**Manuale Tecnico progetto TheKnife**

* **Autori**

Artur Mukhin

De Giorgi Filippo

Magrin Nicolò

Caredda Anna Eleonora

* **Data =** 23/06/2025
* **Versione Documento=** v1.1

**Indice**

1. **Introduzione**

**1.1.** Scopo del Documento

**1.2.** Contesto del Progetto

1. **Architettura dell'Applicazione**

**2.1.** Panoramica Generale e Struttura dei Package

**2.2.** Descrizione dei Layer

**2.3.** Scelte Architetturali

1. **Strutture Dati e Gestione dei File**

**3.1.** Entità del Dominio

**3.2.** Gestione della Persistenza

**3.3.** Formato dei File Dati

1. **Dettaglio delle Classi Significative**

**4.1.** classe TheKnife

**4.2.** menu.GestioneMenu

**4.3.** repository.RistoranteService

**4.4.** gestioneFile.FileRistorante

**4.5.** entita.Ristorante

**4.6.** eccezioni.LocaleGiaPresenteException

1. **Funzionalità e Scelte Algoritmiche**

**5.1.** Flusso di Esecuzione Principale

**5.2.** Ricerca Ristoranti

**5.3.** Gestione Recensioni

1. **Limiti della Soluzione**
2. **JavaDoc**

**1. Introduzione**

**1.1. Scopo del Documento**

Questo manuale tecnico fornisce una descrizione dettagliata della struttura interna, delle scelte di progettazione e delle implementazioni tecniche dell'applicazione "The Knife". È destinato a un pubblico di sviluppatori e tecnici che necessitano di comprendere il funzionamento del software, la sua architettura e le logiche implementative.

**1.2. Contesto del Progetto**

"The Knife" è una piattaforma software che simula le funzionalità di base di un'applicazione per la ricerca e la recensione di ristoranti. Il progetto consente agli utenti di cercare locali, inserire recensioni e gestire una lista di preferiti, e ai ristoratori di aggiungere i propri ristoranti e rispondere alle recensioni ricevute.

**2. Architettura dell'Applicazione**

**2.1. Panoramica Generale e Struttura dei Package**

L'architettura del software è stata progettata per essere modulare e stratificata, separando le responsabilità in diversi package, come richiesto dalle specifiche.

* **theknife**: Contiene la classe TheKnife, punto di ingresso dell'applicazione.
* **menu**: Gestisce l'interfaccia utente da riga di comando. Le classi \*UI e GestioneMenu orchestrano l'interazione con l'utente e invocano i servizi appropriati.
* **repository**: Implementa il layer di business logic. Le classi Service fungono da intermediari tra l'interfaccia utente e il layer di accesso ai dati.
* **gestioneFile**: Si occupa della persistenza dei dati, gestendo la lettura e la scrittura degli oggetti da e verso i file di testo.
* **entità**: Modella gli oggetti del dominio del problema (es. Utente, Ristorante, Recensione).
* **eccezioni**: Contiene le eccezioni personalizzate per la gestione di errori specifici dell'applicazione.

**2.2. Descrizione dei Layer**

L'applicazione segue un'architettura a più livelli (multi-layer) per garantire un'elevata coesione e un basso accoppiamento tra i componenti:

1. **Layer di Presentazione (UI)**: Costituito dai package menu e theknife. È responsabile di mostrare le informazioni all'utente e di catturare i suoi input. Non contiene logica di business, ma delega ogni operazione al layer di servizio.
2. **Layer di Servizio (Business Logic)**: Il package repository. Contiene la logica di business dell'applicazione (es. come cercare un ristorante, come registrare un utente). Agisce da facciata per il layer di accesso ai dati.
3. **Layer di Accesso ai Dati (Persistence)**: Il package gestioneFile. Fornisce un'astrazione sull'archiviazione fisica dei dati, nascondendo i dettagli di come gli oggetti vengono letti e scritti su file.
4. **Layer di Dominio (Entities)**: Il package entita. Rappresenta il cuore del software, contenendo le classi che modellano i dati e le loro relazioni.

**2.3. Scelte Architetturali**

* **Service-Oriented**: L'uso di classi "Service" permette di centralizzare la logica di business, rendendo il codice della UI più snello e focalizzato sulla presentazione. Questa scelta promuove la riusabilità del codice e facilita i test.
* **Astrazione della Persistenza**: Il layer gestioneFile astrae completamente il modo in cui i dati vengono salvati. Questo significa che in futuro si potrebbe sostituire la gestione tramite file con un database senza dover modificare i layer di servizio e di presentazione.
* **Ereditarietà**: Le classi Cliente e Gestore ereditano dalla classe Utente, condividendo attributi comuni come username e password ma differenziandosi per il ruolo. Questa scelta riduce la duplicazione del codice.