Crear la Aplicación ConectAsoc con las características siguientes:

Como experto en flutter, dart y los servicios utilizados en esta app, tendrás en cuenta siempre que tanto el código existente como el nuevo debe cumplir:

UI moderno y profesional

Usabilidad optima

Clean architecture

Patrón BloC: paquete Flutter Bloc

Utilización paquete dartz (Either/Failure)

Uso de Firestore

Utilización de Cloudinary en lugar de Firebase Storage

Utilización y complección de archivos 'barrel'

Generación de textos en multidioma (actualización archivos app\_xx.arb)

cached\_network\_image: ^3.3.0

cloud\_firestore: ^6.0.2

collection: ^1.19.1

crypto: ^3.0.3

cupertino\_icons: ^1.0.2

dartz: ^0.10.1

equatable: ^2.0.5

firebase\_analytics: ^12.0.2

firebase\_auth: ^6.1.0

firebase\_core: ^4.1.1

firebase\_messaging: ^16.0.2

flutter:

sdk: flutter

flutter\_bloc: ^9.1.1

flutter\_form\_builder: ^10.2.0

flutter\_local\_notifications: ^19.4.2

flutter\_localizations:

sdk: flutter

flutter\_quill: ^11.4.2

flutter\_svg: ^2.0.9

form\_builder\_validators: ^11.2.0

get\_it: ^8.2.0

go\_router: ^16.2.2

http: ^1.1.0

image: ^4.1.3

image\_cropper: ^11.0.0

image\_picker: ^1.0.4

injectable: ^2.3.2

intl: ^0.20.2

logger: ^2.0.2+1

path: ^1.8.3

path\_provider: ^2.1.1

permission\_handler: ^12.0.1

photo\_view: ^0.15.0

rxdart: ^0.28.0

shared\_preferences: ^2.5.3

uuid: ^4.1.0

Descripción General

Estamos desarrollando una aplicación móvil Flutter llamada ConectAsoc - un portal de gestión de asociaciones con las siguientes características principales

Funcionalidades Core

Sistema de Autenticación Multi-Modo

Registro de usuarios con Firebase Authentication

Login con email/password y verificación de email

Modo de solo lectura (sin autenticación)

Gestión de sesiones con BLoC pattern

Gestión de Asociaciones

CRUD completo de asociaciones

Listado con búsqueda

Creación durante el registro o posterior

Eliminación con validaciones (no permitir si tiene usuarios)

Actualización de datos y logos

Gestión de Usuarios y Perfiles

CRUD completo de usuarios

Perfiles de usuario editables

Foto de perfil con recorte

Membresías a asociaciones

Roles (admin, miembro)

Listado de usuarios por asociación

Arquitectura Técnica

Clean Architecture con 3 Capas

Capa de Dominio (Domain Layer)

Entities: Modelos de negocio puros

Repositories: Interfaces abstractas

UseCases: Lógica de negocio aislada

Capa de Datos (Data Layer)

Models: Implementaciones con serialización JSON

DataSources: Remote (Firestore) y Local

Repository Implementations: Implementan interfaces del dominio

Capa de Presentación (Presentation Layer)

Pages: Pantallas de la UI

Widgets: Componentes reutilizables

BLoC: Gestión de estado

Patrón BLoC con flutter\_bloc

Implementar BLoCs para:

AuthBloc: Autenticación y sesiones

AssociationBloc: Gestión de asociaciones

UserBloc: Gestión de usuarios y perfiles

Cada BLoC debe:

Emitir estados inmutables

Manejar eventos específicos

Usar BlocProvider para inyección

Implementar BlocListener para efectos secundarios

Implementar BlocBuilder para UI reactiva

Manejo de Errores con Dartz users\_repository\_impl.dart:34-44

Usar Either<Failure, Success> en todos los repositorios y use cases

Tipos de Failure:

ServerFailure: Errores de backend/red

ValidationFailure: Errores de validación

Pattern matching con fold() para manejar resultados

Gestión de Imágenes con Cloudinary users\_repository\_impl.dart:54-80

NO usar Firebase Storage.

Implementar CloudinaryService con:

uploadImage(): Subida de imágenes con tipos (avatar, logoAssociation)

deleteImage(): Eliminación por publicId

getPublicIdFromUrl(): Extracción de ID desde URL

Gestión de imágenes antiguas: eliminar después de subir la nueva exitosamente

Internacionalización (i18n) app\_es.arb:1-13

Soporte para 3 idiomas: Catalán (ca), Inglés (en), Español (es)

Archivos ARB requeridos:

lib/l10n/app\_ca.arb

lib/l10n/app\_en.arb

lib/l10n/app\_es.arb

Estructura de cada archivo: app\_es.arb:1-20

{

"@@locale": "es",

"keyName": "Traducción",

"@keyName": {

"placeholders": {

"paramName": {}

}

}

}

Categorías de strings a incluir:

Navegación (appTitle, homePage, associations, users, myProfile)

Autenticación (login, logout, createAccount, welcomeReadOnlyTitle)

Perfiles (editProfile, name, lastname, email, phone, language)

Asociaciones (createAssociation, deleteAssociation, shortName, longName)

Validaciones y errores (invalidEmailFormat, unexpectedError)

Acciones (search, cancel, retry, delete, undo)

Configuración en MaterialApp:

localizationsDelegates: Incluir AppLocalizations.localizationsDelegates

supportedLocales: AppLocalizations.supportedLocales

locale: Dinámico según preferencia del usuario autenticado

localeListResolutionCallback: Prioridad: usuario > dispositivo > fallback

Archivos Barrel

Crear archivos barrel.dart o [feature].dart en cada feature para exportar todas las clases públicas:

// lib/features/auth/presentation/presentation.dart

export 'bloc/bloc.dart';

export 'pages/pages.dart';

export 'widgets/widgets.dart';

Inyección de Dependencias

Implementar injection\_container.dart con GetIt:

Registrar todos los BLoCs como factories

Registrar repositories como singletons

Registrar data sources como singletons

Registrar use cases como singletons

Inicializar con await init() antes de runApp()

UI/UX Moderno y Profesional

Theme Configuration

Material 3 con:

primarySwatch: Colors.blue

useMaterial3: true

AppBar Theme:

Elevation: 0

Center title: true

Background: Colors.blue

Foreground: Colors.white

Input Decoration Theme:

Filled: true con Colors.grey[50]

Border radius: 12px

Focused border: Blue con width 2

Content padding: 16px horizontal y vertical

Elevated Button Theme:

Elevation: 2

Padding: 24px horizontal, 16px vertical

Border radius: 12px

Card Theme:

Elevation: 2

Border radius: 12px

Componentes UI Clave

FloatingActionButton para crear nuevas entidades association\_list\_page.dart:34-43

SnackBar Service global para notificaciones association\_list\_page.dart:54-60

CachedNetworkImage para optimizar carga de imágenes association\_list\_page.dart:3

Diálogos de confirmación para acciones destructivas

Búsqueda en tiempo real en listados

Backend: Firebase

Firebase Authentication: Gestión de usuarios

Cloud Firestore: Base de datos NoSQL

Colección users: Perfiles de usuario

Colección associations: Datos de asociaciones

Colección memberships: Relación usuarios-asociaciones

Navegación GoRouter.

Sistema de rutas nombradas con AppRouter:

RouteNames.splash: Pantalla inicial

RouteNames.welcome: Pantalla de bienvenida (no autenticado)

RouteNames.home: Pantalla principal (autenticado)

RouteNames.associationEdit: Edición/creación de asociación

Navegación con GoRouter y argumentos

Flujo de Registro Complejo register

El registro debe:

Crear usuario en Firebase Authentication

Determinar si crea nueva asociación o se une a existente

Si crea asociación: llamar a CreateAssociationUseCase

Asignar rol (admin si crea, asociado si se une)

Crear documento de usuario en Firestore con membresía

Validar datos completos antes de proceder

Validaciones Importantes

Email con formato válido

Nombres corto y largo obligatorios para asociaciones

No permitir eliminar asociaciones con usuarios asignados

Gestión de errores traducidos según idioma activo

Notes

Este prompt describe una aplicación completa con arquitectura limpia, gestión de estado profesional con BLoC, internacionalización completa en 3 idiomas, y servicios cloud modernos (Firebase + Cloudinary).

La estructura sigue las mejores prácticas de Flutter con separación clara de responsabilidades, manejo robusto de errores con Either/Failure, y una UI moderna con Material 3.

El sistema de autenticación incluye un modo de solo lectura único que permite explorar contenido sin registro.

El árbol de ficheros es

├── .dart\_tool

├── package\_config.json

├── package\_graph.json

├── version

├── dartpad

├── web\_plugin\_registrant.dart

├── extension\_discovery

├── devtools.json

├── vs\_code.json

├── flutter\_build

├── dart\_plugin\_registrant.dart

├── 24d31f5bcd598d52a33554adb56ec8e9

├── .filecache

├── app.dill

├── dart\_build\_result.json

├── dart\_build.d

├── dart\_build.stamp

├── debug\_android\_application.stamp

├── flutter\_assets.d

├── gen\_dart\_plugin\_registrant.stamp

├── gen\_l10n\_inputs\_and\_outputs.json

├── gen\_localizations.d

├── gen\_localizations.stamp

├── install\_code\_assets.d

├── install\_code\_assets.stamp

├── kernel\_snapshot\_program.d

├── kernel\_snapshot\_program.stamp

├── native\_assets.json

├── outputs.json

├── .idea

├── modules.xml

├── workspace.xml

├── libraries

├── Dart\_SDK.xml

├── KotlinJavaRuntime.xml

├── runConfigurations

├── main\_dart.xml

├── .vscode

├── launch.json

├── assets

├── fonts

├── Roboto-Bold.ttf

├── Roboto-Regular.ttf

├── images

├── generi\_asoc-256.png

├── generi\_asoc-32.png

├── generi\_asoc-80.png

├── logo\_conectasoc\_t.png

├── logo\_conectasoc.png

├── lib

├── firebase\_options.dart

├── injection\_container.dart

├── main.dart

├── app

├── router

├── app\_router.dart

├── route\_names.dart

├── router.dart

├── core

├── constants

├── cloudinary\_config.dart

├── errors

├── errors.dart

├── exceptions.dart

├── failures.dart

├── services

├── image\_picker\_service.dart

├── local\_storage\_service.dart

├── utils

├── validators.dart

├── features

├── articles

├── data

├── models

├── article\_model.dart

├── category\_model.dart

├── models.dart

├── repositories

├── article\_repository\_impl.dart

├── domain

├── entities

├── article\_entity.dart

├── article\_filter\_entity.dart

├── entities.dart

├── repositories

├── article\_repository.dart

├── usecases

├── create\_article\_usecase.dart

├── delete\_article\_usecase.dart

├── get\_article\_by\_id\_usecase.dart

├── get\_articles\_usecase.dart

├── get\_categories\_usecase.dart

├── get\_subcategories\_usecase.dart

├── update\_article\_usecase.dart

├── usecases.dart

├── presentation

├── bloc

├── article\_bloc.dart

├── article\_event\_bloc.dart

├── article\_state\_bloc.dart

├── bloc.dart

├── edit

├── article\_edit\_bloc.dart

├── article\_edit\_event.dart

├── article\_edit\_state.dart

├── pages

├── article\_edit\_page.dart

├── pages.dart

├── associations

├── data

├── datasources

├── association\_remote\_datasource.dart

├── models

├── association\_model.dart

├── repositories

├── association\_repository\_impl.dart

├── domain

├── entities

├── association\_entity.dart

├── entities.dart

├── repositories

├── association\_repository.dart

├── usecases

├── create\_association\_usecase.dart

├── delete\_association\_usecase.dart

├── get\_all\_associations\_usecase.dart

├── get\_association\_by\_id\_usecase.dart

├── undo\_delete\_association\_usecase.dart

├── update\_association\_usecase.dart

├── usecases.dart

├── presentation

├── bloc

├── association\_bloc.dart

├── association\_event.dart

├── association\_state.dart

├── bloc.dart

├── edit

├── association\_edit\_bloc.dart

├── association\_edit\_event.dart

├── association\_edit\_state.dart

├── pages

├── association\_edit\_page.dart

├── association\_list\_page.dart

├── pages.dart

├── widgets

├── association\_avatar\_widget.dart

├── auth

├── data

├── datasources

├── auth\_local\_datasource.dart

├── auth\_remote\_datasource.dart

├── datasources.dart

├── models

├── models.dart

├── user\_model.dart

├── repositories

├── auth\_repository\_impl.dart

├── domain

├── domain.dart

├── entities

├── entities.dart

├── local\_user\_entity.dart

├── membership\_entity.dart

├── user\_entity.dart

├── repositories

├── auth\_repository.dart

├── repositories.dart

├── usecases

├── login\_usecase.dart

├── logout\_usecase.dart

├── register\_usecase.dart

├── save\_local\_user\_usecase.dart

├── usecases.dart

├── presentation

├── presentation.dart

├── bloc

├── auth\_bloc.dart

├── auth\_event.dart

├── auth\_state.dart

├── bloc.dart

├── local\_user

├── local\_user\_setup\_bloc.dart

├── local\_user\_setup\_event.dart

├── local\_user\_setup\_state.dart

├── register

├── register\_bloc.dart

├── register\_event.dart

├── register\_state.dart

├── pages

├── local\_user\_setup\_page.dart

├── login\_page.dart

├── pages.dart

├── register\_page.dart

├── splash\_page.dart

├── verification\_page.dart

├── welcome\_page.dart

├── widgets

├── auth\_text\_field\_widget.dart

├── login\_form\_widget.dart

├── register\_form\_widget.dart

├── widgets.dart

├── home

├── presentation

├── bloc

├── bloc.dart

├── home\_bloc.dart

├── home\_event\_bloc.dart

├── home\_state\_bloc.dart

├── pages

├── home\_page.dart

├── pages.dart

├── widgets

├── association\_provider\_widget.dart

├── home\_drawer.dart

├── widgets.dart

├── notifications

├── presentation

├── bloc

├── notification\_bloc.dart

├── users

├── data

├── datasources

├── user\_remote\_datasource.dart

├── repositories

├── users\_repository\_impl.dart

├── domain

├── entities

├── entities.dart

├── i\_user.dart

├── profile\_entity.dart

├── user.dart

├── repositories

├── repositories.dart

├── users\_repository.dart

├── usecases

├── create\_user\_usecase.dart

├── delete\_user\_usecase.dart

├── get\_all\_users\_usecase.dart

├── get\_user\_by\_id\_usecase.dart

├── get\_users\_by\_association\_usecase.dart

├── join\_association\_usecase.dart

├── update\_user\_usecase.dart

├── usecases.dart

├── presentation

├── bloc

├── bloc.dart

├── user\_bloc.dart

├── user\_event.dart

├── user\_state.dart

├── edit

├── user\_edit\_bloc.dart

├── user\_edit\_event.dart

├── user\_edit\_state.dart

├── list

├── user\_list\_bloc.dart

├── user\_list\_event.dart

├── user\_list\_state.dart

├── profile

├── profile\_bloc.dart

├── profile\_event.dart

├── profile\_state.dart

├── pages

├── join\_association\_page.dart

├── pages.dart

├── profile\_page.dart

├── user\_edit\_page.dart

├── user\_list\_page.dart

├── l10n

├── app\_ca.arb

├── app\_en.arb

├── app\_es.arb

├── app\_localizations\_ca.dart

├── app\_localizations\_en.dart

├── app\_localizations\_es.dart

├── app\_localizations.dart

├── l10n.yaml

├── services

├── cloudinary\_service.dart

├── snackbar\_service.dart

├── widgets

├── test\_cloudinary\_widget.dart

Teniendo en cuenta esto, continuamos con la nueva funcionalidad, de la que parte ya está implementada:

La pantalla home realmente muestra los artículos publicados.

Será una lista de artículos de actuales a más antiguos.

Tendrá un filtro de búsqueda

Tendrá una barra abajo donde aparezcan las categorías. Se desplazará horizontalmente si hay más categorías de las que caben en la pantalla.

Al seleccionar una categoría sólo se mostrarán los de esa categoría.

La lista de subcategorías asociadas a la categoría seleccionada sustituirá la barra de abajo. Al seleccionar una subcategoría se mostrarán sólo los de esa categoría/subcategoría

Cada tarjeta tendrá

a la izquierda,

el cover

resto

título, 2 líneas máximo

abstract, 3 líneas máximo (añadir ... al final si se trunca el texto)

categoría/subcategoría (pequeño)

fecha publicación (pequeño)

estado (pequeño)

Los artículos a mostrar serán

Artículos en situación de publicado y fecha del día entre fecha de efecto y fecha de expiración

- superadmin ve todos los articulos

- admin ve los artículos genéricos y los de su asociacion

- editor, asociado ve los artículos genéricos y los de su asociacion

- los usuarios locales ven los artículos genéricos y los de su asociacion

- los usuarios no logueados ni locales ven los artículos genéricos

Superadmin, admin y editores tendrás un switch que pasa de lectura a edición.

Cuando esté en edición, cada artículo que tenga permitido tendrá un icono (lápiz) de editar

En edición se mostrará un botón flotante para crear un artículo nuevo.

Cada tarjeta de la lista tendrá un color de fondo según el estado

redacción: verde clarito

publicado: normal

revisión: amarillo clarito

expirado: naranja clarito

anulado: rojo clarito

Se mostrarán todos los artículos, sea cual sea el estado, para los que estén autorizados

superadmin puede crear, editar y borrar cualquier artículo

admin puede crear, editar y borrar cualquier artículo de su asociación

editor puede crear, editar y borrar cualquier artículo de su asociación del que sea autor

Habrá artículos creados por:

superadmin: artículos generales. Todos, de cualquier asociación, pueden leer, sólo superadmin editar

admin (de una asociación): todos los de la asociación los pueden leer, sólo admin y superadmin editar

editor (de una asociación): todos los de la asociación los pueden leer, editor, admin y superadmin editar

asociado (de una asociación): sólo pueden leer de la asociación y generales. No pueden editar nada.

Un artículo estará compuesto por

- Título (texto enriquecido de 100 caracteres)

- abstract (texto enriquecido de 200 caracteres)

- Imagen cover del artículo

- Ubicación

- atributos

userid del creador (puede ser superadmin,admin o editor)

asocid de userid

email de User,

profile del User,

name User,

lastName User,

avatar User,

longNameAsociation,

shortNameAsociation,

itemsArticle,

categoría

información

noticias

actas

subcategoría

noticias

municipio

urbanización

actas

asambleas

reuniones

información

servicios

cultura

reuniones

asambleas

clase

fecha de creación

fecha de modificación

fecha de borrado

fecha de publicación (posterior a hoy)

fecha de efecto a partir de la cual es visible (posterior a la de publicación)

fecha de expiración a partir de la cual deja de estar visible ( posterior a la de publicación, se contempla que pueda no vencer)

estado:

redacción

publicado

revisión

expirado

anulado

- Secciones

cada sección está formada por

- Imagen

- Campo de texto enriquecido sin limite

Puede tener imagen y/o texto

Al crear o editar un artículo

obligatorio

cover

título

abstract

categoría

subcategoría

fecha de publicación (igual o posterior a hoy)

fecha de efecto a partir de la cual es visible (igual o posterior a la de publicación)

fecha de expiración a partir de la cual deja de estar visible (igual o posterior a la de publicación, se contempla que pueda no vencer)

opcional

secciones

las secciones pueden tener rellenos el campo de texto y/o la imagen

Las secciones han de ser reordenables, moviéndolas de abajo arriba y viceversa

Los campos de texto han de ser de tipo enriquecido (flutter\_quill)

Las imágenes, cover y secciones se han de gestionar como los avatares de usuario y los logos de asociación. Cámara o galería con crop y resize.

Se guardarán en cloudinary, con el servicio y herramientas ya implementadas en la app

Los textos se guardarán en el idioma en que lo escriba el usuario.

Se usará el paquete flutter translator, que traducirá, cuando el idioma sea distinto (o siempre si no está claro) los artículos al idioma seleccionado por el usuario.