Ionic – Liste mit Bildern anlegen, Navigation erstellen

Neue Seite erstellen - Team

1. Liste mit Bildern und Text einfügen – Bilder im Ordner „assets“
2. Rechts unten einen Button einfügen (action Button)
3. Neue Seite erstellen – Standort – Bild und Text ZENRTIEREN in der card
4. Navigation – Grundlagen, automatische Weiterleitung und timeout
5. Navigation nach Klick auf einen Button – click-event
6. Einfache Navigation mit „routerLink()“
7. Back-Link erstellen – Zurück zur vorigen Seite
8. Side-Menü anpassen
9. Side Menü bearbeiten
10. Icons ändern

Info: Router

In einer Ionic-Angular-Application“ benutzt man den „router“ von Angular, um zwischen den Seiten zu navigieren. Zwei Routen werden hier z.B. verwendet:

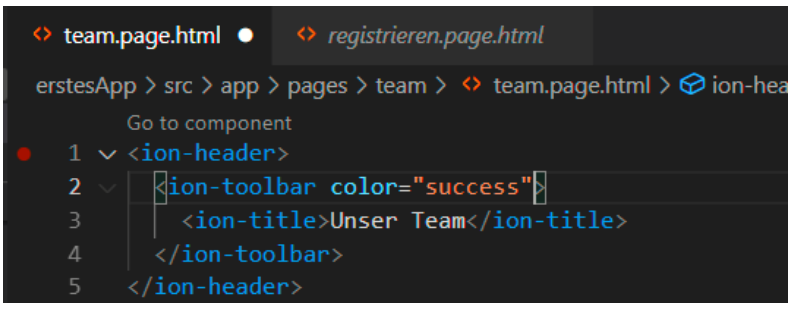
- /mitarbeiter
- /mitarbeiter/:id der Parameter :id dient zur dynamischen Auswahl

1.)Erstelle eine neue Seite „team“

```
PS D:\ionic_start\erstesApp> ionic g page pages/team
```

Öffne diese team.page.html und ändere den Titel auf „Unser Team“

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



2) Liste mit Bildern und Text einfügen

Vorbereitung: Bilder im richtigen Ordner ablegen:

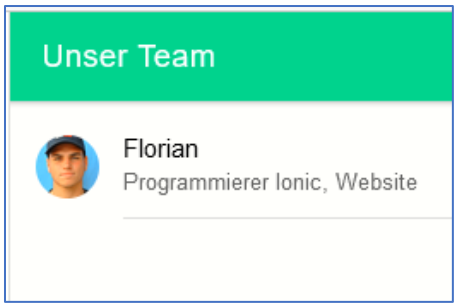
Im Ordner „assets“ einen Ordner „fotos“ anlegen und dort die Bilder deponieren:

Nun soll im Content eine „ion-list“ erstellt werden.

- Die einzelnen Einträge werden wieder mit <ion-item> benutzt
- Das Bild wird mit <ion-avatar> benutzt
- Da das Bild links steht, verwende slot=“start“
- Der Pfad richtet sich auf die assets im Ordner „src“
- Rechts daneben steht mit <ion-label> der Text

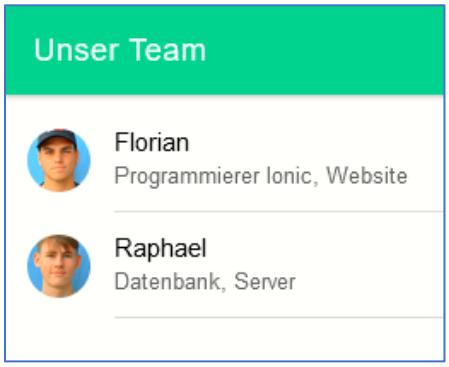
```
7   <ion-content>
8     <ion-list>
9       <ion-item>
10        <ion-avatar slot="start">
11          <img src="../../../assets/fotos/florian.JPG">
12        </ion-avatar>
13        <ion-label>
14          <h2>Florian</h2>
15          <p>Programmierer Ionic, Website</p>
16        </ion-label>
17      </ion-item>
18    </ion-list>
19  </ion-content>
```

```
<ion-list>
  <ion-item>
   <ion-avatar slot="start">
    <img src="../../../assets/fotos/florian.JPG">
   </ion-avatar>
   <ion-label>
    <h2>Florian</h2>
    <p>Programmierer Ionic, Website</p>
   </ion-label>
  </ion-item>
 </ion-list>
```



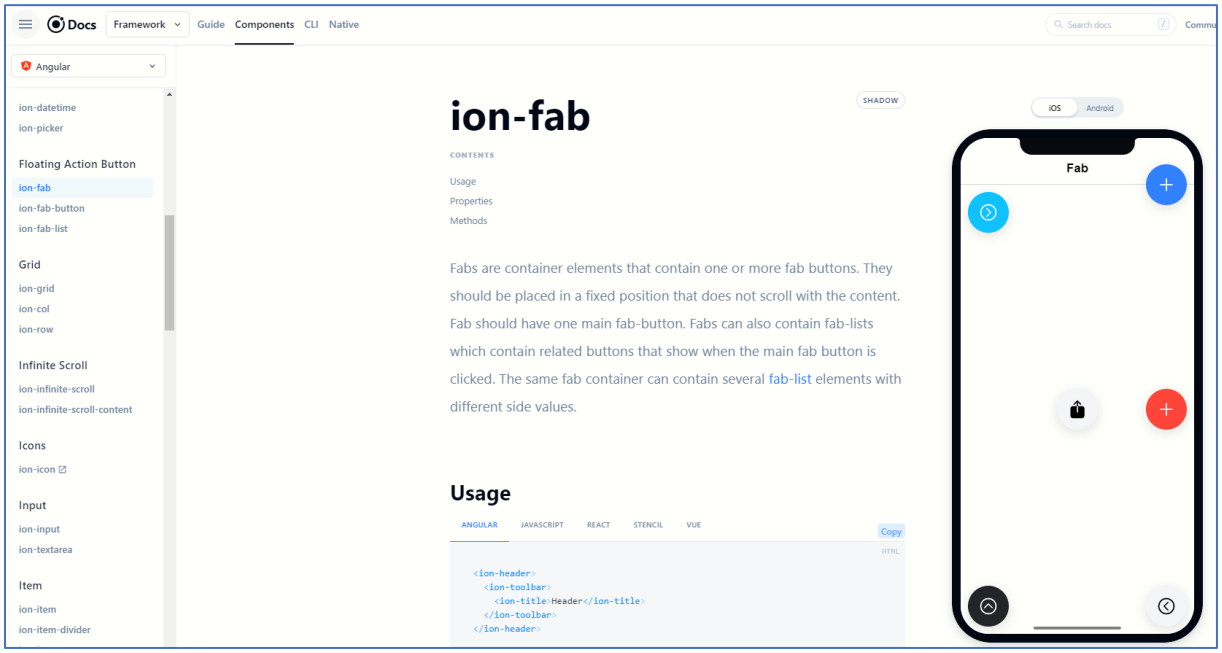
Dann den zweiten Eintrag.

Kopiere den „item“ und ändere den Namen und das Foto.



3.)action Button rechts unten

Die Info findet man unter „floating action button“ in den Components.



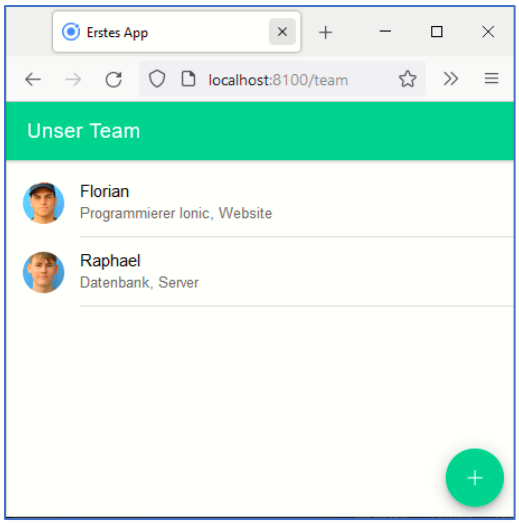
Dieser soll unterhalb der Liste erstellt werden.

- Vertical – bottom
- Horizontal rechts
- Fixiert dort unten rechts

- Farbe: grün
- Inhalt des Buttons ist ein „add“

```
27      </ion-list>
28      <ion-fab vertical="bottom" horizontal="end" slot="fixed">
29        <ion-fab-button color="success">
30          <ion-icon name="add"></ion-icon>
31        </ion-fab-button>
32      </ion-fab>
33    </ion-content>
```

```
<ion-fab vertical="bottom" horizontal="end" slot="fixed">
  <ion-fab-button color="success">
    <ion-icon name="add"></ion-icon>
  </ion-fab-button>
</ion-fab>
```



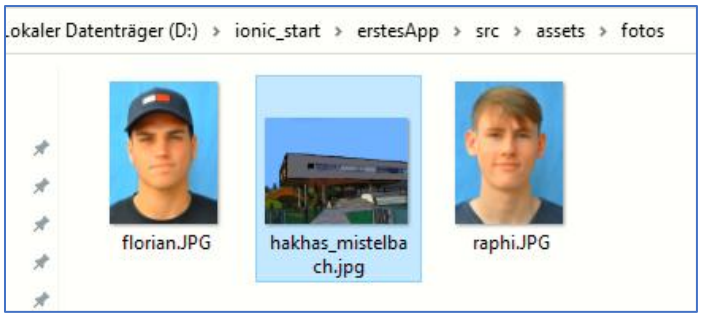
4.)Erstelle eine neue Seite „standort“ und Bild und Text zentrieren

Diese soll eine Bild enthalten, welches von links nach rechts geht und darunter in einer Card die genaue Adresse.

```
PS D:\ionic_start\erstesApp> ionic g page pages/standort
```

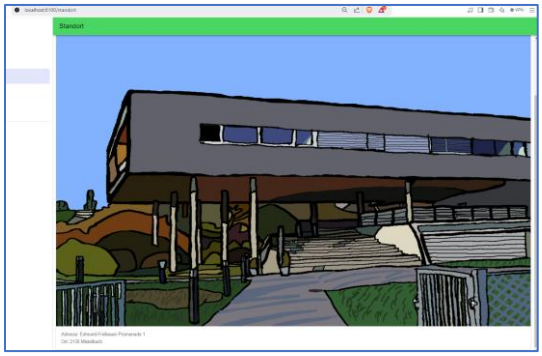
Der Haeder oben soll wieder grün sein und der Titel: Standort

Darunter füge das Bild der Hak ein.



Darunter in einer eigenen <ion-card> der genaue Text für den Standort.

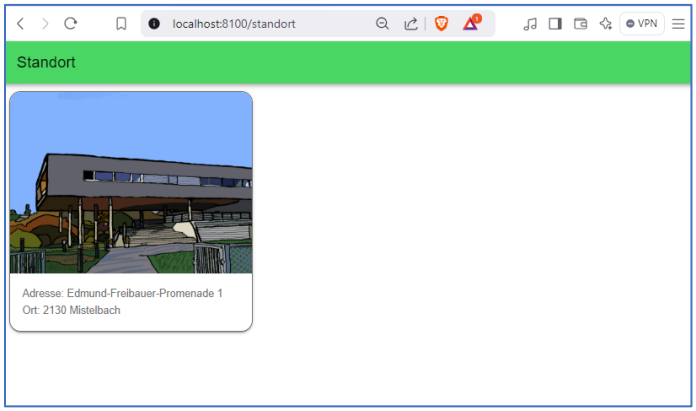
```
<ion-content [fullscreen]="true">
  <ion-card>
    <img src="../../../assets/img/hakhas-mistelbach.jpg">
    <ion-card-content>
      <p>Adresse: Edmund-Freibauer-Promenade 1</p>
      <p>Ort: 2130 Mistelbach</p>
    </ion-card-content>
  </ion-card>
</ion-content>
```



Das Bild ist viel zu groß.

Nun sollen Klassen benutzt werden, die dann in der SCSS-Datei verschönert werden sollen.

Das Bild soll ca. 35% in der Breite einnehmen:



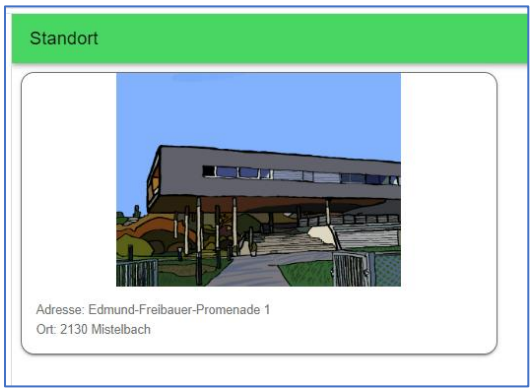

```
ion-card {
  width: 35%;
  height:auto;
  border-radius: 15px;
  // box-shadow: none;
  border: 1px solid;

}
```

```
ion-card {
  width: 35%;
  height:auto;
  border-radius: 15px;
  // box-shadow: none;
```

```
  border: 1px solid;
}
```

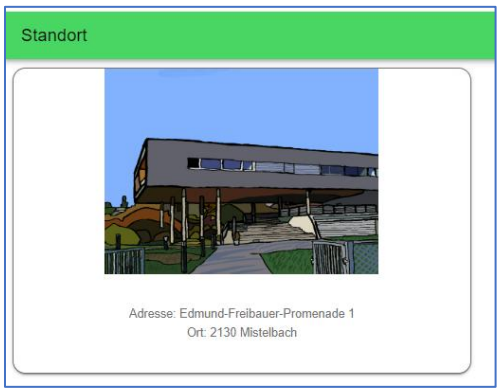
Bild zentrieren




```
9  ion-thumbnail {
10   width: 60%;
11   height: 60%;
12   margin:auto;  //kein Leerzeichen vor auto
13 }
```

```
ion-thumbnail {
  width: 60%;
  height: 60%;
  margin:auto;  //kein Leerzeichen vor auto
}
```

Text zentrieren



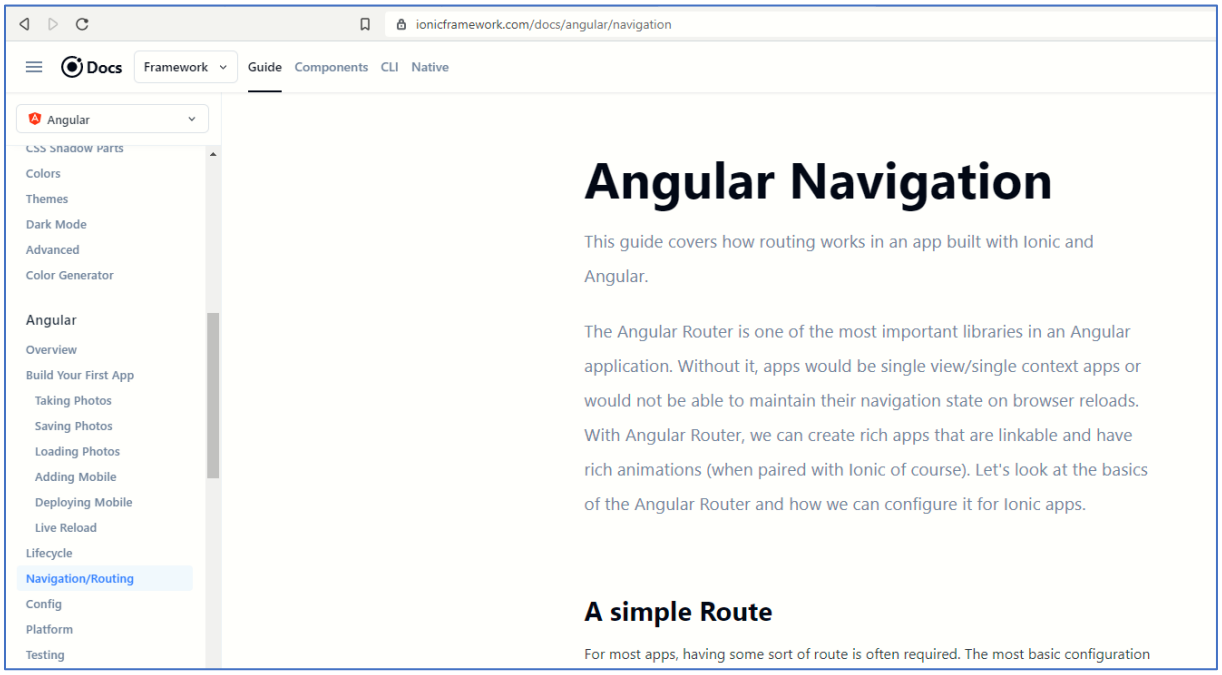

```
14
15  ion-card-content {
16    margin: 20px;
17    text-align: center;
18  }
19
```

```
ion-card-content {
  margin: 20px;
  text-align: center;
}
```

5.)automatische Weiterleitung zur Startseite

Hier erfolgt kein Klicken auf einen Button. Sondern es soll nach einer gewissen Zeit in der das Loadersymbol sich dreht, auf eine Seite weitergeleitet werden.

Hintergrund-Info findet man auf der Website von https://ionicframework.com/
Die Navigation ist in der Dokumentation beschrieben. Dafür muss man statt den „Components“ den „Guide“ anklicken und dann bei Angular, welches wir im Hintergrund ja verwenden, auf „Navigation/Routing“ klicken:



Scrollt man etwas nach unten, findet man wie man von Seite zu Seite navigiert:

```
})
export class LoginComponent {

  constructor(private router: Router){}

  navigate(){
    this.router.navigate(['/detail'])
  }
}
```

Erklärung:
Hierfür benötigt man den Router von Angular. Dieser muss importiert werden.
Im Constructor steht, dass der Router genutzt wird. Das ganz automatisch. Dieser nutzt dann die Navigationsfunktion mit navigate() zu dieser angegebenen Seite.

Die Funktion „navigate“ beinhaltet als Parameter ein „array“ (viereckige Klammern) mit dem String „detail“.

Erklärung:

- in der TS-Seite, in der die Navigation stattfindet (in der HTML) wird in den Funktionen „constructor()“ und „ ngOnInit()“ der Router und die Navigtion eingebaut.

Daher öffne die Seite „loader.page.ts“

```
TS loader.page.ts 1 ×
erstesApp > src > app > pages > loader > TS loader.page.ts > ...
1   import { Component, OnInit } from '@angular/core';
2
3   @Component({
4     selector: 'app-loader',
5     templateUrl: './loader.page.html',
6     styleUrls: ['./loader.page.scss'],
7   })
8   export class LoaderPage implements OnInit {
9
10    constructor() { }
11
12    ngOnInit() {
13    }
14
15  }
16
```

Hier muss man nun die Funktion „constructor()“ und die Funktion „ngOnInit()“ bearbeiten.

IMPORT:

Da man den **Angular-Router** dazu benötig, muss man zuerst im „constructor“ diesen Router vorstellen. Dabei sollte automatisch, wenn man es richtig und geschickt anstellt, der Router importiert werden.

Gibt man beim constructor „privat router: Rou“ ein, noch ohne es vollständig aus zu schreiben, wird der „Router“ vorgeschlagen. In einer Liste, bestätige mit der Maus den ersten Eintrag.

Damit wird automatisch der „import“ dieses „Router“ in die Zeile 2 vorgenommen, was man sonst händisch schreiben müsste. Dies isgt unbedingt nötig, da damit dieser Router überhaupt erst funktioniert, da er importiert wird.

```
10
11    constructor(private router: Rout) { }
12
13    ngOnInit() {                    Router                    (alias) class Routerimport Router
14    }                               RouteConfigLoadEnd        erstesApp/node_modules/@angular/rou…
15                                    RouteConfigLoadStart      erstesApp/node_modules/@angular/r…
                                      RouteReuseStrategy        erstesApp/node_modules/@angular/rou…
```

Variante 2 für IMPORT:
wenn im constuctor alles richtig und vollständig geschrieben wurde, kann man sich an das Ende stellen (vor der schließenden Klammer) und drücke

- STRG
- Und Leertaste

Dann kann man den Vorschlag daneben leicht übernehmen.

```
1  import { Component, OnInit } from '@angular/core';
2  import { Router } from '@angular/router';
3
4  @Component({
5    selector: 'app-loader',
6    templateUrl: './loader.page.html',
7    styleUrls: ['./loader.page.scss'],
8  })
9  export class LoaderPage implements OnInit {
10
11   constructor(private router: Router) { }
```

In der darunter liegenden Funktion „ngOnInit()“ füge nun die Umleitung (Navigation) auf die Seite „team“ ein:

```
13   ngOnInit() {
14       this.router.navigate(['team']);
15   }
```

Strichpunkt nicht vergessen.

Ergebnis:

Zuerst wird ganz kurz der Loader geöffnet, dann erfolgt schon die Navigation automatisch zur Seite „team“.

Um den Loader etwas länger sehen zu können, erstelle einen „Timeout“ mit 2 Sekunden.

setTimeout(() => {}, 2000)

wobei man in die geschwungen Klammern die naviagtion reinsetzen muss:

```
13   ngOnInit() {
14     setTimeout(() => {
15     this.router.navigate(['team']);
16     }, 2000);
17   }
```

6)Navigation nach Klicken eines Buttons

Hier erfolgt die Weiterleitung nachdem ein Button geklickt wurde.

Es erfolgt der Prozess genauso wie oben beschrieben, nur dass die Funktion „navigate()“ **nicht in der „ngOnInit()“ erfolgt**, sondern eine **eigene (der Name ist beliebig) Funktion** erstellt werden muss, die diese „navigate()“ beinhalten muss.

Der Action-Button auf der Seite „team“ soll zur seite „loader“ navigieren. Auch wenn das nun nicht gerade sehr sinnvoll ist, soll es zumindest diese Button-Navigation zeigen.

HTML:
Erstelle in der „**team.page.html**“ den Button-Link mit der click-Funktion und der noch zu erstellenden Funktion „zumLoader()“.

```
30    <ion-fab-button color="success">
31      <ion-icon name="add" (click)="zumLoader()"></ion-icon>
32    </ion-fab-button>
```

Da es diese neue Funktion ja noch nicht gibt, wird diese unterwellt, was bedeutet, dass hier ein Fehler vorliegt. Wir wissen aber warum das so ist.

TS:
Daher soll nun in der dazupassenden **„ts“-Datei diese Funktion erstellt** werden. Öffne dafür die „team.page.ts“.

Da man den **Angular-Router** dazu benötig, muss man zuerst im „constructor“ diesen Router vorstellen. Dabei sollte automatisch, wenn man es richtig und geschickt anstellt, der Router importiert werden.

- Gibt man beim constructor „privat router: Rou“ ein, noch ohne es vollständig aus zu schreiben, wird der „Router“ vorgeschlagen. In einer Liste, bestätige mit der Maus den ersten Eintrag.

```
10
11  constructor(private router: Rout) { }
12
13  ngOnInit() {                    Router                (alias) class Routerimport Router
14  }                               RouteConfigLoadEnd    erstesApp/node_modules/@angular/rou…
15                                  RouteConfigLoadStart  erstesApp/node_modules/@angular/r…
                                    RouteReuseStrategy    erstesApp/node_modules/@angular/rou…
```

- Damit wird automatisch der „import“ dieses „Router“ in die Zeile 2 vorgenommen, was man sonst händisch schreiben müsste. Dies isgt unbedingt nötig, da damit dieser Router überhaupt erst funktioniert, da er importiert wird.

```
TS team.page.ts 1 ×
erstesApp > src > app > pages > team > TS team.page.ts > TeamPage
1   import { Component, OnInit } from '@angular/core';
2   import { Router } from '@angular/router';
```

Dann erstelle die neue Funktion, die schon oben beim Button mit dem Namen „zumLoader()“ vorgestellt wurde. Diese soll nun diese Weiterleitung zur Seite „loader“ durchführen.

```
11    constructor(private router: Router) { }
12
13    ngOnInit() {
14    }
15
16  zumLoader(){
17    this.router.navigate(['loader']);
18  }
19
20  }
```

7)einfachste Navigation mit „routerLink“

Das ist die einfachste Variante für eine Weiterleitung auf eine andere Seite.

```
routerLink="/standort"
```

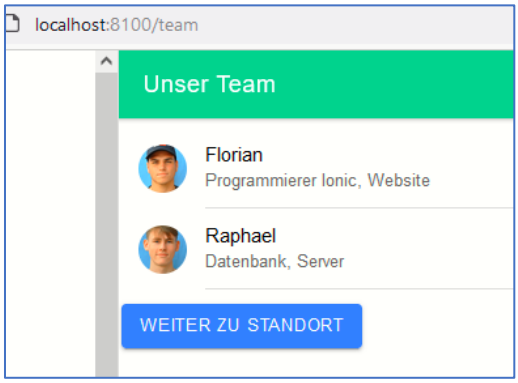
Hierfür benötig man keine weiteren Vorbereitungen oder Funktionen im Hintergrund.

In einen Button schreibe

- routerLink gefolgt von einem Istgleichzeichen
- Anführungszeichen und einem Slash
- dann der Name der Seite, auf die geleitet werden soll
- Anführungszeichen zu

Übung:
Erstelle in der Seite „team“ einen Button, der auf die Seite „standort“ weiterleiten soll. Der einfache Button soll nach der Liste mit den beiden Bildern sich befinden aber noch vor dem „ion-fab“:

```
27    </ion-list>
28    <ion-button>Weiter zu Standort</ion-button>
29    <ion-fab vertical="bottom" horizontal="end" slot="fixed">
30      <ion-fab-button color="success">
```



Dieser Button soll nun diese Weiterleitung auf erhalten.

```
28    <ion-button routerLink="/standort">Weiter zu Standort</ion-button>
```

8)Back-Link erstellen – Zurück zur vorigen Seite

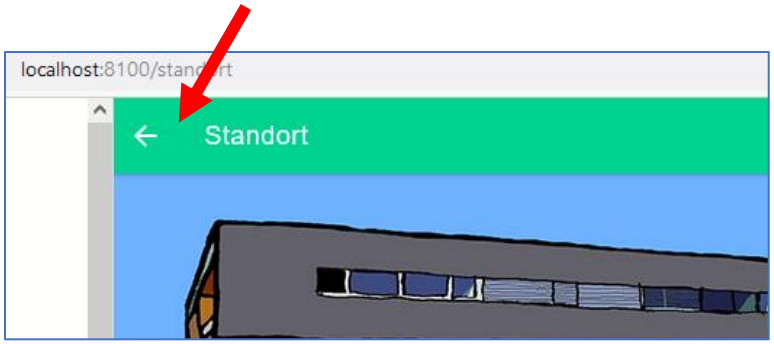
In der Toolbar des Headers soll ganz oben ein „Zurück-Link“ erscheinen. Der Link funktioniert nur dann, wenn man von einer anderen Seite kommt, weil dann ja eine „Historie“ vorhanden ist. Sie funktioniert natürlich nicht, wenn man diese direkt im Browser aufruft.

Dafür erzeuge in der Seite „standort“ diesen back-link.

Öffne die Seite „standort.page.html“ und füge in der „toolbar“ den „ion-back-button“ ein. Damit er linksbündig steht, muss man den slot auf „start“ setzen.

```
2   <ion-toolbar color="success">
3     <ion-back-button slot="start"></ion-back-button>
4     <ion-title>Standort</ion-title>
```

<ion-back-button slot="start"></ion-back-button>



9) Side Menü bearbeiten

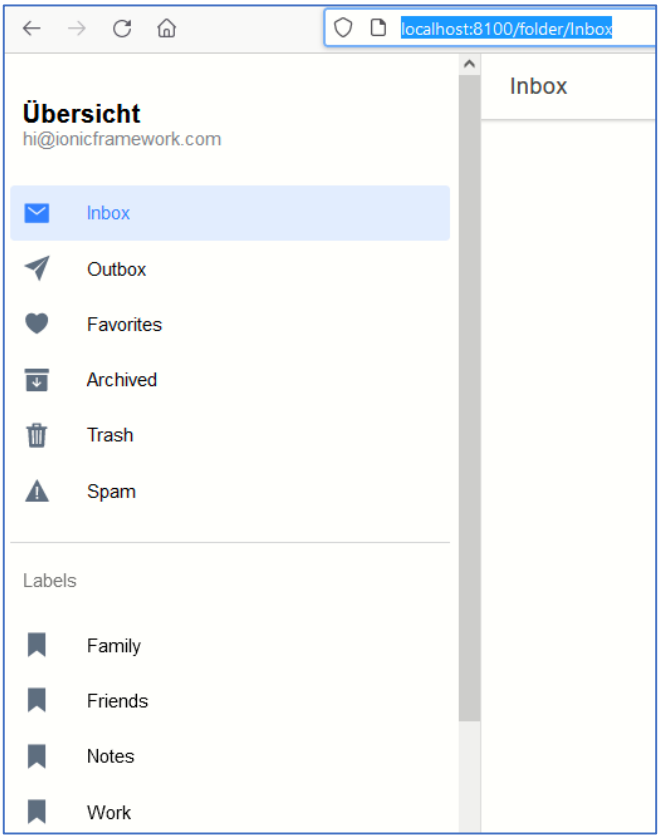
Da wir am Beginn ein automatisches Side-Menü erstellen haben lassen, sollte wir uns das zu Nutze machen und die beiden neuen Seiten hier einarbeiten.

Die alten Vorschläge könnten wir entfernen, zumindest zum größten Teil.

Betrachte dieses Side-Menü durch Eingabe der URL

http://localhost:8100/folder/Inbox

Ergebnis:



Die oberen Eintragungen von „Inbox“ bis „Spam“ werden wir durch die beiden neuen Seiten „Team“ und „Standort“ ersetzen.

Die Labels unten lassen wir noch.

Öffne dafür die Datei:

app.component.ts

```
TS app.component.ts 1 ×
erstesApp > src > app > TS app.component.ts > ...
1   import { Component } from '@angular/core';
2   @Component({
3     selector: 'app-root',
4     templateUrl: 'app.component.html',
5     styleUrls: ['app.component.scss'],
6   })
7   export class AppComponent {
8     public appPages = [
9       { title: 'Inbox', url: '/folder/Inbox', icon: 'mail' },
10      { title: 'Outbox', url: '/folder/Outbox', icon: 'paper-plane' },
11      { title: 'Favorites', url: '/folder/Favorites', icon: 'heart' },
12      { title: 'Archived', url: '/folder/Archived', icon: 'archive' },
13      { title: 'Trash', url: '/folder/Trash', icon: 'trash' },
14      { title: 'Spam', url: '/folder/Spam', icon: 'warning' },
15    ];
16    public labels = ['Family', 'Friends', 'Notes', 'Work', 'Travel', 'Reminders'];
17    constructor() {}
18  }
```

Hier ändere die ersten beiden folgendermaßen, wobei die anderen 4 Elemente darunter gelöscht werden können:

```
})
export class AppComponent {
  public appPages = [
    { title: 'Team', url: '/team', icon: 'mail' },
    { title: 'Standort', url: '/standort', icon: 'paper-plane' },
  ];
```

Beachte:

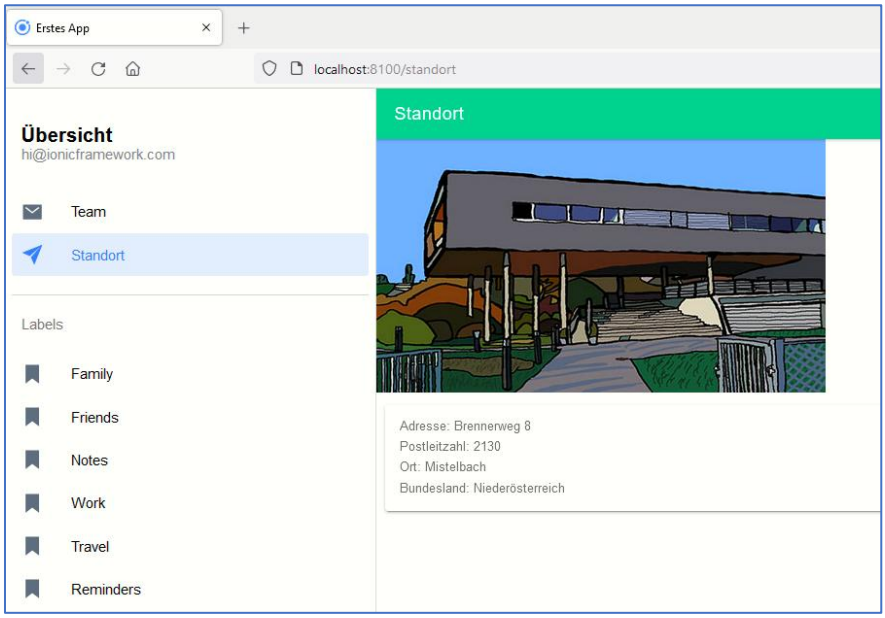
- titel: der name, der dann auch angezeigt wird
- url: beginnt mit dem Slash und es folgt die Bezeichung der Seite

Speichern und testen.

INFO: Den Pfad für z.B. das Team wird ausgelesen aus der Datei app-routing.modul.ts.

```
{
  path: 'team',
  loadChildren: () => import('./pages/team/team.module').then( m => m.TeamPageModule)
},
{
```

Ergebnis:



10)Ändern der Icons:

Nun passen aber die Icons nicht mehr zum Team bzw. Standort.

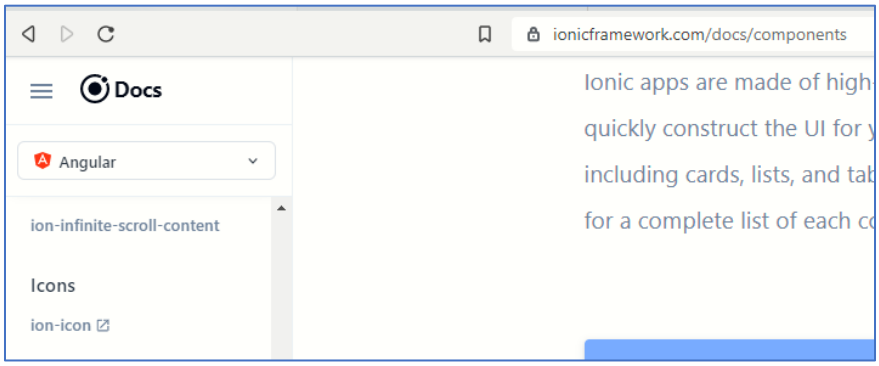
Die Bezeichnungen waren hier : mail und paper-plane

```
{ title: 'Team', url: '/team', icon: 'mail' },
{ title: 'Standort', url: '/standort', icon: 'paper-plane' },
```

Suche die besseren Icons hier aus:

Klicke auf der Website https://ionicframework.com/docs/components

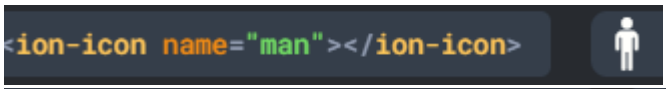
Im linken Bereich auf „ion-icon“ .

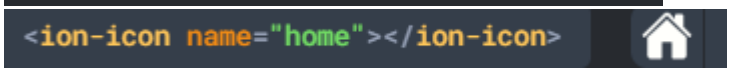


Man kommt auf einen externen Link nämlich https://ionic.io/ionicons

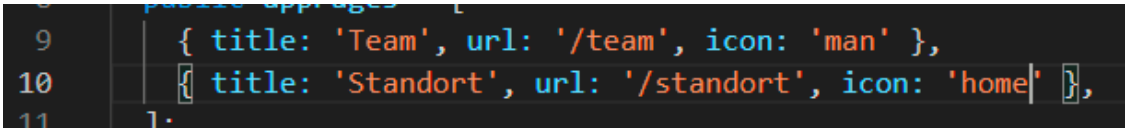
Gehe dort auf „Filled“ und man könnte folgende wählen:

Die „outline“ funktionieren hier leider nicht!

- 

- 

Ändere daher auf:



Ergebnis: