Razor Pages (SchoolLocker.Web) - Spindverwaltung

### Bookings\Index

**Achtung:** Diese Seite darf nur für angemeldete Benutzer zugänglich sein.

Erweitern Sie die bestehende Seite, um zusätzlich die eigenen Buchungen des Benutzers anzuzeigen:

**Lockers Overview**

Number	CountBookings	From	To	
101	3	16.08.2019	02.09.2019	Book
102	2	14.03.2021		Book
103	2	12.11.2019		Book
104	2	29.08.2019		Book
105	2	13.03.2021		Book
106	2	14.03.2021		Book
107	2	13.03.2021		Book
108	1	28.04.2019	27.06.2019	Book
109	2	13.03.2021	15.03.2021	Book
110	1	29.03.2019	25.06.2019	Book

**My Bookins**

LockerNumber	From	To
2	14.03.2021	
6	14.03.2021	

© 2020/21 - SchoolLocker.WebApp

Per Klick auf „Book“ kann eine neue Buchung durchgeführt werden.

### Bookings\Create

**Achtung:** Diese Seite darf nur für angemeldete Benutzer zugänglich sein.

In diesem Formular soll eine Spind-Buchung für den aktuellen Benutzer durchgeführt werden

**Create Booking**

Number: 101

PupilId: Fürlinger Josef

From: 03/14/2021

To: mm/dd/yyyy

Add Booking

[Back to List](#)

© 2020/21 - SchoolLocker.WebApp

Bei Klick auf „Add Booking“ kommt der Benutzer wieder zurück zur Übersichtsseite der Buchungen.

## ASP.NET Web API (SchoolLocker.Web)

Sichern Sie die Web API per JwtBearer ab.

Implementieren Sie dazu folgende Actions bzw. wenden Sie das geforderte Sicherheitsmodell an:

## AuthController

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

Response body

```
{
  "auth_token":
    "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZWl1cy54bWxzb2FwLm9yZy93cy8yMDAxLzA1L21kZW50aXRSL2NsYWltcy91bWVpY2FkZHI1c3MiOiJhZG1pbkR0dG9uYXQ1Ij0JodHRwOi8vc2NoZWl1cy5taHMyb3NvZnQyZ9tL3dzLzIwMDg1MDY1bnRpdHkVY2hwaW1zL3Jvbm91b3JlZG1pb1IsImV4cCI6MTYxNTcwODczNSwiaXNzIjoiaWRR1bnRpdHkuaHRsLmF0IiwiaXVkIjoic2NoZD9sbG9ja2VYLnR0bCShdC9.qaDFXcuUQ7w3bpEOMfXCj1HQ1etLe2R3nNxmNshA59g",
  "name": "User Admin"
}
```

Download

Response headers

```
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
```

Responses

Code

Description

Links

200

Success

No links

401

Unauthorized

No links

Hinweis:

Dekorieren Sie die Action-Methode mit den entsprechenden Response-Type Attributen (200, 401)!

**POST** `/api/Auth/register` Neuen Benutzer registrieren. 

Hiermit können neue Benutzer registriert werden.

Alle Properties (Username, Firstname, Lastname und Password) sind verpflichtend!

Curl

```
curl -X POST "https://localhost:44344/api/Auth/register" -H "accept: */*" -H "Content-Type: application/json" -d '{"username":"marcel.hirscher@t1.at","password":"Start12345","firstname":"Marcel","lastname":"Hirscher"}'
```

Request URL

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

Server response

Code	Details
200	<p>Response headers</p> <pre>date: Sun14 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:

Curl

```
curl -X POST "https://localhost:44344/api/Auth/register" -H "accept: */*" -H "Content-Type: application/json" -d '{"username":"ludwig.hirscher@t1.at","password":"x","firstname":"Ludwig","lastname":"Hirsch"}'
```

Request URL

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

Server response

Code	Details
400	<p>Error:</p> <p>Response body</p> <pre>{   "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')."} </pre> <p>Response headers</p> <pre>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</pre>

Responses

Code	Description	Links
200	Success	No links

### Hinweis:

Dekorieren Sie die Action-Methode mit den entsprechenden Response-Type Attributen (200, 400)!



## BookingsController

**GET** `/api/Bookings/GetOverlappingBookings` Lists all overlapping bookings

Achtung: Diese Action darf nur für angemeldete Benutzer mit der Rolle „Admin“ zugänglich sein.

```
Curl  
curl -X GET "https://localhost:44344/api/bookings/getoverlappingbookings?lockerNumber=110&from=2019-03-28&to=2020-03-28" -H "accept: test/plain" -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpVCy9oLmVudCwibWxzczY9cyBMDA1LzAxMTkzMjZlbnNvYXRldGVzdGFpbGUobDQ1MnVmdltcy9lbWFpbGZhc2hhbmFpdHRhdidR01SiXYVVtjc0ctMOb2NoZWdjaWVmbG9zaWVubHNbcShCdChS9r.kqAAqc65ICG3IGI8qITSE3LoagTQWakf8ckJo-U16A"
```

Request URL

```
https://localhost:44344/api/bookings/getoverlappingbookings?lockerNumber=110&from=2019-03-28&to=2020-03-28
```

Server response

	Code	Details
200		<p>Response body</p> <pre>{   {     "lockerNumber": 110,     "lastName": "Meier",     "firstName": "Edith",     "from": "2019-03-29T00:00:00",     "to": "2019-06-25T00:00:00"   } }</pre>

Download

Hinweis:

Dekorieren Sie die Action-Methode mit den entsprechenden Response-Type Attributen (200, 401, 403, 404)!

**GET** `/api/Bookings/Mine` Lists all bookings of the current user

Achtung: Diese Action darf nur für angemeldete Benutzer zugänglich sein.

Implementieren Sie die Action, sodass alle Bookings des aktuell angemeldeten Benutzers präsentiert werden.

[illegible]

## LockersController

**GET** `/api/Lockers/{lockerNr}` Get Details to a locker.

Achtung: Diese Action darf nur für angemeldete Benutzer zugänglich sein.

```
Curl
curl -X GET "https://localhost:44344/api/Lockers/104" -H "accept: */*" -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpVCV9yJm9kaWw0IjYvc2NoZmlkcyS4bWxzczFwLnBye3cybyMDA1LzAlL2lkZnE5bWZlcnR1LnBvbmQxLCJpc3MlOjIzc2VudG8leS5odG9wYXQ1LCJzdHdnQ10iOiY2bnVzb2xsb2N-XiUsR5LeF0In0.Gqf-AjnXPSR01lyx-6n7AMw3ks5PNC83YQFwHhAs"
```

Request URL

https://localhost:44344/api/Lockers/104

Server response

Code Details

200

Response body

```
{
  "number": 104,
  "cntOfBookings": 2
}
```

Hinweis:

Dekorieren Sie die Action-Methode mit den entsprechenden Response-Type Attributen (200, 401, 404)!

**GET** `/api/Lockers` Lists all existing lockers

**Achtung:** Diese Action darf nur für angemeldete Benutzer zugänglich sein.

```
Curl
curl -X GET "https://localhost:44344/api/Lockers" -H "accept: text/plain" -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyb2UiOiJlbnVudCk9cy44bmZsb2Fudm9yYy9kaW1la111k2dS0xNS12NsYWI1tcy9lbWVpbG9kZD1lc3M0Lj1qb2VAdmRsLnR0IiwiaXNjaXNpdCI6ImNhdCkiLCJpc3M0Lj1pZGVudG9wSSodG9wYXQ1Cl13bW9Q1017yZ2hw2sxb2NrZXIuaHRleF8iIn0.G6qF-AjwXPSR01lyx-6n7AMbu3ksSPNC81YQFuwhAs"
```

Request URL

https://localhost:44344/api/Lockers

Server response

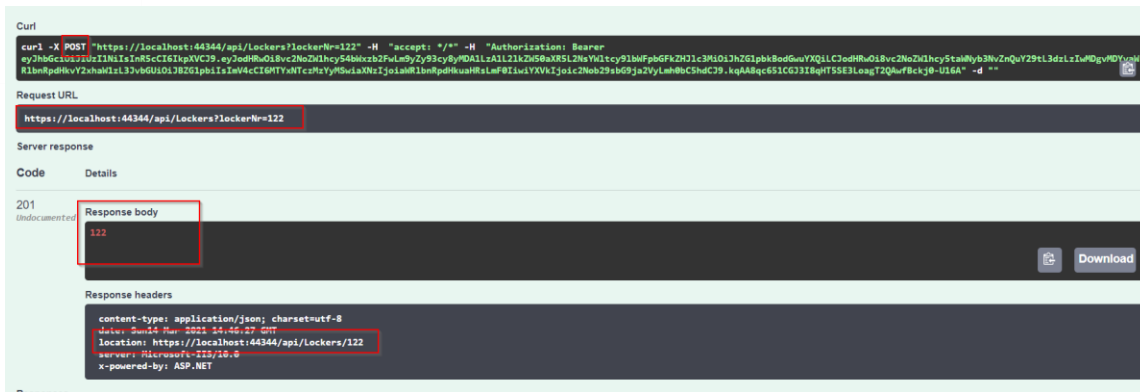
Code	Details
200	<p>Response body</p> <pre>{   101,   102,   103,   104,   105,   106,   107,   108,   109,   110 }</pre>

Hinweis:

Dekorieren Sie die Action-Methode mit den entsprechenden Response-Type Attributen (200, 401)!

**POST** `/api/Lockers` Adds a new locker.

**Achtung:** Diese Action darf nur für angemeldete Benutzer mit der Rolle „Admin“ zugänglich sein.



Hinweis:

```
[HttpPost]
public async Task<ActionResult> AddLocker(int lockerNr)
{
    var lockerInDb = await _unitOfWork.LockerRepository.GetByLockerNrAsync(lockerNr);
    if (lockerInDb != null)
    {
        return Conflict(new { Message = "The locker already exists!" });
    }

    var locker = new Locker { Number = lockerNr };
    await _unitOfWork.LockerRepository.AddAsync(locker);
    await _unitOfWork.SaveChangesAsync();

    return CreatedAtAction(nameof(GetLockerDetails), new { lockerNr = locker.Number }, lockerNr);
}
```

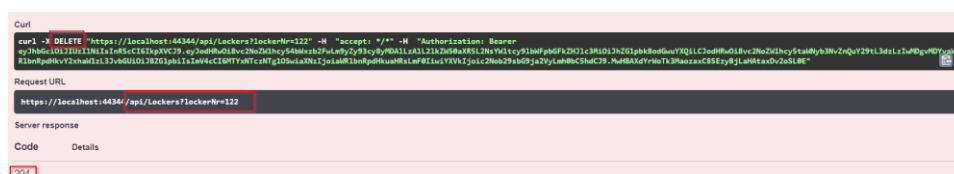
*CreateAtAction*(*..*, *..*, *..*) Parameter:

1. <Name der Action zum Auslesen der Details>
2. <Parameter für die Action>
3. <Daten des erstellten Datensatzes>

Dekorieren Sie die Action-Methode mit den entsprechenden Response-Type Attributen (201, 400, 401, 403, 409)!



**Achtung:** Diese Action darf nur für angemeldete Benutzer mit der Rolle „Admin“ zugänglich sein.



Hinweis:

Dekorieren Sie die Action-Methode mit den entsprechenden Response-Type Attributen (204, 400, 401, 403, 404)!

### Hinweise

- Verwenden Sie zum Sicherstellen der Datenkonsistenz (siehe Screenshots der Website) die notwendigen Validierungsattribute.
- Verwenden Sie dort wo sinnvoll DataTransferObjects
- Achten Sie auf eine korrekte Schichtentrennung (Core, Persistence und Web)
- Verwenden Sie das UnitOfWork-Muster
- Dependency Injection (IoC) verwenden
- Erweitern Sie, wo notwendig die Repositories