TheKnife

Manuale tecnico

Università degli Studi dell'Insubria
Progetto laboratorio A – TheKnife A.A. 2024-2025
Autori:
. Strazzullo Ciro Andrea, matricola 763603
. Matteo Mongelli, matricola 760960
. Riccardo Giovanni Rubini, matricola 761126
Data: 06 Luglio 2025
Versione documento: 2.0.0

Sommario

Introduzione

TheKnife è una piattaforma digitale che consente di individuare ristoranti in tutto il mondo, offrendo la possibilità di filtrarli in base a diversi criteri: posizione geografica, tipologia di cucina, fascia di prezzo, disponibilità di prenotazione del tavolo e opzioni di asporto.
TheKnife**, dunque, **simula alcune delle funzionalità della celebre piattaforma TheFork.

Report tecnico

STRUTTURA DELL'APPLICAZIONE

L'applicazione è organizzata secondo **un'architettura MVC** (model-view-controller) **unita all'architettura DAO** (data access object) per la gestione della persistenza dei dati, seppur, a volte, in modo non completo. Attraverso questa struttura, viene facilitata la **separazione delle responsabilità**, rendendo il codice più **leggibile**, **manutenibile** ed **estensibile**.
Package principali:
. **src/theknife:** contiene tutte le classi che permettono il funzionamento dell'applicazione, suddivisa secondo le architetture precedentemente esposte
. **lib/:** contiene le librerie necessarie come la libreria gson
. **bin/:** contiene il jar dell'applicazione, pronta ad essere eseguita
. **data/:** contiene i file .json che memorizzano i dati in modo persistente
. **doc/:** contiene tutte le documentazioni per il progetto (tecnica, utente e la javadoc)

SCELTE ARCHITETTURALI

*Il progetto **TheKnife** è stato realizzato utilizzando una **struttura modulare** e ben definita utilizzando i **pattern MVC** (Model, View, Controller), anche se in formato semplificato, e **DAO**, definendo quindi dei package per ciascuno, **facilitando** così la sua **implementazione**, la **riusabilità** del codice, la **manutenibilità** e la **scalabilità** (possibilità di estendere di funzionalità le classi del programma).*

Model:

Attraverso il Model vengono implementate tutte le classi che rappresentano gli oggetti dell'applicazione nel mondo reale (Cliente, Utente, Ristoratore, Ristorante, Recensione).

View:

Attraverso la View vengono implementate tutte le classi che gestiscono la view del progetto, vista dall'utente finale (ViewBase, ViewCliente, ViewRistoratore).

Controller:

Il Controller gestisce tutta la logica di programma (Main).

DAO:

Il pattern DAO invece, gestisce tutta la logica di salvataggio/modifica/eliminazione dei dati dell'applicazione.
Nel progetto viene implementata la classe GestoreFile, che si occupa di implementare tutte le logiche precedentemente descritte.

STRUTTURE DATI UTILIZZATE

Le strutture dati utilizzate nel progetto sono:
*. **Liste:** utilizzate per contenere tutti gli oggetti creati di una determinata classe (ristoranti, utenti, recensioni)*
*. **JsonArray:** utilizzati per contenere gli oggetti salvati nel file json*
*. **Oggetti personalizzati:** tutte le istanze create di Ristoranti, Utenti e Recensioni che permettono l'esecuzione del programma*
*. **File e Path:** permettono l'accesso al filesystem e l'utilizzo dei file memorizzati o da memorizzare*

*. **URL/http:** permettono di collegarsi ad un sito (API) che permette il reverse geocoding (da indirizzo a coordinate – da coordinate a indirizzo)*

SCELTE ALGORITMICHE

*Il progetto TheKnife fa uso di diversi tipi di algoritmi per garantire **affidabilità al cliente**. Tra questi algoritmi abbiamo:*

*. **Salvataggio Automatico:** Ad ogni modifica ai dati (utenti, recensioni, ristoranti) equivale un salvataggio su file. Questa operazione, però, risulta essere pesante quando le liste aumentano notevolmente di dimensione.*
*. **Reverse Geocoding:** Per garantire l'esistenza degli indirizzi inseriti, l'applicazione ottiene le coordinate partendo da un indirizzo, o viceversa, sfruttando un API gratuita.*
*. **Formula di Haversine:** Attraverso questa formula matematica è possibile calcolare la distanza tra due punti sulla terra sapendo le loro coordinate geografiche. Utile per determinare se un ristorante è vicino o meno all'utente.*

FORMATO FILE E GESTIONE

*I dati del progetto vengono memorizzati in formato JSON, uno **standard molto diffuso** soprattutto nel web grazie alla sua leggibilità e facilità d'uso. Questo formato permette di **rappresentare in modo semplice e strutturato** sia oggetti singoli che collezioni complesse di dati.*

*Poiché Java non include una gestione nativa per i file JSON, il progetto utilizza la **libreria Gson di Google**, che consente di serializzare e deserializzare facilmente gli oggetti Java in formato JSON e viceversa.*

I file dati sono salvati all'interno della cartella data, che contiene due file principali:

*. **Utenti.json**, dove sono memorizzati tutti gli utenti del sistema, inclusi ristoratori e clienti;*

*. **Ristoranti.json**, che contiene tutte le informazioni relative ai ristoranti.*

Attenzione!
Al primo avvio dell'applicazione, la cartella data e i file JSON non sono ancora presenti nel filesystem. Saranno creati automaticamente dal programma per garantire il corretto funzionamento.

Per assicurare la massima portabilità tra diversi sistemi operativi (Windows, macOS, Linux), la classe GestoreFile implementa un metodo che restituisce il percorso (path) corretto, utilizzando il separatore di file appropriato in base al sistema su cui l'applicazione è in esecuzione.

USO DI PATTERN E CODICE SIGNIFICATIVO

Come accennato nelle SCELTE ARCHITETTURALI il progetto fa uso del pattern MVC (Model, View, Controller), semplificato, e DAO (Data Access Object), ognuno dei quali "implementa" una propria logica.

Codice significativo:

View (ViewBase, metodo ridotto):

```
/**
    * Metodo per visualizzare l'interfaccia grafica di base
    *
    * @param pathUtenti     path del file contenente gli utenti
    * @param PATHRISTORANTI path del file contenente i ristoranti
    * @throws Exception eccezione lanciata in qualsiasi caso di errore di esecuzione
con messaggio
    */
   public static void view(String pathUtenti, String PATHRISTORANTI) throws
Exception {
       try (Scanner s = new Scanner(System.in)) {
          System.out.println("Avvio del progetto....");
          System.out.println("Progetto avviato con successo!");
          System.out.println("Carico i ristoranti..");
          List<Ristorante> ristoranti = GestoreFile.caricaRistoranti(PATHRISTORANTI);
          System.out.println("Ristoranti caricati con successo!");
          System.out.println("Carico l'interfaccia grafica di base, inzio programma...");
          TheKnife.svuotaConsole();
          System.out.println("Benvenuto in TheKnife!");
          System.out.println("Ecco il menu principale:");
          boolean continua = true;
          Utente u;
          while (continua) {
             int scelta = convertiScannerIntero("""
                   \n\n
                   Menù:
                   1. Visualizza ristoranti (luogo, fascia prezzo, servizi, recensioni) in
modalità guest
                   2. Login come cliente o ristoratore
                   3. Registrati come cliente o ristoratore
```

```
                    4. Chiudi l'applicazione
                    La tua scelta:
                """, s);
            switch (scelta) {
                case 1 -> {

                    String luogo;
                    do {
                        luogo = gestisciInput("Inserisci il luogo di ricerca:", s);
                        double[] latLon = ReverseGeocoding.getLatitudineLongitudine(luogo);
                        if (latLon[0] == -1 && latLon[1] == -1) {
                            luogo = "";
                            System.out.println("Luogo non valido, riprova! Assicurati di
inserire un luogo esistente.");
                        }
                    } while (luogo.isEmpty());

                    boolean continuaInterno = true;
                    do {
                        int sceltaIn = convertiScannerIntero("""
                                \n\n
                                Menù Ristoranti guest :
                                1. Visualizza ristoranti vicini al luogo specificato con i
relativi dettagli
                                2. Visualizza ristoranti secondo un filtro e i relativi dettagli
                                3. Modifica luogo
                                4. Torna al menù principale
                                La tua scelta:
                            """, s);
                        switch (sceltaIn) {
                            case 1 -> {
                                List<Ristorante> tmp = ristorantiVicini(ristoranti, luogo);
                                if (tmp != null && !tmp.isEmpty()) {
                                    for (Ristorante r : tmp) {
                                        System.out.println(r.visualizzaRistorante());
                                        System.out.println("Recensioni:");
                                        for (Recensione rec : r.getRecensioni()) {
                                            System.out.println(rec);
                                        }
                                        String continuaRicerca;
                                        do {
                                            System.out.println("Digita 'c' per continuare la ricerca o
'q' per tornare al menù ");
                                            continuaRicerca = s.nextLine();
```

```
                        } while ((!continuaRicerca.equalsIgnoreCase("c")
&& !continuaRicerca.equalsIgnoreCase("q")));
                        TheKnife.svuotaConsole();
                        if (continuaRicerca.equalsIgnoreCase("q")) {
                            System.out.println("Tornando al menù ...");
                            break;
                        }
                    }
                } else {
                    System.out.println("Nessun ristorante trovato!");
                }

            }
            case 2 -> {
                System.out.println("Inserisci i parametri di ricerca:");
                String locazione;
                do {
                    locazione = gestisciInput("Inserisci la locazione. Parametro
obbligatorio:", s);
                    if (locazione.isEmpty()) {
                        System.err.println("La locazione è obbligatoria!");
                    } else {
                        double[] latLon =
ReverseGeocoding.getLatitudineLongitudine(locazione);
                        if (latLon[0] == -1 && latLon[1] == -1) {
                            locazione = "";
                            System.out.println("Locazione non valida, riprova!
Assicurati di inserire un luogo esistente.");
                        }
                    }
                } while (locazione.isEmpty());
                System.out.println("Inserisci il tipo di cucina (premi invio per
saltare):");
                String tipoCucina = s.nextLine();
                double minPrezzo = convertiScannerDouble("Prezzo minimo (0
per ignorare):", s);
                double maxPrezzo = convertiScannerDouble("Prezzo massimo (0
per ignorare):", s);
                if (maxPrezzo > 0 && minPrezzo > maxPrezzo) {
                    System.err.println("Attenzione: il prezzo minimo supera il
massimo. Imposto entrambi a 0.");
                    minPrezzo = 0;
                    maxPrezzo = 0;
                }
```

```
                        System.out.println("Vuoi includere il filtro delivery? (si/no - predefinito: no):");
                        boolean conDelivery = s.nextLine().equalsIgnoreCase("si");
                        System.out.println("Vuoi includere solo ristoranti con servizio delivery? (si/no - predefinito: no):");
                        boolean filtroDelivery = s.nextLine().equalsIgnoreCase("si");
                        System.out.println("Vuoi includere il filtro prenotazione? (si/no - predefinito: no):");
                        boolean conPrenotazione = s.nextLine().equalsIgnoreCase("si");
                        System.out.println("Vuoi includere solo ristoranti con servizio prenotazione? (si/no - predefinito: no):");
                        boolean filtroPrenotazione = s.nextLine().equalsIgnoreCase("si");
                        int minStelle = convertiScannerIntero("Numero minimo di stelle (0-5, default: 0):", s);
                        if (minStelle < 0 || minStelle > 5) {
                            System.out.println("Valore non valido per le stelle. Impostato a 0.");
                            minStelle = 0;
                        }
                        List<Ristorante> risultati = Ristorante.combinata(
                                ristoranti, locazione, tipoCucina, minPrezzo, maxPrezzo,
                                conDelivery, filtroDelivery, conPrenotazione, filtroPrenotazione, minStelle
                        );

                        if (risultati.isEmpty()) {
                            System.out.println("Nessun ristorante trovato con i criteri specificati.");
                        } else {
                            for (Ristorante r : risultati) {
                                System.out.println(r.visualizzaRistorante());
                                System.out.println("Recensioni:");
                                for (Recensione rec : r.getRecensioni()) {
                                    System.out.println(rec);
                                }
                                String sceltaInterna;
                                do {
                                    System.out.println("Digita 'c' per continuare a vedere i risultati o 'q' per tornare al menu principale:");
                                    sceltaInterna = s.nextLine();
                                } while (!sceltaInterna.equalsIgnoreCase("c") && !sceltaInterna.equalsIgnoreCase("q"));

                                TheKnife.svuotaConsole();
```

```
                            if (sceltaInterna.equalsIgnoreCase("q")) {
                                System.out.println("Torno al menu principale...");
                                break;
                            }
                        }
                    }
                }
                case 4 -> {
                    System.out.println("Tornando al menù principale...");
                    continuaInterno = false;
                    TheKnife.svuotaConsole();
                }
                case 3 -> {
                    do {
                        luogo = gestisciInput("Inserisci il luogo di ricerca:", s);
                        double[] latLon =
ReverseGeocoding.getLatitudineLongitudine(luogo);
                        if (latLon[0] == -1 && latLon[1] == -1) {
                            luogo = "";
                            System.out.println("Luogo non valido, riprova! Assicurati di
inserire un luogo esistente.");
                        }
                    } while (luogo.isEmpty());
                    System.out.println("Luogo modificato con successo!");
                }
            }
        } while (continuaInterno);
    }
    case 2 -> {
        u = login(s, pathUtenti, PATHRISTORANTI);
        if (u instanceof Cliente) {
            continua = false;
            ViewCliente.view((Cliente) u, pathUtenti, PATHRISTORANTI);
        } else if (u instanceof Ristoratore) {
            continua = false;
            ViewRistoratore.view((Ristoratore) u, pathUtenti, PATHRISTORANTI);
        } else {
            System.err.println("Login non avvenuto con successo!");
        }
    }
    case 3 -> {
        List<Utente> utenti = caricaUtenti(pathUtenti, PATHRISTORANTI);
        u = registrati((utenti == null ? new ArrayList<Utente>() : utenti), s);
        if (u != null) {
            utenti.add(u);
```

```
                    GestoreFile.salvaUtenti(utenti, pathUtenti);
                    if (u instanceof Cliente) {
                        continua = false;
                        ViewCliente.view((Cliente) u, pathUtenti, PATHRISTORANTI);
                    } else if (u instanceof Ristoratore) {
                        continua = false;
                        ViewRistoratore.view((Ristoratore) u, pathUtenti,
PATHRISTORANTI);
                    } else {
                        System.err.println("Registrazione non avvenuta con successo!");
                    }
                }
            }
            case 4 -> {
                System.out.println("Grazie per aver utilizzato TheKnife!");
                System.out.println("Arrivederci!");
                continua = false;
            }
            default ->
                    System.err.println("Attenzione, scelta non valida, riprova!");
        }
      }
    } catch (IOException e) {
        System.err.println("Errore durante l'esecuzione del programma: " +
e.getMessage());
    } catch (Exception e) {
        System.err.println("Errore imprevisto: " + e.getMessage());
    }
  }
```

DAO:

```
/**
   * Metodo per ottenere la lista di ristoranti salvati su file
   *
   * @param path path al file contenente i ristoranti
   * @return lista di ristoranti
   * @throws IOException eccezione lanciata in caso di errore
   */
  public static List<Ristorante> caricaRistoranti(String path) throws IOException {
      creaFile(path, "[]");
      try (Reader reader = new FileReader(path)) {

          JsonArray array = JsonParser.parseReader(reader).getAsJsonArray();
```

```
        List<Ristorante> lista = new ArrayList<>();

        int maxId = -1;

        for (JsonElement elem : array) {
            JsonObject obj = elem.getAsJsonObject();

            int id = obj.get("id").getAsInt();
            String nome = obj.get("nome").getAsString();
            String nazione = obj.get("nazione").getAsString();
            String citta = obj.get("citta").getAsString();
            String indirizzo = obj.get("indirizzo").getAsString();
            String tipoCucina = obj.get("tipoCucina").getAsString();
            boolean delivery =  obj.get("delivery").getAsBoolean();
            boolean prenotazioneOnline =  obj.get("prenotazioneOnline").getAsBoolean();
            double minPrezzo = obj.get("minPrezzo").getAsDouble();
            double maxPrezzo = obj.get("maxPrezzo").getAsDouble();

            if (id > maxId) {
                maxId = id;
            }

            Ristorante ristorante  = new Ristorante(
                    id,
                    nome,
                    nazione,
                    citta,
                    indirizzo,
                    tipoCucina,
                    delivery,
                    prenotazioneOnline,
                    minPrezzo,
                    maxPrezzo
            );

            if (obj.has("recensioni")) {
                JsonArray recensioniArray = obj.getAsJsonArray("recensioni");
                for (JsonElement recElem : recensioniArray) {
                    JsonObject recObj = recElem.getAsJsonObject();
                    String descrizione = recObj.get("descrizione").getAsString();
                    int stelle = recObj.get("stelle").getAsInt();
                    int idRec = recObj.get("id").getAsInt();
                    String risposta = recObj.has("risposta") ?
recObj.get("risposta").getAsString() : "";
```

```
                    try {
                        Recensione recensione;
                        if (risposta.isEmpty()) {
                            recensione = new Recensione(descrizione, stelle, idRec);
                        } else {
                            recensione = new Recensione(descrizione, stelle, risposta, idRec);
                        }
                        ristorante.recensisciRistorante(recensione);
                    } catch (Exception e) {
                        System.err.println("Errore nel caricamento recensione: " +
e.getMessage());
                    }
                }
            }

            lista.add(ristorante);
        }

        if (maxId >= 0) {
            Ristorante.aggiornaContatore(maxId);
        }

        return lista;

    } catch (Exception e) {
        throw new IOException("Errore caricamento ristoranti: "  + e.getMessage());
    }
}
```

Limiti della soluzione sviluppata

*Il progetto TheKnife, così come progettato e sviluppato, presenta alcune l**imitazioni strutturali e tecnologiche**, descritte di seguito:*

. Assenza di interfaccia grafica (GUI):
L'applicazione è interamente basata su Java standard, senza l'uso di framework grafici o librerie per GUI. Di conseguenza, l'interazione con

l'utente avviene esclusivamente tramite terminale, rendendo l'esperienza utente meno intuitiva rispetto a una classica interfaccia visiva.
Tuttavia, questa scelta ha permesso di semplificare lo sviluppo e ridurre la complessità del codice.

. Gestione dei dati tramite file di testo*:*
I dati relativi ai ristoranti sono memorizzati in file json, che vengono letti e scritti direttamente dall'applicazione.
Questo approccio è semplice da implementare, ma comporta una minore efficienza nelle operazioni di lettura, modifica e salvataggio, specialmente con dataset di grandi dimensioni. Ogni operazione comporta infatti la lettura dell'intero file, la traduzione, la modifica in memoria e la riscrittura completa del file.

. Utilizzo locale e non distribuito:
L'applicazione è stata pensata per l'utilizzo locale tramite IDE o terminale, e non prevedere quindi componenti client-server che permettono la sincronizzazione dei dati delle esecuzioni su diversi dispositivi.

Sitografia e Bibliografia

*Durante lo sviluppo del programma sono stati utilizzati **diversi siti di riferimento**, utili sia per l'installazione corretta degli strumenti necessari che per la gestione del codice e dei dati, utile anche per l'utente che lo esegue nel caso di problemi o per cultura personale:*

. Installare Java su Windows:
https://www.geeksforgeeks.org/download-and-install-java-development-kit-jdk-on-windows-mac-and-linux/

. Come installare Homebrew su MacOs:
https://brew.sh

. Come installare Java su MacOs con Homebrew:
https://formulae.brew.sh/formula/openjdk

. Come installare git e Github su MacOs:
https://formulae.brew.sh/formula/git
https://medium.com/@bykov.tech/git-github-tutorial-basics-of-working-with-github-on-a-mac-f7817ff0d0da

. *Come installare git e Github su Windows:*
https://it.siteground.com/kb/come-installare-git-windows/

. *Cos'è git e cos'è Github:*
https://docs.github.com/en/get-started/start-your-journey/about-github-and-git

. *Libreria Gson:*
https://github.com/google/gson?tab=readme-ov-file
https://www.javadoc.io/doc/com.google.code.gson/gson/latest/com.google.gson/module-summary.html

. *Cos'è il JDK:*
https://www.geeksforgeeks.org/jdk-in-java/

. *Cos'è un IDE:*
https://www.codecademy.com/article/what-is-an-ide