

**E-Cycle**  
**Object Design Document**  
**Versione 1.0**



Data: 16/12/2024

## Coordinatore del progetto

Nome	Matricola
Sinicario Gennaro	05121-16134

## Partecipanti

Nome	Matricola
Gragnaniello Francesco	05121-16465
Margio Antonio	05121-16137
Sinicario Gennaro	05121-16134

<b>Scritto da</b>	Gragnaniello Francesco, Margio Antonio, Sinicario Gennaro
-------------------	---

## Revision History

Data	Versione	Descrizione	Autore
16/12/2024	1.0	Prima Stesura Object Design Document	Gragnaniello Francesco, Margio Antonio, Sinicario Gennaro

# Indice

<b>1. Introduzione</b>	<b>4</b>
1.1 Compromessi nella progettazione degli oggetti	4
1.2 Linee guida per la documentazione dell'interfaccia	4
1.3 Definizioni, acronimi e abbreviazioni	5
<b>2. Pacchetti</b>	<b>5</b>
2.1 Pacchetto control	5
2.2 Pacchetto model	5
<b>3. Glossario delle interfacce di classe</b>	<b>6</b>

# 1. Introduzione

## 1.1 Compromessi nella progettazione degli oggetti

La progettazione della piattaforma e-cycle ha richiesto un'analisi approfondita dei compromessi tra diverse soluzioni tecniche e operative. I principali compromessi affrontati includono:

- **Acquisto vs. Build:** Si è deciso di utilizzare una combinazione di soluzioni preesistenti e componenti personalizzati per bilanciare costi, tempi di sviluppo e flessibilità della piattaforma.
- **Spazio di memoria vs. Tempo di risposta:** Abbiamo scelto di ottimizzare il tempo di risposta per garantire un'esperienza utente fluida, accettando un leggero incremento nell'utilizzo di memoria per cache e indicizzazioni.
- **Sicurezza vs. Usabilità:** Implementazioni di sicurezza avanzate come l'hash della password nel database sono state integrate. Questi compromessi sono stati documentati per garantire trasparenza e coerenza nelle decisioni progettuali.

## 1.2 Linee guida per la documentazione dell'interfaccia

Le linee guida seguenti mirano a migliorare la comunicazione tra sviluppatori durante la progettazione degli oggetti:

- **Nomenclatura:**
  - Le classi sono denominate con **nomi singolari** che riflettono chiaramente la loro responsabilità (es. PagaOra, Elimina Prodotto).
  - I campi e i parametri utilizzano **frasi nominali** che identificano chiaramente i dati che rappresentano (es. prezzo, email,password).
- **Gestione degli errori:**
  - Lo stato di errore viene restituito tramite un'**eccezione**, non tramite un valore di ritorno, per garantire una gestione strutturata degli errori.
- **Raccolte e contenitori:**
  - Le classi che rappresentano raccolte implementano un metodo che restituisce un'enumerazione degli elementi.
  - Le enumerazioni restituite sono **resistenti alle rimozioni** di elementi per evitare comportamenti inattesi durante l'iterazione.
- **Gestione dei casi limite:**
  - Ogni interfaccia considera esplicitamente i casi limite, come input nulli, raccolte vuote e valori fuori intervallo.

Stabilire queste regole prima della progettazione consente a tutti gli sviluppatori di contribuire in modo coerente e di seguire un approccio uniforme.

## 1.3 Definizioni, acronimi e abbreviazioni

- **UI (User Interface):** Interfaccia Utente.
- **UX (User Experience):** Esperienza Utente.
- **ID :** Identificativo unico per i prodotti disponibili sulla piattaforma.

## 2. Pacchetti

### 2.1 Pacchetto control

**Panoramica:** Il pacchetto **control** contiene tutte le classi e i metodi responsabili della gestione della logica applicativa. Esso si occupa di coordinare le interazioni tra il modello (model) e la vista (view).

**Dipendenze:**

- Dipende dal pacchetto **model** per accedere ai dati e gestire le operazioni legate al dominio applicativo.

**Utilizzo previsto:**

- Implementare i controller per gestire eventi e azioni degli utenti.
- Comunicare con il pacchetto **model** per aggiornare o recuperare dati.

### 2.2 Pacchetto model

**Panoramica:** Il pacchetto **model** contiene tutte le classi che rappresentano la logica di dominio e i dati dell'applicazione. Si occupa della gestione e persistenza dei dati relativi agli oggetti elettronici ricondizionati.

**Dipendenze:**

- Indipendente da altri pacchetti. Fornisce un'API che il pacchetto **control** utilizza.

**Utilizzo previsto:**

- Modellare gli oggetti del dominio come prodotti, utenti e ordini.
- Fornire metodi per la gestione di operazioni di persistenza dei dati.

### 3. Glossario delle interfacce di classe

#### Utente

##### Attributi:

- **ID:** Identificatore univoco per l'utente.
- **Nome:** Nome dell'utente.
- **Cognome:** Cognome dell'utente.
- **Email:** Indirizzo email per contatto e accesso.
- **Password:** Password criptata per l'autenticazione.
- **Ruolo:** Specifica se l'utente è un Cliente o un Amministratore.

##### Metodi Pubblici:

- **login():** Autentica l'utente in base alle credenziali.
  - **Precondizioni:** Email e password devono essere validi.
  - **Postcondizioni:** Se l'autenticazione ha successo, lo stato della sessione viene aggiornato.
- **logout():** Termina la sessione dell'utente.
  - **Postcondizioni:** La sessione dell'utente viene invalidata.

#### Cliente (eredita da Utente)

##### Attributi:

- Eredita tutti gli attributi da Utente.

##### Metodi Pubblici:

- **visualizzaCatalogo():** Mostra l'elenco dei prodotti disponibili.
  - **Postcondizioni:** Viene restituita una lista di oggetti *Prodotto* disponibili.
- **confermaOrdine():** Conferma un ordine selezionato.
  - **Precondizioni:** Deve essere selezionato un prodotto disponibile.
  - **Postcondizioni:** L'ordine viene creato e lo stato aggiornato a "Confermato".
- **visualizzaStoricoOrdini():** Mostra lo storico degli ordini del cliente.
  - **Postcondizioni:** Viene restituita una lista di ordini effettuati.

## Amministratore (eredita da Utente)

### Attributi:

- Eredita tutti gli attributi da Utente.

### Metodi Pubblici:

- **aggiungiProdotto():** Aggiunge un nuovo prodotto al catalogo.
  - **Precondizioni:** L'amministratore deve essere autenticato e avere i permessi adeguati.
  - **Postcondizioni:** Il prodotto viene aggiunto al sistema.
- **modificaProdotto():** Modifica i dettagli di un prodotto esistente.
  - **Precondizioni:** Il prodotto deve esistere nel catalogo.
  - **Postcondizioni:** I dettagli del prodotto vengono aggiornati.
- **eliminaProdotto():** Rimuove un prodotto dal catalogo.
  - **Precondizioni:** Il prodotto deve esistere nel catalogo.
  - **Postcondizioni:** Il prodotto viene rimosso dal catalogo.
- **approvaOrdine():** Approva gli ordini dei clienti.
  - **Precondizioni:** L'ordine deve essere nello stato "In attesa".
  - **Postcondizioni:** Lo stato dell'ordine viene aggiornato a "Approvato".

## Prodotto

### Attributi:

- **ID:** Identificatore univoco per il prodotto.
- **Tipo:** Categoria del prodotto (es. Bicicletta, Accessorio).
- **Marca:** Marca del prodotto.
- **Modello:** Nome del modello.
- **Prezzo:** Prezzo del prodotto.
- **Descrizione:** Descrizione dettagliata, incluse le specifiche tecniche.

### Metodi Pubblici:

- **visualizzaDettagli():** Mostra informazioni dettagliate sul prodotto.
  - **Postcondizioni:** Vengono restituiti i dettagli completi del prodotto.

## Ordine

### Attributi:

- **IDOrdine:** Identificatore univoco per l'ordine.
- **Data:** Data dell'ordine.
- **Stato:** Stato corrente (es. Confermato, In Attesa).

- **Cliente:** Riferimento al cliente che ha effettuato l'ordine.
- **Prodotto:** Riferimento al prodotto associato all'ordine.

#### Metodi Pubblici:

- **confermaOrdine():** Finalizza l'ordine.
  - **Precondizioni:** Lo stato deve essere "In Attesa" e il prodotto deve essere disponibile.
  - **Postcondizioni:** Lo stato viene aggiornato a "Confermato".
- **annullaOrdine():** Annulla l'ordine se non ancora approvato.
  - **Precondizioni:** Lo stato dell'ordine deve essere "In Attesa".
  - **Postcondizioni:** Lo stato dell'ordine viene aggiornato a "Annullato".

## Dipendenze tra Classi

- **Utente** è una superclasse con **Cliente** e **Amministratore** come sottoclassi specializzate.
- **Cliente** è associato a più oggetti **Ordine** (relazione 1:N).
- **Prodotto** è referenziato da **Ordine** in una relazione 1:1.
- **CatalogoProdotti** aggrega più oggetti **Prodotto** e fornisce metodi per la ricerca e il recupero.

## Contratti delle Classi

- **Utente:**
  - **Invariante:** Ogni utente deve avere un ID univoco e un'email valida.
- **Cliente:**
  - **Invariante:** Un cliente deve avere uno storico ordini consistente con i dati nel sistema.
- **Amministratore:**
  - **Invariante:** Un amministratore deve avere privilegi elevati per gestire il catalogo e gli ordini.
- **Prodotto:**
  - **Invariante:** Ogni prodotto deve avere un prezzo maggiore di zero e un modello valido.
- **Ordine:**
  - **Invariante:** Ogni ordine deve essere associato a un cliente e a un prodotto valido.

## Gestione delle Eccezioni

- **Errori di Login:**
  - Sollevati quando vengono fornite credenziali errate.



- **Accesso Non Autorizzato:**
  - Attivato quando un utente tenta azioni oltre le autorizzazioni del proprio ruolo.
- **Errori di Validazione dei Dati:**
  - Garantisce che tutti i campi obbligatori siano compilati.
  - Gestisce le inconsistenze, come il tentativo di ordinare prodotti non disponibili.
- **Eccezioni sugli Ordini:**
  - Impedisce conferme doppie o modifiche agli ordini già finalizzati.