

## MEMORIA ESCRITA DEL PROYECTO

CFGS Desarrollo de Aplicaciones Multiplataforma

EggStat



**Autor:** Celia Pérez Vargas

**Tutor:** Alfonso Marín Marín

**Fecha de entrega:** 24/04/2025

**Convocatoria:** 2s2425

**Documentos del proyecto:**

[https://drive.google.com/drive/folders/1G5glktjQgRavjAMYs8N6fXR63SdQoncO?usp=drive\\_link](https://drive.google.com/drive/folders/1G5glktjQgRavjAMYs8N6fXR63SdQoncO?usp=drive_link)



# Índice

1. Introducción	2
1.1. Motivación	2
1.2. Abstract	3
1.3. Objetivos propuestos	3
2. Estado del Arte	5
2.1. Análisis de las aplicaciones más relevantes	7
3. Metodología usada	9
3.1. Metodología Kanban	9
3.2. Ciclo de vida iterativo-incremental	9
4. Tecnologías y herramientas utilizadas en el proyecto	12
5. Planificación, Diagnóstico y Contexto Laboral	14
5.1. Planificación	14
5.2. Diagnóstico y contexto laboral	16
6. Análisis del proyecto	18
6.1. Requisitos funcionales y no funcionales	18
6.2. Diagrama entidad-relación	21
6.3. Casos de uso	22
6.4. Diagrama de clases	31
7. Diseño del proyecto	32
7.1. Diseño previo	32
7.2. Base de datos: Firebase Realtime Database y Firebase Auth	33
7.3. Diseño y desarrollo de la aplicación base	35
7.4. Implementación de funcionalidades	37
7.5. Mejoras de la interfaz gráfica	42
7.6. Mejoras generales y puesta a punto	43
8. Despliegue y pruebas	46
9. Conclusiones	51
10. Vías futuras	52
11. Bibliografía/Webgrafía	53
12. Anexos	56
12.1. Manual de Instalación	56
12.2. Manual de Usuario	58
12.3. Glosario	65
12.4. Imágenes	67

# 1. Introducción

En este documento se presenta el desarrollo completo de la aplicación móvil **EggStat**, desde su idea inicial hasta su implementación final. En él se detallan las distintas fases del proyecto: el análisis de necesidades, el diseño de la interfaz y de la estructura técnica, la codificación de sus funcionalidades, las pruebas realizadas, así como una reflexión sobre su contexto educativo y tecnológico.

La memoria recoge, por tanto, todo el proceso de desarrollo de una aplicación móvil interactiva y gamificada, diseñada para fomentar en los centros educativos el cuidado de un gallinero escolar y, con ello, promover valores como la responsabilidad, el respeto por los animales y la concienciación sobre el origen de los alimentos. Esta app, pensada para el sistema operativo Android, permite a los alumnos y profesores registrar tareas como la recogida de huevos o la limpieza de las estancias del gallinero, todo ello acompañado por un sistema de puntos, estadísticas visuales y una interfaz adaptada a un público infantil.

*EggStat* ha sido diseñada pensando en ofrecer una experiencia intuitiva, colorida y cercana. La aplicación prioriza la sencillez de uso a través de elementos visuales llamativos, botones claros, mensajes animados y navegación por iconos. El desarrollo de la app ha seguido una metodología organizada, con una arquitectura modular y escalable, y utilizando tecnologías actuales como *Firebase* y *Android Studio*, basado en el lenguaje moderno *Kotlin*.

Si bien esta memoria documenta todas las funcionalidades implementadas en esta primera versión, se deja abierta la posibilidad de futuras ampliaciones y mejoras. El objetivo es que EggStat siga creciendo como una herramienta educativa útil, divertida y con un fuerte componente de conciencia ecológica.

## 1.1. Motivación

La motivación para desarrollar este proyecto nace de una **pasión muy personal** y, a la vez, muy cercana: **las gallinas**. A lo largo de mi vida siempre he estado rodeada de animales, pero las gallinas han sido, sin duda, una de mis debilidades. Lejos de los prejuicios que muchas veces las rodean, las gallinas son animales divertidos, inteligentes, agradecidos y con una personalidad única. Actualmente tengo seis gallinas, y no solo forman parte de mi día a día, sino que también han sido la fuente directa de inspiración y protagonistas absolutas de esta aplicación.

Además de mi vínculo personal con estos animales, soy profesora de iniciación a la programación, y en mis clases suelo utilizar ejemplos con gallinas para explicar conceptos básicos o para que sean las protagonistas de los videojuegos que creamos, porque resultan cercanos, curiosos y siempre arrancan una sonrisa al alumnado. De ahí surgió la idea de ir un paso más allá y crear una herramienta real que permita integrar a las gallinas en el aula de forma práctica y tecnológica.

Con *EggStat*, quería acercar este mundo al entorno escolar: ofrecer a los niños y niñas una forma divertida de aprender sobre el cuidado animal, la producción responsable de alimentos, y a la vez fomentar valores como la constancia, la empatía y el **trabajo en equipo**. Creo firmemente que la tecnología también puede tener un enfoque ético y educativo, y este proyecto es mi pequeño granito de arena en esa dirección.

## 1.2. Abstract

*EggStat* is an educational mobile application designed for Android that aims to promote awareness about animal care and egg production through the daily maintenance of a school chicken coop. The app allows students to register tasks such as collecting eggs and cleaning the coop, earning points for each action and visualizing their progress through interactive statistics. Teachers have access to special management features, such as adding students, teams, and hens, or redeeming points.

The development process followed an incremental and iterative life cycle using the *Kanban* methodology and the MVVM architecture, integrating technologies such as Firebase Realtime Database, Firebase Auth or Firebase Storage, and was developed using *Kotlin*, one of the most modern and widely adopted programming languages for Android development. The user interface is friendly, colorful, and adapted to young users, including animated images and intuitive dialogs.

The idea was born from the developer's personal connection with hens and her experience as a programming teacher, aiming to merge education, technology, and nature. The app is intended to be scalable to other contexts (such as school gardens or animal shelters), offering a unique approach to digital learning based on real-life responsibility and sustainability.

## 1.3. Objetivos propuestos

Para la realización del proyecto se ha planteado un objetivo general y varios específicos:

**Objetivo general:**

- Promover la educación sobre la producción de huevos y el bienestar animal mediante una aplicación móvil que facilite el registro y monitoreo del cuidado de un gallinero, incentivando la responsabilidad mediante un sistema de recompensas y estadísticas.

**Objetivos específicos:**

- Desarrollar una aplicación móvil en Android que permita a los niños registrar los huevos que ponen sus gallinas, además de registrar cuándo realizan limpieza de las estancias del gallinero.
- Implementar Firebase Firestore para almacenar la información de los huevos, estancias, gallinas y usuarios en la nube, asegurando la actualización en tiempo real para todos los dispositivos.
- Integrar un sistema de autentificación combinando Firebase Auth y Firestore con roles definidos, en el que estará el rol del profesor y el rol de los alumnos, los cuales tendrán acciones limitadas, mientras que el profesor podrá modificar más datos de las tablas.
- Motivar a los niños a cuidar el gallinero con un sistema de puntos que obtendrán al realizar el mantenimiento de alguna de las estancias.
- Implementar una pantalla de estadísticas con gráficos dinámicos que permitan analizar la producción de huevos de cada gallina y la actividad de los alumnos en el cuidado del gallinero.
- Configurar un sistema visual que muestre a los niños el estado de las estancias, para saber cuándo necesitan mantenimiento, incentivando el cuidado regular del gallinero.

## 2. Estado del Arte

Para entender mejor el contexto en el que se sitúa esta aplicación y comprobar si realmente aporta algo nuevo al panorama educativo y tecnológico, es fundamental realizar una revisión del **estado del arte**. Este proceso implica investigar qué herramientas similares existen actualmente, qué soluciones se ofrecen ya en el ámbito de la educación digital o el cuidado animal, y en qué medida esta app puede cubrir vacíos, aportar innovación o adaptarse a tendencias recientes.

En primer lugar, es importante destacar que el enfoque temático de esta aplicación es poco común. Actualmente, no se encuentran en el mercado aplicaciones móviles pensadas específicamente para que niños y niñas registren el cuidado de gallinas o la producción de huevos en un entorno escolar. La mayoría de apps educativas centradas en animales se enfocan en contenido más teórico (vídeos, fichas o juegos interactivos) o en simulaciones generales tipo “granja virtual”, pero sin conexión con la vida real ni con una experiencia práctica de campo.

En el ámbito de la educación basada en gamificación, sí existen bastantes ejemplos y herramientas consolidadas. Plataformas como **ClassDojo** o **Kahoot!** son líderes en este terreno, permitiendo a los docentes incentivar la participación del alumnado a través de puntos, insignias o rankings. Sin embargo, estas aplicaciones están diseñadas para actividades genéricas del aula, y no permiten conectar esa motivación con el cuidado de un espacio físico como un gallinero.

Por otro lado, desde el punto de vista tecnológico, se ha optado por integrar *Firebase* como motor central de la aplicación. Esta elección va muy en línea con las tendencias actuales en desarrollo móvil: se trata de una herramienta ampliamente usada por su facilidad de integración, sus servicios de autenticación, almacenamiento en tiempo real y su escalabilidad. Aunque muchos proyectos educativos utilizan *Firebase* como backend, no se encuentran ejemplos específicos que combinen *Firebase* con educación, bienestar animal y gamificación, lo cual aporta a este proyecto un enfoque original y poco explorado.

Además, el proyecto no se limita solo a permitir interacciones básicas, sino que introduce funcionalidades más avanzadas como la visualización de estadísticas mediante gráficos (con *MPAndroidChart*), el uso de imágenes animadas (mediante *Glide* y *Firebase Storage*), y la gestión de roles diferenciados (alumnos y profesor). Esta combinación de herramientas modernas con un enfoque práctico y educativo en un contexto real (una escuela con gallinero) lo convierte en un proyecto con una propuesta única.

Uno de los aspectos clave en el que me gustaría hacer hincapié es en la **necesidad de diferenciar claramente entre los roles de profesor y alumno** en el sistema de inicio de sesión. Esta decisión no solo responde a criterios técnicos o de funcionalidad, sino también a mi experiencia en el ámbito educativo, donde es habitual trabajar con herramientas como *Scratch* o *CoSpaces*, que implementan estructuras de cuentas específicas para docentes y estudiantes. Además, existen consideraciones legales importantes, ya que muchas plataformas impiden que menores de 16 años utilicen direcciones de correo electrónico personales, lo que obliga a crear un sistema de acceso adaptado a su edad. En general, la mayoría de plataformas educativas modernas incorporan este tipo de segmentación de usuarios para controlar permisos, asegurar la privacidad y gestionar la experiencia de aprendizaje de forma más efectiva. Esta experiencia previa y la observación de esas buenas prácticas fueron determinantes a la hora de implementar en *EggStat* un sistema que permite que los profesores tengan acceso a funcionalidades avanzadas, como la creación de usuarios, gallinas o equipos, mientras que los alumnos tienen una experiencia más simplificada y centrada en el cuidado y seguimiento del gallinero.

A nivel social y pedagógico, la app tiene potencial para alinearse con los **Objetivos de Desarrollo Sostenible de la Agenda 2030**, en especial con los objetivos 4 (educación de calidad), 12 (producción y consumo responsables) y 15 (vida de ecosistemas terrestres). Este tipo de iniciativas, que combinan tecnología, sostenibilidad y educación, son cada vez más valoradas tanto en el ámbito académico como en propuestas institucionales.



Tabla de objetivos de desarrollo sostenible. Fuente: [un.org](http://un.org)

Por supuesto, tras analizar la situación de este tipo de aplicaciones en el mercado, soy consciente de las limitaciones del proyecto. Su nicho tan específico (centros con gallinero real y con interés por digitalizar su seguimiento) limita su implantación a gran escala. Sin embargo, esto no reduce su valor: al contrario, lo posiciona como una herramienta altamente especializada y adaptable, que puede evolucionar fácilmente a otros escenarios como huertos escolares, refugios de animales o granjas educativas.

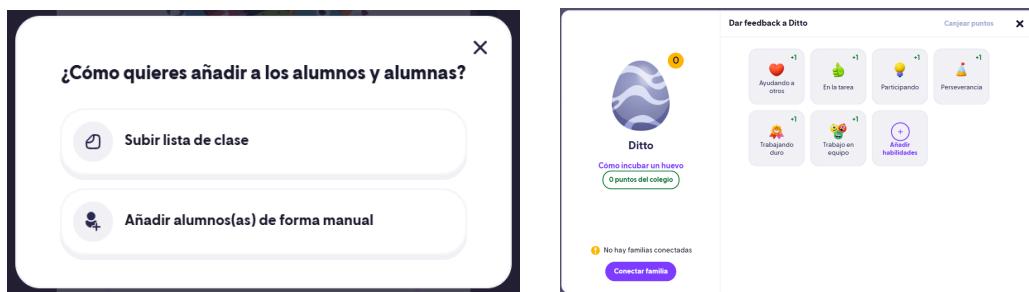
En conclusión, este proyecto se sitúa en un punto intermedio entre innovación educativa, tecnología accesible y concienciación ambiental. No pretende competir con las grandes plataformas de educación, sino ofrecer una solución sencilla, realista y útil para un entorno escolar que quiera aprender desde la experiencia directa con los animales. Y precisamente ahí, en esa mezcla de naturaleza, responsabilidad, tecnología y diversión, está su mayor fortaleza.

## 2.1. Análisis de las aplicaciones más relevantes

Para valorar correctamente el alcance y la originalidad de *EggStat*, se ha realizado una revisión de algunas de las aplicaciones más populares en el ámbito educativo y de gestión animal, analizando sus funcionalidades, su enfoque y su posible relación con los objetivos de esta app.

Como se mencionó anteriormente, entre las apps educativas basadas en gamificación más conocidas se encuentran ***ClassDojo*** y ***Kahoot!***.

*ClassDojo* permite a los docentes asignar puntos a los alumnos por comportamientos positivos, enviar mensajes a las familias o gestionar actividades del aula, todo ello en un entorno visual muy amigable. Además, al comenzar a crearte una cuenta de profesor, te pregunta cómo quieres añadir a tus alumnos. Esta práctica es muy habitual en aplicaciones educativas, quedando en las manos del profesor el añadir los usuarios “alumno” de forma manual tras registrarse ellos mismos. Esta práctica ha sido clave para desarrollar el sistema de creación de alumnos de *EggStat*.



Capturas de pantalla del sistema de registro de alumnos y de puntuación de *ClassDojo*. Fuente: [teach.classdojo.com](http://teach.classdojo.com)

Otras plataformas educativas enfocadas al aprendizaje de programación, como ***Delightex*** (anteriormente conocida como ***CoSpaces***), tienen un proceso de registro que me inspiró para realizar el inicio de sesión especial en la app. Esta plataforma ofrece, al igual que en EggStat, la oportunidad de elegir si acceder con una cuenta de profesor o de alumno (de una forma más avanzada), permitiendo así que el profesor tenga control sobre los alumnos

que se registran en la aplicación, proporcionando un código de clase al alumno que quiere registrarse.



Capturas de pantalla del sistema de registro de alumnos en Delightex. Fuente: [edu.delightex.com/Auth/Signup](http://edu.delightex.com/Auth/Signup)

En el campo de las aplicaciones más vinculadas al mundo animal, existen apps como **Farm Simulators** o **My Chicken Coop**, que aunque tienen temáticas similares (granjas, animales, cuidado), se presentan como videojuegos más que como herramientas educativas reales. Son juegos cerrados, sin posibilidad de interacción con datos reales ni de aplicación en contextos escolares. Por tanto, su utilidad educativa es más limitada, al no conectar con experiencias reales ni permitir que los datos reflejen acciones realizadas físicamente por el alumnado.

Frente a estas propuestas, **EggStat se diferencia por ofrecer una herramienta educativa enfocada al cuidado animal en un entorno escolar real**, utilizando datos en tiempo real, almacenamiento en la nube, gráficos interactivos y una interfaz adaptada al público infantil. Además, introduce una estructura de roles (profesor/alumno), lo que permite ajustar las funcionalidades según el perfil de usuario, algo que no todas las apps educativas ofrecen de forma nativa, aunque suele ser una práctica habitual en el sector.

Tras este análisis se concluye que *EggStat* puede llegar a ser una solución muy especializada, con un enfoque único, adaptado a la realidad de las aulas, y con potencial de ampliación a otros entornos como huertos escolares o refugios animales.

### 3. Metodología usada

Para el desarrollo de la aplicación *EggStat*, se ha seguido una metodología de trabajo basada en **Kanban**, combinada con un **ciclo de vida iterativo-incremental**. Este enfoque ha permitido planificar y visualizar el avance del proyecto de forma flexible, adaptándose a los retos que surgían durante la programación y permitiendo integrar mejoras constantes en cada nueva iteración.

#### 3.1. Metodología Kanban

El uso de Kanban como sistema de organización ha facilitado la división del trabajo en tareas más pequeñas, visibles y manejables, que se iban moviendo por columnas:

- “**To-do**”: donde se colocan las tareas que se deben realizar.
- “**Doing**”: donde se colocan las tareas que se están realizando.
- “**Done**”: donde se colocan las tareas ya realizadas.

Además, conforme iba avanzando el proyecto, y para no desviarnos del objetivo de conseguir una aplicación funcional viable para la fecha de entrega, se añadieron unas columnas extra referidas a cada pantalla de la aplicación, para ir indicando posibles mejoras de futuro en cada una de ellas.

Esta metodología visual resultó especialmente útil en las fases intermedias del desarrollo, donde las funcionalidades se iban implementando de forma progresiva pero no necesariamente en un orden lineal. Gracias a *Kanban*, fue más fácil priorizar tareas urgentes, reorganizar el flujo de trabajo en función del tiempo disponible, separar tareas para realizar en un futuro, e identificar claramente lo que ya estaba hecho y lo que quedaba pendiente.

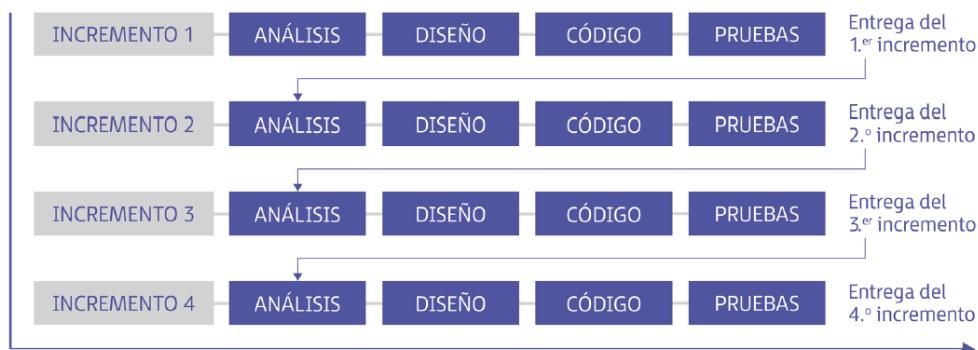
#### 3.2. Ciclo de vida iterativo-incremental

A nivel de desarrollo, se optó por un **ciclo de vida iterativo-incremental**, que consistía en construir el proyecto poco a poco, añadiendo nuevas funcionalidades, y probándolas de forma continua. Cada iteración partía de una versión funcional anterior sobre la que se añadían nuevas características, lo que permitió realizar ajustes rápidamente sin tener que esperar al final del proyecto para probarlo todo. Esta metodología me ayudó a permitir experimentar con herramientas más desconocidas para mí, como Firebase o librerías externas, sin comprometer la estabilidad del sistema.

Con todo ello, a continuación se muestran las iteraciones más significativas que marcaron el proceso:

- 1. Diseño previo y validación de ideas:** Creación de un primer esquema visual sobre cómo se quería que se viera la app, definición de las funcionalidades “innegociables”, realización de pequeñas aplicaciones de prueba para comprobar la viabilidad de lo que se quería conseguir, verificación de conocimientos necesarios para implementar cada componente.
- 2. Configuración de Firebase y primeras pruebas:** Integración de *Firebase Realtime Database* y *Firebase Authentication*, creación de los nodos principales, implementación de lectura/escritura, pruebas para comprobar la sincronización de datos con la interfaz, resolución de errores como la necesidad de emails válidos.
- 3. Implementación del sistema de login y estructura base:** Implementación de *LoginActivity* y *MainActivity*, configuración del archivo *AndroidManifest.xml* para definir la pantalla inicial, creación de la navegación inferior con sus fragments principales, primera toma de contacto con *MVVM*, investigando su funcionamiento y aplicándolo al proyecto.
- 4. Visualización de datos iniciales en pantalla:** Desarrollo de la pantalla Home con puntos del usuario y saludo personalizado, implementación de funcionalidades para mostrar información desde *Firebase* en tiempo real, comienzo de diseño estructural de *layouts XML*, sin estilos todavía, pruebas con escuchadores en tiempo real.
- 5. Implementación de funcionalidades del profesor (Control):** Creación del *fragment de Control*, accesible solo por usuarios profesor, desarrollo de los formularios para crear gallinas, crear equipos, crear alumnos, canjear puntos y resetear huevos de gallinas, investigación e implementación de *Firebase Storage* para las imágenes progresiva mejora de validaciones y feedback visual en diálogos.
- 6. Funcionalidades generales:** desarrollo del *fragment de Gallinas*, *fragment de Huevos* con botón animado y spinner para registrar huevos, *fragment de Estancias* con lógica de estado (limpio, medio, sucio) y revisión de zonas, suma de puntos según acción, aplicación de métodos en la clase *Utils* para evitar duplicación de código.
- 7. Estadísticas y gráficos interactivos:** Integración de la librerías externas, creación de gráficos circulares y de barras, implementación de métodos de actualización, personalización visual con colores personalizados.
- 8. Mejoras de interfaz y estética final:** creación del logo y extracción de su paleta de colores, aplicación en *colors.xml* y elementos visuales, mejora del layout visual, creación de un fondo de pantalla personalizado, personalización de los diálogos informativos, creación de iconos personalizados para algunos botones.

**9. Revisión y puesta a punto:** Limpieza de datos de prueba en Firebase, comprobación del rendimiento en gráficos con muchos datos, documentación del código y mejora de estructura de carpetas, revisión de layouts y reutilización de código donde fue posible.



*Esquema de la metodología iterativo incremental. Fuente: Libro Entornos de desarrollo (Ilerna)*

Cada una de estas iteraciones ha contado con una fase:

- **Fase de análisis:** donde se definen los objetivos y funcionalidades a desarrollar en cada iteración.
- **Fase de diseño:** donde se diseña toda la estructura de cada funcionalidad.
- **Fase de código:** donde se desarrolla el código de estas funcionalidades.
- **Fase de pruebas:** donde se prueban todas estas funcionalidades y se realizan los cambios necesarios para solventar los posibles errores.

## 4. Tecnologías y herramientas utilizadas en el proyecto

### Lenguaje de programación: Kotlin.

Lenguaje de programación usado en Android Studio, de sintaxis clara, moderno, fácil de mantener, que permitirá implementar las funciones de registro de huevos, autenticación de usuarios y estadísticas. Ofrece mejor rendimiento que Java en muchos casos y facilita la integración con *Firebase*.

### Entorno de programación: Android Studio.

Para el desarrollo de la aplicación en Android, por su soporte para Kotlin, su integración con Firebase y por disponer de emulador para probar la app antes de implementarla en un dispositivo real.

### Firebase

Se utilizaron tres de sus servicios: Realtime Database, para la base de datos, donde almacenar información sobre las gallinas, huevos y registros de usuarios; Authentication, para gestionar la autenticación con diferentes roles (profesor o alumno); Storage: para almacenar las imágenes de las gallinas y las imágenes animadas de los equipos y diálogos informativos. Ofrece solución en la nube, y es muy fácil de integrar con Android Studio y Kotlin, lo cual hace que el desarrollo de la app sea más rápido.

### Herramienta para la redacción de la memoria y diapositivas: Google Docs

Al ser en la nube y gratuito me permite acceder desde cualquier dispositivo a la documentación.

### Herramienta para la grabación de vídeo: OBS Studio.

Programa gratuito de grabación de pantalla utilizado para capturar el funcionamiento de la app junto con la explicación en vídeo mediante webcam integrada, y con el que ya tengo experiencia. Utilizado para la presentación final.

### LucidApp

Aplicación online utilizada para la creación del diagrama de clases, permitiendo representar de forma visual la estructura de la app y las relaciones entre entidades.

### Canva

Herramienta de diseño gráfico muy intuitiva utilizada para múltiples fines: diseño del esquema inicial, edición del fondo de pantalla, diseño de iconos, creación de infografías y elaboración del cronograma.

**Trello**

Aplicación usada para aplicar la metodología Kanban, facilitando la organización de tareas en columnas ("Por hacer", "En progreso", "Hecho"), lo que permite visualizar el progreso del desarrollo de manera clara.

**Dall-E (OpenAI)**

Herramienta de inteligencia artificial usada para la generación del logo de la aplicación. A partir de descripciones textuales, se creó una imagen representativa de un huevo relleno de otros huevos de colores, que sirvió de inspiración para la paleta visual del proyecto.

**MPAndroidChart**

Librería externa utilizada para la visualización de estadísticas en forma de gráficos (circular y de barras). Fue la mejor opción encontrada para la representación clara e interactiva de los datos almacenados en Firebase.

**Glide**

Librería de carga de imágenes optimizada para Android. Permite mostrar imágenes estáticas y animadas (GIF) desde Firebase Storage o desde el propio proyecto. Fue utilizada para mostrar imágenes de gallinas, equipos y mensajes visuales en los diálogos.

## 5. Planificación, Diagnóstico y Contexto Laboral

Antes de comenzar con la planificación temporal del proyecto **EggStat**, se realizó un esquema visual de cómo se vería la aplicación, y qué funciones tendría, acordes a los tiempos de los que se disponían. Sin esta primera idea hubiera sido imposible realizar una planificación real del proceso de creación del proyecto, ya que siempre surgían nuevas funcionalidades que se deseaban implementar. A continuación, se detalla toda la planificación, el diagnóstico y el contexto laboral de la aplicación.

### 5.1. Planificación

Con esta primera idea mental hecha, se realizó una planificación, permitiendo organizar el trabajo de forma estructurada y realista. Esta planificación permite facilitar el control de los tiempos y tareas, a la vez que permite anticiparse a posibles dificultades y adaptarse de forma flexible a los imprevistos. Teniendo en cuenta que el desarrollo del proyecto se compagina con una jornada laboral completa, la organización del tiempo fue un reto para poder conseguir la realización de un proyecto mínimo viable.

Se han implementado dos **diagrama de Gantt** (uno inicial y otro final), donde se puede visualizar la distribución temporal de las diferentes fases y tareas que componen el proyecto. Estos diagramas se realizaron con la herramienta **Canva**, y han servido tanto para planificar como para hacer seguimiento real del progreso.

En un primer momento, se estableció una estimación temporal para cada fase del proyecto, distribuyendo las tareas en bloques semanales durante un periodo de ocho semanas. Las fases de realización, en orden, incluyeron: configuración del entorno de desarrollo y Firebase, login y registro de usuarios, modelo de datos y base de datos en Firebase, pantalla de gallinas y registro de huevos, pantalla de estancias y lógica de estado, sistema de puntuación por equipo, pantalla de estadísticas, mejora de interfaz y experiencia visual y, finalmente, pruebas y ajustes finales.

Todas estas tareas se han ido marcando y desglosando en la herramienta de gestión **Trello**, para facilitar la visualización de tareas a realizar, tareas en realización y tareas realizadas. Esta herramienta ha ayudado mucho a mantener un orden en el desarrollo de todas las funcionalidades de la app, además de ayudar a ir recopilando ideas para realizar en un futuro que finalmente no han sido posibles de realizar con el tiempo disponible.

Tras realizar la planificación, se procedió a ir desarrollando el código de la app, siguiendo en su mayoría la temporalización marcada en el diagrama de Gantt, aunque algunas funcionalidades, como la de creación de usuarios y autenticación ralentizaron unos días las

expectativas iniciales de temporalización. Tanto la configuración de Firebase, que dio algunos problemas al inicio, y la implementación de Firebase Storage, inicialmente no planificada, ralentizaron unos días la expectativa de consecución de una primera app inicial funcional. Debido a ello, se redujeron los tiempos previstos para otras tareas dejando de implementar algunas funcionalidades que finalmente se plasmarán en vías futuras.

Además, se cambió el orden de implementación de funcionalidades, ya que se estableció como prioridad el funcionamiento correcto de la base de datos, por lo que se hizo hincapié en el funcionamiento correcto de la creación de usuarios, equipos y gallinas, y en el sistema de puntuación. Se dejó para el final toda la parte estética de la app, lo cual también llevó más tiempo del esperado, ya que al ser una app de gamificación y educativa, enfocada a un público infantil, una prioridad era que esta fuera llamativa, y es por ello que se le ha dedicado un tiempo extra tanto al control de errores como a que en diferentes notificaciones de la app se visualicen imágenes y mensajes divertidos.

Finalmente, teniendo todo esto en cuenta, se realiza un segundo diagrama de Gantt, en el que también se incluye en la semana final la realización de la memoria escrita, previamente no planificada en el primer diagrama.

Tras la comparativa de un diagrama y otro, se concluye que las fases quedan de la siguiente manera:

- **Semana 1 y 2:** Configuración de Android Studio, Firebase y Firebase Storage, tarea previamente estimada para una semana, en el que se configura mediante varias pruebas tanto Android Studio, como Firebase Realtime Database, Firebase Auth y Firebase Storage.
- **Semana 2 y 3:** Programación de la lógica de logueo de usuarios, creación de usuarios, creación de equipos y creación de gallinas (pantalla “control”), tarea previamente estimada para una semana.
- **Semana 3:** A la vez, estructuración del modelo de datos en Firebase, toma de decisiones sobre todos los nodos, relaciones y atributos de cada entidad, previamente estimado en una semana y media.
- **Semana 3 y 4:** Programación de la lógica de puntuación de usuarios y equipos, además de la pantalla “home”, tarea adelantada ya que estaba previamente estimada para comenzar más tarde.
- **Semana 4 y 5:** Programación de la pantalla “gallinas” y “huevos”, atrasada una semana respecto al planning inicial.

- **Semana 5:** Programación de la pantalla “estancias” y la lógica del estado de la estancia, atrasada una semana respecto al planning inicial.
- **Semana 5 y 6:** Programación de la pantalla “estadísticas”.
- **Semana 6 y 7:** Realización de mejoras de la interfaz visual y control de errores, previamente planificada para una semana y media de duración, ampliada a dos semanas.
- **Semana 7 y 8:** Realización de las pruebas finales y realización de la memoria escrita, junto a la presentación en diapositivas y vídeo.



Diagrama de Gantt inicial y final del proyecto. Elaboración propia. Ampliado en el [anexo](#).

## 5.2. Diagnóstico y contexto laboral

Junto a la planificación temporal, se ha realizado un análisis DAFO, clave para realizar un diagnóstico sobre el proyecto, con el objetivo de comprender el contexto general del proyecto, detectar posibles riesgos y reforzar aquellos aspectos positivos que pueden potenciarse. A continuación se muestra el análisis realizado, plasmando las fortalezas, debilidades, oportunidades y amenazas que podría tener la app en el mercado laboral:

### Fortalezas:

- Temática original y educativa, centrada en promover el cuidado de los animales y el conocimiento de la producción de huevos.
- Gamificación para ayudar a profesores a motivar a los alumnos.
- Adaptable a distintos centros educativos y niveles de alumnos.

### Debilidades:

- Nicho de uso muy específico, enfocado principalmente a entornos escolares con espacio y recursos para poder tener un gallinero.

- Dependencia de conexión a internet para el uso de Firebase, lo que hace que en momentos las imágenes tarden en cargar o no se pueda tener acceso a Firebase Database.
- Al estar en fase de desarrollo inicial, podría requerir de mejoras de rendimiento o de estabilidad.

**Oportunidades:**

- Potencial para fomentar la obtención de algunos Objetivos de Desarrollo Sostenible de la Agenda 2030 de la ONU, como el 4 (Educación de calidad), el 12 (Producción y consumo responsables) o el 15 (vida de ecosistemas terrestres).
- Escalabilidad a otras especies o tipos de cuidados (granjas educativas, refugios, huertos escolares...)
- Posibilidad de mejorar la gamificación para aumentar el interés de su uso en alumnos.

**Amenazas:**

- Baja demanda si no se encuentra el entorno adecuado para su aplicación.
- Existencia de otras herramientas educativas más generales.
- Cambios en la infraestructura o gestión escolar que limite su implantación.

## 6. Análisis del proyecto

En este apartado se procede a hacer un análisis exhaustivo de la aplicación **EggStat**. Esta aplicación ha sido desarrollada con el propósito de gamificar el entorno de un colegio que quiera fomentar el cuidado de animales de granja, en este caso gallinas, permitiendo a los alumnos obtener recompensas en forma de puntos canjeables al cuidar de estos animales. Teniendo en cuenta estos objetivos, a continuación se analizan los requisitos funcionales y no funcionales de la aplicación, además de realizar un diagrama entidad-relación, diagrama de casos de uso y diagrama de clases para completar el análisis.

### 6.1. Requisitos funcionales y no funcionales

Las acciones y comportamientos que deben permitir la aplicación serán los **requisitos funcionales** que ésta deba cumplir. En este caso, al tratarse de una aplicación educativa, es fundamental diferenciar entre roles, profesor y alumno concretamente, que permitan a uno controlar determinados aspectos de la app (como crear nuevos usuarios, canjear los puntos, etc.) y a otro interactuar con las estancias y huevos, respectivamente.

Es por ello, que esta aplicación tendrá funcionalidades generales (comunes a todos los usuarios), y funcionalidades para profesores (solo los usuarios registrados con email y contraseña). Se detallan a continuación:

Funcionalidades generales:

Identificador	Funcionalidad	Descripción
RF01	Inicio de sesión y cierre de sesión	La aplicación permite que los usuarios inicien sesión con los datos “usuario” y “contraseña”, y cierran sesión en el botón “Cerrar sesión”.
RF02	Visualización de puntos	Los usuarios registrados podrán visualizar los puntos disponibles en tiempo real en la pantalla “Home”.
RF03	Visualizar las gallinas registradas y sus datos	Los usuarios registrados podrán visualizar las gallinas registradas en la base de datos en la pantalla “gallinas”, además de ver todos sus datos: <ul style="list-style-type: none"> <li>- Nombre</li> <li>- Raza</li> <li>- Edad</li> <li>- Huevos totales registrados</li> </ul>
RF04	Visualizar estado de las estancias	Los usuarios registrados podrán visualizar en qué estado se encuentran las estancias del gallinero en la pantalla “Estancias”, y así saber si

		necesitan o no una revisión.
RF05	Visualizar las estadísticas de puntos y huevos	Los usuarios registrados podrán visualizar en la pantalla “Estadísticas” a todos los equipos registrados en la app y sus puntos totales en tiempo real, además de los huevos de todas las gallinas, en gráficos de barras y circulares.
RF06	Registrar huevos	Los usuarios registrados podrán registrar cuándo una de las gallinas ha puesto un huevo, indicándolo en un <i>dialog</i> que le preguntará qué gallina lo ha puesto. El usuario tendrá que seleccionar la gallina desde un <i>spinner</i> que cargará todas las disponibles para registrarla.
RF07	Registrar revisiones de estancias	Los usuarios registrados podrán registrar, haciendo clic sobre la imagen de la estancia correspondiente, que han revisado la estancia (bebedero, comedero o gallinero).
RF08	Obtener puntos por acciones completadas	Los usuarios registrados obtendrán puntos dependiendo de la acción realizada: <ul style="list-style-type: none"> <li>- 5 puntos cada vez que registren un huevo</li> <li>- 10 puntos cada vez que rellenen el bebedero</li> <li>- 10 puntos cada vez que rellenen el comedero</li> <li>- 50 puntos cada vez que limpien el gallinero</li> </ul>

Funcionalidades para profesores:

Identificador	Funcionalidad	Descripción
RF09	Registro de cuenta de profesor	Los usuarios que sean en principio profesores, tendrán acceso a la clave secreta indicada en el manual de usuario para poder registrarse como profesor, a través del botón “crear cuenta profesor”, proporcionando los siguientes datos: <ul style="list-style-type: none"> <li>- Correo electrónico</li> <li>- Nombre de usuario</li> <li>- Contraseña</li> <li>- Equipo (seleccionado en un spinner)</li> <li>- Clave secreta de profesor</li> </ul>
RF10	Añadir nuevos usuarios “alumno”	Podrá añadir nuevos usuarios “alumno”, mediante la pantalla de control, indicando los siguientes datos: <ul style="list-style-type: none"> <li>- Nombre de usuario</li> <li>- Contraseña</li> </ul>

		- Equipo (seleccionado en un spinner)
RF11	Añadir nuevas gallinas	Podrá añadir nuevas gallinas, mediante la pantalla de control, indicando los siguientes datos: - Nombre - Raza - Fecha de nacimiento (mediante un datepicker) - Total huevos - Imagen de gallina
RF12	Añadir nuevos equipos a la base de datos	Podrá añadir equipos, mediante la pantalla de control, de una lista de equipos ya existentes recopilados en un spinner, que al seleccionar éste pasará al nodo “equipo”.
RF13	Canjear los puntos de un equipo	Podrá canjear los puntos de un equipo de los disponibles, siempre y cuando estos puntos no superen el total de los puntos del equipo.
RF14	Canjear los puntos de un alumno	Podrá canjear los puntos de un alumno de los disponibles, siempre y cuando estos puntos no superen el total de los puntos del alumno.
RF15	Resetear huevos gallinas	Podrá cambiar el valor del atributo “total_huevos” de las gallinas a 0 para comenzar una nueva estadística de huevos.

Los requisitos **no funcionales** describen las características generales que debe tener la app para que sea fiable, eficiente y fácil de usar. Es por ello que se definen los siguientes:

Identificador	Funcionalidad	Descripción
RNF01	Usabilidad	La aplicación debe ser visualmente atractiva y fácil de usar para niños, con una interfaz intuitiva y divertida.
RNF02	Compatibilidad	La aplicación debe estar diseñada para funcionar en dispositivos Android, asegurando su rendimiento en diversas versiones del sistema operativo.
RNF03	Rendimiento	Los datos deben almacenarse en tiempo real usando Firebase Realtime Database, lo que garantiza un acceso rápido y en tiempo real a la información.
RNF04	Almacenamiento de imágenes	La subida de imágenes debe realizarse mediante Firebase Storage, con una carga eficiente y

		segura de imágenes para los equipos y gallinas.
RNF05	Adaptabilidad de funcionalidad	Las funcionalidades de la aplicación deben adaptarse a los distintos roles de usuario (alumno/profesor), ofreciendo una experiencia diferenciada según el tipo de usuario.
RNF06	Validaciones y mensajes informativos	El sistema debe proporcionar mensajes informativos claros (y divertidos en este caso), además de validaciones en caso de errores o acciones incompletas, asegurando que los usuarios comprendan qué hacer.
RNF07	Visualización de estadísticas	La aplicación debe utilizar gráficos visuales (como <i>MPAndroidChart</i> ) para representar estadísticas de manera clara y comprensible.
RNF08	Modularidad y mantenimiento	El código debe estar organizado según el patrón MVVM, lo que facilita su mantenimiento, escalabilidad y futuras mejoras en la aplicación.
RNF09	Seguridad y autenticación	Los usuarios deben estar autenticados mediante Firebase Auth para realizar operaciones de lectura/escritura en la base de datos, garantizando la seguridad y privacidad de los datos.

## 6.2. Diagrama entidad-relación

Se realiza un diagrama entidad-relación para visualizar las entidades clave: Equipo, Usuarios, Gallinas y Estancia. En este diagrama se puede observar la relación entre ellas, concluyendo que:

- Un Usuario pertenece a un solo Equipo, y un Equipo puede tener muchos Usuarios: relación uno a muchos.
- Un Usuario puede registrar huevos en muchas Gallinas, y los huevos de Gallinas pueden ser registradas por muchos Usuarios: relación muchos a muchos.
- Un Usuario puede realizar revisiones en muchas Estancias, y las Estancias pueden ser revisadas por muchos Usuarios: relación muchos a muchos.
- De forma indirecta, la entidad Equipo se relaciona con la entidad Estancia, ya que cuando un Usuario con un Equipo revisa una estancia, el atributo puntos\_equipo cambia.
- Al igual pasaría con la entidad Equipo y Gallina, ya que cuando un usuario registra un huevo de Gallina, el atributo puntos\_equipo cambia.

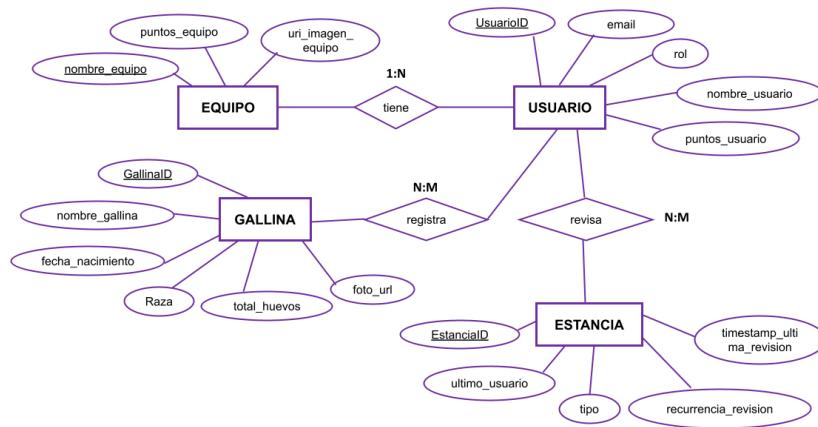


Diagrama entidad-relación. Elaboración propia. Ampliado en el [anexo](#).

### 6.3. Casos de uso

Se realiza el análisis de varios casos de uso para representar las interacciones posibles entre los usuarios y la aplicación, ayudando así al control del flujo del funcionamiento de la misma, control de errores, usabilidad... A continuación, veremos los casos de uso más importantes, describiendo el caso de uso, su precondición (situación previa a la realización de la acción), cómo debería ser su secuencia normal, su postcondición (situación posterior a la realización de la acción) y excepciones que puedan producirse:

#### CU01: Login

<b>Descripción</b>	El sistema permite que el usuario registrado inicie sesión en la aplicación para poder acceder a la misma.
<b>Precondición</b>	El usuario no está registrado.
<b>Secuencia normal</b>	<ol style="list-style-type: none"> <li>El sistema solicita al usuario que ingrese su usuario y contraseña.</li> <li>El usuario introduce los datos y selecciona <i>Iniciar Sesión</i>.</li> <li>El sistema verifica que los datos sean válidos en la base de datos.</li> <li>Si los datos son los correctos, el usuario es redirigido a la pantalla principal de la aplicación.</li> </ol>
<b>Postcondición</b>	El usuario está registrado y ya tiene acceso a la pantalla principal.
<b>Excepciones</b>	El caso de uso no se completa:

	<ol style="list-style-type: none"> <li>1. Si el usuario no introduce los datos obligatorios, el sistema muestra una notificación informando de que debe introducir todos los campos.</li> <li>2. Si los datos no son los correctos, se muestra un diálogo con una imagen y un mensaje indicando el error.</li> <li>3. Si hay un error en la base de datos, se muestra un mensaje de error.</li> </ol>
--	---

**CU02: Recuperar contraseña (solo profesores)**

<b>Descripción</b>	El sistema permite que el usuario recupere su contraseña si la olvida.
<b>Precondición</b>	El usuario debe haberse registrado con correo electrónico previamente.
<b>Secuencia normal</b>	<ol style="list-style-type: none"> <li>1. El usuario selecciona la opción <i>¿Olvidaste tu contraseña?</i>.</li> <li>2. El sistema solicita al usuario que indique si es un <i>Alumno</i> o un <i>Profesor</i>.</li> <li>3. Dependiendo de la opción elegida:           <ol style="list-style-type: none"> <li>a. Si elige alumno, se muestra un mensaje indicando que debe pedirle a su profesor la contraseña.</li> <li>b. Si elige profesor, el sistema le pide que introduzca el correo electrónico con el que se registró.</li> </ol> </li> <li>4. El usuario ingresa su correo y selecciona <i>Enviar</i> para solicitar la recuperación.</li> <li>5. El sistema muestra un aviso y envía un enlace al correo electrónico para restablecer la contraseña.</li> </ol>
<b>Postcondición</b>	El usuario recibe un correo con el enlace para cambiar contraseña.
<b>Excepciones</b>	<p>El caso de uso no se completa:</p> <ol style="list-style-type: none"> <li>1. Si el usuario introduce un correo electrónico con mal formato, se muestra un aviso.</li> <li>2. Si el usuario introduce un correo electrónico no registrado en la base de datos, no recibirá ningún correo de recuperación.</li> </ol>

	<ol style="list-style-type: none"> <li>3. Si hay un error en la base de datos, se muestra un mensaje de error.</li> <li>4. Si el usuario selecciona <i>Cancelar</i>, se cierra el diálogo sin realizar ninguna acción.</li> </ol>
--	---

**CU03: Crear cuenta**

<b>Descripción</b>	El sistema permite que un nuevo usuario se registre en la aplicación si conoce la clave para hacerlo.
<b>Precondición</b>	El usuario no tiene una cuenta previamente.
<b>Secuencia normal</b>	<ol style="list-style-type: none"> <li>1. El usuario selecciona <i>Crear cuenta profesor</i>.</li> <li>2. El usuario introduce todos los datos necesarios para crear la cuenta.</li> <li>3. El usuario selecciona <i>Crear</i> para añadir los datos a la base de datos.</li> </ol>
<b>Postcondición</b>	El usuario tiene cuenta en la base de datos y puede iniciar sesión con las credenciales proporcionadas.
<b>Excepciones</b>	<p>El caso de uso no se completa:</p> <ol style="list-style-type: none"> <li>1. Si falta alguno de los campos por llenar, se muestra un aviso.</li> <li>2. Si el usuario introduce un correo electrónico con mal formato, se muestra un aviso.</li> <li>3. Si el usuario introduce un correo electrónico que ya está en la base de datos, se muestra un aviso.</li> <li>4. Si las contraseñas no coinciden, se muestra un aviso.</li> <li>5. Si la clave de profesor no es la correcta, se muestra un aviso (malvado).</li> <li>6. Si hay un error en la base de datos, se muestra un mensaje de error.</li> </ol>

**CU04: Visualizar todas las pantallas de la aplicación.**

<b>Descripción</b>	El sistema permite que un usuario logueado navegue entre todas las pantallas disponibles en el navegador.
<b>Precondición</b>	El usuario debe estar logueado.
<b>Secuencia normal</b>	<ol style="list-style-type: none"> <li>1. El usuario accede a la pantalla <i>Home</i> y visualiza sus puntos y su imagen de equipo.</li> <li>2. El usuario selecciona el ícono inferior de <i>Gallinas</i> y visualiza todas las gallinas disponibles.</li> <li>3. El usuario selecciona el ícono inferior de <i>Huevos</i> y visualiza la imagen de la aplicación en movimiento.</li> <li>4. El usuario selecciona el ícono inferior de <i>Estancias</i> y visualiza las imágenes y los datos de las estancias.</li> <li>5. El usuario selecciona el ícono inferior de <i>Estadísticas</i> y visualiza los dos gráficos con las estadísticas de los puntos de equipos y huevos de las gallinas.</li> </ol>
<b>Postcondición</b>	El usuario conoce todos los datos de las gallinas y
<b>Excepciones</b>	<p>El caso de uso no se completa:</p> <ol style="list-style-type: none"> <li>1. Si no hay gallinas registradas, no aparecen en la pantalla gallinas y no se puede realizar la acción.</li> <li>2. Si hay un error en la base de datos, se muestra un mensaje de error.</li> </ol>

**CU05: Registrar un huevo**

<b>Descripción</b>	El sistema permite que un usuario registre un huevo de una gallina seleccionada.
<b>Precondición</b>	El usuario debe estar logueado y debe haber gallinas en la base de datos.
<b>Secuencia normal</b>	<ol style="list-style-type: none"> <li>1. El usuario accede a la pantalla <i>Huevos</i>.</li> <li>2. El sistema muestra las gallinas disponibles.</li> <li>3. El usuario selecciona una gallina.</li> </ol>

	4. El usuario confirma el registro del huevo. 5. El sistema actualiza la base de datos con el nuevo huevo registrado.
<b>Postcondición</b>	El huevo es registrado en la base de datos, actualizando la gallina correspondiente, y sumando puntos a usuario y equipo de usuario.
<b>Excepciones</b>	El caso de uso no se completa: 1. Si no hay gallinas registradas, no aparecen en la pantalla gallinas y no se puede realizar la acción. 2. Si hay un error en la base de datos, se muestra un mensaje de error.

**CU06: Revisar estancia**

<b>Descripción</b>	El sistema permite que un usuario registre una revisión de una estancia (bebedero, comedero, gallinero).
<b>Precondición</b>	El usuario debe estar logueado.
<b>Secuencia normal</b>	1. El usuario accede a la pantalla <i>Estancias</i> . 2. El sistema muestra las estancias disponibles y su estado (lleno, vacío, limpio, sucio...) 3. El usuario selecciona una estancia. 4. El usuario confirma la revisión. 5. El sistema actualiza la base de datos con la información de la revisión.
<b>Postcondición</b>	La estancia seleccionada es marcada como recién revisada en la base de datos, sumando los puntos correspondientes tanto al usuario como al equipo del usuario.
<b>Excepciones</b>	El caso de uso no se completa: 1. Si el usuario pulsa en cancelar, se cierra el diálogo. 2. Si hay un error en la base de datos, no se muestran los datos de la estancia y se muestra un mensaje de error.

**CU07: Acciones del profesor: añadir usuario**

<b>Descripción</b>	El sistema permite que un usuario profesor añada usuarios alumno.
<b>Precondición</b>	El usuario debe estar logueado como profesor.
<b>Secuencia normal</b>	<ol style="list-style-type: none"> <li>1. El usuario accede a la pantalla <i>Control</i> a través del menú superior de la pantalla <i>Home</i>.</li> <li>2. En esta pantalla, el usuario selecciona el botón <i>Añadir alumno</i>.</li> <li>3. Se muestra un diálogo, donde el usuario debe introducir todos los datos necesarios.</li> <li>4. El usuario confirma seleccionando <i>Crear</i>.</li> <li>5. El sistema actualiza la base de datos con la información del nuevo usuario registrado y muestra un aviso confirmándolo.</li> </ol>
<b>Postcondición</b>	El sistema crea un nuevo usuario en la base de datos con los datos proporcionados por el usuario profesor.
<b>Excepciones</b>	<p>El caso de uso no se completa:</p> <ol style="list-style-type: none"> <li>1. Si falta alguno de los campos por llenar, se muestra un aviso.</li> <li>2. Si las contraseñas no coinciden, se muestra un aviso.</li> <li>3. Si el usuario pulsa en cancelar, se cierra el diálogo.</li> <li>4. Si hay un error en la base de datos, se muestra un mensaje de error.</li> </ol>

**CU08: Acciones del profesor: añadir gallina**

<b>Descripción</b>	El sistema permite que un usuario profesor añada gallinas a la base de datos.
<b>Precondición</b>	El usuario debe estar logueado como profesor.
<b>Secuencia normal</b>	<ol style="list-style-type: none"> <li>1. El usuario accede a la pantalla <i>Control</i> a través del menú superior de la pantalla <i>Home</i>.</li> <li>2. En esta pantalla, el usuario selecciona el botón <i>Añadir</i></li> </ol>

	<p><i>gallina.</i></p> <ol style="list-style-type: none"> <li>3. Se muestra un diálogo, donde el usuario debe introducir todos los datos necesarios, seleccionar una fecha en un <i>Datepicker</i>, y seleccionar una imagen de la galería de su dispositivo móvil.</li> <li>4. El usuario confirma seleccionando <i>Crear</i>.</li> <li>5. El sistema muestra un diálogo de carga mientras la imagen se está subiendo a la base de datos.</li> <li>6. El sistema actualiza la base de datos con la información de la nueva gallina y muestra un aviso confirmándolo.</li> </ol>
<b>Postcondición</b>	El sistema crea una nueva gallina en la base de datos con los datos proporcionados por el usuario profesor.
<b>Excepciones</b>	<p>El caso de uso no se completa:</p> <ol style="list-style-type: none"> <li>1. Si falta alguno de los campos por llenar, se muestra un aviso.</li> <li>2. Si el usuario pulsa en cancelar, se cierra el diálogo.</li> <li>3. Si hay un error en la base de datos, se muestra un mensaje de error.</li> </ol>

#### **CU09: Acciones del profesor: añadir equipo**

<b>Descripción</b>	El sistema permite que un usuario profesor añada equipos a la base de datos.
<b>Precondición</b>	El usuario debe estar logueado como profesor.
<b>Secuencia normal</b>	<ol style="list-style-type: none"> <li>1. El usuario accede a la pantalla <i>Control</i> a través del menú superior de la pantalla <i>Home</i>.</li> <li>2. En esta pantalla, el usuario selecciona el botón <i>Añadir equipo</i>.</li> <li>3. Se muestra un diálogo, donde el usuario debe seleccionar uno de los equipos existentes en un menú desplegable.</li> <li>4. El usuario confirma seleccionando <i>Crear</i>.</li> <li>5. El sistema actualiza la base de datos con la información del nuevo equipo registrado y muestra un aviso confirmándolo.</li> </ol>

<b>Postcondición</b>	El sistema añade un nuevo equipo a la base de datos con los datos proporcionados por el usuario profesor.
<b>Excepciones</b>	<p>El caso de uso no se completa:</p> <ol style="list-style-type: none"> <li>1. Si el usuario no selecciona un equipo, se muestra un aviso.</li> <li>2. Si el usuario pulsa en cancelar, se cierra el diálogo.</li> <li>3. Si hay un error en la base de datos, se muestra un aviso.</li> </ol>

#### CU10: Acciones del profesor: canjear puntos (equipo o usuario)

<b>Descripción</b>	El sistema permite que un usuario profesor canjee los puntos de los usuarios o los equipos.
<b>Precondición</b>	El usuario debe estar logueado como profesor.
<b>Secuencia normal</b>	<ol style="list-style-type: none"> <li>1. El usuario accede a la pantalla <i>Control</i> a través del menú superior de la pantalla <i>Home</i>.</li> <li>2. En esta pantalla, el usuario selecciona el botón <i>Canjear puntos equipo</i> o <i>Canjear puntos usuario</i>.</li> <li>3. Se muestra un diálogo, donde el usuario debe seleccionar en un menú desplegable uno de los equipos o usuarios existentes.</li> <li>4. El usuario escribe los puntos que desea canjear, teniendo que ser estos inferiores a la cantidad de puntos que tiene el equipo o usuario actualmente.</li> <li>6. El usuario confirma seleccionando <i>Canjear</i>.</li> <li>7. El sistema actualiza la base de datos haciendo las operaciones de resta de los puntos.</li> </ol>
<b>Postcondición</b>	El sistema actualiza los puntos del usuario o equipo en la base de datos.
<b>Excepciones</b>	<p>El caso de uso no se completa:</p> <ol style="list-style-type: none"> <li>1. Si el usuario no selecciona un equipo o no indica una cantidad de puntos, se muestra un aviso.</li> <li>2. Si el usuario indica una cantidad de puntos superior a la que tiene el usuario o el equipo actualmente, se muestra un</li> </ol>

	aviso. 3. Si el usuario pulsa en cancelar, se cierra el diálogo. 4. Si hay un error en la base de datos, se muestra un aviso.
--	---

**CU11: Acciones del profesor: resetear huevos gallinas**

<b>Descripción</b>	El sistema permite que un usuario profesor resetee la cantidad de huevos de las gallinas de la base de datos.
<b>Precondición</b>	El usuario debe estar logueado como profesor.
<b>Secuencia normal</b>	1. El usuario accede a la pantalla <i>Control</i> a través del menú superior de la pantalla <i>Home</i> . 2. En esta pantalla, el usuario selecciona el botón <i>Resetear huevos gallinas</i> . 3. Se muestra un diálogo, donde el usuario debe confirmar que desea resetear estos datos, seleccionando <i>Sí, resetear</i> . 4. El sistema actualiza la base de datos.
<b>Postcondición</b>	El sistema resetea la cantidad de puntos de las gallinas existentes, poniendo a 0 todos los huevos.
<b>Excepciones</b>	El caso de uso no se completa: 1. Si el usuario pulsa en cancelar, se cierra el diálogo. 2. Si hay un error en la base de datos, se muestra un aviso.

Finalmente, se crea un diagrama visual de casos de uso de la aplicación, con la herramienta *Canva*, donde se puede visualizar el flujo de acciones que puede realizar tanto un usuario alumno como un usuario profesor:

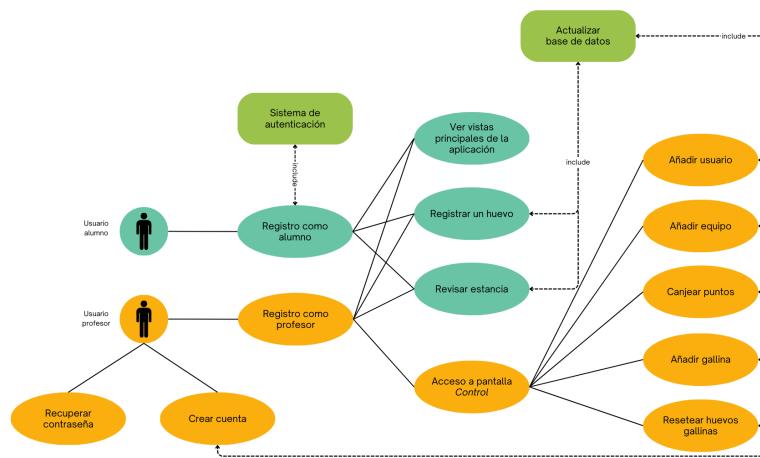


Diagrama de casos de uso. Elaboración propia. Ampliado en el [anexo](#).

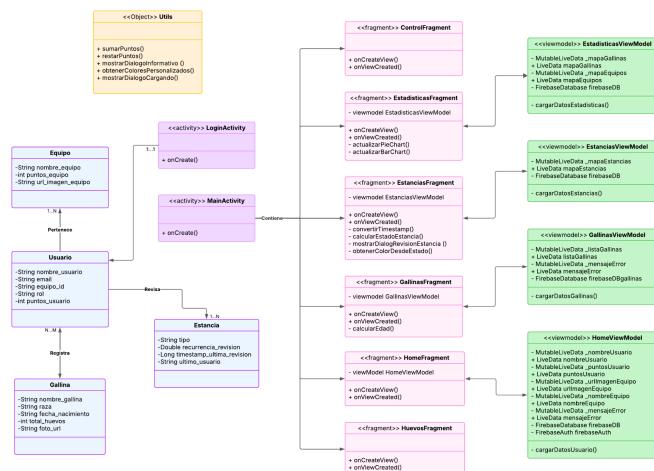
## 6.4. Diagrama de clases

Se realiza un diagrama de clases para visualizar la organización del código de la aplicación, donde se representan las principales clases que intervienen en el funcionamiento de la misma. Al comenzar a plantear qué arquitectura debía tener la aplicación para poder ser escalable, por lo que se decidió usar la arquitectura *MVVM (Model-View-ViewModel)* tras estudiar la *Guía de Arquitectura de Google*. Se decidió utilizar clases visualización de datos, y así separar funcionalidades de visualización con tratamiento de datos, además de asegurarnos de que los datos se visualizarán a tiempo real.

Es por ello que finalmente se utilizaron cinco tipos de clases:

- **Entidades** para la base de datos: Equipo, Usuario, Estancia, Gallina.
- **Activities** principales: LoginActivity y MainActivity.
- **Fragments** dentro de MainActivity: ControlFragment, EstadisticasFragment, EstanciasFragment, GallinasFragment, HomeFragment y HuevosFragment.
- **ViewModels** que interactúan con su fragment para visualizar datos en tiempo real: EstadisticasViewModel, EstanciasViewModel, GallinasViewModel y HomeViewModel.
- **Clase objeto** Utils, donde sepáramos métodos estáticos que todas las clases puedan usar sin necesidad de instanciar a la clase.

Se genera con la herramienta *Lucid.app* un diagrama de clases visual:

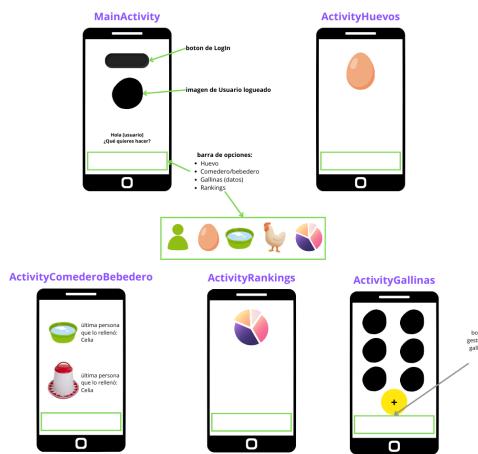
Diagrama de clases de EggStat. Elaboración propia. Ampliado en el [anexo](#).

## 7. Diseño del proyecto

En este apartado se procede a detallar todo el diseño del proyecto, comenzando por el diseño previo a la utilización de código, pasando por la configuración de la base de datos, y acabando por el desarrollo final de la aplicación, tanto la parte de código como la visual.

### 7.1. Diseño previo

Como se indicó en el apartado 5 de este documento, previamente se realizó un esquema visual de cómo quería que se viera la aplicación, y qué funcionalidades tendría. Este esquema se plasmó en la herramienta Canva, y es el siguiente:

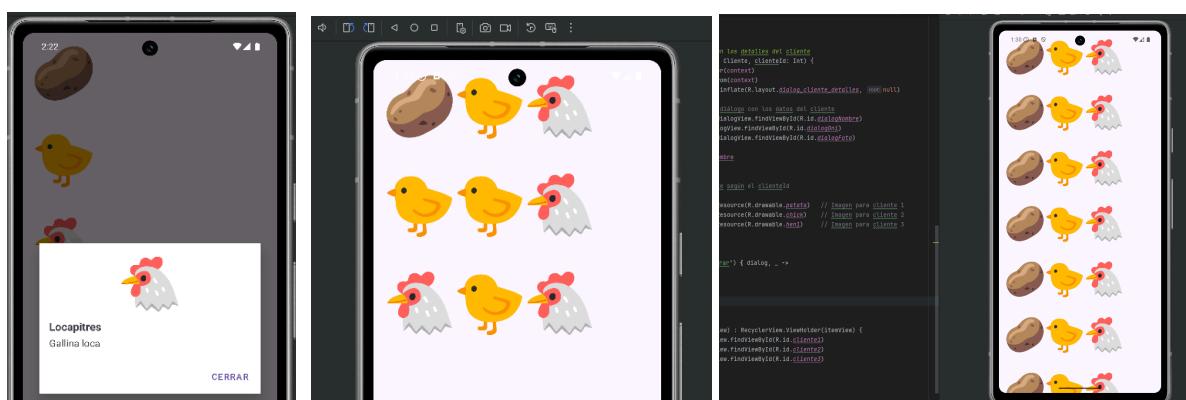
Esquema visual inicial. Elaboración propia. Ampliado en el [anexo](#).

Tras realizar este primer diseño, se concluyen las funcionalidades básicas que quiero que tenga la aplicación, algunas de las “innegociables” son: abrir diálogos informativos, mostrar

varias imágenes en una tabla y que pueden pulsarse, usar una barra inferior de navegación, que el profesor tenga una pantalla especial donde aparezcan funcionalidades extra, que haya una pantalla donde se muestren las estadísticas de los puntos de los equipos y los huevos de las gallinas, y que en las pantallas haya datos que se muestran en tiempo real.

Para verificar que las funcionalidades que se deseaban implementar eran asequibles, respecto a conocimientos y tiempo, es necesario acceder a la referencia de Android para desarrolladores, tanto para aprender a implementar un *AlertDialog* (los *Toast* me resultaban muy repetitivos y aburridos), para aprender a manejar un *NavigationBar*, implementar y llenar con datos de la base de datos un *Spinner*, manejar los datos de un *Datepicker*, implementación de gráficos con la librería externa *MPAndroidChart*, etc.

Tras aprender los conceptos básicos para la implementación de estos componentes, se procede a realizar pequeñas aplicaciones de prueba, concluyendo los componentes que son asequibles para su implementación en la aplicación final.



Imágenes de aplicaciones de prueba previas. Elaboración propia.

Tras realizar estas pruebas, se decide dar el siguiente paso para avanzar en la realización de la aplicación, configurando la base de datos.

## 7.2. Base de datos: Firebase Realtime Database y Firebase Auth

Para llevar a cabo el almacenamiento de los datos y la autenticación de los usuarios en la aplicación, se decidió utilizar el servicio de **Google Firebase**, que ofrece múltiples herramientas enfocadas a desarrolladores de aplicaciones móviles.

Desde el principio el objetivo fue que la base de datos tuviese las entidades: usuarios, gallinas, equipos y estancias. Además, cada usuario debía tener una cuenta personal, lo que implicaba implementar un sistema de autenticación. Por todo ello, se concluyó que Firebase podría ser una buena opción, en especial por lo bien que se integra con Android Studio, que es gratuita, y por la flexibilidad que ofrece con su base de datos en tiempo real.

### 7.2.1. Firebase Realtime Database

En este momento se procede a **crear la aplicación base real** de la futura aplicación final, implementando **Firebase Realtime Database**, y previamente habiendo revisado toda la documentación disponible en la página oficial de Firebase, aunque he de decir que el entorno de desarrollo Android Studio facilitó mucho los pasos a seguir con su asistente, no necesitando demasiado el consultar dicha documentación. Esta aplicación base constaba ya con las partes principales de la futura aplicación: *MainActivity.kt* y *LoginActivity.kt*, por lo que se tuvo que proceder a la configuración de estas dos en el *AndroidManifest.xml* para que apareciese al comienzo la pantalla de Login y no la MainActivity.

En Firebase Realtime Database, tras la configuración previa necesaria siguiendo los pasos de Android Studio, se comenzó añadiendo los nodos principales (equipo, estancia, gallinas y usuarios), con varios atributos para realizar pruebas de lectura y escritura desde Android Studio. En este proceso se trabajó con *ValueEventListener* y *addOnSuccessListener* para escuchar cambios y responder a ellos dentro de la interfaz de usuario. Uno de los mayores desafíos fue asegurarse de que las operaciones de escritura y lectura se ejecutaban de forma sincronizada con la interfaz gráfica, ya que en varias ocasiones los datos no se visualizaban o la aplicación se cerraba inesperadamente, teniendo que ir aprendiendo a controlar los casos de error.

### 7.2.2. Firebase Authentication

Una vez hechas varias pruebas con la base de datos sin necesidad de registrarse, se procede a implementar el registro e inicio de sesión de usuarios utilizando **Firebase Authentication**. En esta fase, se detectaron algunos errores recurrentes, como intentos fallidos de iniciar sesión debido a que no se había habilitado el método de autenticación por correo y contraseña desde la consola de Firebase (algo que no viene activado por defecto). Otro problema que se detectó es que Firebase Authentication, con su método *createUserWithEmailAndPassword*, no permite la autenticación de usuarios sin un correo electrónico. Esto suponía un **problema importante**, ya que los usuarios principales de esta aplicación son niños de entre seis y doce años, por lo que legalmente no pueden disponer de un correo electrónico. Se realizó una investigación sobre cómo habían hecho otros desarrolladores para resolver este problema, y se concluyó en realizar una programación “trampa” para crear un correo falso a los usuarios que el profesor cree, y así esquivar esta gran piedra en el camino. A continuación, se muestra cómo quedó esta lógica, en la que en el momento de añadir al usuario en la base de datos, se añade al final de la cadena de texto

“@eggstat.com”, de forma que Firebase Authentication crea que el usuario se registró con un email válido:

```
// variable para crear un email falso al usuario (requisito de FirebaseAuth)
val emailFake = "$usuario@eggstat.com"

firebaseAuth.createUserWithEmailAndPassword(emailFake, password)
    .addOnSuccessListener { result ->
        val uid = result.user?.uid ?: return@addOnSuccessListener
        val nuevoUsuario = Usuario(
            nombre_usuario = usuario,
            rol = "alumno",
            equipo_id = equipoSeleccionado,
            puntos_usuario = 0,
            email = emailFake // aquí sería su email "falso" con el que se registra en FirebaseAuth
    )
}
```

*Captura de pantalla de la lógica “crear usuario alumno”. Elaboración propia.*

De esta forma, se concluye que se tiene que controlar en el inicio de sesión esta característica de los usuarios, siendo los profesores los que se registran con un correo electrónico real, y siendo los alumnos los que se registran con un nombre de usuario y contraseña, previamente añadidas a la base de datos por un profesor. De esta forma definiremos más adelante los roles de la aplicación, para poder realizar unas acciones u otras, como indicamos previamente en los casos de uso.

A lo largo del desarrollo, los nodos de la base de datos no se mantuvieron estáticos. De hecho, algunos atributos se añadieron o modificaron conforme se implementan nuevas funcionalidades o se ajustaban a las ideas iniciales. Por ejemplo, el atributo *foto\_url* de las gallinas no se planteó desde el inicio, pero se añadió cuando se decidió incluir imágenes animadas de las gallinas en el grid. También se fue decidiendo qué nodos iban a tener un id generado por Firebase y cuales no, concluyendo en que los nodos propensos a tener muchos hijos, se generarían con ese ID (gallinas y usuarios), y el resto de nodos, al ser de cantidad “controlada” utilizarán como ID su nombre (estancias y equipos).

También hubo que revisar las reglas de seguridad de Firebase Auth y Firebase Realtime Database, configurando que solo los usuarios registrados pudiesen leer o modificar datos de la base de datos. Esto forzó a añadir la funcionalidad de *Usuario anónimo* de Firebase Authentication para el momento de creación de una cuenta profesor.

### 7.3. Diseño y desarrollo de la aplicación base

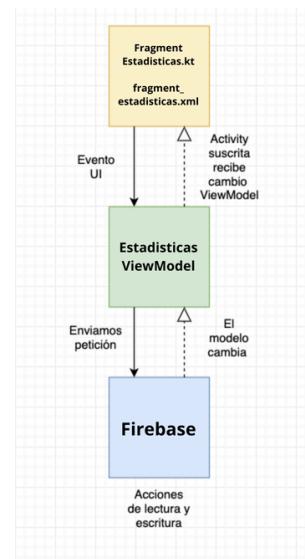
Tras configurar exitosamente la base de datos y el sistema de autenticación, se procedió al diseño y desarrollo de la estructura base de la app. En esta etapa se tomaron decisiones clave que afectaron directamente a la organización del código, ya que se quería utilizar una arquitectura escalable y moderna.

La primera decisión importante fue la elección de una barra de navegación inferior (**Bottom Navigation Bar**) como sistema de navegación principal. Esta decisión vino motivada tras consultar la guía oficial de *Material Design* sobre este componente (disponible en la bibliografía). En dicha guía explican las ventajas de este patrón, especialmente en aplicaciones con pocas secciones principales, lo cual encajaba perfectamente con el caso de EggStat, ya que la aplicación tendría cinco vistas principales: Home, Gallinas, Huevos, Estancias y Estadísticas.

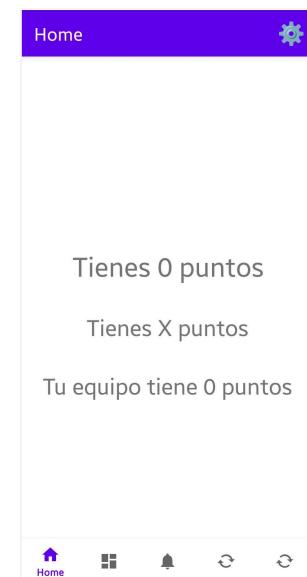
Al investigar cómo implementar correctamente este componente, se descubrió que Android Studio ofrecía una plantilla predefinida con *Bottom Navigation Bar*, por lo que decidí partir de ella para estructurar la aplicación. Esta plantilla incluía una arquitectura desconocida para mí, **MVVM (Model-View-ViewModel)**, ya implementada de forma básica. Me pareció una oportunidad interesante para aprenderlo y adaptarlo al proyecto. Tras investigar tanto en la documentación oficial de Android, como en páginas como en la documentación de desarrolladores senior de Android como *AristiDevs* sobre MVVM, considero que esta arquitectura resulta especialmente útil en este caso, ya que la aplicación debe mostrar datos en tiempo real y reaccionar automáticamente a los cambios en Firebase.

En el caso de esta aplicación, esta arquitectura se adapta utilizando Firebase Realtime Database y Firebase Auth como parte del *Model*, ya que representan la fuente de datos y lógica de acceso. A diferencia de otras aplicaciones que utilizan Room o Retrofit, aquí el acceso es directo a una base de datos remota sincronizada en tiempo real, lo que permite una visualización constante y actualizada de los datos en la interfaz del usuario.

Una vez aprendidos los conceptos básicos, como el papel de un *Fragment* dentro de una *activity*, el papel del *ViewModel* y la visualización de los datos con *LiveData*, se procede a ir creando tanto los fragments como los viewmodel de cada una de las pantallas de la futura *EggStat*, habiendo configurado también previamente la navegación del *BottomNavigationBar* en



Esquema MVVM. Elaboración propia.



Captura de la pantalla Home antes de añadir mejoras visuales. Elaboración propia.

*res/menu/bottom\_nav\_menu.xml*, donde deben declararse todos los ítems por los que navegará el menú.

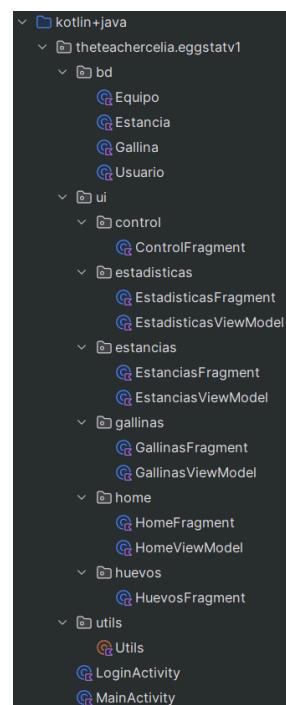
A su vez, se fueron desarrollando los **layouts XML** de cada Fragment, inicialmente centrados únicamente en el aspecto estructural (dónde colocar los elementos) y no tanto en el diseño visual definitivo. Al mismo tiempo, se iban desarrollando los layouts, se iba implementando la carga de datos básicos desde Firebase para poder probar que todo funcionaba correctamente. Por ejemplo, se comenzó a mostrar los datos de los puntos disponibles, ficticios de momento, en la pantalla *Home*, como se muestra en la imagen de la derecha.

Este enfoque progresivo (primero visualizar correctamente los datos, luego añadir otras funcionalidades) me permitió testear de forma modular cada pantalla y asegurarme de que el flujo de datos desde Firebase hacia la interfaz funcionaba como se esperaba. Fue en estas primeras versiones donde se realizaron pruebas importantes con *LiveData*, *MutableLiveData*, y con los escuchadores en tiempo real (*addValueEventListener*), fundamentales para lograr la reactividad esperada en la aplicación.

## 7.4. Implementación de funcionalidades

Una vez establecida la estructura base de la aplicación y tras probar la arquitectura MVVM, se dio paso a la implementación progresiva de las diferentes funcionalidades que darían forma y vida a la aplicación. Como punto de partida, se decidió comenzar con el desarrollo de la **pantalla Gallinas**, ya que son el núcleo de la aplicación y todo gira en torno a su cuidado, seguimiento y estadísticas.

El primer paso fue la implementación de una cuadrícula de imágenes en el *fragment\_gallinas.xml*. A pesar de que puede parecer una tarea sencilla, el control del espaciado, los márgenes y la correcta alineación en distintos tamaños de pantalla supuso un proceso de prueba y error bastante constante, y hasta ahora no considero que haya quedado visualmente correcta. Cada imagen era pulsable, y al hacerlo se abría un *AlertDialog* personalizado que mostraba información relevante sobre la gallina (raza, edad, número de huevos). Esto permitió comprobar que la interacción con los datos era fluida y que el acceso a *Firebase* desde el *ViewModel* funcionaba correctamente con las gallinas añadidas de prueba.



Captura de la pantalla *Home* antes de añadir mejoras visuales. Elaboración propia.

He de decir que en mitad del proceso de implementación notifiqué que algunos fragmentos de código podrían ser reutilizados, por lo que decidí implementar una clase **object Utils**, en la que implementar algunos de los métodos utilizados en diferentes pantallas y a los que se puede acceder de forma estática sin necesidad de instanciarla, quedando finalmente un directorio de clases como el de la imagen de la derecha. Hice hincapié en la separación por paquetes para tener una mejor organización de las mismas.

Una vez configurada esta pantalla, se comenzó con la funcionalidad clave de la aplicación: la **pantalla de Control** accesible únicamente por profesores. A esta pantalla solo se puede acceder desde un botón en la barra superior de la pantalla *Home*, que debía estar visible únicamente si el usuario autenticado ha iniciado sesión con un correo electrónico (lo que lo identifica como profesor). Este botón es configurado en el directorio *res/menu* desde el layout *menu\_home.xml*, en el que se utiliza una imagen

Desde la pantalla se implementaron tres primeras funcionalidades que se consideraron prioritarias para ir avanzando con el resto de la aplicación:

1. **Crear nuevas gallinas:** incluyendo el uso de *DatePicker* para seleccionar una fecha de nacimiento, un campo para añadir el nombre, la raza y los huevos con los que consideramos que va a comenzar, y además otro botón para añadir una imagen de gallina. En este punto nos encontramos con una decisión clave: ¿Dónde subimos la imagen de la gallina? ¿a Firebase o a la aplicación? En este punto se procede a investigar sobre **Firebase Storage**, y al tener un plan gratuito y ser fácil de implementar, se decide utilizar esta herramienta para ir almacenando ahí las imágenes de las gallinas. Por ello, se implementa tanto su subida a la plataforma, como un **ProgressBar**, ya que tras realizar unas pruebas, si no se esperaba a la subida, se podía generar un conflicto con los procesos si el usuario seguía realizando acciones. Con todos estos datos, se implementa que
2. **Crear equipos:** que en un principio se implementó para que el profesor escribiese un nombre de equipo, y finalmente se ha optado por crear una lista de equipos base y que el profesor elija uno de esos equipos desde un *Spinner*. Este spinner contendrá la lista de equipos que se han añadido a un nodo *img\_equipos*, que contienen el nombre de los equipos y una url de una imagen animada divertida que aparecerá en la pantalla *Home* al loguearte en la aplicación.
3. **Crear usuarios alumno:** introduciendo nombre, contraseña, y eligiendo uno de los equipos disponibles, se añade un usuario nuevo a la base de datos, teniendo el usuario profesor la responsabilidad de facilitarle estos datos al alumno.

4. **Canjear puntos de usuario y equipo:** se selecciona un equipo o un usuario desde un *Spinner* que carga todos los hijos del nodo que se seleccione, se muestra en un *TextView* todos los puntos de los que dispone el usuario consultando la base de datos, y se indican los puntos a canjear, llamando al método de utils *restarPuntos()* para realizar la operación.
5. **Resetear huevos gallinas:** se confirma tras seleccionar esta opción que se quieren resetear todos los huevos de las gallinas, modificando el atributo *total\_huevos* de todas las gallinas existentes en la base de datos.

Todas estas funcionalidades se desarrollaron al hacer clic en un *ImageButton* para que aparezca un *AlertDialog* con los campos correspondientes para su guardado en Firebase Realtime Database y/o Firebase Storage posterior.

Tras el desarrollo de estas funcionalidades, se procede a implementar la lógica que impide que un usuario alumno pueda acceder a esa pantalla. La verificación se realiza comprobando el dominio del correo del usuario logueado. Solo si el usuario tiene un correo válido (no genérico tipo @eggstat.com), se le permite ver y pulsar el botón de ajustes para acceder al *Fragment* de *Control*.

Otra funcionalidad esencial era la **revisión de estancias**. En esta pantalla se muestran las tres zonas principales del gallinero (bebadero, comedero y gallinero), junto con su estado, la última fecha de revisión y el nombre del usuario que la revisó. Cada imagen es pulsable, y al hacerlo se muestra un *Dialog* de confirmación. Si el usuario acepta, se actualizan los datos en Firebase, se registra el usuario y la nueva fecha de revisión, y se otorgan puntos al usuario. Se implementaron además funciones que calculan el estado actual de la estancia en base al tiempo transcurrido desde la última revisión (*calcularEstadoEstancia()*), teniendo en cuenta el uso del formato *Timestamp*. Al no poder añadir un dato tipo *Date* a Firebase, se tuvo que optar por utilizar este tipo de formato de fecha, y convertirlo posteriormente desde un método *convertirTimestamp()* a una fecha legible por un usuario. Nos ayudamos de la web *EpochConverter* para realizar pruebas de conversión y utilizar una fecha base.

Finalmente, se procede a la implementación de la pantalla que daba nombre a la aplicación, la **pantalla de estadísticas** (llamada “*EggStadísticas*” en alguno de los *AlertDialog* para añadir un toque divertido). En este apartado se integró la **librería MPAndroidChart**, una herramienta especializada en la creación de gráficos en Android. Tras estudiar la documentación oficial, se comprendió que esta librería permitía implementar de forma relativamente sencilla gráficos circulares y de barras con muchas posibilidades de personalización.

La primera implementación consistió en un gráfico de barras (*BarChart*) que muestra los puntos acumulados por cada equipo. Cada barra representa un equipo distinto y se construye dinámicamente a partir de los datos *nombre\_equipo* y *puntos\_equipo* del nodo equipo de Firebase . El segundo gráfico es un gráfico circular (*PieChart*) que representa la distribución total de huevos puestos por todas las gallinas.

Uno de los aspectos más útiles de esta librería es su capacidad para sincronizarse con listas de datos sin necesidad de implementar lógica de dibujo personalizada, permitiendo simplemente cargar los datos en objetos *BarEntry* o *PieEntry* y personalizar el estilo a través de métodos de configuración. Para la actualización de estos gráficos, se implementaron dos métodos: *actualizarBarChart()* y *actualizarPieChart()*, en los que se tuvo que ir probando diferentes configuraciones hasta dar con la que se adaptaba mejor a la pantalla del dispositivo, como mostrar los nombres de los equipos inclinados con el atributo *labelRotationAngle*, o modificar el tamaño de los datos con *setValueTextSize*. Además, para el parámetro de los colores en estos métodos, se implementa un método en *Utils obtenerColoresPersonalizados()* en el que se crea una lista de los colores personalizados de la aplicación (implementado más adelante).

#### 7.4.1. Funcionalidad extra: Firebase Storage y Glide

Con el objetivo de que la aplicación tuviera un enfoque divertido, visual y adaptado a niños, se consideró “necesidad” de incluir imágenes animadas que acompañaran a ciertas acciones dentro de la app. Esto no solo aportaba un toque más divertido, sino que también ayudaba a reforzar la gamificación que debía caracterizar a *EggStat*. Para conseguirlo, se integró el uso de **Firebase Storage** como sistema de almacenamiento en la nube, y la librería **Glide** como herramienta para cargar imágenes desde URLs o desde recursos locales.

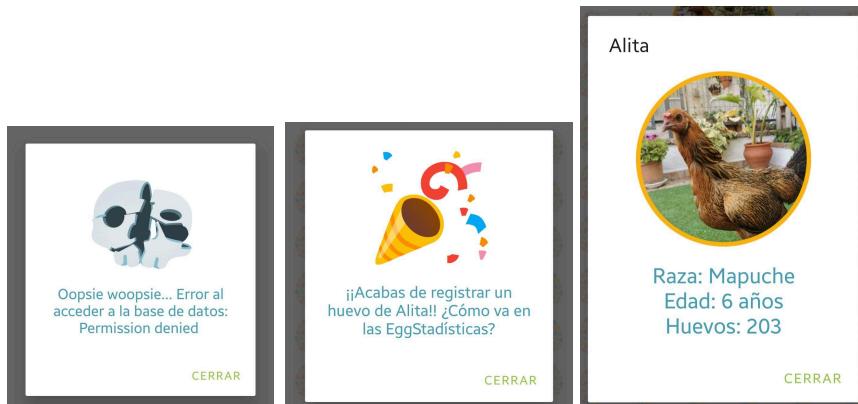
En una primera etapa se investigaron colecciones de imágenes animadas que fueran gratuitas y libres de derechos, y se optó por utilizar algunos *GIFs* de la colección **Google Noto Animated**, que son imágenes diseñadas por Google y publicadas como parte de su iniciativa de emojis accesibles. Estas imágenes están catalogadas como libres de uso no comercial y se encuentran alojadas públicamente por Google, por lo que su utilización en un proyecto educativo como este es apropiada.

Las imágenes se subieron directamente a *Firebase Storage*, generando una URL accesible que luego podía ser utilizada desde cualquier fragmento de la app. A partir de aquí, fue necesario aprender a utilizar la librería *Glide*, ya que Android no admite la carga de *GIFs*.

animados directamente desde *ImageView*. Glide resultó ser una herramienta muy potente y sencilla de utilizar, permitiendo cargar GIFs con una sola línea de código (*Glide.with(contexto).load(imagen).into(contenedor)*) y facilitando el uso de animaciones fluidas en diversas partes de la interfaz.

Inicialmente se utilizó *Glide* para mostrar una imagen animada del equipo del usuario en la pantalla Home. Por ejemplo, si el usuario pertenecía al equipo “*Pulpos*”, aparecía un pulpo animado saludando. Más adelante, también se implementó *Glide* para mostrar las imágenes de las gallinas, que se cargan desde *Firebase* a través de su URL correspondiente.

Otro uso importante fue en los *AlertDialogs* personalizados, donde se mostraban GIFs tanto en mensajes de confirmación divertida (por ejemplo, al registrar un huevo o revisar una estancia), como en los mensajes de error (cuando fallaba el login o había algún error en la base de datos).



Capturas de la pantalla de imágenes cargadas con *Glide* desde *Firebase Storage*. Elaboración propia.

Finalizo esta sección de implementación de funcionalidades indicando que fue necesario realizar una serie de modificaciones en los **archivos de configuración del proyecto** para poder integrar correctamente las librerías y herramientas utilizadas. Uno de los primeros ajustes se hizo en el archivo *libs.versions.toml*, donde se definieron las versiones utilizadas de *Kotlin*, *Firebase*, *Glide* y *MPAndroidChart*. En el caso de *Firebase*, estas versiones se añadieron automáticamente al seguir el asistente de configuración oficial de Android Studio. Para *Glide* y *MPAndroidChart*, se añadieron manualmente sus respectivas líneas bajo las secciones *[versions]* y *[libraries]*, de modo que luego pudieran ser invocadas fácilmente desde el archivo *build.gradle.kts*.

En el archivo *build.gradle.kts* del módulo principal también se realizaron varios cambios. Se actualizó la versión de *Kotlin* para asegurar la compatibilidad con las últimas librerías utilizadas. Se añadieron las dependencias correspondientes a *Firebase (Authentication,*

*Realtime Database y Storage), a Glide para la carga de imágenes dinámicas, y a MPAndroidChart para permitir el uso de gráficos estadísticos.*

## 7.5. Mejoras de la interfaz gráfica

Tras el desarrollo de las funcionalidades principales de la aplicación, se comenzó a trabajar en aspectos relacionados con los colores, los márgenes, los iconos, los fondos y la personalización general de la experiencia de usuario.

El primer paso fue diseñar un **logo para la aplicación**, con la idea de que reflejara la temática central de *EggStat*, en la que los huevos fueran los protagonistas. Para ello, utilicé la herramienta de generación de imágenes con IA Dall-E, tras varios intentos fallidos de realizar de forma autónoma este archivo tan importante. El logo generado fue la imagen de un huevo relleno de pequeños huevos de colores. Este logo no solo se convirtió en el ícono principal de la app, sino que además sirvió como inspiración para la paleta de colores. A partir del logo extraje los tonos principales y los definí en el archivo **colors.xml** como colores personalizados: rojo (#f05240), naranja (#f46d26), amarillo (#ffbb30f), verde (#9bc54c), azul-verde claro (#6dc4a9) y azul verdoso (#489eb3). Estos colores se convirtieron en el eje visual de toda la aplicación.

A continuación, se configuró el nuevo logo como ícono oficial de la app en Android Studio, actualizando los archivos de *res/mipmap/ic\_control*, *ic\_launcher*, *ic\_launcher\_foreground* e *ic\_launcher\_round*. Además, se utilizó este logo como imagen dentro de la pantalla *Huevos*, actuando así como botón clicable para registrar un huevo. Para darle más dinamismo, se decide utilizar la librería de Android **ObjectAnimator**, para hacer que la imagen simule un efecto “latido” en bucle, implementando el siguiente bloque de código, tras realizar varias pruebas con diferentes efectos proporcionados por esta librería, en *HuevosFragment*:

```
// animación efecto latido (solo en el eje Y)
val animacionEjeY = ObjectAnimator.ofFloat(imgHuevo, "scaleY", ...values: 1f, 1.1f, 1f).apply {
    duration = 3000
    repeatCount = ObjectAnimator.INFINITE
    repeatMode = ObjectAnimator.REVERSE
    interpolator = AccelerateDecelerateInterpolator()
}
animacionEjeY.start()
```

Captura de pantalla de la utilización de ObjectAnimator. Elaboración propia.

A partir de ahí se comenzó a refinar todos los layouts: se ajustaron márgenes, alineaciones (*gravity*), paleta de colores, y se mejoraron los elementos visuales para lograr una presentación más cuidada y homogénea.

Una de las mejoras más destacadas fue la **implementación de cambio de color dinámico en el estado de las estancias**. Según el tiempo transcurrido desde la última revisión, el color del estado cambia automáticamente: verde si ha sido revisada recientemente, naranja si está en un punto medio, y rojo si requiere atención urgente. Esta función se lleva a cabo con el método implementado en *EstanciasFragment obtenerColorDesdeEstado()*, dependiendo del valor obtenido por *calcularEstadoEstancia()*. Esto no solo hace la app más intuitiva, sino que también refuerza el componente visual y de urgencia.

Además, se revisaron y mejoraron todos los **mensajes de error o confirmación**, sustituyendo los clásicos *Toast* por *AlertDialogs* personalizados, con textos más simpáticos y amigables, y acompañados de imágenes animadas obtenidas desde *Firebase Storage* y cargadas con *Glide*. También se creó, usando la herramienta *Canva*, un **fondo de pantalla** con el logo de *EggStat* en formato mosaico, al que se le aplicó transparencia para no entorpecer la lectura, pero sí mantener una identidad visual reconocible en todas las pantallas.

Finalmente, se diseñaron algunos de los **iconos del BottomNavigationView** y las imágenes finales de las estancias y botones de *Control* utilizando de nuevo la herramienta *Canva*. En el caso de las imágenes de los botones de Control, se utilizan Emojis de la biblioteca de *Google Color Noto* con alguna modificación, como un símbolo “+”. En el caso de los iconos, fue necesario un proceso adicional para pasarlo a formato SVG y después convertirlos a XML para poder integrarlos correctamente en Android Studio, lo cual resultó más complejo de lo esperado, al ser la primera vez que había modificado el código de un documento SVG. Con todos estos ajustes, la aplicación al fin cumple con el objetivo principal de ser amigable para niños, e intuitiva para todo usuario que la utilice.

En el apartado [Comparativa versiones interfaz](#) de los anexos se puede visualizar la diferencia entre una primera versión de aplicación, y la versión posterior a aplicar todas las mejoras visuales de la aplicación.

## 7.6. Mejoras generales y puesta a punto

En esta fase se procede a realizar pequeñas mejoras generales, como la limpieza de usuarios y equipos de la **base de datos**, comprobación de la cantidad de equipos que se pueden visualizar en los gráficos, además de la **limpieza de código**, realización de comentarios en todas las clases, simplificación de código en los casos en los que fuera necesario, y repaso general de todos los ficheros de la aplicación.

Finalmente, la aplicación cuenta con un total de diecisiete clases Kotlin, de las cuales: cuatro entidades de tipo *dataclass* (Equipo, Estancia, Gallina y Usuario), dos clases que heredan de *AppCompatActivity* (LoginActivity y MainActivity), una clase *object* Utils, seis clases que heredan de *Fragment* (ControlFragment, EstadisticasFragment, EstanciasFragment, GallinasFragment, HomeFragment, HuevosFragment) y cuatro que heredan de *ViewModel* (EstadisticasViewModel, EstanciasViewModel, GallinasViewModel y HomeViewModel).

Respecto a los archivos de layout (*res/layout*), contamos con un total de veinte archivos XML encargados de mostrar diferentes vistas en la pantalla del dispositivo. *activity\_main.xml* y *activity\_login.xml* mostrarán las vistas de MainActivity y LoginActivity respectivamente. Todos los fragments tienen su propio layout (*fragment\_home.xml*, *control*, *gallinas*, *huevos*, *estadísticas* y *estancias*). Además, se cuenta con un total de doce layouts para mostrar diálogos informativos, como *dialog\_registro\_estancia.xml* que muestra el aviso de confirmación de que vas a revisar una estancia, el *dialog\_registrar\_huevo.xml*, que muestra un spinner donde elegir la gallina que puso el huevo, etc. En este punto se es consciente de que quizás algunos de ellos podrían reutilizarse, pero llegados a este punto no queda otra que dejarlo para vías futuras.

Otros archivos con los que cuenta la aplicación final son las imágenes de *res/drawable*, como son las de las estancias, imagen del logo o las de los botones de *Control*, y los iconos como *ico\_gallina.xml* o *ico\_estancias.xml*, que son los iconos de la barra inferior de navegación. Respecto a esta barra, en *res/menu* se encuentran *menu\_home.xml*, responsable de la vista de la parte superior de *Home*, y *bottom\_nav\_menu.xml*, responsable de la vista de la barra inferior de navegación.

Cabe destacar uno de los archivos más importantes de la app, en *res/navigation*, donde se encuentra el archivo *mobile\_navigation.xml* y donde se declaran todos los fragments por los que debe navegar la barra inferior de navegación, además de la vista de *Control*.

## 8. Despliegue y pruebas

Durante todo el desarrollo de la aplicación, se han ido realizando pruebas constantes y progresivas para verificar que cada funcionalidad implementada se comportaba como se esperaba. Estas pruebas se realizaron tanto en el emulador de Android Studio como en un dispositivo físico real, para asegurar que realmente la aplicación funcionaba en un caso real.

La mayoría de las pruebas realizadas fueron **pruebas funcionales de caja negra**, ya que no se evalúa el código fuente directamente, sino el comportamiento visible de la app desde el punto de vista del usuario. A lo largo de la implementación de cada funcionalidad (como el login, la creación de usuarios, el registro de huevos o las revisiones de estancias), se comprobó que el flujo fuera correcto, que los datos se guardaran adecuadamente en *Firebase*, que los *Spinners* se llenaran con la información esperada, y que los diálogos se mostraran correctamente al interactuar con los diferentes botones o elementos visuales.

También surgieron casos en los que fue necesario realizar **pruebas de caja blanca**. Estas pruebas se centraron en analizar directamente el comportamiento del código, y resultaron especialmente útiles en situaciones donde las funcionalidades no fallaban de forma evidente, pero sí había comportamientos inesperados. Un ejemplo destacado fue la prueba relacionada con la carga dinámica de imágenes mediante *Glide* al seleccionar equipos desde el *Spinner* en el diálogo de creación de equipos. A través del uso de *Toasts* de depuración y el análisis en *Logcat*, se pudo detectar que ciertas funciones aparentemente estaban siendo llamadas (como los botones de canjear puntos o añadir gallinas), pero no se ejecutaban correctamente cuando no se había abierto previamente el diálogo que contenía carga de imágenes. Finalmente, se concluyó que este comportamiento se debía a una sobrecarga de recursos o conflictos con *Glide*, y se optó por eliminar la funcionalidad en este diálogo para mantener la estabilidad del resto de la aplicación.

También se prestó especial atención a la **gestión de errores**, comprobando que si faltaban datos, si ocurría un problema de conexión o si había un fallo en la lectura de *Firebase*, el sistema reaccionaba adecuadamente mostrando mensajes informativos mediante *AlertDialogs* personalizados, lo cual ayudó a mantener una experiencia de usuario amigable incluso en situaciones inesperadas.

A continuación, se muestran algunos ejemplos de casos de uso y las pruebas realizadas sobre ellos:

<b>Caso de uso:</b>	Iniciar sesión
<b>Objetivo probado:</b>	Verificar que un usuario puede acceder con datos correctos
<b>Requisitos probados:</b>	<ul style="list-style-type: none"> <li>- El sistema solicita usuario y contraseña</li> <li>- Se comprueban credenciales con Firebase Auth</li> </ul>
<b>Pruebas realizadas</b>	
<ol style="list-style-type: none"> <li>1. <b>Introducir usuario y contraseña válidos:</b> el usuario puede acceder con normalidad a la aplicación.</li> <li>2. <b>Seleccionar iniciar sesión sin introducir todos los campos:</b> se muestra un aviso y no se permite el acceso.</li> <li>3. <b>Introducir un usuario que no existe:</b> se muestra un aviso con el mensaje de error y no se permite el acceso.</li> <li>4. <b>Introducir una contraseña errónea:</b> se muestra un aviso con el mensaje de error y no se permite el acceso.</li> </ol>	

<b>Caso de uso:</b>	Acceso a Control
<b>Objetivo probado:</b>	Verificar que un alumno no puede visualizar el botón situado en la parte superior de la pantalla <i>Home</i>
<b>Requisitos probados:</b>	<ul style="list-style-type: none"> <li>- Se comprueban las credenciales con Firebase Auth para identificar si el rol es “profesor”</li> </ul>
<b>Pruebas realizadas</b>	
<ol style="list-style-type: none"> <li>1. <b>Se accede con un usuario profesor:</b> se verifica que el botón se puede visualizar y seleccionar para acceder a la pantalla Control.</li> <li>2. <b>Se accede con un usuario alumno:</b> se verifica que el botón no se puede visualizar y no se puede acceder por tanto a la pantalla Control.</li> </ol>	

<b>Caso de uso:</b>	Revisar una estancia
<b>Objetivo probado:</b>	Comprobar que se actualiza la fecha y el estado con los datos

	correctos.
<b>Requisitos probados:</b>	<ul style="list-style-type: none"> <li>- Se actualiza el timestamp y nombre de usuario</li> <li>- Se calcula nuevo estado</li> <li>- Se suman puntos al usuario y equipo</li> </ul>
<b>Pruebas realizadas</b>	
<ol style="list-style-type: none"> <li>1. <b>Pulsar imagen de estancia y confirmar:</b> se actualizan los datos en Firebase y cambia el color del estado, la fecha y el último usuario.</li> <li>2. <b>Cancelar el diálogo:</b> no se realizan cambios.</li> </ol>	

<b>Caso de uso:</b>	Canjear puntos usuario o equipo
<b>Objetivo probado:</b>	Verificar que se pueden cambiar los puntos actuales mediante una resta, y que estos no pueden ser superiores a los que tiene.
<b>Requisitos probados:</b>	<ul style="list-style-type: none"> <li>- Se carga un diálogo donde indicar todos los datos</li> <li>- Se muestran en un <i>TextView</i> los datos del equipo seleccionado en el <i>Spinner</i> del diálogo</li> </ul>
<b>Pruebas realizadas</b>	
<ol style="list-style-type: none"> <li>1. <b>Seleccionar canjear sin proporcionar datos:</b> se cierra el diálogo y se muestra un aviso. No se actualizan los datos.</li> <li>2. <b>Introducir más puntos de los que tiene el usuario o equipo:</b> se cierra el diálogo y se muestra un aviso. No se actualizan los datos.</li> <li>3. <b>Introducir una cantidad de puntos menor a la disponible:</b> se actualizan los puntos y se muestra un aviso confirmándolo.</li> </ol>	

<b>Caso de uso:</b>	Añadir una gallina
<b>Objetivo probado:</b>	Verificar que un objeto gallina se añade a la base de datos con todos los datos correctos, incluida la imagen
<b>Requisitos probados:</b>	<ul style="list-style-type: none"> <li>- Se carga un diálogo emergente solicitando los datos</li> </ul>

	<ul style="list-style-type: none"> <li>- Se proporciona un <i>DatePicker</i> para elegir fecha de nacimiento</li> <li>- Se proporciona un botón de selección de imagen de la galería del dispositivo</li> </ul>
<b>Pruebas realizadas</b>	
<ol style="list-style-type: none"> <li>1. <b>Seleccionar crear sin proporcionar datos:</b> se cierra el diálogo y se muestra un aviso. No se actualizan los datos en la base de datos.</li> <li>2. <b>Buscar imagen de gallina y volver atrás:</b> aparece un aviso y no se actualiza la base de datos.</li> <li>3. <b>Buscar imagen de gallina y seleccionarla:</b> aparece un aviso de que se seleccionó la gallina y el sistema queda en espera de confirmación.</li> <li>4. <b>Introducir todos los datos y la imagen correctamente:</b> aparece un <i>ProgressBar</i> que indica que la imagen se está subiendo. Al finalizar, se muestra un aviso. Se actualiza la base de datos.</li> </ol>	

<b>Caso de uso:</b>	Crear cuenta profesor
<b>Objetivo probado:</b>	Verificar que un usuario puede crear una cuenta
<b>Requisitos probados:</b>	<ul style="list-style-type: none"> <li>- Se solicitan los datos necesarios para la creación de un nuevo usuario</li> <li>- El sistema actualiza la base de datos correctamente</li> </ul>
<b>Pruebas realizadas</b>	
<ol style="list-style-type: none"> <li>1. <b>Seleccionar botón <i>Crear cuenta</i>:</b> aparece un diálogo con los datos necesarios para crear la cuenta.</li> <li>2. <b>Seleccionar <i>crear sin proporcionar todos los datos</i>:</b> aparece un diálogo y no se actualiza la base de datos.</li> <li>3. <b>Seleccionar <i>crear sin proporcionar la clave secreta correcta</i>:</b> aparece un diálogo informativo con una imagen malvada y un mensaje divertido. No se actualiza la base de datos.</li> <li>4. <b>Seleccionar <i>crear sin proporcionar un correo electrónico</i>:</b> aparece un diálogo informativo con el mensaje de error. No se actualiza la base de datos.</li> </ol>	

5. **Seleccionar crear con todos los datos correctos:** aparece un diálogo informando de que se ha añadido un profesor con éxito. Se actualiza la base de datos.

<b>Caso de uso:</b>	Recuperar contraseña
<b>Objetivo probado:</b>	Verificar que si se pierde la contraseña se envía un mensaje de recuperación al correo electrónico
<b>Requisitos probados:</b>	<ul style="list-style-type: none"> <li>- El sistema pregunta si el usuario es alumno o profesor</li> <li>- Si es profesor, el sistema pide un correo electrónico</li> </ul>
<b>Pruebas realizadas</b>	
<ol style="list-style-type: none"> <li>1. <b>Seleccionar “soy alumno”:</b> se muestra un diálogo informando al alumno que debe pedirle su contraseña a su profesor.</li> <li>2. <b>Seleccionar “soy profesor” e introducir un texto que no es un email:</b> aparece un diálogo informando del error. No se envía ningún correo electrónico de recuperación.</li> <li>3. <b>Seleccionar “soy profesor” e introducir un correo electrónico no registrado en la base de datos:</b> aparece un mensaje de que se ha enviado un mensaje a esa dirección de correo si está registrada, y que si no, es porque no lo está. No se envía correo electrónico a esa cuenta.</li> <li>4. <b>Seleccionar “soy profesor” e introducir un correo electrónico registrado en la base de datos:</b> aparece un mensaje de que se ha enviado un mensaje a esa dirección de correo si está registrada, y que si no, es porque no lo está. Se envía un correo electrónico a la dirección de correo con un enlace de recuperación. El usuario debe verificar su bandeja de entrada y cambiar la contraseña.</li> </ol>	

## 9. Conclusiones

El desarrollo de **EggStat** ha sido una experiencia enriquecedora tanto a nivel técnico como personal. A lo largo de las diferentes fases del proyecto, se han alcanzado los objetivos propuestos, y se ha logrado crear una aplicación funcional, educativa y visualmente atractiva que permite a niños y niñas registrar sus acciones en el cuidado de un gallinero escolar de manera gamificada.

Desde la elección de herramientas como Firebase y Android Studio, hasta la adopción de la arquitectura *MVVM*, cada decisión se ha tomado con el objetivo de garantizar la escalabilidad, claridad en el código y facilidad de mantenimiento. Estas elecciones han demostrado ser adecuadas para el tipo de aplicación planteada, permitiendo una interacción fluida con la base de datos en tiempo real y diferenciando claramente los roles de los usuarios, una funcionalidad clave para su contexto educativo.

Durante el proceso de desarrollo, se han enfrentado numerosos retos, como la implementación de *Firebase Authentication* en un entorno donde los usuarios no pueden disponer de correo electrónico, la sincronización correcta de datos y la personalización visual para adaptarse al público infantil. Sin embargo, cada uno de estos obstáculos ha supuesto una oportunidad para aprender y mejorar, contribuyendo directamente a la evolución y madurez del proyecto.

Asimismo, se ha trabajado con especial dedicación en la interfaz gráfica, cuidando los colores, las imágenes animadas y los detalles visuales para conseguir una experiencia atractiva, accesible y coherente con los valores de la aplicación: educación, sostenibilidad, cuidado animal y trabajo en equipo.

*EggStat* no solo cumple su función como herramienta educativa, sino que también refleja un proyecto con una identidad propia, motivado por una experiencia real y una pasión personal. Esto le otorga un valor añadido que va más allá de lo técnico: es una aplicación pensada desde la práctica docente, con cariño por los animales y con ganas de hacer del aula un espacio donde aprender también puede ser divertido.

Sin duda, *EggStat* ha sido un proyecto de gran valor formativo, y deja la puerta abierta a futuras mejoras y ampliaciones. Su base sólida y su enfoque centrado en el usuario la convierten en una herramienta con mucho potencial para seguir creciendo y adaptándose a nuevos contextos educativos.

## 10. Vías futuras

Aunque EggStat ha logrado cumplir los objetivos inicialmente planteados, el potencial del proyecto abre muchas posibilidades de evolución y mejora que podrían implementarse en futuras versiones de la aplicación.

Una de las funcionalidades que se plantea incorporar es la posibilidad de que el usuario con rol de profesor pueda eliminar gallinas, equipos o usuarios alumnos. Actualmente, estos datos pueden añadirse desde la app, pero no eliminarse, lo que puede dificultar la gestión cuando se desea hacer limpieza de datos antiguos o reiniciar el sistema para un nuevo curso escolar.

Además, se plantea una línea de desarrollo especialmente ambiciosa e interesante, pero que mi experiencia con estos dispositivos me hace realmente necesitarlos en una aplicación futura: **la integración de sensores IoT en el gallinero real**. Con esta mejora, el estado de las estancias dejaría de calcularse a partir del tiempo transcurrido desde la última revisión, y pasaría a actualizarse en función de datos reales obtenidos a través de sensores (por ejemplo, sensores de nivel del agua para el bebedero/comedero). Esto haría que la app se convirtiera en un sistema de monitorización mucho más preciso y automatizado.

En esa misma línea de innovación, se contempla la posibilidad de implementar un modelo de **inteligencia artificial** que, a partir de imágenes o características de los huevos puestos, **sea capaz de identificar automáticamente qué gallina ha puesto cada huevo**. Esto requeriría un proceso de entrenamiento del sistema con una base de datos de ejemplos reales, pero supondría una revolución tanto a nivel técnico como educativo.

Otro avance deseado sería la inclusión de una pantalla con acceso a una **webcam** en tiempo real, instalada en el propio gallinero escolar, para que los usuarios pudieran observar a sus gallinas en directo. Esto reforzaría el vínculo emocional con los animales y aumentaría el sentido de responsabilidad de los alumnos.

También se considera importante introducir mecanismos de validación por parte del profesor, especialmente para ciertas acciones clave como el registro de una revisión. Esta funcionalidad permitiría asegurar que las acciones realizadas por los alumnos han sido realmente ejecutadas, añadiendo un componente de supervisión muy valioso en contextos educativos reales.

Todas estas mejoras no sólo aportarían nuevas funcionalidades y profundidad al proyecto, sino que también lo convertirían en una herramienta educativa aún más completa, interdisciplinar y conectada con el mundo real.

## 11. Bibliografía/Webgrafía

### Recursos bibliográficos

Ilerna Online. (2023). *Entornos de desarrollo* (Grado Superior DAM/DAW). Maquetado e impreso por Ilerna Online S. L.

Ilerna Online. (2023). *Programación multimedia y dispositivos móviles* (Grado Superior DAM). Maquetado e impreso por Ilerna Online S. L.

### Documentación oficial de Android y Firebase

Android Developers. (2025). *BottomNavigationView*.

<https://developer.android.com/reference/com/google/android/material/bottomnavigation/BottomNavigationView>

Android Developers. (2025). *Dialogs*.

<https://developer.android.com/develop/ui/views/components/dialogs?hl=es-419>

Android Developers. (2025). *RecyclerView*.

<https://developer.android.com/develop/ui/views/layout/recyclerview?hl=es-419>

Android Developers. (2025). *App architecture*.

<https://developer.android.com/topic/architecture/intro?hl=es-419>

Android Developers. (2024). *ViewModel overview*.

<https://developer.android.com/topic/architecture/recommendations?hl=es-419#viewmodel>

Android Developers. (2025). *Themes and styles*.

<https://developer.android.com/develop/ui/views/theming/themes?hl=es-419>

Android Developers. (2025). *LayoutInflater*.

<https://developer.android.com/reference/android/view/LayoutInflator>

Android Developers. (2025). *NavController*.

<https://developer.android.com/reference/androidx/navigation NavController?authuser=1>

Android Developers. (2025). *ActivityResultLauncher*.

<https://developer.android.com/reference/androidx/activity/result/ActivityResultLauncher>

Android Developers. (2025). *Spinner*.

<https://developer.android.com/reference/android/widget/Spinner>

Android Developers. (2025). *DatePicker*.

<https://developer.android.com/reference/android/widget/DatePicker>

Android Developers. (2024). *View Binding*.

<https://developer.android.com/topic/libraries/view-binding?hl=es-419>

Android Developers. (2025). *GridLayout*.

<https://developer.android.com/reference/android/widget/GridLayout?authuser=1>

Firebase. (2025). *Authentication for Android*. <https://firebase.google.com/docs/auth?hl=es>

Firebase. (2025). *Firebase Realtime Database*.

<https://firebase.google.com/docs/database?hl=es>

Firebase. (2025). *Firebase Storage*.

[https://firebase.google.com/docs/storage/?hl=es&authuser=1#implementation\\_path](https://firebase.google.com/docs/storage/?hl=es&authuser=1#implementation_path)

Android Developers. (2024). *Vector Asset Studio*.

<https://developer.android.com/studio/write/vector-asset-studio?hl=es-419>

Android Developers. (2025). *ObjectAnimator*.

<https://developer.android.com/reference/android/animation/ObjectAnimator>

Android Developers. (2025). *Adapter*.

<https://developer.android.com/reference/android/widget/Adapter>

Android Developers. (2024). *Providing resources*.

<https://developer.android.com/guide/topics/resources/providing-resources?hl=es-419>

## Tutoriales y documentación externa

PhilJay. (2021). *MPAAndroidChart library*. <https://github.com/PhilJay/MPAndroidChart>

Programmer Click. (s.f.). *Tutorial MPAndroidChart*.

<https://programmerclick.com/article/41521437531/>

TutsPlus. (2017). *Code an image gallery Android app with Glide*.

<https://code.tutsplus.com/es/code-an-image-gallery-android-app-with-glide--cms-28207t>

Develou. (s.f.). *Crear un GridView en Android*.

<https://www.develou.com/tutorial-para-crear-un-gridview-en-android/>

Desarrollador Android. (2019). *Funciones estándar en Kotlin: let*.

<http://kotlin.desarrollador-android.com/modismos-y-patrones/funciones-estandar/let/>

CursoKotlin.com. (2024). *Curso Kotlin para Android*. <https://cursokotlin.com/>

CursoKotlin.com. (2021). *MVVM en Android con Kotlin, LiveData y View Binding*.

<https://cursokotlin.com/mvvm-en-android-con-kotlin-livedata-y-view-binding-android-architecture-components/>

EpochConverter. (2025). *Convertir timestamp a fecha*. <https://www.epochconverter.com/>

## Contenido audiovisual

Android Coding with Harsha. (2022). *Navigation Bar - Bottom NavigationView en Android [Video]*. YouTube. <https://www.youtube.com/watch?v=ohAMfjC4a0Y>

## Diseño y recursos gráficos

Emojipedia. (2025). *Animated Noto Color Emoji*.

<https://emojipedia.org/animated-noto-color-emoji>

## Información complementaria

Hablemos de Aves. (2020). *Huevos azules: raza Mapuche*.

<https://hablemosdeaves.com/huevos-azules/>

Naciones Unidas. (s. f.). *Objetivos de Desarrollo Sostenible*.

<https://www.un.org/sustainabledevelopment/es/sustainable-development-goals/>

## 12. Anexos

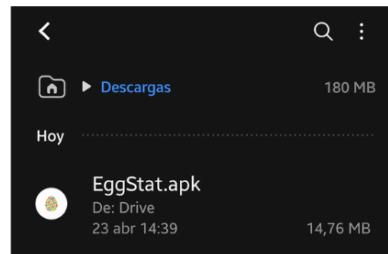
### 12.1. Manual de Instalación



A continuación, se muestran los pasos para instalar la aplicación EggStat. Teniendo en cuenta que aún no se ha subido a ninguna tienda de aplicaciones, **es imprescindible contar con el archivo .apk** para poder disfrutar de la aplicación.

#### 1. DESCARGA EL ARCHIVO .APK

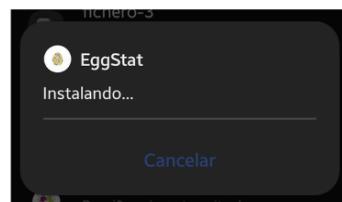
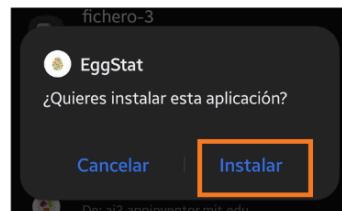
Descarga el archivo **EggStat.apk**, y búscalo en tu dispositivo, generalmente en la carpeta **Descargas**.



#### 2. INSTALA EL ARCHIVO .APK

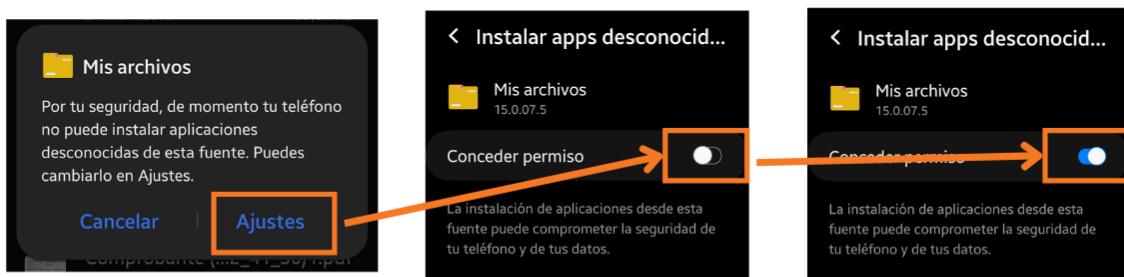
Pulsa sobre el archivo **EggStat.apk**, y confirma su instalación haciendo tap en **Instalar**.

Si todo va bien, la aplicación se instalará y ¡ya podrás disfrutar de EggStat!



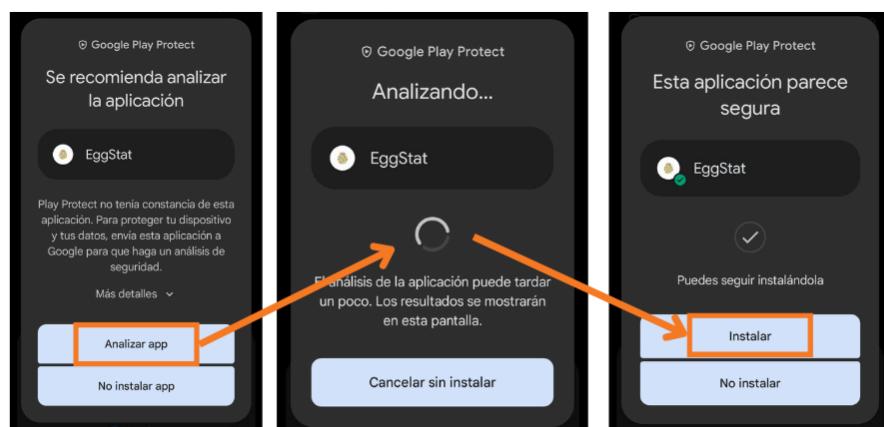
## EXTRA - CONCEDER PERMISOS

Si es la primera vez que se instala una aplicación de terceros, se tendrá que **conceder permisos de instalación**. Para ello, en el diálogo emergente, pulsa sobre **Ajustes**, y luego activa “**Conceder permiso**”.

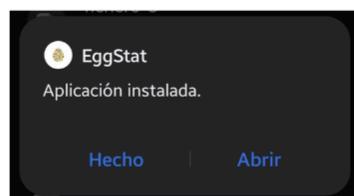


## EXTRA - ANALIZAR LA APLICACIÓN

También es posible que el sistema recomiende analizar la aplicación antes de instalarla. Para ello, pulsa en **Analizar app**, espera que el proceso finalice, y luego pulsa sobre **Instalar**.



Ahora sí... ¡Ya puedes disfrutar de **EggStat**!



## 12.2. Manual de Usuario



# EGGSTAT

## MANUAL DE USUARIO: VERSIÓN PROFE!



¡Atención! ¡Este manual incluye información SÚPER SECRETA de profes! **Solo los profesores pueden tener acceso a esta versión del manual.** Si eres un alumno y estás leyendo esto, ¡rápido, deja de leerlo o una gallina malvada te perseguirá para quitarte la merienda! Las gallinas lo saben todo...

Ahora que estamos seguros de que eres un/a profesor/a... En este manual se mostrará cómo crear una **cuenta de profesor**, y todas las funcionalidades que tendrás al ingresar con dicha cuenta. Las **cuentas de alumno** serán responsabilidad de los profesores, tanto su creación como la transmisión de las credenciales a los alumnos. ¡Vamos a ello!

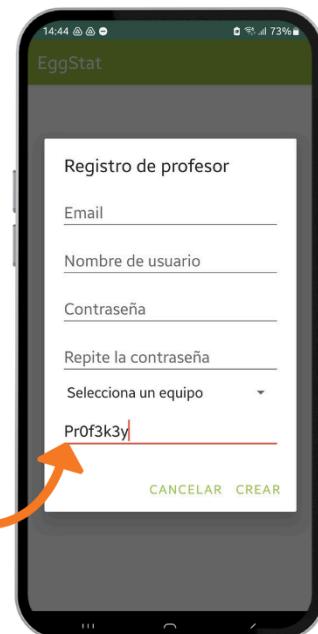
### CREAR UNA CUENTA DE PROFESOR



Haz clic sobre **crear cuenta profesor** para comenzar a crear una cuenta y rellena todos los datos necesarios.

Y aquí viene la información super secreta... la clave para crear la cuenta es “PrOf3k3y”

(¡importante la P mayúscula y las vocales como números cero y tres!)



## ACCEDER COMO PROFESOR



Si se quiere acceder como profesor, se tendrá que ingresar el **correo electrónico** con el que se registró. Los alumnos accederán con las credenciales que el profe cree más adelante.

En el caso de olvidar contraseña, se puede recuperar siguiendo los pasos de **¿Olvidaste tu contraseña?**

## FUNCIONALIDADES DE PROFESOR



En la parte superior de la pantalla principal se encuentra el botón de acceso al **Panel de control**. Desde ahí podrás realizar las siguientes acciones:



Para poner tu app **EggStat** a punto, lo ideal es realizar las acciones en el siguiente orden:

## 1 - AÑADIR GALLINAS



Añade a las protagonistas de la app: **las gallinas**. Indica su nombre, su fecha de nacimiento, su raza, los huevos que crees que lleva puestos y ¡sube una imagen! Recomendaciones:

- Utiliza una imagen cuadrada
- Lleva el conteo de huevos por meses (¿de verdad te acuerdas de cuántos huevos llevan desde que nacieron?)

Al finalizar, ¡la gallina aparecerá en la pantalla **Gallinas**!



## 2 - AÑADIR EQUIPOS



Agrupa a tus alumnos en **equipos**. ¡Es la mejor manera de cuidar de las gallinas! Hay varios equipos pre-configurados con imágenes muy divertidas, hasta un máximo de 8. Selecciona el que quieras añadir a tu app para poder añadir alumnos a estos equipos.

## 3 - AÑADIR ALUMNOS



Una vez que haya equipos disponibles, ¡ya puedes añadir **alumnos**! Tendrás que decidir qué usuario y contraseña tendrán, a qué equipo pertenecerán y ¡ya podrán acceder! Eso sí, tendrás que ser tú como profe el que le **facilite los datos de acceso**.



## + CANJEAR PUNTOS



En los botones **Canjear puntos equipo** y **Canjear puntos usuario** podrás canjear los puntos que deseas. Ofrece a tus alumnos recompensas por cuidar de las estancias o recoger los huevos:

- Que toda la clase salga 5 minutos antes al recreo: ¡50 puntos!
- Que toda la clase elija pista deportiva en el recreo: ¡100 puntos!
- Que un alumno pueda llevarse a casa media docena de los huevos... ¡20 puntos!

También puedes recompensar al equipo que acabe el mes con más puntos... Échale imaginación, adapta el sistema a tu situación real y ¡motiva a tus alumnos a cuidar el gallinero con este sistema de puntos!

## + RESETEAR HUEVOS DE TODAS LAS GALLINAS

Para poder ver una comparación real de los huevos que van poniendo las gallinas, se puede hacer un reseteo de todos los huevos que llevan las gallinas. De esta forma, se puede hacer conteo por mes o por semana, o incluso ¡por estación del año! ¿Sabías que en verano y en invierno es cuando menos huevos ponen las gallinas? ¡Compruébalo con **EggStat**!



# EGGSTAT

¡Gana puntos por cuidar de tus gallinas!





# EGGSTAT

## MANUAL DE USUARIO

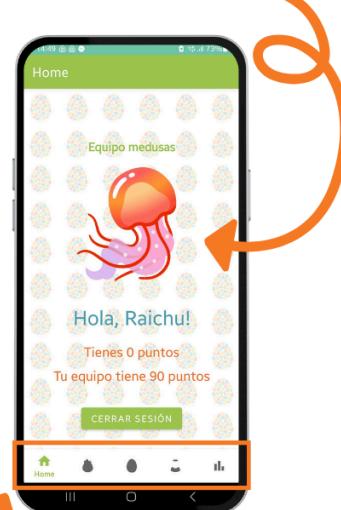
¡Bienvenid@ a **EggStat!** La aplicación que te da puntos por cuidar de tus gallinas. A continuación te mostramos cómo utilizar todas las funcionalidades de la app:

### INICIAR SESIÓN



¡Introduce las credenciales que te ha facilitado tu profe!

Cuando accedas, comenzarás en la pantalla **Home**, donde podrás ver la imagen de tu equipo, y los puntos que tienes disponibles.



En la **parte inferior** podrás ver todas las pantallas por las que podrás navegar si pulsas sobre los iconos.

## GANAR PUNTOS PARA TI Y PARA TU EQUIPO

### REGISTRAR UN HUEVO

Cuando vayas al gallinero y haya un huevo... ¡regístralos! desde la pantalla **Huevos!** Tendrás que conocer bien a las gallinas, porque tienes que decidir a cuál de ellas se lo asignas.



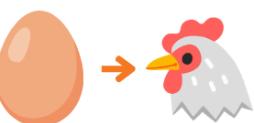
### REVISAR UNA ESTANCIA

¡El bebedero está vacío! Ve al gallinero, rellénalo y ¡registra la revisión en la pantalla **Estancias!** Mantén a las gallinas con agua, comida y ¡con el gallinero bien limpio!



**Suma** todos estos puntos tanto a tu cuenta personal como a tu equipo!

+5



+10



+50



Y ahora... ¡toca negociar con tu profe recompensas para **cambiar puntos**!

## CONOCE A TUS GALLINAS

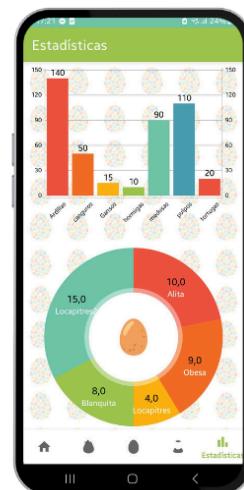


Ve a la pantalla **Gallinas** y visualiza los datos de tus gallinas. ¿Sabías que la raza Mapuche pone huevos azules? ¡Ya sabes identificar los huevos de Alita!



## VISUALIZA LAS "EGGSTADÍSTICAS"

¿Qué gallina lleva más huevos? ¿Qué equipo lleva más puntos? Mantente informado desde la pantalla **Estadísticas** de todos estos datos. ¿Sabes que cuanto mejor cudes a las gallinas pondrán más huevos?



# EGGSTAT

¡Gana puntos por cuidar de tus gallinas!



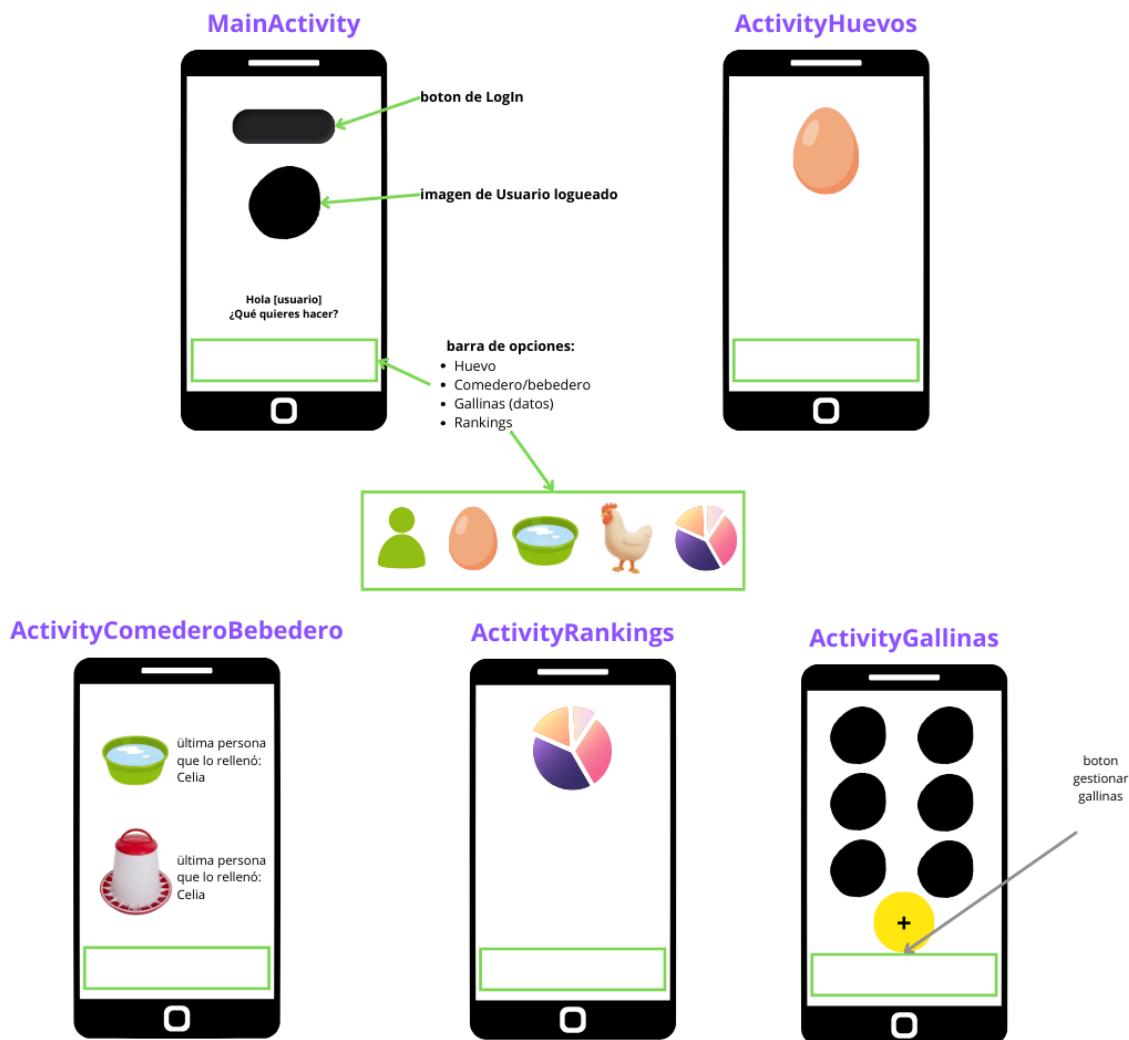
### 12.3. Glosario

Término/Sigla	Descripción
<b>MVVM (Model-View-ViewModel)</b>	Patrón de arquitectura utilizado en el desarrollo de la aplicación, que separa la lógica de negocio (Model), la interfaz de usuario (View) y el controlador intermedio (ViewModel) para mejorar la organización del código.
<b>ViewModel</b>	Componente de MVVM que se encarga de preparar y gestionar los datos para la vista. Permite la comunicación entre la vista (Fragment o Activity) y los datos (Firebase, por ejemplo), y mantiene el estado ante cambios de configuración.
<b>Fragment</b>	Componente modular de Android que representa una parte de la interfaz de usuario dentro de una Activity. Permite reutilizar vistas y gestionar mejor la navegación.
<b>Activity</b>	Unidad principal de una aplicación Android. Representa una pantalla completa y puede contener uno o más Fragments.
<b>Dialog / AlertDialog</b>	Ventana emergente que interrumpe la interfaz actual para mostrar información o pedir confirmación al usuario. Se ha utilizado en múltiples momentos de la app para confirmar acciones o mostrar errores.
<b>Spinner</b>	Elemento gráfico en forma de menú desplegable que permite al usuario seleccionar una opción de una lista. Se usa, por ejemplo, para elegir gallinas o equipos.
<b>Toast</b>	Mensaje breve que se muestra en pantalla para notificar algo al usuario. No interrumpe la interfaz y desaparece automáticamente.
<b>Nodo</b>	Estructura de datos en Firebase Realtime Database. Representa una clave dentro del árbol de datos (por ejemplo: "usuarios", "gallinas").
<b>Grid / GridLayout</b>	Componente visual que permite mostrar elementos en forma

	de cuadrícula, como en el caso de la visualización de gallinas en la app.
<b>Kanban</b>	Metodología de trabajo ágil basada en la visualización de tareas en columnas. Permite gestionar el progreso del proyecto de forma clara y flexible.
<b>Firebase</b>	Plataforma de desarrollo en la nube de Google utilizada en este proyecto para la base de datos en tiempo real, almacenamiento de imágenes y autenticación de usuarios.
<b>Glide</b>	Librería para Android que permite cargar y mostrar imágenes de forma eficiente, incluyendo animaciones y gifs desde URLs o almacenamiento local.
<b>MPAndroidChart</b>	Librería de gráficos utilizada en Android para representar datos visualmente mediante gráficos de barras, circulares, etc.
<b>Firebase Realtime Database</b>	Servicio de base de datos NoSQL que sincroniza datos en tiempo real entre los usuarios conectados. Se ha utilizado para guardar la información de usuarios, gallinas, estancias y más.
<b>Firebase Auth</b>	Módulo de Firebase que permite implementar autenticación de usuarios en la aplicación, diferenciando entre profesores y alumnos.
<b>Firebase Storage</b>	Servicio de Firebase que permite almacenar y acceder a archivos como imágenes en la nube. Se ha usado para guardar las fotos de las gallinas, equipos y recursos animados.

## 12.4. Imágenes

### 12.4.1. Esquema inicial



### 12.4.2. Diagrama de Gantt inicial

## EggStat: Diagrama de Gantt

Desde la semana del 3/3/2025 a la semana del 21/4/2025



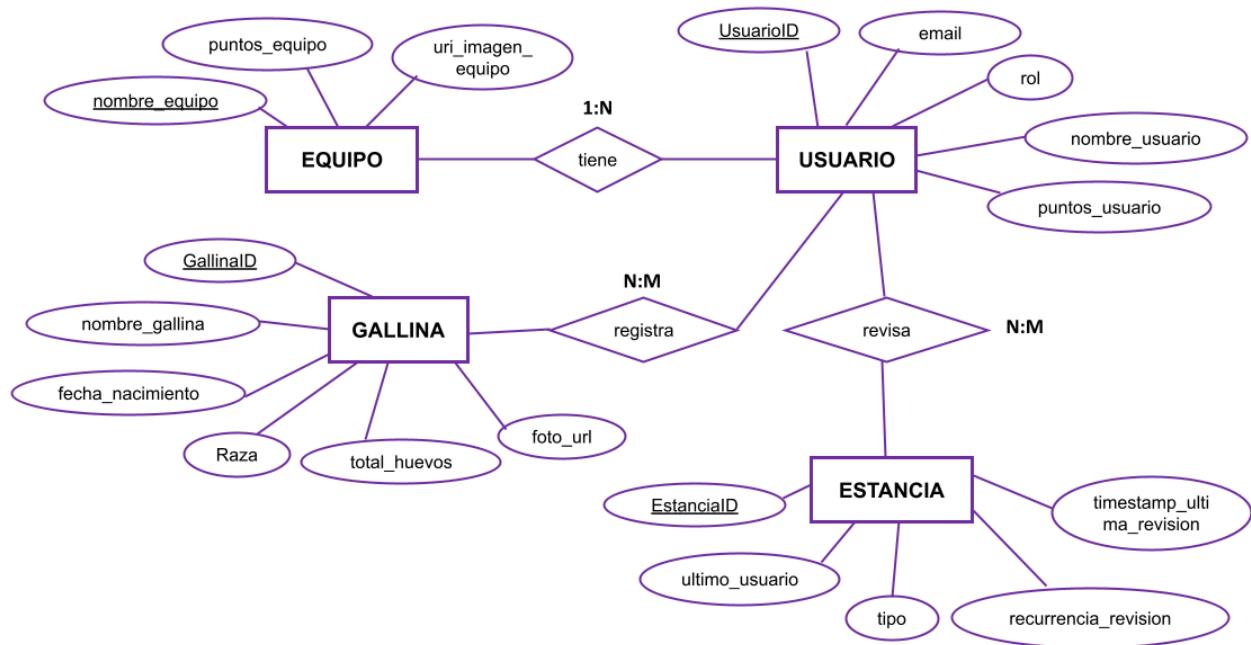
### 12.4.3. Diagrama de Gantt Final

## EggStat: Diagrama de Gantt (final)

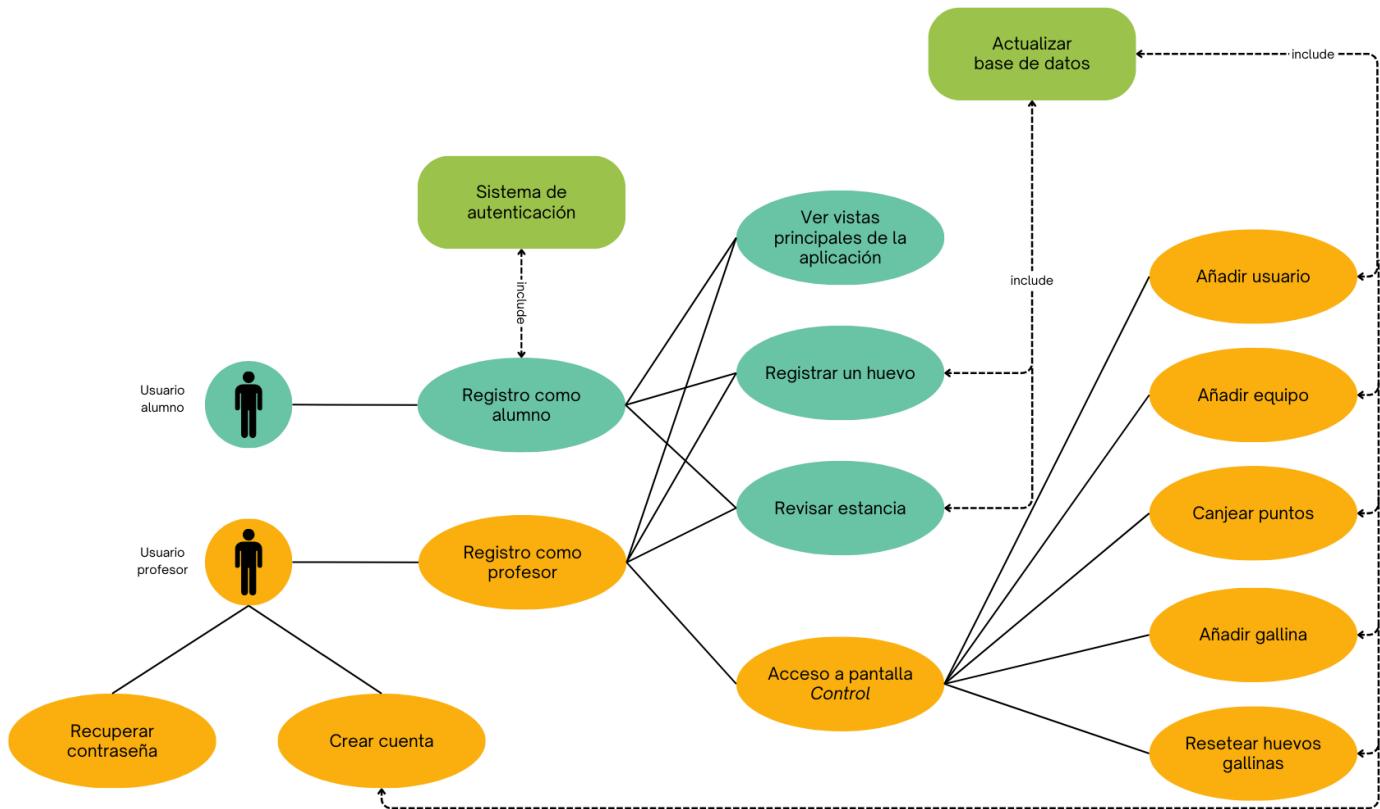
Desde la semana del 3/3/2025 a la semana del 21/4/2025



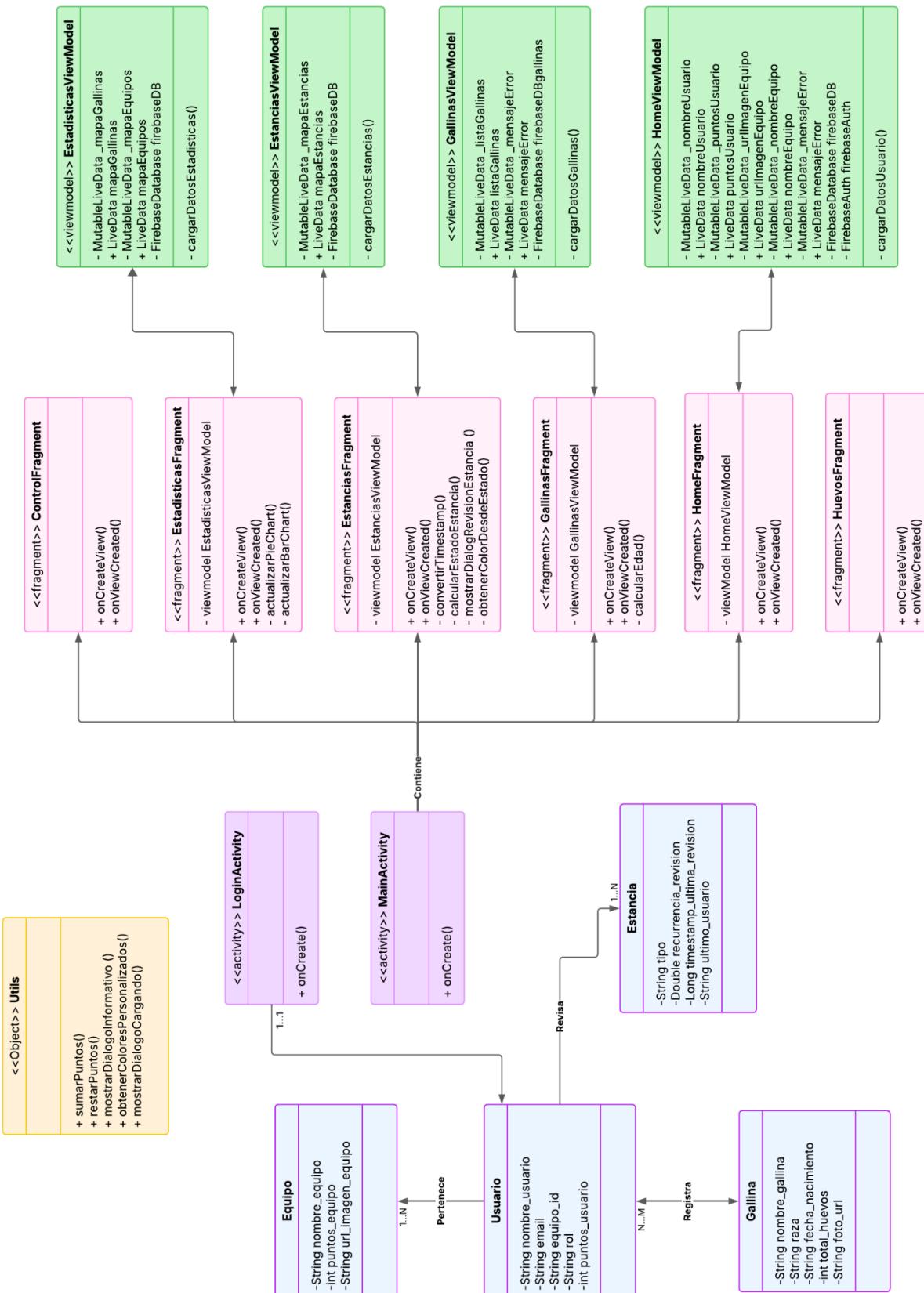
#### 12.4.4. Diagrama entidad-relación



#### 12.4.5. Diagrama de casos de uso



## 12.4.6. Diagrama de clases



#### 12.4.7. Colores EggStat



#489eb3

#6dc4a9

#9bc54c

#fbb30f

#f46d26

#f05240

#### 12.4.8. Las protagonistas: mis gallinas.



## 12.4.9. Comparativa versiones interfaz

### Versión final

Home

Equipo Profesores

Hola, TeacherCelia!

Tienes 205 puntos

Tu equipo tiene 75 puntos

CERRAR SESIÓN

Control

CREAR USUARIO

ASALO EN EL PUEBLO

CANTAR CANCIÓN USUARIO

Gallinas

Locapotes

Alas

Ojos

Bianjita

Haz clic para registrar un huevo

Huevos

Haz clic para registrar un huevo

Estadísticas

Estadísticas

Estadísticas

Estadísticas

Estadísticas

Estadísticas

### Primera versión

Home

Equipo Profesores

Hola, TeacherCelia!

Tienes 205 puntos

Tu equipo tiene 75 puntos

CERRAR SESIÓN

Control

CREAR USUARIO

ASALO EN EL PUEBLO

CANTAR CANCIÓN USUARIO

Gallinas

Locapotes

Alas

Ojos

Bianjita

Haz clic para registrar un huevo

Huevos

Haz clic para registrar un huevo

Estadísticas

Estadísticas

Estadísticas

Estadísticas

Estadísticas

Estadísticas

Celia Pérez Vargas

EggStat

ILERNA  
Online

