

**Università degli Studi di Salerno**

Corso di Ingegneria del Software

# **UNISA-TCG**

## **Object Design Document**



**Data: 28/12/2025**

Progetto: UNISA-TCG	Versione: 1.1
Documento: Object Design Document	Data: 28/12/2025

### Coordinatore del progetto:

Nome	Matricola
Sepe Giuseppe	0512119386

### Partecipanti:

Nome	Matricola
Senatore Francesco	0512120568
Sepe Giuseppe	0512119386
Sullo Diego	0512119455

<b>Scritto da:</b>	Senatore Francesco, Sepe Giuseppe, Sullo Diego
--------------------	--

## Revision History

Data	Versione	Descrizione	Autore
22/12/2025	1.0	Object Design Document	Francesco Senatore, Diego Sullo, Giuseppe Sepe
28/12/2025	1.1	Apportate modifiche legate alla Category Partition	Francesco Senatore, Diego Sullo, Giuseppe Sepe

# Indice

<b>1. Introduzione.....</b>	<b>4</b>
1.1.Object Design Tradeoffs.....	4
1.1.1 Durability vs platform dependance.....	4
1.1.2 Performance versus flexibility.....	4
1.2 Interface documentation guidelines.....	4
1.3 Definitions, acronyms, and abbreviations.....	5
1.4 References.....	7
<b>2. Packages.....</b>	<b>8</b>
<b>2.1 it.unisatcg.model.....</b>	<b>8</b>
<b>2.2 it.unisatcg.control.....</b>	<b>8</b>
<b>2.3 Web Content (Struttura del Presentation Layer).....</b>	<b>8</b>
<b>2.4 Dipendenza tra Package.....</b>	<b>8</b>
<b>3. Class interfaces Glossary.....</b>	<b>9</b>
3.1 Package it.unisatcg.model (Bean).....	9
Classe Prodotto.....	9
Classe Categoria.....	9
Classe Utility: DBConnection.....	10
3.2 Package it.unisatcg.model.dao (DAO).....	11
Interfaccia ProdottoDAO.....	11
Interfaccia CategoriaDAO.....	11
3.3 Package it.unisatcg.control (Servlet).....	12
CategoriaProdottiServlet.....	12
ProductDetailServlet.....	12
ImageServlet.....	12

# 1. Introduzione

Il presente **Object Design Document (ODD)** definisce la progettazione a livello di oggetto della piattaforma e-commerce **UNISA-TCG**. Questo documento funge da ponte tra l'architettura di sistema descritta nel System Design Document (SDD) e l'effettiva implementazione del software. La progettazione si basa rigorosamente sul pattern architetturale **Model-View-Controller (MVC)**, garantendo una netta separazione tra la rappresentazione dei dati, la logica di business e l'interfaccia utente.

Viene descritta la struttura interna dei componenti del sistema, focalizzandosi su come gli oggetti collaborano per fornire le funzionalità di e-commerce (es. gestione del catalogo, carrello e checkout sicuro).soddisfare i requisiti funzionali identificati nel RAD.

## 1.1.Object Design Tradeoffs

Il sistema utilizza lo standard **JDBC** per l'interfacciamento con il DBMS **MySQL**.

- **Vantaggio:** Garantisce un'elevata durabilità e integrità dei dati tramite proprietà **ACID**.
- **Trade-off:** L'uso di query SQL specifiche per MySQL e la configurazione dell'hosting locale (On-Premise) legano parzialmente il software alla piattaforma scelta, ma assicurano il pieno controllo dell'infrastruttura necessario per i backup e il tuning delle performance.

### 1.1.1 Durability vs platform dependance

La scelta di **JDBC** e **MySQL** garantisce un'elevata integrità dei dati (proprietà ACID) al costo di un legame specifico con un sistema di gestione di database relazionali.

### 1.1.2 Performance versus flexibility

Per rispettare il requisito di tempi di risposta inferiori a un secondo (DG\_2), si è scelto di:

- Limitare la complessità dei filtri di ricerca iniziali (Usabilità e Flessibilità).
- Accettare una potenziale **denormalizzazione** dei dati nel database per velocizzare le query di ricerca sul catalogo, riducendo la flessibilità di schemi puramente relazionali in favore della velocità di esecuzione.

## 1.2 Interface documentation guidelines

Oltre alle convenzioni già citate (nomi singolari per le classi, verbi per i metodi), aggiungiamo le seguenti specifiche per il progetto UNISA-TCG:

- **Pattern DAO:** Ogni accesso al database deve avvenire esclusivamente tramite interfacce DAO (es. ProdottoDAO), restituendo oggetti POJO (Plain Old Java Objects).
- **Separazione L1/L2:** Gli oggetti di interfaccia (JSP) non devono contenere logica di business, che è delegata interamente alle Servlet e ai Controller (es. Gestione Carrello Control).

### 1.3 Definitions, acronyms, and abbreviations

Termine/Acronimo	Definizione
<b>ACID</b>	(Atomicity, Consistency, Isolation, Durability). Proprietà fondamentali delle transazioni di database che garantiscono l'integrità dei dati, cruciali per il <b>Checkout</b> (UC3).
<b>CRUD</b>	(Create, Read, Update, Delete). Le quattro operazioni fondamentali di manipolazione dei dati utilizzate dal sottosistema <b>Persistenza Dati</b> e gestite dal <b>Gestore</b> sul Catalogo (UC5).
<b>DAO</b>	<b>Data Access Object</b> . Pattern software (Layer di Astrazione) che gestisce l'accesso ai dati, separando la logica di business dal database (L3).
<b>DBMS</b>	<b>Database Management System</b> . Software utilizzato per gestire e amministrare il database relazionale ( <b>MySQL</b> ).
<b>GDPR</b>	<b>General Data Protection Regulation</b> (Regolamento UE 2016/679). Normativa sulla protezione dei dati personali a cui il sistema deve conformarsi.
<b>HTTPS</b>	<b>Hypertext Transfer Protocol Secure</b> . Protocollo di comunicazione utilizzato per crittografare i dati in transito tra L1 e L2.
<b>JDBC</b>	<b>Java Database Connectivity</b> . API standard Java utilizzata dall'Application Server (L2) per connettersi al Database Server (L3).

<b>L1, L2, L3</b>	I tre livelli dell'architettura (3-Tier): <b>L1</b> (Presentazione), <b>L2</b> (Logica Applicativa), <b>L3</b> (Dati).
<b>PSP</b>	<b>Payment Service Provider.</b> Fornitore di servizi di pagamento esterno (simulato) utilizzato per autorizzare le transazioni finanziarie.
<b>RAD</b>	<b>Requirements Analysis Document.</b> Documento che precede l'SDD e contiene la specifica dei requisiti e i modelli (Classi, Casi d'Uso, Sequenza).
<b>Rollback</b>	Operazione eseguita dal DBMS che annulla tutte le modifiche apportate durante una transazione in caso di fallimento (Gestione Errori 3.7).
<b>SDD</b>	<b>System Design Document.</b> Il documento che specifica l'architettura e la progettazione del sistema.
<b>TCG</b>	<b>Trading Card Game.</b> Il tipo di prodotto gestito dal sistema (es. carte collezionabili).
<b>Tomcat</b>	<b>Apache Tomcat.</b> L'Application Server (container di servlet) utilizzato come ambiente di esecuzione per la logica L2.
<b>UC</b>	<b>Use Case</b> (Caso d'Uso). Sequenza di azioni definita nel RAD (es. UC3: Checkout e Pagamento).

<b>Uptime</b>	Percentuale di tempo in cui il sistema è completamente operativo e disponibile agli utenti.
<b>WAR</b>	<b>Web Archive.</b> Formato di file utilizzato per impacchettare i componenti web dell'applicazione per il deployment su Tomcat.
<b>BLOB</b>	<b>Binary Large Object.</b> Tipo di dato usato nei database per memorizzare grandi quantità di dati binari non strutturati, spesso sono immagini, video, audio o file compressi.

## 1.4 References

- Requirements Analysis Document (RAD) UNISA-TCG.
- System Design Document (SDD) UNISA-TCG.

## 2. Packages

### 2.1 it.unisatcg.model

Questo package costituisce il cuore dei dati del sistema. È l'unico che interagisce direttamente con il database.

- **Bean (Entity):** Classi come **Prodotto.java**, **Utente.java**, **Categoria.java**, **Ordine.java**. Esse rappresentano le tabelle del database sotto forma di oggetti Java (POJO).
- **DAO (Data Access Object):** Interfacce e implementazioni (es. **ProdottoDAO.java**, **ProdottoDAOImp.java**). Contengono la logica SQL (all'interno di un sotto-package "dao").
- **Gestione Connessione:** La classe **DBConnection.java** risiede qui ed è utilizzata dai DAO per ottenere il riferimento al database MySQL.

### 2.2 it.unisatcg.control

Questo package implementa il ruolo di Controller nel pattern MVC. Contiene tutte le Servlet che mediano tra l'utente e il modello.

- Servlet principali: **LoginServlet**, **ProductsServlet**, **CheckoutServlet**, **RegistrationServlet**, ecc.
- Ruolo: Estraggono i parametri dalle richieste HTTP, invocano i metodi dei DAO nel package model e decidono a quale pagina JSP fare il forward dei risultati.

### 2.3 Web Content (Struttura del Presentation Layer)

Rappresenta il **Livello 1 (Vista)**. I file sono organizzati nella root delle risorse web:

- **Pagine JSP:** index.jsp, products.jsp, cart.jsp, ecc. (Generazione dinamica dell'HTML).
- **Asset Statici:** style.css (Style), main.js, ed eventuali ulteriori file javascript o css (Logica client-side).
- **Ruolo:** Visualizzare i dati all'utente e inviare input alle Servlet nel package control.

### 2.4 Dipendenza tra Package

La gerarchia delle dipendenze segue un flusso unidirezionale **top-down** per garantire il disaccoppiamento tra i livelli e prevenire riferimenti circolari:

- **it.unisatcg.control** dipende da **it.unisatcg.model**: le Servlet (Controller) utilizzano le interfacce DAO per l'accesso ai dati e le classi Bean per incapsulare le informazioni da mostrare nelle viste (JSP).
- **it.unisatcg.model (DAO)** dipende dalle classi **Bean** e da **DBConnection**: le implementazioni DAO (DAOImp) utilizzano i Bean per mappare i record del database in oggetti Java e si appoggiano alla classe DBConnection per la gestione della connettività JDBC e del pool di connessioni.
- **it.unisatcg.model (Bean)**: le classi di dominio (es. Prodotto, Utente) sono **indipendenti**. Non hanno dipendenze verso altri pacchetti del sistema, garantendo che la struttura dei dati sia isolata dalla logica di persistenza e di controllo.

### 3. Class interfaces Glossary

In questa sezione vengono dettagliate le interfacce pubbliche delle classi appartenenti allo stralcio del sistema relativo alla **Visualizzazione del catalogo e dei prodotti singoli**. Per ogni classe vengono indicati gli attributi principali, le operazioni disponibili e le eccezioni sollevate.

#### 3.1 Package it.unisatcg.model (Bean)

##### Classe Prodotto

Rappresenta l'entità fondamentale del catalogo (carta collezionabile o accessorio).

- **Attributi:**
  - **private int id:** Identificatore univoco del prodotto.
  - **private String nome:** Nome del prodotto.
  - **private String descrizione:** Descrizione dell'articolo.
  - **private double prezzo:** Prezzo di vendita.
  - **private int quantita:** Numero di pezzi disponibili in magazzino.
  - **private int categoriaId:** Riferimento alla categoria di appartenenza.
  - **private boolean isDisponibile:** Definisce se un prodotto è disponibile o esaurito
  - **private String specifiche:** Descrizione più dettagliata del prodotto
  - **private byte[] foto:** BLOB contenente l'immagine del prodotto.
- **Metodi:**
  - **Getter/Setter:** Metodi standard per l'accesso e la modifica degli attributi sopra elencati.

##### Classe Categoria

Definisce la tipologia del prodotto (Es. Pokemon, Accessori, Magic).

- **Attributi:**
  - **private int id:** Identificatore della categoria.
  - **private String nome:** Nome della categoria.
- **Metodi:**
  - **Getter/Setter:** Metodi standard per la gestione dell'entità.

## Classe Utility: DBConnection

Responsabile della gestione della connettività con il **DBMS MySQL**, utilizza il driver **JDBC** per stabilire una comunicazione sicura e centralizzata.

- **Attributi:**
  - **URL:** Stringa di connessione al JDBC (**jdbc:mysql://localhost:3306/unisa\_cardshop** nel nostro caso)
  - **USER:** Nome utente del database (es. **root**)
  - **PASSWORD:** Password per l'accesso.
- **Metodi:**
  - **public static Connection getConnection() throws SQLException:** Carica dinamicamente il driver **com.mysql.cj.jdbc.Driver** (se non è già stato caricato) e restituisce un oggetto **Connection** attivo. Questo verrà usato dalle varie implementazioni **DAO** per eseguire **query SQL**.

## 3.2 Package it.unisatcg.model.dao (DAO)

### Interfaccia ProdottoDAO

Definisce il contratto per le operazioni di persistenza dei prodotti nel DBMS MySQL.

Metodo	Descrizione
<b>void doSave(Prodotto prodotto)</b>	Memorizza un nuovo prodotto nel database.
<b>List&lt;Prodotto&gt; doRetrieveByCategoria(int categoriaId)</b>	<b>Ricerca filtrata:</b> Recupera tutti i prodotti associati a una specifica categoria.
<b>List&lt;Prodotto&gt; doRetrieveByCategoriaDisp(int categoriaId, String disp)</b>	<b>Ricerca filtrata:</b> Recupera tutti i prodotti che combaciano con una data categoria e una data disponibilità
<b>Prodotto doRetrieveByKey(int id)</b>	Recupera un singolo prodotto tramite il suo identificativo univoco
<b>List&lt;Prodotto&gt; doRetrieveAll()</b>	Recupera l'elenco completo dei prodotti presenti nel catalogo.
<b>List&lt;Prodotto&gt; doRetrieveAllDisp(String disp)</b>	<b>Ricerca filtrata:</b> Recupera l'elenco completo dei prodotti che combaciano con la disponibilità ricevuta come input.
<b>void doUpdate(Prodotto prodotto)</b>	Aggiorna le informazioni (nome, prezzo, ecc.) di un prodotto esistente.
<b>void doUpdateQuantita(int id, int qta, Connection c)</b>	Aggiorna la disponibilità a magazzino (usato durante la transazione di checkout).
<b>void doDelete(int id)</b>	Rimuove un prodotto dal catalogo tramite il suo ID.

### Interfaccia CategoriaDAO

Gestisce la persistenza delle categorie necessarie per la navigazione del catalogo.

Metodo	Descrizione
<b>void doSave(Categoria categoria)</b>	Inserisce una nuova categoria nel sistema.
<b>Categoria doRetrieveByKey( int id)</b>	Recupera i dettagli di una categoria (es. "pokemon") tramite ID.
<b>List&lt;Categoria&gt;doRetrieveAll()</b>	Recupera tutte le categorie per popolare i filtri della pagina products.jsp
<b>void doUpdate(Categoria categoria)</b>	Modifica il nome di una categoria esistente.
<b>void doDelete(int id)</b>	Elimina una categoria dal database.

### 3.3 Package it.unisatcg.control (Servlet)

#### CategoriaProdottiServlet

Si occupa di gestire la visualizzazione principale del catalogo e il filtraggio per categoria e disponibilità.

- **Input:** La servlet può ricevere dei parametri **categoriaId** e **disp** per filtrare i risultati.
- **Logica:**
  1. Utilizza **CategoriaDAO** per recuperare l'elenco completo delle categorie necessarie per popolare il menù di filtraggio nella sezione prodotti
  2. Se i parametri **categoriaId** e **disp** sono validi interroga **ProdottoDAO.doRetrieveByCategoriaDisp** per ottenere i prodotti pertinenti
  3. Se i parametri sono assenti, invoca **prodottoDAO.doRetrieveAll** per mostrare l'intero catalogo non filtrato.
- **Metodi:**
  - **protected void doGet(HttpServletRequest request, HttpServletResponse response):** prepara gli attributi **products**, **categorie**, **categoriaSelezionata** e **disponibilitasel** ed effettua il forward verso **products.jsp**.

#### ProductDetailServlet

Gestisce la visualizzazione della scheda tecnica dettagliata di un singolo articolo.

- **Input:** Riceve obbligatoriamente il parametro id del prodotto tramite il metodo **GET**.
- **Logica:** Recupera i dati chiamando **ProdottoDAO.doRetrieveByKey**. Se il prodotto esiste, carica i dettagli (nome, desc, prezzo etc.) nell'oggetto richiesta, in caso contrario avviene un errore.
- **Metodi:**
  - **protected void doGet(HttpServletRequest request, HttpServletResponse response):** Gestisce il recupero dell'entità **Prodotto** ed esegue un forward verso **product\_detail.jsp**.

#### ImageServlet

Servlet dedicata alla gestione e il caricamento dinamico delle immagini salvate nel DB.

- **Input:** riceve il parametro **id** del prodotto di cui si vuole visualizzare l'immagine.
- **Logica:** Si connette al DB tramite **DBConnection**, recupera la foto in formato BLOB dalla tabella prodotto e lo converte in uno stream binario in uscita. Imposta il **contentType** della risposta su **image/jpeg**.
- **Metodi:**
  - **protected void doGet(HttpServletRequest request, HttpServletResponse response):** Apre lo stream verso il DB e scrive i byte dell'immagine direttamente nell'oggetto **OutputStream** della risposta HTTP.

