GET-Request mit Ionic 7 und Datenbankzugriff mittels API

Ionic – Liste aus der Datenbank befüllen

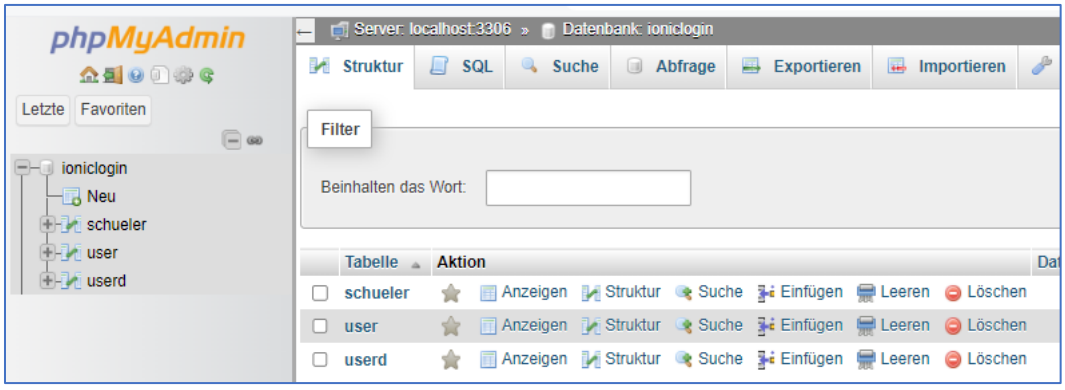
1. Neue Datei „mitarbeiter“ erstellen mit der List-Darstellung
   a. *ngFor
   b. Interpolation {{ }}
2. In mitarbeiter.page.ts mit der passenden Funktion „getMitarbeiter()“ ausformen, die auf das Service (apiService) zugreift
3. apiService mit der passenden GET-Abfrage befüllen
4. Side in Sidemenü aufnehmen
5. API erstellen

Ziel:

- Zugriff mittels Ionic auf Daten per GET-Request über eine API.
- Die API liegt „irgendwo“ außerhalb der App – z.B. auf einem Server (hosttech.at) und ist schon vorbereitet
- Verwendung eines „services“, welches die Verbindung zur API herstellt
- Daten in der Seite „mitarbeiter.page.html“ anzeigen lassen

Testen und ansehen auf: https://ionic.digbizmistelbach.info/login

DB:
„userd“ ist nur zum Üben gewesen, kann man hier vergessen :-)



datum und startzeit kann „null“ sein:

#	Name	Typ	Kollation	Attribute	Null	Standard	Kommentare	Extra
1	iduser	int(11)			Nein	*kein(e)*		AUTO_INCREMENT
2	benutzername	varchar(50)	utf8mb3_general_ci		Nein	*kein(e)*		
3	mail	varchar(50)	utf8mb3_general_ci		Nein	*kein(e)*		
4	passwort	varchar(100)	utf8mb3_general_ci		Nein	*kein(e)*		
5	datum	date			Ja	*NULL*		
6	startzeit	time			Ja	*NULL*		

1.)Erstelle im Ordner „pages“ eine neue Seite „mitarbeiter“

```
kauf3> ionic g page pages/mitarbeiter
```

Öffne diese mitarbeite.page.html und ändere den Titel:

Der Hintergrund des Headers (toolbar) soll ebenfalls grün sein:

```
<ion-header [translucent]="true">
  <ion-toolbar color="success">
    <ion-title>Das sind unsere Mitarbeiter</ion-title>
  </ion-toolbar>
</ion-header>
```

Im Content erfolgt die Darstellung einer Liste. Diese holt sich mit einer FOR-Schleife die Daten und zeigt diese in <ion-label> an.

*ngFor:

- Das ist eine Direktive, um durch die Liste der Manager zu iterieren (durchlaufen)
- Der **vorangestellte Stern „*“** gibt Auskunft darüber, dass es sich beim Inhalt des aktuellen Elements um ein sogenanntes Template handelt.
- ngFor wiederholt das Element und dessen Inhalte für jedes Element.
- „let“ bedeutet in TypeScript soviel wie „Variable“
- dann eine sinnvolle Bezeichnung, hier bietet sich „mitarbeiter“ an. Natürlcih sieht man auch häufig „item“.
- Nach dem „mitarbeiter of“ steht oft die Mehrzahl davon, hier „mitarbeiters“
- In den Label verwendet man die Anzeige mittels INTERPOLATION aus dem Bereich von Angular. Das ist nicht Ionic, sondern ein Code von Angular.
- „mitarbeiter“ in Einzahl, sie wie in der FOR Schleife darüber.
- Nach dem Punkt folgt das Element, wie es in der Datenbank steht angeschrieben, damit es ausgegeben werden kann

Interpolation

Kommt aus der Open-Source-Software ANGULAR, welche hier mit Ionic die Grundlage bildet.

Dabei werden eingesetzt:

- doppelte geschwungene Klammern vorne und hinten
- Name der Variable aus der For-Schleife
- Nach dem Punkt der Name des Elements aus der Datenbank – die in der .ts – Datei mithilfe des Services aus dem API geholt wird.

Es wird die „**Interpolation**“ („{{}}“) verwendet, um Propertys der Komponentenklasse im Template auszugeben. Die Komponenten-Klasse wird später in der passenden „ts“ in „@Component“ erstellt werden.

{{mitarbeiter.mail}} passt nicht in das input, nur in das label

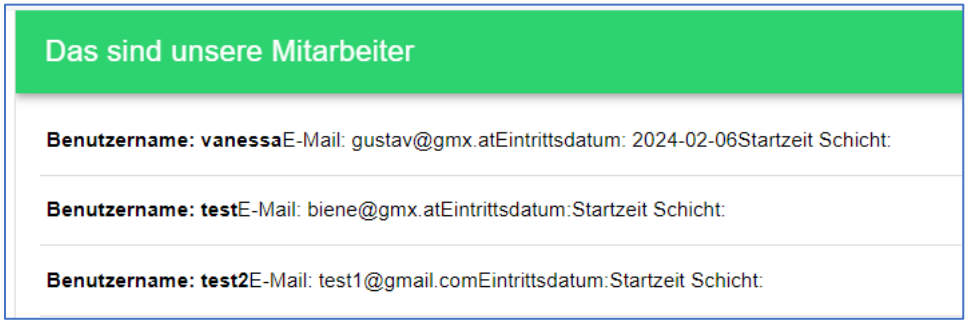
Info:
Das Element aus der Datenbank (nach dem Punkt) wird in der „manager.page.ts“ über die Funktion „getmanager()“ und einem „Observable“ (mit „subscribe“) über das „apiService“ aus der Datenbank geholt. Dazwischen steht hier, zur Absicherung, das API, welches per http-Request im Service angesteuert wird.

```
7   <ion-content [fullscreen]="true">
8     <ion-list>
9       <ion-item lines="insert" *ngFor="let mitarbeiter of mitarbeiters">
10        <ion-label><h5><b>Benutzername:  {{mitarbeiter.benutzername }}</b></h5></ion-label><br>
11        <ion-label><h5>E-Mail:  {{mitarbeiter.mail }}</h5></ion-label><br>
12        <ion-label><h5>Eintrittsdatum:  {{mitarbeiter.datum }}</h5></ion-label><br>
13        <ion-label><h5>Startzeit Schicht:  {{mitarbeiter.startzeit }}</h5></ion-label><br>
14
15      </ion-item>
16    </ion-list>
17
```

- Ansicht verschönern:
  Nutze die „row“ und „grid“:

```
8     <ion-list>
9       <ion-item lines="insert" *ngFor="let mitarbeiter of mitarbeiters">
10        <ion-grid>
11          <ion-row>
12        <ion-label><h5><b>Benutzername:  {{mitarbeiter.benutzername }}</b></h5></ion-label><br>
13        <ion-label><h5>E-Mail:  {{mitarbeiter.mail }}</h5></ion-label><br>
14        <ion-label><h5>Eintrittsdatum:  {{mitarbeiter.datum }}</h5></ion-label><br>
15        <ion-label><h5>Startzeit Schicht:  {{mitarbeiter.startzeit }}</h5></ion-label><br>
16
17      </ion-row>
18      </ion-grid>
19      </ion-item>
```

Ergebnis:

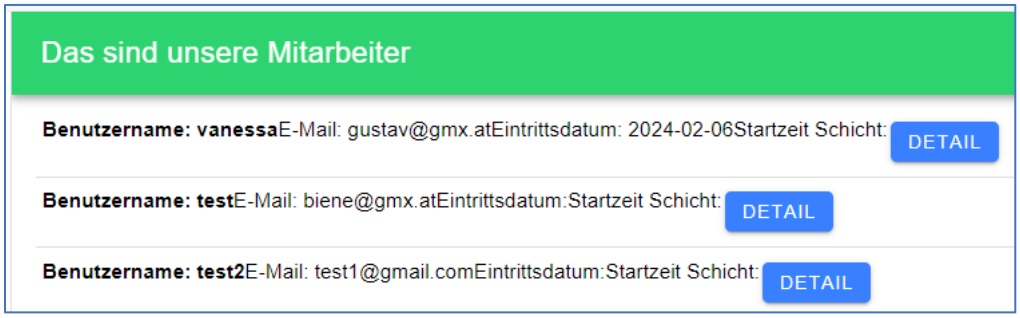


- Verlinkung zur Detailsseite eines jeden Mitarbeiters

Füge einen Button mit der Verlinkung ein:

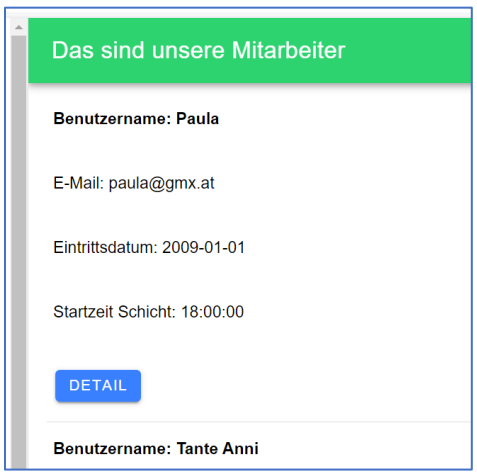
```
      <ion-label><h5>Eintrittsdatum:  {{mitarbeiter.datum }}</h5></ion-label><br>
      <ion-label><h5>Startzeit Schicht:  {{mitarbeiter.startzeit }}</h5></ion-label><br>
      <ion-button color="primary" routerLink="/mitarbdetail/{{mitarbeiter.iduser}}">Detail</ion-button>
    </ion-row>
    </ion-grid>
    </ion-item>
  </ion-list>
```

Ergebnis:



Verbesserung im Layout: die Daten sollen nicht nebeneinander stehen, sondern untereinander. Dazu wird die „row“ entfernt und jede Zeile mit <ion-list> umrhamt.

```
<ion-content [fullscreen]="true">
  <ion-list>
    <ion-item lines="insert" *ngFor="let mitarbeiter of mitarbeiters">
      <ion-grid>

    <ion-list>
      <ion-label><h5><b>Benutzername:  {{mitarbeiter.benutzername }}</b></h5></ion-label>
    </ion-list>
    <ion-list>
      <ion-label><h5>E-Mail:  {{mitarbeiter.mail }}</h5></ion-label><br>
     </ion-list>
     <ion-list>
      <ion-label><h5>Eintrittsdatum:  {{mitarbeiter.datum }}</h5></ion-label><br>
    </ion-list>
    <ion-list>
      <ion-label><h5>Startzeit Schicht:  {{mitarbeiter.startzeit }}</h5></ion-label><br>
    </ion-list>
    <ion-list>
      <ion-button color="primary" routerLink="/mitarbdetail/{{mitarbeiter.iduser}}">Detai
    </ion-list>

    </ion-grid>
    </ion-item>
```



Info bzw. Tipp: bezüglich „Jahr“

Vielleicht ist es einfacher den Namen „item“ zu verwenden, statt Einzahl und Mehrzahl:

```
<ion-list>
  <ion-item button *ngFor="let item of movies">
    <ion-label> {{ item.title }}</ion-label>
```

z.B. Nur das Jahr ausgeben lassen: (typisch in Angular)

```
<ion-list>
  <ion-item button *ngFor="let item of movies">
    <ion-label>
      {{ item.title }}
      <p>{{ item.release_date | date:'y' }}</p>
    </ion-label>
```

Die HTML ist somit fast fertig.

2)mitarbeiter.page.ts vorbereiten

Im Constructor auf die „apiService“ ininzieren und gleichzeitig den Import dafür erstellen lassen (durch langsames Erstellen im constructor und Anklicken in der aufpopenden Hilfe)

```
import { Component, OnInit } from '@angular/core';
import { ApiService } from 'src/app/api.service';

@Component({
  selector: 'app-mitarbeiter',
  templateUrl: './mitarbeiter.page.html',
  styleUrls: ['./mitarbeiter.page.scss'],
})
export class MitarbeiterPage implements OnInit {

  constructor(
    private apiService: ApiService
  ) {  }
```

In der export-Klasse dieser Seite muss man ein Array erstellen, welches jeden Wert annehmen kann (any) und den Namen hat, der als 2. Name in der FOR-Schleife aus der HTML-Datei vorkommt, hier, wie immer, die Mehrzahl „mitarbeiters“. Diese beiden müssen übereinstimmen.

```
export class MitarbeiterPage implements OnInit {
  mitarbeiters: any[] | undefined;

  constructor(
```

Unterhalb der Funktione „ngOnInit()“ wird eine neue Funktion erstellt.

- Diese greift auf des Service zu und
- repliziert auf sich (this) und nutzt die Methode „apiService“ welches „getMitarbeiter“ heißen wird
  Mit „subscribe“ wird die Möglichkeit von „Observables“ genutzt.
- darin wird „mitarbeiters“ (in Mehrzahl, so wie oben in der For-Schleife der HTML)

```
18    ngOnInit() {
19    }
20
21    getMitarbeiter(){
22      this.apiService.getMitarbeiter().subscribe((response: any) => {
23        this.mitarbeiters = response;
24      });
25    };
26
```

```
getMitarbeiter(){
  this.apiService.getMitarbeiter().subscribe((response: any) => {
    this.mitarbeiters = response;
  });
};
```

Diese Funktion oberhalb im „constructor“ aufrufen:

```
12    constructor(
13      private apiService: ApiService
14    ) {
15      this.getMitarbeiter();
16    }
17
18    ngOnInit() {
```

3)apiService - Funktion erstellen

Erstelle ganz unten, noch vor der abschließenden Klammer eine neue Funktion

```
50    getMitarbeiter() {}
```

- Diese Funktion erstellt eine GET-Abfrage in die API, die in diesem Fall schon auf dem Hosttech-Server liegt. Ansonsten kann man das natürlich auch „localhost“ betreiben.
- Diese Abfrage zielt auf die API. Im Detail auf die index.php mit dem Pfad „mitarbeiter“. Dort wird in PHP die Datenbankabfrage erstellt und das Ergebnis als JSON-Datei übergeben.

Funktion für die Abfrage aus der Datenbank

Diese Funktion soll über den direkten Link zum API auf die Datenbank kommen.

- Dabei ist es EGAL wo die API liegt, weil man mit dem Http-Protokoll ja sowieso überall hinkommt.

Erstelle eine Funktion mit einem passenden aussagekräftigen Namen, wie z.B. getMitarbeitereApi() oder getMitarbeiter() – beachte die CamelCase-Schreibweise.

- get – weil es eine get-Abfrage sein wird, im Gegensatz zu post oder update oder delete
- Mitarbeiter – weil es um die Darstellung der Mitarbeiter geht.
- Api wäre gut, damit man es von der später zu erstellenden Funktion „getMitarbeiter“ in der „mitarbeiter.page.ts“ besser unterscheiden kann, weil dort beide unmittelbar aufeinander treffen werden.

Die Methode „http.get()“ fragt die Daten per GET vom Server ab.

- Diese Methode „get“ soll ein Array mit verschiedenen Beratern aus der Datenbank zurückgeben
- In der get-Methode wird die URL eingefügt.
- Die „get-Methode“ gibt eine Observable zurück. Dieses Obsevabel besteht aus einem Array mit allen Beratern aus der Datenbank.

```
50  getMitarbeiter()
51  {
52    return this.http.get('https://ionic.digbizmistelbach.info/api_einkaufsliste/public/index.php/mitarbeiter');
53  }
```

getMitarbeiter()
{
 return
this.http.get('https://ionic.digbizmistelbach.info/api_einkaufsliste/public/index.php/mitarbeiter');
}

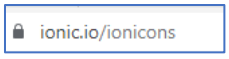
INFO zum API am Ende dieser Datei!

Info:
Es gibt kein Probelm, da wir schon vorher in der Übung den „http Client“ erstellt haben und auch in der „app.modules.ts“ den Import dafür erstellt haben. Ansonsten müsste man das hier noch durchführen.

4)Seite im sidemenü aufnehmen

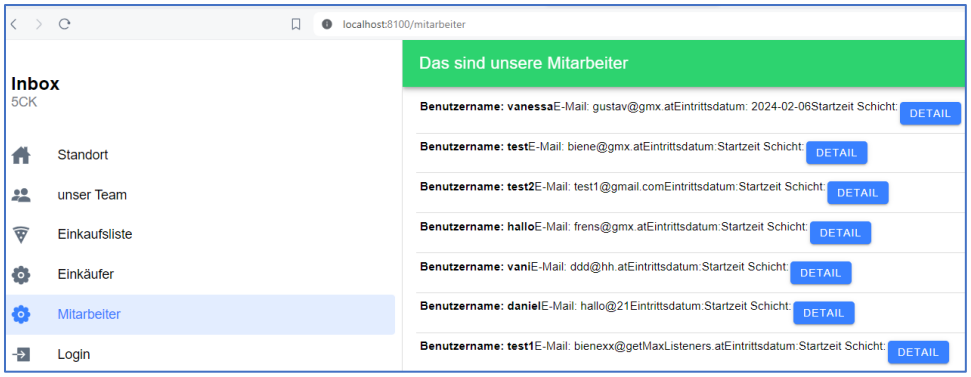
Öffne die Datei „app.component.ts“ und füge einen neuen Eintrag ein.

Als icon suche ein passendens unter: https://ionic.io/ionicons




```
7   export class AppComponent {
8     public appPages = [
9       { title: 'Standort', url: '/standort', icon: 'home' },
10      { title: 'unser Team', url: '/team', icon: 'people' },
11      { title: 'Einkaufsliste', url: '/einkauf', icon: 'pizza' },
12      { title: 'Einkäufer', url: '/buyer', icon: 'flower' },
13      { title: 'Mitarbeiter', url: '/mitarbeiter', icon: 'flower' },
14      { title: 'Login', url: '/login', icon: 'enter' },
```

Ergebnis:



5)API erstellen

API:

```php
$app->get('/mitarbeiter', function( Request $request, Response $response){
  // echo 'Benutzer';

  $sql = "SELECT * FROM user";

  try {
    // Get DB Object
    $db = new db();

    $db = $db->connect();
    $stmt = $db->query( $sql );
    $user = $stmt->fetchAll( PDO::FETCH_OBJ );
    $db = null; // clear db object
    // print out the result as json format
   echo json_encode( $user );
       //print_r($user);


  } catch( PDOException $e ) {

    // show error message as Json format
    echo '{"error": {"msg": ' . $e->getMessage() . '}';
  }

});
```


$app->get('/mitarbeiter', function(Request $request, Response $response){
 // echo 'Benutzer';

 $sql = "SELECT * FROM user";

 try {
  // Get DB Object
  $db = new db();

  $db = $db->connect();
  $stmt = $db->query($sql);
  $user = $stmt->fetchAll(PDO::FETCH_OBJ);
  $db = null; // clear db object
  // print out the result as json format

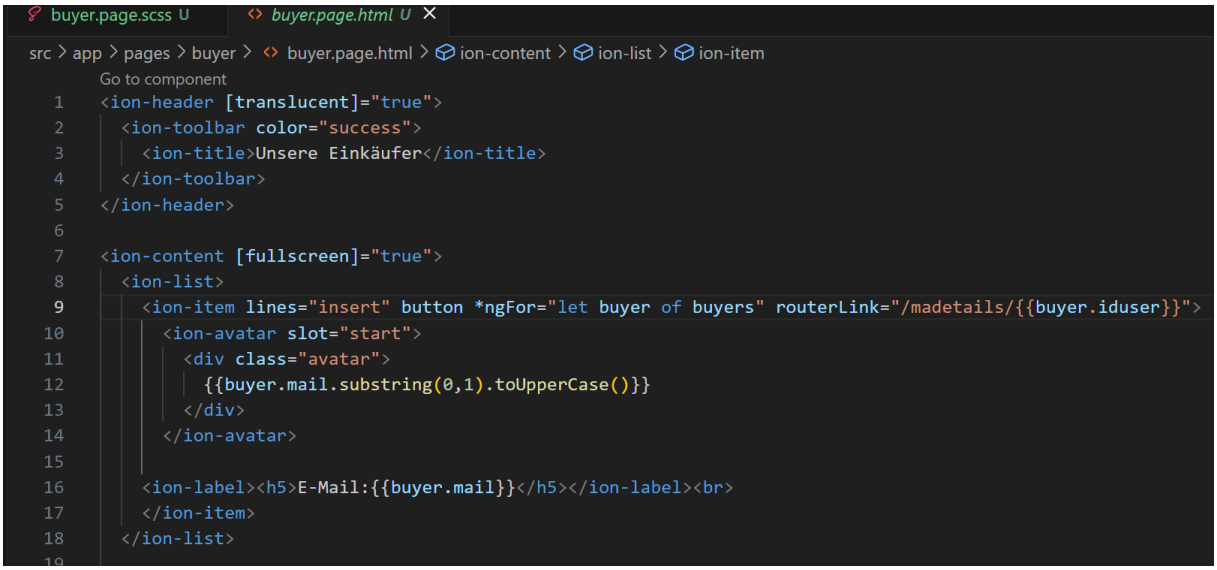
```
   echo json_encode( $user );
                //print_r($user);

  } catch( PDOException $e ) {
    // show error message as Json format
    echo '{"error": {"msg": ' . $e->getMessage() . '}';
  }
});
```

Erweiterung mit Kreis und Anfangsbuchstaben

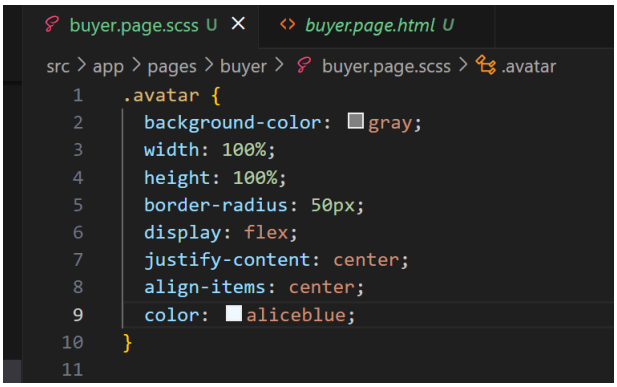
den ersten Buchstaben des Namens in einem Avatar anzeigen lassen:

In der HTML:

```
<ion-avatar slot="start">
 <div class="avatar">
  {{buyer.mail.substring(0,1).toUpperCase()}}
 </div>
</ion-avatar>
```

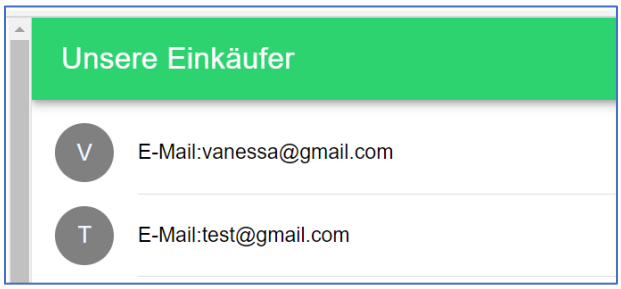


In der CSS:




```
.avatar {
 background-color: gray;
 width: 100%;
 height: 100%;
 border-radius: 50px;
 display: flex;
 justify-content: center;
 align-items: center;
 color: aliceblue;
}
```

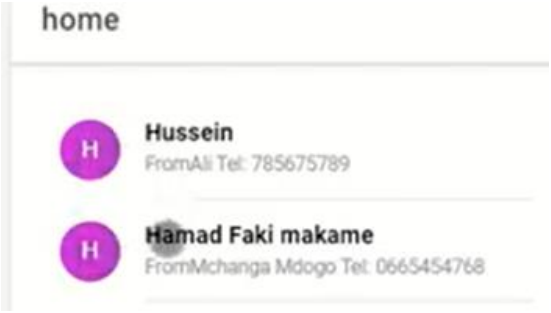
Ergebnis:



https://www.youtube.com/watch?v=PM2uR_eqSvk 16:00

```
<ion-content class="ion-padding">
  <ion-list>
    <ion-item *ngFor="let student of students">
      <ion-avatar slot="start">
        <div class="avatar">
          {{student.name.substring(0, 1).toUpperCase()}}
        </div>
      </ion-avatar>
      <ion-label>
        <h3 class="title">{{student.name}}</h3>
        <p class="subtext">
          {{'From' + student.address + ' Tel: ' + student.phone}}
        </p>
      </ion-label>
    </ion-item>
  </ion-list>
</ion-content>
```

Ergebnis:



Quelle:

Student anlegen, anzeigen, editieren und löschen:

https://www.youtube.com/watch?v=bqiHfIBh8Xk