



On-By-One

Verbale esterno 2026/10/28

On-By-One:

Sara Ballardini	2101048
Shuhui Chen	2138001
Simone Giuliano	2145591
Alvise Greggio	2141004
Daniele Nuzzaci	2139039
Federico Temporin	2146764
Tommaso Zorzetto	2148629

Informazioni documento			
Versione	Data	Stato	Destinatari
1.0.0	2026/10/29	Approvato	On-By-One Prof. Tullio Vardanega Prof. Riccardo Cardin MIRIADE srl

Contatti: onbyone.swe@gmail.com

Registro delle modifiche

Versione	Data	Autore	Descrizione	Verificatore
1.0.0	2026/10/29	Shuhui Chen	Revisione del verbale	Alvise Greggio
0.1.0	2026/10/28	Shuhui Chen	Prima stesura del verbale	-

Indice

1	Informazioni generali	3
2	Introduzione	4
3	Discussioni	4
3.1	Origine del progetto e stato di partenza	4
3.2	Dati di test e documentazione medica	4
3.3	Formazione sul dominio e <i>design sprint</i>	4
3.4	Attori, ruoli e sistema dei permessi	4
3.5	Sicurezza, privacy e autenticazione	4
3.6	Accessibilità e uso dell'IA nello sviluppo	5
3.7	Architettura, linguaggi e supporto tecnico	5
3.8	Intelligenza artificiale e dati sensibili	5
3.9	Priorità del <i>Minimum Viable Product</i>	5
3.10	Account AWS e costi	5

1 Informazioni generali

Dettagli della riunione	
Data	2026/10/28
Ora inizio	15:00
Ora fine	16:00
Modalità	Google Meet

Partecipanti		
Nome e cognome	Ruolo	Presenza
Sara Ballardín	-	Presente
Shuhui Chen	-	Presente
Simone Giuliano	-	Presente
Alvise Greggio	Responsabile	Presente
Daniele Nuzzaci	-	Assente
Federico Temporin	-	Assente
Tommaso Zorzetto	-	Presente

2 Introduzione

La riunione si è svolta con il referente di MIRIADE srl Emanuele Righetto con l'obiettivo di chiarire il contesto del capitolato, i requisiti funzionali e non funzionali, le scelte tecnologiche e il supporto che l'azienda metterà a disposizione del gruppo durante il progetto.

3 Discussioni

3.1 Origine del progetto e stato di partenza

Non esiste alcun prototipo: il progetto parte da zero. Miriade collabora con San Filippo Fighters da due anni, tramite finanziamenti e progetti minori come pagine web e landing page. A inizio anno l'associazione aveva avanzato una richiesta più piccola e mirata, che l'azienda non era riuscita a soddisfare per mancanza di risorse; la proposta è stata poi ampliata e presentata come tema del corso.

3.2 Dati di test e documentazione medica

Non è previsto l'accesso a dati reali di pazienti. L'azienda fornirà, tramite l'associazione, documenti su trattamenti, tipi di cura e procedure, che costituiranno fonti affidabili per alimentare la parte di IA (RAG) e i *disease pack*, oltre a guidare il gruppo nel flusso di supporto al malato. I casi d'uso (pazienti fittizi) dovranno invece essere generati dal gruppo. In alternativa, si può chiedere ai volontari dell'associazione di provare l'applicazione inserendo un caso con informazioni a loro note, il più realistico possibile, tenendo conto che si tratta in prevalenza di volontari con disponibilità limitata.

3.3 Formazione sul dominio e *design sprint*

Per colmare la mancanza di competenze mediche l'azienda propone due attività facoltative. La prima consiste in uno o più incontri con l'associazione, in cui viene spiegato il flusso che il *care navigator* e il *caregiver* affrontano per assistere il paziente, sia per la parte tecnica sia per quella medica. La seconda è una versione ridotta di *design sprint*, già apprezzata negli anni scorsi, con uno o due workshop che coinvolgono il gruppo, Miriade e l'associazione per disegnare i flussi di lavoro di tutti gli attori. Al termine il gruppo dovrebbe avere un'idea più chiara della navigazione e delle azioni del prodotto, oltre a una bozza di interfaccia a livello di componenti e sezioni, senza elementi grafici definitivi.

3.4 Attori, ruoli e sistema dei permessi

Gli attori di base sono quattro o cinque: *caregiver*, *care navigator*, professionista sanitario, struttura ed, eventualmente, un ente pubblico come la scuola (preside o docente referente), profilo che non ci si aspetta nella fase iniziale. L'azienda ritiene auspicabile un sistema di permessi espandibile senza modificare il codice, in cui l'aggiunta di un nuovo ruolo avvenga da configurazione. L'approccio tipico prevede permessi molto granulari associati alle singole operazioni, con il ruolo inteso come insieme di permessi (sì/no). Una volta realizzato il sistema dinamico, la stessa interfaccia potrà essere usata per creare i ruoli richiesti durante il progetto, a patto che sia progettata con cura. Il controllo degli accessi è invece obbligatorio; per esempio un medico può vedere solo i referti dei propri pazienti.

3.5 Sicurezza, privacy e autenticazione

I dati sensibili devono essere trasmessi in modo protetto. L'allegato 1 del capitolato elenca una serie di standard, non tutti da implementare; come primo livello è sufficiente HTTPS al

posto di HTTP e non è richiesta la cifratura aggiuntiva del payload. La cifratura *end-to-end* sulla parte LLM non è facilmente realizzabile e potrebbe risultare assente o parziale; l'azienda supporterà il gruppo sui dubbi tecnici. Per l'autenticazione, che il gruppo non aveva ancora affrontato, sono state indicate due strade: un database locale con tabelle di utenti e ruoli e password cifrate oppure AWS Cognito, che gestisce un pool di utenti, il reset della password e la federazione (single sign-on tramite Google, OAuth2 e OpenID Connect). Nelle architetture serverless dell'azienda una funzione Lambda posta davanti alle altre verifica l'utente tramite Cognito e, in base ai ruoli, autorizza o nega le operazioni. L'uso di Cognito è consigliato perché semplifica il lavoro.

3.6 Accessibilità e uso dell'IA nello sviluppo

L'interfaccia deve essere accessibile per gli attori; l'azienda può fornire librerie e template grafici (*design system*). Progettare un'interfaccia accessibile non è banale e le linee guida sono molto estese: si consiglia di farsi aiutare da strumenti di IA esplicitando i requisiti di accessibilità, e di verificare il risultato con un tool di controllo, che segnala problemi di colore, contrasto, font e dimensioni. L'uso dell'IA non è un problema, a condizione che resti un supporto: le decisioni progettuali e tecniche devono essere prese dal gruppo, che deve mantenere il controllo del codice e saper motivare le proprie scelte.

3.7 Architettura, linguaggi e supporto tecnico

Si richiede un'architettura a microservizi: non un unico artefatto monolitico con tutte le API, ma tanti servizi autonomi e piccoli, ciascuno con una singola responsabilità, che insieme formino il *backend*. Il gruppo può scegliere tra container (Docker, eseguiti ad esempio su AWS con ECS), facili da testare in locale, e funzioni serverless (AWS Lambda, Google Cloud Functions), ognuna con pro e contro. Con Lambda è più naturale mantenere servizi molto piccoli, mentre i container, più costosi, tendono ad aggregare più funzioni. Va evitata la miscela incoerente tra microservizi e serverless. Per il *backend* l'azienda consiglia Java e Python (anche Node/TypeScript, con minor supporto). Per il *frontend* suggerisce Angular per il web e Flutter per il mobile, oppure Flutter per entrambi, con maggiore lavoro di adattamento ma un'unica base di codice. Miriade può offrire supporto su Java, Python, Angular, Flutter e AWS, in parte anche su React e Node; è possibile usare altre tecnologie (ad esempio Azure), ma con un supporto più limitato.

3.8 Intelligenza artificiale e dati sensibili

Sono stati analizzati i concetti di RAG (risposte fondate su documentazione autorevole e controllata) e di *guardrail* (meccanismi di protezione contro allucinazioni ed esfiltrazione di dati sensibili). Data la limitata esperienza del gruppo sulle architetture ad agenti, l'azienda eventualmente fornirà supporto nella classificazione delle informazioni riservate e nella definizione di un sistema sicuro per il controllo degli accessi e delle risposte del modello.

3.9 Priorità del *Minimum Viable Product*

Le priorità andranno definite insieme nel workshop iniziale; secondo l'azienda le funzioni per *caregiver* e *care navigator* sono le più importanti, mentre funzioni importanti quali quelle per il medico sono negoziabili. Poiché il capitolato è ampio, il gruppo può specializzarsi senza completare ogni parte.

3.10 Account AWS e costi

L'azienda fornirà un account AWS gestito da Miriade, con utenze nominali e permessi iniziali di base limitati ai servizi standard (ad esempio Lambda ed ECS) e non ai servizi più costosi (ad

esempio Bedrock), per evitare costi elevati. I permessi aggiuntivi andranno richiesti in modo puntuale all'azienda, mettendo in conto alcuni tempi di attesa. I costi dell'account sono sostenuti dall'azienda, che chiede però attenzione al loro contenimento.

Firma del revisore esterno

A handwritten signature in black ink, appearing to read 'R. F. E.', is written over a solid horizontal line.