SDD
SYSTEM DESIGN DOCUMENT

ModuLink

Riferimento	NC08_SDD_ver.1.2
Versione	1.2
Data	19/11/2025
Destinatario	Studenti di Ingegneria del Software 2025/26
Presentato da	Buzi Arjel, Carpentieri Daniele, Chikviladze Aleksandre, Cito Roberto.
Codice Gruppo	NC08
Approvato da	

Revision History

Data	Versione	Descrizione	Autori
19/11/2025	0.1	Prima stesura	Roberto e Arjel
24/11/2025	0.2	Introduzione	Roberto e Aleksandre
24/11/2025	0.3	Design Goals	Aleksandre
26/11/2025	0.4	TradeOff, Definizioni	Daniele
28/11/2025	0.5	Organizzazione del documento	Daniele e Arjel
01/12/2025	0.6	Diagramma Architetturale e Ristrutturazione del Class Diagram	Tutti
02/12/2025	0.7	Correzione e Modifiche	Daniele
03/12/2025	0.8	Schema ER Dizionario di dati	Aleksandre
06/12/2025	0.9	Completamento degli Argomenti	Aleksandre e Arjel
09/12/2025	1.0	Correzione e Modifiche	Daniele
20/12/2025	1.1	Revisione Finale	Tutti
18/01/2026	1.2	Aggiornamento Spiegazione “Architettura Sistema Proposto” post Suggerimenti	Daniele

Team members

Nome	Ruolo nel progetto	Acronimo	Informazioni di contatto
Roberto Cito	Team Member	RC	r.cito@studenti.unisa.it
Daniele Carpentieri	Team Member	DC	d.carpentieri8@studenti.unisa.it
Aleksandre Chikviladze	Team Member	AC	a.chikviladze@studenti.unisa.it
Arjel Buzi	Team Member	AB	a.buzi@studenti.unisa.it

Sommario

1 Introduzione

1.1 Obiettivo del Sistema

L'obiettivo del progetto è lo sviluppo di una piattaforma web, semplice ed intuitiva, con una struttura base modulare e interamente personalizzabile. È concepita per le aziende che desiderano ottimizzare la gestione dei processi interni e delle risorse digitali.

Il sistema si propone come uno strumento flessibile e scalabile, in grado di adattarsi dinamicamente alle esigenze operative specifiche di ogni organizzazione, incrementando l'efficienza operativa quotidiana; evitando la necessità di utilizzare software su misura o servizi che possono risultare onerosi

La piattaforma permette alle aziende, previa registrazione, di configurare in piena autonomia la propria dashboard, selezionando e integrando moduli applicativi (app) da un catalogo predefinito. Questo catalogo potrà essere periodicamente aggiornato e ampliato con lo sviluppo di nuove funzionalità.

Ciascun modulo è dedicato a una funzionalità specifica, come ad esempio:

- Gestione di calendari, appuntamenti e scadenze.
- Organizzazione di attività (task) e progetti.
- Visualizzazione e consultazione dei dati aziendali.
- Strumenti di analisi e reporting per il monitoraggio degli andamenti.

Il sistema supporta una gestione granulare degli accessi. L'azienda può registrare i propri dipendenti e collaboratori, associando a ciascuno un ruolo che permette di accedere a dashboard e funzionalità a loro dedicate. Ciò garantisce un controllo sui flussi informativi e una chiara definizione delle responsabilità operative.

Oltre alla selezione da catalogo, la piattaforma prevede la possibilità di richiedere lo sviluppo di moduli personalizzati. Questi moduli verranno progettati ad hoc per rispondere a necessità uniche del cliente, assicurando la continuità operativa con i processi e le infrastrutture preesistenti nell'azienda.

In sintesi, ModuLink si configura come un ecosistema digitale integrato, pensato per offrire una gestione centralizzata e intuitiva delle applicazioni e dei dati interni.

1.2 Obiettivi di Design (Design Goals)

Questa sezione delinea i principi tecnici fondamentali e obiettivi di qualità che caratterizzano lo sviluppo della piattaforma. Questa tabella collega gli obiettivi di design ai requisiti non funzionali di origine, classificandoli in Rank (in ordine di importanza) e categorie.

I design goal si suddividono nelle seguenti categorie:

- **Performance**: includono i requisiti di spazio e velocità imposti sul sistema.
- **Dependability**: determinano quanto sforzo deve essere speso per minimizzare i fallimenti del sistema (crash, falle di sicurezza) e le loro conseguenze.
- **Maintenance**: determina quanto sforzo è necessario per modificare il sistema dopo il suo rilascio.
- **End User**: includono qualità che sono desiderabili dal punto di vista dell'utente, ma che non sono state coperte dai criteri di Performance e Dependability.

Ciascun design goal è descritto da:

- **Rank**, che ne specifica un valore di priorità compreso tra 1 e 16 (1 massima e 16 minima).
- **ID Design Goal**, un identificatore univoco e un nome esplicativo.
- **Descrizione,** una descrizione del design goal.
- **Categoria**, ovvero la categoria di appartenenza del design goal.
- **RNF di origine**, ovvero il requisito non funzionale che lo ha generato.

Design Goals

RANK	ID Design Goal	Descrizione	Categoria	RNF di origine
1	DG_1 Segregazione Multi-Tenant	Il sistema deve garantire un isolamento tra i dati delle diverse aziende e deve impedire accessi incrociati e non autorizzati	Dependability	RNF-SIC.2
5	DG-2 Modularità	La modifica di un modulo nel catalogo non deve richiedere il riavvio o la ricompilazione del sistema centrale	Maintenance	RNF-AFF.7
2	DG_3 Sicurezza	L'accesso a ogni funzionalità/risorsa dev'essere governato da un sistema granulare di ruoli e permessi, configurabili dal client	Dependability	RNF-SIC.3
9	DG_4 Configurabilità	La dashboard dev'essere personalizzabile dall'utente tramite configurazione	End User	RNF.USA.3
7	DG_5 Scalabilità	Il sistema deve supportare l'aumento del carico di lavoro aggiungendo nuove stanze server senza modificare l'architettura del software	Performance	RNF-PRE.3
8	DG_6 Tempi di risposta	Le operazioni di lettura e salvataggio sulla dashboard e sui moduli devono garantire un tempo di risposta sotto 2 secondi	Performance	RNF-PRE.1

10	DG_7 usabilità	Tutti i moduli default/custom devono avere un design unificato per garantire un'esperienza coerente	End User	RNF-USA.1 RNF-USA.2 RNF-USA.4
3	DG_8 Disponibilità	Il sistema deve garantire un uptime elevato durante gli orari lavorativi standard	Dependability	RNF-AFF.1
4	DG_9 Integrità	Il sistema deve prevenire la corruzione dei dati in caso di concorrenza nell'accesso alle stesse risorse	Dependability	RNF-AFF.2
6	DG_10 Facilità di deployment	L'utente deve sempre sapere cosa sta succedendo: se il sistema sta caricando o elaborando un report, deve mostrarlo chiaramente.	Maintenance	RNF-USA.4

Trade-Off

Trade-Off	Descrizione
Segregazione multi-tenant vs scalabilità	Per garantire un forte isolamento tra i tenant (aziende) si possono adottare meccanismi di separazione logica o fisica dei dati, con controlli aggiuntivi su ogni richiesta. Questo aumenta la sicurezza ma può rendere più complessa la gestione dell'infrastruttura e limitare la scalabilità orizzontale, richiedendo più risorse e una progettazione più avanzata per supportare un alto numero di aziende e utenti concorrenti
Aggiornamenti automatici del catalogo vs stabilità del sistema	Automatizzare l'installazione e l'aggiornamento dei moduli riduce i costi di manutenzione e semplifica la gestione, ma introduce il rischio di incompatibilità o malfunzionamenti improvvisi. Un modulo aggiornato senza supervisione potrebbe compromettere temporaneamente la stabilità o generare errori nelle dashboard attive.

1.3 Definizioni, acronimi e abbreviazioni

Vengono riportati di seguito alcune definizioni presenti nel documento corrente:

- **Sottosistema:** un sottoinsieme dei servizi del dominio applicativo, formato da servizi legati da una relazione funzionale.
- **Design Goal:** le qualità sulle quali il sistema deve essere focalizzato.
- **Dati Persistenti**: dati che sopravvivono all'esecuzione del programma che li ha creati e che dunque vengono salvati.
- **Mapping Hardware/Software**: studio della connessione tra parti fisiche e logiche di cui si compongono il sistema.
- **SDD:** System Design Document
- **RAD:** Requirements Analysis Document

1.4 Riferimenti

Di seguito sono riportati i riferimenti a risorse che possono agevolare la lettura del documento, o fornire maggiori informazioni.

Nome Riferimento	Riferimento
Libro	"Object-Oriented Software Engineering: Conquering Complex and Changing Systems", terza edizione, Bernd Bruegge & Allen Dutoit, 2014.
SOW	Statement Of Work (SOW) Chiuso
RAD	Requirement Analysis Document Chiuso
ODD	Object Design Document Chiuso
TPD	Test Plan Document Chiuso

1.5 Organizzazione del documento

Il presente documento di System Design è organizzato in quattro sezioni principali:

- **Introduzione:** Presenta una panoramica generale del sistema, delineando lo scopo del progetto e gli obiettivi di design che si intendono raggiungere.
- **Architettura Software Corrente:** Analizza lo stato dell'arte e la struttura tecnica del software esistente, fornendo il contesto necessario per le evoluzioni proposte.
- **Architettura Software Proposta:** Descrive la configurazione futura del sistema. Questa sezione approfondisce la partizione in sottosistemi, il mapping Hardware/Software e la strategia di gestione dei dati persistenti. Include inoltre il dettaglio dei singoli sottosistemi e la definizione delle *boundary conditions* (condizioni al contorno) dell'intero ecosistema.
- **Glossario:** Fornisce una lista dettagliata dei termini tecnici e degli acronimi utilizzati nel documento, con le relative definizioni.

2 Architettura del sistema corrente

Attualmente, non si riscontra una piattaforma in grado di integrare in un unico ambiente l'intero spettro delle funzionalità offerte da ModuLink, quali: gestione utenti e ruoli, moduli dinamici, task, calendario, magazzino e dashboard configurabile.

Il mercato offre soluzioni parzialmente sovrapponibili, tuttavia caratterizzate da un'elevata frammentazione; si tratta spesso di strumenti distinti per il task management, il CRM, la gestione del magazzino, i calendari o moduli personalizzati, frequentemente privi di interoperabilità tra loro.

Sistemi evoluti, come Salesforce, approssimano il concetto di ecosistema modulare fornendo un'estesa copertura funzionale e consentendo l'espansione della piattaforma mediante componenti aggiuntivi. Cionondimeno, tali soluzioni comportano una complessità considerevole e costi rilevanti. ModuLink, di contro, mira a preservare un approccio più snello, configurabile e intuitivo, pur garantendo i benefici fondamentali: modularità, personalizzazione della dashboard, isolamento multi-tenant e gestione centralizzata dei flussi aziendali.

3 Architettura del sistema proposto

3.1 Panoramica sulla sezione

Il sistema proposto adotta un'architettura logica di tipo Three-Tier, suddivisa in Client (Presentation Tier), Server (Logic Tier) e Database (Data Tier).

All'interno del Logic Tier (Server), l'applicazione è strutturata internamente seguendo il design pattern MVC (Model-View-Controller), implementato tramite il framework Spring Boot. Questa distinzione permette di separare chiaramente l'infrastruttura di distribuzione (i Tier) dall'organizzazione interna del codice (il Pattern), garantendo scalabilità, manutenibilità e una gestione efficiente delle dipendenze.

Tale struttura favorisce miglioramenti significativi in termini di:

- leggibilità del codice
- semplicità di manutenzione
- possibilità di riuso e sostituzione modulare dei componenti

Per il front-end saranno utilizzati **HTML5**, **CSS3** e **Thymeleaf**, che consentono una generazione dinamica delle viste perfettamente integrata con Spring.

La logica applicativa (back-end) sarà sviluppata interamente in Java, sfruttando i moduli del framework Spring Boot, che offrono un modello coerente per la gestione dei controller, dei servizi e delle dipendenze.

Per la persistenza dei dati verranno utilizzati:

- **Spring Data JPA**, per l'accesso strutturato al database e la gestione delle entità
- **MySQL**, eseguito all'interno di un container **Docker**
- infrastruttura cloud fornita da **Google Cloud**, che ospita il container e ne garantisce disponibilità, sicurezza e scalabilità

Questa configurazione permette di mantenere un ambiente stabile, portabile e facilmente replicabile, semplificando sia il ciclo di sviluppo sia il dispiegamento del sistema.

3.2 Decomposizione in sottosistemi

I sottosistemi individuati sono:

- **Gestione Responsabile (GDR):** si occupa di gestire il processo di registrazione iniziale del Responsabile e della creazione della relativa istanza privata per l'Azienda (Tenant), configurando l'ambiente di lavoro iniziale.

- **Gestione Utenti e Autenticazione (GDU):** è responsabile delle funzionalità di sicurezza e accesso, inclusi il Login, il Logout, il primo accesso con cambio password obbligatorio, e la gestione completa del ciclo di vita degli account dei Dipendenti (registrazione, modifica e rimozione).

- **Gestione Moduli Aziendali (GMA):** è il cuore della modularità del sistema; è responsabile della visualizzazione dello "store" dei moduli, dell'installazione o rimozione dei moduli (App) nella dashboard aziendale e della gestione delle richieste per lo sviluppo di moduli personalizzati.

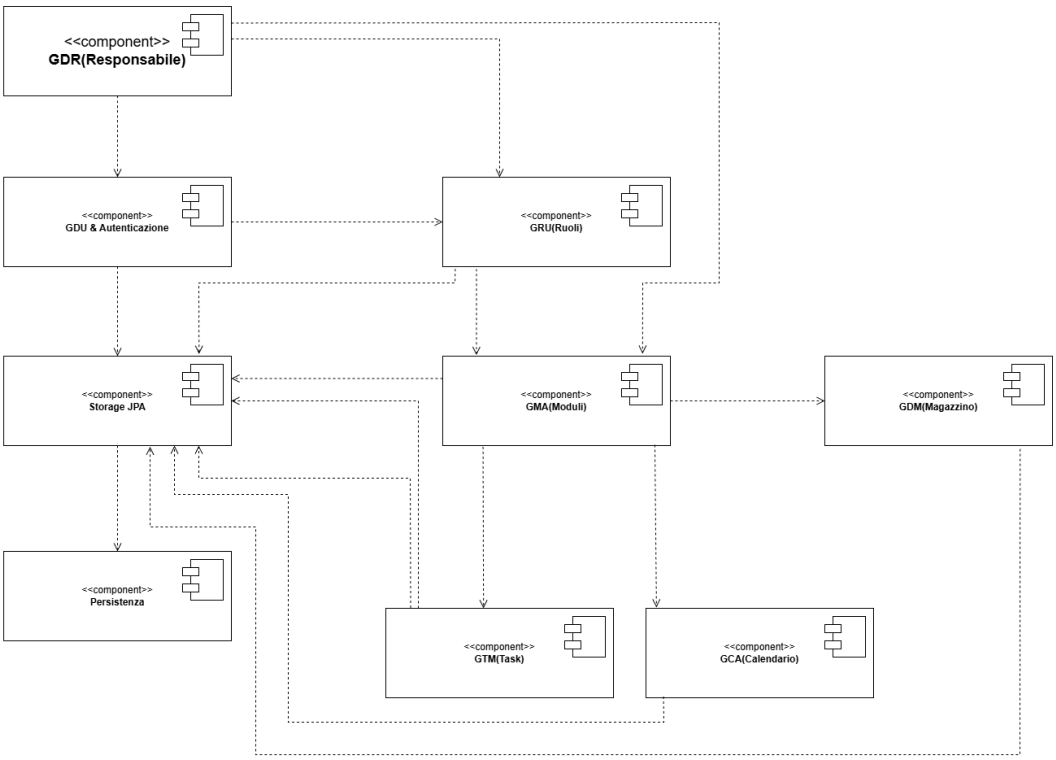
- **Gestione Ruoli (GRU):** si occupa della creazione e definizione dei ruoli aziendali (RBAC) e della mappatura dei permessi, permettendo di associare specifici privilegi di accesso ai moduli per i vari dipendenti.

- **Task Manager (GTM):** si occupa delle funzioni operative riguardanti la creazione, l'assegnazione, la modifica, il monitoraggio dello stato e la visualizzazione dei task lavorativi tra i dipendenti.

- **Gestione Magazzino (GDM):** è responsabile della logica di business relativa all'inventario, occupandosi della creazione, aggiornamento e rimozione dei prodotti, nonché della registrazione delle operazioni di acquisto e vendita.

- **Gestione Calendario (GCA):** si occupa della gestione temporale delle attività, permettendo la creazione di eventi, la gestione degli inviti per utenti o ruoli specifici e la sincronizzazione delle scadenze.

- **Persistenza e Storage:** si occupa di gestire la persistenza dei dati strutturati (tramite database relazionale come PostgreSQL/MySQL) e non strutturati, garantendo l'isolamento dei dati tra i vari tenant (Aziende) e interfacciandosi con il backend sviluppato in Spring Boot.



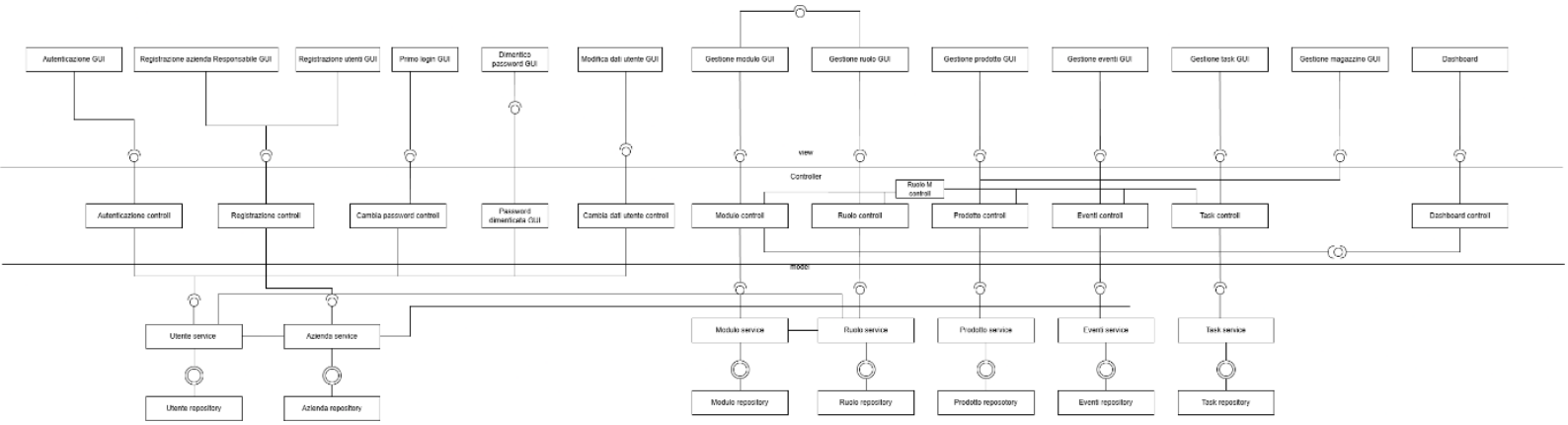
Alcuni sottosistemi saranno gestiti da componenti COTS (Commercial off the shelf), di seguito un elenco:

- Storage JPA verrà gestito da Spring Data JPA
- Persistenza sarà gestita attraverso un DBMS relazionale su sistema **Google Cloud**

Di seguito una vista dettagliata di ciascun sottosistema evidenziando le componenti principale:

- **GUI**: Graphic User Interface, che contiene le varie view che saranno renderizzate per creare le pagine web da mostrare al cliente.
- **Controller**: si occupa della logica per il controllo del sistema.
- **Service**: si occupa della logica di business.
- **DAO**: Data Access Object, che si occupa di fornire accesso ai dati persistenti.

Diagramma Architetturale

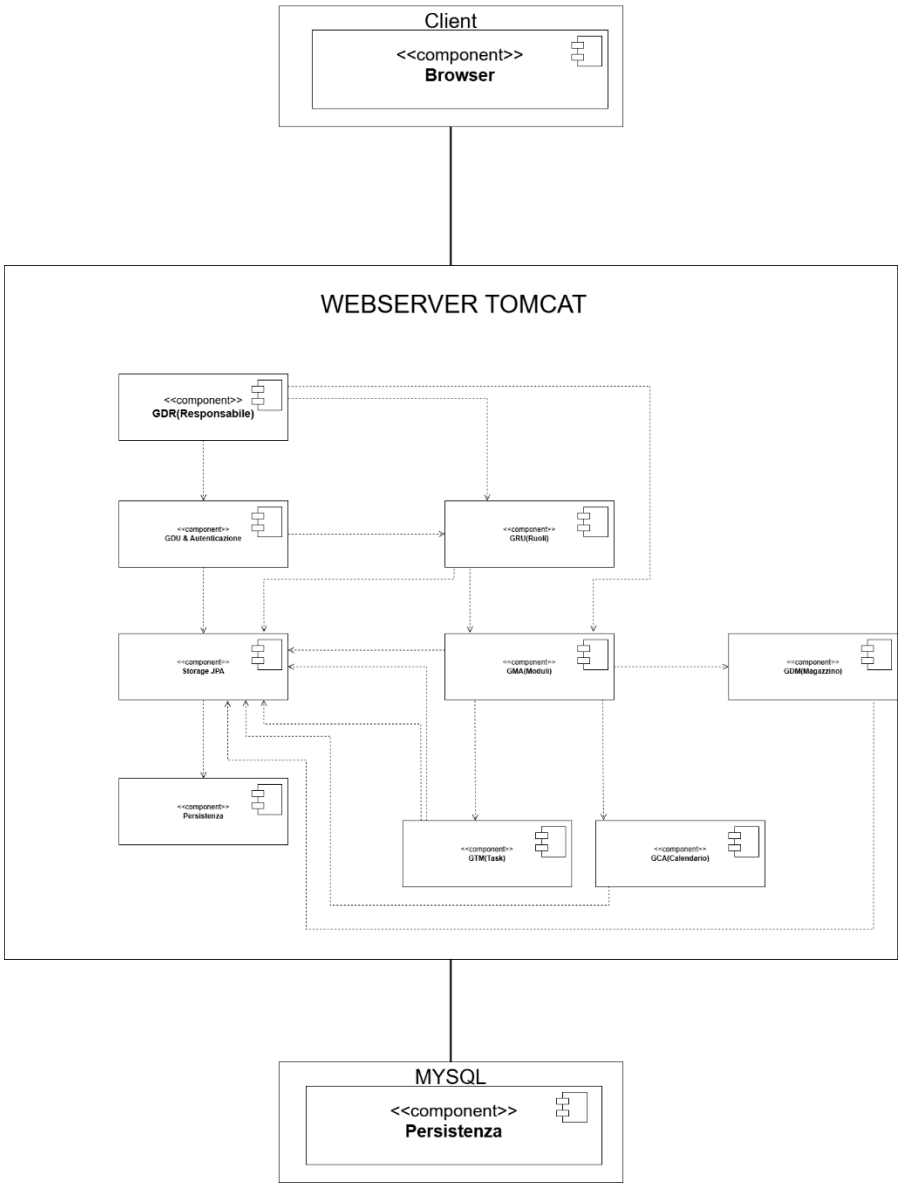


3.3 Mapping Hardware/Software

L'infrastruttura di sistema si basa su un modello architettonico di tipo *Client-Server*. Il backend dell'applicazione è ospitato su un unico nodo computazionale (Server), configurato per gestire le richieste provenienti dai client tramite protocollo HTTP/HTTPS.

Poiché il sistema adotta un'architettura non distribuita, l'intero stack applicativo risiede in un unico ambiente di runtime. Dal lato utente, l'accesso alle funzionalità è garantito in modalità *cross-platform* attraverso un comune browser web, rendendo il sistema indipendente dall'hardware locale del cliente.

La distribuzione delle componenti software sulle risorse hardware e le relative relazioni di interdipendenza sono illustrate nel seguente **Diagramma di Deployment UML**.



3.4 Gestione dei dati persistenti

Introduzione

Per assicurare una gestione efficace dei dati persistenti del sistema, la scelta è ricaduta su un **database relazionale gestito tramite DBMS**. Questa soluzione consente di affrontare simultaneamente due esigenze critiche: facilitare l'accesso concorrente alle informazioni e preservare l'**integrità e la coerenza dei dati** attraverso i meccanismi di controllo centralizzato offerti dal DBMS stesso.

Questa decisione architetturale si allinea perfettamente con i **design goals del progetto**, garantendo molteplici vantaggi operativi.

Innanzitutto, il DBMS permette di definire e applicare automaticamente **vincoli di integrità su vari livelli**, assicurando che i dati mantengano sempre uno stato valido e coerente indipendentemente dalle operazioni effettuate.

Secondariamente, la **privatezza e la sicurezza dell'accesso** sono gestite tramite sistemi di autorizzazione granulare: ogni membro del team può disporre di permessi differenziati, accedendo solo alle porzioni di dati rilevanti per il proprio ruolo e alle operazioni specifiche di cui ha necessità.

Inoltre, il DBMS fornisce **strumenti affidabili per il backup e il disaster recovery**, proteggendo i dati da potenziali guasti hardware o software e consentendo il ripristino rapido dello stato consistente della base di dati.

Infine, l'**atomicità delle transazioni** garantisce che sequenze complesse di operazioni si completino interamente oppure non lascino alcun effetto sui dati, mantenendo la base di dati sempre in uno stato logicamente coerente con il modello di business.

Purtroppo, lavorare con un database usando solamente l'hosting in locale rallenta lo sviluppo. Per evitare problemi di velocità si è optato per un'infrastruttura database in cloud.

La scelta è ricaduta su **Google Cloud Platform**, sfruttando i **300 dollari di credito gratuito messo a disposizione**, un ammontare ampiamente sufficiente per coprire i costi del servizio gestito DBMS.

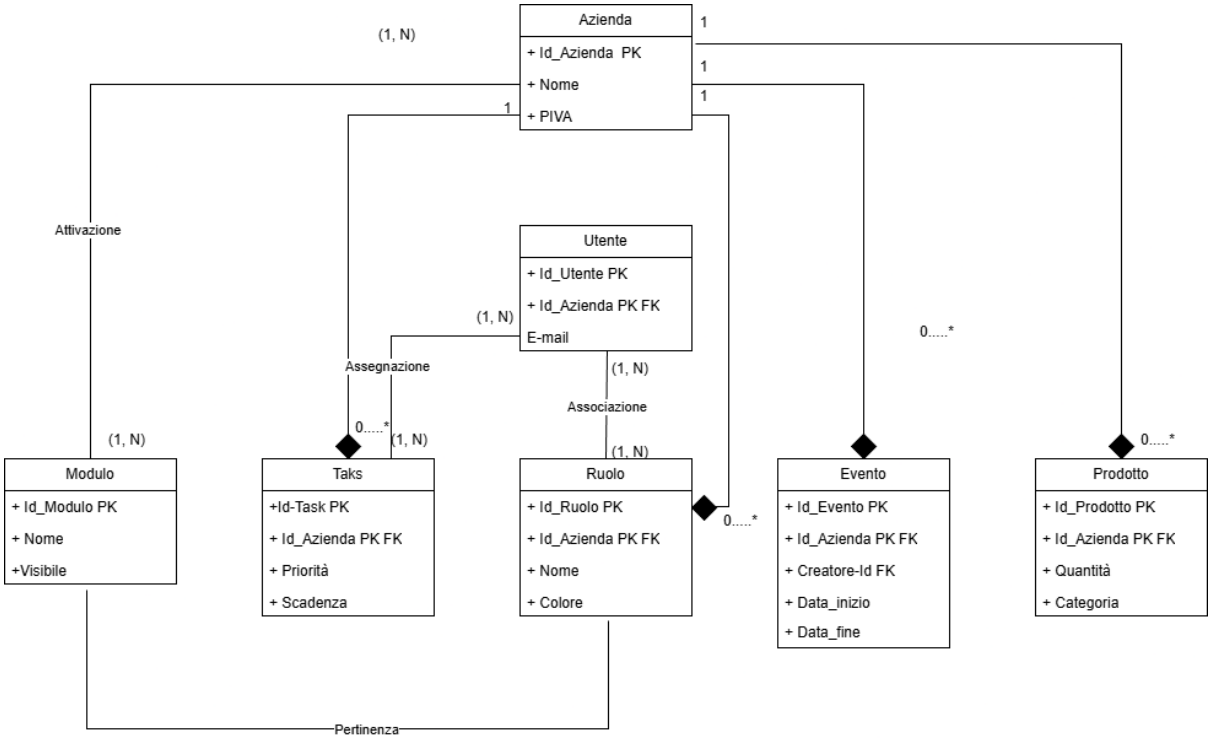
Questo approccio offre molteplici benefici: permette a tutti i membri del team di operare su una **medesima istanza condivisa** senza sincronizzazioni manuali, elimina la necessità di mantenere copie locali della base di dati sui singoli dispositivi, e rimuove sistematicamente gli errori di configurazione individuale legati all'hosting locale.

Di conseguenza, il tempo dedicato alla risoluzione di problemi infrastrutturali viene significativamente ridotto, permettendo al team di concentrarsi completamente sulla logica applicativa.

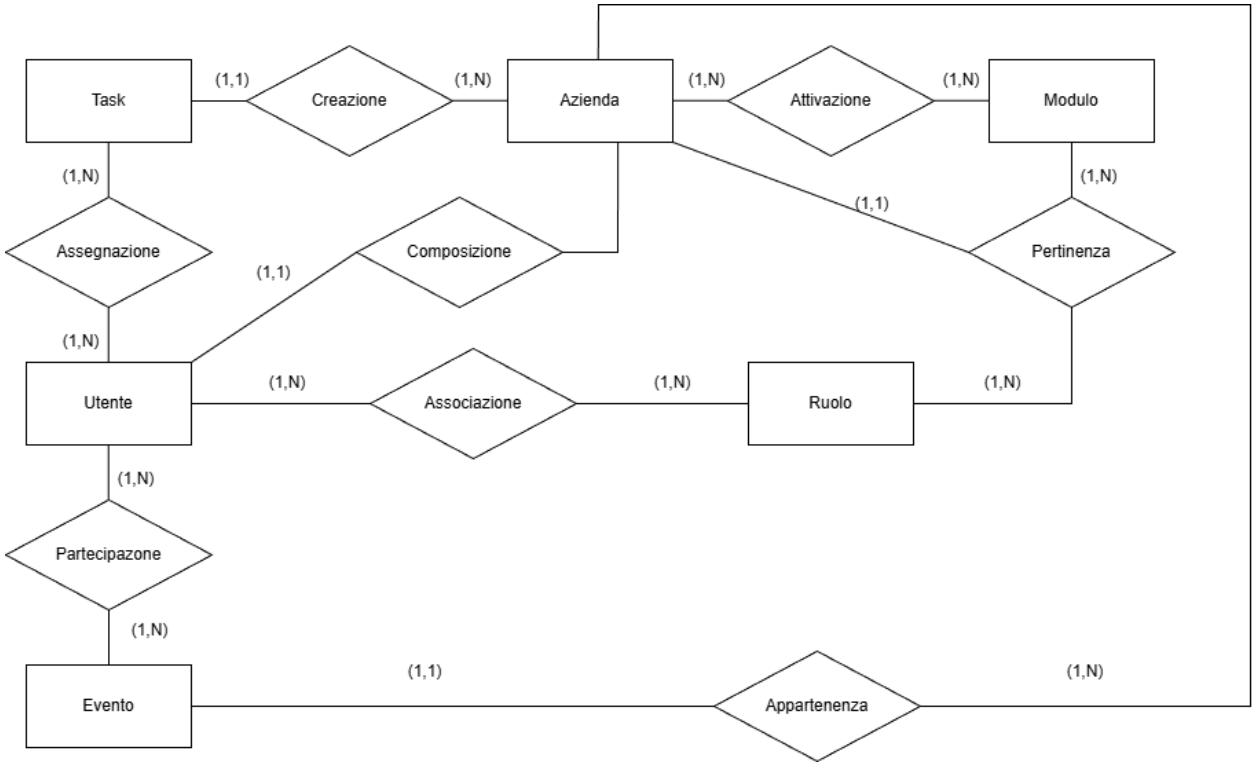
Inoltre, la medesima piattaforma cloud verrà successivamente utilizzata anche per l'**hosting del progetto nel suo complesso**, sfruttando efficientemente lo stesso credito disponibile.

CD_SDD: Entity Class Diagram ristrutturato

Nota: Sono visualizzati solo gli attributi principali. Dati anagrafici o descrittivi (es. nome, cognome, indirizzo) non sono rappresentati per non appesantire il diagramma.



ER: Schema ER del database



Dizionario dei dati

Di seguito si mostrano gli attributi per ogni entità individuata:

Nome Entità	**Utente**		
Descrizione	Contiene i dati relativi ad un utente		
Nome campo	**Tipo**	**Vincolo di chiave**	**Altri vincoli**
ID_Utente	Int	PRIMARY KEY	NOT NULL
ID_Azienda	Int	PRIMARY KEY FOREIGN KEY(Azienda)	NOT NULL
Email	varchar(50)		NOT NULL
Password	varchar(255)		NOT NULL
Nome	varchar(50)		NOT NULL
Cognome	varchar(50)		NOT NULL
Telefono	varchar(50)		NOT NULL
Immagine_Profilo	varchar(50)		NULL

Nome Entità	**Azienda**		
Descrizione	Contiene i dati relativi ad un'azienda		
Nome campo	**Tipo**	**Vincolo di chiave**	**Altri vincoli**
ID_Azienda	Int	PRIMARY KEY	NOT NULL
Nome	varchar(50)		NOT NULL
P_IVA	varchar(50)		NOT NULL
Indirizzo	varchar(50)		NULL
Città	varchar(50)		NULL
CAP	varchar(50)		NULL
Telefono	varchar(50)		NULL
Logo	varchar(50)		NULL

Nome Entità	**Ruolo**		
Descrizione	Contiene i dati relativi a ruolo		
Nome campo	**Tipo**	**Vincolo di chiave**	**Altri vincoli**
ID_Ruolo	Int	PRIMARY KEY	NOT NULL
ID_Azienda	Int	PRIMARY KEY FOREIGN KEY(Azienda)	NOT NULL
Nome	varchar(50)		NOT NULL
Colore	varchar(50)		NOT NULL
Descrizione	Text		NULL

Nome Entità	**Modulo**		
Descrizione	Contiene i dati relativi a un modulo		
Nome campo	**Tipo**	**Vincolo di chiave**	**Altri vincoli**
ID_Modulo	Int	PRIMARY KEY	NOT NULL
Nome	varchar(50)		NOT NULL
Descrizione	Text		NOT NULL
URL_Icona	varchar(50)		NULL

Nome Entità	**Task**		
Descrizione	Contiene i dati relativi a un task		
Nome campo	**Tipo**	**Vincolo di chiave**	**Altri vincoli**
ID_Task	Int	PRIMARY KEY	NOT NULL
ID_Azienda	Int	PRIMARY KEY FOREIGN KEY(Azienda)	NOT NULL
Titolo	varchar(150)		NOT NULL
Priorità	Int		NOT NULL DEFAULT 0
Scadenza	Date		NULL
ID_Utente_Creatore	Int	FOREIGN KEY	NOT NULL
Data_Creazione	Datetime		NOT NULL
Data_Completamento	Datetime		NULL

Nome Entità	**Evento**		
Descrizione	Contiene i dati relativi a un evento		
Nome campo	**Tipo**	**Vincolo di chiave**	**Altri vincoli**
ID_Evento	Int	PRIMARY KEY	NOT NULL
ID_Azienda	Int	PRIMARY KEY FOREIGN KEY(Azienda)	NOT NULL
Nome	varchar(200)		NOT NULL
Luogo	varchar(300)		NULL
Data_Ora_inizio	Datetime		NOT NULL
Data_FIne	Datetime		NULL

Nome Entità	**Prodotto**		
Descrizione	Contiene i dati relativi a un prodotto		
Nome campo	**Tipo**	**Vincolo di chiave**	**Altri vincoli**
ID_Prodotto	Int	PRIMARY KEY	NOT NULL
ID_Azienda	Int	PRIMARY KEY FOREIGN KEY(Azienda)	NOT NULL
Nome	varchar(50)		NOT NULL
Quantità	Int		NOT NULL
Prezzo	Float		NULL
Descrizione	Text		NULL
Categoria	Text		NULL

3.5 Controllo degli Accessi e Sicurezza

Di seguito viene mostrata la matrice degli accessi per poter tenere traccia di quali attori possono accedere ai quali dei servizi offerti dal sistema.

Ricordiamo che il Responsabile non è altro che un utente che possiede tutti i permessi a tutti i moduli dell'azienda; quindi, nel caso in cui parliamo di "Utente con Permessi per il modulo" stiamo parlando non solo di utente con i permessi attivi ma anche del responsabile stesso.

GDR (Gestione del Responsabile)		
Attori **Oggetti**	Responsabile Registrato	Responsabile Non Registrato
Registrazione		RegistrazioneAziendaEdResponsabile
Autenticazione	Login Logout VisualizzaDashBoard ModificaDatiResponsabile CancellazioneAccount CambioPassword PasswordDimenticata	

GDU (Gestione degli Utenti)		
Attori **Oggetti**	Utente Senza Permessi per il modulo	Utente Con Permessi per il modulo
Registrazione		RegistraNuovoUtente
Autenticazione	Login Logout FirstLogin VisualizzaAreaUtente ModificaDatiUtente CancellazioneAccount CambioPassword PasswordDimenticata	Login Logout FirstLogin VisualizzaAreaUtente ModificaDatiUtente CancellazioneAccount CambioPassword PasswordDimenticata
Gestione Utenti		ModificaDatiUtenti ElimazioneUtenti Assegnazione/rimozioneRuoli CambiaPasswordUtenti

In questo caso abbiamo dovuto distinguere "*Responsabile*" e "*Utente con Permessi al modulo*" poiché il responsabile ha dei privilegi maggiore rispetto all'utente che gli concede l'accesso a funzioni aggiuntive.

GMA (Gestione dei Moduli Aziendali)			
Attori **Oggetti**	Responsabile	Utente con Permessi al modulo	Utente senza Permessi al modulo
Visualizzazione	VisualizzaModuliDashbord VisualizzaStoreModuli	VisualizzaModuliDashbord	VisualizzaModuliDashbord
Richieste	MandaRichiestaNuovoModulo		
Gestione	Aggiunta/RimozioneModuli		
Gestione Assegnazione	Aggiunta/RimozioneRuoloAUnModulo	Aggiunta/RimozioneRuoloAdModulo	

GRU (Gestione dei Ruoli)		
Attori **Oggetti**	Utente Senza Permessi per il modulo	Utente Con Permessi per il modulo
Visualizzazione		VisualizzaRuoli
Gestione		CreazioneRuolo Modifica/RimozioneRuolo

GTM (Gestione del Task Manager)		
Attori **Oggetti**	Utente Senza Permessi per il modulo	Utente Con Permessi per il modulo
Visualizza	VisualizzaGestioneTask	VisualizzaGestioneTask
Gestione	CreazioneTask RimozioneTask ModificaTask	CreazioneTask RimozioneTask ModificaTask AssegnazioneTask

GDM (Gestione del Magazzino)		
Attori **Oggetti**	Utente Senza Permessi per il modulo	Utente Con Permessi per il modulo
Visualizza		VisualizzaGestioneMagazzino
Gestione		CreazioneProdotto ModificaProdotto RimozioneProdotto AquistoProdotto VenditaProdotto

GCA (Gestione del Calendario)		
Attori **Oggetti**	Utente Senza Permessi per il modulo	Utente Con Permessi per il modulo
Visualizzazione	VisualizzaCalendario	VisualizzaCalendario
Gestione	CreaEvento ModificaEvento CancellaEvento InvitaPersona	CreaEvento ModificaEvento CancellaEvento InvitaPersona invitaRuoli

3.6 Controllo globale del software

Il sistema ModuLink è una piattaforma web interattiva in cui ogni funzionalità viene attivata in risposta alle azioni compiute dall'utente tramite l'interfaccia grafica. Quando un utente desidera accedere a una funzione del sistema, interagisce con la GUI, che genera l'evento corrispondente. Questo evento viene intercettato dal relativo handler, il quale gestisce il flusso e lo instrada verso il sottosistema responsabile della logica di controllo.

Il sottosistema di controllo elabora la richiesta e, quando necessario, delega l'esecuzione ai servizi applicativi che implementano le funzionalità di business dei vari moduli (es. gestione utenti, ruoli, task, calendario, magazzino). Questo modello garantisce una separazione chiara tra interfaccia, logica di controllo e logica applicativa, semplificando l'evoluzione dei moduli e l'estensibilità della piattaforma.

Poiché ModuLink è una web application strutturata secondo un approccio modulare, l'intero sistema si basa su un paradigma event-driven: l'interazione dell'utente genera eventi che attivano i vari componenti dell'architettura, assicurando un flusso di elaborazione reattivo, coerente e facilmente scalabile.

3.7 Condizioni limite

Avvio del sistema

Identificativo	UCBC_1 – Avvio del Sistema	**Data**	03/12/2025
		Versione	1.0
		Autori	Daniele Carpentieri
Descrizione	Lo UC permette l'avvio del sistema		
Attore principale	Amministratore		
Attori secondari	NA		
Entry condition	L'Amministratore accede al Server		
Exit condition **On success**	Il sistema viene avviato correttamente		
Exit condition **On failure**	Il sistema non si avvia		
Flusso di eventi principale			
1	Amministratore	Esegue sulla macchina il comando che avvia il sistema.	
2	Sistema	Verifica la disponibilità e sanità dei dati persistenti e, se disponibili e sani, rende disponibili i suoi servizi e rende le sue funzionalità disponibili agli utenti.	
I Flusso di Eventi Alternativo: I Dati Persistenti sono danneggiati			
2.a1	Sistema	Notifica l'Amministratore di problemi ai dati persistenti e non effettua l'avvio.	
2.a2	Amministratore	Verifica e corregge l'indisponibilità dei dati persistenti.	
2.a3	Amministratore	Esegue il Passaggio 1	

Aggiornamento del sistema

Identificativo	UCBC_2 – Upload di una nuova versione del sistema	**Data**	03/12/2025
		Versione	1.0
		Autori	Arjel Buzi
Descrizione	L'amministratore avvia la procedura pubblicando una nuova versione del codice su GitHub; il sistema, tramite GitHub Actions, genera e pubblica l'immagine aggiornata su Docker Hub, che viene poi scaricata e avviata dal server.		
Attore principale	Amministratore		
Attori secondari	GitHub, GitHub Actions, Docker Hub, Server		
Entry condition	La nuova versione del codice sorgente è pronta e caricata dalla repository locale dell'amministratore.		
Exit condition **On success**	La nuova immagine è stata compilata, pubblicata su Docker Hub, scaricata dal server ed è ora in esecuzione.		
Exit condition **On failure**	La pipeline non completa la compilazione dell'immagine oppure il server non riesce a scaricare o avviare la nuova versione.		
Flusso di eventi principale			
1	Amministratore	Effettua un push della nuova versione del codice sulla repository GitHub.	
2	GitHub Actions	Avvia automaticamente la pipeline di build dell'immagine Docker.	
3	GitHub Actions	In caso di build completata, pubblica l'immagine su Docker Hub.	
4	Server	Verifica la presenza di una nuova immagine su Docker Hub.	
5	Server	Esegue un pull dell'immagine aggiornata.	
6	Server	Avvia la nuova versione del sistema utilizzando l'immagine appena scaricata.	

Flusso alternativo: Errore nella pipeline GitHub Actions		
2.a1	GitHub Actions	La compilazione dell'immagine fallisce.
2.a2	GitHub Actions	L'immagine non viene pubblicata su Docker Hub e il processo si interrompe.
Flusso di Eventi Alternativo: Il Server non vede nessuna nuova immagine		
4.a	Server	Non ci sono nuove versioni
Flusso di Eventi Alternativo: La nuova versione fallisce all'avvio nel server		
6.a	Server	Il server carica la vecchia versione ancora salvata in memoria
6.b	Server	Avvia la precedente versione

Spegnimento del sistema

Identificativo	UCBC_3 –– Spegnimento del Sistema	**Data**	03/12/2025
		Versione	1.0
		Autori	Roberto Cito
Descrizione	Lo UC permette lo spegnimento del sistema		
Attore principale	Amministratore		
Attori secondari	NA		
Entry condition	L'Amministratore accede al Server e invia il comando di spegnimento del container		
Exit condition On success	Il sistema viene spento correttamente		
Exit condition On failure	Il sistema rimane attivo		
Flusso di eventi principale			
1	Amministratore	Esegue sulla macchina il comando che spegne il sistema.	
2	Sistema	Verifica che non ci siano eventi o transazioni in corso, se non ci sono, spegne il container.	
I Flusso di Eventi Alternativo: Ci sono transazioni/eventi ancora in corso			
2.a1	Sistema	Notifica l'Amministratore della problematica e attiva un contatore di 30 secondi	
2.a2	Sistema	Una volta scaduto il tempo ritorna al passaggio due.	

Fallimento del sistema

Identificativo		UCBC_4 – Fallimento del Sistema	**Data**	24/11/2020
			Versione	1.0
			Autori	Aleksandre Chikviladze
Descrizione		Il sistema fallisce durante il normale funzionamento		
Attore principale		Sistema		
Attori secondari		Amministratore, Server		
Entry condition		Il Sistema viene terminato inaspettatamente		
Exit condition On success		Il Sistema viene riavviato correttamene		
Exit condition On failure		Il Sistema non viene riavviato		
Flusso di eventi principale				
1	Sistema	Rileva un Problema		
2	Sistema	Si spegne, mandando un'e-mail all'amministratore con il relativo problema rilevato.		
3	Amministratore	L'amministratore accede al server		
4	Amministratore	Risolve la problematica		
5	Amministratore	**UCBC_1**		
Flusso di eventi Alternativo: Il sistema non rileva Errori				
5	Sistema	Non fa niente		