Università degli Studi di Salerno

Corso di Ingegneria del Software

UNISA-TCG

Test Plan

Data: 22/12/2025

Coordinatore del progetto:

Nome	Matricola
Sepe Giuseppe	0512119386

Partecipanti:

Nome	Matricola
Senatore Francesco	0512120568
Sepe Giuseppe	0512119386
Sullo Diego	0512119455

Scritto da:	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**
27/12/2025	**1.1**	**Cambio della tecnica della specifica dei casi di test (Category Partition)**	**Francesco Senatore, Diego Sullo, Giuseppe Sepe**

Indice

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 (Rarità, Set, Prezzo).

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 su **UNISA-TCG**. Utilizzeremo casi di test che combinano diverse classi di input per massimizzare l'efficienza.

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**
Brand (ID Categoria)	Pokémon (G1), Magic (G2), Yu-Gi-Oh! (G3), Non Specificato	[Property BrandSet]
Tipo Prodotto	Booster Box (T1), Deck Box (T2), Accessorio (T3)	[if BrandSet]
Disponibilità	In Stock, Esaurito	-

5.2 Test Case: Filtri di Ricerca Incrociati

Applichiamo la strategia **Weak Equivalence** per coprire ogni classe almeno una volta con il minor numero di test.

ID	Azione Utente (Input)	Classi Coperte	Risultato Atteso (Output)
TC_01	Filtra: **Pokémon** + **Booster Box**	G1,T1	Visualizzazione di box sigillati Pokémon (es. *Set Base*).

TC_02	Filtra: **Magic** + **Deck Box**	G2,T2	Visualizzazione di porta-mazzi a tema Magic.
TC_03	Filtra: **Yu-Gi-Oh!** + **Accessorio**	G3,T3	Visualizzazione di bustine protettive (sleeves) Yu-Gi-Oh!.

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 clicca su un prodotto "Booster Box Pokémon Evoluzioni".
- **Choices Verificate**: Brand: [Pokémon], Tipo: [Booster Box].
- **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 booleano 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 Tipo: [Booster Box]).
- **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.