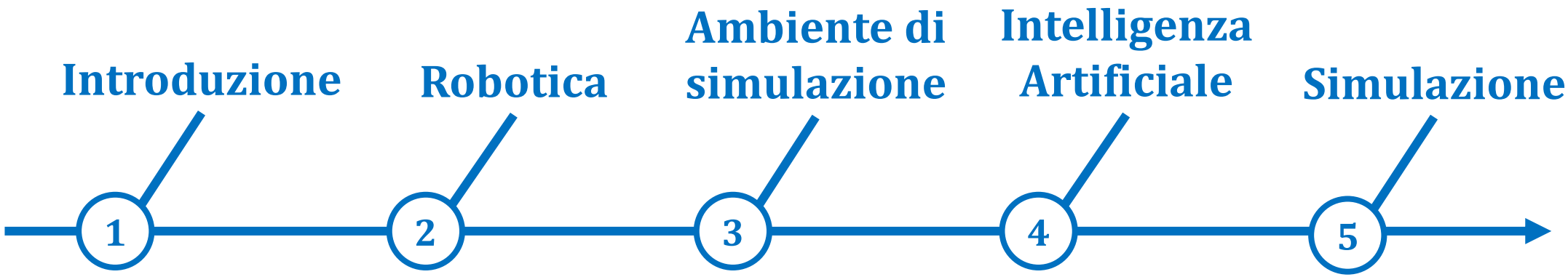
Progetto: Robotica & Intelligenza Artificiale 2

HrR (Hotel receptionist Robot)

Università degli Studi di Palermo

Gabriele Bova, Paolo Manuele Gulotta, Andrea Spinelli, Vincenzo Zizzo

Flusso di Presentazione



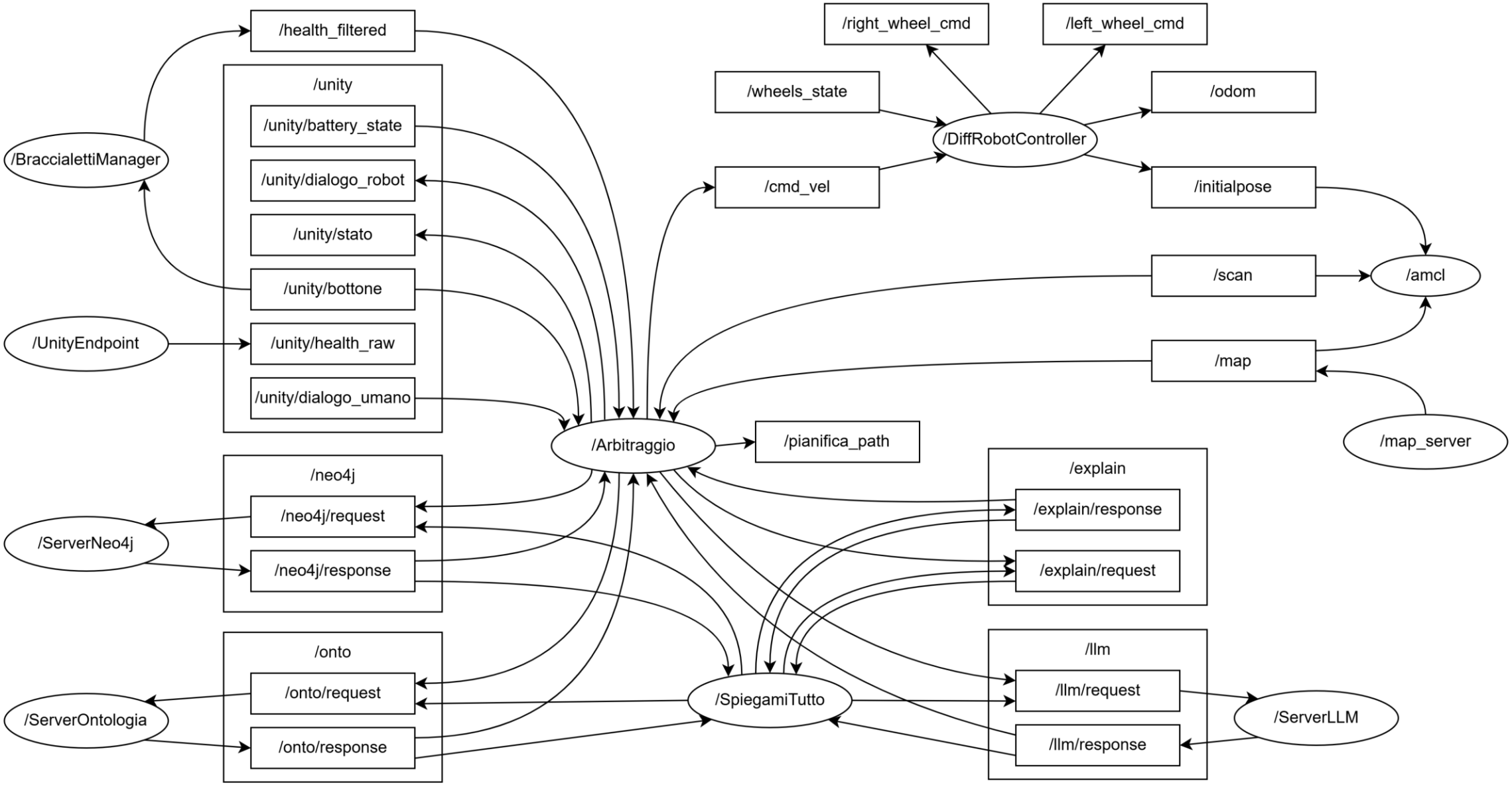
1. Introduzione

- **Pippor – Robot sociale**: dalla gestione dell'accoglienza alla rilevazione di anomalie, Pippor opera come punto di riferimento costante per l'ospite.
- **Ecosistema connesso**: Integrazione in tempo reale tra robotica e dispositivi indossabili per il monitoraggio dei parametri vitali e la gestione ottimizzata delle emergenze.

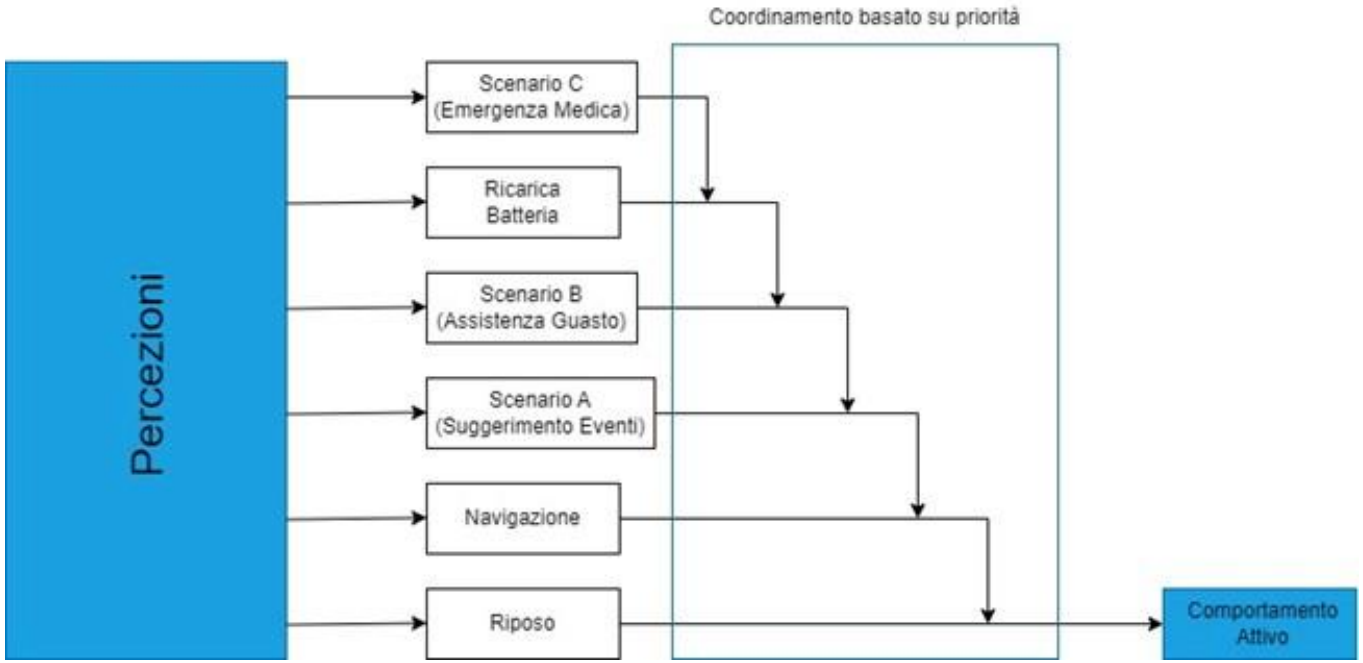
- **Scenario A: Accoglienza e profilazione dinamica**
  - Pippor riconosce l'ospite e ne profila interessi e preferenze in tempo reale, utilizzando un'intelligenza ibrida per offrire suggerimenti personalizzati e gestire ogni esigenza logistica.
- **Scenario B: Assistenza tecnica e manutenzione**
  - Il robot gestisce le segnalazioni di oggetti guasti effettuando un triage diagnostico in camera, filtrando le necessità di intervento tecnico per ottimizzare il lavoro del personale.
- **Scenario C: Gestione emergenze e anomalie**
  - Attraverso il monitoraggio biometrico costante e l'analisi dei trend vitali, il sistema rileva autonomamente situazioni di pericolo o malori, garantendo un'attivazione immediata.

2. Robotica

1. **Architettura**
2. **Arbitraggio**
3. **Pianificazione e Navigazione**
4. **TF e Localizzazione**
5. **Filtro di Kalman**

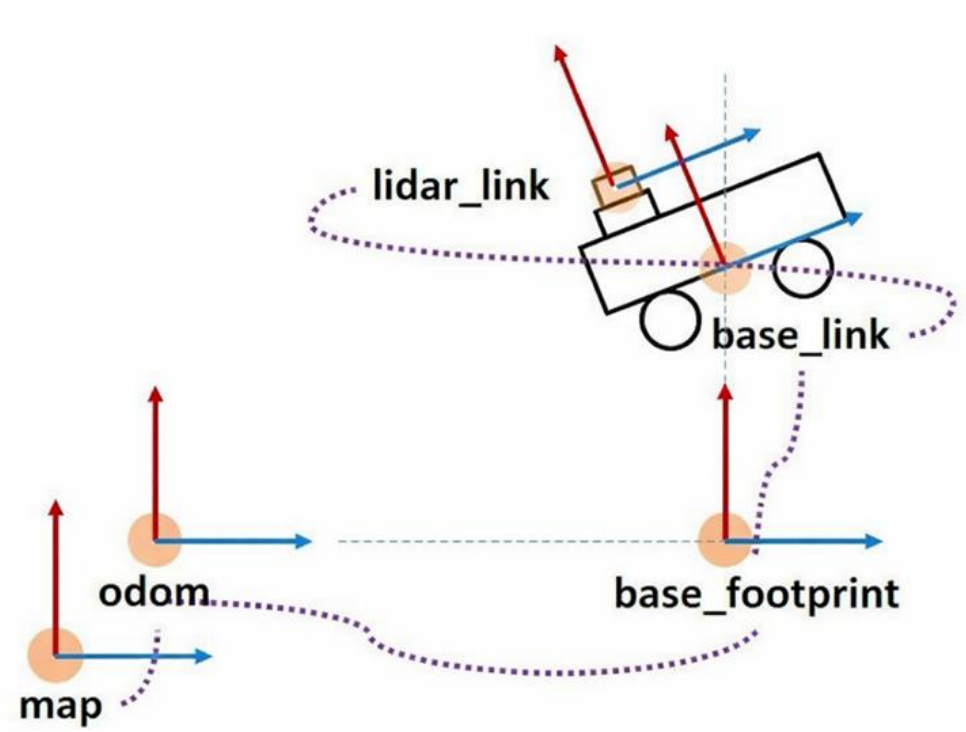

/health_filtered
/unity
/unity/battery_state
/unity/dialogo_robot
/unity/stato
/unity/bottone
/unity/health_raw
/unity/dialogo_umano
/BraccialettiManager
/UnityEndpoint
/right_wheel_cmd
/left_wheel_cmd
/wheels_state
/odom
/DiffRobotController
/cmd_vel
/initialpose
/scan
/amcl
/map
/map_server
/Arbitraggio
/pianifica_path
/neo4j
/neo4j/request
/neo4j/response
/ServerNeo4j
/explain
/explain/response
/explain/request
/onto
/onto/request
/onto/response
/ServerOntologia
/SpiegamiTutto
/llm
/llm/request
/llm/response
/ServerLLM

- **Sistema di Arbitraggio**: Architettura a soppressione (Subsumption) basata su priorità fisse, non una semplice macchina a stati.
- Gerarchia delle priorità:
  - **Sicurezza (Scenario C)**: Salvaguardia dell'ospite (priorità assoluta).
  - **Energia**: Ricarica batteria (sacrificabile in caso di emergenza medica).
  - **Interazione sociali**: Assistenza e Concierge.



- **Pianificazione**: Algoritmo A* custom su griglia 8-connected con prevenzione del taglio degli angoli ("Corner Cutting Prevention").
    - **Corner Cutting Prevention**: Verifica celle adiacenti per evitare collisioni sugli spigoli in diagonale.
    - **Gestione Mappa**: Flag allow_unknown per attraversamento dinamico di aree inesplorate.
    - **Bounding Box (bbox)**: Ricerca confinata in sottofinestre locali per ridurre il carico computazionale.
- **Loop di Controllo**
    - **Percezione**: Ricezione Occupancy Grid su topic /map.
    - **Localizzazione**: Aggiornamento posa via buffer TF2 (map → base_link).
- **Elaborazione e Traiettoria**
    - **Map Inflation**: "Gonfiaggio" ostacoli (SAFETY_RADIUS) con algoritmo ottimizzato per Python (calcolo pre-offset).
    - **Semplificazione**: Metodo simplify_path per ridurre i waypoint ai soli punti di svolta essenziali.
- **Macchina a Stati (Controllo Ibrido)**
    - **Turn-in-place**: Rotazione sul posto (controllore PI) se l'errore angolare supera la soglia.
    - **Drive-to-goal**: Avanzamento proporzionale con correzione di rotta dinamica.
    - **Fluidità**: Passaggio continuo tra waypoint (senza stop) entro il raggio di tolleranza.

- **Localizzazione (AMCL)**: Filtro particellare adattivo che stima la posa (x, y, θ) unendo dati odometrici e laser.
- **Gerarchia spaziale (TF Tree)**
  - **Coerenza spaziale**: Modulo TF2 per mantenere l'albero delle relazioni tra i sistemi di riferimento.
  - **Trasformate statiche (Intra-Robot)**: Definiscono la geometria fisica immutabile (base_link → lidar_link).
- **Catena dinamica e Correzione**
  - **Odometria locale**: Il controller pubblica odom → base_link (fluida ma soggetta a drift).
  - **Correzione globale**: Il sistema di localizzazione pubblica map → odom.
  - **Meccanismo anti-salto**: L'errore accumulato viene compensato facendo "scivolare" l'origine del sistema odometrico rispetto alla mappa.



- **Analisi Biometrica (Filtro di Kalman)**: Nodo custom per filtrare il rumore dei braccialetti smart e predire trend di pressione/battito, scartando anomalie momentanee.

- **Predizione**: La matrice di transizione proietta lo stato nel futuro in base al *dt*, stimando tendenza e valore atteso.

- **Logica di Sicurezza (Anomaly detection)**

  - **Gestione outlier**: Confronto tra misura grezza e predizione; scarto del dato se l'errore residuo supera la soglia dinamica (filtraggio artefatti di movimento).

  - **Persistenza dell'anomalia**: Reset del filtro solo se la deviazione persiste oltre MAX_OUTLIERS (garanzia di reattività a vere emergenze come tachicardie).

  - **Watchdog**: Invalidazione automatica dello stato ospite dopo RESET_TIMEOUT di silenzio radio (disconnessione dispositivo).

3. Ambiente di simulazione

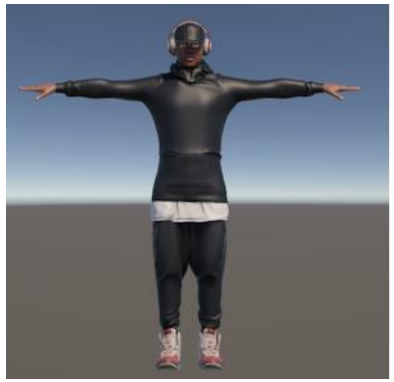
1. **Ambiente**
2. **Robot**
3. **Mappa**

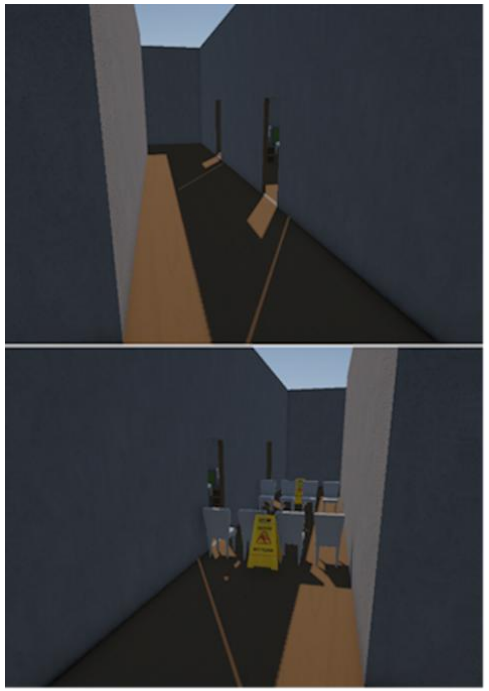
- **Design funzionale**: La scelta di questo setting non è puramente estetica, ma funzionale. L'Hotel è diviso in Reception, Corridoi (sfida per path planning) e CoffeeRoom (ambiente denso non strutturato).

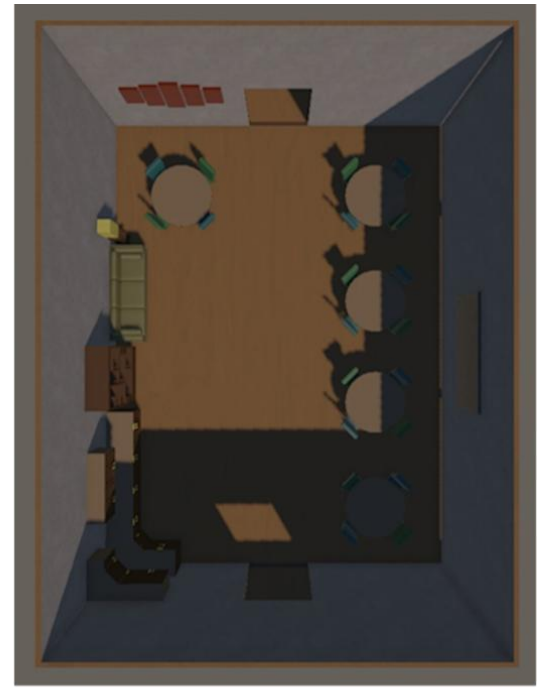
- Ostacoli: La navigazione nell'hotel si basa sul sistema NavMesh (Navigation Mesh) di Unity, che definisce le aree camminabili.

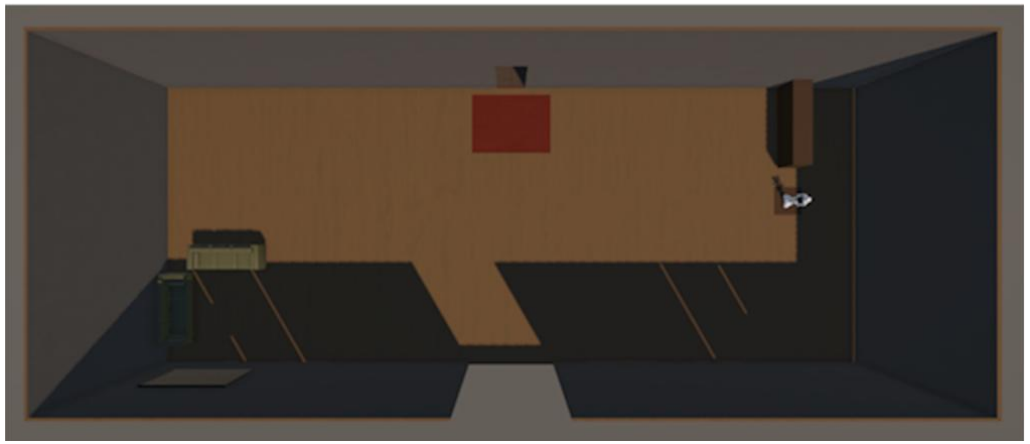
    - **Statici**: Muri e arredi "bakerizzati" nella NavMesh.

    - **Dinamici (NPC)**: Agenti intelligenti con identità e parametri vitali (da Neo4j) che agiscono come ostacoli improvvisi.

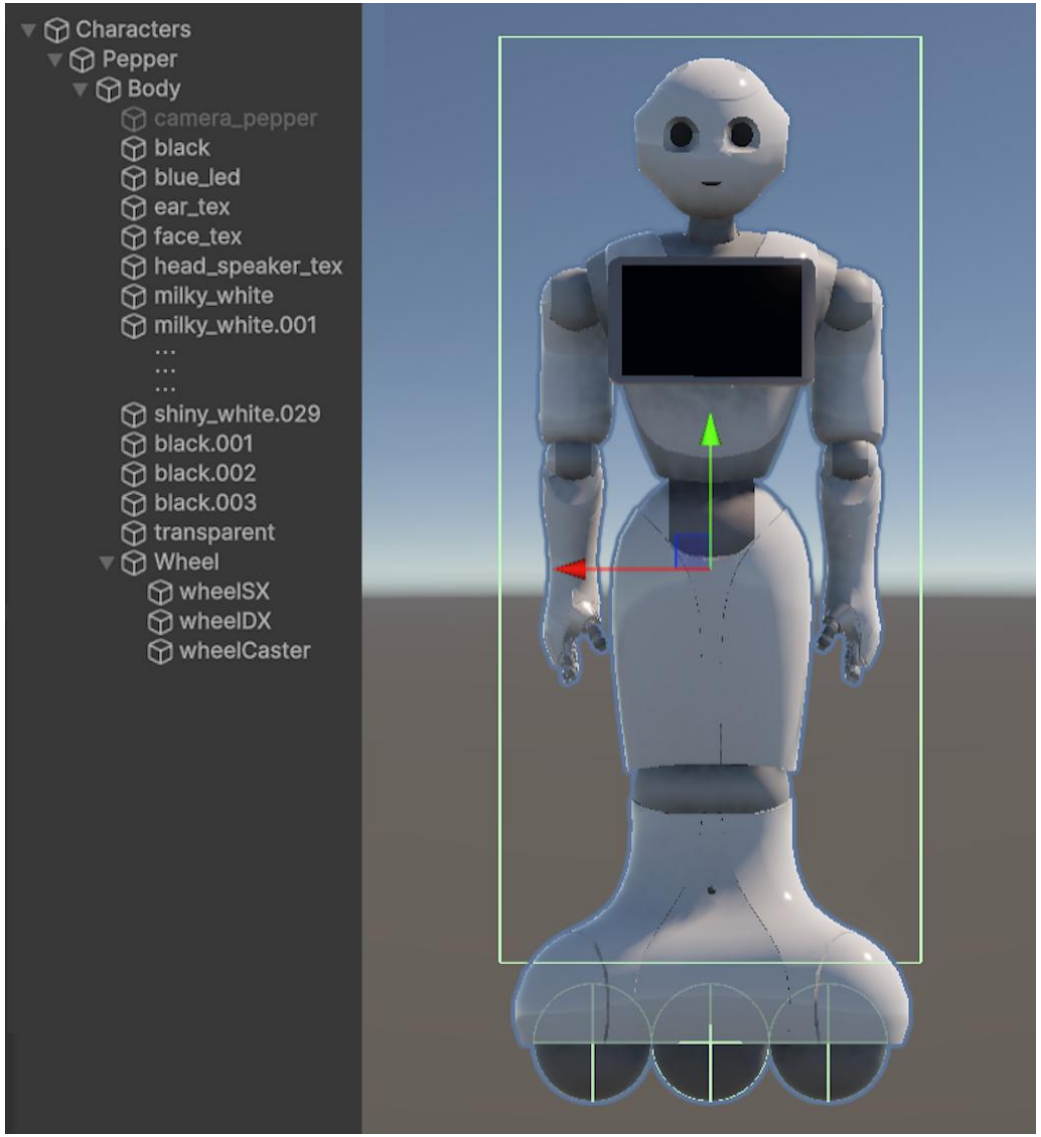




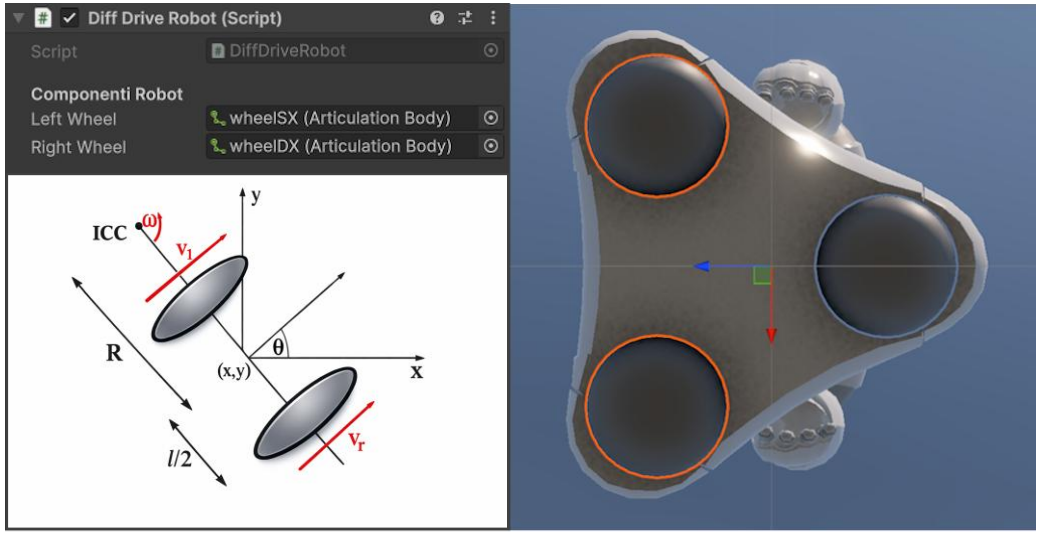


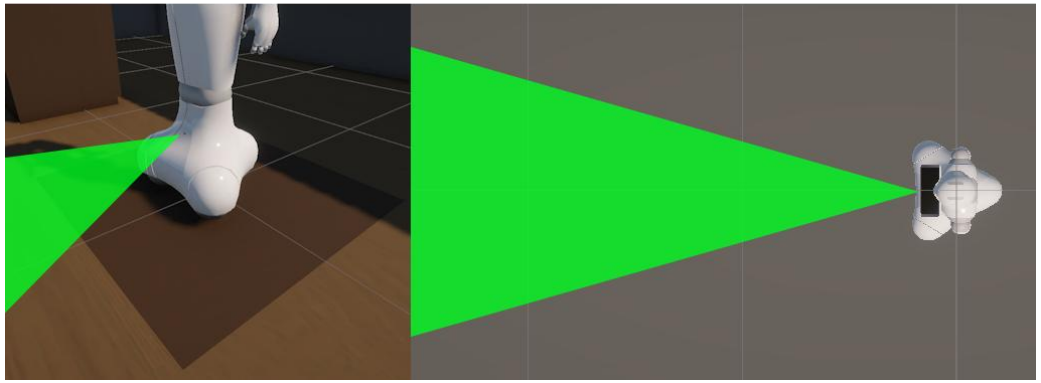


- **Fisica realistica:** L'implementazione non è una semplice animazione, ma una simulazione fisica completa basata su giunti, forze e comunicazione bidirezionale con ROS2. Il robot è stato modellato come un sistema multi-corpo (Multi-Body System):
  - **Chassis (base)**: Il corpo principale, che ospita il computer di bordo simulato e le batterie. Ha una massa definita di circa 28 kg.
  - **Ruote (wheels)**: Il sistema di locomozione, nonché il cuore ingegneristico della simulazione. Usa una configurazione a guida differenziale (2 ruote motrici e 1 omnidirezionale passiva per l'equilibrio).
  - **Batteria (power management)**: Sottosistema implementato nel chassis. Non si limita a decrementare un contatore, ma simula cicli di carica/scarica basati sul tempo e sull'interazione.



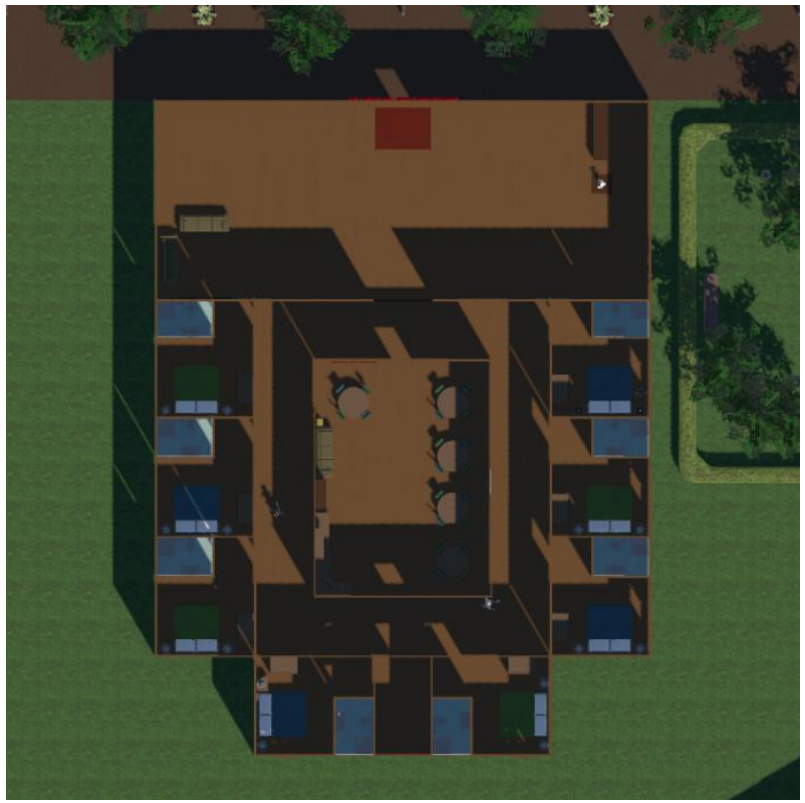
- **Guida differenziale**: Utilizzo di componenti ArticulationBody per stabilità cinematica superiore rispetto ai joint classici.
    - La sfida della massa: Inizialmente il robot risultava "pigro", lento ad accelerare e incapace di frenare tempestivamente; scivolando o muovendosi a scatti.
    - Soluzioni:
        1. Ricalibrare i parametri di guida (force limit, damping, massa delle ruote)
        2. Ricalibrare il Safety Watchdog per la frenata.
- **Sensori (LiDAR)**: Simulazione via Raycasting che interseca i Collider fisici e serializza i dati in messaggi ROS2 standard.

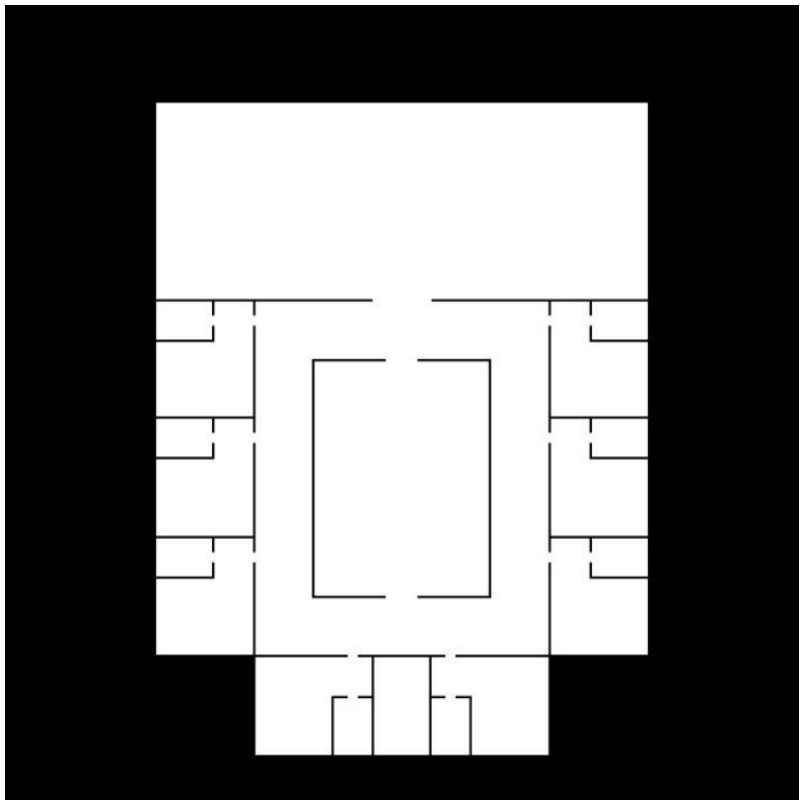




- In un ambiente reale, un robot costruisce la mappa esplorando (SLAM). In un ambiente simulato, abbiamo già la verità assoluta (la scena Unity), ma ROS2 non può leggerla direttamente.

- Il modulo MapExporter è stato sviluppato per colmare questo divario, permettendo di convertire istantaneamente l'ambiente 3D di Unity in una mappa 2D standard (formato .png + .yaml) comprensibile dallo stack di navigazione di ROS2.

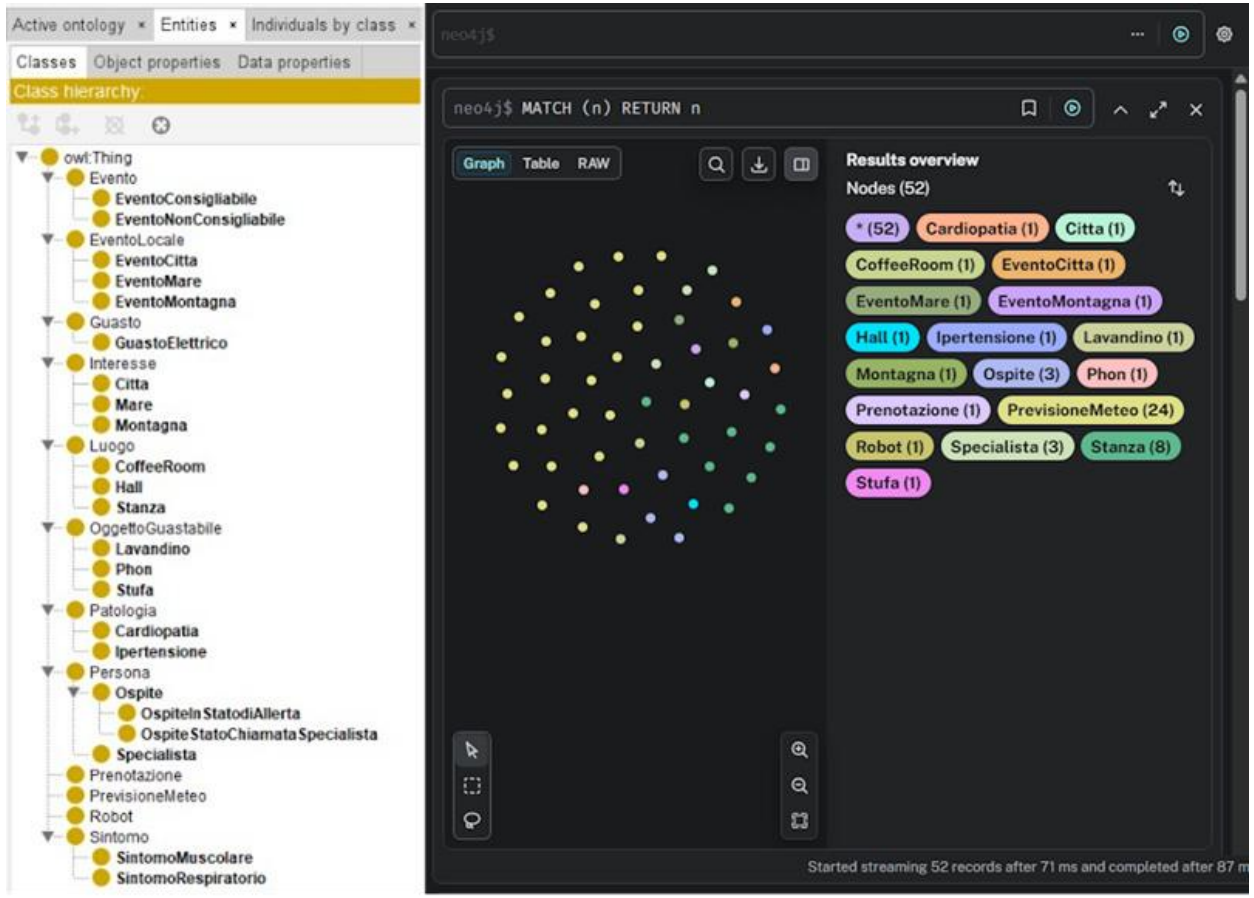




4. Intelligenza Artificiale

1. **Base di conoscenza**
2. **LLM**
3. **Explainability**

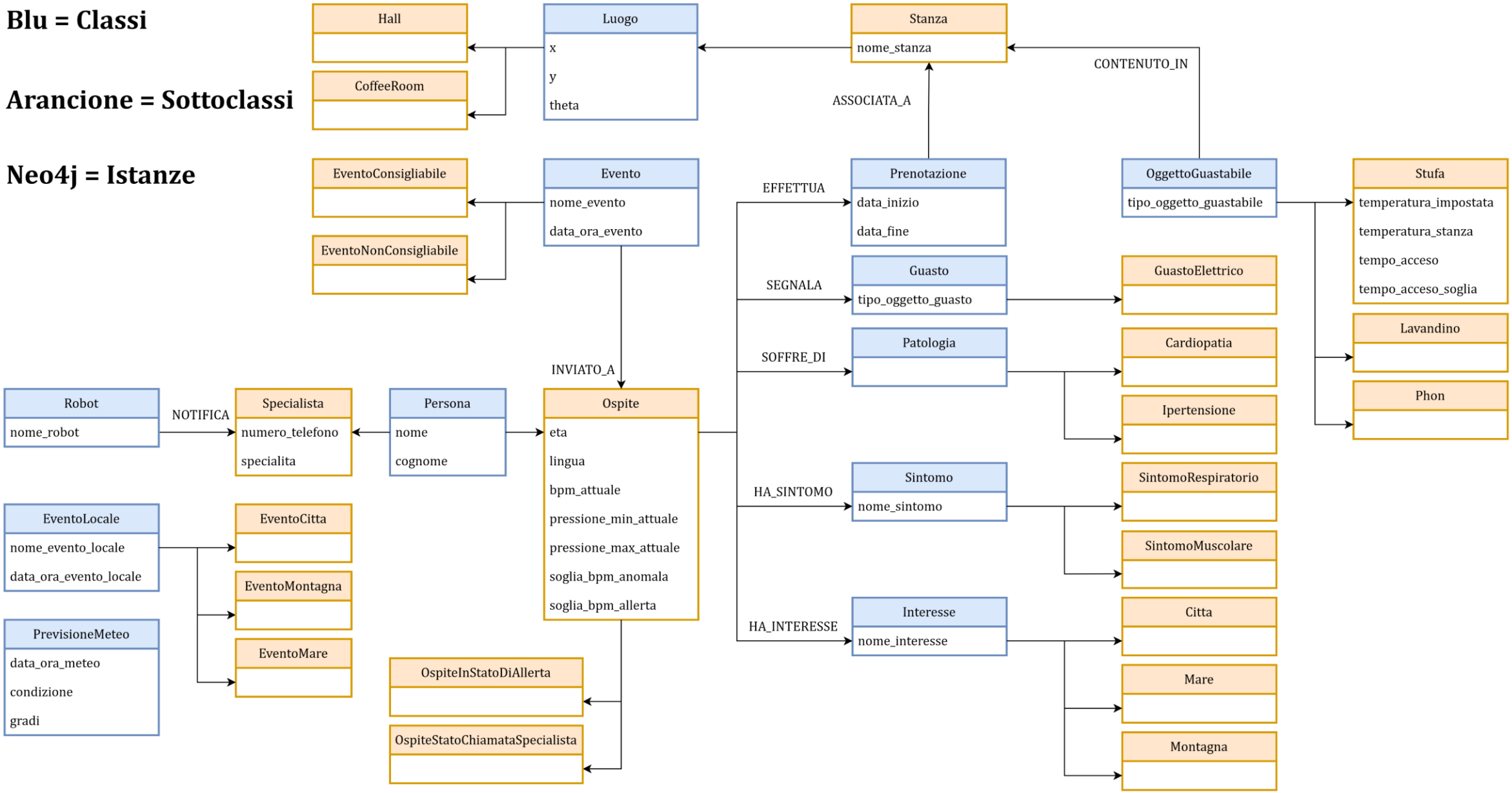
- **Strategia architetturale**
  - L'architettura della base di conoscenza è stata ingegnerizzata separando la fase di definizione strutturale (**TBox**) da quella di popolamento e persistenza delle istanze (**ABox**), al fine di ottimizzare sia il rigore logico che l'efficienza operativa.
- **Stack tecnologico**
  - **Modellazione (Protégé)**: Definizione formale di classi, proprietà e assiomi. Garantisce coerenza semantica e validazione "a monte".
  - **Persistenza (Neo4j)**: Gestione dinamica dei dati e iniezione delle istanze nel sistema. Permette di spostare la struttura logica dell'ontologia in un graph database capace di gestire un grande numero di nodi e relazioni



Blu = Classi

Arancione = Sottoclassi

Neo4j = Istanze



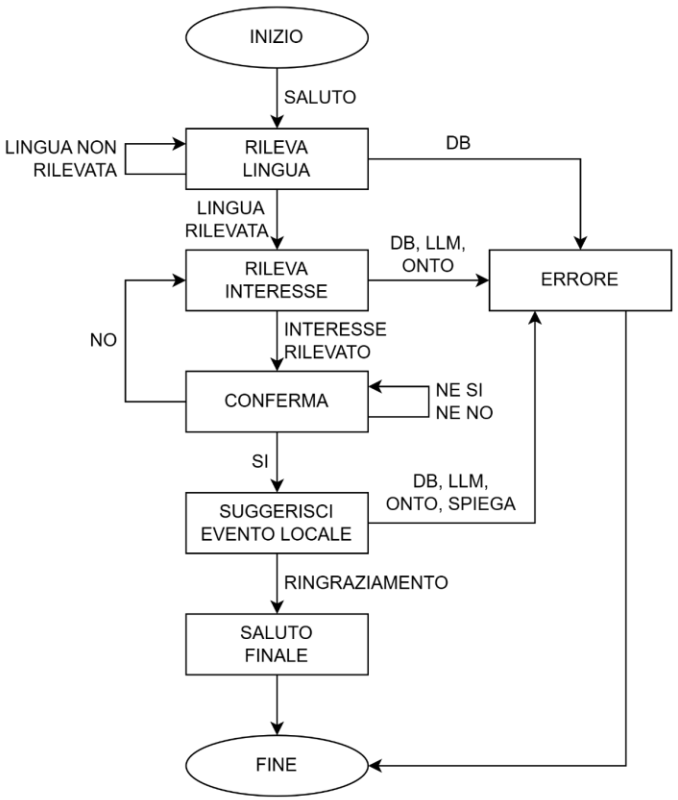
- **Ruolo e confini**
  - **Scope limitato**: Adozione strettamente confinata a livello di presentazione (Interazione Uomo-Robot).
  - **Vincolo di sicurezza**: Esclusione tassativa dai processi decisionali e pianificazione delle azioni.
  - **Tecnologia**: Integrazione API gemini-2.5-flash.
- **Funzionamento (Forma vs Contenuto)**
  - **Natural Language Generation (NLG)**: L'LLM cura solo la forma (risposte discorsive, empatiche).
  - **Contenuto deterministico**: Il contenuto informativo proviene esclusivamente dalla base di conoscenza.
  - **Mitigazione rischi**: Drastica riduzione delle "allucinazioni" operative disaccoppiando la logica dalla generazione del testo.

- **Strategia logica (OWA)**
  - **Problema**: Ambiguità della Open-World Assumption.
  - **Classificazione attiva**: Distinzione netta tra classi positive e negative per evitare deduzioni incerte.
- **Pipeline architetturale**:
  1. Estrazione on-the-fly del sottografo rilevante Neo4j.
  2. Transcodifica in OWL temporaneo
  3. OWLAPI e Openllet per robustezza inferenziale.
  4. Uso di Java per accedere a Default-ExplanationGenerator (non disponibile in Python).
- **Output e sicurezza**
  - **Tracciabilità**: Ricostruzione della catena di assiomi inferita de regole SWRL.
  - **Explainability del Reasoner**: Output JSON con mappatura Soggetto + Spiegazione stato soggetto.

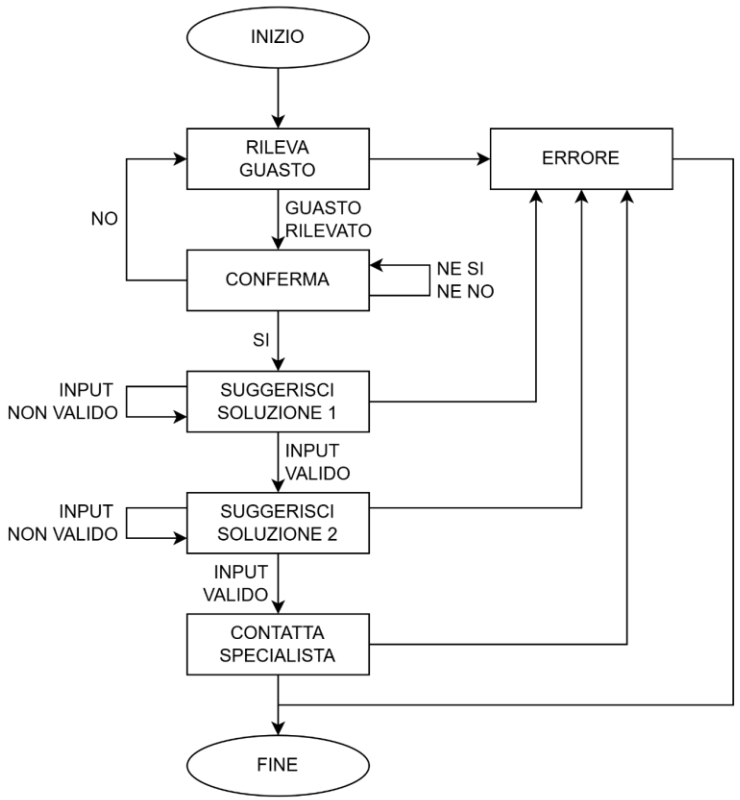
5. Simulazione

1. **Scenario A**
2. **Scenario B**
3. **Scenario C**

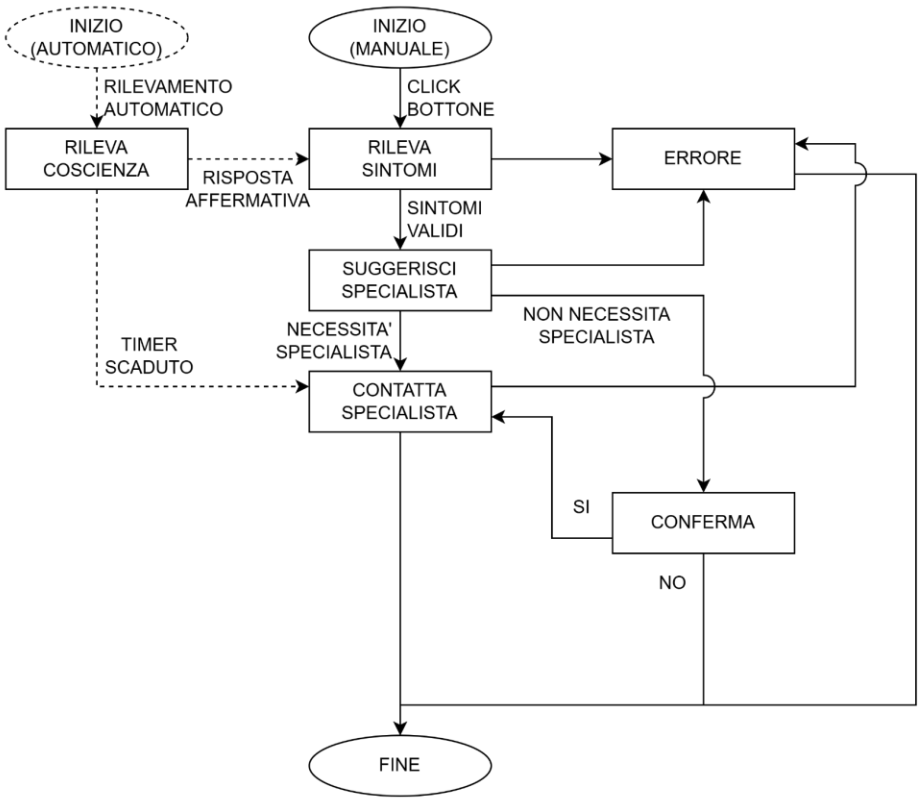
Scenario A



Scenario B



Scenario C



Università degli Studi di Palermo

Grazie per l'attenzione

Gabriele Bova, Paolo Manuele Gulotta, Andrea Spinelli, Vincenzo Zizzo