AgroClima Jalapa

Manual técnico

Arquitectura, configuración, despliegue y mantenimiento del sistema

El clima de su parcela, antes de que pase.

Aplicación versión 1.1.0 · Android 8.0 o superior · octubre de 2026

Elaborado por

Marlon Steven Quintana Recinos

Universidad Mariano Gálvez de Guatemala, Campus Jutiapa
Ingeniería en Sistemas de Información
Jalapa, Guatemala

Información del documento

Dato	Detalle
Documento	Manual técnico
Sistema	AgroClima Jalapa: aplicación Android y ciclo automático en la nube
Versión del sistema	1.1.0 (compilación 2)
Versión del documento	1.1
Dirigido a	Desarrolladores, administradores del sistema y personal técnico que dé mantenimiento a la aplicación
Repositorio	https://github.com/StevenQuintanaGT/Agroclima_Jalapa
Elaborado por	Marlon Steven Quintana Recinos
Fecha	Octubre de 2026

Historial de versiones

Versión	Fecha	Autor	Descripción
1.0	Octubre de 2026	Marlon Steven Quintana Recinos	Primera versión del manual.
1.1	Octubre de 2026	Marlon Steven Quintana Recinos	Versión 1.1.0: colección encuestas, sus reglas y la exportación de respuestas.

Documentos relacionados

Documento	Contenido
Manual de instalación	Descarga e instalación de la aplicación en el teléfono del productor. https://stevenquintanagt.github.io/Pag_Agroclima/manuales/manual-de-instalacion.pdf
Manual de usuario	Uso de la aplicación por parte del productor. https://stevenquintanagt.github.io/Pag_Agroclima/manuales/manual-de-usuario.pdf

Contenido

1. Introducción **6**
1.1 Objetivo del manual 6
1.2 A quién va dirigido 6
1.3 Alcance 6
1.4 Convenciones 6

2. Descripción general del sistema **6**
2.1 Tecnologías 7

3. Arquitectura **8**
3.1 Patrones de diseño 9
3.2 Módulos 9

4. Estructura del repositorio **10**
4.1 Archivos de configuración principales 11

5. Modelo de datos **12**
5.1 Valores de las enumeraciones 13
5.2 Índices compuestos 14
5.3 Borrado en cascada 14

6. Ciclo automático **14**
6.1 Pasos de cada ejecución 15
6.2 Flujos de GitHub Actions 16
6.3 Parámetros del ciclo 17
6.4 Motor de umbrales 17
6.5 Control de duplicados y envío 18

7. Seguridad **18**
7.1 Reglas de Firestore 19
7.2 Claves y archivos que no se suben al repositorio 20

8. Configuración del entorno de desarrollo **20**
8.1 Requisitos 20
8.2 Pasos de configuración 21
8.3 Trabajo con los emuladores de Firebase 21

9. Compilación y publicación del APK **21**
9.1 Crear la llave de firma (una sola vez) 21
9.2 Compilar 22
9.3 Publicar 22

10. Despliegue y operación en la nube **22**
10.1 Proyecto de Firebase 22
10.2 Secretos de GitHub 23
10.3 Carga inicial de umbrales 24

10.4 Prueba de notificaciones..... 24
10.5 Exportar las respuestas de la encuesta..... 24
11. Pruebas..... 25
12. Mantenimiento..... 25
13. Solución de problemas..... 26
14. Glosario técnico..... 26
15. Soporte y referencias..... 27

Índice de figuras

Figura 1. Diagrama de despliegue del sistema..... 7
Figura 2. Arquitectura del sistema por capas y servicios..... 8
Figura 3. Módulos y componentes del sistema..... 10
Figura 4. Estructura de colecciones de Cloud Firestore..... 12
Figura 5. Diagrama de actividades del ciclo automático..... 15
Figura 6. Historial de ejecuciones en GitHub Actions..... 16
Figura 7. Mecanismos de seguridad del sistema..... 18
Figura 8. Reglas de seguridad publicadas en la consola de Cloud Firestore..... 19
Figura 9. Consola del proyecto de Firebase en el plan Spark..... 23
Figura 10. Secretos del repositorio en GitHub..... 24

1. Introducción

Este manual describe cómo está construido AgroClima Jalapa y qué se necesita para configurarlo, compilarlo, ponerlo en funcionamiento y darle mantenimiento. Reúne la arquitectura, la organización del código, el modelo de datos, el ciclo automático de alertas, la seguridad y los procedimientos de operación.

1.1 Objetivo del manual

Permitir que una persona con conocimientos de desarrollo de software pueda entender el sistema, reproducir el entorno de trabajo, publicar una versión nueva de la aplicación y mantener el servicio en la nube sin depender del autor.

1.2 A quién va dirigido

A desarrolladores, administradores del sistema y personal técnico. Se asume conocimiento básico de Flutter y Dart, JavaScript con Node.js, Git y GitHub, y de los servicios de Firebase.

1.3 Alcance

El manual cubre la versión 1.1.0 de la aplicación Android, el código del ciclo automático en la carpeta functions, los flujos de trabajo de GitHub Actions y la configuración del proyecto de Firebase. No cubre la instalación por parte del productor ni el uso diario, que se explican en los manuales de instalación y de usuario.

1.4 Convenciones

- Los nombres de archivos, carpetas, campos y comandos aparecen en **negrita** o en recuadros de código con letra de ancho fijo.
- Los códigos como D-37 o RNF-04 remiten a las decisiones de diseño (docs/DECISIONES.md) y a los requisitos (docs/REQUISITOS.md) del repositorio.
- Las rutas se escriben desde la raíz del repositorio.

2. Descripción general del sistema

AgroClima Jalapa es una aplicación móvil para productores de los siete municipios de Jalapa que muestra el clima de cada parcela y avisa cuando el pronóstico supera lo que aguanta el cultivo. El sistema no tiene un servidor propio: la aplicación se comunica directamente con servicios en la nube, y un ciclo automático que se ejecuta cada tres horas en GitHub Actions consulta el clima, evalúa los umbrales y envía las notificaciones.

La Figura 1 muestra los nodos del sistema: el teléfono con la aplicación y su copia local de datos, los servicios de Firebase y GitHub en la nube, y los dos servicios externos, OpenWeather para el clima y Esri para el mapa base.

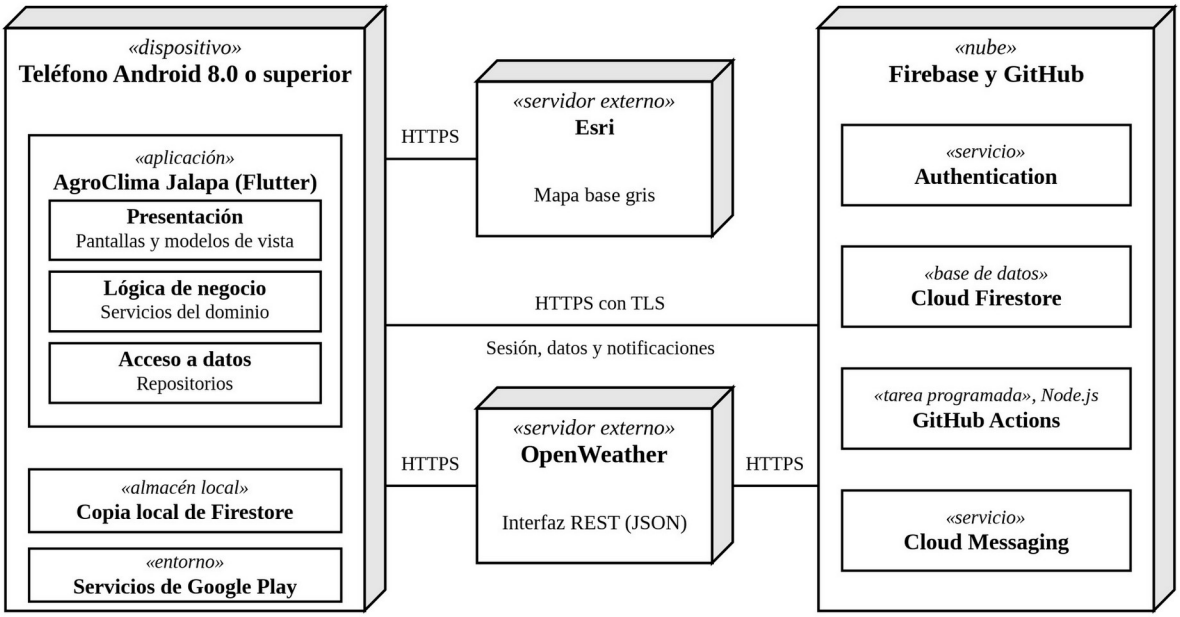


Figura 1. *Diagrama de despliegue del sistema.*

2.1 Tecnologías

Capa	Tecnología	Uso
Aplicación	Flutter y Dart	Aplicación Android, mínimo Android 8.0 (minSdk 26). Paquete gt.umg.agroclima_jalapa.
Estado	provider	Patrón MVVM: cada pantalla con su modelo de vista e inyección de dependencias.
Navegación	go_router	Rutas, guardas de sesión y barra inferior de cinco destinos.
Autenticación	Firebase Authentication	Correo y contraseña, y cuenta de Google.
Base de datos	Cloud Firestore	Base de datos documental con persistencia sin conexión en el teléfono.
Notificaciones	Firebase Cloud Messaging	Envío de avisos; canales locales con flutter_local_notifications.
Ciclo automático	Node.js 22 y firebase-admin	Código en functions/ que ejecuta GitHub Actions cada tres horas.
Clima	OpenWeather, plan gratuito	Clima actual, pronóstico de 5 días cada 3 horas y capas del mapa.
Mapas	flutter_map con teselas de Esri	Mapa satelital y gris sin clave de acceso.
Ubicación	geolocator y geocoding	Ubicación del teléfono y búsqueda de lugares.

Capa	Tecnología	Uso
Reportes	pdf, printing, excel y share_plus	Generación de PDF, Excel y CSV en el teléfono, impresión y compartir.
Gráficas	fl_chart	Gráficas de temperatura y lluvia.
Versiones	Git y GitHub	Repositorio, flujos de trabajo y secretos.

El sistema está diseñado para costo cero. Firebase se mantiene en el plan gratuito Spark y OpenWeather en su plan gratuito. No se deben activar servicios que exijan el plan Blaze o una cuenta de facturación, como Cloud Functions, Cloud Scheduler o Google Maps (decisión D-37).

3. Arquitectura

La aplicación sigue una arquitectura cliente sin servidor propio. Dentro del teléfono se organiza en tres capas que solo se comunican con la inmediata: presentación, lógica de negocio y acceso a datos. Los repositorios son los únicos que tocan Firestore, OpenWeather o el almacenamiento local, como se ve en la Figura 2.

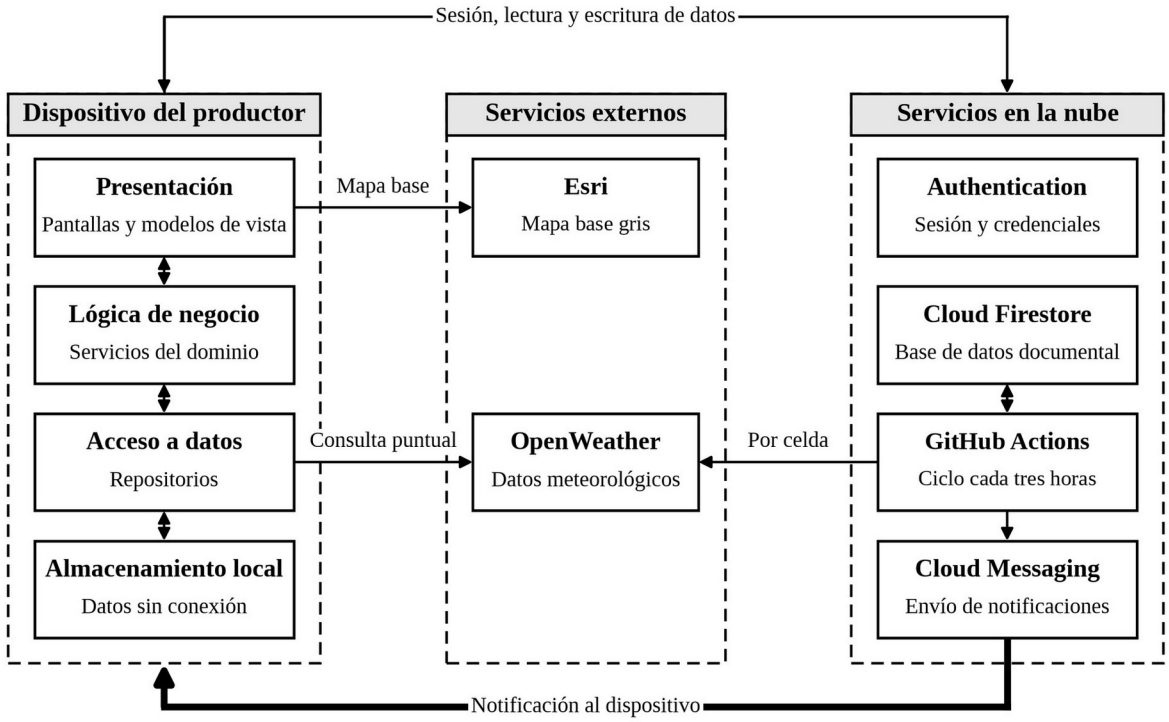


Figura 2. *Arquitectura del sistema por capas y servicios.*

Capa	Carpeta	Responsabilidad
Presentación	lib/pantallas, lib/componentes	Pantallas y modelos de vista. No contienen reglas de negocio ni consultas.
Lógica de negocio	lib/servicios	Reglas del dominio: registro de parcelas, validación geográfica, vigencia del dato, reportes, avisos.
Acceso a datos	lib/repositorios	Contratos abstractos y sus implementaciones para Firestore, OpenWeather y almacenamiento local.
Modelos	lib/modelos	Objetos de datos con fromMap y toMap, compartidos por las tres capas.

3.1 Patrones de diseño

Patrón	Dónde se aplica
MVVM	Cada pantalla tiene su modelo de vista (archivo *_vm.dart) con ChangeNotifier.
Repositorio	Contratos en lib/repositorios para cambiar de proveedor sin tocar la presentación (RNF-20).
Caché primero	Se lee el dato local, se revisa su vigencia y solo si caducó se consulta la red; ante una falla se muestra el último dato con aviso.
Fachada	openweather_cliente.dart en la aplicación y src/openweather.js en el ciclo encapsulan las peticiones al proveedor.
Estrategia	Una regla por tipo de riesgo en functions/src/motor/reglas.
Observador	Flujos de Firestore para parcelas, alertas y perfil en vivo.
Objeto de transferencia	Modelos con fromMap y toMap.

3.2 Módulos

El sistema se divide en cinco módulos funcionales y un grupo de componentes transversales, como muestra la Figura 3.

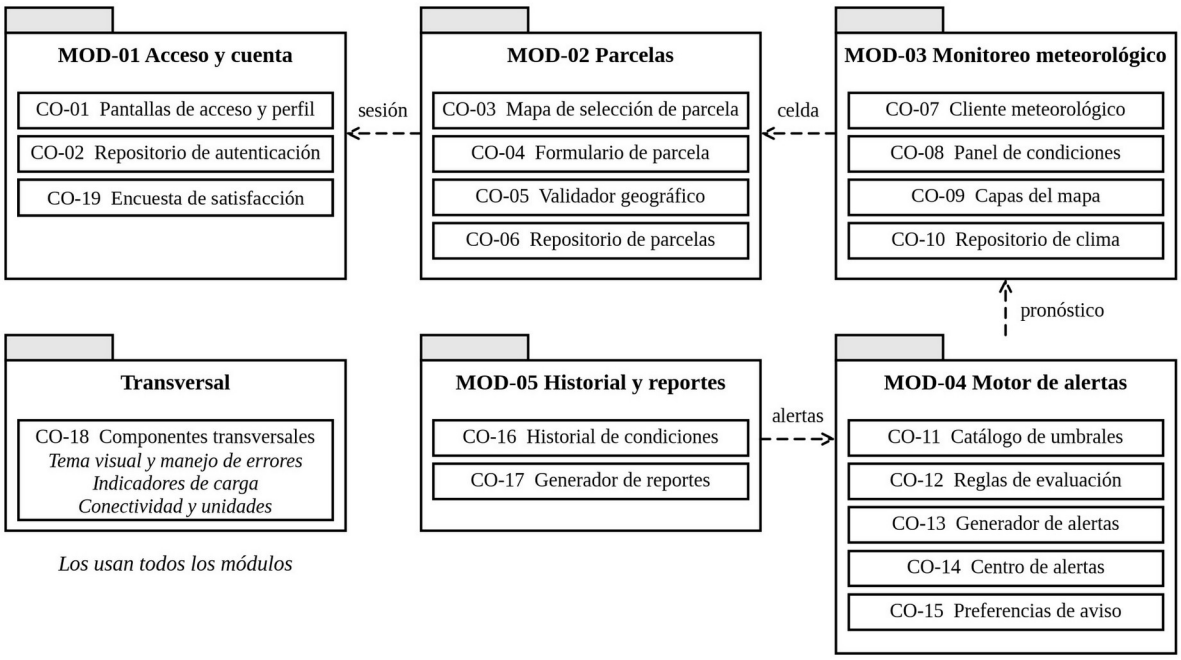


Figura 3. *Módulos y componentes del sistema.*

Módulo	Contenido	Carpeta
MOD-01 Acceso y cuenta	Bienvenida, inicio de sesión, registro, recuperación de contraseña, permisos y la encuesta Su opinión (CO-19).	lib/pantallas/acceso; la encuesta, en lib/pantallas/perfil
MOD-02 Parcelas	Registro en cuatro pasos, validación geográfica, lista y detalle.	lib/pantallas/parcelas
MOD-03 Monitoreo meteorológico	Panel principal, pronóstico detallado y mapa del clima.	lib/pantallas/clima
MOD-04 Motor de alertas	Centro de alertas, detalle, preferencias y el motor del ciclo.	lib/pantallas/alertas y functions/src
MOD-05 Historial y reportes	Registro día por día, resumen y exportación.	lib/pantallas/reportes
Transversal	Perfil, ajustes, ayuda, acerca de y componentes reutilizables.	lib/pantallas/perfil y lib/componentes

4. Estructura del repositorio

El repositorio contiene la aplicación Flutter, el código del ciclo automático y la documentación técnica. Los archivos con claves o credenciales no se suben; en su lugar hay plantillas con el sufijo example.

```
agroclima_jalapa/
├── .github/workflows/       Flujos de GitHub Actions (ciclo, umbrales, alertas de prueba)
├── android/                 Proyecto Android (minSdk 26, firma y canales de aviso)
├── assets/geo/              Límites de los 7 municipios (geoBoundaries) para validar parcelas
├── assets/img/              Logotipo e ilustraciones
├── docs/                    Requisitos, modelo de datos, umbrales, decisiones y diseño
├── env/dev.example.json     Plantilla de la clave de OpenWeather para la aplicación
├── functions/               Ciclo automático en Node.js
│   ├── index.js             Punto de entrada del ciclo
│   ├── src/                 Adquisición, motor, alertas, avisos, limpieza y cuentas
│   ├── seed/umbrales.json   Catálogo inicial de umbrales
│   ├── scripts/             Sembrar umbrales, alertas de prueba, simulación y exportar encuestas
│   └── test*/               Pruebas unitarias, de reglas y con emulador
├── lib/
│   ├── main.dart            Inicializa Firebase, persistencia, avisos y dependencias
│   ├── app.dart             Aplicación, rutas y tema claro u oscuro
│   ├── config/              Entorno, rutas, textos, constantes y tema
│   ├── modelos/             Objetos de datos y enumeraciones
│   ├── repositorios/        Acceso a Firestore, OpenWeather y almacenamiento local
│   ├── servicios/           Lógica de negocio
│   ├── pantallas/           Una carpeta por módulo, cada pantalla con su modelo de vista
│   ├── componentes/         Componentes visuales reutilizables
│   └── utilidades/          Unidades, fechas, validadores y errores
├── test/                    Pruebas de la aplicación
├── firebase.json            Emuladores, reglas e índices
├── firestore.rules          Reglas de seguridad de Firestore
├── firestore.indexes.json   Índices compuestos
└── pubspec.yaml             Dependencias de Flutter (versión 1.1.0+2)
```

4.1 Archivos de configuración principales

Archivo	Contenido
lib/config/constantes.dart	Versión mostrada, vigencia del dato (60 min, 180 min con ahorro de datos, 3 h el pronóstico), días de pronóstico (5), tiempo máximo de consulta (5 s), tamaño de celda (0.05°) y rangos válidos.
lib/config/entorno.dart	Lee la clave de OpenWeather y el uso de emuladores desde --dart-define.
lib/config/textos.dart	Todos los textos visibles de la aplicación.
lib/config/rutas.dart	Rutas de go_router y guardas de sesión.
functions/src/config.js	Zona horaria, tamaño de celda, reintentos, pausa entre celdas, días de historial para sequía y valores permitidos.
functions/seed/umbrales.json	Catálogo de umbrales que se carga en Firestore.

5. Modelo de datos

Los datos se guardan en Cloud Firestore. Como Firestore no impone un esquema, el esquema oficial está en docs/MODELO_DATOS.md y se respeta igual en la aplicación y en el ciclo. La Figura 4 muestra las colecciones y sus relaciones.

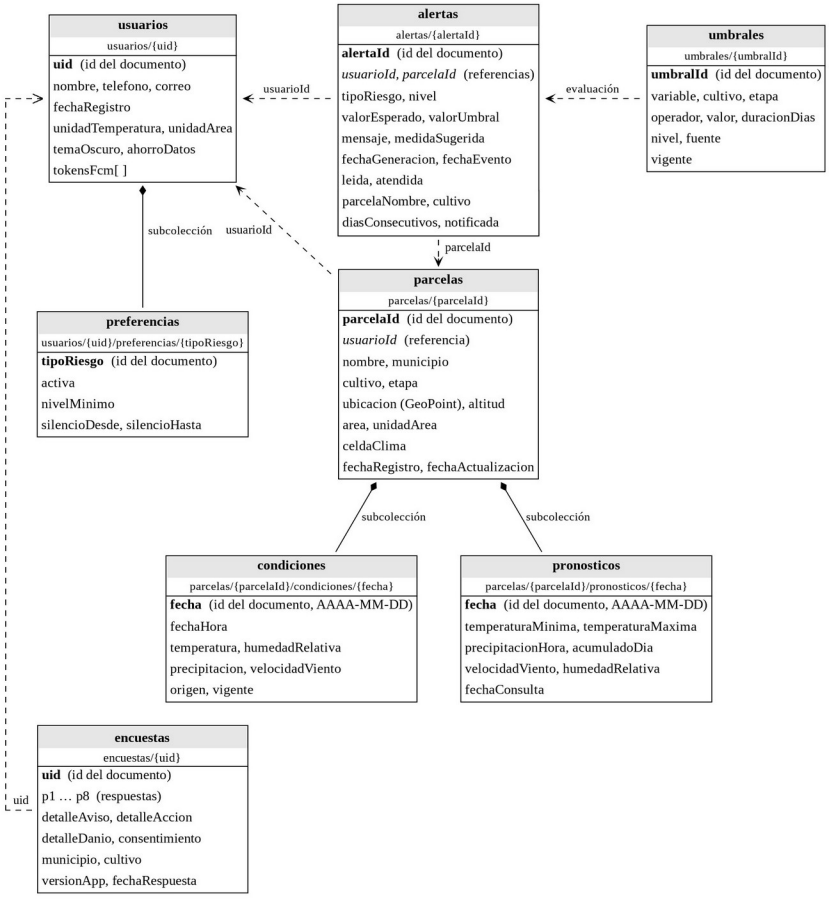


Figura 4. *Estructura de colecciones de Cloud Firestore.*

```
usuarios/{uid}
    └── preferencias/{tipoRiesgo}
parcelas/{parcelaId}
    ├── condiciones/{yyyyMMdd}
    └── pronosticos/{yyyyMMdd}
alertas/{alertaId}
umbrales/{umbralId}
encuestas/{uid}
```

Colección	Contenido	Quién escribe
usuarios	Perfil, unidades, tema, ahorro de datos, tokens de notificación y solicitud de eliminación.	Aplicación

Colección	Contenido	Quién escribe
usuarios/.../preferencias	Un documento por tipo de riesgo: activo, nivel mínimo y horario de silencio.	Aplicación
parcelas	Nombre, municipio, cultivo, etapa, ubicación, altitud, área y celda climática.	Aplicación
parcelas/.../condiciones	Clima observado por día: temperatura, humedad, lluvia acumulada y viento.	Ciclo y aplicación
parcelas/.../pronosticos	Pronóstico por día: mínima, máxima, lluvia por hora y del día, viento y humedad. Único insumo del motor.	Solo el ciclo
alertas	Alerta generada: riesgo, nivel, valor esperado, umbral, mensaje y medida sugerida.	Solo el ciclo
umbrales	Catálogo de criterios por variable, cultivo y etapa, con su fuente.	Solo el flujo de siembra
encuestas	Respuestas de Su opinión: preguntas p1 a p8, detalles opcionales, permiso, municipio y cultivo de la primera parcela, versión de la aplicación y fecha. Un documento por usuario, con su uid.	Aplicación

Las condiciones y los pronósticos son subcolecciones de cada parcela. En la consola de Firebase no aparecen en la raíz: se ven al abrir un documento de la colección parcelas. Sus identificadores son la fecha yyyyMMdd en hora de Guatemala, de modo que una segunda escritura del mismo día reemplaza a la anterior.

5.1 Valores de las enumeraciones

Campo	Valores guardados
municipio	jalapa, sanPedroPinula, sanLuisJilotepeque, sanManuelChaparron, sanCarlosAlzatate, monjas, mataquescuintla
cultivo	maiz, frijol, cafe, hortalizas (vacío: sin cultivo)
etapa	siembra, desarrolloVegetativo, floracion, llenado, cosecha
tipoRiesgo	lluviaIntensa, vientoFuerte, sequia, temperaturaBaja, temperaturaAlta, humedadAlta
nivel	informativa (verde), preventiva (precaución), critica (peligro)

5.2 Índices compuestos

Colección	Campos	Uso
alertas	usuarioId ascendente, fechaGeneracion descendente	Centro de alertas de la aplicación.
alertas	parcelaId, tipoRiesgo, fechaEvento ascendentes	Control de duplicados en el ciclo.

5.3 Borrado en cascada

Como el plan Spark no permite funciones que reaccionen al borrado, las limpiezas las hace el ciclo. Al borrar una parcela desde la aplicación, la siguiente ejecución elimina sus condiciones, sus pronósticos y sus alertas. Al pedir la eliminación de la cuenta, la aplicación marca el campo eliminacionSolicitada y cierra la sesión; el ciclo borra primero el usuario de Authentication y después sus parcelas, alertas, preferencias, encuesta y perfil. Si algo falla, la marca se conserva y se reintenta en la siguiente ejecución.

6. Ciclo automático

El ciclo automático es un programa de Node.js (functions/index.js) que usa el SDK de administración de Firebase. No es una Cloud Function: lo ejecuta GitHub Actions cada tres horas, por lo que funciona en el plan gratuito y no depende de que la aplicación esté abierta. La Figura 5 muestra su flujo.

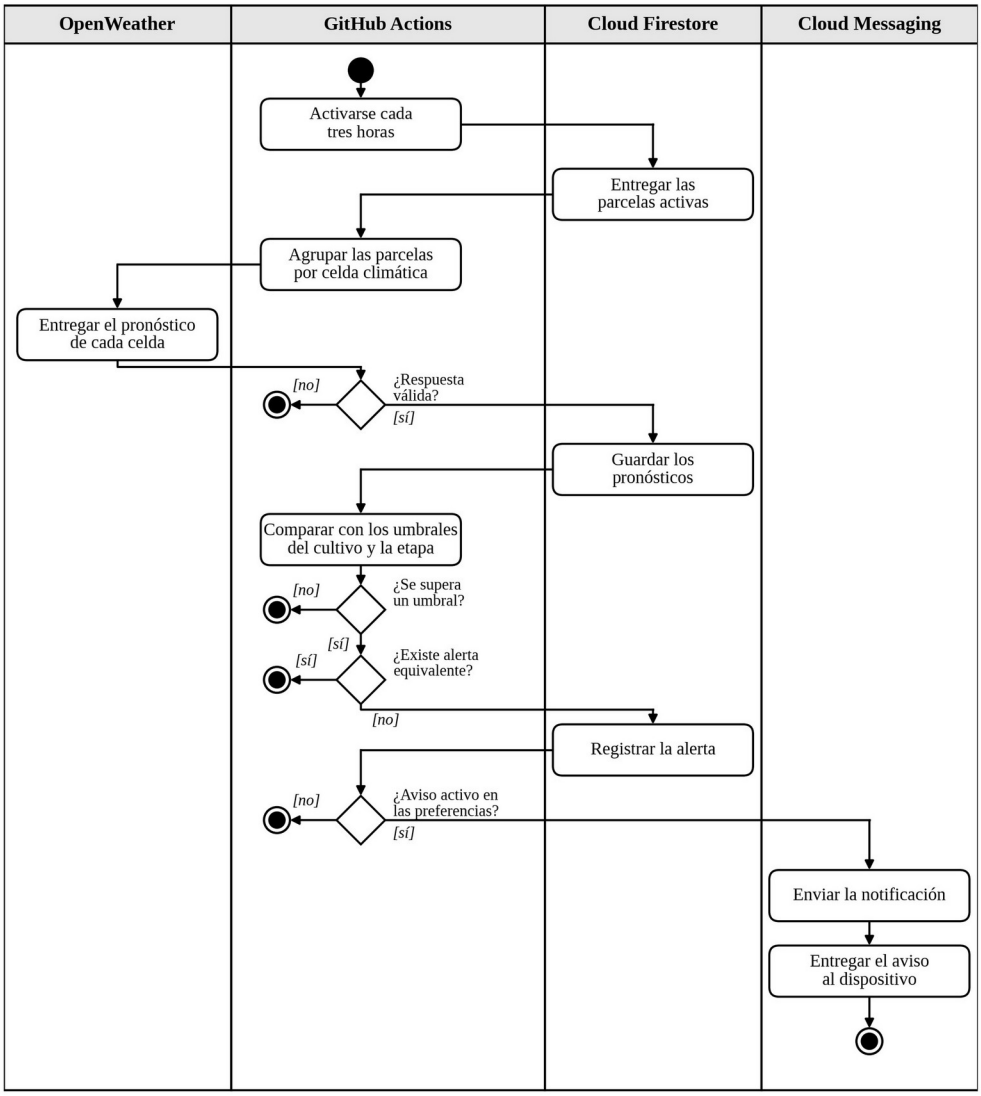


Figura 5. *Diagrama de actividades del ciclo automático.*

6.1 Pasos de cada ejecución

1. Elimina las cuentas que pidieron su eliminación (src/cuentas.js).
2. Limpia los datos de las parcelas borradas (src/limpieza.js).
3. Agrupa las parcelas activas por celda climática de 0.05° y consulta a OpenWeather una vez por celda; guarda las condiciones y los pronósticos de cada parcela (src/adquisicion.js).
4. Evalúa los umbrales de las celdas actualizadas con el catálogo vigente (src/evaluacion.js y src/motor).
5. Genera las alertas sin duplicar (src/alertas.js).
6. Envía las notificaciones por Cloud Messaging según las preferencias de cada usuario (src/notificaciones.js).

Si ninguna celda obtuvo clima, por ejemplo por una clave inválida o una caída del proveedor, el programa termina con error para que GitHub avise por correo al dueño del repositorio.

6.2 Flujos de GitHub Actions

Archivo	Nombre	Se ejecuta	Qué hace
ciclo-clima.yml	Ciclo del clima	Cada 3 horas (cron 0 */3 * * *, en UTC) y a mano	Ejecuta node index.js con los secretos del repositorio.
sembrar-umbrales.yml	Sembrar umbrales	Solo a mano	Carga o actualiza el catálogo de functions/seed/umbrales.json en Firestore.
alertas-de-prueba.yml	Alertas de prueba	Solo a mano	Envía o borra dos alertas de prueba (helada y lluvia fuerte) en la primera parcela de una cuenta.

El horario en UTC equivale a las 18:00, 21:00, 00:00, 03:00, 06:00, 09:00, 12:00 y 15:00 de Guatemala. GitHub puede retrasar unos minutos las tareas programadas en momentos de mucha demanda. La Figura 6 muestra el historial de ejecuciones del repositorio.

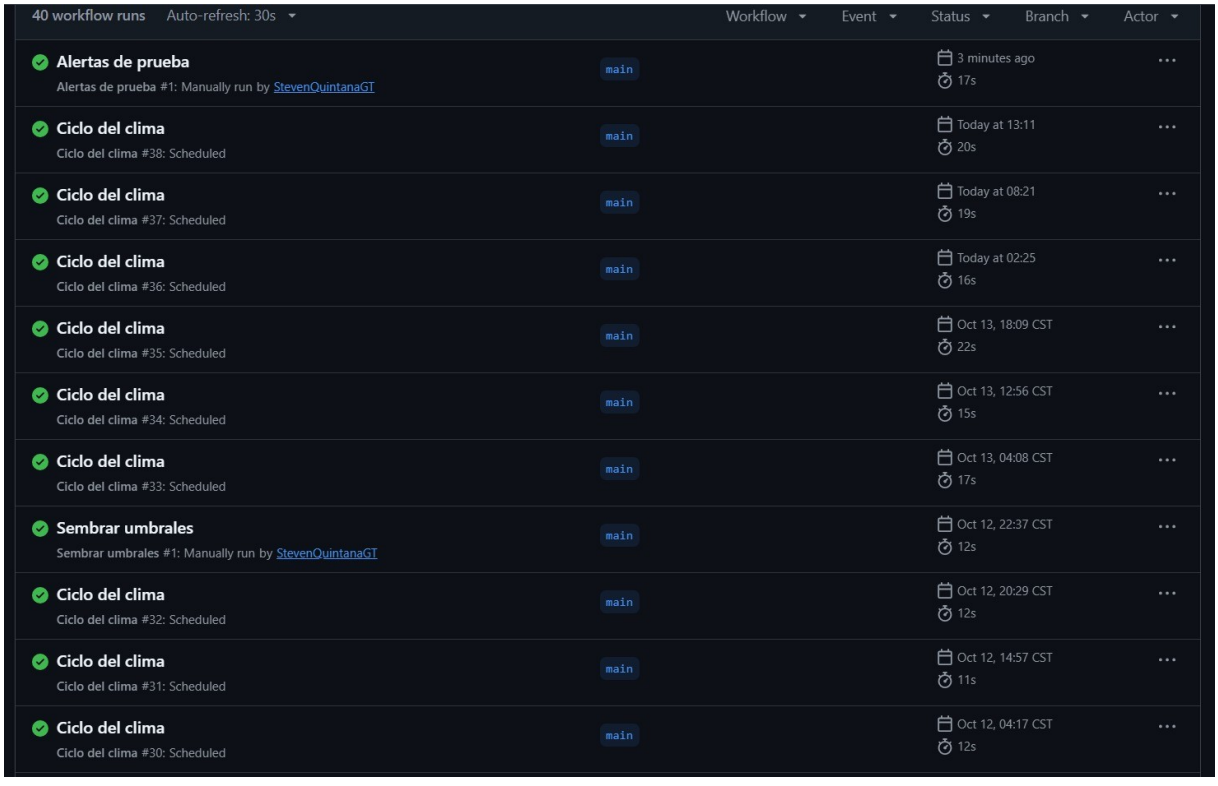


Figura 6. *Historial de ejecuciones en GitHub Actions.*

6.3 Parámetros del ciclo

Parámetro	Valor	Motivo
ZONA_HORARIA	America/ Guatemala	Identificadores diarios y horario de silencio.
TAMANO_CELDA_GRADOS	0.05	Cuadros de unos 5.5 km de lado; las parcelas del mismo cuadro comparten la consulta.
ESPERA_CLIMA_MS	5000	Tiempo máximo por consulta (RNF-07).
REINTENTOS_CLIMA	3	Reintentos ante fallas temporales.
PAUSA_ENTRE_CELDAS_MS	2400	Dos llamadas por celda cada 2.4 s: 50 por minuto, bajo el límite de 60 del plan gratuito.
DIAS_PRONOSTICO	5	Hoy y los cuatro días siguientes.
DIAS_HISTORIAL_SEQUIA	20	Días de condiciones que se unen al pronóstico para contar días secos.
UMBRAL_DIA_SECO_MM	1	Un día con menos de 1 mm cuenta como seco.

6.4 Motor de umbrales

Los umbrales no están en el código: se leen de la colección umbrales. Para cada parcela aplican los umbrales generales y los de su cultivo, de todas las etapas o de su etapa. Cada tipo de riesgo tiene su propia regla, que compara el pronóstico de los cinco días con los umbrales y devuelve el nivel más alto cumplido por día. La sequía une los días secos del historial con los del pronóstico para contar la racha.

Catálogo inicial de umbrales

Variable	Aplica a	Precaución	Peligro
Lluvia por hora	Todos	> 15 mm/h	> 30 mm/h
Viento	Todos	> 15 km/h	> 30 km/h
Días secos seguidos	Todos (informativa de 5 a 10)	11 a 15 días	> 15 días
Días secos seguidos	Maíz en floración; frijol en floración o llenado	5 a 10 días	> 10 días
Temperatura mínima	Todos	—	≤ 0 °C (helada)
Temperatura mínima	Café / frijol	< 15 °C / < 18 °C	—
Temperatura máxima	Café / frijol	> 30 °C / > 28 °C	Igual, 3 días o más
Temperatura máxima	Maíz	> 35 °C en desarrollo	> 35 °C en floración o llenado
Humedad relativa	Café	> 85 %	—

Para cambiar un umbral se edita functions/seed/umbrales.json, se sube el cambio al repositorio y se ejecuta el flujo **Sembrar umbrales**. La carga es idempotente: usa umbralId como identificador, así que repetirla no duplica documentos.

6.5 Control de duplicados y envío

- Antes de crear una alerta se busca otra con la misma parcela, el mismo riesgo y el mismo día del evento. Si existe con igual o mayor nivel, no se crea; si el riesgo empeoró, se crea una nueva con el nivel mayor.
- Se notifica solo si el aviso está activo en las preferencias, si el nivel es igual o mayor al mínimo elegido y si no es horario de silencio (por defecto de 22:00 a 05:00). Las alertas de peligro suenan siempre.
- Una alerta que no se notifica igual queda guardada con notificada en falso.
- Se envía a todos los tokens del usuario, y los tokens inválidos que devuelve Cloud Messaging se eliminan.
- Varios días seguidos del mismo riesgo y nivel en una parcela se avisan con una sola notificación.

7. Seguridad

La seguridad descansa en tres elementos: la autenticación de Firebase, las reglas de Firestore que limitan cada documento a su dueño y el manejo de las claves fuera del código. La comunicación con todos los servicios usa HTTPS, como resume la Figura 7.

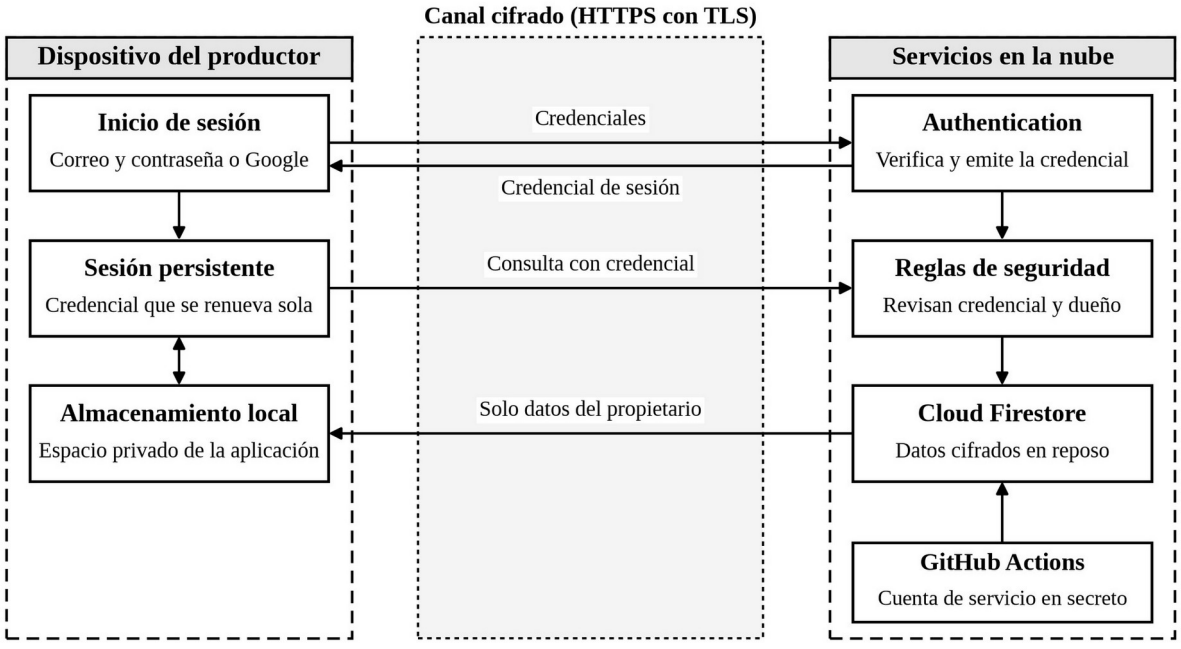


Figura 7. *Mecanismos de seguridad del sistema.*

7.1 Reglas de Firestore

Las reglas están en el archivo firestore.rules del repositorio y se publican en el proyecto de Firebase, donde se pueden revisar en Firestore › Reglas, como en la Figura 8. La tabla siguiente resume lo que permite cada colección.

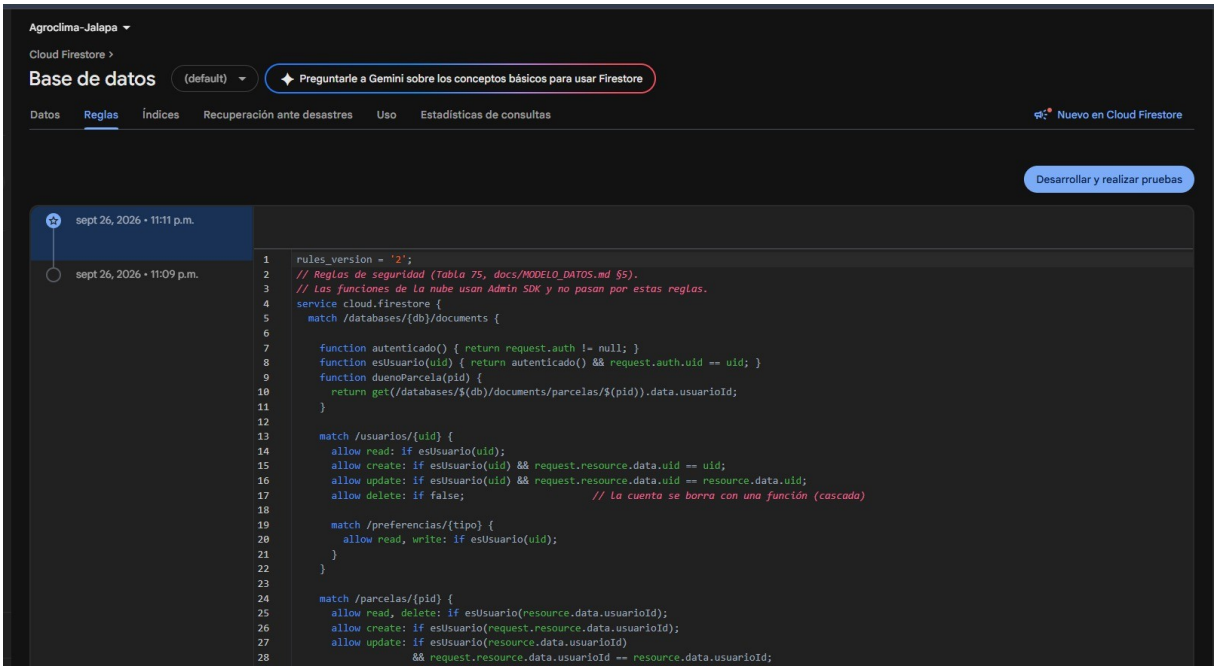


Figura 8. *Reglas de seguridad publicadas en la consola de Cloud Firestore.*

Colección	Lectura	Escritura desde la aplicación
usuarios y preferencias	Solo el dueño	Crear y actualizar su propio perfil. No puede borrarlo.
parcelas	Solo el dueño	Crear, actualizar y borrar las suyas sin cambiar el dueño.
condiciones	Solo el dueño de la parcela	Solo registros de origen consulta.
pronosticos	Solo el dueño de la parcela	Ninguna: solo el ciclo.
alertas	Solo el destinatario	Solo los campos leida y atendida.
umbrales	Cualquier usuario con sesión	Ninguna.
encuestas	Solo el dueño	Crear y reemplazar la suya, con su propio uid como identificador y en el campo uid. No puede borrarla.

El ciclo usa el SDK de administración con una cuenta de servicio, por lo que no pasa por estas reglas. Las consultas de la aplicación siempre deben filtrar por usuarioId o las reglas las rechazan.

7.2 Claves y archivos que no se suben al repositorio

Elemento	Dónde va	Uso
env/dev.json y env/prod.json	Equipo del desarrollador	Clave de OpenWeather de la aplicación (OPENWEATHER_API_KEY).
android/app/google-services.json y lib/firebase_options.dart	Equipo del desarrollador	Conexión de la aplicación con el proyecto de Firebase.
android/key.properties y *.jks	Equipo del desarrollador y respaldo seguro	Llave de firma del APK.
android/local.properties	Equipo del desarrollador	Rutas del SDK de Android y de Flutter.
OPENWEATHER_KEY	Secreto de GitHub	Clave de OpenWeather del ciclo.
FIREBASE_SERVICE_ACCOUNT	Secreto de GitHub	JSON de la cuenta de servicio de Firebase para el ciclo.

Guarde la llave de firma (archivo .jks y sus contraseñas) en un lugar seguro fuera del repositorio. Sin ella no se pueden publicar actualizaciones que se instalen encima de la versión actual: los usuarios tendrían que desinstalar y volver a instalar.

8. Configuración del entorno de desarrollo

8.1 Requisitos

Herramienta	Detalle
Flutter	Canal estable con Dart 3.13 o superior.
Android SDK	Con el NDK instalado (Android Studio › SDK Manager › SDK Tools › NDK).
Node.js	Versión 22 o superior, para el ciclo y sus pruebas.
Firebase CLI	npm install -g firebase-tools y firebase login.
FlutterFire CLI	dart pub global activate flutterfire_cli, para conectar la aplicación con Firebase.
Java	Para los emuladores de Firebase; sirve el que incluye Android Studio.
Git	Para clonar el repositorio.
Cuentas	Proyecto de Firebase en plan Spark, clave de OpenWeather del plan gratuito y cuenta de GitHub.

8.2 Pasos de configuración

1. Clone el repositorio y entre a la carpeta del proyecto.
2. Copie android/local.properties.example como android/local.properties y llene sdk.dir y flutter.sdk.
3. Copie env/dev.example.json como env/dev.json y escriba la clave de OpenWeather.
4. Ejecute flutterfire configure para generar google-services.json y firebase_options.dart del proyecto de Firebase.
5. Instale las dependencias con flutter pub get y, en la carpeta functions, con npm install.
6. Ejecute la aplicación en un emulador o en un teléfono con los comandos siguientes.

```
git clone https://github.com/StevenQuintanaGT/Agroclima_Jalapa.git
cd Agroclima_Jalapa
flutter pub get
flutter run --dart-define-from-file=env/dev.json
flutter analyze && flutter test
```

8.3 Trabajo con los emuladores de Firebase

Para probar sin tocar los datos reales, se encienden los emuladores de Authentication y Firestore y se ejecuta la aplicación apuntando a ellos. Las cuentas de prueba están en functions/seed/cuentas_prueba_emulador.json y los datos se pueden revisar en http://localhost:4000.

```
cd functions
npm run emuladores

# en otra terminal, desde la raíz del proyecto
flutter run --dart-define-from-file=env/dev.json --dart-define=USAR_EMULADORES=true

# alertas de prueba en el emulador (helada, calor, lluvia, viento...)
npm run simular:emulador -- <parcelaId> helada
```

9. Compilación y publicación del APK

La aplicación se distribuye como APK firmado desde la página del proyecto, sin Google Play. Para una versión nueva, primero se actualiza el número de versión en pubspec.yaml y en lib/config/constantes.dart (una prueba verifica que coincidan).

9.1 Crear la llave de firma (una sola vez)

```
keytool -genkey -v -keystore android/agroclima-entrega.jks -keyalg RSA \
        -keysize 2048 -validity 10000 -alias agroclima
```

Después se copia android/key.properties.example como android/key.properties con las contraseñas de la llave, y env/dev.example.json como env/prod.json con la clave de OpenWeather de producción.

9.2 Compilar

```
flutter build apk --release \
  --target-platform android-arm,android-arm64 \
  --obfuscate --split-debug-info=build/simbolos \
  --dart-define-from-file=env/prod.json
```

El archivo resultante queda en build/app/outputs/flutter-apk/app-release.apk y pesa cerca de 39 MB. Se compila solo para procesadores ARM, que son los de todos los teléfonos Android reales; la arquitectura x86_64 es de emuladores y aumentaba el tamaño en 23 MB. Sin key.properties el APK se firma con la llave de prueba, que sirve para probar pero no para entregar.

9.3 Publicar

1. Renombre el archivo como AgroClima-Jalapa.apk.
2. Cree un Release nuevo en el repositorio de la aplicación y adjunte el archivo con ese mismo nombre. Los botones de la página descargan siempre el último Release (releases/latest/download/AgroClima-Jalapa.apk), así que no hay que cambiar el enlace.
3. Actualice la versión y el tamaño que muestra la página en sitio/index.html del repositorio Pag_Agroclima.
4. Guarde la carpeta build/simbolos junto con la versión: sirve para leer los errores de la versión ofuscada.
5. Registre los cambios en el historial de versiones de los manuales.

10. Despliegue y operación en la nube

10.1 Proyecto de Firebase

El proyecto de Firebase usa Authentication con los proveedores de correo y Google, Cloud Firestore y Cloud Messaging, todo en el plan Spark. La Figura 9 muestra la consola del proyecto con el uso de lecturas y escrituras.



Figura 9. *Consola del proyecto de Firebase en el plan Spark.*

Las reglas y los índices se publican desde la carpeta del proyecto con el comando siguiente, que funciona en el plan gratuito:

```
firebase deploy --only firestore:rules,firestore:indexes
```

10.2 Secretos de GitHub

1. En el repositorio, abra **Settings** › **Secrets and variables** › **Actions**.
2. Cree el secreto **OPENWEATHER_KEY** con la clave de OpenWeather.
3. En la consola de Firebase, genere una clave privada de la cuenta de servicio (Configuración del proyecto › Cuentas de servicio) y pegue el JSON completo en el secreto **FIREBASE_SERVICE_ACCOUNT**.
4. Ejecute a mano el flujo **Ciclo del clima** (Actions › Ciclo del clima › Run workflow) y revise que termine en verde.

La Figura 10 muestra los dos secretos ya creados en el repositorio. GitHub solo muestra su nombre y la fecha de actualización; el valor no se puede volver a ver, solo reemplazar con el ícono del lápiz.

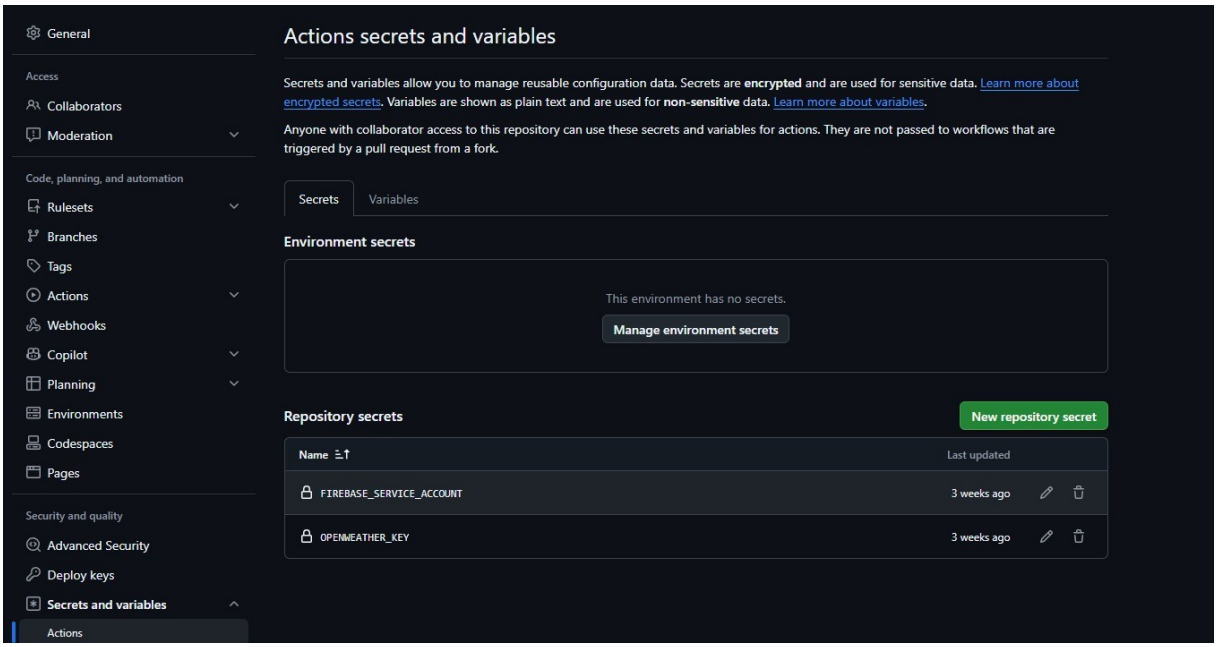


Figura 10. *Secretos del repositorio en GitHub.*

Si falta alguno de los secretos, el flujo no falla: muestra una advertencia y no ejecuta el ciclo.

10.3 Carga inicial de umbrales

Antes de la primera ejecución del ciclo se debe cargar el catálogo: Actions › **Sembrar umbrales** › Run workflow. Se repite cada vez que se cambie functions/seed/umbrales.json.

10.4 Prueba de notificaciones

Para revisar los avisos en un teléfono sin esperar un evento real se usa Actions › **Alertas de prueba** › Run workflow, con el correo de la cuenta y la acción enviar. Llegan dos alertas marcadas como prueba en la primera parcela de esa cuenta. Con la acción borrar se eliminan. El correo se oculta en el registro de la ejecución.

10.5 Exportar las respuestas de la encuesta

Las respuestas de Su opinión se descargan a un archivo CSV con el script functions/scripts/exportar-encuestas.js, que usa la misma cuenta de servicio del ciclo. Se ejecuta desde la raíz del repositorio con el comando siguiente; si no se indica un nombre, el archivo se llama encuestas.csv.

```
FIREBASE_SERVICE_ACCOUNT='<JSON de la cuenta de servicio>' node functions/scripts/exportar-encuestas.js [archivo.csv]
```

- El archivo se guarda en UTF-8 con BOM, para que Excel muestre bien las tildes.
- No incluye el correo ni el uid. Agrega el nombre del perfil, o «(sin permiso)» si el productor no dio el permiso.
- El CSV tiene datos personales: .gitignore lo excluye y no debe subirse al repositorio.

11. Pruebas

Comando	Dónde	Qué prueba
flutter test	Raíz	Modelos, servicios, modelos de vista y pantallas de la aplicación.
flutter analyze	Raíz	Análisis estático con flutter_lints.
npm test	functions	Pruebas unitarias del cliente de OpenWeather, motor, avisos, umbrales y lluvia del día.
npm run test:reglas	functions	Reglas de Firestore con el emulador.
npm run test:emulador	functions	Reglas, adquisición, alertas, cuentas, evaluación y limpieza con el emulador.

Las pruebas de reglas verifican, entre otros casos, que no se pueda leer una parcela ajena, crear una parcela a nombre de otro usuario, escribir un pronóstico desde la aplicación ni cambiar el mensaje de una alerta, y que sí se pueda marcar como leída una alerta propia.

12. Mantenimiento

Tarea	Frecuencia	Cómo
Revisar las ejecuciones del ciclo	Semanal	Actions › Ciclo del clima. Una ejecución en rojo genera un correo de GitHub.
Mantener activo el repositorio	Antes de 60 días	GitHub desactiva las tareas programadas de un repositorio público sin actividad durante 60 días. Un cambio o una ejecución manual lo evita.
Revisar cuotas	Mensual	Uso de Firestore en la consola de Firebase y de llamadas en OpenWeather (60 por minuto y 1 000 000 al mes).
Revisar umbrales	Por temporada	Ajustar functions/seed/umbrales.json con criterio técnico y ejecutar Sembrar umbrales.
Renovar claves	Si se exponen	Generar una clave nueva en OpenWeather o en Firebase y actualizar el secreto de GitHub y env/prod.json.
Actualizar dependencias	Semestral	flutter pub outdated y npm outdated; probar antes de publicar.

Tarea	Frecuencia	Cómo
Respaldar la llave de firma	Al crearla y al cambiarla	Copia del archivo .jks y de key.properties en un lugar seguro.

13. Solución de problemas

Problema	Causa probable	Solución
El ciclo termina en rojo con «Ninguna celda obtuvo clima».	Clave de OpenWeather inválida o servicio caído.	Revisar el secreto OPENWEATHER_KEY y el estado de OpenWeather; ejecutar de nuevo.
El ciclo muestra que faltan secretos.	No existen OPENWEATHER_KEY o FIREBASE_SERVICE_ACCOUNT.	Crear los secretos en Settings › Secrets and variables › Actions.
El ciclo dejó de ejecutarse solo.	GitHub desactivó la tarea por 60 días sin actividad.	Actions › Ciclo del clima › Enable workflow.
La aplicación no muestra el clima.	Falta la clave en env/*.json al compilar.	Compilar con --dart-define-from-file y una clave válida.
La aplicación no se conecta a Firebase.	Falta google-services.json.	Ejecutar flutterfire configure y compilar de nuevo.
No llegan notificaciones.	Permiso desactivado, token vencido o preferencia apagada.	Revisar el permiso del teléfono, el campo tokensFcm y la colección de preferencias; usar Alertas de prueba.
La actualización no se instala encima.	El APK se firmó con otra llave.	Compilar con la llave original de key.properties.
Una consulta de la aplicación es rechazada.	No filtra por usuarioId.	Agregar el filtro por usuarioId, como exigen las reglas.

14. Glosario técnico

Término	Significado
APK	Paquete de instalación de una aplicación Android.
Celda climática	Cuadro de 0.05° de latitud y longitud que agrupa parcelas cercanas para consultar el clima una sola vez.
Ciclo automático	Programa que cada tres horas consulta el clima, evalúa umbrales y envía avisos.

Término	Significado
Cuenta de servicio	Credencial de Firebase que permite al ciclo trabajar con privilegios de administración.
Emulador de Firebase	Versión local de los servicios de Firebase para probar sin tocar los datos reales.
FCM	Firebase Cloud Messaging, servicio de envío de notificaciones.
Flujo de trabajo	Archivo .yml de GitHub Actions que define una tarea automática.
MVVM	Patrón que separa la pantalla (vista) de su estado y lógica (modelo de vista).
Plan Spark	Plan gratuito de Firebase.
Secreto	Valor cifrado guardado en GitHub que los flujos usan sin mostrarlo.
Token FCM	Identificador de un teléfono para enviarle notificaciones.
Umbral	Valor del clima a partir del cual se genera una alerta para un cultivo y una etapa.

15. Soporte y referencias

Recurso	Dirección
Repositorio	https://github.com/StevenQuintanaGT/Agroclima_Jalapa
Página del proyecto	https://stevenquintanagt.github.io/Pag_Agroclima/
Documentación del repositorio	README.md, ESTRUCTURA_PROYECTO.md y la carpeta docs
Flutter	https://docs.flutter.dev
Firebase	https://firebase.google.com/docs
OpenWeather	https://openweathermap.org/api
GitHub Actions	https://docs.github.com/actions
Desarrollador	Marlon Steven Quintana Recinos
Correo de contacto	msquintana.gt@gmail.com