

Ionic Shop

Erstelle eine neue Seite im Ordner „seiten“ namens „shop“

```
PS C:\ionic_start\ersteapp> ionic g page pages/shop
```

Öffne die „shop.html“ und entferne vieles, damit das über bleibt:

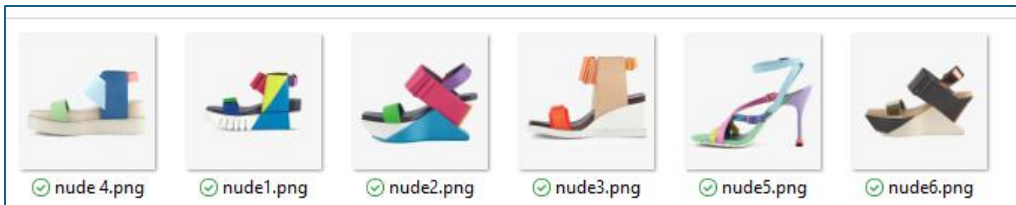
```
<> shop.page.html U X
src > app > pages > shop > <> shop.page.html > ion-header
Go to component
1 <ion-header>
2   <ion-toolbar>
3     <ion-title>Shop</ion-title>
4   </ion-toolbar>
5 </ion-header>
6
7 <ion-content >
8
9 </ion-content>
```

1) Daten im Shop anlegen

a) Kopiere die Bilder in den Ordner „assets“ in einen Ordner „shop“



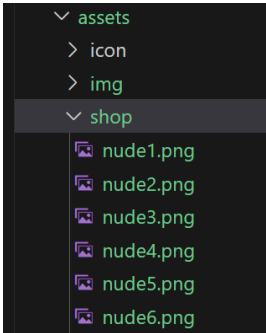
---> sorry doch lieber Schuhe 😊



360 x 360 Pixel

Folgende Modelle der Marke „united nude“:

- <https://unitednude.com/> <https://unitednude.at/>
- Nude1 delta run, Keilsandalette, rainbow, 250,00
- Nude2 delta, Plateausandalette, rainbow 280,00
- Nude3 Ko Mid, Keilsandalette, malibu 200,00
- Nude4 rico, Plateausandalette, summer 240,00
- Nude5 strappy hi, Hi Heel Sandalette, 180,00
- Nude6 delta, Plateausandalette, bronze 280,00



Code ist unterhalb zum Kopieren.

Service erstellen

Um die Daten für den Shop anzeigen lassen zu können, soll nun ein „Service“ erstellt werden.

Dazu soll mit Hilfe des Terminals die Dateien „api“ im Unterfolder von den „services“ erstellt werden.

```
eapp> ionic g service services/api
```

Ergebnis:

```
services\api
├── TS api.service.spec.ts  U
├── TS api.service.ts      U
└── TS app-routing.module.ts  M
```

Erstelle für alle Einträge (Produkte) im Shop ein Array: mit dem Namen „item“ – Strichpunkt am Ende:

```
src > app > services > api > TS api.service.ts > ...
4   |   providedIn: 'root'
5   |   })
6   |   export class ApiService {
7   |
8   |     items: any[] = [
9   |       {
10  |
11  |     };
12  |     constructor() { }
13  |   }
```

Innerhalb der geschwungenen Klammern sollen nun alle 6 Artikel angeführt werden. Hier ist das erste Element dargestellt, darunter als Kopiervorlage der Rest:

```
8   |     items: any[] = [
9   |       {
10  |         id: '1',
11  |         name: 'Delta run',
12  |         preis: 250,
13  |         rating: 4.5,
14  |         cover: 'assets/shop/nude1.png',
15  |         beschreibung: 'Keilsandalette, rainbow'
16  |       },
17  |       {
18  |         id: '2',
```

```
items: any[] = [
  {
    id: '1',
    name: 'Delta run',
    preis: 250,
    rating: 4.5,
    cover: 'assets/shop/nude1.png',
    beschreibung: 'Keilsandalette, rainbow'
  },
  {
```

```

id: '2',
name: 'Delta',
preis: 280,
rating: 4.7,
cover: 'assets/shop/nude2.png',
beschreibung: 'Plateausandalette, rainbow'
},
{
id: '3',
name: 'Ko Mid',
preis: 200,
rating: 4.2,
cover: 'assets/shop/nude3.png',
beschreibung: 'Keilsandalette, malibu'
},
{
id: '4',
name: 'Rico',
preis: 240,
rating: 4.8,
cover: 'assets/shop/nude4.png',
beschreibung: 'Plateausandalette, summer'
},
{
id: '5',
name: 'Strappy Hi',
preis: 180,
rating: 4.5,
cover: 'assets/shop/nude5.png',
beschreibung: 'Hi Heel Sandalette'
},
{
id: '6',
name: 'Delta',
preis: 280,
rating: 4.6,
cover: 'assets/shop/nude6.png',
beschreibung: 'Plateausandalette, bronze'
}
}

```

2) Daten in shop.page.ts in eine Funktion einbauen

In der Export-Klasse füge mittels „inject“ das eben angelegte API-Service ein und beachte den gleichzeitigen IMPORT durch Anklick des Vorschlages:

```

7
8   export class ShopPage implements OnInit {
9
10    private api = inject(Api)
11
12    constructor() { }

```

The screenshot shows a dropdown menu for the `inject(Api)` call. The menu items are:

- `ApiService` (with a link icon) `src/app/services/api/api.service`
- `api` (with a cube icon)

Dadurch wird der Import eingefügt, der unbedingt nötig ist: siehe Zeile 2 auch in Zeile 1: inject - beachten

```
<> shop.page.html U TS shop.page.ts U X TS api.service.ts U
src > app > pages > shop > TS shop.page.ts > ...
1  import { Component, OnInit, inject } from '@angular/core';
2  import { ApiService } from 'src/app/services/api/api.service';
3
4  @Component({
```

```
9  export class ShopPage implements OnInit {
10
11     private api = inject(ApiService);
12
13     constructor() { }
14
15     ngOnInit() {
16
17
18     }
19 }
```

Funktion erstellen

In die „ngOnInit-Funktion“ schreibe

`this.getItems();`

```
15     constructor() { }
16
17     ngOnInit() {
18         this.getItems();
19     }
```

Erstelle danach eine neue Funktion, die die Items aufrufen wird:

```
17     ngOnInit() {
18         this.getItems();
19     }
20
21     getItems(){
22     }
23
24 }
25
```

Darin werden „allItems“ mit den Ergebnissen aus dem API „angefüttert“.

`this.allItems = this.api.items;`

`this.items = [...this.allItems];`

```
21     getItems(){
22         this.allItems = this.api.items;
23         this.items = [...this.allItems];
24     }
25
```

Dafür aber in der Klasse oben müssen diese Variablen angelegt werden:

Items vom Typ „any“ und initialisiert als „leeres Array“, genauso die Variable „allItems“

```

9  export class ShopPage implements OnInit{
10
11     items: any[] = [];
12     allItems: any[] = [];
13     private api = inject(ApiService);

```

3)Anzeigen in der „shop.page.html“:

Dafür lege im Content typische Elemente an wie

- ion-row
- ion-col

In jeder „col“ soll dann ein Artikel mit Bild und Beschreibung angezeigt werden. Es wird auch eine Variation eingebaut werden, die auf die Bildschirmgröße Rücksicht nehmen wird.

```

7  <ion-content >
8  <ion-row>
9  <ion-col>
10
11 </ion-col>
12 </ion-row>
13 </ion-content>

```

Um durch alle Einträge durchzugehen (looping) benutze eine Schleife. Die Ordnung (track) könnte auch nach „item.id“ erfolgen, hier aber etwas großzügiger durch „i“, welche nach einem Index erfolgt.

@for(item of items; track i; let i = \$index){}

```

7  <ion-content >
8  <ion-row>
9  @for(item of items; track i; let i = $index){
10 <ion-col>

```

ABER: in die Funktion muss jede „col“ hinein, daher muss die schließende geschwungenen Klammer nach dem Ende der <ion-col> gezogen werden!!!

```

7  <ion-content >
8  <ion-row>
9  @for(item of items; track i; let i = $index){
10 <ion-col>
11 </ion-col>
12 } <!-- hier schließende Klammer-->
13 </ion-row>
14 </ion-content>

```

Innerhalb der „col“ dann „thumbnail“:

```

10 <ion-col>
11   <ion-thumbnail></ion-thumbnail>
12 </ion-col>
13 </ion-row>

```

In das „thumbnail“ (Fingernagel = Mini-Bild) wird ein Bild gestellt:

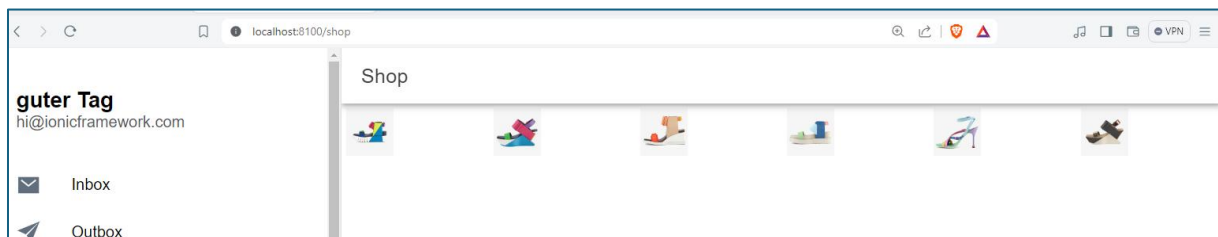
```

10 <ion-col>
11   <ion-thumbnail>
12     <ion-img [src]="item?.cover"></ion-img>
13   </ion-thumbnail>
14 </ion-col>
15 } <!-- hier schließende Klammer-->

```

Ergebnis:

die Bilder werden schon angezeigt – bei Eingabe „shop“ in der URL



Verbesserung: Erstellen von „cards“

Diese Cards bieten die schöne Möglichkeit von einzelnen Rahmen, die dann später pro Shop-Item mit Bild und Beschreibung und entsprechender Hintergrundfarbe eine gute Abgrenzung bieten werden.

```

10 <ion-col>
11   <ion-card>
12     <ion-thumbnail>
13       <ion-img [src]="item?.cover"></ion-img>
14     </ion-thumbnail>
15   </ion-card>
16 </ion-col>
17 } <!-- hier schließende Klammer-->

```

Verbesserung durch die Anpassung an die Bildschirmgrößen

Bei einer Anzahl von 12 Spalten pro Reihe ergibt sich z.B. bei einem „sizeXS“ für ein Smartphone eine Aufteilung bei 6 eine Anzeige von 2 Bildern pro Zeile, was sinnvoll ist. Die Berechnung erfolgt nämlich: $12 / 6 = 2$ pro Zeile

```

9   @for(item of items; track i; let i = $index){
10  <ion-col sizeLg="3" sizeMd="4" sizeSm="6" sizeXL="3" sizeXs="6">
11    <!--Aufteilung bei 12 Spalten pro Reihe-->
12    <ion-card>

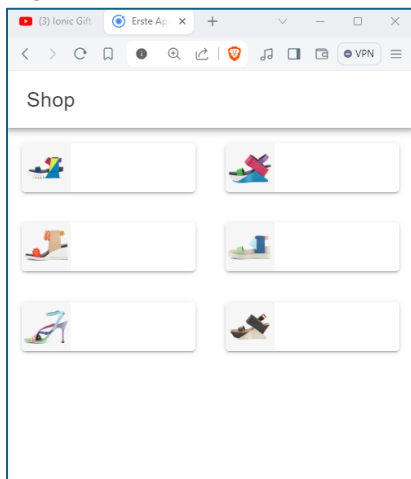
```

```

<ion-col sizeLg="3" sizeMd="4" sizeSm="6" sizeXL="3" sizeXs="6">
  <!--Aufteilung bei 12 Spalten pro Reihe-->

```

Ergebnis bei XS:



Beschreibung bei den Bildern anzeigen

Erstelle innerhalb der Card aber unter dem Thumbnail ein <ion-label> für den Textbereich.

```

<ion-label>
  <ion-text color="dark"><strong>
    {{item?.name}}
  </strong></ion-text>
</ion-label>

```

```

15   </ion-thumbnail>
16   <ion-label>
17     <ion-text color="dark"><strong>
18       {{item?.name}}
19     </strong></ion-text>
20   </ion-label>
21   </ion-card>
22 </ion-col>

```

Darunter der Preis:

```

<p>
  <ion-text color="dark">
    <strong>
      {{item?.preis}}
    </strong>
  </ion-text>
</p>

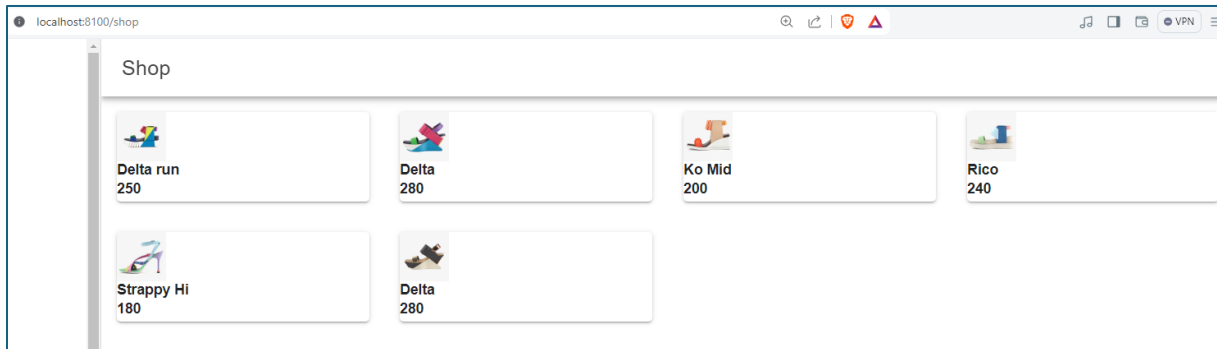
```

```

19     </strong></ion-text>
20   <.....><p>
21     <.....><ion-text color="dark">
22     <.....><strong>
23     <.....>{{item?.preis}}
24     <.....></strong>
25     <.....></ion-text>
26   <.....></p>
27
28 </ion-label>

```

Ergebnis:



Innerhalb vom <p> wird nun das Rating und ein Icon „star“ angezeigt:

```

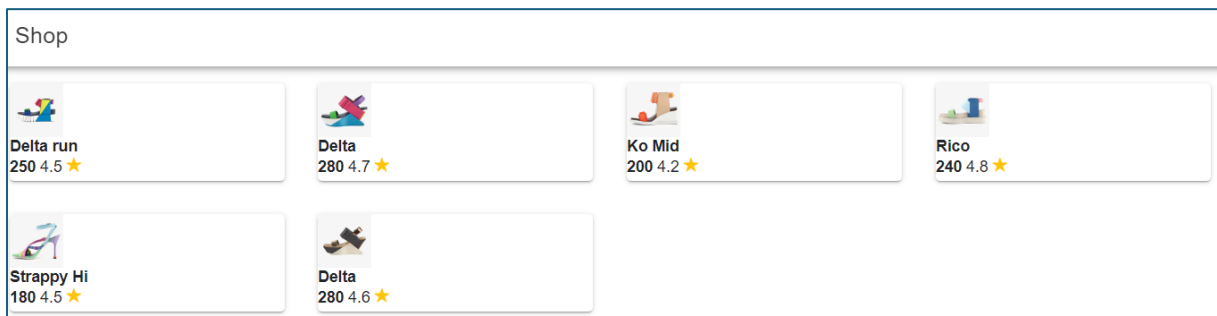
<ion-text class="rating" color="dark">
  {{item?.rating}}
  <ion-icon name="star" color="warning"></ion-icon>
</ion-text>

```

```

25     </ion-text>
26     <ion-text class="rating" color="dark">
27     <.....>{{item?.rating}}
28     <.....><ion-icon name="star" color="warning"></ion-icon>
29     <.....></ion-text>
30   <.....></p>
31 </ion-label>

```



Beim Preis füge ein Eurozeichen hinzu:

```

21     <ion-text color="dark">
22         <strong>
23             € {{item?.preis}}
24         </strong>
25     </ion-text>

```

Design per CSS

Öffne die „shop.page.scss“.

Erstelle die ion-card und diese wird nun designed.

```

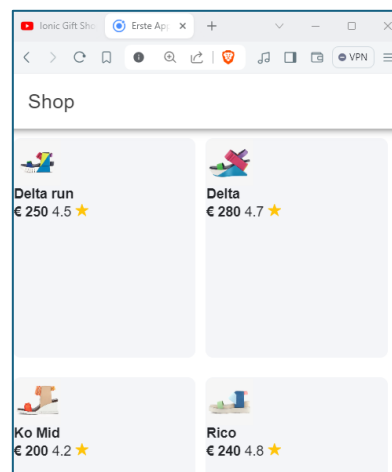
shop.page.scss U X
src > app > pages > shop > shop.page.scss > ion-card
1  ion-card {
2      --background: var(--ion-color-light);
3      height: 14rem;
4      margin: 0.3rem 0; // top und bottom, null links und rechts
5      border-radius: 8px;
6      box-shadow: none;
7      border: 1px solid var(--ion-color-light);
8  }
9

```

```

ion-card {
  --background: var(--ion-color-light);
  height: 14rem;
  margin: 0.3rem 0; // top und bottom, null links und rechts
  border-radius: 8px;
  box-shadow: none;
  border: 1px solid var(--ion-color-light);
}

```



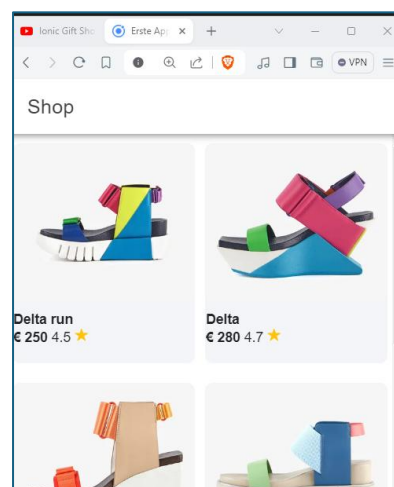
Verbesserung, da Bild zu klein:

Innerhalb der Klasse, erstelle:

```

7      border: 1px solid var(--ion-color-light);
8
9      ion-thumbnail {
10         width: 100%;
11         height: 10rem;
12         margin-bottom: 10px;
13         ion-img {
14             height: 100%;
15             border-radius: 8px;
16         }
17     }
18 }

```



```
ion-thumbnail {
```

```

width: 100%;
height: 10rem;
margin-bottom: 10px;
ion-img{
  height: 100%;
  border-radius: 8px;
}
}

```

Design vom Label – alles ebenfalls noch in der „card“

Die font-size und auch den Abstand der beiden Zeile, daher „label UND p“ darunter mit einem Abstand von links und rechts

```

18
19   ion-label {
20     font-size: 0.9rem;
21   }
22   ion-label, p {
23     margin: 0 10px;
24   }
25 }
26

```

Die Zeile von „p“ ebenfalls extra designen inkl. „rating und Stern“

Stern und rating soll rechtsbündig sein

```

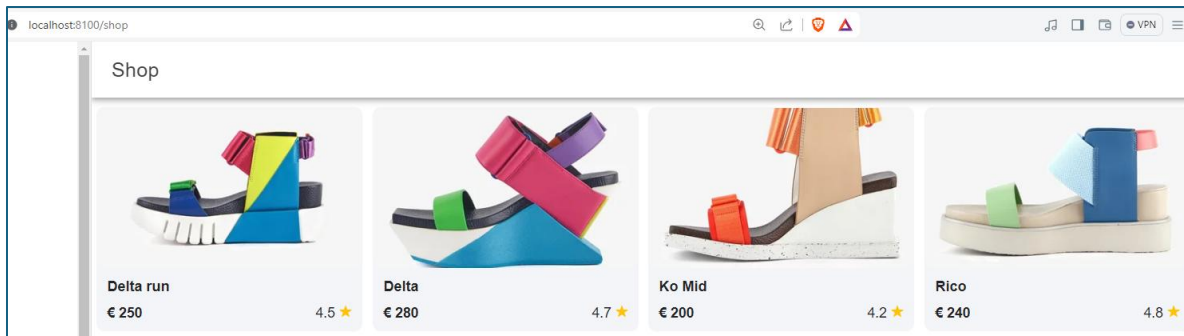
25
26   p {
27     font-size: 0.85;
28     margin-top: 8px;
29     font-weight: 500;
30     ion-text.rating {
31       float: right;
32       ion-icon {
33         font-size: 0.8rem; //damit der Stern kleiner ist
34       }
35     }
36   }

```

```

p{
font-size: 0.85;
margin-top: 8px;
font-weight: 500;
ion-text.rating{
float: right;
ion-icon{
font-size: 0.8rem; //damit der Stern kleiner ist
}
}
}

```



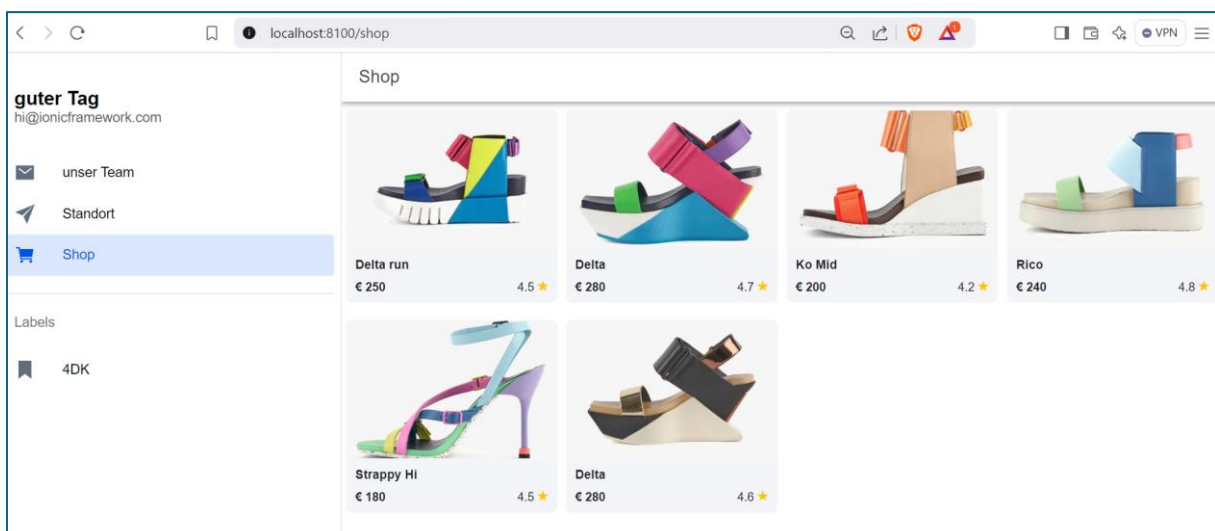
In die Side-Navigation aufnehmen

Öffne die „app.component.ts“ und erstelle einen neuen Eintrag

```

app.component.ts 8
app.component.spec.ts 9
app.component.ts M 10
app.module.ts M 11
12
public appPages = [
  { title: 'unser Team', url: '/team', icon: 'mail' },
  { title: 'Standort', url: '/standort', icon: 'paper-plane' },
  { title: 'Shop', url: '/shop', icon: 'cart' },
];

```



Searchbar

folgt

Shop

<https://www.youtube.com/watch?v=tx0ZM6UA464>

https://www.youtube.com/watch?v=BHM7_NmX_LA