Labo 3 – Reactive forms en Pipes

In dit labo bouw je vier autonome Angular 20-toepassingen: twee rond Reactive Forms en twee rond Pipes.

Na dit labo kan je:

- Een FormGroup opzetten met FormControl en Validators, inclusief foutmeldingen in de template.
- Een FormArray gebruiken voor dynamische rijen en totalen berekenen.
- Ingebouwde pipes toepassen (zoals date, currency, number, percent, uppercase, titlecase) en pipes combineren.
- Een custom pipe schrijven en toepassen met parameters.

In elke opdracht krijg je het form-model (voor de form-oefeningen) of een pipe-skelet (voor de custom pipe). Dit verlaagt de instapdrempel en laat je focussen op logica en binding. Je mag er uiteraard voor kiezen om deze zelf van nul te schrijven.

Opdracht 1 – Klantformulier

Doel: werken met FormGroup, FormControl en standaard Validators. Toon live de form-waarde en fouten.

Component

- KlantFormulierComponent – bevat het volledige formulier.

Form-model (startcode)

```
form = new FormGroup({
  naam: new FormControl<string>('', { nonNullable: true, validators:
[Validators.required, Validators.minLength(2)] }),
  email: new FormControl<string>('', { nonNullable: true, validators:
[Validators.required, Validators.email] }),
  geboortedatum: new FormControl<string>('', { nonNullable: true,
validators: [Validators.required] }),
  land: new FormControl<string>('BE', { nonNullable: true, validators:
[Validators.required] }),
});
landen = ['BE', 'NL', 'FR', 'DE'];
```

Taken

- Bind het formulier aan de template met [formGroup] en formControlName.
- Toon live JSON-weergave onder het formulier:

  ```
  <pre>{{ form.value | json }}</pre>
  ```

- Toon foutmeldingen met @if, bijvoorbeeld: “Naam is verplicht (min. 2 letters)”, “E-mail is ongeldig”.

- De Submit-knop is enkel actief als form.valid is.
- Voeg een Reset-knop toe die het formulier leegt of naar standaardwaarden terugzet.
- Styling is van kleiner belang maar het is toegelaten je applicatie netjes te stylen 😀

Opdracht 2 – Bestellijst met rijen (Reactive Forms – FormArray)

Doel: dynamische rijen beheren met FormArray, totalen berekenen en validatie toepassen.

Component

- BestellingComponent – volledig formulier + lijst van orderregels.

Form-model (startcode)

```
regelGroup() {
  return new FormGroup({
    product: new FormControl<string>('', { nonNullable: true, validators:
[Validators.required] }),
    aantal: new FormControl<number>(1, { nonNullable: true, validators:
[Validators.required, Validators.min(1)] }),
    prijs: new FormControl<number>(0, { nonNullable: true, validators:
[Validators.required, Validators.min(0)] }),
  });
}

form = new FormGroup({
  klant: new FormControl<string>('', { nonNullable: true, validators:
[Validators.required] }),
  regels: new FormArray<FormGroup<any>>([ this.regelGroup() ]),
  opmerkingen: new FormControl<string>('', { nonNullable: true })
});

get regels() { return this.form.controls.regels as FormArray<FormGroup>; }
get totaal(): number {
  return this.regels.controls.reduce((sum, g) =>
    sum + (g.value.aantal ?? 0) * (g.value.prijs ?? 0), 0);
}
```

Taken

- Toon de rijen met @for over regels.controls.
- Voorzie knoppen “Rij toevoegen” en “Verwijderen”.
- Toon onderaan het totaalbedrag (live herberekend).
- Bonus: disable de submit-knop als totaal === 0 of bij ongeldige rijen.
- Bonus: gebruik valueChanges om een melding te tonen bij totaal ≥ €100 (“Gratis verzending!”).
- Styling is van kleiner belang maar het is toegelaten je applicatie netjes te stylen 😀

Opdracht 3 – Productenlijst met ingebouwde Pipes

Doel: ingebouwde pipes toepassen en combineren (date, currency, percent, uppercase, titlecase).

Component

- ProductenLijstComponent – toont een lijst of tabel van producten.

Dataset

```
producten = [
  { id: 1, naam: 'Laptop Pro', prijs: 1299.99, voorraad: 0.25, toegevoegd:
new Date(2025, 0, 15) },
  { id: 2, naam: 'Mechanisch Toetsenbord', prijs: 139.5, voorraad: 0.6,
toegevoegd: new Date(2025, 2, 3) },
  { id: 3, naam: 'USB-C Dock', prijs: 89, voorraad: 0.05, toegevoegd: new
Date(2024, 10, 28) },
  { id: 4, naam: 'Noise-cancelling Koptelefoon', prijs: 299, voorraad:
0.8, toegevoegd: new Date(2025, 4, 1) },
];
```

Taken

- Toon de naam met titlecase of uppercase.
- Toon de prijs met currency:'EUR'.
- Toon de voorraad als percent met 0 of 1 decimal.
- Toon de toevoegdatum met date:'shortDate' of date:'dd/MM/yyyy'.
- Laat pipe-chaining zien, bv. {{ naam | uppercase | slice:0:12 }}.
- Bonus: stel locale in op nl-BE voor Europese notatie van currency en date.

Opdracht 4 – Custom Pipe: filteren op zoekterm

Doel: een custom pipe schrijven die een lijst filtert op basis van een zoekterm. Combineer met [(ngModel)] voor live filtering.

Componenten

- ZoekLijstComponent – inputveld + lijst.
- FilterPipe – custom pipe voor filtering.

Dataset

```
items = [
  { id: 1, titel: 'Angular voor Starters' },
  { id: 2, titel: 'Professioneel TypeScript' },
  { id: 3, titel: 'Websecurity Basis' },
  { id: 4, titel: 'Frontend Patterns' },
  { id: 5, titel: 'Reactive Forms in de Praktijk' },
];
zoekterm = '';
```

Pipe-skelet

```
@Pipe({ name: 'filter' })
export class FilterPipe implements PipeTransform {
  transform<T extends { [k: string]: any }>(lijst: T[], term: string,
veld: keyof T = 'titel' as any): T[] {
    if (!lijst || !term) return lijst ?? [];
    const lower = term.toLowerCase();
    return lijst.filter(item => String(item[veld] ??
'').toLowerCase().includes(lower));
  }
}
```

Taken

- Gebruik de pipe in de template:

  ```
  @for (item of (items | filter:zoekterm:'titel'); track item.id) {
  <p>{{ item.titel }}</p>
  }
  ```

- Voorzie een input met [(ngModel)]="zoekterm" om live te filteren.
- Bonus: breid de pipe uit zodat een tweede parameter bepaalt op welk veld gefilterd wordt (bv. 'auteur').
- Bonus: toon “Geen resultaten” met @if wanneer de lijst leeg is.
  M