3 Señales y control

Cambios en los datos producen cambios en la presentación

3.1 Actualización de datos y refresco de pantalla usando Signals.

3.1.1 Señales por todas partes

```
export class BookingsComponent {
  readonly activity: Activity = {
    name: "Paddle surf",
    location: "Lake Leman at Lausanne",
    price: 125,
    date: new Date(2025, 7, 15),
    minParticipants: 5,
    maxParticipants: 9,
    status: "published",
    id: 1,
    slug: "paddle-surf",
    duration: 2,
    userId: 1,
  };
  readonly alreadyParticipants = 3;

  readonly newParticipants = signal(0);
  readonly booked = signal(false);

  onNewParticipantsChange(newParticipants: number) {
    this.newParticipants.set(newParticipants);
  }

  onBookClick() {
    this.booked.set(true);
  }
}
```

```
<article>
  <header>
    <h2>{{ activity.name }}</h2>
    <div [class]="activity.status">
```

```
      <span>{{ activity.location }}</span>
      <span>{{ activity.price | currency }}</span>
      <span>{{ activity.date | date: 'dd-MMM-yyyy' }}</span>
      <span>{{ activity.status | uppercase }}</span>
    </div>
  </header>
  <main>
    <h4>Participants</h4>
    <div>Already Participants: {{ alreadyParticipants }}</div>
    <div>New participants: {{ newParticipants() }}</div>
  </main>
  <footer>
    <h4>New Bookings</h4>
    <label for="newParticipants">How many participants want to book?</label>
    <input type="number" [ngModel]="newParticipants()"
(ngModelChange)="onNewParticipantsChange($event)" />
    <button [disabled]="booked() || newParticipants() === 0"
(click)="onBookClick()">Book now!</button>
  </footer>
</article>
```

3.2 Programación reactiva con Signals.

3.2.1 Computación derivada

```
maxNewParticipants = this.activity.maxParticipants - this.alreadyParticipants;
totalParticipants = computed(() => this.alreadyParticipants +
this.newParticipants());
remainingPlaces = computed(() => this.activity.maxParticipants -
this.totalParticipants());
bookingAmount = computed(() => this.newParticipants() * this.activity.price);
canNotBook = computed(() => this.booked() || this.newParticipants() === 0);
```

```
<article>
  <header>
    <h2>{{ activity.name }}</h2>
    <div [class]="activity.status">
      <span>{{ activity.location }}</span>
      <span>{{ activity.price | currency }}</span>
      <span>{{ activity.date | date: 'dd-MMM-yyyy' }}</span>
      <span>{{ activity.status | uppercase }}</span>
    </div>
  </header>
  <main>
    <h4>Participants</h4>
    <div>Already Participants: {{ alreadyParticipants }}</div>
```

```
    <ul>
      <li>New Participants: {{ newParticipants() }}</li>
      <li>Total participants: {{ totalParticipants() }}</li>
      <li>Remaining places: {{ remainingPlaces() }}</li>
    </ul>
  </main>
  <footer>
    <label for="newParticipants">How many participants want to book?</label>
    <input
      type="number"
      name="newParticipants"
      [ngModel]="newParticipants()"
      (ngModelChange)="onNewParticipantsChange($event)"
      min="0"
      [max]="maxNewParticipants" />
    <button [disabled]="canNotBook()" (click)="onBookClick()">Book now for {{
bookingAmount() | currency }}!</button>
    <button>Cancel</button>
  </footer>
</article>
```

3.2.2 Efectos colaterales

```
constructor() {
 effect(() => {
    const totalParticipants = this.totalParticipants();
    const activity = this.activity;
    if (totalParticipants >= activity.maxParticipants) {
      activity.status = 'sold-out';
    } else if (totalParticipants >= activity.minParticipants) {
      activity.status = 'confirmed';
    } else {
      activity.status = 'published';
    }
  });
}
```

3.3 Estructuras de control declarativas.

3.3.1 Condicionales

```
@if(remainingPlaces() > 0) {
<label for="newBookings">How many bookings?</label>
<input type="number" [ngModel]="newBookings()"
(ngModelChange)="onNewBookings($event)" min="0" [max]="maxBookings()" />
} @else {
```

```
<div>
  <button class="secondary outline"
(click)="onNewParticipantsChange(0)">Reset</button>
  <span>No more places available</span>
</div>
}
```

3.3.2 Repetitivas

```
class CookingComponent {
  participants = signal<{ id: number }[]>([{ id: 1 }, { id: 2 }, { id: 3 }]);

  onNewParticipantsChange(newParticipants: number) {
    if (newParticipants > this.maxNewParticipants) {
      newParticipants = this.maxNewParticipants;
    }
    this.newParticipants.set(newParticipants);
    this.participants.update((participants) => {
      const updatedParticipants = participants.slice(0,
this.alreadyParticipants);
      for (let i = 0; i < newParticipants; i++) {
        updatedParticipants.push({ id: participants.length + 1 });
      }
      return updatedParticipants;
    });
  }
}
```

```
<div>
  @for (participant of participants(); track participant.id) {
  <span [attr.data-tooltip]="participant.id">🏃</span>
  } @empty {
  <span>🕸</span>
  }
</div>
```

3.3.3 Extras

```
@Component({
  selector: "lab-footer",
  standalone: true,
  imports: [],
  template: ``,
  styles: ``,
```

```
  changeDetection: ChangeDetectionStrategy.OnPush,
})
export class FooterComponent {
  author = {
    name: "Alberto Basalo",
    homepage: "<https://albertobasalo.dev>",
  };

  year = new Date().getFullYear();

  cookiesAccepted = signal(false);

  onAcceptClick() {
    console.log("Cookies accepted!");
    this.cookiesAccepted.set(true);
  }
}
```

```
<footer>
  <nav>
    <span>
      <a [href]="author.homepage" target="_blank">© {{ year }} {{ author.name }}
</a>
    </span>
    <span>
      @if (cookiesAccepted()) {
      <small>🍪</small>
      } @else {
      <button (click)="onAcceptClick()" class="secondary outline">Accept
Cookies</button>
      }
    </span>
  </nav>
</footer>
```