Angular – Backend für Rotes Kreuz (localhost) – „REST-Call“

Tools:

- Xampp
- MyPhpAdmin
- Datenbank: schueler
- Tabelle: schueler

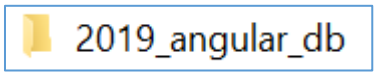
Ausgangspunkt ist folgende Datenbank:

	s_id	vorname	nachname	geschlecht	klasse	gehalt	geburtsdatum	foto
Bearbeiten Kopieren Löschen	1	Ulrich	Plach	m	4CK	3500	2000-09-01	[BLOB - 44,3 KiB]
Bearbeiten Kopieren Löschen	2	Gregor	Hechberger	m	4CK	3750	2001-10-12	[BLOB - 29,4 KiB]
Bearbeiten Kopieren Löschen	3	Carina	Kramberger	w	4CK	4000	2001-06-07	[BLOB - 32,3 KiB]
Bearbeiten Kopieren Löschen	4	Florian	Siegel	m	4CK	3300	2001-02-08	[BLOB - 30,6 KiB]

Unser Interesse gilt nur den ersten 4 Datensätzen:

#	Name	Typ	Kollation	Attribute
1	s_id	int(11)		
2	vorname	varchar(30)	utf8_general_ci	
3	nachname	varchar(50)	utf8_general_ci	
4	geschlecht	char(1)	utf8_general_ci	

1)Erstelle in Xampp/htdocs den Ordner „2019_angular_db“



Darin erstelle eine PHP-Datei: „angular1.php“

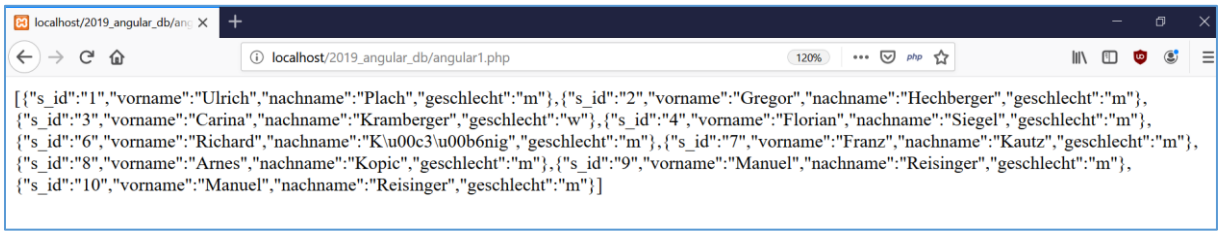
Wichtig ist die Zeile 25, damit das Ergebnis als JSON ausgegeben wird.

```php
<?php

$con = new MySQLi("localhost", "root", "", "schueler");
if ($con->connect_error) {
    echo "Fehler bei der Verbindung: " . mysqli_connect_error();
    exit();
    }
if (!$con->set_charset("utf8")){
    echo "Fehler bei utf8" . $con->error;
    }

$students = [];
$sql = "SELECT * FROM schueler";
$result = mysqli_query($con, $sql);

$nr = 0;  //damit er weiterzählt
while($row = mysqli_fetch_assoc($result))
{
    $students[$nr]['s_id'] = $row['s_id'];
    $students[$nr]['vorname'] = $row['vorname'];
    $students[$nr]['nachname'] = $row['nachname'];
    $students[$nr]['geschlecht'] = $row['geschlecht'];
    $nr++;
}
echo json_encode($students);
?>
```

Code:

```
<?php
$con = new MySQLi("localhost", "root", "", "schueler");
if ($con->connect_error) {
        echo "Fehler bei der Verbindung: " . mysqli_connect_error();
        exit();
   }
if (!$con->set_charset("utf8")){
   echo "Fehler bei utf8" . $con->error;
   }
$students = [];
$sql = "SELECT * FROM schueler";
$result = mysqli_query($con, $sql);

$nr = 0;  //damit er weiterzählt
while($row = mysqli_fetch_assoc($result))
{
   $students[$nr]['s_id'] = $row['s_id'];
   $students[$nr]['vorname'] = $row['vorname'];
   $students[$nr]['nachname'] = $row['nachname'];
   $students[$nr]['geschlecht'] = $row['geschlecht'];
   $nr++;
}
echo json_encode($students);
?>
```

Ergebnis im Browser:



Quelle:
https://www.youtube.com/watch?v=wUydqSvaopU&list=PLGt1lxwGVOI40me32bmhBS2kjkdTJ1-Mn&index=4&t=0s

2)Nachinstallieren von

npm install --save rxjs-compat

```
npm install --save rxjs-compat
```

Dazu öffne VisualStudioCode und das Terminal dort im Projekt. Hier gib den Befehl ein.

RxJS ist eine externe Bibliothek.

3)HttpClient anlegen

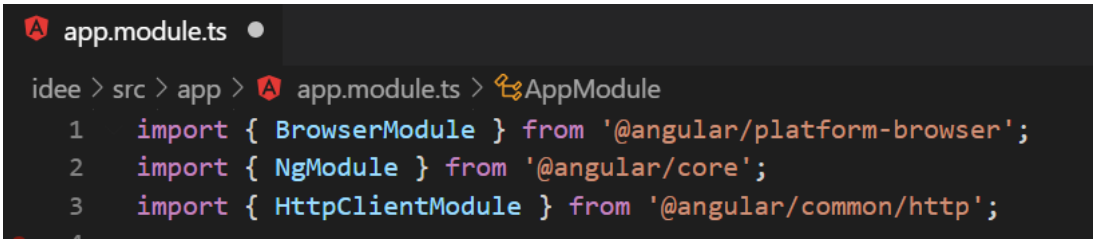
Für den Austausch der Daten über eine serverbasierte Schnittstelle stellt Angular die Klasse „http“ zur Verfügung. Die Anbindung basiert dabei auf der Klasse „Observable“ aus der Bibliothek RxJS.

Dafür gibt es ein eigenes Modul, das HttpClientModule. Dieses muss eingebunden werden in „app.module.ts“.

HttpClientModul einbinden in „app.module.ts“

Öffne die Datei „app.module.ts“ und füge zu den Importen hinzu:

```
import { HttpClientModule } from '@angular/common/http';
```



Dazu muss man es auch noch im „@NgModule“-Array bei den „imports“ einbinden:
Achtung: Beistrich davor (=nach AppRoutingModule)

```
14   imports: [
15     BrowserModule,
16     AppRoutingModule,
17     HttpClientModule
18   ],
```

4) Ein Service erstellen

Diese neue Datei soll als Auffangbecken für den Import der Daten dienen.

Mittels Services kann man Code in eigenständige Klassen auslagern.

Definition: Eine Funktion oder Klasse, welche eine Funktionalität für eine andere Funktion oder Klasse bereitstellt wird als Service bezeichnet.

Das ist dann der Grundstein dafür, die Daten in verschiedenen Komponenten zur Verfügung zu haben.

Um den Code zu vereinfachen, soll die Bereitstellung der Daten ausgelagert werden. Der Vorteil ist, dass Daten aus einer zentralen Quelle geladen werden.

Dadurch ist es auch viel einfacher möglich, später Daten von einer REST-Schnittstelle einzubinden.

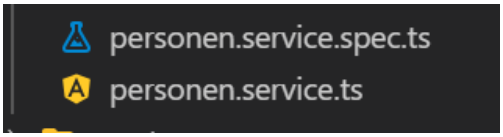
4.1.)Service erstellen

ng g service personen

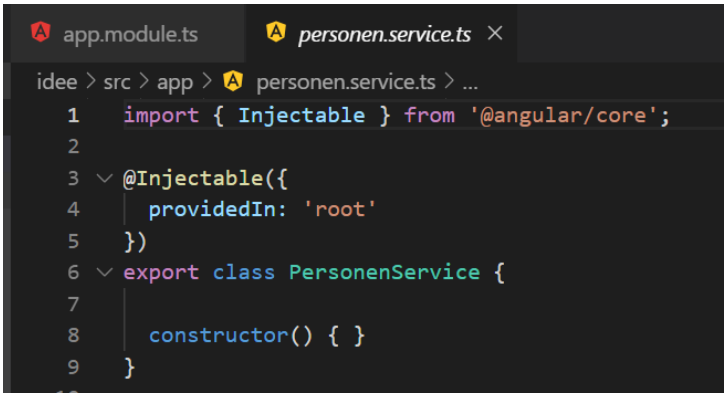
Gib diesen Code mit Hilfe der Angular CLI ein.

```
PS C:\angular_rk\idee> ng g service personen
```

Ergebnis: kommt neu dazu



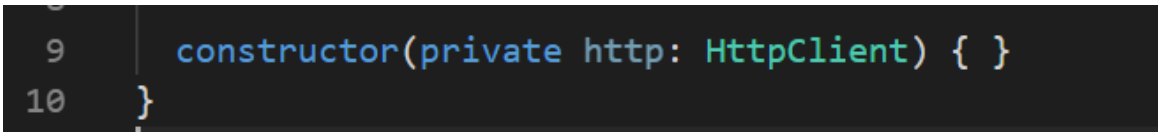
Öffne die „personen.service.ts“:



- Nun legt man den Import an: Zeile 2 und 3 (HttpClient und auch Observable)

- Im Konstruktor injiziere die private Klasse „http“. Danach hat man eine lokale Variable http.
- Der Service „HttpClient“ stellt dann diverse Schnittstellen zur Kommunikation mit einem HTTP-Backend zur Verfügung.

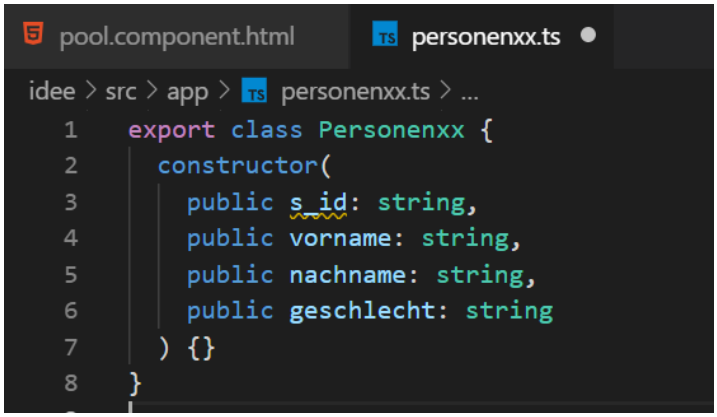
constructor(private http: HttpClient) { }



4.2)Klasse erstellen

- Erstelle eine Klasse „personenxx“, damit man dort bei der folgenden „get“-Methode Daten ablegen kann.

- Öffne die „personenxx.ts“ und definiere die Variablen, die aus der Datenbank kommen werden:



Code:

```
export class Personenxx {
  constructor(
    public s_id: string,
    public vorname: string,
    public nachname: string,
    public geschlecht: string
  ) {}
}
```

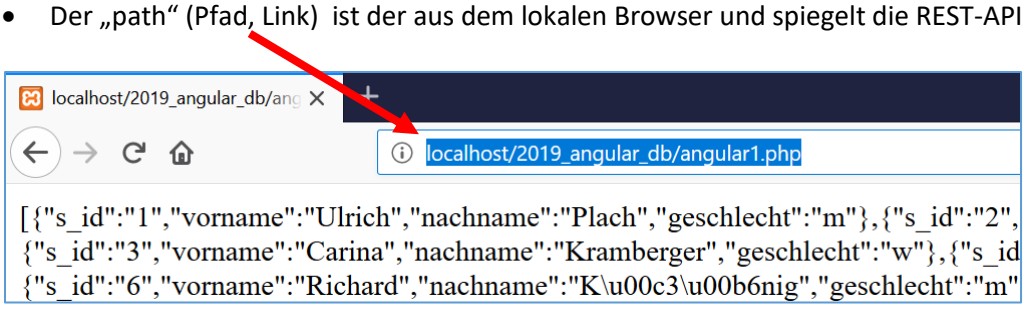
- Diese Klasse muss man auch gleich in der noch geöffneten „personen.service.ts“ importieren: Personenxx….groß geschrieben

personen.service.ts

idee > src > app > personen.service.ts > ...

```
1  import { Injectable } from '@angular/core';
2  import { HttpClient } from '@angular/common/http';
3  import { Personenxx } from './personenxx';
4  import { Observable } from 'rxjs/Observable';
```

- Lege für die spätere „get“-Methode gleich eine private API an. Auf diese wird dann drunter nur mehr verwiesen. Hier wird somit die Adresse des Servers in die Konstante „api“ ausgelagert.
- Der „path“ (Pfad, Link) ist der aus dem lokalen Browser und spiegelt die REST-API wieder.




```
9  export class PersonenService {
10   private api = 'http://localhost/2019_angular_db/angular1.php';
11
```

- Erstelle die „get“ Methode:

```
12   constructor(private http: HttpClient) { }
13
14   loadPersonen(): Observable<Personenxx[]> {
15     return this.http.get<Personenxx[]>(this.api);
16   }
```

Code:

```
loadPersonen(): Observable<Personenxx[]> {
   return this.http.get<Personenxx[]>(this.api);
 }
```

Info:

- Der Name der Methode „loadPersonen“ ist frei wählbar und bechreibt die Schnittstelle klar.
- Diese Methode liefert ein Observabe vom „Personenxx“-Objekt zurück. Diese Methode muss man dann später auch beim „subscribe“ verwenden.
- Das „http“ kommt von der Instanz aus der Zeile 12.

- Das „get“ greift auf die Klasse „Personenxx“ zu und lädt alle im http-Backend gespeicherten „Personen“ und speichert.
- Die spitzen Klammern beschreiben hier einen sogenannten Type-Cast, durch den man dem TypeScript-Compiler mitteilt, dass er das Ergebnis des get-Aufrufes als Liste von „Personenxx“-Objekten behandeln soll.

4.3.)Test ob bis jetzt alles funktioniert:

Nun muss man die Komponente aktualisieren, damit sie das Observable abonnieren, um die Daten zu erhalten. Dazu brauch man ein „subscribe()“.

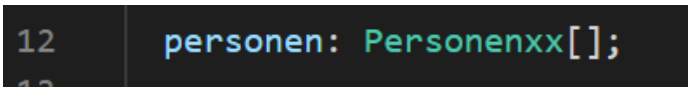
Die Übersicht soll in der Datei „pool.component.html“ angezeigt werden. Dazu muss man mit der „pool.component.ts“ zuerst arbeiten.

Öffne „pool.component.ts“.

Zuerst die beiden wichtigen Importe tätigen:

Info:
Diese Listansicht muss jetzt die Antwort (Request) der von dem Service gestellten Anfrage mittels http an den Server irgendwie bekommen. Dazu dient das „subscribe“.

Das „Personenxx“ in Zeile 12 ist die Klasse, die oben importiert wurde.



Im „constructor“ erstelle

Ändere die „ngOnInit“, um die Servicemethode in der Listenansicht nun zu verwenden.

Hier wird die Servicemethode „loadPersonen()“ aufgerufen, das Observable abonniert und dann die empfangenen Daten in die Eigenschaft „this.personen“ gespeichert:

```
10  export class PoolComponent implements OnInit {
11
12    personen: Personenxx[];
13
14    constructor(private bs: PersonenService) { }
15
16    ngOnInit() {
17      this.bs.loadPersonen().subscribe(res => this.personen = res);
18
19    }
```

Das Linke „res“ ist das Argument zur Funktion, das rechte „this.personen = res“ der Body der Funktion.

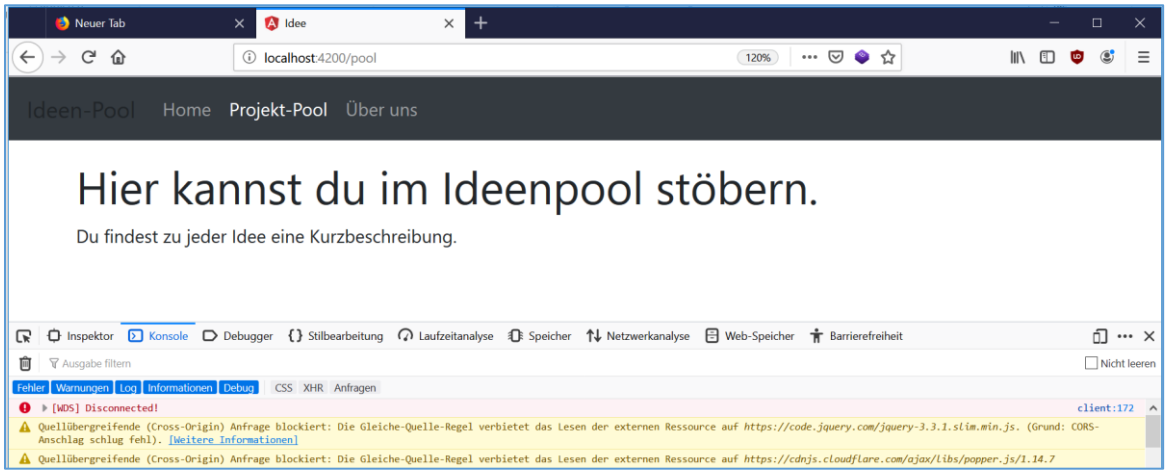
Info: Mithilfe der subscribe-Methode kann man sich bei diesem Observable anmelden, um benachrichtigt zu werden, wenn die Antwort vom Server eintrifft. Hier wird schließlich die Liste der Komponentenklasse (this.personen) auf den neuen Wert gesetzt, sodass Angular nach der Rückkehr des http-Aufrufs automatisch dafür sorgt, dass die Liste in der Oberfläche aktualisiert wird.

Nun kann man den ersten Test durchführen:

Gib ein im Terminal:

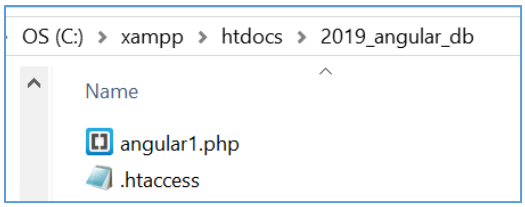
```
PS C:\angular_rk\idee> ng serve --open
```

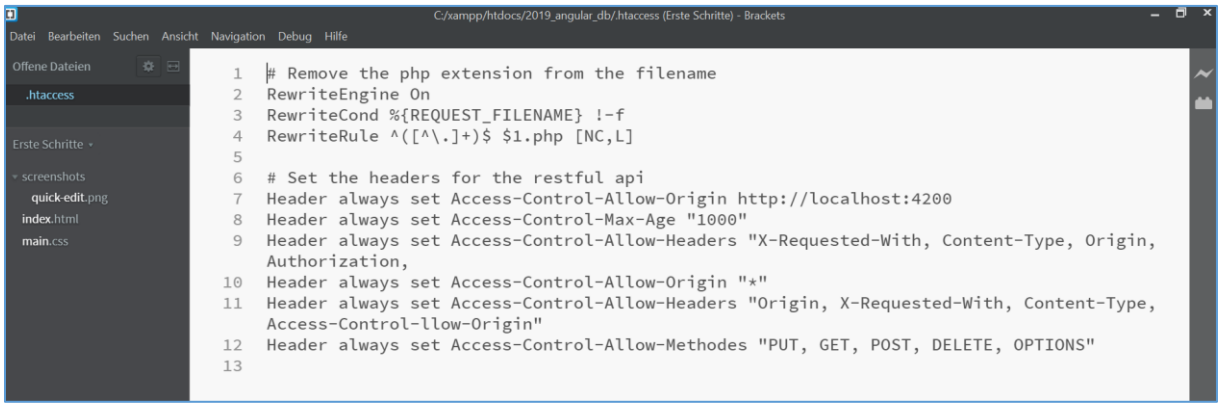
Klicke auf „Projekt-Pool“ und dann auf die Funktionstaste 12, um die Konsole ansehen zu können. Hier sieht man das Problem.



4.4.) htaccess erstellen (API Ersatz)

Man benötigt als API-Ersatz eine „.htaccess“ Datei. Diese muss man neben die „php“-Datei in XAMPP speichern.

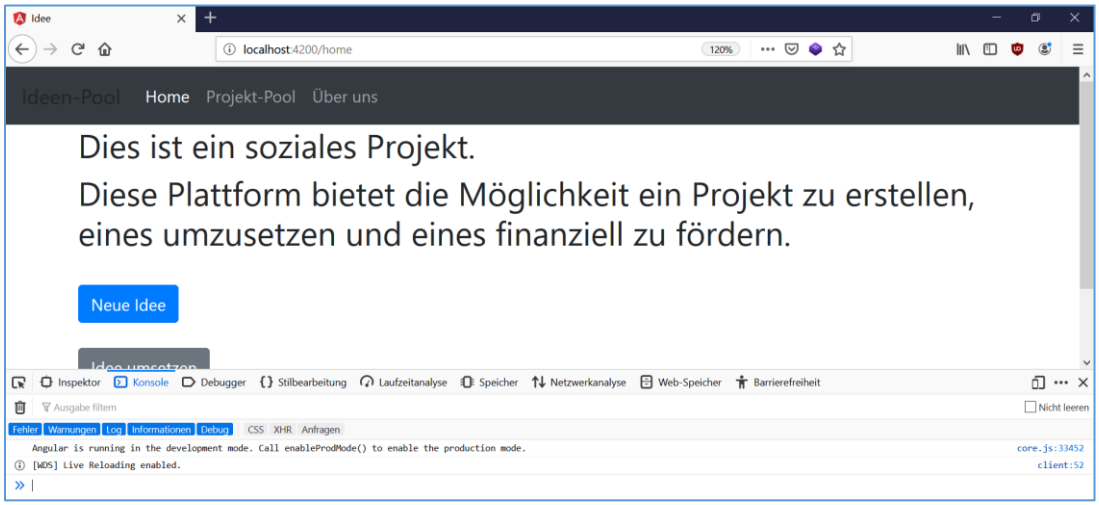




Code:

```
# Remove the php extension from the filename
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^([^\.]+)$ $1.php [NC,L]

# Set the headers for the restful api
Header always set Access-Control-Allow-Origin http://localhost:4200
Header always set Access-Control-Max-Age "1000"
Header always set Access-Control-Allow-Headers "X-Requested-With, Content-Type, Origin,
Authorization,
Header always set Access-Control-Allow-Origin "*"
Header always set Access-Control-Allow-Headers "Origin, X-Requested-With, Content-Type, Access-
Control-llow-Origin"
Header always set Access-Control-Allow-Methodes "PUT, GET, POST, DELETE, OPTIONS"
```

Nun gibt es kein Problem mehr:
letzte Zeile ist nun ok:



4.5.)Anzeige der Tabelle mit ngFor

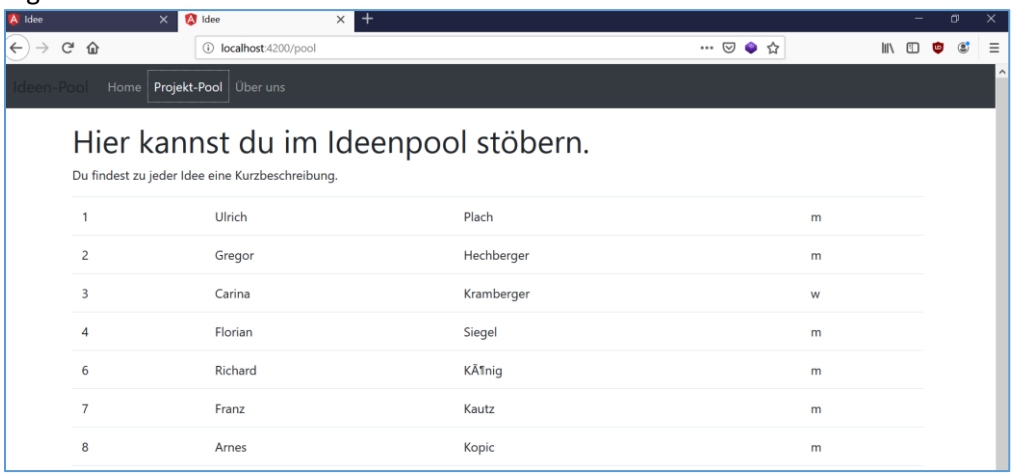
Füge in der „pool.component.html“ die Tabelle mit dem „*ngFor“ ein:

pool.component.html

src > app > pool > pool.component.html > div.container > table.table.table-st

```
1   <div class="container">
2     <br><br><br>
3     <h1>Hier kannst du im Ideenpool stöbern.</h1>
4     <p>Du findest zu jeder Idee eine Kurzbeschreibung.</p>
5
6     <table class="table table-striped">
7       <tr *ngFor="let pers of personen">
8           <td>{{i + 1}}</td>
9           <td>{{pers.s_id}}</td>
10          <td>{{pers.vorname}}</td>
11          <td>{{pers.nachname}}</td>
12          <td>{{pers.geschlecht}}</td>
13      </tr>
14    </table>
15  </div>
```

Ergebnis:



4.6.)Verbesserung durch Anzeige des Reihenfolge der Listeneinträge

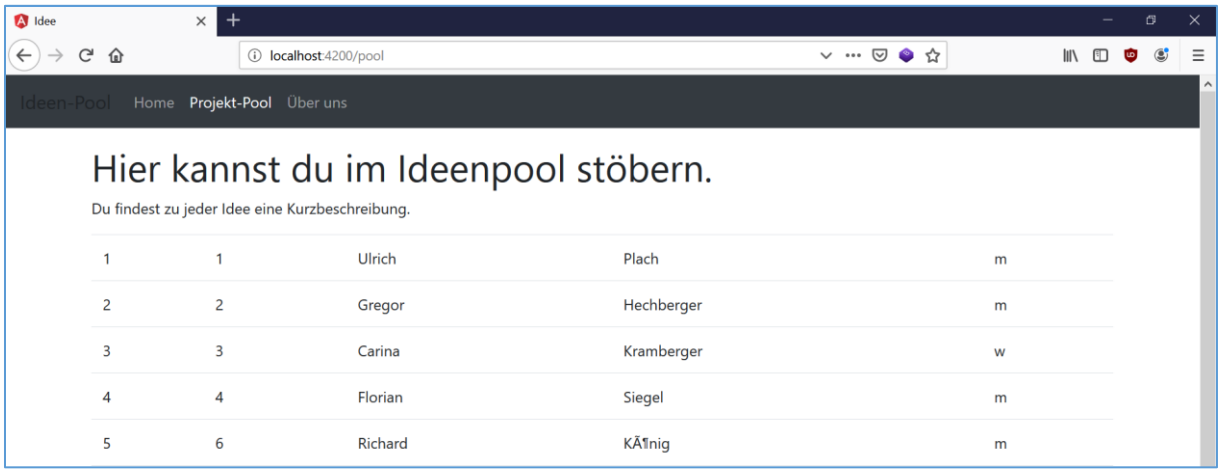
Erstelle eine zweite Variable im Bereich der *ngFor:

let i = index“

```
6   <table class="table">
7     <tr *ngFor="let pers of personen; let i = index">
8         <td>{{i + 1}}</td>
9         <td>{{pers.s_id}}</td>
```

Lass das Ergebnis am Beginn der Zeile ausgeben – siehe Zeile 8

Ergebnis:



Wenn man per Bootstrap die Tabelle verschönern möchte, kann man z.B. „table-bordered“ nutzen.