DuocUC® ESCUELA DE INFORMÁTICA Y TELECOMUNICACIONES

Modelo de Base de Datos (Firebase Firestore)

Integrantes:
David Mardones
Arturo Rojas
Maria Morales

Índice

Introducción

La base de datos utilizada para el proyecto AvicolApp está implementada en Firebase Firestore, una base de datos NoSQL en la nube, de tipo documento-colección, que permite almacenar, sincronizar y consultar datos en tiempo real.

Firestore fue elegida por su escalabilidad, bajo mantenimiento y facilidad de integración con Ionic + Angular, el framework usado para desarrollar la aplicación móvil

Objetivo del diseño

El objetivo de la base de datos es registrar y organizar la recolección diaria de huevos en los diferentes galpones de la granja avícola, de forma simple, rápida y confiable.

Cada recolección incluye los distintos tipos de huevos (incubables, sucios, trizados y dobles), separados por origen (nido o piso), y registrados por fecha, galpón, turno y recolector

¿Por qué Firebase?

Se eligió Firebase Firestore por las siguientes razones:

- Sincronización en tiempo real: Los datos se actualizan automáticamente entre los dispositivos de los recolectores y el panel del encargado.
- Escalabilidad: Permite manejar grandes volúmenes de registros sin pérdida de rendimiento.
- Integración nativa con Firebase Authentication: Facilita la gestión de usuarios y roles.
- Modelo flexible: Al ser NoSQL, permite adaptarse fácilmente a cambios en la estructura del proyecto.
- Acceso multiplataforma: Es compatible con Ionic/Angular, el framework usado para la app móvil.

Tipo de Base de Datos

La base de datos es NoSQL (No relacional) y está basada en un modelo jerárquico de colecciones y documentos.

Esto permite organizar la información en niveles lógicos que representan:

- Granja (Farm)
- Galpón (Shed)
- Registros de huevos (Egg Records)

Firestore no usa tablas ni relaciones estrictas como SQL, sino que trabaja con documentos JSON que pueden anidar subcolecciones

Estructura del Modelo

La base de datos está compuesta por varias colecciones principales y subcolecciones, las cuales representan las entidades y relaciones del negocio avícola.

Colección: users

Guarda los datos de los usuarios registrados en el sistema, tanto recolectores como encargados.

Campo	Tipo	Descripción
uid	String	Identificador único del usuario (Firebase Auth)
name	String	Nombre del usuario
role	String	Rol asignado (encargado o recolector)
createdAt	Timestap	Galpón asignado (solo para recolectores)
assignedShed	String/Null	Fecha de creacion del usuario
email	String	Correo electronico

Colección: Farms

Representa cada granja registrada en el sistema.

Campo	Tipo	Descripción
farmId	ID (String)	Identificador único de la granja
name	String	Nombre de la granja

Subcolección: sheds

Contiene los galpones o sectores de cada granja.

Campo	Tipo	Descripción
shedId	ID (String)	Identificador del galpón (ej. “S1A”)
shedLabel	String	Nombre completo del galpón
sectorId	Number	Número del sector asociado

Subcolección: eggRecords

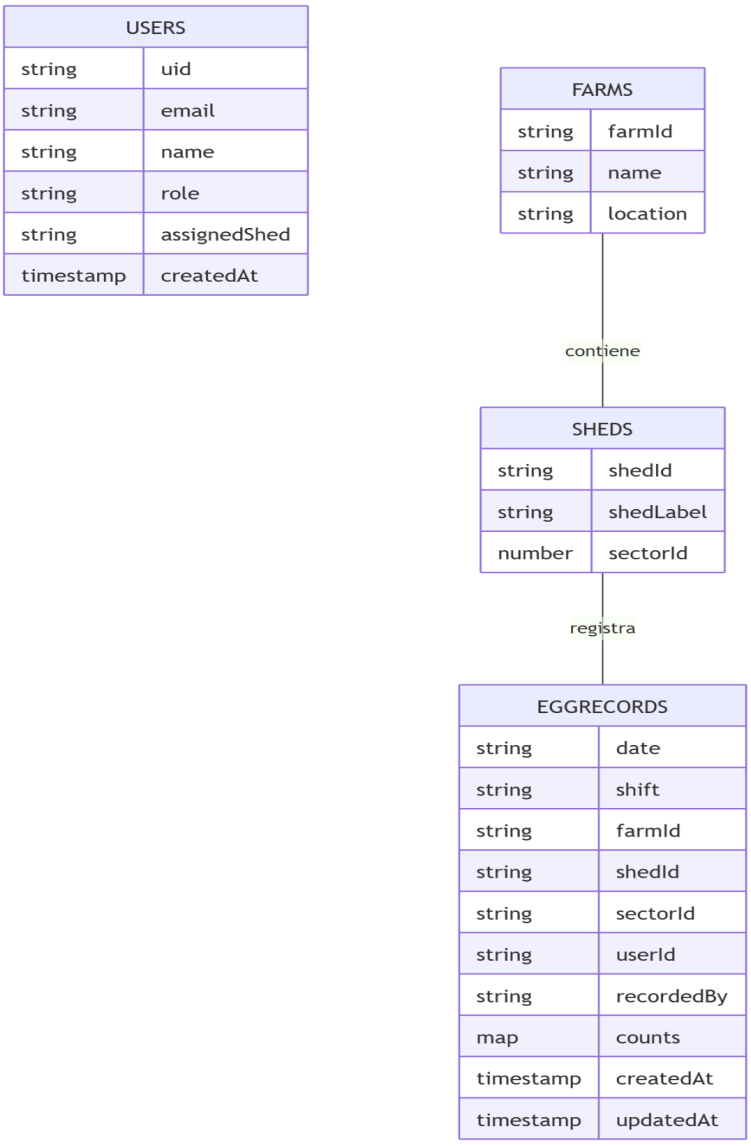
Guarda los registros diarios de recolección de huevos.
Cada documento representa un registro para un día y turno específico.

Campo	Tipo	Descripción
date	String	Fecha del registro
counts	Map	Contiene las cantidades por tipo de huevo
farmId	String	ID de la granja
shedId	String	ID del galpón
shedLabel	String	Nombre completo del galpón
sectorId	Number	Sector donde se realizó la recolección
userId	String	ID del usuario (Firebase Auth)
userId	String	Nombre del recolector
createdAt	Timestamp	Fecha y hora de creacion
updatedAt	Timestamp	Fecha y hora de actualizacion

Relaciones entre colecciones

Relación	Descripción
`users.uid` →`eggRecords.userId`	Identifica qué usuario registró la recolección
`farms.farmId` → `eggRecords.farmId`	Asocia el registro con la granja correspondiente
`sheds.shedId` → `eggRecords.shedId`	Indica el galpón donde se realizó la recolección

Diagrama del modelo



Justificación del modelo

El modelo se diseñó para reflejar la estructura real de la granja avícola, utilizando Firebase Firestore por su naturaleza jerárquica y escalable. La organización en colecciones y subcolecciones (Farms → sheds → eggRecords) permite consultas rápidas y reduce redundancia, mientras que la colección Users centraliza la gestión de roles y permisos. Este enfoque facilita la integración con autenticación, soporta datos en tiempo real y se adapta a cambios sin necesidad de migraciones complejas.

Conclusión

El modelo implementado en Firebase Firestore cumple con los objetivos del proyecto al ofrecer una estructura flexible, escalable y alineada con la operación real de la granja avícola. La organización jerárquica en colecciones y subcolecciones facilita la gestión de datos, optimiza las consultas y permite integrar funcionalidades como autenticación y control de roles.