

**Università degli Studi di Salerno**

Corso di Ingegneria del Software

**UNISA-TCG**

**Test Plan**



**Data: 28/12/2025**

Progetto: UNISA-TCG	Versione: 1.1
Documento: Test Plan	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	Test Plan	Francesco Senatore, Diego Sullo, Giuseppe Sepe
28/12/2025	1.1	Cambio della tecnica della specifica dei casi di test (Category Partition)	Francesco Senatore, Diego Sullo, Giuseppe Sepe

# Indice

<b>UNISA-TCG.....</b>	<b>1</b>
<b>Test Plan.....</b>	<b>1</b>
<b>1. Introduzione.....</b>	<b>4</b>
1.1 Obiettivi del test.....	4
1.2 Scope del test.....	4
1.3 Analisi del sistema.....	4
<b>2. Approccio.....</b>	<b>5</b>
2.1 Vincoli.....	5
2.2 Copertura.....	5
2.3 Test Tools.....	5
2.5 Test Data.....	5
<b>3. Test Plan.....</b>	<b>6</b>
3.1 Test Team.....	6
3.2 Major Task.....	6
3.3 Ambiente di Test.....	6
<b>4. Feature da Testare.....</b>	<b>7</b>
<b>5. Procedure di Testing.....</b>	<b>7</b>
5.1 Definizione di Categorie, Choices e Vincoli.....	7
5.2 Test Case: Filtri di Ricerca Incrociati.....	8
5.3 Test di Visualizzazione Prodotto (Deep Dive).....	8
5.4 Test di "Nessun Risultato" (Robustezza).....	9
<b>6. Rischi e contingenze.....</b>	<b>9</b>

# 1. Introduzione

Il presente documento descrive la strategia di testing per la piattaforma UNISA-TCG. L'obiettivo è validare le funzionalità core di navigazione e ricerca per garantire un'esperienza utente fluida e priva di errori.

## 1.1 Obiettivi del test

- Garantire che la visualizzazione dei prodotti sia coerente con i dati presenti nel database.
- Verificare l'accuratezza della ricerca tramite filtri.
- Assicurarsi che la navigazione tra catalogo e pagina prodotto sia priva di interruzioni.

## 1.2 Scope del test

Lo scope è limitato al catalogo, attraverso le classi Categoria, DBConnection e prodotto specificamente:

- Caricamento della lista prodotti.
- Visualizzazione dettagliata del singolo prodotto.
- Funzionamento dei filtri (Disponibilità, Tipo).

## 1.3 Analisi del sistema

Il sistema è un'applicazione web basata su architettura Model-View-Controller (MVC). Il front-end comunica con un database relazionale per recuperare le informazioni sui prodotti e le relative immagini.

## 2. Approccio

L'approccio sarà di tipo **Black Box** per i test funzionali (testando l'interfaccia come un utente finale) e **White Box** per la validazione delle query di ricerca a livello di codice.

### 2.1 Vincoli

- I test devono essere completati entro la data di consegna del progetto.
- Limitazione dei dati: i test verranno eseguiti su un set di dati controllato (popolamento DB di test).

### 2.2 Copertura

La copertura dei test sarà garantita tramite il metodo **Category Partition**. Questa tecnica sistematica permette di:

- Identificare i parametri di input (Categorie) e le loro partizioni significative (Choices).
- Definire vincoli formali tra le scelte per eliminare combinazioni impossibili o ridondanti.
- Garantire che ogni scelta valida sia testata almeno una volta e che i casi di errore siano isolati.

### 2.3 Test Tools

- **Browser (Chrome/Firefox):** Per l'esecuzione manuale o automatizzata dei test.
- **Selenium IDE / Katalon Recorder:** Per registrare e riprodurre le azioni dell'utente sul browser.
- **Fogli di calcolo (Excel/Google Sheets):** Per la tracciabilità della matrice dei casi di test e dei risultati.

### 2.4 Test Type

- **Functional Testing:** Verifica che i filtri mostrino i prodotti corretti.
- **Boundary Value Analysis (Integrata):** Test sui limiti dei filtri (es. prezzo minimo e massimo).

### 2.5 Test Data

Un set di prodotti pre-caricati nel sistema con caratteristiche note (nomi, rarità, prezzi) per confrontare l'output del sistema con il risultato atteso.

## 3. Test Plan

### 3.1 Test Team

- **Responsabile Testing:** Giuseppe Sepe
- **Sviluppatori:** Francesco Senatore, Diego Sullo, Giuseppe Sepe

### 3.2 Major Task

1. Predisposizione dell'ambiente di test e del DB.
2. Scrittura dei casi di test per filtri e visualizzazione.
3. Esecuzione dei test manuali e automatizzati.
4. Reportistica dei bug e ri-test dopo il bug-fixing.

### 3.3 Ambiente di Test

- **Hardware:** PC con architettura x64.
- **Software:** Server locale (Apache/Tomcat), Browser (Chrome, Firefox).
- **DB:** MySQL (Xampp).

## 4. Feature da Testare

- **Visualizzazione Catalogo:** Visualizzazione corretta della griglia prodotti.
- **Visualizzazione Prodotto:** Correttezza dei dati tecnici nella pagina singola.
- **Ricerca e Filtri:** Precisione dei risultati filtrati.

## 5. Procedure di Testing

In questa fase, il sistema viene sollecitato simulando le azioni di un utente reale che naviga sulla piattaforma UNISA-TCG. Al fine di massimizzare l'efficienza e la copertura del test, utilizzeremo casi di test generati sistematicamente combinando le diverse **categorie** (parametri di input) e le relative **choice** (partizioni di valori) individuate tramite il metodo **Category Partition**.

### 5.1 Definizione di Categorie, Choices e Vincoli

Abbiamo scomposto il dominio degli input per la ricerca e visualizzazione nel catalogo:

Categoria	Choice (Partizione)	Vincoli / Proprietà
<b>Brand (ID Categoria)</b>	Pokémon (G1), Magic (G2), Yu-Gi-Oh! (G3), Non specificato	[Property BrandSet]
<b>Disponibilità</b>	Disponibile, Esaurito, Tutto	-

## 5.2 Test Case: Filtri di Ricerca Incrociati

Utilizzando la partizione sopra definita, sono stati generati i seguenti test frame:

ID	Combinazione Choice (Input)	Partizioni (Categorie) Coperte	Risultato Atteso (Output)
TC_CP_01	<b>Brand:</b> [Pokémon]  <b>Disponibilità:</b> [Disponibile]	<b>Brand</b>  <b>Disponibilità</b>	Visualizzazione di prodotti del brand Pokémon disponibili.
TC_CP_02	<b>Brand:</b> [Magic]  <b>Disponibilità:</b> [Esaurito]	<b>Brand</b>  <b>Disponibilità</b>	Visualizzazione di del brand Magic The Gathering esauriti..
TC_CP_03	<b>Brand:</b> [Yu-Gi-Oh!]  <b>Disponibilità:</b> [Tutto]	<b>Brand</b>  <b>Disponibilità</b>	Visualizzazione di del brand Yu-Gi-Oh! disponibile e non.

## 5.3 Test di Visualizzazione Prodotto (Deep Dive)

In questo scenario, testiamo se il sistema mostra correttamente i dati tecnici (metadati) recuperati dal database quando l'utente seleziona un prodotto specifico dal catalogo. Questo test verifica che la mappatura tra l'oggetto Prodotto (Bean) e la vista product\_detail.jsp sia corretta.

- **Scenario:** L'utente seleziona come filtri Pokémon e Esaurito.
- **Choices Verificate:** Brand: [Pokémon], Disponibilità:[Esaurito].
- **Dati attesi in uscita (attributi del Bean Prodotto):**
  1. **Immagine:** Visualizzazione del BLOB foto come stream binario.
  2. **Descrizione:** Corrispondenza con l'attributo descrizione (es. 36 pack).
  3. **Disponibilità:** Etichetta basata sul valore isDisponibile.
  4. **Prezzo:** Valore dell'attributo prezzo comprensivo di IVA.



---

## 5.4 Test di "Nessun Risultato" (Robustezza)

Fondamentale per la robustezza del metodo **Category Partition** è capire come il sistema gestisce combinazioni di **Choice** che non producono match nel database. In questo caso, si testa il comportamento della `ProdottiServlet` quando la query prodotta dal `ProdottoDAO` restituisce una lista vuota.

- **Azione:** Applicare una combinazione di filtri che non ha corrispondenze (es. Categoria Brand: [Yu-Gi-Oh!] + Categoria Disponibilit : [Esaurito]).
- **Risultato Atteso:** La pagina non deve generare errori di runtime (es. `NullPointerException`); non deve apparire alcun prodotto e deve essere mostrato un messaggio di cortesia all'utente.

## 6. Rischi e contingenze

- **Rischio:** Mancato caricamento delle immagini delle carte da server esterni.
- **Contingenza:** Utilizzo di immagini segnaposto (placeholder) locali per i test di layout.
- **Rischio:** Prestazioni lente della ricerca con molti utenti.
- **Contingenza:** Ottimizzazione degli indici del database.