# Documentazione Tecnica del Progetto: Piattaforma di Clustering Gerarchico Distribuito

Lascaro Gianluca
Matricola: 758367

Esame di Metodi Avanzati di Programmazione - Anno Accademico 2023/24
Aprile 2025

## Indice

# Introduzione

Il progetto realizza un sistema distribuito **client-server** per l'analisi avanzata di dataset tramite algoritmi di **clustering gerarchico**. L'applicazione consente di:

- Caricare dati da database MySQL
- Generare dendrogrammi con algoritmi **Single-Link** e **Average-Link**
- Visualizzare, salvare e ricaricare modelli serializzati
- Gestire connessioni concorrenti tramite architettura multithreading

Il sistema integra componenti di **elaborazione distribuita**, **GUI interattiva** (Java Swing), e **persistenza dati**, seguendo i principi del pattern **MVC** (Model-View-Controller).

# 1 Scopo e Obiettivi

## 1.1 Obiettivi Funzionali

- **Comunicazione Client-Server**:
  - Connessione TCP/IP con gestione di porte customizzabili
  - Serializzazione di oggetti complessi (dendrogrammi, dataset) tramite `ObjectOutputStream`
- **Clustering Gerarchico**:
  - Supporto per **distanze Single-Link** (minima tra cluster) e **Average-Link** (media tra cluster)
  - Costruzione di dendrogrammi con profondità personalizzabile
- **Integrazione Database**:
  - Lettura di tabelle MySQL e conversione in oggetti `Data` per l'elaborazione
- **Gestione File**:
  - Salvataggio e caricamento di dendrogrammi in formato binario (`.ser`)

## 1.2 Obiettivi Non Funzionali

- **Scalabilità**: Server multithreading per gestire più client
- **Robustezza**: Gestione centralizzata di eccezioni (es. `InvalidDepthException`, `NoDataException`)
- **Usabilità**: Interfaccia grafica intuitiva con validazione input

# 2 Architettura del Sistema

## 2.1 Diagramma dei Componenti

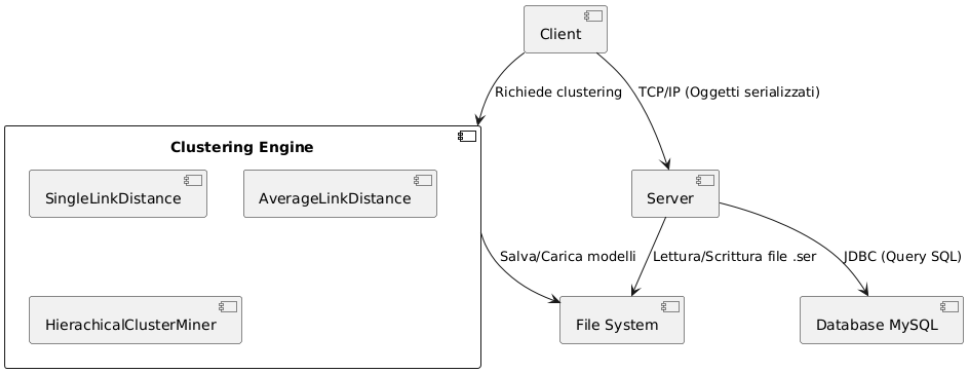


Figura 1: Architettura ad alto livello del sistema

## 2.2 Diagramma delle Classi

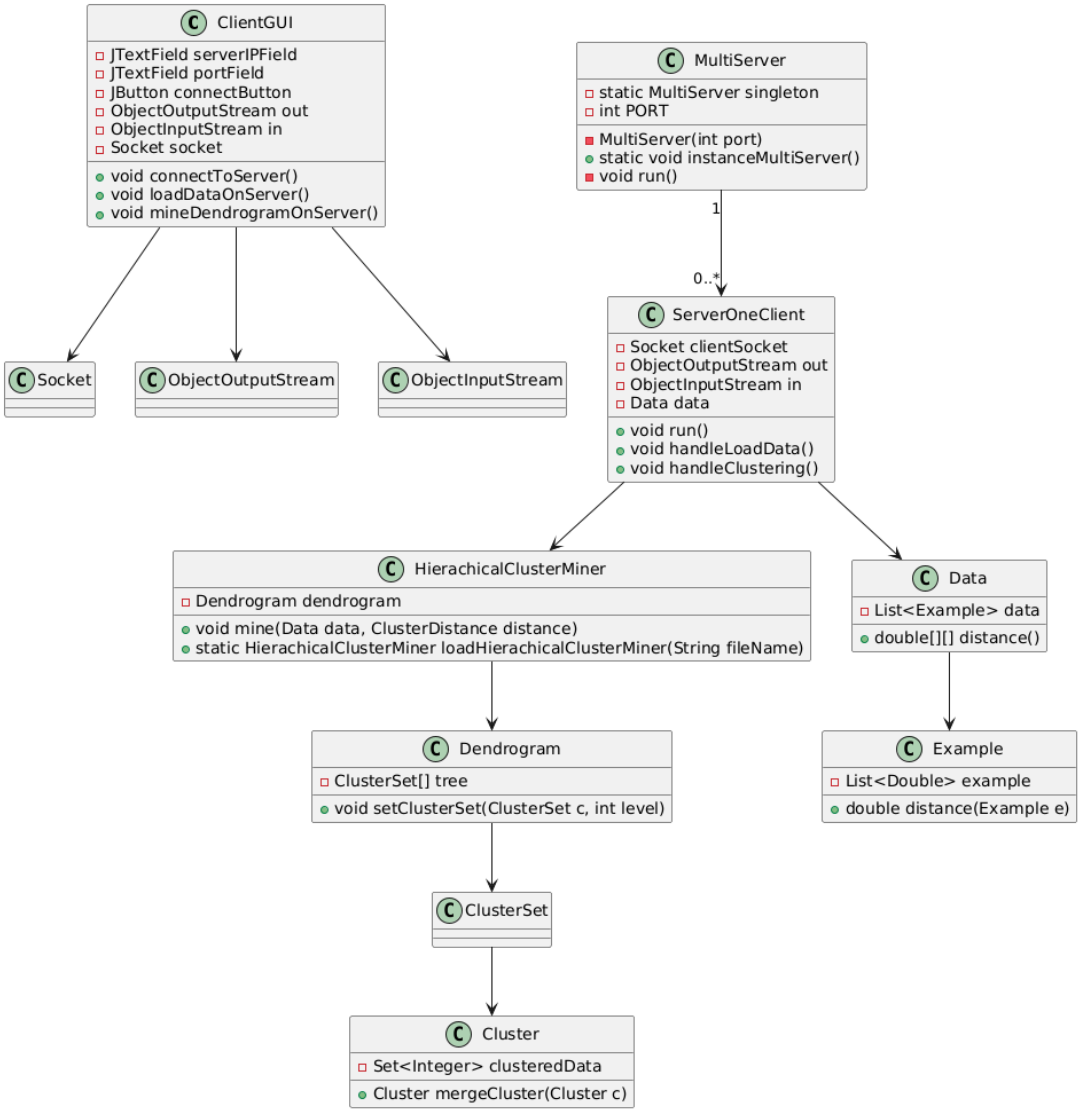


Figura 2: Struttura dettagliata delle classi e delle loro relazioni

Principali componenti:

- **ClientGUI**: Gestione interfaccia utente e comunicazione
- **MultiServer**: Implementazione pattern Singleton per il server
- **HierarchicalClusterMiner**: Core algorithm implementation

- **ClusterDistance**: Gerarchia per le strategie di calcolo distanze

### 2.3 Design Pattern Utilizzati

- **Singleton**: Classe `MultiServer` per garantire un'unica istanza del server
- **Factory Method**: Creazione di oggetti `ClusterDistance` (Single/Average-Link)
- **Observer**: Aggiornamento dinamico della GUI in risposta agli eventi del server
- **Strategy**: Implementazione delle diverse metriche di distanza

## 3 Dettagli Implementativi

### 3.1 Componente Client (`ClientGUI`)

**Struttura GUI**:

- **JTabbedPane** per organizzare operazioni (connessione, clustering, salvataggio)
- **JTextArea** con scroll per log dettagliati
- Controlli di validazione (es. campi vuoti, formati numerici)

**Gestione Connessione**:

```
private void connectToServer() {
    if (isConnected) closeConnection();
    try {
        socket = new Socket(serverIPField.getText(), Integer.parseInt(
            portField.getText()));
        out = new ObjectOutputStream(socket.getOutputStream());
        in = new ObjectInputStream(socket.getInputStream());
        // Abilita pulsanti post-connessione
    } catch (IOException e) {
        outputArea.append("Errore: " + e.getMessage());
    }
}
```

### 3.2 Componente Server (`MultiServer`, `ServerOneClient`)

**Multithreading**:

```
public class MultiServer {
    private static MultiServer instance;
    private MultiServer(int port) { /* ... */ }
    public static void instanceMultiServer() { // Singleton pattern }
    private void run() {
        while (true) {
            Socket socket = serverSocket.accept();
```

```
            new ServerOneClient(socket); // Nuovo thread per client
        }
    }
}
```

**Algoritmo di Clustering** (`HierarchicalClusterMiner`):

```
public void mine(Data data, ClusterDistance distance) throws
    InvalidDepthException {
    ClusterSet initialClusters = createInitialClusters(data);
    dendrogram.setClusterSet(initialClusters, 0);
    for (int level = 1; level < depth; level++) {
        ClusterSet merged = mergeClosestClusters(level-1, distance, data);
        dendrogram.setClusterSet(merged, level);
    }
}
```

## 3.3 Descrizione delle Schermate

L'interfaccia grafica del client è organizzata in tre schermate principali che guidano l'utente attraverso il workflow di clustering. Di seguito una descrizione dettagliata.

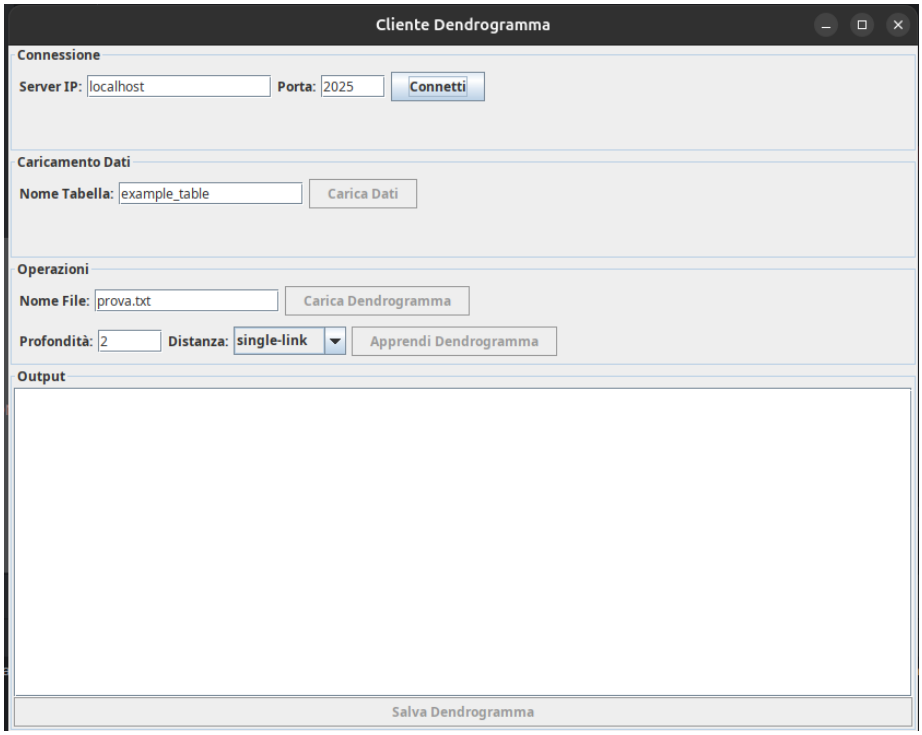


Figura 3: Schermata iniziale di connessione e configurazione. L'utente può specificare l'IP del server (default: localhost), la porta (2025), e avviare la connessione. Sono presenti sezioni per il caricamento di dati da tabelle MySQL e l'impostazione dei parametri di clustering.

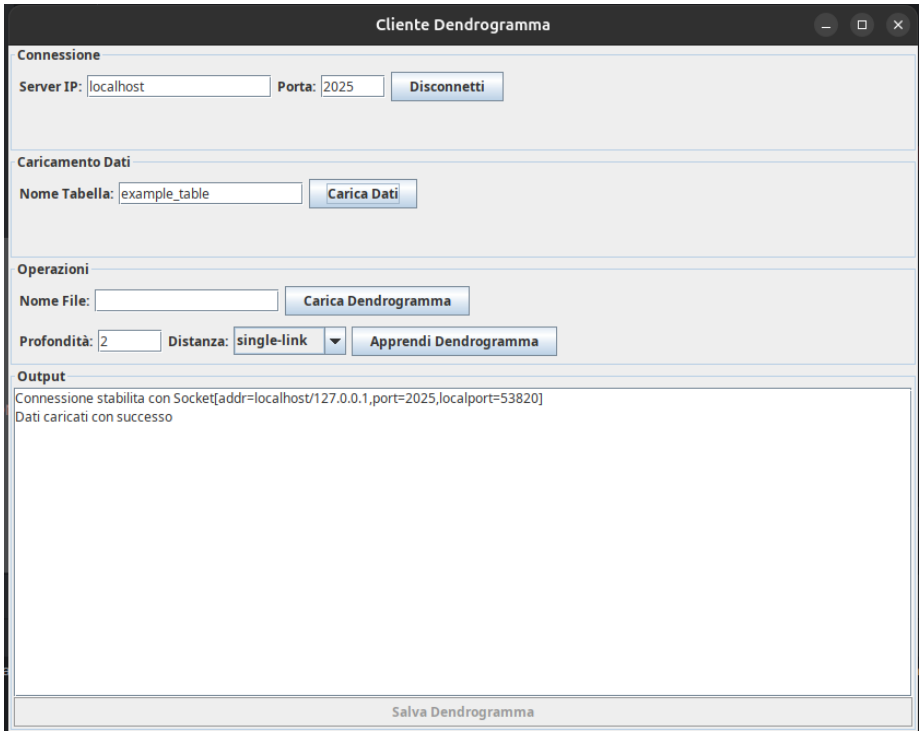


Figura 4: Schermata operativa post-connessione. Mostra la conferma della connessione stabilita e l'esito del caricamento dati. L'utente può procedere alla generazione del dendrogramma impostando profondità (es. 2) e metrica di distanza (single-link).

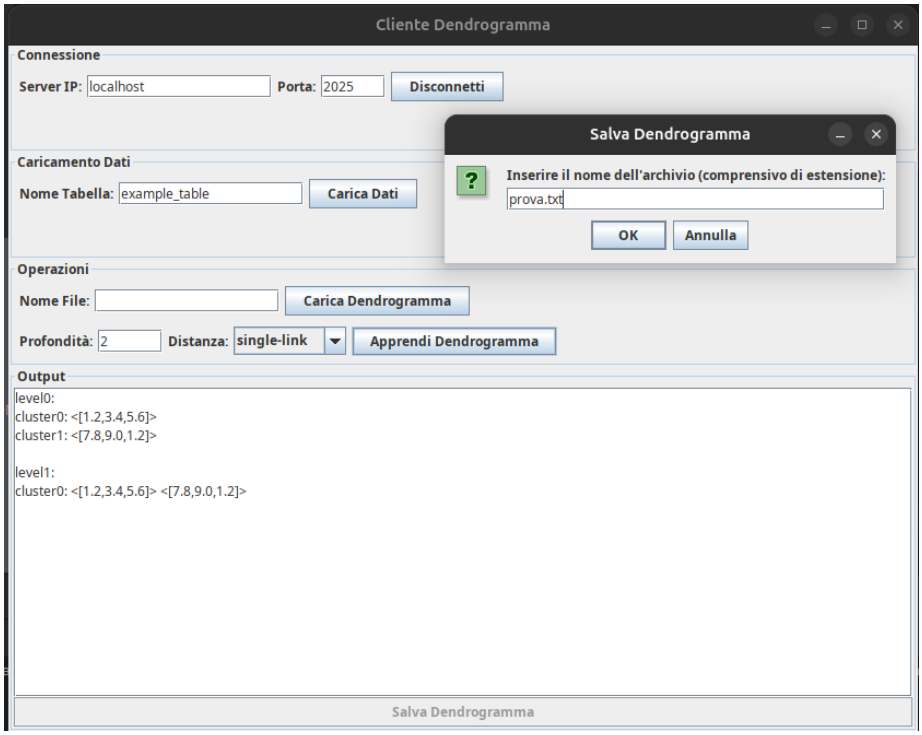


Figura 5: Risultato del clustering gerarchico. Visualizza i cluster a diversi livelli di profondità (level0 e level1), con i relativi centroidi. Include opzioni per salvare il dendrogramma in formato binario o annullare l'operazione.

### 3.3.1 Flusso Interattivo

- **Schermata 1**: Configurazione iniziale e caricamento dati. I pulsanti `Connetti` e `Carica Dati` abilitano le successive fasi.
- **Schermata 2**: Esecuzione dell'algoritmo. La sezione `Output` mostra log di stato in tempo reale (es. "Dati caricati con successo").
- **Schermata 3**: Analisi dei risultati. I cluster sono rappresentati in formato testuale con sintassi `<f1,f2,f3>` per i centroidi. L'utente può navigare tra i livelli del dendrogramma.

# 4 Flusso di Lavoro (Workflow)

## 4.1 Diagramma di Sequenza

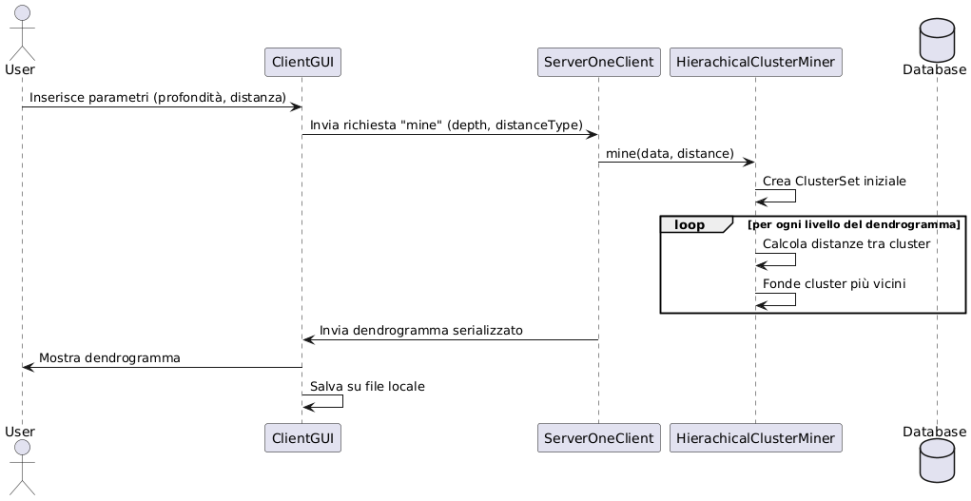


Figura 6: Interazione client-server durante il processo di clustering

### 4.2 Diagramma di Attività

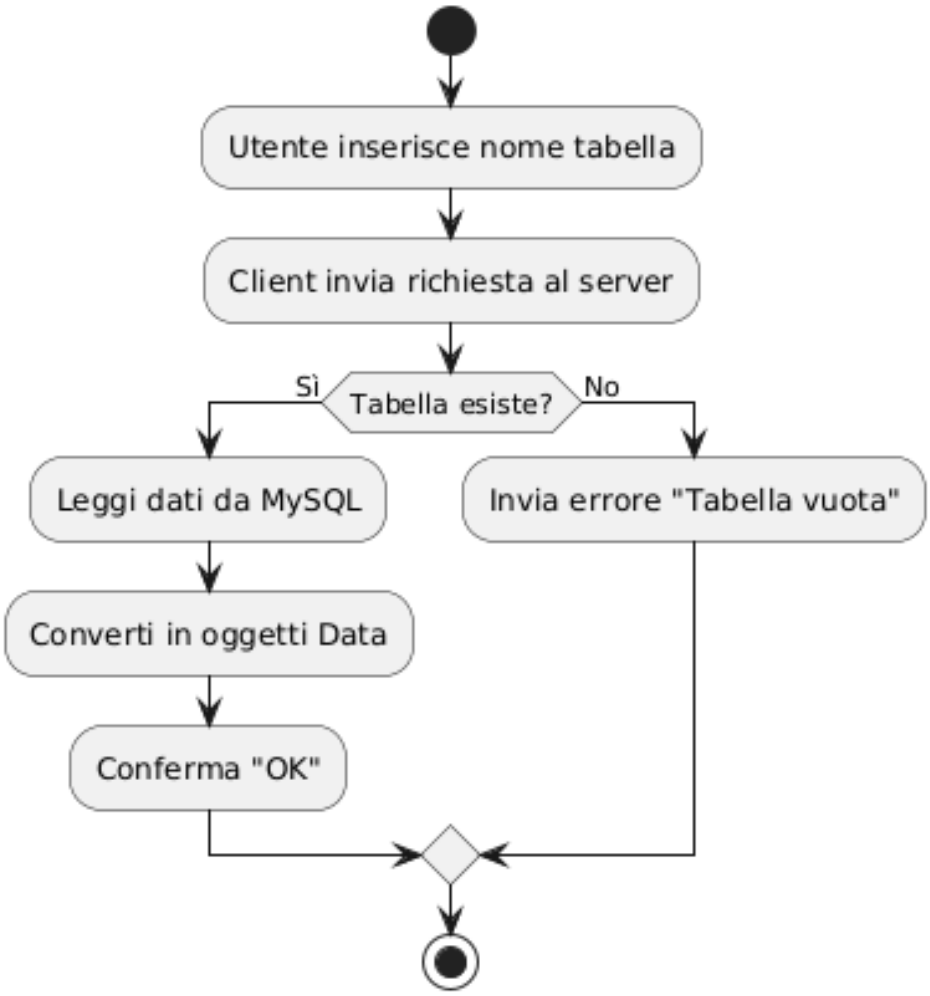


Figura 7: Flusso decisionale per la generazione del dendrogramma

## 5 Gestione degli Errori

### 5.1 Eccezioni Personalizzate

- `InvalidDepthException`: Profondità non valida (es. 0)
- `NoDataException`: Tabella database vuota o inesistente
- `InvalidClustersNumberException`: Tentativo di fusione con un solo cluster

### 5.2 Logging e Feedback

- Output in `JTextArea` con messaggi di errore dettagliati
- Utilizzo di `JOptionPane` per dialoghi modali in caso di errori critici
- Sistema di logging centralizzato nel server

## 6 Deployment e Configurazione

### 6.1 Requisiti di Sistema

- **MySQL Server**: Configurazione con tabelle secondo schema specificato
- **Librerie**: MySQL Connector/J inclusa nel classpath

### 6.2 Configurazione Database

- Creare un utente `root` con privilegi sulla tabella target
- Esempio di struttura tabella:

```
CREATE TABLE example_table (
    feature1 FLOAT,
    feature2 FLOAT,
    feature3 FLOAT
);
```

## Conclusioni

Il progetto dimostra un'implementazione robusta di un sistema distribuito per l'analisi dati, combinando tecnologie avanzate (socket, multithreading, algoritmi di clustering) in un'architettura modulare. Le scelte progettuali, come l'uso della serializzazione per i dendrogrammi e il pattern Singleton per il server, garantiscono scalabilità e manutenibilità.