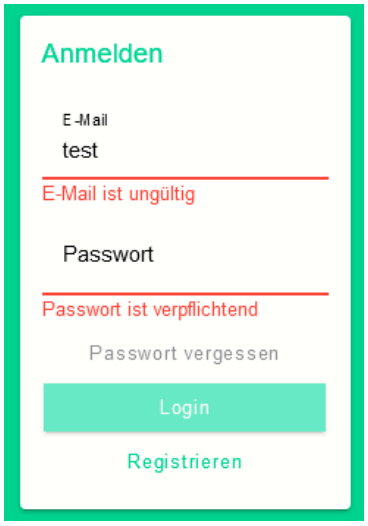
Ionic 5 – Validierung mit Reactive Forms

https://www.youtube.com/watch?v=3gaVbroD-I8

Inhalt:

1. **den „Form Builder“ erstellen**
    a. **Diesen FormBuilder verwenden, um die Form zu erstellen (create)**
    b. **Dieses Formular in der Login-Page verwenden**
    c. **das ReactiveFormsModul einbauen**
2. **Das Formular in der HTML nutzen**
3. **login Button disable**
4. **Passwort vergessen Button disabled**
5. **Error-Anzeige was falsch ist – für E-Mail (*ngIf)**
6. **Error-Anzeige was falsch ist – für Passwort**



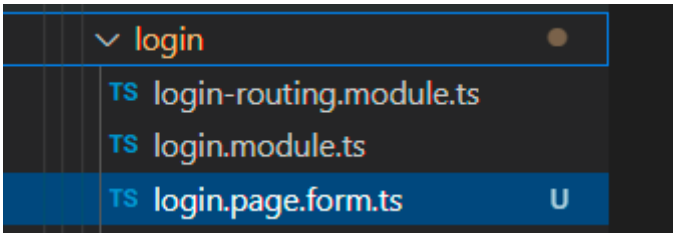
Ziel:

- Input Felder sollen zeigen, dass ein Fehler vorliegt.
- Der „Absenden“-Button soll nur klick bar sein, wenn das Formular valide ist.

1)den „Form Builder“ erstellen

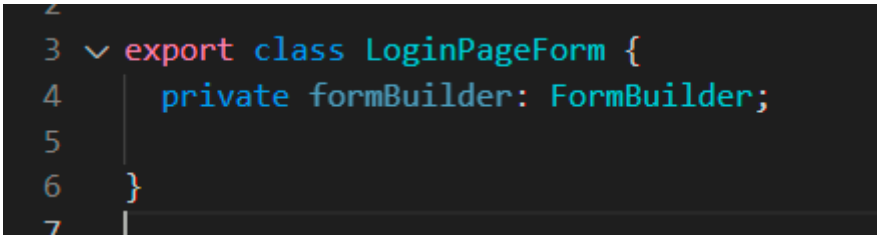
Erstelle im Ordner „login“ eine neue Site namens „login.page.form.ts“

Rechter Mausklick auf „login“-Ordner Symbol und dann „Neue Datei“:

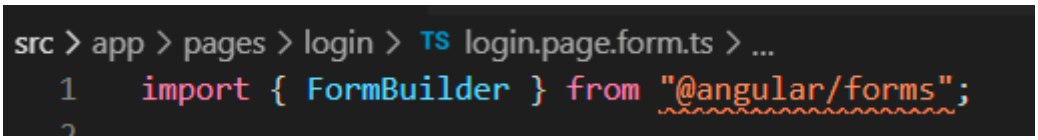


Hier soll eine export-Klasse angelegt werden und zwar ganz unten. Diese soll den Namen „LoginPageForm“ haben.

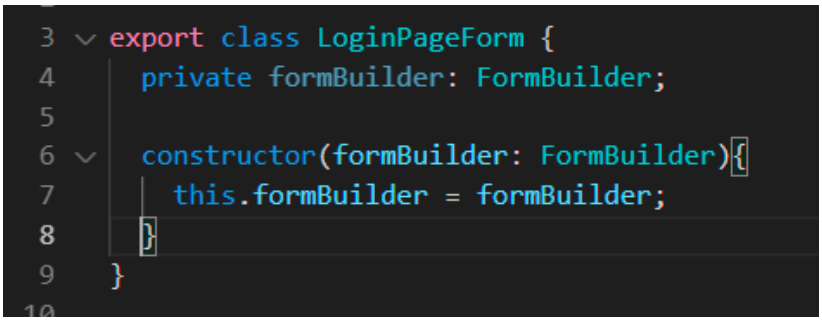
Erstelle diese Klasse und die private Variable, welche den Typ „FormBuilder“ hat: wenn man diese schreibt, sollte man die Ausfüllhilfe nutzen, damit sich der IMPORT ganz oben automatisch erstellt – siehe Zeile 2



export class LoginPageForm {}



Danach wird der constructor erstellt, wo der Typ festgelegt wird als „formBuilder“. In Zeile 7 wird mit „this“ der formBuilder referenziert, der den Wert vom Konstruktor bekommt.

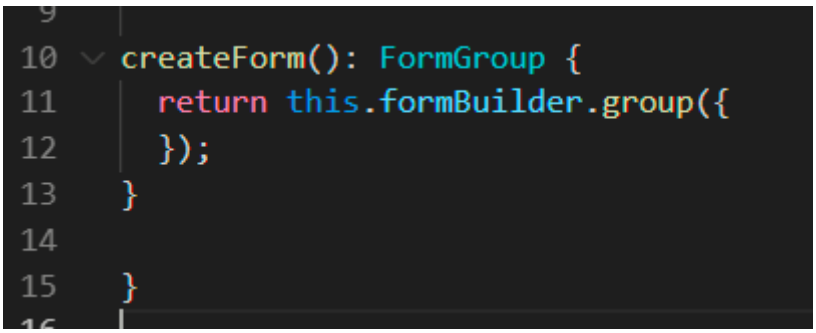


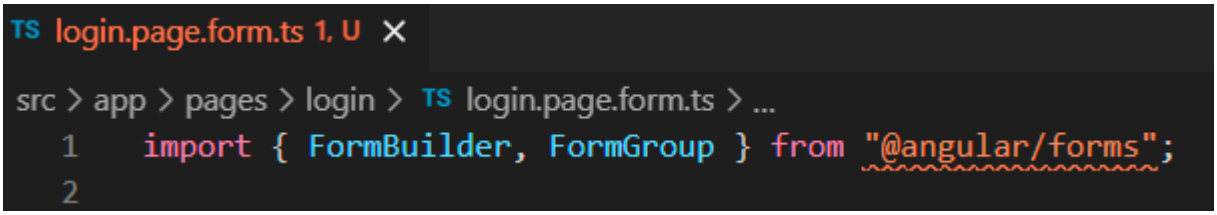

```
constructor(formBuilder: FormBuilder){
  this.formBuilder = formBuilder;
 }
```

INFO: Der Import wird unterwellt angezeigt, was auf einen Fehler hindeutet. Ignoriere diesen und er verschwindet, solange die Datei später dann geschlossen wird. Er stört nicht.

1a)Diesen FormBuilder verwenden, um die Form zu erstellen (create)

- Erstelle die Funktion „createForm()“ die eine Funktion namens FormGroup aufweist
- Beim Erstellen nach dem Doppelpunkt soll die „FormGroup“ langsam geschrieben werden, damit man wieder den Vorschlag erhält, den man übernimmt, damit in Zeile 1 im Import die „FormGroup“ importiert wird.
- Dann return diesen FormBuilder mit der Funktion „group“





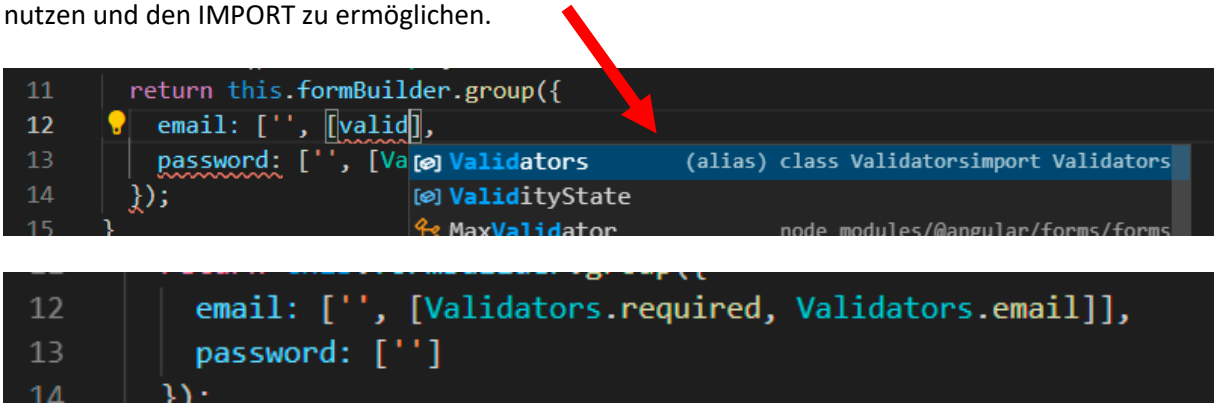
In dieser Funktion „group“ wird ein Objekt erhalten, das die einzelnen Eigenschaften (properties) validieren wird.

```
createForm(): FormGroup {
  return this.formBuilder.group({
    email: [''],
    password: ['']
  });
```

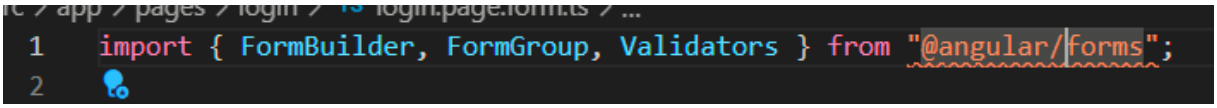
Zuerst wird das E-Mail überprüft:

- leer gelassen? – Beachte den Beistrich danach
- Ist es zwingend gefordert (required)
- Ist es in Form eines typischen E-Mails?

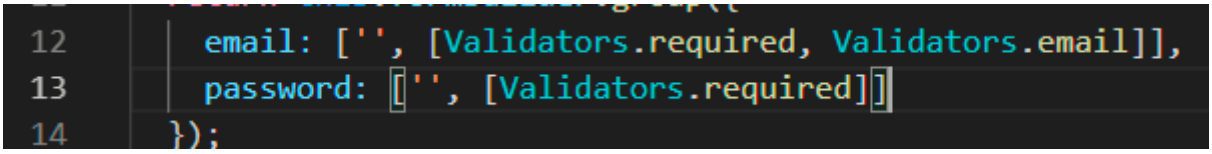
Beachte, dass „Validators“ langsam geschrieben wird um die automatische Vervollständigung zu nutzen und den IMPORT zu ermöglichen.



Zeile 1: IMPORT



Für das Passwort ebenfalls, jedoch ohne dem letzten Validator.



1b)Dieses Formular in der Login-Page verwenden

Öffne die „login.page.ts“

Erstelle in der export class eine

- Property als eine FormGroup

```
10  export class LoginPage implements OnInit {
11
12    form: FormGroup;
13
```

Dabei soll wiederum in Zeile 2 der passende Import stattfinden.

Dann muss diese Form initialisiert werden, was in der „ngOnInit“-Funktion passieret. Dabei wird die Form kreiert (erstellt):

```
18  ngOnInit() {
19    this.form = new LoginPageForm(this.formBuilder).createForm();
20  }
```

Beachte: Auch hier muss der IMPORT zugelassen werden. Siehe Zeile 4.

Damit hier keine Fehlermeldung erscheint (rote Welle) muss im „constructor“ ein FormBuilder erstellt werden.

```
15    constructor(private router: Router,
16      private formBuilder: FormBuilder ) { }
17
```

Auch dieser benötigt den IMPORT:

```
2  import { FormBuilder, FormGroup } from '@angular/forms';
3  import { Router } from '@angular/router';
4  import { LoginPageForm } from './login.page.form';
```

1c)das ReactiveFormsModul einbauen

Öffne die „login.module.ts“.
Hier muss folgendes in den „imports“ hinzugefügt werden: Beachte den Beistrich vorher, damit die Aufzählung funktioniert

```
11  @NgModule({
12    imports: [
13      CommonModule,
14      FormsModule,
15      IonicModule,
16      LoginPageRoutingModule,
17      ReactiveFormsModule
18    ],
```

Dadurch soll auch der IMPORT wieder automatisch erfolgen

```
2  import { CommonModule } from '@angular/common';
3  import { FormsModule, ReactiveFormsModule } from '@angular/forms';
```

2)Das Formular in der HTML nutzen

Öffne die „login.page.html“ und füge an passender Stelle dieses neue Formular ein, was im Hintergrund jetzt vorbereitet wurde:

```
Go to component
1  <ion-content>
2    <div class="flex-center">
3      <form [formGroup]="form"></form>
4      <ion-card>
```

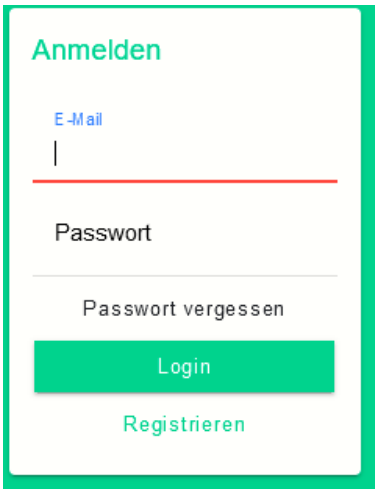
Ziehe den schließenden Tag nach unten zur Zeile:

```
21          </ion-card-con
22        </ion-card>
23      </form>
24    </div>
25  </ion-content>
```

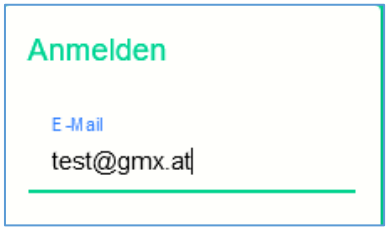
Beim E-Mail und Passwortfeld erstelle den passenden „formControlName“

```
<ion-label position="floating">E-Mail</ion-label>
<ion-input type="email" formControlName="email"></ion-input>
</ion-item>
<ion-item>
<ion-label position="floating">Passwort</ion-label>
<ion-input type="password" formControlName="password"></ion-input>
```

Ergebnis:
Ruft man die Seite auf, wird bereits eine rote Warnlinie angezeigt, wenn man in das Feld hineinklickt:



wird eine E-Mail eingeben:



Es funktioniert!

Das hängt an unseren Validierungen:

```
email: ['', [Validators.required, Validators.email]],
password: ['', [Validators.required]]
});
```

3)login Button disable

Der Login-Button soll erst klick bar sein, wenn die Validierung keinen Fehler erhält.
Dafür füge in den Button folgenden Code ein:

[disabled]="!form.valid"

```
<ion-button color="success" size="full" [disabled]="!form.valid">Login</ion-b
```

Die vorgefertigte Funktion „disabled“ greift hier genau dann, wenn die Validierung NICHT zutrifft, ausgedrückt durch das Rufzeichen am Beginn.

4)Passwort vergessen Button disabled

Hier soll das nur angezeigt werden, wenn das E-Mail valide ist. Denn es hängt ja an der E-Mail, die vorhanden sein soll. Ob es wirklich ein korrektes E-Mail ist, welches in der Datenbank vorhanden ist, wird hier nicht geprüft, sondern nur, ob es eine E-Mail sein könnte.
[disabled]="!form.get('email').valid"

```
18  tton color="dark" size="full" fill="clear" [disabled]="!form.get('email').valid">Passwort verg
19  tton color="success" size="full" [disabled]="!form.valid">Login</ion-button>
```

5)Error-Anzeige was falsch ist – für E-Mail

Das ist jeweils unter dem passenden Input-Feld ein ion-label mit einer roten Farbe.

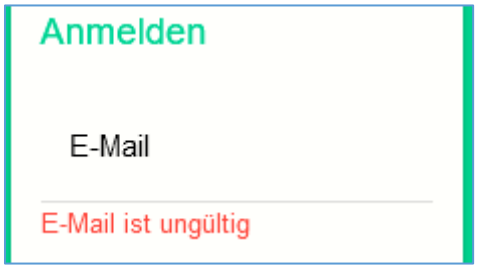
<ion-label color="danger"></ion-label>

```
12      <ion-input type="email" formControlName="email"></ion-input>
13    </ion-item>
14    <ion-label color="danger"></ion-label>
15
16    <ion-item>
```

Der Text ist frei wählbar.

```
</ion-item>
<ion-label color="danger">E-Mail ist ungültig</ion-label>
```

Das Problem ist aber jetzt, dass dieser Text IMMER angezeigt wird.



ABER er soll nur angezeigt werden, wenn

- Die Validierung nicht korrekt ist

Daher muss man mit einer IF arbeiten, die in IONIC folgender maßen heißt:

- *ngIf-Statement

Beachte das Sternchen vor den n

```
<ion-label color="danger" *ngIf="">E-Mail ist ungültig</ion-label>
```

Folgende Elemente sind zu berücksichtigen:

- Wenn das Input-Feld noch nicht geklickt (berührt) wurde, soll die Meldung nicht angezeigt werden:

  ```
  *ngIf="form.get('email').touched">E-Mail ist ungültig<
  ```

  form.get('email').touched

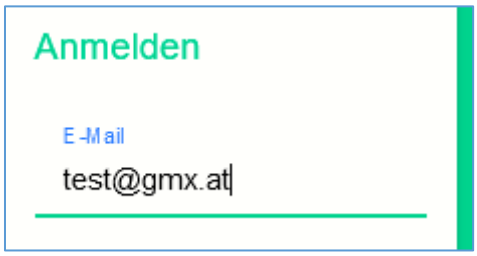
- Überprüfe, ob es einen Error gibt, der von Typ her „email“ ist

  ```
  && form.get('email').errors?.email">E-Mail i
  ```

Daher verknüpft mit einem UND (&&):

```
er" *ngIf="form.get('email').touched && form.get('email').errors?.email">E-Mail i
```

Ergebnis:



bzw.

Verfeinerung:
Zwei unterschiedliche Meldungen:

1. Das Feld ist verpflichtend.
2. E-Mail ist ungültig.

Das unterscheidet sich nur in der Error-Beziehung, dass bei der ersten Meldung statt „errors?.email“ steht

- errors?.required

Man muss aber beide gleichbedeutend verwenden. Daher die ganze Zeile kopieren und ändere auch den Text zu „E-Mail ist verpflichend“:

```
uched && form.get('email').errors?.required">E-Mail ist verpflichtend</ion-label>
uched && form.get('email').errors?.email">E-Mail ist ungültig</ion-label>
```

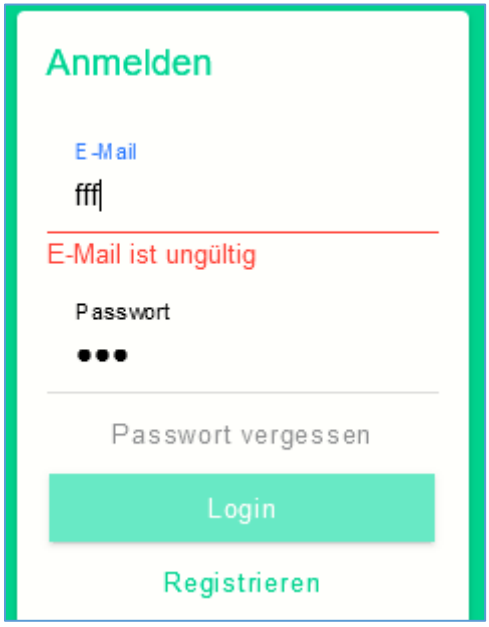
6)Error-Anzeige was falsch ist – für Passwort

Hier ist das Ziel, eine Error-Meldung zu erzeugen, wenn das Passwort nicht eingegeben wird.

Kopiere die Zeile von gerade eben und ändere es passend:

```
19        </ion-item>
20        <ion-label color="danger" *ngIf="form.get('password').touched &&
21        form.get('password').errors?.required">Passwort ist verpflichtend</ion-label>
```

Ergbnis:



Aufgabe:

Diese Validierung soll nun auch für ALLE Input-Felder der „Registrierung“ vorgenommen werden.