4 Rutas y SPA

Páginas de contenido dinámico

4.1 Conceptos de enrutado y Single Page Applications.

4.1.1 Configuración y router outlet

```
// provideRouter en app.config
export const appConfig: ApplicationConfig = {
  providers: [provideRouter(routes), provideClientHydration()],
};
//  app.routes.json
// / (bookingsComponent)
export const routes: Routes = [
  {
    path: "",
    loadComponent: () => import("./bookings/bookings.component").then((m) =>
m.BookingsComponent),
  },
];
```

```
// App component y router outlet
@Component({
  selector: "lab-root",
  standalone: true,
  imports: [RouterOutlet, HeaderComponent, FooterComponent],
})
export class AppComponent {}
```

```
<lab-header />
<main>
  <router-outlet />
</main>
<lab-footer />
```

```
main {
    margin-top: 2rem;
    margin-bottom: 2rem;
}
```

4.1.2 Router link

`ng g c routes/auth/login`

```
// /auth/login LoginComponent
{
  path: 'auth/login',
  loadComponent: () => import('./auth/login.component').then((m) => m.LoginComponent),
},

// HeaderComponent [routerLink] / /login
@Component({
  selector: 'lab-header',
  standalone: true,
  imports: [RouterLink],
  template: ``,
  styles: ``,
  changeDetection: ChangeDetectionStrategy.OnPush,
})
export class HeaderComponent {
  title = 'Activity Bookings';
}
```

```
<header>
  <nav>
    <a [routerLink]="['/']">
      <strong> {{ title }} </strong>
    </a>
    <a [routerLink]="['/auth', 'login']">Login</a>
  </nav>
</header>
```

4.1.3 Page components

`ng g c routes/auth/register --skip-selector --type=page`

```
// register.page.ts
@Component({
  standalone: true,
  imports: [],
  template: ` <p>register works!</p> `,
  styles: ``,
  changeDetection: ChangeDetectionStrategy.OnPush,
})
export default class RegisterPage {}

// .eslintrc.json
"rules": {
    "prettier/prettier": "warn",
    "@angular-eslint/component-class-suffix": [
      "error",
      {
        "suffixes": ["Component", "Page", "Template", "Widget"]
      }
    ],
// app.routes.es
{
  path: 'auth/register',
  loadComponent: () => import('./auth/register.page'),
},

// login.component.ts

@Component({
  selector: 'lab-login',
  standalone: true,
  imports: [RouterLink],
  template: ` `,
  styles: ``,
  changeDetection: ChangeDetectionStrategy.OnPush,
})
export class LoginComponent {}
```

```
<article>
  <header>
    <h2>Login</h2>
  </header>
  <main>
    <form>
      <label for="email">
        <span>Email</span>
        <input id="email" type="email" />
      </label>
      <label for="password">
        <span>Password</span>
        <input id="password" type="password" />
      </label>
      <button type="submit">Login</button>
    </form>
  </main>
  <footer>
    <a [routerLink]="['/auth', 'register']">Register if don't have an account</a>
  </footer>
</article>
```

```
@Component({
  standalone: true,
  imports: [RouterLink],
  template: `   `,
  styles: ``,
  changeDetection: ChangeDetectionStrategy.OnPush,
})
export default class RegisterPage {}
```

```
<article>
  <header>
    <h2>Register</h2>
  </header>
  <main>
    <form>
      <label for="username">
        <span>Username</span>
        <input id="username" type="text" />
      </label>
      <label for="email">
        <span>Email</span>
        <input id="email" type="email" />
      </label>
      <label for="password">
        <span>Password</span>
        <input id="password" type="password" />
      </label>
      <label for="confirm">
        <span>Confirm Password</span>
        <input id="confirm" type="password" />
      </label>
      <label for="terms">
        <span>Accept the terms and conditions</span>
        <input id="terms" type="checkbox" />
      </label>
      <button type="submit">Login</button>
    </form>
  </main>
  <footer>
    <a [routerLink]="['/auth', 'login']">Login if already have an account</a>
  </footer>
</article>
```

4.2 Parámetros en las rutas, señales en los componentes.

4.2.1 Configuración y envío

activities.data.ts

```
// activities array TypeScript constant
export const ACTIVITIES: Activity[] = [
  {
    name: 'Paddle surf',
    location: 'Lake Leman at Lausanne',
    price: 125,
    date: new Date(2023, 7, 15),
    minParticipants: 5,
    maxParticipants: 9,
    status: 'done',
    id: 1,
    slug: 'paddle-surf-lake-leman-at-lausanne',
    duration: 2,
    userId: 1,
  },...
```

`ng g c routes/home --skip-selector --type=page`

```
// config with routed params
export const routes: Routes = [
    {
      path: '',
      loadComponent: () => import('./routes/home.page'),
    },
    {
      path: 'bookings/:slug',
      loadComponent: () => import('./routes/bookings/bookings.page'),
    },
]
```

```
// homePage activity list
@Component({
  standalone: true,
  imports: [CurrencyPipe, DatePipe, RouterLink],
  template: ``,
  styles: ``,
  changeDetection: ChangeDetectionStrategy.OnPush,
})
export default class HomePage {
  activities = ACTIVITIES;
}
```

```
<article>
```

```
  <header>
    <h2>Activities</h2>
  </header>
  <main>
    @for (activity of activities; track activity.id) {
    <div>
      <span>
        <a [routerLink]="['/bookings', activity.slug]">{{ activity.name }}</a>
      </span>
      <span>{{ activity.location }}</span>
      <span>{{ activity.price | currency }}</span>
      <span>{{ activity.date | date : "dd-MMM-yyyy" }}</span>
    </div>
    }
  </main>
</article>
```

4.2.2 Recepción reactiva de parámetros como señales

```
provideRouter(routes, withComponentInputBinding())

// slug input en BookingsPage
slug = input<string>();

activity = computed(
    () => ACTIVITIES.find((a) => a.slug === this.slug()) || NULL_ACTIVITY,
);
```

```
// full reactive signal based component
export default class BookingsPage {
  slug = input<string>();
  activity = computed(() => ACTIVITIES.find((a) => a.slug === this.slug()) ||
NULL_ACTIVITY);
  alreadyParticipants = computed(() => Math.floor(Math.random() *
this.activity().maxParticipants));
  maxNewParticipants = computed(() => this.activity().maxParticipants -
this.alreadyParticipants());
  isBookable = computed(() => ["published",
"confirmed"].includes(this.activity().status));

  newParticipants = signal(0);
  booked = signal(false);
  participants = signal<{ id: number }[]>([]);

  totalParticipants = computed(() => this.alreadyParticipants() +
this.newParticipants());
  remainingPlaces = computed(() => this.activity().maxParticipants -
this.totalParticipants());
```

```
  bookingAmount = computed(() => this.newParticipants() * this.activity().price);

  bookedMessage = computed(() => {
    if (this.booked()) return `Booked USD ${this.bookingAmount()}`;
    return "";
  });

  constructor() {
    effect(
      () => {
        this.participants.update((participants) => {
          participants.splice(0, participants.length);
          for (let i = 0; i < this.totalParticipants(); i++) {
            participants.push({ id: participants.length + 1 });
          }
          return participants;
        });
      },
      {
        allowSignalWrites: true,
      }
    );
    effect(() => {
      if (!this.isBookable()) {
        return;
      }
      const totalParticipants = this.totalParticipants();
      const activity = this.activity();
      let newStatus = activity.status;
      if (totalParticipants >= activity.maxParticipants) {
        newStatus = "sold-out";
      } else if (totalParticipants >= activity.minParticipants) {
        newStatus = "confirmed";
      } else {
        newStatus = "published";
      }
      activity.status = newStatus;
    });
  }

  onNewParticipantsChange(newParticipants: number) {
    if (newParticipants > this.maxNewParticipants()) {
      newParticipants = this.maxNewParticipants();
    }
    this.newParticipants.set(newParticipants);
  }

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

```
@if (activity(); as activity) {
  <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>Max Participants: {{ activity.maxParticipants }}</div>
      <ul>
        <li>New Participants: {{ newParticipants() }}</li>
        <li>Remaining places: {{ remainingPlaces() }}</li>
        <li>Total participants: {{ totalParticipants() }}</li>
      </ul>
      <div>
        @for (participant of participants(); track participant.id) {
          <span [attr.data-tooltip]="participant.id">🏃</span>
        } @empty {
          <span>🕸️</span>
        }
      </div>
    </main>
    <footer>
      @if (isBookable()) {
        <h4>New Bookings</h4>
        @if (remainingPlaces() > 0) {
          <label for="newParticipants">How many participants want to book?</label>
          <input
            type="number"
            name="newParticipants"
            [ngModel]="newParticipants()"
            (ngModelChange)="onNewParticipantsChange($event)"
            min="0"
            [max]="maxNewParticipants()"
          />
        } @else {
          <div>
            <button class="secondary outline" (click)="onNewParticipantsChange(0)">
              Reset
            </button>
            <span>No more places available</span>
          </div>
        }
        <button [disabled]="booked() || newParticipants() === 0"
(click)="onBookClick()">
          Book {{ newParticipants() }} places now for {{ bookingAmount() | currency }}!
        </button>
        <div>{{ bookedMessage() }}</div>
      }
    </footer>
  </article>
}
```

4.3 SEO y Server Side Rendering.

4.3.1 SPA y navegación local y offline

```
// navegación local desconectada
# dev mode
npm start
# chunks
// primero server, después browser
```

4.3.2 Indexación de contenido SSR

```

# build and node serve
npm run build
npm run serve:ssr:ActivityBookings
# full
npm run serve

```