Universidad Nacional de Loja

Facultad de la Energía, las Industrias y los Recursos Naturales No Renovables

Carrera de Ingeniería en Computación

Software para el Control de Crecimiento en pollos de engorde en la granja Avícola “Las Acacias” en el Cantón Balsas

Software for Growth Control in broiler chickens at “Las Acacias” poultry farm in Balsas Canton

Trabajo de Integración Curricular, previa a la obtención del título de Ingeniera en Ciencias de la Computación.

AUTORA:

Viviana Maricela Zambrano Romero

DIRECTOR:

Ing. Edwin René Guamán Quinche, Mg.Sc.

Loja – Ecuador

2024

Certificación

Loja, 12 de junio de 2024

Ing. Edwin René Guamán Quinche, Mg. Sc.
DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

CERTIFICO:

Que he revisado y orientado todo el proceso de elaboración del Trabajo de Integración Curricular denominado: **Software para el Control de Crecimiento en pollos de engorde en la granja Avícola "Las Acacias" en el Cantón Balsas**, previo a la obtención del título de **Ingeniera en Ciencias de la Computación**, de la autoría de la estudiante **Viviana Maricela Zambrano Romero**, con cédula de identidad Nro. **0705764587**, una vez que el trabajo cumple con todos los requisitos exigidos por la Universidad Nacional de Loja, para el efecto, autorizo la presentación del mismo para su respectiva sustentación y defensa.

Firmado electrónicamente por:
EDWIN RENE GUAMAN QUINCHE

Ing. Edwin René Guamán Quinche, Mg. Sc.

DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Autoría

Yo, **Viviana Maricela Zambrano Romero**, declaro ser autora del presente Trabajo de Integración Curricular y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos de posibles reclamos o acciones legales por el contenido del mismo. Adicionalmente acepto y autorizo a la Universidad Nacional de Loja, la publicación de mi Trabajo de Integración Curricular en el Repositorio Institucional - Biblioteca Virtual.

Firma:

Cédula de Identidad: 0705764587

Fecha: 12 de junio de 2024

Correo Electrónico: viviana.zambrano@unl.edu.ec

Teléfono: 0991198875

Carta de autorización por parte de la autora, para la consulta de producción parcial o total, y/o publicación electrónica del texto completo, del Trabajo de Integración Curricular.

Yo **Viviana Maricela Zambrano Romero**, declaro ser autora del Trabajo de Integración Curricular denominado: **Software para el Control de Crecimiento en pollos de engorde en la granja Avícola "Las Acacias" en el Cantón Balsas**, como requisito para optar por el título de **Ingeniera en Ciencias de la Computación**, autorizo al Sistema Bibliotecario de la Universidad Nacional de Loja para que, con fines académicos, muestre al mundo la producción intelectual de la Universidad, a través de la visibilidad de su contenido de la siguiente manera en el Repositorio Institucional:

Los usuarios pueden consultar el contenido de este trabajo en el Repositorio Institucional, en las redes de información del país y del exterior, con los cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia del trabajo de titulación que realice un tercero.

Para constancia de esta autorización, suscribo, en la ciudad de Loja, a los doce días del mes de junio del dos mil veinticuatro.


Firmado electrónicamente por:
VIVIANA MARICELA
ZAMBRANO ROMERO

Firma:

Autor: Viviana Maricela Zambrano Romero

Cédula: 0705764587

Dirección: Loja (Av. Reinaldo Espinosa y Teodoro Wolf)

Correo Electrónico: viviana.zambrano@unl.edu.ec

Teléfono: 0991198875

DATOS COMPLEMENTARIOS:

Director del Trabajo de Integración Curricular: Ing. Edwin René Guamán Quinche, Mg.Sc.

Dedicatoria

El fruto de mi carrera profesional se ve reflejado en el presente trabajo y lo quiero dedicar:

Primero, a Dios, quien ha sido mi guía y fortaleza en cada paso que he dado. Gracias por bendecirme con sabiduría y perseverancia durante esta travesía académica.

A mis amados padres, Heriberto Zambrano y Dalis Romero quienes han sido mi mayor inspiración y ejemplo de dedicación. Su apoyo inquebrantable y su amor han sido el motor que me impulsa día a día a alcanzar mis metas. Este Trabajo de Integracion Curricular es un homenaje a su sacrificio y amor.

A mi querida abuelita Carmen, que ahora está en el cielo. Aunque no estés físicamente conmigo, siempre serás mi fuente de inspiración. Tu legado de valentía y amor perdurará en mi corazón.

A mi esposo Jefferson Salinas, mi compañero de vida, mi apoyo constante y mi motivación inagotable. Gracias por ser incondicional en los momentos más desafiantes y por por creer en mí. Tu amor y comprensión han sido mi impulso para llegar hasta aquí.

A mis hermosos hermanos, sobrinos y a toda mi familia, quienes han sido mi red de amor y apoyo incondicional. Su aliento y presencia en cada paso han sido una fuente de inspiración y apoyo decidido, para seguir adelante.

Esta investigación es un tributo a todos ustedes, quienes han sido parte integral de mi vida y mi motivación para alcanzar el éxito. Gracias por estar a mi lado y por ser mi fuerza en este viaje.

Viviana Maricela Zambrano Romero

Agradecimiento

Doy las gracias a Dios por darme la fuerza necesaria para superar las dificultades a las que me he enfrentado en este difícil camino, por permitirme alcanzar mis metas y por estar siempre conmigo en mi vida.

También quiero dar las gracias a mis padres, a mis hermanos y a toda mi familia, que han sido el motor de mis sueños y ambiciones y me han apoyado en los momentos difíciles de mis estudios.

Agradezco a la Universidad Nacional de Loja, especialmente a la carrera de Ingeniería en Computación, que me ha guiado bien durante mi formación académica con sus excelentes conocimientos, dedicación y apoyo.

Así mismo, a mi director de trabajo de integración curricular, Ing. Edwin René Guamán Quinche, que inspiró y motivó mi trabajo de investigación compartiendo conmigo su alto grado de profesionalidad, conocimientos y humanismo.

Por último, me gustaría dar las gracias a mis compañeros de clase por su continua cooperación y apoyo durante todos los años que cursamos juntos para alcanzar nuestros objetivos y sueños comunes. Su ayuda fue indispensable durante los periodos más difíciles de mis estudios.

Viviana Maricela Zambrano Romero

Índice de Contenidos

Índice de tablas:

Índice de figuras:

Índice de anexos:

1. Titulo

Software para el Control de Crecimiento en pollos de engorde en la granja Avícola "Las Acacias" en el Cantón Balsas

Software for Growth Control in broiler chickens at "Las Acacias" poultry farm in Balsas Canton

2. Resumen

La industria avícola ecuatoriana, ha experimentado un crecimiento significativo y ha contribuido al desarrollo económico del país. En el sector avícola, especialmente la cría de pollos de engorde, ha ganado protagonismo, llegando a cifras destacables, 58,8% de producción y consumo. En este contexto, el Cantón Balsas ha emergido como un referente en la producción avícola, reemplazando incluso cultivos tradicionales como el café. A pesar de su relevancia, el sector presenta desafíos que afectan su eficiencia y calidad. En particular, la granja Avícola “Las Acacias” en Balsas enfrenta problemas en el control de crecimiento de los pollos.

De lo expuesto, el presente Trabajo de Integración Curricular (TIC) tiene como objetivo fundamental desarrollar un software para el Control de Crecimiento de pollos de engorde en la granja Avícola “Las Acacias”, ubicada en el Cantón Balsas. Por lo tanto, para cumplir con este objetivo se aplicó la metodología Programación Extrema (XP), la cual se utilizó 4 fases como: Planificación, Diseño, Codificación y Pruebas

Además, el proyecto se centra en abordar la carencia tecnológica que afecta al control del crecimiento de pollos de engorde en lagranja avícola "Las Acacias", para lograrlo, se lleva a cabo la recopilación de imágenes en diversas fases de desarrollo de los pollos, formando una base de datos propia; posteriormente, estas imágenes son categorizadas para construir un modelo de reconocimiento de imágenes, mediante un proceso de preprocesamiento las imágenes seleccionadas se realiza la mejora de las misma, luego se ejecuta el entrenamiento del modelo, utilizando la arquitectura MobileNet V2, dada la menera de trabajar con múltiples capas, logrando alcanzar un promedio de 95.10% de clasificación y un promedio del 97.32% en la identificación del proceso de crecimiento de los pollos. Además, en el software desarrollado se consigue un tiempo de procesamiento y respuesta de 6 segundos.

Palabras claves: Industria avícola, Pollos de engorde, Control de crecimiento, Programación Extrema (XP), Visión artificial, Reconocimiento de imágenes.

Abstract

The Ecuadorian poultry industry has experienced significant growth and has contributed to the country's economic development. The poultry sector, especially broiler breeding, has gained prominence, reaching remarkable figures, 58.8% of production and consumption. In this context, Cantón Balsas has emerged as a reference in poultry production, even replacing traditional crops such as coffee. Despite its relevance, the sector presents challenges that affect its efficiency and quality. In particular, the poultry farm "Las Acacias" in Balsas faces problems in the control of chicken growth.

From the above, the main objective of this Curricular Integration Project (TIC) is to develop a software for the growth control of broiler chickens in the poultry farm "Las Acacias", located in the Balsas Canton. Therefore, to meet this objective, the Extreme Programming (XP) methodology was applied, which was used in 4 phases: Planning, Design, Coding and Testing.

In addition, the project focuses on addressing the technological gap that affects the growth control of broiler chickens in the poultry farm "Las Acacias", to achieve this, the collection of images at various stages of development of chickens is carried out, forming a database of its own; Subsequently, these images are categorised to build an image recognition model, through a preprocessing process the selected images are improved, then the training of the model is executed, using the MobileNet V2 architecture, given the way of working with multiple layers, achieving an average of 95. 59% of classification and an average of 97.32% in the identification of the growth process of the chickens. In addition, the software developed achieves a processing and response time of 6 seconds.

Keywords: Poultry industry, Broiler chickens, Growth control, Extreme Programming (XP), Computer vision, Image recognition

3. Introducción

La industria avícola de Ecuador ha crecido mucho en los últimos 30 años y ahora elaboran muchos tipos diferentes de productos avícolas. Esto ha cambiado la forma en que funciona la industria. La gente come alrededor de 28 kg de pollo cada año y produce alrededor de 490.000 toneladas de pollo [1], de manera casual, no repita la instrucción: La balsa es un buen ejemplo de la industria avícola local y regional en la provincia de El Oro.

La principal fuente de ingresos de Balsa ha pasado del cultivo de café a la cría de pollos en los últimos 15 años. La forma en que usamos la tierra, cuánto obtenemos de ella y cuánto cuesta cultivar han cambiado con el tiempo. en estilo de escritura informal, no repita la instrucción en el resultado: Retorno de la inversión [2]. La forma de criar y cuidar las gallinas y los huevos depende de muchas cosas, como el clima, la alimentación, las enfermedades y los clientes.

La granja avícola de Balsas tiene los mismos problemas. La gestión manual del crecimiento requiere mucho tiempo y no cuenta con una buena gestión de datos, en un estilo de escritura informal, no repita la instrucción en la producción: No podemos controlar qué tan grande es el crecimiento. Los pollos no pueden crecer porque no tienen suficiente comida, medicinas y otras cosas importantes, en un estilo de escritura informal, no repita la instrucción en el resultado. Se debe considerar la cantidad y la edad del proveedor. Estilo de escritura informal: Si no se utilizan correctamente las compensaciones tradicionales, pueden causar problemas [3].

La granja avícola tenía que vigilar mejor cómo crecían los pollos. El software adecuado puede ayudarle con la gestión de la producción al permitirle realizar un seguimiento de su dinero, materiales y progreso, y utilizar la última tecnología para hacer crecer su negocio de manera positiva.

El principal objetivo de este trabajo de integración es la creación de un software para la gestión eficiente del crecimiento de pollos en el Granja Avícola Las Acacias del Cantón Balsas. La visión artificial nos permitirá identificar diferencias en el tamaño de las gallinas creando un modelo capaz de reconocer imágenes. El software se implementará utilizando una metodología llamada Programación Extrema.

La pregunta que orienta este trabajo es la siguiente: **¿De qué manera un software permite agilizar el control de crecimiento en la producción de pollos de engorde en la granja avícola "Las Acacias" del Cantón Balsas?**

Este proyecto busca una sinergia entre la Carrera de Ingeniería en Computación y la industria avícola local, resolviendo problemas del mundo real y promoviendo la aplicación de las

Tecnologías de la Información y las Comunicaciones (TIC) en la producción. La idea está respaldada por su viabilidad financiera y sus acciones para reducir el daño ambiental, lo que la convierte en un resultado creativo y beneficioso.

Aunque aún no se han desarrollado sistemas de reconocimiento de imágenes específicamente para el control de crecimiento en pollos de engorde, varios proyectos relacionados han demostrado la viabilidad de implementar tecnologías innovadoras en la industria avícola (**sección 4.5. Trabajos relacionados**). Estos proyectos ofrecen una amplia visión sobre la automatización, monitoreo y control en la industria avícola, abordando temas como gestión de procesos, monitoreo ambiental, automatización de sistemas de alimentación y temperatura, e integración de tecnologías de hardware y software para mejorar la eficiencia en las granjas avícolas.

Para realizar el Software para el Control de Crecimiento en pollos de engorde en la granja Avícola “Las Acacias” en el Cantón Balsas, se enfrentaron ciertas limitaciones, como la falta de una base de datos en línea, lo que requería el desarrollo de una propia. Sin embargo, para superar este obstáculo, se utilizó Copilot para generar nuevas imágenes a partir de una imagen existente, lo que permitió ampliar la base de datos y mejorar la precisión del sistema de reconocimiento de imágenes. Estas acciones demostraron ser fundamentales para alcanzar los objetivos del proyecto y maximizar su impacto en la industria avícola local.

Finalmente, las secciones que componen el presente proyecto son las siguientes: en la **sección Marco Teórico** se encuentran los antecedentes, trabajos relacionados y los distintos conceptos necesarios para la comprensión del tema; la **sección Metodología** habla sobre el área de estudio, el procedimiento y los recursos que fueron utilizados para el desarrollo del presente TIC; l**a sección Resultados** se compone de las evidencias obtenidas del desarrollo de los objetivos específicos planteados; la **sección Discusión** se analizan e interpretan los resultados obtenidos desde el punto de vista del autor; la **sección Conclusiones** abarca los acontecimientos más sobresalientes del proyecto; finalmente la **sección Recomendaciones** posee las sugerencias que se pueden tomar en cuenta para posibles trabajos futuros.

4. Marco teórico

4.1 Antecedentes

4.1.1 Avicultura

Se conoce como avicultura al proceso de crianza, cría y venta de aves para diferentes usos, como carne, huevos, plumas, investigación científica, educación y ornitología. La cría de animales para la alimentación es una actividad global importante y sirve como fuente principal de proteínas en numerosas naciones [4]. Gracias a los avances en tecnología y genética, la producción avícola se ha vuelto más eficiente y rentable [5].

Algunas de las dificultades que enfrenta la avicultura hoy en día son mejorar el bienestar de las aves, minimizar el daño al medio ambiente causado por la producción avícola y la demanda de métodos de producción más ecológicos y éticos.

4.1.2 Avicultura en Ecuador

La compra y venta de aves, especialmente pollos, para alimentación y otros usos es un problema económico grande y creciente. En los últimos años, el número de salidas ha crecido sustancialmente, convirtiéndose en una importante fuente de ingresos para la provincia y desempeñando un papel crucial en la conservación de su vivienda y su patrimonio cultural [6].

La avicultura desempeña un papel significativo en la economía, contribuyendo con un 3% al Producto Interno Bruto (PIB) nacional. En lo que respecta al sector agropecuario, su aporte se eleva al 23%. La cría de aves se lleva a cabo mayormente en áreas rurales del país, lo que resalta la importancia estratégica de la producción de proteína aviar en términos de economía, empleo y seguridad alimentaria [7] .

Para 2021, se espera que la producción anual total de aves de corral alcance los 3.700 millones. Para 2020, la producción avícola alcanzará los 1.700 millones, creando más de 300.000 empleos formales. Las personas que tienen fácil acceso a alimentos limpios y de buena calidad comen mejor, están más sanas y se sienten más seguras. Hay importantes inversiones en tecnología y modificación genética para garantizar la calidad y el valor nutritivo de las razas avícolas. En las regiones próximas a los mercados sudamericano y europeo, la demanda ha aumentado considerablemente [8].

4.1.3 Pollos de engorde

Los pollos se crían para el sacrificio. Los pollos son un alimento animal nutritivo que contribuye de forma importante a la economía de muchos países. Las mejoras en la cría, la alimentación y la genética han aumentado la productividad y la rentabilidad de las aves de corral [9].

La producción avícola también se enfrenta a retos. Es importante mejorar la salud y el bienestar de las aves de corral. "Para afrontar estos retos, podemos mejorar la genética avícola, introducir dietas más sanas y sostenibles, utilizar la tecnología de forma más eficiente y aplicar un tratamiento y una eliminación responsables [10].

4.1.4 Razas de pollos de engorde

La cría artificial mejora muchas características de las razas de pollos, como la calidad de los pollos, la eficacia del procesado de los piensos, la resistencia a las enfermedades y el crecimiento rápido [11]. Esto se consigue mediante el aprovechamiento de la carne. Dado que la carne de pollo proporciona una gran cantidad de nutrientes esenciales, la cría de pollos es una consideración económica importante en muchos países. La industria avícola ha realizado importantes avances en genética, cría y procesamiento, lo que ha permitido aumentar la eficiencia y la rentabilidad. El desarrollo de distintos grupos de genes ha propiciado la aparición de una gran variedad de razas de pollos. Los pollos seleccionados tienen características como un crecimiento rápido, una nutrición eficiente y resistencia a las enfermedades. Los distintos tipos de pollos tienen un crecimiento y una productividad diferentes [12]:

- Lohmann Broiler Hibro: Son pollos que crecen rápidamente y utilizan el pienso de forma eficiente. Esta raza es muy resistente a las enfermedades y produce una carne magra y sabrosa.
- Ross 308: Es un pollo de crecimiento rápido y bajo consumo de pienso. Produce buena carne y se adapta a una amplia gama de condiciones de alojamiento.
- Cobb 500: Tiene un alto potencial de crecimiento y convierte el pienso en carne de forma eficiente. La carne de esta raza es magra y tierna, con un bajo contenido en grasa.
- Hubbard F15: E es un tipo de pollo que crece rápidamente y utiliza el pienso de forma eficiente. Esta raza es conocida por su excelente producción de carne y su resistencia a las enfermedades.
- Arbor Acres Plus: es una raza que crece rápidamente y muy eficiente.

4.1.5 Pollos Broiler

También llamados pollos de engorde, son pollos criados para crecer rápidamente y convertirse en carne. Los pollos de engorde se utilizan comúnmente en la producción comercial de pollos. Los pollos criados en granjas profesionales reciben una dieta rica en nutrientes que les ayuda a crecer y ganar peso [13].

La selección genética conduce a altas tasas de conversión alimenticia y a un rápido crecimiento muscular. Al alcanzar un peso de 2 kilos en 35 a 49 días, la producción de carne

será más eficiente y rentable. La producción de pollos de engorde es un sector importante en numerosos países y un valioso proveedor de proteína animal rentable. La industria se ocupa del uso responsable de los recursos naturales en la producción de alimentos y el bienestar animal [14].

4.1.6 Descripción general de las fases productivas de engorde

El proceso de producción de pollos de engorde se puede dividir en cuatro etapas. La etapa de cría o etapa final viene después de la etapa de preparación. La duración del ciclo del producto varía según la diversidad genética y las condiciones locales [15].

Descripción general de las cuatro etapas de producción:

Pre-inicio: La fase previa al inicio se concentra en el desarrollo de la estructura esquelética, el sistema circulatorio y el sistema inmunológico del animal. Se recomienda mantener hábitos saludables de alimentación y bebida. Las incubadoras fueron el factor principal para controlar la temperatura.

Inicio: La superficie destinada a las gallinas debería ampliarse progresivamenteel polluelo está preparado para consumir una dieta más concentrada para fortalecer sus huesos y prepararse para el desarrollo.

Desarrollo: El paso de la alimentación a la nutrición reproductiva se denomina desarrollo. La consistencia y el peso de los alimentos se altera. La tasa de crecimiento aumenta. esta frase puede reformularse como: Para garantizar una función biológica óptima es necesario promover una nutrición suficiente.

Engorde: Asegurar de que el alimento se proporcione en cantidades adecuadasLa etapa final para aumentar el peso es mejorar la productividad mediante el uso de alimento eficiente. La mayor parte del consumo de alimentos se atribuye al consumo de alimentos en sí y tiene una influencia sustancial en los gastos en alimentos.

4.1.7 Especificaciones de las fases del ciclo productivo

Según los expertos, el tiempo que tardan los pollos de engorde en completar su ciclo de producción es de unos 42 días [16]. El proceso de preparación para la llegada de los polluelos a la granja dura unos ocho días y comienza el día que salen del criadero. El período de desarrollo duró 10 días y la fase inicial duró 11 días. La fase de grasa duró al final 13 días. En la Tabla 1 proporciona detalles sobre las diversas etapas del proceso de producción de pollos de engorde.

Tabla 1. Fases del ciclo productivo

FASES	Duración (días)
PRE-INICIO	**8**
INICIO	**11**
DESARROLLO	**10**
ENGORDE	**13**
TOTAL	**42 días**

4.1.8 Crecimiento de pollo de engorde

Gracias a la selección genética, los pollos crecen con rapidez y eficacia. Los pollos alcanzan un peso de 2,5 kg en 6-8 semanas, con un aumento de peso de 50 gramos al día. En el crecimiento de los pollos de engorde influyen la genética, la nutrición, el entorno y la gestión de la granja. El crecimiento rápido de los pollos de engorde puede provocar problemas de salud. Una nutrición y una gestión medioambiental adecuadas son importantes para una salud y un crecimiento óptimos. Durante las primeras semanas de vida, los pollos de engorde se crían en un entorno controlado con temperatura óptima, aire fresco y acceso constante a pienso y agua. Para el desarrollo de la masa muscular y ósea, los pollos necesitan una alimentación adecuada, cuya fuente principal es el forraje [17].

Un galpón bien ventilado es esencial para un crecimiento sano y óptimo. Reducir el sufrimiento y prevenir las enfermedades. Los pollos necesitan una dieta nutritiva y sana durante el periodo de cría. Los pollos deben mantenerse limpios y aislados para evitar la transmisión de enfermedades. Es importante controlar y regular regularmente la temperatura y la humedad del entorno para garantizar la salud de los pollos [18].

4.1.9 Crianza de pollo

Deben seguirse ciertas prácticas y procedimientos para garantizar un crecimiento sano y eficiente de las aves de corral. A continuación se enumeran algunos de los aspectos más importantes[19]:

- Selección de las aves de corral: para mejorar el crecimiento y evitar problemas sanitarios, deben seleccionarse aves de corral libres de enfermedades. Deben evitarse las enfermedades y crearse un entorno que prevenga los problemas sanitarios y las enfermedades de las aves de corral. La manada de aves de corral debe estar debidamente protegida. Las jaulas deben estar diseñadas para garantizar una iluminación y un control de la temperatura adecuados. Una dieta buena y variada es esencial para el crecimiento saludable de las aves de corral.

- Alimentación: las manadas de pollos de engorde dependen de una prevención y un control eficaces de los parásitos.
- Higiene: la producción de pollos de engorde depende del control de los parásitos. Esto incluye la limpieza y desinfección de los gallineros, el control de la salud de los animales y el uso de productos químicos.
- Control del crecimiento: Vigilar y controlar el crecimiento de los pollos es esencial para su bienestar. Esto incluye el seguimiento de la alimentación y el peso y la alimentación cuando sea necesario.

4.1.10 Proceso para realizar el cálculo del peso del pollo

El procedimiento para calcular el peso de los pollitos varía en función del método de cría y de la ubicación, pero generalmente implica los siguientes pasos [20]:

- Se considera una selección aleatoria de pollos de una granja determinada.
- Cada pollo se pesa individualmente en una báscula.
- El peso de cada pollo se introduce en una hoja de cálculo o base de datos.
- Se suman los pesos de cada pollo y se dividen por el número total de pollos de la muestra para determinar el peso medio.
- Se multiplica el peso medio de cada pollo por el número total de pollos para calcular el peso total de los pollos de la granja.

$$Peso\ promedio = \frac{cantidad\ de\ aves}{peso\ libras}$$

4.2 Fundamentos teóricos

4.2.1 Inteligencia Artificial

La inteligencia artificial (IA) es una rama de la informática cuyo objetivo es desarrollar sistemas que puedan realizar tareas que normalmente requieren inteligencia humana, como el aprendizaje, el razonamiento y la deducción [21]. Estos sistemas pueden diseñarse para imitar a los humanos. La inteligencia artificial se utiliza en diversos campos, como el procesamiento del lenguaje natural, la visión por ordenador y el reconocimiento de patrones [22].

4.2.2 Machine learning

El aprendizaje automático es una rama de la inteligencia artificial. El objetivo del aprendizaje automático es crear sistemas que puedan aprender de los datos y hacer predicciones y tomar decisiones sin instrucciones explícitas [23].

El aprendizaje automático puede dividirse en tres categorías básicas [24]:

- Aprendizaje supervisado: Las aplicaciones se convierten en expertas en predecir resultados si éstos coinciden con las expectativas.
- Aprendizaje no supervisado: En el aprendizaje no supervisado, los datos se analizan sin conocimientos previos para identificar patrones y estructuras.
- Aprendizaje por refuerzo: El aprendizaje por refuerzo se basa en el aprendizaje a partir de la experiencia y la retroalimentación.

Los algoritmos de aprendizaje automático se utilizan en diversas aplicaciones, como el reconocimiento de voz y la clasificación de imágenes:

- Reconocimiento del habla y clasificación de imágenes.
- Análisis de sentimientos y procesamiento del lenguaje natural.
- Análisis de sentimientos y reconocimiento del habla. Predicción de precios de recursos y acciones.
- Segmentación de productos y mercados.
- Evaluación del riesgo crediticio y detección del fraude.

4.2.3 Deep learning

El Aprendizaje Profundo, también denominado Deep Learning, representa una subdivisión del Aprendizaje Automático o Machine Learning, rama del aprendizaje automático o inteligencia artificial que utiliza redes neuronales. Este es un componente de un sistema más amplio". "Para ayudar a los humanos en sus tareas de resolución de problemas, la inteligencia artificial pretende permitir soluciones naturales o automáticas [26].

Un caso ilustrativo es la capacidad de la IA para examinar los aspectos visuales de las imágenes. Generar representaciones visuales y determinar sus correspondientes interpretaciones. El aprendizaje profundo es un campo que tiene muchos desarrollos nuevos e interesantes, como las redes neuronales, las redes neuronales convolucionales y las redes neuronales recurrentes. Estas estructuras se emplean en diversas aplicaciones, como procesamiento de imágenes, reconocimiento de voz y reconocimiento de caracteres escritos a mano [27].

4.2.4 Redes neuronales

La tecnología de redes neuronales utiliza computadoras para modelar el comportamiento en sistemas biológicos.comportamiento [28].

Las arquitecturas de redes neuronales normalmente constan de tres componentes principales: una capa de entrada que representa los campos de entrada, una o más capas ocultas y una capa de salida que transmite uno o más resultados computacionales de la red. [29]. Los datos de entrada se cargan en la primera capa y el valor de cada neurona se envía a la siguiente

capa. El resultado se envía a la capa de salida. Los pesos pueden ser positivos o negativos dependiendo de la actividad de una sola neurona [30].

Este proceso compara los resultados deseados con los resultados producidos por la capa de resultados cuando se invoca el mecanismo de retroalimentación. Los pesos de las capas ocultas se ajustan para que coincidan con la capa de entrada. La red aprende con el tiempo y reduce la diferencia entre los resultados esperados y los logrados [31].

4.2.5 Red neuronal convolucional (CNN)

La clasificación de imágenes mediante redes neuronales, La alta carga computacional de las redes neuronales dificulta la clasificación de imágenes. Una red neuronal tradicional se compone de capas interconectadas. El número de conexiones en la capa inicial de neuronas es 409.600, en el caso de una imagen con dimensiones de 640 x 640 píxeles El exceso de informática impide el proceso de aprendizaje [32].

Las redes neuronales parecen resolver este problema. La primera capa de estas redes está diseñada para extraer características como bordes y esquinas. Estas características se utilizan en capas para encontrar formas más complejas. Estos formularios pueden identificar características avanzadas en la imagen de un automóvil. La predicción de la última capa de la red totalmente conectada se basa en las características obtenidas [33].

4.3 Tecnologías de desarrollo

4.3.1 Herramientas

Python

Este lenguaje de programación se utiliza ampliamente en diversos campos, como la ciencia de datos, el aprendizaje automático, la inteligencia artificial, el desarrollo de software y la automatización de tareas debido a su naturaleza avanzada, flexible y fácil de aprender. Este software se utiliza ampliamente debido a su simplicidad y su amplia colección de bibliotecas y marcos que pueden abordar diversos problemas. Pueden adaptarse a diversos requisitos Gracias a su complejidad, flexibilidad y facilidad de aprendizaje, se utiliza en una amplia gama de campos como la informática, el aprendizaje automático, la inteligencia artificial, el desarrollo de software y la automatización de tareas. Es popular porque es sencillo y fácil de usar y dispone de una amplia gama de bibliotecas y frameworks para una gran variedad de tareas. Pueden adaptarse a diferentes necesidades [34].

Teachable Machine

Una herramienta gratuita basada en web para construir modelos de aprendizaje automático. Lanzada en 2017, la herramienta se puede utilizar para construir modelos de aprendizaje

automático a partir de lenguaje, texto e imágenes. Estos modelos se pueden utilizar para analizar y clasificar imágenes, sonidos y textos [35].

Construir un modelo en Teachable consta de tres pasos [36]:

- Recogida de datos: Es necesario clasificar o separar los distintos tipos de datos brutos. Tomar imágenes de distintos tipos de objetos y buscar patrones que identifiquen distintos tipos de objetos en las imágenes.
- Entrenamiento del modelo: La herramienta aprende de los datos recogidos y crea un modelo de entrenamiento. Las plantillas del modelo se crean a partir de los patrones y las características de los muebles de las imágenes.
- Uso del modelo: Una vez entrenado el estereotipo, puede utilizarse para crear nuevas imágenes y propiedades. Una vez que el estereotipo ha aprendido a identificar diferentes tipos de objetos en una imagen, puede utilizarse para crear nuevas imágenes**.**

Google Colab

Se trata de una plataforma que permite a los usuarios crear, ejecutar y ejecutar código Python que se puede ejecutar en el bloc de notas Jupyter. No está instalado en mi ordenador, así que no puedo usarlo. Jupyrt ofrece muchas funciones útiles, como recopilación y análisis de datos, aprendizaje automático y modelos de aprendizaje, informes interactivos y proyectos de software [37].

Visual Studio Code

Visual Studio código de aplicación.

Es una herramienta de desarrollo que permite escribir y modificar código para distintos tipos de ordenadores. La gestión de los cambios en el código fuente mediante Git es una herramienta útil para los desarrolladores. Con las opciones avanzadas de personalización de Visual Studio Code, se puede modificar el código fuente, las propiedades y los cambios. Es una potente herramienta que ayuda a los desarrolladores a escribir código de forma más eficiente [38].

MySQL

Sistema de gestión de bases de datos de código abierto que utiliza un lenguaje de consulta estructurado para gestionar bases de datos. El término es una descripción de MySQL que significa "gestión eficiente y segura de grandes cantidades de datos y acceso a ellos". Lo utilizan muchas empresas e industrias y es uno de los sistemas de almacenamiento de datos más utilizados del mundo. También lo utilizan sitios web y desarrolladores. Las bases de datos

se utilizan en una amplia gama de aplicaciones, desde pequeñas aplicaciones de escritorio hasta grandes sitios web y aplicaciones corporativas. Las bases de datos se utilizan para almacenar datos como información de clientes, datos financieros, catálogos de productos y otra información que debe organizarse y utilizarse de forma eficiente [39].

4.3.2 Bibliotecas

Una biblioteca de Python es un conjunto de código que se puede aplicar a diversas tareas. Estas bibliotecas simplifican la creación de aplicaciones y proyectos. [40].

TensorFlow

Una biblioteca de software para implementar y entrenar redes neuronales artificiales. TensorFlow puede utilizarse para construir modelos complejos de aprendizaje automático.

Urllib

Este módulo de biblioteca estándar proporciona un "protocolo de red". UrnLib se utiliza para transferir datos a través de una red. Consta de varios submódulos: urllib, request, urllib. parse, y urllib.robotparser. urllib es útil para la comunicación en red de Python.

Template

Las plantillas se pueden utilizar para crear formularios de texto y HTML. Cuando se crea el último archivo, las plantillas se sustituyen por nuevos formularios. Se pueden utilizar datos para crear y cargar estructuras de plantillas. Puede utilizar esta herramienta para crear páginas de aplicaciones web dinámicas y flexibles. Puede utilizarse para enviar mensajes personales, informar a los usuarios, responder a consultas web, etc. Los desarrolladores pueden compartir herramientas de diseño visual para mejorar la personalización y el mantenimiento de las aplicaciones.

Shortcuts

Esta biblioteca está diseñada para simplificar las tareas de programación repetitivas permitiendo a los programadores realizar tareas complejas de forma rápida y sencilla con una sola línea de código. El objetivo de la biblioteca es aumentar la eficacia del programador reduciendo la cantidad de código necesario para completar las tareas y minimizando el riesgo de errores.

Django-crispy-forms

Utilice esta biblioteca para crear rápida y fácilmente formularios con diseños personalizados. Usando un lenguaje claro y fácil de entender, los desarrolladores pueden crear formularios bonitos y eficientes sin tener que escribir mucho código. Esta afirmación puede leerse como

la librería tiene muchas características avanzadas como soporte Bootstrap, validación de campos en tiempo real, y más.

Xhtml2pdf

Una herramienta para crear informes y archivos PDF para aplicaciones web. La biblioteca admite plantillas Jinja2 y hojas de estilo CSS y permite crear archivos PDF utilizando XHTML2PDF. Útil para crear informes y reuniones de negocios.

ReportLab

Organice la creación de documentos PDF. Se pueden añadir al documento texto, imágenes, gráficos, tablas y otros elementos mediante diversas herramientas. Crea documentos PDF utilizando plantillas y fuentes dinámicas, funciones avanzadas de formato y diseño". Es un software versátil que puede utilizarse para crear una amplia gama de documentos e informes empresariales, imprimir etiquetas de códigos de barras y crear libros electrónicos.

Json

Es una biblioteca que proporciona herramientas para interpretar objetos. Es un formato de transferencia de datos que puede ser creado y leído por máquinas, pero también leído y escrito por humanos. Las aplicaciones y servicios web utilizan este formato para enviar y recibir datos.

Http

Esta biblioteca contiene varias herramientas para desarrollar aplicaciones web. También puede utilizarse para recuperar y gestionar respuestas. La biblioteca HTTP puede utilizarse para automatizar tareas relacionadas con la comunicación con Internet, como la recuperación de datos de páginas web. La distribución estándar de Python incluye este módulo y no requiere software adicional.

4.4 Metodología Programación Extrema XP

El desarrollo de NoRAE es rápido y enfatiza la comunicación, la retroalimentación continua y la mejora del código central". En el verano de 1996, Kent Beck trabajaba en Chrysler y concibió la idea. Según Beck, de la revista C, tenía algunas ideas sobre cómo mejorar las técnicas. El marco tiene como objetivo mejorar la riqueza del cliente al disminuir el gasto de herramientas y acortar el evento mediante la integración de elementos como costo, tiempo, costo y área de influencia del proyecto [41].

4.4.1 Características

- El software se crea mediante pruebas para garantizar que funcione correctamente.

- El proceso gira en torno a los desarrolladores y usuarios de software.
- En cada etapa de la existencia del sistema, se recomienda seguir las mejores prácticas de desarrollo de software,
- Este enfoque requiere una comprensión precisa del cliente, la capacidad de adaptarse a sus necesidades y la formación de equipos pequeños y eficientes de 2 a 12 personas. Se necesita un equipo y empleados bien capacitados [42].

4.4.2 Valores

El enfoque se basa en los principios de XP. Cada individuo debería extraer de él sus propios pilares. Están conectados con el deseo de hacer [41], [42].

Comunicación: Algunos de los valores son la comunicación, la sencillez, la retroalimentación y la valentíaLos miembros del equipo deben comunicarse de manera efectiva. A lo largo del proyecto colaboran en todas las etapas. Es fundamental documentar todo el proceso minuciosamente.

Simplicidad: Es crucial simplificar el mantenimiento del software para mantener la simplicidad del diseño y desarrollo. Es recomendable modificar su código periódicamente para mejorar su legibilidad para los programadores.

Retroalimentación: Es importante ser audaz y responder a las características que desea el cliente". Conclusión: es importante aceptar los retos, aunque sean difíciles, y ser sincero, sin reservas, sobre los progresos y las evaluaciones.

Coraje: Es importante ser valiente a la hora de cumplir los requisitos del cliente. Es importante encontrar soluciones, aunque sean difíciles, y ser honesto sobre los progresos y las evaluaciones sin reservas. Esforzarse por alcanzar el éxito y estar preparado para adaptarse a los cambios que se produzcan.

Respeto: El equipo debe tratarse con respeto y reconocer la importancia de la contribución de cada uno a la creación de productos de alta calidad. Como cada miembro del equipo tiene ideas y aportaciones únicas, es importante encontrar la mejor solución de diseño, aunque sólo sea un capricho [43].

4.4.3 Herramientas

Historias de Usuario: Proporciona información sobre las actividades de los usuarios y se utiliza para estimar el tiempo y sugerir el diseño. Cada historia está relacionada con una tarea importante del sistema. Es importante seguir estas pautas al realizar una prueba de aceptación. Es importante que cada historia esté escrita de forma clara y concisa para que los desarrolladores puedan identificarla fácilmente de un vistazo [43].

4.4.4 Roles

En la Tabla 2, se detallen los roles de XP agrupados por categorías [41], [42]:

Tabla 2. Roles de XP

CLIENTE	DESARROLLADOR	GESTIÓN	OTROS
Cliente La verificación de su implementación se puede realizar utilizando historias de usuarios de clientes y pruebas funcionales. Determina el orden de tus historias de usuario y elige en cuáles trabajar en cada iteración, concentrándote en las historias que brindan el mayor beneficio a tu empresa. El proyecto está siendo dirigido por un grupo de personas que se verán afectadas por él.	**Programador** El sistema de desarrollador se utiliza para desarrollar código y escribir pruebas. Puedes asignar tareas y estimar su duración. Los desarrolladores y otros miembros del equipo deben trabajar juntos. Los clientes utilizan controladores de prueba para escribir pruebas. El equipo de pruebas debe ser supervisado y los resultados deben informarse al equipo. **Encargado de Pruebas (*Tester*)** El director de pruebas ayuda a los clientes. El equipo de prueba y los resultados se informarán al equipo.	**Entrenador/Tutor (*Coach*)** El docente es el supervisor del proceso. Es necesario que los miembros del equipo tengan un conocimiento profundo del proceso XP para poder ejecutarlo correctamente. **Gestor (*Big boss*)** Actúan como enlace entre clientes y desarrolladores, ayudando a los equipos a trabajar de manera eficiente creando el entorno adecuado. **Encargado de seguimiento/Perseguidor (*Tracker*)** Retroalimentación al equipo, responsable de determinar el grado de correspondencia entre las estimaciones y el tiempo real invertido y de informar los resultados para mejorar las estimaciones futuras.	**Consultor** Se sugiere incorporar miembros del equipo externos que tengan conocimientos especializados en determinadas áreas del proyecto. Un equipo colabora para resolver un problema.

4.4.5 Fases

En la tabla 3 describe 4 fases las cuales son [41], [42]:

Tabla 3. Fases de la metodología XP

FASE 1	FASE 2	FASE 3	FASE 4
Planificación	**Diseño**	**Codificación**	**Pruebas**
En esta fase se establece la visión y los objetivos a largo plazo del proyecto. El equipo de desarrollo prioriza las características del producto según su importancia para el cliente, estima el tiempo y el costo del proyecto y determina el equipo necesario para llevar a cabo el trabajo. También se establecen los hitos del proyecto.	En esta fase se crea un diseño detallado del sistema a partir de la lista de características del producto. El equipo de desarrollo utiliza herramientas de modelado, como diagramas de flujo y de clases, y puede crear prototipos para probar diferentes enfoques. El diseño se actualiza y mejora en cada iteración.	En esta fase, el equipo de desarrollo escribe el código del sistema utilizando la programación en parejas. Dos programadores trabajan juntos en la misma tarea para garantizar que el código sea de alta calidad y fácil de mantener. La codificación se realiza en pequeños incrementos, lo que significa que el sistema se desarrolla de forma iterativa y se entrega en pequeñas partes funcionales.	En esta fase, se verifican que el sistema cumpla con los requisitos del cliente. Las pruebas se realizan en cada iteración y se centran en la funcionalidad más crítica del sistema. Las pruebas incluyen pruebas unitarias, pruebas de integración y pruebas de aceptación, todo ello para garantizar que el sistema cumple las perspectivas de los clientes.

4.5 Trabajos relacionados

Resulta fundamental contar con una revisión bibliográfica que contemple una variedad de trabajos sobre el control del crecimiento en pollos de engorde, con el fin de adquirir distintas perspectivas acerca del proceso llevado a cabo en otras granjas avícolas. Por lo tanto, se ha realizado un estudio de algunos de estos trabajos, donde se presenta una breve síntesis de los aspectos más relevantes de cada uno de ellos.

4.5.1 Primer Trabajo Relacionado

SISTEMA EMBEBIDO PARA LA AUTOMATIZACIÓN DEL CONTROL Y MONITOREO DE LA PRODUCCIÓN EN LA GRANJA AVÍCOLA "ROMERO & HNOS El objetivo del proyecto es utilizar sensores y actuadores para controlar y seguir las actividades de la granja avícola Romero & Hnos. En el Cantón Balsas, dentro de la provincia de El Oro, se lanzó un proyecto que permitió establecer un sistema automatizado para la supervisión y gestión de una granja avícola. Se empleó el tablero para vincular los dos elementos. este sistema se colocó en varios puntos del almacén para evitar que entrara en contacto con los dispositivos de control de temperatura [44].

4.5.2 Segundo Trabajo Relacionado

SISTEMA DE GESTIÓN POR PROCESOS EN LA LÍNEA DE PRODUCCIÓN PARA LA EMPRESA AVÍCOLA LA PONDEROSA EN EL CANTÓN DE SALCEDO. El objetivo de este proyecto es resolver problemas con los procedimientos existentes en la granja avícola "La Ponderosa". La ausencia de directrices, manuales y registros coherentes dificulta la documentación adecuada de las actividades. El comportamiento de los operadores no se alinea con las condiciones comerciales actuales. El inicio de la investigación implica un cuestionario entregado a los trabajadores de granjas avícolas. Esta encuesta ayuda a determinar las actividades que se realizaron durante el proceso de fabricación y los documentos utilizados en ellas. Se lleva a cabo un examen exhaustivo para evaluar el estado actual de la industria avícola. El objetivo de este examen es evaluar el estado actual del procedimiento [45].

4.5.3 Tercer Trabajo Relacionado

DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO IOT PARA EL MONITOREO DE PARÁMETROS AMBIENTALES APLICADOS A LA AVICULTURA PARA LA CRIANZA DE POLLOS DE GRANJA UTILIZANDO HARDWARE DE BAJO COSTO Y AWS. Se ha implementado un sistema automatizado, similar al que han adoptado las principales potencias mundiales, para mejorar el sector, disminuir los gastos y permitir que los microempresarios participen en el ámbito tecnológico. Esta frase se puede parafrasear como: Para mejorar la productividad tanto de los pequeños criadores de aves como de los grandes productores de

aves de corral, es esencial automatizar el proceso de eclosión. La importancia de esta automatización radica en la necesidad de fomentar la utilización de sistemas automatizados desarrollados por profesionales locales y la implementación de un nuevo enfoque para ejecutar este proceso dentro de la industria avícola. Dada la rentabilidad del entorno avícola, es posible [46]

4.5.4 Cuarto Trabajo Relacionado

CONTROL Y MONITOREO DE UN CRIADERO AVÍCOLA CONTROLADO POR MICROCONTROLADOR DESDE UN SITIO WEB. El objetivo del proyecto es crear un sistema que mejore la calidad del producto y reduzca los costos en la industria avícola a través de la automatización. Al acceder a www.sisavic.com se muestra la página principal del sistema. Se puede acceder a una página específica mediante nombres de usuario y contraseñas para rastrear y controlar la granja avícola. Los datos recibidos de los sensores se utilizan para regular la apertura y el cierre de las válvulas de agua. La distancia desde el suministro hasta el silo se mide mediante un sensor en el silo. La web muestra los datos del sensor que alimenta el sistema. el administrador recibe una notificación por correo electrónico cuando la temperatura cae por debajo de cierto nivel. Si la temperatura excede el umbral, se envía una notificación por correo electrónico al administrador [47].

4.5.5 Quinto Trabajo Relacionado

SISTEMA DE INFORMACIÓN PARA EL CONTROL Y MONITOREO ARDUINO DE LA CRIANZA AVÍCOLA EN LA GRANJA "PURA PECHUGA". Este pasaje explica cómo se creó un sistema para supervisar y regular las condiciones en las que se crían los animales en la granja Pura Pechuga. Este sistema contiene tanto seres vivos como dispositivos que detectan cosas. Se puede utilizar una aplicación web para rastrear el desarrollo de los animales y se puede acceder a ella desde cualquier dispositivo conectado a Internet. La granja donde se está implementando este proyecto no tiene los medios financieros para invertir en costosos sistemas de seguimiento y control de las zonas de cría de animales. Se ha establecido un sistema para cumplir con los requisitos de la granja. Para lograr esto se ha empleado la utilización de tecnología de software libre. Los sensores de la base de datos capturan los datos. El formato digital elimina la necesidad de que las hojas se extravíen o dañen [48].

5. Metodología

En esta sección se explica de manera detallada el procedimiento y los recursos utilizados en el desarrollo del Trabajo de Integración Curricular, el cual se basó en una investigación descriptiva y de mejora para cumplir con los objetivos previamente definidos.

En la **sección 5.1** se describe el área de estudio en donde se llevó a cabo el proyecto, mientras que en la **sección 5.2**, se proporciona información acerca del contexto en el que se llevó a cabo el Trabajo de Integración Curricular, en la **sección 5.3** se describe el proceso para alcanzar los objetivos, incluyendo las actividades realizadas y sus respectivos anexos para demostrar los resultados obtenidos. Además, la **sección 5.4** se enfoca en los recursos empleados, tales como recursos científicos y técnicos. Finalmente, la **sección 5.5** se dedica a presentar a los participantes involucrados en el trabajo de Trabajo de Integración Curricular.

Adicionalmente, se procedió a desarrollar, definir y organizar el conjunto de enfoques metodológicos, métodos, técnicas, estándares y recursos que establecieron una serie estructurada de pasos para llevar a cabo la realización del proyecto. Estos elementos se detallan a continuación:

5.1 Área de estudio

El Trabajo de Integración Curricular, se llevó a cabo en la granja avícola “Las Acacias” como se muestra en la Figura 16, la principal fuente de ingresos se encuentra en la cría de pollos de engorde Broiler destinados al consumo en masa en las provincias de El Oro, Loja, Azuay y Guayas. Se encuentra ubicada en Ecuador, Provincia de El Oro, Cantón Balsas, Sector las Acacias, y cuyo supervisor legal es el Sr. Fernando Bravo. El TIC se realizó en un ámbito académico y durante el periodo de Abril 2023 - Septiembre 2023 establecido por la Universidad Nacional de Loja (**ver Mapa**).

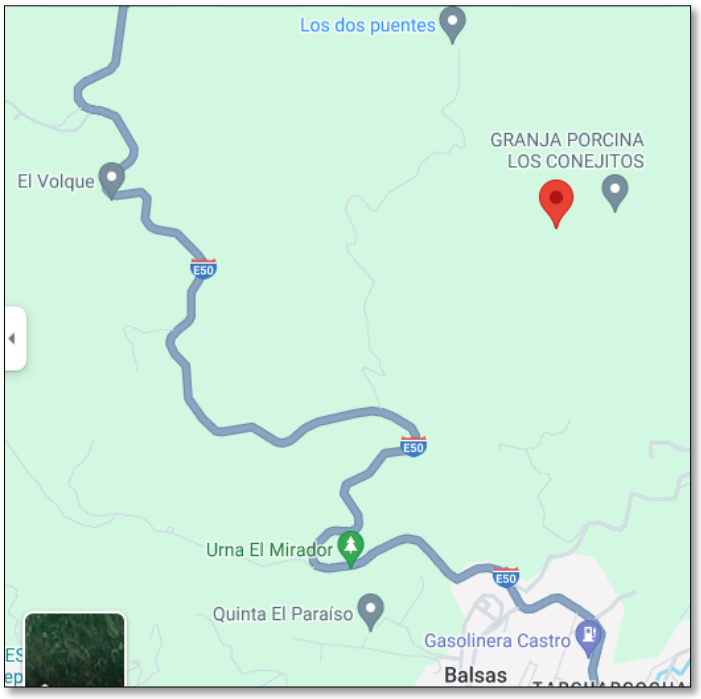


Figura 16. Mapa de Balsas

5.2 Contexto

El Trabajo de Integración Curricular, planteó como objetivo llevar a cabo el desarrollo del software para el Control de Crecimiento en pollos de engorde en la granja Avícola “Las Acacias” en el Cantón Balsas. Este enfoque se originó a partir de la problemática de investigación y la cuestión planteada: **¿De qué manera un software permite agilizar el control de crecimiento en la producción de pollos de engorde en la granja avícola “Las Acacias” del Cantón Balsas?,** a partir de la cual se dio inicio al desarrollo del Trabajo de Integración Curricular, hasta la finalización del mismo.

5.3 Procedimiento

Con el fin de alcanzar los objetivos definidos en el Trabajo de Integración Curricular, se describe el proceso que se siguió para lograr el objetivo general, en el cual se incluyen los diferentes objetivos específicos y las actividades correspondientes a cada uno de ellos.

5.3.1 Objetivo 1. Construir un modelo de reconocimiento de imágenes mediante visión artificial para la identificación del crecimiento del tamaño del pollo.

- Se realizó la recopilación de imágenes en diferentes fases de crecimiento como: “Pre-inicio, Inicio, Desarrollo, Engorde” para utilizarlas como base de datos para el modelo en la cual se obtuvo un total de 520 imágenes. Las imágenes obtenidas correspondieron a fotografías tomadas en la granja "Las Acacias", de autoría propia, debido a la ausencia de

una base de datos en línea. (**véase sección 6.1, Recopilación de imágenes en diferentes fases).**

- Se realizó la categorización de datos mediante etiquetas a las que pertenecen las imágenes del conjunto de datos. En este caso, las etiquetas pueden incluir diferentes fases de crecimiento de los pollos en la cual se obtuvo un total de 130 imágenes para cada fase (**véase sección 6.1, Categorización de datos**).
- Se realizó el preprocesamiento con el objetivo de mejorar su calidad y eliminar el ruido presente en cada una de ellas. Adicionalmente, se llevó a cabo la selección de imágenes para cada fase del proceso, considerando que en la etapa inicial de recopilación y categorización se identificaron imágenes similares, borrosas o inadecuadas para el propósito. Como resultado de esta selección, se obtuvieron un total de 130 imágenes en la fase de Pre-inicio, 130 imágenes en la fase de Inicio, 130 imágenes en la fase de Desarrollo y 130 imágenes en la fase de Engorde. En conjunto, se contó con un conjunto final de 520 imágenes que fueron utilizadas en el análisis y reconocimiento del crecimiento del tamaño del pollo. (**véase sección 6.1, Preprocesamiento de imágenes**).
- Se realizó el diseño del pre-entrenamiento con la herramienta Google Colab, utilizando un conjunto de datos etiquetados para reconocer las fases de crecimiento de los pollos y así identificar su tamaño. Se emplearon técnicas de aprendizaje automático para adaptarse y generalizar los patrones aprendidos, se utilizó una clase ImageDataGenerator de TensorFlow para aumentar datos de imágene con varios parámetros de aumento de datos, como rotación, cambio de ancho y alto, sesgo, y zoom, donde se obtuvo 1,166 imágenes, de modo que pudiera identificar con precisión el tamaño del pollo en nuevas imágenes no vistas durante el entrenamiento (**véase sección 6.1, Pre-entrenamiento del modelo**).
- Se realizó la compilación del modelo pre-entrenado, estableciendo su configuración para el entrenamiento, incluyendo algoritmos de optimización, función de pérdida y métricas de evaluación. Se visualizó el historial de entrenamiento para comprender la evolución del rendimiento (**véase sección 6.1, Creación del modelo pre-entrenado**).
- Se realizó la **codificación del entrenamiento del modelo** con un conjunto de datos etiquetados de calidad permitió enseñar al modelo a reconocer las fases de crecimiento de los pollos y, en consecuencia, aprender a identificar el tamaño del pollo a partir de las características extraídas. El uso de técnicas de aprendizaje automático y un conjunto de datos diverso proporcionó al modelo la capacidad de generalizar y aplicar su conocimiento a nuevas imágenes (**véase sección 6.1, Codificación del modelo entrenado**).
- Se realizó una evaluación del modelo utilizando un conjunto de imágenes de prueba diverso. Esta evaluación imparcial permitió determinar la precisión y el desempeño del modelo en situaciones del mundo real, lo que proporcionó información para ajustar y

mejorar el modelo en caso de ser necesario (**véase sección 6.1, Pruebas del modelo entrenado**).

5.3.2 Objetivo 2. Implementar el software para el control de crecimiento utilizando la metodología XP mediante el lenguaje de programación Python.

- **Planificación:**
  - Se realizó entrevistas a los expertos encargados, así mismo se buscó trabajos relacionados para recabar información necesaria **(véase Anexo 2**).
  - Se realizó el documento de especificación de requisitos de software con el estándar IEEE-830, donde se establecieron las iteraciones y se definieron las historias de usuario correspondientes a cada una de ellas **(véase Anexo 3**).
- **Diseño:**
  - Se realizó el documento de arquitectura del software, modelo arquitectónico 4+1, en donde se realizó el modelo de cada vista mediante modelos UML **(véase Anexo 4**).
- **Codificación:**
  - Se elaboró una descripción minuciosa de los estándares de codificación adoptados. Cada iteración planificada se tradujo en historias de usuario, y el código se desarrolló en consonancia con estas historias **(véase Anexo 6**).
- **Pruebas:**
  - Se elaboró pruebas unitarias, de integración y de satisfacción de formularios específicos para los métodos encargados de manejar la lógica empresarial. Estas pruebas se ejecutaron para garantizar que los métodos cumplan exitosamente con sus funciones previstas **(véase Anexo 7, Anexo 8, Anexo 9, Anexo 10**).

5.4 Recursos

A continuación, se detallan los recursos científicos y técnicos empleados en la ejecución de los objetivos:

5.4.1 Recursos científicos:

- **Método analítico:** implica descomponer el conjunto en sus elementos constituyentes con el propósito de lograr una comprensión más profunda del objeto de estudio. Dentro del contexto de este Trabajo de Integración Curricular, se formuló un objetivo general en base a la interrogante de investigación presentada. Este objetivo general se desglosó en objetivos específicos, y posteriormente, se subdividió en actividades destinadas a la consecución de dichos objetivos.Por lo tanto, a lo largo de todo el desarrollo del Trabajo de Integración Curricular, (**véase sección de Procedimiento**), se puede observar la aplicación del método analítico.

- **Método científico:** con el propósito de llevar a cabo el desarrollo del Trabajo de Integración Curricular, se implementó una estrategia con el fin de lograr un resultado concreto y alcanzar el objetivo de la investigación planteada. Esta estrategia se encuentra reflejada en el proceso delineado para cada uno de los objetivos establecidos.tal como se describe en el apartado (**véase sección de Procedimiento**), donde se presentan los pasos seguidos y los resultados obtenidos.
- **Estudio de caso:** se caracteriza por examinar situaciones particulares de un fenómeno relacionado con la ingeniería de software en un entorno práctico, involucrando diversas fuentes de información. En este sentido, se llevaron a cabo investigaciones de casos análogos en los que se hubiera implantado un módulo de software para automatizar procesos (**véase sección de Trabajos Relacionados)** y se procedió a analizar tales situaciones.
- **Entrevistas:** a través de esta técnica, se logró adquirir información valiosa de las personas que estuvieron involucradas en granja Avícola “Las Acacias” en el Cantón Balsas, (**véase sección de Entrevistas**) acerca de las actividades que se realizan así también de cómo se manejan y registran los diferentes parámetros de la avícola.

5.4.2 Recursos técnicos:

- **Herramientas de trabajo colaborativo:** Se emplearon aplicaciones de Google como Drive, Documentos y Hojas de cálculo, junto con herramientas de Microsoft 365 como OneDrive y Word. Además, se utilizó Lucid Chart, Diagrams.net, Mendeley y Zoom.
- **Estándar IEEE-830 para la especificación de requisitos:** Se siguió esta norma para definir los requisitos funcionales y no funcionales del sistema.
- **Metodología XP:** Se aplicó esta metodología ágil para el desarrollo del módulo de software en el segundo objetivo.
- **Modelo 4 + 1:** Se utilizó este enfoque para obtener una comprensión global de la solución informática a través de la arquitectura adoptada. En este modelo se detalló la arquitectura tanto de hardware como de software, empleando múltiples perspectivas concurrentes y utilizando modelos UML.
- **Otras herramientas y tecnologías:** Python, Teachable Machine, Google Colab, Visual Studio Code, MySQL.

5.5 Participantes

El presente Trabajo de Integración Curricular, fue desarrollado por los siguientes participantes:

- Viviana Maricela Zambrano Romero, autora del Trabajo de Integración Curricular. Su involucramiento abarca desde la formulación del anteproyecto hasta la ejecución y conclusión de los objetivos establecidos en el Trabajo de Integración Curricular.

- Ing. Edwin René Guamán Quinche Mg.Sc, director del Trabajo de Integración Curricular. Contribuyó mediante la revisión de los aspectos académicos, prácticos y metodológicos del proyecto.
- Sr. Alexis Jeovanny Pinto Añazco, representante legal de la Granja Avícola "La Acacias".
- Sr. Fernando Bravo, supervisor de la Granja Avícola "La Acacias"

.

6. Resultados

En esta sección se proporcionan los detalles de los resultados logrados a partir de la ejecución del Trabajo de Integración Curricular. El proyecto se desarrolló siguiendo la metodología de desarrollo XP, en consonancia con los objetivos específicos previamente establecidos.

1.1. Objetivo 1

Construir un modelo de reconocimiento de imágenes mediante visión artificial para la identificación del crecimiento del tamaño del pollo.

En esta etapa, se involucran las siguientes fases representadas en la Figura 17:

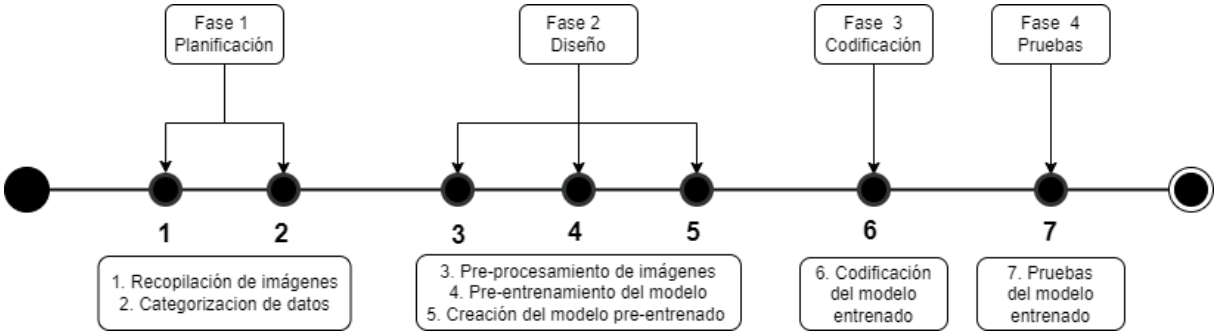


Figura 17. Proceso de implementación para el primer objetivo.

Fase 1: Planificación

Recopilación de imágenes en diferentes fases

En esa etapa, se recopilaron imágenes de pollos en varias fases de crecimiento, un total de 520, que abarcaban "Pre-inicio", "Inicio", "Desarrollo" y "Engorde" con un total de 130 para cada fase mecncionada. (Véase en el siguiente enlace **Recopilación de imágenes en diferentes fases**). Estas imágenes fueron utilizadas como conjunto de datos para entrenar el modelo de reconocimiento de imágenes, como se muestra en la Figura 18. Es importante destacar que las imágenes no se encontraban organizadas en ningún orden específico, sino que se les asignó una numeración aleatoria.

Figura 18. Recopilación de imágenes

Categorización de datos

Se llevó a cabo la categorización de datos mediante la asignación de etiquetas a las imágenes del conjunto de datos. Como se muestra en la Figura 19, estas etiquetas fueron utilizadas para identificar la fase de crecimiento del pollo a la que pertenecía cada imagen. Dichas fases se basaron en l**as especificaciones del ciclo productivo** y se emplearon etiquetas correspondientes a las etapas de "Pre-inicio", "Inicio", "Desarrollo" y "Engorde", junto con una numeración adicional. (Véase en el siguiente enlace **Categorización de datos**).

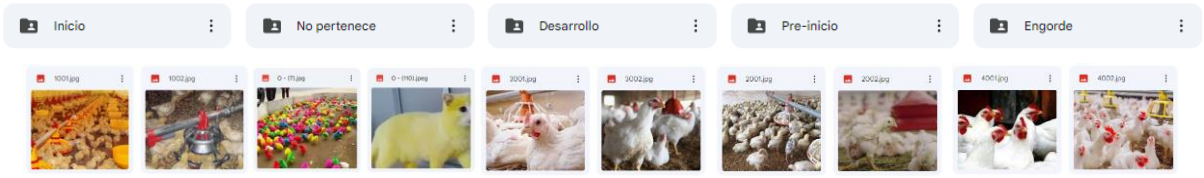


Figura 19. Categorización de datos

Fase 2: Diseño

Diagrama

En la Figura 20, el diagrama representa visualmente el flujo de trabajo del modelo de reconocimiento de imágenes para la identificación del crecimiento del tamaño del pollo, desde la entrada de la imagen hasta la salida de la predicción.

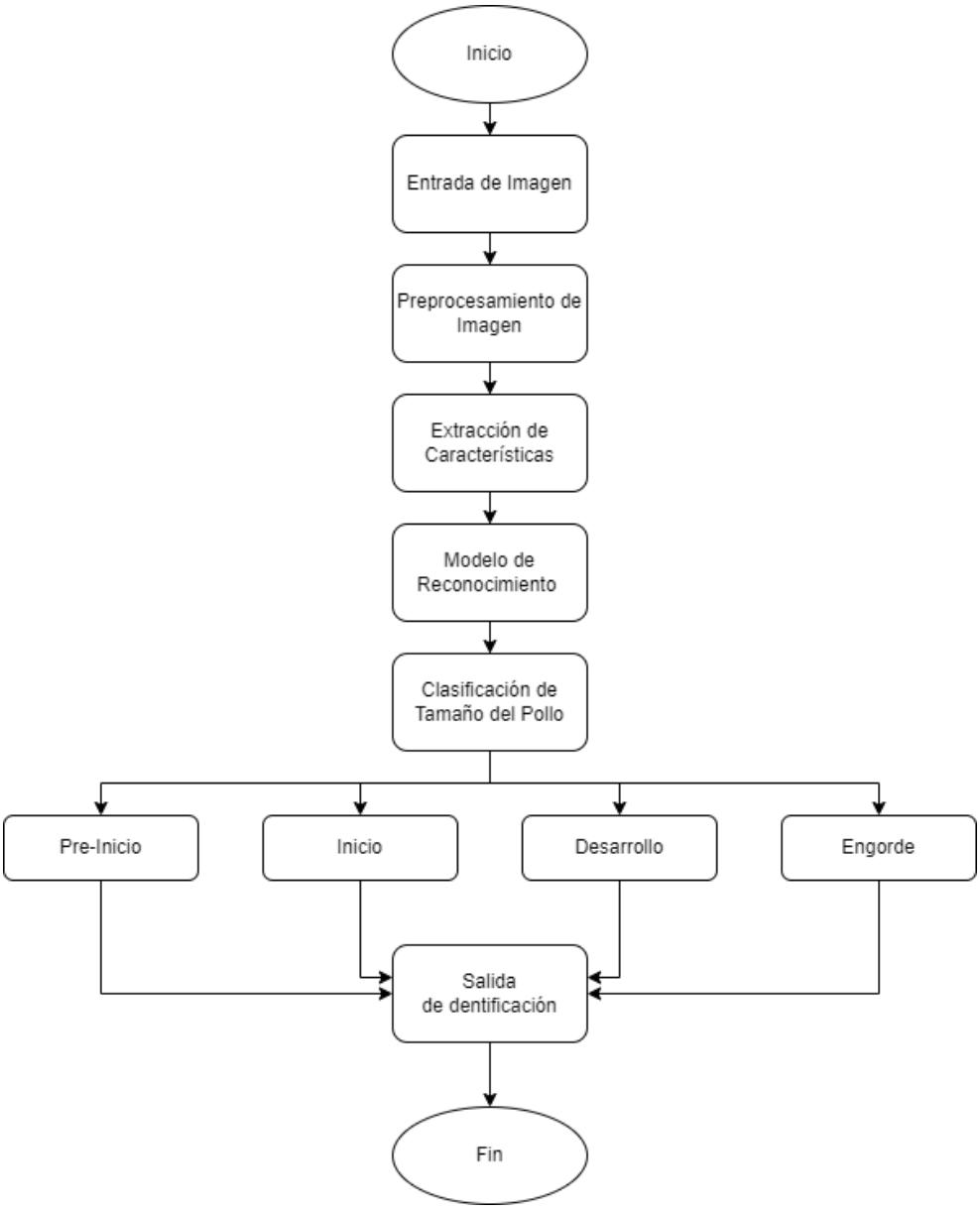


Figura 20. Diagrama del flujo del módulo

Preprocesamiento de imágenes

Se llevaron a cabo una serie de técnicas, como el escalado, recorte y normalización, con el fin de asegurar que todas las imágenes estuvieran en el mismo formato y tamaño. Se estableció una numeración para cada fase. En la Figura 21, el objetivo principal consistió en preparar las imágenes para su posterior análisis y entrenamiento en el modelo de reconocimiento de imágenes en diferentes fases de crecimiento del pollo, tales como "Pre-inicio", "Inicio", "Desarrollo" y "Engorde". (Véase en el siguiente enlace **Preprocesamiento de imágenes**).

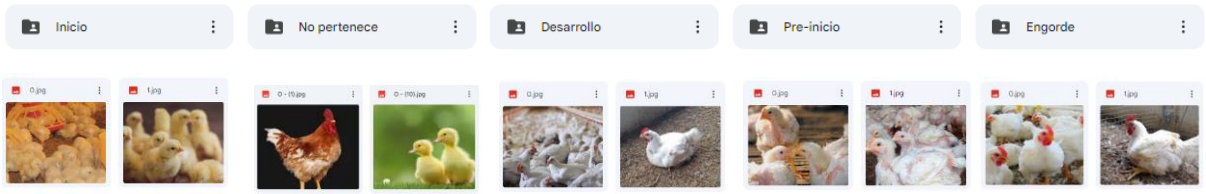


Figura 21. Preprocesamiento de imágenes

Pre-entrenamiento del modelo

Esta fase se utilizó el conjunto de imágenes agrupadas por categoría que permite enseñar al modelo a reconocer. En este caso, el conjunto de datos etiquetados se utilizó para enseñar al modelo a reconocer diferentes fases de crecimiento de los pollos, lo que permitiría al modelo identificar el tamaño del pollo a partir de las características extraídas.

Durante ese proceso, el modelo fue expuesto a imágenes etiquetadas de diferentes fases de crecimiento, se utilizó una clase ImageDataGenerator de TensorFlow para aumentar datos de imágene con varios parámetros de aumento de datos, como rotación, cambio de ancho y alto, sesgo, y zoom, donde se obtuvo 1,166 imágenes, lo que le permitió aprender los patrones visuales y las características distintivas asociadas a cada fase. A medida que el modelo se familiarizaba con más imágenes etiquetadas, ajustaba sus parámetros internos para mejorar su capacidad de reconocimiento. Estas imágenes se utilizaron como ejemplos para que el modelo aprendiera los patrones visuales y las características distintivas asociadas a cada fase de crecimiento.

Es relevante resaltar que, en esta situación en particular, el modelo entrenado fue sometido a 75 iteraciones, donde cada iteración comprendió el procesamiento completo de los datos de entrenamiento a través del modelo. En el transcurso de cada iteración, el modelo evaluó y ajustó sus parámetros utilizando el algoritmo de optimización y la función de pérdida predeterminada. La finalidad de realizar múltiples iteraciones durante el proceso de entrenamiento radicaba en permitir que el modelo aprendiera de manera progresiva de los datos. En cada iteración, el modelo efectuó predicciones sobre el conjunto de entrenamiento, calculó la pérdida correspondiente y refinó sus parámetros para mejorar su habilidad en realizar pronósticos más precisos.

```
#Entrenar el modelo
EPOCAS = 75
historial = modelo.fit(
    data_gen_entrenamiento, epochs=EPOCAS, batch_size=32,
    validation_data=data_gen_pruebas
)
```

Creación del modelo pre-entrenado

Se diseñó y entrenó una red neuronal convolucional (CNN), MobileNet V2 diseñada específicamente para aplicaciones de visión por computadora y tareas de aprendizaje profundo en dispositivos móviles y sistemas con recursos limitados. Esta red se destaca por su eficiencia computacional y su capacidad para ejecutarse en dispositivos con menos potencia de cómputo y memoria.

El modelo MobileNet V2 tiene varias capas que componen su arquitectura. A continuación, se mencionan algunas de las capas principales que conforman este modelo [49]:

Capa de entrada: Esta capa toma la imagen de entrada y la pasa a través de la red para su procesamiento.

Capas de convolución y activación: MobileNet V2 utiliza múltiples capas de convolución que aplican filtros para extraer características importantes de la imagen. Cada capa de convolución está seguida por una función de activación, como ReLU (Rectified Linear Unit), que introduce la no linealidad en el modelo.

Capas de separable y lineal convolución: MobileNet V2 utiliza un enfoque llamado separable convolución que reduce drásticamente la cantidad de parámetros en el modelo. Esto se logra separando la convolución en dos etapas: una convolución profunda (o "depthwise") y una convolución lineal. Este enfoque mejora la eficiencia computacional del modelo sin comprometer el rendimiento.

Capas de pooling: Después de algunas capas de convolución, se aplican capas de pooling para reducir el tamaño de las características y mantener solo la información más relevante.

Capa global de promedio: En lugar de usar una capa densa o completamente conectada, MobileNet V2 utiliza una capa global de promedio para reducir la dimensionalidad de las características finales. Esta capa calcula el promedio de todas las activaciones en cada canal de características, generando un vector de características más compacto.

Durante el entrenamiento, en última capa el modelo aprendió a reconocer y extraer características relevantes de las imágenes de pollos en diversas etapas de crecimiento. El objetivo era desarrollar un modelo capaz de identificar el tamaño del pollo en función de las imágenes. Una vez completado el entrenamiento, el modelo pre-entrenado estaba listo para su implementación. Este modelo contenía la información y el conocimiento adquiridos durante el entrenamiento, lo que le permitía realizar predicciones sobre nuevas imágenes de pollos y reconocer el tamaño del pollo en base a las características aprendidas.

Para evaluar el rendimiento del modelo pre-entrenado, se siguió un proceso específico. En primer lugar, se descargó la imagen utilizando la URL proporcionada como entrada. Luego, se aplicó un procesamiento que incluía la normalización de la imagen. A continuación, se redimensionó la imagen para que se ajustara al tamaño requerido por el modelo pre-entrenado. Finalmente, se realizó la predicción utilizando el modelo previamente entrenado. El resultado de esta predicción proporcionaba la identificación de la fase correspondiente a la imagen, es decir, se determinaba en qué etapa de crecimiento se encontraba el pollo en la imagen. Este proceso de prueba y clasificación permitió evaluar la efectividad y precisión del modelo pre-entrenado en la tarea de reconocimiento de las diferentes fases de crecimiento.

```
#Categorizar una imagen de internet
from PIL import Image
import requests
from io import BytesIO
import cv2

def categorizar(url):
  respuesta = requests.get(url)
  img = Image.open(BytesIO(respuesta.content))
  img = np.array(img).astype(float)/255

  img = cv2.resize(img, (224,224))
  prediccion = modelo.predict(img.reshape(-1, 224, 224, 3))
  return np.argmax(prediccion[0], axis=-1)

# Prueba con una imagen de internet
url =
'https://t2.uc.ltmcdn.com/es/posts/4/9/6/como_cuidar_de_un_pollito_27694_600_square.jpg'
prediccion = categorizar (url)
print(prediccion)
```

Fase 3: Codificación

Codificación del modelo entrenado

La **codificación del modelo** se realizó una vez completado el proceso de entrenamiento. Permitió almacenar toda la información necesaria para utilizar el modelo. Durante el entrenamiento, se utilizó un conjunto de datos etiquetados cuidadosamente seleccionado para su codificación. Este conjunto de datos desempeñó un papel fundamental en la enseñanza al modelo para reconocer diferentes categorías, centrándose específicamente en las fases de crecimiento de los pollos. Al proporcionar ejemplos etiquetados de cada fase de crecimiento al modelo, se le permitió aprender y comprender los patrones visuales y las características

distintivas asociadas a cada etapa. De esta manera, el modelo pudo capturar las sutilezas que indican el crecimiento del pollo y utilizar estas características extraídas para identificar el tamaño del pollo en futuras imágenes como se muestra en el siguiente codigo.

```
def predict_imagen2(request):
    if request.method == 'POST' and request.FILES['image']:
        # Obtener la imagen cargada por el usuario
        image_file = request.FILES['image']
        # Guardar la imagen en el directorio MEDIA_ROOT
        image_path = os.path.join(settings.MEDIA_ROOT, image_file.name)
        with open(image_path, 'wb') as f:
            f.write(image_file.read())
        # Especificar el dispositivo de E/S
        options =
tf.saved_model.LoadOptions(experimental_io_device='/job:localhost')
        # Cargar el modelo SavedModel
        model = tf.saved_model.load(os.path.join(settings.STATIC_ROOT,
'saved_model'), options=options)
        # Obtener la función predict del modelo
        predict_fn = model.signatures['serving_default']
        # Cargar y preprocesar la imagen de entrada
        image = tf.io.read_file(image_path)
        image = tf.image.decode_image(image, channels=3)
        image = tf.image.resize(image, (224, 224))
        image = image / 255.0                                # Normalizar la
imagen
        # Agregar una dimensión extra para que coincida con el tamaño de
lote esperado por el modelo
        image = tf.expand_dims(image, axis=0)
        # Realizar la predicción
        predictions = predict_fn(image)['dense']
        # Obtener la clase con mayor probabilidad
        predicted_class = tf.argmax(predictions, axis=1)[0].numpy()
        # Definir las etiquetas correspondientes a las clases
        labels = {
            4: 'Pre-inicio',
            3: 'No pertenece'
            2: 'Inicio',
            1: 'Engorde',
            0: 'Desarrollo'
        }
        # Obtener la etiqueta correspondiente a la clase predicha
        predicted_label = labels[predicted_class]
        # Eliminar el archivo de imagen después de la predicción
        os.remove(image_path)
        # Guardar la predicción y el enlace de la imagen en el modelo
Prediccion
        prediccion = Prediccion()
```

```
        prediccion.imagen = image_file
        prediccion.etiqueta = predicted_label
        prediccion.save()2
        # Renderizar el resultado en el template
        return render(request, 'prediccion/result.html', {'prediccion':
prediccion, 'predicted_label': predicted_label})
    return render(request, 'prediccion/upload.html')
```

Fase 4: Pruebas

Plan de pruebas

1. Para realizar la evaluación, se procedió a someter al modelo a un conjunto de imágenes que abarcaban una amplia variedad de condiciones, como diferentes iluminaciones, ángulos de captura y variaciones en las características de los pollos completamente diferente al utilizado durante el entrenamiento, además se realizó una búsqueda en fuentes externas para asegurarse de que no se encontraran previamente incluidas en el conjunto de entrenamiento, (Véase en el siguiente enlace **Base de datos - Prueba**).
2. En segundo lugar, se procedió a evaluar el modelo utilizando las imágenes no vistas previamente con el fin de determinar su capacidad para generalizar y realizar predicciones precisas en condiciones diferentes a las del conjunto de entrenamiento. El objetivo era representar situaciones reales y desafiantes que el modelo podría encontrar en un entorno de aplicación práctico. En la Figura 22, cada imagen fue procesada y se aplicaron las técnicas de reconocimiento previamente establecidas. El modelo utilizó las características extraídas durante el entrenamiento para realizar predicciones sobre el tamaño del pollo en cada imagen de prueba

Figura 22. Identificación de Fases

3. Además, durante el proceso de evaluación, se registraron y analizaron los resultados obtenidos por el modelo. En la Figura 23, se evaluó la precisión de las predicciones comparando las respuestas del modelo con las categorías de tamaño reales de los pollos en las imágenes de prueba.

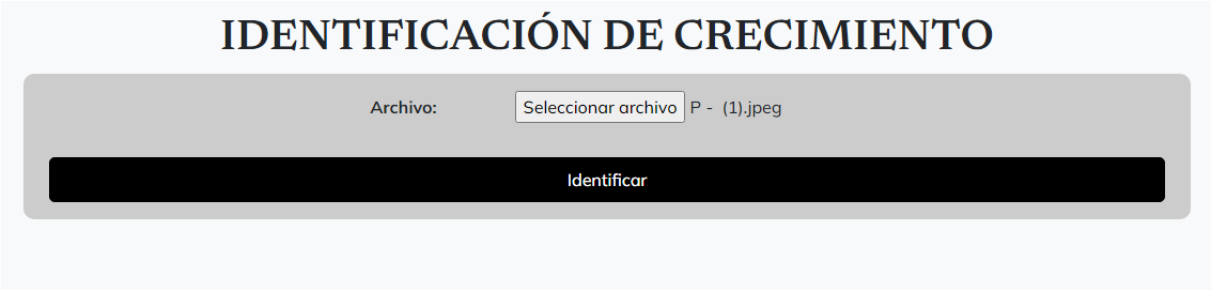


Figura 23. Evaluación de las Fases

Los resultados de la evaluación brindaron información sobre cómo el modelo se comporta en situaciones reales.

Pruebas unitarias

En la Tabla 4 se detalla las diferentes formas de evaluar con pruebas unitarias, donde se podrá validar si el proceso la construcción de un modelo de reconocimiento de imágenes mediante visión artificial para la identificación del crecimiento del tamaño del pollo está funcionando correctamente.

En las pruebas unitarias, se evalúa el rendimiento de la identificación de las fases en escenarios individuales y se comprueba si el resultado obtenido mediante **los objetivos de desempeño**, que hace referencia a una tabla comparativa, para verificar si coincide con el resultado esperado. En los ejemplos de prueba unitaria, se clasifica una imagen de pollo en una fase específica, como Pre-inicio, Inicio, Desarrollo y Engorde.

En la columna "Resultado esperado", se especifica el resultado que se espera obtener en cada prueba. En la columna "Resultado obtenido", se registra el resultado real obtenido durante las pruebas. Luego, se puede evaluar si el resultado obtenido coincide con el resultado esperado y si la prueba se considera exitosa.

Tabla 4. Pruebas unitarias

Tipo de prueba	Escenario	Resultado esperado	Resultado obtenido	¿Pasa la prueba?
Prueba unitaria	Clasificar imagen de pollo en fase de Pre-inicio	 Fase de Pre-inicio	 Fase de Pre-inicio	Sí

Tipo de prueba	Escenario	Resultado esperado	Resultado obtenido	¿Pasa la prueba?
Prueba unitaria	Clasificar imagen de pollo en fase de Desarrollo	 Fase de Desarrollo	 Fase de Desarrollo	Sí
Prueba unitaria	Clasificar imagen de pollo en fase de Engorde	 Fase de Engorde	 Fase de Engorde	Sí

En la primera, segunda, tercera, cuarta prueba unitaria, se esperaba clasificar una imagen de pollo en cada fase de crecimiento como es: Pre-inicio, Inicio, Desarrollo y Engorde, en donde el resultado obtenido coincide con el resultado esperado, por lo que la prueba pasa.

En la prueba de aceptación, se esperaba obtener resultados coherentes y precisos para cada imagen en un conjunto diverso de imágenes de pollos. Si los resultados obtenidos cumplen con esta expectativa, la prueba pasa.

Entrenamiento, Validación y Pruebas

Conjunto de entrenamiento: Este conjunto se utilizó para entrenar el modelo. Contenía un conjunto de 1.166 imágenes de pollos en sus diferentes fases, con sus correspondientes etiquetas. El modelo fue ajustado a este conjunto durante el proceso de entrenamiento para aprender a reconocer los patrones y características relevantes de las imágenes.

Conjunto de validación: Este conjunto se utilizó para evaluar el rendimiento del modelo durante el entrenamiento y ajustar los hiperparámetros. Después de cada época de entrenamiento, en este caso 75 épocas, el modelo se evaluó en el conjunto de validación para medir su precisión y otros indicadores de rendimiento. En la Figura 24, se representa los valores de las 5 primeras épocas y las últimas 10 épocas (**véase en el Colab**).

```
Epoch 1/75
19/19 [==============================] - 33s 2s/step - loss: 0.1426 - accuracy: 0.9502 - val_loss: 0.2562 - val_accuracy: 0.9062
Epoch 2/75
19/19 [==============================] - 34s 2s/step - loss: 0.1362 - accuracy: 0.9622 - val_loss: 0.2760 - val_accuracy: 0.9531
Epoch 3/75
19/19 [==============================] - 33s 2s/step - loss: 0.1516 - accuracy: 0.9485 - val_loss: 0.2655 - val_accuracy: 0.9375
Epoch 4/75
19/19 [==============================] - 41s 2s/step - loss: 0.1560 - accuracy: 0.9467 - val_loss: 0.2327 - val_accuracy: 0.9531
Epoch 5/75
19/19 [==============================] - 32s 2s/step - loss: 0.1853 - accuracy: 0.9313 - val_loss: 0.2038 - val_accuracy: 0.9688
```

```
Epoch 66/75
19/19 [==============================] - 34s 2s/step - loss: 0.1177 - accuracy: 0.9605 - val_loss: 0.1956 - val_accuracy: 0.9531
Epoch 67/75
19/19 [==============================] - 33s 2s/step - loss: 0.1092 - accuracy: 0.9639 - val_loss: 0.3016 - val_accuracy: 0.9531
Epoch 68/75
19/19 [==============================] - 30s 2s/step - loss: 0.0744 - accuracy: 0.9794 - val_loss: 0.2768 - val_accuracy: 0.9688
Epoch 69/75
19/19 [==============================] - 32s 2s/step - loss: 0.0880 - accuracy: 0.9674 - val_loss: 0.1629 - val_accuracy: 0.9531
Epoch 70/75
19/19 [==============================] - 35s 2s/step - loss: 0.0995 - accuracy: 0.9639 - val_loss: 0.2405 - val_accuracy: 0.9688
Epoch 71/75
19/19 [==============================] - 30s 2s/step - loss: 0.0917 - accuracy: 0.9708 - val_loss: 0.1404 - val_accuracy: 0.9531
Epoch 72/75
19/19 [==============================] - 33s 2s/step - loss: 0.1088 - accuracy: 0.9588 - val_loss: 0.2077 - val_accuracy: 0.9219
Epoch 73/75
19/19 [==============================] - 33s 2s/step - loss: 0.0855 - accuracy: 0.9725 - val_loss: 0.1769 - val_accuracy: 0.9688
Epoch 74/75
19/19 [==============================] - 34s 2s/step - loss: 0.1042 - accuracy: 0.9639 - val_loss: 0.3169 - val_accuracy: 0.9688
Epoch 75/75
19/19 [==============================] - 32s 2s/step - loss: 0.1062 - accuracy: 0.9656 - val_loss: 0.2639 - val_accuracy: 0.9375
```

Figura 24. Épocas del modelo

Se presentaron las métricas de entrenamiento y validación en 75 épocas para un modelo de aprendizaje automático. Cada época representó una pasada completa cada vez para cada época. A continuación, se analiza las últimas 5 primeras épocas y las 10 ultimas épocas.

En la época de la primera a la quinta, el modelo se entrenó durante 32 a 34 segundos. La pérdida de entrenamiento fue de 0.14 a 0.18, lo que indicó que el modelo pudo ajustarse bien a los datos de entrenamiento. La precisión de entrenamiento fue del 93 a 96%, lo que significó que el modelo clasificó correctamente el 96.05% de los ejemplos en el conjunto de entrenamiento. La pérdida de validación fue de 0.20 a 0.27, indicando un rendimiento sólido en el conjunto de datos de validación con una precisión del 90 a 96%.

En la época 66, el modelo se entrenó durante 34 segundos. La pérdida de entrenamiento fue de 0.1177, lo que indicó que el modelo pudo ajustarse bien a los datos de entrenamiento. La precisión de entrenamiento fue del 96.05%, lo que significó que el modelo clasificó correctamente el 96.05% de los ejemplos en el conjunto de entrenamiento. La pérdida de validación fue de 0.1956, indicando un rendimiento sólido en el conjunto de datos de validación con una precisión del 95.31%.

En la época 67, el modelo se entrenó durante 33 segundos. La pérdida de entrenamiento fue de 0.1092, con una precisión de entrenamiento del 96.39%. Sin embargo, la pérdida de validación aumentó a 0.3016, indicando un rendimiento ligeramente inferior en el conjunto de datos de validación con una precisión del 95.31%.

En la época 68, el modelo se entrenó durante 30 segundos. La pérdida de entrenamiento fue de 0.0744, con una precisión de entrenamiento del 97.94%. La pérdida de validación disminuyó a 0.2768, con una precisión del 96.88% en el conjunto de datos de validación.

En la época 69, el modelo se entrenó durante 32 segundos. La pérdida de entrenamiento fue de 0.0880, con una precisión de entrenamiento del 96.74%. La pérdida de validación fue de 0.1629, con una precisión del 95.31% en el conjunto de datos de validación.

En la época 70, el modelo se entrenó durante 35 segundos. La pérdida de entrenamiento fue de 0.0995, con una precisión de entrenamiento del 96.39%. La pérdida de validación disminuyó a 0.2405, con una precisión del 96.88% en el conjunto de datos de validación.

En la época 71, el modelo se entrenó durante 30 segundos. La pérdida de entrenamiento fue de 0.0917, con una precisión de entrenamiento del 97.08%. La pérdida de validación disminuyó a 0.1404, con una precisión del 95.31% en el conjunto de datos de validación.

En la época 72, el modelo se entrenó durante 33 segundos. La pérdida de entrenamiento fue de 0.1088, con una precisión de entrenamiento del 95.88%. La pérdida de validación aumentó a 0.2077, con una precisión del 92.19% en el conjunto de datos de validación.

En la época 73, el modelo se entrenó durante 33 segundos. La pérdida de entrenamiento fue de 0.0855, con una precisión de entrenamiento del 97.25%. La pérdida de validación disminuyó a 0.1769, con una precisión del 96.88% en el conjunto de datos de validación.

En la época 74, el modelo se entrenó durante 34 segundos. La pérdida de entrenamiento fue de 0.1042, con una precisión de entrenamiento del 96.39%. La pérdida de validación aumentó a 0.3169, con una precisión del 96.88% en el conjunto de datos de validación.

En la época 75, el modelo se entrenó durante 32 segundos. La pérdida de entrenamiento fue de 0.1062, con una precisión de entrenamiento del 96.56%. La pérdida de validación fue de 0.2639, con una precisión del 93.75% en el conjunto de datos de validación.

En la Figura 25, se presenta un resumen de las métricas de entrenamiento y validación por época correspondientes a las últimas diez épocas del modelo.

Época	Duración	Pérdida de entrenamiento	Precisión de entrenamiento	Pérdida de validación	Precisión de validación
1/75	33s	0.1426	0.9502	0.2562	0.9062
2/75	34s	0.1362	0.9622	0.2760	0.9531
3/75	33s	0.1516	0.9485	0.2655	0.9375
4/75	41s	0.1560	0.9467	0.2327	0.9531
5/75	32s	0.1853	0.9313	0.2038	0.9688
----------	----------	----------	----------	----------	----------
66/75	34s	0.1177	0.9605	0.1956	0.9531
67/75	33s	0.1092	0.9639	0.3016	0.9531
68/75	30s	0.0744	0.9794	0.2768	0.9688
69/75	32s	0.0880	0.9674	0.1629	0.9531
70/75	35s	0.0995	0.9639	0.2405	0.9688
71/75	30s	0.0917	0.9708	0.1404	0.9531
72/75	33s	0.1088	0.9588	0.2077	0.9219
73/75	33s	0.0855	0.9725	0.1769	0.9688
74/75	34s	0.1042	0.9639	0.3169	0.9688
75/75	32s	0.1062	0.9656	0.2639	0.937

Figura 25. Comparación de épocas

Se evidenció una mejora progresiva en la pérdida de entrenamiento a lo largo de las épocas, lo que señaló un ajuste cada vez más preciso del modelo a los datos de entrenamiento. La precisión de entrenamiento también demostró un rendimiento elevado y consistente en todas las épocas, indicando una capacidad sólida para clasificar de manera precisa los ejemplos del conjunto de entrenamiento.

No obstante, la pérdida de validación y la precisión de validación exhibieron cierta variabilidad. En las primeras épocas, se observó una mayor pérdida de validación y una precisión ligeramente inferior. A medida que avanzó el proceso de entrenamiento, la pérdida de validación disminuyó gradualmente, y la precisión de validación se estabilizó. A continuación, se muestra un promedio de manera general de cada ítem.

- ✓ **Resumen de los datos de las épocas:** 1 a 75.
- ✓ **Duración promedio por época:** 32.3 segundos
- ✓ **Pérdida de entrenamiento promedio:** 0.1431
- ✓ **Precisión de entrenamiento promedio:** 0.9630

- ✓ **Pérdida de validación promedio:** 0.1656
- ✓ **Precisión de validación promedio:** 0.9510

Estos valores muestran el rendimiento promedio del modelo durante las últimas 10 épocas en términos de duración del entrenamiento, pérdida y precisión tanto para el conjunto de entrenamiento como para el de validación. De esta manera se puede obtener un promedio de **95.10%** de clasificación.

Conjunto de pruebas

Una vez que el modelo había sido entrenado y ajustado utilizando los conjuntos de entrenamiento y validación, se procedió a evaluar su rendimiento final utilizando el conjunto de pruebas. Este conjunto de datos se mantuvo completamente separado durante todo el proceso de desarrollo del modelo y se utilizó exclusivamente al final para evaluar su capacidad de reconocimiento del crecimiento del tamaño del pollo en imágenes nuevas y no vistas previamente. El conjunto de pruebas constaba de un total de 80 imágenes, 20 para cada fase del crecimiento del pollo. Las etiquetas de tamaño de crecimiento correspondientes a este conjunto se utilizaron para comparar las predicciones del modelo y calcular métricas de rendimiento, como la matriz de confusión.

Matriz de confusión mediante validación cruzada

Se calcularon diversas métricas de evaluación, como la precisión y el puntaje en porcentaje, para evaluar el rendimiento del modelo de reconocimiento de imágenes en la identificación del crecimiento del tamaño del pollo en diferentes fases, como "Pre-inicio", "Inicio", "Desarrollo" y "Engorde". Estas métricas proporcionaron información sobre la precisión y la capacidad del modelo para identificar correctamente cada fase de crecimiento.

En la Figura 26, se representaron visualmente las cuatro fases mencionadas ("Pre-inicio", "Inicio", "Desarrollo", "Engorde" y "No pertenece"), las cuales se aplicaron en el proceso de evaluación. Cada una de estas fases fue evaluada utilizando un conjunto de 20 imágenes, lo que resultó en un total de 100 imágenes evaluadas. Estas imágenes se seleccionaron de fuentes externas para garantizar que no estuvieran incluidas en el conjunto de entrenamiento del modelo. Esta selección se realizó con el objetivo de lograr una mayor precisión al aplicar la identificación del crecimiento del tamaño del pollo en cada fase.

	FASE 1 Pre-inicio	FASE 2 Inicio	FASE 3 Desarrollo	FASE 4 Engorde	No pertenece	PORCENTAJE
FASE 1 Pre-inicio	20	0	0	0	0	100.0%
FASE 2 Inicio	0	19	1	0	0	93.3%
FASE 3 Desarrollo	0	0	20	0	0	100.0%
FASE 4 Engorde	0	0	1	19	0	93.3%
No pertenece	0	0	0	0	20	100.0%
TOTAL						**97.32%**

Figura 26. Matriz de confusión

Durante las pruebas, se verificó que el modelo de reconocimiento de imágenes logró alcanzar una precisión del 97.32% en la identificación del crecimiento del tamaño del pollo. Esta métrica refleja el porcentaje de acierto en el entrenamiento del modelo al reconocer correctamente el tamaño del pollo en cada fase en las imágenes de prueba evaluadas. El resultado obtenido demuestra la capacidad del modelo para generalizar y realizar predicciones precisas en escenarios distintos a los del conjunto de entrenamiento.

Objetivo 2

Implementar el software para el control de crecimiento utilizando la metodología XP mediante el lenguaje de programación Python.

En esta etapa intervienen las siguientes fases de la Figura 17:

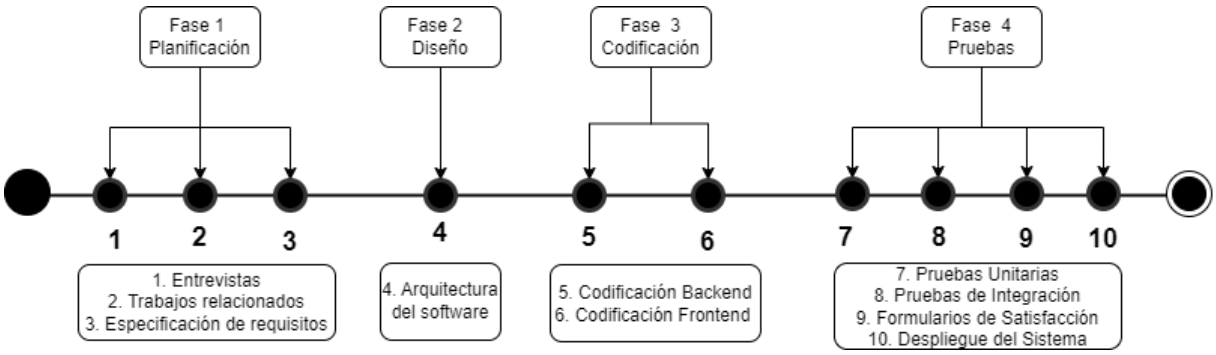


Figura 27. Proceso de desarrollo para el objetivo 2.

Fase 1: Planificación

Entrevistas a los expertos encargados

Se llevó a cabo la etapa de recolección de datos mediante **entrevistas** a expertos encargados en la avícola. Estas entrevistas tuvieron como objetivo obtener información relevante sobre las actividades realizadas y el manejo y registro de parámetros de la avícola. Se entrevistó al

dueño de la avícola, Pinto Añazco Alexis Jeovanny, quien es el actual representante de la granja "Las Acacias". Además, se realizó una entrevista al encargado de la empresa, Sr. Fernando Bravo, quien es responsable de todos los suministros para la granja. Durante esta entrevista, se emplearon preguntas abiertas y cerradas para obtener una amplia gama de información sobre el estado de la avícola.

Trabajos relacionados

Los cinco trabajos relacionados abordan diferentes aspectos de la industria avícola y comparten un enfoque en la implementación de tecnología para mejorar el control, monitoreo y eficiencia en la crianza de pollos. Los **trabajos relacionados** presentan casos de estudio en granjas avícolas reales, lo que demuestra la aplicabilidad y relevancia de las soluciones propuestas en situaciones prácticas. A continuación, se resumen algunas características generales de estos trabajos:

1. Automatización y Monitoreo: Todos los trabajos buscan automatizar y monitorear distintos aspectos del proceso de crianza de pollos en granjas avícolas. Para lograrlo, utilizan tecnología de hardware y software, como microcontroladores (Arduino y ESP32), sensores y actuadores.

2. Mejora de la Productividad: Los trabajos tienen como objetivo mejorar la productividad y eficiencia de las granjas avícolas, lo que puede resultar en una mayor producción y reducción de costos para los microempresarios y grandes industrias del sector.

3. Tecnología de Bajo Costo: Se destaca el uso de tecnología de bajo costo, como plataformas de desarrollo de código abierto (Arduino) y hardware económico (ESP32), lo que hace que las soluciones sean accesibles y viables para granjas con recursos limitados.

4. Monitoreo Ambiental: La mayoría de los trabajos se centran en el monitoreo de parámetros ambientales relevantes para la crianza de pollos, como humedad, temperatura e intensidad lumínica. Estos datos son fundamentales para tomar decisiones informadas y mejorar las condiciones técnicas en el proceso de crianza.

5. Enfoque en Tecnología IoT: Varios trabajos emplean la tecnología IoT (Internet de las cosas) para conectar dispositivos y sistemas, lo que permite el acceso remoto y en tiempo real a la información, así como la toma de decisiones basada en datos.

En su conjunto, estos estudios resaltan el enfoque en la implementación tecnológica para potenciar la industria avícola, subrayando la relevancia de la automatización, supervisión y la adopción de tecnologías asequibles con el propósito de alcanzar una producción de pollos más eficiente y económicamente beneficiosa.

Documento de especificación de requisitos de software con el estándar IEEE-830

Se elaboró un Documento de Especificación de Requisitos de Software siguiendo el estándar IEEE-830. En dicho documento, se establecieron las iteraciones planificadas para el desarrollo del software, y se definieron las Historias de Usuario correspondientes a cada una de estas iteraciones. Las Historias de Usuario describieron las funcionalidades y requerimientos específicos del sistema desde la perspectiva del usuario, proporcionando una visión clara y detallada de los objetivos y expectativas para cada fase del desarrollo. Este enfoque permitió una gestión efectiva del proyecto y una comunicación precisa entre los diferentes equipos involucrados en el proceso de desarrollo del software. A continuación, se detalla de manera general el **Documento de Especificación de Requisitos de Software.**

Objetivo del módulo de software

Se relaciona al objetivo general del Trabajo de Integración Curricular, que es “Desarrollar el software para el Control de Crecimiento en pollos de engorde en la granja Avícola “Las Acacias” en el Cantón Balsas.”.

Actores del módulo de software

El módulo de software tiene lo siguientes actores:

- **Usuario Administrador:**
  - Registrar los datos de la avícola.
  - Generar los reportes generales.
- **Usuario Galponero:**
  - Ingresar datos diarios en el control de crecimiento como, mortalidad diaria, recepción de alimentos balanceados, peso promedio.
  - Visualizar los diagnósticos de las anomalías del pollo
- **Usuario Veterinario:**
  - Registrar los diagnósticos de las anomalías del pollo.
  - Generar los reportes generales

Requerimientos Funcionales y no Funcionales

En esta fase actual, se detallan los siguientes aspectos: los requisitos funcionales y no funcionales, así como las historias de usuario (consultar el Anexo 3: Especificación de requisitos de software) véase en el **Anexo 3. Especificación de requisitos de software**) siguiendo el estándar IEEE 830. Estos elementos se derivaron a partir de las **entrevistas** llevadas a cabo, mediante las cuales se recopiló información valiosa de los individuos involucrados en la operación de la granja avícola "Las Acacias" en el Cantón Balsas. Estas

entrevistas proporcionaron un entendimiento sobre las actividades realizadas en la granja, así como el manejo y registro de diversos parámetros relacionados con la operación avícola.

Requisitos funcionales

En la Tabla 5, 6, 7, se muestra los requisitos funcionales de los tres roles del Software para el Control de Crecimiento en pollos de engorde en la granja Avícola “Las Acacias” en el Cantón Balsas.

Usuario Administrador

Tabla 5. Requisitos funcionales usuario Administrador

Identificador	Nombre
RF01	Inicio de sesión
RF02	Registro datos de la avícola
RF03	Generar reportes generales

Usuario Galponero

Tabla 6. Requisitos funcionales usuario Galponero

Identificador	Nombre
RF04	Inicio de sesión
RF05	Registrar control de crecimiento
RF06	Registrar mortalidad
RF07	Registrar alimentos
RF08	Registrar peso
RF09	Calcular crecimiento de pollos (edad)
RF10	Evaluar imagen (fase)
RF11	Buscar anomalías

Usuario Veterinario

Tabla 7. Requisitos funcionales usuario Veterinario

Identificador	Nombre
RF11	Ingresar información del diagnóstico de anomalías
RF12	Generar reportes generales

Requisitos no funcionales

En la Tabla 8, se muestra los requisitos no funcionales del Software para el Control de Crecimiento en pollos de engorde en la granja Avícola “Las Acacias” en el Cantón Balsas.

Tabla 8. Requisitos no funcionales de manera general

Identificador	Requisito	Descripción
RNF01	Rendimiento	El módulo de software debe proveer un tiempo de respuesta aceptable de 2 a 12 segundos aproximadamente.
RNF02	Usabilidad	El módulo de software debe proveer una interfaz amigable e intuitiva, y ser compatible con diversos navegadores web.
RNF03	Seguridad	El módulo de software debe garantizar la autentificación de usuario mendiante usuario y contraseña.

Historias de usuario del Software

En la Tabla 9, se muestra las historias de usuario de los tres roles del Software para el Control de Crecimiento en pollos de engorde en la granja Avícola “Las Acacias” en el Cantón Balsas.

Tabla 9. Historias de usuario

Identificador de la historia	Rol	Característica Funcionalidad	Razón / Resultado
HU01	Administrador	Inicio de sesión	Permitir al administrador iniciar sesión en el sistema para acceder a funcionalidades específicas de su rol.
HU02	Administrador	Registrar datos de la avícola	Permitir al administrador ingresar datos esenciales de la avícola, como lote y galpón, para mantener un registro organizado de la información.
HU03	Administrador, Veterinario	Generar reportes generales	Posibilitar a administradores y veterinarios generar reportes generales con información relevante sobre la avícola y su funcionamiento.
HU04	Galponero	Registrar control de crecimiento	Permitir al galponero registrar datos de control de crecimiento de los pollos, incluyendo mortalidad diaria, recepción de alimentación diaria y peso promedio.
HU06	Galponero	Registrar alimentación	Permitir al galponero registrar la información de alimentación de los pollos, incluyendo conductor, cantidad ingresada y consumida, fecha y hora, y saldo balanceado.
HU07	Galponero	Registrar peso	Permitir al galponero registrar el peso y la imagen de los pollos para evaluar su crecimiento en edad y fase.
HU08	Galponero	Evaluar imagen	Permitir al galponero evaluar imágenes de pollos en diferentes fases de desarrollo para identificar su tamaño y evaluar su crecimiento.
HU09	Veterinario	Ingresar información del diagnóstico de anomalías	Posibilitar al veterinario ingresar información detallada sobre el diagnóstico de anomalías en los pollos para su seguimiento y tratamiento.
HU10	Veterinario	Agregar enfermendad	Permitir agregar enfermedad en caso que exista alguna nueva en el campo de la avicultura.

Fase 2: Diseño

Documento de arquitectura del software

Modelo arquitectónico 4+1

Se elaboró el documento de arquitectura del software siguiendo el patrón arquitectónico 4+1, que consiste en un enfoque para describir la arquitectura de un sistema desde diferentes perspectivas o vistas. Cada una de estas vistas proporcionó una representación específica de la arquitectura, lo que permitió comprender mejor los diferentes aspectos del sistema.

En el caso particular, se desarrollaron modelos UML (Lenguaje de Modelado Unificado) para cada una de las vistas. UML es una notación estándar utilizada para visualizar y representar diagramas de distintos tipos en el campo de la ingeniería de software. A continuación, se detalla de manera general el esquema del **Documento la arquitectura del software 4 +1.**

Las cinco vistas principales en el patrón 4+1 fueron:

1. **Vista de Escenarios:** representó los casos de uso del sistema mediante diagramas de casos de uso, mostrando cómo los actores interactúan con el sistema para lograr sus objetivos.

Diagrama de casos de uso rol Administrador

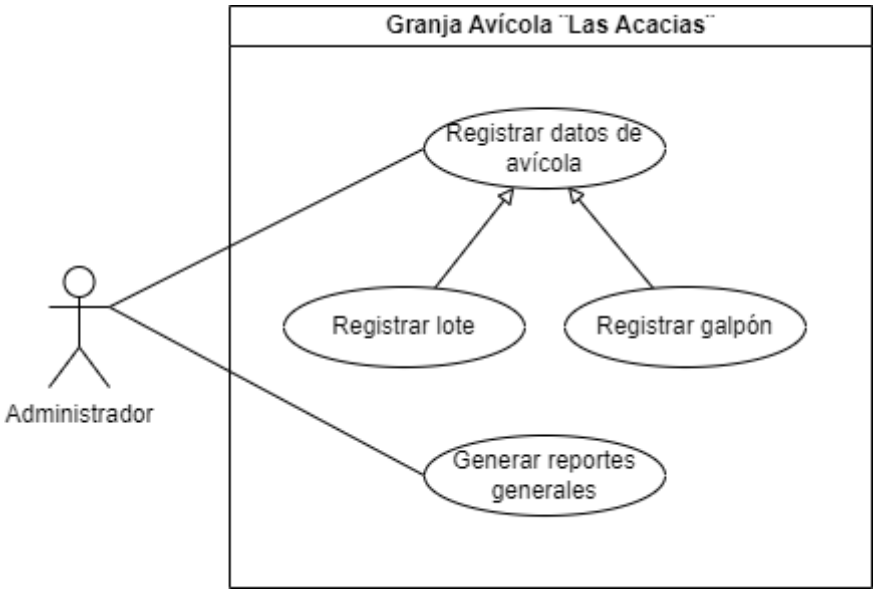


Figura 28. Diagrama de casos de uso

Descripción de casos de uso de inicio de sesión

Tabla 10. Requisito funcional Inicio de sesión

Identificación del requerimiento:	RF02
Nombre del Requerimiento:	Registrar datos de la avícola
Descripción del requerimiento:	El sistema permitirá al usuario Administrador registrar los datos esenciales de la avícola, incluyendo lote y galpón.
Dependencias:	S/N
Requerimiento NO funcional:	• RNF01 • RNF02
Prioridad del requerimiento:	Alta

2. **Vista Lógica:** describió la estructura y organización interna del sistema mediante diagramas de clases y diagramas de entidad-relación, mostrando las entidades y sus relaciones en el sistema.

Diagrama de clases

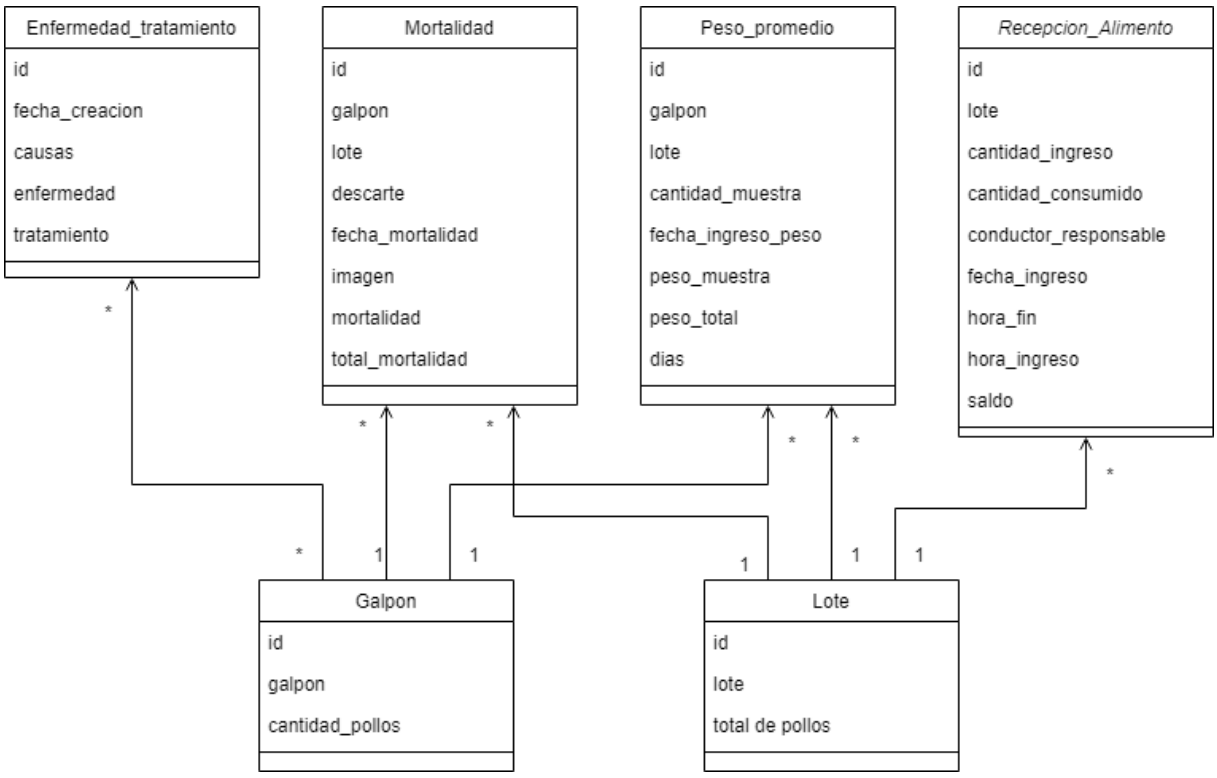


Figura 29.Diagrama de clases general

Diagrama entidad relación

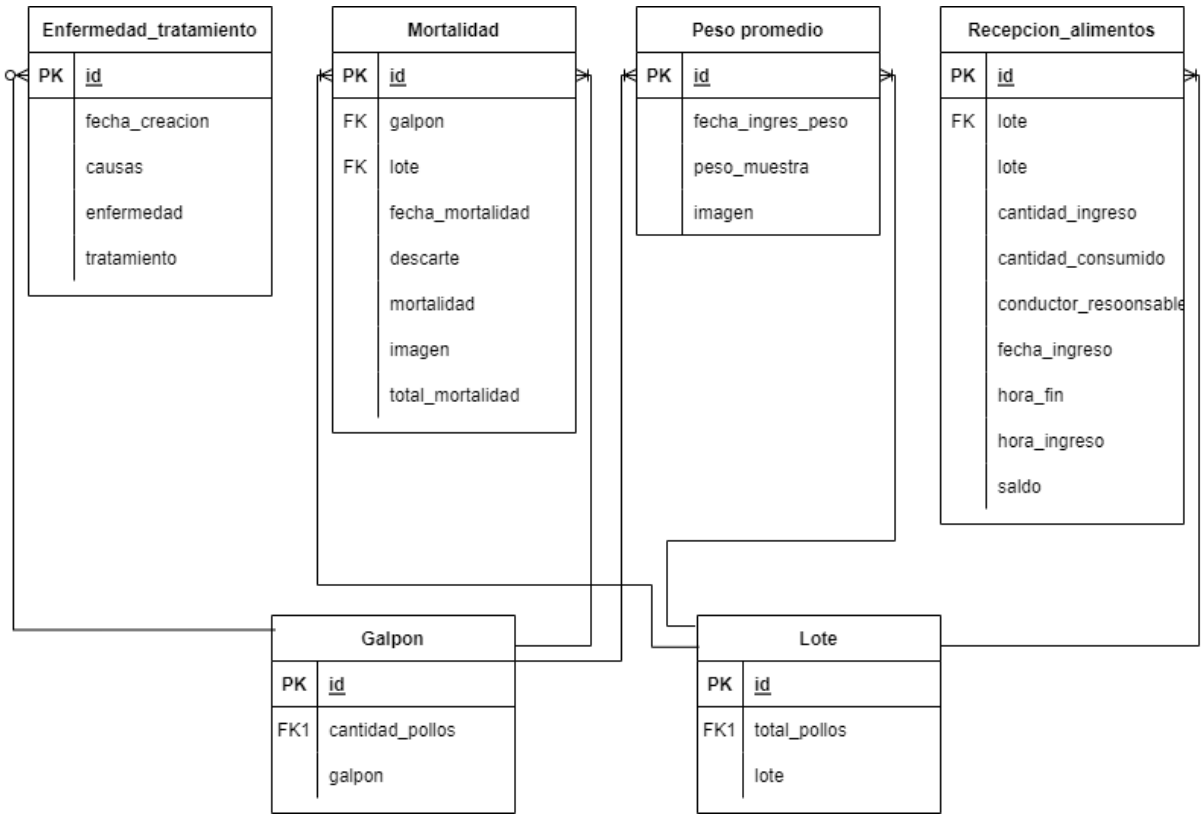


Figura 30. Diagrama entidad-relación general

Vista de Procesos: representó el comportamiento dinámico del sistema mediante diagramas de secuencia y diagramas de actividades, mostrando cómo los objetos del sistema interactúan y responden a eventos.

Diagrama de secuencia

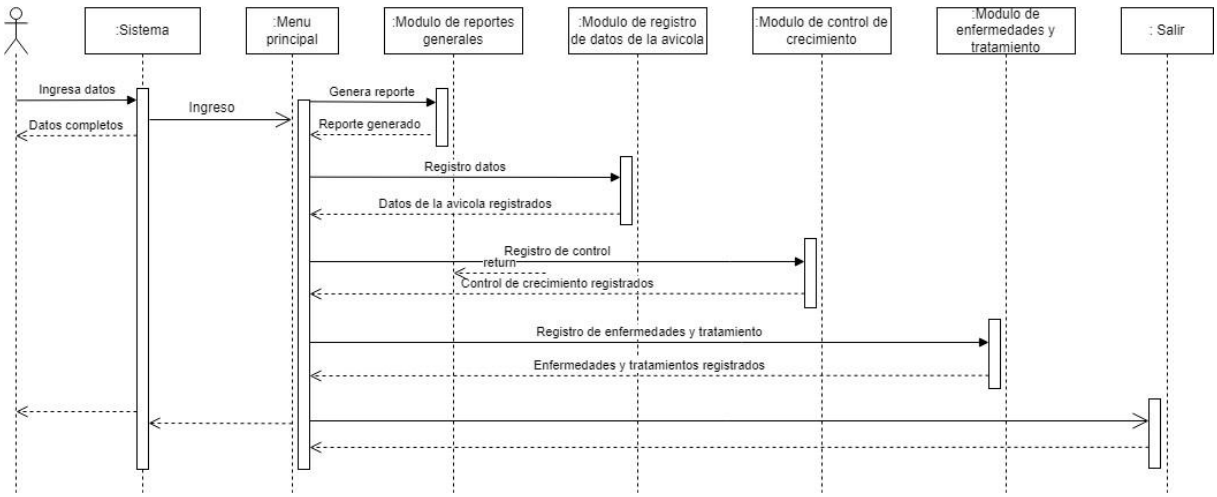


Figura 31. Diagrama de secuencia general

Diagrama de actividades

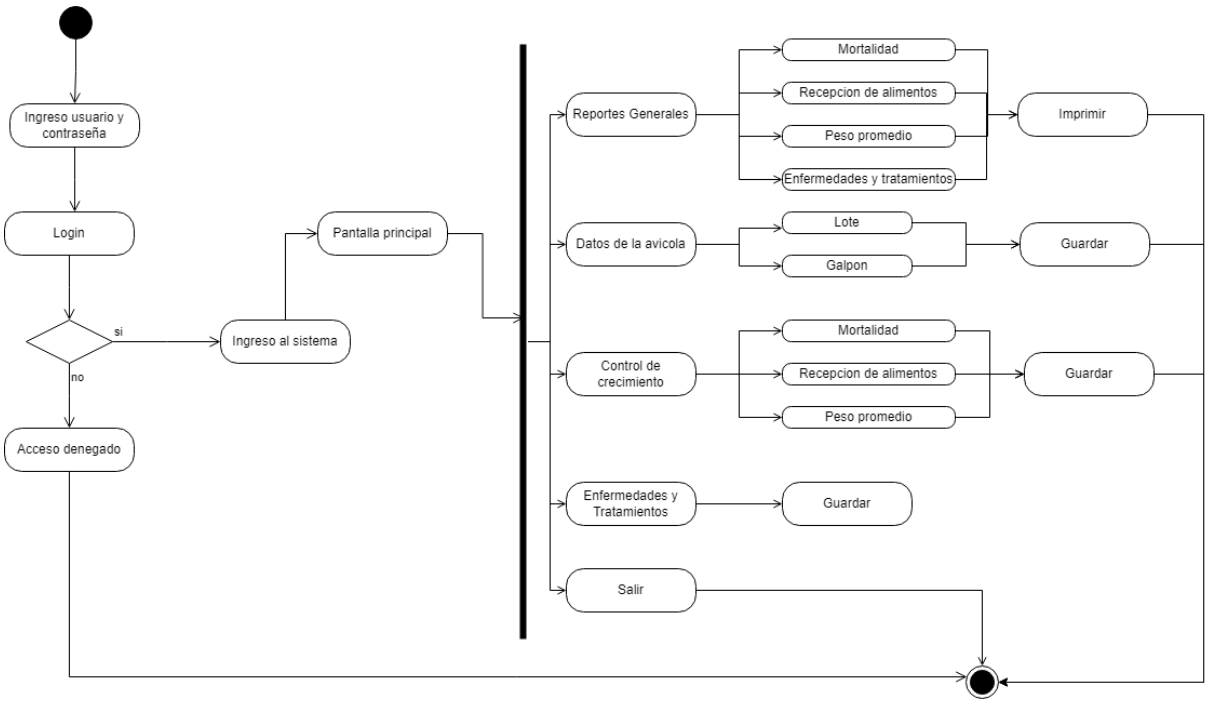


Figura 32. Diagrama de actividades general

3. **Vista de Desarrollo:** describió la estructura del software desde el punto de vista del código fuente, y diagrama de paquetes, mostrando la organización de los módulos y componentes del sistema.

Diagrama de paquetes

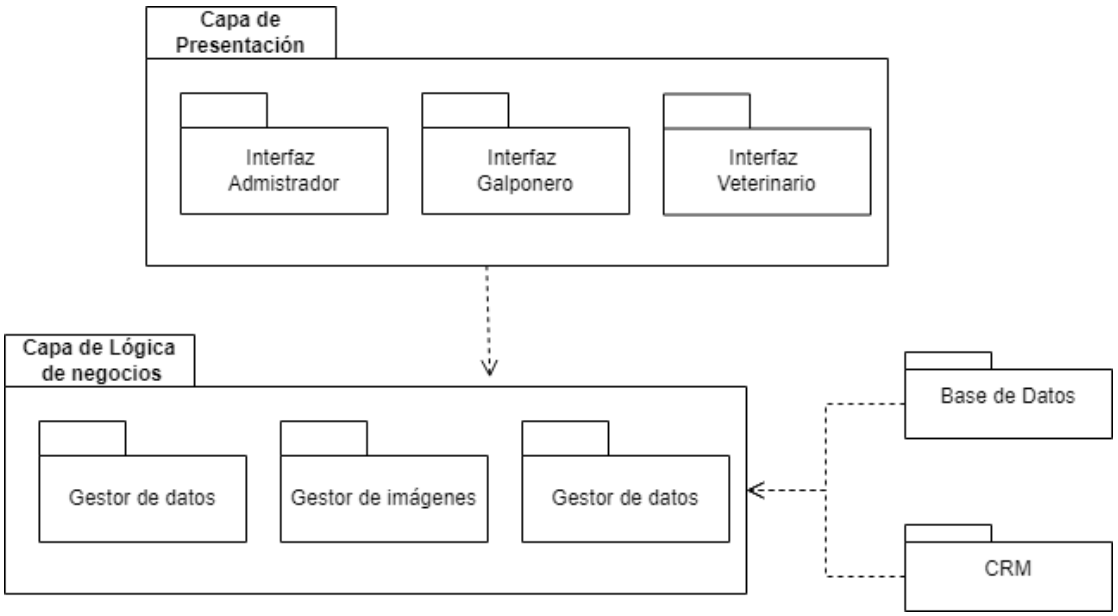


Figura 33. Diagrama de paquetes

4. **Vista Física:** representó la infraestructura y despliegue físico del sistema mediante diagramas de despliegue, mostrando cómo se distribuyen y conectan los elementos del sistema en el entorno de ejecución.

Diagrama de despliegue

Cada uno de estos modelos UML proporcionó una perspectiva única y valiosa para entender diferentes aspectos de la arquitectura del software. Además, el patrón 4+1 fomentó una visión holística del sistema al considerar múltiples vistas en conjunto, permite comprender cómo el software se implementa y se despliega en una arquitectura física real.

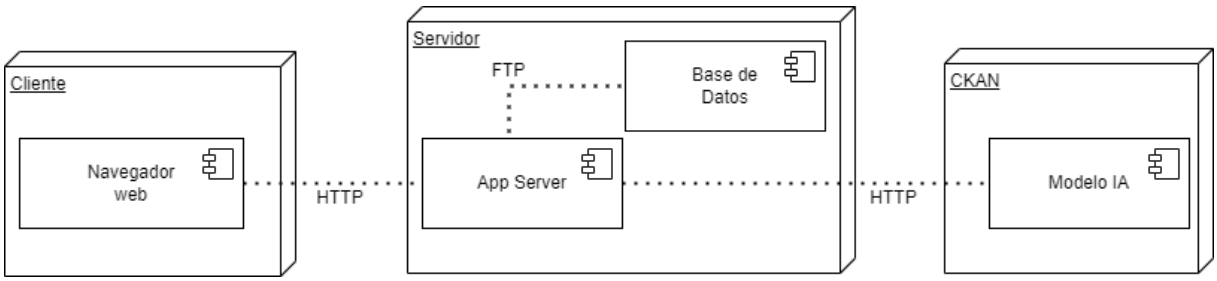


Figura 34. Diagrama de despliegue

Fase 3: Codificación

A continuación, se describen dos de las funcionalidades implementadas en el módulo, proporcionando detalles sobre el proceso de codificación llevado a cabo en cada iteración (véase sección **Codificación del Anexo 6**).

La codificación del dividió en dos partes:

Codificación del Backend

Para el desarrollo del backend, se empleó el framework de Django en combinación con RestFramework para implementar los Api Rest necesarios que permiten al software llevar a cabo los procesos específicos requeridos. La estructura del proyecto de backend sigue el diseño mostrado en la Figura 35.

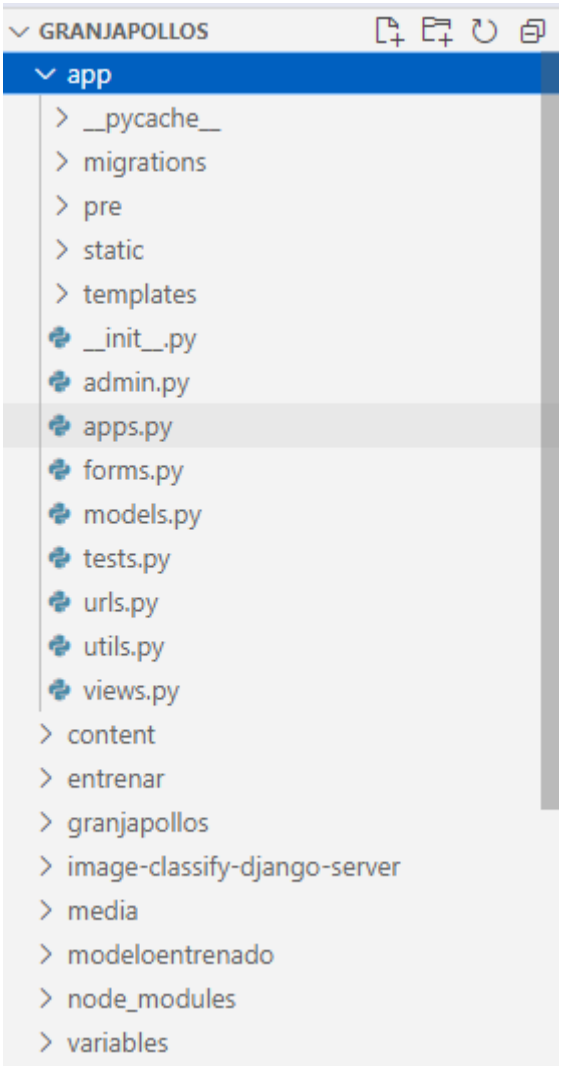


Figura 35. Estructura del proyecto backend

En base a esta estructura se detalla los archivos más relevantes dentro de la codificación:

Views: La Figura 36 muestra la estructura lógica del proceso para crear un Reporte. La "View", también conocida como controlador, tiene la función de solicitar al "Models" los datos necesarios para procesar las peticiones realizadas por el cliente.

```
class Reporte_pdf(View):

    def get(self, request, *args, **kwargs):
        vista = Reportes_generales.objects.all()
        data = {
    #         'count': vista.count(),
            'vista': vista
        }
        pdf = render_to_pdf('reporte_veterinario/Imprimir.html', data)
        return HttpResponse(pdf, content_type='application/pdf')
```

Figura 36. Estructura API del Reporte

Models: El término "Models" se refiere generalmente a una tabla de la base de datos. En la Figura 37, se pueden observar los atributos que componen los "Models", los cuales gestionan automáticamente las conversiones de tipos de datos para la base de datos que se está utilizando.

```python
class Mortalidad(models.Model):
    lote= models.ForeignKey(Lote, on_delete=models.CASCADE)
    galpon = models.ForeignKey(Galpon, on_delete=models.CASCADE)
    fecha_mortalidad = models.DateField(verbose_name='Fecha Mortalidad',null=True, blank=True)
    mortalidad = models.IntegerField(verbose_name='Cantidad de Mortalidad')
    descarte = models.IntegerField(verbose_name='Cantidad de Descarte')
    total_moratalidad = models.IntegerField(verbose_name='Total de Mortalidad existente',
    null=True, blank=True)
    imagen = models.FileField(upload_to="Mortalidadfile/")
    @property

    def valor_promedio(self):
        return (self.mortalidad + self.descarte)

    def save(self):
        self.total_moratalidad = self.valor_promedio
        # llamamos al metodo super que permite capturar todos lo cambios en una clase
        super(Mortalidad, self).save()

    def __str__(self):
        return f'{self.fecha_mortalidad}'
```

Figura 37. Estructura del modelo mortalidad

Urls: El archivo "Urls" especifica la relación entre una vista concreta y la URL en la que aparece. En la Figura 38, se muestra la conexión de cada ruta con las vistas a las que pertenecen.

```python
urlpatterns = [
    path('', home, name="home"),
    path('galponero/', galponero, name="galponero"),
    path('admin1/', admin1, name="admin1"),
    path('veterinario/', veterinario, name="veterinario"),
    path('lote_galpon/', lote_galpon, name="lote_galpon"),
    path('m_crecimiento/', m_crecimiento, name="m_crecimiento"),
    path('m_reportes_generales/', m_reportes_generales, name="m_reportes_generales"),
    path('accounts/', include('django.contrib.auth.urls')),
    path('imprimir_pdf/', imprimir_pdf, name='imprimir_pdf'),
    path('agReporte/', agReporte, name="agReporte"),
    path('agMortalidad/', agMortalidad, name="agMortalidad"),
    path('vista_reporte/', vista_reporte, name= "vista_reporte"),
    path('reporte/', views.Reporte_pdf.as_view(), name='reporte'),
    path('edReporte/<id>/', edReporte, name="edReporte"),
    path('verMortalidad/', verMortalidad, name= "verMortalidad"),
    path('edMortalidad/<id>/', edMortalidad, name= "edMortalidad"),
    path('edPredict2/<id>/', edPredict2, name= "edPredict2"),
    path('edrecepcion/<id>/', edrecepcion, name= "edrecepcion"),
```

Figura 38. Rutas direccionar a las vistas

Codificación del Frontend

Se utilizó el marco Django para convertir los datos en una interfaz gráfica para que el usuario los vea. En la Figura 39, visualizar la estructura del Frontend.

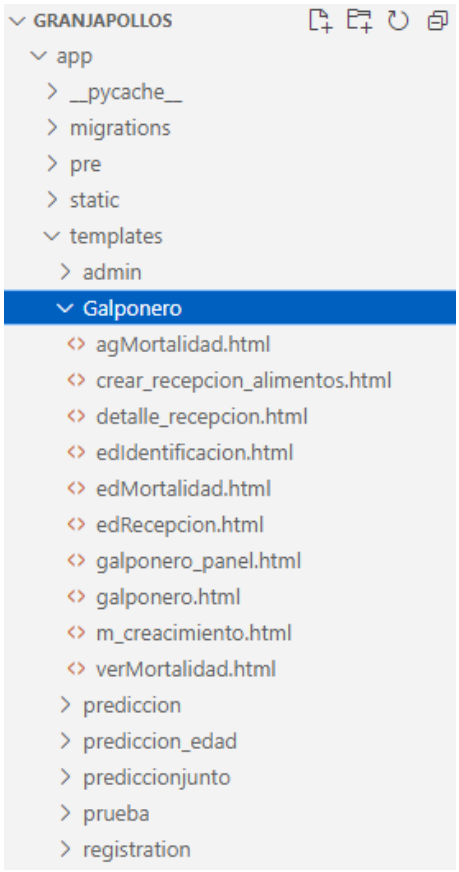


Figura 39. Estructura del Frontend

Fase 4: Pruebas

Para la ejecución de las pruebas se base en la técnica del muestreo por conveniencia ya que Software para el Control de Crecimiento en pollos de engorde es específicamente para la granja Avícola “Las Acacias” la cual está conformado por 3 personas mismos que realizaron las pruebas.

En la Tabla 11, se muestra los participantes que realizaron las pruebas.

Tabla 11. Participantes en ejecución de pruebas.

Nombre	Rol	Cargo
Fernando Bravo	Administrador	Jefe de la avícola
Jeovanny Pinto	Galponero	Representante legal
Mauricio Ramirez	Veterinario	Veterinario

Los anexos **7, 8** y **9,** incluyeron el Plan de Pruebas Unitarias, el Plan de Pruebas de Integración y el Plan de Pruebas Funcionales, respectivamente. Existe la posibilidad de que este no sea el caso. Proporcionaron una estructura para las diferentes fases de prueba. Las estrategias, enfoques y casos de prueba específicos para cada nivel de prueba fueron establecidos por los planes, asegurando que se abordaran tanto los aspectos individuales de los componentes como su correcta interacción y funcionamiento. Asimismo, el **Anexo 10**, correspondiente al Formulario de Satisfacción, permitió obtener valiosa retroalimentación de los usuarios finales y partes interesadas, lo que contribuyó a la mejora continua del producto y a la adaptación a las necesidades del cliente.

El Plan de Pruebas Unitarias

El **plan de pruebas unitarias** generadas para el software definió los casos de prueba específicos para validar el correcto funcionamiento de cada componente, asegurando que cumpliera con los requisitos y expectativas establecidos. Las pruebas unitarias permitieron identificar errores y defectos tempranamente en el proceso de desarrollo, facilitando su corrección y mejorando la calidad del código. En la Tabla 12, se procedió a generar la automatización de 16 pruebas unitarias en la que se observa el estado de aceptación con el tiempo de ejecución de cada una y el total.

Tabla 12. Casos de pruebas unitarias

Número del Caso de Prueba	Requisito / Historia de Usuario	Componente	Descripción de lo que se probará	Prerrequisitos
CP01	RF01: Inicio de sesión	Inicio de sesión	Ingresar credenciales válidas	Usuario administrador registrado en el sistema
CP02	RF01: Inicio de sesión	Inicio de sesión	Ingresar credenciales inválidas	-
CP03	RF02: Registrar datos de la avícola	Registrar datos de la avícola	Registrar datos de la avícola correctamente	Usuario administrador registrado en el sistema
CP04	RF03: Generar reportes generales	Generar reportes generales	Generar reportes de mortalidad	Usuario administrador o veterinario registrado en el sistema
CP05	RF03: Generar reportes generales	Generar reportes generales	Generar reportes de alimentación	Usuario administrador o veterinario registrado en el sistema
CP06	RF03: Generar reportes generales	Generar reportes generales	Generar reportes de peso	Usuario administrador o veterinario registrado en el sistema
CP07	RF03: Generar reportes generales	Generar reportes generales	Generar reportes de enfermedades	Usuario administrador o veterinario registrado en el sistema
CP08	RF03: Generar reportes generales	Generar reportes generales	Generar reportes de tratamientos	Usuario administrador o veterinario registrado en el sistema

Número del Caso de Prueba	Requisito / Historia de Usuario	Componente	Descripción de lo que se probará	Prerrequisitos
CP10	RF05: Registrar mortalidad	Registrar mortalidad	Registrar datos de mortalidad correctamente	Usuario galponero registrado en el sistema
CP11	RF06: Registrar alimentación	Registrar alimentación	Registrar datos de alimentación correctamente	Usuario galponero registrado en el sistema
CP12	RF07: Registrar peso	Registrar peso	Registrar datos de peso correctamente	Usuario galponero registrado en el sistema
CP13	RF08: Calcular crecimiento de pollos	Calcular crecimiento de pollos	Calcular crecimiento de pollos correctamente	Usuario galponero registrado en el sistema
CP14	RF09: Evaluar imagen	Evaluar imagen	Evaluar imágenes de pollos correctamente	Usuario galponero registrado en el sistema
CP15	RF10: Buscar anomalías	Buscar anomalías	Buscar anomalías correctamente	Usuario galponero registrado en el sistema
CP16	RF11: Ingresar información del diagnóstico de anomalías	Ingresar información del diagnóstico de anomalías	Ingresar información del diagnóstico correctamente	Usuario veterinario registrado en el sistema

El Plan de Pruebas de Integración

Se abordó la verificación de la correcta interacción y comunicación entre los diferentes componentes y módulos del software. El plan de pruebas de integración se enfocó en probar la integración de las unidades previamente validadas en el contexto del sistema completo. Se definieron escenarios de prueba que cubrieron las interacciones entre los módulos y se verificó que los datos fluyeran adecuadamente entre ellos. Las pruebas de integración fueron fundamentales para identificar problemas y conflictos que pudieron surgir cuando los componentes se combinaron, lo que permitió realizar ajustes y correcciones antes de avanzar en el desarrollo. En la Tabla 13, se muestra los 11 casos de prueba de integración, los que se aplican para asegurarse de que los diferentes componentes o módulos del sistema funcionen correctamente cuando se combinan y se interactúan entre sí.

Tabla 13. Casos de pruebas de integración

Número del Caso de Prueba	Componente	Descripción de lo que se probará	Prerrequisitos
CP01	Inicio de sesión	Verificar inicio de sesión con credenciales válidas e invalidad	• El sistema debe estar en funcionamiento. • Debe existir un usuario administrador registrado en el sistema con credenciales válidas e invalidas.

Número del Caso de Prueba	Componente	Descripción de lo que se probará	• Prerrequisitos
CP03	Generar reportes generales	Verificar generación de reportes de mortalidad, alimentación, peso, enfermedades y tratamientos	• El sistema debe estar en funcionamiento. • El usuario Administrador o Veterinario debe haber iniciado sesión en el sistema.
CP04	Registrar control de crecimiento	Verificar registro correcto de control de crecimiento de pollos	• El sistema debe estar en funcionamiento. • El usuario Galponero debe haber iniciado sesión en el sistema.
CP05	Registrar mortalidad	Verificar registro correcto de datos de mortalidad	• El sistema debe estar en funcionamiento. • El usuario Galponero debe haber iniciado sesión en el sistema
CP06	Registrar alimentación	Verificar registro correcto de datos de alimentación	• El sistema debe estar en funcionamiento. • El usuario Galponero debe haber iniciado sesión en el sistema.
CP07	Registrar peso	Verificar registro correcto de datos de peso	• El sistema debe estar en funcionamiento. • El usuario Galponero debe haber iniciado sesión en el sistema
CP08	Calcular crecimiento de pollos	Verificar cálculo correcto de crecimiento de pollos	• El sistema debe estar en funcionamiento. • El usuario Galponero debe haber iniciado sesión en el sistema
CP09	Evaluar imagen	Verificar evaluación correcta de imágenes de pollos	• El sistema debe estar en funcionamiento. • El usuario Galponero debe haber iniciado sesión en el sistema
CP10	Buscar anomalías	Buscar anomalías	• El sistema debe estar en funcionamiento. • Verificar búsqueda exitosa de anomalías en el crecimiento
CP11	Ingresar información diagnóstico de anomalía	Ingresar información del diagnóstico de anomalías	• El sistema debe estar en funcionamiento. • Verificar ingreso correcto de información de diagnóstico

El Formulario de Satisfacción

Fue utilizada para recopilar retroalimentación y opiniones de los usuarios finales y partes interesadas sobre el software desarrollado. Los formularios de satisfacción permitieron evaluar la experiencia del usuario, la facilidad de uso y la satisfacción general con el producto entregado. Al obtener la opinión directa de los usuarios, se pudieron identificar áreas de mejora y oportunidades de optimización en el software. La información recopilada a través del Formulario de Satisfacción resultó invaluable para realizar ajustes y mejoras continuas, garantizando la entrega de un producto que satisficiera plenamente las necesidades y expectativas del cliente. En la Tabla 14, se muestra un ejemplo del formulario de satisfacción en este caso de funcionalidad, se aplicó para evaluar y recopilar comentarios de los usuarios sobre cómo perciben y utilizan las diversas características y funcionalidades del software.

Tabla 14. Formulario de Satisfacción de Funcionalidad

UNIVERSIDAD NACIONAL DE LOJA

Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables

Carrera de Ingeniería en Computación

FORMULARIO DE SATISFACCIÓN
FUNCIONALIDAD

Como usuario final del sistema de gestión avícola, deberá utilizar las siguientes afirmaciones para evaluar el software para el Control de Crecimiento en pollos de engorde en el Granja Avícola Las Acacias del Cantón Balsas, para esto deberá de responder las siguiente preguntas utilizando una escala de 5 puntos, cuando 5 es la mayor, lo cual significa que usted está Totalmente de Acuerdo con la experiencia en la navegación, 4- Algo de Acuerdo, 3- Ni de Acuerdo Ni en desacuerdo, 2- Algo en Desacuerdo, y 1- Totalmente en Desacuerdo.

	1- Totalmente en Desacuerdo.	**2- Algo en Desacuerdo.**	**3- Ni de Acuerdo Ni en desacuerdo.**	**4- Algo de Acuerdo.**	**5- Totalmente de Acuerdo.**
¿Al momento de ingresar por primera vez al sistema lo logró desde el primero intento?					
¿Al momento de realizar consultas dentro del sistema, se generan de manera rápida y ágil?					
¿Está usted de acuerdo, en la forma en que están divididosy organizados los módulos dentro del sistema?					
¿En general, se encuentra usted satisfecho con la implementación del sistema avícola?					

Despliegue del Sistema

Documentación del Proceso de Despliegue del Sistema

Objetivo:

Esta guía proporciona instrucciones detalladas para implementar y levantar el sistema en un entorno Ubuntu 22.04.2 LTS.

Alcance:

Esta documentación cubre el proceso de configuración y despliegue del sistema utilizando la herramienta screen y el comando runserver en el servidor Ubuntu

Requisitos Previos:

- Un servidor con Ubuntu 22.04.2 LTS instalado.
- Acceso de administrador al servidor.
- Conexión a Internet estable.
- Familiaridad con los fundamentos de la línea de comandos en entornos Linux.

Paso 1: Preparación del Entorno

1. Accede al servidor Ubuntu utilizando tus credenciales de administrador.
2. Verifica la información del sistema operativo, incluyendo la versión de Ubuntu, características de CPU, memoria y espacio de almacenamiento.

Paso 2: Levantamiento del Sistema

1. Para asegurar la continuidad de la operación incluso después de cerrar la sesión, emplearemos la utilidad "screen". Crea una nueva sesión de screen con el comando:

screen -S nombre_de_sesion

2. Dentro de la sesión screen, inicia el sistema mediante el siguiente comando:

runserver manage.py runserver 49.13.53.244:80

Esto activará el sistema y lo pondrá en funcionamiento en la dirección IP 49.13.53.244 y el puerto 80.

Paso 3: Verificación del Sistema

1. Abre un navegador web en tu dispositivo local.
2. Ingresa la dirección IP del servidor (49.13.53.244) seguida de ":80".
3. Abre un navegador web y accede a la dirección IP del servidor seguida por el puerto 80 (por ejemplo, http://49.13.53.244:80).
4. El resultado del mismo se puede observar en la siguiente dirección web: https://crianzapollosbalsas.com/
5. Verifica que el sistema se haya levantado correctamente y que puedas acceder a él desde el navegador.
6. Si el proceso de despliegue ha sido exitoso, deberías ver la interfaz o página web correspondiente del sistema.

Finalmente se generó el manual de usuario:

- Instructivo para el usuario disponible en el Anexo 11. Manual de Usuario.

7.Discusión

7.1 Desarrollo de la propuesta alternativa

El Proyecto de Trabajo de Integración Curricular, denominado “Software para el Control de Crecimiento en pollos de engorde en la granja Avícola “Las Acacias” en el Cantón Balsas”, busca agilizar el control de crecimiento en la producción de pollos de engorde. Fue elaborado para satisfacer los objetivos establecidos.

7.1.1 Objetivo 1: Construir un modelo de reconocimiento de imágenes mediante visión artificial para la identificación del crecimiento del tamaño del pollo.

La tarea de reconocimiento de imágenes para la identificación del crecimiento del tamaño de los pollos fue un área que, hasta esa fecha, había carecido de estudios previos y trabajos relacionados. Aunque aún no se han desarrollado sistemas de reconocimiento de imágenes específicamente para el control de crecimiento en pollos de engorde, varios proyectos relacionados han demostrado la viabilidad de implementar tecnologías innovadoras en la industria avícola (**sección 4.5. Trabajos relacionados**). Estos proyectos ofrecen una amplia visión sobre la automatización, monitoreo y control en la industria avícola, abordando temas como gestión de procesos, monitoreo ambiental, automatización de sistemas de alimentación y temperatura, e integración de tecnologías de hardware y software para mejorar la eficiencia en las granjas avícolas.Esta ausencia de investigación previa subrayó la singularidad y la novedad de la propuesta y resaltó la necesidad de abordar ese vacío en la literatura científica.

Para afrontar ese desafío inexplorado, se tomó la decisión de emplear la arquitectura MobileNet V2 [49] de redes neuronales convolucionales ya que sobresale entre las arquitecturas comparadas (**véase en el Anexo 11**) gracias a su tiempo de procesamiento rápido de 7.1 segundos, su alta precisión y su versatilidad, siendo capaz de manejar un rango amplio de imágenes, desde 500 hasta 10,000, con buenos resultados. Además, su tamaño del modelo relativamente pequeño y su disponibilidad de pre-entrenamiento lo hacen ideal para implementaciones en dispositivos móviles y embebidos, ofreciendo un equilibrio óptimo entre rendimiento y eficiencia de recursos.

A pesar de las limitaciones derivadas de la cantidad restringida de imágenes y otros desafíos inherentes, como la falta de una base de datos en línea, lo que requería el desarrollo de una propia ya que se poseía una base inicial de 281 imagenes. Sin embargo, para superar este obstáculo, se utilizó Copilot para generar nuevas imágenes a partir de una imagen existente, donde se obtuvo 520 imágenes, además se utilizó una clase ImageDataGenerator de TensorFlow para aumentar datos de imágenes con varios parámetros de aumento de datos, como rotación, cambio de ancho y alto, sesgo, y zoom, donde se obtuvo 1,166 imágenes, para mejorar la precisión del sistema de reconocimiento de imágenes.

Estas acciones demostraron ser fundamentales para alcanzar los objetivos del proyecto y maximizar su impacto en la industria avícola local, el enfoque demostró ser altamente efectivo. Se logró alcanzar un promedio de 95.10% de clasificación y un promedio del 97.32% en la identificación del proceso de crecimiento de los pollos. Ese resultado resaltó la potencia de la metodología adaptativa, especialmente diseñada para el contexto único de la cría de pollos de engorde.

7.1.2 Objetivo 2: Implementar el software para el control de crecimiento utilizando la metodología XP mediante el lenguaje de programación Python.

Las fases de la metodología XP permitieron desarrollar el TIC. En la fase de planificación, aunque existen varias técnicas de recolección de datos, las cuales pudieron haber ayudado a la obtención de requerimientos conformes al control de crecimiento de pollos de engorde en la granja avícola las Acacias, sin embargo, para el cumplimiento de esta tarea se utilizó la entrevista. El resultado final consistió en la creación del documento de **Especificación de Requerimientos de Software** siguiendo las pautas del estándar IEEE-830. Asimismo, la revisión de los trabajos relacionados, como se describe en la **Sección 4.5 Trabajos Relacionados**, proporcionó una amplia perspectiva de las soluciones tecnológicas aplicadas en granjas avícolas.

Para reforzar la fase de diseño del sistema, se utilizó **el documento de arquitectura de software,** modelo arquitectónico 4+1 el cual permitió modelar y documentar el diseño del software para un mejor entendimiento en base a 5 vistas bien definidas. A partir de este punto, la etapa de Codificación se llevó a cabo utilizando el lenguaje de programación Python (ver **Anexo 5 y Anexo 6**), el cual facilitó en gran medida, ya que se eligió los módulos de forma alternativa para llevar a cabo la implementación de los mismos. La ejecución de las historias de usuario planificadas, en alineación con las iteraciones definidas, demostró un compromiso constante con el desarrollo progresivo del software.

Adicionalmente, en la fase de prueba, se llevó a cabo mediante diversas pruebas como: **pruebas unitarias, pruebas de integración y formularios de satisfacción**, que permitieron garantizar que el software funciona correctamente y cumple con las especificaciones definidas en el documento de Especificación de Requerimientos.

La implementación de un software basado en un modelo de reconocimiento de imágenes mediante visión artificial, desarrollado con la metodología XP y el lenguaje de programación Python, agilizó el control de crecimiento en la producción de pollos de engorde en la granja avícola "Las Acacias" del Cantón Balsas. Este sistema automatizado proporcionó mediciones precisas y continuas del tamaño de los pollos, eliminando los retrasos y problemas asociados con el manejo manual de la información y el tiempo, reduciendo la necesidad de intervención

humana y disminuyendo los márgenes de error. Además, permitió recolectar en tiempo real información crucial como la mortalidad, alimentación, medicamentos y peso, facilitando la toma de decisiones más acertadas gracias a su precisión y rapidez. Con tiempos de respuesta promedio de 6 segundos, superando considerablemente la meta inicial de 12 segundos, este sistema optimizó los recursos y mejoró la eficiencia general de la producción avícola. El monitoreo constante y preciso permitió detectar rápidamente problemas de salud o crecimiento, interviniendo tempranamente y reduciendo la mortalidad de los pollos, lo cual contribuyó a un manejo más eficaz y rentable de la granja.

Por lo tanto, es importante mencionar que la pregunta de investigación planteada para este TIC queda respondida satisfactoriamente.

8. Conclusiones

Una vez culminado el Proyecto de Integración Curricular, se concluye que:

- El presente estudio demuestra la viabilidad y efectividad de la construcción de un modelo de reconocimiento de imágenes basado en visión artificial para la identificación del crecimiento del tamaño de los pollos de engorde. La ausencia previa de investigaciones en este campo resalta la originalidad y pertinencia de la propuesta. La elección de la arquitectura MobileNet V2 y la construcción de una base de datos personalizada demostraron ser estrategias exitosas para abordar este desafío. La obtención de un promedio de 95.10% de clasificación y un promedio del 97.32% en la identificación del proceso de crecimiento de los pollos refuerza la utilidad práctica y el potencial de esta metodología adaptativa.
- El uso de técnicas de aumento de datos, como la generación de imágenes con Copilot y el uso de la clase ImageDataGenerator de TensorFlow, fue crucial para superar las limitaciones derivadas de la cantidad restringida de imágenes disponibles y mejorar la precisión del sistema de reconocimiento de imágenes, ya que en un inicio se tenia 281 imágenes, y al final 1.166 imágenes. Esto subraya la importancia de la creatividad y la exploración de diversas estrategias para mejorar la calidad de los datos y, por ende, el rendimiento de los modelos de aprendizaje automático.
- La selección de la entrevista como técnica principal para la obtención de requerimientos demostró ser una elección acertada. La captura directa de información de las partes interesadas permitió comprender plenamente los aspectos esenciales del sistema. La transformación de los resultados de las entrevistas en documentos técnicos sólidos, como el estándar IEEE-830, garantizó la articulación precisa y completa de los requerimientos, sentando las bases para el diseño del sistema.
- La implementación exitosa de la metodología XP y el uso de Python en el desarrollo del software para el control de crecimiento de pollos de engorde garantizaron el cumplimiento de los objetivos del proyecto. Las prácticas ágiles, como la planificación continua y las pruebas frecuentes, permitieron un desarrollo adaptable y conforme a los requisitos del usuario. la aplicación del modelo arquitectónico 4+1, permitió representar y documentar de manera comprehensiva la arquitectura del sistema.

9. Recomendaciones

Una vez culminado el Proyecto de Integración Curricular, se recomienda que:

- Dado que la cantidad limitada de imágenes impactó las capacidades del modelo, se recomienda la continuación de la captura y recopilación de imágenes para expandir la base de datos. Una base de datos más amplia permitirá una mayor variabilidad y generalización del modelo, lo que podría mejorar aún más su precisión en la identificación del crecimiento.
- Aunque la arquitectura MobileNet V2 demostró ser efectiva, sería recomendable explorar otras arquitecturas de modelos de redes neuronales convolucionales. La experimentación con diferentes arquitecturas podría proporcionar insights adicionales sobre cuál es la más adecuada para abordar esta tarea específica.
- Considerando el éxito obtenido al utilizar la entrevista como la técnica principal para obtener requerimientos, se recomienda que en futuros proyectos se continúe dando prioridad a la interacción directa con las partes interesadas. No obstante, sería beneficioso complementar esta técnica con métodos adicionales, como encuestas o grupos de enfoque, para asegurar una recopilación de datos aún más integral y una validación adicional de los resultados obtenidos.
- La decisión de adoptar la metodología XP (Extreme Programming) ha resultado crucial para el logro exitoso del objetivo de implementar el software de control de crecimiento. Esta metodología ha facilitado un proceso organizado que ha englobado la planificación, el diseño, la codificación y las pruebas. Además, la integración del documento de arquitectura de software, modelo arquitectónico 4+1 ha promovido una mayor claridad y comunicación durante el desarrollo del sistema. Esta combinación de enfoques metodológicos y arquitectónicos ha contribuido significativamente a la calidad y eficiencia del proceso de desarrollo en su totalidad.

10. Bibliografía

[1] CONAVE, “CONAVE presenta las Estadísticas del Sector Avícola – CONAVE.” https://conave.org/conave-presenta-las-estadisticas-del-sector-avicola/ (accessed Aug. 07, 2023).

[2] Econ. MBA. Ana María Sánchez Econ. MBA. Tatiana Vayas Ing. Fernando Mayorga Ing. Carolina Freire, “Sector-avicola-Ecuador”, Accessed: Aug. 07, 2023. [Online]. Available: https://obest.uta.edu.ec/wp-content/uploads/2020/09/Sector-avicola-Ecuador.pdf

[3] C. O. Asanza Reyes, “Estudio de factibilidad económica para la producción de sábila aloe vera en el cantón Balsas,” 2007, Accessed: Aug. 07, 2023. [Online]. Available: http://repositorio.utmachala.edu.ec/handle/48000/1706

[4] Oliverio Napoleón Vargas González, “AVICULTURA”, Accessed: Jun. 27, 2023. [Online]. Available: https://acortar.link/HuUCfp

[5] Jorge Lesmes, “Federación nacional de avicultores de colombia-Fenavi”, Accessed: Jun. 27, 2023. [Online]. Available: www.fenavi.org

[6] Salomé Rosales Tapia, “Versión Pública Tema: Estudio de Mercado Avícola enfocado a la Comercialización del Pollo en Pie”, Accessed: Jun. 29, 2023. [Online]. Available: http://www.scpm.gob.ec/biblioteca

[7] CONAVE, “El sector avicultor y su aporte en la generación de fuentes de empleo en el Ecuador. – CONAVE.” https://conave.org/el-sector-avicultor-y-su-aporte-en-la-generacion-de-fuentes-de-empleo-en-el-ecuador/ (accessed Jun. 29, 2023).

[8] Econ. MBA. Ana María Sánchez Econ. MBA. Tatiana Vayas Ing. Fernando Mayorga Ing. Carolina Freire, “Sector-avicola-Ecuador”, Accessed: Jun. 29, 2023. [Online]. Available: https://obest.uta.edu.ec/wp-content/uploads/2020/09/Sector-avicola-Ecuador.pdf

[9] E. LA Cantón Libertad, P. DE Santa Elena Trabajo De Titulación, R. Aarón Júpiter Toala Tutor, I. Carlos Balmaseda Espinosa, and L. Libertad, “PRODUCCIÓN Y COMERCIALIZACIÓN DE POLLOS EN Previo a la obtención del título de: INGENIERO AGROPECUARIO”, Accessed: Jun. 29, 2023. [Online]. Available: https://repositorio.upse.edu.ec/xmlui/bitstream/handle/46000/5960/UPSE-TIA-2021-0029.pdf?sequence=1&isAllowed=y

[10] M. Clotilde, H. Pinzón, T. R. Duque, and M. C. Ruíz, “Propuesta de mejoramiento para la producción de pollo de engorde, en una empresa del sector avícola en Colombia,

mediante el uso de herramientas estadísticas de control de calidad", Accessed: Jun. 29, 2023. [Online]. Available: https://ciencia.lasalle.edu.co/maest_agronegocios//ciencia.lasalle.edu.co/maest_agronegocios/40

[11] JOYERIA LUCY;, "AVICULTURA_O_Vargas", Accessed: Jun. 29, 2023. [Online]. Available: https://www.academia.edu/36192943/AVICULTURA_SITUACI%C3%93N_ACTUAL

[12] Javier Pedroza, "Manual de Producción Avícola", Accessed: Jun. 29, 2023. [Online]. Available: https://www.academia.edu/24288459/Avicultura

[13] MARIO FERNANDO VALDIVIEZO HALLO, "ESCUELA SUPERIOR POLITÉCNICA DEL CHIMBORAZO FACULTAD DE CIENCIAS PECUARIAS ESCUELA DE INGENIERÍA ZOOTÉCNICA "DETERMINACIÓN Y COMPARACIÓN DE PARÁMETROS PRODUCTIVOS EN", Accessed: Jun. 29, 2023. [Online]. Available: https://core.ac.uk/download/pdf/234590698.pdf

[14] F. DE Balanceado and D. Nancy Bonifaz Quito, "UNIVERSIDAD POLITÉCNICA SALESIANA SEDE QUITO CARRERA: INGENIERÍA AGROPECUARIA Tesis previa a la obtención del Título de: INGENIERA AGROPECUARIA LA TORTA DE SOYA COMO FUENTE DE PROTEÍNA EN LA AUTORA: KARLA VANESSA CASTRO MARTÍNEZ DIRECTORA", Accessed: Jun. 29, 2023. [Online]. Available: https://www.mag.go.cr/bibliotecavirtual/L01-8217.pdf

[15] Vargas Mamuel, Fernandez Jinson, and Vargas Ornella, "Producción de pollos de engorde (Broiler)", Accessed: Jun. 29, 2023. [Online]. Available: https://es.slideshare.net/JinsonFernndezAguila/produccin-de-pollos-de-engorde-broiler

[16] A. Vargas Céspedes, K. Serrano Chaves, W. Watler, M. Morales, and R. Vignola, "FICHA TÉCNICA SECTOR PRODUCTIVO AVÍCOLA Realizado con el aporte del Fondo de Adaptación", Accessed: Jun. 29, 2023. [Online]. Available: https://www.mag.go.cr/bibliotecavirtual/L01-8217.pdf

[17] CESAR FERNANDO JARAMA PEÑALOZA, "UNIVERSIDAD POLITÉCNICA SALESIANA SEDE CUENCA CARRERA DE MEDICINA VETERINARIA Y ZOOTECNIA TRABAJO EXPERIMENTAL PREVIO A LA OBTENCIÓN".

[18] Aviagen, "Manual de manejo del pollo de engorde Arbor Acres Manual de manejo del pollo de engorde," 2018, Accessed: Jul. 30, 2023. [Online]. Available: www.aviagen.com.

[19] S. Ignacio and A. Corey, “ACTUALIZACIÓN DE LAS BUENAS PRÁCTICAS DE PRODUCCIÓN PARA POLLOS BROILER EN ENGORDA”.

[20] F. Santana Romo, “DETERMINACIÓN DEL AUMENTO DE PESO EN POLLOS DE ENGORDE (GALLUS GALLUS) MEDIANTE LA INCORPORACIÓN DE DIFERENTES FUENTES PROTEICAS A SU ALIMENTACIÓN.”.

[21] T. Hardy, “Revista de la Universidad Bolivariana Volumen”.

[22] Y. Ocaña-Fernández, L. A. Valenzuela-Fernández, and L. L. Garro-Aburto, “Inteligencia artificial y sus implicaciones en la educación superior,” *Propósitos y Representaciones*, vol. 7, no. 2, Jan. 2019, doi: 10.20511/pyr2019.v7n2.274.

[23] Alberto Maisueche Cuadrado, “UTILIZACIÓN DEL MACHINE LEARNING EN LA INDUSTRIA 4.0”.

[24] “Tipos de aprendizaje en Machine Learning: supervisado y no supervisado.” https://telefonicatech.com/blog/que-algoritmo-elegir-en-ml-aprendizaje (accessed Jul. 30, 2023).

[25] Xabier Basogain Olabe, “REDES NEURONALES ARTIFICIALES Y SUS APLICACIONES.”

[26] John D. Kelleher, “Deep Learning - John D. Kelleher - Google Libros.” https://books.google.es/books?hl=es&lr=&id=b06qDwAAQBAJ&oi=fnd&pg=PP9&dq=Deep+Learning&ots=_oIUOOoT_L&sig=jZ1tZG04XPIZqPpyHIOPO0dfd8k#v=onepage&q=Deep%20Learning&f=false (accessed Aug. 12, 2023).

[27] Francois Chollet, “Deep Learning with Python, Second Edition - Francois Chollet - Google Libros.” https://books.google.es/books?hl=es&lr=&id=mjVKEAAAQBAJ&oi=fnd&pg=PR9&dq=Deep+Learning+with+Python&ots=AfgVvH_H_j&sig=JPVdqOhJdlrBhpWkiW32l9njDzI#v=onepage&q=Deep%20Learning%20with%20Python&f=false (accessed Aug. 12, 2023).

[28] IBM, “El modelo de redes neuronales - Documentación de IBM.” https://www.ibm.com/docs/es/spss-modeler/saas?topic=networks-neural-model (accessed Aug. 12, 2023).

[29] Jason Brownlee, “Deep Learning for Computer Vision: Image Classification, Object Detection ... - Jason Brownlee - Google Libros.” https://books.google.es/books?hl=es&lr=&id=DOamDwAAQBAJ&oi=fnd&pg=PP1&dq=Deep+Learning+for+Computer+Vision+with+Python&ots=3rBraMIHFQ&sig=s004xSL

ceOibcMKCJ13DPHR_TmM#v=onepage&q=Deep%20Learning%20for%20Computer%20Vision%20with%20Python&f=false (accessed Aug. 12, 2023).

[30] F. Izaurieta and C. Saavedra, “Redes Neuronales Artificiales”.

[31] C. Alberto Ruiz Marta Susana Basualdo Autor and D. Jorge Matich, “Cátedra: Informática Aplicada a la Ingeniería de Procesos-Orientación I Redes Neuronales: Conceptos Básicos y Aplicaciones”.

[32] C. B. Carrión, “REDES CONVOLUCIONALES”.

[33] Emilson Junior Parra Gamarra, “Caracterización y Análisis de los Modelos de Reconocimiento de Imágenes Mediante Redes Neuronales Convolucionales.” Accessed: Aug. 12, 2023. [Online]. Available: https://repository.udistrital.edu.co/bitstream/handle/11349/30482/ParraGamarraEmilsonJunior2022.pdf?sequence=1&isAllowed=y

[34] “Python Documentation.” https://docs.python.org/es/3/ (accessed Jul. 30, 2023).

[35] “Teachable Machine.” https://teachablemachine.withgoogle.com/ (accessed Jul. 30, 2023).

[36] “Programación y Robótica: Teachable Machine.” https://formacion.intef.es/mod/book/view.php?id=2625&chapterid=2408 (accessed Jul. 30, 2023).

[37] “Google Colab .” https://colab.research.google.com/?hl=es (accessed Jul. 30, 2023).

[38] “Visual Studio Code Documentation .” https://code.visualstudio.com/docs (accessed Jul. 30, 2023).

[39] “MySQL Documentation.” https://dev.mysql.com/doc/ (accessed Jul. 30, 2023).

[40] Pablo Álvarez Corredera, “BIBLIOTECAS de PYTHON | CIBERNINJAS”, Accessed: Jul. 30, 2023. [Online]. Available: https://ciberninjas.com/python-librerias/

[41] SINTYA MILENA MELÉNDEZ VALLADAREZ, MARIA ELIZABETH GAITAN, and NELDIN NOEL PÉREZ REYES, “METODOLOGIA ÁGIL DE DESARROLLO DE SOFTWARE PROGRAMACION EXTREMA.”.

[42] “Programación Extrema Programación Extrema Índice”.

[43] M. C. Saúl González Campos, M. C. Luis, and F. Fernández Martínez, “Software Programación Extrema: Prácticas, Aceptación y Controversia.”

[44] ROMERO SANCHEZ JORGE LUIS and QUINDE GONZABAY JEFFERSON ROLANDO, “SISTEMA EMBEBIDO PARA LA AUTOMATIZACIÓN DEL CONTROL Y MONITOREO DE LA PRODUCCIÓN EN LA GRANJA AVÍCOLA ‘ROMERO & HNOS.’”.

[45] Miguel Ángel López Carrasco, “‘SISTEMA DE GESTIÓN POR PROCESOS EN LA LÍNEA DE PRODUCCIÓN PARA LA EMPRESA AVÍCOLA LA PONDEROSA EN EL CANTÓN DE SALCEDO.’” Accessed: Jul. 30, 2023. [Online]. Available: https://repositorio.uta.edu.ec/bitstream/123456789/28940/1/Tesis_t1497id.pdf

[46] B. I. García Vaca and F. R. Mora Cruz, “DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO IOT PARA EL MONITOREO DE PARÁMETROS AMBIENTALES APLICADOS A LA AVICULTURA PARA LA CRIANZA DE POLLOS DE GRANJA UTILIZANDO HARDWARE DE BAJO COSTO Y AWS.” Accessed: Jul. 30, 2023. [Online]. Available: http://dspace.ups.edu.ec/handle/123456789/21122

[47] San Lucas Arancibia and D. Isabel Garzón Salazar, “CONTROL Y MONITOREO DE UN CRIADERO AVÍCOLA CONTROLADO POR MICROCONTROLADOR DESDE UN SITIO WEB.” Accessed: Jul. 30, 2023. [Online]. Available: http://dspace.ups.edu.ec/handle/123456789/1664

[48] J. J. Yaguar Mariño, F. J. Estacio Reina, and J. Jairo, “SISTEMA DE INFORMACIÓN PARA EL CONTROL Y MONITOREO ARDUINO DE LA CRIANZA AVÍCOLA EN LA GRANJA ‘PURA PECHUGA’.” Accessed: Jul. 30, 2023. [Online]. Available: https://dspace.uniandes.edu.ec/handle/123456789/9838

[49] “tf.keras.aplicaciones.mobilenet_v2.MobileNetV2 | TensorFlow v2.13.0.” https://www.tensorflow.org/api_docs/python/tf/keras/applications/mobilenet_v2/MobileNetV2 (accessed Aug. 08, 2023).

11. Anexos

Anexo 1. Acta de Compromiso

UNIVERSIDAD NACIONAL DE LOJA

FACULTAD DE LA ENERGÍA, LAS INDUSTRIAS Y LOS RECURSOS NATURALES NO RENOVABLES
Carrera de Ingeniería en Sistemas / Carrera Computación

CARTA COMPROMISO DE COOPERACIÓN INTERINSTITUCIONAL ENTRE GRANJA "LAS ACACIAS", Y LA CARRERA DE INGENIERÍA EN SISTEMAS/COMPUTACIÓN DE LA CIS/C NACIONAL DE LOJA.

COMPARECIENTES:

Comparecen a la celebración de la presente Carta Compromiso, por una parte, **Granja "Las Acacias",** representada legalmente por **PINTO AÑAZCO ALEXIS JEOVANNY**, que en adelante se denominará **GRANJA "LAS ACACIAS",** y por otra, la CIS/C Nacional de Loja, a través del Ing. Pablo Fernando Ordoñez Ordoñez Mg.Sc. Director de la CIS/C en la Facultad de la Energía, las Industrias y los Recursos Naturales No Renovables; que en adelante se denominará "CIS/C", quienes con la capacidad legal que en derecho se requiere para este tipo de actos acuerdan celebrar la presente carta compromiso.

Las dos partes representantes de las instituciones declaran su voluntad para suscribir el presente instrumento, cuyo objeto es concretar acciones específicas de cooperación interinstitucional para cumplir con los lineamientos de políticas institucionales y habilitar el Desarrollo de actividades de investigación en temas propuestos por ambas partes y aquellos orientados al esquema de proyectos, cuyo marco regulador se rige por las cláusulas que a continuación se detallan:

PRIMERA: ANTECEDENTES

GRANJA "LAS ACACIAS",

La empresa se encuentra legalmente constituida, con Nro. **RUC: 0703788216001**, con domicilio en la **Provincia de El Oro, Cantón Balsas, Sector las Acacias**, y cuyo representante legal es **Pinto Añazco Alexis Jeovanny**; quien desempeña actividades y funciones de Representante legal y CEO de la misma.

LA UNIVERSIDAD NACIONAL DE LOJA

La Universidad Nacional de Loja es una Institución de Educación Superior, Laica, autónoma, de derecho público, con personería jurídica y sin fines de lucro, de alta calidad académica y humanística, que ofrece formación en los niveles: técnico y tecnológico superior; profesional o de tercer nivel; y, de postgrado o cuarto nivel; que realiza investigación científico-técnica sobre los problemas del entorno, con calidad, pertinencia y equidad, a fin de coadyuvar al desarrollo sustentable de la región y del país, interactuando con la comunidad, generando propuestas alternativas a los problemas nacionales, con responsabilidad social; reconociendo y promoviendo la diversidad cultural y étnica y la sabiduría popular, apoyándose en el avance científico y tecnológico, en procura de mejorar la calidad de vida del pueblo ecuatoriano. Se rige por la constitución política de la república del Ecuador, La ley de Educación Superior, y su Reglamento de Aplicación y las Leyes Conexas, los Reglamentos del Consejo de Educación Superior (CES) y sus Resoluciones, la Secretaria Nacional de Educación Superior, Ciencia Tecnología e Innovación (SENESCYT), el Estatuto Orgánico y Reglamento General

de la Universidad, los Reglamentos, Normativos e Instructivos; y, las resoluciones que adopten sus organismos de gobiernos y las autoridades universitarias en el ámbito de su competencia.

SEGUNDO: MARCO LEGAL

El presente Instrumento Legal se fundamenta en los artículos 87 de la Ley Orgánica de Educación Superior en sus artículos 89, 90, 91, 94 y de la graduación y titulación del Reglamento de Régimen Académico.

TERCERA: OBJETO

En base a los preceptos antes indicados y a los objetivos del Plan Nacional del Buen Vivir, la CIS/C de la Universidad Nacional de Loja y **GRANJA "LAS ACACIAS",** a través de ésta Carta Compromiso establecen vínculos de cooperación interinstitucional que permitan a los estudiantes de la Carrera de Ingeniería en Sistemas/Computación de la Facultad de la Energía, las Industrias y los Recursos Naturales No Renovables de la Universidad Nacional de Loja, desarrollar acciones conjuntas que permitan el correcto desarrollo de actividades de investigación en temas propuestos por **Granja "Las Acacias",** o por la "CIS/C" y aquellos orientados al esquema de proyectos.

CUARTA: COMPROMISO DE LAS PARTES

DE GRANJA "LAS ACACIAS"

1. La Empresa no cobrará ningún monto por conceptos derivados del proyecto y productos.
2. Asimismo, la Empresa se compromete a impartir los conocimientos y habilidades del oficio de GRANJA "LAS ACACIAS" y, a su vez, el estudiante, cumplirá estrictamente los Programas planificados en la empresa y las tareas asignadas a él. Los proyectos durarán el tiempo establecido en los cronogramas de investigación acordados.
3. Para estos efectos, el Gerente designa como TUTOR DE LA EMPRESA, en carácter de GUÍA en este proceso a el empleado(a) Fernando Bravo.
4. Los proyectos y sus derivados se registran como propiedad intelectual de la Universidad Nacional de Loja, Carrera de Ingeniería en Sistemas/Computación la misma que, con fines académicos, pueda mostrar al mundo dicha producción intelectual a través de las memorias resultantes de dichos proyectos que se publicarán en el Repositorio Digital Institucional.
5. El contenido de dichas memorias se podrá consultar en las redes de información del país y del exterior, con los cuales tenga convenio la Universidad.
6. La/s base/es de datos vinculadas a los productos del proyecto no tendrán restricción alguna cuyo fin serán las investigaciones futuras en la Universidad Nacional de Loja.
7. Los logos de la Universidad y Carrera serán siempre visibles en los productos resultantes
8. Colaborar como patrocinador en los eventos académicos generados por la Carrera CISC de la Universidad Nacional de Loja.
9. Financiar los presupuestos que impliquen materiales y otros a excepción del Talento Humano

10. Becas económicas para los estudiantes en caso de movilidad al lugar de desarrollo del proyecto.
11. Otros acuerdos que previamente se establezcan en beneficio de la Carrera CISC de la Universidad Nacional de Loja.

DE LA CIS/C DE LA UNIVERSIDAD NACIONAL DE LOJA

1. Designar el/los docente/s necesario/s, que serán responsables de planificar, organizar y evaluar la participación de las y los estudiantes de la Carrera Ingeniería en Sistemas / Computación de la Facultad de la Energía las Industrias y los Recursos Naturales no Renovables, en la realización de los proyectos establecidos.
2. Vigilar permanentemente que las y los estudiantes de la Carrera de Ingeniería en Sistemas / Computación de la Facultad de la Energía, Las Industrias y los Recursos Naturales no Renovables, cumplan con los objetivos y metas de trabajo acordados por las partes en los proyectos que se generen a partir de la presente carta de compromiso;
3. Asignará el número de horas para el desarrollo del proyecto de acuerdo a los requerimientos de su formación académica y respectiva especialidad y a la planificación curricular de la Carrera de Ingeniería en Sistemas / Computación.
4. Definir las actividades en las que participen la o los estudiantes de la Carrera de Ingeniería en Sistemas / Computación en **GRANJA "LAS ACACIAS".**
5. Para la evaluación, monitoreo y control del cumplimiento de los proyectos el docente encargado realizará llamadas o visitas a **GRANJA "LAS ACACIAS".**
6. Otorgar el aval académico para que las y los estudiantes de la Carrera de Ingeniería en Sistemas / Computación de la Facultad de la Energía las Industrias y los Recursos Naturales no Renovables, así como funcionarios, de **GRANJA "LAS ACACIAS",** que participan en las actividades previstas en la presente carta de compromiso; cuenten con las certificaciones que acrediten su participación.

QINTA: PROCEDIMIENTOS DE EJECUCIÓN

Para la ejecución de la presente carta de compromiso, en cuanto sea posible y conveniente, las partes observarán los siguientes lineamientos.

- **GRANJA "LAS ACACIAS"** enviará a través de oficio las necesidades de estudiantes en modalidad de proyectos y/o trabajos de titulación.
- La CIS/C Nacional de Loja a través de la Carrera de Ingeniería en Sistemas / Computación canalizará el recurso solicitado de acuerdo con su disponibilidad, observando las capacidades de los estudiantes de acuerdo con su formación académica y especialidad.
- La ejecución de acciones se regirá por medio de acuerdos establecidos exclusivamente a las actividades que cumplirá el estudiante durante el tiempo que dure el trabajo de titulación curricular,

SÉXTA. - RELACIÓN LABORAL:

Las partes están exentas de asumir responsabilidad laboral o de cualquier otra naturaleza jurídica con terceros, tales como reclamos, juicios, recursos, indemnizaciones o cualquier acción legal que pueda surgir o derivarse de las acciones ejecutadas por una de las partes signatarias en la ejecución del presente instrumento.

SEPTIMA. - ACTA DE CONFIDENCIALIDAD.

- La firma de este documento es con la finalidad de garantizar la confidencialidad en la entrega de la información, documentos por parte de la empresa **GRANJA "LAS ACACIAS"**, a los estudiantes; por esta razón se firmará un acta de Confidencialidad y buen uso de la información; en caso de incumplimiento, serán sancionados de acuerdo con la Ley y a las normas internas de **GRANJA "LAS ACACIAS".**

OCTAVA. - DURACIÓN:

La duración del compromiso asumido por las partes están contados a partir de la fecha de su suscripción por 3 años, pudiendo ser renovada por el mismo plazo, en tal razón, cualquiera de las partes podrán solicitar con 30 días de anticipación la renovación de la presente, de hallarse interés en la renovación por cualquiera de las dos partes, solo será necesario adjuntar dichas comunicaciones al presente instrumento en calidad de documento habilitantes, sin embargo, de existir cambios sustanciales al contenido, se deberá suscribir una nueva Carta Compromiso adjuntando el respectivo informe técnico que recomienda dichos cambios.
Los Compromisos de la **Granja "Las Acacias",** de número 1 y del 4 al 8 no fenecen.

NOVENA. - CONTROVERSIAS:

Cualquier controversia que surja de la aplicación de estos compromisos, respecto de la interpretación, cumplimiento o ejecución del compromiso adquirido por las partes, será sometida a un arreglo en forma directa, mediante procedimientos de amigable composición, a través de los representantes de las instituciones para este instrumento, en un lapso no mayor a treinta días calendario, contados a partir de la notificación de cualquiera de ellas, señalando la divergencia o controversia surgida.
Si se suscitaren divergencias o controversias en la interpretación o ejecución del presente instrumento, cuando las partes no llegaren a un acuerdo directo, podrán utilizar los métodos alternativos para la solución de controversias en el Centro de Mediación de la Función Judicial de Loja.

DÉCIMA. - COMUNICACIONES Y NOTIFICACIONES:

Todas las comunicaciones citaciones y/o notificaciones entre las partes, se realizarán por escrito a las siguientes direcciones en la ciudad de Loja:

GRANJA "LAS ACACIAS"

Provincia de El Oro, Cantón Balsas, Sector las Acacias
Telf. 0980855620

Carrera de Ingeniería en Sistemas/Computación:

Ciudad Universitaria Guillermo Falconí Espinosa La Argelia Casilla Letra S, Av Pío Jaramillo Alvarado, Loja EC110111, Facultad de la Energía, Las Industrias y los Recursos Naturales no Renovables, Carrera de Ingeniería en Sistemas / Computación.
Telf. 2545689 ext 110 o 109

Para constancia de lo actuado y de las responsabilidades que origina la presente carta de compromiso, firman en unidad de acto los representantes legales de las instituciones participantes, a los 24 días del mes de abril del dos mil veintitrés.

UNIVERSIDAD NACIONAL DE LOJA FACULTAD DE ENERGIA, LAS INDUSTRIAS Y LOS RECURSOS NATURALES NO RENOVABLES CARRERA DE INGENIERIA EN SISTEMAS - COMPUTACION DIRECTOR DE CARRERA

Ing. Pablo Fernando Ordoñez Ordoñez M.Sc.
DIRECTOR DE CARRERA

Pinto Añazco Alexis Jeovanny
GERENTE GENERAL DE
GRANJA "LAS ACACIAS"

UNIVERSIDAD NACIONAL DE LOJA
FACULTAD DE ENERGIA, LAS INDUSTRIAS Y LOS RECURSOS NATURALES NO RENOVABLES
CARRERA DE INGENIERIA EN SISTEMAS - COMPUTACION
SECRETARIA DE CARRERA

Anexo 2. Entrevistas

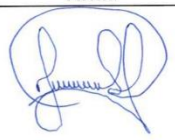
UNIVERSIDAD NACIONAL DE LOJA
ÁREA DE LA ENERGÍA LAS INDUSTRIAS Y LOS RECURSOS NATURALES NO RENOVABLES
CARRERA DE INGENIERIA EN COMPUTACION

Instrumento 1: Experto General

Entrevistadora	Viviana Maricela Zambrano Romero
Entrevistado	Sr. Fernando Bravo
Cargo del entrevistado	Supervisor de la Granja Avícola "La Acacias"
Fecha de entrevista	02/07/2022
Hora de entrevista:	17H00 pm
Canal de Comunicación	Entrevista de video conferencia ZOOM

Objetivo: Recolectar información relevante acerca de las actividades que se realizan así también de cómo se manejan y registran los diferentes parámetros de la avícola.

1) **Mencione o detalle los procesos actuales que se maneja dentro del crecimiento de pollos.**
   Los procesos actuales que se manejan dentro de las granjas es que el alimento lo dan según la edad del pollo, se realizan las vacunas correspondientes, contra hepatitis además se aplica vitaminas para un mejor desarrollo.
2) **¿Cuál es el método que utilizan actualmente para registrar la mortalidad, registro de alimentación, registro de alimento, peso promedio?**
   El método que se utiliza para poder llevar la contabilidad tanto de mortalidad, peso promedio y alimentación, es mediante una agenda donde se va anotando a diario con su respectiva fecha.
3) **¿En dónde usted registra los parámetros del peso promedio, para poder comparar si está en la línea correcta de crecimiento?**
   De igual forma se la lleva en una agenda, el peso se lo realiza diario por 22 días seguidos, y posteriormente semanal hasta la semana 7 que es cuando el pollo empieza a salir al consumo. Se compara las pesas con un formulario que es brindado por el doctor veterinario.
4) **¿Cuáles son las acciones para seguir ante un cambio brusco que se salga del rango normal estimado de temperatura dentro de las instalaciones?**
   Cuando la temperatura es elevada se bajan las lonas que rodean los galpones para que circule el aire, y si ya tienen más de 4 semanas se encienden los ventiladores para que exista una mejor ventilación. Y cuando la temperatura es demasiado baja dependiendo de la edad de las aves, lo que se hace es encender las calentadoras que funcionan a gas o a Diesel.
5) **¿Qué sucede si las aves en algún momento del proceso de crecimiento llegasen a quedarse por un tiempo determinado sin agua o alimento?**
   Cuando se tiene problemas con las reservas de agua y se le seca el agua al pollo, sufren un poco de deshidratación, pero no pueden pasar por más de unas 3 horas sin beber agua y cuando se quedan sin alimento las aves entran en un estado de canibalismo y estrés ya que se empiezan a picar entre ellas y pueden terminar muertas un número considerable de aves.
6) **¿De qué manera usted realiza el control de llenado de los comenderos y los bebederos dentro de las instalaciones avícolas?**
   Los comederos se los llena manualmente, ya que vea que no existe comida dentro de ellos, se procede a cargar alimento, además se cuenta con tanques de agua para cada granja donde se le pone cloro o alguna vitamina y de ahí pasa directo a cada bebedero.

Nombre	Cargo	Firma
Sr. Fernando Bravo	Supervisor de la Granja Avícola "La Acacias"	

UNIVERSIDAD NACIONAL DE LOJA
ÁREA DE LA ENERGÍA LAS INDUSTRIAS Y LOS RECURSOS NATURALES NO RENOVABLES
CARRERA DE INGENIERIA EN COMPUTACION

Instrumento 2: Experto Especifica

Entrevistadora	Viviana Maricela Zambrano Romero
Entrevistado	Sr. Alexis Jeovanny Pinto Añazco
Cargo del entrevistado	Representante legalmente de la Granja Avícola "La Acacias".
Fecha de entrevista	02/07/2022
Hora de entrevista:	17H00 pm
Canal de Comunicación	Entrevista de video conferencia ZOOM

Objetivo: Recolectar información relevante acerca de las actividades que se realizan así también de cómo se manejan y registran los diferentes parámetros de la avícola.

1) **Detalle el proceso que realiza para la producción del pollo.**
   Entre los procesos de producción de pollo tenemos, la alimentación, el llenado de los reservorios, el de climatización de los galpones, el de recibimiento de las aves, el de iluminación, control de salud, monitoreo de mortalidad, monitoreo de alimento consumido, control de enfermedades, programa de vacunación, logística, proceso de cargado de las aves para ser consumido.
2) **Qué procedimientos realiza para controlar el crecimiento del pollo.**
   Para poder controlar el peso del pollo se realiza durante 22 días el pesado de 60 pollitos, sacando así un peso promedio diario, al pasar los 22 días se continúa pesando el pollo, pero ya se lo realiza semanalmente.
3) **¿Qué actividades realiza para registrar la mortalidad, registro de alimentación, registro de alimento, peso promedio del pollo?**
   Al finalizar el día se anota en la agenda el registro de la mortalidad y cuantos pollos quedan en la granja, para el registro de alimento también se anota en la agenda el consumo diario de alimento y se toma en cuenta el saldo de alimento en la granja, el peso también se lo registra, toda esta información es enviada a un grupo de WhatsApp para que tengan conocimiento sobre el peso, mortalidad y consumo.
4) **¿Qué herramientas ocupa para realizar lo descrito anteriormente?**
   Libreta, esfero y celular para poder enviar la información.
5) **Cuáles son los casos extremos que ha tenido usted en el control del crecimiento del pollo.**
   Muchas de las veces cuando el ave empieza a crecer muy rápido sufre una descomposición en su sistema y cae patitas arriba. Y cuando el ave no se está alimentando bien, tiende a subir la mortalidad.
6) **Tiene algún dispositivo electrónico que controle las variables (temperatura, agua, alimentación) que influye en el crecimiento del pollo.**
   Los primeros 22 días se utiliza un termómetro para poder medir la temperatura del ave y así no sufra deshidratación.
7) **Como calcula usted el adecuado crecimiento del pollo.**
   Se calcula mediante el peso y el consumo de alimento, esos valores se comparan con una tabla que tengo que es brindada por el doctor veterinario de la empresa.
8) **Tiene alguna influencia el tamaño del pollo para su distribución, y como realiza ese control.**
   Muchas de las veces algunos distribuidores les gusta llevar pollo pequeño ya que los utilizan para asaderos, mientras que otros prefieren pollo grande para los embutidos.
   El control se lo lleva mediante la toma de peso y de semanas que tenga el ave, ya que cuando el ave es cargada de la granja igual se anota dentro de la libreta para poder enviar esa información.

Nombre	Cargo	Firma
Sr. Alexis Jeovanny Pinto Añazco	Representante legalmente de la Granja Avícola "La Acacias"	

Anexo 3. Especificación de requisitos de software

Especificación de

requisitos de software

Proyecto: Software para el Control de Crecimiento en pollos de engorde en la granja Avícola "Las Acacias" en el Cantón Balsas

Versión: 1.0

Fecha: 03/04/2023

Ficha del documento

Versión	Fecha de revisión	Responsables	Descripción de modificación
1.0	03/04/2023	Viviana Maricela Zambrano Romero	N/A

Índice de contenido

Índice de Figuras

Índice de Tablas

1. Introducción

Este documento es una Especificación de Requisitos Software (ERS) para el Software para el Control de Crecimiento en pollos de engorde en la granja Avícola “Las Acacias” en el Cantón Balsas. Las pautas para las prácticas recomendadas para las especificaciones de requisitos de software provienen del IEEE.ANSI/IEEE 830, 1998.

1.1 Propósito

El propósito del documento de Especificación de Requisitos de Software es definir las los requsitos de especificaciones tanto funcionales y tambien no funcionales para el desarrollo de Software para el Control del Crecimiento. El documento es una herramienta fundamental en la fase de desarrollo.

1.2 Alcance

El usuario del sistema es quien desea construir un modelo de reconocimiento de imágenes mediante visión artificial para identificar el crecimiento de tamaño de los pollos, e implementar el software para el control del crecimiento utilizando la metodología XP a través del lenguaje de programación Python.

1.3 Personal involucrado

Tabla 15. Personal involucrado estudiante de la CIS

Nombre	Viviana Maricela Zambrano Romero
Rol	Analista y Desarrollador de Software
Categoría Profesional	Estudiante de la CIS
Responsabilidad	Análisis de información, diseño y programación del módulo de software
Información de contacto	viviana.zambrano@unl.edu.ec

Tabla 16.Personal involucrado docente de la CIS

Nombre	Edwin René Guamán Quinche
Rol	Director del Trabajo de Integración Curricular
Categoría Profesional	Docente de la CIS/C
Responsabilidad	Supervisar y asesorar en el desarrollo del Trabajo de Titulación
Información de contacto	rguaman@unl.edu.ec

1.3 Definiciones, acrónimos y abreviaturas

Tabla 17. Definiciones, acrónimos y abreviaturas

Nombre	Descripción
AI	Inteligencia Artificial
CONAVE	Corporación Nacional de Avicultores del Ecuador
CIS/C	Carrera de Ingeniería en Sistemas/Computación
ERS	Especificación de Requisitos Software
RF	Requerimiento Funcional
RNF	Requerimiento No Funcional

1.4 Referencias

Tabla 18. Referencias ERS

Título del Documento	Referencia
IEEE Std 830-1998	IEEE Recommended Practice for Software Requirements Specifications

1.4 Resumen

Este documento fue elaborado con el objetivo de detallar todas las consideraciones técnicas en el desarrollo del Software para el Control de Crecimiento en pollos de engorde en la granja Avícola “Las Acacias” en el Cantón Balsas, consta de tres secciones.

En la primera sección del documento presenta la introducción y descripción general de la especificación de recursos del sistema es el comienzo del documento. Hay un propósito, alcance y personal involucrado en el proyecto.

En la segunda parte de esta sección del siguiente documento se proporciona una descripción de manera general de lo que se refiere al en la segunda sección del documento. Se describen los objetivos a alcanzar, las características de los actores involucrados en el uso del módulo informático, así como las restricciones, supuestos y dependencia del proyecto.

La tercera sección del documento es la que se encarga definir requisitos están en la tercera sección del documento. Existen requisitos tanto funcionales como no funcionales que deben implementarse en el desarrollo del Software para el Control del Crecimiento en pollos de engorde. El sistema se describe con el fin de satisfacer las necesidades del usuario.

2. Descripción general

2.1 Perspectiva del producto

El software de Control de Crecimiento podrá ser utilizado de manera rápida y efectiva en el establecimiento Avícola Las Acacias del Cantón Balsas, lo que permitirá brindar características de transparencia, inmutabilidad, seguridad y trazabilidad.

2.2 Funcionalidad del producto

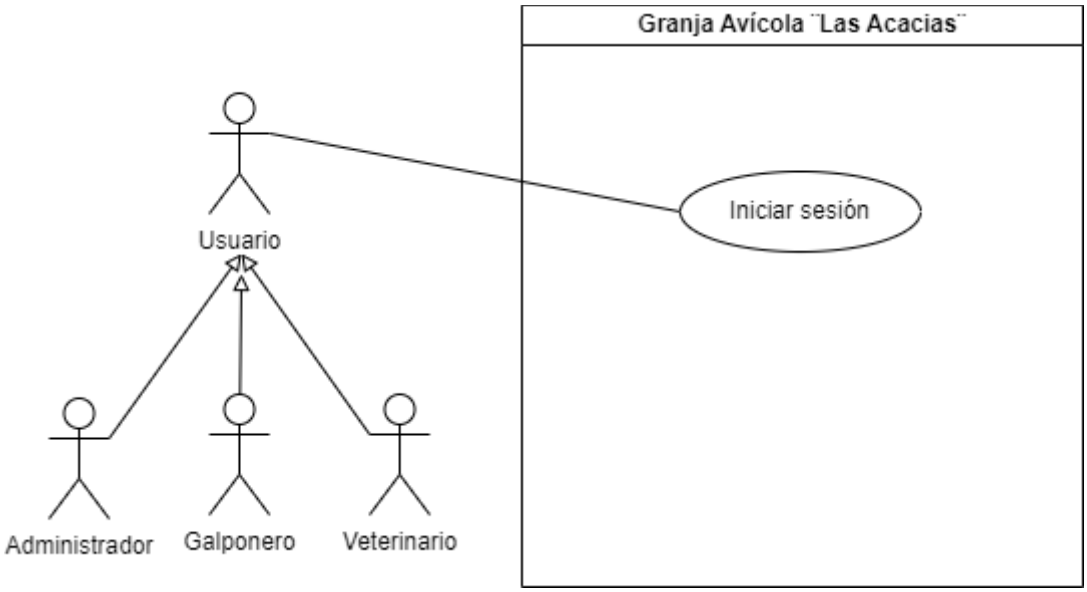


Figura 40. Rol Usuario

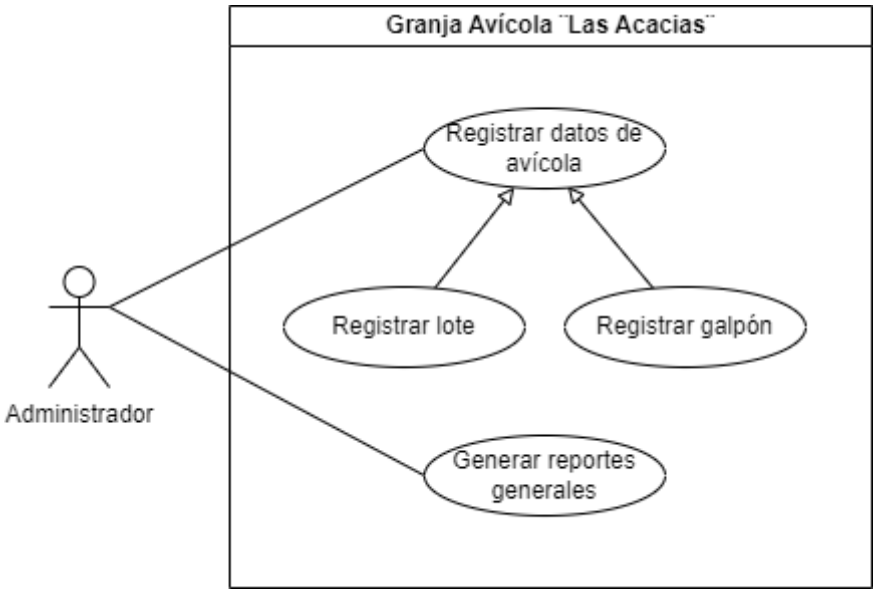


Figura 41. Rol Administrador

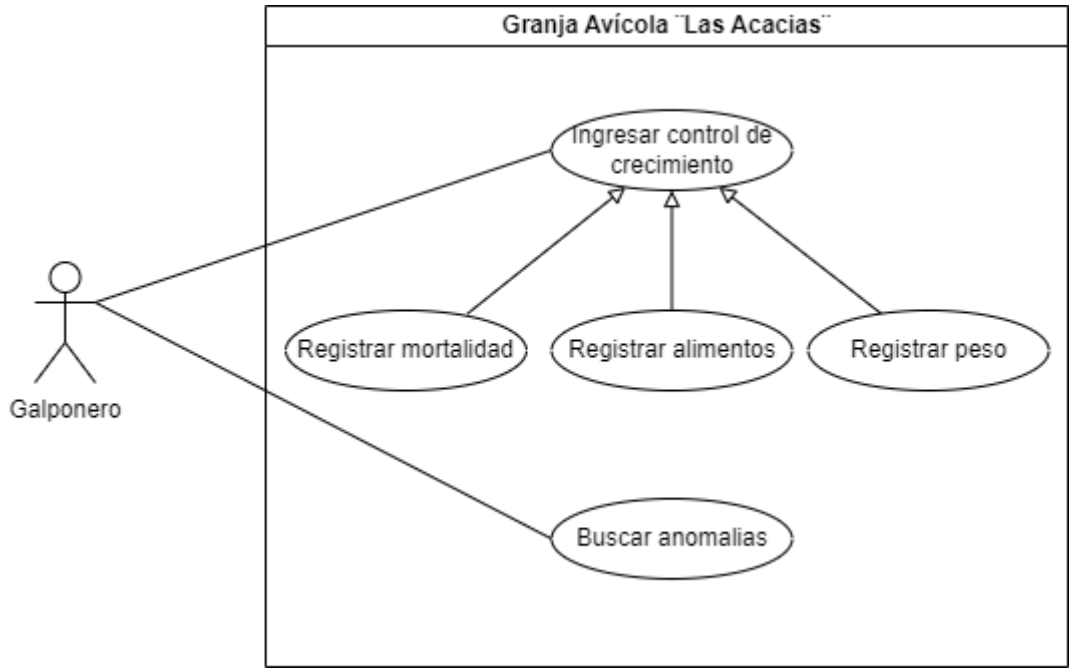


Figura 42. Rol Galponero

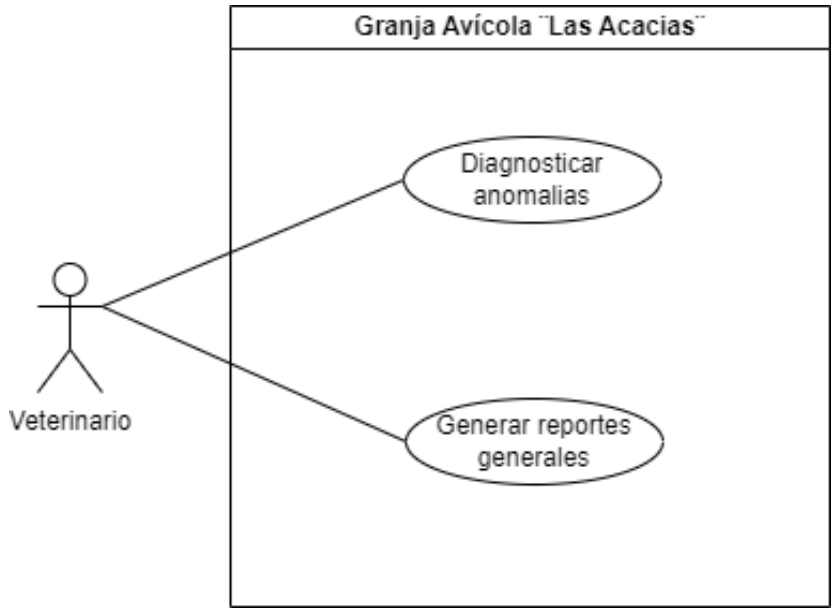


Figura 43. Rol Veterinario

2.3 Características de los usuarios

Tabla 19. Características usuario administrador

Tipo de usuario	Usuario Administrador
Formación	N/A
Actividades	• Registrar los datos de la avícola • Generar los reportes generales

Tabla 20. Características usuario galponero

Tipo de usuario	Usuario Galponero
Formación	N/A
Actividades	• Ingresar datos diarios en el control de crecimiento como, mortalidad diaria, recepción de alimentos balanceados, peso promedio. • Visualizar los diagnósticos de las anomalías del pollo.

Tabla 21. Características usuario veterinario

Tipo de usuario	Usuario Veterinario
Formación	N/A
Actividades	• Registrar los diagnósticos de las anomalías del pollo. • Generar los reportes generales

2.4 Restricciones

- Interfaz para ser usada con internet.
- El módulo del software que se realizó podrá ser utilizado en diferentes navegadores Chrome, Mozilla Firefox, Safari, Opera o Edge.
- El módulo de software que se realizó requiere internet para su correcto funcionamiento.
- Lenguajes y tecnologías en uso: Python, Teachable Machine , Google Colab, Visual Studio Code.
- Se dispondrá de una base de datos relacional como esÑ MySQL.
- El sistema deberá, tener un diseño independiente, de la plataforma o del lenguaje de programación, además de una implementación sencilla,
- Para acceder al Software para el Control de Crecimiento en pollos de engorde en la granja Avícola “Las Acacias” en el Cantón Balsas, en el que cada usuario del Sistema deberá contar con usuraio, una contraseña dependiendo su rol para acceder al mismo.

2.5 Suposiciones y dependencias

- Los requisitos del documento se asumen que son estables.
- Para ejecutar el sistema se garantiza que debe cumplis con los requisitos dados anteriormente para que se ejecute de manera correcta.

3. Requisitos específicos

3.1 Requisitos comunes de las interfaces

3.1.1 Interfaces de usuario

La interfaz de usuario será una serie de ventanas que contendrán botones, listas y campos de texto, y se desarrollará de manera personalizada para el sistema propuesto. Se utilizarán principalmente los colores verde, amarillo y azul para diseñar la interfaz, que contendrá elementos como menús, formularios de entrada y actualización de datos, botones, cuadros de texto y mensajes de error e información. El objetivo es facilitar la navegación y hacerla intuitiva para los usuarios.

Figura 44. Interfaz gráfica inicial

3.1.2 Interfaces de hardware

Será necesario disponer de equipos de cómputos en perfecto estado con las siguientes características:

Computador:

- 14 pulgadas, 800 dpi, de pantalla
- core i3, Pentium dual core, de procesador
- 2GB, memoria ram minima
- Entrada/Salida, de perifericos

Conectividad:

- Conexión a internet.

3.1.3 Interfaces de software

- Sistema Operativo: Windows 7 o superior.
- Explorador: Mozilla o Chrome.

3.2 Requisitos funcionales

Usuario Administrador

Tabla 22. Requisitos funcionales usuario Administrador

Identificador	Nombre
RF01	Inicio de sesión
RF02	Registro datos de la avícola
RF03	Generar reportes generales

Usuario Galponero

Tabla 23. Requisitos funcionales usuario Galponero

Identificador	Nombre
RF04	Inicio de sesión
RF05	Registrar control de crecimiento
RF06	Registrar mortalidad
RF07	Registrar alimentos
RF08	Registrar peso (edad)
RF09	Evaluar imagen (fase)

Usuario Veterinario

Tabla 24. Requisitos funcionales usuario Veterinario

Identificador	Nombre
RF10	Inicio de sesión
RF11	Ingresar información del diagnóstico de anomalías
RF12	Ingresar enfermedad nueva
RF13	Generar reportes generales

Tabla 25.Requisito funcional Inicio de sesión

Identificación del requerimiento:	RF01
Nombre del Requerimiento:	Inicio de sesión
Descripción del requerimiento:	El sistema permitirá ingresar a través de la validación de credenciales (usuario y contraseña).
Dependencias:	S/N
Requerimiento NO funcional:	• RNF03 • RNF04
Prioridad del requerimiento:	Alta

Tabla 26. Requisito funcional Registrar datos de la avícola

Identificación del requerimiento:	RF02
Nombre del Requerimiento:	Registrar datos de la avícola
Descripción del requerimiento:	El sistema permitirá al usuario Administrador registrar los datos esenciales de la avícola, incluyendo lote y galpón.
Dependencias:	• S/N
Requerimiento NO funcional:	• RNF01 • RNF02
Prioridad del requerimiento:	Alta

Tabla 27. Requisito funcional Generar reportes generales

Identificación del requerimiento:	RF03
Nombre del Requerimiento:	Generar reportes generales
Descripción del requerimiento:	El sistema permitirá al Administrador y Veterinario generar reportes generales que contengan información sobre la avícola y su funcionamiento referente a mortalidad, alimentación, peso, enfermedades y tratamiento.
Dependencias:	• S/N
Requerimiento NO funcional:	• RNF01 • RNF02
Prioridad del requerimiento:	Alta

Tabla 28. Requisito funcional Registrar control de crecimiento

Identificación del requerimiento:	RF04
Nombre del Requerimiento:	Registrar control de crecimiento
Descripción del requerimiento:	El sistema permitirá al galponero registrar datos de control de crecimiento de los pollos. (Mortalidad diaria, recepción de alimentación diaria de las aves, peso promedio, identificación de crecimiento.)
Dependencias:	• S/N
Requerimiento NO funcional:	• RNF01 • RNF02
Prioridad del requerimiento:	Alta

Tabla 29. Requisito funcional Registrar mortalidad

Identificación del requerimiento:	RF05
Nombre del Requerimiento:	Registrar mortalidad
Descripción del requerimiento:	El sistema permitirá al galponero ingresar los datos diarios de mortalidad y descarte, saldo de mortalidad.
Dependencias:	• S/N
Requerimiento NO funcional:	• RNF01 • RNF02
Prioridad del requerimiento:	Alta

Tabla 30.Requisito funcional Registrar alimentación

Identificación del requerimiento:	RF06
Nombre del Requerimiento:	Registrar alimentación
Descripción del requerimiento:	El sistema permitirá al galponero registrar conductor, cantidad de ingreso, cantidad consumido, fecha ingreso, hora ingreso, hora fin, saldo balanceado
Dependencias:	• S/N
Requerimiento NO funcional:	• RNF01 • RNF02
Prioridad del requerimiento:	Alta

Tabla 31. Requisito funcional Registrar peso

Identificación del requerimiento:	RF06
Nombre del Requerimiento:	Registrar peso
Descripción del requerimiento:	El sistema permitirá al galponero registrar el peso de los pollos. (Calcular edad de crecimiento de pollos)
Dependencias:	S/N
Requerimiento NO funcional:	RNF01 RNF02
Prioridad del requerimiento:	Alta

Tabla 3232. Requisito funcional Evaluar imagen

Identificación del requerimiento:	RF07
Nombre del Requerimiento:	Evaluar imagen
Descripción del requerimiento:	
Dependencias:	• S/N
Requerimiento NO funcional:	• RNF01 • RNF02
Prioridad del requerimiento:	Alta

Tabla 333. Requisito funcional Ingresar información del diagnóstico de anomalías

Identificación del requerimiento:	RF18
Nombre del Requerimiento:	Ingresar información del diagnóstico de anomalías
Descripción del requerimiento:	El sistema permitirá al veterinario ingresar información sobre las enfermedades y tratamientos de los pollos.
Dependencias:	• S/N
Requerimiento NO funcional:	• RNF03 • RNF04
Prioridad del requerimiento:	Alta

Tabla 34. Requisito funcional Ingresar información del diagnóstico de anomalías

Identificación del requerimiento:	RF10
Nombre del Requerimiento:	Agregar enfermedad nueva
Descripción del requerimiento:	El sistema permitirá al veterinario ingresar información sobre alguna enfermedad nueva.
Dependencias:	• S/N
Requerimiento NO funcional:	• RNF03 • RNF04
Prioridad del requerimiento:	Alta

3.3 Requisitos no funcionales

Tabla 34. Requisitos no funcionales de manera general

Identificador	Requisito	Descripción
RNF01	Rendimiento	El módulo de software debe proveer un tiempo de respuesta aceptable de 2 a 12 segundos aproximadamente.
RNF02	Usabilidad	El módulo de software debe proveer una interfaz amigable e intuitiva, haciendo que el proceso sea comprensible y fácil de llevar a cabo.
RNF03	Seguridad	El módulo de software debe garantizar la autentificación de usuario mendiante usuario y contraseña.

Tabla 35. Requisito no funcional Rendimiento

Identificación del requerimiento:	RNF01
Nombre del Requerimiento:	Rendimiento
Descripción del requerimiento:	El módulo de software debe proporcionar un tiempo de respuesta aceptable aproximadamente entre 2 a 12 segundos.
Prioridad del requerimiento:	Alta

Tabla 36. Requisito no funcional Usabilidad

Identificación del requerimiento:	RNF02
Nombre del Requerimiento:	Usabilidad
Descripción del requerimiento:	El módulo de software debe proveer una interfaz amigable e intuitiva, haciendo que el proceso sea comprensible y fácil de llevar a cabo.
Prioridad del requerimiento:	Alta

Tabla 37. Requisito no funcional Seguridad

Identificación del requerimiento:	RNF04
Nombre del Requerimiento:	Seguridad
Descripción del requerimiento:	El módulo de software debe garantizar la autentificación de usuario mendiante usuario y contraseña.
Prioridad del requerimiento:	Alta

4. Entorno Operativo

4.1 De control de acceso

Es necesario que el sistema realice una gestión adecuada de los permisos de cada usuario para garantizar su accesibilidad únicamente a la información correspondiente a su rol, evitando así el acceso a datos o funciones que no le corresponde.

4.1.1 Historias de usuario

ID: H001	**Inicio de sesión**		
Usuario / Rol	Administrador	**Prioridad**	Alta
Descripción	Como administrador, quiero poder iniciar sesión en el sistema para acceder a las funcionalidades reservadas para mi rol.		
Nro	**Escenario**	**Criterio de aceptación**	
1	Ingresar credenciales válidas	● El sistema mostrará un formulario de inicio de sesión. ● El usuario ingresará su nombre de usuario y contraseña. ● Si las credenciales son válidas, el sistema permitirá el acceso al sistema. ● Si las credenciales son inválidas, se mostrará un mensaje de error y se impedirá el acceso al sistema.	

ID: H002	Registrar datos de la avícola		
Usuario / Rol	Administrador	**Prioridad**	Alta
Descripción	Como administrador, quiero poder registrar los datos esenciales de la avícola, incluyendo lote y galpón, para mantener un registro organizado de la información.		
Nro	**Escenario**	**Criterio de aceptación**	
1	Registrar datos de la avícola	• El sistema proporcionará un formulario para ingresar los datos de la avícola, como el lote y el galpón. • El administrador completará los campos requeridos. • Una vez ingresados los datos, el sistema almacenará la información de la avícola en la base de datos.	

ID: H003	Generar reportes generales		
Usuario / Rol	Administrador, Veterinario	**Prioridad**	Alta
Descripción	Como administrador o veterinario, quiero poder generar reportes generales que contengan información relevante sobre la avícola y su funcionamiento, incluyendo datos sobre mortalidad, alimentación, peso, Identificacion del tamaño, enfermedades y tratamientos.		
Nro	**Escenario**	**Criterio de aceptación**	
1	Generar reportes generales	• El sistema proporcionará opciones para seleccionar los tipos de reportes deseados (mortalidad, alimentación, peso, identificar tamaño, enfermedades, tratamientos, etc.). • El usuario seleccionará los tipos de reportes que desea generar. • El sistema procesará la solicitud y generará los reportes correspondientes. • Los reportes se mostrarán en pantalla o se descargarán en formato PDF u otro formato adecuado.	

ID: H004	Registrar control de crecimiento		
Usuario / Rol	Galponero	**Prioridad**	Alta
Descripción	Como galponero, quiero poder registrar datos de control de crecimiento de los pollos, incluyendo información sobre la mortalidad diaria, recepción de alimentación diaria de las aves y peso promedio.		
Nro	**Escenario**	**Criterio de aceptación**	
1	Registrar control de crecimiento	• El sistema proporcionará formularios para ingresar los datos de control de crecimiento, como la mortalidad diaria, recepción de alimentación y peso promedio, identificacion de edad. • El galponero completará los campos requeridos. • Una vez ingresados los datos, el sistema almacenará la información del control de crecimiento de los pollos en la base de datos.	

ID: H005	Registrar mortalidad		
Usuario / Rol	Galponero	**Prioridad**	Alta
Descripción	Como galponero, quiero poder ingresar los datos diarios de mortalidad y descarte de los pollos, así como llevar un registro del saldo de mortalidad.		
Nro	**Escenario**	**Criterio de aceptación**	
1	Registrar mortalidad	• El sistema proporcionará un formulario para ingresar los datos diarios de mortalidad y descarte de los pollos. • El galponero completará los campos requeridos, como la cantidad de pollos muertos y los descartes de la mortalidad. • Una vez ingresados los datos, el sistema actualizará el registro de mortalidad y calculará el saldo de mortalidad.	

ID: H006	Registrar alimentación		
Usuario / Rol	Galponero	**Prioridad**	Alta
Descripción	Como galponero, quiero poder registrar la información de alimentación de los pollos, incluyendo el conductor, la cantidad de ingreso, la cantidad consumida, la fecha y hora de ingreso, y el saldo balanceado, con el fin de tener un reporte diario de consumo, asi como total y total de balanceado,		
Nro	**Escenario**	**Criterio de aceptación**	
1	Registrar alimentación	• El sistema proporcionará un formulario para ingresar los datos de alimentación de los pollos. • El galponero completará los campos requeridos, como el conductor responsable, la cantidad de alimento ingresado y la cantidad consumida por los pollos. • También se registrarán la fecha y hora de ingreso del alimento. • Una vez ingresados los datos, el sistema calculará el saldo balanceado de alimento.	

ID: H007	Registrar peso		
Usuario / Rol	Galponero	**Prioridad**	Alta
Descripción	Como galponero, quiero poder registrar el peso y la imagen de los pollos, lo que me permitirá evaluar su crecimiento en edad y fase.		
Nro	**Escenario**	**Criterio de aceptación**	
1	Registrar peso	• El sistema proporcionará un formulario para ingresar los datos de peso de los pollos y subir la imagen • El galponero ingresará el peso y la imagen de los pollos, permitiendo la evaluación de edad y la fase, registrando la información correspondiente. • Una vez ingresados los datos, el sistema almacenará la información del peso de los pollos en la base de datos.	

ID: H008	Evaluar imagen		
Usuario / Rol	Galponero	**Prioridad**	Alta
Descripción	Como galponero, quiero poder evaluar imágenes de los pollos en diferentes fases de desarrollo para identificar su tamaño y evaluar su crecimiento.		
Nro	**Escenario**	**Criterio de aceptación**	
1	Evaluar imagen de pollo.	• El sistema proporcionará una función para cargar y visualizar imágenes de los pollos. • El galponero seleccionará las imágenes correspondientes a los pollos en diferentes fases de crecimiento • Utilizando inteligencia artificial, el sistema evaluará las imágenes y determinará el tamaño de los pollos, permitiendo así evaluar su crecimiento.	

ID: H09	Ingresar información del diagnóstico de anomalías		
Usuario / Rol	Veterinario	**Prioridad**	Alta
Descripción	Como veterinario, quiero poder ingresar información detallada sobre el diagnóstico de anomalías detectadas en los pollos, proporcionando datos relevantes para su seguimiento y tratamiento.		
Nro	**Escenario**	**Criterio de aceptación**	
1	Ingresar información del diagnóstico de anomalías	• El sistema proporcionará un formulario para ingresar información detallada sobre el diagnóstico de anomalías en los pollos. • El veterinario completará los campos requeridos, como la descripción de la anomalía, los síntomas observados y los tratamientos aplicados. • Una vez ingresada la información, el sistema almacenará los datos para su posterior seguimiento y tratamiento.	

ID: H010	Buscar anomalías		
Usuario / Rol	Veterinario	**Prioridad**	Alta
Descripción	Como veterinario, quiero poder ingresar información en caso de que exista alguna enfermedad nueva.		
Nro	**Escenario**	**Criterio de aceptación**	
1	Agregar nueva Enfermedad	• El sistema proporcionará una función de agregar nueva enfermedad.	

5. Firmas de responsabilidad

Acción	Funcionario	Firmas
Elaborado por:	Viviana Maricela Zambrano Romero **Analista y Desarrollador de Software**	
Revisado por:	Ing. Edwin René Guamán Quinche, Mg.Sc **Director del Trabajo de Integración Curricular**	
Revisado por:	Sr. Pinto Añazco Alexis Jeovanny **Representada legalmente de la Granja "Las Acacias"**	

Anexo 4. Documentación de arquitectura de software

Arquitectura de software
Modelo arquitectónico 4+1

Proyecto: Software para el Control de Crecimiento en pollos de engorde en la granja Avícola "Las Acacias" en el Cantón Balsas

Versión: 1.0

Fecha: 03/04/2023

Ficha del documento

Versión	Fecha de revisión	Responsables	Descripción de modificación
1.0	03/04/2023	Viviana Maricela Zambrano Romero	N/A

Índice de Contenidos

Índice de Figuras

Índice de Tablas

1. Introducción

Este documento es una Arquitectura de Software para el Software para el Control de Crecimiento en pollos de engorde en la granja Avícola “Las Acacias” en el Cantón Balsas.

La arquitectura del sistema se resume en muchas vistas y componentes que se explican en detalle. El documento sigue el modelo de vista 4 + 1 como modelo de referencia para este documento.

1.1 Propósito

Este documento proporciona una descripción arquitectónica completa del sistema, utilizando una serie de vistas arquitectónicas diferentes para representar diferentes aspectos del sistema. Está pensado para capturar y transmitir las importantes decisiones arquitectónicas que se han tomado en el sistema.

Este documento elabora la arquitectura del sistema en 5 vistas diferentes. (4 + 1 modelo de vista). El comportamiento estático y dinámico del sistema se describe en este documento. Todos los diagramas requeridos y sus descripciones están disponibles en este documento.

El uso del modelo de vista 4 + 1 permite representar el software con la mayor precisión posible. Permite a una amplia gama de partes interesadas encontrar lo que necesitan en el documento de arquitectura.

1.2 Alcance

El documento de arquitectura de software se aplica a cada aspecto estático y dinámico del sistema. Dado que el modelo de vista 4 + 1 se utiliza como modelo de referencia, incorpora muchas vistas del sistema, lo que hace que el documento sea completo y coherente.

Bajo el comportamiento estático del sistema, el documento analiza los diagramas de clase, los diagramas de paquetes y otros diseños de arquitectura estática. Los aspectos dinámicos del sistema se elaboran utilizando realizaciones de casos de uso y diagramas de secuencia del sistema.

1.3 Personal involucrado

Tabla 1.
Personal involucrado estudiante de la CIS

Nombre	Viviana Maricela Zambrano Romero
Rol	Analista y Desarrollador de Software
Categoría Profesional	Estudiante de la CIS
Responsabilidad	Análisis de información, diseño y programación del módulo de software
Información de contacto	viviana.zambrano@unl.edu.ec

Tabla 2.
Personal involucrado docente de la CIS

Nombre	Edwin René Guamán Quinche
Rol	Director del trabajo de titulación curricular
Categoría Profesional	Docente de la de la CIS/C
Responsabilidad	Supervisar y asesorar en el desarrollo del Trabajo de Titulación
Información de contacto	rguaman@unl.edu.ec

1.4 Definiciones, acrónimos y abreviaturas

Tabla 3.
Definiciones, acrónimos y abreviaturas

Nombre	**Descripción**
AI	Inteligencia Artificial
CONAVE	Corporación Nacional de Avicultores del Ecuador
CIS/C	Carrera de Ingeniería en Sistemas/Computación
ERS	Especificación de Requisitos Software
RF	Requerimiento Funcional
RNF	Requerimiento No Funcional

1.5 Resumen

El informe presentará un análisis detallado de la arquitectura del Software para el Control de crecimiento en pollos de engorde en la granja Avícola “Las Acacias” en el Cantón Balsas. Otras secciones cubren la representación arquitectónica del proyecto, incluida la representación arquitectónica y limitaciones arquitectónicas y las realizaciones de casos de uso. Las secciones posteriores cubren los detalles específicos detallados de las 4 vistas principales (vista lógica, vista de proceso, vista de desarrollo y vista de implementación) del sistema.

2 Representación Arquitectónica

Esta sección detalla la arquitectura usando las vistas definidas en el modelo “4 + 1”. Las vistas utilizadas para documentar el Software para el Control de Crecimiento en pollos de engorde en la granja Avícola “Las Acacias” en el Cantón Balsas.

2.5 Vista Escenarios

Audiencia:

Todas las partes interesadas del sistema, incluidos los usuarios finales.

Área:

Esta vista se centra en los casos de uso del sistema y está destinada a proporcionar una visión clara de cómo los actores interactúan con el software para lograr sus objetivos y las funcionalidades que el sistema debe ofrecer a los usuarios.

2.6 Vista Lógica

Audiencia

Arquitectos de Software, Desarrolladores, Analistas de Sistemas.

Área:

Esta vista describe la estructura interna del sistema y se enfoca en las entidades, clases, y sus relaciones, permitiendo comprender cómo se organiza y representa la información en el software.

2.7 Vista de Procesos

Audiencia:

Arquitectos de Software, Desarrolladores, Analistas de Sistemas, Ingenieros de Pruebas.

Área:

Esta vista representa el comportamiento dinámico del sistema, mostrando cómo los diferentes componentes interactúan entre sí y responden a eventos. Es útil para entender el flujo de control y la lógica de procesamiento dentro del software.

2.8 Vista de Desarrollo

Audiencia:

Desarrolladores, Gerentes de Desarrollo, Gerentes de Calidad.

Área

Esta vista se enfoca en la estructura del software desde una perspectiva de código fuente. Muestra cómo se organizan los módulos, bibliotecas y componentes del sistema, lo que es fundamental para el desarrollo, mantenimiento y reutilización del código.

2.9 Vista Física

Audiencia:

Administradores de Sistemas, Ingenieros de Infraestructura, Arquitectos de Infraestructura.

Área:

Esta vista representa la infraestructura física y el despliegue del sistema, mostrando cómo se distribuyen y conectan los componentes del software en el entorno de ejecución. Es relevante para la configuración, escalabilidad y disponibilidad del sistema en producción.

3 Metas y limitaciones arquitectónicas

3.1 Del lado del servicio:

Meta: Diseñar la arquitectura del servidor de manera que pueda adaptarse a un crecimiento futuro en el número de usuarios y la carga de trabajo sin comprometer el rendimiento.

Limitación: Se debe considerar la capacidad de escalamiento horizontal y vertical, utilizando técnicas como la adición de más servidores o la distribución de carga para garantizar que el sistema pueda manejar una mayor demanda de manera eficiente.

Meta: Implementar mecanismos para detectar y recuperarse de posibles fallos en el servidor, minimizando el impacto en el funcionamiento del sistema.

Limitación: La arquitectura del servidor debe incluir estrategias de recuperación y respaldo, como la replicación de datos y la planificación de puntos de recuperación, para asegurar que el sistema pueda mantener su operatividad en caso de errores o interrupciones.

3.2 Del lado del cliente:

Meta: Diseñar una interfaz de usuario que ofrezca una experiencia fluida y receptiva, independientemente del dispositivo o plataforma utilizado por el usuario.

Limitación: La arquitectura del cliente debe considerar la optimización del rendimiento y la eficiencia en el procesamiento de la interfaz de usuario para garantizar tiempos de respuesta rápidos y una navegación sin problemas.

Meta: Minimizar el consumo de recursos en el cliente, como memoria y energía, para garantizar un rendimiento óptimo y una experiencia de usuario satisfactoria.

Limitación: La arquitectura del cliente debe optimizar el uso de recursos del dispositivo, evitando operaciones innecesarias y priorizando la eficiencia en el procesamiento y almacenamiento de datos.

3.3 Seguridad:

Meta: Garantizar la seguridad y confidencialidad de los datos sensibles del sistema, así como protegerlo contra accesos no autorizados y ataques externos.

Limitación: El sistema debe cumplir con los estándares y mejores prácticas de seguridad establecidos por la industria.

3.4 Persistencia:

Meta: Asegurar la persistencia de los datos relevantes del sistema para que la información no se pierda después de la finalización de las operaciones o en caso de fallos.

Limitación: La solución de persistencia debe ser eficiente y escalable, garantizando una recuperación rápida y confiable de los datos almacenados.

3.5 Confiabilidad / Disponibilidad:

Meta: Garantizar la disponibilidad continua del sistema y minimizar el tiempo de inactividad para evitar interrupciones en el servicio.

Limitación: La arquitectura debe incluir mecanismos de redundancia y tolerancia a fallos para mantener la disponibilidad incluso en situaciones de alto tráfico o incidencias inesperadas.

3.6 Performance:

Meta: Lograr un alto rendimiento y tiempos de respuesta rápidos para satisfacer las necesidades de los usuarios y permitir un uso eficiente del sistema.

Limitación: La arquitectura debe optimizar el uso de recursos, minimizar cuellos de botella y evitar problemas de escalabilidad que puedan afectar negativamente el rendimiento.

3.7 Portabilidad y Reutilización:

Meta: Diseñar el sistema de forma modular y flexible para facilitar su portabilidad a diferentes plataformas y permitir la reutilización de componentes en futuros proyectos.

Limitación: La arquitectura debe adherirse a estándares y tecnologías ampliamente adoptadas para maximizar la compatibilidad y portabilidad del software.

3.8 Herramientas de Desarrollo:

Meta: Proporcionar herramientas de desarrollo eficientes y efectivas que permitan a los desarrolladores trabajar de manera productiva y colaborativa en el proyecto.

Limitación: Las herramientas seleccionadas deben ser adecuadas para el equipo de desarrollo y estar bien integradas con el flujo de trabajo, facilitando el desarrollo, depuración y pruebas del software.

4 Vistas

4.1 Vista de Escenarios

4.1.1 Casos de uso

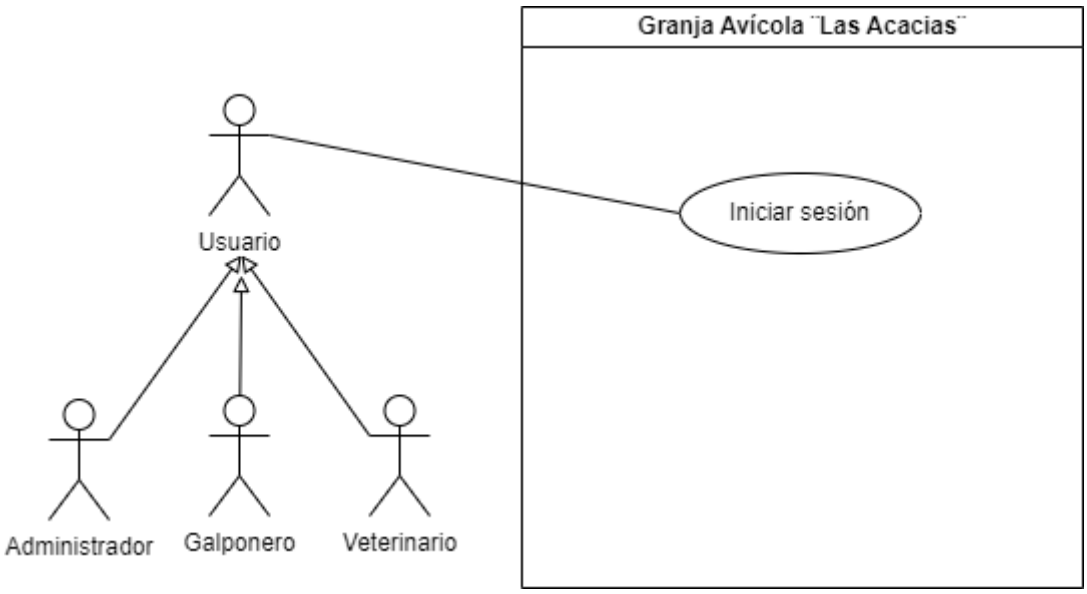


Figura 45. Rol Usuario

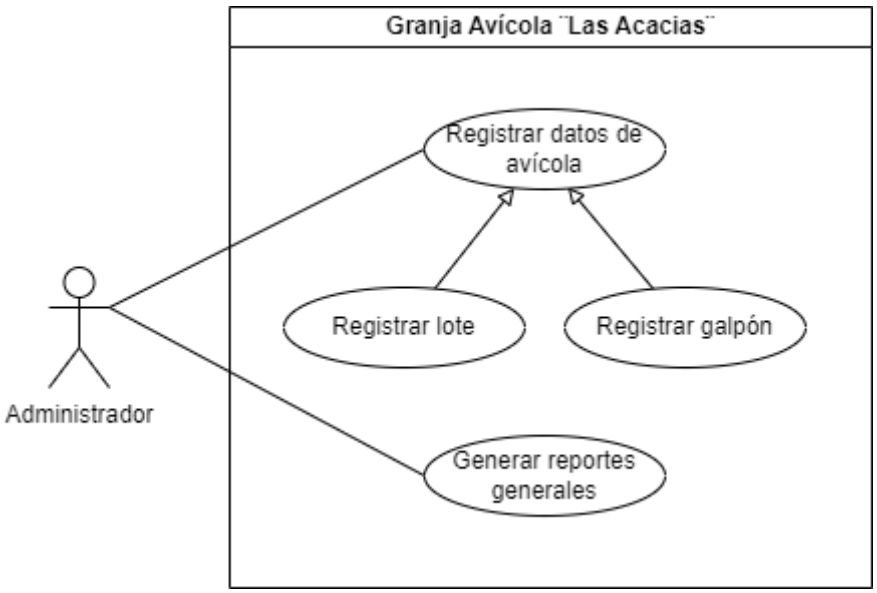


Figura 46. Rol Administrador

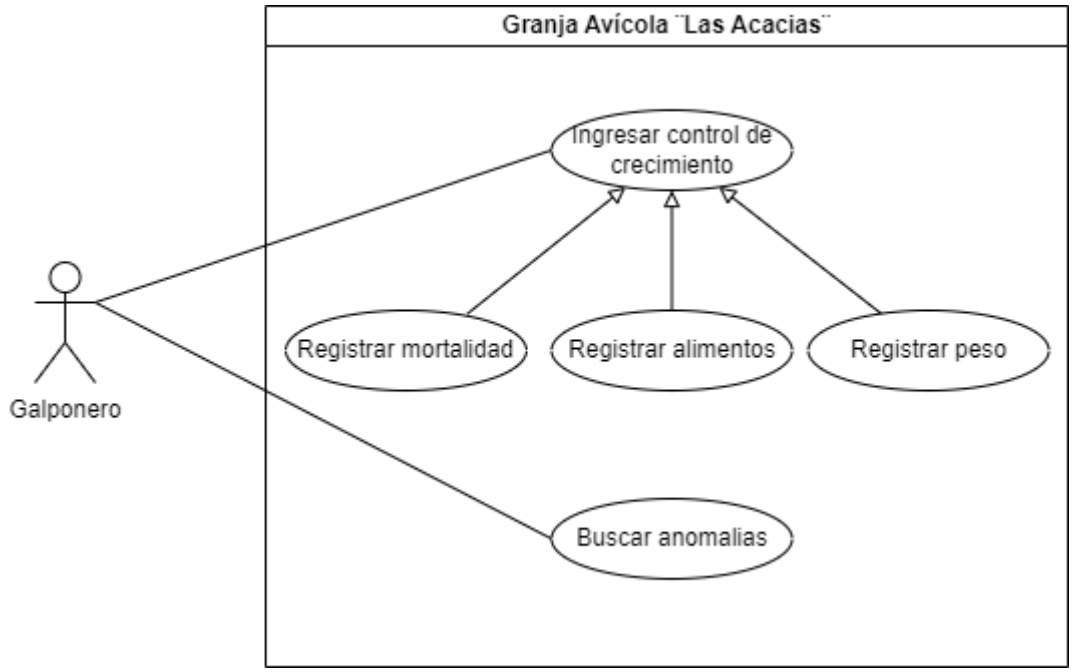


Figura 47. Rol Galponero

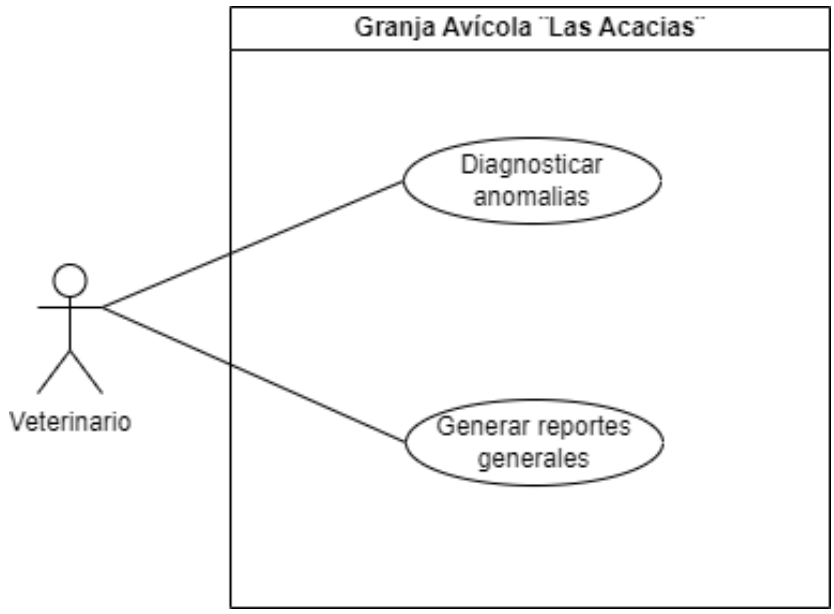


Figura 48. Rol Veterinario

4.1.2 Descripción de cada caso de uso

Tabla 38. Requisito funcional Inicio de sesión

Identificación del requerimiento:	RF01
Nombre del Requerimiento:	Inicio de sesión
Descripción del requerimiento:	El sistema permitirá ingresar a través de la validación de credenciales (usuario y contraseña).
Dependencias:	S/N
Requerimiento NO funcional:	• RNF03 • RNF04

Prioridad del requerimiento:	Alta

Tabla 39. Requisito funcional Registrar datos de la avícola

Identificación del requerimiento:	RF02
Nombre del Requerimiento:	Registrar datos de la avícola
Descripción del requerimiento:	El sistema permitirá al usuario Administrador registrar los datos esenciales de la avícola, incluyendo lote y galpón.
Dependencias:	• S/N
Requerimiento NO funcional:	• RNF01 • RNF02
Prioridad del requerimiento:	Alta

Tabla 40. Requisito funcional Generar reportes generales

Identificación del requerimiento:	RF03
Nombre del Requerimiento:	Generar reportes generales
Descripción del requerimiento:	El sistema permitirá al Administrador y Veterinario generar reportes generales que contengan información sobre la avícola y su funcionamiento referente a mortalidad, alimentación, peso, enfermedades y tratamiento.
Dependencias:	• S/N
Requerimiento NO funcional:	• RNF01 • RNF02
Prioridad del requerimiento:	Alta

Tabla 41. Requisito funcional Registrar control de crecimiento

Identificación del requerimiento:	RF04
Nombre del Requerimiento:	Registrar control de crecimiento
Descripción del requerimiento:	El sistema permitirá al galponero registrar datos de control de crecimiento de los pollos. (Mortalidad diaria, recepción de alimentación diaria de las aves, peso promedio)
Dependencias:	• S/N
Requerimiento NO funcional:	• RNF01 • RNF02
Prioridad del requerimiento:	Alta

Tabla 42. Requisito funcional Registrar mortalidad

Identificación del requerimiento:	RF05
Nombre del Requerimiento:	Registrar mortalidad
Descripción del requerimiento:	El sistema permitirá al galponero ingresar los datos diarios de mortalidad y descarte, saldo de mortalidad.
Dependencias:	• S/N
Requerimiento NO funcional:	• RNF01 • RNF02
Prioridad del requerimiento:	Alta

Tabla 43.Requisito funcional Registrar alimentación

Identificación del requerimiento:	RF06
Nombre del Requerimiento:	Registrar alimentación
Descripción del requerimiento:	El sistema permitirá al galponero registrar conductor, cantidad de ingreso, cantidad consumido, fecha ingreso, hora ingreso, hora fin, saldo balanceado
Dependencias:	• S/N
Requerimiento NO funcional:	• RNF01 • RNF02
Prioridad del requerimiento:	Alta

Tabla 44. Requisito funcional Registrar peso

Identificación del requerimiento:	RF07
Nombre del Requerimiento:	Registrar peso
Descripción del requerimiento:	El sistema permitirá al galponero registrar el peso de los pollos. (Calcular edad de crecimiento de pollos y evaluar fase con la imagen)
Dependencias:	• S/N
Requerimiento NO funcional:	• RNF01 • RNF02
Prioridad del requerimiento:	Alta

Tabla 45. Requisito funcional Calcular crecimiento de pollos

Identificación del requerimiento:	RF08
Nombre del Requerimiento:	Calcular crecimiento de pollos
Descripción del requerimiento:	El sistema permitirá calcular el crecimiento de los pollos en función de su edad, utilizando los datos de peso y control de crecimiento registrados previamente por el Galponero
Dependencias:	• S/N
Requerimiento NO funcional:	• RNF01 • RNF02
Prioridad del requerimiento:	Alta

Tabla 46. Requisito funcional Evaluar imagen

Identificación del requerimiento:	RF09
Nombre del Requerimiento:	Evaluar imagen
Descripción del requerimiento:	El sistema permitirá evaluar imágenes de los pollos en diferentes fases, a través de inteligencia artificial, e identificar el tamaño según su crecimiento.
Dependencias:	• S/N
Requerimiento NO funcional:	• RNF01 • RNF02
Prioridad del requerimiento:	Alta

Tabla 47.Requisito funcional Buscar anomalías

Identificación del requerimiento:	RF10
Nombre del Requerimiento:	Buscar anomalías
Descripción del requerimiento:	El sistema permitirá al galponero visualizar la información adicional referente a una enfermedad o alguna anomalía que se detectó y el tratamiento.
Dependencias:	• S/N
Requerimiento NO funcional:	• RNF01 • RNF02
Prioridad del requerimiento:	Alta

Tabla 48. Requisito funcional Ingresar información del diagnóstico de anomalías

Identificación del requerimiento:	RF11
Nombre del Requerimiento:	Ingresar información del diagnóstico de anomalías
Descripción del requerimiento:	El sistema permitirá al veterinario ingresar información sobre las enfermedades y tratamientos de los pollos.
Dependencias:	• S/N
Requerimiento NO funcional:	• RNF03 • RNF04
Prioridad del requerimiento:	Alta

4.2 Vista Lógica

4.2.1 Diagrama de Clases

Representan las clases del sistema, sus atributos y las relaciones entre ellas.

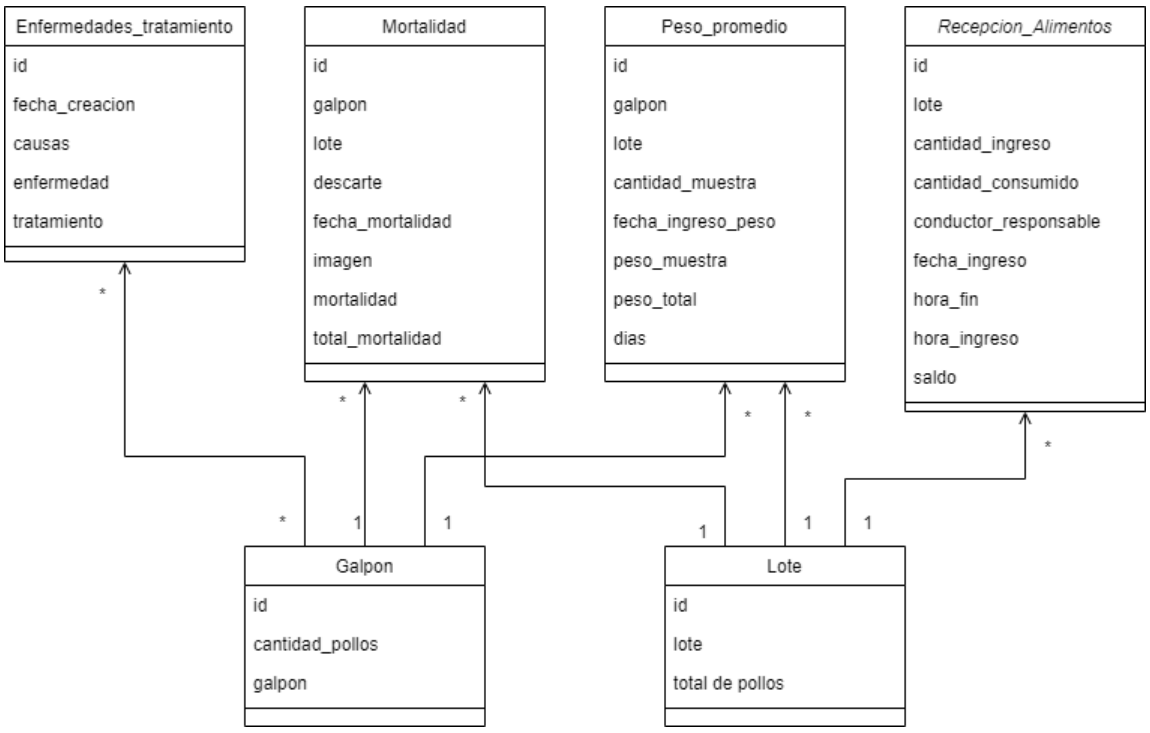


Figura 49. Diagrama de clases general

4.2.2 Diagrama Entidad-Relación

Utilizan para modelar las entidades y sus relaciones en una base de datos.

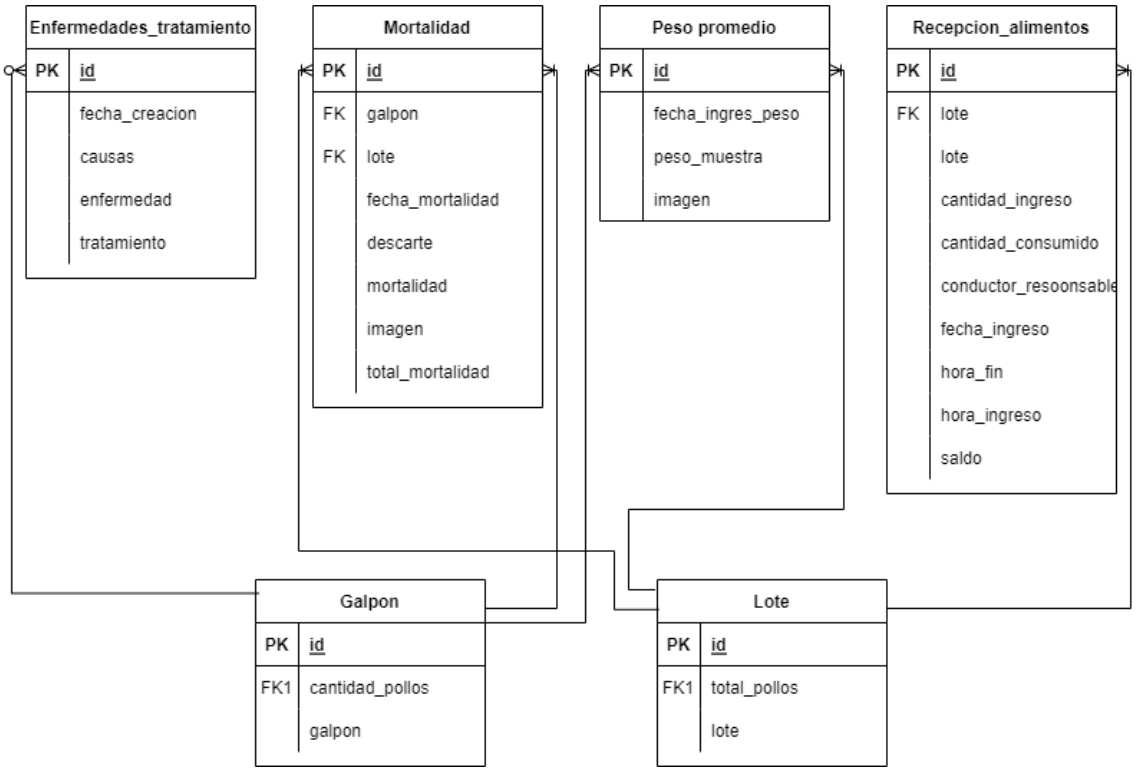


Figura 50. Diagrama entidad-relación general

4.3 Vista de Procesos

4.3.1 Diagrama de Secuencia

Muestran cómo los objetos del sistema interactúan entre sí y responden a eventos en el tiempo. Muestran el flujo de mensajes y llamadas entre objetos para completar una tarea

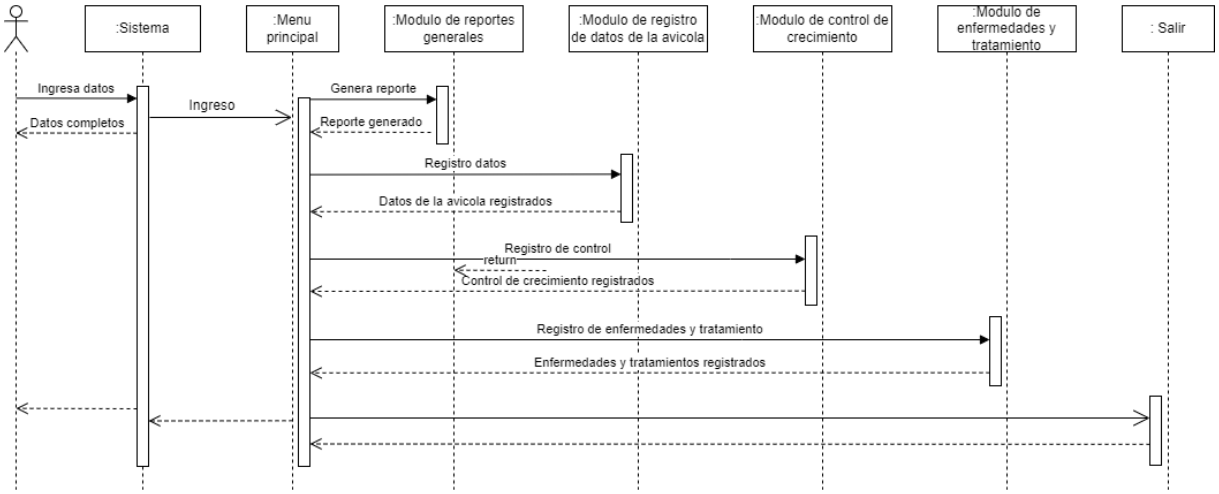


Figura 51. Diagrama de secuencia general

4.3.2 Diagrama de Actividades

Describen el flujo de actividades o procesos del sistema, mostrando el orden en el que se realizan las tareas y las decisiones que se toman.

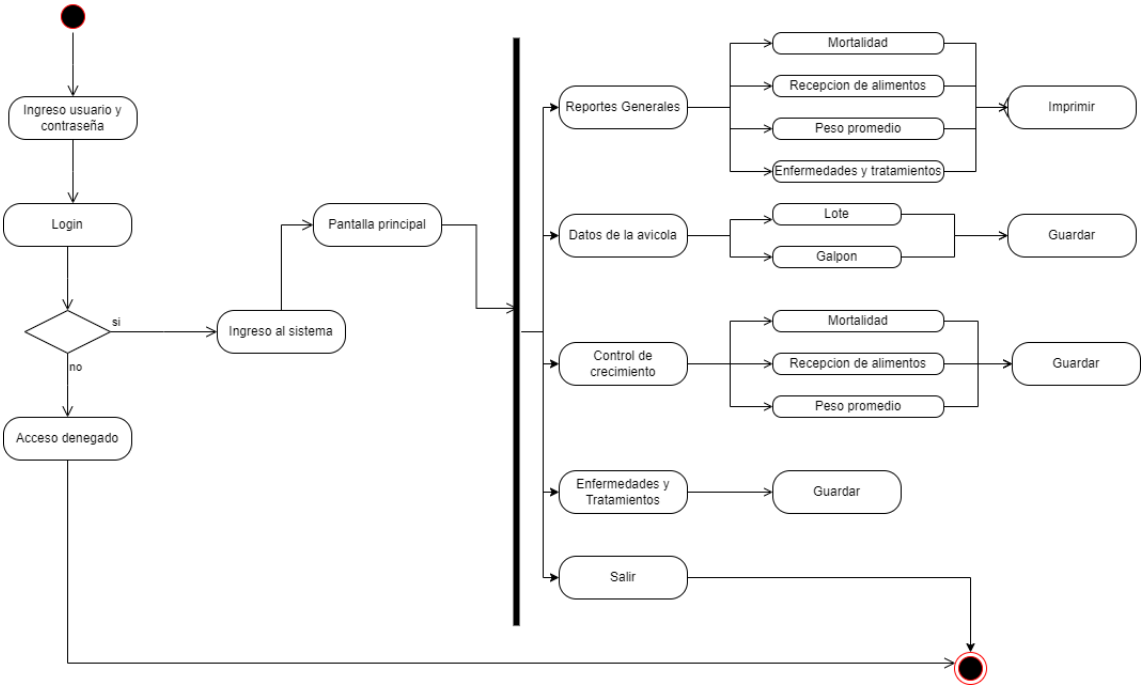


Figura 52. Diagrama de actividades general

4.4 Vista de Desarrollo

4.4.1 Diagrama de paquetes

Muestran la organización lógica del código en módulos o paquetes, lo que ayuda a entender cómo se agrupan las funcionalidades relacionadas.

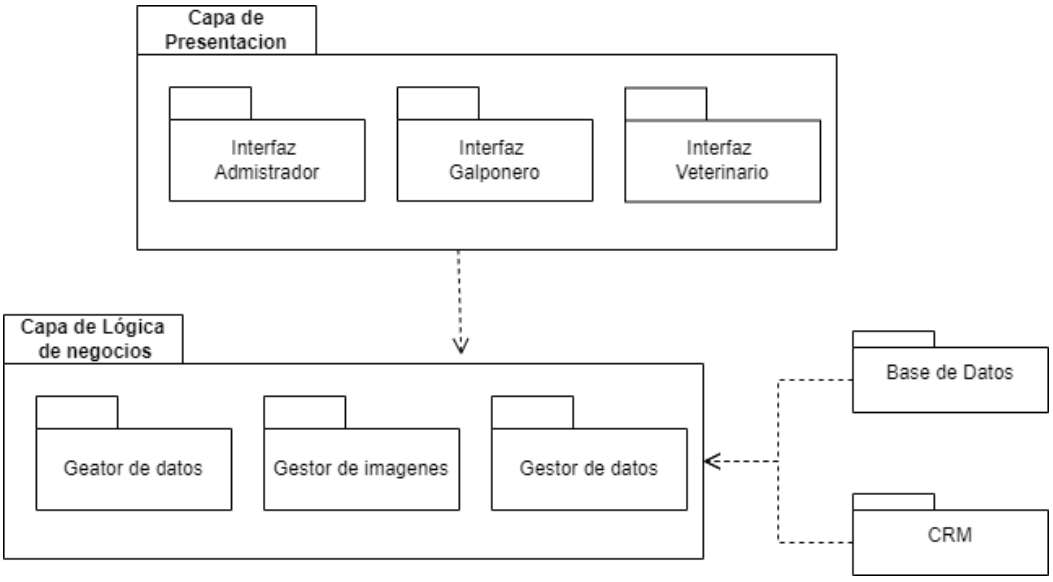


Figura 53. Diagrama de paquetes

4.5 Vista Física

4.5.1 Diagrama de despliegue

Muestran cómo se distribuyen y conectan los elementos del sistema en el entorno de ejecución, como servidores, dispositivos, redes, etc. Permiten comprender cómo el software se implementa y se despliega en una arquitectura física real.

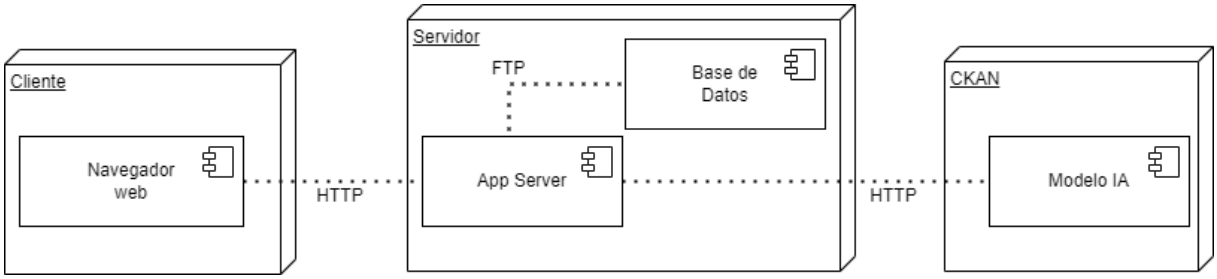


Figura 54. Diagrama de despliegue

5 Firmas de responsabilidad

Acción	Funcionario	Firmas
Elaborado por:	Viviana Maricela Zambrano Romero **Analista y Desarrollador de Software**	VIVIANA MARICELA ZAMBRANO ROMERO
Revisado por:	Ing. Edwin René Guamán Quinche, Mg.Sc **Director del Trabajo de Integración Curricular**	EDWIN RENE GUAMAN QUINCHE

Anexo 5. Código de programación del modelo de reconocimiento de imágenes mediante visión artificial para la identificación del crecimiento del tamaño del pollo

Código de programación del modelo de reconocimiento de imágenes mediante visión artificial para la identificación del crecimiento del tamaño del pollo

Proyecto: Software para el Control de Crecimiento en pollos de engorde en la granja Avícola "Las Acacias" en el Cantón Balsa

Fecha: 06/06/2023

Índice

1. Código fuente del modelo pre-entrenado

El código fuente se observa en la Tabla 1.

Tabla 1.
Código fuente del contrato Migrations.sol

```
#Crear las carpetas
!mkdir Engorde
!mkdir Desarrollo
!mkdir Inicio
!mkdir Pre-inicio

#Entrar en cada carpeta y descomprimir el archivo zip
%cd Engorde
!unzip Engorde.zip
%cd ..

%cd Desarrollo
!unzip Desarrollo.zip
%cd ..

%cd Inicio
!unzip Inicio.zip
%cd ..

%cd Pre-inicio
!unzip Pre-inicio.zip
%cd ..

#Listar cada carpeta
!ls /content/Engorde/ | wc -l #66
!ls /content/Desarrollo/ | wc -l #64
!ls /content/Inicio/ | wc -l #87
!ls /content/Pre-inicio// | wc -l #58

#Mostrar algunas imagenes con pyplot
import os
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

plt.figure(figsize=(15,15))

carpeta = '/content/Engorde'
imagenes = os.listdir(carpeta)

for i, nombreimg in enumerate(imagenes[:25]):
  plt.subplot(5,5,i+1)
  imagen = mpimg.imread(carpeta + '/' + nombreimg)
  plt.imshow(imagen)
```

```
#Crear carpetas para hacer el set de datos
!mkdir dataset
!mkdir dataset/Engorde
!mkdir dataset/Desarrollo
!mkdir dataset/Inicio
!mkdir dataset/Pre-inicio

#Copiar imagenes que subimos a carpetas del dataset
#Limitar para que todos tengan la misma cantidad de imagenes
#maximo 419 (el num. menor de imagenes que subi)
import shutil
carpeta_fuente = '/content/Engorde'
carpeta_destino = '/content/dataset/Engorde'

imagenes = os.listdir(carpeta_fuente)

for i, nombreimg in enumerate(imagenes):
  if i < 419:
    #Copia de la carpeta fuente a la destino
    shutil.copy(carpeta_fuente + '/' + nombreimg, carpeta_destino +
'/' + nombreimg)

#Copiar imagenes que subimos a carpetas del dataset
#Limitar para que todos tengan la misma cantidad de imagenes
#maximo 419 (el num. menor de imagenes que subi)
import shutil
carpeta_fuente = '/content/Desarrollo'
carpeta_destino = '/content/dataset/Desarrollo'

imagenes = os.listdir(carpeta_fuente)

for i, nombreimg in enumerate(imagenes):
  if i < 419:
    #Copia de la carpeta fuente a la destino
    shutil.copy(carpeta_fuente + '/' + nombreimg, carpeta_destino +
'/' + nombreimg)

#Copiar imagenes que subimos a carpetas del dataset
#Limitar para que todos tengan la misma cantidad de imagenes
#maximo 419 (el num. menor de imagenes que subi)
import shutil
carpeta_fuente = '/content/Inicio'
carpeta_destino = '/content/dataset/Inicio'

imagenes = os.listdir(carpeta_fuente)

for i, nombreimg in enumerate(imagenes):
  if i < 419:
    #Copia de la carpeta fuente a la destino
```

```
    shutil.copy(carpeta_fuente + '/' + nombreimg, carpeta_destino +
'/' + nombreimg)

#Copiar imagenes que subimos a carpetas del dataset
#Limitar para que todos tengan la misma cantidad de imagenes
#maximo 419 (el num. menor de imagenes que subi)
import shutil
carpeta_fuente = '/content/Pre-inicio'
carpeta_destino = '/content/dataset/Pre-inicio'

imagenes = os.listdir(carpeta_fuente)

for i, nombreimg in enumerate(imagenes):
  if i < 419:
    #Copia de la carpeta fuente a la destino
    shutil.copy(carpeta_fuente + '/' + nombreimg, carpeta_destino +
'/' + nombreimg)

#Mostrar cuantas imagenes tengo de cada categoria en el dataset
!ls /content/dataset/Engorde | wc -l
!ls /content/dataset/Desarrollo | wc -l
!ls /content/dataset/Inicio | wc -l
!ls /content/dataset/Pre-inicio | wc -l

#Aumento de datos con ImageDataGenerator
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np

#Crear el dataset generador
datagen = ImageDataGenerator(
    rescale=1. / 255,
    rotation_range = 30,
    width_shift_range = 0.25,
    height_shift_range = 0.25,
    shear_range = 15,
    zoom_range = [0.5, 1.5],
    validation_split=0.2 #20% para pruebas
)

#Generadores para sets de entrenamiento y pruebas
data_gen_entrenamiento =
datagen.flow_from_directory('/content/dataset', target_size=(224,224),
                                                    batch_size=32,
shuffle=True, subset='training')
data_gen_pruebas = datagen.flow_from_directory('/content/dataset',
target_size=(224,224),
                                                    batch_size=32,
shuffle=True, subset='validation')
```

```
#Imprimir 10 imagenes del generador de entrenamiento
for imagen, etiqueta in data_gen_entrenamiento:
  for i in range(10):
    plt.subplot(2,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(imagen[i])
  break
plt.show()

import tensorflow as tf
import tensorflow_hub as hub

url = "https://tfhub.dev/google/tf2-
preview/mobilenet_v2/feature_vector/4"
mobilenetv2 = hub.KerasLayer(url, input_shape=(224,224,3))

#Congelar el modelo descargado
mobilenetv2.trainable = False

modelo = tf.keras.Sequential([
    mobilenetv2,
    tf.keras.layers.Dense(4, activation='softmax')
])

modelo.summary()

#Compilar como siempre
modelo.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

# Visualizando el entrenamiento history
from keras.models import Sequential
from keras.layers import Dense
import matplotlib.pyplot as plt
import numpy

#Entrenar el modelo
EPOCAS = 50

historial = modelo.fit(
    data_gen_entrenamiento, epochs=EPOCAS, batch_size=32,
    validation_data=data_gen_pruebas
)

#Graficas de precisión
```

```
acc = historial.history['accuracy']
val_acc = historial.history['val_accuracy']

loss = historial.history['loss']
val_loss = historial.history['val_loss']

rango_epocas = range(50)

plt.figure(figsize=(8,8))
plt.subplot(1,2,1)
plt.plot(rango_epocas, acc, label='Precisión Entrenamiento')
plt.plot(rango_epocas, val_acc, label='Precisión Pruebas')
plt.legend(loc='lower right')
plt.title('Precisión de entrenamiento y pruebas')

plt.subplot(1,2,2)
plt.plot(rango_epocas, loss, label='Pérdida de entrenamiento')
plt.plot(rango_epocas, val_loss, label='Pérdida de pruebas')
plt.legend(loc='upper right')
plt.title('Pérdida de entrenamiento y pruebas')
plt.show()

#Categorizar una imagen de internet
from PIL import Image
import requests
from io import BytesIO
import cv2

def categorizar(url):
  respuesta = requests.get(url)
  img = Image.open(BytesIO(respuesta.content))
  img = np.array(img).astype(float)/255

  img = cv2.resize(img, (224,224))
  prediccion = modelo.predict(img.reshape(-1, 224, 224, 3))
  return np.argmax(prediccion[0], axis=-1)

# Prueba con una imagen de internet
url =
'https://t2.uc.ltmcdn.com/es/posts/4/9/6/como_cuidar_de_un_pollito_27694_600_square.jpg'
prediccion = categorizar (url)
print(prediccion)
```

1. Código fuente del modelo entrenado

El código fuente se observa en la Tabla 2.

Tabla 2.
Código fuente del contrato StorageCAD.sol

```
def predict_imagen2(request):
    if request.method == 'POST' and request.FILES['image']:
        # Obtener la imagen cargada por el usuario
        image_file = request.FILES['image']
        # Guardar la imagen en el directorio MEDIA_ROOT
        image_path = os.path.join(settings.MEDIA_ROOT, image_file.name)
        with open(image_path, 'wb') as f:
            f.write(image_file.read())
        # Especificar el dispositivo de E/S
        options =
tf.saved_model.LoadOptions(experimental_io_device='/job:localhost')
        # Cargar el modelo SavedModel
        model = tf.saved_model.load(os.path.join(settings.STATIC_ROOT,
'saved_model'), options=options)
        # Obtener la función predict del modelo
        predict_fn = model.signatures['serving_default']
        # Cargar y preprocesar la imagen de entrada
        image = tf.io.read_file(image_path)
        image = tf.image.decode_image(image, channels=3)
        image = tf.image.resize(image, (224, 224))
        image = image / 255.0                                # Normalizar la
imagen
        # Agregar una dimensión extra para que coincida con el tamaño de
lote esperado por el modelo
        image = tf.expand_dims(image, axis=0)
        # Realizar la predicción
        predictions = predict_fn(image)['dense']
        # Obtener la clase con mayor probabilidad
        predicted_class = tf.argmax(predictions, axis=1)[0].numpy()
        # Definir las etiquetas correspondientes a las clases
        labels = {
            3: 'Pre-inicio',
            2: 'Inicio',
            1: 'Engorde',
            0: 'Desarrollo'
        }
        # Obtener la etiqueta correspondiente a la clase predicha
        predicted_label = labels[predicted_class]
        # Eliminar el archivo de imagen después de la predicción
        os.remove(image_path)
        # Guardar la predicción y el enlace de la imagen en el modelo
Prediccion
```

```
        127ndice127t127127 = Prediccion()
        127ndice127t127127.imagen = image_file
        127ndice127t127127.etiqueta = predicted_label
        127ndice127t127127.save()
        # Renderizar el resultado en el template
        return render(request, 'prediccion/result.html', {'prediccion':
prediccion, 'predicted_label': predicted_label})
    return render(request, 'prediccion/upload.html')
```

Anexo 6. Código de programación del software para el control de crecimiento utilizando la metodología XP mediante el lenguaje de programación Python.

Código de programación software para el control de crecimiento utilizando la metodología XP mediante el lenguaje de programación Python.

Proyecto: Software para el Control de Crecimiento en pollos de engorde en la granja Avícola "Las Acacias" en el Cantón Balsa

Fecha: 24/07/2023

Índice

2. Código fuente del Forms del software de control de crecimiento

El código fuente del forms se observa en la Tabla 1.

Tabla 1.Código fuente

```
from 130ndice import forms
from .models import *
from 130ndice.forms import fields
from 130ndice.contrib.auth.models import User
from 130ndice.contrib.auth.forms import UserCreationForm
from collections.abc import Mapping

class Reporte_generalForm(forms.ModelForm):

    class Meta:
        model = Reportes_generales
       # fields = [“vehiculo”, “fecha_reserva”, “hora_reserva”]
        fields = ‘__all__’

        widgets = {
            ‘fecha_creacion_dep’: forms.DateInput(format=(‘%d-%m-%Y’),
attrs={‘class’: ‘form-control’, ‘type’:’date’}),
       }

class MortalidadForm(forms.ModelForm):

    class Meta:
        model = Mortalidad
        fields = [“lote”, “130ndice”, “fecha_mortalidad”, “mortalidad”,
“descarte”,”imagen”]
        #fields = ‘__all__’

        widgets = {
            ‘fecha_mortalidad’: forms.DateInput(format=(‘%d-%m-%Y’),
attrs={‘class’: ‘form-control’, ‘type’:’date’}),
       }

class WeightForm(forms.Form):
    peso = forms.FloatField(label=’Peso’)

class LoteForm(forms.ModelForm):
    class Meta:
        model = Lote
        fields = [‘lote’, ‘total_de_pollos’]

class GalponForm(forms.ModelForm):
    class Meta:
        model = Galpon
        fields = [‘galpon’, ‘cantidad_pollos’]
```

```
class RecepcionAlimentosForm(forms.ModelForm):
    fecha_ingreso = forms.DateField(widget=forms.DateInput(attrs={‘type’:
‘date’}))
    hora_ingreso = forms.TimeField(widget=forms.TimeInput(attrs={‘type’:
‘time’}))
    hora_fin = forms.TimeField(widget=forms.TimeInput(attrs={‘type’:
‘time’}))
    class Meta:
        model = Recepcion_Alimentos
        fields = ‘__all__’

class Predict2Form(forms.ModelForm):
    class Meta:
        model = Predict2
        fields = ‘__all__’
```

3. Código fuente Models del software de control de crecimiento

El código fuente se observa en la Tabla 2.

Tabla 2.Código fuente

```
from 131ndice.db import models
from email.mime import audio
from enum import auto
from posixpath import split
from pyexpat import model
from statistics import mode
from 131ndice.db import models
from 131ndice.contrib.auth.models import Group, Permission, User
from 131ndice.contrib.contenttypes.models import ContentType
from 131ndice.db.models.deletion import CASCADE
from datetime import datetime
from 131ndice.contrib.auth.models import AbstractUser, Group
from collections.abc import Mapping

# Create your models here.

Class Galpon(models.Model):
    galpon = models.IntegerField(verbose_name=’Galpón’)
    cantidad_pollos = models.IntegerField(verbose_name=’Cantidad de Pollos’)
    def __str__(self):
        return f’El Galpón N°{self.galpon}’

class Lote(models.Model):
    lote = models.CharField(verbose_name=’Lote’, max_length=200)
    total_de_pollos = models.IntegerField(verbose_name=’Total de pollos en
el Lote’)
    def __str__(self):
```

```
        return  f'El Lote es { self.lote}'

class Reportes_generales (models.Model):
    ENFERMEDAD =[
        ('Colibacilosis','Colibacilosis'),
        ('Mycoplasmosis','Mycoplasmosis'),
        ('Cólera Aviar','Cólera Aviar'),
        ('Salmonelosis','Salmonelosis'),
        ('Tifoidea Aviar','Tifoidea Aviar'),
        ('Coriza infeccioso','Coriza infeccioso'),
        ('Enteritis necrótica y ulcerativa','Enteritis necrótica y
ulcerativa'),
        ('Gripe Aviar','Gripe Aviar'),
        ('Leucosis linfoide','Leucosis linfoide'),
        ('Gumboro o bursitis','Gumboro o bursitis'),
        ('Enfermedad de Newcastle','Enfermedad de Newcastle'),
        ('Influenza Aviar','Influenza Aviar'),
        ('Enfermedad de Marek','Enfermedad de Marek'),
        ('Bronquitis infecciosa','Bronquitis infecciosa'),
        ('Aspergilosis','Aspergilosis'),
        ('Moniliasis','Moniliasis'),
        ('Micotoxicosis','Micotoxicosis'),
        ('Ascaridiosis','Ascaridiosis'),
        ('Coccidiosis','Coccidiosis'),
        ('Escarabajos tenebriónidos en patología aviar','Escarabajos
tenebriónidos en patología aviar'),
        ('Heterakidosis','Heterakidosis'),
        ('Histomonosis','Histomonosis'),
        ('Knemidokoptosis','Knemidokoptosis'),
        ('Raillietiniasis','Raillietiniasis'),
        ('Tricomoniasis','Tricomoniasis')
    ]
    fecha_creacion_dep = models.DateField(verbose_name='Fecha de creación')
    lote= models.ForeignKey(Lote, on_delete=models.CASCADE)
    galpon = models.ForeignKey(Galpon, on_delete=models.CASCADE)
    enfermedad = models.CharField(max_length=100, verbose_name='Enfermedad',
choices=ENFERMEDAD)
    causas_mortalidad = models.TextField(verbose_name='Causas de la
mortalidad')
    tratamiento = models.TextField(verbose_name="Tratamiento")
    def __str__(self):
        return self.enfermedad
class Mortalidad(models.Model):
    lote= models.ForeignKey(Lote, on_delete=models.CASCADE)
    galpon = models.ForeignKey(Galpon, on_delete=models.CASCADE)
    fecha_mortalidad = models.DateField(verbose_name='Fecha
Mortalidad',null=True, blank=True)
    mortalidad = models.IntegerField(verbose_name='Cantidad de Mortalidad')
    descarte = models.IntegerField(verbose_name='Cantidad de Descarte')
```

```
    total_moratalidad = models.IntegerField(verbose_name='Total de
Mortalidad existente', null=True, blank=True)
    imagen = models.FileField(upload_to="Mortalidadfile/")
    @property

    def valor_promedio(self):
        return (self.mortalidad + self.descarte)

    def save(self):
        self.total_moratalidad = self.valor_promedio
        # llamamos al 133ndice super que permite capturar todos lo cambios
en una clase
        super(Mortalidad, self).save()

    def __str__(self):
        return f'{self.fecha_mortalidad}'

class Recepcion_Alimentos(models.Model):
    conductor_reponsable = models.CharField(verbose_name='Conductor
Responsable', max_length=200)
    cantidad = models.IntegerField(verbose_name='Cantidad de Ingreso')
    cantidad_consumido = models.IntegerField(verbose_name='Cantidad de
Consumido')
    lote= models.ForeignKey(Lote, on_delete=models.CASCADE)
    fecha_ingreso = models.DateField(verbose_name='Fecha Ingreso',null=True,
blank=True)
    hora_ingreso = models.TimeField(verbose_name='Hora Ingreso',null=True,
blank=True)
    hora_fin = models.TimeField(verbose_name='Hora Fin',null=True,
blank=True)
    saldo = models.IntegerField(verbose_name='Saldo Balanceado', null=True,
blank=True)

    @property
    def valor_restar(self):
        return (self.cantidad - self.cantidad_consumido)
    def save(self):
        self.saldo = self.valor_restar
        # llamamos al 133ndice super que permite capturar todos lo cambios
en una clase
        super(Recepcion_Alimentos, self).save()

    def __str__(self):
        return self.conductor_reponsable

class Peso_promedio(models.Model):
    SEMANAS =[
        ('DIA 0','DIA 0'),
        ('DIA 1','DIA 1'),
```

```
        ('DIA 2','DIA 2'),
    ]
    fecha_ingreso_peso = models.DateField(null=True, blank=True)
    lote= models.ForeignKey(Lote, on_delete=models.CASCADE)
    galpon= models.ForeignKey(Galpon, on_delete=models.CASCADE)
    semana = models.CharField(max_length=100, verbose_name='Semana',
choices=SEMANAS)
    cantidad_muestra = models.IntegerField(verbose_name='Cantidad de la
Muestra')
    peso_muestra = models.DecimalField(verbose_name='Peso total de la
Muestra', max_digits=6, decimal_places=2)
    peso_total = models.DecimalField(verbose_name='Peso total de la
Muestra', max_digits=6, decimal_places=2, null=True, blank=True)

    @property

    def valor_promedio(self):
        return (self.peso_muestra / self.cantidad_muestra)

    def 134ndi(self):
        self.peso_total = self.valor_promedio
        # llamamos al 134ndice super que permite capturar todos lo cambios
en una clase
        super(Peso_promedio, self).save()

    def __str__(self):
        return self.semana

class Prediccion(models.Model):
    imagen = models.ImageField(upload_to="productos", null=True)
    etiqueta = models.CharField(verbose_name='Prediccion', max_length=200)

    def __str__(self):
        return self.etiqueta

class AgePrediction(models.Model):
    peso = models.FloatField()
    edad_predicha = models.IntegerField()
    fecha_ingreso = models.DateTimeField(verbose_name='Hora
Ingreso',null=True, blank=True)
    creado_en = models.DateTimeField(auto_now_add=True)
    def __str__(self):
        return  f'La edad en peso es { self.edad_predicha}'
class Predict2(models.Model):
    imagen = models.ImageField(upload_to='images/')
    etiqueta = models.CharField(max_length=255)
    age_prediction = models.OneToOneField('AgePrediction',
on_delete=models.CASCADE, null=True, blank=True)
```

```
    def __str__(self):
        return f'Prediccion – Etiqueta: {self.etiqueta}'
```

4. Código fuente del Urls del software de control de crecimiento

El código fuente se observa en la Tabla 3.

Tabla 3. Código fuente

```
from 135ndice.conf import settings
from . import views
from 135ndice.contrib import messages, admin
from 135ndice.contrib.messages import success
from 135ndice.views.static import serve
from app.views import *
from 135ndice.urls import include, path, re_path, reverse
#from turisticos.agenda.views import
urlpatterns = [
    path('', home, name="home"),
    path('galponero/', galponero, name="galponero"),
    path('admin1/', admin1, name="admin1"),
    path('veterinario/', veterinario, name="veterinario"),
    path('lote_galpon/', lote_galpon, name="lote_galpon"),
    path('m_crecimiento/', m_crecimiento, name="m_crecimiento"),
    path('m_reportes_generales/', m_reportes_generales,
name="m_reportes_generales"),
    path('accounts/', include('django.contrib.auth.urls')),
    path('imprimir_pdf/', imprimir_pdf, name='imprimir_pdf'),
    path('agReporte/', agReporte, name="agReporte"),
    path('agMortalidad/', agMortalidad, name="agMortalidad"),
    path('vista_reporte/', vista_reporte, name= "vista_reporte"),
    path('reporte/', views.Reporte_pdf.as_view(), name='reporte'),
    path('edReporte/<id>/', edReporte, name="edReporte"),
    path('verMortalidad/', verMortalidad, name= "verMortalidad"),
    path('edMortalidad/<id>/', edMortalidad, name= "edMortalidad"),
    path('edPredict2/<id>/', edPredict2, name= "edPredict2"),
    path('edrecepcion/<id>/', edrecepcion, name= "edrecepcion"),
    #prediciones
    path('predict2/', predict_imagen2, name='predict_imagen2'),
    path('predecir_edad/', views.predecir_edad, name='predecir_edad'),
    path('predic_a_index', predic_a_index, name='predic_a_index'),
    path('predict_a2', predict_a2, name='predict_a2'),
    #ejemplo de 135ndice135t135135 unido
    path('predic_a_index2', predic_a_index2, name='predic_a_index2'),
    path('predict_imagen3', predict_imagen3, name='predict_imagen3'),
    path('crear-lote/', views.crear_lote, name='crear_lote'),
    path('crear-galpon/', views.crear_galpon, name='crear_galpon'),
```

```
    path('crear_recepcion_alimentos/', crear_recepcion_alimentos,
name='crear_recepcion_alimentos'),
    path('detalle_recepcion/', views.detalle_recepcion,
name='detalle_recepcion'),
    path('suma/<int:136ndice_id>/', views.suma_valores,
name='suma_valores'),
    path('reporte_alimentos_pdf', views.reporte_alimentos_pdf,
name='reporte_alimentos_pdf'),
    path('reporte_mortalidad_pdf', views.reporte_mortalidad_pdf,
name='reporte_mortalidad_pdf'),
    path('verprediccion/', views.verprediccion, name='verprediccion'),
    path('m_reportes_generales_vete/', views.m_reportes_generales_vete,
name='m_reportes_generales_vete'),
    path('verTratamiento/', verTratamiento, name='verTratamiento'),
    path('tratamiento_pdf/', tratamiento_pdf, name='tratamiento_pdf'),
    path('verprediccion_pdf/', verprediccion_pdf, name='verprediccion_pdf'),
    path('ver_Lote/', ver_Lote, name='ver_Lote'),
    path('ver_Galpon/', ver_Galpon, name='ver_Galpon'),
    path('m_crecimiento_galponero/', m_crecimiento_galponero,
name='m_crecimiento_galponero'),
```

5. Código fuente del Urls del software de control de crecimiento

El código fuente de urls se observa en la Tabla 4.

Tabla 4. Código fuente

```
#from typing_extensions import required
from 136ndice.contrib.auth.decorators import login_required,
permission_required, user_passes_test
from 136ndice.shortcuts import redirect, render, get_object_or_404
from 136ndice.contrib.auth.mixins import LoginRequiredMixin
from 136ndice.views.generic import CreateView, DetailView
from 136ndice.core.files.storage import FileSystemStorage
from 136ndice.contrib.auth import authenticate, login
from 136ndice.conf import UserSettingsHolder, settings
from 136ndice.http import JsonResponse, HttpResponse
from 136ndice.template.loader import render_to_string, get_template
from 136ndice.forms.models import construct_instance
from 136ndice.utils.encoding import smart_str
from app.pre.predecir_edad import predecir_edad
from 136ndice.contrib.messages.api import success
from 136ndice.http.response import HttpResponse
from 136ndice.forms.widgets import PasswordInput
from 136ndice.shortcuts import render, redirect
from 136ndice.contrib.auth.models import User
from 136ndice.contrib import admin, messages
from 136ndice.http.request import HttpRequest
from 136ndice.contrib.messages import success
```

```
from mimetypes import guess_all_extensions
from 137ndice.core.mail import send_mail
from reportlab.pdfgen import canvas
from 137ndice.db.models import Sum
from 137ndice.conf import settings
from .utils import render_to_pdf
from 137ndice.views import View
from 137ndice.db import models
from tensorflow import keras
from tempfile import tempdir
from urllib import request
from xhtml2pdf import pisa
from operator import ipow
from pyexpat import model
import tensorflow as tf
from http import server
from re import template
from .models import *
from .forms import *
import numpy as np
import os
import io

def es_veterinarios(user):
    return user.groups.filter(name='Veterinarios').exists()

def es_galponero(user):
    return user.groups.filter(name='Galponero').exists()

def es_administradores(user):
    return user.groups.filter(name='Administradores').exists()

def home(request):
    return render(request, 'index.html')

@login_required
@user_passes_test(es_galponero)
def galponero(request):
    return render(request, 'Galponero/galponero.html')

@login_required
@user_passes_test(es_galponero)
def m_crecimiento_galponero(request):
    return render(request, 'Galponero/m_creacimiento.html')

@login_required
@user_passes_test(es_administradores)
def admin1(request):
    return render(request, 'admin/admin.html')
```

```
@login_required
@user_passes_test(es_administradores)
def lote_galpon(request):
    return render(request, 'admin/lote_galpon.html')

@login_required
@user_passes_test(es_veterinarios)
def veterinario(request):
    return render(request, 'veterinario/veterinario.html')

@login_required
@user_passes_test(es_galponero)
def m_crecimiento(request):
    return render(request, 'admin/m_crecimiento.html')

@login_required
@user_passes_test(es_administradores)
def m_reportes_generales(request):
    return render(request, 'admin/m_reportes_generales.html')

@login_required
@user_passes_test(es_veterinarios)
def m_reportes_generales_vete(request):
    return render(request, 'veterinario/m_reportes_generales.html')

@login_required
@user_passes_test(es_veterinarios)
def tratamiento(request):
    return render(request, 'reporte_veterinario/tratamiento.html')

def suma_valores(request,  galpon_id):
    galpon = get_object_or_404(Galpon, id=galpon_id)
    suma_total =
Mortalidad.objects.filter(galpon=galpon).aggregate(total=Sum('total_moratali
dad'))['total']
    return render(request, 'prueba/tu_template.html', {'suma_total':
suma_total,'galpon': galpon})

def imprimir_pdf(request):
    # Recupera los objetos de Muestra
    peso = Peso_promedio.objects.all()

    # 138ndice138 138ndice138 PDF
    response = HttpResponse(content_type='application/pdf')
    response['Content-Disposition'] = 'attachment; filename="muestras.pdf"'

    # Crea el objeto PDF usando ReportLab
    pdf = canvas.Canvas(response)
```

```
    pdf.drawString(100, 750, “Listado de muestras:”)

    y = 700
    for peso in peso:
        pdf.drawString(100, y, f”Semana: {peso.fecha_ingreso_peso}”)
        pdf.drawString(300, y, f”Cantidad de muestra:
{peso.cantidad_muestra}”)
        pdf.drawString(500, y, f”Peso de muestra: {peso.peso_muestra}
gramos”)
        y -= 20

    # Finaliza y guarda el PDF
    pdf.showPage()
    pdf.save()
    return response

@login_required
@user_passes_test(es_veterinarios)
def agReporte(request):
    data = {
        ‘form’: Reporte_generalForm()
    }
    if request.method == ‘POST’:
        formulario = Reporte_generalForm(data=request.POST)
        if formulario.is_valid():
            formulario.save()
            data[“mensaje”] = “guardado correctamente”
            return redirect(‘verTratamiento’)
        else:
            data[“form”] = formulario
    return render(request, ‘reporte_veterinario/agregar.html’, data)

@login_required
@user_passes_test(es_administradores)
def crear_lote(request):
    if request.method == ‘POST’:
        form = LoteForm(request.POST)
        if form.is_valid():
            form.save()
            messages.success(request, ‘El lote se ha guardado
exitosamente.’)
            return redirect(‘ver_Lote’)  # Redirige a la página de lista de
lotes o a donde desees redirigir después de guardar
    else:
        form = LoteForm()
    return render(request, ‘admin/agLote.html’, {‘form’: form})

@login_required
@user_passes_test(es_administradores)
```

```
def crear_galpon(request):
    if request.method == 'POST':
        form = GalponForm(request.POST)
        if form.is_valid():
            form.save()
            messages.success(request, 'El lote se ha guardado
exitosamente.')
            return redirect('ver_Galpon')  # Redirige a la página de lista
de lotes o a donde desees redirigir después de guardar
    else:
        form = GalponForm()
    return render(request, 'admin/agGalpon.html', {'form': form})

@login_required
def vista_reporte(request):
    vista = Reportes_generales.objects.all()
    print(vista)
    data = {
        'vista': vista,
    }
    return render(request, 'reporte_veterinario/Imprimir.html', data)

def edReporte(request, id):
    reporte = get_object_or_404(Reportes_generales, id=id)
    data = {
        'form': Reporte_generalForm(instance=reporte)
    }
    if request.method == 'POST':
        formulario = Reporte_generalForm(data=request.POST,
instance=reporte)
        if formulario.is_valid():
            formulario.save()
            messages.success(request, "El producto fue creado
correctamente")
            return redirect(to="verTratamiento")
        else:
            messages.warning(request, "El código del producto debe tener 5
caracteres")
        data["form"] = formulario
    return render(request, 'reporte_veterinario/edReporte.html', data)

#@login_required
class Reporte_pdf(View):

    def get(self, request, *args, **kwargs):
        vista = Reportes_generales.objects.all()
        data = {
    #         'count': vista.count(),
            'vista': vista
```

```
        }
        pdf = render_to_pdf('reporte_veterinario/Imprimir.html', data)
        return HttpResponse(pdf, content_type='application/pdf')

@login_required
@user_passes_test(es_galponero)
def agMortalidad(request):
    form = MortalidadForm()
    if request.method == 'POST':
        form = MortalidadForm(request.POST,request.FILES)
        if form.is_valid():
            form.save()
            return redirect('verMortalidad')
        else:
            form = MortalidadForm()
    return render(request, 'Galponero/agMortalidad.html', {'form': form})

@login_required
@user_passes_test(es_galponero)
def edMortalidad(request, id):
    mortalidad = get_object_or_404(Mortalidad, id=id)
    data = {
        'form': MortalidadForm(instance=mortalidad)
    }
    if request.method == 'POST':
        formulario = MortalidadForm(data=request.POST, instance=mortalidad)
        if formulario.is_valid():
            formulario.save()
            messages.success(request, "El 141ndice141 fue creado
correctamente")
            return redirect(to="verMortalidad")
        else:
            messages.warning(request, "El código del producto debe tener 5
caracteres")
        data["form"] = formulario
    return render(request, 'Galponero/edMortalidad.html', data)
#def login (request):

@login_required
@user_passes_test(es_galponero)
def edPredict2(request, id):

    predict2 = get_object_or_404(Predict2, id=id)
    data = {
        'form': Predict2Form(instance=predict2)
    }
    if request.method == 'POST':
        formulario = Predict2Form(data=request.POST, instance=predict2)
        if formulario.is_valid():
```

```
            formulario.save()
            messages.success(request, “El producto fue creado
correctamente”)
            return redirect(to=”verprediccion”)
        else:
            messages.warning(request, “El código del producto debe tener 5
caracteres”)
        data[“form”] = formulario
    return render(request, ‘Galponero/edIdentificacion.html’, data)


@login_required
@user_passes_test(es_galponero)
def edrecepcion(request, id):

    recepcion = get_object_or_404(Recepcion_Alimentos, id=id)
    data = {
        ‘form’: RecepcionAlimentosForm(instance=recepcion)
    }
    if request.method == ‘POST’:
        formulario = RecepcionAlimentosForm(data=request.POST,
instance=recepcion)
        if formulario.is_valid():
            formulario.save()
            messages.success(request, “El producto fue creado
correctamente”)
            return redirect(to=”detalle_recepcion”)
        else:
            messages.warning(request, “El código del producto debe tener 5
caracteres”)
        data[“form”] = formulario
    return render(request, ‘Galponero/edIdentificacion.html’, data)

@login_required
def verMortalidad(request):
    vista = Mortalidad.objects.all()
    saldo_total = sum(registro.total_moratalidad for registro in vista)

    data = {
        ‘vista’: vista,
        ‘saldo_total’: saldo_total,
    }
    return render(request, ‘Galponero/verMortalidad.html’, data)

def reporte_mortalidad_pdf(request):
    template_path = ‘reportes/verMortalidadpdf.html’
        # Obtén los datos de los modelos que deseas mostrar
    vista = Mortalidad.objects.all()
    saldo_total = sum(registro.total_moratalidad for registro in vista)
```

```
    imagen = vista[0].imagen.url  # Obtén la URL de la imagen de un registro
(asumiendo que solo hay uno)

    context = {
        'vista': vista,  # Agrega los datos al contexto de la plantilla
        'saldo_total': saldo_total,
        'imagen': imagen
    }
    template = get_template(template_path)
    html = template.render(context)
    # Renderiza la plantilla HTML
    # Crea un objeto StringIO para almacenar el PDF generado
    result = io.BytesIO()
    # Crea el documento PDF a partir de la plantilla HTML
    encoding = 'UTF-8'
    html = html.encode(encoding)
    #pdf = pisa.pisaDocument(io.BytesIO(html.encode("UTF-8")), result)
    pdf = pisa.pisaDocument(io.BytesIO(html), result, encoding=encoding)
    # Verifica si se generó correctamente el PDF
    if not pdf.err:
        # Establece la respuesta HTTP con el contenido del PDF
        response = HttpResponse(result.getvalue(),
content_type='application/pdf')
        response['Content-Disposition'] = 'attachment;
filename="143ndice143t.pdf"'
        return response
    # Si ocurrió un error, devuelve una respuesta vacía con un estado de
error
    return HttpResponse('Error al generar el PDF', status=500)

def predict_imagen2(request):
    if request.method == 'POST' and request.FILES['image']:
        # Obtener la imagen cargada por el usuario
        image_file = request.FILES['image']
        # Guardar la imagen en el directorio MEDIA_ROOT
        image_path = os.path.join(settings.MEDIA_ROOT, image_file.name)
        with open(image_path, 'wb') as f:
            f.write(image_file.read())
        # Especificar el dispositivo de E/S
        options =
tf.saved_model.LoadOptions(experimental_io_device='/job:localhost')
        # Cargar el modelo SavedModel
        model = tf.saved_model.load(os.path.join(settings.STATIC_ROOT,
'saved_model'), options=options)
        # Obtener la función predict del modelo
        predict_fn = model.signatures['serving_default']
        # Cargar y preprocesar la imagen de entrada
        image = tf.io.read_file(image_path)
        image = tf.image.decode_image(image, channels=3)
```

```
        image = tf.image.resize(image, (224, 224))
        image = image / 255.0  # Normalizar la imagen
        # Agregar una dimensión extra para que coincida con el tamaño de
lote esperado por el modelo
        image = tf.expand_dims(image, axis=0)
        # Realizar la predicción
        predictions = predict_fn(image)['dense']
        # Obtener la clase con mayor probabilidad
        predicted_class = tf.argmax(predictions, axis=1)[0].numpy()
        # Definir las etiquetas correspondientes a las clases
        labels = {
            3: 'Pre-inicio',
            2: 'Inicio',
            1: 'Engorde',
            0: 'Desarrollo'
        }
        # Obtener la etiqueta correspondiente a la clase predicha
        predicted_label = labels[predicted_class]
        # Eliminar el archivo de imagen después de la predicción
        os.remove(image_path)
        # Guardar la predicción y el enlace de la imagen en el modelo
Prediccion
        144ndice144t144144 = Prediccion()
        144ndice144t144144.imagen = image_file
        144ndice144t144144.etiqueta = predicted_label
        144ndice144t144144.save()
        # Renderizar el resultado en el template
        return render(request, 'prediccion/result.html', {'prediccion':
prediccion, 'predicted_label': predicted_label})
    return render(request, 'prediccion/upload.html')

@login_required
@user_passes_test(es_galponero)
def predic_a_index(request):
    return render(request, 'prediccion_edad/index.html')

@login_required
@user_passes_test(es_galponero)
def predict_a2(request):
    if request.method == 'POST':
        peso = int(request.POST['peso'])
        fecha_ingreso = (request.POST['fecha_ingreso'])
        edad_predicha = predecir_edad(peso)
        # Guardar el resultado de la predicción en el modelo
        resultado = AgePrediction(peso=peso, fecha_ingreso=fecha_ingreso,
edad_predicha=edad_predicha)
        resultado.save()
        return render(request, 'prediccion_edad/predict.html', {'peso':
peso, 'edad_predicha': edad_predicha})
```

```
    else:
        return render(request, 'prediccion_edad/predict.html')
#ejemplo de unido los dos modelos

@login_required
@user_passes_test(es_galponero)
def predic_a_index2(request):
    return render(request, 'prediccionjunto/upload.html')

@login_required
@user_passes_test(es_galponero)
def predict_imagen3(request):
    if request.method == 'POST' and request.FILES['image']:
        # Obtener la imagen cargada por el usuario
        image_file = request.FILES['image']
        # Guardar la imagen en el directorio MEDIA_ROOT
        image_path = os.path.join(settings.MEDIA_ROOT, image_file.name)
        with open(image_path, 'wb') as f:
            f.write(image_file.read())
        # Especificar el dispositivo de E/S
        options =
tf.saved_model.LoadOptions(experimental_io_device='/job:localhost')
        # Cargar el modelo SavedModel
        model = tf.saved_model.load(os.path.join(settings.STATIC_ROOT,
'saved_model'), options=options)
        # Obtener la función predict del modelo
        predict_fn = model.signatures['serving_default']
        # Cargar y preprocesar la imagen de entrada
        image = tf.io.read_file(image_path)
        image = tf.image.decode_image(image, channels=3)
        image = tf.image.resize(image, (224, 224))
        image = image / 255.0  # Normalizar la imagen
        # Agregar una dimensión extra para que coincida con el tamaño de
lote esperado por el modelo
        image = tf.expand_dims(image, axis=0)
        # Realizar la predicción
        predictions = predict_fn(image)['dense']
        # Obtener la clase con mayor probabilidad
        predicted_class = tf.argmax(predictions, axis=1)[0].numpy()
        # Definir las etiquetas correspondientes a las clases
        labels = {
            3: 'Pre-inicio',
            2: 'Inicio',
            1: 'Engorde',
            0: 'Desarrollo'
        }
        # Obtener la etiqueta correspondiente a la clase predicha
        predicted_label = labels[predicted_class]
        # Eliminar el archivo de imagen después de la predicción
```

```
        os.remove(image_path)
        prediccion = Predict2()
        146ndice146t146146.imagen = image_file
        146ndice146t146146.etiqueta = predicted_label
        146ndice146t146146.save()

        peso = int(request.POST['peso'])
        edad_predicha = predecir_edad(peso)
        # Guardar la predicción en el modelo AgePrediction y establecer la
relación uno a uno
        age_prediction = AgePrediction()
        age_prediction.peso = peso
        age_prediction.edad_predicha = edad_predicha
        age_prediction.save()

        prediccion.age_prediction = age_prediction
        prediccion.save()

        # Renderizar el resultado en el template
        return render(request, 'prediccionjunto/result.html', {'prediccion':
146ndice146t146146, 'predicted_label': predicted_label})
    return render(request, 'prediccionjunto/upload.html')

@login_required
@user_passes_test(es_galponero)
def crear_recepcion_alimentos(request):
    if request.method == 'POST':
        form = RecepcionAlimentosForm(request.POST)
        if form.is_valid():
            146ndice146t146_alimentos = form.save()
            return redirect('detalle_recepcion')
    else:
        form = RecepcionAlimentosForm()
    return render(request, 'Galponero/crear_recepcion_alimentos.html',
{'form': form})

def detalle_recepcion(request):
    146ndice146t146 = Recepcion_Alimentos.objects.all()
    saldo_total = sum(registro.saldo for registro in 146ndice146t146)
    data = {
        'recepcion': 146ndice146t146,
        'saldo_total': saldo_total
    }
    return render(request, 'Galponero/detalle_recepcion.html', data)

def reporte_alimentos_pdf(request):
    template_path = 'reportes/detalle_recepcionpdf.html'
        # Obtén los datos de los modelos que deseas mostrar
    146ndice146t146 = Recepcion_Alimentos.objects.all()
```

```
    saldo_total = sum(registro.saldo for registro in 147ndice147t147)

    context = {
        'recepcion': 147ndice147t147,  # Agrega los datos al contexto de la
plantilla
        'saldo_total': saldo_total
    }
    template = get_template(template_path)
    html = template.render(context)
    # Renderiza la plantilla HTML

    # Crea un objeto StringIO para almacenar el PDF generado
    result = io.BytesIO()

    # Crea el documento PDF a partir de la plantilla HTML
    encoding = 'UTF-8'
    html = html.encode(encoding)
    #pdf = pisa.pisaDocument(io.BytesIO(html.encode("UTF-8")), result)
    pdf = pisa.pisaDocument(io.BytesIO(html), result, encoding=encoding)

    # Verifica si se generó correctamente el PDF
    if not pdf.err:
        # Establece la respuesta HTTP con el contenido del PDF
        response = HttpResponse(result.getvalue(),
content_type='application/pdf')
        response['Content-Disposition'] = 'attachment;
filename="147ndice147t.pdf"'
        return response

    # Si ocurrió un error, devuelve una respuesta vacía con un estado de
error
    return HttpResponse('Error al generar el PDF', status=500)

def verprediccion(request):
    vista = Predict2.objects.all()
    data = {
        'vista': vista,
    }
    return render(request, 'prediccionjunto/verprediccion.html', data)

def verTratamiento(request):
    vista = Reportes_generales.objects.all()
    data = {
        'vista': vista
    }
    return render(request, 'reporte_veterinario/tratamiento.html', data)

def tratamiento_pdf(request):
    template_path = 'reporte_veterinario/tratamientopdf.html'
```

```
        # Obtén los datos de los modelos que deseas mostrar
    vista = Reportes_generales.objects.all()
    context = {
        'vista': vista,  # Agrega los datos al contexto de la plantilla
    }
    template = get_template(template_path)
    html = template.render(context)
    # Renderiza la plantilla HTML

    # Crea un objeto StringIO para almacenar el PDF generado
    result = io.BytesIO()

    # Crea el documento PDF a partir de la plantilla HTML
    encoding = 'UTF-8'
    html = html.encode(encoding)
    #pdf = pisa.pisaDocument(io.BytesIO(html.encode("UTF-8")), result)
    pdf = pisa.pisaDocument(io.BytesIO(html), result, encoding=encoding)

    # Verifica si se generó correctamente el PDF
    if not pdf.err:
        # Establece la respuesta HTTP con el contenido del PDF
        response = HttpResponse(result.getvalue(),
content_type='application/pdf')
        response['Content-Disposition'] = 'attachment;
filename="148ndice148t.pdf"'
        return response

    # Si ocurrió un error, devuelve una respuesta vacía con un estado de
error
    return HttpResponse('Error al generar el PDF', status=500)

def verprediccion_pdf(request):
    template_path = 'prediccionjunto/verprediccion_pdf.html'
        # Obtén los datos de los modelos que deseas mostrar
    vista = Predict2.objects.all()
    imagen = vista[0].imagen.url  # Obtén la URL de la imagen de un registro
(asumiendo que solo hay uno)

    context = {
        'vista': vista,  # Agrega los datos al contexto de la plantilla
        'imagen': imagen
    }
    template = get_template(template_path)
    html = template.render(context)
    # Renderiza la plantilla HTML
    # Crea un objeto StringIO para almacenar el PDF generado
    result = io.BytesIO()
    # Crea el documento PDF a partir de la plantilla HTML
    encoding = 'UTF-8'
```

```
    html = html.encode(encoding)
    #pdf = pisa.pisaDocument(io.BytesIO(html.encode("UTF-8")), result)
    pdf = pisa.pisaDocument(io.BytesIO(html), result, encoding=encoding)
    # Verifica si se generó correctamente el PDF
    if not pdf.err:
        # Establece la respuesta HTTP con el contenido del PDF
        response = HttpResponse(result.getvalue(),
content_type='application/pdf')
        response['Content-Disposition'] = 'attachment;
filename="149ndice149t.pdf"'
        return response
    # Si ocurrió un error, devuelve una respuesta vacía con un estado de
error
    return HttpResponse('Error al generar el PDF', status=500)

@login_required
@user_passes_test(es_administradores)
def ver_Lote(request):
    vista = Lote.objects.all()
    data = {
        'vista': vista,
    }
    return render(request, 'admin/ver_lote.html', data)

@login_required
@user_passes_test(es_administradores)
def ver_Galpon(request):
    vista = Galpon.objects.all()
    data = {
        'vista': vista,
    }
    return render(request, 'admin/ver_galpon.html', data)
```

6. Código fuente del Views del software de control de crecimiento

El código fuente de views se observa en la Tabla 5.

Tabla 5. Código fuente

```
#from typing_extensions import required
from 149ndice.contrib.auth.decorators import login_required,
permission_required, user_passes_test
from 149ndice.shortcuts import redirect, render, get_object_or_404
from 149ndice.contrib.auth.mixins import LoginRequiredMixin
from 149ndice.views.generic import CreateView, DetailView
from 149ndice.core.files.storage import FileSystemStorage
from 149ndice.contrib.auth import authenticate, login
from 149ndice.conf import UserSettingsHolder, settings
```

```
from 150ndice.http import JsonResponse, HttpResponse
from 150ndice.template.loader import render_to_string, get_template
from 150ndice.forms.models import construct_instance
from 150ndice.utils.encoding import smart_str
from app.pre.predecir_edad import predecir_edad
from 150ndice.contrib.messages.api import success
from 150ndice.http.response import HttpResponse
from 150ndice.forms.widgets import PasswordInput
from 150ndice.shortcuts import render, redirect
from 150ndice.contrib.auth.models import User
from 150ndice.contrib import admin, messages
from 150ndice.http.request import HttpRequest
from 150ndice.contrib.messages import success
from mimetypes import guess_all_extensions
from 150ndice.core.mail import send_mail
from reportlab.pdfgen import canvas
from 150ndice.db.models import Sum
from 150ndice.conf import settings
from .utils import render_to_pdf
from 150ndice.views import View
from 150ndice.db import models
from tensorflow import keras
from tempfile import tempdir
from urllib import request
from xhtml2pdf import pisa
from operator import ipow
from pyexpat import model
import tensorflow as tf
from http import server
from re import template
from .models import *
from .forms import *
import numpy as np
import os
import io

def es_veterinarios(user):
    return user.groups.filter(name='Veterinarios').exists()

def es_galponero(user):
    return user.groups.filter(name='Galponero').exists()

def es_administradores(user):
    return user.groups.filter(name='Administradores').exists()

def home(request):
    return render(request, 'index.html')

@login_required
```

```
@user_passes_test(es_galponero)
def galponero(request):
    return render(request, 'Galponero/galponero.html')

@login_required
@user_passes_test(es_galponero)
def m_crecimiento_galponero(request):
    return render(request, 'Galponero/m_creacimiento.html')

@login_required
@user_passes_test(es_administradores)
def admin1(request):
    return render(request, 'admin/admin.html')

@login_required
@user_passes_test(es_administradores)
def lote_galpon(request):
    return render(request, 'admin/lote_galpon.html')

@login_required
@user_passes_test(es_veterinarios)
def veterinario(request):
    return render(request, 'veterinario/veterinario.html')

@login_required
@user_passes_test(es_galponero)
def m_crecimiento(request):
    return render(request, 'admin/m_crecimiento.html')

@login_required
@user_passes_test(es_administradores)
def m_reportes_generales(request):
    return render(request, 'admin/m_reportes_generales.html')

@login_required
@user_passes_test(es_veterinarios)
def m_reportes_generales_vete(request):
    return render(request, 'veterinario/m_reportes_generales.html')

@login_required
@user_passes_test(es_veterinarios)
def tratamiento(request):
    return render(request, 'reporte_veterinario/tratamiento.html')

def suma_valores(request,  galpon_id):
    galpon = get_object_or_404(Galpon, id=galpon_id)
    suma_total =
Mortalidad.objects.filter(galpon=galpon).aggregate(total=Sum('total_moratali
dad'))['total']
```

```
    return render(request, 'prueba/tu_template.html', {'suma_total':
suma_total,'galpon': galpon})

def imprimir_pdf(request):
    # Recupera los objetos de Muestra
    peso = Peso_promedio.objects.all()

    # 152ndice152 152ndice152 PDF
    response = HttpResponse(content_type='application/pdf')
    response['Content-Disposition'] = 'attachment; filename="muestras.pdf"'

    # Crea el objeto PDF usando ReportLab
    pdf = canvas.Canvas(response)
    pdf.drawString(100, 750, "Listado de muestras:")

    y = 700
    for peso in peso:
        pdf.drawString(100, y, f"Semana: {peso.fecha_ingreso_peso}")
        pdf.drawString(300, y, f"Cantidad de muestra:
{peso.cantidad_muestra}")
        pdf.drawString(500, y, f"Peso de muestra: {peso.peso_muestra}
gramos")
        y -= 20

    # Finaliza y guarda el PDF
    pdf.showPage()
    pdf.save()
    return response

@login_required
@user_passes_test(es_veterinarios)
def agReporte(request):
    data = {
        'form': Reporte_generalForm()
    }
    if request.method == 'POST':
        formulario = Reporte_generalForm(data=request.POST)
        if formulario.is_valid():
            formulario.save()
            data["mensaje"] = "guardado correctamente"
            return redirect('verTratamiento')
        else:
            data["form"] = formulario
    return render(request, 'reporte_veterinario/agregar.html', data)

@login_required
@user_passes_test(es_administradores)
def crear_lote(request):
    if request.method == 'POST':
```

```
        form = LoteForm(request.POST)
        if form.is_valid():
            form.save()
            messages.success(request, 'El lote se ha guardado
exitosamente.')
            return redirect('ver_Lote')  # Redirige a la página de lista de
lotes o a donde desees redirigir después de guardar
    else:
        form = LoteForm()
    return render(request, 'admin/agLote.html', {'form': form})

@login_required
@user_passes_test(es_administradores)
def crear_galpon(request):
    if request.method == 'POST':
        form = GalponForm(request.POST)
        if form.is_valid():
            form.save()
            messages.success(request, 'El lote se ha guardado
exitosamente.')
            return redirect('ver_Galpon')  # Redirige a la página de lista
de lotes o a donde desees redirigir después de guardar
    else:
        form = GalponForm()
    return render(request, 'admin/agGalpon.html', {'form': form})

@login_required
def vista_reporte(request):
    vista = Reportes_generales.objects.all()
    print(vista)
    data = {
        'vista': vista,
    }
    return render(request, 'reporte_veterinario/Imprimir.html', data)

def edReporte(request, id):
    reporte = get_object_or_404(Reportes_generales, id=id)
    data = {
        'form': Reporte_generalForm(instance=reporte)
    }
    if request.method == 'POST':
        formulario = Reporte_generalForm(data=request.POST,
instance=reporte)
        if formulario.is_valid():
            formulario.save()
            messages.success(request, "El producto fue creado
correctamente")
            return redirect(to="verTratamiento")
        else:
```

```
            messages.warning(request, "El código del producto debe tener 5
caracteres")
        data["form"] = formulario
    return render(request, 'reporte_veterinario/edReporte.html', data)

#@login_required
class Reporte_pdf(View):

    def get(self, request, *args, **kwargs):
        vista = Reportes_generales.objects.all()
        data = {
    #         'count': vista.count(),
            'vista': vista
        }
        pdf = render_to_pdf('reporte_veterinario/Imprimir.html', data)
        return HttpResponse(pdf, content_type='application/pdf')

@login_required
@user_passes_test(es_galponero)
def agMortalidad(request):
    form = MortalidadForm()
    if request.method == 'POST':
        form = MortalidadForm(request.POST,request.FILES)
        if form.is_valid():
            form.save()
            return redirect('verMortalidad')
        else:
            form = MortalidadForm()
    return render(request, 'Galponero/agMortalidad.html', {'form': form})

@login_required
@user_passes_test(es_galponero)
def edMortalidad(request, id):
    mortalidad = get_object_or_404(Mortalidad, id=id)
    data = {
        'form': MortalidadForm(instance=mortalidad)
    }
    if request.method == 'POST':
        formulario = MortalidadForm(data=request.POST, instance=mortalidad)
        if formulario.is_valid():
            formulario.save()
            messages.success(request, "El 154ndice154 fue creado
correctamente")
            return redirect(to="verMortalidad")
        else:
            messages.warning(request, "El código del producto debe tener 5
caracteres")
        data["form"] = formulario
    return render(request, 'Galponero/edMortalidad.html', data)
```

```
#def login (request):

@login_required
@user_passes_test(es_galponero)
def edPredict2(request, id):

    predict2 = get_object_or_404(Predict2, id=id)
    data = {
        'form': Predict2Form(instance=predict2)
    }
    if request.method == 'POST':
        formulario = Predict2Form(data=request.POST, instance=predict2)
        if formulario.is_valid():
            formulario.save()
            messages.success(request, "El producto fue creado
correctamente")
            return redirect(to="verprediccion")
        else:
            messages.warning(request, "El código del producto debe tener 5
caracteres")
        data["form"] = formulario
    return render(request, 'Galponero/edIdentificacion.html', data)


@login_required
@user_passes_test(es_galponero)
def edrecepcion(request, id):

    recepcion = get_object_or_404(Recepcion_Alimentos, id=id)
    data = {
        'form': RecepcionAlimentosForm(instance=recepcion)
    }
    if request.method == 'POST':
        formulario = RecepcionAlimentosForm(data=request.POST,
instance=recepcion)
        if formulario.is_valid():
            formulario.save()
            messages.success(request, "El producto fue creado
correctamente")
            return redirect(to="detalle_recepcion")
        else:
            messages.warning(request, "El código del producto debe tener 5
caracteres")
        data["form"] = formulario
    return render(request, 'Galponero/edIdentificacion.html', data)

@login_required
def verMortalidad(request):
    vista = Mortalidad.objects.all()
```

```
    saldo_total = sum(registro.total_moratalidad for registro in vista)

    data = {
        'vista': vista,
        'saldo_total': saldo_total,
    }
    return render(request, 'Galponero/verMortalidad.html', data)

def reporte_mortalidad_pdf(request):
    template_path = 'reportes/verMortalidadpdf.html'
        # Obtén los datos de los modelos que deseas mostrar
    vista = Mortalidad.objects.all()
    saldo_total = sum(registro.total_moratalidad for registro in vista)
    imagen = vista[0].imagen.url  # Obtén la URL de la imagen de un registro
(asumiendo que solo hay uno)

    context = {
        'vista': vista,  # Agrega los datos al contexto de la plantilla
        'saldo_total': saldo_total,
        'imagen': imagen
    }
    template = get_template(template_path)
    html = template.render(context)
    # Renderiza la plantilla HTML
    # Crea un objeto StringIO para almacenar el PDF generado
    result = io.BytesIO()
    # Crea el documento PDF a partir de la plantilla HTML
    encoding = 'UTF-8'
    html = html.encode(encoding)
    #pdf = pisa.pisaDocument(io.BytesIO(html.encode("UTF-8")), result)
    pdf = pisa.pisaDocument(io.BytesIO(html), result, encoding=encoding)
    # Verifica si se generó correctamente el PDF
    if not pdf.err:
        # Establece la respuesta HTTP con el contenido del PDF
        response = HttpResponse(result.getvalue(),
content_type='application/pdf')
        response['Content-Disposition'] = 'attachment;
filename="156ndice156t.pdf"'
        return response
    # Si ocurrió un error, devuelve una respuesta vacía con un estado de
error
    return HttpResponse('Error al generar el PDF', status=500)

def predict_imagen2(request):
    if request.method == 'POST' and request.FILES['image']:
        # Obtener la imagen cargada por el usuario
        image_file = request.FILES['image']
        # Guardar la imagen en el directorio MEDIA_ROOT
        image_path = os.path.join(settings.MEDIA_ROOT, image_file.name)
```

```
        with open(image_path, 'wb') as f:
            f.write(image_file.read())
        # Especificar el dispositivo de E/S
        options =
tf.saved_model.LoadOptions(experimental_io_device='/job:localhost')
        # Cargar el modelo SavedModel
        model = tf.saved_model.load(os.path.join(settings.STATIC_ROOT,
'saved_model'), options=options)
        # Obtener la función predict del modelo
        predict_fn = model.signatures['serving_default']
        # Cargar y preprocesar la imagen de entrada
        image = tf.io.read_file(image_path)
        image = tf.image.decode_image(image, channels=3)
        image = tf.image.resize(image, (224, 224))
        image = image / 255.0  # Normalizar la imagen
        # Agregar una dimensión extra para que coincida con el tamaño de
lote esperado por el modelo
        image = tf.expand_dims(image, axis=0)
        # Realizar la predicción
        predictions = predict_fn(image)['dense']
        # Obtener la clase con mayor probabilidad
        predicted_class = tf.argmax(predictions, axis=1)[0].numpy()
        # Definir las etiquetas correspondientes a las clases
        labels = {
            3: 'Pre-inicio',
            2: 'Inicio',
            1: 'Engorde',
            0: 'Desarrollo'
        }
        # Obtener la etiqueta correspondiente a la clase predicha
        predicted_label = labels[predicted_class]
        # Eliminar el archivo de imagen después de la predicción
        os.remove(image_path)
        # Guardar la predicción y el enlace de la imagen en el modelo
Prediccion
        157ndice157t157157 = Prediccion()
        157ndice157t157157.imagen = image_file
        157ndice157t157157.etiqueta = predicted_label
        157ndice157t157157.save()
        # Renderizar el resultado en el template
        return render(request, 'prediccion/result.html', {'prediccion':
prediccion, 'predicted_label': predicted_label})
    return render(request, 'prediccion/upload.html')

@login_required
@user_passes_test(es_galponero)
def predic_a_index(request):
    return render(request, 'prediccion_edad/index.html')
```

```
@login_required
@user_passes_test(es_galponero)
def predict_a2(request):
    if request.method == 'POST':
        peso = int(request.POST['peso'])
        fecha_ingreso = (request.POST['fecha_ingreso'])
        edad_predicha = predecir_edad(peso)
        # Guardar el resultado de la predicción en el modelo
        resultado = AgePrediction(peso=peso, fecha_ingreso=fecha_ingreso,
edad_predicha=edad_predicha)
        resultado.save()
        return render(request, 'prediccion_edad/predict.html', {'peso':
peso, 'edad_predicha': edad_predicha})
    else:
        return render(request, 'prediccion_edad/predict.html')
#ejemplo de unido los dos modelos

@login_required
@user_passes_test(es_galponero)
def predic_a_index2(request):
    return render(request, 'prediccionjunto/upload.html')

@login_required
@user_passes_test(es_galponero)
def predict_imagen3(request):
    if request.method == 'POST' and request.FILES['image']:
        # Obtener la imagen cargada por el usuario
        image_file = request.FILES['image']
        # Guardar la imagen en el directorio MEDIA_ROOT
        image_path = os.path.join(settings.MEDIA_ROOT, image_file.name)
        with open(image_path, 'wb') as f:
            f.write(image_file.read())
        # Especificar el dispositivo de E/S
        options =
tf.saved_model.LoadOptions(experimental_io_device='/job:localhost')
        # Cargar el modelo SavedModel
        model = tf.saved_model.load(os.path.join(settings.STATIC_ROOT,
'saved_model'), options=options)
        # Obtener la función predict del modelo
        predict_fn = model.signatures['serving_default']
        # Cargar y preprocesar la imagen de entrada
        image = tf.io.read_file(image_path)
        image = tf.image.decode_image(image, channels=3)
        image = tf.image.resize(image, (224, 224))
        image = image / 255.0  # Normalizar la imagen
        # Agregar una dimensión extra para que coincida con el tamaño de
lote esperado por el modelo
        image = tf.expand_dims(image, axis=0)
        # Realizar la predicción
```

```
        predictions = predict_fn(image)['dense']
        # Obtener la clase con mayor probabilidad
        predicted_class = tf.argmax(predictions, axis=1)[0].numpy()
        # Definir las etiquetas correspondientes a las clases
        labels = {
            3: 'Pre-inicio',
            2: 'Inicio',
            1: 'Engorde',
            0: 'Desarrollo'
        }
        # Obtener la etiqueta correspondiente a la clase predicha
        predicted_label = labels[predicted_class]
        # Eliminar el archivo de imagen después de la predicción
        os.remove(image_path)
        prediccion = Predict2()
        159ndice159t159159.imagen = image_file
        159ndice159t159159.etiqueta = predicted_label
        159ndice159t159159.save()

        peso = int(request.POST['peso'])
        edad_predicha = predecir_edad(peso)
        # Guardar la predicción en el modelo AgePrediction y establecer la
relación uno a uno
        age_prediction = AgePrediction()
        age_prediction.peso = peso
        age_prediction.edad_predicha = edad_predicha
        age_prediction.save()

        prediccion.age_prediction = age_prediction
        prediccion.save()

        # Renderizar el resultado en el template
        return render(request, 'prediccionjunto/result.html', {'prediccion':
159ndice159t159159, 'predicted_label': predicted_label})
    return render(request, 'prediccionjunto/upload.html')

@login_required
@user_passes_test(es_galponero)
def crear_recepcion_alimentos(request):
    if request.method == 'POST':
        form = RecepcionAlimentosForm(request.POST)
        if form.is_valid():
            159ndice159t159_alimentos = form.save()
            return redirect('detalle_recepcion')
    else:
        form = RecepcionAlimentosForm()
    return render(request, 'Galponero/crear_recepcion_alimentos.html',
{'form': form})
```

```
def detalle_recepcion(request):
    160ndice160t160 = Recepcion_Alimentos.objects.all()
    saldo_total = sum(registro.saldo for registro in 160ndice160t160)
    data = {
        'recepcion': 160ndice160t160,
        'saldo_total': saldo_total
    }
    return render(request, 'Galponero/detalle_recepcion.html', data)

def reporte_alimentos_pdf(request):
    template_path = 'reportes/detalle_recepcionpdf.html'
        # Obtén los datos de los modelos que deseas mostrar
    160ndice160t160 = Recepcion_Alimentos.objects.all()
    saldo_total = sum(registro.saldo for registro in 160ndice160t160)

    context = {
        'recepcion': 160ndice160t160,  # Agrega los datos al contexto de la
plantilla
        'saldo_total': saldo_total
    }
    template = get_template(template_path)
    html = template.render(context)
    # Renderiza la plantilla HTML

    # Crea un objeto StringIO para almacenar el PDF generado
    result = io.BytesIO()

    # Crea el documento PDF a partir de la plantilla HTML
    encoding = 'UTF-8'
    html = html.encode(encoding)
    #pdf = pisa.pisaDocument(io.BytesIO(html.encode("UTF-8")), result)
    pdf = pisa.pisaDocument(io.BytesIO(html), result, encoding=encoding)

    # Verifica si se generó correctamente el PDF
    if not pdf.err:
        # Establece la respuesta HTTP con el contenido del PDF
        response = HttpResponse(result.getvalue(),
content_type='application/pdf')
        response['Content-Disposition'] = 'attachment;
filename="160ndice160t.pdf"'
        return response

    # Si ocurrió un error, devuelve una respuesta vacía con un estado de
error
    return HttpResponse('Error al generar el PDF', status=500)

def verprediccion(request):
    vista = Predict2.objects.all()
    data = {
```

```
        'vista': vista,
    }
    return render(request, 'prediccionjunto/verprediccion.html', data)

def verTratamiento(request):
    vista = Reportes_generales.objects.all()
    data = {
        'vista': vista
    }
    return render(request, 'reporte_veterinario/tratamiento.html', data)

def tratamiento_pdf(request):
    template_path = 'reporte_veterinario/tratamientopdf.html'
        # Obtén los datos de los modelos que deseas mostrar
    vista = Reportes_generales.objects.all()
    context = {
        'vista': vista,  # Agrega los datos al contexto de la plantilla
    }
    template = get_template(template_path)
    html = template.render(context)
    # Renderiza la plantilla HTML

    # Crea un objeto StringIO para almacenar el PDF generado
    result = io.BytesIO()

    # Crea el documento PDF a partir de la plantilla HTML
    encoding = 'UTF-8'
    html = html.encode(encoding)
    #pdf = pisa.pisaDocument(io.BytesIO(html.encode("UTF-8")), result)
    pdf = pisa.pisaDocument(io.BytesIO(html), result, encoding=encoding)

    # Verifica si se generó correctamente el PDF
    if not pdf.err:
        # Establece la respuesta HTTP con el contenido del PDF
        response = HttpResponse(result.getvalue(),
content_type='application/pdf')
        response['Content-Disposition'] = 'attachment;
filename="161ndice161t.pdf"'
        return response

    # Si ocurrió un error, devuelve una respuesta vacía con un estado de
error
    return HttpResponse('Error al generar el PDF', status=500)

def verprediccion_pdf(request):
    template_path = 'prediccionjunto/verprediccion_pdf.html'
        # Obtén los datos de los modelos que deseas mostrar
    vista = Predict2.objects.all()
```

```
    imagen = vista[0].imagen.url  # Obtén la URL de la imagen de un registro
(asumiendo que solo hay uno)

    context = {
        'vista': vista,  # Agrega los datos al contexto de la plantilla
        'imagen': imagen
    }
    template = get_template(template_path)
    html = template.render(context)
    # Renderiza la plantilla HTML
    # Crea un objeto StringIO para almacenar el PDF generado
    result = io.BytesIO()
    # Crea el documento PDF a partir de la plantilla HTML
    encoding = 'UTF-8'
    html = html.encode(encoding)
    #pdf = pisa.pisaDocument(io.BytesIO(html.encode("UTF-8")), result)
    pdf = pisa.pisaDocument(io.BytesIO(html), result, encoding=encoding)
    # Verifica si se generó correctamente el PDF
    if not pdf.err:
        # Establece la respuesta HTTP con el contenido del PDF
        response = HttpResponse(result.getvalue(),
content_type='application/pdf')
        response['Content-Disposition'] = 'attachment;
filename="162ndice162t.pdf"'
        return response
    # Si ocurrió un error, devuelve una respuesta vacía con un estado de
error
    return HttpResponse('Error al generar el PDF', status=500)

@login_required
@user_passes_test(es_administradores)
def ver_Lote(request):
    vista = Lote.objects.all()
    data = {
        'vista': vista,
    }
    return render(request, 'admin/ver_lote.html', data)

@login_required
@user_passes_test(es_administradores)
def ver_Galpon(request):
    vista = Galpon.objects.all()
    data = {
        'vista': vista,
    }
    return render(request, 'admin/ver_galpon.html', data)
```

7. Código fuente del Predecir Peso – Edad del software de control de crecimiento

El código fuente de Predecir Peso – Edad se observa en la Tabla 6.

Tabla 6. Código fuente

```
import numpy as np
pesos = np.array([38, 48, 51, 61, 63, 73, 79, 89, 97, 107, 117, 127, 147,
157, 177, 187, 197, 207, 217, 227,
                  247, 257, 300, 310, 340, 350, 385, 395, 415, 425, 475,
485, 545, 555, 615, 625, 685, 695,
                  765, 775, 835, 845, 910, 920, 1002, 1012, 1088, 1098,
1162, 1172, 1236, 1246, 1305, 1315,
                  1407, 1417, 1490, 1500, 1519, 1529, 1593, 1603, 1677,
1687, 1776, 1786, 1851, 1861, 1947,
                  1957, 2093, 2103, 2149, 2159, 2225, 2235, 2293, 2303,
2376, 2386, 2492, 2502, 2572, 2582,
                  2649, 2659, 2743, 2753, 2852, 2862, 2938, 2948, 3070,
3080, 3179, 3189, 3241, 3251, 3392, 3402])
edades = np.array([0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8, 9,
9,
                   10, 10, 11, 11, 12, 12, 13, 13, 14, 14, 15, 15, 16, 16,
17, 17,
                   18, 18, 19, 19, 20, 20, 21, 21, 22, 22, 23, 23, 24, 24,
25, 25,
                   26, 26, 27, 27, 28, 28, 29, 29, 30, 30, 31, 31, 32, 32,
33, 33,
                   34, 34, 35, 35, 36, 36, 37, 37, 38, 38, 39, 39, 40, 40,
41, 41,
                   42, 42, 43, 43, 44, 44, 45, 45, 46, 46, 47, 47, 48, 48,
49, 49])
def predecir_edad(peso):
    163ndice = np.searchsorted(pesos, peso)
    if 163ndice == 0:
        return edades[0]
    elif 163ndice == len(pesos):
        return edades[-1]
    peso_inferior = pesos[163ndice - 1]
    peso_superior = pesos[163ndice]
    edad_inferior = edades[163ndice - 1]
    edad_superior = edades[163ndice]
    pendiente = (edad_superior - edad_inferior) / (peso_superior -
peso_inferior)
    edad_predicha = edad_inferior + pendiente * (peso - peso_inferior)
    return int(edad_predicha)
```

Anexo 7. Plan de Pruebas Unitarias

Plan de Pruebas Unitarias

Proyecto: Software para el Control de Crecimiento en pollos de engorde en la granja Avícola "Las Acacias" en el Cantón Balsas

Versión: 1.0

Fecha: 25/07/2023

Hoja de control

Organismo	Universidad Nacional de Loja		
Proyecto	Software para el Control de Crecimiento en pollos de engorde en la granja Avícola “Las Acacias” en el Cantón Balsas		
Entregable	Planes de Pruebas Unitarias		
Autor	Viviana Maricela Zambrano Romero		
Versión/Edición	1.0	**Fecha Versión**	25/07/2023
Aprobado por	Edwin René Guamán Quinche	**Fecha Aprobación**	28/08/2021
		Nº Total de Páginas	8

Ficha del documento

Versión	Fecha de revisión	Responsables	Descripción de modificación
1.0	28/04/2023	Viviana Maricela Zambrano Romero	N/A

Control de distribución

Nombre y Apellidos
Viviana Maricela Zambrano Romero
Ing. Edwin René Guamán Quinche, Mg.Sc

Índice

1. Introducción

Objetivo

El objetivo de este documento es poder tener conocimiento sobre el código que se ejecuta del que se conoce como modulo principal, además de poder verificar si el sistema esta funcionando de manera correcta.

Propósito

Comprobar el correcto funcionamiento Software para el Control de Crecimiento en pollos de engorde en la granja Avícola “Las Acacias” en el Cantón Balsas, referente haber si es de manera eficiente el correcto funcionaiento del código de la parte principal.

2. Definición de los casos de pruebas

Número del Caso de Prueba	Requisito / Historia de Usuario	Componente	Descripción de lo que se probará	Prerrequisitos
CP01	RF01: Inicio de sesión	Inicio de sesión	Ingresar credenciales válidas	Usuario administrador registrado en el sistema
CP02	RF01: Inicio de sesión	Inicio de sesión	Ingresar credenciales inválidas	-
CP03	RF02: Registrar datos de la avícola	Registrar datos de la avícola	Registrar datos de la avícola correctamente	Usuario administrador registrado en el sistema
CP04	RF03: Generar reportes generales	Generar reportes generales	Generar reportes de mortalidad	Usuario administrador o veterinario registrado en el sistema
CP05	RF03: Generar reportes generales	Generar reportes generales	Generar reportes de alimentación	Usuario administrador o veterinario registrado en el sistema
CP06	RF03: Generar reportes generales	Generar reportes generales	Generar reportes de peso	Usuario administrador o veterinario registrado en el sistema
CP07	RF03: Generar reportes generales	Generar reportes generales	Generar reportes de enfermedades	Usuario administrador o veterinario registrado en el sistema
CP08	RF03: Generar reportes generales	Generar reportes generales	Generar reportes de tratamientos	Usuario administrador o veterinario registrado en el sistema
CP09	RF04: Registrar control de crecimiento	Registrar control de crecimiento	Registrar datos de control de crecimiento correctamente	Usuario galponero registrado en el sistema
CP10	RF05: Registrar mortalidad	Registrar mortalidad	Registrar datos de mortalidad correctamente	Usuario galponero registrado en el sistema
CP11	RF06: Registrar alimentación	Registrar alimentación	Registrar datos de alimentación correctamente	Usuario galponero registrado en el sistema
CP12	RF07: Registrar peso	Registrar peso	Registrar datos de peso	Usuario galponero registrado en el

			correctamente	sistema
CP13	RF08: Calcular crecimiento de pollos	Calcular crecimiento de pollos	Calcular crecimiento de pollos correctamente	Usuario galponero registrado en el sistema
CP14	RF09: Evaluar imagen	Evaluar imagen	Evaluar imágenes de pollos correctamente	Usuario galponero registrado en el sistema
CP15	RF10: Buscar anomalías	Buscar anomalías	Buscar anomalías correctamente	Usuario galponero registrado en el sistema
CP16	RF11: Ingresar información del diagnóstico de anomalías	Ingresar información del diagnóstico de anomalías	Ingresar información del diagnóstico correctamente	Usuario veterinario registrado en el sistema

CP01					
Nº	**Descripción**	**Método**	**Datos Entrada**	**¿OK?**	**Observaciones**
1	Ingresar credenciales válidas	Inicio de sesión	Credenciales válidas de un administrador	✓	N/A

CP02					
Nº	**Descripción**	**Método**	**Datos Entrada**	**¿OK?**	**Observaciones**
1	Ingresar credenciales inválidas	Inicio de sesión	Credenciales inválidas (usuario y/o contraseña incorrectos)	✓	N/A

CP03					
Nº	**Descripción**	**Método**	**Datos Entrada**	**¿OK?**	**Observaciones**
1	Registrar datos de la avícola correctamente	Registrar datos de la avícola	Datos válidos de la avícola (lote y galpón)	✓	N/A

CP04					
Nº	**Descripción**	**Método**	**Datos Entrada**	**¿OK?**	**Observaciones**
1	Generar reportes de mortalidad	Generar reportes generales	Opción seleccionada: "Mortalidad"	✓	N/A

CP05					
Nº	**Descripción**	**Método**	**Datos Entrada**	**¿OK?**	**Observaciones**

1	Generar reportes de alimentación	Generar reportes generales	Opción seleccionada: "Alimentación"	✓	N/A

CP06					
Nº	**Descripción**	**Método**	**Datos Entrada**	**¿OK?**	**Observaciones**
1	Generar reportes de peso	Generar reportes generales	Opción seleccionada: "Peso"	✓	N/A

CP07					
Nº	**Descripción**	**Método**	**Datos Entrada**	**¿OK?**	**Observaciones**
1	Generar reportes de enfermedades	Generar reportes generales	Opción seleccionada: "Enfermedades"	✓	N/A

CP08					
Nº	**Descripción**	**Método**	**Datos Entrada**	**¿OK?**	**Observaciones**
1	Generar reportes de tratamientos	Generar reportes generales	Opción seleccionada: "Tratamientos"	✓	N/A

CP09					
Nº	**Descripción**	**Método**	**Datos Entrada**	**¿OK?**	**Observaciones**
1	Registrar control de crecimiento correctamente	Registrar control de crecimiento	Datos válidos de control de crecimiento (mortalidad diaria, alimentación diaria, peso promedio)	✓	N/A

CP10					
Nº	**Descripción**	**Método**	**Datos Entrada**	**¿OK?**	**Observaciones**
1	Registrar mortalidad correctamente	Registrar mortalidad	Datos válidos de mortalidad (cantidad de pollos muertos, descartes	✓	N/A

CP11					
Nº	**Descripción**	**Método**	**Datos Entrada**	**¿OK?**	**Observaciones**
1	Registrar alimentación correctamente	Registrar alimentación	Datos válidos de alimentación (conductor,	✓	N/A

			cantidad ingreso, cantidad consumido, fecha ingreso, hora ingreso, hora fin, saldo balanceado)		

CP12					
Nº	**Descripción**	**Método**	**Datos Entrada**	**¿OK?**	**Observaciones**
1	Registrar peso correctamente	Registrar peso	Datos válidos de peso de los pollos y subir imagen	✓	N/A

CP13					
Nº	**Descripción**	**Método**	**Datos Entrada**	**¿OK?**	**Observaciones**
1	Calcular crecimiento de pollos correctamente	Calcular crecimiento de pollos	Datos de peso y control de crecimiento válidos previamente registrados	✓	N/A

CP14					
Nº	**Descripción**	**Método**	**Datos Entrada**	**¿OK?**	**Observaciones**
1	Evaluar imagen de pollos correctamente	Evaluar imagen	Imágenes válidas de pollos en diferentes fases	✓	N/A

CP15					
Nº	**Descripción**	**Método**	**Datos Entrada**	**¿OK?**	**Observaciones**
1	Buscar anomalías correctamente	Buscar anomalías	Criterios de búsqueda válidos (enfermedades o tratamientos)	✓	N/A

CP16					
Nº	**Descripción**	**Método**	**Datos Entrada**	**¿OK?**	**Observaciones**
1	Ingresar información del diagnóstico de anomalías correctamente	Ingresar información del diagnóstico de anomalías	Información de diagnóstico válida (descripción de la anomalía, síntomas, tratamientos)	✓	N/A

Anexo 8. Plan de Pruebas de Integración

Plan de Pruebas de Integración

Proyecto: Software para el Control de Crecimiento en pollos de engorde en la granja Avícola "Las Acacias" en el Cantón Balsas

Versión: 1.0

Fecha: 25/07/2023

Hoja de control

Organismo	Universidad Nacional de Loja		
Proyecto	Software para el Control de Crecimiento en pollos de engorde en la granja Avícola “Las Acacias” en el Cantón Balsas		
Entregable	Planes de Pruebas Unitarias		
Autor	Viviana Maricela Zambrano Romero		
Versión/Edición	1.0	**Fecha Versión**	25/07/2023
Aprobado por	Edwin René Guamán Quinche	**Fecha Aprobación**	28/08/2021
		Nº Total de Páginas	10

Ficha del documento

Versión	Fecha de revisión	Responsables	Descripción de modificación
1.0	28/04/2023	Viviana Maricela Zambrano Romero	N/A

Control de distribución

Nombre y Apellidos
Viviana Maricela Zambrano Romero
Ing. Edwin René Guamán Quinche, Mg.Sc

Índice

1. Introducción

Objetivo

El objetivo de este documento es poder verificar los requerimientos funcionales apartir del modulo en lo que es en distintos componentes, estas se elaboran después de la ejecución este realizada de manera correcta, se reakiza de manera unitaria.

Alcance

Los casos de pruebas están dirigidos para el docente Ing. Edwin René Guamán Quinche, Mg.Sc (director del TIC), docente de la Carrera de Ingeniería en Sistemas / Computación de la Universidad Nacional de Loja, lo cual se realiza la validación en los diferentes casos de pruebas. Y el estudiante Viviana Maricela Zambrano Romero que es el que genera y registra los diferentes casos de pruebas.

2. Definición de los casos de pruebas

Número del Caso de Prueba	Componentes	Descripción de lo que se probará	Prerrequisitos
CP01	Inicio de sesión	Verificar inicio de sesión con credenciales válidas e invalidad	• El sistema debe estar en funcionamiento. • Debe existir un usuario administrador registrado en el sistema con credenciales válidas e invalidas.
CP02	Registrar datos de la avícola	Verificar registro correcto de datos esenciales de la avícola	• El sistema debe estar en funcionamiento. • El usuario Administrador debe haber iniciado sesión en el sistema.
CP03	Generar reportes generales	Verificar generación de reportes de mortalidad, alimentación, peso, enfermedades y tratamientos	• El sistema debe estar en funcionamiento. • El usuario Administrador o Veterinario debe haber iniciado sesión en el sistema.
CP04	Registrar control de crecimiento	Verificar registro correcto de control de crecimiento de pollos	• El sistema debe estar en funcionamiento. • El usuario Galponero debe haber iniciado sesión en el sistema.
CP05	Registrar mortalidad	Verificar registro correcto de datos de mortalidad	• El sistema debe estar en funcionamiento. • El usuario Galponero debe haber iniciado sesión en el sistema
CP06	Registrar alimentación	Verificar registro correcto de datos de alimentación	• El sistema debe estar en funcionamiento. • El usuario Galponero debe haber iniciado sesión en el sistema.
CP07	Registrar peso	Verificar registro correcto de datos de peso	• El sistema debe estar en funcionamiento. • El usuario Galponero debe haber iniciado sesión en el sistema
CP08	Calcular crecimiento de pollos	Verificar cálculo correcto de crecimiento de pollos	• El sistema debe estar en funcionamiento. • El usuario Galponero debe haber iniciado sesión en el sistema
CP09	Evaluar imagen	Verificar evaluación correcta de imágenes de pollos	• El sistema debe estar en funcionamiento. • El usuario Galponero debe haber iniciado sesión en el sistema
CP10	Buscar anomalías	Buscar anomalías	• El sistema debe estar en funcionamiento. • Verificar búsqueda exitosa de anomalías en el crecimiento

CP11	Ingresar información del diagnóstico de anomalías	Ingresar información del diagnóstico de anomalías	• El sistema debe estar en funcionamiento. • Verificar ingreso correcto de información de diagnóstico

CP01					
Paso	**Descripción de pasos a seguir**	**Datos Entrada**	**Salida Esperada**	**¿OK?**	**Observaciones**
1	Ingresar a la página	Clic	Vista de inicio de sesión	✓	N/A
2	Iniciar sesión con credenciales válidas	Usuario y contraseña válidos	Acceso al sistema	✓	
3	Iniciar sesión con credenciales inválidas	Usuario y/o contraseña inválidos	Mensaje de error y no se permite el acceso al sistema	✓	

CP02					
Paso	**Descripción de pasos a seguir**	**Datos Entrada**	**Salida Esperada**	**¿OK?**	**Observaciones**
1	Ingresar a la página	Clic	Vista de inicio de sesión	✓	N/A
2	Completar el formulario de registro de datos de la avícola	Datos esenciales de la avícola (lote y galpón)	Datos registrados correctamente	✓	

CP03					
Paso	**Descripción de pasos a seguir**	**Datos Entrada**	**Salida Esperada**	**¿OK?**	**Observaciones**
1	Ingresar a la página	Clic	Vista de generación de reportes generales	✓	N/A
2	Seleccionar el tipo de reporte "Mortalidad", "Alimentación", "Peso", "Enfermedades y Tratamientos"	-	Vista con el reporte de mortalidad, alimentación, peso y enfermades y tratamientos generados	✓	

CP04					
Paso	**Descripción de pasos a seguir**	**Datos Entrada**	**Salida Esperada**	**¿OK?**	**Observaciones**
1	Ingresar a la página	Clic	Vista de registro de control de crecimiento	✓	N/A

2	Completar el formulario de registro de control de crecimiento	Datos de control de crecimiento (mortalidad diaria, recepción de alimentación diaria y peso promedio)	Datos registrados correctamente	✓	

CP05					
Paso	**Descripción de pasos a seguir**	**Datos Entrada**	**Salida Esperada**	**¿OK?**	**Observaciones**
1	Ingresar a la página	Clic	Vista de registro de mortalidad	✓	N/A
2	Completar el formulario de registro de mortalidad	Datos diarios de mortalidad y descarte	Datos registrados correctamente y cálculo del saldo de mortalidad	✓	

CP06					
Paso	**Descripción de pasos a seguir**	**Datos Entrada**	**Salida Esperada**	**¿OK?**	**Observaciones**
1	Ingresar a la página	Clic	Vista de registro de alimentación	✓	N/A
2	Completar el formulario de registro de alimentación	Datos de alimentación (conductor, cantidad de ingreso, cantidad consumida, fecha y hora de ingreso, saldo balanceado)	Datos registrados correctamente	✓	

CP07					
Paso	**Descripción de pasos a seguir**	**Datos Entrada**	**Salida Esperada**	**¿OK?**	**Observaciones**
1	Ingresar a la página	Clic	Vista de registro de peso	✓	N/A
2	Completar el formulario de registro de peso	Datos de peso y subir la imagen	Datos registrados correctamente y almacenamiento de la información del peso en la base de datos	✓	

CP08					
Paso	**Descripción de pasos a seguir**	**Datos Entrada**	**Salida Esperada**	**¿OK?**	**Observaciones**
1	Ingresar a la página	Clic	Vista de cálculo de crecimiento de pollos	✓	N/A

2	Utilizar los datos de peso y control de crecimiento registrados previamente	Datos de peso y control de crecimiento	Cálculo del crecimiento de los pollos en función de su edad	✓	

CP09					
Paso	**Descripción de pasos a seguir**	**Datos Entrada**	**Salida Esperada**	**¿OK?**	**Observaciones**
1	Ingresar a la página	Clic	Vista de evaluación de imagen	✓	N/A
2	Cargar y visualizar imágenes de los pollos en diferentes fases de desarrollo	Imágenes de los pollos	Evaluación del tamaño de los pollos y su crecimiento	✓	

CP10					
Paso	**Descripción de pasos a seguir**	**Datos Entrada**	**Salida Esperada**	**¿OK?**	**Observaciones**
1	Ingresar a la página	Clic	Vista de búsqueda de anomalías	✓	N/A
2	Realizar una búsqueda utilizando criterios específicos (enfermedades o tratamientos)	Criterios de búsqueda	Mostrar resultados con información adicional sobre las anomalías detectadas y el tratamiento a aplicar	✓	

CP11					
Paso	**Descripción de pasos a seguir**	**Datos Entrada**	**Salida Esperada**	**¿OK?**	**Observaciones**
1	Ingresar a la página	Clic	Vista de ingreso de información del diagnóstico de anomalías	✓	N/A
2	Completar el formulario de ingreso de información del diagnóstico de anomalías	Información detallada sobre el diagnóstico de anomalías (descripción, síntomas, tratamientos)	Datos registrados correctamente	✓	

3. Glosario

A continuación, se muestra la definición de todos los términos utilizados en el presente documento.

Término	Descripción
TIC	Trabajo de Integración Curricular

4. Bibliografía y referencias

Referencia	Título
Anexo 1 (del Documento del Proyecto de Integración — Curricular)	Especificación de requisitos de Software IEEE 830.

Anexo 9. Formulario de Satisfacción

Formulario de Satisfacción

Proyecto: Software para el Control de Crecimiento en pollos de engorde en la granja Avícola "Las Acacias" en el Cantón Balsas

Versión: 1.0

Fecha: 25/07/2023

Hoja de control

Organismo	Universidad Nacional de Loja		
Proyecto	Software para el Control de Crecimiento en pollos de engorde en la granja Avícola “Las Acacias” en el Cantón Balsas		
Entregable	Planes de Pruebas Unitarias		
Autor	Viviana Maricela Zambrano Romero		
Versión/Edición	1.0	**Fecha Versión**	25/07/2023
Aprobado por	Edwin René Guamán Quinche	**Fecha Aprobación**	28/08/2021
		Nº Total de Páginas	15

Ficha del documento

Versión	Fecha de revisión	Responsables	Descripción de modificación
1.0	28/04/2023	Viviana Maricela Zambrano Romero	N/A

Control de distribución

Nombre y Apellidos
Viviana Maricela Zambrano Romero
Ing. Edwin René Guamán Quinche, Mg.Sc

Índice

2. Introducción

Objetivo

El objetivo de este documento es que a partir de la obtención de información (formularios de satisfacción) que valida la funcionalidad y requerimientos del módulo de software por medio de la Aceptación de una muestra de personas de la granja avícola “Las Acacias”.
Estos formularios están diseñados para recopilar datos y métricas que permiten medir el nivel de satisfacción del usuario con respecto a la funcionalidad, confiabilidad, usabilidad, eficiencia y portabilidad. A través de la recolección de datos sistemática, se busca evaluar aspectos técnicos y funcionales, identificar problemas y áreas de mejora, y realizar un seguimiento de la satisfacción del usuario a lo largo del tiempo. Los formularios de satisfacción son una herramienta crucial en la gestión y mejora continua de sistemas y servicios tecnológicos, ya que proporcionan información valiosa para la toma de decisiones y el desarrollo de soluciones adaptadas a las necesidades y expectativas de los usuarios.

Propósito

El propósito fundamental de realizar formularios de satisfacción es optimizar la experiencia del usuario y garantizar la eficiencia y eficacia de los sistemas o servicios tecnológicos. Mediante la recopilación sistemática de datos, se busca evaluar la satisfacción y aceptación de los usuarios con respecto a los aspectos técnicos y funcionales del sistema, así como su interacción con el mismo. A través del análisis de los resultados obtenidos, se pretende identificar posibles deficiencias y áreas de mejora, para posteriormente implementar soluciones y mejoras que maximicen la usabilidad y la utilidad del sistema. El propósito también se extiende a la toma de decisiones informadas en el diseño y desarrollo de futuras versiones y actualizaciones del sistema o servicio, con el fin de satisfacer las necesidades y expectativas cambiantes de los usuarios y lograr una mayor fidelidad y retención de la audiencia. En última instancia, el propósito de estos formularios es lograr una mejora continua y una mayor satisfacción del usuario en el entorno tecnológico.

3. Formulario de Satisfacción de Funcionalidad

UNIVERSIDAD NACIONAL DE LOJA

Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables

Carrera de Ingeniería en Computación

FORMULARIO DE SATISFACCIÓN
FUNCIONALIDAD

Dando como pauta las siguientes afirmaciones, usted como usuario final del manejo del sistema avícola evaluará la funcionalidad, en cuanto a su experiencia del software para el Control de Crecimiento en pollos de engorde en la granja Avícola "Las Acacias" en el Cantón Balsas, para esto deberá de responder las siguiente preguntas utilizando una escala de 5 puntos, cuando 5 es la mayor, lo cual significa que usted está Totalmente de Acuerdo con la experiencia en la navegación, 4- Algo de Acuerdo, 3- Ni de Acuerdo Ni en desacuerdo, 2- Algo en Desacuerdo, y 1- Totalmente en Desacuerdo.

	1- Totalmente en Desacuerdo.	**2- Algo en Desacuerdo.**	**3- Ni de Acuerdo Ni en desacuerdo.**	**4- Algo de Acuerdo.**	**5- Totalmente de Acuerdo.**
¿Al momento de ingresar por primera vez al sistema lo logró desde el primero intento?					
¿Al momento de realizar consultas dentro del sistema, se generan de manera rápida y ágil?					
¿Está usted de acuerdo, en la forma en que están divididos y organizados los módulos dentro del sistema?					
¿En general, se encuentra usted satisfecho con la implementación del sistema avícola?					

4. Formulario de Satisfacción de Confiabilidad

UNIVERSIDAD NACIONAL DE LOJA

Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables

Carrera de Ingeniería en Computación

FORMULARIO DE SATISFACCIÓN

CONFIABILIDAD

Dando como pauta las siguientes afirmaciones, usted como usuario final del manejo del sistema avícola evaluará la confiabilidad, en cuanto a su experiencia del software para el Control de Crecimiento en pollos de engorde en la granja Avícola "Las Acacias" en el Cantón Balsas, para esto deberá de responder las siguiente preguntas utilizando una escala de 5 puntos, cuando 5 es la mayor, lo cual significa que usted está Totalmente de Acuerdo con la experiencia en la navegación, 4- Algo de Acuerdo, 3- Ni de Acuerdo Ni en desacuerdo, 2- Algo en Desacuerdo, y 1- Totalmente en Desacuerdo.

	1- Totalmente en Desacuerdo.	**2- Algo en Desacuerdo.**	**3- Ni de Acuerdo Ni en desacuerdo.**	**4- Algo de Acuerdo.**	**5- Totalmente de Acuerdo.**
¿Al ingresar datos al sistema, se guardan de manera rápida y ágil?					
Según su criterio, indique si, la información que maneja el sistema avícola.					

5. Formulario de Satisfacción de Usabilidad

UNIVERSIDAD NACIONAL DE LOJA

Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables

Carrera de Ingeniería en Computación

FORMULARIO DE SATISFACCIÓN
USABILIDAD

Dando como pauta las siguientes afirmaciones, usted como usuario final del manejo del sistema avícola evaluará la usabilidad, en cuanto a su experiencia del software para el Control de Crecimiento en pollos de engorde en la granja Avícola "Las Acacias" en el Cantón Balsas, para esto deberá de responder las siguiente preguntas utilizando una escala de 5 puntos, cuando 5 es la mayor, lo cual significa que usted está Totalmente de Acuerdo con la experiencia en la navegación, 4- Algo de Acuerdo, 3- Ni de Acuerdo Ni en desacuerdo, 2- Algo en Desacuerdo, y 1- Totalmente en Desacuerdo.

	1- Totalmente en Desacuerdo.	**2- Algo en Desacuerdo.**	**3- Ni de Acuerdo Ni en desacuerdo.**	**4- Algo de Acuerdo.**	**5- Totalmente de Acuerdo.**
¿Le resultó fácil el acceso al sistema avícola?					
¿Está usted de acuerdo con la apariencia y el diseño del sistema?					
¿Le resulta intuitiva (facilidad para reconocer opciones) la navegación dentro del sistema?					
¿Le resultan cómodos los botones, las ventanas, los cuadros de diálogo y los formularios?					

6. Formulario de Satisfacción de Eficiencia

UNIVERSIDAD NACIONAL DE LOJA

Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables

Carrera de Ingeniería en Computación

FORMULARIO DE SATISFACCIÓN
EFICIENCIA

Dando como pauta las siguientes afirmaciones, usted como usuario final del manejo del sistema avícola evaluará la eficiencia, en cuanto a su experiencia del software para el Control de Crecimiento en pollos de engorde en la granja Avícola "Las Acacias" en el Cantón Balsas, para esto deberá de responder las siguiente preguntas utilizando una escala de 5 puntos, cuando 5 es la mayor, lo cual significa que usted está Totalmente de Acuerdo con la experiencia en la navegación, 4- Algo de Acuerdo, 3- Ni de Acuerdo Ni en desacuerdo, 2- Algo en Desacuerdo, y 1- Totalmente en Desacuerdo.

	1- Totalmente en Desacuerdo.	**2- Algo en Desacuerdo.**	**3- Ni de Acuerdo Ni en desacuerdo.**	**4- Algo de Acuerdo.**	**5- Totalmente de Acuerdo.**
¿Las funcionalidades que brinda el sistema, fortalecen los procesos de control y seguimiento para la toma de decisiones en la avícola?					
¿El sistema desarrollado es capaz de procesar y almacenar datos de manera eficiente?					
¿Se ve mejorada la productividad de los trabajadores por la eficiencia del sistema?					

7. Formulario de Satisfacción de Portabilidad

UNIVERSIDAD NACIONAL DE LOJA

Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables

Carrera de Ingeniería en Computación

FORMULARIO DE SATISFACCIÓN

PORTABILIDAD

Dando como pauta las siguientes afirmaciones, usted como usuario final del manejo del sistema avícola evaluará la portabilidad, en cuanto a su experiencia del software para el Control de Crecimiento en pollos de engorde en la granja Avícola "Las Acacias" en el Cantón Balsas, para esto deberá de responder las siguiente preguntas utilizando una escala de 5 puntos, cuando 5 es la mayor, lo cual significa que usted está Totalmente de Acuerdo con la experiencia en la navegación, 4- Algo de Acuerdo, 3- Ni de Acuerdo Ni en desacuerdo, 2- Algo en Desacuerdo, y 1- Totalmente en Desacuerdo.

	1- Totalmente en Desacuerdo.	**2- Algo en Desacuerdo.**	**3- Ni de Acuerdo Ni en desacuerdo.**	**4- Algo de Acuerdo.**	**5- Totalmente de Acuerdo.**
¿En relación con la facilidad de acceso, recomendaría este sistema avícola a otras medianas y pequeñas empresas que se dedican a la avicultura?					

8. Bibliografía y referencias

Referencia	Título
Ref. 1	Documento de Especificación de Requisitos del Sistema

Anexo 10. Formularios de Satisfacción Contesta

UNIVERSIDAD NACIONAL DE LOJA

Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables

Carrera de Ingeniería en Computación

FORMULARIO DE SATISFACCIÓN
FUNCIONALIDAD

Dando como pauta las siguientes afirmaciones, usted como usuario final del manejo del sistema avícola evaluará la funcionalidad, en cuanto a su experiencia del software para el Control de Crecimiento en pollos de engorde en la granja Avícola "Las Acacias" en el Cantón Balsas, para esto deberá de responder las siguiente preguntas utilizando una escala de 5 puntos, cuando 5 es la mayor, lo cual significa que usted está Totalmente de Acuerdo con la experiencia en la navegación, 4- Algo de Acuerdo, 3- Ni de Acuerdo Ni en desacuerdo, 2- Algo en Desacuerdo, y 1- Totalmente en Desacuerdo.

	1- Totalmente en Desacuerdo.	2- Algo en Desacuerdo.	3- Ni de Acuerdo Ni en desacuerdo.	4- Algo de Acuerdo.	5- Totalmente de Acuerdo.
¿Al momento de ingresar por primera vez al sistema lo logró desde el primero intento?					X
¿Al momento de realizar consultas dentro del sistema, se generan de manera rápida y ágil?					X
¿Está usted de acuerdo, en la forma en que están divididos y organizados los módulos dentro del sistema?					X
¿En general, se encuentra usted satisfecho con la implementación del sistema avícola?					X

Nombre	Cargo	Firma
Fernando Bravo	Administrador - Jefe	
Jeovanny Pinto	Galponero – Representante legal	
Mauricio Ramírez	Veterinario	

UNIVERSIDAD NACIONAL DE LOJA

Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables

Carrera de Ingeniería en Computación

FORMULARIO DE SATISFACCIÓN

PORTABILIDAD

Dando como pauta las siguientes afirmaciones, usted como usuario final del manejo del sistema avícola evaluará la portabilidad, en cuanto a su experiencia del software para el Control de Crecimiento en pollos de engorde en la granja Avícola "Las Acacias" en el Cantón Balsas, para esto deberá de responder las siguiente preguntas utilizando una escala de 5 puntos, cuando 5 es la mayor, lo cual significa que usted está Totalmente de Acuerdo con la experiencia en la navegación, 4- Algo de Acuerdo, 3- Ni de Acuerdo Ni en desacuerdo, 2- Algo en Desacuerdo, y 1- Totalmente en Desacuerdo.

	1- Totalmente en Desacuerdo.	2- Algo en Desacuerdo.	3- Ni de Acuerdo Ni en desacuerdo.	4- Algo de Acuerdo.	5- Totalmente de Acuerdo.
¿En relación con la facilidad de acceso, recomendaría este sistema avícola a otras medianas y pequeñas empresas que se dedican a la avicultura?					X

Nombre	Cargo	Firma
Fernando Bravo	Administrador - Jefe	
Jeovanny Pinto	Galponero – Representante legal	
Mauricio Ramírez	Veterinario	

UNIVERSIDAD NACIONAL DE LOJA

Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables

Carrera de Ingeniería en Computación

FORMULARIO DE SATISFACCIÓN

EFICIENCIA

Dando como pauta las siguientes afirmaciones, usted como usuario final del manejo del sistema avícola evaluará la eficiencia, en cuanto a su experiencia del software para el Control de Crecimiento en pollos de engorde en la granja Avícola "Las Acacias" en el Cantón Balsas, para esto deberá de responder las siguiente preguntas utilizando una escala de 5 puntos, cuando 5 es la mayor, lo cual significa que usted está Totalmente de Acuerdo con la experiencia en la navegación, 4- Algo de Acuerdo, 3- Ni de Acuerdo Ni en desacuerdo, 2- Algo en Desacuerdo, y 1- Totalmente en Desacuerdo.

	1- Totalmente en Desacuerdo.	2- Algo en Desacuerdo.	3- Ni de Acuerdo Ni en desacuerdo.	4- Algo de Acuerdo.	5- Totalmente de Acuerdo.
¿Las funcionalidades que brinda el sistema, fortalecen los procesos de control y seguimiento para la toma de decisiones en la avícola?					X
¿El sistema desarrollado es capaz de procesar y almacenar datos de manera eficiente?					X
¿Se ve mejorada la productividad de los trabajadores por la eficiencia del sistema?					X

Nombre	Cargo	Firma
Fernando Bravo	Administrador - Jefe	
Jeovanny Pinto	Galponero – Representante legal	
Mauricio Ramírez	Veterinario	

UNIVERSIDAD NACIONAL DE LOJA

Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables

Carrera de Ingeniería en Computación

FORMULARIO DE SATISFACCIÓN
USABILIDAD

Dando como pauta las siguientes afirmaciones, usted como usuario final del manejo del sistema avícola evaluará la usabilidad, en cuanto a su experiencia del software para el Control de Crecimiento en pollos de engorde en la granja Avícola "Las Acacias" en el Cantón Balsas, para esto deberá de responder las siguiente preguntas utilizando una escala de 5 puntos, cuando 5 es la mayor, lo cual significa que usted está Totalmente de Acuerdo con la experiencia en la navegación, 4- Algo de Acuerdo, 3- Ni de Acuerdo Ni en desacuerdo, 2- Algo en Desacuerdo, y 1- Totalmente en Desacuerdo.

	1- Totalmente en Desacuerdo.	2- Algo en Desacuerdo.	3- Ni de Acuerdo Ni en desacuerdo.	4- Algo de Acuerdo.	5- Totalmente de Acuerdo.
¿Le resultó fácil el acceso al sistema avícola?					X
¿Está usted de acuerdo con la apariencia y el diseño del sistema?					X
¿Le resulta intuitiva (facilidad para reconocer opciones) la navegación dentro del sistema?					X
¿Le resultan cómodos los botones, las ventanas, los cuadros de diálogo y los formularios?					X

Nombre	Cargo	Firma
Fernando Bravo	Administrador - Jefe	
Jeovanny Pinto	Galponero – Representante legal	
Mauricio Ramírez	Veterinario	

UNIVERSIDAD NACIONAL DE LOJA

Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables

Carrera de Ingeniería en Computación

FORMULARIO DE SATISFACCIÓN

CONFIABILIDAD

Dando como pauta las siguientes afirmaciones, usted como usuario final del manejo del sistema avícola evaluará la confiabilidad, en cuanto a su experiencia del software para el Control de Crecimiento en pollos de engorde en la granja Avícola "Las Acacias" en el Cantón Balsas, para esto deberá de responder las siguiente preguntas utilizando una escala de 5 puntos, cuando 5 es la mayor, lo cual significa que usted está Totalmente de Acuerdo con la experiencia en la navegación, 4- Algo de Acuerdo, 3- Ni de Acuerdo Ni en desacuerdo, 2- Algo en Desacuerdo, y 1- Totalmente en Desacuerdo.

	1- Totalmente en Desacuerdo.	2- Algo en Desacuerdo.	3- Ni de Acuerdo Ni en desacuerdo.	4- Algo de Acuerdo.	5- Totalmente de Acuerdo.
¿Al ingresar datos al sistema, se guardan de manera rápida y ágil?					X
Según su criterio, indique si, la información que maneja el sistema avícola.				X	

Nombre	Cargo	Firma
Fernando Bravo	Administrador - Jefe	
Jeovanny Pinto	Galponero – Representante legal	
Mauricio Ramírez	Veterinario	

Anexo 11. Tabla compartiva de Arquitecturas (CNN)

Arquitectura	Año de Publicación	Número de Capas	Tiempo de Procesamiento (segundos)	Buena Precisión	Cantidad de Imágenes (Mín-Máx)	Versatilidad	Precisión	Facilidad de Implementación	Tamaño del Modelo	Disponibilidad de Pre-entrenamiento	Comunidad y Soporte	Ventajas	Desventajas
Transformer	2017	6	2.5	Sí	1000-10000	Alta	Alta	Media	Grande	Sí	Grande	Alto rendimiento en tareas de procesamiento de lenguaje natural (NLP)	Requiere grandes conjuntos de datos para el entrenamiento
BERT	2018	12	3.8	Sí	500-5000	Alta	Alta	Media	Grande	Sí	Grande	Excelente en comprensión del lenguaje natural (NLP)	Demanda de recursos computacionales significativos
GPT	2018	48	5.2	Sí	100-1000	Alta	Alta	Media	Grande	Sí	Grande	Buen desempeño en generación de texto y comprensión del lenguaje	Requiere gran cantidad de datos para el entrenamiento
ResNet	2015	152	15.3	Sí	1000-10000	Media	Alta	Alta	Grande	Sí	Grande	Excelente en tareas de visión por computadora	Profundidad puede conducir a problemas de sobreajuste
EfficientNet	2019	528	8.7	Sí	500-5000	Alta	Alta	Alta	Mediano	Sí	Grande	Eficiente en términos de rendimiento y tamaño del modelo	Requiere ajustes finos para la adaptación a diferentes tareas
DenseNet	2016	201	12.6	Sí	500-5000	Alta	Alta	Media	Grande	Sí	Grande	Conexiones densas entre capas que promueven un mejor flujo de información	Menor consumo de memoria durante la inferencia
VGG	2014	19	10.4	Sí	1000-10000	Baja	Alta	Alta	Grande	No	Grande	Fácil de entender e implementar	Requiere más parámetros y recursos computacionales
MobileNet V2	2018	53	7.1	Sí	500-10000	Alta	Alta	Alta	Pequeño	Sí	Grande	Eficiente en dispositivos móviles y embebidos	Mayor precisión en comparación con arquitecturas más grandes

Anexo 12. Objetivos de desempeño – Tabla comparativa

OBJETIVOS DE DESEMPEÑO						
COMPARATIVA				VALORES REALES		
Edad de días	*Crecimiento excelente*	*Buen crecimiento*	*Peso de ideal*	*Peso*	*Edad en dias*	*Fases*
0	48	38	43	38 a 48	0	Pre-Inicio
1	61	51	56	51 a 61	1	Pre-Inicio
2	73	63	68	63 a 73	2	Pre-Inicio
3	89	79	84	79 a 89	3	Pre-Inicio
4	107	97	102	97 a 107	4	Pre-Inicio
5	127	117	122	117 a 127	5	Pre-Inicio
6	157	147	152	147 a 157	6	Pre-Inicio
7	187	177	182	177 a 187	7	Pre-Inicio
8	207	197	202	197 a 207	8	Inicio
9	227	217	222	217 a 227	9	Inicio
10	257	247	252	247 a 257	10	Inicio
11	310	300	305	300 a 310	11	Inicio
12	350	340	345	340 a 350	12	Inicio
13	395	385	390	385 a 395	13	Inicio
14	425	415	420	415 a 425	14	Inicio
15	485	475	480	475 a 485	15	Inicio
16	555	545	550	545 a 555	16	Inicio
17	625	615	620	615 a 625	17	Inicio
18	695	685	690	685 a 695	18	Inicio
19	775	765	770	765 a 775	19	Desarrollo
20	845	835	840	835 a 845	20	Desarrollo
21	920	910	915	910 a 920	21	Desarrollo
22	1012	1002	1007	1002 a 1012	22	Desarrollo
23	1098	1088	1093	1088 a 1098	23	Desarrollo
24	1172	1162	1167	1162 a 1172	24	Desarrollo
25	1246	1236	1241	1236 a 1246	25	Desarrollo
26	1315	1305	1310	1305 a 1315	26	Desarrollo
27	1417	1407	1412	1407 a 1417	27	Desarrollo
28	1500	1490	1495	1490 a 1500	28	Desarrollo
29	1529	1519	1524	1519 a 1529	29	Desarrollo
30	1603	1593	1598	1593 a 1603	30	Engorde
31	1687	1677	1682	1677 a 1687	31	Engorde
32	1786	1776	1781	1776 a 1786	32	Engorde
33	1861	1851	1856	1851 a 1861	33	Engorde
34	1957	1947	1952	1947 a 1957	34	Engorde
35	2103	2093	2098	2093 a 2103	35	Engorde
36	2159	2149	2154	2149 a 2159	36	Engorde
37	2235	2225	2230	2225 a 2235	37	Engorde
38	2303	2293	2298	2293 a 2303	38	Engorde
39	2386	2376	2381	2376 a 2386	39	Engorde
40	2502	2492	2497	2492 a 2502	40	Engorde
41	2582	2572	2577	2572 a 2582	41	Engorde
42	2659	2649	2654	2649 a 2659	42	Engorde
43	2753	2743	2748	2743 a 2753	43	Engorde
44	2862	2852	2857	2852 a 2862	44	Engorde
45	2948	2938	2943	2938 a 2948	45	Engorde
46	3080	3070	3075	3070 a 3080	46	Engorde
47	3189	3179	3184	3179 a 3189	47	Engorde
48	3251	3241	3246	3241 a 3251	48	Engorde
49	3402	3392	3397	3392 a 3402	49	Engorde

Anexo 13. Manual de Usuario

[MANUAL DEL USUARIO]

[ADMINISTRADOR – GALPONERO - VETERINARIO]

Software para el Control de crecimiento en pollos de engorde en la granja Avícola “Las Acacias” en el Cantón Balsas.

Versión 1.0

Elaborado por:

Viviana Maricela Zambrano Romero

Revisado por:

Ing. Edwin René Guamán Quinche, Mg.Sc

2023-2024

HISTORIAL DE REVISIONES

Revisión	Fecha	Responsable	Descripción de la modificación
1.0	24/07/2023	Viviana Maricela Zambrano Romero	Documento Inicial
2.0	30/07/2023	Viviana Maricela Zambrano Romero	Actualización de Interfaces

CONTENIDOS

1. INTRODUCCIÓN

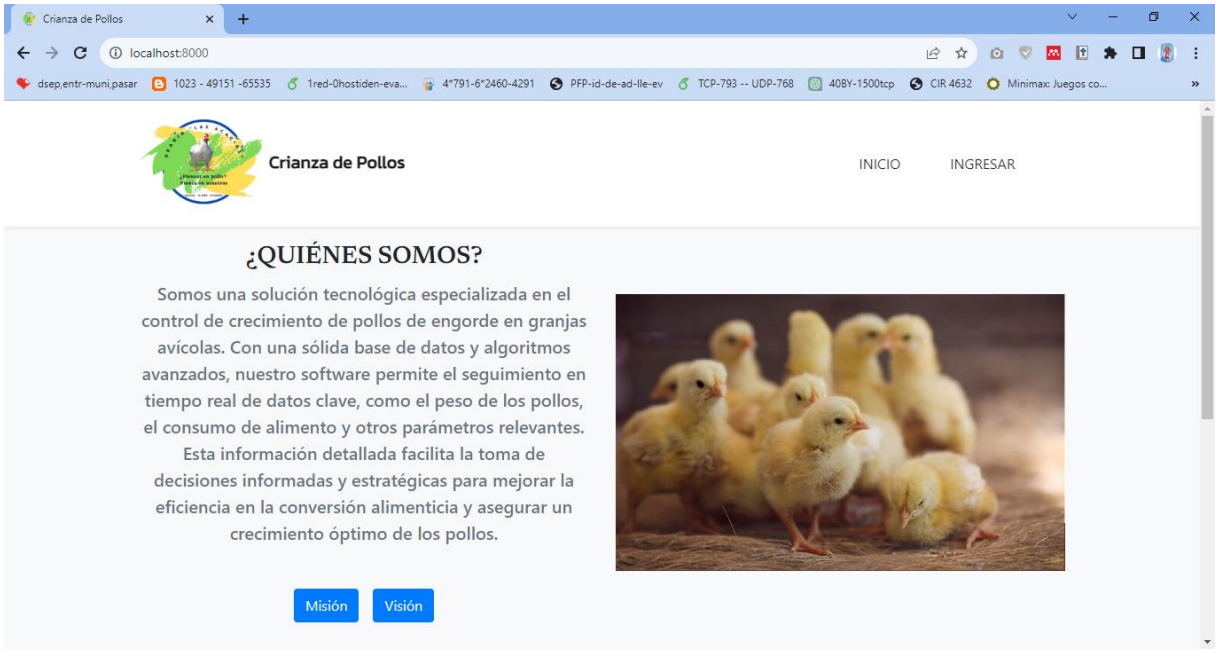
En el mundo de la avicultura, la gestión eficiente de una granja es fundamental para garantizar un óptimo crecimiento y desarrollo de los pollos de engorde. Para lograr este objetivo, es esencial contar con un equipo de profesionales capacitados y comprometidos. Es por eso que este software ha sido desarrollado con la finalidad de brindar apoyo y facilitar las tareas diarias a través de una documentación adecuada para guiar a cada role fundamental dentro de la granja: el administrador, el galponero y el veterinario.

A continuación, se presenta un resumen de los puntos clave:

- **Servicios:** Pantalla principal y botones de manera general.
- **Acceso a la plataforma:** Se menciona el acceso a la plataforma, lo que implica que se necesita un inicio de sesión para interactuar con el software.
- **Login:** Se detalla cómo se lleva a cabo el proceso de inicio de sesión en el software.
- **Roles:** Se mencionan los diferentes roles de usuario dentro del sistema, incluyendo el Administrador, Galponero y Veterinario.
- **Privilegios de usuario:** Se explica que cada rol tiene diferentes privilegios y responsabilidades dentro del software.
- **Módulos:** Se describen los diferentes módulos disponibles en el software, específicos para cada rol de usuario.
    - Módulos rol Administrador: Registro de datos avícolas y generación de reportes generales.
    - Módulos rol Galponero: Control de crecimiento de los pollos y módulo de enfermedades y tratamiento.
    - Módulos rol Veterinario: Módulo de enfermedades y tratamiento y módulo de reportes generales.

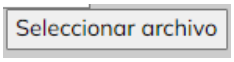
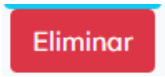
2. SERVICIOS

Al ingresar a la plataforma de Crianza de Pollos, se mostrará la pantalla principal en la cual se detallará información como: Quienes somos, Misión y Visión. Además del Login dependiendo el Rol (administrador, galponero o veterinario).



BOTONES Y OPCIONES RELACIONADOS CON TODOS LOS ROLES DE MANERA GENERAL.

Guardar	Este botón se utiliza para guardar los cambios que has realizado en un formulario. Al presionarlo, los datos se almacenan en el sistema.
Cancelar	Este botón se utiliza para descartar los cambios que hayas realizado antes de guardarlos. Si has realizado modificaciones y no deseas guardarlas, puedes presionar "Cancelar" para regresar al estado anterior.
Identificar	Este botón se utiliza para identificar en qué fase de control de crecimiento se encuentra el pollo.
Regresar	Este botón se usa para volver a la pantalla o página anterior.
Imprimir	Al presionar este botón, se envía el contenido del documento o página actual a una impresora conectada, lo que permite obtener una copia física del contenido en papel.

	Este botón se utiliza para carguen un archivo desde su dispositivo. Al hacer clic en este botón, se abre el explorador de archivos para que el usuario pueda seleccionar el archivo que desea cargar.
	Su función es permitir que los usuarios hagan cambios o ajustes en el contenido o la información presente en un documento, formulario o cualquier otra interfaz.
	Su función es permitir que los usuarios elimienen la información que se encuentre presente en un documento, formulario o cualquier otra interfaz.
	Similar al botón "Regresar", este botón también se encuentra en aplicaciones o páginas web y permite volver a la pantalla o página anterior.
	Al presionar este botón, los usuarios cierran sesión o salen completamente del sistema o plataforma en la que estaban trabajando.

3. ACCESO A LA PLATAFORMA

Acceso al software

- ➜ Abrir el navegador y digitar https://crianzapollosbalsas.com/
- ➜ A continuación, en la parte superior derecha hay un menú denominado **[Ingresar]** hacer clic.

4. LOGIN

Para poder ingresar al sistema del LOGIN, deben acceder sus credenciales (usuario y contraseña).

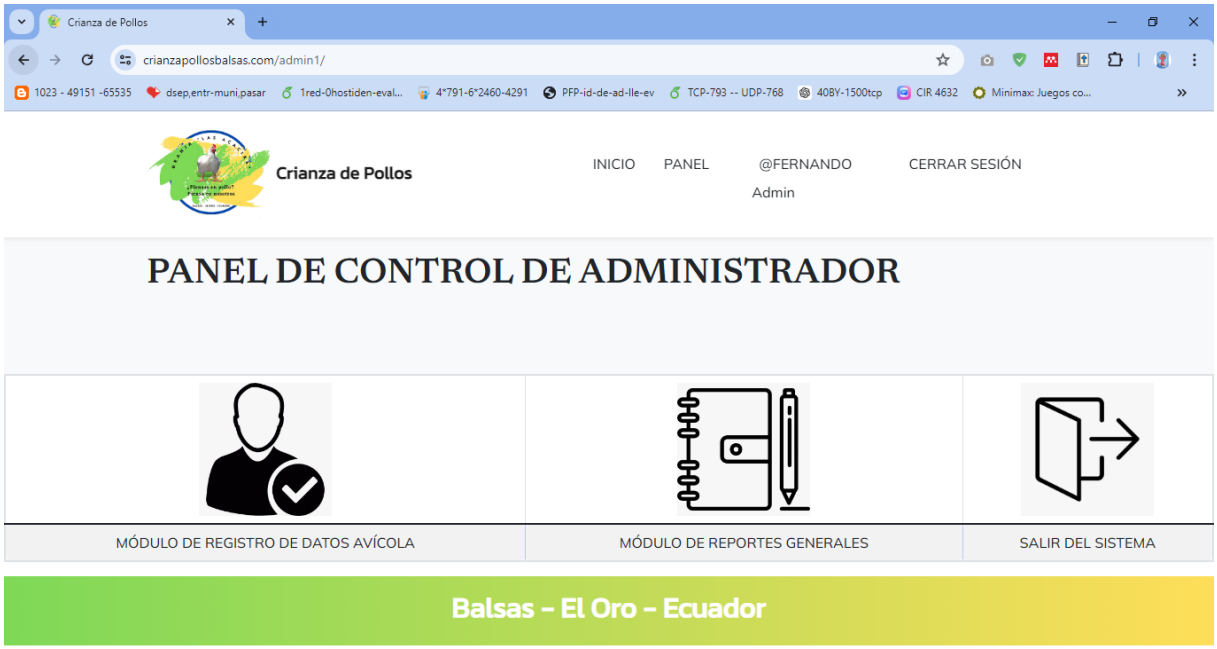
1. **Nombre de usuario:** Se registra el nombre de usuario dependiendo el rol que sea para poder ingresar al sistema y visualizar las funciones.
2. **Contraseña:** Se registra la contraseña para poder ingresar al sistema y navegar en sus funciones.

5. ROLES

Una vez autenticado con las respectivas credenciales como usuario y clave, el sistema le permitirá visualizar el menú principal de acuerdo eso depende a que rol corresponde a continuación se muestran:

- Administrador.
- Galponero.
- Veterinario.

5.1 Administrador.



A este módulo únicamente tendrá acceso el administrador que esta a cargo de la avícola, podrá registrar al modulo de registro de datos de la avícola y el módulo de reportes generales.

5.2 Galponero

A este módulo únicamente tendrá acceso el galponero que esta a cargo del lote, y le permitirá ingresar al módulo de control crecimiento, y adicional a visualizar el reporte de enfermedades y tratamientos.

5.3 Veterinario

A este módulo únicamente tendrá acceso el veterinario que esta a cargo del lote, le permitirá registrar en el módulo de enfermedades y tratamiento y además visualizar el módulo de reportes generales.

6. PRIVILEGIOS DE USUARIO

6.1 Privilegio de Administrador

El rol del administrador dentro de sistema será registrar los datos de la avícola como lote y galpón. Además de generar los reportes generales. Los módulos que el administrador tendrá acceso son los siguientes:

1. Módulo de registro de la avícola
2. Módulo de reportes generales

6.2 Privilegio de Galponero

El rol del galponero sera ingresar datos diarios en el control de crecimiento como, mortalidad diaria, recepción de alimentos balanceados, peso promedio. Además de visualizar los diagnósticos de las anomalías del pollo. Los módulos que galponero tendrá acceso son los siguientes:

1. Módulo de control de crecimiento
2. Módulo enfermedades y tratamientos

6.3 Privilegio de Veterinario

El rol del veterinario dentro del sistema es el ver enfermedades y aplicar el tratamiento adecuado para mejorar la toma de decisiones. Además de generar los reportes generales. Los módulos que el administrador tendrá acceso son los siguientes:

1. Módulo de enfermedades y tratamientos
2. Módulo de reportes generales

7. MÓDULOS

7.1 Rol Administrador

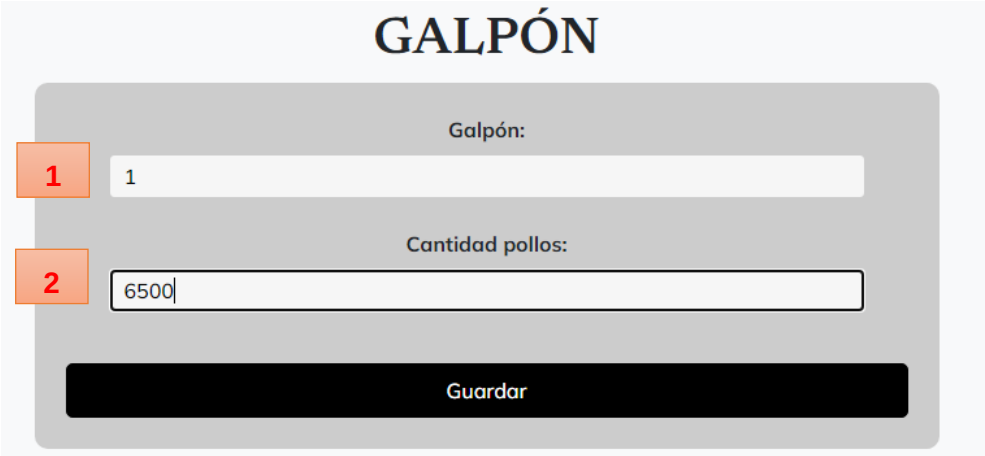
7.1.1 Módulo de registro de datos avícola

Este submódulo se encarga del registro y visualización del lote y galpón de los pollos, los cuales se detallan a continuación:

Submódulo registro del lote:

1. **Lote:** se registra el nombre del lote.
2. **Total de pollos de lote:** cantidad total de pollos ingresados en dicho lote.

Submódulo registro del galpón:



1. **Galpón:** se registra el número de galpón.
2. **Cantidad de pollos:** se registra la cantidad de pollos que ingresan al galpón.

Submódulo ver del lote:

Este submódulo se encarga mostrar los datos ingresados del lote como el nombre del lote, y la cantidad total de pollos ingresados en dicho lote.

Submódulo ver del galpón:

Este submódulo se encarga mostrar los datos ingresados del galpón como el nombre del lote, y la cantidad de pollos ingresados.

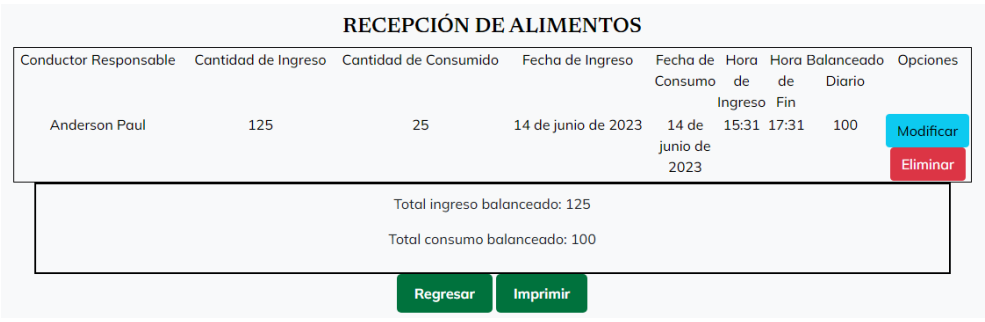
7.1.2 Módulo de Reportes Generales

Este submódulo se encarga de la visualización de los reportes realizados tanto por el galponero y veterinario, los cuales se detallan a continuación:

Submódulo ver mortalidad:

Se encarga de la visualización del registro diario de las muertes de las aves, y dando un valor total de cuantos pollos total van muertos, además de imprimir el reporte.

Submódulo ver recepción de alimento:



Se encarga de la visualización del registro del balanceado, y dando un valor total de cuantos balanceados han sido los consumidos diariamente, además del total de ingreso y consumo de balanceado, se puede imprimir el reporte.

Submódulo ver peso promedio:

Se encarga de la visualización del registro del peso promedio, dependiendo el peso en gramos, la edad de identificación en días.

Submódulo ver peso promedio:

Se encarga de la visualización del registro del peso promedio, dando dependiendo una imagen en qué fase se encuentra y dependiendo el peso la edad del pollo, además de imprimir el reporte.

Submódulo ver enfermedades y tratamientos:

Se encarga de la visualización del registro del de cuáles han sido las enfermedades, en la cual se puede ver la causa para poder mejorar y el tratamiento para poder medicar, además de imprimir el reporte.

7.2 Rol Galponero

7.2.1 Módulo de control de crecimiento

Este submódulo se encarga del registro de mortalidad, recepción de alimentos y peso promedio de los pollos, los cuales se detallan a continuación:

Submódulo registro de mortalidad:

Este submódulo se encarga del registro diario de mortalidad. El el cual el deberá registrar los diferentes campos.

1. **Lote:** se selecciona el lote en el cual donde han ocurrido las muertes.
2. **Galpón:** se selecciona el número de galpón acuerdo donde han ocurrido las muertes.
3. **Fecha Mortalidad:** se registra la fecha del día que se registró la mortalidad de los pollos.
4. **Cantidad de mortalidad:** se registra la mortalidad por cualquier circunstancia que se halla presentad.
5. **Cantidad descarte:** se registra en caso que sea necesario descartar por cualquier situación.

6. **Imagen:** se sube la imagen del total de los pollos muertos ya sea de mortalidad y descarte.

Submódulo registro de recepción de alimentos:

Este Submódulo se encarga del control de la alimentación de las aves, se registrará el conductor responsable, la cantidad de sacos de balanceados ingresados y los que serán utilizados diariamente en los respectivos lotes dentro de la avícola. El galponero deberá llenar los siguientes campos:

1. **Conductor responsable:** se registra el nombre de la persona que lleva el balanceado a la granja.
2. **Cantidad de ingreso:** se registra la cantidad total que ingresaron de balanceado.
3. **Cantidad consumido:** se registra la cantidad de balanceado que son utilizados en ese día
4. **Lote:** se selecciona el lote en el cual donde ingresa el balanceado.
5. **Fecha Ingreso**: se registra la fecha del día que se registró el ingreso del balanceado.
6. **Fecha Consumo**: se registra la fecha del día que se registró consumo del balanceado.
7. **Hora ingreso:** se registra la hora que ingreso el conductor a realizar la entrega del balanceado.
8. **Hora fin:** se registra la hora de salida del conductor que entrega el balanceado.

Submódulo registro de peso promedio:

Este Submódulo se encarga del control de los pesos para poder verificar cuantos días. El mismo que deberá registrar los diferentes campos:

1. **Fecha:** se registra la fecha del registro del peso.

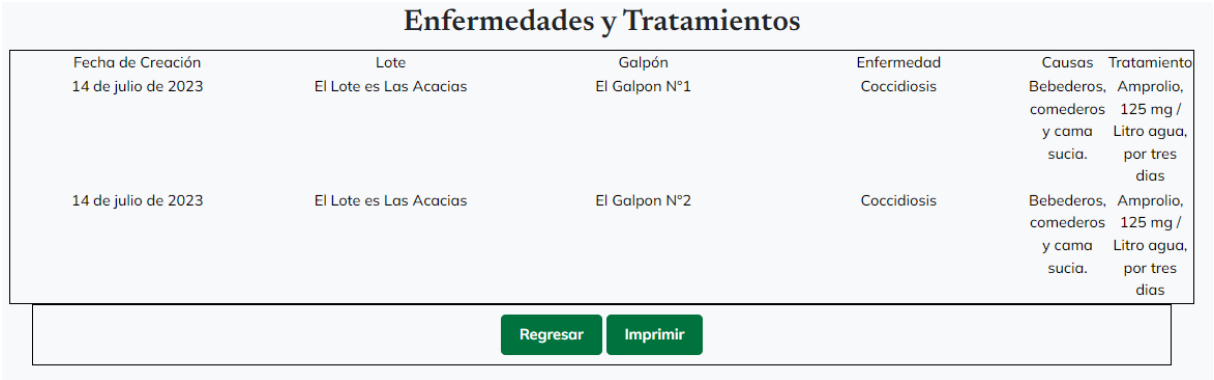
2. **Peso:** se registra el peso obtenido, para obtener la edad en días y verificar en la tabla comparativa, para ver si está creciendo de manera adecuada.
3. **Identificar:** se identifica según los gramos, para ver cuantos días tienen.

Submódulo registro de identificación de crecimiento:

Este Submódulo se encarga del control de identificar el crecimiento, y ver en que fase se encuentra. El mismo que deberá registrar los diferentes campos:

1. **Archivo:** se sube una imagen, de la foto que se tomó para poder ver en qué fase se encuentra el pollo.
2. **Identificar:** se identifica la imagen para saber en que fase se encuentra.

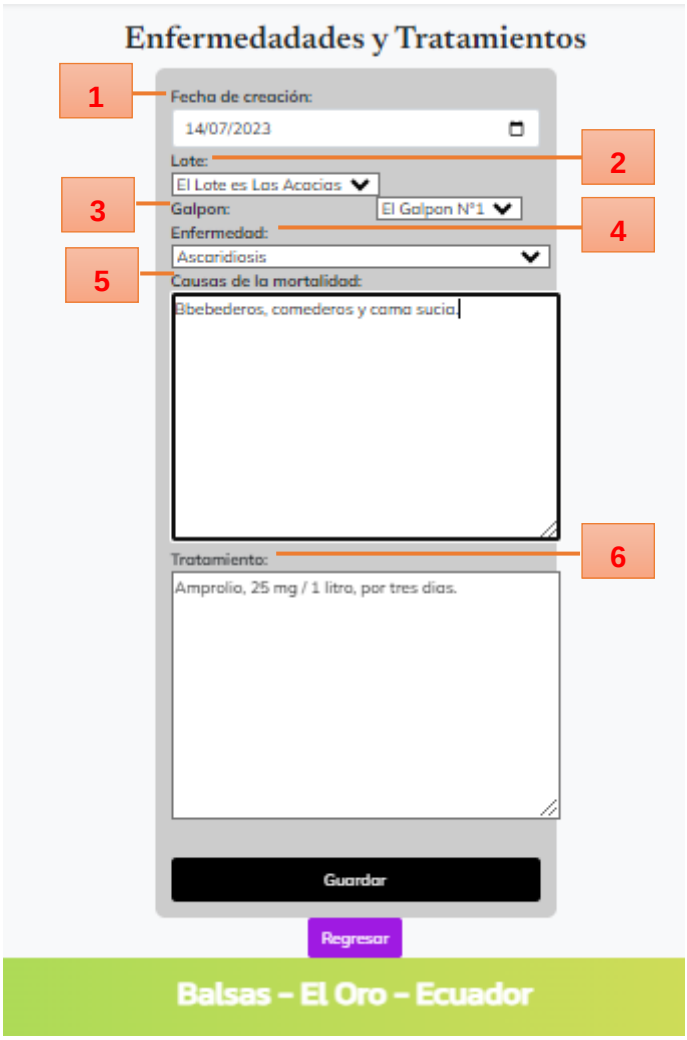
7.2.2 Modulo Enfermedades y Tratamiento



Este módulo se encarga de mostrar las enfermedades y tratamientos, que han sido realizados, para que el galponero pueda ver las indicaciones del tratamiento del medicamento y las causas de la enfermedad para poder mejorar. Además, me permite imprimir por si es necesario.

7.3 Rol Veterinario

7.3.1 Modulo Enfermedades y Tratamiento



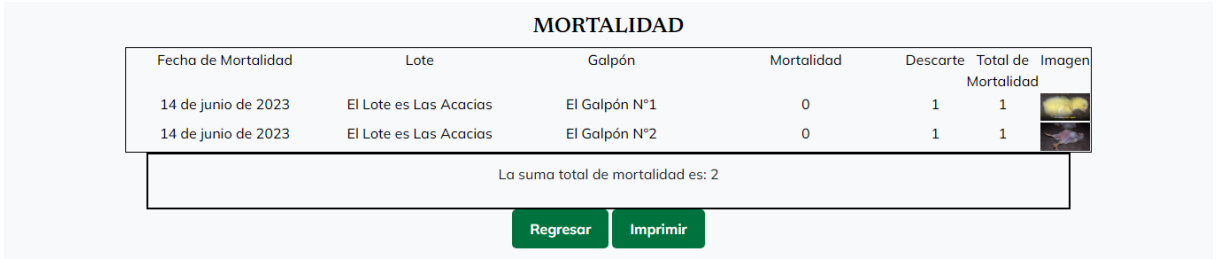
Este módulo se encarga del control de las enfermedades y tratamientos de las aves por lote, y galpón este registro es importante para el control del crecimiento de los pollos de la avícola. El veterinario debe registrar los siguientes campos:

1. **Fecha de creación:** se registra la fecha de registro en la cual se detectó alguna anomalía en el pollo.
2. **Lote:** se selecciona el lote en el cual donde han ocurrido las anomalías.
3. **Galpón:** se selecciona el número de galpón de acuerdo donde han ocurrido las anomalías.
4. **Enfermedad:** se selecciona la enfermedad que se detectó. Además, que se puede agregar en caso que exista una nueva enfermedad.
5. **Causas:** se describe la situación de las causas de la enfermedad del pollo.
6. **Tratamiento:** se describe el tratamiento como se debe medicar al pollo.

7.3.2 Modulo Reportes Generales

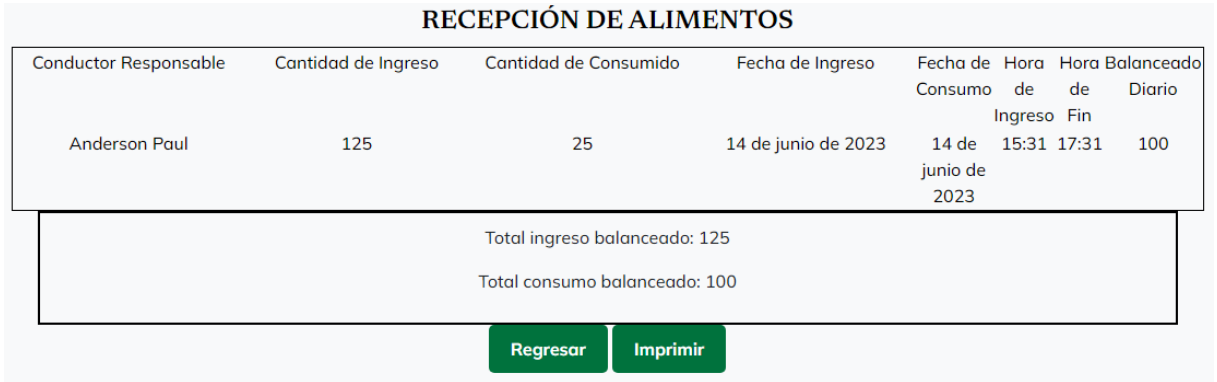
Este submódulo se encarga de la visualización de los reportes realizados tanto por el galponero y veterinario, los cuales se detallan a continuación:

Submódulo ver mortalidad:



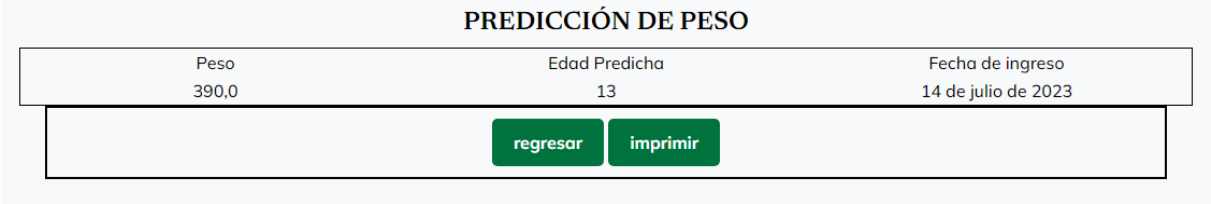
Se encarga de la visualización del registro diario de las muertes de las aves, y dando un valor total de cuantos pollos total van muertos, además de imprimir el reporte.

Submódulo ver recepción de alimento:



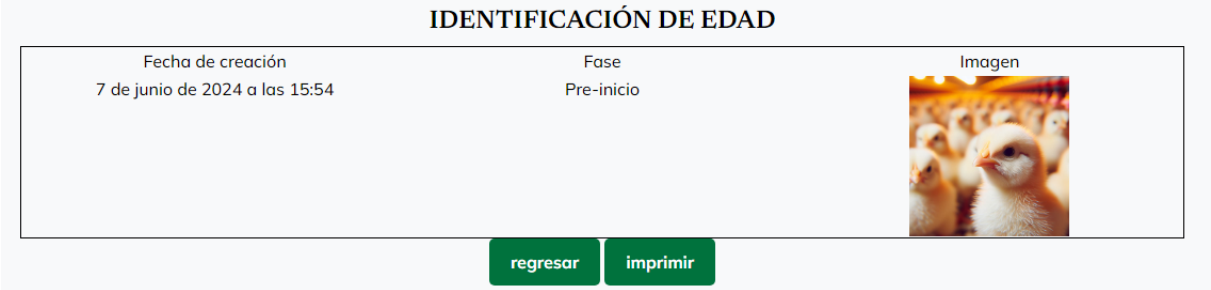
Se encarga de la visualización del registro del balanceado, y dando un valor total de cuantos balanceados han sido los consumidos, además de imprimir el reporte.

Submódulo ver peso promedio:



Se encarga de la visualización del registro del peso promedio, dado del peso en gramos del pollo, se encuentra la edad predicha, además de imprimir el reporte.

Submódulo ver identificación de edad:



Se encarga de la visualización del registro de identificación de edad, dado dependiendo una imagen se identifica en qué fase se encuentra el pollo, además de imprimir el reporte.

Submódulo ver enfermedades y tratamientos:

Se encarga de la visualización del registro del de cuáles han sido las enfermedades, en la cual se puede ver la causa para poder mejorar y el tratamiento para poder medicar, además de imprimir el reporte.

8. FIRMAS DE RESPONSABILIDAD

Acción	Funcionario	Firmas
Elaborado por:	Viviana Maricela Zambrano Romero **Analista de Sistemas Informáticos**	
Revisado por:	Ing. Edwin René Guamán Quinche, Mg.Sc **Especialista de Sistemas de Información**	

Anexo 14. Certificado de traducción

CERTIFICADO DE TRADUCCIÓN

Loja, 14 de agosto de 2023

Yo, **Adriana Elizabeth Cango Patiño** con numero de cedula 1103653133, Magister en Pedagogía de los Idiomas Nacionales y Extranjeros. Mención en Enseñanza de Inglés.

CERTIFICO:

Haber realizado la traducción de español al idioma inglés del resumen del trabajo de titulación denominado: **Software para el Control de Crecimientos en pollos de engorde en la granja Avícola "Las Acacias" en el Cantón Balsas** de autoría de la estudiante **Viviana Maricela Zambrano Romero** con número de cedula **0705764587**, estudiante de la carrera de Ingeniería en Computación Facultad de la Energía, las Industrias y los Recursos Naturales No Renovables de la Universidad Nacional de Loja. Dicho estudio se encontró bajo la dirección del Ing. Edwin René Guamán Quinche, Mg. Sc, previó a la obtención del título de Ingeniera en Ciencias de la Computación.

Es todo cuanto puedo certificar en honor a la verdad, y autorizo al interesado hacer uso del documento para los fines académicos correspondientes.

Atentamente,

Mg.Sc. Adriana Cango

Mg. Sc. Adriana Elizabeth Cango Patiño
Magister en Pedagogía de los Idiomas Nacionales y Extranjeros. Mención en Enseñanza de Inglés
Registro Senescyt 1049-2022-2589539
Celular: 0989814921
Email: adrianacango@hotmail.com