





## Agenda personal con Angular y Firebase

### Objetivo




Diseñar e implementar una aplicación web con Angular que funcione como una **agenda personal visual tipo Kanban**, permitiendo gestionar tareas de forma organizada en columnas y almacenarlas en tiempo real usando Firebase como base de datos.

### Descripción

En esta actividad vas a desarrollar una aplicación Angular conectada a Firebase para crear y gestionar tareas en una interfaz de tipo tablero (*board*). La aplicación permitirá:

-  Crear tareas en la columna "Nueva".
-  Mover tareas entre tres columnas: **Nueva**, **En proceso** y **Completada** mediante **drag and drop**.
-  Visualizar las tareas con información básica: **título**, **descripción** y **fecha de creación**.
-  Al mover una tarea a otra columna, su color de fondo cambiará automáticamente para reflejar su estado.

### Requisitos funcionales

1. Mostrar tres columnas:
  -  **Nueva**
  -  **En proceso**
  -  **Completada**
2. Cada columna muestra solo las tareas que coincidan con su estado.
3. En la columna **Nueva** habrá un botón **+** que al pulsarlo mostrará un formulario modal para crear una nueva tarea.
4. Las tareas deben poder moverse entre columnas con arrastrar y soltar (**drag & drop**). **Módulo:** `npm install @angular/cdk`
5. Cada columna debe tener un color de fondo diferente según el estado.
6. Toda la información se almacena en tiempo real en Firebase.

## firebase/firebase.config.ts

```
import { provideFirebaseApp, initializeApp } from '@angular/fire/app';
import { provideFirestore, getFirestore } from '@angular/fire/firestore';
import { provideStorage, getStorage } from '@angular/fire/storage';
```

```
const firebaseConfig = {
  apiKey: "",
  authDomain: "",
  projectId: "",
  storageBucket: "",
  messagingSenderId: "",
  appId: ""
};
```

```
export const firebaseProviders = [
  provideFirebaseApp(() => initializeApp(firebaseConfig)),
  provideFirestore(() => getFirestore()),
  provideStorage(() => getStorage())
];
```

## app.config.ts

```
import { ApplicationConfig, provideZoneChangeDetection } from '@angular/core';
import { provideRouter, withViewTransitions } from '@angular/router';
import { firebaseProviders } from './firebase/firebase.config'; // Importar la configuración de Firebase
```

```
import { routes } from './app.routes';
```

```
export const appConfig: ApplicationConfig = {
  providers: [provideZoneChangeDetection({ eventCoalescing: true }), provideRouter(routes,
    withViewTransitions()), ...firebaseProviders]
};
```

## models/note.model.ts

```
export interface Note {
  id?: string;
  title: string;
  description: string;
  datetime: string;
  status: 'nueva' | 'en_proceso' | 'completada';
}
```

## services/note.service.ts

```
// note.service.ts
import { Injectable } from '@angular/core';
import {
  Firestore,
  getDocs,
  collection,
  addDoc,
  doc,
  updateDoc,
  deleteDoc,
  where,
  query
} from '@angular/fire/firestore';
import { Note } from '../models/note.model';

@Injectable({ providedIn: 'root' })
export class NoteService {
  constructor(private firestore: Firestore) {}

  async getNotes(): Promise<Note[]> {
    const notesRef = collection(this.firestore, 'notes');
    const snapshot = await getDocs(notesRef);
    return snapshot.docs.map(doc => ({ id: doc.id, ...(doc.data() as Note) }));
  }

  addNote(note: Note) {
    const notesRef = collection(this.firestore, 'notes');
    return addDoc(notesRef, note);
  }

  updateNoteStatus(id: string, status: Note['status']) {
    const noteRef = doc(this.firestore, 'notes', id);
    return updateDoc(noteRef, { status });
  }

  deleteNote(id: string) {
    const noteRef = doc(this.firestore, 'notes', id);
    return deleteDoc(noteRef);
  }
}
```

## board.component.ts

```
// board.component.ts
import { Component, OnInit } from '@angular/core';
import { CommonModule } from '@angular/common';
import { NoteService } from '../services/note.service';
import { Note } from '../models/note.model';
import { CdkDragDrop, DragDropModule } from '@angular/cdk/drag-drop';
import { NoteCardComponent } from '../note-card/note-card.component';
import { FormsModule } from '@angular/forms';

@Component({
  selector: 'app-board',
  standalone: true,
  imports: [CommonModule, FormsModule, NoteCardComponent, DragDropModule],
  templateUrl: './board.component.html',
  styleUrls: ['./board.component.css']
})
export class BoardComponent implements OnInit {
  notes: Note[] = [];
  statusList: Note['status'][] = ['nueva', 'en_proceso', 'completada'];

  showModal = false;
  newTitle = '';
  newDescription = '';

  constructor(private noteService: NoteService) {}

  ngOnInit(): void {
    this.loadNotes();
  }

  async loadNotes() {
    try {
      this.notes = await this.noteService.getNotes();
      console.log('DATA LLEGÓ ✅', this.notes);
    } catch (err: any) {
      console.error('ERROR AL LEER DATOS ❌', err.message, err);
    }
  }

  getNotesByStatus(status: Note['status']) {
    return this.notes.filter(note => note.status === status);
  }
}
```

## Agenda personal con Angular y Firebase

```
async drop(event: CdkDragDrop<Note[]>, newStatus: Note['status']) {
  const note = event.previousContainer.data[event.previousIndex];

  if (note.status !== newStatus) {
    // 1. Actualiza el estado local inmediatamente
    note.status = newStatus;

    // 2. Mueve la tarjeta visualmente
    this.notes = [
      ...this.notes.filter(n => n.id !== note.id),
      note
    ];

    // 3. Actualiza en Firestore (sin esperar el resultado para redibujar)
    this.noteService.updateNoteStatus(note.id!, newStatus).catch(err => {
      console.error('Error al actualizar nota:', err);
    });
  }
}

getStatusLabel(status: string): string {
  return status
    .replace('_', ' ')
    .replace(/\b\w/g, l => l.toUpperCase()); // Capitaliza cada palabra
}

openModal() {
  this.newTitle = '';
  this.newDescription = '';
  this.showModal = true;
}

closeModal() {
  this.showModal = false;
}

async saveNote() {
  if (this.newTitle.trim() && this.newDescription.trim()) {
    const nuevaNota: Note = {
      title: this.newTitle,
      description: this.newDescription,
      datetime: new Date().toISOString(),
      status: 'nueva'
    };
    await this.noteService.addNote(nuevaNota);
    await this.loadNotes();
    this.closeModal();
  }
}
```

```
}  
}  
}
```

## board.component.html

```
<div class="board" cdkDropListGroup>  
  <div class="column" *ngFor="let status of statusList">  
    <h3>  
      {{ getStatusLabel(status) }}  
      <button *ngIf="status === 'nueva'" (click)="openModal()" style="margin-left: 8px; background-color: #28a745; border: none; border-radius: 50%; color: white; width: 24px; height: 24px; font-size: 16px; line-height: 20px; cursor: pointer;">  
        +  
      </button>  
    </h3>  
  
    <div class="note-container" cdkDropList [cdkDropListData]="getNotesByStatus(status)" [cdkDropListConnectedTo]="statusList" (cdkDropListDropped)="drop($event, status)">  
  
      <div *ngFor="let note of getNotesByStatus(status)" cdkDrag>  
        <app-note-card [note]="note"></app-note-card>  
      </div>  
  
    </div>  
  </div>  
  
  <!-- Modal -->  
  <div *ngIf="showModal" class="modal-backdrop">  
    <div class="modal">  
      <h4>Nueva Nota</h4>  
      <label>Título:</label>  
      <input [(ngModel)]="newTitle" />  
      <label>Descripción:</label>  
      <textarea [(ngModel)]="newDescription" rows="6"></textarea>  
      <div class="modal-buttons">  
        <button (click)="saveNote()">Guardar</button>  
        <button (click)="closeModal()">Cancelar</button>  
      </div>  
    </div>  
  </div>  
</div>
```

## board.component.css

```
.board {
  display: flex;
  justify-content: space-between;
  gap: 2rem;
  padding: 2rem;
  flex-wrap: wrap;
}

.column {
  flex: 1;
  min-width: 300px;
  max-width: 100%;
  background: #ffffffcc;
  backdrop-filter: blur(6px);
  padding: 1rem;
  border-radius: 12px;
  box-shadow: 0 4px 20px rgba(0, 0, 0, 0.1);
  transition: transform 0.2s ease;
}

.column:hover {
  transform: translateY(-2px);
}

.column h3 {
  display: flex;
  align-items: center;
  justify-content: space-between;
  margin-bottom: 1rem;
  font-size: 1.2rem;
  color: #333;
  font-weight: 600;
  border-bottom: 2px solid #e0e0e0;
  padding-bottom: 0.5rem;
}

.note-container {
  min-height: 300px;
  padding: 1rem;
  border: 2px dashed #dcdcdc;
  border-radius: 10px;
```

## Agenda personal con Angular y Firebase

```
background-color: #fafafa;
transition: background 0.2s ease;
}

.note-container.cdk-drop-list-dragging {
  background-color: #f0fff0;
  border-color: #4caf50;
}

.note-card {
  background: white;
  border-radius: 8px;
  padding: 1rem;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
  margin-bottom: 0.75rem;
  transition: transform 0.2s ease;
}

.note-card:hover {
  transform: scale(1.02);
}

.cdk-drag-preview {
  box-shadow: 0 5px 15px rgba(0, 0, 0, 0.3);
  border-radius: 8px;
  background: white;
  padding: 0.75rem;
  transform: rotate(2deg);
  opacity: 0.95;
}

.cdk-drag-animating {
  transition: transform 0.25s ease;
}

.note-container.cdk-drop-list-dragging .note-card {
  opacity: 0.4;
}

.modal-backdrop {
  position: fixed;
  inset: 0;
  background: rgba(0, 0, 0, 0.4);
  backdrop-filter: blur(4px);
  display: flex;
  justify-content: center;
  align-items: center;
  z-index: 1000;
```



```
}
```

```
.modal {  
  background: #ffffffcc;  
  backdrop-filter: blur(10px);  
  padding: 2rem;  
  border-radius: 12px;  
  box-shadow: 0 8px 32px rgba(0, 0, 0, 0.25);  
  width: 100%;  
  max-width: 400px;  
  display: flex;  
  flex-direction: column;  
  gap: 1rem;  
  animation: fadeIn 0.3s ease-out;  
}
```

```
.modal h4 {  
  margin: 0;  
  font-size: 1.25rem;  
  font-weight: 600;  
  color: #333;  
  text-align: center;  
}
```

```
.modal input,  
.modal textarea {  
  padding: 0.75rem;  
  border: 1px solid #ddd;  
  border-radius: 6px;  
  font-size: 1rem;  
  background: #f9f9f9;  
  transition: border-color 0.2s ease;  
}
```

```
.modal input:focus,  
.modal textarea:focus {  
  border-color: #4caf50;  
  outline: none;  
  background: #fff;  
}
```

```
.modal-buttons {  
  display: flex;  
  justify-content: space-between;  
  gap: 0.5rem;  
}
```

```
.modal-buttons button {
```

## Agenda personal con Angular y Firebase

```
flex: 1;
padding: 0.6rem 1rem;
border: none;
border-radius: 6px;
font-size: 1rem;
cursor: pointer;
transition: background 0.2s ease;
}
```

```
.modal-buttons button:first-child {
  background-color: #4caf50;
  color: white;
}
```

```
.modal-buttons button:last-child {
  background-color: #e0e0e0;
  color: #333;
}
```

```
.modal-buttons button:hover {
  opacity: 0.9;
}
```

```
@keyframes fadeIn {
  from {
    opacity: 0;
    transform: translateY(10px);
  }

  to {
    opacity: 1;
    transform: translateY(0);
  }
}
```

## note-card.component.ts

```
import { Component, Input } from '@angular/core';
import { CommonModule } from '@angular/common';
import { Note } from '../models/note.model';
```

```
@Component({
  selector: 'app-note-card',
  standalone: true,
  imports: [CommonModule],
  templateUrl: './note-card.component.html',
  styleUrls: ['./note-card.component.css']
})
export class NoteCardComponent {
  @Input() note!: Note;

  getStatusClass() {
    return {
      nueva: 'nueva',
      en_proceso: 'en_proceso',
      completada: 'completada',
    }[this.note.status];
  }
}
```

## note-card.component.html

```
<div class="note" [ngClass]="getStatusClass()">
  <h3>{{ note.title }}</h3>
  <p>{{ note.description }}</p>
  <small>{{ note.datetime | date: 'short' }}</small>
</div>
```

note-card.component.css

```
.note {
  padding: 1rem;
  margin-bottom: 0.5rem;
  border-radius: 8px;
  color: #333;
}
```

## Agenda personal con Angular y Firebase

```
.note:hover{
  cursor: pointer;
  opacity: 0.8;
}

.nueva {
  background-color: #e0f7fa;
}

.en_proceso {
  background-color: #fff9c4;
}

.completada {
  background-color: #c8e6c9;
}
```

### **app.component.ts**

```
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';
import { BoardComponent } from './board/board.component';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [RouterOutlet, BoardComponent],
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'kanban-firebase';
}
```