Politecnico di Torino

Politecnico di Torino

Ingegneria Informatica

A.a. 2025/2026

Sessione di laurea Marzo 2026

Realizzazione di un Decision Support System per il benessere animale

Relatore: Luigi De Russis

Candidato: Giancarlo Virga

Indice

Elenco delle figure IV

1 Introduzione 1
- 1.1 Contesto . . . 1
- 1.2 Obiettivo della tesi . . . 2
  - 1.2.1 Soluzione proposta . . . 2
  - 1.2.2 Contesto Operativo: Bando SWIch . . . 3
  - 1.2.3 Inquadramento tecnico del progetto . . . 3

2 Metodologia 5
- 2.1 Studio del Dominio, Identificazione degli Stakeholders e Needfinding 5
  - 2.1.1 Dominio e Contestualizzazione del Sito Operativo . . . 5
  - 2.1.2 Identificazione degli Stakeholders . . . 8
  - 2.1.3 Needfinding . . . 9
- 2.2 Raccolta Requisititi . . . 10
  - 2.2.1 Interviste . . . 10
  - 2.2.2 Requisiti . . . 11
  - 2.2.3 Definizione del Processo "To-Be" . . . 12
- 2.3 Analisi della Complessità e Prioritizzazione . . . 13
  - 2.3.1 Valutazione dei Vincoli Tecnici . . . 13
  - 2.3.2 Analisi della complessità . . . 14
  - 2.3.3 Prioritizzazione Strategica . . . 15
  - 2.3.4 Applicazione del Metodo MoSCoW . . . 16
  - 2.3.5 Definizione del Minimum Viable Product (MVP) . . . 18
- 2.4 Development Lifecycle e Specifica Formale . . . 19
  - 2.4.1 Approccio Metodologico . . . 19
  - 2.4.2 Definizione dei Requisiti Funzionali (FR) . . . 20
  - 2.4.3 Definizione dei Requisiti Non Funzionali (NFR) . . . 21

3 Progettazione Architetturale e Backend 22
3.1 Dai Requisiti al Design Architetturale . . . 22
3.1.1 Specifica dei Requisiti Funzionali (RF) . . . 22
3.1.2 Specifica dei Requisiti Non Funzionali (RNF) . . . 24
3.1.3 Architettura di Sistema ad Alto Livello: Design Driven by Requirements . . . 25
3.2 Il Backend, lo Storage e la Sfida dell'Asincronia . . . 28
3.2.1 Ingestione e Streaming Video (Storage-Service) . . . 28
3.2.2 Orchestrazione AI e il Pattern "Docker Volume" (s3fs) . . . 30
3.2.3 Elaborazione Risultati: Claim-Check Pattern e Redis Cache 31
3.3 Livello Dati: Modellazione Ibrida e *Data Fusion* . . . 32
3.3.1 Il Problema: Volumetria, Eterogeneità e Frequenza dei Dati 32
3.3.2 La Soluzione: Persistenza Ibrida (PostgreSQL e S3) . . . 33
3.3.3 La *Data Fusion* e l'Ottimizzazione tramite Redis Cache . . . 33

4 Livello di Presentazione e Progettazione UX/UI 35
4.1 Il Problema: Carico Cognitivo e Complessità del Dato . . . 35
4.2 Metodologia di Design Iterativo (HCI) e Prototipazione . . . 36
4.2.1 Low-Fidelity Prototype . . . 36
4.2.2 Mid-Fidelity Prototype . . . 38
4.2.3 High-Fidelity Prototype . . . 40

5 Futura valutazione della UI 42
5.1 Pianificazione . . . 42
5.2 Metodologia di Valutazione della Dashboard . . . 43
5.3 Metriche di Risultato e Analisi Euristica . . . 44
5.4 Questionario SUS (System Usability Scale) . . . 45

6 Conclusioni 47
6.1 Traguardi Raggiunti . . . 47
6.2 Sviluppi Futuri . . . 48

A Script per la Valutazione del Prototipo 49

B Questionario System Usability Scale (SUS) 51

Bibliografia 54

Elenco delle figure

2.1 Planimetria del sito di studio. 6
2.2 Lely Astronaut 7
2.3 Estratto dei log grezzi esportati dal sistema di mungitura Lely Astronaut. 8
2.4 Estratto dell'output generato dal modello di Computer Vision. . . . 13

3.1 Diagramma architetturale ad alto livello del sistema SIMBA. Sono evidenziati il livello di *routing* (Spring Cloud Gateway), i microservizi di dominio (Java Spring Boot), il *layer* di persistenza (PostgreSQL e S3/MinIO), l'infrastruttura Redis (Code e *Cache*) e il modulo di elaborazione AI (Python YOLO). 27
3.2 *Sequence diagram* del flusso di *upload*. Il video viene caricato sull'Object Storage (S3) e i metadati relazionali salvati in PostgreSQL. 29
3.3 *Sequence diagram* del flusso di *streaming chunked*. Il frontend riceve il video a porzioni, evitando il sovraccarico della RAM. 29
3.4 *Sequence diagram* dell'avvio del Job AI. Evidenzia l'orchestrazione asincrona e l'accesso diretto al *filesystem* tramite Docker Volume. . 30
3.5 *Sequence diagram* dell'elaborazione dei risultati. L'`analysis-service` consuma la coda Redis, legge i CSV massivi direttamente da S3 e popola la *Cache*. 31

4.1 Evoluzione del *wireframe* Lo-Fi. In questa fase ci si concentra esclusivamente sulla disposizione spaziale dei macro-componenti e sul raggruppamento in "Progetti" 37
4.2 Viste di dettaglio della *dashboard* clinica. 39
4.3 Prototipi Hi-Fi dell'interfaccia utente (schermate di *Login* e di Gestione). 41

Capitolo 1

Introduzione

1.1 Contesto

Negli ultimi decenni, il settore agroalimentare e zootecnico ha vissuto un percorso di transizione digitale, muovendosi in direzione dell'industria 4.0. In questa evoluzione, in primo piano si colloca il **Precision Livestock Farming (PLF)**, ovvero la zootecnia di precisione. L'obiettivo cardine del PLC è l'applicazione delle tecnologie dell'informazione e della comunicazione (ICT) per permettere un monitoraggio continuo, automatico, e individualizzato dei capi di bestiame all'interno degli allevamenti.

Nel contesto dell'allevamento di bovini da latte, il monitoraggio tecnologico permette non soltanto di ridurre il costo di monitoraggio delle strutture, ma permette anche il controllo costante di mandrie di dimensioni sempre maggiori. Parallelamente, l'ottenimento di dati da parte di questi sistemi ha permesso lo studio del **benessere animale**. A tal propostito la letteratura zootecnica impone una distinzione concettuale tra *Animal Welfare* inteso come garanzia delle cure di base e il rispetto dei bisogni primari dell'animale, e *Animal Wellbeing*, che definisce inceve l'equilibrio psico-fisico del bovino.

Questo interesse non è soltanto relativo a questioni etiche, ma di profitto. Una vacca in salute, che riposa bene, in un ambiente controllato e privo di fattori di stress ambientali o gerarchici, massimizza la produzione di latte in termini sia qualitativi che quantitativi, aumentando i ricavi ed abbattendo i costi legati agli interventi veterinari e all'uso di antibiotici.

Per rispondere all'esigenza, le stalle a stabulazione libera, si sono rapidamente adeguate ai tempi, dotandosi di dispositivi all'avanguardia: sensori, telecamere di videosorveglianza perimetrali, avanzati sistemi di mungitura. Tutti questi dispositivi permettono di ricavare dati spaziali, fisiologici e clinici legati al bovino, i quali diventano una preziosa risorsa per allevatori e veterinari.

La difficoltà di accesso ad una semplice lettura di questi dati frammentati si pone tuttavia come barriera in questo contesto, impedendo ad addetti del settore un pratico accesso alle informazioni

In questo processo frammentato, lento e soggetto ad errore, si inserisce SIMBA, con l'intenzione di fornire un punto di accesso unificato alle informazioni. Il sistema permette di estrarre conoscenza derivata dai dati grezzi e tradurli in indicatori clinici. Così facendo SIMBA si presenta come un vero e proprio **Decision Support System**, capace di abbattere il carico cognitivo degli operatori e guidarli verso il riconoscimento di anomalie comportamentali o insorgenze di problemi clinici.

1.2 Obiettivo della tesi

L'obiettivo principale dell'elaborato non si limita esclusivamente alla descrizione di quanto prodotto, bensì si pone con lo scopo di descrivere l'intero processo di **progettazione architetturale e analisi funzionale** che porta alla creazione di una piattaforma software di supporto decisionale (DSS - Decision Support System) in un contesto zootecnico reale.

Il lavoro svolto ripercorre metodologie dell'ingegneria del Software applicate per trasformare un problema di business in una soluzione tecnologica viabile.

1.2.1 Soluzione Proposta

Per rispondere alle criticità legate alla frammentazione dei dati, la soluzione proposta si articola attraverso la progettazione di un DSS in ambiente cloud. La piattaforma è stata studiata in modo tale da offrire all'utente la possibilità di aggregare dati e permettere la loro elaborazione, traducendo poi i dati in informazioni fruibili dall'utilizzatore.

La soluzione proposta si fonda su tre pilastri architetturali e funzionali:

- **Integrazione ed Eterogeneità:** Il sistema è progettato per fondere in un unico punto dati eterogenei provenienti dai dispositivi installati in loco. SIMBA, unendo flussi video e dati clinici derivati dall'analisi dei dati in output dal robot di mungitura, fornisce all'utente anomalie comportamentali e parametri fisiologici.

- **Architettura Cloud e Asincronia:** Per far fronte all'ingestione di pesanti flussi multimediali, SIMBA permette il disaccoppiamento tra il caricamento e l'elaborazione video, garantendo scalabilità tramite un'infrastruttura distribuita a microservizi.

- ***Data Extraction e Visualization*:** Al fine di fornire all'utente informazioni facilmente fruibili, SIMBA estrae dati eterogenei e li trasforma visivamente, fornendo all'utente grafi, mappe e tabelle facilmente interpretabili.

In sintesi, la soluzione proposta permette di trasformare la stalla in un ecosistema intelligente, restituendo all'allevatore il controllo sulla mandria attraverso i dati fornitigli. Tramite SIMBA l'operatore viene supportato nelle decisioni e nella tempestiva identificazione di criticità, massimizzando il benessere animale e ottimizzando l'intero ciclo produttivo.

1.2.2 Contesto Operativo: Bando SWIch

La realizzazione di SIMBA è avvenuta all'interno di una iniziativa di ricerca industriale co-finanziata dalla Regione Piemonte, tramite il **Bando SWIch**. L'obiettivo esplicito del bando regionale consiste nel favorire il trasferimento tecnologico da un ambiente di laboratorio ad un ambiente operativo reale. Per soddisfare questo requisito e garantire la validazione scientifica e informatica del prodotto, il progetto è stato condotto da un consorzio multidisciplinare di eccellenza, all'interno del quale il candidato ha operato attivamente. I tre attori principali del consorzio sono stati:

- **Interlogic SRL (Capofila Tecnologico):**Azienda informatica responsabile dell'intera Ingegneria del Software, della progettazione dell'infrastruttura in cloud e dello sviluppo del Decision Support System, ambiente nel quale è stato svolto il lavoro di analisi, architettura e sviluppo descritto in questa tesi.
- **Università di Torino - Dipartimento di Scienze Veterinarie (Organismo di Ricerca):** L'organismo di ricerca che ha messo a disposizione la sua conoscenza del dominio clinico ed etologico. È responsabile della definizione dei parametri significativi per il benessere dei bovini. È inoltre responsabile della realizzazione dell'algoritmo di Computer Vision, necessario per il tracciamento spaziale dei bovini.
- **Azienda Agricola Vanzetti (End User):** Realtà zootecnica d'eccellenza che ha messo a disposizione il sito di studio, oltre che le proprie infrastrutture hardware (telecamere, robot di mungitura Lely) e la propria mandria per l'acquisizione dei dati sul campo.

1.2.3 Inquadramento tecnico del progetto

Per comprendere la natura del lavoro svolto, è necessario delineare il contesto in cui il progetto SIMBA ha preso forma.

Il sistema si avvale di un avanzato modello di Intelligenza Artificiale per il tracciamento visivo (Computer Vision), necessario per l'estrazione di coordinate spaziali a partire dai video del sistema di videosorveglianza installato in loco. Lo sviluppo e l'addestramento di tale rete neurale sono stati condotti interamente dai ricercatori del Dipartimento di Scienze Veterinarie dell'Università di Torino (UniTo). All'interno dell'architettura software, questo modello è stato integrato ed orchestrato sotto forma di componente *Black-Box* esterna.

Il contributo tecnico e scientifico di questo elaborato risiede nell'intero processo di ingegnerizzazione e sviluppo del software. Nello specifico, il lavoro di tesi si articola attorno a tre temi principali:

- **Analisi Funzionale e Ingegneria dei Requisiti:** Il contributo primario è consistito nel fungere da ponte tra gli esperti di dominio e il prodotto software. Questa fase ha richiesto un'intensa attività di modellazione per tradurre concetti clinici ed etologici complessi in logiche matematiche computabili. Attraverso analisi dei requisiti e prioritizzazione, è stato definito il perimetro del *Minimum Viable Product* (MVP).

- **Progettazione dell'Architettura di Sistema (Backend e Dati):** Il cuore ingegneristico ha riguardato l'ideazione dell'ecosistema applicativo *Cloud-Native*, necessario per orchestrare l'ingestione massiva di dati. Per gestire il peso dei file video in 4K, l'architettura è stata progettata attorno a logiche di asincronia, sfruttando un *Object Storage* isolato. Parallelamente, sono state affrontate le problematiche legate al database e alla gestione dei dati in output, per sincronizzare algoritmicamente le coordinate spaziali fornite dall'AI con le metriche cliniche di mungitura estratte dal robot Lely.

- **Sviluppo del Livello di Presentazione (Frontend):** L'ultimo pilastro ha riguardato la progettazione e l'implementazione dell'interfaccia utente. Strutturata come *Single Page Application* (SPA) in React, l'interfaccia è stata concepita specificamente per abbattere il carico cognitivo dell'allevatore. Il lavoro si è concentrato sulla traduzione della complessità dei dati moduli intuitivi, sviluppando mappe di calore (*Heatmap*) per l'occupazione spaziale, tracciamenti dinamici in 2D e indicatori a semaforo per segnalare le criticità sanitarie.

Capitolo 2

Metodologia

La realizzazione di un sistema software complesso in un contesto di ricerca industriale non può prescindere dall'applicazione di un rigoroso framework metodologico. Per affrontare lo sviluppo di SIMBA è stato seguito un rigoroso processo di analisi guidato dai principi cardine dell'ingegneria dei requisiti (Requirement Engineering).

2.1 Studio del Dominio, Identificazione degli Stakeholders e Needfinding

Il primo passo dell'analisi si è concentrato sulla fase di scoperta. Per definire i requisiti e le funzionalità del software è infatti necessaria la piena comprensione di ciò che si vuole descrivere, in modo tale da mappare interessi e problemi da risolvere tramite la successiva implementazione.

2.1.1 Dominio e Contestualizzazione del Sito Operativo

Ispirandosi ai concetti del Domain-Driven Design (DDD), l'analisi è partita dall'assimilazione del dominio. Ciò ha permesso di presentarsi agli incontri con gli esperti di settore con la possibilità di utilizzare un vocabolario comune. Lo studio del dominio si è diviso in due parti: Lo studio dei comportamenti bovini e lo studio del sito di test (in questo caso le stalle a stablulazione libera dell'Azienda Agricola Vanzetti Holstein).

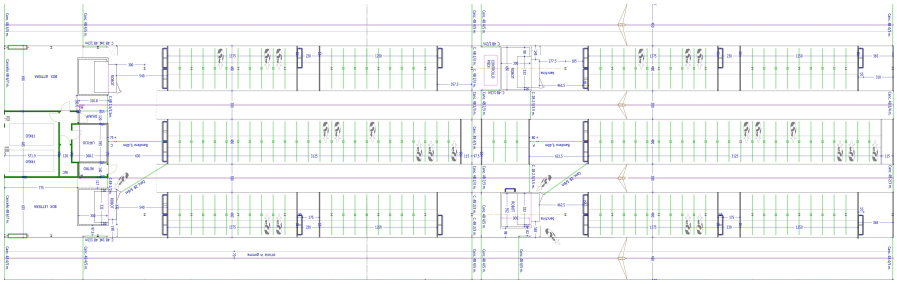

Figura 2.1: Planimetria del sito di studio.

- **Habitat e le dinamiche spaziali**: È stato analizzato un modello a "stabulazione libera", ovvero un ambiente all'interno della quel i bovini non sono legati, ma si muovono autonomamente, La comprensione degli spazi è stata fondamentale per mappare le coordinate digitali. La stalla è infatti divisa in macro aree interconnesse: la zona bivacco (costituita da cuccette per il riposo), le corsie di camminamento e la zona di alimentazione (dove il cibo è accessibile volontariamente dal bovino in qualsiasi momento) .

- **Etologia, Riposo, Ruminazione e Socialità:** Dal punto di vista comportamentale, il bovino è un animale fortemente gregario. Nello specifico, il dominio applicativo del progetto si concentra su capi di razza *Holstein* (la caratteristica vacca pezzata, nota in Italia come Frisona). Il sistema software doveva tenere conto del fatto che i tempi di riposo (idealmente 12-14 ore al giorno) sono direttamente correlati alla produzione di latte. Le dinamiche sociali, influenzate da leadership e stress da competizione, guidano l'accesso alle risorse alimentari e alle zone di riposo.

- **Sistemi di Mungitura (Lely Astronaut):** L'infrastruttura di stalla è dotata di un sistema di mungitura automatizzata (AMS - *Automated Milking System*), nello specifico il modello Astronaut, prodotto da Lely [1]. L'accesso a tale postazione avviene su base strettamente volontaria: l'animale viene infatti incentivato a recarsi autonomamente al robot tramite l'erogazione mirata di mangime concentrato, che funge da stimolo alimentare [2]. Ogni interazione dell'animale con il robot gerena una vasta mole di dati tabellari che tracciano metriche cliniche e produttive quali: frequenza di ingresso, produzione di latte, tassi di rifiuto ed altro. L'estrazione e la valorizzazione di questi registri assumono un'importanza strategica che travalica il semplice monitoraggio aziendale: esiste infatti un forte potenziale di interesse da parte di aziende come Lely nell'integrazione dei propri robot con ecosistemi come quello di SIMBA

Figura 2.2: Lely Astronaut

L'ingresso dell'animale nella postazione attiva un processo di identificazione automatizzata, tramite un sensore RFID presente nel marchio auricolare di ogni capo . Questa scansione permette al sistema di riconoscere in modo univoco l'animale .

Una volta stabilita l'identità della vacca, il software del robot avvia il tracciamento della sessione di mungitura. Tutti i parametri fisiologici e produttivi rilevati durante la mungitura (come la quantità di latte munto, la conducibilità elettrica per quarto, la velocità di flusso e gli indici di rischio mastite) vengono registrati e associati all'ID dell'animale. Questo meccanismo genera quotidianamente un file in formato CSV con i dati delle mungiture effettuate.

```
Animale_ID  Robot_ID  Session_UUID                          Inizio_Mungitura          Fine_Mungitura
        Rit(h)  Stato              Prod(kg) Cond_1 Cond_2 Cond_3 Cond_4
1954        102       EBDE1B97-CB30-4A07-BA0F-FBECF20661FD  2025-11-23T21:43:31.000Z  2025-11-23T21:52:05.000
     Z  7       Successful milking  10       64     63     64     63     ...
1397        104       F8F2E933-51B2-4B4A-99B5-5091828085FB  2025-11-24T10:58:17.000Z  2025-11-24T11:04:17.000
     Z  10.5    Successful milking  16.1     70     68     69     70     ...
1788        102       BCE095DC-7ADF-4877-AC37-9370F41F5DD5  2025-11-23T05:21:38.000Z  2025-11-23T05:32:25.000
     Z  5       Successful milking  23.5     72     72     72     70     ...
1873        102       23D4EE56-1B29-4B39-B0BA-4401491996C5  2025-11-19T01:45:53.000Z  2025-11-19T01:46:01.000
     Z  9       Refused milking     null   null   null   null   null   ...
1840        101       FF9E94BC-77BE-4CD1-BD7B-4CBFEC422847  2025-11-17T08:28:46.000Z  2025-11-17T08:33:01.000
     Z  0.1     Successful milking  12.2     66     64     64     64     ...
```

Figura 2.3: Estratto dei log grezzi esportati dal sistema di mungitura Lely Astronaut.

- **Le Sfide Cliniche (Mastiti e Zoppie):** Lo studio del dominio si è focalizzato sulla comprensione delle due patologie primarie che affliggono le mandrie da latte. Le **mastiti** sono infiammazioni della ghiandola mammaria, generalmente di origine batterica (ambientale o contagiosa) , che si diffondono per contatto diretto o tramite le attrezzature della stalla; esse causano dolore acuto e un drastico calo nella produzione e qualità del latte . Le **zoppie** (*lameness*) sono invece dolorose affezioni podali o articolari , provocate da stress meccanici prolungati, pavimentazioni inadeguate o cause metaboliche. L'insorgenza della patologia altera profondamente il comportamento dell'animale: il dolore inibisce la normale locomozione, costringendo il bovino ad aumentare i tempi di decubito e riducendo drasticamente il suo accesso volontario alla zona di alimentazione.

2.1.2 Identificazione degli Stakeholders

Acquisita padronanza del dominio fisico, il passo successivo è stato procedere con la mappatura degli **stakeholders**. Lo sviluppo di un software è infatti il prodotto della mediazione tra entità con obiettivi, background e metriche profondamente differenti. Nel contesto del progetto SIMBA, co-finanziato dal bando regionale, l'ecosistema ruota attorno a 4 quattro attori principali, tutti interessi comuni e obbiettivi distinti:

- **Il Dipartimento di Scienze Veterinarie (UniTo):**Rappresentano l'anima scientifica e metodologica del progetto . Il loro interesse primario è il rigore analitico e la validazione formale delle tesi etologiche. I ricercatori di UniTo necessitano di un sistema capace di elaborare calcoli complessi, tracciare grafi comportamentali ed esportare moli massicce di dati grezzi (strutturati in file CSV) per condurre successive indagini statistiche indipendenti.

- **L'Azienda Agricola Vanzetti-Holstein:**Rappresentano il committente sul campo e il caso di studio reale. Il loro ruolo è fondamentale per soddisfare le direttive del bando regionale: validare la tecnologia in un ambiente operativo per innalzarne il Technology Readiness Level (TRL). Il loro interesse è focalizzato esclusivamente sull'usabilità pratica e sul ritorno d'investimento operativo. Necessitano di un supporto decisionale rapido (es. avvisi a semaforo per il rischio mastite), fruibile in mobilità senza dover interpretare tabelle complesse durante il faticoso lavoro in stalla. Inoltre, l'azienda ha imposto il vincolo operativo più stringente del progetto: il principio della "zero interferenza". Hanno escluso categoricamente l'adozione di dispositivi **wearable** (collari o podometri IoT), obbligando il team a sviluppare una soluzione di monitoraggio totalmente passiva basata sulla **Computer Vision**.

- **InterLogic S.R.L.** È l'azienda fornitrice del software e capofila tecnologico del progetto. Il suo interesse non si limita al mero rilascio del codice nei tempi previsti, ma abbraccia la "valorizzazione economica dell'innovazione". ProLogic punta a creare un **Minimum Viable Product** (MVP) dotato di un'architettura modulare e scalabile, capace di gestire l'elaborazione video asincrona. L'obiettivo strategico è acquisire un **know-how** specialistico e un **framework** riutilizzabile per posizionarsi come eccellenza nel nascente mercato dell'AgriTech, trasformando una ricerca finanziata in un futuro prodotto commerciale.

- **Lely Industries N.V.**Azienda Olandese produttrice dei robot di mungitura automatizzata Astronaut. Pur non essendo un utente diretto che interagisce con il progetto SIMBA, Lely rappresenta uno stakeholder infrastrutturale e di dominio critico. L'interesse in gioco in questo caso è l'interoperabilità: il successo clinico del software dipende dalla capacità di interfacciarsi indirettamente con i loro sistemi chiusi, al fine di estrarre in modo affidabile i log fisiologici necessari per alimentare gli algoritmi del Decision Support System.

2.1.3 Needfinding

Identificati gli attori e il perimetro del dominio, la fase successiva è stata quella di ricerca dei bisogni (**Needfinding**) . In questa fase l'obiettivo è stato quello di fare emergere le necessità degli stakeholders, non curandosi in questa fase della fattibilità. Nell'Ingegneria dei Requisiti è noto che i committenti raramente si esprimono in termini tecnici riguardo a ciò di cui hanno realmente bisogno; spesso propongono soluzioni pre-confezionate ("vorrei un sito per vedere le telecamere") invece di descrivere il problema alla radice.

Attraverso i primi tavoli di confronto e l'osservazione indiretta dei processi "As-Is", oltre a tener conto richiesta fatte, è stata eseguita una indagine sulla necessità di

una facile fruizione dei dati frammentati già disponibili. Il monitoraggio tradizionale soffre infatti di due grandi limitazioni:

- Da un lato l'osservazione umana diretta necessaria per il monitoraggio comportamentale si è rivelata di base empirica, ma soprattutto non scalabile su mandrie di grandi dimensioni, in quanto risulta impossibile il monitoraggio costante durante tutto l'arco della gionata.
- D'altro canto la difficoltà di trovare un nesso tra eventi apparentemente scollegati, come i dati del robot di mungitura, in quanto privi del contesto spaziale relativo all'animale.

Il vero bisogno latente, emerso da questo processo di analisi, consiste quindi nell'**offrire all'operatore dati che correlassero tra loro eventi apparentemente scollegati**. Il software richiesto non deve quindi limitarsi ad essere un visualizzatore di report estratti dall'Intelligenza Artificiale, ma agire da ponte. La necessità è quindi quella di unire flussi di dati provenienti da fonti diverse e convogliarle in un unico punto.

2.2 Raccolta Requisiti

Il passaggio logico successivo alla comprensione del dominio è l'elicitazione attiva dei bisogni. In questa fase del progetto, l'approccio metodologico adottato è stato volutamente divergente: l'obiettivo primario era raccogliere il maggior numero di informazioni, aspettative e visioni cliniche senza applicare alcun filtro ingegneristico preventivo. Da un punto di vista operativo l'intero processo è stato documentato su fogli di testo in modo tale da produrre documentazione legata al progetto.

Lo scopo era far emergere la pura necessità di dominio e il reale potenziale di ricerca. Pertanto, lo studio della reale fattibilità algoritmica e dei *trade-off* implementativi è stato strutturalmente posticipato a una fase successiva e dedicata, garantendo così ai ricercatori la massima libertà esplorativa durante la mappatura degli obiettivi scientifici.

2.2.1 Interviste

Il nucleo della raccolta informativa si è basato su una serie di interviste semi-strutturate che hanno visto confrontarsi in modo diretto Interlogic SRL e i ricercatori del Dipartimento di Scienze Veterinarie (UniTo).

È fondamentale precisare che l'Azienda Agricola Vanzetti non ha partecipato attivamente a questi tavoli di progettazione logica. Il partner agricolo aveva infatti già definito a monte il proprio ruolo di fornitore del sito di test, ponendo un unico

pre-requisito infrastrutturale: il divieto assoluto di installare hardware, sensori o cavi all'interno degli spazi vitali della stalla o direttamente sugli animali (il principio di "zero interferenza").

Di conseguenza, i tavoli tecnici con UniTo si sono concentrati in via esclusiva su **cosa gli esperti volessero concretamente studiare riguardo ai bovini**. L'obiettivo di Interlogic è stato ascoltare, comprendere e mappare i bisogni etologici. Durante gli incontri, ai medici veterinari è stato chiesto di descrivere quali parametri ritenessero vitali per valutare il benessere della mandria.

Sono emerse tematiche di studio di altissimo valore clinico: i ricercatori hanno espresso la forte volontà di indagare la frequenza e la durata del riposo, le dinamiche di dominanza nell'accesso al cibo e i primissimi segnali di isolamento sociale (spesso sintomo di patologie sub-cliniche). Spingendosi nei dettagli, gli etologi hanno manifestato l'interesse di voler studiare la postura esatta dell'animale (distinguendo tra vacca eretta o sdraiata), l'orientamento spaziale del corpo e le interazioni sociali di micro-dettaglio, come il reciproco annusamento (contatto frontale "muso-muso"). Tutto questo prezioso bacino di idee comportamentali è stato meticolosamente registrato per essere successivamente analizzato.

2.2.2 Requisiti

Le complesse richieste etologiche emerse durante le interviste con i ricercatori si possono raggruppare in tre macro-categorie di studio:

- **Monitoraggio del Riposo tramite Occupazione Spaziale:**
  - **Frequenza e durata del decubito:** Impossibilitato a determinare la postura esatta, il sistema deve dedurre il tempo di riposo (target 12-14 ore) calcolando l'intersezione continua tra le coordinate del centroide del bovino e i poligoni statici pre-calibrati che delimitano le cuccette sulla mappa.

- **Analisi di Prossimità e Dinamiche di Gruppo:**
  - **Social Network Analysis (SNA):** Per sopperire all'assenza del riconoscimento direzionale ("muso-muso"), il sistema deve tracciare le interazioni sociali calcolando la distanza euclidea tra i centroidi nel tempo. Questo *workaround* permette di mappare la rete di contatti della mandria basandosi sulla pura prossimità spaziale.
  - **Segnali di isolamento sociale:** Identificare i capi che mantengono costantemente distanze elevate dal baricentro del gruppo, evidenziando comportamenti anomali o stati depressivi.

  - **Dinamiche di dominanza in mangiatoia:** Sfruttando la prossimità all'interno dell'area di alimentazione, il software deve identificare le gerarchie di accesso al cibo e gli eventuali capi sottomessi che vengono allontanati dalla corsia.

- **Data Fusion e Correlazioni Cliniche:**

  - **Incrocio clinico-comportamentale (Mastiti e Zoppie):** Il requisito cardine del sistema consiste nell'incrociare i dati di mobilità (chilometri percorsi, tempi di riposo prolungati e cali di attività generale) con i dati fisiologici estratti dal robot Lely (come l'innalzamento della conducibilità elettrica o il calo di produzione). Questa sincronizzazione temporale permette di intercettare tempestivamente le infezioni mammarie e le affezioni podali.

2.2.3 Definizione del Processo "To-Be"

La comprensione degli obiettico di studio emersi dalle interviste hanno permesso di tracciare in linea generale la visione futura, permettendo di definire il processo "To-Be".

Processo "As-Is"

Attualmente lo studio del benessere animale richiede un enorme sforzo di controllo manuale e cognitivo. Il ricercatore deve recuperare pesanti file video dalla stalla, analizzarli visivamente ed estararne i dati unendoli con informazioni derivanti dai log del robot di mungitura. Solo successivamente possono essere dedotte eventuali conclusioni clinico-comportamentali.

Il processo "To-Be" (La visione del nuovo sistema)

Il sistema software da progettare dovrà ribaltare questo paradigma, trasferendo la responsabilità del calcolo e la trasformazione dei dati grezzi alla macchina. Nel nuovo flusso di lavoro immaginato, l'utente dovrà prelevare i blocchi video e i log di Lely per caricarli su un portale web nel quale essi verranno successivamente raggruppati e mandati in elaborazione. A questo punto l'utente potrà disconnettersi in attesa del termine dell'elaborazione. Una volta che il sistema avrà processato i dati, l'utente verrà notificato, ed una volta effettuato nuovamente l'accesso alla piattaforma, avrà la possibilità di visionare i dati elaborati e pronti per lo studio. Il veterinario potrà così concentrarsi in modo esclusivo sulla pura analisi clinica.

2.3 Analisi della Complessità e Prioritizzazione

Una volta conclusa la fase di raccolta dei bisogni sono stati effettuati una serie di confronti tecnici per analizzare la complessità dei singoli requisiti.

Appurando che i requisiti di interesse clinico-etologico riguardassero l'elaborazione dei dati, è emersa la necessità di fornire all'utente una dashboard sicura (quindi protetta da una procedura di login) che permettesse all'utente l'upload e la consultazione dei parametri oggetti di studio. I singoli requisiti sono stati quindi analizzati in funzione della loro complessità e in funzione delle informazioni in output dal modello di Computer Vision, con l'intensione di fornire una stima delle tempistiche di sviluppo per ognuno di essi.

2.3.1 Valutazione dei Vincoli Tecnici

Il primo passo dell'analisi di fattibilità è consistito in una valutazione oggettiva dei mezzi tecnologici a disposizione, con particolare attenzione al motore di analisi video. Come anticipato nello studio del dominio, il modulo di *Computer Vision* (Intelligenza Artificiale) fornito dall'Università di Torino doveva essere trattato dal team di sviluppo di Interlogic SRL come un componente *Black-Box*.

Per definire cosa fosse realmente implementabile nel software aziendale, è stato necessario analizzare il tracciato dei dati grezzi (*output*) generato dalla rete neurale.

```
1 frame_id det_index id class_id x1          y1          x2          y2          w           h           area
            centroid_x  centroid_y  proj_x proj_y proj_z
2 0        1          1  0         1627.355713 703.8161621 1821.713867 952.0407715 194.3581543 248.2246094
      48244.47693 1709.851196 815.5977783 3854   158    100
3 0        2          2  0         1143.605103 462.6463013 1223.724121 641.0177612 80.11901855 178.37146
      14290.94631 1186.050537 556.2169189 3076   504    100
4 0        3          3  0         994.118042  461.4572144 1114.892334 648.0535889 120.774292  186.5963745
      22536.04502 1060.141968 561.6160278 2833   499    100
5 0        4          4  0         1353.55481  434.2424011 1420.360596 701.47052   66.80578613 267.2281189
      17852.38456 1387.244995 572.3399048 3457   466    100
6
```

Figura 2.4: Estratto dell'output generato dal modello di Computer Vision.

Dalla lettura dei campi emerge chiaramente la natura puramente spaziale e quantitativa del dato. La *Black-Box* è in grado di fornire esclusivamente:

- Il **tracciamento temporale** (`frame_id` e `id` per mantenere la traccia dell'animale nel tempo).
- L'**ingombro in pixel** (*Bounding Box* definita dalle coordinate `x1, y1, x2, y2`, larghezza `w` e altezza `h`).

- La **localizzazione bidimensionale** (il centro di massa `centroid_x, centroid_y` e la proiezione sul piano reale della stalla tramite i campi `proj_x, proj_y`).

L'assenza di metriche semantiche più avanzate ha evidenziato criticità nella soddisfazione di due dei requisiti clinici precedentemente richiesti dai veterinari:

1. **Impossibilità di determinare la postura esatta:** L'algoritmo si limita a tracciare un rettangolo (*Bounding Box*) attorno alla massa corporea del bovino, ma non effettua una classificazione dello stato posturale. Benché in teoria si potrebbe tentare di dedurre la postura dal rapporto larghezza/altezza (`w/h`) del rettangolo, le continue occlusioni visive generate dagli altri animali e le distorsioni prospettiche delle telecamere rendono tale calcolo complesso e poco affidabile per un uso clinico.

2. **Impossibilità di calcolare l'orientamento spaziale:** Il centro di massa (`centroid`) è un singolo punto (X, Y). L'Intelligenza Artificiale non fornisce vettori direzionali né effettua il riconoscimento anatomico delle estremità dell'animale. Di conseguenza, è matematicamente impossibile stabilire dove si trovi la testa e dove la coda, rendendo inattuabile lo studio dei contatti "muso-muso" o l'orientamento del bovino all'interno delle cuccette.

2.3.2 Analisi della complessità

Sulla base delle valutazioni tecniche, le proposte cliniche sono state classificate utilizzando una metrica basata su una **scala di complessità a 5 livelli** , concepita per stimare l'impatto algoritmico e temporale di ciascuna di esse:

- **Livello 1 (Molto Bassa):** Dato già nativamente esposto o derivabile con operazioni matematiche elementari.
- **Livello 2 (Bassa):** Richiede l'implementazione di logiche spaziali o temporali standard lato *backend*.
- **Livello 3 (Media):** Richiede la progettazione di algoritmi dedicati, l'uso intensivo di elaborazioni asincrone in *background* o una complessa strutturazione del database relazionale (*Data Fusion*).
- **Livello 4 (Alta):** Richiede un pesante riaddestramento dei modelli di *Computer Vision* o l'integrazione di sensori hardware aggiuntivi, con un elevato margine di incertezza clinica.
- **Livello 5 (Molto Alta):** Supera i limiti fisici dell'hardware e del software attuale (es. risoluzione ottica, occlusioni massive), richiedendo veri e propri salti tecnologici nell'infrastruttura di base.

La seguente tabella riassume l'esito dell'analisi, associando un livello di complessità a ciascun requisito: La seguente tabella riassume la valutazione delle proposte cliniche iniziali, analizzate in base ai rigorosi vincoli imposti dalla *Black-Box* algoritmica e dalla fisica delle riprese, evidenziando le motivazioni tecniche alla base della stima di complessità:

Proposta Analitica	**Lvl.**	**Motivazione Tecnica (Vincoli AI)**
Interazioni di micro-dettaglio *(es. contatto "muso-muso")*	**5**	L'attuale AI riconosce la massa globale dell'animale ma non estrae l'orientamento direzionale o singole parti anatomiche. Le occlusioni visive renderebbero inaffidabile il calcolo del contatto intenzionale.
Riconoscimento posturale *(eretta / seduta)*	**4**	L'output fornisce solo *Bounding Box* bidimensionali. Lo schiacciamento prospettico delle telecamere perimetrali impedisce di dedurre l'asse Z (altezza) con accuratezza clinica.
Interazioni di prossimità *(Social Network Analysis)*	**3**	Richiede l'uso dei centroidi per il calcolo continuo della distanza euclidea nel tempo. La complessità è legata all'elevato carico computazionale, risolvibile delegando l'elaborazione a code asincrone in *background*.
Incrocio dati fisiologici *(es. conducibilità latte)*	**3**	Operazione indipendente dal tracciamento video. Richiede la costruzione di un *database Time-Series* e logiche di *Data Fusion* robuste per sincronizzare l'AI con i *log* testuali di Lely.
Occupazione cuccette *(monitoraggio del riposo)*	**2**	La *Black-Box* fornisce con precisione i centroidi. È sufficiente un calcolo logico-spaziale lato *backend* per misurarne le intersezioni con aree statiche pre-calibrate.

Tabella 2.1: Valutazione della complessità delle proposte cliniche in relazione ai limiti algoritmici della *Computer Vision*.

2.3.3 Prioritizzazione Strategica

Conclusa l'analisi di complessità e definiti i **trade-off** ingegneristici, il team di sviluppo si è trovato di fronte a un set di funzionalità, profondamente eterogenee per impatto clinico e sforzo realizzativo. Per fronteggiare il rischio di modificare continuamente i requisiti e garantire il successo architetturale, è stato necessario applicare un rigoroso framework di prioritizzazione. L'obiettivo era stabilire una

gerarchia di sviluppo inequivocabile, garantendo che le risorse aziendali venissero allocate primariamente sui componenti vitali del Decision Support System (DSS).

2.3.4 Applicazione del Metodo MoSCoW

Per categorizzare le funzionalità emerse e filtrate durante i workshop, è stato adottato il **Metodo MoSCoW** [3], una tecnica standard e ampiamente consolidata nell'ambito dello sviluppo e dell'Ingegneria dei Requisiti. Questo approccio ha permesso di classificare i requisiti in quattro categorie di priorità decrescente, allineando le aspettative degli stakeholder clinici con le reali tempistiche di ingegnerizzazione. Di seguito viene riportata l'applicazione del metodo al dominio specifico del progetto SIMBA:

Must Have (Requisiti Vitali): Rappresentano gli elementi non negoziabili, senza i quali il sistema software non avrebbe alcun valore o non rispetterebbe i vincoli minimi di sicurezza e operatività. In questa categoria fondante sono stati inseriti:

- L'infrastruttura di autenticazione e sicurezza (basata su token JWT) per proteggere i dati aziendali sensibili.
- Il sistema di archiviazione asincrona (*Storage* e gestione code) per supportare l'upload dei pesanti file video 4K senza mandare in *time-out* l'applicativo web.
- La struttura del database *Time-Series* (PostgreSQL) per la persistenza delle coordinate visive.
- Il motore logico di *Data Fusion* per incrociare i file CSV dell'Intelligenza Artificiale con i *log* del robot Lely.
- Il calcolo e la visualizzazione a semaforo del primo e più importante indicatore clinico: l'Indice di Rischio Mastite (MR).

Should Have (Requisiti ad Alto Valore): Funzionalità estremamente importanti per l'indagine etologica, ma la cui assenza temporanea non comprometterebbe l'esecuzione del motore di base (il tracciamento e il rischio mastite). Comprendono:

- L'analisi spaziale geometrica per il monitoraggio del riposo (le *Heatmap* dell'occupazione delle cuccette).
- Il calcolo del *Robot Refusal Rate* per monitorare i cali di motivazione o le sospette zoppie.
- L'algoritmo di calcolo delle distanze euclidee per la *Social Network Analysis* (interazioni di prossimità).

Categoria	Priorità	Funzionalità Architetturali	Motivazione Strategica
Must Have	**Vitale** *(Perimetro MVP)*	• Autenticazione (JWT) • Upload e Storage (Code) • DB Time-Series • Motore di Data Fusion • Rischio Mastite (MR)	Requisiti non negoziabili. Costituiscono le fondamenta e garantiscono i vincoli del bando e della sicurezza.
Should Have	**Alta** *(Release future)*	• Occupazione cuccette • Robot Refusal Rate • Interazioni prossimità	Alto valore clinico. Non bloccano l'infrastruttura di base, ma completano lo studio etologico (UniTo).
Could Have	**Desiderabile** *(Nice-to-have)*	• Navigazione avanzata • Export CSV filtrabile	Elementi di rifinitura per la UX. Sviluppo subordinato alle tempistiche delle priorità superiori.
Won't Have	**Sospesa** *(Backlog)*	• Postura esatta • Contatto "muso-muso"	Fuori scala per complessità computazionale e limiti hardware. In attesa di upgrade dell'AI.

Tabella 2.2: Matrice di Prioritizzazione MoSCoW applicata ai requisiti del progetto SIMBA.

Could Have (Requisiti Desiderabili): Elementi di rifinitura o *nice-to-have* che migliorano la *User Experience* o offrono flessibilità analitiche aggiuntive. La loro implementazione è subordinata al mancato esaurimento del budget temporale dedicato alle priorità superiori:

- Funzioni di filtraggio avanzato e navigazione temporale complessa sui grafici della *dashboard*.

- Esportazione massiva formattata (in CSV) filtrabile per singoli capi o per finestre orarie specifiche da parte del ricercatore.

Won't Have (Requisiti Sospesi): Questa categoria ha assorbito tutte le proposte cliniche valutate con livello di complessità 4 e 5 durante la precedente fase di analisi tecnica.

- Il riconoscimento posturale esatto (differenziazione eretta/seduta tramite asse Z).

- La rilevazione dell'orientamento spaziale direzionale e il conseguente contatto intenzionale "muso-muso".

Queste funzionalità, come concordato, sono state congelate nel *backlog* in attesa di evoluzioni future della *Black-Box* AI da parte dell'Organismo di Ricerca.

2.3.5 Definizione del Minimum Viable Product (MVP)

La rigorosa classificazione MoSCoW è stata lo strumento cardine per definire il perimetro del **Minimum Viable Product (MVP)**. Nel contesto dell'Ingegneria del Software, l'MVP non va inteso come un prodotto incompleto, instabile o difettoso, ma come la versione più snella e focalizzata dell'applicativo, contenente esclusivamente le funzionalità strettamente necessarie per validare l'idea fondante del progetto e fornire valore immediato agli *stakeholder*.

Per il progetto SIMBA, l'obiettivo fondante (la *value proposition*) da validare era la **Data Fusion asincrona**: dimostrare all'ente finanziatore e alla comunità scientifica che fosse tecnologicamente possibile, e architetturalmente stabile, unire il dato visivo al dato fisiologico di stalla.

Di conseguenza, il perimetro dell'MVP è stato tracciato racchiudendo esclusivamente tutti i requisiti appartenenti alla categoria **"Must Have"**. Il primo traguardo di rilascio del team di sviluppo non avrebbe compreso le mappe di calore o i grafi relazionali complessi, ma doveva garantire un flusso logico *end-to-end* inattaccabile: un operatore doveva poter accedere in modo protetto, caricare un video in 4K, avviare l'elaborazione invisibile dell'Intelligenza Artificiale orchestrata dal *backend*, e infine visualizzare il cruscotto con l'incrocio dei dati e il calcolo del Rischio Mastite.

Rilasciare con successo questo MVP ha garantito due risultati strategici di inestimabile valore: da un lato, ha fornito ai veterinari di UniTo un primissimo strumento funzionante e validato per avviare le loro ricerche sul campo; dall'altro, ha permesso a Interlogic SRL di certificare l'avanzamento del *Technology Readiness Level* (TRL) richiesto dal bando regionale, gettando fondamenta architetturali stabili su cui innestare iterativamente le funzionalità "Should Have" nelle fasi di sviluppo successive.

2.4 Development Lifecycle e Specifica Formale

Il passaggio logico conclusivo della fase metodologica consiste nella formalizzazione del ciclo di vita dello sviluppo (*Software Development Life Cycle* - SDLC) e nella conseguente specifica dei requisiti. Terminata la fase di esplorazione del dominio, di negoziazione tecnica e di prioritizzazione (MoSCoW), è necessario definire la filosofia operativa con la quale il lavoro di analisi verrà tradotto in artefatti ingegneristici concreti. In questa fase l'obiettivo è trasformare i desideri degli stakeholder in documenti tecnici rigorosi che guideranno successiva scrittura del codice.

2.4.1 Approccio Metodologico

Da un lato dello spettro si colloca l'approccio puramente predittivo, frequentemente implementato attraverso un'interpretazione rigida del modello a cascata (il cosiddetto **Naive Waterfall**), il quale assume che i requisiti possano essere elicitati in modo esaustivo, documentati e validati a priori. Dall'altro lato si posizionano gli approcci adattivi, caratterizzati da cicli di vita iterativi e incrementali; questi trovano la loro massima espressione contemporanea nelle filosofie e nei framework **Agile**, concepiti per accogliere il cambiamento fisiologico dei requisiti in corso d'opera attraverso continui cicli di retroazione (**feedback loop**) [4].

Nello scenario specifico del progetto SIMBA, l'adozione "pura" di uno degli estremi di questo spettro si sarebbe rivelata inadeguata. Un approccio orientato al Naive Waterfall era incompatibile con la profonda incertezza della ricerca sperimentale: l'utilizzo di algoritmi di Intelligenza Artificiale su comportamenti etologici avrebbe inevitabilmente richiesto aggiustamenti architetturali in risposta ai dati empirici emersi sul campo. Al contrario, un approccio Agile puro (in cui il perimetro del progetto e i costi non vengono fissati rigidamente in fase di avvio) si scontrava con i severi vincoli burocratici, economici e di scadenza imposti dal bando regionale SWIch.

Il team di sviluppo ha adottato organicamente un **approccio metodologico ibrido (sviluppo a due velocità)**. Questa separazione di paradigma ha fornito un vantaggio ingegneristico cruciale, permettendo di parallelizzare il lavoro su due binari distinti:

- **Approccio Predittivo (Infrastruttura di base):** Le logiche predittive, tipiche dello sviluppo a cascata, sono state applicate alle componenti strutturali e deterministiche del sistema. Poiché i requisiti legati all'ingestione dei dati erano ben definiti e indipendenti dall'evoluzione dell'indagine clinica, il team ha potuto avviare immediatamente la scrittura del codice. In questo modo si è consolidato il *backend* architetturale, e parte del *frontend* bloccandone le specifiche nelle fasi iniziali del progetto.

- **Approccio Adattivo (Analisi e Visualizzazione):** L'approccio iterativo e incrementale, tipico delle metodologie *Agile*, è stato invece riservato esclusivamente alla gestione delle informazioni estratte dal modulo AI e alla loro visualizzazione (*Frontend*). Poiché la rete neurale era un componente in continua evoluzione da parte dell'Organismo di Ricerca (UniTo), le metriche cliniche derivate erano soggette a forte volatilità. Gestire il motore di *Data Fusion* e la *dashboard* tramite iterazioni rapide e *feedback* continui dei veterinari ha permesso all'applicativo di adattarsi dinamicamente ai limiti dell'Intelligenza Artificiale, senza mai invalidare il lavoro infrastrutturale pregresso.

Questa scelta metodologica ha influenzato direttamente il modo in cui sono state redatte le specifiche formali. Per mantenere l'agilità operativa richiesta, i requisiti non sono stati prodotti come un documento monolitico di difficile consultazione, ma sono stati formalizzati e divisi nelle due canoniche macro-categorie dell'Ingegneria del Software: **Requisiti Funzionali (FR)** e **Requisiti Non Funzionali (NFR)**.

2.4.2 Definizione dei Requisiti Funzionali (FR)

La definizione dei Requisiti Funzionali (*Functional Requirements* - FR) rappresenta il processo di specificazione formale di **cosa il sistema deve fare operativamente**. In questa fase, le funzionalità classificate come *Must Have* e *Should Have* durante l'analisi MoSCoW vengono prese in carico dagli analisti di sistema e spogliate di qualsiasi ambiguità linguistica.

L'obiettivo metodologico di questo *step* non è descrivere le scelte tecnologiche (es. quale linguaggio di programmazione usare), ma definire con assoluto rigore i comportamenti, le interazioni e le logiche di *business* che l'applicativo dovrà esporre agli utenti finali. Nel contesto del *Decision Support System* per la stalla, la definizione degli FR ha comportato la scomposizione del processo "To-Be" in unità logiche testabili, stabilendo esplicitamente:

- **Le interazioni di Input:** Come l'utente o i sistemi esterni (es. l'operatore tramite l'interfaccia o il NAS locale) immettono i flussi video e i file di *log* nel sistema.
- **Le logiche di elaborazione:** Le regole matematiche di *Data Fusion* che il motore logico deve applicare per trasformare un dato spaziale grezzo (es. le coordinate di un centroide) in un indicatore clinico di sintesi (es. il rischio di mastite).
- **Gli output attesi:** Le modalità esatte e le visualizzazioni grafiche con cui il sistema deve restituire l'informazione elaborata (es. cruscotti a semaforo, mappe di calore, esportazioni in CSV).

L'adozione di un linguaggio strutturato in questa fase ha permesso di creare un vero e proprio "contratto logico" tra l'Organismo di Ricerca (che ha convalidato la correttezza scientifica delle logiche) e il team di sviluppo.

2.4.3 Definizione dei Requisiti Non Funzionali (NFR)

Parallelamente alle funzioni operative, la metodologia ingegneristica impone la definizione dei Requisiti Non Funzionali (*Non-Functional Requirements* - NFR). Se i requisiti funzionali descrivono *cosa* il sistema fa, i requisiti non funzionali dettano **come e quanto bene deve farlo**.

Questa fase della specifica si concentra sugli **attributi di qualità** dell'architettura (*Quality Attributes*), parametri sistemistici che spesso risultano invisibili all'utente finale, ma che determinano la reale sopravvivenza del software in un ambiente di produzione. Trascurare l'elicitazione degli NFR in un ecosistema ad alto carico di rete come quello zootecnico avrebbe portato alla creazione di un prodotto magari ineccepibile dal punto di vista del calcolo clinico, ma del tutto inutilizzabile nella pratica quotidiana.

La metodologia di definizione degli NFR si è concentrata sull'isolare e regolamentare parametri critici quali:

- **Sicurezza (*Security*):** Le direttive architetturali per garantire la rigida confidenzialità dei dati aziendali e sanitari, definendo le *policy* di autenticazione e i sistemi di autorizzazione.
- **Prestazioni e Asincronia (*Performance*):** I vincoli temporali architetturali necessari per garantire che l'interfaccia utente rimanga reattiva e navigabile anche durante il concomitante caricamento di file video in 4K del peso di decine di gigabyte.
- **Scalabilità (*Scalability*):** La capacità dell'infrastruttura di adattarsi agilmente all'aumento esponenziale del carico di lavoro, prevedendo l'inevitabile accumulo di serie storiche (milioni di record nel database) e terabyte di dati video multimediali nel corso dei mesi di monitoraggio.
- **Affidabilità (*Reliability*):** La resilienza del sistema nel gestire l'orchestrazione complessa tra i vari microservizi (es. il *backend*, lo *storage* e la *Black-Box* AI) assicurando l'integrità del dato.

La formalizzazione congiunta di FR e NFR ha sigillato definitivamente la fase metodologica di analisi. Gli artefatti ingegneristici così prodotti costituiscono le fondamenta documentali su cui si basa l'intero Capitolo 3 , che andrà a dettagliare puntualmente l'architettura dei requisiti del progetto SIMBA.

Capitolo 3

Progettazione Architetturale e Backend

3.1 Dai Requisiti al Design Architetturale

Il passaggio metodologico illustrato nel capitolo precedente ha permesso di filtrare le innumerevoli richieste cliniche emerse durante i tavoli di lavoro, giungendo a una definizione chiara del perimetro di progetto. In questa fase ingegneristica, l'applicazione del *framework* MoSCoW (*Must, Should, Could, Won't*) si è tradotta nella stesura formale del documento di specifica, che ha guidato l'intero ciclo di sviluppo del *Decision Support System* (DSS).

Di seguito vengono presentati gli artefatti documentali che definiscono il comportamento atteso del sistema (Requisiti Funzionali) e i vincoli infrastrutturali a cui esso deve sottostare (Requisiti Non Funzionali).

3.1.1 Specifica dei Requisiti Funzionali (RF)

I Requisiti Funzionali (RF) descrivono esplicitamente le operazioni, le interazioni e i calcoli che l'applicativo software deve garantire per soddisfare gli obiettivi aziendali e di ricerca.

I requisiti sono stati raggruppati e classificati in base alla loro priorità architetturale. È importante sottolineare che i requisiti legati al tracciamento puro e all'addestramento algoritmico (come la calibrazione ottica delle telecamere e il modello Python di reti neurali) sono stati formalmente trattati dal team di sviluppo come **Dipendenze Esterne (Black-Box)** fornite dall'Organismo di Ricerca (UniTo). Pertanto, essi non figurano nei requisiti di sviluppo del software aziendale, il cui compito è invece l'ingestione e l'elaborazione del loro *output*.

Tabella 3.1: Specifica dei Requisiti Funzionali (RF) e relativa classificazione MoSCoW.

ID	Nome Requisito	Descrizione Architetturale e Logica	Priorità
RF-001	Gestione Accessi (Autenticazione)	Il sistema deve esporre un meccanismo sicuro di Login, Logout e recupero credenziali per proteggere i dati sanitari aziendali.	**Must**
RF-002	Deployment e Accessibilità	L'applicativo deve essere distribuito e accessibile via web tramite un'infrastruttura Cloud dedicata (es. *simba.interlogic.it*).	**Must**
RF-003	Ingestione Multimediale Asincrona	Il sistema deve permettere il caricamento di flussi video ad alta risoluzione, gestendo l'upload in *background* per non bloccare l'interfaccia.	**Must**
RF-004	Elaborazione e Ingestione Dati AI	L'applicativo deve orchestrare l'analisi dei video tramite il modulo AI esterno e permettere il download dei risultati posizionali (CSV).	**Must**
RF-005	Mappa 2D Posizionale (Statica)	Generazione di una vista aerea della stalla, sincronizzata con una barra temporale, per mostrare le posizioni dei bovini.	**Must**
RF-006	Rilevamento Interazioni di Prossimità	Calcolo algoritmico delle interazioni sociali se la distanza euclidea tra due bovini è $< 1{,}5$m per finestre temporali di 20 secondi.	**Must**
RF-007	Data Fusion: KPI Robot Mungitura	Ingestione automatizzata (o massiva) dei *file* CSV del robot Lely, per l'incrocio con i dati spaziali.	**Should**
RF-008	Analisi Spaziale: Occupazione Zone	Calcolo computazionale e visualizzazione del tempo di occupazione di specifiche zone (cuccette, abbeveratoio, ecc.).	**Should**
RF-009	Analisi Spaziale: Heatmap	Generazione di mappe di calore statiche e dinamiche, filtrabili per singolo capo, per evidenziare le aree di maggior stazionamento.	**Should**

(Continua nella pagina successiva)

(Continua dalla pagina precedente)

ID	Nome Requisito	Descrizione Architetturale e Logica	Priorità
RF-010	Tracciamento Dinamico Avanzato	Trasformazione della Mappa 2D in uno strumento interattivo per visualizzare i tracciati continui (*Tracking 2D*).	**Could**
RF-011	Social Network Analysis: Grafo	Rappresentazione visiva delle interazioni sociali (RF-006) mediante un grafo relazionale a nodi e archi.	**Could**
RF-012	Visual Data Querying (ChatBOT)	Implementazione di un modulo interrogabile in linguaggio naturale per l'estrazione semplificata dei dati statistici.	**Won't**

3.1.2 Specifica dei Requisiti Non Funzionali (RNF)

Parallelamente alle logiche di *business*, l'Ingegneria dei Requisiti ha imposto la definizione rigorosa dei Requisiti Non Funzionali (RNF). Questi parametri non descrivono *cosa* il sistema fa, ma dettano i vincoli qualitativi, infrastrutturali e prestazionali (*Quality Attributes*) entro i quali il software deve operare.

Nel contesto del progetto SIMBA, trascurare gli RNF avrebbe portato allo sviluppo di un applicativo matematicamente corretto ma del tutto inutilizzabile nel mondo reale, a causa dell'enorme mole di dati multimediali (video in 4K) e tabellari (milioni di record posizionali) generati dalla stalla.

I seguenti requisiti rappresentano i pilastri che hanno guidato la progettazione dell'architettura a microservizi e la scelta delle tecnologie di persistenza:

Tabella 3.2: Specifica dei Requisiti Non Funzionali (RNF) per l'architettura SIMBA.

ID	Categoria	Descrizione del Vincolo	Priorità
RNF-001	Prestazioni (Performance)	La *dashboard* clinica deve garantire tempi di caricamento inferiori al secondo (es. tramite Redis Cache) anche nell'elaborazione di metriche complesse.	**Must**
RNF-002	Archiviazione (Storage)	Sostenere l'ingestione di massivi flussi multimediali e CSV (stima 50-70 GB/giorno) tramite storage a scalabilità orizzontale (es. Amazon S3).	**Must**

(Continua nella pagina successiva)

(Continua dalla pagina precedente)

ID	Categoria	Descrizione del Vincolo	Priorità
RNF-003	Gestione RAM e I/O	I file video e i CSV non devono essere caricati in RAM. Richiesta lettura diretta dal disco (*s3fs*) e trasmissione in *Streaming Chunked*.	**Must**
RNF-004	Affidabilità (Reliability)	I fallimenti dei job IA non devono bloccare il sistema; i task pesanti vanno disaccoppiati tramite un Message Broker (*Event-Driven*).	**Must**
RNF-005	Sicurezza e Privacy	I dati clinici devono essere isolati logicamente, la comunicazione cifrata e l'autenticazione *stateless* (JWT).	**Must**
RNF-006	Modularità	Architettura suddivisa in Microservizi (Spring Boot) indipendenti per permettere l'aggiornamento isolato dei singoli domini.	**Should**

3.1.3 Architettura di Sistema ad Alto Livello: Design Driven by Requirements

L'architettura del sistema SIMBA rappresenta la traduzione ingegneristica dei Requisiti Funzionali (RF) e dei vincoli operativi (RNF) definiti in fase di analisi. Per soddisfare la necessità di reattività del *client* e l'elaborazione intensiva dei file video, si è optato per un ecosistema *Cloud-Native* basato sul paradigma a **Microservizi**, progettato per il *deployment* su infrastruttura **Amazon Web Services (AWS)** [5].

La topologia di questa infrastruttura, i protocolli di comunicazione e l'orchestrazione dei moduli sono illustrati in Figura 3.1.

L'ecosistema si compone di moduli disaccoppiati, governati da **Consul** [6] come strumento di *Service Discovery* per la risoluzione dinamica degli indirizzi di rete interni. Il sistema è suddiviso nei seguenti macro-componenti operativi:

- **Livello di Accesso e Sicurezza (Web + Gateway):** L'interfaccia utente è una *Single Page Application* sviluppata in **React** [7]. Le richieste vengono instradate allo **Spring Cloud Gateway** (porta 8080), che agisce da *API Gateway* [8] e gestisce il *routing* dinamico verso i microservizi sottostanti, validando centralmente l'autenticazione tramite token JWT (*Stateless authentication*).
- **Microservizi Core di Dominio (Java Spring Boot):** Le logiche di *business* sono frammentate in tre servizi indipendenti, *deployati* su istanze EC2 all'interno di *subnet* private:

  - `authentication-service`: Gestisce il ciclo di vita degli utenti, la validazione su database e l'emissione dei JWT. Gestisce inoltre l'invio asincrono delle comunicazioni (es. *reset password*) pubblicando *job* su un server SMTP (Amazon SES) tramite code Redis.

  - `storage-service`: Regola il caricamento (*upload*) e lo *streaming chunked* dei file video richiesti dal *Frontend*, interfacciandosi con l'*Object Storage* e mantenendo i metadati referenziali nel database.

  - `analysis-service`: È il cuore algoritmico del sistema. Consuma i messaggi di completamento analisi, interroga l'*Object Storage* per recuperare i CSV generati dall'AI, esegue il *parsing* dei dati ed effettua la *Data Fusion*. Poiché si fa carico dei calcoli per le viste cliniche, implementa meccanismi di *Caching* per le elaborazioni pesanti (come i grafi relazionali e le *Heatmap*).

- **Livello di Persistenza e Caching (Database, S3, Redis):** L'eterogeneità dei dati ha imposto un netto sezionamento dello *storage*, delegato a servizi gestiti ad alte prestazioni:

  - **PostgreSQL (AWS RDS):** Struttura relazionale utilizzata per le anagrafiche, la storicizzazione dei *log* di mungitura e il salvataggio dei risultati di inferenza AI associati alle procedure [9].

  - **Object Storage (S3 / MinIO):** Spazio di archiviazione dedicato ai flussi video e ai pesanti file CSV di *output* [10] [11].

  - **Redis (AWS ElastiCache):** Svolge un duplice ruolo cruciale. Agisce da strato di **Cache** per velocizzare le letture dei dati spaziali e funge da **Message Broker** (*Publish/Subscribe*) gestendo le code asincrone (`ai-jobs`, `ai-results-topic`, `email_dispatch_queue`), disaccoppiando totalmente il *frontend* dai pesanti *task* di elaborazione [12] [13].

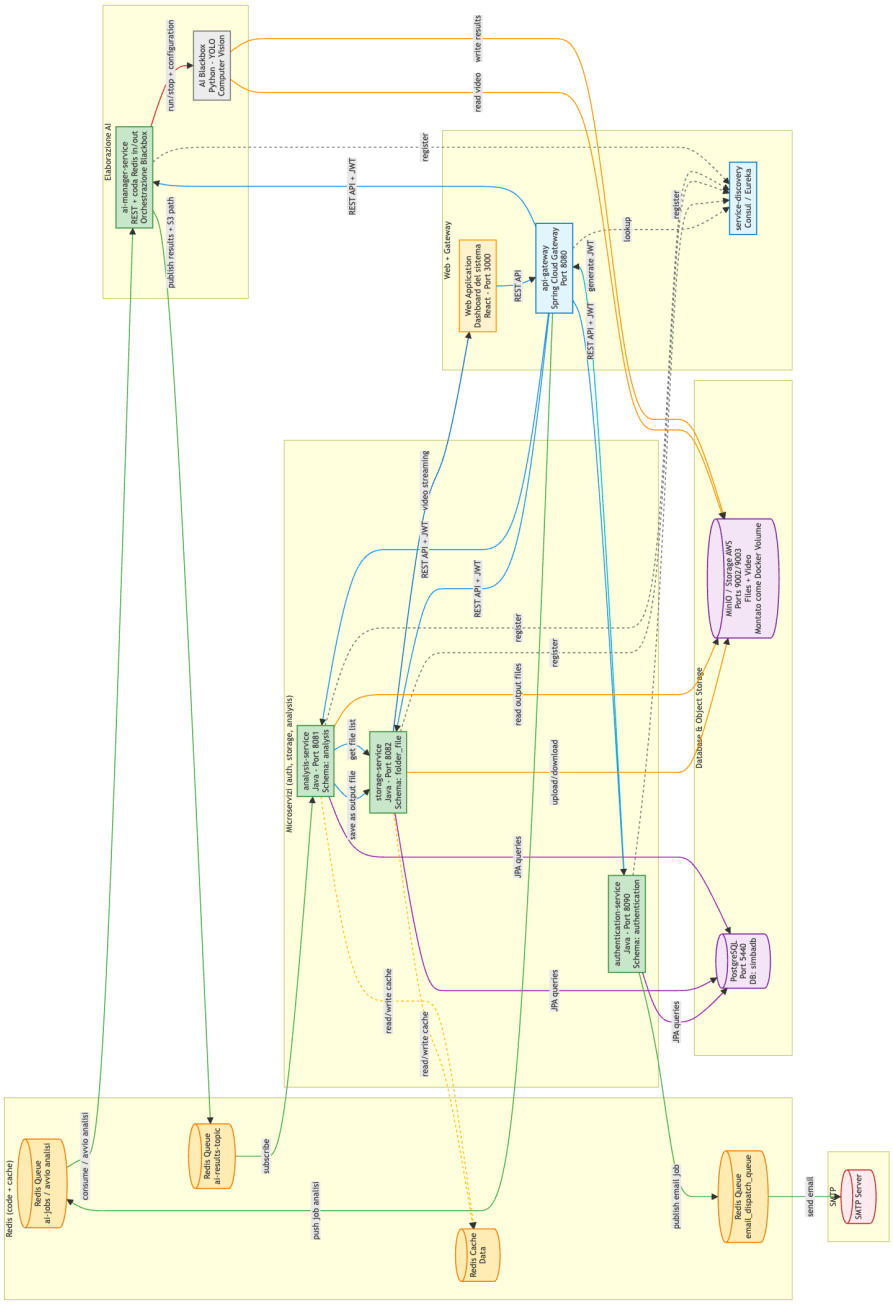

Figura 3.1: Diagramma architetturale ad alto livello del sistema SIMBA. Sono evidenziati il livello di *routing* (Spring Cloud Gateway), i microservizi di dominio (Java Spring Boot), il *layer* di persistenza (PostgreSQL e S3/MinIO), l'infrastruttura Redis (Code e *Cache*) e il modulo di elaborazione AI (Python YOLO).

- **Elaborazione AI e Ottimizzazione I/O (GPU Node):** Il modello di *Computer Vision* (YOLO in Python) è ospitato su un nodo EC2 dedicato con accelerazione hardware (GPU). L'integrazione con il *core* aziendale è gestita dall'`ai-manager-service`, che orchestra l'avvio della *Blackbox*. Un'ottimizzazione architetturale fondamentale risiede nell'accesso ai dati: per evitare pesanti trasferimenti HTTP (con video di decine di GB), **lo *storage* S3 è stato montato direttamente come *Docker Volume***. La *Blackbox* legge i video ed esporta i CSV comportandosi come se stesse interagendo con un *file system* locale, delegando al sistema operativo la sincronizzazione trasparente con il *cloud*. Al termine, l'*AI Manager* notifica l'`analysis-service` pubblicando il *path* del risultato sulla coda Redis, chiudendo il ciclo in totale asincronia.

3.2 Il Backend, lo Storage e la Sfida dell'Asincronia

Il soddisfacimento dei vincoli prestazionali (RNF-001) e di archiviazione (RNF-002) ha rappresentato la sfida ingegneristica più complessa del progetto. La gestione di flussi video in 4K e dei massivi *file* di *output* generati dall'Intelligenza Artificiale (spesso superiori ai 30 GB) ha imposto un ripensamento radicale delle classiche architetture sincrone, spingendo il *design* verso soluzioni *Cloud-Native* avanzate su ecosistema AWS.

3.2.1 Ingestione e Streaming Video (Storage-Service)

Per separare il carico computazionale da quello di puro I/O, la gestione dei file multimediali è stata isolata all'interno dello `storage-service`.

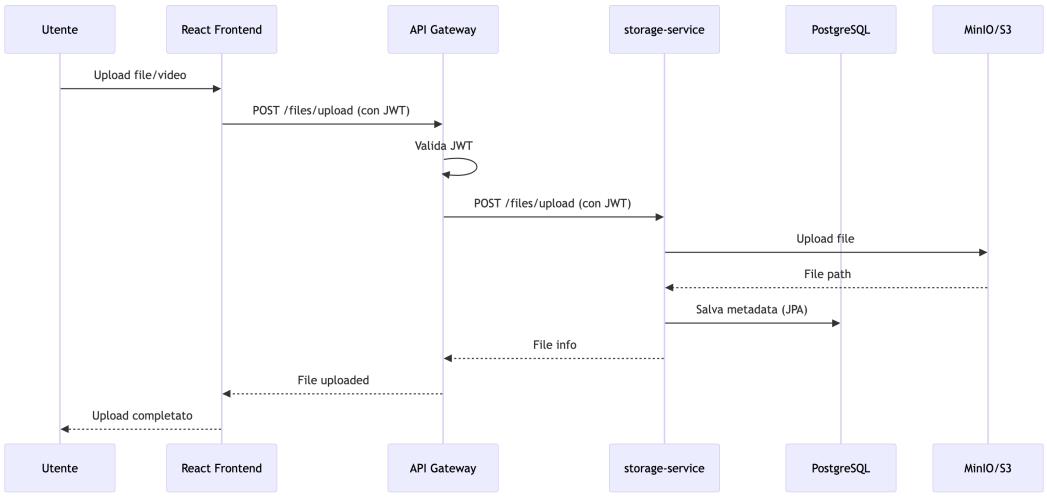

Figura 3.2: *Sequence diagram* del flusso di *upload*. Il video viene caricato sull'Object Storage (S3) e i metadati relazionali salvati in PostgreSQL.

Durante la fase di *upload*, il *file* intercetta l'API Gateway e viene consegnato allo `storage-service`, il quale lo trasferisce direttamente sull'*Object Storage* (Amazon S3 / MinIO). In PostgreSQL (su AWS RDS) vengono salvati unicamente i metadati (identificativo, stalla, data), mantenendo il database relazionale leggero ed efficiente.

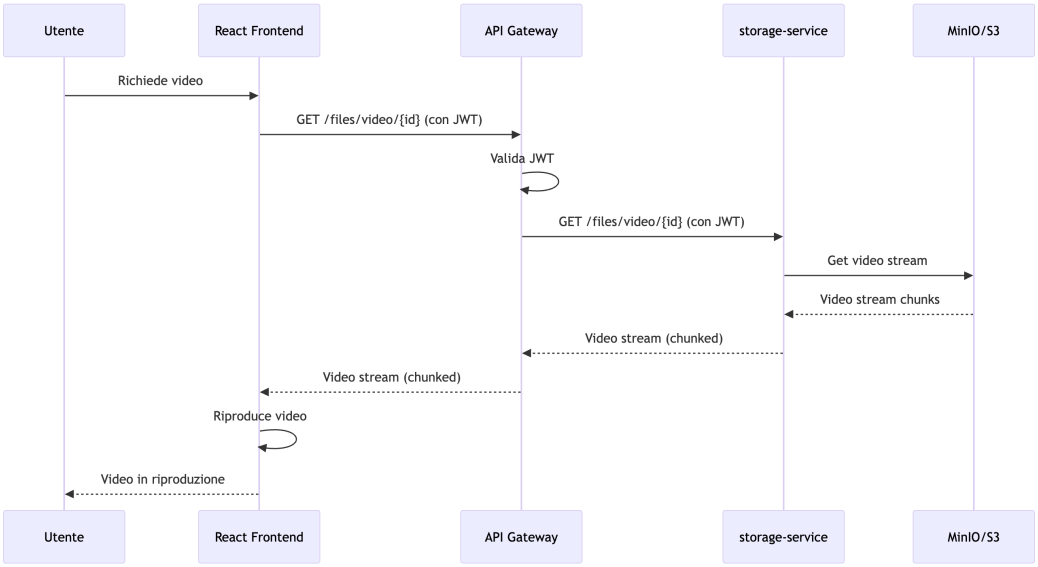

Figura 3.3: *Sequence diagram* del flusso di *streaming chunked*. Il frontend riceve il video a porzioni, evitando il sovraccarico della RAM.

Simmetricamente, per non bloccare il *browser* dell'utente in fase di visualizzazione, lo `storage-service` non scarica l'intero video per poi inviarlo al *client*. Implementa invece un meccanismo di **Streaming Chunked**: legge il flusso video da S3 "a pezzi" (*chunks*) e lo trasmette in tempo reale al *Frontend* in React, garantendo una riproduzione immediata e un consumo di RAM lato server quasi nullo.

3.2.2 Orchestrazione AI e il Pattern "Docker Volume" (s3fs)

La vera potenza dell'architettura si manifesta nella gestione del *job* di analisi video, orchestrato tramite il *pattern* a eventi (*Event-Driven*).

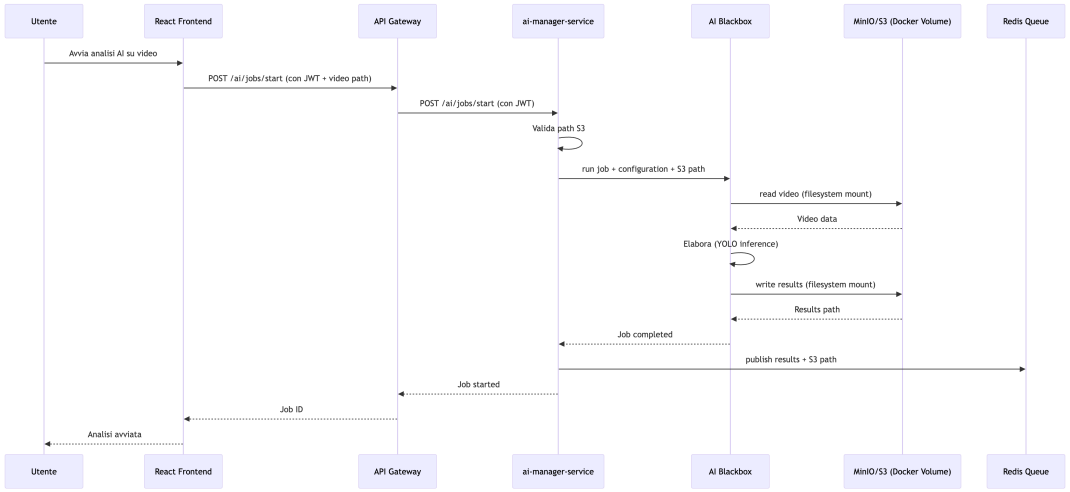

Figura 3.4: *Sequence diagram* dell'avvio del Job AI. Evidenzia l'orchestrazione asincrona e l'accesso diretto al *filesystem* tramite Docker Volume.

Quando l'utente richiede l'analisi, la richiesta viene instradata all'`ai-manager-service`, situato su un nodo EC2 dedicato con accelerazione hardware (GPU). L'innovazione ingegneristica introdotta in questa fase riguarda l'accesso ai dati: **la Blackbox Python (YOLO) non scarica il video via rete HTTP**. Al contrario, il *bucket* S3 è stato montato direttamente sul nodo tramite `s3fs` (come *Docker Volume*).

La rete neurale legge il video e scrive i risultati comportandosi come se stesse interagendo con un normale disco fisso locale. Questo aggira problemi legati alla *Blackbox*, che necessita input strutturati in un sistema di cartelle UNIX . Al termine dell'elaborazione, l'AI Manager pubblica semplicemente il *path* del risultato (S3) sulla coda Redis (`ai-results-topic`), chiudendo il *task* asincrono in totale sicurezza.

3.2.3 Elaborazione Risultati: Claim-Check Pattern e Redis Cache

L'ultima fase del processo ingegneristico è la cosiddetta *Data Fusion*, dove i milioni di coordinate estratte diventano metriche cliniche. Questo compito è affidato all'`analysis-service`.

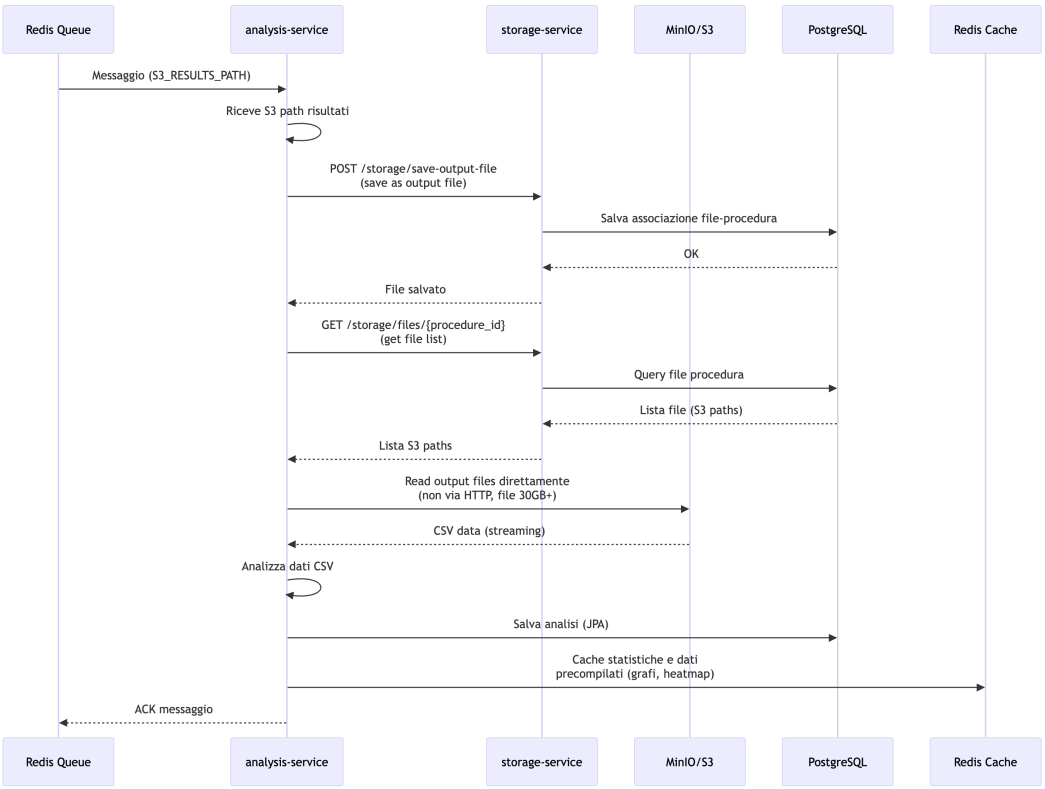

Figura 3.5: *Sequence diagram* dell'elaborazione dei risultati. L'`analysis-service` consuma la coda Redis, legge i CSV massivi direttamente da S3 e popola la *Cache*.

L'implementazione segue rigorosamente il **Claim-Check Pattern**:

1. L'`analysis-service` agisce da *consumer* (*subscriber*) sulla coda Redis. Riceve un messaggio piccolissimo contenente solo l'indirizzo (`S3_RESULTS_PATH`) in cui si trovano i dati.
2. Invece di richiedere il file tramite API HTTP (che andrebbe in *timeout* per file di decine di GB), il microservizio apre uno *stream* di lettura **direttamente** dall'Object Storage S3, elaborando le righe CSV in modo sequenziale.
3. Le associazioni tra procedure e coordinate vengono persistite in PostgreSQL.

4. **Il ruolo di Redis Cache:** Poiché calcolare grafi relazionali e mappe di calore (*Heatmap*) su milioni di record richiede una potenza di calcolo enorme, l'`analysis-service` pre-calcola queste metriche statistiche e le salva su **Redis Cache** (AWS ElastiCache). Quando il *Frontend* React richiederà la *dashboard*, l'API Gateway risponderà in pochi millisecondi leggendo dalla RAM di Redis, invece di lanciare pesantissime *query* aggreganti sul database relazionale.

3.3 Livello Dati: Modellazione Ibrida e *Data Fusion*

Risolto il problema dell'ingestione multimediale e dell'orchestrazione asincrona, l'architettura si è trovata a dover gestire l'*output* generato da tali processi. Il cuore del *Decision Support System* non risiede nella semplice archiviazione, ma nella capacità di incrociare informazioni di natura radicalmente diversa per generare nuovo valore clinico, un processo noto in letteratura come *Data Fusion*.

3.3.1 Il Problema: Volumetria, Eterogeneità e Frequenza dei Dati

Il soddisfacimento dei requisiti funzionali legati al calcolo dei KPI richiedeva la fusione di due flussi informativi caratterizzati da domini semantici e granularità temporali completamente asimmetriche: da un lato gli eventi clinici discreti (i *log* della mungitrice Lely), dall'altro le serie storiche spaziali continue generate dall'Intelligenza Artificiale.

A questa eterogeneità semantica si è aggiunto un severo vincolo infrastrutturale legato al **dimensionamento dei dati** . Durante la fase di progettazione del *deployment* su AWS, è stata condotta una stima volumetrica per ogni ciclo di elaborazione:

- Il flusso video grezzo, non compresso, genera un peso di circa 8 GB all'ora per singola telecamera.
- Considerando l'utilizzo in stalla di 4 telecamere, si raggiungono i 32 GB di dati all'ora.
- Una tipica sessione di analisi da 2 ore produce quindi un *input* di circa 64 GB al giorno.
- A questi si sommano i risultati algoritmici dell'AI (le annotazioni spaziali CSV), stimati nell'ordine di 80 GB per ciclo operativo.

In sintesi, una singola giornata di analisi genera tra i 70 e i 100 GB di dati. Provare a salvare questa mole di informazioni all'interno di un database relazionale tradizionale ne avrebbe causato l'immediato collasso.

3.3.2 La Soluzione: Persistenza Ibrida (PostgreSQL e S3)

Per far fronte a questa sfida, il livello di persistenza è stato ingegnerizzato seguendo un approccio ibrido, separando lo *storage* dei *blob* binari dai metadati relazionali:

1. **Il dato pesante su Object Storage (S3):** Come anticipato, i file video e gli enormi CSV di *output* (spesso superiori a 10 GB) risiedono esclusivamente su Amazon S3 (o MinIO). Questo livello garantisce scalabilità orizzontale illimitata a costi contenuti.

2. **Il dato strutturato su PostgreSQL (RDS):** Il database relazionale viene interrogato dai microservizi (tramite *query* JPA) esclusivamente per salvare l'anagrafica aziendale, i *log* Lely e le **associazioni file-procedura**. In questo modo, il database rimane snello, garantendo tempi di risposta inferiori al secondo.

3.3.3 La *Data Fusion* e l'Ottimizzazione tramite Redis Cache

Il vero "motore" clinico del sistema prende vita all'interno dell'`analysis-service`. Questo microservizio entra in azione al termine del *job* AI, quando riceve sulla coda Redis il messaggio contenente il *path* S3 dei risultati appena generati.

Il flusso di elaborazione e fusione dei dati è stato ingegnerizzato per massimizzare l'efficienza:

1. **Recupero Metadati e Lettura Diretta:** L'`analysis-service` contatta prima lo `storage-service` per ottenere da PostgreSQL la lista esatta dei percorsi S3 associati alla procedura clinica. Successivamente, per evitare *timeout* dovuti al peso dei file (10+ GB), il servizio **legge il flusso CSV direttamente da S3**, *bypassando* del tutto il protocollo HTTP convenzionale.

2. **Algoritmo di Sincronizzazione:** Il *backend* esegue il *parsing* sequenziale del CSV (i milioni di punti X, Y generati dall'AI) e interroga PostgreSQL per estrarre gli eventi Lely. Attraverso *query* spaziali e temporali, il sistema "allinea" i due flussi usando i *timestamp* come chiave relazionale primaria.

3. **Caching Strategico (Redis):** Il calcolo delle interazioni sociali (es. il grafo delle distanze) e l'aggregazione dei milioni di record necessari a disegnare

le mappe di calore (*Heatmap*) sul *frontend* richiedono uno sforzo computazionale massivo. Per evitare di bloccare il database relazionale ri-eseguendo questi calcoli a ogni ricaricamento della *dashboard*, l'`analysis-service` **precompila le statistiche e le mappa all'interno di Redis Cache** (su AWS ElastiCache).

Grazie a questa architettura, quando l'allevatore o il veterinario accede al sistema tramite la SPA in React, i microservizi non interrogano i file originari, ma prelevano i dati clinici già aggregati direttamente dalla RAM di Redis, garantendo tempi di caricamento quasi istantanei nonostante le moli di dati generate a monte.

Capitolo 4

Livello di Presentazione e Progettazione UX/UI

L'architettura di *backend*, le code asincrone e il database *Time-Series* illustrati nel capitolo precedente costituiscono il solido "motore" del progetto SIMBA. Tuttavia, un *Decision Support System* (DSS) fallisce il suo obiettivo primario se la potenza computazionale non viene tradotta in un'interfaccia accessibile. Questo capitolo analizza l'ultimo tassello ingegneristico del sistema: la progettazione e lo sviluppo del Livello di Presentazione (*Frontend*).

4.1 Il Problema: Carico Cognitivo e Complessità del Dato

Come anticipato nel Capitolo 1, il problema cronico della zootecnia moderna non è la mancanza di dati, ma l'eccesso di informazioni destrutturate, che genera un profondo **sovraccarico cognitivo** per l'operatore umano.

Dal punto di vista puramente informatico, l'architettura di *Data Fusion* (eseguita su PostgreSQL) svolge il suo compito , incrociando i *log* del robot Lely con l'*output* spaziale del modello AI (YOLO). Tuttavia, il risultato grezzo di questa computazione è rappresentato da tabelle relazionali contenenti milioni di record testuali e numerici: coordinate X e Y campionate al secondo, codici identificativi dei bovini, *timestamp* millisecondo per millisecondo e valori decimali di conducibilità del latte.

Consegnare questo dato "grezzo" a un medico veterinario o a un allevatore durante una normale ispezione in stalla equivarrebbe a rendere il software del tutto inutile. L'utente finale opera in un ambiente fisico dinamico, spesso utilizzando dispositivi mobili (*tablet* o *smartphone*) in condizioni di luce non ottimali, e necessita

di risposte immediate a domande cliniche complesse: *"Quale vacca rischia la mastite ?"*, *"Il gruppo sta riposando abbastanza nelle cuccette?"*.

Il problema ingegneristico affrontato in questa fase non è stato quindi la gestione del dato, ma la sua **traduzione semantica e visuale**. L'obiettivo è stato progettare un'interfaccia capace di astrarre la complessità matematica sottostante, trasformando milioni di coordinate grezze in indicatori visivi (*Data Visualization*) immediatamente azionabili, riducendo il tempo di diagnosi da diverse ore di analisi manuale a pochi secondi di consultazione a schermo.

4.2 Metodologia di Design Iterativo (HCI) e Prototipazione

L'implementazione del *client* web ha seguito i rigorosi principi dell'Ingegneria dell'Usabilità (*Human-Computer Interaction* - HCI). In perfetta coerenza con l'approccio *Agile* adottato per lo sviluppo del *Frontend* (come descritto nel Capitolo 2), la progettazione dell'interfaccia ha seguito un ciclo di vita iterativo e incrementale.

Invece di procedere direttamente con la stesura del codice React, col rischio di sviluppare cruscotti incomprensibili per i medici veterinari, il team ha validato le interfacce attraverso tre fasi di prototipazione a fedeltà crescente, raccogliendo continui *feedback* dagli *stakeholder* (i ricercatori di UniTo).

4.2.1 Low-Fidelity Prototype

Il processo è iniziato con la creazione di *wireframe* a bassa fedeltà. In questa fase esplorativa, l'obiettivo primario non era la resa estetica, bensì la definizione della corretta **Architettura dell'Informazione**. Attraverso bozze essenziali (schemi a blocchi e schizzi concettuali), il team si è confrontato con i veterinari per stabilire le gerarchie e le priorità operative. Nello specifico, si è delineato il flusso logico dell'applicativo: la schermata principale è stata progettata per consentire la creazione di "Progetti", all'interno dei quali l'utente può raggruppare ordinatamente i *file* video. Questa struttura permette di mandare in elaborazione massiva i dati verso il modulo di Intelligenza Artificiale e, una volta terminato il processo, di sbloccare l'accesso alla fase di visualizzazione clinica.

Figura 4.1: Evoluzione del *wireframe* Lo-Fi. In questa fase ci si concentra esclusivamente sulla disposizione spaziale dei macro-componenti e sul raggruppamento in "Progetti" .

Per quanto riguarda l'ambiente di visualizzazione clinica vero e proprio, l'interfaccia è stata ingegnerizzata adottando un *design pattern* basato su **strutture a *widget*** (moduli a schede indipendenti). Questa scelta permette di ridurre il sovraccarico cognitivo: ogni modulo si presenta inizialmente in una forma compatta, offrendo una *preview* sintetica degli indicatori chiave. Qualora il veterinario necessiti di indagare un'anomalia, può espandere il singolo *widget* per accedere al dettaglio analitico profondo.

Tra i moduli informativi spiccano:

- ***Widget* Heatmap Spaziale:** Un modulo dedicato alla generazione delle mappe di calore, essenziale per la valutazione immediata delle zone di stazionamento e dell'occupazione delle cuccette.
- ***Widget* Grafo delle Interazioni (SNA):** Una rappresentazione visiva interattiva a nodi per esplorare le reti di prossimità sociale e isolare i capi emarginati dal gruppo.
- ***Widget* Mappa 2D:** Una mappa in pianta che visualizzasse spazialmente i bovini permettondo all'utente di filtrarli per ID.

4.2.2 Mid-Fidelity Prototype

Superata la fase concettuale dei *wireframe*, il processo di *design* è avanzato verso la prototipazione a media fedeltà (Mi-Fi). Se il *Low-Fidelity* aveva lo scopo di definire "cosa" mostrare all'utente (l'Architettura dell'Informazione), il *Mid-Fidelity* ha l'obiettivo di stabilire **"come" strutturare visivamente queste informazioni** all'interno dello spazio a disposizione sullo schermo.

Nel contesto del progetto SIMBA, la prototipazione Mi-Fi è risultata determinante per ottimizzare il dimensionamento spaziale dei *widget* analitici e per riorganizzare le gerarchie visive all'interno della schermata dedicata ai "Progetti". Nello specifico, questa iterazione ha permesso di perfezionare la *User Experience* legata all'ingestione dei dati, rendendo fluido e a prova di errore il delicato processo di associazione logica video e i *log* CSV estratti dal robot Lely.

Come si evince dalle interfacce realizzate, la *dashboard* funge da vera e propria cabina di regia. La presenza della barra di scorrimento temporale globale (*Time Slider*) permette di sincronizzare simultaneamente la riproduzione del flusso video reale con il movimento dei *marker* sulla planimetria digitale (Mappa 2D).

Scorrendo l'interfaccia, l'utente ha accesso immediato ai moduli di analisi profonda:

- Il **Tracciamento percorso**, che permette di isolare la traiettoria di un singolo capo (es. ID: 91) per valutarne la fluidità di movimento e le distanze percorse.

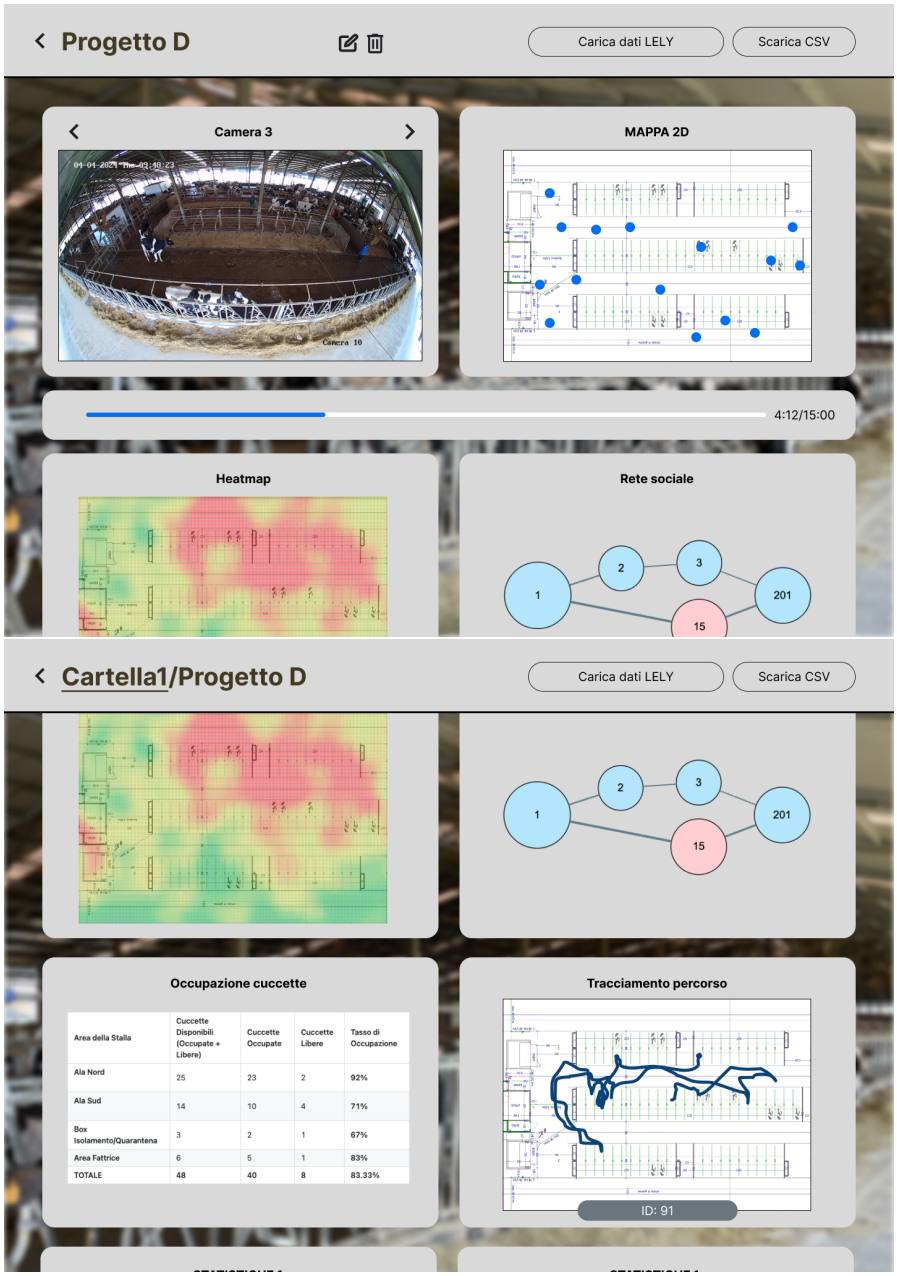

Figura 4.2: Viste di dettaglio della *dashboard* clinica.

- La **Rete sociale**, dove la topologia a grafo identifica le interazioni di prossimità. Risulta evidente l'efficacia del contrasto cromatico: i capi integrati nel gruppo sono azzurri, mentre il nodo isolato o anomalo (es. ID: 15) viene segnalato in rosso, suggerendo un potenziale stato di malessere o allontanamento volontario.

- La **Tabella di occupazione cuccette**, che traduce la complessità spaziale in metriche gestionali pure, indicando il tasso percentuale di occupazione per ogni singola cuccetta della stalla.

4.2.3 High-Fidelity Prototype

L'ultimo step del processo iterativo ha portato alla realizzazione dei prototipi ad alta fedeltà (Hi-Fi). In linea con i paradigmi di sviluppo aziendale strutturato, questa fase ha previsto una netta separazione delle competenze . Mentre la definizione dell'architettura dell'informazione, dei flussi operativi e dei vincoli tecnici (fasi Lo-Fi e Mi-Fi) è stata condotta integralmente dal *team* di Ingegneria del Software, la declinazione estetica finale è stata affidata a un *UI/Graphic Designer* specializzato.

L'intervento di questa figura professionale ha permesso di trasformare l'impalcatura grezza a *widget* in un'interfaccia utente moderna, coerente e accessibile. In questa fase sono state normalizzate le direttive visive di dettaglio: la gerarchia tipografica, la consistenza delle spaziature (*padding* e *margin*) e, fattore cruciale per un *Decision Support System*, la definizione di una **palette cromatica funzionale**.

Nel contesto clinico del progetto SIMBA, infatti, l'uso del colore travalica la pura estetica per diventare uno strumento diagnostico immediato:

- È stata implementata una rigorosa scala cromatica termica per la renderizzazione delle *Heatmap* spaziali (con transizioni dal blu per le zone fredde al rosso per le aree ad altissima densità di occupazione).
- È stato standardizzato un sistema di allerta visiva "a semaforo" (Verde, Giallo, Rosso) per segnalare a colpo d'occhio metriche critiche, quali il Rischio Mastite estratto dai *log* del robot Lely.

La validazione di questi artefatti visivi ha sancito il completamento del *Design Handoff*, ovvero il trasferimento formale delle specifiche grafiche (*asset* vettoriali, codici esadecimali e direttive CSS) al *team* di sviluppo.

Allo stato attuale del progetto, la traduzione di questi *mockup* statici in componenti interattivi tramite la libreria JavaScript **React** è **in fase di implementazione**. Questa transizione architetturale verso una *Single Page Application* (SPA) pienamente reattiva costituisce lo stadio conclusivo dello sviluppo *Frontend*, propedeutico al *deployment* dell'intero ecosistema SIMBA in ambiente di produzione.

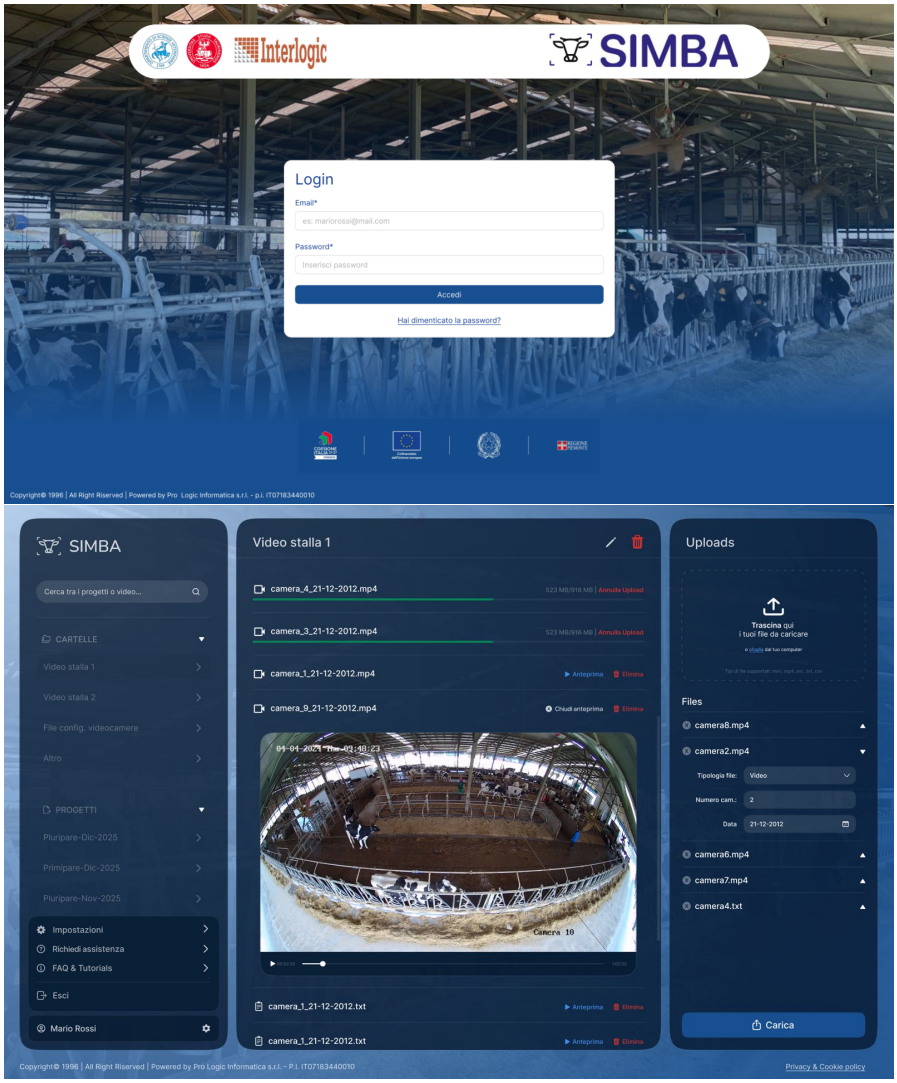

Figura 4.3: Prototipi Hi-Fi dell'interfaccia utente (schermate di *Login* e di Gestione).

Capitolo 5

Futura valutazione della UI

A conclusione del percorso di progettazione e sviluppo del prototipo, è stato definito un rigoroso protocollo per la futura valutazione del sistema tramite test di usabilità. Questi test sono considerati fondamentali per poter valutare e iterare l'esperienza utente (*User Experience*) e l'interfaccia grafica (*User Interface*) prima del rilascio definitivo in ambiente di produzione.

Come evidenziato in letteratura, i test di usabilità servono a comprendere come gli utenti interagiscono con il prodotto e come lo percepiscono, mirando a stabilire quanto sia facile e intuitivo l'utilizzo del *Decision Support System* clinico sviluppato. L'usabilità, infatti, non è una caratteristica intrinseca del codice sorgente, ma una qualità che emerge unicamente dall'interazione tra il sistema e l'utente finale [14, 15, 16, 17].

5.1 Pianificazione

È stato formalizzato un protocollo per l'esecuzione dei test di usabilità al fine di valutare il sistema in termini di efficienza, efficacia e soddisfazione. La valutazione si concentrerà sull'interfaccia web (*Single Page Application* in React) presentata nel capitolo precedente.

Per garantire una valutazione mirata e aderente al dominio di progetto, saranno coinvolti gli *stakeholder* primari del sistema: i medici veterinari, gli allevatori e i ricercatori dell'Università di Torino.

In accordo con le ricerche condotte da Tom Landauer e Jakob Nielsen nell'ambito della *Human-Computer Interaction*, i test verranno effettuati coinvolgendo un campione di **cinque partecipanti per categoria**. La letteratura dimostra infatti che l'incremento della numerosità del campione oltre la quinta persona testata comporta una drastica diminuzione delle nuove informazioni raccolte, poiché le problematiche segnalate tendono a diventare ripetitive.

Per l'esecuzione dei test, si prevede l'allestimento di un ambiente di valutazione controllato, per limitare distrazioni esterne. I partecipanti saranno supportati da un moderatore (facilitatore), il cui ruolo sarà quello di spiegare le attività da eseguire, osservare passivamente i comportamenti e intervenire per indirizzare l'utente solo in caso di blocchi critici nell'esecuzione dei *task*. I test si svolgeranno sia in presenza (direttamente in stalla) sia da remoto tramite piattaforme di videoconferenza (es. Zoom) tramite la condivisione dello schermo.

5.2 Metodologia di Valutazione della Dashboard

Gli strumenti che verranno utilizzati per la valutazione dell'usabilità del prototipo includono:

- **Dispositivo hardware:** computer o *tablet* con l'applicativo React attivo e funzionante;
- **Software di monitoraggio:** applicazione Zoom per consentire il controllo remoto e la registrazione della sessione (ove applicabile);
- **Prototipo del sistema:** l'applicativo SIMBA con dati precaricati (video e log Lely);
- **Materiale di supporto:** carta e penna per registrare le osservazioni (*thinking aloud*) durante il processo.

Le sessioni avranno una durata stimata di 30 minuti per partecipante. Per valutare il sistema nella sua interezza, sono state pianificate quattro attività principali (*Task*), progettate per coprire l'intero flusso operativo dell'applicativo: dall'ingestione dei dati all'analisi clinica complessa. I dettagli del protocollo sono riportati nella Tabella 5.1.

TITOLO	DESCRIZIONE	CRITERI DI SUCCESSO	METRICA
Attività 1: Comprensione pagina e Creazione Progetto	Al partecipante viene chiesto di creare un nuovo progetto, assegnando fisicamente le telecamere sulla planimetria e caricando i file video di calibrazione.	Il partecipante riesce a individuare la funzionalità di *Spatial Mapping* e ad associare correttamente i file senza errori di caricamento.	Successo (1) o fallimento (0).
Attività 2: Analisi cruscotto clinico Lely	Il partecipante deve navigare nella sezione anagrafica e individuare lo stato di salute di un bovino specifico, analizzando gli indicatori a semaforo.	Il partecipante comprende immediatamente le informazioni di rischio mastite e la conducibilità del latte.	Successo (1) o fallimento (0).
Attività 3: Interazione con Mappa 2D e *Heatmap*	Viene richiesto di attivare la *Heatmap* termica e di utilizzare la barra temporale (*Time Slider*) per valutare l'occupazione delle cuccette.	Il partecipante riesce a gestire i filtri visivi e a interpretare correttamente le zone ad alta densità cromatica (in rosso).	Successo (1) o fallimento (0).
Attività 4: Comprensione Grafo Rete Sociale	Il partecipante deve visualizzare il grafo relazionale generato dall'AI e identificare i bovini isolati dal gruppo.	Il partecipante individua correttamente i nodi anomali (colorati in rosso) e deduce lo stato di potenziale malessere.	Successo (1) o fallimento (0).

Tabella 5.1: Attività pianificate per la valutazione dell'applicazione web SIMBA.

5.3 Metriche di Risultato e Analisi Euristica

Durante le sessioni, verrà adottata la tecnica del *Thinking Aloud*: agli utenti sarà richiesto di commentare ad alta voce i propri ragionamenti durante l'interazione. I risultati del processo di valutazione andranno a popolare una matrice di successo (strutturata come in Tabella 5.2), che permetterà al *team* di calcolare il tasso di completamento medio per ogni *task* e per singolo partecipante.

Partecipante	**Attività 1**	**Attività 2**	**Attività 3**	**Attività 4**	**Succ. medio**
P01	[0/1]	[0/1]	[0/1]	[0/1]	[%]
P02	[0/1]	[0/1]	[0/1]	[0/1]	[%]
P03	[0/1]	[0/1]	[0/1]	[0/1]	[%]
P04	[0/1]	[0/1]	[0/1]	[0/1]	[%]
P05	[0/1]	[0/1]	[0/1]	[0/1]	[%]
%Task	[%]	[%]	[%]	[%]	**-**

Tabella 5.2: Struttura predefinita per la raccolta degli esiti delle attività dei partecipanti.

Qualora l'utente non dovesse riuscire a completare un *task*, o nel caso in cui le informazioni fornite dai grafici non risultassero sufficienti per una completa comprensione (violazione dell'euristica numero 10 di Nielsen: "Supporto e documentazione"), l'anomalia verrà registrata per consentire una revisione strutturale del componente React. Tutte le criticità emerse verranno classificate secondo le **10 Euristiche di Nielsen** (ad esempio, analizzando la corrispondenza tra il sistema e il mondo reale - Euristica 2, o la coerenza degli standard visivi - Euristica 4) assegnando a ciascun errore un grado di severità (bassa, media, alta).

5.4 Questionario SUS (System Usability Scale)

Al termine di ogni sessione guidata, è prevista la somministrazione di un questionario **SUS (System Usability Scale)** per raccogliere opinioni e impressioni in modo rapido ed efficace.

Il questionario SUS è uno strumento standardizzato e ampiamente riconosciuto in letteratura, composto da dieci affermazioni (cinque in forma positiva e cinque in forma negativa) che misurano vari aspetti dell'usabilità: dall'apprendimento dell'utente, all'efficienza, fino alla soddisfazione generale . Le risposte vengono raccolte tramite una scala Likert a 5 punti (da "Fortemente in disaccordo" a "Fortemente d'accordo") e successivamente elaborate da un algoritmo di *scoring* per determinare un punteggio totale da 0 a 100.

L'obiettivo ingegneristico per il rilascio in produzione del sistema SIMBA è il raggiungimento di un **punteggio medio superiore a 80/100**. Un punteggio di tale portata confermerebbe l'avvenuto abbattimento del sovraccarico cognitivo, certificando che l'interfaccia a *widget* e le mappe termiche risultano pienamente

comprensibili e azionabili dagli operatori zootecnici, chiudendo così con successo il ciclo di Ingegneria del Software del progetto.

Capitolo 6

Conclusioni

L'obiettivo primario di questo lavoro di tesi è stato la progettazione e lo sviluppo di SIMBA, un *Decision Support System* (DSS) innovativo destinato al settore della zootecnia di precisione (Agricoltura 4.0), partendo dall'analisi di un problema cronico del settore: il sovraccarico cognitivo generato dall'enorme mole di dati destrutturati. Il progetto ha mirato a trasformare le serie storiche spaziali estratte dall'Intelligenza Artificiale e i *log* del robot di mungitura Lely in conoscenza clinica immediatamente azionabile.

6.1 Traguardi Raggiunti

Il percorso ingegneristico illustrato nei capitoli precedenti ha permesso di definire un'infrastruttura solida e scalabile. Applicando rigorosamente i paradigmi dell'Ingegneria del Software e dell'Interazione Uomo-Macchina (*Human-Computer Interaction*), sono stati raggiunti i seguenti traguardi:

- **Gestione della Complessità Architetturale:** è stata definita un'architettura a microservizi *Cloud-Native* in grado di supportare l'ingestione asincrona di file video massivi, orchestrando in *background* i calcoli della *Data Fusion* senza impattare sulla reattività del *client*.
- **Ottimizzazione della User Experience (UX):** l'interfaccia utente è stata progettata attraverso un ciclo iterativo (Lo-Fi, Mi-Fi, Hi-Fi) per prevenire l'errore umano in fase di inserimento dati (es. la mappatura spaziale delle telecamere) e per garantire una curva di apprendimento rapida.
- **Data Visualization Avanzata:** i risultati dei complessi algoritmi di visione artificiale sono stati tradotti in strumenti visivi ad alto impatto cognitivo. L'implementazione concettuale di mappe di calore (*Heatmap*), grafi per l'analisi

della rete sociale (SNA) e cruscotti a semaforo per il Rischio Mastite fornisce ora al medico veterinario un supporto diagnostico istantaneo.

- **Pianificazione del Collaudo:** è stato formalizzato un protocollo di validazione empirica basato sul *System Usability Scale* (SUS), ponendo le basi per una futura misurazione oggettiva delle interfacce.

6.2 Sviluppi Futuri

Allo stato attuale, l'impalcatura logica e visiva del sistema ha superato con successo le fasi di validazione del *design* ed è in corso la sua completa trasposizione in codice, seppur attualmente in fase embrionale .

A testimonianza della validitá e dell'impatto concreto del lavoro svolto durante questo periodo di tesi, la collaborazione con l'azienda si è evoluta in un inserimento professionale a tutti gli effetti. Il sottoscritto è stato formalmente assunto all'interno dell'organico di **Prologic SRL** con il ruolo di *Software Engineer*.

Nei prossimi mesi, il mio impegno professionale sarà dedicato interamente ai seguenti sviluppi futuri:

1. **Sviluppo Frontend in React:** l'implementazione architetturale della *Single Page Application* (SPA) verrà portata a compimento, traducendo i *mockup* Hi-Fi in componenti interattivi e integrando lo strato di gestione dello stato globale per il consumo delle API REST esposte dall'API Gateway.

2. **Integrazione Real-Time:** verranno consolidati i meccanismi di *streaming chunked* per la riproduzione video e il rendering dinamico delle *Heatmap* direttamente lato *client*.

3. **Deployment** il sistema verrà rilasciato nell'infrastruttura Cloud AWS aziendale e introdotto in stalla e reso operativo. In questa fase, verranno condotti i test di usabilità (SUS) delineati nel Capitolo 5, iterando il codice React sulla base dei *feedback* reali raccolti dagli allevatori.

In conclusione, il progetto SIMBA non rappresenta soltanto il culmine di un percorso di studi magistrale, ma il punto di partenza di una carriera professionale mirata a colmare il divario tra la ricerca e la creazione di interfacce software capaci di generare un reale valore per l'utente finale.

Appendice A

Script per la Valutazione del Prototipo

Questo documento rappresenta lo *script* standardizzato che il moderatore utilizzerà per introdurre i test di usabilità agli *stakeholder* (allevatori e medici veterinari) prima dell'interazione con l'applicativo SIMBA.

Fase 1: Presentazione e Accoglienza

"Buongiorno e grazie per aver accettato di partecipare a questo studio. Il progetto che stiamo sviluppando, denominato SIMBA, è un *Decision Support System* pensato per aiutarvi a monitorare il benessere della mandria integrando i dati spaziali con i *log* del robot Lely.

Oggi testeremo insieme l'interfaccia di questo sistema. Vorrei sottolineare che l'obiettivo di questa sessione è **testare il software, non le sue abilità**. Se incontra delle difficoltà o trova delle schermate poco chiare, è un difetto del nostro *design* e i suoi *feedback* ci aiuteranno a migliorarlo. La prego di comportarsi come farebbe in una normale giornata di lavoro."

Fase 2: Istruzioni Operative (***Thinking Aloud***)

"Il sistema che utilizzerà oggi è un prototipo avanzato. Le chiederò di completare 4 specifiche attività (ad esempio, creare un nuovo progetto o visualizzare una mappa di calore). Mentre esegue queste operazioni, le chiedo la cortesia di **pensare ad alta voce** (*Thinking Aloud*): mi descriva cosa sta guardando, cosa sta cercando di fare e se qualcosa non le è chiaro. Io sarò qui accanto a lei per osservare, ma interverrò solo se dovesse trovarsi in un blocco totale."

Fase 3: Debriefing

"Abbiamo concluso le attività previste. Prima di salutarci, le chiedo qualche minuto per compilare un breve questionario di valutazione (SUS) per quantificare la sua esperienza d'uso. Ha qualche domanda, dubbio o consiglio aggiuntivo su quanto ha appena visto?"

Appendice B

Questionario System Usability Scale (SUS)

Di seguito è riportato il questionario standardizzato SUS (*System Usability Scale*) che verrà sottoposto agli utenti al termine delle sessioni di *Beta Testing* dell'applicativo SIMBA. I partecipanti dovranno esprimere il proprio grado di accordo utilizzando una scala Likert a 5 punti, dove 1 corrisponde a "Fortemente in disaccordo" e 5 a "Fortemente d'accordo".

Affermazione	**1**	**2**	**3**	**4**	**5**
1. Penso che mi piacerebbe utilizzare questo sistema frequentemente per il monitoraggio della stalla.	◯	◯	◯	◯	◯
2. Trovo che questo sistema sia inutilmente complesso.	◯	◯	◯	◯	◯
3. Penso che il sistema sia facile da utilizzare.	◯	◯	◯	◯	◯
4. Penso che avrei bisogno del supporto di un tecnico informatico per poter utilizzare il sistema.	◯	◯	◯	◯	◯
5. Trovo che le diverse funzioni di questo sistema (es. mappe 2D, cruscotti Lely) siano ben integrate.	◯	◯	◯	◯	◯
6. Trovo che ci sia troppa incoerenza in questo sistema.	◯	◯	◯	◯	◯
7. Immagino che la maggior parte dei colleghi imparerebbe ad utilizzare il sistema molto rapidamente.	◯	◯	◯	◯	◯
8. Trovo il sistema molto faticoso e macchinoso da utilizzare.	◯	◯	◯	◯	◯
9. Mi sono sentito molto sicuro nell'utilizzare il sistema per valutare i dati clinici.	◯	◯	◯	◯	◯
10. Ho avuto bisogno di imparare molte cose prima di poter cominciare ad utilizzare il sistema in autonomia.	◯	◯	◯	◯	◯

Tabella B.1: Modulo cartaceo del questionario SUS (*System Usability Scale*) tradotto e contestualizzato per il progetto SIMBA.

Note per il calcolo del punteggio (ad uso del team di sviluppo):

- Per gli *item* dispari (1, 3, 5, 7, 9), il punteggio è la risposta dell'utente meno 1.
- Per gli *item* pari (2, 4, 6, 8, 10), il punteggio è 5 meno la risposta dell'utente.
- Sommare i punteggi di tutti gli *item* e moltiplicare il totale per 2,5 per ottenere l'indice globale (da 0 a 100).

Bibliografia

[1] Lely International N.V. *Lely Astronaut: Robot di mungitura automatico*. https://www.lely.com/it/soluzioni/mungitura/astronaut/. 2025 (cit. a p. 6).

[2] Laura Ozella, Mario Giacobini, Elena Vicuna Diaz, Achille Schiavone e Claudio Forte. «A comparative study of social behavior in primiparous and multiparous dairy cows during automatic milking». In: *Applied Animal Behaviour Science* 268 (2023), p. 106065. ISSN: 0168-1591. DOI: https://doi.org/10.1016/j.applanim.2023.106065. URL: https://www.sciencedirect.com/science/article/pii/S016815912300237X (cit. a p. 6).

[3] TechTarget. *MoSCoW Method*. https://www.techtarget.com/searchsoftwarequality/definition/MoSCoW-method. 2024 (cit. a p. 16).

[4] PTC. *When, Why, and How to Use the Agile-Waterfall Hybrid Model*. https://www.ptc.com/en/blogs/alm/when-why-how-to-use-the-agile-waterfall-hybrid-model. 2023 (cit. a p. 19).

[5] Amazon Web Services. *Amazon Simple Storage Service (S3) Documentation*. https://docs.aws.amazon.com/s3/. 2026 (cit. a p. 25).

[6] HashiCorp. *Consul by HashiCorp: Service Discovery and Mesh*. https://www.consul.io/. 2026 (cit. a p. 25).

[7] Meta Platforms, Inc. *React: The library for web and native user interfaces*. https://react.dev/. 2026 (cit. a p. 25).

[8] Baeldung. *Spring Boot Reference and Tutorials*. https://www.baeldung.com/spring-boot. 2026 (cit. a p. 25).

[9] PostgreSQL Global Development Group. *PostgreSQL: The World's Most Advanced Open Source Relational Database*. https://www.postgresql.org/. 2026 (cit. a p. 26).

[10] MinIO, Inc. *MinIO: High Performance Object Storage*. https://min.io/. 2026 (cit. a p. 26).

[11] s3fs-fuse Contributors. *s3fs-fuse: FUSE-based file system backed by Amazon S3*. https://github.com/s3fs-fuse/s3fs-fuse. 2026 (cit. a p. 26).

[12] Redis Ltd. *Redis: The open source, in-memory data store*. https://redis.io/. 2026 (cit. a p. 26).

[13] Amazon Web Services. *Amazon ElastiCache for Redis Documentation*. https://aws.amazon.com/elasticache/redis/. 2026 (cit. a p. 26).

[14] Unguess. *Cosa sono i test di usabilità e perché sono importanti*. https://blog.unguess.io/it/what-is-usability-testing. Consultato: Marzo 2026. 2026 (cit. a p. 42).

[15] Jakob Nielsen. «Why you only need to test with 5 users». In: *Nielsen Norman Group* (2000) (cit. a p. 42).

[16] Designers Italia - AgID. *Test di usabilità: Manuale operativo di design*. https://docs.italia.it/italia/designers-italia/manuale-operativo-design-docs/it/versione-corrente/doc/design-research/test-usabilita.html. 2026 (cit. a p. 42).

[17] WebManWalking. *Le 10 Euristiche dell'usabilità di Jakob Nielsen*. https://www.webmanwalking.it/le-10-euristiche-di-nielsen/. 2026 (cit. a p. 42).