Università degli Studi di Salerno
Corso di Ingegneria del Software

Future Forge Gear
System Design Document
Versione 1.2

Data: 27/01/2026

Progetto: Future Forge Gear	Versione: 1.2
Documento: System Design Document	Data: 27/01/2026

Coordinatore del progetto:

Nome	Matricola

Partecipanti:

Nome	Matricola
Francesco Mascolo	0512117352
Francesco Lamanna	0512117166
Luigi Mazza	0512118612

Scritto da:	Francesco Mascolo

Revision History

Data	Versione	Descrizione	Autore
19/11/2025	0.7	Scomposizione del sistema in componenti autonome, fortemente coese e debolmente accoppiate. Definizione dei design goal.	Full Team
21/11/2025	0.8	Component diagram, deployment diagram.	Francesco Lamanna
21/11/2025	0.9	Design Goal e Boundary Conditions	Luigi Mazza
23/11/2025	1.0	Tavola degli accessi e deployment diagram	Francesco Mascolo
25/11/2025	1.1	Modifica dei Servizi dei sottosistemi	Francesco Mascolo & Francesco Lamanna
27/01/2026	1.2	Modifica della gestione dei Dati persistenti e del Mapping Hardware e Software e dei Servizi dei sottosistemi	Francesco Mascolo

Indice

1. Introduzione

1.1. Purpose of the system

Lo scopo del progetto è suddividere il sistema in componenti modulari, progettate per essere il più autonome possibile l'una dall'altra a livello implementativo. La comunicazione tra questi moduli utilizzerà il meccanismo del passaggio per riferimento proprio della programmazione orientata agli oggetti, per assicurare un bassissimo livello di accoppiamento e un'alta coesione interna in ogni componente.
L'architettura sarà di tipo stratificato e chiuso: ogni livello offrirà le proprie funzionalità a quello superiore, basandosi a sua volta sui servizi forniti dai livelli sottostanti.
L'obiettivo è fornire una gestione dati affidabile a basso livello, affiancata da una logica di business efficiente e coerente. Questa logica farà da intermediario per l'interazione tra utente e sistema, con il fine di offrire un'esperienza d'uso gradevole e intuitiva.

1.2 Design goals

1.2.1 Criteri di usabilità

- **Guida visiva**: Ciascuna schermata dovrà presentare messaggi e suggerimenti contestuali per accompagnare l'utente durante le varie operazioni.

  Priorità: Alta.

- **Navigazione strutturata**: La barra di navigazione dovrà essere sempre visibile in tutte le pagine e consentire di raggiungere la homepage o la pagina del carrello con non più di due click.

  Priorità: Alta.

- **Accesso rapido**: Gli utenti loggati devono avere a disposizione un accesso immediato alle aree principali (come il catalogo, il carrello e la cronologia ordini) attraverso collegamenti diretti posizionati nella propria area personale.

  Priorità: Alta.

- **Responsive**: Il sito web deve adattarsi fluidamente a schermi di diverse dimensioni, garantendo una corretta visualizzazione e fruizione da qualsiasi dispositivo.
  Priorità: Alta.

1.2.2 Criteri di affidabilità

- **Disponibilità**: Il sistema deve essere disponibile per almeno il 99.9% del tempo durante l'anno.

  Priorità: Bassa.

- **Robustezza**: Ogni dato inserito dall'utente deve essere sottoposto a validazione sia sul client che sul server, restituendo messaggi di errore chiari e dettagliati per ogni specifica anomalia (es. campo vuoto, formato non valido).

  Priorità: Alta.

- **Sicurezza**: La piattaforma deve assicurare che le risorse, le informazioni sensibili e le funzionalità siano accessibili soltanto da parte di utenti aventi le opportune autorizzazioni.

  Priorità: Alta.

- **Tolleranza agli errori**: In situazioni di malfunzionamento, il sistema deve ripristinare autonomamente le normali condizioni operative o, in alternativa, segnalare l'anomalia senza che si verifichi la perdita di dati critici.

  Priorità: Bassa.

1.2.3 Criteri di prestazione

- **Tempo di risposta**: Il caricamento delle pagine fondamentali (quali catalogo, carrello e procedura di acquisto) non deve superare i due secondi a seguito di un'azione dell'utente.

  Priorità: Bassa.

- **Throughput**: Sotto un carico di lavoro ordinario, il sistema deve supportare almeno 1000 richieste concorrenti.

  Priorità: Bassa.

- **Scalabilità**: In periodi di picco della domanda, la piattaforma deve continuare a operare gestendo un carico di lavoro doppio rispetto al normale, senza che le prestazioni subiscano un calo superiore al 10%.

  Priorità: Bassa.

1.2.4 Criteri di manutenzione

- **Modello Three-Tier**: Il sistema deve avere un'architettura a tre livelli: il tier presentazione, il tier applicazione e il tier dati. Questo modello architetturale assicura che la piattaforma possa essere aggiornata, modificata o ampliata con facilità.

  Priorità: Alta.

- **Estensibilità**: La piattaforma deve essere concepita per permettere l'aggiunta agevole di nuove classi e funzionalità.

  Priorità: Alta.

- **Modificabilità**: Il sistema deve permettere di apportare modifiche semplici alle classi e alle funzionalità esistenti.

  Priorità: Alta.

- **Adattabilità**: La piattaforma deve poter essere adattata senza grandi sforzi a contesti applicativi differenti.

  Priorità: Bassa.

- **Portabilità**: Il sistema deve essere progettato per essere trasferito con facilità su diverse piattaforme tecnologiche.

  Priorità: Bassa.

- **Leggibilità**: La scrittura del codice deve essere chiara e di immediata comprensione durante la lettura.

  Priorità: Media.

1.3. Definitions, acronyms, and abbreviations

- **ORM:** Object-Relational Mapping
- **JavaEE:** Java Platform, Enterprise Edition
- **DAO:** Data Access Object

1.4. References

Lo sviluppo del documento si è basato sul modello di analisi proposto nel Requirements Analysis Document (RAD) Future Forge Gear Versione 1.5

1.5 Overview

Il documento descrive l'architettura proposta, la decomposizione in sottosistemi, la mappatura hardware/software, la gestione dei dati persistenti, il controllo degli accessi e la sicurezza, il controllo globale del software e le condizioni limite.

2. Proposed software architecture

2.1. Overview

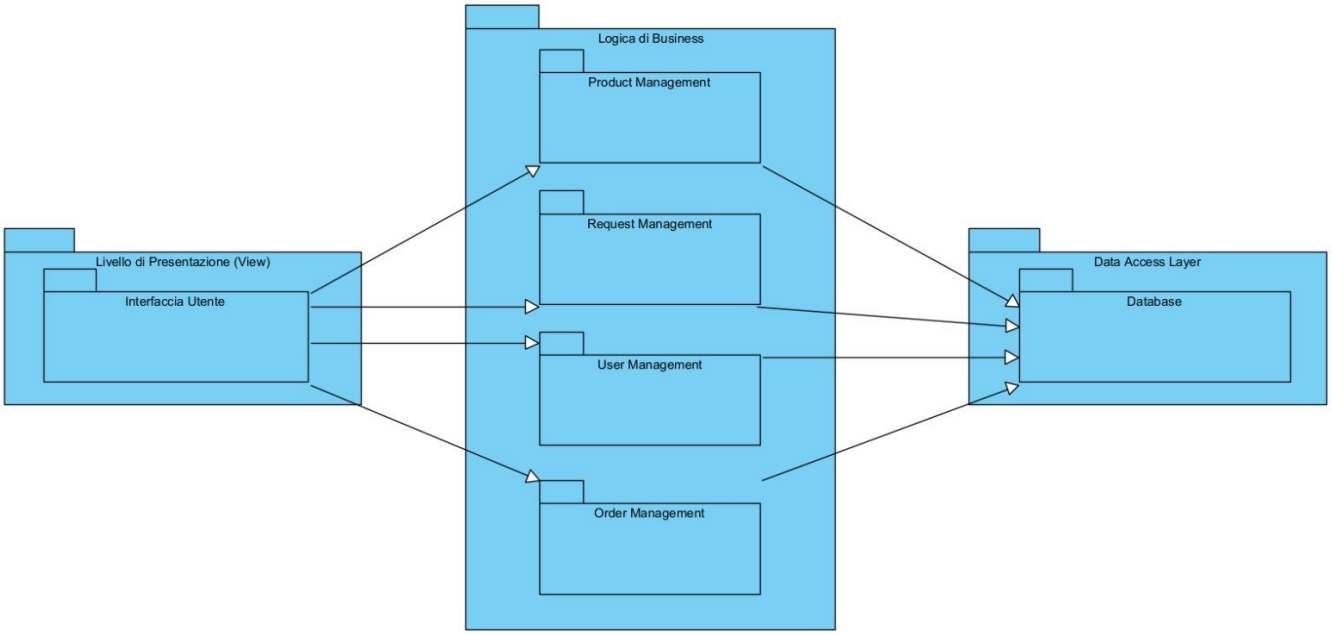
La proposta progettuale prevede l'adozione di un'architettura a tre livelli (three-tier), nella quale il livello di Presentazione (View) gestisce l'interfaccia utente, il livello di Controllo (Control) implementa la logica di business, e il livello Modello (Model) è dedicato alla strutturazione dei dati di dominio.
L'identificazione dei servizi e dei relativi sottosistemi è scaturita da un processo preliminare di Layering, che ha definito un'architettura organizzata su tre strati:

- **Presentation**: responsabile della gestione dell'interazione con l'utente finale.
- **Business**: dedicato all'implementazione della logica applicativa.
- **Data Access Layer**: incaricato della persistenza delle informazioni e della loro manipolazione sul database.

Al processo di stratificazione è stata affiancata una tecnica di partizionamento, che ha individuato 4 moduli funzionali principali:

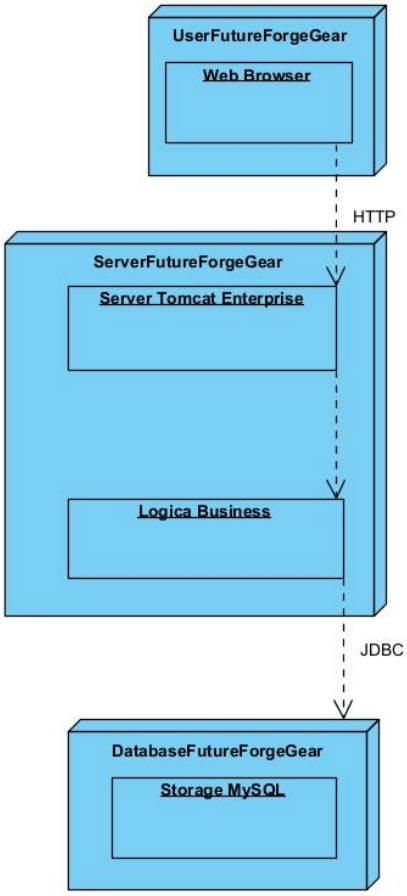
- **la gestione degli utenti** - **la gestione dell'acquisto** – **la gestione della richiesta**- **la gestione del catalogo**.

2.2. Subsystem decomposition



Come anticipato, il sistema è strutturato secondo un'architettura a tre livelli (Three-Tier), che si articola in: il livello di Presentazione (View), il livello di Logica Applicativa (Logica di Business) e il livello di Accesso ai Dati (Data Access Layer). I moduli principali identificati nell'architettura sono i seguenti:

- **Gestione Utenti**: Si occupa di tutte le procedure legate all'account, come la creazione di un nuovo profilo, l'accesso al sistema e il ripristino delle credenziali e la modifica dei dati personali.
- **Gestione Ordine**: Si incarica dell'intero ciclo di vita di un ordine, dalla sua accettazione, alla sua elaborazione, fino alla fase di spedizione e emissione della relativa documentazione fiscale.
- **Gestione Prodotti**: Amministra l'anagrafica degli articoli, tiene sotto controllo le giacenze di magazzino e provvede ad aggiornare i dati associati a ciascun prodotto.
- **Gestione Richiesta**: Gestisce il flusso di invio e ricezione delle comunicazioni inoltrate dagli addetti al magazzino e dai partner commerciali.



2.3. Hardware/software mapping

L'infrastruttura si basa su un modello di distribuzione Client-Server, organizzata su tre nodi fisici distinti. La persistenza dei dati è affidata a un database relazionale MySQL ospitato su un'unità dedicata. (Database Server)
Le comunicazioni tra le componenti avvengono attraverso protocolli standard:

- Il client web interagisce con il server applicativo mediante connessioni http/https
- L'accesso ai dati è gestito via connettività JDBC tra il business layer e il database

Il sistema viene distribuito come archivio enterprise su un'istanza Tomcat (TomEE) configurata su macchina server separata, che orchestra tutte le operazioni applicative e gestisce le connessioni al data layer. (Application Server)

2.4. Persistent data management

Nel sistema, le informazioni che devono essere mantenute persistenti – ovvero i dati da salvare perché necessari a ogni utilizzo della piattaforma – sono:

- Le informazioni relative agli **utenti**, ossia tutti i dati a loro associati, come email, password e indirizzo.
- Le informazioni relative ai **prodotti**, cioè tutti i dettagli che li caratterizzano: nome, descrizione, prezzo e ogni altro attributo rilevante.
- Le informazioni relative agli **ordini**, inclusi data, importo totale, e un codice identificativo univoco (primary key) che ne consenta la distinzione.
- Le informazioni sulle **richieste**, con riferimento alle richieste inviate dagli utenti gestori del magazzino ai fornitori.

La tecnologia di archiviazione scelta è il **database relazionale**, particolarmente adatto a gestire dati strutturati e interconnessi. Questo tipo di database offre il supporto di un linguaggio di interrogazione, come **MySQL**, che consente di eseguire query per recuperare in modo rapido e preciso le informazioni desiderate.

2.5. Access control and security

- **Autenticazione**: Procedura di accesso all'account convalidata in modo sicuro.
- **Autorizzazione**: Sistema di verifica dei permessi fondato su profili utente.
- **Sicurezza dei dati**: Cifratura delle credenziali protezione da attacchi di tipo SQL injection.

La suddivisione delle funzionalità tra i diversi attori del sistema è sintetizzata nelle tabelle seguenti, che associano ciascun ruolo alle operazioni che è autorizzato a svolgere quindi di conseguenza l'accesso a determinati oggetti.

Oggetti Attori	Carrello	Richiesta	Ordine	Prodotto
Cliente	aggiungiProdotto() svuota() visualizza()		Crea() Visualizza()	Visualizza()
Guest	aggiungiProdotto() svuota() visualizza()			Visualizza()
Fornitore		Gestisci()		Visualizza()
Gestore Ordini		Gestisci()	modificaStato() visualizza()	
Magazziniere		Crea()	Visualizza()	aggiungiProdottoAlmagazzino() modificaProdotto() rimuoviProdottoDalMagazzino()

2.6. Global software control

Il sistema FutureForgeGear adotta un modello **web request/response**: ogni operazione è avviata da un'azione dell'utente che genera una richiesta HTTP. Il controllo del flusso è gestito dai **controller Servlet**, che raccolgono input da request e sessione e invocano i **service EJB**. I servizi applicativi applicano le regole di business e accedono ai dati tramite **JPA (EntityManager)** per eseguire operazioni CRUD e query sul database. In base all'esito, il controller seleziona la vista (JSP) da mostrare tramite forward/redirect, mantenendo separati presentazione, logica e persistenza.

2.7. Boundary condition

Le **boundary conditions** relative al funzionamento della piattaforma sono:

- **Installazione::** l'applicazione viene distribuita come WAR su un application server Tomcat EE**,** configurando datasource e persistence unit per il DB MySQL.
- **Avvio:** all'avvio del server vengono inizializzati i componenti applicativi (EJB), la configurazione JPA e la connessione al database MySQL tramite MySQL Connector/J. Completata l'inizializzazione, l'applicazione è accessibile via browser.
- **Manutenzione**: Gli eventuali aggiornamenti e patch vengono applicati in finestre di manutenzione con downtime controllato, seguiti da redeploy/riavvio.
- **Terminazione**: la chiusura del browser o l'inattività portano alla scadenza della sessione; il logout "automatico" dipende dalla gestione sessione. I dati non confermati possono andare persi se non persistiti; nel progetto il carrello può essere salvato a fine sessione (listener di sessione), ma non è garantito in caso di errori/crash.
- **Fallimenti del sistema**: guasti infrastrutturali (server down) sono a carico del provider/gestione infrastruttura; errori applicativi non gestiti generano risposte di errore (es. HTTP 500) e richiedono correzioni software (validazioni, gestione eccezioni, logging).

3 Servizi dei sottosistemi

Di seguito sono descritti i sottosistemi identificati e alcune delle funzionalità da essi fornite:

Gestione Utenti – User Management

Servizi Offerti	
Servizio	Descrizione
Gestione Account	Permette di visualizzare e modificare i dati personali dell'utente
AuthService	Contiene le operazioni per l'accesso al sistema e l'uscita dal sistema
Registrazione	Permette la registrazione di un utente non registrato

Gestione Prodotti – Product Management

Servizi Offerti	
Servizio	Descrizione
Visualizzazione Prodotti	Consente la visualizzazione dei prodotti a catalogo per Cliente e Guest.
Gestione Prodotti	Contiene le operazioni di manutenzione del catalogo prodotto, compreso le modifiche relative ai prodotti stessi(solo per Magazziniere).

Gestione Ordine – Order Management

Servizi Offerti	
Servizio	Descrizione
Gestione degli ordini	Contiene le operazioni per modifica di stato e visualizzazione degli ordini effettuati dal cliente (Solo per il Gestore Ordini)
Creazione e Visualizzazione ordine	Permette al cliente di creare un ordine acquistando prodotti e visualizzarlo per monitorare lo stato

Gestione Richiesta Prodotto – Request Management

Servizi Offerti	
Servizio	Descrizione
Creazione Richieste	Consente ai magazzinieri di inviare richieste e ai fornitori e di crearne nuove. (Solo per i magazzinieri)
Gestione Richieste	Permette ai fornitori di controllare le richieste e di accettarle nel caso possibile

I service sono definiti come interfacce per promuovere i seguenti vantaggi:

1. **Separazione delle responsabilità**: La logica di business non dipende da implementazioni concrete, ma si appoggia esclusivamente sui contratti definiti dalle interfacce.
2. **Facilità di testing**: Durante la fase di test è possibile utilizzare oggetti mock dei service, simulandone il comportamento senza dover interagire con i DAO/Service reali.
3. **Manutenibilità e flessibilità**: L'implementazione sottostante di un service può essere modificata senza richiedere cambiamenti ai controller o ai DAO/Service che ne fanno uso.