BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

MATERIA: PROGRAMACIÓN DE DISPOSITIVOS MOVILES

PRÁCTICA 3 APLICACIÓN DE NOTICIAS CON IONIC 6

M.C. LUIS YAEL MENDEZ – SANCHEZ

ALUMNOS: JORGE LEON VIVANCO
GUSTAVO ALFREDO AGUILAR GUERRERO

INTERPERIODO 2025

INTRODUCCIÓN

El desarrollo de aplicaciones móviles ha cobrado gran importancia en la actualidad, especialmente cuando se trata de crear soluciones dinámicas, responsivas y conectadas a servicios externos. En esta práctica, el objetivo fue construir una aplicación de noticias completamente funcional utilizando Ionic 6 y Angular, combinando lo aprendido en el curso con herramientas modernas como el consumo de APIs, el manejo del almacenamiento nativo, y la implementación de componentes personalizados.

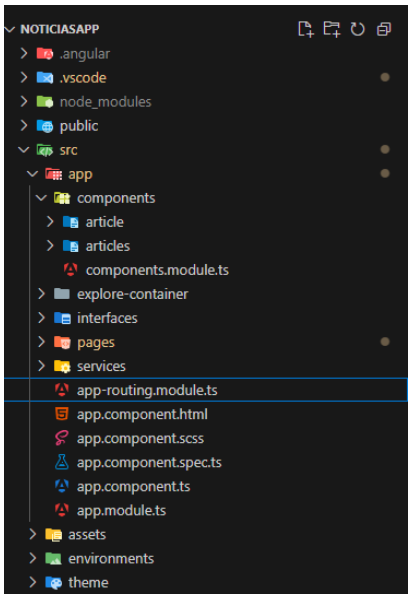
Durante el desarrollo de esta aplicación, tuvimos la oportunidad de enfrentarnos a un proyecto que simula el flujo real de trabajo en una empresa de software, desde la estructura inicial del proyecto, el consumo eficiente de una API (en este caso, NewsAPI.org), hasta la integración de funcionalidades como Infinite Scroll, segmentación por categorías, diseño adaptativo y el guardado local de contenido.

Además de enfocarnos en lo técnico, también nos centramos en ofrecer una buena experiencia de usuario, cuidando la navegación, el diseño responsivo y los detalles visuales con los componentes propios de Ionic. Otro aspecto fundamental fue hacer que la app pudiera ejecutarse tanto en dispositivos móviles reales usando Capacitor, como en navegadores mediante su versión como PWA (Progressive Web App). Todo esto nos permitió aplicar habilidades prácticas de programación orientadas a productos reales.

DESARROLLO

1. Estructura del Proyecto

La aplicación se organiza de forma modular para facilitar el mantenimiento y escalabilidad:



src/app/components/

1. *article/* y *articles/*: Componentes reutilizables para mostrar noticias individuales y listados.
2. *explore-container/*: Componente base inicial.

- **src/app/pages/**

1. *tab1, tab2, tab3, tabs/*: Representan distintas categorías o secciones de navegación.

- **src/app/services/**

1. *news.service.ts*: Se encarga de consumir la API de noticias.
2. *storage.service.ts*: Manejo del almacenamiento local usando Ionic Storage.

- **src/app/interfaces/**

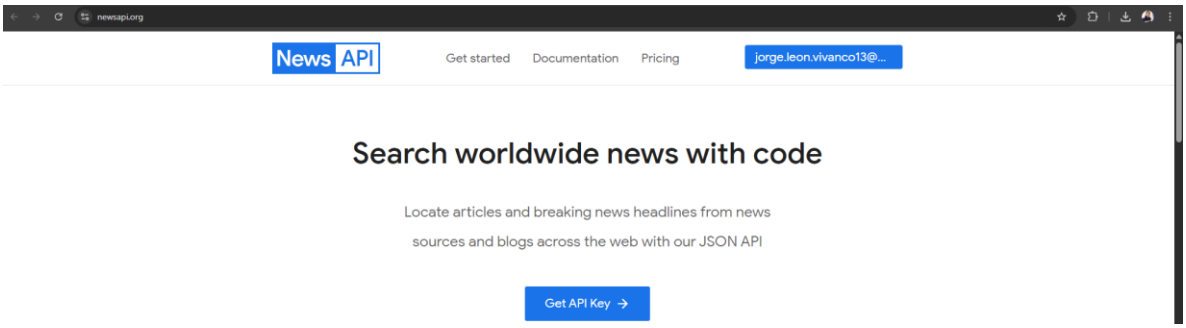
1. *index.ts*: Define interfaces para tipar correctamente las respuestas de la API.

- **theme, assets, environments**: Configuración de estilo global, recursos estáticos y variables de entorno.

2. Desarrollo e Implementación

1. Consumo de API

Utilizamos la API de NewsAPI.org para obtener las noticias. Mediante el servicio news.service.ts, realizamos peticiones HTTP y manipulamos los resultados con interfaces personalizadas.



2. Componentes personalizados

Creamos componentes article y articles para mostrar de forma dinámica y reutilizable los datos de cada noticia. Estos componentes manejan la presentación visual y la lógica mínima para recibir @Input() desde las páginas principales.

```
<ion-grid fixed>
    <ion-row>
      <ion-col *ngFor=" let article of articles; let i = index"
      size="12" sizeLg="3" sizeMd="4" sizeSm="6" sizeXs="12">
            <app-article [article]="article" [index]= "i + 1"> </app-article>
      </ion-col>
    </ion-row>
  </ion-grid>
```

3. Diseño Responsivo

Se implementaron estilos en global.scss y en los componentes para asegurar compatibilidad visual en móviles, tabletas y escritorios, aprovechando la grilla de Ionic y el layout flexible con CSS.

global.scss > ...

```
/* Core CSS required for Ionic components to work properly */
@import "@ionic/angular/css/core.css";

/* Basic CSS for apps built with Ionic */
@import "@ionic/angular/css/normalize.css";
@import "@ionic/angular/css/structure.css";
@import "@ionic/angular/css/typography.css";
@import "@ionic/angular/css/display.css";

/* Optional CSS utils that can be commented out */
@import "@ionic/angular/css/padding.css";
@import "@ionic/angular/css/float-elements.css";
@import "@ionic/angular/css/text-alignment.css";
@import "@ionic/angular/css/text-transformation.css";
@import "@ionic/angular/css/flex-utils.css";

/**
 * Ionic Dark Mode
 * -----------------------------------------------------
 * For more info, please see:
 * https://ionicframework.com/docs/theming/dark-mode
 */

/* @import "@ionic/angular/css/palettes/dark.always.css"; */
/* @import "@ionic/angular/css/palettes/dark.class.css"; */
@import "@ionic/angular/css/palettes/dark.system.css";


.text-primary {
    color: rgb(8, 132, 234);
}

.article-source-name{
        font-size: 12px;
```

4. Infinite Scroll

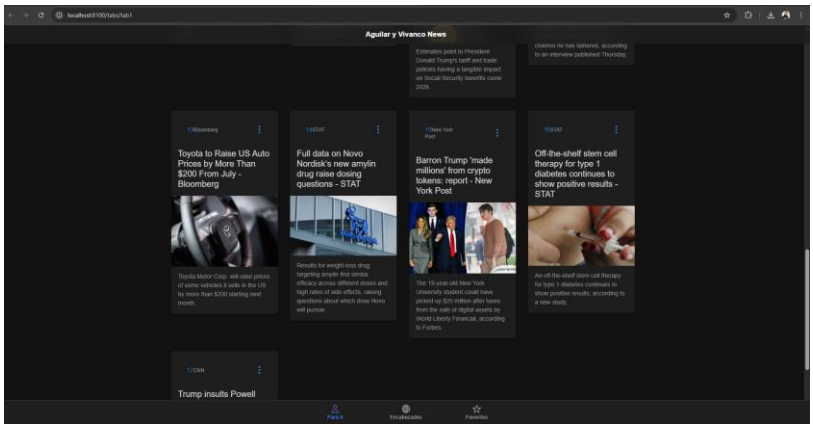
Usamos <ion-infinite-scroll> para cargar noticias adicionales al llegar al final del scroll. Esto se enlaza con la paginación de la API, actualizando los datos sin recargar la vista.

6. Segmentos con Scroll

En las pestañas principales se integró <ion-segment> con scroll para que el usuario pueda filtrar noticias por categorías (tecnología, ciencia, salud, etc.).

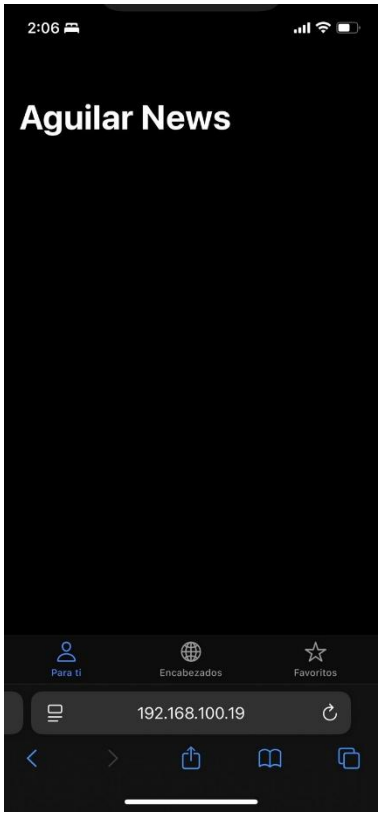
7. Almacenamiento local

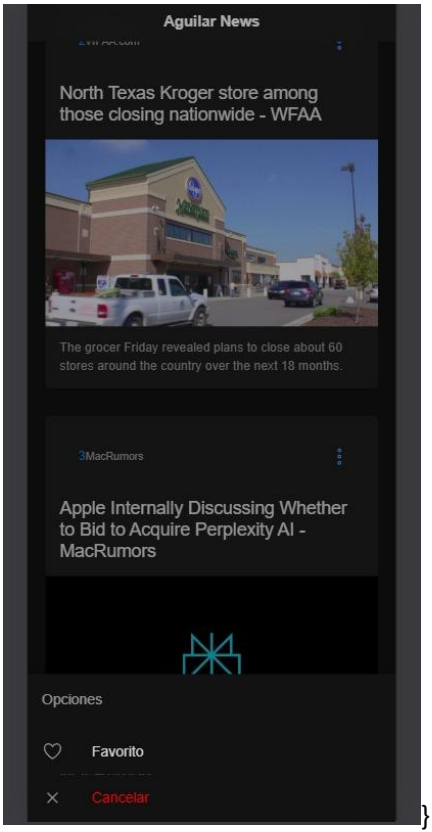
Mediante storage.service.ts y el plugin de Ionic Storage, almacenamos noticias marcadas como favoritas. Esto persiste incluso tras cerrar la app, y evita duplicados con validaciones previas.



8. Plugins (Capacitor)

Probamos la app en dispositivo Android real, integrando Capacitor para compilarla como aplicación nativa. También preparamos la versión PWA.





}

CONCLUSION

Esta práctica representó un ejercicio completo de desarrollo móvil moderno, en el que logramos llevar a cabo una app de noticias que no solo consume datos de forma externa, sino que también responde a la interacción del usuario con fluidez y eficiencia. A lo largo del proyecto aplicamos técnicas fundamentales como la separación por módulos y componentes, la integración de servicios, el uso de interfaces, y el almacenamiento persistente de datos, lo que en conjunto contribuyó a una arquitectura limpia y escalable.

Uno de los mayores aprendizajes fue el uso del almacenamiento local con Ionic Storage, que nos permitió guardar y recuperar noticias favoritas incluso después de cerrar la app, así como implementar mecanismos de validación para evitar duplicados. Además, el uso de elementos como toasts, action-sheets y alerts nos ayudó a mejorar la comunicación con el usuario, aportando valor a la experiencia final.

Otro punto importante fue el diseño responsivo. Al usar los estilos de Ionic y buenas prácticas de CSS, logramos una aplicación que funciona y se adapta correctamente a teléfonos, tabletas y pantallas más grandes. La implementación de segmentos con scroll y el infinite-scroll también nos permitieron brindar una experiencia más fluida y organizada para la navegación por categorías.

REFERENCIAS BIBLIOGRÁFICAS

- Ionic Framework. (s.f.). *Ionic Documentation*. Recuperado el 10 de junio de 2025, de https://ionicframework.com/docs
- Angular. (s.f.). *Angular Documentation*. Recuperado el 10 de junio de 2025, de https://angular.io/docs
- CapacitorJS. (s.f.). *Capacitor Documentation*. Recuperado el 10 de junio de 2025, de https://capacitorjs.com/docs
- Ionic Team. (2022). *Ionic 6.0 Release Notes*. Recuperado de https://ionicframework.com/blog/announcing-ionic-v6
- Wieruch, R. (2019). *The Road to React: Your journey to master plain yet pragmatic React.js*. Independently published. *(Referente a buenas pr⟨cticas en componentes reutilizables, aplicables tambi⌈n en Angular/Ionic)*
- Freeman, A., & Sanderson, P. (2020). *Pro Angular 9*. Apress. https://doi.org/10.1007/978-1-4842-6037-2
- Hartl, M. (2020). *Web Development with JavaScript and the Ionic Framework*. Learn Enough. Recuperado de https://www.learnenough.com/