UNMdP - Facultad de Ingeniería

Departamento de Informática

Sistema para la medición y monitoreo de pasturas: Pastech

Subsistema Cloud-Web

Alumnos

Demoor, Tobías <tobias.demoor@gmail.com>

Lening Celaya, Leopoldo <leopoldoleningcelaya@gmail.com>

Director

Hinojal, Hernán

Co-Director

Genin, Fernando

Proyecto final para optar por el grado de Ingeniero en Informática

Mar del Plata, 11 de octubre del 2023

UNMdP - Facultad de Ingeniería

Departamento de Informática

Sistema para la medición y monitoreo de pasturas: Pastech

Subsistema Cloud-Web

Alumnos

Demoor, Tobías <tobias.demoor@gmail.com>

Lening Celaya, Leopoldo <leopoldoleningcelaya@gmail.com>

Director

Hinojal, Hernán

Co-Director

Genin, Fernando

Proyecto final para optar por el grado de Ingeniero en Informática

Mar del Plata, 11 de octubre del 2023

Índice

Agradecimientos

A nuestros familiares y amigos. En especial, a nuestros padres y hermanos, por el apoyo y el afecto brindados.

A los compañeros que conocimos a lo largo de la carrera por la colaboración y los momentos compartidos.

Al cuerpo docente de la Facultad de Ingeniería de la Universidad Nacional de Mar del Plata por la formación y el apoyo que nos han brindado a lo largo de la carrera. En especial, a Hernán Hinojal y Fernando Genin, directores de este proyecto, por su buena predisposición al momento de solicitar su participación y ante cualquier inquietud o consulta.

A los investigadores de la Facultad de Ciencias Agrarias, del INTA y del CONICET por darnos la oportunidad de participar en este proyecto. En especial, a Juan Insúa y Alejandra Marino, quienes nos presentaron la problemática y su propuesta.

Resumen

El proyecto Pastech fue ideado por investigadores de la facultad de Ciencias Agrarias, del INTA y del CONICET. Tiene como objetivo desarrollar una solución que permita realizar la medición y monitoreo de las pasturas de los establecimientos ganaderos de una manera rápida, precisa y automatizada haciendo uso de las tecnologías de la información. Se espera obtener un producto mínimo viable útil que tenga la capacidad de ser extendido y escalado fácilmente y cuyo precio sea accesible para ser lanzado al mercado.

El sistema desarrollado está compuesto por: un pasturómetro electrónico que permitirá a los operarios del campo tomar las mediciones de la altura del pasto; una aplicación mobile que se conectará al pasturómetro para poder capturar las mediciones y enviarlas a la nube; un sistema cloud que se encargará del almacenamiento y procesamiento de los datos y, por último, una aplicación web para realizar la gestión y visualización de la información.

Dada la magnitud del sistema, se tomó la decisión de dividir el desarrollo en dos subproyectos. Por un lado, el subsistema *mobile* y su interacción con el pasturómetro fue desarrollado por el grupo integrado por Matías Veitch y Nicolás Escudé, mientras que el pasturómetro fue diseñado y construido por integrantes del departamento de Ingeniería Electrónica de la Facultad de Ingeniería de la UNMdP. Por otro lado, en este trabajo final se llevó a cabo el desarrollo de los servicios *cloud* y la aplicación *web*.

El diseño del subsistema *cloud-web* fue guiado principalmente por los requerimientos de escalabilidad y extensibilidad mencionados anteriormente, por lo que se implementó una arquitectura conformada por microservicios con responsabilidades claramente definidas. Además, se desarrolló una aplicación *web* con un diseño moderno e intuitivo para poder acceder, gestionar y visualizar la información mediante imágenes satelitales, representaciones gráficas y georeferenciadas de los datos, cálculos y análisis estadísticos, entre otros.

El desarrollo de este trabajo final permitió enfrentar una problemática real y atravesar las etapas necesarias de planificación y gestión de un proyecto de *software* con el fin brindar una solución acorde a las necesidades del dominio. Esto permitió incorporar experiencias y conocimientos que son de suma importancia en la vida profesional de un Ingeniero en Informática.

Introducción

La eficacia de la producción de los establecimientos ganaderos de base pastoril depende en gran medida de la calidad y la cantidad de pasto, debido a que es la materia prima a partir de la que se genera el producto animal (carne, leche, lana, entre otros). Por lo tanto, el hecho de conocer la cantidad de pasto y su distribución geográfica resulta fundamental para llevar a cabo una planificación eficiente del pastoreo.

En el presente informe, se tiene como objetivo partir de un análisis de la situación actual en Argentina respecto a la problemática mencionada anteriormente y planificar el diseño y desarrollo de un sistema *cloud* y un cliente *web* que formará parte de una solución integral para la medición de pasto en los establecimientos ganaderos. Se espera que esta herramienta permita estimar de forma rápida y precisa la variación espacio temporal de pasto para poder realizar un pastoreo planificado. Con esta información, se podrán aplicar los suplementos necesarios en la medida justa de manera que se genere el menor impacto ambiental posible. A su vez, el sistema desarrollado deberá tener la capacidad de ser extensible para que, en un futuro, se puedan agregar múltiples funcionalidades nuevas sin mayores inconvenientes.

Por otro lado, se tiene como finalidad conocer y enfrentar los inconvenientes que aparecen al momento de planificar y gestionar un proyecto de desarrollo de *software* en cuanto a la estimación de tiempos, la toma de decisiones, la validación del sistema con los usuarios, etc. Cabe destacar que la solución presentada en este informe formará parte de un sistema más grande, en el que intervienen otros proyectos: el desarrollo de una aplicación *mobile* y el diseño de un instrumento de medición electrónico que trabajarán en conjunto para tomar los valores de la altura del pasto y enviarlos al sistema *cloud*. Se espera tener una interacción constante con los demás equipos de trabajo y poner en práctica las competencias profesionales requeridas para lograr una comunicación eficaz y poder diseñar las interfaces y protocolos a utilizar entre los diferentes subsistemas.

Para llevar a cabo el proyecto, se cuenta con la ayuda de los referentes funcionales, quienes son expertos en materia agronómica y conocen en profundidad la problemática y la forma de trabajar en los establecimientos ganaderos. Por lo tanto, se podrán validar los requerimientos y aspectos técnicos del sistema

constantemente, con el fin de que la solución desarrollada se ajuste de la mejor manera posible a la realidad de los trabajadores de la actividad ganadera.

Análisis del problema

Dominio del problema

La producción ganadera es una actividad sumamente importante para la economía argentina, dado que representa alrededor del 30% del valor bruto de producción de las cadenas agroalimentarias en Argentina[1]. La rentabilidad de los establecimientos ganaderos de base pastoril está definida en gran medida por la cantidad de pasto que se convierte en producto animal (carne, leche o lana). Por lo tanto, el éxito en esta actividad depende no solo de producir más pasto, sino también de utilizarlo de manera conveniente a través de un eficiente manejo del pastoreo[2].

Para poder gestionar las pasturas de manera eficiente, es crucial contar con el conocimiento preciso y actualizado de la cantidad de pasto que hay en cada potrero del campo. Sin embargo, en la práctica a nivel local, las mediciones frecuentes (es decir, recorridas semanales de los potreros) generalmente no se realizan debido a que las técnicas 'tradicionales' para estimar la cantidad de pasto (por ejemplo, la medición de la altura y el posterior corte, secado y pesaje) requieren mucho tiempo y trabajo. Esta problemática se ve agravada al considerar la generalizada escasez de mano de obra en los establecimientos ganaderos. Además, estas dificultades se amplifican cuanto más grande es el área a medir, como ocurre en los sistemas extensivos de nuestro país.

La estimación rápida y precisa de la cantidad de pasto disponible es uno de los grandes desafíos que enfrenta la producción ganadera argentina para poder garantizar una adecuada soberanía alimentaria.

Problema a resolver

Actualmente en Argentina, las técnicas tradicionales que se utilizan para realizar el seguimiento de la cantidad y tasa de crecimiento del pasto requieren mucho tiempo y esfuerzo manual (corte, secado y pesado, por ejemplo). Es por ello que, en general, estas tareas de medición no son realizadas en absoluto. Esto afecta principalmente a la rentabilidad de los establecimientos ganaderos de nuestro país, cuya actividad económica representa una gran parte del producto bruto agropecuario, dado que una nula o mala administración del pasto se traduce en una menor cantidad de producto animal.

Por otro lado, al no tener la información necesaria para la gestión de las pasturas, los suplementos para el pasto no son utilizados de manera eficiente, y generalmente se tiende a malgastarlos. Esto no solo tiene consecuencias económicas por el uso excesivo de los productos, sino que también genera un fuerte impacto ambiental negativo.

Por lo tanto, resulta necesario el desarrollo de herramientas que permitan estimar de manera rápida y precisa la cantidad de pasto y su variación espacio temporal en cada establecimiento ganadero. De esta manera, se podrá consumir el pasto de manera planificada y asignar la cantidad justa y necesaria de suplementos que se requiere para una producción eficiente de alimentos con el menor impacto ambiental posible.

Objetivos del proyecto

Objetivo principal

El objetivo principal del proyecto consiste en desarrollar una solución integral para la medición y monitoreo de las pasturas en los establecimientos ganaderos. Se espera que esta solución permita realizar una estimación objetiva, rápida y precisa de la variación espacio-temporal del pasto en cada uno de los terrenos. En particular, en este proyecto se desarrollará el subsistema *cloud-web*, que deberá tener un diseño moderno y preparado para adaptarse a la incorporación de nuevas funcionalidades y servicios para poder ser extendido y escalado en el futuro. A su vez, el producto debe ser accesible tanto para el mercado local como para el mercado extranjero.

Se busca que esta solución incentive a realizar las tareas de control y seguimiento de las pasturas a aquellos productores que actualmente no las llevan a cabo debido al tiempo y costo que implican las técnicas tradicionales. Además, a partir de la visualización y análisis de los datos almacenados en el sistema, se espera que los usuarios puedan asignar de forma exacta la cantidad de pasto y los suplementos que requieran los animales para lograr una producción de alimentos más eficiente y que genere el menor impacto ambiental posible.

Por otro lado, para generar el impacto esperado, es fundamental lanzar un producto que cumpla con los requerimientos establecidos en el menor tiempo posible, ya que el tiempo de llegada al mercado es sumamente importante para que la solución sea exitosa en materia comercial.

Alcance

Pastech tiene como objetivo desarrollar una solución que permita llevar a cabo el control de las pasturas en los establecimientos ganaderos mediante el uso de las tecnologías de la información. La funcionalidad esperada abarca: la toma de mediciones georeferenciadas de la altura del pasto; la definición geográfica de los potreros del campo y otros datos como las especies, fechas de pastoreo, rodeo, etc.; la definición de curvas de calibración que transformen la medición de la altura en cantidad de pasto; el almacenamiento de los datos y la generación de informes

estadísticos básicos que indiquen las métricas necesarias para apoyar la toma de decisiones de los usuarios.

En particular, el subsistema *cloud-web* debe proveer un servicio capaz de almacenar los datos, procesar y generar los informes estadísticos, gestionar y autentificar usuarios y brindar una interfaz de comunicación que permita el correcto funcionamiento del resto de aplicaciones del sistema. Además, se debe proveer una aplicación *web* que permita visualizar los informes y gráficos estadísticos, definir los potreros en el campo, visualizar las mediciones en un mapa, definir las curvas de calibración y las especies de pasto y cargar datos tales como el rodeo y las fechas de pastoreo. A su vez, debe permitir la administración de los clientes que contraten el producto.

El proyecto no abarca la incorporación de medidas de seguridad en la infraestructura de los microservicios. Se hará uso de buenas prácticas de seguridad informática y se validará que cada consulta incluya las credenciales (token de acceso) en su cabecera, pero no se autenticará la identidad de cada microservicio en las consultas internas. Esto solo podría suponer un problema en el caso de que los microservicios no se ejecuten en la misma red local y, por lo tanto, se deba exponer sus interfaces a internet de manera individual para permitir que se comuniquen entre sí.

Tampoco estarán dentro del alcance de este trabajo el desarrollo de los requerimientos deseados para una versión futura del sistema, la realización de un *testing* integral e intensivo en conjunto con el equipo responsable del desarrollo de la aplicación *mobile*, la implementación de pruebas automatizadas, la definición de políticas y mecanismos de *backup* de la información ni la puesta en producción con el posterior lanzamiento al mercado del producto.

Entregables del proyecto

A continuación, se listan los entregables resultantes de este proyecto:

- Lista de requerimientos funcionales y no funcionales.
- Lista de requerimientos para una versión futura del sistema.
- Documentación:
    - Diagrama de arquitectura del sistema.
    - Protocolo de errores de los microservicios. (Apéndice B)
    - Interfaces de las API de los microservicios. (Apéndice C)

    - Diagrama de entidades. (Apéndice D)
    - Manual de *deployment*.
    - Manual de administración del sistema.
- Código fuente del sistema:
    - *UserService.* Microservicio encargado de la gestión de usuarios y grupos de usuarios.
    - *PaddockService.* Microservicio encargado del almacenamiento de datos de potreros, calibraciones y mediciones.
    - *PastureBalanceService.* Microservicio encargado de la generación de reportes estadísticos.
    - *BFFMobileApp.* Servidor que actúa como interfaz entre la aplicación *mobile* y los microservicios.
    - *BFFWebApp.* Servidor que actúa como interfaz entre la aplicación *web* y los microservicios.
    - Aplicación *web* para administración de usuarios, potreros, mediciones, calibraciones, especies y visualización de reportes y datos en general.

División del proyecto

Pastech es una solución integral de gran alcance, por lo que se tomó la decisión de dividirla en dos subproyectos: uno que se abarque el desarrollo del dispositivo electrónico para tomar las mediciones y la aplicación *mobile* con la que se comunicará y otro que se enfoque en el desarrollo de los servicios *cloud* y la aplicación *web*.

Una de las primeras tareas a realizar fue la definición de los requerimientos funcionales y el alcance de cada subsistema. Además, se definió la arquitectura general de la solución y la comunicación necesaria entre los componentes. Esto fue llevado a cabo mediante una serie de reuniones con los referentes funcionales y los encargados del desarrollo del pasturómetro y la aplicación *mobile*. Por otro lado, antes de iniciar el diseño, se realizaron reuniones con los miembros del grupo encargado del desarrollo de la aplicación *mobile* con el fin de definir las interfaces en común y las responsabilidades de cada subsistema.

El hecho de dividir el proyecto y trabajar colaborativamente con otro grupo para el desarrollo de la solución permitió ampliar su alcance y brindar producto final con una mayor funcionalidad en un menor tiempo. En el caso de haber abordado la totalidad del desarrollo bajo un solo equipo de trabajo, la extensión del proyecto hubiera sido demasiado prolongada para los tiempos requeridos.

Sin embargo, esta decisión implica un incremento en la complejidad del desarrollo, debido a que se debe dedicar tiempo para llevar a cabo las definiciones comunes a ambos subsistemas y coordinar la toma de decisiones que afecten a la solución general. A pesar de esto, se consideró que la división del proyecto era una decisión acertada en relación a los objetivos de Pastech.

Aporte del proyecto

Beneficiarios

El proyecto Pastech fue ideado y presentado al departamento de informática de la Facultad de Ingeniería por investigadores de la Facultad de Ciencias Agrarias, el INTA y el CONICET. Este grupo de investigadores se encarga de acercarse a los establecimientos agrónomos locales y analizar sus necesidades con el fin de encontrar puntos de mejora e innovación en los que se pueda hacer uso de las tecnologías de la información.

En este caso en particular, los investigadores observaron la oportunidad de introducir al mercado una solución para la medición y control de las pasturas de los establecimientos ganaderos que resulte accesible para el productor local, dado que no existe un competidor fuerte y gran parte de los productores no hace uso de ninguna herramienta similar.

Impacto

La solución a desarrollar brindará un medio para realizar la toma de mediciones de las pasturas de una manera precisa empleando el menor tiempo posible. Se pueden identificar tres puntos principales en los que se espera generar un impacto considerable: sobre el productor local, sobre la economía y sobre el ambiente.

El productor local tendrá acceso a una solución accesible y que se ajuste a sus necesidades. Si bien fuera del país existen algunas opciones similares, las condiciones bajo las que deben utilizarse no siempre se adecúan correctamente a los establecimientos locales y el precio resulta demasiado elevado, por lo que el uso es prácticamente nulo.

Por otro lado, se espera generar un impacto económico al mejorar la productividad de los establecimientos ganaderos locales. El producto permitirá ver los datos de manera tal que se facilite la toma de decisiones estratégicas para planificar el pastoreo a futuro. De esta manera, se espera hacer un uso más eficiente tanto de la materia prima como de los suplementos que se aplican para que el pasto cumpla con las condiciones necesarias para el consumo animal.

A su vez, el buen manejo de pasturas y pastizales aporta servicios ecosistémicos al medio ambiente, ya que regula el ciclo del agua, aumenta la captura de las partículas de carbono y reduce la emisión de gases efecto invernadero. Además, se espera reducir el impacto ambiental generado por el uso de suplementos para el pasto al brindarle al usuario la información requerida para administrarlos en la cantidad justa y necesaria.

Análisis FODA

A continuación se presentan los resultados del análisis FODA realizado por el equipo al inicio del proyecto y que fue utilizado como guía para la toma de decisiones a lo largo del desarrollo.

Fortalezas:

- Los integrantes poseen experiencia en el desarrollo de sistemas *web*.
- Se cuenta con la asistencia técnica de los referentes funcionales, quienes son expertos en el sector agrónomo. A su vez, los referentes cuentan con experiencia práctica en relación a las técnicas tradicionales históricamente utilizadas para el control de las pasturas, por lo que se podrá obtener información de alta calidad para validar el producto.
- El proyecto brindará una solución que puede ser aplicada y comercializada de manera general a todos los productores ganaderos a nivel nacional e internacional.

Oportunidades:

- Si bien existen soluciones similares en países extranjeros, ninguna de ellas se adecúa al poder adquisitivo y las características de los establecimientos locales. Por lo tanto, existe un mercado de productores agrónomos muy extenso que puede demandar el producto a desarrollar.

Debilidades:

- Los integrantes poseen un conocimiento muy escaso acerca del dominio de la problemática.
- El presupuesto para la puesta en producción del sistema es bajo.

- Pueden existir retrasos en el cronograma debido a dificultades en la comprensión y validación de conceptos relacionados a tecnicismos del dominio del problema.

Amenazas:

- El producto puede tener una baja adopción en el mercado por parte de los productores agrónomos. Esto se debe a que, en el ámbito local y regional, generalmente no se realiza ningún tipo de tarea relacionada al seguimiento y administración de las pasturas, ni siquiera haciendo uso de las técnicas tradicionales. Por lo tanto, es posible que los productores no logren ver el valor agregado de la solución desarrollada en este proyecto y se nieguen a destinar recursos a una tarea que nunca realizaron.
- La apertura de importaciones puede generar una disminución del costo de soluciones preexistentes.

Análisis de riesgos

Además de los riesgos contemplados en el análisis FODA, se han analizado otros factores que pudieran hacer inviable el proyecto o afectar su desarrollo en gran medida previo a su finalización. Para cada uno de ellos, se ha ponderado su probabilidad de ocurrencia (P) y su impacto sobre el proyecto (Imp) en una escala de valores del 1 al 3, siendo éste último el de mayor valor. Ambos valores son utilizados para calcular el peso del riesgo como el producto entre la probabilidad de ocurrencia y el impacto.

Finalmente, si el peso calculado para un riesgo potencial resulta mayor o igual a 6, se debe elaborar un plan de contingencia, ya que se considera que su impacto y probabilidad son elevados y requieren tomar acciones al respecto.

Riesgo	Descripción	Consecuencia	P	Imp	Peso
R01	Apertura de importaciones	Las soluciones competidoras reducen sus costos y el producto no tiene el impacto esperado en el mercado	1	3	3

R02	El proyecto es absorbido por una empresa tecnológica	Imposibilidad de continuar con el proyecto del trabajo final	2	3	6
R03	Un integrante abandona el proyecto	Pérdida de mano de obra y experiencia	1	3	3
R04	Imposibilidad de coordinar reuniones con los referentes funcionales	Retrasos en el cronograma por validaciones y cambios inoportunos en los requerimientos	2	3	6
R05	Aparición de errores al integrar los componentes del sistema	Demoras por retrabajo	2	2	4
R06	Ausencia momentánea de uno o más integrantes por razones personales.	Retrasos en el cronograma por pérdida temporal de mano de obra	3	2	6
R07	Retrasos en el desarrollo del subsistema *mobile.*	Imposibilidad de realizar pruebas de integración y retrasos en el cronograma	3	2	6
R08	Abandono de la totalidad del equipo del proyecto paralelo correspondiente al subsistema *mobile*	Imposibilidad de continuar con el desarrollo integral del sistema	2	3	6
R09	Obsolescencia o abandono del mantenimiento de alguna de las tecnologías utilizadas para el desarrollo por parte de sus creadores	Retrasos en el cronograma por migración del sistema a tecnologías nuevas	1	3	3

Figura 1 - Análisis de riesgos

Planes de contingencia desarrollados

- **R02.** Acordar con los referentes funcionales un alcance mínimo para garantizar la obtención de un producto que cumpla con los objetivos del proyecto. En caso de que una empresa absorba el desarrollo, se buscará llegar a un acuerdo para que permitan finalizar la solución esperada en este trabajo final antes de que tomen la dirección del proyecto.
- **R04.** Pactar las reuniones con la suficiente anterioridad. En caso de que aún así se dificulte coincidir para realizarlas, se establecerán comunicaciones por otros medios asíncronos para disminuir el impacto en el cronograma.
- **R06.** Compensar las horas de desarrollo perdidas con horas extras en las semanas previas o posteriores a la ausencia del o los integrantes.
- **R07.** En caso de que el retraso no sea considerable, se buscará reorganizar las tareas correspondientes en el cronograma. En caso contrario, se realizarán pruebas con prototipos simples que simulen el comportamiento de la aplicación *mobile*.
- **R08.** En caso de abandono de la totalidad del equipo del proyecto paralelo, se llevará a cabo el desarrollo haciendo uso de prototipos simples que simulen el comportamiento de la aplicación *mobile*. Por otro lado, se debe encontrar otro grupo de desarrolladores que absorba y continúe con el proyecto *mobile* con el fin de obtener un producto completo y funcional.

Estimación inicial

Para poder realizar las estimaciones sobre los tiempos del proyecto, se trabajó en conjunto con el equipo del subsistema *mobile* y los referentes funcionales para definir, a partir de las funcionalidades deseadas, una lista de requerimientos y una arquitectura que permitiera cumplir con ellos. De esta manera, se logró obtener una visión más acertada del dominio del problema a resolver y del alcance del proyecto. Por otro lado, la definición de la arquitectura del subsistema *cloud* fue de gran importancia para la estimación de tiempos debido a que permitió dividir el desarrollo en la implementación de componentes más pequeños y con responsabilidades bien definidas.

Parte del equipo contaba con experiencia en el desarrollo de servicios *backend*, por lo que se pudo estimar con mayor certeza el tiempo necesario para implementar los componentes correspondientes de la arquitectura. Sin embargo, al carecer de los conocimientos adecuados para estimar el tiempo de desarrollo de la aplicación *web*, la planificación de las tareas relacionadas se realizó de una forma más aproximada. Finalmente, para definir la carga horaria del proyecto, se tuvieron en cuenta las actividades laborales y personales de cada integrante para definir una dedicación semanal que pueda llevarse a cabo en la práctica sin mayores inconvenientes.

La estructura de la estimación inicial del cronograma se dividió en las siguientes etapas: análisis, diseño, implementación, *testing* y cierre. Para la disposición temporal de las etapas, se decidió adoptar una metodología de cascada debido al escaso conocimiento del dominio del problema por parte de los integrantes. De esta manera, se realizaría un análisis y estimación de la complejidad del proyecto antes de comenzar a diseñar cada uno de los componentes del producto final.

Por otro lado, se programó el diseño de las interfaces de comunicación entre el subsistema *cloud* y la aplicación *mobile* al inicio del cronograma con el fin de desacoplar el avance de ambos proyectos en una etapa temprana del desarrollo. De esta manera, cada equipo podría avanzar de forma independiente, por lo que se disminuye considerablemente la probabilidad de que los retrasos en uno de los proyectos afecten el avance del otro.

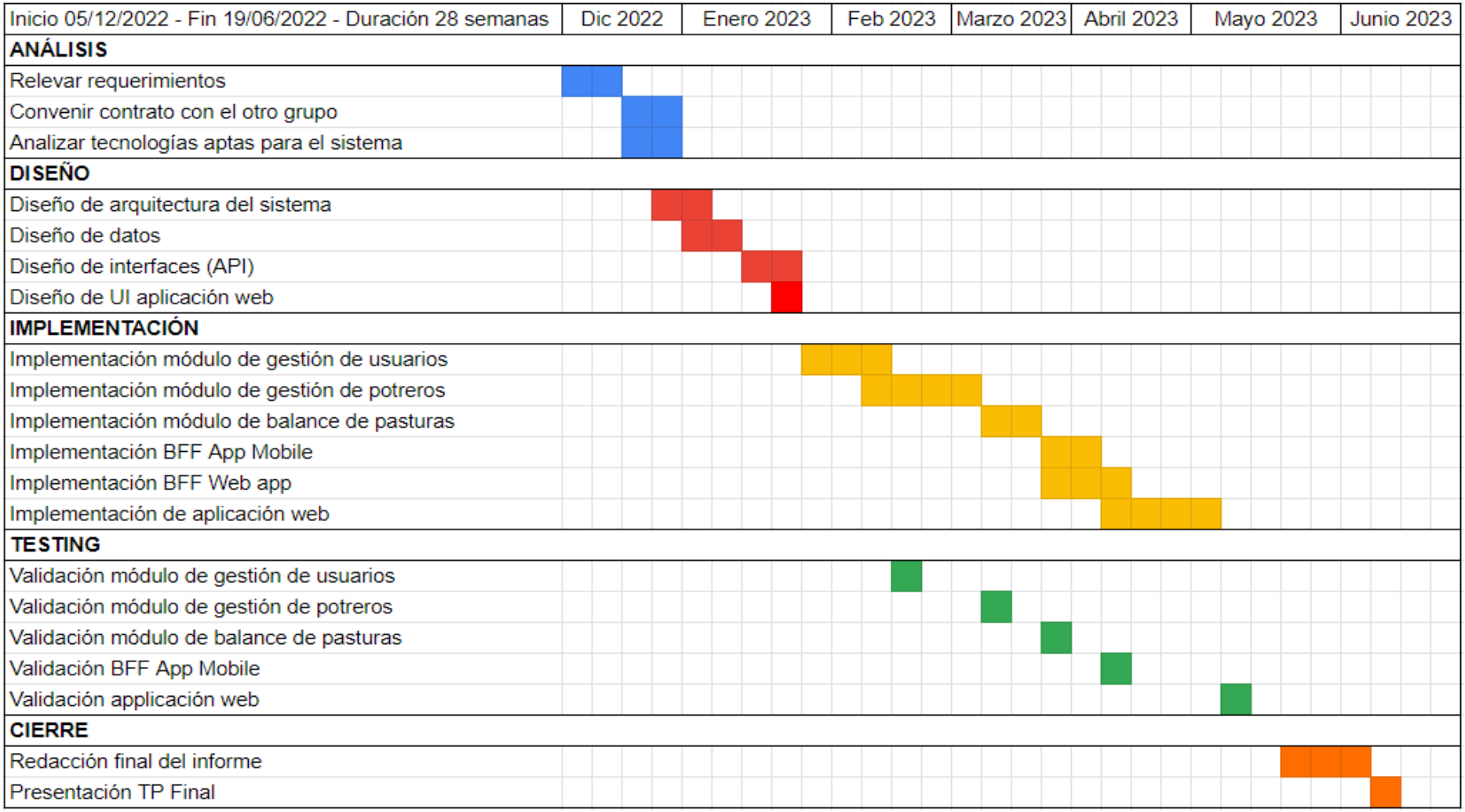

Figura 2 - Diagrama de Gantt presentado con el protocolo

Metodologías

Metodología de trabajo

En las etapas iniciales del proyecto se adoptó una metodología de trabajo en cascada para poder obtener información y conocimiento acerca del dominio del problema antes de comenzar con el diseño e implementación. Por lo tanto, las primeras tareas se llevaron a cabo de forma secuencial y no se avanzaba a la siguiente hasta terminar la que se encontraba en curso. Si bien este esquema tiene sus desventajas frente a los esquemas ágiles debido a que no se mantiene una retroalimentación constante con el cliente, permitió seguir una estructura clara al inicio del proyecto frente a la falta de experiencia del equipo en el ámbito agrónomo. Sin embargo, las etapas referidas al análisis y relevamiento del módulo de balance y estadísticas de pasturas se postergaron para una etapa más avanzada del proyecto. Esto se debió a que, durante el desarrollo, se consideró que era conveniente esperar a tener un mayor grado de madurez de los conocimientos para comprender de una mejor manera los conceptos referidos a dicho módulo. De esta forma, se pudo tener una etapa intermedia en la que no solo se pudieron relevar los requerimientos necesarios, sino que también se pudieron validar los entregables parciales del proyecto y recibir una retroalimentación por parte de los referentes funcionales.

Para el resto de las etapas, se decidió utilizar una metodología iterativa e incremental, con el objetivo de desarrollar el producto de manera progresiva y obtener entregables parciales antes de la fecha final, pero siguiendo el orden del desarrollo planteado en la estimación inicial. De esta manera, a diferencia de la metodología en cascada inicialmente planteada, se podrían realizar modificaciones, rediseños y mejoras continuas al producto a lo largo del desarrollo. Además, este esquema favorece la productividad del equipo, dado que los objetivos a cumplir son a corto plazo y de menor alcance.

En primer lugar, se definió un plazo de una semana para los incrementos, también conocidos como *sprints*. Al inicio de cada *sprint* se realizó una reunión en la que se tomaba un objetivo o requerimiento del listado general definido en el cronograma y se establecían las subtareas necesarias para llevarlo a cabo. Cada tarea consta de una descripción, una fecha límite, un estado (pendiente, vencida o completada) y un

responsable a cargo de realizarla. Por otro lado, en estas reuniones se realizaba un diseño más detallado de los componentes o funcionalidades a desarrollar en ese *sprint*, así como también se definían las tecnologías específicas para utilizar para cumplir con los objetivos.

En segundo lugar, se definieron los puntos de validación del proyecto, es decir, las instancias en las que debía realizar reuniones con los referentes funcionales o con los directores del proyecto para presentar los avances y recibir retroalimentación de su parte. De esta manera, se aseguraba una comunicación fluida y constante entre los distintos actores involucrados que permitiese ajustar y mejorar el proyecto en tiempo de desarrollo. Además, estas reuniones de validación contribuyeron a garantizar que los resultados estuvieran alineados con las expectativas y necesidades de los futuros usuarios.

Metodología de análisis

Al inicio del proyecto se trabajó en conjunto con el grupo encargado del desarrollo de la aplicación *mobile* y los referentes funcionales con el fin de realizar el análisis y relevamiento de los requerimientos de la solución general. Los referentes aportaron su conocimiento acerca del dominio, la problemática, las soluciones existentes, el panorama del mercado en la región, la presencia de posibles competidores, el prototipo con el que contaban y la forma de trabajo con este.

A partir de este análisis inicial, se definieron las responsabilidades de cada subsistema (*cloud-web* y *mobile*) junto a los integrantes del equipo encargado del desarrollo de la aplicación *mobile.* Luego, se realizó un análisis más profundo y se agregaron, eliminaron y modificaron requerimientos hasta llegar a una lista que se ajuste de la mejor manera a la totalidad de la solución esperada. A su vez, se acordó con los referentes funcionales cuáles serían las funcionalidades desarrolladas en el alcance de este trabajo final y cuáles entrarían en una etapa posterior a este proyecto.

Durante el desarrollo, se validaron los requerimientos existentes y se incorporaron nuevos a partir de las reuniones programadas con los directores y los referentes funcionales. En particular, se definieron nuevas funcionalidades en relación al módulo de cálculos y reportes estadísticos, dado que no había podido ser analizado en su totalidad durante la etapa de análisis inicial ya que requería una madurez de los conceptos con la que no se contaba en ese momento.

Metodología de diseño

Tras completar la fase de análisis y tener claros los requisitos, se dio inicio a la etapa de diseño del sistema. Durante este proceso, se construyó un esquema arquitectónico con el fin de satisfacer las necesidades planteadas. El diagrama de arquitectura obtenido fue una pieza clave para avanzar en el proyecto, ya que brindó una visión completa del sistema y permitió separarlo en módulos más pequeños con responsabilidades claramente definidas. Además, se definió cómo se comunicarían cada uno de los componentes entre sí. Durante todo el proceso de diseño, se puso especial énfasis en crear un producto escalable, fácil de mantener y con posibilidad de ampliarse en el futuro sin mayores inconvenientes. El siguiente paso fue definir las entidades principales del sistema, las relaciones entre ellas, el modelo de datos, la información a almacenar en cada uno de los componentes de la arquitectura y las principales tecnologías y herramientas a utilizar para el desarrollo.

Al momento de validar este diseño inicial surgieron algunos cambios relacionados a la información a almacenar y el modelo de datos del sistema. Luego, a lo largo del desarrollo del proyecto, se realizaron una serie de mejoras y modificaciones a este diseño con el fin de incorporar los requerimientos nuevos y las modificaciones a los existentes que fueron surgiendo a partir de las reuniones en las que se presentaron los entregables parciales del producto.

Por otro lado, con el fin de que cada equipo de trabajo pudiera continuar su desarrollo de manera independiente, se diseñaron y se documentaron las interfaces de comunicación entre los subsistemas en una etapa temprana del proyecto. De esta manera, cada grupo pudo continuar el desarrollo de las funcionalidades requeridas de manera independiente siempre que se cumpliera con el contrato definido anteriormente.

Metodología de desarrollo y pruebas

Los integrantes del grupo poseían experiencia en desarrollo e implementación de *software* adquirida tanto en el ámbito laboral como en el universitario. En particular, los integrantes tenían conocimiento en el uso de GitHub[3] para alojar y administrar los repositorios donde se mantendría el código fuente de los componentes a desarrollar y Todoist[4] como herramienta principal para registrar, asignar y controlar las tareas del proyecto.

En cuanto al uso de GitHub, se definió que cada repositorio contaría con dos ramas base: una rama principal llamada *master,* donde se mantendría la versión estable del código, y una rama llamada *dev,* que se utilizaría para desarrollar y realizar las pruebas necesarias. De esta manera, para implementar nuevas funcionalidades, se debe crear una nueva rama a partir de la rama de desarrollo. Una vez terminado el trabajo, se debe crear una solicitud de incorporación de cambios (llamada *pull request)* a la rama *dev*, donde posteriormente se realizarán las tareas de *testing*. Una vez que las funcionalidades desarrolladas fueron testeadas y/o corregidas, se debe enviar un *pull request* desde la rama de desarrollo a la rama principal. Al momento de crear un *pull request*, se debe incluir una breve descripción de los cambios realizados y se debe asignar a una persona para que realice la revisión y apruebe o no el pedido para incluirlos. Esta persona puede incluir anotaciones y solicitar modificaciones en el código antes de aprobar el *pull request*. De esta manera, se mantiene un control y una revisión constante de los nuevos cambios introducidos en la aplicación.

Antes de comenzar las tareas de implementación, se definieron reglas de *linting* personalizadas con el fin de lograr un estilo de código unificado y evitar errores de tipeo y de semántica. Estas reglas definen la longitud máxima de una línea de código, buenas prácticas para la definición de funciones, el formato de indentación, entre otras. Para la implementación y corrección automatizada de estas reglas se utilizó una herramienta de análisis de código estático llamada ESLint[5], debido a su excelente integración con el resto de tecnologías de desarrollo utilizadas. Esto permite obtener un código fuente estructurado y legible, por lo que favorece ampliamente la mantenibilidad del sistema.

Para realizar las tareas de *testing,* se optó por realizar pruebas exploratorias manuales debido a que los integrantes no tenían conocimiento acerca de las herramientas necesarias para realizar pruebas automatizadas, y uno de los objetivos del proyecto es lanzar el producto al mercado lo antes posible. La investigación y aprendizaje de una nueva herramienta y la definición de los escenarios de pruebas necesarios para utilizarla implicaba destinar una cantidad considerable de tiempo que no había sido prevista en el cronograma. Al realizar pruebas exploratorias de *testing* y validación con una colección de datos de prueba, se pudo destinar una mayor cantidad de tiempo para el desarrollo de las funcionalidades necesarias.

Las pruebas realizadas en la aplicación *web* se centraron principalmente en comprobar la consistencia de la información del sistema, las alertas mostradas al cargar, modificar y eliminar datos, el comportamiento de la interfaz en los diferentes tamaños de pantallas de escritorio donde la aplicación será utilizada y la autentificación de los roles y sesiones del usuario. Además, se realizaron pruebas para validar la usabilidad y la experiencia del cliente al momento de llenar los formularios de carga y edición, haciendo énfasis en las pantallas que requieren una interacción con los mapas e imágenes satelitales.

Para comprobar la integración con el subsistema *mobile,* durante el desarrollo de este proyecto se realizaron una serie de pruebas para simular los casos de uso de la aplicación en el campo y el envío de mediciones y calibraciones con datos de prueba. Al finalizar la implementación del subsistema *cloud-web* correspondiente a este trabajo final, se realizó una puesta en marcha de los microservicios y la aplicación *web* en un servicio de *hosting* contratado por los referentes funcionales. Este entorno de ejecución fue utilizado para realizar algunas pruebas exploratorias de integración con el pasturómetro y la aplicación *mobile,* con el fin de facilitar el desarrollo y las pruebas de estos componentes.

Diseño del sistema

En esta sección se abordan las decisiones de diseño tomadas durante el desarrollo de este proyecto. Se analizan tanto la arquitectura de la solución general como la del subsistema *cloud-web,* correspondiente a este trabajo final. Además, se detallan las decisiones acerca de las tecnologías seleccionadas para el desarrollo y el modelo de datos utilizado.

Arquitectura general

A continuación, se presenta un diagrama en el que muestran los componentes principales del sistema y cómo se encuentran relacionados entre sí. La arquitectura general sigue un modelo de cliente-servidor, en el que los clientes son la aplicación *web* y la aplicación *mobile,* mientras que el *backend* está compuesto por un esquema de microservicios. Los servicios principales son: el servicio de usuarios (*UserService*), el servicio de potreros (*PaddockService*) y el servicio de balance de pasturas y estadísticas (*PastureBalanceService*). Por otro lado, se diseñaron dos servicios adicionales (*BFFWebApp* y *BFFMobileApp*) que funcionan como mediadores de la comunicación entre los clientes del sistema y los microservicios. A su vez, los microservicios se comunican entre sí para obtener la información necesaria para llevar a cabo sus funciones.

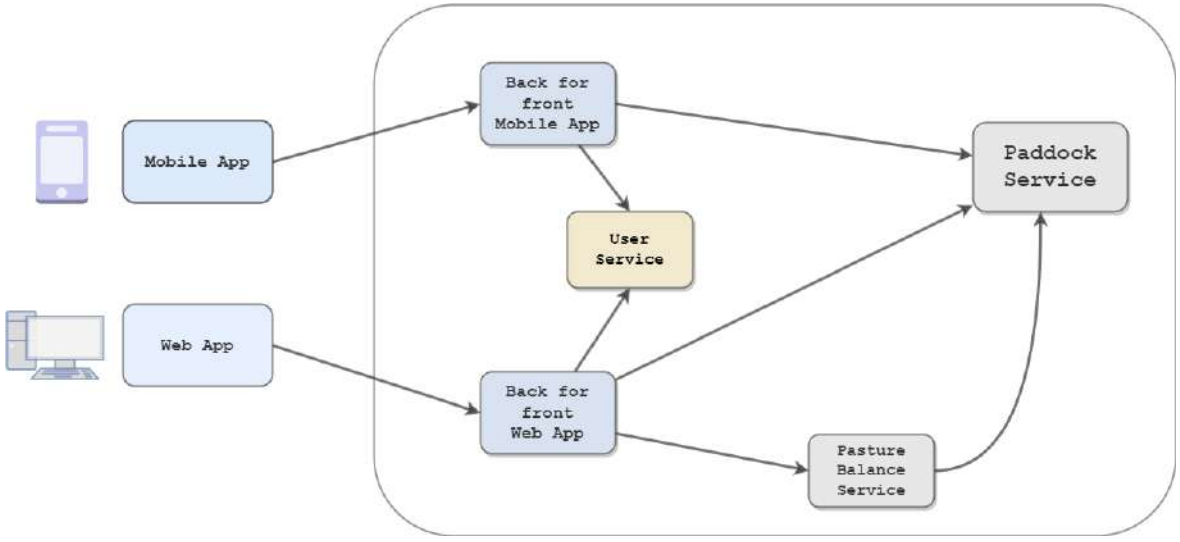

Figura 3 - Diagrama de arquitectura

Arquitectura subsistema *cloud*

El diseño de arquitectura del subsistema *cloud* fue mayormente guiado por dos requerimientos principales relevados durante la etapa de análisis. En primer lugar, el sistema debe estar preparado para escalar fácilmente a futuro con el fin de agregar nuevas funcionalidades y servicios. En segundo lugar, la arquitectura debe contemplar la posibilidad de escalar en el mercado, por lo que debe estar preparada para funcionar tanto a nivel nacional como internacional.

Con el fin de cumplir con los requerimientos indicados previamente, se optó por una arquitectura de microservicios[6] para el diseño del subsistema *cloud.* En este tipo de arquitecturas no existe un único servidor monolítico que contenga toda la funcionalidad del sistema, sino que existen varios servidores más pequeños que se comunican entre sí y cuya responsabilidad es más acotada. En este caso, cada microservicio desarrollado consiste en una *API REST* y puede ser modificado o reemplazado por otras implementaciones que cumplan con las interfaces de comunicación definidas. A su vez, se pueden agregar nuevos microservicios que consuman la información de los existentes para introducir nuevas funcionalidades en el sistema.

User Service

El *User Service* es el microservicio encargado de la gestión de usuarios y grupos. Su función principal es autenticar las credenciales de un usuario y gestionar la generación y revocación de *tokens* de acceso al sistema. A través de este servicio, un administrador del sistema puede crear las cuentas para los distintos clientes que adquieran el producto. A su vez, los usuarios finales del sistema pueden crear un grupo de usuarios bajo su dependencia para gestionar su equipo de trabajo.

Paddock Service

El *Paddock Service* es el microservicio que concentra la mayor parte de la lógica del subsistema *cloud*, ya que es el encargado de almacenar y gestionar los datos de los potreros, las especies, las calibraciones, las mediciones y los reportes generados a partir de estas. Además, brinda las interfaces necesarias para consultar, modificar y eliminar los datos según corresponda, con el fin de otorgar la flexibilidad necesaria para desarrollos futuros que requieran hacer uso de este componente.

Pasture Balance Service

En este microservicio se realizan todos los cálculos relacionados a los reportes estadísticos de stock y tasa de crecimiento de los distintos potreros y del campo en general. Para ello, consume activamente las interfaces del servicio de potreros para hacer uso de los datos almacenados en él. En el alcance de este trabajo final se buscó obtener una implementación de este servicio con una funcionalidad suficiente para lanzar el producto al mercado, pero es el componente donde se estima que se implementará la mayor parte de los requerimientos deseados a futuro.

Aplicación web

La aplicación *web* del sistema es una interfaz gráfica en la que los usuarios finales podrán consumir los servicios del subsistema *cloud.* El objetivo principal de diseño de este componente fue el de lograr una aplicación que resulte atractiva e intuitiva al usuario y que brinde la funcionalidad necesaria para apoyar la toma de decisiones y planificación del pastoreo en base a los datos y mediciones almacenados en los servicios. Cuenta con una serie de pantallas que permiten realizar altas, bajas y modificaciones de los datos del sistema tales como los potreros, las especies, las curvas de calibración, el ganado y los pastoreos, entre otros. A su vez, permite al usuario visualizar las recorridas de mediciones y definir las regiones geográficas de los potreros de una manera interactiva haciendo uso de imágenes satelitales. Por otro lado, también cuenta con gráficos en los que se pueden visualizar los datos de los reportes estadísticos actuales e históricos respecto al stock y tasa de crecimiento del pasto.

Esta aplicación no consume los datos directamente de los microservicios, sino que utiliza únicamente las interfaces provistas por el servicio llamado *BFFWebApp*.

Patrón Backends for Frontends

Cuando se trabaja con una arquitectura orientada a microservicios y se necesita utilizar los datos almacenados en el sistema, es probable que una simple consulta requiera desencadenar una serie de llamadas a más de un servicio. A su vez, cuando existe más de una aplicación cliente en el sistema (*mobile* y *web* en este caso), es probable que las necesidades de cada una de ellas no sean exactamente las mismas ya sea por su funcionalidad, capacidad de procesamiento, etc. Para

evitar que las aplicaciones cliente del sistema soliciten los datos a cada uno de los microservicios de manera individual, se tomó la decisión de aplicar el patrón de diseño llamado *Backends for Frontends*[7]. Esta técnica consiste en desarrollar un componente adicional que funcione como un único punto de acceso para cumplir con los requerimientos específicos de una aplicación cliente en particular. Su función principal es recibir las solicitudes, distribuirlas en los servicios adecuados, recopilar los datos necesarios y devolver una respuesta. Además, actúa como una barrera de protección del sistema ya que a cada *frontend* se le permite acceder únicamente a la información que requiere para satisfacer sus necesidades.

Como se mencionó anteriormente, en la arquitectura general de la solución existen dos *frontends*: la aplicación *web* desarrollada en este trabajo final y la aplicación *mobile* comprendida en el alcance del proyecto paralelo correspondiente. Por lo tanto, se diseñaron dos servicios adicionales, uno para cada *frontend* del sistema.

- ***Back for front web***. Este componente es el encargado de responder a las solicitudes provenientes del cliente *web* del sistema.
- ***Back for front mobile***. Este componente es el encargado de responder a las solicitudes provenientes de la aplicación *mobile* del sistema. Tiene un acceso más limitado a la información debido a las características y funcionalidades inherentes a este cliente en particular.

Tecnologías

Backend

Cada microservicio del *backend* puede ser implementado utilizando tecnologías diferentes siempre que cumpla con el contrato de interfaces establecido. Esta característica es sumamente positiva para desarrollos futuros ya que permite una gran flexibilidad en los equipos de trabajo. Sin embargo, en este proyecto se consideró adecuado homogeneizar y utilizar las mismas tecnologías para implementar todos los servicios debido a que esto permite agilizar el desarrollo en equipos pequeños de trabajo y reduce los tiempos de investigación y aprendizaje de nuevas herramientas.

Se decidió utilizar el lenguaje de programación TypeScript con el entorno de ejecución Node.js y el *framework* de código abierto Express.js. En la actualidad, este set de herramientas es el más popular para el desarrollo *backend*, por lo que

existe una comunidad de desarrolladores muy grande y activa. Express.js es un *framework* minimalista que dispone de un conjunto de facilidades para el desarrollo de servidores HTTP y APIs RESTful altamente escalables. Por otro lado, el equipo contaba con experiencia en el uso de estas tecnologías tanto en el ámbito laboral como en el universitario, por lo que se consideró que esta elección era la más conveniente.

Aplicación web

El lenguaje de programación utilizado para el desarrollo de la aplicación *web* fue TypeScript con Node.js y se eligió el *framework* de código abierto React.js para el desarrollo de los componentes visuales en conjunto con la librería Redux que permite gestionar el estado de la aplicación del lado del cliente. Los motivos principales de esta elección residen en que este conjunto de tecnologías es actualmente la más utilizada y ambos integrantes del equipo contaban con experiencia en el uso de ellas. Además, son herramientas cuyas curvas de aprendizaje son ideales para adquirir los conocimientos necesarios para construir aplicaciones robustas y modernas en un corto período de tiempo.

Por otro lado, se mantuvieron conversaciones con el equipo de desarrollo de la aplicación *mobile* para llegar a un acuerdo y optar por tecnologías similares que brinden al usuario una experiencia de uso consistente en ambos clientes del sistema. Además, en un futuro, un mismo equipo de desarrollo podría tener a cargo el mantenimiento de ambas aplicaciones al tratarse de implementaciones similares.

Tecnologías para implementación de mapas

Uno de los requerimientos principales de la aplicación *web* era el de poder visualizar las recorridas de mediciones en el campo y poder definir las regiones geográficas de los potreros en un mapa. Para lograr esta funcionalidad, se debe hacer uso de imágenes satelitales y librerías que permitan manipularlas y trabajar con ellas.

Luego de investigar y analizar las diferentes alternativas, se optó por utilizar la librería Leaflet[8] y las imágenes satelitales provistas por OpenStreetMap[9]. Ambas herramientas se encuentran entre las más utilizadas en la comunidad de desarrolladores y están en constante mantenimiento. Esta elección requiere un poco más de esfuerzo de desarrollo para las funcionalidades específicas de este proyecto, pero su principal ventaja es que dichas librerías son de código abierto y

100% gratuitas. De esta manera, el producto puede ser lanzado al mercado sin costos operativos adicionales.

Las otras alternativas analizadas fueron Google Maps[10] y MapBox[11], que son consideradas como las opciones más populares para trabajar con mapas. La principal desventaja de estas herramientas son los costos de uso. Ambas cuentan con un límite de cantidad de peticiones mensuales a partir del que se debe comenzar a abonar una suscripción en dólares. Como se espera que el producto pueda escalar de manera internacional, esto supondría un aumento del costo de uso del servicio a futuro.

TypeScript

Como se mencionó anteriormente, el lenguaje de programación elegido tanto para los microservicios como para la aplicación *web* fue TypeScript[12]. TypeScript es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft. Es un superconjunto de JavaScript que principalmente añade validaciones estáticas de tipos de datos y objetos basados en clases. Esta decisión se fundamentó en la mejora sustancial que TypeScript aporta a la mantenibilidad y depuración del código, ya que las validaciones de tipos permiten definir interfaces más estructuradas en código que son validadas en tiempo de compilación.

Base de datos

Al igual que en la elección de tecnologías para el desarrollo de los microservicios, se contaba con la libertad de optar por tecnologías de bases de datos diferentes para cada implementación. Sin embargo, se tomó nuevamente la decisión de utilizar la misma tecnología para todos los servicios que lo requieran bajo los mismos fundamentos: facilitar el desarrollo y mantenimiento y reducir los tiempos de aprendizaje e investigación que serían mayores en el caso de optar por tecnologías diferentes.

Una vez definidos los modelos de datos a utilizar en cada uno de los servicios, se consideró que las entidades del dominio y los procesamientos de datos necesarios se ajustaban de una forma más acertada a un modelo no relacional o NoSQL que a un modelo relacional. Además, gran parte de los datos a almacenar y procesar en el sistema son estructuras de datos geoespaciales, que se ajustan de mejor manera a este tipo de modelos de datos. Es por esto que, en base a la experiencia de los

integrantes, se consideró que trabajar con una base de datos NoSQL favorecería la eficiencia y velocidad del desarrollo, así como también el rendimiento del producto final.

La tecnología elegida fue MongoDB[13], que es un sistema de base de datos orientado a documentos altamente escalable y flexible que se encuentra entre los más utilizados, por lo que posee una gran base de usuarios en la comunidad. Si bien los integrantes del equipo no contaban con experiencia en el uso profesional de esta herramienta, estaban familiarizados con ella por haberla utilizado por motivos académicos. Las principales ventajas de su elección son su fuerte integración con el lenguaje de programación elegido para el desarrollo, dado que el formato de almacenamiento y consulta de datos está basado en documentos de tipo JSON (JavaScript *Object Notation*) y en sus características de optimización para almacenamiento y consultas de datos geoespaciales. Además, es una tecnología que posee un lenguaje de consulta sencillo y fácil de entender, por lo que los tiempos necesarios para aprender a utilizarla son bajos.

Seguridad

Autenticación

Para implementar la gestión de cuentas y credenciales de usuarios, que comprende su creación, administración y autenticación, se decidió utilizar una herramienta preexistente y adaptarla a las necesidades de este proyecto en lugar de realizar un desarrollo completamente desde cero.

El motivo principal en el que se basó esta decisión fue en el ahorro de tiempo y recursos que se obtiene al utilizar una herramienta preexistente en lugar de desarrollar una solución completa. Implementar un sistema de gestión de usuarios y seguridad desde cero requiere un esfuerzo considerable de diseño, desarrollo, pruebas y mantenimiento que pueden implicar un proyecto en sí mismo con una duración considerable. Además, el servicio de usuarios de este proyecto no cuenta con requerimientos específicos que justifiquen destinar tantos recursos para realizar un desarrollo desde cero o que supongan un problema al momento de utilizar soluciones estandarizadas.

La herramienta elegida fue Keycloak[14], una plataforma de código abierto que el equipo de desarrollo conocía por experiencias previas a este trabajo final. La

principal ventaja es que su uso es gratuito, a diferencia de servicios similares como Firebase[15]. Es una de las más utilizadas del mercado y es una solución altamente probada en términos de seguridad y autenticación. A su vez, brinda una serie de librerías e interfaces para manipular y adaptar la herramienta a los requerimientos específicos de cada proyecto.

Confidencialidad

El producto será vendido en forma de servicio a numerosos clientes de distintos establecimientos ganaderos, y cada uno de ellos tendrá sus propios datos y grupos de usuarios para que sus trabajadores puedan hacer uso del sistema. Dichos datos serán almacenados en los mismos servidores, es decir, no se montará una instancia del sistema exclusiva para cada cliente. Por lo tanto, se definieron los mecanismos de seguridad necesarios para que cada cliente y su grupo de usuarios accedan únicamente a su colección de datos.

Para asegurar que no se filtre información, se diseñó un modelo de datos en el que cada documento está relacionado explícitamente a un grupo de usuarios. De esta manera, en cada consulta se puede filtrar por este atributo para evitar retornar información indebida. Para ello, el repositorio de datos está configurado para que se filtren todas las consultas de manera automática en base al grupo del usuario que la realice, y este dato se obtiene a partir del *token* de autenticación que se encuentra en la cabecera de cada petición a los servicios, por lo que no es necesario validar esta restricción en cada funcionalidad nueva que se implemente.

Integridad

Para asegurar la integridad de los datos almacenados, se implementaron las validaciones necesarias al momento de realizar altas y bajas con el fin de verificar que estas acciones no dejen el sistema en un estado inconsistente respecto a los datos almacenados. Además, los procesos llevados a cabo en los servicios se implementaron de manera transaccional, por lo que sus acciones se ejecutarán de manera atómica y no se realizarán cambios parciales que puedan generar inconsistencias en los datos.

Testing

Para realizar las pruebas exploratorias en los microservicios se optó por hacer uso de Postman[16]. Se eligió esta herramienta debido a que es una plataforma muy completa que permite diseñar, documentar y probar APIs. A su vez, posee funcionalidades para automatizar las llamadas a los servicios y realizar acciones en base a la respuesta obtenida. Por otro lado, los integrantes ya contaban con experiencia en el uso de esta tecnología.

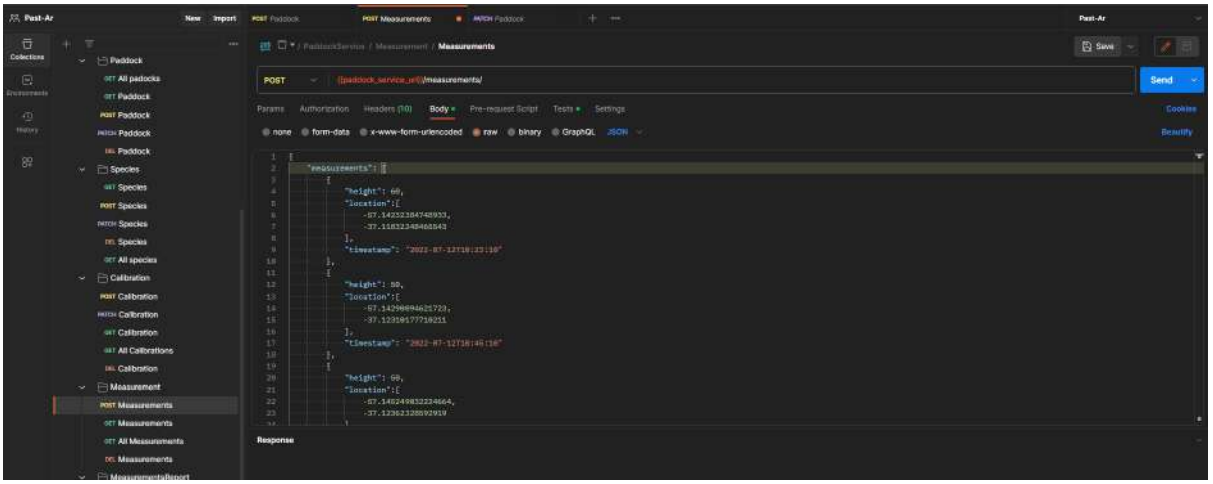

Figura 4 - Ejemplo de envío de mediciones al servicio de potreros del backend.

En cuanto a las pruebas realizadas en la aplicación *web*, se utilizaron las herramientas de desarrollo básicas existentes en los navegadores como Firefox y Chrome y la extensión Redux DevTools que permite analizar en detalle el estado de la aplicación en tiempo real.

Modelo de datos

A partir de los requerimientos relevados en la etapa de análisis, se trabajó en conjunto con el equipo del proyecto *mobile* para diseñar un modelo de datos que cumpla con ellos. En esta etapa se consideró que la colaboración entre ambos subsistemas era fundamental dado que, si bien la implementación del modelo es parte del subsistema *cloud*, la aplicación *mobile* necesitará conocerlo en profundidad para poder enviar y consultar los datos.

Se definieron las entidades del dominio con sus campos, sus nombres y las relaciones entre ellas. El modelo de datos obtenido puede verse representado en el diagrama de entidad-relación (DER) presente en el Apéndice D.

Producto

Requerimientos

A continuación se listan todos los requerimientos del sistema relevados durante la etapa de análisis. Los requerimientos funcionales y no funcionales esperados fueron implementados en el alcance de este proyecto, mientras que los requisitos deseados fueron relevados para ser implementados en una etapa posterior a este trabajo final.

Requerimientos funcionales esperados

- **RF01.** El sistema deberá autenticar a los usuarios mediante *email* y contraseña.
- **RF02.** El sistema permitirá a los administradores generar cuentas de usuario para los clientes que adquieran el producto.
- **RF03**. El sistema permitirá a los clientes generar cuentas de usuario con diferentes permisos para su equipo de trabajo.
- **RF04.** El sistema permitirá definir potreros cuyos datos serán: un nombre, una región geográfica, una especie de pasto y una curva de calibración asignada.
- **RF05.** El sistema permitirá definir especies cuyos datos serán: un nombre y una lista de curvas de calibración asociadas.
- **RF06.** El sistema permitirá definir curvas de calibración que serán utilizadas para convertir las mediciones de altura de pasto registradas en el campo en valores expresados en unidades de $\frac{kgms}{ha}$ (kilogramos de masa seca por hectárea). Las calibraciones tendrán asociadas un nombre, un conjunto de mediciones con sus pesos correspondientes y el grado del polinomio utilizado para aproximar una función matemática que represente dichos datos.
- **RF07.** El sistema permitirá registrar mediciones expresadas como un conjunto de valores georeferenciados.
- **RF08.** El sistema deberá procesar las mediciones, identificar los potreros donde fueron tomadas y utilizar la curva de calibración asignada a éste para calcular la cantidad de pasto medida.

- **RF09.** El sistema permitirá registrar datos de ganado, cuyos atributos serán: un nombre, cantidad de animales y consumo diario de pasto por animal.
- **RF10.** El sistema permitirá registrar la fecha, hora y potrero en el que se envió a los animales a pastorear para poder usar esta información al momento de calcular la tasa de crecimiento del pasto en cada uno de los potreros del campo.
- **RF11.** El sistema brindará información respecto al stock actual e histórico tanto de los potreros como del promedio del campo.
- **RF12.** El sistema brindará información respecto a la tasa de crecimiento actual e histórica tanto de los potreros como del promedio del campo.
- **RF13.** La aplicación web permitirá realizar la carga y edición de los potreros a través de un formulario y un mapa para poder definir las regiones geográficas.
- **RF14.** La aplicación web permitirá visualizar las series de mediciones georeferenciadas en un mapa.
- **RF15.** La aplicación web poseerá gráficos que permitan visualizar los datos de stock y tasa de crecimiento actuales y su evolución histórica.
- **RF16.** La aplicación web permitirá completar los pesos asociados a las mediciones de las curvas de calibración, modificar su nombre y el grado del polinomio utilizado para calcular la curva de ajuste.
- **RF17.** La aplicación web permitirá la creación y edición de especies de pasto.

Requerimientos no funcionales esperados

- **RNF01.** La aplicación web tendrá una interfaz de usuario que sea adaptable a distintos tamaños de pantallas de escritorio y debe correr en un navegador.
- **RNF02**. La aplicación web tendrá una experiencia de usuario intuitiva y contará con alertas descriptivas.
- **RNF03.** La aplicación web tendrá un aspecto atractivo y moderno.
- **RNF04.** La arquitectura del sistema estará preparada para escalar y dar servicio a clientes nacionales e internacionales.
- **RNF05.** La arquitectura del sistema estará preparada para ser expandida en proyectos futuros y facilitar la incorporación de los requerimientos funcionales deseados no implementados.

- **RNF06.** Los componentes del sistema podrán ser fácilmente reemplazables por nuevas implementaciones sin requerir alteraciones en los módulos relacionados.

Requerimientos funcionales deseados

- **RFD01.** El sistema permitirá el procesamiento de imágenes satelitales para la estimación de la cantidad de pasto disponible en el campo.
- **RFD02.** El sistema implementará modelos de predicción clima-dependientes para la tasa de crecimiento que hagan uso de servicios meteorológicos externos.
- **RFD03.** El sistema permitirá la definición de potreros con secciones internas que no deban ser tomadas en cuenta para el cálculo de su área total (por ejemplo, un lago en el centro de un potrero).
- **RFD04.** El sistema permitirá a los administradores predefinir una colección estándar de curvas de calibración que podrán ser utilizadas por todos los clientes del sistema.
- **RFD05.** El sistema identificará las situaciones en las que el usuario tome mediciones en un potrero que no posea una calibración asignada y recomendará una curva según la estación actual del año y la especie de pasto asignada a ese potrero.

Flujo de trabajo

En el siguiente diagrama de Modelo y Notación de Procesos de Negocio se muestra cómo es la interacción principal del usuario con el sistema. Se identifican dos actores principales: el usuario administrador del campo, quien accede a la aplicación *web* para realizar la gestión de los datos del establecimiento, y el usuario que trabaja en el campo y hace uso de la aplicación *mobile* junto al pasturómetro para tomar las mediciones.

El objetivo principal de este flujo de trabajo es que el usuario final pueda visualizar los datos y reportes estadísticos generados a partir de la realización de mediciones periódicas de la cantidad de pasto existente en el campo. De esta manera, se espera que el productor del campo pueda tener información actualizada del estado de las pasturas en su establecimiento y utilizarla para elaborar un plan de administración eficiente de sus pasturas.

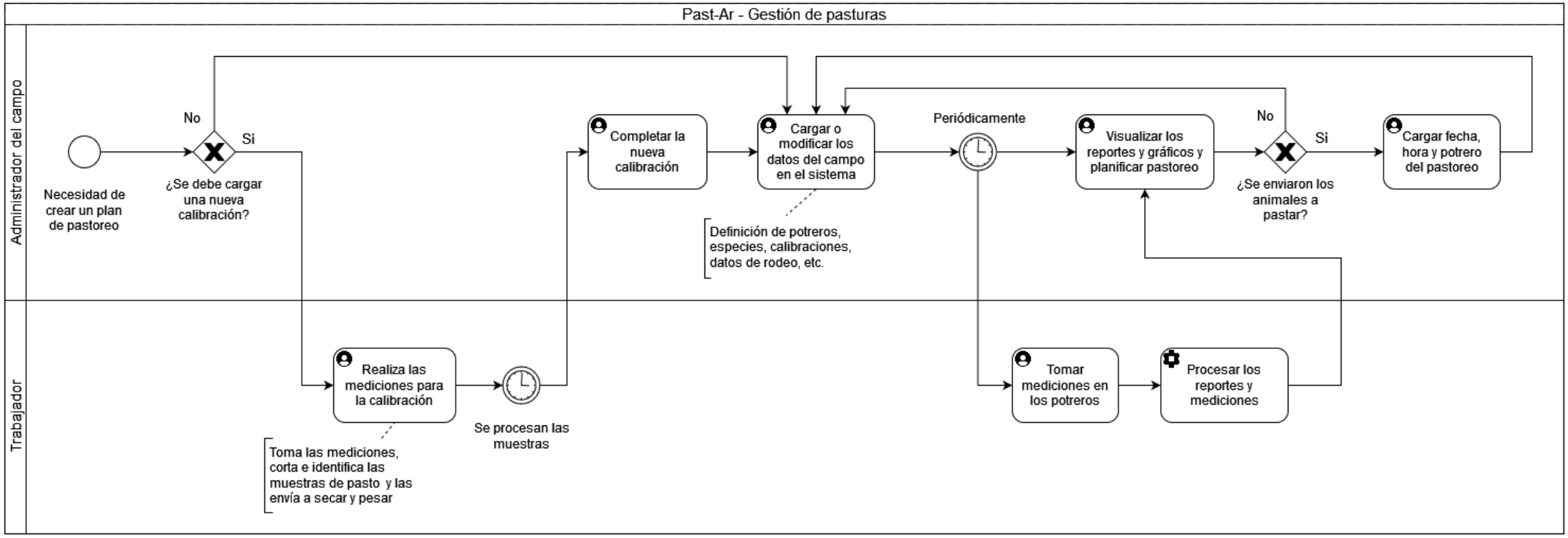

Figura 5 - BPMN principal

Producto obtenido

El producto desarrollado consta de una aplicación *web* y un *backend* compuesto de varios microservicios. A continuación, se detallan las características y funcionalidades de ambos entregables.

La aplicación *web* es la interfaz principal del sistema con la que el usuario accederá y hará uso de los servicios que se exponen en el *backend.* Esta aplicación cliente posee las siguientes características diferenciadas por las pantallas de la interfaz:

- **Pantalla de inicio de sesión.** Permite autenticar el acceso al sistema con usuario y contraseña para una cuenta preexistente.
- **Pantalla de stock actual**. Posee una serie de gráficos y métricas que permiten visualizar un resumen de los datos más relevantes para la toma de decisiones. Se muestran:
    - Gráfico de barras que indica la cantidad actual de pasto por potrero.
    - Gráfico de barras que indica la tasa de crecimiento actual por potrero.
    - Gráfico de barras que indica la tasa de crecimiento actual por especie de pasto.
    - Valor del stock promedio actual en kilogramos de masa seca por hectárea.
    - Valor de la demanda diaria de pasto calculado en base al consumo de los animales cargados.
    - Valor de la oferta diaria de pasto, es decir, la cantidad de pasto producida por día en el campo.
    - Valor promedio de la tasa de crecimiento.

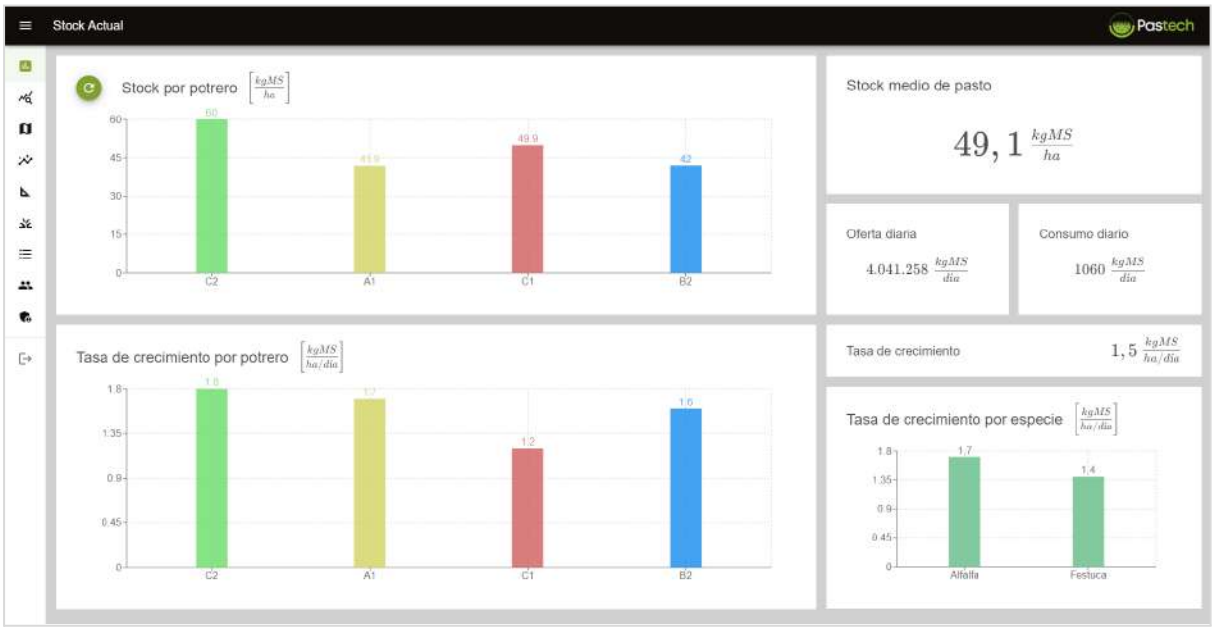

Figura 6 - Pantalla de visualización de stock actual y estadísticas

- **Pantalla de evolución histórica**. Permite visualizar mediante dos gráficos de barras los datos históricos del stock y la tasa de crecimiento. Se permite filtrar por un rango de fechas, elegir entre ver el detalle de todos los potreros, filtrar por un potrero en particular o ver únicamente la evolución de los valores promedios del campo en ambos casos.
- **Pantalla de carga y edición de potreros.** Consta de un mapa en el que se pueden visualizar, crear y modificar los potreros de una manera interactiva. Además, se muestra un listado de los potreros con sus datos y un formulario que permite editarlos. Los datos editables además de la región geográfica en el mapa son: la especie y calibración asociadas, el nombre y el color con el que se representa en el mapa.

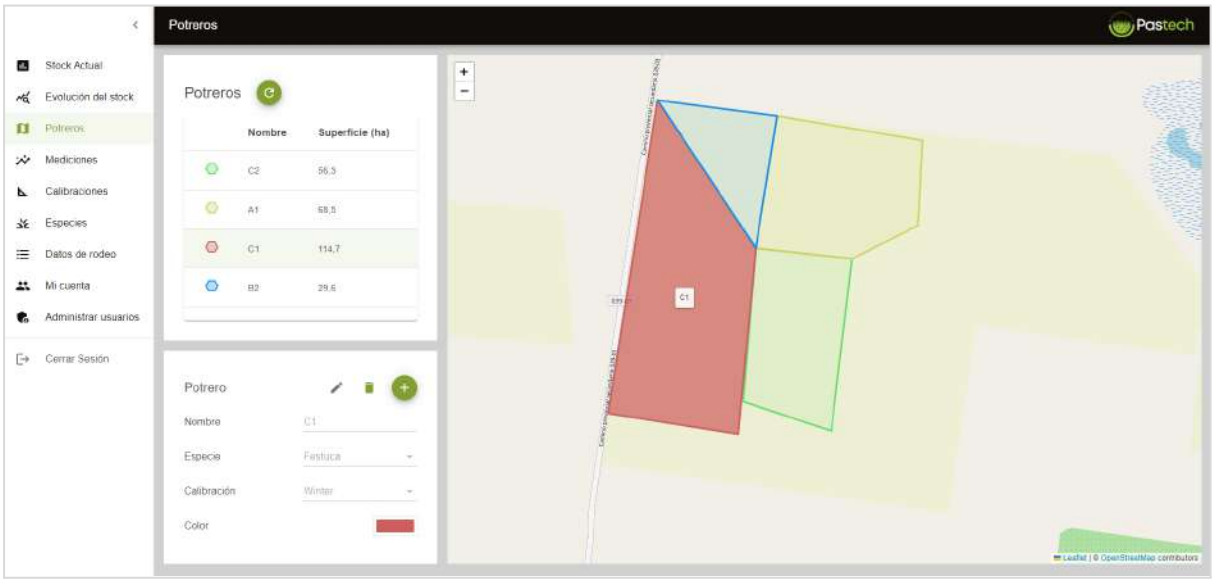

Figura 7 - Pantalla de visualización y edición de potreros

- **Pantalla de visualización de mediciones.** Permite visualizar en un mapa las series de mediciones realizadas y los potreros definidos. Además, se muestra un listado de las series de mediciones que puede filtrarse por un rango de fechas. Para la serie seleccionada, se muestra un listado de los resultados obtenidos para cada uno de los potreros donde se tomaron y se permite recalcular dichos reportes en caso de que, por ejemplo, se modifique la curva de calibración asignada a un potrero o se cambie su región geográfica y se desee ver los cambios plasmados en esa serie de mediciones.

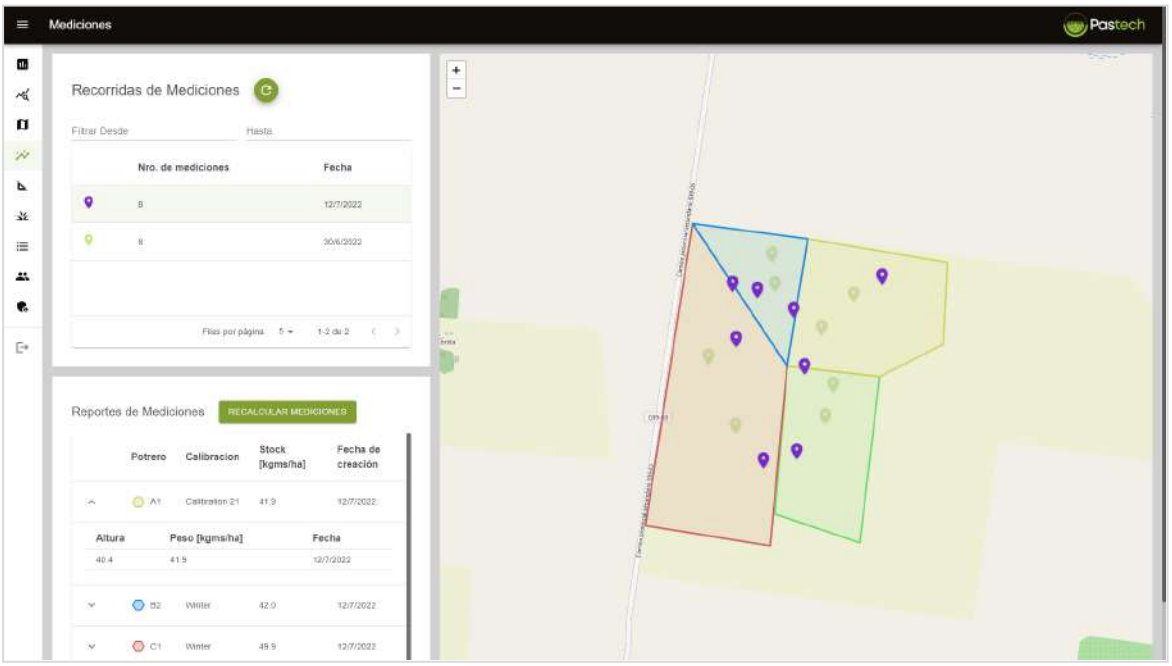

Figura 8 - Pantalla de visualización de mediciones

- **Pantalla de carga y edición de calibraciones.** Permite visualizar las calibraciones existentes y completar o editar las mediciones asociadas. Se muestra, además, un gráfico de ejes cartesianos con las mediciones y la curva de aproximación calculada.
- **Pantalla de carga y edición de especies.** Permite añadir, editar y eliminar especies de pasto y asignar a ellas una lista de calibraciones.
- **Pantalla de carga y edición de datos de rodeo.** Por un lado, permite cargar y modificar los datos de rodeo (tipos de animales). Cada tipo de animal tiene un nombre, un consumo diario por animal que es utilizado para realizar cálculos estadísticos y la cantidad que se poseen en el establecimiento. Por otro lado, permite cargar la fecha, hora y potrero de los pastoreos realizados. Este dato es utilizado para el cálculo de la tasa de crecimiento del pasto, debido a que, si se envían los animales a pastar, se reduce la cantidad de pasto en el potrero y la siguiente serie de mediciones dará un stock de pasto menor. Esto debe identificarse para no realizar un cálculo erróneo de la tasa entre dos series de mediciones si en el medio hubo pastoreo.
- **Pantalla de cuenta de usuario.** Permite modificar los datos propios del usuario, incluyendo contraseña, nombre y apellido. En el caso de que el

usuario autenticado posea los permisos correspondientes, permite gestionar los usuarios del grupo correspondiente al establecimiento ganadero, con la posibilidad de agregar, habilitar y deshabilitar nuevas cuentas para los trabajadores y asignarles determinados permisos para hacer uso del sistema.

- **Pantalla de administrador del sistema.** Permite al administrador o dueño del sistema gestionar los usuarios clientes con la posibilidad de agregar, habilitar o deshabilitar cuentas nuevas para los clientes que contraten el sistema.

Por otro lado, el *backend* posee todas las funcionalidades necesarias para recibir y almacenar las mediciones provenientes de la aplicación *mobile*, realizar la gestión las credenciales de usuario y autenticar el acceso, crear y modificar los datos correspondientes a las entidades del dominio y consultar todos los datos almacenados y los reportes y métricas estadísticas.

Benchmarking

En el mercado objetivo de la problemática en cuestión no se pudo encontrar un competidor cuyo producto combine la toma de mediciones mediante un plato o pasturómetro y el procesamiento y análisis de los datos obtenidos. Por un lado, existe un cierto número de empresas que comercializan pasturómetros, pero que no brindan ningún servicio de procesamiento integrado. Por otro lado, hay empresas que brindan servicios *cloud* de gestión, pero que hacen uso de imágenes satelitales o cargas manuales de datos para estimar la cantidad de pasto en los establecimientos y no ofrecen ninguna otra solución para la toma de mediciones. En este segundo grupo de empresas se destacan Pasture.io y Kelpie por la similitud con el servicio *cloud* que se desarrolló en este proyecto.

Pasture.io

Pasture.io[17] es una empresa australiana cuyo producto y modelo de negocios tiene un grado de madurez muy elevado. Tiene clientes a lo largo de todo el mundo, en países como Estados Unidos, Argentina, Brasil, Sudáfrica, Australia, Noruega, entre otros. Ofrece un sistema de gestión *web* muy completo que cuenta con una versión *mobile* y permite definir las regiones geográficas de los potreros, registrar datos de pastoreo, estimar la tasa de crecimiento y la demanda de pasto, etc. Además, cuenta con pronósticos de la tasa de crecimiento y del stock a futuro y con la

posibilidad de registrar eventos como plantación y sembrado de pasto, fertilización, fumigación y cosecha por potrero. Esta información se muestra tanto en gráficos como en listas y mapas.

Para tomar los datos de la cantidad de pasto, utiliza imágenes provistas por más de 150 satélites con el fin de poder tener actualizaciones diarias de la información. Ofrece tres tipos de suscripciones cuyo precio se calcula en base a las hectáreas que se deseen contratar y se diferencian principalmente en la resolución de las imágenes satelitales (de ella depende la precisión de los datos) y la frecuencia de actualización de esta información. Además, los planes más caros poseen características adicionales tales como la posibilidad de administrar más de una empresa o establecimiento, prioridad en las colas de procesamiento en la nube e integración con servicios meteorológicos y de fertilización externos. Los precios de los planes de Pasture.io por mes para 200 hectáreas son 143 USD, 203 USD y 473 USD respectivamente para los tres planes (conversión aproximada de dólares australianos a dólares americanos). Cabe destacar que la relación entre el precio y las hectáreas no es lineal y ofrecen planes de pago cuatrimestrales, anuales y bianuales con descuentos.

Kelpie

Kelpie[18] es una pequeña empresa argentina que fue creada a partir de que, en un viaje a Australia, sus fundadores observaran las prácticas de cuantificar y gestionar el forraje para la producción ganadera utilizadas en la región para la planificación del pastoreo. El producto ofrecido por esta empresa es el más similar a la solución desarrollada en este proyecto y consta de una plataforma *web* y una aplicación *mobile*. Permite definir los lotes o potreros del campo mediante un mapa interactivo y asignarles una especie de pasto predominante. Además, permite cargar datos de rodeo, planificar circuitos de pastoreo por tipo de animal y potrero y registrar eventos tales como tratamientos realizados en el campo y otras acciones. Cuenta con tableros y gráficos para visualizar la evolución de los datos del campo e indicadores productivos claves. La información de la disponibilidad de pasto por potrero debe ser cargada manualmente, por lo que el usuario deberá utilizar alguna otra herramienta o solución para obtener este dato y finalmente poder mantener actualizada la información en este sistema. Adicionalmente, ofrece una aplicación *mobile* cuya funcionalidad se limita a la visualización del campo en el mapa y la carga de ciertos datos como la disponibilidad de pasto y eventos por potrero y posee la capacidad de trabajar de manera *offline*.

En cuanto a las licencias de uso y comercialización, Kelpie ofrece un plan básico gratuito y un plan pago completo que ofrece un período de prueba de 15 días. El plan gratuito permite definir los datos de los potreros, cargar mediciones y visualizar los mapas y datos históricos, mientras que el plan completo ofrece la totalidad de las funcionalidades. Ambos planes permiten hacer uso de la aplicación *mobile.* No se logró obtener información acerca de los precios y la modalidad de pago (si depende de la cantidad de hectáreas del campo, si es un pago periódico o un pago único, etc.) a pesar de haber contactado al soporte, dado que no se recibió respuesta alguna. Por otro lado, se creó una cuenta para acceder al período de prueba del plan completo y no pudo ser utilizada debido a que la plataforma tiene actualmente un error que imposibilita el acceso a las funcionalidades.

Comparativa

A partir de la descripción de los productos competidores más representativos del mercado, podemos notar que Pasture.io es un sistema sumamente completo que posee una funcionalidad mayor al producto desarrollado en este trabajo final, mientras que Kelpie brinda una solución que se asimila más en cuanto a sus características. Sin embargo, Kelpie no tiene la posibilidad de trabajar de manera integrada con un sistema de medición calibrado, por lo que puede considerarse que la solución general propuesta por Pastech es superior. Cabe destacar que esta primera versión de Pastech, y uno de los objetivos es obtener un producto con la capacidad ampliar fácilmente sus funcionalidades, por lo que tiene potencial suficiente para competir en cuanto a prestaciones con el sistema de Pasture.io.

En cuanto al costo de los servicios, Pastech es competitiva. Si bien Pasture.io es un producto muy maduro y utilizado a nivel mundial, su precio puede resultar sumamente elevado para los productores locales y de la región. Por otro lado y como se comentó anteriormente, no se ha logrado obtener la información de los precios del sistema Kelpie y tampoco se ha podido hacer uso de su versión de prueba por errores en la plataforma y la falta de respuesta por parte del soporte.

	Pasture.io	Kelpie	Pastech
Origen	Tasmania, Australia	Buenos Aires, Argentina	Buenos Aires, Argentina
Grado de madurez	Alto	Bajo	Bajo
Obtención de datos de disponibilidad de pasto	Imágenes satelitales	Carga manual en la plataforma	Sistema integrado de medición por pasturómetro
Precio (200 ha)	143 USD por mes	Sin información	10 USD por mes + pago único por el pasturómetro (*)

Figura 9 - Resumen de benchmarking

(*). El precio indicado es un valor preliminar definido por Pastech acorde a su estrategia comercial. El precio del pasturómetro aún no ha sido definido.

Trabajo futuro

Al momento de definir el alcance de este trabajo final, se decidió que algunos requerimientos relevados no serían incluídos en el producto a entregar, principalmente porque extenderían considerablemente el tiempo de desarrollo de este proyecto debido a su complejidad. Sin embargo, se incluyeron en la documentación para que puedan ser implementados en una etapa futura de Pastech.

A continuación, se listan las funcionalidades destinadas a ser implementadas en un trabajo futuro:

- Mediciones mediante imágenes (*drones*/satelitales). Añadir un servicio que permita la utilización de imágenes para obtener información de la disponibilidad de pasto. Se deben obtener imágenes actualizadas del establecimiento y procesarlas para obtener un valor puntual que pueda ser corregido mediante una curva de calibración para finalmente obtener los kilogramos de masa seca por unidad de superficie.

- Modelos de simulación clima-dependientes. Cruzar la información histórica de la evolución de la cantidad de pasto y la tasa de crecimiento almacenada con datos meteorológicos con el fin de poder realizar simulaciones y estimaciones a futuro, de manera de lograr obtener predicciones más fiables acerca del crecimiento y *stock* esperado.
- Potreros definidos mediante multi polígonos. El producto entregado cuenta únicamente con la posibilidad de definir el área de un potrero mediante un polígono simple, sin polígonos internos. Esto puede suponer un inconveniente si, por ejemplo, se cuenta con algún potrero que posea un cuerpo de agua dentro de él, ya que no debería ser considerado para el cálculo de la superficie de esa región de pasto. Para evitar esto, se debe permitir la creación de potreros a partir de estructuras definidas por múltiples polígonos, de manera que se pueda delimitar un área dentro del potrero que no deba ser tomada en cuenta.
- Curvas de calibración predefinidas. En el sistema desarrollado, cada usuario debe generar sus propias curvas de calibración. En la práctica, las mismas especies de pasto sembradas en distintos establecimientos suelen tener una densidad de crecimiento similar. Por lo tanto, un administrador del sistema podría cargar curvas predefinidas para que los usuarios puedan utilizarlas sin tener que realizar las tareas asociadas a la creación de una nueva.
- Sugerencias de curvas de calibración. En el sistema actual, si un usuario toma mediciones en un potrero que no posee una calibración asignada, no se podrá calcular el stock de pasto hasta que se asigne una curva y se recalculen las mediciones. Por lo tanto, el sistema podría detectar estos casos en particular y sugerir la asignación de una calibración que se ajuste a la especie asignada al potrero y a las condiciones climáticas de la estación del año en la que se encuentre.

Por otro lado, para que este producto pueda ser implementado en un ambiente de producción con el menor riesgo de fallas posible, debe realizarse una serie de tareas de *testing* exhaustivas que incluyan pruebas automatizadas sobre la solución completa con el fin de asegurar el correcto funcionamiento del sistema en cada escenario posible. Esto es necesario debido a que en este proyecto, por requerimientos referidos a los tiempos del mercado, las pruebas realizadas sobre el producto fueron exploratorias. Además, resultaría ideal definir las políticas y mecanismos de *backup* para proteger los datos de los futuros clientes ante contingencias del sistema o de los servicios de *hosting*.

Finalmente, con el fin de cumplir con la lista de entregables del proyecto, se deben terminar de elaborar los manuales de *deployment* y de administración. Estos manuales son fundamentales para asegurar una correcta transferencia del conocimiento referido a la puesta en producción y gestión del sistema.

Memoria del proyecto

Participación de Pastech en financiamientos y eventos

La propuesta de la solución general fue presentada por los referentes funcionales en numerosos eventos en búsqueda de financiamiento para poder avanzar con el desarrollo del producto y su posterior lanzamiento al mercado.

El evento que atravesó en mayor medida a este trabajo final fue el *Hackathon* realizado en los días 11 y 12 de noviembre del año 2022 en el hall de la Municipalidad de General Pueyrredón. Esta actividad fue organizada por la Facultad de Ingeniería de la UNMdP, la Secretaría de Desarrollo Productivo e Innovación de la Municipalidad de General Pueyrredón y ATICMA. También participaron en la organización otras instituciones locales relacionadas al ámbito socio-productivo, profesional, tecnológico y educativo de la ciudad. El objetivo de este evento era que los grupos de trabajo desarrollaran y presentaran en 24 horas una propuesta que brinde una solución a una problemática existente en la sociedad argentina haciendo uso de las tecnologías de la información. Los tres ejes sobre los que se debía trabajar eran Turismo, Agricultura y Movilidad Urbana. Finalmente, las propuestas ganadoras recibirían financiamiento y serían incubadas por la Incubadora de Empresas de la UNMdP para continuar con el proyecto.

Durante el evento, se trabajó junto a los referentes funcionales y los integrantes del equipo encargado del subsistema *mobile* para analizar la problemática en profundidad y definir un prototipo de producto para lanzar al mercado. Finalmente, Pastech fue uno de los proyectos ganadores del evento y actualmente se encuentra bajo el mando y apoyo de la Incubadora de Empresas.

Cumplimiento de objetivos

A partir del desarrollo de este trabajo, se obtuvo un sistema que cumple con todos los requerimientos comprendidos en el alcance pactado. Además, cumple con las características de escalabilidad y extensibilidad, dado que posee el potencial para adaptarse en el futuro según sea requerido gracias al diseño de la arquitectura y la modularidad de los componentes de *software*. Por lo tanto, el objetivo principal de este proyecto ha sido alcanzado con éxito.

El producto obtenido permite estimar de forma rápida y precisa la oferta, demanda y tasa de crecimiento del pasto en el campo y visualizar estos datos mediante gráficos detallados, que es de suma importancia para que los usuarios puedan planificar el pastoreo y tomar las decisiones correspondientes. Sin embargo, no se ha podido medir el impacto de la solución sobre los productores debido a que el sistema aún no se está comercializando. No obstante, tiene un gran potencial para generar el incentivo deseado ya que, luego de realizar el *benchmarking*, se pudo constatar que sus características lo hacen altamente competitivo.

Finalmente, al haberse retrasado la entrega del producto por aproximadamente dos meses, el tiempo de llegada al mercado no fue el esperado. A pesar de que este resultado es considerado como satisfactorio y adecuado para una correcta inserción en el ámbito comercial, no se logró cumplir con los tiempos pactados.

Planificación esperada vs ejecución

Inicialmente, el cronograma se planteó bajo un esquema en cascada, en el que se consideraron etapas de análisis, diseño, implementación, testing y redacción del informe y se asumió una dedicación de 14 horas semanales por integrante. De esta manera, se estimó que la duración del proyecto sería de 27 semanas (714 horas totales de desarrollo).

Finalmente, en la práctica se tomó la decisión de mantener la metodología de cascada salvo para las tareas relacionadas al módulo de estadísticas del sistema, para el que se decidió postergar las etapas de análisis y diseño. Esta decisión se fundamentó en la complejidad asociada a los requerimientos de este componente, que requerían tener una mayor madurez de los conceptos del dominio. De esta manera, se comenzó a trabajar en dicho módulo luego de haber aplicado y validado los conceptos principales en los demás componentes del sistema.

El inicio del proyecto tuvo lugar en la fecha prevista, pero el desarrollo se extendió 11 semanas más allá de lo planificado, en las que la dedicación horaria fue menor a la estimada. En total, se dedicaron 824 horas a lo largo de 38 semanas.

A continuación, se presenta el cronograma estimado y el cronograma real. Los números dentro de cada barra representan la cantidad total de horas que se invirtieron en ese período. En la siguiente sección se aborda el análisis en profundidad de cada etapa del proyecto.

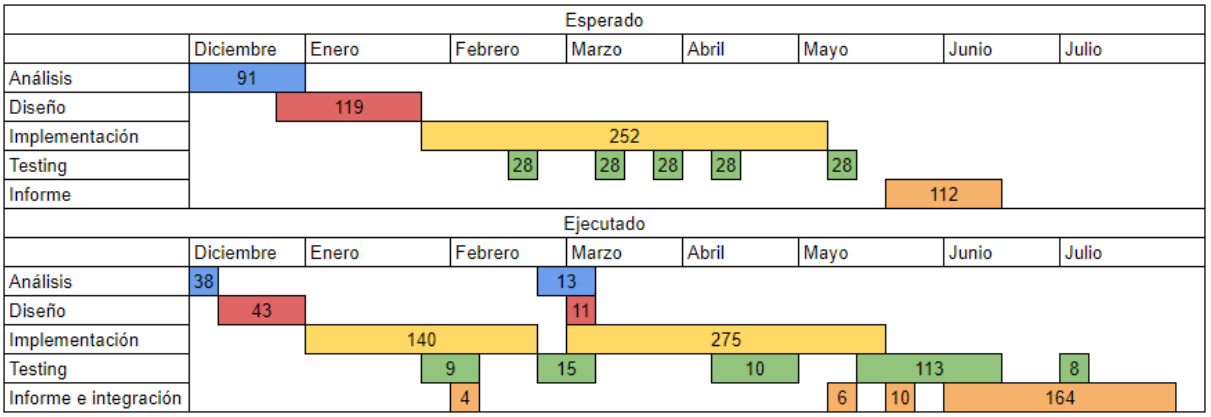

Figura 10 - Comparación entre cronogramas esperado y ejecutado

Análisis de etapas

Análisis

La mayor parte del análisis se realizó al inicio del proyecto durante el *Hackathon* de noviembre de 2022 junto a los integrantes del equipo *mobile* y los referentes funcionales. En total, se invirtieron 38 horas en una sola semana en la que se trabajó de manera intensiva. Estos valores son considerablemente menores a los estimados en un principio, tanto en la cantidad de horas totales como en la cantidad de semanas. Esto se debió principalmente a una sobreestimación de tiempos a causa de la falta de experiencia por parte del equipo en relación a las tareas de relevamiento de requerimientos y al desconocimiento del dominio del problema. Además, en el *Hackatón* se tuvieron numerosas charlas en conjunto con todos los integrantes y referentes funcionales para realizar el análisis, lo que agilizó el avance de las primeras tareas de esta etapa. Este evento permitió realizar varias iteraciones presenciales para finalmente obtener un listado preliminar de los requerimientos del subsistema a desarrollar en este trabajo final. Esta lista fue revisada y corregida en reuniones que tuvieron lugar en las semanas posteriores al evento.

Por otro lado, aproximadamente a la mitad del proyecto, se realizó una segunda etapa de análisis que consistió en varias reuniones virtuales con los referentes funcionales en las que se relevaron todos los requerimientos referidos al módulo de estadísticas y balance de pasturas. El hecho de haber postergado esta etapa permitió realizar un análisis más profundo previo a la implementación del

componente debido a que los integrantes ya poseían cierto grado de madurez de los conceptos del dominio del problema.

Diseño

La etapa de diseño tuvo una duración total de 54 horas de trabajo, que resultó ser menos de la mitad de lo estimado. Se considera que esto sucedió en gran parte debido a un error de estimación, pero también por una documentación de las horas de trabajo realizadas que no fue del todo correcta. Este error en el reporte de horas surge por no tener en cuenta que, al momento de iniciar con la implementación de un requerimiento o servicio, se dedicaban algunas horas a validar y pulir los diseños realizados previamente. Estos tiempos finalmente fueron reportados y documentados como horas en las que se realizaron tareas de implementación, que no coinciden exactamente con lo que ocurrió en la práctica.

El primer paso en esta etapa fue definir los prototipos de las pantallas de la aplicación *web* junto a los referentes funcionales y establecer los componentes principales del sistema y las interacciones entre ellos, de manera que la arquitectura resultante permita cumplir con los objetivos de escalabilidad y mantenibilidad. Posteriormente, se definieron las responsabilidades de los servicios y los datos que se almacenarían en cada uno de ellos. A su vez, se identificaron los distintos puntos en los que sería necesario tomar decisiones respecto a tecnologías a utilizar, tales como el motor de base de datos y las herramientas para implementar los servicios, la aplicación web y la gestión de usuarios y credenciales. Además, se definió un modelo de datos inicial para luego poder validarlo con el otro grupo al momento de definir las interfaces de comunicación entre la aplicación *mobile* y los servicios *cloud*.

Una vez definidas las responsabilidades de los componentes de la arquitectura *cloud-web,* se comenzó a trabajar de manera colaborativa con los integrantes del grupo *mobile* para establecer cómo se comunicarían ambos subsistemas. Se buscó formalizar la interfaz compartida lo antes posible de manera de que ambos equipos pudieran trabajar de manera independiente, sin que las necesidades de uno interrumpiera o bloqueara el flujo de trabajo del otro. Finalmente, a partir de varias reuniones, se logró obtener un diseño acorde a las necesidades de la aplicación *mobile*. Esta decisión permitió agilizar el avance del proyecto ya que, como se tenían claramente definidos los contratos de comunicación entre los subsistemas

desde una etapa temprana, cada grupo podía continuar su desarrollo sin interrupciones siempre que cumpliera con ellos.

Posteriormente, se realizó el diseño de las API del servicio de potreros y de usuarios y se añadieron las interfaces necesarias para dar lugar a otras funcionalidades que no serían utilizadas por la aplicación *mobile* (Apéndice C). Además, se diseñó un protocolo para gestionar adecuadamente los errores esperados del sistema (Apéndice B). Una vez completado este diseño, se realizaron las reuniones de validación correspondientes con los directores del proyecto y los referentes funcionales para corroborar que sea adecuado.

Como se comentó anteriormente, se optó por postergar el análisis y diseño de los aspectos relacionados al servicio de estadísticas y balance de pasturas para una etapa más avanzada del proyecto. Esta decisión permitió realizar un diseño más acertado debido a que en las etapas previas se lograron consolidar y comprender los conceptos base del dominio del problema de una mejor manera. Además, como el desarrollo de este módulo no bloqueaba ninguna otra tarea del cronograma, no supuso ningún retraso en la planificación.

Implementación y pruebas

Para las tareas de implementación se dedicaron 415 horas de trabajo en total, por lo que esta etapa se extendió 163 horas respecto a los valores estimados. A su vez, el plazo fue mayor al esperado, debido a que la etapa de implementación tuvo una duración de 19 semanas, 5 semanas más allá de lo planificado.

El orden de las tareas de desarrollo fue respetado según lo dispuesto en el cronograma y se mantuvieron reuniones semanales en las que se revisaban los avances del proyecto, se comunicaban los inconvenientes y se coordinaban las tareas de la semana siguiente. La decisión de utilizar una sistema preexistente para la gestión y autenticación de usuarios fue fundamental para simplificar y acelerar las tareas de implementación y poder dedicar más tiempo a las funcionalidades específicas del dominio del problema. De esta manera, se logró mantener un ritmo de avance acorde a los tiempos planificados durante la etapa de desarrollo de los microservicios.

Sin embargo, para las tareas de implementación de la aplicación *web* se requirió una dedicación mucho mayor a la estimada inicialmente. Esto se debe a que, por la falta de experiencia del equipo, se subestimó la cantidad de horas necesarias para

desarrollar este componente. El excedente se concentró principalmente en la investigación y aprendizaje de las tecnologías necesarias para poder lograr las funcionalidades requeridas. Además, se tuvo que llevar a cabo un análisis comparativo para decidir qué librerías utilizar para implementar las características referidas al uso de mapas interactivos e imágenes satelitales. Por otro lado, al ser una interfaz a utilizar por los clientes finales, se puso el foco en obtener una experiencia de usuario que resulte intuitiva y moderna y, por lo tanto, fue el componente que más modificaciones sufrió debido a peticiones por parte de los referentes funcionales.

En lo que respecta a las tareas de *testing*, se dedicaron 155 horas de trabajo en total. Este valor indica que se tuvo excedente de 15 horas respecto a lo estimado, pero se considera que es una variación aceptable y que se encuentra dentro del rango esperado. Por otro lado, en la práctica, la distribución real de estas horas no fue la planificada. Se realizaron pruebas menos intensivas a cada componente individual durante el desarrollo y finalmente se llevó a cabo un *testing* más completo sobre la arquitectura del sistema una vez que se completaron las tareas de implementación.

La primera versión completa del subsistema *cloud-web* fue presentada a los referentes funcionales con un retraso de dos semanas en relación a los tiempos pactados. A partir de esta presentación, surgió la adición de nuevas funcionalidades al producto. Dichas funcionalidades no fueron tenidas en cuenta en la planificación inicial del proyecto, pero se consideró que el valor que éstas agregaban a la solución final justificaba el hecho de extender la etapa de desarrollo algunas semanas para poder implementarlas.

Integración y redacción de documento

Esta etapa consistió principalmente en la redacción del informe del trabajo final. Esto llevó más tiempo del esperado a pesar de que se contaba con casi la totalidad de la información y documentos necesarios, debido a que fueron redactados a lo largo del desarrollo. Durante esta etapa, se recopiló y se ordenó dicha información, se elaboraron los apoyos visuales correspondientes y se validó la estructura del informe con los directores del proyecto. En total, se invirtieron 164 horas, 52 horas más de lo estimado.

A su vez, durante esta etapa se pusieron en funcionamiento todos los servicios y la aplicación *web* en una máquina virtual contratada por los referentes funcionales. El

objetivo principal de esta puesta en marcha del sistema fue generar un entorno de ejecución real que pueda ser utilizado para llevar a cabo pruebas de integración con el pasturómetro y la aplicación *mobile* en un campo y, de esta manera, facilitar el desarrollo de estos componentes. En total, se dedicaron 20 horas para las tareas de *deployment*, que no habían sido previstas en el cronograma original.

Esta etapa se extendió a lo largo de 11 semanas, 7 semanas más de lo estimado originalmente, que se debió principalmente a una subestimación de las tareas de redacción. Además, se tuvo que postergar parcialmente la confección del informe para destinar tiempo a las nuevas tareas relacionadas a la integración y *deployment* del sistema que no habían sido estimadas en el cronograma inicial.

Resumen

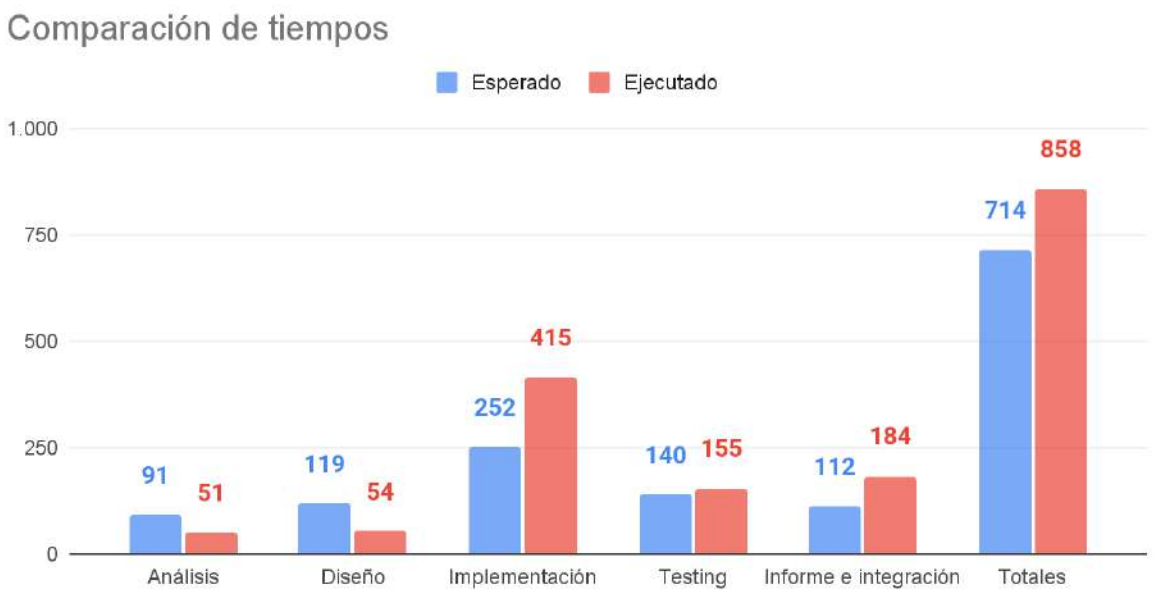

Figura 11 - Gráfico de barras: Horas esperadas vs Horas ejecutadas

En resumen, la estimación inicial de la dedicación horaria por etapas no fue del todo correcta, debido a que se pueden observar desvíos considerables respecto a lo que ocurrió en la realidad. En algunas etapas, como en las de análisis y diseño, se dedicaron menos horas de las estimadas, mientras que en otras, como en las de implementación, *testing* e informe e integración, la cantidad de tiempo dedicado fue mayor al esperado. Si bien el desvío en cantidad de horas totales fue del 20%, la duración final del proyecto se extendió en un 40% (11 semanas de excedente sobre

27 semanas totales inicialmente estimadas). Esta diferencia entre los desvíos porcentuales se debe a que la dedicación promedio fue de aproximadamente 11 horas semanales por integrante (3 horas por debajo de la planificada) debido a cuestiones laborales y personales de los miembros del equipo de desarrollo.

El desvío en los plazos se debió principalmente a los errores en la estimación de tiempos de la etapa de desarrollo, la incorporación de nuevos requerimientos que no fueron previstos en la planificación inicial y la extensión de la etapa de integración y redacción del informe.

Definir la duración total de un proyecto implica comprometerse a finalizar su desarrollo en un período de tiempo pactado. Las variaciones en los plazos de entrega por errores en la estimación pueden tener consecuencias tanto para el cliente como para aquellos que participen en el desarrollo del proyecto. Desde un punto de vista económico, una subestimación en los tiempos necesarios puede generar una pérdida de dinero y recursos que deben destinarse a tareas que no fueron previstas. Por otro lado, una sobreestimación de tiempos puede resultar en un proyecto con costos elevados, por lo que el cliente puede rechazar la propuesta y, en ese caso, el equipo de trabajo pierde una oportunidad económica.

División de tareas

La división del desarrollo de la solución general en dos subproyectos ha permitido diseñar y obtener un producto completo y funcional. Sin esta decisión, no hubiera sido posible alcanzar el desarrollo de la solución dentro de los tiempos requeridos. Ambos grupos de trabajo participaron en las reuniones iniciales de la etapa de análisis con los referentes funcionales en las que colaborativamente se definieron los requerimientos, las interfaces y la arquitectura general, de manera que ambos subsistemas puedan funcionar de forma conjunta una vez integrados. Por otro lado, la definición de la comunicación entre los subsistemas en una etapa temprana del proyecto permitió desacoplar el avance de ambos equipos. A su vez, a lo largo del desarrollo se mantuvieron reuniones en conjunto para validar y debatir las decisiones que afectaban a ambos subsistemas.

En cuanto al desarrollo de este trabajo final, se mantuvo la misma metodología durante todo el proyecto: se tuvieron reuniones al inicio de cada semana en las que se validaban los desarrollos de la semana previa y se analizaba el avance cronograma. Posteriormente, se seleccionaban las tareas a desarrollar en el

transcurso de esa semana y se definían las subtareas correspondientes, que luego eran asignadas a uno de los integrantes y se les establecía una fecha límite para su cumplimiento. De esta manera, los integrantes tenían definidas sus responsabilidades y sus objetivos de una forma clara en cada iteración del desarrollo. A su vez, se planificaban las reuniones con los referentes funcionales, con los directores o con el grupo del subsistema *mobile* en caso de que fuera necesario para validar o tomar ciertas decisiones. Por otro lado, los integrantes del grupo habían trabajado juntos numerosas veces a lo largo de la carrera, por lo que se tenía conocimiento del potencial de trabajo de cada uno de ellos. Además, se contaba con experiencia en el uso de esta metodología de trabajo, que ha sido mejorada y pulida a lo largo del tiempo.

Una vez finalizado el desarrollo del subsistema *cloud*, se montó una instancia local de los microservicios para que los integrantes del otro equipo pudieran realizar pruebas y continuar su desarrollo haciendo uso del *backend* real de la aplicación. Más adelante en el proyecto, a pedido de los directores, los referentes funcionales contrataron los servicios de *hosting* necesarios para poder hacer la puesta en marcha de la arquitectura *cloud* y la aplicación *web* desarrolladas en este trabajo final.

Utilización de nuevas tecnologías

Cuando se analizó la propuesta de la solución general y se tomó la decisión de que el desarrollo debía ser dividido en dos subsistemas, cada grupo eligió el subproyecto que le resultaba más atractivo e interesante en base a las herramientas que podían llegar a utilizarse para su desarrollo. En particular, los integrantes de este trabajo final tenían mayor interés en desarrollar una plataforma *cloud* y una aplicación *web* debido a que tenían experiencia en el uso de las tecnologías que se utilizarían para la implementación y les resultaba desafiante el hecho de tener que trabajar con nuevas herramientas para cumplir con ciertas funcionalidades específicas del dominio.

Una de las tecnologías nuevas más significativas que se utilizó fue el motor de base de datos NoSQL llamado MongoDB, que no había sido utilizada por ningún integrante más allá de algunas pruebas dentro del contexto educativo. Por lo tanto, debido a la falta de experiencia en la utilización e integración de esta herramienta, se optó por incorporar una pequeña etapa de aprendizaje. Se hizo uso de la documentación oficial y del soporte de la comunidad para poder conocer cómo

utilizarla y cuáles eran las mejores prácticas recomendadas. A su vez, se llevó a cabo la investigación en lo que respecta a las capacidades y facilidades del motor para el manejo de datos y consultas geoespaciales, que era una característica fundamental y necesaria para el procesamiento de las mediciones y el almacenamiento de los datos de los potreros.

Esta nueva herramienta fue de suma importancia para poder lograr las funcionalidades esperadas del producto de manera eficiente gracias a su excelente integración con el resto de las herramientas de desarrollo y sus características relacionadas a la consulta de datos geoespaciales. Además, la utilización de esta nueva tecnología supuso una gran experiencia en el uso de motores de base de datos NoSQL, que es una aptitud altamente buscada en el mercado laboral en la actualidad y que puede resultar útil para el desarrollo de nuevos proyectos personales.

En cuanto al desarrollo de la aplicación *web,* se contaba con experiencia en el uso de las tecnologías de desarrollo elegidas, pero en el último tiempo éstas habían sufrido cambios y habían sido actualizadas. Esto supuso atravesar una pequeña etapa de adaptación para comprender qué aspectos habían cambiado y debían utilizarse de una manera distinta. Por otro lado, se utilizaron herramientas y librerías que el equipo de desarrollo desconocía para llevar a cabo la implementación de las funcionalidades referidas al uso de mapas interactivos con imágenes satelitales. Esto permitió actualizar y ampliar los conocimientos del equipo en el uso de herramientas que en el mercado laboral son altamente solicitadas.

Conclusiones

El objetivo principal del proyecto era el de analizar la problemática relacionada al monitoreo y control de las pasturas de los establecimientos ganaderos para diseñar e implementar una solución que permita realizar estas tareas de una manera rápida, simple y precisa. Además, el producto obtenido debía estar preparado para poder crecer fácilmente en desarrollos futuros y debía poder ser lanzado al mercado con un precio accesible para fomentar su uso en la región. El sistema obtenido como resultado del desarrollo de este proyecto cumple con las funcionalidades y características indicadas según los referentes funcionales, por lo que el objetivo fue alcanzado con éxito. Además, en base a las comparaciones realizadas con productos similares del mercado, se observó que la solución desarrollada es altamente competitiva.

Debido a la magnitud del sistema, el desarrollo se dividió en dos proyectos. Esta decisión permitió obtener una solución más completa y con un alcance más amplio dentro de los tiempos esperados. A lo largo del proyecto se mantuvieron comunicaciones constantes con el otro grupo de trabajo para definir los puntos comunes a ambos subsistemas. Además, según lo planificado inicialmente, se realizaron una serie de reuniones con los referentes funcionales, quienes brindaron la información necesaria para comprender el dominio y definir y validar los requerimientos del sistema, y con los directores del trabajo final, quienes brindaron asesoramiento sobre tecnologías y la gestión del proyecto. Esto permitió adquirir y reforzar ciertas competencias profesionales que son de suma importancia para el perfil de un Ingeniero en Informática. En particular, se pusieron en práctica las capacidades de negociación con los referentes funcionales al momento de acordar cuáles requerimientos serían efectivamente implementados dentro del alcance de este trabajo y cuáles entrarían en una etapa posterior. A su vez, el hecho de trabajar colaborativamente con otro equipo de desarrollo permitió desarrollar las habilidades comunicativas necesarias para dialogar de manera eficaz. Por otro lado, durante el proyecto se lograron poner en práctica los conceptos adquiridos a lo largo de la carrera, así como también se incorporaron nuevos conocimientos referidos a tecnologías necesarias para la implementación.

Una de las tareas más difíciles de llevar a cabo fue la planificación y estimación de los tiempos. Si bien la extensión del proyecto en cuanto a cantidad de horas totales dedicadas fue mayor a la estimada, se mantuvo dentro de un rango aceptable. Sin

embargo, la distribución de tiempos para cada una de las etapas y la dedicación semanal no fue la esperada, por lo que la duración del proyecto se extendió varias semanas y no se logró cumplir con los plazos pactados en la planificación. La causa principal de estos errores en la estimación fue la falta de experiencia de los integrantes, debido a que no habían realizado una planificación de esta magnitud anteriormente.

En conclusión, el resultado fue exitoso. Desde el punto de vista técnico, se logró implementar un producto trabajando de forma colaborativa e interdisciplinaria que cumple con las funcionalidades acordadas y se reforzaron las habilidades técnicas necesarias para su desarrollo. Por otro lado, desde el punto de vista personal, el aprendizaje fue muy importante: se tuvo la primera experiencia respecto a la gestión y administración de un proyecto que abarca casi todas las etapas del ciclo de vida del *software* y se logró integrar muchos de los conocimientos vistos en la carrera.

Apéndices

Apéndice A - Glosario

API[19]. Del inglés, *Application Programming Interface*, es una herramienta que permite a dos componentes de software comunicarse entre sí mediante un conjunto de definiciones y protocolos. La interfaz puede considerarse como un contrato de servicio entre dos aplicaciones que define cómo se comunican entre sí mediante solicitudes y respuestas.

Backend. Se refiere a la parte de un sistema informático que se encarga del procesamiento y almacenamiento de datos y el manejo de la lógica para el correcto funcionamiento de la aplicación.

Calibración. Es una función que relaciona la variable que se quiere predecir con la variable predictora (en este caso, kilogramos de masa seca por hectárea y la altura medida por el pasturómetro respectivamente). Para obtener la curva de calibración se utiliza la técnica de doble muestreo, que consiste en elegir varias muestras con diferente cantidad de pasto, medir su altura y luego cortarlas y pesarlas. A partir de este proceso, se obtiene un conjunto de puntos de altura vs. cantidad de pasto asociada, a partir de los que se puede obtener una función matemática utilizando métodos numéricos.

Frontend. Se refiere a la parte visual e interactiva de una aplicación o sitio *web*. Es la capa con la que los usuarios interactúan de manera directa con los datos y funcionalidades del sistema, a través de un navegador u otra interfaz gráfica.

HTTP[20]. Del inglés, *Hypertext Transfer Protocol*, es un protocolo de comunicación diseñado para transferir información entre dispositivos en red. Sigue un esquema de solicitud-respuesta, en el que un cliente envía una petición con un determinado formato al servidor y éste envía una respuesta.

JSON[21]*.* Del inglés, *JavaScript Object Notation,* es un formato de texto para el intercambio de datos basado en un subconjunto del lenguaje de programación JavaScript. Su lectura y procesamiento es sencillo tanto para los humanos como para los analizadores sintácticos.

Linting. Proceso en el que un programa inspecciona el código fuente de una pieza de software para encontrar errores comunes y hacer cumplir estándares y buenas

prácticas de escritura tales como longitudes máximas de línea, cantidad de tabulaciones, entre otros.

NoSQL[22]. Es un término general que se refiere a todos los sistemas de almacenamiento y gestión de datos que difieren del modelo relacional en su diseño, enfoque y lenguaje de consulta.

Pasto. Es el material de origen vegetal que se utiliza como alimento para los animales herbívoros.

Pastura. Es el pasto o la hierba sembrada por el hombre. A diferencia de lo que puede ser un pastizal natural (no implantado por el hombre) que suele estar compuesto por decenas de especies de pasto, las pasturas generalmente tienen una, dos o tres especies en particular (festuca, alfalfa, etc.).

REST[23]. Es un estilo de arquitectura de software para realizar una comunicación entre cliente y servidor que utiliza HTTP sin mantener un estado entre solicitudes, es decir, sin almacenar datos relacionados al cliente en el servidor.

Superconjunto (de un lenguaje de programación)[24]. Se refiere a un lenguaje de programación que toma otro lenguaje como base y añade una serie de funcionalidades con el fin de extenderlo.

Testing. Es el proceso de realizar pruebas sobre un sistema o una aplicación para verificar que su funcionamiento sea el esperado y que cumpla con los requisitos establecidos.

Transacción[25]. Es un grupo de operaciones que tienen las siguientes propiedades: atomicidad, coherencia, aislamiento y durabilidad (ACID). El uso de transacciones permite obtener un sistema más robusto y que preserva en mayor medida la integridad de los datos.

Valor Bruto de Producción[26]. Es el valor total de todos los productos generados por una actividad económica, incluidos los primarios y los secundarios.

Apéndice B - Protocolo de errores

Status	Code	Description
400	VALIDATION_ERROR	Invalid request
401	AUTH_INVALID_LOGIN	Invalid user credentials
409	USER_ALREADY_EXISTS	User exists with the same username
401	INVALID_TOKEN	Invalid auth token
400	INVALID_REFRESH_TOKEN	Refresh token is invalid
404	USER_NOT_FOUND	User not found
409	DELETE_CALIBRATION_REFERENCED_BY_PADDOCK	Cannot delete calibration referenced by paddock
409	DELETE_CALIBRATION_REFERENCED_BY_SPECIES	Cannot delete calibration referenced by species
409	DELETE_SPECIES_REFERENCED_BY_PADDOCK	Cannot delete species referenced by paddock
404	CALIBRATION_NOT_FOUND	Calibration not found
404	MEASUREMENT_NOT_FOUND	Measurement not found
404	MEASUREMENT_REPORT_NOT_FOUND	Measurement report not found
404	PADDOCK_NOT_FOUND	Paddock not found
404	SPECIES_NOT_FOUND	Species not found
404	CATTLE_NOT_FOUND	Cattle not found
404	GRAZING_LOG_NOT_FOUND	Grazion log not found

Figura 12 - Protocolo de errores implementado

Apéndice C - APIs

User Service

Endpoint	Roles	Method	Request	Response
/api/user/v1/auth/login	*	POST	{ email: string, password: string }	LoginGrant
/api/user/v1/auth/refreshToken	*	POST	{ refresh_token: string }	LoginGrant
/api/user/v1/auth/verifyToken	*	GET		UserDto
/api/user/v1/users/	SUPER_ADMIN	POST	{ email: string, password: string, firstName: string, lastName: string, roles: string[] }	{ message: string }
/api/user/v1/users/{uid}	SUPER_ADMIN	PATCH	{ firstName?: string, lastName?:string, password?: string }	{ message: string }
/api/user/v1/users/{uid}/enabled	SUPER_ADMIN	PATCH	{ enabled: boolean }	{ message: string }
/api/user/v1/users/{uid}	*	PATCH	{ firstName?: string, lastName?:string, password?: string }	{ message: string }
/api/user/v1/dependants	OWNER, ADMIN	GET		{ message: string, data: DependantDTO[] }
/api/user/v1/dependants/	OWNER, ADMIN	POST	{ groupUid: string, email: string, password: string, firstName: string, lastName: string, roles: string[] }	{ message: string }
/api/user/v1/dependants/{uid}/roles	OWNER, ADMIN	PATCH	{ roles: string[] }	{ message: string }
/api/user/v1/dependants/{uid}/enabled	OWNER, ADMIN	PATCH	{ enabled: boolean }	{ message: string }

Figura 13 - API implementada para el *UserService*

Paddock Service

Endpoint	Roles	Method	Request	Response
/api/paddock/v1/paddocks	OWNER, ADMIN, WORKER, VIEWER	GET	query parameters: { pageNumber?: number, pageSize?: number}	{ data: Page<PaddockDto>, message: string }
/api/paddock/v1/paddocks/{uid}	OWNER, ADMIN, WORKER, VIEWER	GET		{ data: PaddockDto, message: string }
/api/paddock/v1/paddocks/{uid}	OWNER, ADMIN	DELETE		{ message: string }
/api/paddock/v1/paddocks/	OWNER, ADMIN	POST	PaddockDto	{ data: string (new paddock uid), message: string }
/api/paddock/v1/paddocks/{uid}	OWNER, ADMIN	PATCH	Partial<PaddockDto>	{ message: string }
/api/paddock/v1/calibrations	OWNER, ADMIN, WORKER, VIEWER	GET	query parameters: { pageNumber?: number, pageSize?: number}	{ data: Page<CalibrationDto>, message: string }
/api/paddock/v1/calibrations/{uid}	OWNER, ADMIN, WORKER, VIEWER	GET		{ data: CalibrationDto, message: string }
/api/paddock/v1/calibrations/{uid}	OWNER, ADMIN	DELETE		{ message: string }
/api/paddock/v1/calibrations/	OWNER,ADMIN,WORKER	POST	{ id: string, measurement: MeasurementDto }[]	{ message: string }
/api/paddock/v1/calibrations/{uid}	OWNER, ADMIN	PATCH	{ id: string, weight: number }[]	{ message: string }
/api/paddock/v1/measurements	OWNER, ADMIN, VIEWER	GET	query parameters: { pageNumber?: number, pageSize?: number}	{ data: Page<MeasurementDto>, message: string }
/api/paddock/v1/measurements/{uid}	OWNER, ADMIN,	GET		{ data: MeasurementDto, message: string }

	VIEWER			
/api/paddock/v1/measurements/	OWNER, ADMIN, WORKER	POST	MeasurementDto[]	{ message: string }
/api/paddock/v1/species	OWNER, ADMIN, WORKER, VIEWER	GET		{ data: SpeciesDto[], message: string }
/api/paddock/v1/species/{uid}	OWNER, ADMIN, WORKER, VIEWER	GET		{ data: SpeciesDto, message: string }
/api/paddock/v1/species/{uid}	OWNER, ADMIN	DELETE		{ message: string }
/api/paddock/v1/species/	OWNER, ADMIN	POST	SpeciesDto	{ data: string (new species uid), message: string }
/api/paddock/v1/species/{uid}	OWNER, ADMIN	PATCH	Partial<SpeciesDto>	{ message: string }
/api/paddock/v1/measurement-report	OWNER, ADMIN, VIEWER	GET	query parameters: { pageNumber?: number, pageSize?: number}	{ data: Page<MeasurementReportDto>, message: string }
/api/paddock/v1/measurement-report/{uid}	OWNER, ADMIN, VIEWER	GET		{ data: MeasurementReportDto, message: string }
/api/paddock/v1/measurement-report/	OWNER, ADMIN	POST	{ measurementsUid: string }	{ data: string (uid), message: string }
/api/paddock/v1/measurement-report/{uid}	OWNER, ADMIN	DELETE		{ message: string }

Figura 14 - API implementada para el *PaddockService*

Pasture Balance Service

Endpoint	Roles	Method	Request	Response
/api/paddock/v1/cattle	OWNER, ADMIN, VIEWER	GET		{ data: CattleDto[], message: string }
/api/paddock/v1/cattle/{uid}	OWNER, ADMIN, VIEWER	GET		{ data: CattleDto, message: string }
/api/paddock/v1/cattle/	OWNER, ADMIN	POST	CattleDto	{ data: string (new paddock uid), message: string }
/api/paddock/v1/cattle/{uid}	OWNER, ADMIN	PATCH	Partial<CattleDto>	{ message: string }
/api/paddock/v1/cattle/{uid}	OWNER, ADMIN	DELETE		{ data: string }
/api/paddock/v1/grazing-log	OWNER, ADMIN, VIEWER	GET		{ data: GrazingLogResponse[], message: string }
/api/paddock/v1/grazing-log/{uid}	OWNER, ADMIN, VIEWER	GET		{ data: GrazingLogResponse, message: string }
/api/paddock/v1/grazing-log/	OWNER, ADMIN	POST	GrazingLogDto	{ data: string (new grazing_log uid), message: string }
/api/paddock/v1/grazing-log/{uid}	OWNER, ADMIN	DELETE		{ data: string }
/api/paddock/v1/stats/stock	OWNER, ADMIN, VIEWER	GET		{ data: StockResponse, message: string }
/api/paddock/v1/stats/stock-by-paddock	OWNER, ADMIN, VIEWER	GET		{ data: StockByPaddockResponse[], message: string }
/api/paddock/v1/stats/demand	OWNER, ADMIN, VIEWER	GET		{ data: DemandResponse, message: string }
/api/paddock/v1/stats/offer	OWNER, ADMIN, VIEWER	GET		{ data: OfferResponse, message: string }
/api/paddock/v1/stats/growth-rate	OWNER, ADMIN, VIEWER	GET		{ data: GrowthRateResponse, message: string }
/api/paddock/v1/stats/growth-rate-by-species	OWNER, ADMIN, VIEWER	GET		{ data: GrowthRateBySpeciesResponse[], message: string }
/api/paddock/v1/stats/growth-rate-by-paddock	OWNER, ADMIN, VIEWER	GET		{ data: GrowthRateByPaddockResponse[], message: string }

Figura 15 - API implementada para el *PastureBalanceService*

Referencias

Name	Description
Page<T>	{ content: T[], totalPages: number, totalElements: number, pageNumber: number }
MeasurementDTO	{ height: float, location: GeoJSON Point, timestamp: string }
UserDto	{ uid: string, name:string, email: string, roles: string[], groupUid: string }
PaddockDto	{ uid: string, name: string, groupUid: string, speciesUid: string, calibrationUid: string, geofence: GeoJSON Polygon }
CalibrationDto	{ uid: string, measurements: { id: string, measurement: MeasurementDto, weight: number }, curve: number[], timestamp: string }
SpeciesDto	{ uid: string, name: string, calibrationsUid: string[] }
LoginGrant	{ access_token: string, expires_in: int, refresh_expires_in: int, refresh_token: string }
CattleDto	{ uid: string, groupUid?:string, name: string, count:number, dailyIntakePerAnimal: number, totalDailyIntake: number, createdAt: string }
MeasurementReportDto	{ uid: string, paddockUid: string, measurementUid: string, status: string, stock?: number, measurements?: { weight: number, location: GeoJSON Point, timestamp: string }[], createdAt: string }

Figura 16 - Referencias a interfaces mencionadas dentro de las API previas

Apéndice D - Diagrama entidad-relación

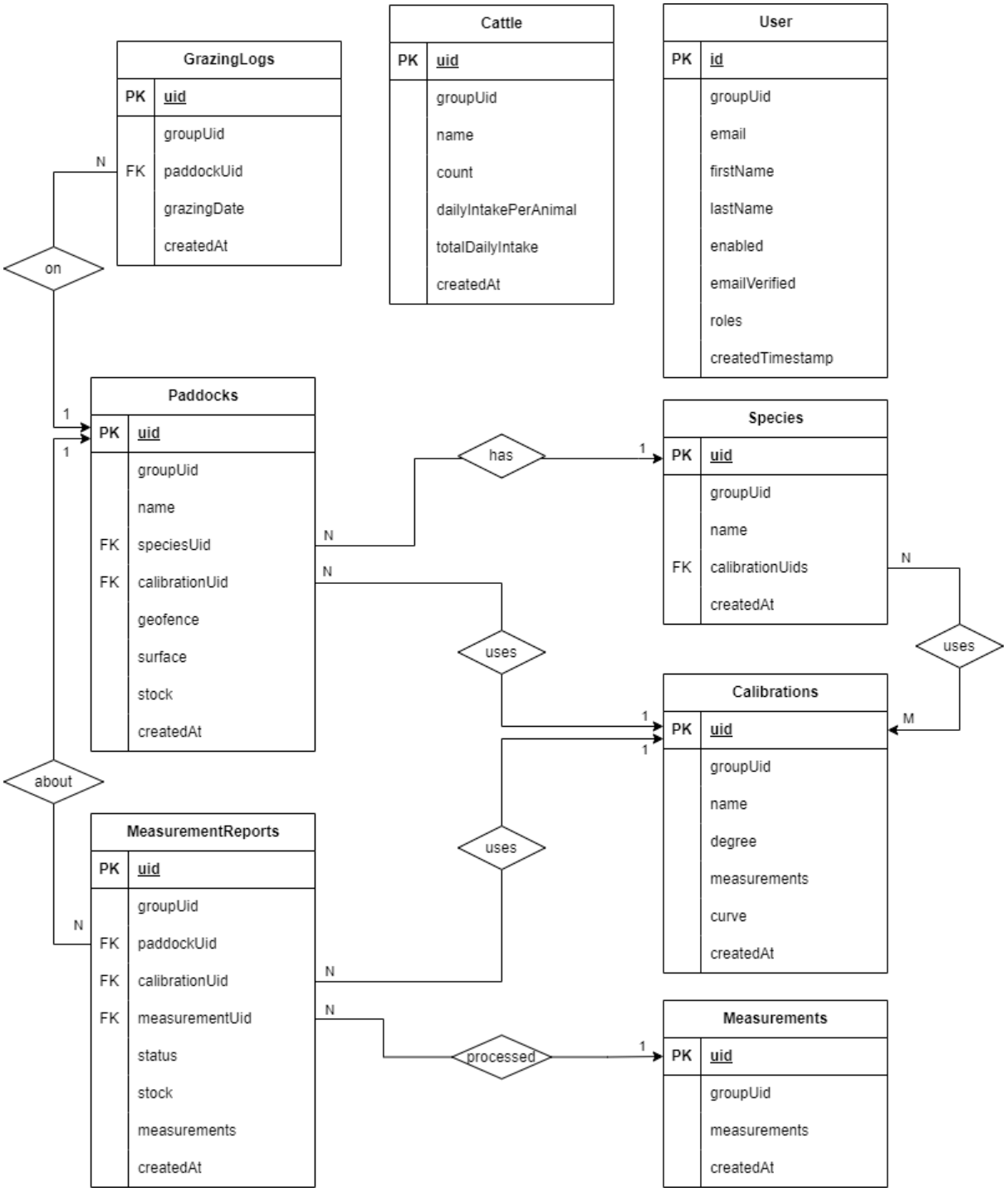

Figura 17 - Diagrama entidad-relación

Bibliografía

1. Desiré, S.; Emilce, T (2021). La importancia de la ganadería para la economía argentina. Bolsa de Comercio de Rosario.

2. Fulkerson, W. J., McKean, K., Nandra, K. S., & Barchia, I. M. (2005). Benefits of accurately allocating feed on a daily basis to dairy cows grazing pasture. Australian Journal of Experimental Agriculture.

3. Web oficial GitHub [en línea]. GitHub. Consultado el 11 de agosto del 2023 en https://github.com/

4. Web oficial Todoist [en línea]. Todoist. Consultado el 11 de agosto del 2023 en https://todoist.com/es

5. Web oficial ESlint [en línea]. ESlint. Consultado el primero de agosto del 2023 en https://eslint.org/

6. Estilo de arquitectura de microservicios [en línea]. Microsoft. Consultado el primero de julio del 2023 en https://learn.microsoft.com/es-es/azure/architecture/guide/architecture-styles/microservices

7. Patrón Backends for Frontends [en línea]. Microsoft. Consultado el dos de julio del 2023 en https://learn.microsoft.com/es-es/azure/architecture/patterns/backends-for-frontends

8. Web oficial de Leaflet [en línea]. Leaflet. Consultado el dos de julio del 2023 en https://leafletjs.com/

9. Web oficial de OpenStreetMap [en línea]. Consultado el dos de julio del 2023 en https://www.openstreetmap.org/

10. Web oficial de Google Maps [en línea]. Consultado el dos de julio del 2023 en https://mapsplatform.google.com/pricing/

11. Web oficial de MapBox. MapBox [en línea]. Consultado el dos de julio del 2023 en https://www.mapbox.com/pricing

12. Web oficial de Typescript [en línea]. Typescript Lang. Consultado el tres de julio del 2023 en https://www.typescriptlang.org/

13. Web oficial de MongoDB [en línea]. MongoDB. Consultado el 22 de junio del 2023 en https://www.mongodb.com/es

14. Web oficial de Keycloak [en línea]. Keycloak. Consultado el 23 de junio del 2023 en https://www.keycloak.org/

15. Web oficial de Firebase Authentication [en línea]. Firebase. Consultado el 30 de junio del 2023 en https://firebase.google.com/docs/auth

16. Web oficial Postman [en línea]. Postman. Consultado el primero de agosto del 2023 en https://www.postman.com/

17. Web oficial de Pasture.io [en línea]. Pasture.io. Consultado el dos de agosto del 2023 en https://pasture.io/

18. Sobre Kelpie [en línea]. Kelpie. Consultado el 22 de julio del 2023 en http://kelpie.com.ar/sobre-kelpie/

19. ¿Qué es una API y cómo funciona? [en línea]. Red Hat. Consultado el primero de julio del 2023 en https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces

20. HTTP [en línea]. MDN. Consultado el primero de julio del 2023 en https://developer.mozilla.org/es/docs/Web/HTTP

21. Introducción a JSON [en línea]. JSON.org. Consultado el primero de julio del 2023 en https://www.json.org/json-es.html

22. ¿Qué es NOSQL? [en línea]. Oracle. Consultado el primero de julio del 2023 en https://www.oracle.com/ar/database/nosql/what-is-nosql/

23. ¿Qué es una API de REST? [en línea]. Red Hat. Consultado el 30 de junio del 2023 en https://www.redhat.com/es/topics/api/what-is-a-rest-api

24. Superset [en línea]. The Free Dictionary. Consultado el ocho de septiembre del 2023 en https://encyclopedia2.thefreedictionary.com/superset

25. ¿Qué es una transacción? [en línea]. Microsoft. Consultado el 14 de agosto del 2023 en https://learn.microsoft.com/es-es/windows/win32/ktm/what-is-a-transaction

26. Valor bruto de la produccion [en línea]. Conogasi. Consultado el 11 de agosto del 2023 en https://conogasi.org/t%C3%A9rminos/valor-bruto-de-la-produccion/