Chapter 1

Conclusiones, Limitaciones y Trabajos Futuros

1.1 Conclusión

En este proyecto se desarrolló un sistema funcional y extensible diseñado para su implementación en entornos ganaderos, logrando mejorar la precisión y eficiencia en la predicción del BCS. Este desarrollo supuso enfrentar diferentes niveles de desafío según el componente en cuestión. En el caso del servidor, mi experiencia de más de 10 años como desarrollador backend fue determinante para diseñar e implementar este componente de manera ágil, utilizando tecnologías como Java, Spring Boot, JMS Template y STOMP. Además, esta experiencia facilitó la comprensión del dominio y los requerimientos funcionales, permitiendo una interacción efectiva con el modelo de negocio y sus particularidades.

Por otro lado, el desarrollo de la aplicación móvil representó un desafío mayor, debido a mi falta de experiencia con Flutter, Dart y el desarrollo de interfaces visuales. Esta situación implicó una curva de aprendizaje significativa, que demandó más tiempo y esfuerzo en comparación con el desarrollo del servidor. Sin embargo, esta experiencia se convirtió en una oportunidad valiosa para ampliar mis habilidades en áreas complementarias y explorar nuevos enfoques en el desarrollo de software.

En total, el proyecto involucró el desarrollo de 113 clases java y 4748 lineas de código en el servidor y 52 archivos dart y 37 clases java y 6006 lineas de código en la aplicación móvil, alcanzando un total de 10754 líneas de código, reflejando su complejidad técnica y su alcance.

La modularidad y escalabilidad del sistema abren puertas para futuras ampliaciones, como la integración con dispositivos IoT, la optimización de modelos predictivos según distintas razas bovinas y el desarrollo de estrategias avanzadas de tolerancia a fallos. En conjunto, este proyecto no solo cumplió con los objetivos iniciales, sino que también sienta las bases para nuevas innovaciones en la automatización y gestión del sector ganadero. Además, representa un avance significativo tanto en términos técnicos como en el aprendizaje y adaptación a tecnologías y dominios desconocidos.

1.1.1 Rendimientos en dispositivos móviles

Para evaluar la eficiencia del sistema, se midieron los tiempos de predicción del BCS en dispositivos móviles, un aspecto crítico para asegurar una experiencia fluida al obtener los resultados de la predicción. Antes de la implementación del prototipo, el algoritmo fue probado mediante una versión preliminar diseñada para Android. Estas pruebas se llevaron a cabo en diversos dispositivos móviles, variando tanto la cantidad de dispositivos como el tamaño de los lotes de procesamiento ("jobs").

En la versión preliminar, los datos se transmitían de forma continua desde un servidor a los dispositivos móviles usando round-robin, operando dentro de la misma red local. En el prototipo

Job por imágenes	versión	Dispositivo	tiempo de tranferencia por imagen (ms)
2	Preliminar	Note 7	697 ± 610
		Moto g9 play	746 ± 447
2	Preliminar	Note 7	983 ± 910
		Moto g6 play	807 ± 265
		Moto g9 play	1065 ± 3092
15	Preliminar	Note 7	57 ± 13
		Moto g6 play	58 ± 11
		Moto g9 play	64 ± 14
1	Prototipo	g9 plus	34 ± 92
		moto g32	41 ± 99

Table 1.1: Transferencia de datos

final, el flujo de datos comenzó con una máquina que simulaba el sistema de envíos hacia el servidor. Posteriormente, el servidor redistribuía los trabajos a los dispositivos móviles usando round robin, permitiendo evaluar el rendimiento en condiciones más representativas del entorno real en una red local.

En 1.1 se puede ver una comparación de la versión preliminar y el prototipo con respecto a la transferencia de datos. La tabla muestra que, a medida que aumenta la cantidad de imágenes por job, el tiempo de transferencia de datos disminuye en la versión preliminar. Esto se debe a que el cliente obtenía cada imagen a procesar mediante una llamada HTTP, lo que generaba mayores tiempos de transferencia. Sin embargo, al cambiar el mecanismo de transferencia a una conexión WebSocket, se logra una reducción significativa en el tiempo de transferencia. Además, la menor variabilidad en la versión del prototipo sugiere una gestión más eficiente de la transferencia de imágenes.

En 1.2 y 1.1se puede ver un análisis comparativo entre las diferentes pruebas en términos del tiempo de detección por imágen. En la comparación se puede ver que Note 7 y el Moto G9 Play obtuvieron un mejor tiempo promedio de detección de imágenes y una menor variabilidad en la versión preliminar, en comparación con la versión prototipo, que incorpora dispositivos más modernos. Esta diferencia se debe a que, en la versión preliminar, el dispositivo se centraba exclusivamente en la ejecución del algoritmo, mientras que en el prototipo se realizaron tareas adicionales, como la visualización de todas las imágenes procesadas (no solo las utilizadas para la predicción) y el envío de los resultados al servidor. A pesar de estas diferencias en la carga de procesamiento, es un aspecto clave a evaluar con el objetivo de optimizar tanto el tiempo de detección como la desviación estándar en futuras iteraciones.

A continuación, se presenta la comparación del tiempo total de procesamiento.

Se observa que la versión preliminar logra un procesamiento más rápido a medida que los jobs contienen un mayor número de imágenes, lo cual se debe a que, como se analizó anteriormente, la transferencia de datos representaba el mayor costo en términos de tiempo. En la versión prototipo, este aspecto se optimizó mediante el uso de una conexión WebSocket, reduciendo significativamente los tiempos de transferencia. Sin embargo, dado que cada job en el prototipo contiene una única imagen y es necesario visualizarla en el dispositivo del observador, el tiempo de procesamiento por imagen es mayor.

A lo largo de las pruebas, se redujo progresivamente el tiempo de espera entre ráfagas, como se observa en la prueba 3 del prototipo, sin afectar la eficiencia del sistema. En futuras evaluaciones, sería recomendable continuar disminuyendo este tiempo de espera y mejorar el prototipo para permitir el procesamiento de múltiples imágenes por job. Asimismo, sería útil incorporar la opción de deshabilitar la visualización de las imágenes, mostrando únicamente los resultados de la predicción, con el objetivo de realizar comparaciones más precisas y optimizar el rendimiento.

Imágenes por job	Versión	Stream	Cluster de smartphones marca-modelo	Tiempo de detección por imágen (ms)
Stream de dos imágenes por job	preliminar	10 segundos entre ráfagas de 12 job	Note7	235 ± 24
			moto g9 play	243 ±15
Stream de dos imágenes por job	preliminar	10 segundos entre ráfagas de 12 job	Note 7	233 ± 23
			moto g6	300 ± 54
			moto g9 play	244 ± 30
Stream de 15 imágenes por job	preliminar	10 segundos entre ráfagas de 12 job	Note 7	217 ± 3
			moto g6	355 ± 14
			moto g9 play	215 ± 13
Stream de 1 imágen por job	prototipo	10 segundos entre ráfagas de 12 job	moto g9 plus	336 ± 161
			Motorola moto g32	255 ± 78
Stream de 1 imagen por job	prototipo	10 segundos entre ráfagas de 12 job	moto g9 plus	249 ± 48
			moto g32	298 ± 47
Stream de 1 imagen por job	prototipo	5 segundos entre ráfagas de 12 job	Motorala moto g9 plus	241 ± 35
			Motorola moto g32	256 ± 73

Table 1.2: Tiempos de ejecución móvil

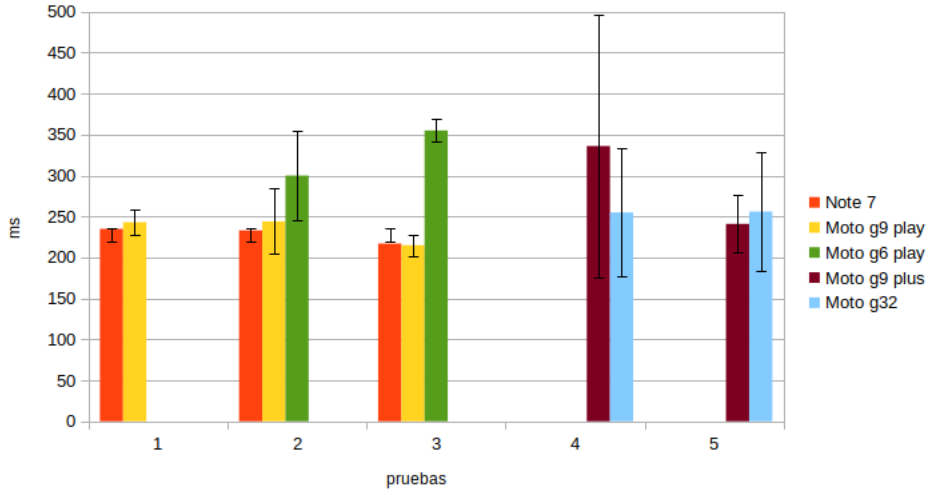

Figure 1.1: Tiempo de detección

Cantidad de dispositivos	Imágenes por job	Versión	Imágenes procesadas	Job Recibidos	Tiempo total
2	2	Preliminar	1752	876	26:43
3	2	Preliminar	1752	876	26:44
3	3	Preliminar	1740	116	3:47
2	1	Prototipo	876	876	876
2	1	Prototipo	432	432	432
2	2	Prototipo	432	432	432

Table 1.3: Tiempo de procesamiento total

Adicionalmente, las pruebas realizadas con el prototipo permitieron evaluar el consumo de memoria en los dispositivos utilizados. Los resultados obtenidos fueron los siguientes:

- En la primera prueba, el Moto G32 alcanzó un consumo máximo de 663.2 MB, mientras que el G9 Plus registró un pico de 445.49 MB.
- En la segunda prueba, el consumo máximo del Moto G32 fue de 558.29 MB, mientras que el G9 Plus redujo su uso de memoria a 425.64 MB.
-
-
-
-

1.1.2 Rendimientos en el servidor

Como se mencionó anteriormente el sistema fue probado en un entorno con dos observadores, sin calificadores para verificar el consumo de recursos durante la ejecución del algoritmo de predicción, y demostró los siguientes consumos de memoria:

- ActiveMQ: fue incrementando desde 277 MB de memoria hasta 490 MB de memoria.
- MySQL: fue incrementando desde 395 MB de memoria hasta 403 MB de memoria.
- Servidor (API principal): no superaron los 500 MB de memoria las pruebas realizadas.

1.2 Limitaciones

1.2.1 Seguridad

El sistema de inicio de sesión presenta una limitación en términos de seguridad. Actualmente, este componente únicamente permite identificar el tipo de usuario y asociar calificaciones a un experto específico, pero no incluye mecanismos de autenticación robustos ni validaciones avanzadas para garantizar la protección del acceso.

Asimismo, no se han implementado restricciones para controlar quiénes pueden registrarse como calificadores. Esto permite que cualquier usuario pueda asumir este rol, lo que podría comprometer la calidad de las evaluaciones realizadas. La falta de validación previa de las credenciales o experiencia de los calificadores representa una limitación para la confiabilidad del sistema en entornos reales.

1.2.2 Tolerencia a fallos

El sistema actual presenta limitaciones en su capacidad de tolerancia a fallos. Ante interrupciones en la conexión WebSocket o errores relacionados, no existen mecanismos implementados para realizar reintentos automáticos o redirigir los trabajos ("jobs"), lo que puede resultar en predicciones no realizadas por el modelo.

Asimismo, el proceso de calificación carece de estrategias que manejen fallos en llamadas REST, lo que puede ocasionar la pérdida de evaluaciones realizadas por los expertos y comprometer la integridad de los datos almacenados.

1.2.3 Mejora en la Gestión de Jobs para Calificadores

Una limitación en la implementación actual es que los calificadores no pueden procesar los trabajos ("jobs") generados antes de su ingreso a una sesión de calificación. Esto significa que los trabajos existentes antes del inicio de la sesión quedan sin calificar, lo que afecta la completitud del proceso de calificación.

Además, no se contempla la carga automática de trabajos previos desde la base de datos hacia la cola JMS del usuario al momento de iniciar sesión, lo que limita la capacidad de los calificadores para abordar tareas pendientes acumuladas antes de su participación activa en la sesión.

1.2.4 Observadores y Algoritmo de Predicción

En la implementación actual, la predicción del puntaje de condición corporal (BCS) está limitada a ser ejecutada exclusivamente en la aplicación móvil por usuarios con logueados con el rol de observador. Esto implica que, en ausencia de observadores, el sistema no genera predicciones. En etapas de prueba preliminares del prototipo en entornos reales en los que el apartamiento de animales según condición corporal se realice de forma manual, sin mecanización automática, es decir, por humanos, esto puede no verse como una limitación ya que se puede asumir que el sistema contará con el poder de cómputo de los dispositivos de quienes realizan dicha labor. Sin embargo, en un entorno que cuenta con apartamiento automático, es deseable que el sistema cuente con una forma alternativa de generar la predición. Para esto, se prevee que en el futuro el componente servidor actue como un observador por defecto.

1.3 Trabajos futuros

1.3.1 Múltiples calificadores

En futuras versiones del servidor, se planea implementar mejoras que permitan a los calificadores compartir una única cola JMS, facilitando la distribución equitativa de las tareas de calificación. Esto incluiría la opción de dividir un grupo de vacas (rodeo) entre calificadores o mantener la funcionalidad actual, donde todos los calificadores evalúan todas las vacas. Además, esta mejora permitiría que los calificadores puedan abandonar una sesión y unirse a otra según sea necesario, ofreciendo mayor flexibilidad en la gestión de las sesiones.

Actualmente, aunque la aplicación móvil no permite que dos calificadores trabajen simultáneamente, el servidor ha sido diseñado para admitir múltiples calificadores en una misma sesión. Cada calificador dispone de su propia cola JMS y tiene la capacidad de calificar todas las vacas, lo que sienta las bases para la implementación de las mejoras mencionadas.

1.3.2 Sistema generador de Jobs

En las pruebas de rendimiento realizadas, fue necesario la creación de dicho componente para simular un escenario de ejecución. Sin embargo, la implementación real de un sistema de generación de jobs quedó fuera del alcance de este proyecto.Este sistema debiera configurarse para notificar al servidor sobre la existencia de una secuencia de imágenes capturadas por una cámara y asociadas

unívocamente a una vaca, lo que da lugar a la creación de un "job" que deberá ser tratado por el componente servidor desarrollado en este proyecto. El generador de jobs deberá permitir, además, configurar la cantidad de imágenes por vaca a ser capturadas, teniendo en cuenta calidad de las mismas, por ejemplo, mediante la aplicación de filtros para determinar borrosidad de la imágen, presencia total o parcial de la parte del cuerpo que se emplea en el cálculo del BCS, etc..

1.3.3 Procesamiento de imágenes

Actualmente, el sistema está diseñado para recibir y almacenar múltiples imágenes asociadas a cada vaca; sin embargo, sólo la primera imagen es enviada a los usuarios calificadores y observadores. Una posible mejora futura sería habilitar la capacidad de enviar todas las imágenes relacionadas con cada vaca, permitiendo un análisis más exhaustivo por parte de los calificadores y observadores. Además, esto abriría la posibilidad de optimizar el algoritmo de predicción para procesar múltiples imágenes dentro de un mismo job, lo que podría mejorar la precisión de los resultados generados.

1.3.4 Integración con IoT

El sistema tiene el potencial de integrarse en un entorno IoT, permitiendo la conexión con dispositivos y elementos del entorno ganadero, como mangas, corrales y comederos. Esto requerirá desarrollar una capa de comunicación que facilite la captura automatizada de datos y acciones de contingencia, estructurando el sistema como un Sistema Integral de Gestión de la Información (SIGI) para apoyar el monitoreo, la planificación y la toma de decisiones.

1.3.5 Múltiples modelos

La aplicación móvil está diseñada con una arquitectura flexible que puede permtir la integración de diferentes modelos de predicción. A través de una configuración sencilla, se posible seleccionar y definir qué modelo utilizar en cada caso específico, adaptándose así a diversas necesidades y escenarios.

1.3.6 Multiples plataformas

Al desarrollar la aplicación móvil en Flutter, la portabilidad a iOS se facilita significativamente debido a la naturaleza multiplataforma del framework. Sin embargo, para lograr esta transición, es necesario analizar y ajustar el modelo predictivo, ya que actualmente está diseñado y probado específicamente dispositivos Android.

1.3.7 Configuración de la participación de observadores en el cálculo de BCS

En futuras implementaciones, se planea incorporar a la aplicación móvil un interruptor (switch) junto con un campo de configuración que permita a un usuario observador decidir si aportar o no recursos de cómputo para el cálculo del BCS, o incluso que dicha opción pueda ser habilitada dependiendo del nivel de batería con el que cuenta el dispositivo. Este mecanismo permitirá al dispositivo móvil notificar al servidor si puede ofrecer recursos de procesamiento para el cálculo del BCS como así también fijar cuotas de uso de batería.

Al activar este interruptor, se deberá iniciar un servicio en segundo plano encargado de gestionar un protocolo de intercambio de mensajes entre la aplicación y el servidor, asegurando una coordinación eficiente para el uso compartido de recursos. Aunque esta funcionalidad estaba contemplada desde el inicio, no se logró implementar debido a las dificultades encontradas durante el desarrollo de la aplicación móvil.