

Progetto: Registro Elettronico "Didattix"

Committenza: Ambiente dimostrativo per registro elettronico scolastico

Team di sviluppo: Alterra Studios

Periodo di esecuzione: Giugno 2025 – Dicembre 2025

Project Manager: Sara Tantucci

## 1. Scopo del documento e perimetro tecnico

Il presente diario di bordo tecnico documenta in modo puntuale e cronologico le attività svolte per la realizzazione dell'ambiente "Registro Elettronico Didattix", descrivendo architettura, configurazioni, scelte progettuali, criticità e relative soluzioni. Il documento è destinato a figure tecniche (system engineer, DevOps, responsabili IT) e a stakeholder che necessitano di valutare l'aderenza del sistema a buone pratiche di sicurezza, osservabilità, automazione e manutenibilità.

Il perimetro include: provisioning e hardening di una VM Ubuntu Server 22.04.5 LTS su VirtualBox, realizzazione di uno stack Docker multi-servizio (web, database, monitoring, logging, LLM, API REST), configurazione di meccanismi di backup e monitoring periodico basati su script e cron, e produzione della documentazione infrastrutturale e operativa.

## 2. Panoramica dell'architettura

### 2.1 Componenti principali

L'architettura è costruita su una singola macchina virtuale denominata registro che funge da nodo unico di esecuzione per tutti i servizi necessari alla dimostrazione del registro elettronico. Su questa VM è installato Docker Engine con Docker Compose, utilizzato per orchestrare i container relativi a:

- applicazione web del registro (PHP 8.3 + Apache);
- database MySQL 8 con interfaccia di amministrazione phpMyAdmin;
- sistema di monitoring basato su Prometheus e Node Exporter;
- logging centralizzato tramite Loki e Promtail;
- dashboard di osservabilità tramite Grafana;
- stack LLM basato su Ollama e Open WebUI;
- API REST di monitoring sviluppata in Node.js e gestita tramite PM2.

La scelta di concentrare l'ambiente su un singolo host virtuale risponde all'obiettivo di semplificare deployment, manutenzione e dimostrazioni, mantenendo al contempo un elevato livello di realismo architetturale e aderenza alle best practice di containerizzazione e monitoring.

### 2.2 Vincoli progettuali e requisiti non funzionali

I vincoli prevedono l'utilizzo di Ubuntu Server LTS come sistema operativo, Docker come tecnologia di containerizzazione e l'adozione di strumenti open source per il monitoring e il logging (Prometheus, Grafana, Loki, Promtail). Sul piano non funzionale, il sistema deve essere:

- riproducibile (ambiente ricostruibile da documentazione e file di configurazione);
- osservabile (metriche e log centralizzati);
- sicuro (accesso SSH a chiave, firewall, utenze dedicate);
- automatizzato (backup schedulati, monitoraggio e reportistica via cron).

## 3. Ruoli, responsabilità e modello organizzativo

Per garantire un corretto governo del progetto è stata adottata una chiara separazione tra area applicativa e area infrastrutturale, in coerenza con i modelli consigliati nella reportistica di progetto professionale.

- Area applicativa (frontend/web)
  - Responsabili: Filippo Hinceanu, Daniele Strino.
  - Deliverable: configurazione e deploy dell'applicazione "Registro elettronico Didattix" nei container PHP/Apache, integrazione delle pagine e dei flussi di utilizzo (login, gestione classi, registrazione e consultazione lezioni), verifica dell'esperienza utente da browser.
- Area infrastrutturale (VM, networking, sicurezza, Docker, monitoring, LLM, automazioni)
  - Responsabili: Sara Tantucci, Nicolò Fiorucci.
  - Deliverable: realizzazione della VM registro, configurazione di rete su VirtualBox, hardening SSH e firewall, installazione di Docker e Compose, definizione e manutenzione dei file docker-compose.yml, setup dei sistemi di monitoring/logging e delle dashboard Grafana, integrazione/contenimento dell'LLM, implementazione dell'API di monitoring e degli script di backup/monitoraggio.

## 4. Giugno 2025 – Analisi dei requisiti e definizione architetturale

### 4.1 Raccolta requisiti funzionali e non funzionali

In questa fase è stata effettuata una raccolta strutturata dei requisiti funzionali dell'applicazione di registro elettronico: gestione delle classi, anagrafiche, registro lezioni, interfaccia di consultazione orientata a docenti e studenti, accesso via browser con autenticazione. Parallelamente, sono stati identificati i requisiti non funzionali: prestazioni accettabili su VM singola, semplicità di manutenzione, logging centralizzato, monitoraggio sistematico delle risorse e procedure di backup automatico per prevenire la perdita di dati.

Sono stati inoltre classificati i moduli del project work in:

- moduli obbligatori: setup VM, Docker, configurazioni di sicurezza, monitoring, automazione (cron, script);
- moduli "bonus": API REST di monitoring, sistema di backup automatico, composizione LLM in Docker, dashboard avanzate per le metriche.

#### 4.2 Scelta architetturale e dimensionamento

È stata adottata una topologia a "singolo nodo" in cui la VM registro funge sia da host virtuale sia da nodo di esecuzione per tutti i container. Questa scelta riduce la complessità di gestione, semplifica la formazione e la presentazione del progetto, pur mantenendo una separazione logica dei servizi tramite Docker e reti virtuali interne.

Il dimensionamento definito è:

- 16 GB RAM per supportare simultaneamente i servizi core (web, DB), lo stack di monitoring e il runtime LLM;
- 8 vCPU virtuali per garantire reattività anche in presenza di carichi concorrenti (query DB, scraping metriche, richieste al modello LLM);
- disco VDI dinamico  $\geq 40-60$  GB per ospitare OS, immagini Docker, volumi persistenti (MySQL, Loki, Grafana) e log generati dai processi periodici.

L'architettura è stata sintetizzata in un diagramma logico (layer VM / layer Docker / layer servizi) e discussa con il team applicativo per definire URL, porte esposte, variabili d'ambiente e punti di integrazione (es. endpoint API di monitoring, accesso al DB).

### 5. Luglio 2025 – Provisioning della VM e setup sistema operativo

#### 5.1 Configurazione VirtualBox e networking

È stata creata la VM registro in Oracle VirtualBox con le seguenti impostazioni:

- Tipo: Linux
- Versione: Ubuntu (64-bit)
- RAM: 16 GB
- vCPU: 8
- Disco: VDI dinamico, dimensione minima 40–60 GB
- Controller grafico: VMSVGA, 80 MB video, accelerazione 3D abilitata.

A livello di rete, per garantire il pieno accesso dalla LAN, è stata selezionata la modalità Scheda con bridge sulla NIC fisica dell'host, con scheda emulata Intel PRO/1000 MT Desktop; la modalità promiscua è stata configurata su "Permetti tutto" e il "cavo connesso" attivato. Questa configurazione consente di assegnare alla VM un indirizzo IP nell'intervallo della LAN, rendendola raggiungibile come se fosse un host fisico.

#### 5.2 Installazione Ubuntu Server 22.04.5 LTS

L'installazione del sistema operativo è stata eseguita tramite ISO di Ubuntu Server 22.04.5 LTS con partizionamento guidato sull'intero disco.

Durante il setup sono stati effettuati:

- impostazione hostname registro;
- creazione dell'utente amministrativo iniziale;
- installazione del server OpenSSH come componente base per la gestione remota.

A valle dell'installazione sono stati effettuati controlli tramite `hostnamectl` (verifica hostname e virtualizzazione), `uname -a` (verifica kernel), `lsb_release -a` (versione distribuzione) e `ip a` per verificare interfacce e indirizzo IP. È stato quindi fissato l'IP della VM in 10.10.13.244, da utilizzare come riferimento statico per tutti i successivi file di configurazione.

#### 5.3 Issue / Fix

- Issue: la VM inizialmente era configurata in modalità NAT, impedendo l'accesso diretto dalla LAN e limitando le possibilità di test multi-client.  
Fix: passaggio alla modalità di rete "Scheda con bridge" e rifissazione dell'indirizzo IP, con successivi test di ping e accesso SSH da più host della rete.

### 6. Agosto 2025 – Hardening identità, SSH e firewall

#### 6.1 Gestione delle identità e gruppi di sistema

È stato definito l'utente operativo principale registro, destinatario delle attività di manutenzione applicativa e infrastrutturale. L'account è stato aggiunto ai gruppi:

- `adm` e `sudo` per i privilegi amministrativi;
- `docker` per l'uso dei comandi Docker senza `sudo`;
- `lxd` e `ollama` per eventuali estensioni legate a container e LLM;
- `plugdev` per la gestione di eventuali periferiche collegate.

Questa scelta evita l'utilizzo dell'utente root in produzione, in linea con le raccomandazioni di sicurezza per host Linux.

## 6.2 Configurazione accesso SSH a chiave

La procedura di hardening SSH ha previsto:

- generazione delle chiavi SSH sul/i client autorizzato/i;
- copia della chiave pubblica nel file `~/.ssh/authorized_keys` dell'utente registro;
- verifica e correzione dei permessi sulla directory `.ssh` (700) e sui file (600).

Nel file di configurazione di SSH (`/etc/ssh/sshd_config`) è stata disabilitata l'autenticazione via password e vietato l'accesso diretto dell'utente root, lasciando come unico metodo di accesso remoto le chiavi SSH per l'utente dedicato. La configurazione è stata validata con `sshd -t` e applicata tramite riavvio del servizio (`systemctl restart sshd`).

## 6.3 Firewall UFW

Per la gestione del firewall è stato adottato UFW (Uncomplicated Firewall), abilitato con profilo predefinito `deny-incoming/allow-outgoing`. Le regole applicate includono:

- `ufw allow OpenSSH` per consentire l'accesso SSH;
- `ufw deny 23` per bloccare esplicitamente Telnet;
- `ufw deny 3389` per bloccare RDP non necessario in contesto Linux server.

Lo stato del firewall e l'ordine delle regole sono stati verificati tramite `ufw status numbered`, garantendo coerenza con le esigenze di esposizione dei servizi.

## 6.4 Issue / Fix

- Issue: dopo l'attivazione di UFW, l'accesso SSH risultava bloccato per mancanza della regola di allow.  
Fix: accesso via console VirtualBox, inserimento della regola `ufw allow OpenSSH`, nuova verifica tramite connessione SSH da host esterno.
- Issue: errori di autenticazione legati a permessi errati sulla directory `.ssh`.  
Fix: sistemazione dei permessi in conformità alla documentazione OpenSSH, test ripetuti di accesso da client multipli.

# 7. Settembre 2025 – Introduzione di Docker e stack applicativo web

## 7.1 Installazione Docker Engine e Compose

L'host registro è stato aggiornato e sono stati installati Docker Engine e Docker Compose dai repository ufficiali, seguendo le linee guida per l'utilizzo in ambienti server. Il servizio `docker.service` è stato abilitato per l'avvio automatico all'accensione della VM e testato con:

- `docker run --rm hello-world` per verificare la correttezza dell'installazione;
- `docker info` per analizzare configurazione e parametri del daemon.

L'utente registro è stato aggiunto al gruppo `docker` per semplificare l'utilizzo dei comandi senza `sudo`, secondo le best practice operative.

## 7.2 Definizione del file `docker-compose.yml` (web + DB)

È stato definito un file `docker-compose.yml` dedicato alla parte applicativa, comprendente:

- Servizio db (MySQL 8)
  - immagine MySQL 8 ufficiale;
  - volume montato per `/var/lib/mysql` per garantire persistenza;
  - variabili d'ambiente per utente, password e database del registro.
- Servizio web (PHP 8.3 + Apache)
  - immagine PHP 8.3 con Apache;
  - mount del codice dell'applicazione "Registro elettronico Didattix";
  - porta esterna 8080 mappata sulla porta HTTP del container.
- Servizio phpmyadmin
  - immagine phpMyAdmin;
  - variabile `PMA_HOST=db` per connettività al DB;
  - porta esterna 8081 per interfaccia di amministrazione.
- Rete Docker interna dedicata per isolare il traffico applicativo rispetto ad altri stack.

## 7.3 Test funzionali applicazione

Una volta avviati i container con `docker compose up -d`, sono stati eseguiti test applicativi:

- accesso all'interfaccia web via <http://10.10.13.244:8080>;
- login con credenziali di test;
- creazione e gestione delle classi;
- registrazione delle lezioni e consultazione delle informazioni archiviate.

L'accesso a phpMyAdmin tramite <http://10.10.13.244:8081> ha consentito di verificare la corretta creazione degli schemi e la coerenza dei dati persistenziati.

## 7.4 Issue / Fix

- Issue: conflitti di porta causati da processi locali o configurazioni preesistenti.  
Fix: standardizzazione delle porte 8080 (web) e 8081 (phpMyAdmin), aggiornamento delle configurazioni e della documentazione interna.
- Issue: phpMyAdmin non riusciva a connettersi al database a causa di host errato nelle variabili d'ambiente.  
Fix: impostazione PMA\_HOST=db nel file docker-compose.yml e riavvio dello stack per ristabilire la connettività.

## 8. Ottobre 2025 – Monitoring e logging centralizzato

### 8.1 Obiettivi di osservabilità

Questa fase ha introdotto un sistema di monitoring e logging centralizzato, in linea con le best practice per ambienti Docker: metriche di host e container, log aggregati e visualizzazione tramite dashboard. L'obiettivo è identificare colli di bottiglia, anomalie e tendenze, con particolare attenzione a CPU, memoria, I/O e stato dei servizi.

### 8.2 Stack Prometheus, Node Exporter, Loki, Promtail, Grafana

Lo stack di osservabilità include:

- Prometheus: raccolta e storage delle metriche da host e servizi;
- Node Exporter: esportazione delle metriche della VM (CPU, RAM, disco, rete);
- Loki: backend per lo storage dei log a bassa cardinalità;
- Promtail: agente di raccolta log, configurato per leggere da /var/log e dalla directory del progetto;
- Grafana: frontend per la visualizzazione di tempi serie e log tramite dashboard configurabili.

Sono stati creati i seguenti file di configurazione:

- prometheus.yml: definizione dei job di scraping (endpoint di Prometheus, Node Exporter, eventuali altri target);
- loki-config.yaml: configurazione di ingesters, queriers, retention e percorsi di storage (loki-data, loki-data/wal);
- promtail-config.yaml: definizione dei percorsi di log da seguire (file di sistema e log dell'applicazione).

Volumi persistenti:

- grafana-data (database SQLite grafana.db);
- grafana-provisioning per datasources e provisioning iniziale;
- loki-data e loki-data/wal per i log centralizzati.

### 8.3 Dashboard Grafana

In Grafana sono state create dashboard per:

- monitoraggio CPU, memoria, spazio disco della VM;
- stato e risorse dei container Docker (CPU per container, memoria, riavvii);
- analisi dei log tramite Loki, con filtri per servizio e per severità.

Queste dashboard consentono una visione unificata dell'intero ambiente, facilitando la diagnosi di problemi e l'analisi post-incident.

### 8.4 Issue / Fix

- Issue: warning DNS nei container, con rallentamenti nella risoluzione nomi e nei pull delle immagini.  
Fix: ottimizzazione dei resolver a livello di host (es. definizione di DNS affidabili in /etc/resolv.conf) e configurazione coerente del bridge Docker.
- Issue: iniziale raccolta log limitata ai soli log di sistema.  
Fix: ampliamento di promtail-config.yaml per includere la directory del progetto, permettendo la raccolta dei log applicativi e degli script custom.

## 9. Novembre 2025 – Stack LLM, API REST di monitoring e backup

### 9.1 Stack LLM (Ollama + Open WebUI)

Per abilitare scenari di supporto tramite intelligenza artificiale locale, è stato configurato uno stack LLM dedicato:

- container llm basato su immagine ollama/ollama:latest, in ascolto sulla porta 11434;
- container webui basato su immagine ghcr.io/open-webui/open-webui:main, in ascolto sulla porta 3500;
- rete Docker dedicata llm-docker\_default per isolare il traffico LLM dal resto della rete applicativa.

Questa soluzione permette, a regime, di integrare nell'applicazione o nel flusso operativo strumenti di assistenza contestuale (es. suggerimenti, analisi log, supporto alla documentazione), mantenendo il controllo sui dati grazie a un LLM locale.

### 9.2 API REST di monitoring (Node.js + PM2)

È stato implementato un backend Node.js nel percorso /home/registro/project-work/backend/src/server.js, con API REST esposta sulla porta 3443 per la fornitura di endpoint di health e status dell'ambiente. L'applicazione Node.js è gestita tramite PM2, che consente:

- pm2 list per elenco processi;
- pm2 info monitoring-api per dettagli del processo;
- pm2 restart monitoring-api per riavvii controllati;
- pm2 logs monitoring-api per consultare i log applicativi.

Le API di monitoring possono essere integrate in futuro con sistemi esterni (es. dashboard custom, sistemi di alerting) grazie a un'interfaccia REST standard.

### 9.3 Backup automatico

Per la messa in sicurezza dei dati è stato creato lo script `/home/registro/project-work/backup/backup.sh`, progettato per:

- raccogliere configurazioni e dati rilevanti;
- creare archivi compressi all'interno di `backup/archive/`;
- scrivere log di esecuzione in `backup/logs/cron.log`.

È stato configurato il cronjob:

```
0 1 * * * /home/registro/project-work/backup/backup.sh >> /home/registro/project-work/backup/logs/cron.log 2>&1
```

eseguito ogni notte, che consente di avere una copia regolare delle informazioni critiche dell'ambiente.

### 9.4 Issue / Fix

- Issue: presenza di una API Flask sperimentale non più allineata con la soluzione definitiva.  
Fix: consolidamento architetturale sull'unico backend Node.js per l'API di monitoring, semplificando manutenzione e riducendo il rischio di configurazioni divergenti.
- Issue: prime esecuzioni di backup fallite per permessi mancanti e utilizzo di path relativi negli script.  
Fix: aggiunta dei permessi eseguibili, aggiornamento a percorsi assoluti e verifica dei log in `backup/logs/cron.log` per confermare la riuscita.

## 10. Dicembre 2025 – Monitoraggio periodico, reportistica e consolidamento

### 10.1 Struttura `/opt/project` e script di monitoraggio

Per il monitoraggio periodico è stata definita la struttura `/opt/project/` con:

- `scripts/monitor.sh` per la raccolta dei parametri di sistema;
- `scripts/generate-report.sh` per la generazione del report giornaliero;
- `logs/system.log` come log centralizzato;
- `reports/report-YYYY-MM-DD.txt` come report sintetico per data.

`monitor.sh` raccoglie: timestamp, load average CPU, utilizzo memoria, spazio disco e stato rete (ad es. output ridotto di ip o ss). Il cronjob configurato (ogni 10 minuti) permette di avere una traccia temporale fine dell'andamento dell'host.

`generate-report.sh` elabora le ultime 50 righe di `system.log` producendo un report testuale giornaliero in `reports/`, facilitando l'analisi macro dell'andamento della VM nel giorno precedente. È stato pianificato un cronjob alle 2:00 per l'esecuzione dello script, con possibilità di estensione per invio via e-mail mediante `mailutils` e comando `mail`.

### 10.2 Validazioni e consolidamento

Sono stati verificati:

- il corretto funzionamento del cron di backup notturno;
- la periodicità di esecuzione di `monitor.sh` e `generate-report.sh`;
- i permessi eseguibili sugli script e i log generati.

È stato infine redatto un README tecnico che descrive l'architettura, i comandi principali, i percorsi di configurazione, i suggerimenti per reinstallazione/restore e le istruzioni per riprodurre l'ambiente da zero, completato dal presente diario di bordo come traccia di project history.

### 10.3 Issue / Fix

- Issue: `system.log` inizialmente molto verboso, con crescita rapida e scarsa leggibilità.  
Fix: ottimizzazione di `monitor.sh` per limitare la raccolta a metrica chiave e formati sintetici, migliorando leggibilità e contenimento dimensionale.
- Issue: alcuni cronjob non venivano eseguiti a causa dell'assenza del bit eseguibile sugli script.  
Fix: normalizzazione permessi (`chmod +x`) e test manuale degli script prima dell'attivazione definitiva in cron.

## 11. Valutazione finale dell'infrastruttura

La VM registro si presenta come un ambiente dimostrativo completo, configurato secondo buone pratiche: accesso SSH a chiave, firewall UFW attivo, utenza dedicata e separazione netta tra host e servizi containerizzati. Lo stack Docker offre un ecosistema coerente che integra applicazione web, DB, monitoring, logging, dashboard, LLM e API di monitoring, allineato ai modelli moderni di gestione di infrastrutture containerizzate.

Le automazioni di backup, monitoraggio e reportistica garantiscono un livello di resilienza e osservabilità adeguato a un progetto dimostrativo orientato al mondo enterprise, mentre la documentazione tecnica e questo diario di bordo permettono a un'azienda interessata di comprendere a fondo scelte architetturali, implementazioni e livello di maturità del progetto "Registro Elettronico Didattix".