Práctica 5: Aplicación de Gestión de Tareas con Navegación y Carga Diferida

Kirbi Xavier Huerta Salinas

Octubre 2025

1 Introducción

El presente documento detalla el desarrollo de una aplicación móvil para la gestión de tareas, construida con Ionic y Angular. El objetivo de la práctica es aplicar conceptos clave de desarrollo de aplicaciones móviles, como la creación de componentes, navegación, persistencia de datos y optimización de carga.

La aplicación permite a los usuarios crear, visualizar, editar y eliminar tareas, las cuales se almacenan localmente en el dispositivo para garantizar la persistencia de la información entre sesiones.

2 Cumplimiento de Requisitos

La aplicación implementa todos los requisitos solicitados en el documento de la práctica:

- **Creación de Páginas:** Se crearon tres páginas principales: `Home` (para listar tareas), `Tarea` (para crear una nueva tarea) y `TareaDetalles` (para editar una tarea existente).
- **Navegación:** Se configuró un sistema de rutas en Angular para navegar entre las páginas mencionadas.
- **Lazy Loading:** Todas las rutas de las páginas se implementaron con carga diferida (lazy loading) utilizando la función `loadComponent` para optimizar el rendimiento inicial de la aplicación.
- **Servicio y Persistencia:** Se desarrolló un servicio (`TareaService`) que centraliza la lógica de negocio (CRUD) y utiliza el `localStorage` del navegador para guardar y recuperar las tareas.
- **Diseño de UI:** La interfaz es sencilla y funcional, utilizando los componentes de Ionic para una experiencia de usuario nativa.
- **Validaciones:** El formulario de creación y edición de tareas incluye validaciones para campos requeridos y para asegurar que la fecha de vencimiento sea siempre una fecha futura.

3 Estructura y Organización del Proyecto

El proyecto sigue la estructura estándar de una aplicación de Angular, con los componentes de la aplicación ubicados en la carpeta `src/app`. Las páginas, servicios y modelos de datos se organizaron en sus respectivas carpetas para mantener el código ordenado y mantenible.

- `src/app/pages/`: Contiene los componentes de página (`TareaPage`, `TareaDetallesPage`).
- `src/app/services/`: Contiene el servicio de lógica de negocio (`TareaService`).
- `src/app/models/`: Define la interfaz del modelo de datos (`Tarea`).
- `src/app/validators/`: Contiene validadores de formularios personalizados.

4 Detalles de Implementación

4.1 Uso de Lazy Loading

La carga diferida se configuró en `src/app/app.routes.ts`. Cada ruta utiliza `loadComponent` para cargar los componentes de página solo cuando son necesarios.

```
import { Routes } from '@angular/router';

export const routes: Routes = [
  {
    path: 'home',
    loadComponent: () => import('./home/home.page').then((m) => m.HomePage),
  },
  {
    path: 'tarea',
    loadComponent: () => import('./pages/tarea/tarea.page').then( m => m.
        TareaPage)
  },
  {
    path: 'tarea-detalles/:id',
    loadComponent: () => import('./pages/tarea-detalles/tarea-detalles.page').
        then( m => m.TareaDetallesPage)
  },
];
```

4.2 Persistencia con LocalStorage

El servicio `TareaService` es responsable de interactuar con `localStorage` para persistir los datos. Los métodos `saveTareas` y `getTareas` se encargan de la serialización y deserialización de los datos de las tareas.

```
@Injectable({
  providedIn: 'root'
})
export class TareaService {
  private readonly storageKey = 'tareas';

  getTareas(): Tarea[] {
    const tareasJson = localStorage.getItem(this.storageKey);
    if (tareasJson) {
      return JSON.parse(tareasJson).map((tarea: any) => ({
        ...tarea,
        fechaVencimiento: new Date(tarea.fechaVencimiento)
      }));
    }
    return [];
  }

  private saveTareas(tareas: Tarea[]): void {
    localStorage.setItem(this.storageKey, JSON.stringify(tareas));
  }
  // ... otros metodos CRUD
}
```

4.3 Validaciones de Formulario

Se utilizó un validador personalizado (`fechaFuturaValidator`) para el campo de fecha, asegurando que el usuario no pueda seleccionar una fecha que ya ha pasado. Este validador se combina con los validadores estándar de Angular para campos requeridos.

```
// En el componente TareaPage o TareaDetallesPage
this.tareaForm = this.fb.group({
  titulo: ['', Validators.required],
  descripcion: ['', Validators.required],
  fechaVencimiento: [null, [Validators.required, fechaFuturaValidator]]
});

// Mensaje de error en el HTML
```

```
<div *ngIf="tareaForm.get('fechaVencimiento')?.hasError('fechaPasada')">
  La fecha debe ser futura.
</div>
```

5 Capturas de Pantalla

Figure 1: Página principal con la lista de tareas.

6 Compilación para Android con Capacitor

Para empaquetar la aplicación para Android, se utilizó Capacitor. Este proceso convierte la aplicación web en un proyecto nativo de Android que puede ser gestionado con Android Studio.

Los pasos seguidos fueron los siguientes:

1. **Instalar dependencias de Capacitor:**

   ```
   npm install @capacitor/core @capacitor/cli
   ```

2. **Inicializar Capacitor:** Se crea el archivo de configuración principal.

   ```
   npx cap init
   ```

Figure 2: Formulario de creación de una nueva tarea.

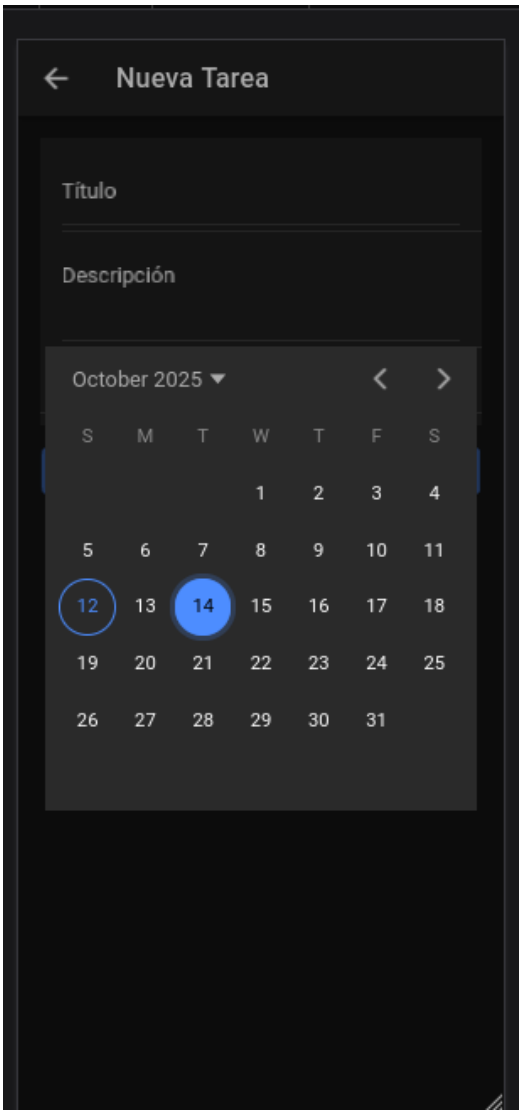

Figure 3: ion-datetime para seleccionar la fecha de vencimiento.

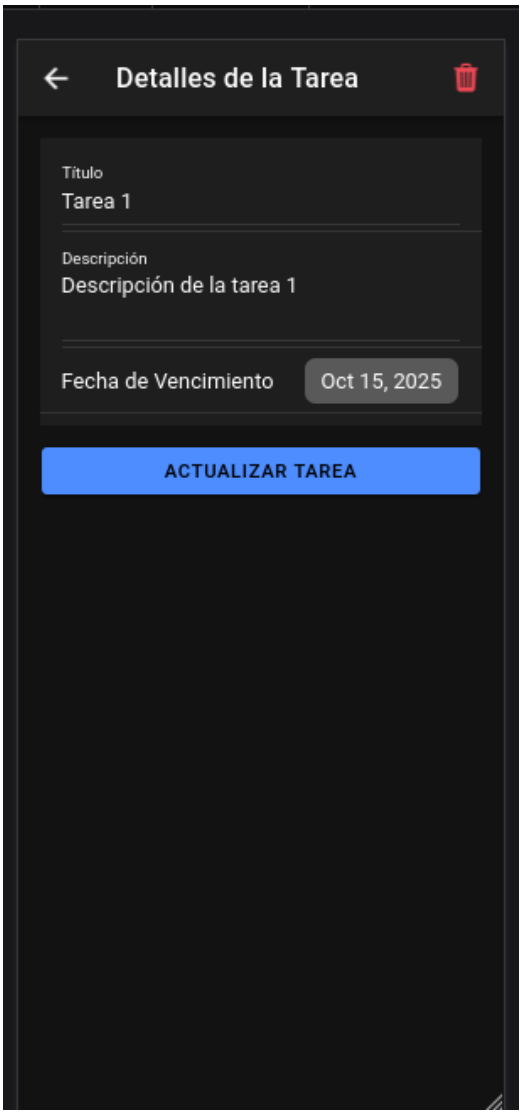

Figure 4: Formulario de edición de una tarea existente.

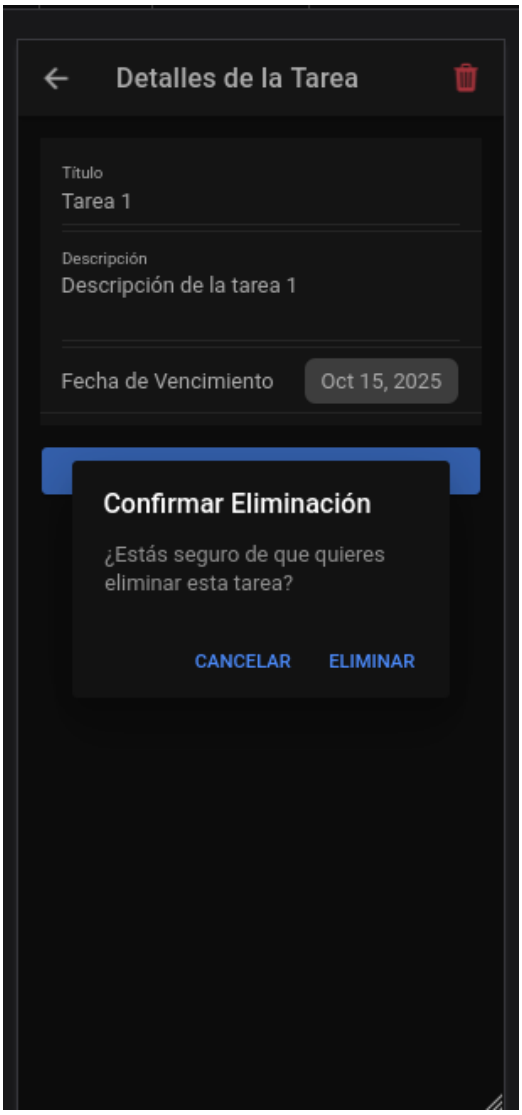

Figure 5: Mensaje al querer eliminar una tarea.

3. **Construir la aplicación web:** Se generan los archivos estáticos del proyecto de Angular.

   ```
   npm run build
   ```

4. **Añadir la plataforma Android:** Se crea el proyecto nativo de Android.

   ```
   npx cap add android
   ```

5. **Sincronizar los recursos web:** Se copian los archivos web al proyecto nativo.

   ```
   npx cap sync
   ```

6. **Abrir en Android Studio:** Finalmente, se abre el proyecto en el IDE de Android para compilar y ejecutar.

   ```
   npx cap open android
   ```

7 Demostración en Video

Se grabó un video que muestra la aplicación funcionando en un dispositivo móvil Android. El video demuestra la creación, edición, y eliminación de tareas, así como la correcta funcionalidad de las validaciones y la persistencia de datos.

El video se puede visualizar en el siguiente enlace: Ver video de la aplicación en funcionamiento