Università degli Studi di Padova
Laurea: Informatica
Corso: Ingegneria del Software
Anno Accademico: 2025/2026

Gruppo: NullPointers Group
Email: nullpointersg@gmail.com

Verbale Riunione

10 Dicembre 2025

Presenze	Marco Brunello Laura Pieripolli Matteo Mazzaretto Lisa Casagrande Luca Marcuzzo Tommaso Ceron
Destinatari	NullPointers Group Ergon Informatica Srl

Indice

1 Informazioni generali

- **Tipo di riunione:** Esterna
- **Luogo:** Google Meet
- **Data:** 10/12/2025
- **Ora inizio:** 15:00
- **Ora fine:** 16:30
- **Scriba:** Matteo Mazzaretto
- **Partecipanti:**
  - Carlesso Gianluca

2 Ordine del giorno

Secondo colloquio con la proponente^G Ergon Informatica.
In questo incontro, svolta a distanza tramite la piattaforma Google Meet, discutiamo principalmente gli scenari dei casi d'uso e dei requisiti.
Oltretutto, poniamo alcune domande e dubbi che ci sono sorti:

- È preferibile utilizzare il formato TOON oppure il più classico formato JSON?
- Con quale tecnologia si implementa l'API^G REST? Suggerimenti a riguardo
- Si può considerare il sistema di machine learning come un qualcosa definito e modificato da noi, e quindi non considerabile né attore^G principale né attore^G secondario?

3 Diario della riunione

3.1 Formato TOON oppure formato JSON

È attualmente in corso un ampio dibattito sull'argomento. Sicuramente è utilissimo per risparmiare token, però i modelli attuali, essendo stati addestrati su JSON^G o CSV potrebbero avere difficoltà a comprenderlo. consiglia comunque di rimanere sul JSON.

3.2 API REST, quale tecnologia lo implementa e suggerimenti a riguardo

API^G REST^G è il passaggio fra front-end e backend. Una tecnologia consigliata è FastAPI su Python, molto utilizzato al giorno d'oggi.

3.3 Il sistema di machine learning è a carico nostro o ci sono delle parti non controllate pienamente da noi?

Non si va a modificare il modello con tutti i parametri, lo si addestra sui nostri dati nel senso che si fanno conoscere i dati che abbiamo ma deve imparare a gestirlo. Semplicemente si passano informazioni aggiuntive.

3.4 Come funziona il caricamento dei dati?

Langchain, framework^G Python^G per caricare i dati (modello e versione). Tramite questa serie di API^G c'è la possibilità di passare il contesto su cui noi vogliamo operare. Si passa il dataset.

3.5 Scenario dei casi d'uso

Abbiamo discusso dei vari scenari. Il primo scenario trovato è la registrazione, che comprende sia il caso di utente già registrato che in registrazione.
Questo porta direttamente all'autenticazione con il caso d'uso^G di autenticazione non riuscita.
Successivamente si è discusso dello scenario di invio dell'ordine, che può essere inviato tramite input testuale, immagine e audio.
Per quanto riguarda lo scenario dell'ordine, dopo l'inserimento si può visualizzare l'anteprima dell'ordine, confermare o non confermare quello che si è generato, chiedere l'intervento di un operatore, selezionare l'opzione corretta fra le ambiguità. Ma l'utente non può scrivere una form ed inserire prodotti manualmente. Si può inserire una conversazione fra utente e chatbot, sia separata che stesso flusso. È stato chiarito che per "gestione delle ambiguità" si intende la scelta del prodotto corretto oppure questa chat. L'azienda ha confermato come obbligatoria la scelta da parte dell'utente del prodotto corretto ma opzionale l'implementazione del chatbot. Lo scenario termina con la conferma e l'invio dell'ordine.

È stato discusso lo scenario di invio dati da parte dell'admin per il training, l'esportazione del log degli ordini per analisi esterne, il consulto dello ordini, sia da parte dell'admin relativamente a tutti i clienti sia da parte del cliente per il suo storico. Questo caso d'uso^G si soddisfa con una pagina relativa.
L'admin può inoltre verificare lo stato dei servizi AI, creare nuovi utenti, cambiare i permessi di questi ultimi.
Lo scenario del logout è facilmente comprensivo del caso d'uso^G del logout dall'utente.
Ci è stato detto dall'azienda che visualizzare o terminare le sessioni attive è un caso d'uso^G che si può tranquillamente trascurare perché non molto sensato.
Il caso d'uso^G di duplicazione ordine è stato confermato, poiché utile per alcuni utenti che ordinano buona parte delle volte gli stessi prodotti.
Il filtro degli ordini è stato mantenuto, che può essere per periodo (da giorno x a giorno y), dei mesi, oppure per prezzo (crescente o decrescente). L'admin può inoltre scaricare il log della pipeline di un ordine, addestrare manualmente il modello AI, pianificare retraining automatizzato, configurare integrazione ERP, pulire i dati nel training (rimozione di dati rumorosi nel dataset), testare connessione ERP, monitorare utilizzo modelli AI.
Il gruppo ha anche proposto lo scenario di gestione notifiche, sia da parte dell'admin sia da parte del cliente.
Gli ultimi scenari di cui abbiamo discusso sono l'eliminazione dell'account, visualizzare l'help e la documentazione e segnalazione di un bug.

3.6 Requisiti funzionali

3.6.1 Requisiti funzionali obbligatori

Si è discusso successivamente di una parte dei requisiti funzionali.

1. È stato confermato l'obbligatorietà di invio input tramite testo oppure audio eseguendo una fase di pre-processing.
2. L'ordine deve essere generato in formato JSON^G solo dopo la conferma dell'ordine.
3. Nel caso di mancata quantità il chatbot richiede il numero di colli voluti dal cliente, nel caso di ambiguità si richiede la scelta.
4. È stata discussa l'obbligatorietà della registrazione dei log completi e tracciamento delle fasi di elaborazione, confermando l'obbligo di questo. Potrebbero essere interessanti le informazioni come timestamp, invio utente.
5. Il sistema deve essere in grado di scalare fra ordini specifici e meno specifici per permettere a tutti gli utenti di completare l'ordine.
6. Il formato audio deve avere una durata massima di 2 minuti, accettare solo audio con dimensione massima 10MB, supporto dei formati .mp3,

.m4a, .wav. Oltretutto dentro la chat deve essere permessa la registrazione per non scomodare l'utente con i limiti mantenuti precedentemente. La trascrizione dell'audio deve essere in codifica utf-8.

7. È stato deciso di non occuparsi di eventuale indisponibilità dei prodotti.

3.6.2 Requisiti funzionali desiderabili

1. Fusione multimodale avanzata tramite embedding unificati, ovvero possibilità di invio ordine con più modalità. È stato detto che è effettivamente complessa come cosa.
2. Confidence score per ogni entità e ordine totale.
3. Arricchimento automatico delle informazioni mancanti.
4. Interfaccia con chatbot.
5. Supporto al caricamento di dataset aziendali per test^G o benchmarking.
6. Sincronizzazione periodica con ERP^G per catalogo prodotti e stati ordine.
7. Consultazione storico attività: ordini processati, errori, ambiguità risolte.

3.6.3 Requisiti funzionali opzionali

1. Supporto multilingua, integrazione con canali aggiuntivi (WhatsApp, Telegram).
2. Dashboard avanzata per l'admin.
3. Retraining automatico basato sul feedback degli utenti (stile ChatGPT).
4. Riconoscimento avanzato delle immagini come barcode, etichette, QR.
5. Notifiche automatiche in caso di errori.
6. OCR^G obbligatorio se la modalità immagine è attiva.
7. Modalità simulazione/sandbox senza inserimento in ERP.
8. Esportazione log/ordini in CSV/JSON^G per analisi esterne.
9. Cambio modelli AI (GPT4, GPT5, Gemini ecc).
10. Connettori plug-in per input da email/chat.
11. Supporto multi-tenant (dati e cataloghi separati per cliente). Comunque è centrale nel progetto^G il DB iniziale e non necessario separare i cataloghi.
12. Per quanto riguarda l'invio d'immagine l'azienda ha accettato l'invio di .png, .jpg, .jpeg, ma è stato chiesto di modificare il limite dei 5MB a 15MB e una risoluzione pari o superiore a 800x600 pixel.

3.7 Requisiti non funzionali

3.7.1 Requisiti non funzionali obbligatori

1. Il sistema deve essere scalabile.
2. Deve garantire affidabilità.
3. Deve garantire manutenibilità con strumenti come i log e mantenendo la separazione delle componenti nel codice.
4. È stato chiesto se ha senso mantenere l'audit trail, è stato suggerito di trasferirlo come opzionale oppure di cancellarlo.
5. Tempi di risposta rapidi di minore di 2 sec e 1 sec. Ci è stato suggerito di alzare la soglia a 8 secondi per l'accettabile e 4 secondi per l'ottimo.
6. Garantire la sicurezza dei dati trattati, chiaramente le password non devono essere trasmesse in chiaro. Se tutti i collegamenti fra API^G vengono gestite con HTTPS la sicurezza è garantita.
7. Garantire la disponibilità 24/24 7/7 del servizio. Il sistema deve quindi gestire correttamente errori e fallimenti della pipeline tramite meccanismi di recovery.
8. Hashing per password.

3.7.2 Requisiti non funzionali desiderabili

1. Latenza ridotta anche sotto carichi elevati.
2. Strumenti di monitoring dei componenti.
3. Supporto a meccanismi di caching per ridurre tempi di risposta.
4. Logging distribuito e centralizzato.
5. Supporto a sistemi di alerting automatici.
6. Deve supportare backup e ripristino dei dati e delle configurazioni critiche, già fatto da GitHub.

3.7.3 Requisiti non funzionali opzionali

1. Ottimizzazione avanzata per la riduzione dei costi computazionali.
2. Sistema di throttling intelligente per evitare sovraccarichi in input massivi. In questo caso si impedisce al chatbot di rispondere in immediato ma si aspetta.

3.8 Requisiti di vincolo

3.8.1 Requisiti di vincolo obbligatori

1. Utilizzo di un database^G relazionale.
2. La documentazione deve essere conforme.
3. Il sistema deve adottare un'architettura modulare.
4. Il sistema deve esporre interfacce di integrazione documentate tramite API^G REST.
5. Il progetto^G deve utilizzare un sistema di versionamento^G del codice.
6. I formati di scambio dati devono essere strutturati in un formato compatibile con il database^G gestionale.
7. Utilizzo di un database^G vettoriale.
8. È necessario costruire un'interfaccia web.
9. UI completamente responsive e mobile first.
10. Containerizzazione.

3.8.2 Requisiti di vincolo desiderabili

1. Adozione di API^G REST^G anche per comunicazione interne tra moduli.
2. Separazione netta tra frontend, backend^G e motore AI per maggiore manutenibilità.
3. Utilizzo di un formato standard per il logging (JSON^G logging).

3.8.3 Requisiti di vincolo opzionali

1. Utilizzo di un middleware per separare ulteriormente la comunicazione tra i componenti oppure attraverso connettori standard come una fonte dati ODBC.

4 Decisioni e Azioni

Codice	Descrizione
VE^G 2.1	Si utilizzerà il formato JSON^G per la comunicazione con l'API.
VE^G 2.2	Per definire correttamente l'API^G REST^G si utilizzerà il framework^G "Fast API" di Python.
VE^G 2.3	Il caricamento dei dati nel modello verrà effettuato tramite il framework^G "Langchain" di Python.
VE^G 2.4	Decisi i casi d'uso, requisiti funzionali, non funzionali e di vincolo con la loro classificazione.

5 Approvazione Esterna

Si attesta che il seguente verbale della riunione è stato approvato da parte dei rappresentati di Ergon Informatica Srl.
Tale approvazione è comprovata dalla presenza della firma di almeno uno dei rappresentanti:

11/12/2025

ERGON INFORMATICA S.r.l.
Via per Salvatronda, 21
31033 CASTELFRANCO V.TO
Partita I.V.A. 02229190265