smartlearn
virtual learning. real success.

Challenge-03-Agente IA su MedFinder

ICT-Formazione professionale Svizzera

Indice

Situazione iniziale **3**

Esercizio **4**

Sezione a) Requisiti di integrazione del middleware 5

Sezione b) Fattori determinanti concreti: indicatori, proprietà del sistema e vincoli 7

Sezione c) Classificazione dei dati e raccomandazione per il database 8

Ambiente virtuale **9**

Installazione, Setup e configurazione 9

Ubuntu Server 9

Upload dell'applicazione sulla VM 10

Windows 11 Client 13

Situazione iniziale

Situazione iniziale

Un’azienda di medie dimensioni gestisce una piattaforma attiva in tutta la Svizzera per mettere in contatto pazienti con medici e cliniche. L’elenco comprende 28’000 voci in quattro lingue nazionali. Nonostante la base dati sia completa, molti utenti abbandonano l’applicazione mobile senza ottenere risultati: la ricerca testuale è difficile da usare, non tollera errori di battitura e presuppone che nomi e termini specialistici siano inseriti correttamente.

L’azienda ha deciso di affrontare questo problema con un agente IA a comando vocale. In futuro, gli utenti dovranno poter formulare la propria richiesta semplicemente a voce, così come la pensano: *«Ho mal di schiena: consigliami per favore un medico a Zurigo»*.

L’agente comprende la richiesta dell’utente, la confronta con l’elenco dei medici e fornisce una risposta diretta e filtrata. A tale scopo, viene introdotto un nuovo livello intermedio (Middleware) tra il sistema legacy e l’agente IA.

Condizioni tecniche di base:

Interfaccia dati**:** l’unico accesso disponibile al backend esistente (`GET /v1/providers/dump`) restituisce sempre l’intero set di dati sotto forma di un unico file JSON da 383 MB. Nell’attuale ambiente operativo, il tempo di risposta osservato è generalmente compreso tra 40 e 90 secondi, a seconda del carico di rete e del carico del server. Non sono disponibili parametri di filtro o paginazione. Un adeguamento dell’endpoint prima del go-live è escluso.

Riconoscimento vocale**:** analisi empiriche basate su 200 registrazioni vocali in svizzero tedesco mostrano un tasso di errore del 23% per i nomi di professionisti sanitari e del 34% per i nomi di località pronunciati in dialetto. Nel set di dati mancano completamente campi di indicizzazione fonetica. Il middleware deve tuttavia essere in grado di risolvere correttamente anche trascrizioni errate.

Architettura**:** l'agente IA comunica esclusivamente tramite API REST con risposte JSON standardizzate. Per ragioni architetturali, non è previsto un accesso diretto a database o sistemi interni. Il middleware assume il ruolo di unica interfaccia tra il backend legacy e l'agente.

Ambiente e credenziali di accesso

Funzione	Host	Indirizzo IP	OS	Applicazioni & Strumenti	Utente	Passw
Legacy-API-Server	ubu-srv-exam	192.168.110.20/24	Ubuntu 24.04.2 LTS	Provider Directory API Swagger UI: http://192.168.110.20/docs (http://192.168.110.20/docs) Backend (Port 80): http://192.168.110.20/v1/providers/dump (http://192.168.110.20/v1/providers/dump)	Nessun accesso	
Testclient	win-cli-exam	192.168.110.101/24	Windows 11 Client	Browser Edge curl 8.19.0 Postman 12.11.3	vmadmin	sml1

Nel playground API fornito è disponibile l'endpoint legacy GET `/v1/providers/dump`. Le credenziali di accesso del client di test sono riportate nella tabella sottostante. Utilizzate l'endpoint tramite il client di test e osservate attentamente i dati restituiti.

Ambiente e credenziali di accesso

Funzione	Host	Indirizzo IP	OS	Applicazioni & Strumenti	Utente	Passw
Legacy-API-Server	`ubu-srv-exam`	`192.168.110.20/24`	Ubuntu 24.04.2 LTS	Provider Directory API Swagger UI: http://192.168.110.20/docs (http://192.168.110.20/docs) Backend (Port 80): http://192.168.110.20/v1/providers/dump (http://192.168.110.20/v1/providers/dump)	Nessun accesso	
Testclient	`win-cli-exam`	`192.168.110.101/24`	Windows 11 Client	Browser Edge curl 8.19.0 Postman 12.11.3	`vmadmin`	`sml1`

Sezione a) Requisiti di integrazione del middleware

Analizza la situazione iniziale e i dati di esempio recuperati tramite l'endpoint. Da questa analisi ricava quattro requisiti centrali di integrazione per il nuovo middleware, uno per ciascuna categoria:

- latenza/performance
- qualità dei dati/ricerca
- compatibilità tecnica
- resilienza alle trascrizioni errate

Formula ogni requisito come un requisito obbligatorio preciso e tecnicamente motivato, secondo il seguente schema:

- Dimensione: [nome della categoria]
- Requisito: il middleware deve…
- Motivazione: perché… [riferimento alla situazione iniziale o ai dati API]

Distribuzione dei punti: 4 × 1 punto = 4 punti

Un requisito obbligatorio completo per ciascuna dimensione, composto da requisito e motivazione. Viene assegnato un punto per ciascuna dimensione solo se entrambe le parti sono presenti e coerenti. Non vengono assegnati punti parziali.

Requisito relativo a latenza/performance

Requisito relativo alla qualità dei dati / ricerca

Requisito relativo alla compatibilità tecnica

Requisito relativo alla resilienza alle trascrizioni errate

Sezione b) Fattori determinanti concreti: indicatori, proprietà del sistema e vincoli

Su cosa si basano i tuoi requisiti?

Per ciascuno dei quattro requisiti indica il motivo concreto per cui è necessario. Fai riferimento a un dato, a una caratteristica del sistema o a un vincolo tecnico presente nella situazione iniziale o nei dati di esempio.

Usa il seguente schema di risposta:

- Dimensione: [categoria]
- Motivo concreto: [numero, nome di campo o caratteristica specifica ricavata dalla situazione iniziale o dai dati di esempio]
- Spiegazione: questo valore o questa caratteristica rende il requisito necessario perché…

Distribuzione dei punti: 4 × 1 punto = 4 punti

Viene assegnato 1 punto per ciascuna dimensione. **Importante:** la semplice ripetizione del requisito indicato nella sezione a) vale 0 punti. È necessario indicare un dato concreto ricavato dalla situazione iniziale o dai dati di esempio dell'API.

Fattore scatenante per latenza / performance

Fattore scatenante per la qualità dei dati / ricerca

Fattore scatenante per la compatibilità tecnica

Fattore scatenante per la resilienza alle trascrizioni errate

Sezione c) Classificazione dei dati e raccomandazione per il database

Dopo aver interrogato l'endpoint nel playground API, identifica il tipo di dati presenti nell'estratto restituito e la relativa classificazione. Spiega poi in 1-2 frasi quale tipo di database è adatto per il middleware e perché.

Distribuzione dei punti:
1 punto per il tipo di dati e la classificazione, 1 punto per la motivazione della scelta del database.

Tipo di dati

Tipo di database

Ambiente virtuale

Vengono utilizzate VM del set `Master-Set PDS 2026`, perché contengono già configurazioni e script utili (purtroppo attualmente descritti altrove).

L'ambiente virtuale comprende le seguenti VM:

- `192.168.110.2` – Gateway – `bsd-inf-exam` – invisibile ai candidati (-)
- `192.168.110.20` – Ubuntu Server – `ubu-srv-exam` – visibile ai candidati, ma non controllabile (R)
- `192.168.110.101` – Windows 11 Client – `win-cli-exam` – visibile e controllabile dai candidati (RW)

Installazione, Setup e configurazione

Ubuntu Server

- Correzione del bug relativo al blocco su `eth1` durante l'avvio.
- Aggiornamenti tramite script standard.
- Configurazione delle impostazioni locali (lingue e layout di tastiera) tramite script standard.
- Installazione di Docker:
    - `curl -fsSL https://get.docker.com | sudo sh && sudo usermod -aG docker $USER && newgrp docker`
    - Test: `sudo systemctl status docker` `docker compose version`
- Adeguamenti al file `docker-compose.yml`:
    - Binding sulla NIC, non su `127.0.0.1` porta 80: "`80:8080`"
- Permessi sulla directory `data`: `sudo chmod 777 data` (Nota: il container viene eseguito con l'utente non root `vmadmin`)
- Build e avvio dell'ambiente: `docker compose down && sudo rm -rf data/* && docker compose up --build`

Upload dell'applicazione sulla VM

- Per effettuare il build del container Docker è necessario disporre di una connessione Internet.
- Attiva l'interfaccia `wan0` sul gateway `bsd-inf-exam` e avvia la VM del gateway.



- Avvia la VM dell'Ubuntu Server.
- Carica sulla VM, tramite trasferimento file, l'applicazione compressa in formato ZIP insieme al file `docker-compose.yaml`.



- Connessione al server tramite console remota
- Login come `vmadmin` con la password standard
- Passaggio a `sudo -i`
- Estrazione dell'applicazione nella directory `/home/vmadmin/app`
- Adeguamenti in `docker-compose.yml` relativi al binding: NIC, (non `127.0.0.1`), e porta 80: "`80:8080`"
- Build dell'applicazione tramite: `docker compose build`
- Avvio del container Docker tramite: `docker compose up -d` (In alternativa, riavvia il server: il servizio Docker avvia automaticamente l'applicazione)
- Test dal client, sia con connessione Internet sia senza.
- Disattivazione della connessione Internet sul gateway (la VM deve essere arrestata)

Windows 11 Client

- Aggiornamenti con script standard.
- Configurazione delle impostazioni locali (lingue e layout di tastiera) con script standard.
- Bookmark e pagina iniziale in Edge:
    - `http://192.168.110.20/docs`
    - `http://192.168.110.20/v1/providers/dump`
- Download e installazione di Postman (come amministratore). Fonte: `https://www.postman.com/downloads/`