
Hustle Hub

Aplicación para permitir la gestión y creación de eventos



TRABAJO FINAL DE CICLO

CFGS DESARROLLO DE APLICACIONES WEB

Autor/a: Ángel Martín Rubio

Tutor/a: Tomás Martínez Ruiz

Junio de 2025

Índice

Capítulo 1: Introducción y Objetivos	3
1.1 Objetivos	3
1.2 Ejemplo	4
Capítulo 2: Especificación de Requisitos	4
2.1 Bocetos , prototipos	4
2.2 Funcionalidades	4
Capítulo 3: Planificación Temporal y Evaluación de Costes	5
3.1 Tiempo empleado	5
3.1.1 Diseño	5
3.1.2 Configuración del proyecto	6
3.1.3 Autenticación	6
3.1.4 Gestión de eventos	6
3.1.5 Diseño responsive	6
3.1.6 Testing y ajustes	6
3.1.7 Despliegue	7
3.2 Evaluación de costes	7
Capítulo 4: Tecnologías Utilizadas	8
4.1 Firebase	8
4.2 Angular	9
4.3 Bootstrap	10
4.4 Ng-Bootstrap	11
4.5 RXJS	12
4.6 JWT	13
4.7 GitHub Actions	13
Capítulo 5: Desarrollo e Implementación	14
5.1 Integración de Angular y Firebase	14
5.2 Autenticación	15
5.3 Modal genérico	16
5.4 Slider de iconos	17
5.5 Offcanvas como dashboard	18
5.6 Servicio de inactividad	20
5.7 Routing	21
5.8 Recogida de datos de firebase	22
5.9 Inserción de datos	23
5.10 Gestión de suscripciones	25

5.11 Gestión de eventos de usuario	28
Capítulo 6: Conclusiones y Líneas Futuras	30
Conclusiones	30
Líneas Futuras	30

Capítulo 1: Introducción y Objetivos

HustleHub es una aplicación web pensada para gestionar eventos de distintos tipos. Viendo el problema actual que tenemos las personas a la hora de buscar un evento que nos guste entre las distintas plataformas cada una para un estilo u otro , diferentes artistas etc..

HustleHub busca unir todo en uno , sin distinciones de tipo de eventos , todos tienen su lugar! .

Los artistas tendrán al alcance de su mano las personas que van a sus eventos para motivarlas aún mas .

1.1 Objetivos

- 1.Creación de una interfaz amigable para la navegación del usuario.
- 2.Acercar esta aplicación a creadores de eventos.
- 3.Facilitar la creación de eventos de manera online.
- 4.Aprender nuevas tecnologías no impartidas en el ciclo

1.2 Ejemplo

Pensemos en la típica gestora de nuestro pueblo o ciudad que simplemente quiere dar una charla de por qué los seguros que ofrece son mejores que otros, pero que no dispone ni de los medios ni del tiempo para configurar plataformas complejas. Aquí es donde HustleHub marca la diferencia: frente a otras aplicaciones de gestión de eventos, HustleHub ofrece una solución mucho más amigable, pensada específicamente para pequeños comercios y profesionales locales. Su interfaz intuitiva, la facilidad para publicar eventos y la ausencia de configuraciones complicadas hacen que cualquier persona, sin necesidad de conocimientos técnicos, pueda crear, promocionar y gestionar sus eventos en cuestión de minutos.

Capítulo 2: Especificación de Requisitos

2.1 Bocetos , prototipos

<https://www.figma.com/design/YGgiEHafC838rYWVMg8bRr/HustleHub?node-id=0-1&t=gnBN5kGPHCVfu7Mf-1>

Wireframe preeliminar de como tiene que ser la aplicación

2.2 Funcionalidades

2.2.1 Funcionales

Estas son características que describen lo que el sistema hace:

1. Iniciar sesión con correo electrónico y contraseña.
2. Iniciar sesión mediante Google (OAuth 2.0).
3. Registro de nuevos usuarios (con correo y contraseña).
4. Cierre de sesión seguro.
5. Visualizar listado de eventos disponibles.
6. Filtros por categoría de evento (educación, viajes, videojuegos, cine, etc.).

7. Acceso rápido a eventos destacados o favoritos.
8. Añadir eventos a "favoritos" para guardarlos y consultarlos luego.

2.2.2 No funcionales

Estas describen cómo debe comportarse el sistema o sus restricciones técnicas:

1. Navegación rápida y responsive (adaptado a dispositivos móviles y escritorio).
2. Datos de usuarios y eventos almacenados en Firebase Realtime Database.
3. Imágenes de eventos almacenadas mediante URLs en la base de datos.

Capítulo 3: Planificación Temporal y Evaluación de Costes

3.1 Tiempo empleado

A continuación se va a estimar el tiempo que pueden llevar cada tarea

3.1.1 Diseño

6 horas en :

- Diseño de wireframes básicos
- Prototipo de pantallas principales (inicio, login, registro, listado de eventos, detalle de evento, favoritos)

3.1.2 Configuración del proyecto

5 horas en :

- Crear proyecto en Angular
- Integrar Firebase (Authentication + Realtime Database)
- Configuración inicial de rutas y servicios

3.1.3 Autenticación

6 horas en:

- Implementar login con correo/contraseña
- Implementar login con Google
- Implementar registro de usuario
- Cierre de sesión

3.1.4 Gestión de eventos

8 horas en:

- Mostrar eventos en la vista principal
- Crear filtros por categoría y búsqueda
- Favoritos de usuario

3.1.5 Diseño responsive

5 horas en :

- Ajustar estilos para móvil y escritorio
- Integrar Bootstrap para los componentes UI

3.1.6 Testing y ajustes

5 horas en:

- Testear flujos de login, registro, favoritos, filtros
- Corrección de errores menores
- Mejoras de usabilidad

3.1.7 Despliegue

5 horas en :

- Configurar hosting en Firebase
- Configurar Github actions
- Implementar CI/CD

3.2 Evaluación de costes

Frameworks y tecnologías	Angular, Bootstrap, TypeScript: software libre	0 €
Diseño	Figma gratuito	0 €
Backend Servicios	Firebase Authentication, Realtime Database y Hosting (plan Spark gratuito)	0 €
Despliegue	Firebase Hosting gratuito en plan Spark (hasta 1 GB almacenamiento/10 GB de tráfico mensual)	0 €
Herramientas Colaborativas	GitHub	0 €
Costes totales		0 €

Capítulo 4: Tecnologías Utilizadas

La aplicación se realizará en Angular 16 con ngBootstrap , bootstrap además de guardar los datos en Firebase junto al hosting y despliegue .

4.1 Firebase



A la hora de plantearme cómo construir HustleHub, una de las decisiones más importantes fue elegir qué herramientas utilizar para que el proyecto fuera rápido de desarrollar, sencillo de mantener y al mismo tiempo potente. Ahí es donde apareció Firebase.

Firebase es una plataforma creada por Google que ofrece un montón de servicios listos para usar: login de usuarios, base de datos en tiempo real, alojamiento web, almacenamiento de archivos y más. La gran ventaja es que todo esto funciona sin tener que montar un servidor propio o complicarse con configuraciones difíciles, algo que me venía perfecto porque mi objetivo era centrarme en el desarrollo del frontend.

Para HustleHub, quería que el usuario pudiera registrarse, iniciar sesión, guardar sus eventos favoritos y navegar de forma rápida, sin tener que esperar a grandes tiempos de carga o sufrir errores. Firebase me permite tener toda esta parte montada de forma segura y escalable, usando solo herramientas del lado del cliente.

Además, su plan gratuito (el Spark Plan) cubre de sobra todo lo que necesita una aplicación como HustleHub en su primera fase: autenticación de usuarios, base de datos para los eventos, y hosting gratuito para desplegar la aplicación online. Todo esto, sin tener que pagar servidores externos ni complicarme con un backend propio.

Otra razón de peso fue la facilidad de integración con Angular, el framework que elegí para construir el proyecto. Firebase y Angular trabajan muy bien juntos, lo que hizo que todo el proceso de conexión entre la app y la base de datos fuera rápido y fluido.

En resumen, elegí Firebase porque me permitía centrarme en lo más importante: construir una aplicación pensada para usuarios reales, pequeña en complejidad pero grande en funcionalidad. Y todo sin perder tiempo ni recursos en tareas que Firebase ya resuelve de forma automática.

4.2 Angular



Desde el primer momento que empecé a plantear el desarrollo de HustleHub, tenía claro que necesitaba un framework que fuera sólido, moderno y que me permitiera crear una aplicación rápida, escalable y fácil de mantener. Después de valorar varias opciones, me decidí por Angular, y en concreto, por su versión 16.

Una de las principales razones para elegir Angular 16 es que actualmente es el framework que se utiliza en la empresa donde estoy realizando mis prácticas. Esto me permitió aprovechar todo el conocimiento y experiencia que estoy adquiriendo allí, aplicándolo directamente a mi proyecto. Además, me ayudó a mantener una continuidad en las tecnologías que utilizo en mi formación profesional, lo cual considero fundamental para seguir creciendo como desarrollador.

Angular 16 me ofrecía justo lo que buscaba: un entorno de trabajo estructurado, donde es muy fácil organizar el proyecto en módulos, componentes y servicios. Esta organización me ha ayudado muchísimo a mantener el código limpio y a que la aplicación sea fácil de ampliar o modificar en el futuro. Además, Angular incorpora herramientas muy potentes de serie,

como el enrutamiento, la gestión de formularios o el control del estado de los datos, lo que me ha ahorrado tener que añadir librerías extra o complicar la arquitectura.

Una de las grandes mejoras que trae Angular 16 es la optimización del rendimiento, sobre todo con las nuevas funciones reactivas y la gestión de cambios más eficiente. Esto era especialmente importante en HustleHub, ya que quería que la navegación entre pantallas fuera fluida y que el usuario sintiera que estaba usando una aplicación moderna y rápida, incluso aunque la conexión no fuera la mejor.

Otro punto clave fue la facilidad de integración de Angular con Firebase. Gracias a librerías oficiales como AngularFire, conectar la app con la autenticación de usuarios, la base de datos y el hosting fue un proceso bastante sencillo y seguro.

Finalmente, Angular también me aportó otra ventaja: el ecosistema de herramientas de desarrollo. El uso de Angular CLI para generar componentes y servicios, y las posibilidades de testing automático, me permitieron trabajar de manera mucho más rápida y ordenada.

En resumen, elegí Angular 16 porque es el framework que mejor se adapta a proyectos modernos como HustleHub, y además porque es el mismo entorno que utilizo en mis prácticas profesionales, lo que me ha permitido aplicar mis conocimientos de forma práctica y coherente.

4.3 Bootstrap



Poco de qué hablar sobre Bootstrap.

En este proyecto, solamente decidí, por temas que se detallan más adelante, utilizar únicamente los **estilos de Bootstrap**, sin necesidad de usar el JavaScript que viene con sus componentes ya preestablecidos.

Bootstrap 5 es uno de los frameworks CSS más utilizados hoy en día para diseñar

páginas web de forma rápida y ordenada. Su principal ventaja es que ofrece un sistema de grillas (grid), clases utilitarias y componentes de interfaz como botones, tarjetas, formularios y alertas, listos para aplicar directamente en el HTML. Además, su última versión permite trabajar de forma completamente responsiva sin depender de librerías externas como jQuery, lo que hace que sea más ligero y fácil de integrar en proyectos modernos.

En HustleHub, se tomó la decisión de usar únicamente los estilos de Bootstrap 5 porque, para este tipo de proyecto, era suficiente aprovechar sus clases de diseño sin cargar todo el JavaScript adicional que Bootstrap proporciona. De esta forma, se mantiene el peso de la aplicación bajo control y se evita añadir funcionalidades que luego serían gestionadas de una forma más adaptada a Angular.

Más adelante, para los pocos componentes que sí necesitaban interacción como modales o offcanvas, se explica brevemente la integración de **ng-bootstrap**, una librería que adapta los componentes de Bootstrap para funcionar de forma nativa dentro de Angular, sin necesidad de depender de JavaScript externo.

4.4 Ng-Bootstrap



Siguiendo con la parte visual del proyecto, llegó un momento en el que fue necesario utilizar algunos componentes interactivos como modales, offcanvas o acordeones. Aquí es donde entra en juego **ng-bootstrap**.

Ng-bootstrap es una librería que adapta los componentes de Bootstrap para que funcionen de manera nativa dentro de aplicaciones Angular. Es decir, no necesita jQuery ni el JavaScript clásico de Bootstrap; todo está construido directamente en Angular, respetando su forma de trabajar con componentes, servicios e inyección de dependencias. Esto facilita mucho la integración,

sobre todo cuando se busca mantener el proyecto ligero y totalmente adaptado al ecosistema Angular.

En el caso de HustleHub, se decidió utilizar ng-bootstrap principalmente porque ya tenía Bootstrap 5 para los estilos, y no tenía sentido cargar scripts externos cuando podía usar una solución pensada para Angular desde el principio. De esta forma, los pocos componentes dinámicos que necesitaba el proyecto, como el offcanvas del usuario o algún modal informativo, pudieron integrarse de forma sencilla, limpia y sin romper la estructura del proyecto.

Además, ng-bootstrap ofrece un control total sobre el comportamiento de los componentes, permitiendo personalizar su funcionamiento mediante directivas y propiedades, algo que encajaba perfectamente con la idea de mantener el proyecto flexible y escalable.

4.5 RXJS



RxJS es una de las bibliotecas más populares en el desarrollo web actual. Ofrece un enfoque potente y funcional para gestionar eventos y puntos de integración en un número creciente de frameworks, bibliotecas y utilidades, lo que hace que aprender Rx sea más atractivo que nunca. Si a esto le sumamos la posibilidad de aplicar tus conocimientos en prácticamente cualquier lenguaje, dominar la programación reactiva y sus múltiples funciones parece una obviedad. Aprender RxJS y programación reactiva es difícil . Hay multitud de conceptos, una amplia superficie de API y un cambio fundamental en la mentalidad, de un estilo imperativo a uno declarativo . En HustleHub se utiliza bastante para los servicios como el de autenticación o el de categorías

4.6 JWT



En HustleHub, la autenticación de usuarios se realiza utilizando Firebase Authentication, que permite registrar e iniciar sesión mediante correo y contraseña o a través de proveedores externos como Google. Una vez el usuario se identifica correctamente, Firebase genera de forma automática un token de sesión en formato JWT (JSON Web Token). Este token contiene información codificada sobre la identidad del usuario y se utiliza para autenticar todas las operaciones que ese usuario realice desde la aplicación. Este proceso ocurre

de forma completamente transparente para el desarrollador. Es decir, no es necesario gestionar manualmente las sesiones ni los tokens, ya que el SDK de Firebase se encarga de: Generar el token JWT al iniciar sesión.

Enviar este token en cada petición a la base de datos (por ejemplo, al leer o escribir datos).
Verificar que el token sea válido.

Refresca automáticamente cuando expira.

Esto significa que cada vez que un usuario consulta sus eventos, favoritos o cualquier otra información protegida, Firebase comprueba que su token JWT sea válido y que tenga permisos para acceder a esa parte de la base de datos. De esta forma, se garantiza tanto la seguridad como la integridad del acceso, sin necesidad de montar un sistema de autenticación personalizado.

4.7 GitHub Actions



GitHub Actions

GitHub Actions es una plataforma de integración y despliegue continuos (IC/DC) que te permite automatizar tu mapa de

compilación, pruebas y despliegue. Puedes crear flujos de trabajo y crear y probar cada solicitud de cambios en tu repositorio o desplegar solicitudes de cambios fusionadas a producción.

GitHub Actions va más allá de solo DevOps y te permite ejecutar flujos de trabajo cuando otros eventos suceden en tu repositorio. Por ejemplo, puedes ejecutar un flujo de trabajo para que agregue automáticamente las etiquetas adecuadas cada que alguien cree una propuesta nueva en tu repositorio.

GitHub proporciona máquinas virtuales Linux, Windows y macOS para que ejecutes tus flujos de trabajo o puedes hospedar tus propios ejecutores auto-hospedados en tu propio centro de datos o infraestructura en la nube.

Capítulo 5: Desarrollo e Implementación

5.1 Integración de Angular y Firebase

Para poder comunicarse fue necesario realizar unos ajustes en el proyecto para que funcionase correctamente

```
you, last week | 2 authors (LENOVO-LMBARKERlamartin and one other)
@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    FooterComponent,
    InactividadComponent
  ],
  imports: [
    BrowserModule,
    provideFirebaseApp(() => initializeApp(environment.firebaseConfig)),
    provideAuth(() => getAuth()),
    provideDatabase(() => getDatabase()),
    AppRoutingModule,
    HttpClientModule,
    FeaturesModule,
    NgbModule,
    CoreModule,
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

En nuestro modulo app debemos llamar al inicio de la aplicación cogiendo las variables de entorno definidas en environment

```
export const environment = {
  production: false,
  firebaseConfig: {
    apiKey: 'AIzaSyCBIIvrdprkiZ3AmW-HjxZP-wMPzMKfjME',
    authDomain: 'hustlehub-55746.firebaseio.com',
    databaseURL: 'https://hustlehub-55746-default-rtdb.europe-west1.firebaseio.com',
    projectId: 'hustlehub-55746',
    messagingSenderId: '582093535603',
    appId: '1:582093535603:web:bc1672128f7eb3831c2d06',
    measurementId: 'G-XHZ77Z1G61'
  }
}
```

Con esta configuración al llamar a esas funciones anonimas en nuestro appmodule nos comunicamos con el servicio de firebase

5.2 Autenticación

En este apartado se ve como se implementa la autenticación de nuestra aplicación , tanto como por email / contraseña como manera clásica o ya sea por autenticación mediante google , esta mas enfocada en un login rapido basandose en oauth2

```
@Injectable({
  providedIn: 'root'
})
export class AuthService {
  private currentUserSubject = new BehaviorSubject<User | null>(null);
  public currentUser$ = this.currentUserSubject.asObservable();

  constructor(private auth: Auth) {
    onAuthStateChanged(this.auth, user => {
      this.currentUserSubject.next(user);
    });
  }

  getCurrentUser(): User | null {
    return this.currentUserSubject.value;
  }

  isLoggedIn(): boolean {
    return !!this.currentUserSubject.value;
  }

  login(email: string, password: string) {
    return signInWithEmailAndPassword(this.auth, email, password);
  }

  register(email: string, password: string) {
    return createUserWithEmailAndPassword(this.auth, email, password);
  }
}
```

Para conocer mejor el funcionamiento de la autenticación con firebase hay que saber que se realiza con JWT generando el token guardandolo en localStorage y comparando con el servicio Backend que nos ofrece Firebase

5.3 Modal genérico

En esta aplicación se a incorporado un modal genérico y reutilizable para muchas de las pestañas para gestión del usuario y eventos .

Está construido usando ng-bootstrap, que facilita la integración de modales en Angular sin depender de JavaScript externo. La principal característica de este modal es que recibe componentes dinámicos como contenido, lo que permite reutilizar la misma estructura para mostrar formularios, listas, detalles o confirmaciones según sea necesario.

Como funciona :

Se crea el componente contenedor usando NgbModal

```
export class GenericModalComponent implements OnInit {  
  constructor(private modalService: NgbModal) { }  
  
  ngOnInit(): void {  
  }  
  open(component: any) {  
    this.modalService.open(component);  
  }  
}
```

En un componente como puede ser el dashboard llamamos a este componente y le inyectamos el deseado

```
openMyEventModal(userId: string): void {  
  const modalRef: NgbModalRef = this.modalService.open(MyEventsComponent, {  
    centered: true,  
    backdrop: true,  
    keyboard: true,  
    size: 'lg'  
  });  
  
  modalRef.componentInstance.userId = userId;  
}
```


para meter atributos usamos componentInstance ademas de tener las opciones del modal como el backdrop que al pinchar fuera del modal se cierra .

5.4 Slider de iconos

La idea de hacer un menu con iconos y que estos sean nuestra navegación en el home de nuestra aplicación , estos efectos se realizan en el ts del componente con eventos y manejando las coordenadas ,

```
export class CategorySliderComponent implements OnInit {
  icons: CategoryIcon[] = [];

  @ViewChild('slider', { static: false }) slider!: ElementRef;

  isDragging = false;
  startX = 0;
  scrollLeft = 0;

  constructor(private categoriesService: CategoriesService) {}

  async ngOnInit(): Promise<void> {
    this.icons = await this.categoriesService.getCategoryIcons();
  }

  onMouseDown(e: MouseEvent): void {
    this.isDragging = true;
    this.slider.nativeElement.classList.add('dragging');
    this.startX = e.pageX - this.slider.nativeElement.offsetLeft;
    this.scrollLeft = this.slider.nativeElement.scrollLeft;

    window.addEventListener('mousemove', this.onMouseMove);
    window.addEventListener('mouseup', this.onMouseUp);
  }

  onMouseMove = (e: MouseEvent): void => {
    if (!this.isDragging) return;
    e.preventDefault();
    const x = e.pageX - this.slider.nativeElement.offsetLeft;
    const walk = (x - this.startX) * 1.5;
    this.slider.nativeElement.scrollLeft = this.scrollLeft - walk;
  };
};
```

5.5 Offcanvas como dashboard

En esta parte para todo el tema de usuario decidí usar un offcanvas , como la típica pantallita que al dar click nos la abre a un lado de nuestra pantalla mostrándonos así las opciones como favoritos , crear eventos etc .

Esto se realiza usando ngbootstrap , concretamente NgbOffCanvasRef para manejar este mismo .

Desde el componente del header se vería tal que así:

```
<div class="login-container">
  <ng-container *ngIf="authService.currentUser$ | async as user; else loginButton">
    <i class="fas fa-user fs-5" (click)="openUserPanel()" id="login"></i>
  </ng-container>

  <ng-template #loginButton>
    <button id="loginButton" type="button" class="btn custom-login-btn" (click)="openmodal()">Acceder</button>
  </ng-template>
</div>
```

En la parte lógica de nuestro archivo se vería tal que así

```
openUserPanel(): void {
  const ref: NgbOffcanvasRef = this.offcanvasService.open(UserPanelComponent, {
    position: 'end'
  });
  ref.componentInstance.offcanvasRef = ref;
}
```

Una vez visto como se abre el componente que abrimos se ve tal que así :

```

<div class="d-flex flex-column justify-content-between h-100 p-4">

  <div class="text-start">
    <div *ngIf="user$ | async as user">
      <h5 class="mb-0 fw-bold">{{ user.displayName || 'Usuario' }}</h5>
      <small class="text-muted">{{ user.email }}</small>
    </div>

    <hr class="my-4">
    <div *ngIf="user$ | async as user">
      <button class="btn border-0 text-dark w-100 mb-3 fw-bold text-start" (click)="openProfile()">Mi perfil</button>
      <button class="btn border-0 text-dark w-100 mb-3 fw-bold text-start" (click)="openChangePassword()">Modificar
        datos</button>
      <button class="btn border-0 text-dark w-100 mb-3 fw-bold text-start">Mis Favoritos</button>
      <button class="btn border-0 text-dark w-100 mb-3 fw-bold text-start" (click)="openMySubscriptionsModal(user.uid)">Mis Subscripciones</button>
      <button class="btn border-0 text-dark w-100 mb-3 fw-bold text-start" (click)="openMyEventModal(user.uid)">Mis
        Eventos</button>
      <button class="btn border-0 text-dark w-100 mb-3 fw-bold text-start" (click)="openCreateEventModal()">Crear Evento
    </button>
    </div>

    You, 6 days ago • 13/05/2025 parte visual Ok a falta de fav subs ...

  </div>

  <div class="mt-auto text-start">
    <button class="btn btn-danger px-5" (click)="logout()">Cerrar sesión</button>
  </div>

</div>

```

Llamando con botones a los otros componentes que se lanzan con el modal genérico .

5.6 Servicio de inactividad

Se implementó un servicio para manejar la inactividad ya que al estar usando un plan gratis de firebase el control de las conexiones es muy importante porque si no nos pasaríamos de los límites de uso y nuestra aplicación dejaría de funcionar a no ser que pasase por caja .

El servicio es sencillo en cuanto a lógica pero eficaz

En este componente se pone que al cumplirse el tiempo de inactividad y lanzar el error de timeout te reenvía a una ruta llamada /inactividad .

```

})
export class IdleService {
  private timer: any;
  private readonly timeout = 2 * 60 * 1000;
  private auth = inject(Auth);

  constructor(private router: Router, private zone: NgZone) {
    this.resetTimer();
    this.initListeners();
  }

  private initListeners(): void {
    ['click', 'keypress', 'scroll'].forEach(event => {
      this.router.navigate(['/']);
      document.addEventListener(event, () => this.resetTimer());
    });
  }

  private resetTimer(): void {
    clearTimeout(this.timer);
    this.timer = setTimeout(() => this.logout(), this.timeout);
  }

  private async logout(): Promise<void> {
    this.zone.run(async () => {
      try {
        await signOut(this.auth);
        this.router.navigate(['/inactividad']);
        console.log('Sesión cerrada por inactividad.');
      } catch (error) {
        console.error('Error al cerrar sesión:', error);
      }
    });
  }
}

```

5.7 Routing

Para gestionar las rutas a diferencia de otros frameworks Angular lo trae en su core de por sí sin necesidad de de paquetes o librerías externas , en este proyecto no existen rutas enrevesadas peros sí útiles para una navegación fluida .

La ruta raíz (") carga la página de inicio (HomeComponent), siendo la primera vista que ve el usuario al acceder a la aplicación.

La ruta 'about' lleva a la sección "Sobre nosotros" (AboutUsComponent), que ofrece información general del proyecto y del equipo.

La ruta 'hot' muestra los eventos más destacados o populares, gestionados desde el componente HotEventsComponent.

En 'inactividad', se encuentra una vista de prueba/control del sistema de detección de inactividad del usuario, útil durante el desarrollo o para futuras funcionalidades de seguridad.

El bloque de rutas 'categories/:categoryId' permite navegar a los eventos de una categoría específica. Dentro de esta ruta, se anida :eventId, que carga directamente los detalles de un evento dentro de esa categoría. Este diseño permite una navegación limpia y jerárquica, como por ejemplo:

/categories/cine/12345.

Adicionalmente, existe una ruta alternativa 'event/:category/:id', pensada para acceder al detalle de un evento de forma directa sin necesidad de estar dentro del contexto de una categoría. Esto es útil cuando se redirige desde otras partes de la aplicación.

Finalmente, la ruta comodín '**' gestiona cualquier URL no válida y carga un componente de error (NotFoundComponent) que informa al usuario de que la página no existe.

Finalmente el archivo de rutas quedaría tal que así:

```
const routes: Routes = [
  { path: '', component: HomeComponent, pathMatch: 'full' },
  { path: 'about', component: AboutUsComponent },
  { path: 'hot', component: HotEventsComponent },
  { path: 'inactividad', component: InactividadComponent },
  {
    path: 'categories/:categoryId',
    component: CategoryDetailComponent,
    children: [
      {
        path: ':eventId',
        component: EventDetailComponent
      }
    ]
  },
  { path: 'event/:category/:id', component: EventDetailComponent },
  { path: '**', component: NotFoundComponent }
];
```

5.8 Recogida de datos de firebase

Para entender cómo recogemos los datos de firebase hay que entender que es una base de datos no SQL es decir guarda los datos en formato json , esto es bueno ya que nos facilita la obtención de datos de manera muy sencilla

Se empieza por importar los módulos necesarios para conectarnos a la base de datos

```
import { Injectable } from '@angular/core';
import { Database, ref, get, child } from '@angular/fire/database';
```

Se prepara un método para así recibir los datos

```
async getAllEvents(): Promise<any[]> {
  const snapshot = await get(child(ref(this.db), 'events'));
  if (!snapshot.exists()) return [];

  const allEventsObj = snapshot.val();

  return Object.entries(allEventsObj).flatMap(
    ([categoryKey, category]: any) =>
      category.items
        ? Object.values(category.items).map((item: any) => ({ ...item, category: categoryKey }))
        : []
  );
}
```

Concretamente este método obtiene todos los eventos de todas las categorías almacenadas en Firebase Realtime Database. Devuelve un array plano con todos los eventos, añadiéndoles además el nombre de la categoría a la que pertenecen ya que a la hora de enrutar la aplicación es necesario tener tanto la categoría como el id del evento en cuestión permitiendo una navegación mas óptima.

5.9 Inserción de datos

Para tener nuestros eventos a los que nos subscribimos o mismamente los que creamos necesitamos hacer inserciones en nuestra base de datos , esto por defecto esta deshabilitado

ya que Firebase no permite realizar inserciones , pero se desactiva en dos clicks cambiando las reglas de la realtime database .

Por otra parte vamos a ver con el ejemplo de crear evento como se realiza una inserción en base de datos

Creamos un formulario vacío con el lifecyclehook ngOnInit

```
ngOnInit(): void {
  this.eventForm = this.fb.group({
    title: ['', Validators.required],
    description: [''],
    eventDate: ['', Validators.required],
    image: [''],
    category: ['', Validators.required]
  });

  this.loadCategories();
}
```

```
<div class="modal-header">
  <h4 class="modal-title">Crear Nuevo Evento</h4>
  <button type="button" class="btn-close" aria-label="Close" (click)="activeModal.close()"></button>
</div>

<div class="modal-body">
  <form [formGroup]="eventForm" (ngSubmit)="onSubmit()">
    <div class="mb-3">
      <label>Título</label>
      <input class="form-control" formControlName="title" required />
    </div>

    <div class="mb-3">
      <label>Descripción</label>
      <textarea class="form-control" formControlName="description" rows="3"></textarea>
    </div>

    <div class="mb-3">
      <label>Fecha del Evento</label>
      <input type="date" class="form-control" formControlName="eventDate" required />
    </div>

    <div class="mb-3">
      <label>Imagen (URL)</label>
      <input class="form-control" formControlName="image" />
    </div>

    <div class="mb-3">
      <label>Categoría</label>
      <select class="form-select" formControlName="category">
        <option *ngFor="let cat of categories" [value]="cat.category">
          {{ cat.category | titlecase }}
        </option>
      </select>
    </div>

    <button type="submit" class="btn btn-success" [disabled]="eventForm.invalid">Guardar</button>
  </form>
</div>
```

Luego construiremos la llamada para mandar a la base de datos la información con los campos insertados en el formulario reactivo de Angular .

```
async onSubmit(): Promise<void> {
  if (this.eventForm.valid) {
    const formValue = this.eventForm.value;
    const categoryPath = `events/${formValue.category}/items`;
    const newRef = push(ref(db, categoryPath));

    const user = auth.currentUser;

    const eventData = {
      id: newRef.key,
      title: formValue.title,
      description: formValue.description,
      eventDate: formValue.eventDate,
      image: formValue.image || 'https://images.pexels.com/photos/3183186/pexels-photo-3183186.jpeg',
      createdAt: new Date().toISOString(),
      createdBy: user?.uid || 'anonymous',
      creatorEmail: user?.email || 'unknown'
    };

    await set(newRef, eventData);
    this.activeModal.close();
  }
}
close() {
  this.activeModal.close();
}
```

Los campos seguidos de las pipes || son los campos default en caso de no poner nada .

5.10 Gestión de suscripciones

Este servicio se encarga de manejar las suscripciones de los usuarios a eventos dentro de la aplicación HustleHub, utilizando Firebase Realtime Database como almacenamiento.

Está decorado con @Injectable({ providedIn: 'root' }), lo que significa que es un servicio global y puede ser inyectado en cualquier componente.


```

@Injectables({ providedIn: 'root' })
export class SubscriptionService {
  constructor(private db: Database) {}

  async subscribeToEvent(userId: string, eventId: string, category: string): Promise<void> {
    const subRef = ref(this.db, `subscriptions/${userId}/${category}/${eventId}`);
    return set(subRef, true);
  }

  async unsubscribeFromEvent(userId: string, eventId: string, category: string): Promise<void> {
    const subRef = ref(this.db, `subscriptions/${userId}/${category}/${eventId}`);
    return remove(subRef);
  }

  async getUserSubscriptions(userId: string): Promise<Record<string, string[]>> {
    const snapshot = await get(ref(this.db, `subscriptions/${userId}`));
    const result: Record<string, string[]> = {};
    if (!snapshot.exists()) return result;

    const rawData = snapshot.val();
    for (const category of Object.keys(rawData)) {
      result[category] = Object.keys(rawData[category] || {});
    }

    return result;
  }
}

```

El uso de este servicio en el componente seria de tal manera que :

```

async ngOnInit(): Promise<void> {
  if (!this.userId) {
    this.isError = true;
    this.isLoading = false;
    return;
  }

  try {
    const subscriptions = await this.subscriptionService.getUserSubscriptions(this.userId);

    for (const category of Object.keys(subscriptions)) {
      const eventIds = subscriptions[category];
      const categoryEvents: any[] = [];

      const itemsSnap = await get(ref(this.db, `events/${category}/items`));

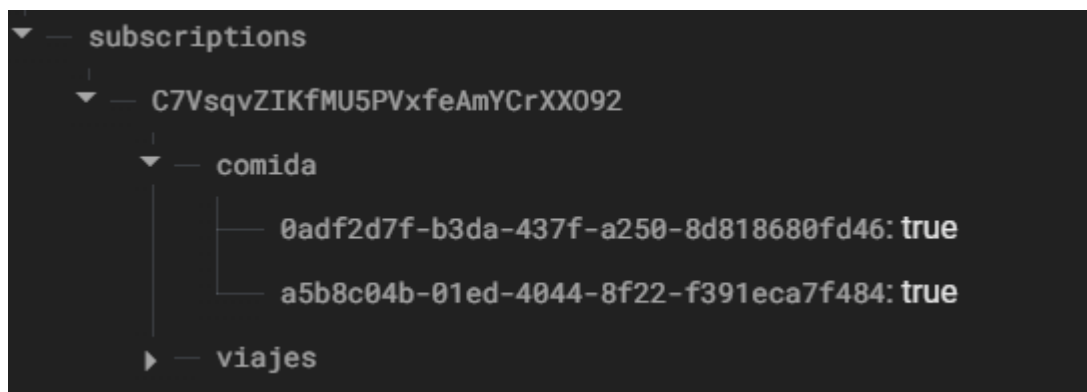
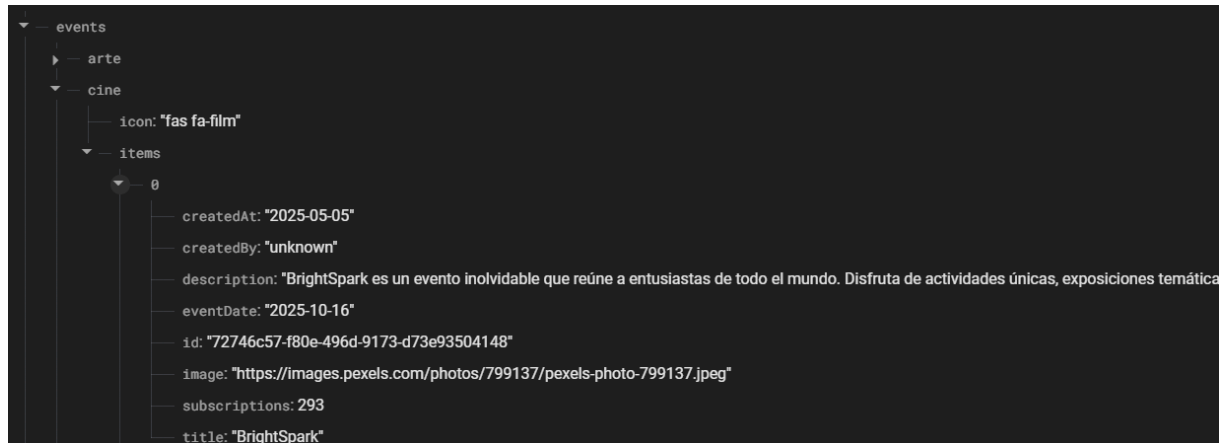
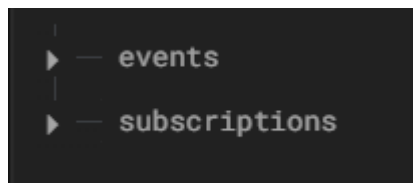
      if (itemsSnap.exists()) {
        const items = Object.values(itemsSnap.val());

        for (const eventId of eventIds) {
          const match = items.find((event: any) => event.id === eventId);
          if (match) {
            categoryEvents.push({ ...match, category });
          }
        }
      }

      if (categoryEvents.length > 0) {
        this.eventsByCategory[category] = categoryEvents;
      }
    }
  } catch (error) {
    console.error('Error cargando suscripciones:', error);
    this.isError = true;
  } finally {
    this.isLoading = false;
  }
}

```

Entendiendo que la estructura de la base de datos esta de tal manera que



En las subscripciones se guarda el uid de usuario con el que se a realizado además de la categoría y el id del evento en concreto , ganando asi modularidad y claridad en la estructura de datos .

5.11 Gestión de eventos de usuario

En este apartado se ve cómo se gestionan los eventos creados por un usuario , mostrandolos en el modal genérico e inyectando el componente al modal , lo importante de este apartado no es el html en sí , si no que lo importante es como recogemos esos eventos

Se realiza mediante un get como :

```
ngOnInit(): void {
  const user = this.auth.currentUser;

  if (!user) {
    this.isLoading = false;
    this.isError = true;
    return;
  }

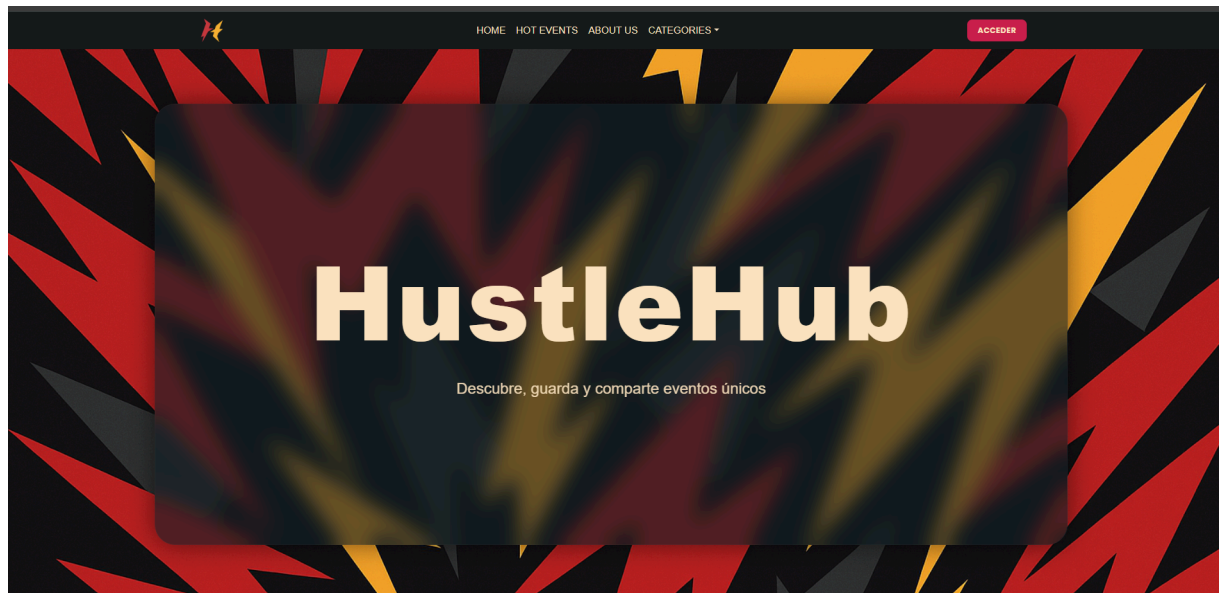
  this.categoriesService.getEventsByUser(user.uid)
    .then(events => {
      this.myEvents = events;
      this.isLoading = false;
    })
    .catch(err => {
      console.error('Error loading events:', err);
      this.isLoading = false;
      this.isError = true;
    });
}
```

Cogiendo así el uid de usuario que nos proporciona firebase y guardándolo de manera sencilla.

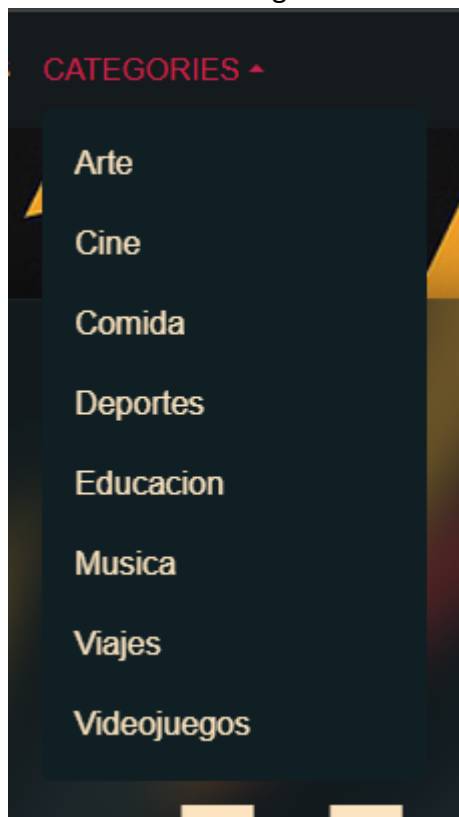
```
-OQCYS2Uil7ktZM22ns8
├── createdAt: "2025-05-14T06:49:05.503Z"
├── createdBy: "C7VsqvZIKfMU5PVxfeAmYCrXXO92"
├── creatorEmail: "amartinr03@gmail.com"
├── description: "asdasfedfawefasdfweafsad"
├── eventDate: "2025-05-22"
├── id: "-OQCYS2Uil7ktZM22ns8"
├── image: "https://images.pexels.com/photos/3183186/pexels-photo-3183186.jpeg"
└── title: "Prueba1"
```

Capítulo 6 : Demo

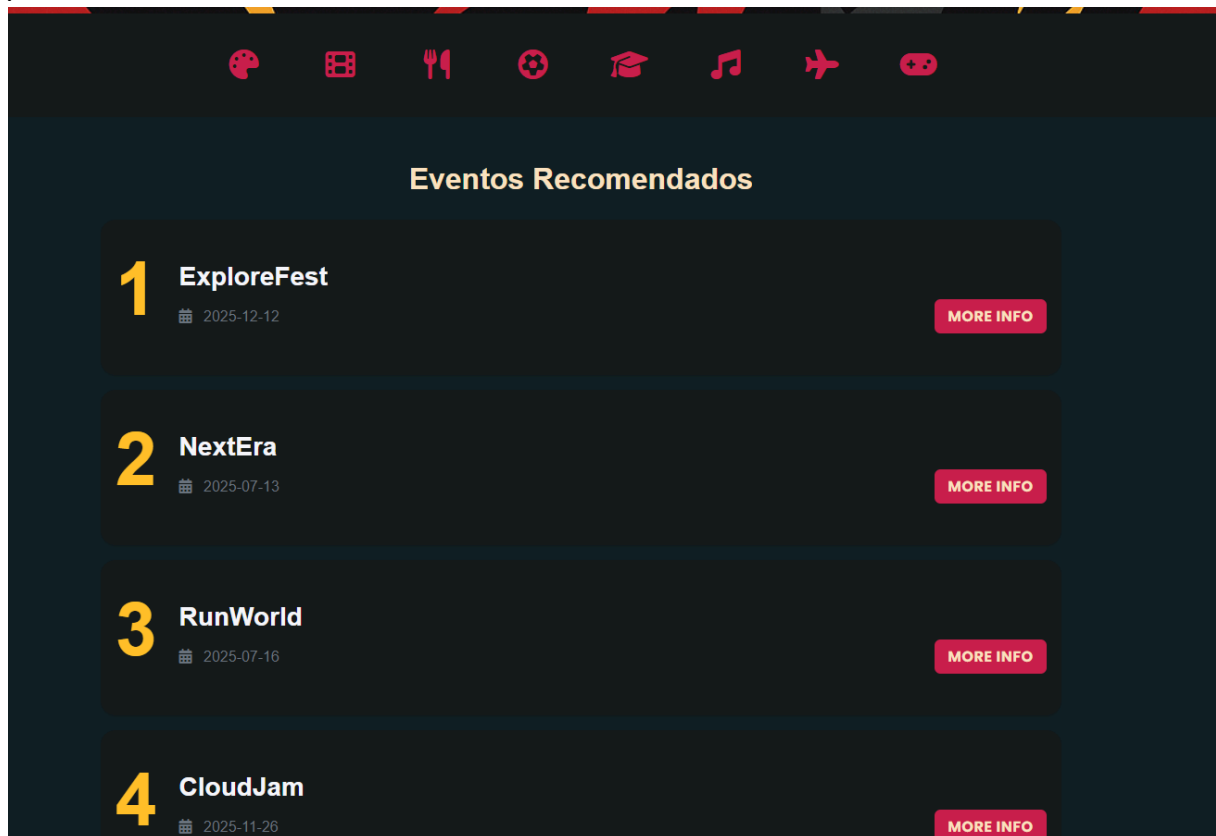
Nada más entrar encontraremos un hero con nuestra aplicación



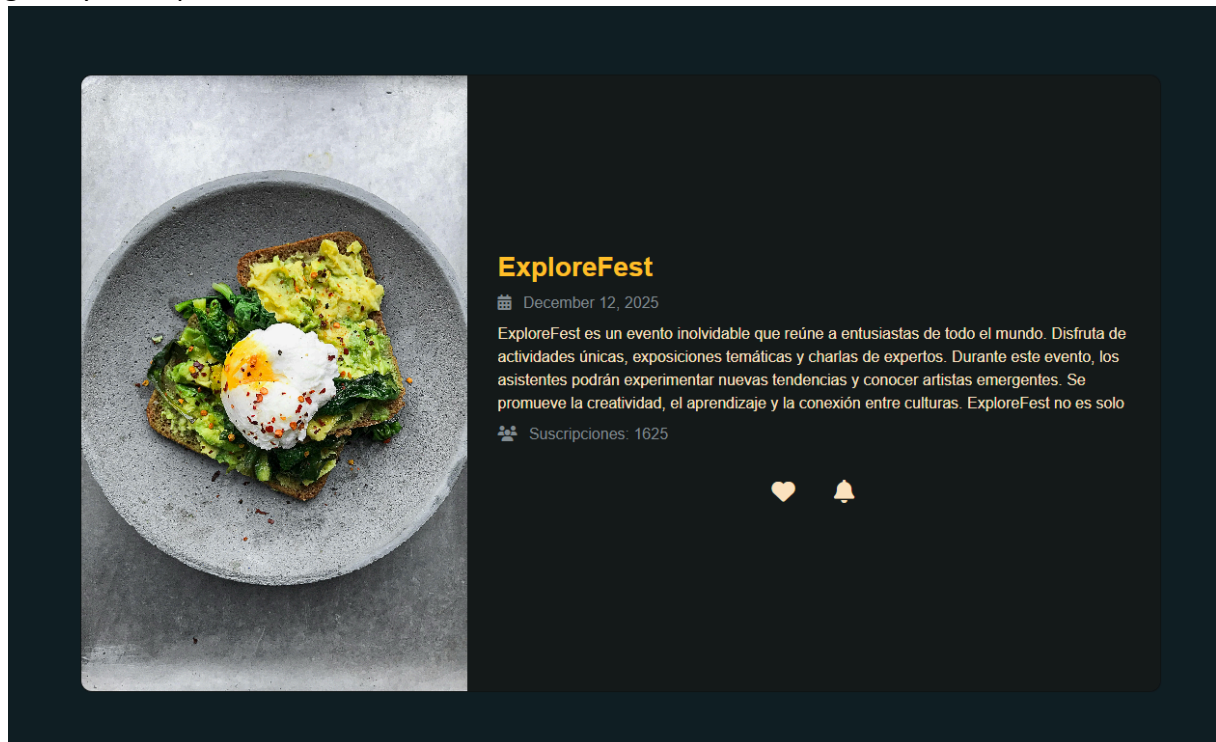
Podemos visitar categorías en el menú desplegable



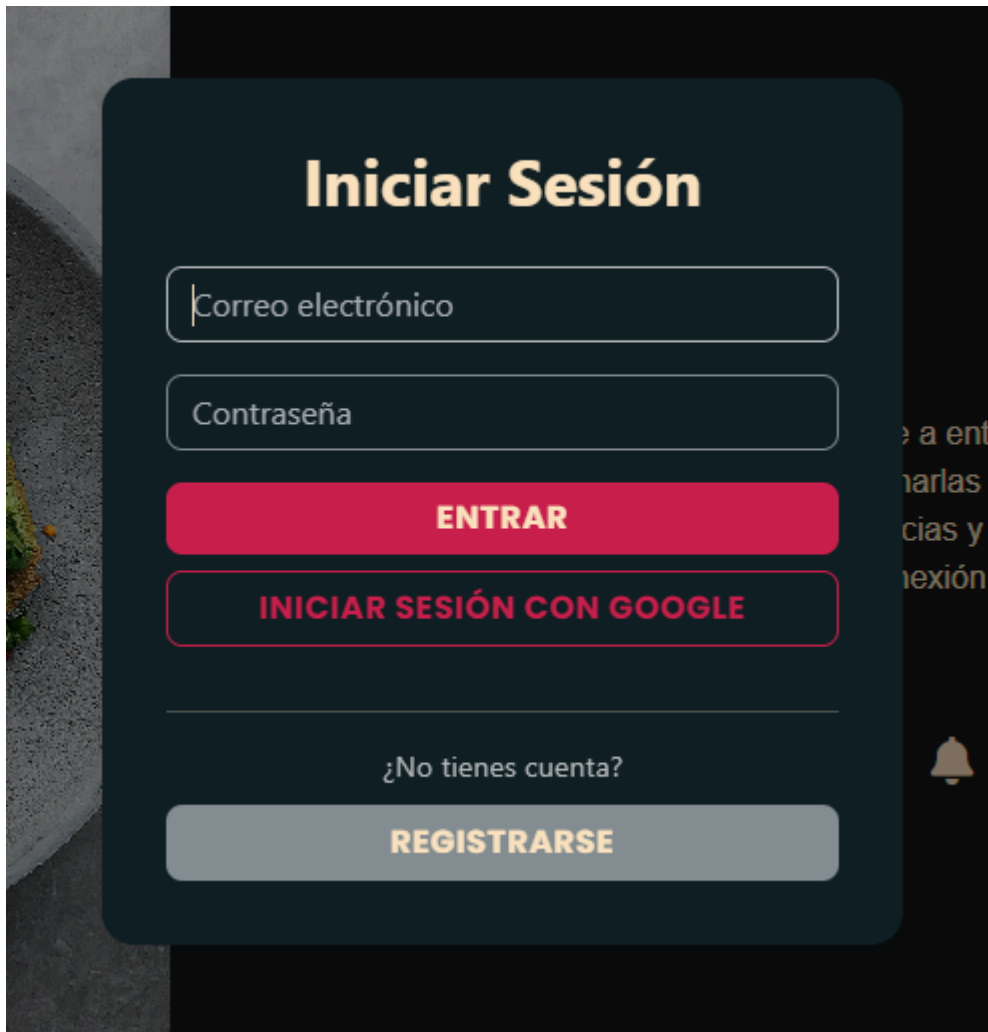
Siguiendo en la pantalla de inicio podemos ver un slider de nuevo con las categorías y unos eventos totalmente aleatorios a modo de información



En el detalle de cada evento podemos ver su fecha y una descripción mas los botones de me gusta y suscripción



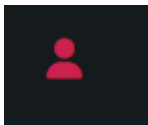
Por otra parte tenemos el inicio de sesión



The image shows a dark-themed login form overlay. At the top, the title "Iniciar Sesión" is displayed in a large, bold, light-colored font. Below the title are two input fields: "Correo electrónico" and "Contraseña". A red button labeled "ENTRAR" is positioned below the password field. Below the "ENTRAR" button is a button labeled "INICIAR SESIÓN CON GOOGLE". A horizontal line separates this section from the registration section below. The registration section starts with the text "¿No tienes cuenta?" and a light gray button labeled "REGISTRARSE". To the right of the form, a small bell icon is visible.

Donde iniciaremos sesión con lo que nos parezca más cómodo en ese momento o si no registrarnos .

Una vez registrados saldria nuestro icono de usuario



pinchando sobre él nos desplegará un offCanvas mostrando información .

Usuario

test@example.com

MI PERFIL

MODIFICAR DATOS

MIS FAVORITOS

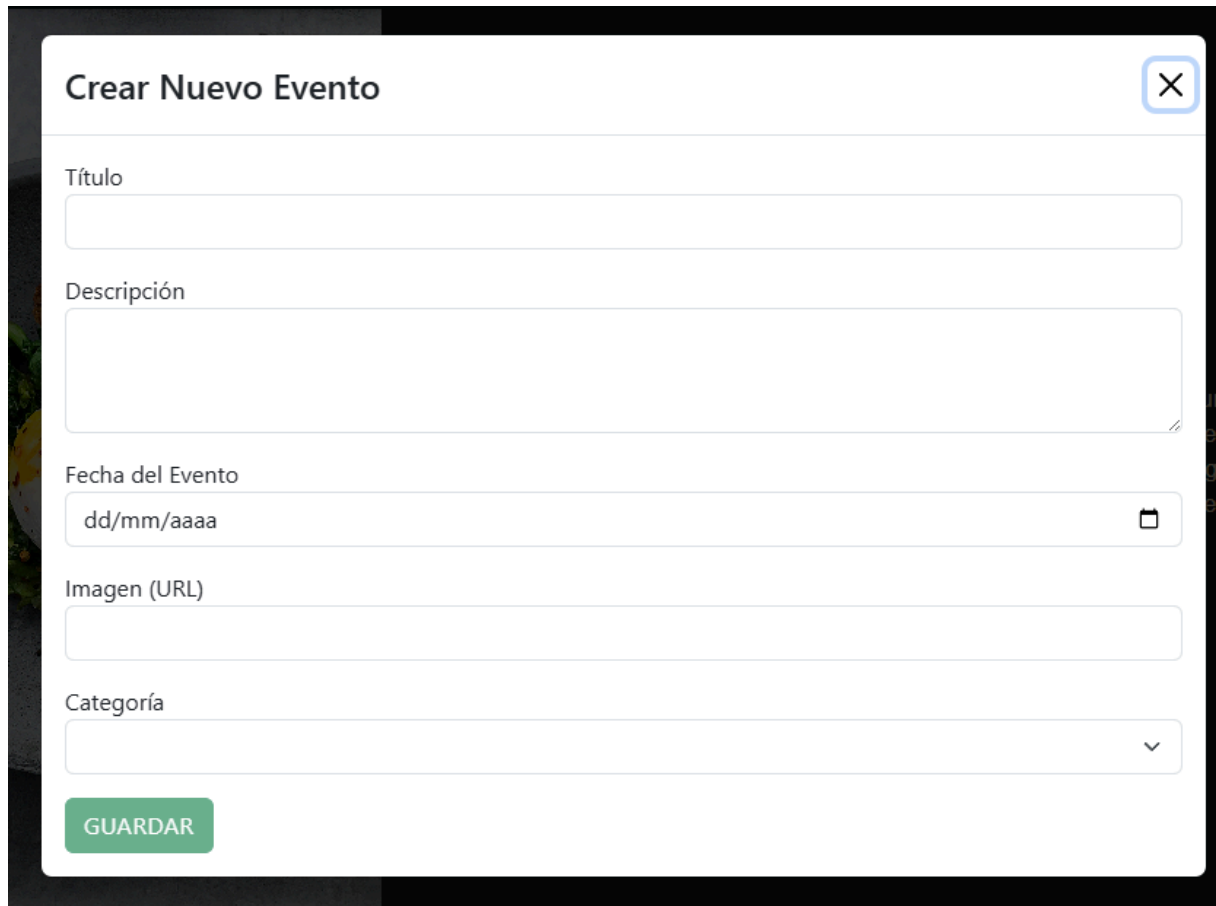
MIS SUBSCRIPCIONES

MIS EVENTOS

CREAR EVENTO

CERRAR SESIÓN

En este podremos ver nuestra Información o gestionar nuestros eventos



Crear Nuevo Evento

Título

Descripción

Fecha del Evento

dd/mm/aaaa

Imagen (URL)

Categoría

GUARDAR

Capítulo 7: Conclusiones y Líneas Futuras

Conclusiones

Con este proyecto se puede ver una evolución en la capacidad de desarrollo con Angular ya que a un inicio todo era mas complicado, una vez e ido aprendiendo el framework las funcionalidades salian de una manera mas fluida y no se demoraba tanto tiempo en su realización .

Este proceso ha sido clave para entender cómo organizar un proyecto real, cómo estructurar los componentes, servicios y módulos, y sobre todo cómo trabajar con tecnologías como Firebase y Angular de forma conjunta. También ha sido una oportunidad para poner en práctica buenas prácticas de desarrollo, como la reutilización de componentes, la separación de lógica y vista, y el uso de herramientas modernas para desplegar y mantener una aplicación web.

Líneas Futuras

-Gestión de favoritos personalizada

Actualmente se permite guardar eventos como favoritos, pero se puede trabajar más en esa funcionalidad: mostrar recomendaciones según los favoritos, ordenarlos por relevancia, o incluso permitir compartirlos con otros usuarios.

-Sistema de notificaciones

Añadir notificaciones en tiempo real (por ejemplo, cuando hay cambios en eventos suscritos) mejoraría mucho la experiencia del usuario.

-Buscador inteligente y filtros avanzados

Incluir búsqueda por ubicación o fecha, con filtros combinables, para facilitar la navegación en grandes volúmenes de eventos.

-Panel de administración (Dashboard)

Una interfaz solo para administradores que permita crear, editar o eliminar eventos directamente, así como gestionar categorías o usuarios.

-Internacionalización (i18n)

Traducir la aplicación a varios idiomas para abrirla a un público más amplio.

Capítulo 8: Bibliografía

Angular. (2025). *Angular - The modern web developer's platform*. Recuperado de: <https://angular.io/>

Firebase. (2025). *Firebase Documentation | Get started with Firebase*. Google. Recuperado de: <https://firebase.google.com/docs>

RxJS. (2025). *ReactiveX - RxJS Documentation*. Recuperado de: <https://rxjs.dev/>

Bootstrap. (2025). *Bootstrap v5.3 · The world's most popular framework for building responsive, mobile-first sites*. Recuperado de: <https://getbootstrap.com/>

ng-bootstrap. (2025). *ng-bootstrap - Bootstrap components built with Angular*. Recuperado de: <https://ng-bootstrap.github.io/>

GitHub. (2025). *GitHub Docs – Collaboration and CI/CD tools*. Recuperado de: <https://docs.github.com/>

Figma. (2025). *Figma - Collaborative interface design tool*. Recuperado de: <https://www.figma.com/>

Stack Overflow. (2025). *Public Q&A on Angular, Firebase, RxJS and web development*. Recuperado de: <https://stackoverflow.com/>

Google Developers. (2025). *OAuth 2.0 for Web Server Applications*. Recuperado de: <https://developers.google.com/identity/protocols/oauth2>

Mozilla Developer Network (MDN). (2025). *Web APIs and frontend concepts*. Recuperado de: <https://developer.mozilla.org/>