Manuale Tecnico – The-Knife

Frontespizio

Titolo: Manuale Tecnico – The-Knife
Autori: Filippo Molteni, Simone Marcarini, Enea Giana
Versione documento: 0.2

Indice

1. Struttura dell'applicazione

Il progetto *The-Knife* non adotta una divisione in pacchetti convenzionali (es. model, controller, view), ma si basa su una struttura semplice e lineare.

- **App.java**
  È il punto di ingresso dell'applicazione. Contiene il metodo main() e invoca il Manager per avviare l'intero flusso logico.
- **Manager.java**
  È la classe centrale che gestisce la logica dell'applicazione. Si occupa del caricamento dei dati, della gestione dell'interazione utente e del coordinamento tra le strutture dati e i file CSV.
- **FileManager.java**
  Si occupa della lettura e scrittura dei dati da e verso i file CSV. Centralizza la gestione delle I/O tramite BufferedReader e BufferedWriter.

2. Scelte architetturali

L'applicazione non segue un pattern architetturale formale come MVC o layered architecture. È progettata come una **applicazione monolitica semplice**, adatta a un prototipo CLI con file system locale.

Non esistono moduli indipendenti; i file sorgente sono organizzati principalmente per comodità visuale.

3. Strutture dati utilizzate

Collezioni

La struttura dati principale utilizzata è:

- **ArrayList<T>** – per la gestione dinamica di liste di ristoranti, utenti, recensioni, ecc.

Classi Custom

Le classi definite sono:

- Utente: rappresenta un utente generico, contiene credenziali e identificatori comuni.
- Cliente: estende Utente, con metodi per gestire recensioni e preferiti.
- Ristoratore: estende Utente, contiene funzionalità per gestire i propri ristoranti.
- Ristorante: contiene informazioni come nome, fascia di prezzo, posizione, disponibilità, premi.
- Recensione: rappresenta una recensione testuale con valutazione numerica.
- Preferito: associazione tra Cliente e Ristorante.
- FasciaPrezzo: enumerazione che rappresenta la fascia di prezzo (es. ECONOMICO, MEDIO, ALTO).

4. Scelte algoritmiche

L'applicazione implementa algoritmi di base per la gestione delle liste, tra cui:

- **Filtri** per tipologia, fascia di prezzo, disponibilità di prenotazione/asporto.
- **Ordinamento** semplice (se richiesto) tramite metodi standard delle collezioni Java.
- **Statistiche** elementari generate dai dati aggregati (es. numero di recensioni, medie voto).

Login

L'autenticazione è gestita in modo semplice:

- Ricerca dell'utente tramite ArrayList<Utente>.
- Verifica del nome utente e confronto diretto con la password salvata.
- Nessuna cifratura, salvataggio plaintext nei CSV.

5. Formato dei file e gestione

Posizione

Tutti i file CSV si trovano nella cartella:

/data/

Gestione file

I file vengono gestiti nella fase di avvio e chiusura dell'applicazione:

- In fase di avvio: caricati in memoria tramite FileManager usando BufferedReader.
- In fase di chiusura/modifica: scritti con BufferedWriter.

I path sono **relativi** e centralizzati in costanti, così da facilitarne la modifica.

Esempio di gestione:

public static final String PATH_UTENTI = "data/utenti.csv";

BufferedReader reader = new BufferedReader(new FileReader(PATH_UTENTI));

6. Pattern utilizzati e codice significativo

Pattern

Non sono stati adottati pattern di progettazione formali.

Codice significativo

```java
/**
 * una volta invocato il manager, che ha il compito di gestire tutti dati
 * prende da ogni file i dati e li trasferisce in delle liste come i ristoranti e gli utenti
 * poi usanto i dati dei file "intermedi" come i ristotantiPreferiti associa i dati degli utenti a quelli dei ristoranti
 * in questo modo una volta avviato il programma i dati sono tutti pronti per essere utilizzati in maniera semplice e veloce
 */
public void avviaApplicazione() {
    FileManager.inizializzaFile();
    this.ristoranti = FileManager.leggiRistorantiDaCSV();
    this.utenti = FileManager.leggiUtentiDaCSV();

    this.recensioni = FileManager.leggiRecensioniDaCSV();
    this.preferiti = FileManager.leggiPreferitiDaCSV();
    this.gestiti = FileManager.leggiGestitiDaCSV();
    this.clienti = filtraClienti();
    CaricaDinamicaPreferiti();
    CaricaDinamicaRecensioni();
    this.ristoratori = filtraRistoratori();
    CaricaDinamicaRistorantiGestiti();

}
```

7. JavaDoc

Tutte le classi e metodi principali sono correttamente documentati tramite **JavaDoc**.

Il JavaDoc di questo progetto è presente nella documentazione di quest'ultimo assieme ai manuali.