Desarrollo de Interfaces

Proyecto Entregable 04 - Aplicación Flutter con integración backend

Autor: Sergi García

Actualizado Septiembre 2025

Licencia

Nomenclatura

A lo largo de este tema se utilizarán diferentes símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

📖 **Importante**

❕ **Atención**

💬 **Interesante**

ÍNDICE

1. Objetivo del Proyecto **3**
2. Descripción General **3**
3. Requisitos técnicos **3**
4. Condiciones de desarrollo y herramientas Permitidas **5**
5. Entregables **6**
6. Criterios de Evaluación **6**
7. Calificación **6**
8. Ideas y pistas por tipo de Backend **7**

Proyecto Entregable 04 - Aplicación Flutter con integración backend

1. Objetivo del Proyecto

Desarrollar una aplicación completa y funcional en Flutter que integre interfaz de usuario avanzada, gestión de estado, navegación, diseño responsive y, fundamentalmente, consumo de datos desde un backend elegido por el alumno. El proyecto demostrará la capacidad para integrar todos los conocimientos adquiridos durante el curso en un producto realista y bien estructurado, abarcando tanto el frontend como la conectividad con servicios externos.

2. Descripción General

Cada estudiante diseñará, programará y desplegará una aplicación móvil y/o web con Flutter que cumpla con requisitos técnicos avanzados. La aplicación debe incluir:

- Interfaz de usuario compleja y bien diseñada.
- Navegación entre múltiples pantallas.
- Gestión de estado profesional (Provider, Bloc, Riverpod o similar).
- Conexión con un backend a elegir entre:
    - API REST de terceros (pública o privada con autorización).
    - Firebase (Firestore, Authentication, Storage).
    - Backend propio desarrollado por el alumno (ej: con FastAPI, Node.js, Spring Boot).
- Diseño adaptable (responsive).
- Funcionalidad completa y útil para el usuario final.

El proyecto culminará con una defensa técnica en la que se explicará el código, se demostrarán las funcionalidades y se responderá a preguntas técnicas profundas.

3. Requisitos técnicos

a) Interfaz de Usuario (Unidades 09-10)

- Mínimo 5 pantallas con navegación fluida.
- Uso de al menos 10 widgets diferentes (TextField, ListView.builder, Card, GridView, Drawer, BottomNavigationBar, etc.).
- Diseño 100% responsive (adaptable a móvil, tablet y web).
- Animaciones básicas o transiciones entre pantallas.
- Paleta de colores y tipografía consistentes en toda la app.

b) Gestión de estado y navegación (Unidad 11)

- Uso de gestión de estado avanzada (Provider, Bloc, Riverpod o GetX).
- Navegación con named routes y paso de argumentos entre pantallas.
- Validación de formularios en tiempo real.

c) Integración con Backend y Servicios Inteligentes (Unidad 12)

Conexión a una fuente de datos externa (obligatorio). El proyecto debe demostrar capacidad para consumir servicios remotos mediante HTTP/REST. Opciones:

Opciones de Backend/Servicios:

1. **APIs de Terceros (Tradicionales):**
    - Ejemplos: The Movie Database (TMDB), Spotify API (lectura), Google Maps API,

REST Countries API, News API, PokeAPI, etc.
    - Recomendación: Ideal para apps centradas en visualizar y buscar datos (catálogos, mapas, noticias).
2. **Firebase (Plataforma como Servicio - BaaS):**
    - Uso obligatorio de al menos dos servicios: Firestore (CRUD) + Authentication (login/registro).
    - Opcional: Firebase Storage para subida de archivos, o Cloud Functions para lógica de backend.
    - Recomendación: Perfecto para apps que requieren datos en tiempo real, user management y un backend robusto sin escribir código de servidor.
3. **Backend Propio (Full-Stack):**
    - Desarrollo de una API REST sencilla con FastAPI (Python), Express.js (Node.js), Spring Boot (Java) o similares.
    - Despliegue: La API debe estar desplegada en un servicio en la nube como Render, Railway, Vercel, Google Cloud Run o AWS Elastic Beanstalk.
    - Recomendación: Para alumnos que quieran demostrar habilidades full-stack y tener control total sobre la lógica del backend.
4. **APIs de Modelos de Lenguaje (LLMs):**
    - Uso de APIs de inteligencia artificial para agregar features inteligentes a la app.
    - Opciones Gratuitas/Freemium:
        - Google Gemini API: Capa gratuita generosa (60 RPM en mayo 2025). Ideal para generar texto, resúmenes, análisis de sentimiento, chatbots simples o clasificación de contenido. Ej: Una app de recetas que pida a Gemini generar una variante vegana.
        - Hugging Face Inference API: Ofrece acceso gratuito o de pago a miles de modelos de IA (texto, imagen, audio). Ej: Una app de diario personal que analice el sentimiento de las entradas con un modelo de Hugging Face.
        - OpenAI API (ChatGPT): Tiene crédito gratuito inicial para nuevos usuarios, luego es de pago.
    - Implementación: La app de Flutter enviará prompts o datos a la API del LLM (por ejemplo, mediante una POST request) y mostrará la respuesta generada en la UI.
    - ¡Importante! Esta opción no reemplaza la necesidad de un backend principal para los datos de la app. Debe ser una característica adicional que se suma a una de las opciones anteriores (1, 2 o 3). Por ejemplo:
        - App con backend en Firebase (Opción 2) que use Gemini (Opción 4) para generar descripciones automáticas de los perfiles de usuario.
        - App de lista de tareas con backend propio (Opción 3) que use un LLM para sugerir desgloses de tareas complejas.

Requisitos técnicos comunes (para cualquier opción):

- **Operaciones CRUD Completas o Lectura Robusta:**
    - Para APIs propias o Firebase: Implementar Create, Read, Update, Delete.
    - Para APIs de terceros de solo lectura (o LLMs): Demostrar una implementación robusta de las operaciones de lectura (GET), incluyendo paginación, filtrado, búsqueda y caching (por ejemplo, con el paquete flutter_cache_manager o dio_http_cache).

- **Manejo de Estado Asíncrono:**
  - Gestión profesional de los estados de carga, éxito y error durante las peticiones de red. Uso de indicadores de progreso (CircularProgressIndicator) y mensajes de error (SnackBar) para feedback al usuario.
- **Persistencia Local (Opcional pero valorado):**
  - Uso de shared_preferences, hive o sqflite para cachear respuestas de la API o persistir datos offline, mejorando la experiencia de usuario.

d) Temática libre pero realista

Algunas ideas:

- Con API de Terceros: App de películas/series (TMDB), app de música (Spotify API), mapa de restaurantes (Google Maps API), climatológica (OpenWeatherMap).
- Con Firebase: Red social, app de tareas colaborativas, chat simple, blog con comentarios.
- Con Backend Propio: Sistema de reservas, gestor de inventario, app de encuestas, portfolio personal con backend.

Ejemplo de Idea Integrando un LLM:

- Nombre de la App: "ReadSmart - Resúmenes Express"
- Backend Principal: Firebase Firestore (para guardar el historial de resúmenes del usuario).
- Feature con LLM: El usuario introduce la URL de un artículo de noticias. La app extrae el texto (o el usuario lo pega) y lo envía a la API de Gemini con el prompt: "Genera un resumen conciso de este texto en 3 bullet points en español". La app muestra el resumen generado y lo guarda en Firestore.
- Tecnologías: Flutter, http o dio para las peticiones, Firebase Core, Cloud Firestore, Firebase Auth.

4. CONDICIONES DE DESARROLLO Y HERRAMIENTAS PERMITIDAS

- Repositorio: El código fuente debe alojarse en un repositorio GitHub privado, debidamente compartido con el profesorado para su evaluación.
- **Backend:**
  - Se permite el uso de APIs de terceros, Firebase o el desarrollo de un backend propio (por ejemplo, con FastAPI en Python para crear endpoints sencillos de CRUD).
- **Asistentes de IA:**
  - Se permite consultar a asistentes de IA con capa gratuita (como Gemini, ChatGPT -versión gratuita-, Claude) exclusivamente para:
    - Resolver dudas conceptuales sobre Flutter o Dart.
    - Depurar errores específicos de código (stack traces).
    - Obtener sugerencias sobre implementaciones concretas y mejores prácticas.
  - Queda terminantemente prohibido:
    - Copiar y pegar soluciones integrales sin comprender su funcionamiento.
- Principio de Autonomía: El trabajo debe ser auténtico e individual. El alumno debe ser capaz de explicar y defender cada línea de código escrita.

5. Entregables

1. **Código Fuente: En un repositorio Git privado (GitHub), que incluya:**
    - Todo el proyecto de Flutter.
    - Si se desarrolla un backend propio, su código en una carpeta separada o en un repositorio adicional (con README para su despliegue).
2. **Defensa Oral Presencial (15-20 minutos):**
    - Explicación de la arquitectura de la app y decisiones de diseño.
    - Demo en vivo de todas las funcionalidades, incluida la interacción con el backend.
    - Ronda de preguntas técnicas sobre el código, la gestión de estado y la integración con el backend. El profesorado podrá solicitar modificaciones en tiempo real.

6. Criterios de Evaluación

Criterio	Peso	Descripción
Funcionamiento y Estabilidad	25%	La app no crashea y todas las funcionalidades trabajan correctamente. La navegación es fluida.
Integración con Backend	25%	Las operaciones con la API/Firebase/Backend propio funcionan sin errores. Manejo correcto de estados de carga y error.
Diseño de UI/UX y Responsive	20%	Interfaz atractiva, usable, coherente y adaptable a diferentes tamaños de pantalla.
Gestión de Estado y Arquitectura	15%	Uso eficiente y correcto de la solución de estado elegida. Código bien estructurado y separación de responsabilidades.
Documentación y Defensa Técnica	15%	Claridad y completitud de la documentación. Solidez y profundidad en las respuestas durante la defensa oral.

7. Calificación

- Apto: Cumple todos los requisitos técnicos de forma satisfactoria y defiende el proyecto con solvencia, demostrando comprensión profunda del código y las decisiones tomadas.
- No Apto: No cumple con los requisitos técnicos mínimos y/o no es capaz de explicar o modificar su propio código durante la defensa.

8. Ideas y pistas por tipo de backend

1. Para APIs de Terceros (Más sencillo para empezar)

a) The Movie Database (TMDB) - https://www.themoviedb.org/documentation/api

- Idea de App: Catálogo de películas y series. Los usuarios pueden buscar películas, ver detalles, guardar sus favoritos (en una lista local), y ver tráilers.
- Endpoints útiles:
  - GET /movie/popular (lista de películas populares)
  - GET /search/movie (búsqueda por título)
  - GET /movie/{id} (detalles de una película)
- Ventaja: API muy bien documentada, gratuita con registro, y no requiere OAuth para las operaciones básicas de lectura.

b) REST Countries API - https://restcountries.com/

- Idea de App: Aplicación educativa sobre países del mundo. Los usuarios pueden buscar países, filtrar por continente, y ver detalles como capital, población, bandera, etc.
- Endpoints útiles:
  - GET /v3.1/all (todos los países)
  - GET /v3.1/name/{name} (buscar por nombre)
  - GET /v3.1/region/{region} (filtrar por región)
- Ventaja: API muy simple, sin necesidad de clave API, perfecta para practicar la paginación, el filtrado y el manejo de listas largas.

c) The CocktailDB - https://www.thecocktaildb.com/api.php

- Idea de App: Libro de recetas de cócteles. Los usuarios pueden buscar cócteles por nombre o ingrediente, ver recetas completas e ingredientes, y marcar sus favoritos.
- Endpoints útiles:
  - GET /api/json/v1/1/search.php?s={nombre} (buscar por nombre)
  - GET /api/json/v1/1/filter.php?i={ingrediente} (filtrar por ingrediente)
- Ventaja: API divertida y fácil de usar, sin autenticación.

2. Para Firebase (Potente y muy integradp en Flutter)

a) App de Tareas Colaborativas

- Estructura de Firestore:

```
collections {
  users { # Colección de usuarios (se crea con Firebase Auth)
    userId: { # Documento por usuario
      name: "Juan"
    }
  }
  taskLists { # Colección de listas de tareas
    listId: { # Documento por lista
      name: "Proyecto Flutter",
      createdBy: userId, # Referencia al usuario
```

```
      users: [userId, ...] # Array de usuarios con acceso
    }
  }
  tasks { # Subcolección de tareas dentro de cada lista
    taskId: {
      title: "Investigar APIs",
      isCompleted: false,
      assignedTo: userId # Opcional: asignar tarea
    }
  }
}
```

- Funcionalidades: Registro/login de usuarios, crear listas compartidas, añadir/editar/eliminar/marcar tareas como completadas en tiempo real.

b) Mini-Blog o Foro Simple

- Estructura de Firestore:

```
collections {
  posts { # Colección de publicaciones
    postId: {
      title: "Mi primera app",
      content: "...",
      author: userId,
      createdAt: Timestamp,
      likes: 0
    }
  }
  comments { # Subcolección de comentarios dentro de cada post
    commentId: {
      text: "¡Genial!",
      author: userId,
      createdAt: Timestamp
    }
  }
}
```

- Funcionalidades: Crear posts, listarlos por fecha, dar "me gusta", comentar. Se puede añadir autenticación para que solo usuarios registrados puedan publicar.

3. Para Backend Propio con FastAPI (Python) - (Máximo control)

Idea: Sistema de Encuestas / Votaciones

- Endpoints de tu API (FastAPI):
  - POST /polls/ (Crear una nueva encuesta) - C
  - GET /polls/ (Obtener listado de todas las encuestas) - R
  - GET /polls/{id} (Obtener detalles de una encuesta y sus opciones) - R

    - POST /polls/{id}/vote (Votar en una opción de una encuesta) - U
    - DELETE /polls/{id} (Eliminar una encuesta - opcional con autenticación) - D
- Estructura de datos simple (JSON):
- python

```
# Para crear una encuesta (POST /polls/)
{
  "question": "¿Cuál es tu lenguaje de programación favorito?",
  "options": ["Dart", "Python", "JavaScript", "Java"]
}

# Respuesta al votar (POST /polls/{id}/vote)
{
  "option_index": 0 # Vota por la opción 0 ("Dart")
}
```

- Despliegue: Puedes desplegar tu API FastAPI de forma gratuita y sencilla en servicios como:
    - Render: Conecta tu repositorio GitHub y despliega automáticamente.
    - Railway: Similar a Render, muy fácil de usar.
    - PythonAnywhere: Otra opción popular para aplicaciones Python.

Consejo Técnico para FastAPI: Usa uvicorn como servidor y pydantic para la validación de datos. Es muy sencillo de configurar.

🎯 **Recomendación Final**

- Si es tu primera vez integrando un backend: Empieza con una API de terceros como The MovieDB (TMDB) o REST Countries. Son las más sencillas.
- Si quieres aprender algo muy demandado y potente: Elige Firebase. La integración con Flutter es excelente y es muy usado en proyectos reales.
- Si te apetece un reto completo y tener control total: Lánzate con un backend propio en FastAPI. Aprenderás mucho sobre cómo funcionan las APIs por dentro.