



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

Gruppo: NullPointers Group
Email: nullpointersg@gmail.com

Verbale Riunione

19 Novembre 2025

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

Indice

1	Informazioni generali	3
2	Ordine del giorno	4
3	Diario della riunione	5
3.1	Presentazione Aziendale	5
3.2	Comunicazione con l'Azienda	5
3.3	Gestione del codice e Versioning	5
3.4	Organizzazione e Pianificazione del Progetto	6
3.5	Tecnologie, Formazione e Infrastruttura	6
3.6	Riscontro su Casi d'Uso e Requisiti	7
4	Decisioni e Azioni	9
5	Approvazione Esterna	10

1 Informazioni generali

- **Tipo di riunione:** Esterna
- **Luogo:** Sede di Ergon Informatica Srl (TV)
- **Data:** 19/11/2025
- **Ora inizio:** 15:00
- **Ora fine:** 17:00
- **Scriba:** Lisa Casagrande
- **Partecipanti:**
 - Carlesso Gianluca
 - Tieppo Anna
 - Santinon Matteo

2 Ordine del giorno

Primo colloquio in presenza di NullPointers Group con l’azienda Ergon Informatica Srl a seguito dell’aggiudicazione dell’Appalto: incontro conoscitivo, discussione relativa ad alcuni dubbi sorti durante la stesura dei documenti “Norme di Progetto” e “Analisi dei Requisiti”.

3 Diario della riunione

3.1 Presentazione Aziendale

L'incontro si è aperto con la presentazione della struttura di Ergon Informatica Srl, caratterizzata da una storia quarantennale e da due divisioni principali: una dedicata allo sviluppo^G del gestionale proprietario “Interspacing” e l'altra specializzata in infrastrutture IT, sicurezza e servizi cloud, operando principalmente su tecnologie Microsoft.

Successivamente, l'incontro ha introdotto il tema del progetto^G di intelligenza artificiale^G, citando sistemi di raccomandazione e modelli previsionali. È stato posto l'accento sulla sfida centrale legata agli assistenti virtuali: mitigare il fenomeno delle “allucinazioni” dei modelli linguistici, che generano risposte errate ma plausibili.

Abbiamo quindi esposto i nostri quesiti, ai quali abbiamo ricevuto risposte esaurienti. Di seguito il dettaglio.

3.2 Comunicazione con l'Azienda

1. Frequenza dei Meeting con l'azienda.

Abbiamo concordato di effettuare Meeting all'occorrenza, possibilmente con un preavviso di qualche giorno.

Qualora l'incontro si svolga telematicamente, verrà utilizzata la piattaforma **Google Meet**.

2. Canale di Comunicazione

Abbiamo concordato l'utilizzo di **WhatsApp** come canale di comunicazione.

3.3 Gestione del codice e Versioning

3. Flusso di lavoro **Git** consigliato per il progetto^G e se il metodo **GitFlow**, attualmente adottato, risulta valido oppure se ne conoscono o utilizzano uno più valido.

I referenti ci hanno confermato che il flusso Git^G adottato dal loro team è sostanzialmente allineato a una variante del GitFlow^G. Anche per loro esiste un branch^G main, che contiene sempre il codice in produzione^G, e un branch^G develop, in cui viene integrato il codice pronto per il testing. Ogni nuova funzionalità^G viene sviluppata in un branch^G dedicato (ad esempio feature/nome-funzionalità) e il merge^G avviene tramite pull request^G, che attiva il processo^G di revisione e verifica^G.

A differenza del nostro repository^G, che elimina automaticamente i branch^G feature dopo il merge^G, il loro non prevede una pulizia automatica; tuttavia non erano contrari a questa pratica e l'hanno considerata una gestione valida e utile per mantenere la repository^G ordinata. In conclusione, i referenti hanno confermato che la strategia da noi descritta è corretta e pienamente compatibile con il loro flusso di lavoro.

4. Come si devono gestire i **conflitti di merge** in un team di più persone e quale sia la procedura corretta quando, mentre si lavora su una funzionalità^G, qualcun altro modifica^G la stessa parte di codice.

È stata fatta luce sul fatto che la miglior prevenzione dei conflitti di merge^G è una buona organizzazione del lavoro: suddividere le funzionalità^G in modo che ciascun membro del team operi su parti di codice il più possibile indipendenti riduce notevolmente il rischio di sovrapposizioni.

La procedura operativa standard per la gestione dei conflitti è chiara: prima di iniziare lo sviluppo^G e, soprattutto, prima di effettuare un push, è fondamentale eseguire un git^G pull per allineare la copia locale con le modifiche presenti nel branch^G remoto. Qualora Git^G segnali un conflitto, è necessario risolverlo manualmente, decidendo quale porzione di codice mantenere o come combinarle correttamente. Una volta risolti tutti i conflitti, si procede con un git^G add dei file aggiornati e si completa il merge^G tramite git^G commit.

È stato inoltre sottolineato che i conflitti non rappresentano un errore, ma una componente normale e fisiologica dello sviluppo^G collaborativo. Ciò che conta è affrontarli correttamente, seguendo con attenzione il processo^G di pull e merge^G prima di proseguire con il proprio lavoro.

3.4 Organizzazione e Pianificazione del Progetto

5. In che modo è possibile organizzare al meglio scadenze e obiettivi e se viene adottato un metodo specifico per la gestione delle **milestone**.

I referenti hanno confermato che la pianificazione^G parte da una stima dell'impegno e dalla suddivisione del progetto^G in attività più piccole e gestibili. Sebbene in azienda queste attività siano guidate da Project Manager o Team Leader, noi dovremo autogestirci, creando una task^G list, assegnando le attività ai membri del team e definendo scadenze interne.

In pratica, abbiamo **libertà di organizzare** il lavoro come riteniamo più efficace.

6. Licenza del Repository

Ci è stato consigliato di utilizzare una Licenza MIT^G, in qualsiasi caso se non viene indicata la Licenza di fatto si applica quella più restrittiva, dunque a prescindere nessuno potrebbe utilizzare il materiale.

3.5 Tecnologie, Formazione e Infrastruttura

7. Quali tra le **tecnologie** elencate nel capitolato^G ci **consigliano** personalmente e, in particolare, quale modello di linguaggio è più indicato.

- **Front-end** (Interfaccia Utente): **React**, scelto per la sua ampia diffusione e le ottime potenzialità nella creazione di interfacce utente dinamiche;
- **Back-end** (Logica e AI): **Python**, ambiente ideale per lo sviluppo^G di modelli di intelligenza artificiale^G e machine learning^G;

- **Database:** consigliato **PostgreSQL**.
- **Modello di Linguaggio (LLM):** **GPT** di **OpenAI**, attualmente il modello più performante e affidabile per questo tipo di applicazioni. Le alternative open-source, come **LlaMA**, possono essere utili in scenari specifici (ad esempio per eseguire il modello in locale o offline per motivi di privacy o costi), ma le loro performance sono generalmente inferiori.
- **Analisi di Immagini:** per il riconoscimento di oggetti nelle foto, si possono utilizzare servizi come **GPT-4V** (con capacità visive) o strumenti specializzati come **Google Vision**. È importante ricordare che questi modelli non sono sempre deterministici: fornendo la stessa immagine più volte, le risposte potrebbero variare leggermente.
- **Trascrizione Vocale:** sempre **OpenAI** o **Whisper**.
- **OCR:** consigliato **EasyOCR** per le funzionalità^G di riconoscimento ottico dei caratteri (OCR).
- **Creazione di UML:** ci è stato consigliato l'utilizzo di **Draw.io**, in qualsiasi caso resta a nostra discrezione.

8. Organizzazione di **sessioni di formazione** sulle tecnologie non conosciute, come Docker^G o React^G.

Per supportarci nell'acquisire familiarità con tecnologie fondamentali possiamo organizzare dei **mini-corsi** o delle **sessioni introduttive** su tecnologie specifiche come Docker^G. Inoltre, possono indicarci risorse e guide mirate, come appunto l'utilizzo di Docker^G che ad oggi è una pratica comune, e online esistono molti materiali ben fatti che possiamo sfruttare per consolidare le nostre competenze.

9. Utilizzo di un database^G di test^G per le verifiche

Ci è stato ragionevolmente suggerito di non lavorare mai su un database^G di produzione^G, ma piuttosto utilizzare **database^G di test**, che sono una copia più o meno completa del database^G reale, contenente dati coerenti ma non sensibili. Avrà la stessa struttura e conterrà i cataloghi prodotti, gli storici ordini dei clienti (fittizi) e tutte le entità necessarie per testare il sistema.

3.6 Riscontro su Casi d'Uso e Requisiti

Per concludere, abbiamo richiesto un riscontro sugli attori principali e secondari, procedendo poi alla verifica^G e alla valutazione dei vrCasi d'Uso, dei vrRequisiti Funzionali, dei vrRequisiti Non Funzionali e dei vrRequisiti di Vincolo precedentemente abbozzati dal gruppo in un documento non ufficiale.

L'obiettivo era quello di poter successivamente redigere in modo più ordinato e completo l'Analisi dei Requisiti^G.

Nel corso di questa revisione sono emerse alcune osservazioni rilevanti, particolarmente utili per la stesura della documentazione finale; tra le più significative si possono evidenziare le seguenti:

10. Chiarimento sul flusso di validazione^G. L'utente convalida ogni singolo prodotto o l'intero ordine alla fine?

Il sistema propone, l'utente dispone. La validazione^G umana è un requisito di sicurezza non negoziabile.

Ci sono due approcci possibili:

- **Validazione^G a Singolo Prodotto:**

Il sistema analizza la richiesta e, per ogni prodotto identificato, propone un match (es. vrCoca Cola Lattina da 33cl). L'utente deve confermare o correggere ogni singola voce prima che venga aggiunta all'ordine.

- **Validazione^G dell'Ordine Completo:**

Il sistema interpreta l'intera richiesta e genera un vrcarrello o un'anteprima dell'ordine completo. L'utente visualizza l'elenco di tutti i prodotti, le quantità e le unità di misura proposte, e ha la possibilità di modificarlo prima di confermare l'invio definitivo.

L'idea di Ergon era più simile alla seconda opzione, ovvero visualizzare l'anteprima dell'ordine e confermarlo esplicitamente. Tuttavia, la scelta implementativa finale può essere discussa. L'importante è che l'utente abbia l'ultima parola sull'intero ordine prima che venga processato.

11. Come deve comportarsi il sistema con **prodotti ambigui**? Ad esempio, se un cliente scrive solo “due Coca”, come facciamo a sapere se intende lattine o bottiglie?

Il sistema deve essere intelligente e consultare lo storico ordini di quel specifico cliente. Se il cliente ha ordinato il 90% delle volte “Coca Cola in bottiglia”, il sistema può proporre quella come opzione primaria.

Nei casi di **ambiguità**, il sistema non deve indovinare. Deve invece **proporre** all'utente una scelta. Ad esempio, potrebbe visualizzare: “Quale prodotto intendevi?” ed elencare le opzioni più probabili (“Coca Cola Lattina”, vrCoca Cola Bottiglia).

L'utente, nella fase di validazione^G, selezionerà l'opzione corretta. Questo meccanismo di “disambiguazione” è una funzionalità^G chiave per rendere il sistema affidabile.

4 Decisioni e Azioni

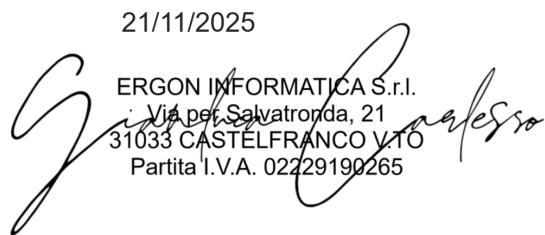
Codice	Descrizione
VE ^G 1.1	I meeting verranno svolti all'occorrenza, con preavviso. Gli incontri online avverranno tramite Google Meet ^G .
VE ^G 1.2	È stato scelto WhatsApp ^G come canale di comunicazione.
VE ^G 1.3	Abbiamo deciso di proseguire con Git ^G Flow e di mantenere il nostro metodo di suddivisione del lavoro in feature branch ^G , che verranno eliminati dopo il merge ^G .
VE ^G 1.4	Abbiamo deciso di adottare la licenza MIT ^G per il repository ^G .
VE ^G 1.5	Utilizzo di React ^G per il front-end.
VE ^G 1.6	Utilizzo di Python ^G per back-end/AI.
VE ^G 1.7	Utilizzo di PostgreSQL ^G per i database ^G .
VE ^G 1.8	Utilizzo di GPT ^G di OpenAI ^G per LLM ^G .
VE ^G 1.9	Per la realizzazione di SmartOrder il gruppo ha deciso di preferire l'implementazione della trascrizione vocale all'analisi di immagini
VE ^G 1.10	Utilizzo di EasyOCR ^G nel caso di implementazione della funzione di riconoscimento delle immagini.
VE ^G 1.11	Utilizzo di Draw.io ^G per la creazione di UML ^G .
VE ^G 1.12	Utilizzo di un database ^G di test ^G per le verifiche.
VE ^G 1.13	È necessaria la validazione ^G dell'ordine: l'utente deve confermare l'ordine completo prima dell'invio.
VE ^G 1.14	Per prodotti ambigui il sistema consulta lo storico del cliente, in mancanza di esso propone opzioni invece di "indovinare".

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:

21/11/2025


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