Laboratorio interdisciplinare A

Climate Monitoring

Un sistema di monitoraggio di parametri climatici

CATTANEO LUCA
FICARA PAOLO
MAURI ANDREA

Indice

Introduzione

Climate monitoring è un progetto sviluppato nell'ambito del progetto di Laboratorio A per il corso di laurea in Informatica dell'Università degli Studi dell'Insubria.

Il progetto è stato sviluppato in Java 17, usa un'interfaccia grafica costruita con Java Swing, libreria inclusa in NetBeans apposita per lo sviluppo del design delle applicazioni desktop, ed è stato sviluppato e testato su Windows 11.

Librerie esterne utilizzate

Durante lo sviluppo di questo progetto è stata utilizzata una libreria di terze parti in aggiunta,

- ## codice-fiscale-java-master

  Il codice sorgente è stato reperito in rete ed è stato adattato e trasformato in libreria. È incluso nel progetto ai fini di denominare il codice fiscale avendo tutte le generalità necessarie fornite dall'utente in fase di registrazione.

STRUTTURA DEL DATABASE

TABELLE PRINCIPALI

parametriclimatici

Descrizione: Memorizza i dati climatici raccolti dai vari centri di monitoraggio.

Campi:

- idcitta: Identificativo della città (chiave primaria).
- nome_centro: Nome del centro di monitoraggio.
- data: Data della registrazione dei parametri.

- ora: Ora della registrazione dei parametri.
- vento: Velocità del vento.
- umidita: Livello di umidità.
- pressione: Pressione atmosferica.
- temperatura: Temperatura.
- precipitazioni: Quantità di precipitazioni.
- altitudine: Altitudine.
- massa: Massa dei ghiacciai.

coordinatemonitoraggio

Descrizione: Contiene le informazioni geografiche e identificative delle aree di interesse disponibili per i centri di monitoraggio.

Campi:

- id: Identificativo unico del centro (chiave primaria).
- name: Nome della città.
- name_ascii: Nome della città in caratteri ASCII.
- country_code: Codice del Paese.
- country_name: Nome del Paese.
- lat: Latitudine.
- lon: Longitudine.

centromonitoraggio

Descrizione: Memorizza informazioni sui centri di monitoraggio.

Campi:

- name: Nome del centro di monitoraggio (chiave primaria).
- address: Indirizzo del centro.

operatoreregistrato

Descrizione: Contiene le informazioni degli operatori registrati che gestiscono i centri di monitoraggio.

Campi:

- cf: Codice fiscale dell'operatore (chiave primaria).
- nome: Nome dell'operatore.
- cognome: Cognome dell'operatore.
- mail: Indirizzo email dell'operatore.
- nick: Nome utente.
- password: Password per l'accesso.
- nome_centro: Nome del centro di monitoraggio presso cui l'operatore lavora.

lavora

Descrizione: Rappresenta la relazione N:N tra i centri di monitoraggio e le loro coordinate.

Campi:

id_coordinate: Identificativo delle coordinate (chiave esterna verso coordinatemonitoraggio).

nome_centro: Nome del centro di monitoraggio (chiave esterna verso centromonitoraggio).

RELAZIONI

parametriclimatici- coordinatemonitoraggio:

La tabella parametriclimatici è collegata alla tabella coordinatemonitoraggio tramite l'identificativo della città (idcitta).

parametriclimatici- centromonitoraggio:

La tabella parametriclimatici è collegata alla tabella centromonitoraggio tramite il campo nome_centro.

operatoreregistrato- centromonitoraggio:

La tabella operatoreregistrato è collegata alla tabella centromonitoraggio tramite il campo nome_centro.

lavora- coordinatemonitoraggio e centromonitoraggio:

La tabella lavora stabilisce una relazione tra la tabella coordinatemonitoraggio e la tabella centromonitoraggio tramite i campi id_coordinate e nome_centro.

Struttura generale del sistema di classi

Nel progetto si possono suddividere le classi in tre macro-rami: classi adibite per la gestione dei dati e la rappresentazione degli oggetti in questione, classi impiegate alla gestione dell'interfaccia grafica e classi per la gestione e scambio dati client/server.

Gestione dei dati e rappresentazioni oggetti

- DatiCondivisi
- DBManager
- Forecast
- InterestingAreas
- MonitoringStation
- User
- ServerInterface

Gestione dell'interfaccia grafica

- AddNotes
- CreateMonitoringStation
- Login
- Menu(main)
- MenuOperatore
- Register
- SearchResult

Gestione connessione client/server

- ServerMain
- ClientHandler

Verranno ora presentate le classi adibite alla gestione dei dati dettagliamente, non verranno invece affrontate nel particolare le classi per la gestione dell'interfaccia grafica essendo che non sono espressamente richieste nei requisiti del progetto.

Classi per la gestione dei dati

InterestingAreas

InterestingAreas è la classe che rappresenta le aree di interesse ai quali si riferisce un utente.

L'oggetto che descrive l'area di interesse viene identificato con i seguenti attributi:

- Id: identificativo numerico univoco rappresentato dal tipo String
- name: nome dell'area di interesse rappresentato dal tipo String
- countryCode: il codice del paese dell'area di interesse rappresentato dal tipo String
- lat: la latitudine dell'area di interesse rappresentata dal tipo String
- lon: la longitudine dell'area di interesse rappresentata dal tipo String

Metodi principali della classe:

1. Metodo contains(String s)
    - Descrizione: Verifica se il nome dell'area contiene una determinata stringa (ignorando maiuscole/minuscole).
    - Parametri:
        - s: la stringa da cercare
    - Complessità: O(n), dove 'n' è la lunghezza del nome dell'area

Nota: La complessità degli accessi e delle modifiche agli attributi è O(1), in quanto si tratta di operazioni dirette sugli attributi dell'oggetto.

MonitoringStation

MonitoringStation è la classe che rappresenta le stazioni di monitoraggio che un operatore può creare.

L'oggetto che descrive la stazione di monitoraggio viene identificato con i seguenti attributi:

- name: il nome della stazione di monitoraggio rappresentato dal tipo String
- address: l'indirizzo della stazione di monitoraggio rappresentato dal tipo String
- interestingAreas: le aree di interesse della stazione di monitoraggio rappresentato da un vettore di String

Tutti i metodi hanno complessità O(1), poiché eseguono operazioni dirette sugli attributi dell'oggetto, senza iterazioni o algoritmi complessi.

User

User è la classe che rappresenta un operatore.

L'oggetto che descrive un operatore viene identificato con i seguenti attributi:

- name: il nome dell'operatore rappresentato dal tipo String
- surname: il cognome dell'operatore rappresentato dal tipo String
- cf: il codice fiscale dell'operatore rappresentato dal tipo String
- mail: l'indirizzo mail dell'operatore rappresentato dal tipo String

- nick: il nickname dell'operatore rappresentato dal tipo String
- password: la password dell'operatore rappresentato dal tipo String
- station: la stazione di monitoraggio dell'operatore rappresentato dal tipo String

Nota: Tutti i metodi hanno complessità O(1), poiché eseguono operazioni dirette sugli attributi dell'oggetto, senza iterazioni o algoritmi complessi.

Forecast

La classe Forecast rappresenta una previsione meteorologica associata a una specifica città e stazione di rilevamento. Questa classe include attributi per descrivere vari parametri meteorologici, oltre a metodi per impostare e ottenere questi parametri. La classe implementa l'interfaccia Serializable, permettendo quindi la serializzazione degli oggetti.

Attributi della Classe

- idCitta: L'ID della città a cui si riferisce la previsione. Tipo: String
- NomeStazione: Il nome della stazione meteorologica. Tipo: String
- data: La data della previsione. Tipo: Date
- ora: L'ora della previsione. Tipo: Timestamp
- vento: I dati relativi al vento. Tipo: String[]
- umidita: I dati relativi all'umidità. Tipo: String[]
- pressione: I dati relativi alla pressione. Tipo: String[]
- temperatura: I dati relativi alla temperatura. Tipo: String[]
- precipitazioni: I dati relativi alle precipitazioni. Tipo: String[]
- altitudine: I dati relativi all'altitudine. Tipo: String[]
- massa: I dati relativi alla massa. Tipo: String[]

Metodi principali della Classe

1. Metodo toQuery()
    - Descrizione: Restituisce una stringa da utilizzare come statement per la costruzione di una query per inserire i dati all'interno di un database.

Note: Accesso e Modifica degli Attributi: Gli attributi della classe possono essere accessi e modificati utilizzando i metodi getter e setter. La complessità di queste operazioni è O(1).

Rappresentazione per interfacciarsi al database: Il metodo toQuery consente di ottenere una stringa che rappresenta la previsione in un formato compatibile con l'inserimento in un database, facilitando l'integrazione con sistemi di archiviazione dati.

DBManager

La classe DBManager è responsabile della gestione delle operazioni di lettura e scrittura sui dati del database relativi agli utenti, aree di interesse, stazioni di monitoraggio e previsioni climatiche. Utilizza connessioni JDBC per interagire con un database PostgreSQL.

Metodi principali della Classe

1. readUser(Connection conn)
    - Descrizione: Legge i dati dalla tabella operatoreregistrato e li trasforma in una lista di oggetti User.

- Parametri:
    - conn - Connessione al database.
- Ritorna: Una lista di oggetti User.
- Eccezioni:
    - SQLException - Errore durante l'esecuzione della query.
    - ClassNotFoundException - Classe non trovata.

2. readAreas(Connection conn)
    - Descrizione: Legge i dati dalla tabella interesting_areas e li trasforma in una lista di oggetti InterestingAreas.
    - Parametri:
        - conn - Connessione al database.
    - Ritorna: Una lista di oggetti InterestingAreas.
    - Eccezioni:
        - SQLException - Errore durante l'esecuzione della query.
3. readStation(Connection conn)
    - Descrizione: Legge i dati dalla tabella centromonitoraggio e li trasforma in una lista di oggetti MonitoringStation.
    - Parametri:
        - conn - Connessione al database.
    - Ritorna: Una lista di oggetti MonitoringStation.
    - Eccezioni:
        - SQLException - Errore durante l'esecuzione della query.
4. readForecast(Connection conn)
    - Descrizione: Legge i dati dalla tabella parametriclimatici e li trasforma in una lista di oggetti Forecast.
    - Parametri:
        - conn - Connessione al database.
    - Ritorna: Una lista di oggetti Forecast.
    - Eccezioni:
        - SQLException - Errore durante l'esecuzione della query.
5. writeUser(User u, Connection conn)
    - Descrizione: Scrive un oggetto User nel database.
    - Parametri:
        - u - Oggetto User da scrivere.
        - conn - Connessione al database.
    - Ritorna: Nessuno.
    - Eccezioni:
        - SQLException - Errore durante l'esecuzione della query.
        - RemoteException - Errore di connessione remota.

6. writeForecast(Forecast f, Connection conn)
    - Descrizione: Scrive un oggetto Forecast nel database.
    - Parametri:
        - f - Oggetto Forecast da scrivere.
        - conn - Connessione al database.
    - Ritorna: Nessuno.

- Eccezioni:
    - SQLException - Errore durante l'esecuzione della query.
    - ClassNotFoundException - Classe non trovata.

7. writeStation(MonitoringStation ms, Connection conn, List<String> areas)
    - Descrizione: Scrive un oggetto MonitoringStation e le relative aree di interesse nel database.
    - Parametri:
        - ms - Oggetto MonitoringStation da scrivere.
        - conn - Connessione al database.
        - areas - Lista di nomi delle aree di interesse.
    - Ritorna: Nessuno.
    - Eccezioni:
        - SQLException - Errore durante l'esecuzione della query.
        - ClassNotFoundException - Classe non trovata.
    - Dettagli di Implementazione
    - Connessioni al Database: La classe utilizza connessioni JDBC per eseguire le operazioni di lettura e scrittura sul database.
    - PreparedStatement: Utilizzato per prevenire SQL Injection e migliorare le performance delle query.

Note

Assicurarsi che le librerie JDBC appropriate siano incluse nel classpath del progetto. Le tabelle nel database devono esistere e avere la struttura corretta come previsto dai metodi. Gestire le eccezioni in modo appropriato per garantire la robustezza dell'applicazione. Questa documentazione fornisce una panoramica delle funzionalità offerte dalla classe DBManager e dei dettagli di implementazione per l'uso efficace e sicuro delle sue capacità di gestione del database.

DatiCondivisi

La classe DatiCondivisi è progettata come un singleton per leggere, memorizzare e gestire i dati necessari per un'applicazione di monitoraggio climatico. Questa classe utilizza JDBC per connettersi a un database PostgreSQL e implementa UnicastRemoteObject per supportare operazioni RMI.

Attributi della classe

instance: Singleton che rappresenta l'istanza unica della classe. Tipo: DatiCondivisi.

monitoringStations: Lista di oggetti MonitoringStation. Tipo: ArrayList<MonitoringStation>.

users: Lista di oggetti User. Tipo ArrayList<User>.

areas: Lista di oggetti InterestingAreas. Tipo: ArrayList<InterestingAreas>.

forecasts: Lista di oggetti Forecast. Tipo: ArrayList<Forecast>.

operatore: L'utente attualmente in sessione. Tipo: User.

conn: Connessione al database PostgreSQL. Tipo: Connection.

dBManager: Istanza della classe DBManager per gestire operazioni sul database. Tipo: DBManager.

JDBC_DRIVER = "org.postgresql.Driver": Nome del driver JDBC. Tipo: String

Metodi Principali della Classe

1. getInstance()
    - Descrizione: Restituisce l'unica istanza della classe DatiCondivisi, creando l'istanza se non esiste.
    - Parametri: Nessuno
    - Complessità: O(1)
2. refresh()
    - Descrizione: Ricarica i dati degli utenti, previsioni e stazioni di monitoraggio dal database.
    - Parametri: Nessuno
    - Complessità: O(n), dove n è la somma del numero di utenti, previsioni e stazioni di monitoraggio.
3. cercaAreaGeografica(String s)
    - Descrizione: Cerca l'area geografica interessata e restituisce un array di stringhe contenenti le città nell'area d'interesse.
    - Parametri:
        - s: la stringa che rappresenta il nome o le coordinate dell'area
    - Complessità: O(n), dove n è il numero di aree di interesse.
4. cercaLimitrofo(double lat1, double lon1)
    - Descrizione: Cerca le città vicine ad un determinato punto dato in latitudine e longitudine.
    - Parametri:
        - lat1: latitudine
        - lon1: longitudine
    - Complessità: O(n), dove n è il numero di aree di interesse.
5. existForecast(String area)
    - Descrizione: Controlla se una determinata area di interesse ha delle rilevazioni.
    - Parametri:
        - area: nome dell'area
    - Complessità: O(n), dove n è il numero di previsioni.
    - Eccezioni:
        - SQLException
6. convertNameToId(String name)
    - Descrizione: Converte il nome di una città nel suo ID corrispondente nel database.
    - Parametri:
        - name: nome della città
    - Complessità: O(1)
    - Eccezioni:
        - SQLException
7. sortAreas()
    - Descrizione: Ordina in modo crescente le aree di interesse.
    - Parametri: Nessuno
    - Complessità: O(n log n), dove n è il numero di aree di interesse.
8. writeForecast(Forecast f)
    - Descrizione: Scrive una previsione climatica nel database.
    - Parametri:
        - f: oggetto Forecast
    - Complessità: O(1)

- Eccezioni: SQLException, ClassNotFoundException

9. writeUser(User u)
    - Descrizione: Scrive un utente nel database.
    - Parametri:
        - u: oggetto User
    - Complessità: O(1)
    - Eccezioni:
        - SQLException, RemoteException
10. writeStation(MonitoringStation ms, List<String> areas)
    - Descrizione: Scrive una stazione di monitoraggio nel database.
    - Parametri:
        - ms: oggetto MonitoringStation
        - areas: lista di aree di interesse
    - Complessità: O(1)
    - Eccezioni: SQLException, RemoteException, ClassNotFoundException

Nota

La complessità degli accessi e delle modifiche agli attributi è O(1), in quanto si tratta di operazioni dirette sugli attributi dell'oggetto.

Classi per la gestione dell'interfaccia grafica

Tra le classi per la gestione dell'interfaccia grafica troviamo principalmente due tipologie: la classe main e le finestre modali che vengono rese visibili durante l'applicazione a seconda del tipo di operazione che un utente deve svolgere.

Classe Menu

La classe Menu estende javax.swing.JFrame e gestisce la finestra del menu principale dell'applicazione.

Descrizione delle Funzionalità

Costruttori

- Menu(): Crea l'oggetto della finestra del menu e inizializza tutti i suoi componenti. Configura la visibilità dei pulsanti in base all'operatore corrente.

Metodi

- grafica(): Configura l'icona e la posizione della finestra.
- initComponents(): Inizializza e configura i componenti dell'interfaccia grafica, inclusi etichette, campi di testo e pulsanti.

Gestione degli Eventi

- jButton2ActionPerformed(evt): Gestisce l'evento di pressione del pulsante per accedere alla finestra di login.
- jButton3ActionPerformed(evt): Gestisce l'evento di pressione del pulsante per accedere alla finestra di registrazione.

- jButton1ActionPerformed(evt): Gestisce l'evento di pressione del pulsante per cercare un'area geografica.
- jList1MouseClicked(evt): Gestisce l'evento di selezione di un elemento dalla lista dei risultati di ricerca.
- jButton4MouseClicked(evt): Gestisce l'evento di pressione del pulsante per accedere al menu dell'operatore.
- jButton5MouseClicked(evt): Gestisce l'evento di pressione del pulsante per disconnettere l'operatore corrente.
- jTextField1KeyTyped(evt): Gestisce l'evento di pressione del tasto invio nel campo di testo per la ricerca.

Variabili di Istanza

Non ci sono variabili di istanza in questa classe.

Componenti Swing

- Etichette (JLabel): jLabel1, jLabel3
- Campi di Testo (JTextField): jTextField1
- Liste (JList): jList1
- Pulsanti (JButton): jButton1, jButton2, jButton3, jButton4, jButton5

Flusso di Esecuzione

1. La finestra del menu viene creata e i componenti vengono inizializzati.
2. L'icona e la posizione della finestra vengono configurate.
3. L'utente può cercare un'area geografica, accedere alla finestra di login o registrazione, o accedere al menu dell'operatore se è già connesso.

Eccezioni Gestite

- ClassNotFoundException: Errore nel caricamento dei driver JDBC.
- SQLException: Errore nella connessione al database o nell'esecuzione della query.
- RemoteException: Errore nelle operazioni di rete.

Dettagli sui Componenti dell'Interfaccia

- **Etichette (JLabel)**: Visualizzano il testo statico.
- **Campi di Testo (JTextField)**: Permette all'utente di inserire il nome dell'area geografica da cercare.
- **Liste (JList)**: Visualizza i risultati della ricerca di un'area geografica.
- **Pulsanti (JButton)**: Avviano il processo di ricerca, accedono alle finestre di login e registrazione, accedono al menu dell'operatore, o disconnettono l'operatore corrente.

Login

La classe `Login` gestisce la finestra di autenticazione dell'applicazione, permettendo agli utenti di effettuare il login tramite e-mail e password.

Descrizione delle Funzionalità

Costruttore

- **Login()**:
  - Inizializza i componenti dell'interfaccia grafica.
  - Imposta l'icona e la posizione della finestra al centro dello schermo.
  - Carica la lista degli utenti tramite il metodo `readUser()` di `ClientHandler`.

Metodo grafica

- Imposta l'icona della finestra e la posiziona al centro dello schermo.

Inizializzazione dei Componenti

- **initComponents()**:
  - Inizializza i componenti dell'interfaccia grafica, inclusi etichette, campi di testo, pulsanti, e altri elementi.
  - Configura le proprietà di base della finestra, come il titolo, le dimensioni e il layout.

Gestione degli Eventi

- **jButton1ActionPerformed**:
  - Gestisce l'evento di pressione del bottone "Accedi".
  - Verifica l'e-mail e la password inserite.
  - Se le credenziali sono valide, effettua l'autenticazione e apre la finestra del menu operatore.
  - Se le credenziali non sono valide, mostra un messaggio di errore.
- **jButton2ActionPerformed**:
  - Gestisce l'evento di pressione del bottone "Cancel".
  - Ritorna al menù principale senza effettuare il login.

Eccezioni

- Il costruttore e i metodi che interagiscono con il database lanciano eccezioni di tipo:
  - `ClassNotFoundException`
  - `SQLException`
  - `RemoteException`

Dettagli sui Componenti dell'Interfaccia

- **Etichette (JLabel)**:
  - Visualizzano il testo statico, come "E-Mail" e "Password".
- **Campi di Testo (JTextField, JPasswordField)**:
  - Permettono all'utente di inserire l'e-mail e la password.

- **Pulsanti (JButton)**:
  - "Accedi": avvia il processo di autenticazione.
  - "Cancel": annulla l'operazione di login e ritorna al menu principale.

Variabili di Istanza

- **users**: Lista degli utenti caricati dal database, utilizzata per verificare le credenziali di login.

Metodi Privati

- **grafica()**: Configura l'icona e la posizione della finestra.
- **initComponents()**: Inizializza e configura i componenti dell'interfaccia grafica.
- **jButton1ActionPerformed(evt)**: Gestisce l'evento di autenticazione.
- **jButton2ActionPerformed(evt)**: Gestisce l'evento di annullamento del login.

Flusso di Esecuzione

1. **Inizializzazione**:
   - La finestra di login viene creata e i componenti vengono inizializzati.
   - L'icona e la posizione della finestra vengono impostate.
   - La lista degli utenti viene caricata dal server.
2. **Interazione con l'Utente**:
   - L'utente inserisce e-mail e password nei campi di testo.
   - Premendo "Accedi", il sistema verifica le credenziali:
     - Se valide, l'utente viene autenticato e viene aperta la finestra del menu operatore.
     - Se non valide, viene mostrato un messaggio di errore.
   - Premendo "Cancel", il sistema ritorna al menu principale senza effettuare il login.

Eccezioni Gestite

- **ClassNotFoundException**: Errore nel caricamento dei driver JDBC.
- **SQLException**: Errore nella connessione al database o nell'esecuzione della query.
- **RemoteException**: Errore nelle operazioni di rete.

Register

La classe `Register` gestisce la finestra di registrazione dell'applicazione, consentendo agli utenti di registrarsi fornendo i loro dati personali.

Descrizione delle Funzionalità

Costruttore

- **Register()**:
  - Inizializza i componenti dell'interfaccia grafica.
  - Imposta l'icona e la posizione della finestra al centro dello schermo.
  - Popola il menu a tendina delle stazioni di monitoraggio.
  - Inizializza il gestore del database (`DBManager`).

Metodo grafica

- Imposta l'icona della finestra e la posiziona al centro dello schermo.

Metodo createComboMonitoringStation

- Popola il menu a tendina delle stazioni di monitoraggio.

Inizializzazione dei Componenti

- **initComponents()**:
  - Inizializza i componenti dell'interfaccia grafica, inclusi etichette, campi di testo, pulsanti, e altri elementi.
  - Configura le proprietà di base della finestra, come il titolo, le dimensioni e il layout.

Gestione degli Eventi

- **jButton1ActionPerformed**:
  - Gestisce l'evento di pressione del bottone "Registrati".
  - Verifica i campi del modulo di registrazione tramite il metodo `controlloCampi()`.
  - Se tutti i campi sono validi, crea un nuovo utente e lo salva nel database.
  - Se ci sono errori nei campi, mostra un messaggio di errore.
- **jButton2ActionPerformed**:
  - Gestisce l'evento di pressione del bottone "Cancel".
  - Ritorna al menù principale senza completare la registrazione.
- **txtDBirthFocusGained** e **txtDBirthFocusLost**:
  - Gestiscono il focus sul campo di testo della data di nascita, mostrando un suggerimento di formato.

Metodi Privati

- **grafica()**: Configura l'icona e la posizione della finestra.
- **createComboMonitoringStation(monitoringStations)**: Popola il menu a tendina delle stazioni di monitoraggio.
- **initComponents()**: Inizializza e configura i componenti dell'interfaccia grafica.
- **jButton1ActionPerformed(evt)**: Gestisce l'evento di registrazione.
- **jButton2ActionPerformed(evt)**: Gestisce l'evento di annullamento della registrazione.
- **txtDBirthFocusGained(evt)**: Gestisce il focus acquisito sul campo di testo della data di nascita.
- **txtDBirthFocusLost(evt)**: Gestisce il focus perso sul campo di testo della data di nascita.
- **controlloCampi()**: Verifica la validità dei campi di registrazione e restituisce eventuali errori.

Variabili di Istanza

- **DBManager dBManager**: Gestore del database per le operazioni di salvataggio degli utenti.
- **Componenti Swing**:
  - Etichette (`JLabel`): Per visualizzare il testo statico, come "Nome", "Cognome", "Email", ecc.
  - Campi di Testo (`JTextField`, `JPasswordField`): Per l'inserimento dei dati da parte dell'utente.
  - Pulsanti (`JButton`): "Registrati" e "Cancel" per gestire le azioni dell'utente.

- Menu a Tendina (`JComboBox<String>`): Per selezionare la stazione di monitoraggio.
- Gruppo di Pulsanti (`ButtonGroup`): Per gestire la selezione del sesso.

Flusso di Esecuzione

1. **Inizializzazione**:
   - La finestra di registrazione viene creata e i componenti vengono inizializzati.
   - L'icona e la posizione della finestra vengono impostate.
   - Il menu a tendina delle stazioni di monitoraggio viene popolato.
   - Il gestore del database viene inizializzato.
2. **Interazione con l'Utente**:
   - L'utente inserisce i propri dati nei campi di testo.
   - Premendo "Registrati", il sistema verifica i campi tramite `controlloCampi()`:
     - Se validi, crea un nuovo utente e lo salva nel database.
     - Se non validi, mostra un messaggio di errore.
   - Premendo "Cancel", il sistema ritorna al menu principale senza completare la registrazione.

Eccezioni Gestite

- **ClassNotFoundException**: Errore nel caricamento dei driver JDBC.
- **SQLException**: Errore nella connessione al database o nell'esecuzione della query.
- **RemoteException**: Errore nelle operazioni di rete.
- **ParseException**: Errore nel parsing della data di nascita.

Dettagli sui Componenti dell'Interfaccia

- **Etichette (JLabel)**:
  - Visualizzano il testo statico, come "Nome", "Cognome", "Data di nascita", "Comune Nascita", "Provincia", "Sesso", "Email", "NickName", "Password", "Centro di monitoraggio di afferenza".
- **Campi di Testo (JTextField, JPasswordField)**:
  - Permettono all'utente di inserire i propri dati personali.
- **Pulsanti (JButton)**:
  - "Registrati": Avvia il processo di registrazione.
  - "Cancel": Annulla l'operazione di registrazione e ritorna al menu principale.
- **Menu a Tendina (JComboBox)**:
  - Permette di selezionare la stazione di monitoraggio di afferenza.

AddNotes

La classe AddNotes gestisce la finestra dell'applicazione che consente agli utenti di aggiungere note relative alle rilevazioni meteorologiche. Questa finestra permette l'inserimento di varie informazioni come vento, umidità, pressione, temperatura, precipitazioni, altitudine e massa dei ghiacciai.

Descrizione delle Funzionalità

Costruttore

AddNotes()

- Inizializza i componenti dell'interfaccia grafica tramite il metodo initComponents().

AddNotes(String idCitta, String nomeStazione, String date, String time, int wind, int humidity, int pressure, int temperature, int rainfall, int glacierAltitude, int massGlaciers) throws ParseException

- Inizializza i componenti dell'interfaccia grafica tramite il metodo initComponents().
- Chiama il metodo grafica() per configurare l'icona e la posizione della finestra.
- Crea un oggetto Forecast con i dati forniti.

Metodo grafica

grafica()

- Configura l'icona della finestra.
- Imposta la posizione della finestra al centro dello schermo.

Inizializzazione dei Componenti

initComponents()

- Inizializza i componenti dell'interfaccia grafica, inclusi etichette, campi di testo e pulsanti.
- Configura le proprietà di base della finestra, come il titolo, le dimensioni e il layout.

Gestione degli Eventi

jButton1ActionPerformed(evt)

- Gestisce l'evento di pressione del pulsante "Inserisci la rilevazione".
- Aggiorna i dati di rilevazione nel campo temp con i valori inseriti dall'utente.
- Invoca il metodo remoto writeForecast(temp) per salvare i dati nel database.
- Passa al MenuOperatore e chiude la finestra corrente.

txtVentoActionPerformed(evt), txtUmiditaActionPerformed(evt), txtTempActionPerformed(evt), txtPrecActionPerformed(evt), txtPressActionPerformed(evt), txtAltActionPerformed(evt), txtMassActionPerformed(evt)

- Gestiscono l'evento di azione sui campi di testo.
- Limitano la lunghezza del testo inserito a 256 caratteri.

txtVentoKeyPressed(evt)

- Gestisce l'evento di pressione dei tasti sul campo di testo del vento.
- Limita la lunghezza del testo inserito a 256 caratteri.

Metodi Privati

- grafica(): Configura l'icona e la posizione della finestra.
- initComponents(): Inizializza e configura i componenti dell'interfaccia grafica.

- jButton1ActionPerformed(evt): Gestisce l'evento di inserimento della rilevazione.
- txtVentoActionPerformed(evt), txtUmiditaActionPerformed(evt), txtTempActionPerformed(evt), txtPrecActionPerformed(evt), txtPressActionPerformed(evt), txtAltActionPerformed(evt), txtMassActionPerformed(evt): Gestiscono l'evento di azione sui campi di testo.

Variabili di Istanza

- Forecast temp: Oggetto che rappresenta le previsioni meteorologiche.
- DBManager dBManager: Gestore del database per le operazioni di salvataggio delle rilevazioni.
- Componenti Swing:
  - Etichette (JLabel): lblTemp, lblPrec, lblAlt, lblUm, lblPress, lblMassa, jLabel1, jLabel2
  - Campi di Testo (JTextField): txtVento, txtUmidita, txtTemp, txtPrec, txtPress, txtAlt, txtMass
  - Pulsante (JButton): jButton1

Flusso di Esecuzione

1. **Inizializzazione:**
   - La finestra di aggiunta note viene creata e i componenti vengono inizializzati.
   - L'icona e la posizione della finestra vengono impostate.
   - Se viene utilizzato il costruttore con parametri, viene creato un oggetto Forecast con i dati di rilevazione forniti.
2. **Interazione con l'Utente:**
   - L'utente inserisce i dati nei campi di testo.
   - Premendo "Inserisci la rilevazione", il sistema aggiorna i dati di rilevazione e li salva nel database tramite il metodo remoto writeForecast(temp).
   - Il sistema passa al MenuOperatore e chiude la finestra corrente.

Eccezioni Gestite

- ParseException: Errore nella scrittura della data o dell'ora.
- RemoteException: Errore nelle operazioni di rete.
- SQLException: Errore nella connessione al database o nell'esecuzione della query.

Dettagli sui Componenti dell'Interfaccia

- **Etichette (JLabel):**
  - Visualizzano il testo statico, come "Temperatura", "Precipitazioni", "Altitudine dei ghiacciai", "Vento", "Umidità", "Pressione", "Massa dei ghiacciai", "Inserisci una nota".
- **Campi di Testo (JTextField):**
  - txtVento, txtUmidita, txtTemp, txtPrec, txtPress, txtAlt, txtMass: Permettono all'utente di inserire i dati di rilevazione.
- **Pulsante (JButton):**
  - jButton1: Avvia il processo di inserimento della rilevazione.

CreateMonitoringStation

La classe CreateMonitoringStation gestisce la finestra dell'applicazione che consente agli utenti di creare una nuova stazione di monitoraggio. Questa finestra permette l'inserimento del nome della stazione, dell'indirizzo e la selezione delle aree di monitoraggio.

Descrizione delle Funzionalità

Costruttori

- **CreateMonitoringStation():**
  - Crea l'oggetto della finestra dove verrà creata una nuova stazione di monitoraggio e inizializza tutti i suoi componenti.
  - Legge le aree di monitoraggio dal database e popola il menu a tendina delle aree.
  - Inizializza l'oggetto User.
  - Inizializza la lista delle aree.
- **CreateMonitoringStation(User u):**
  - Crea l'oggetto della finestra dove verrà creata una nuova stazione di monitoraggio e inizializza tutti i suoi componenti.
  - Legge le aree di monitoraggio dal database e popola il menu a tendina delle aree.
  - Inizializza l'oggetto User con le credenziali dell'operatore.
  - Inizializza la lista delle aree.

Metodo grafica

- **grafica():**
  - Configura l'icona della finestra.
  - Imposta la posizione della finestra al centro dello schermo.

Inizializzazione dei Componenti

- **initComponents():**
  - Inizializza e configura i componenti dell'interfaccia grafica, inclusi etichette, campi di testo e pulsanti.
  - Configura le proprietà di base della finestra, come il titolo, le dimensioni e il layout.
  - Popola il menu a tendina delle aree di monitoraggio.

Gestione degli Eventi

- **buttonAddActionPerformed(evt):**
  - Gestisce l'evento di pressione del pulsante "Registrati".
  - Verifica che tutti i campi siano compilati.
  - Aggiorna le informazioni della stazione di monitoraggio nel database.
  - Passa al Menu e chiude la finestra corrente.
- **buttonAdd1ActionPerformed(evt):**
  - Gestisce l'evento di pressione del pulsante "Cancel".
  - Passa al Menu e chiude la finestra corrente.

- **btnAggiungiActionPerformed(evt):**
  - Gestisce l'evento di pressione del pulsante "Aggiungi area".
  - Aggiunge l'area selezionata alla lista delle aree di monitoraggio.
  - Visualizza un messaggio di conferma.

Variabili di Istanza

- **User current:** Oggetto che rappresenta l'operatore corrente.
- **List<String> areas:** Lista delle aree di monitoraggio.
- **DBManager dBManager:** Gestore del database per le operazioni di salvataggio delle stazioni di monitoraggio.

Componenti Swing

- **Etichette (JLabel):**
  - jLabel1, jLabel2, jLabel3, jLabel4
- **Campi di Testo (JTextField):**
  - name, address
- **Menu a Tendina (JComboBox):**
  - InterestingAreas
- **Pulsanti (JButton):**
  - buttonAdd, buttonAdd1, btnAggiungi

Flusso di Esecuzione

Inizializzazione:

1. La finestra per la creazione di una nuova stazione di monitoraggio viene creata e i componenti vengono inizializzati.
2. L'icona e la posizione della finestra vengono configurate.
3. Le aree di monitoraggio vengono lette dal database e popolate nel menu a tendina.

Interazione con l'Utente:

1. L'utente inserisce il nome, l'indirizzo e seleziona le aree di monitoraggio.
2. Premendo "Registrati", il sistema verifica che tutti i campi siano compilati.
3. Se i campi sono compilati correttamente, le informazioni sulla stazione di monitoraggio vengono aggiornate nel database.
4. Il sistema passa al Menu e chiude la finestra corrente.
5. Premendo "Cancel", il sistema passa al Menu e chiude la finestra corrente.

Eccezioni Gestite

- ClassNotFoundException: Errore nel caricamento dei driver JDBC.
- SQLException: Errore nella connessione al database o nell'esecuzione della query.
- RemoteException: Errore nelle operazioni di rete.

Dettagli sui Componenti dell'Interfaccia

Etichette (JLabel):

Visualizzano il testo statico, come "Nome", "Indirizzo", "Elenco aree" e "Registra centro aree".

Campi di Testo (JTextField):

- name, address: Permettono all'utente di inserire il nome e l'indirizzo della stazione di monitoraggio.

Menu a Tendina (JComboBox):

- InterestingAreas: Permette di selezionare le aree di monitoraggio.

Pulsanti (JButton):

- buttonAdd: Avvia il processo di registrazione.
- buttonAdd1: Annulla l'operazione di registrazione e ritorna al Menu.
- btnAggiungi: Aggiunge l'area selezionata alla lista delle aree di monitoraggio.

MenuOperatore

La classe MenuOperatore estende javax.swing.JFrame e gestisce la finestra dell'applicazione che consente agli operatori di visualizzare e aggiungere rilevazioni meteorologiche. Questa finestra permette la visualizzazione delle rilevazioni esistenti e l'inserimento di nuove rilevazioni.

Descrizione delle Funzionalità

Costruttori

- MenuOperatore(): Crea l'oggetto della finestra dove si andrà ad aggiungere una rilevazione, riempie la combobox con i relativi dati e popola la tabella con i dati salvati in precedenza. Inizializza l'oggetto DBManager.

Metodi

- grafica(): Configura l'icona e la posizione della finestra.
- refreshTable(String area, String stazione): Aggiorna la tabella delle rilevazioni in base all'area e alla stazione specificate.
- createComboMonitoringStation(): Legge le stazioni di monitoraggio dal database e popola il menu a tendina delle stazioni.
- initComponents(): Inizializza e configura i componenti dell'interfaccia grafica, inclusi etichette, campi di testo e pulsanti.

Variabili di Istanza

- List<Forecast> f: Lista delle previsioni.
- DBManager dBManager: Gestore del database per le operazioni di salvataggio delle rilevazioni.

Componenti Swing

- Etichette
  (JLabel): lblWelcome, jLabel1, lblVento, lblUm, lblUmm, lblPress, lblPressi, lblTemp, lblTempe, lblPrec, lblPreci, lblAlt, lblAlti, lblMassa, lblMas, jLabel9
- Campi di Testo (JSlider): sldVento, sldUm, sldPres, sldTemp, sldPrec, sldAlt, sldMassa
- Menu a Tendina (JComboBox): cmbAreas
- Pulsanti (JButton): btnInsert, jButton1, jButton5
- Tabelle (JTable): tblRilevazioni

Flusso di Esecuzione

1. La finestra per la visualizzazione e l'aggiunta di rilevazioni viene creata e i componenti vengono inizializzati.
2. L'icona e la posizione della finestra vengono configurate.
3. Le stazioni di monitoraggio vengono lette dal database e popolate nel menu a tendina.
4. L'utente può visualizzare le rilevazioni esistenti e aggiungere nuove rilevazioni.

Eccezioni Gestite

- ClassNotFoundException: Errore nel caricamento dei driver JDBC.
- SQLException: Errore nella connessione al database o nell'esecuzione della query.
- RemoteException: Errore nelle operazioni di rete.

Dettagli sui Componenti dell'Interfaccia

- **Etichette (JLabel)**: Visualizzano il testo statico, come "Benvenuto", "Vento", "Umidità", "Pressione", "Temperatura", "Precipitazioni", "Altitudine", "Massa".
- **Campi di Testo (JSlider)**: Permettono all'utente di inserire i valori delle rilevazioni.
- **Menu a Tendina (JComboBox)**: Permette di selezionare l'area di monitoraggio.
- **Pulsanti (JButton)**: Avviano il processo di inserimento di una nuova rilevazione o annullano l'operazione.
- **Tabelle (JTable)**: Visualizzano le rilevazioni esistenti.

SearchResult

La classe SearchResult estende javax.swing.JFrame e gestisce la finestra dell'applicazione che consente agli operatori di visualizzare i risultati di una ricerca di previsioni meteorologiche.

Descrizione delle Funzionalità

Costruttori

- SearchResult(): Crea l'oggetto della finestra dei risultati di ricerca e inizializza tutti i suoi componenti.

- SearchResult(String areaName, Menu me): Crea l'oggetto della finestra dei risultati di ricerca con il nome dell'area di ricerca e il menu associato. Inizializza i componenti, configura la grafica e aggiorna la tabella con i risultati della ricerca.

Metodi

- grafica(): Configura l'icona e la posizione della finestra.
- refreshTable(String areaName): Aggiorna la tabella dei risultati di ricerca in base al nome dell'area specificata.
- initComponents(): Inizializza e configura i componenti dell'interfaccia grafica, inclusi etichette, campi di testo e pulsanti.

Variabili di Istanza

- List<Forecast> f: Lista delle previsioni.
- Menu m: Menu associato alla finestra dei risultati di ricerca.

Componenti Swing

- Etichette (JLabel): Non presenti in questa classe.
- Campi di Testo (JTextField): Non presenti in questa classe.
- Menu a Tendina (JComboBox): Non presenti in questa classe.
- Pulsanti (JButton): jButton20
- Tabelle (JTable): tblRilevazioni

Flusso di Esecuzione

1. La finestra per la visualizzazione dei risultati di ricerca viene creata e i componenti vengono inizializzati.
2. L'icona e la posizione della finestra vengono configurate.
3. I risultati della ricerca vengono letti dal database e popolati nella tabella.
4. L'utente può visualizzare i risultati della ricerca.

Eccezioni Gestite

- ClassNotFoundException: Errore nel caricamento dei driver JDBC.
- SQLException: Errore nella connessione al database o nell'esecuzione della query.

Dettagli sui Componenti dell'Interfaccia

- **Tabelle (JTable)**: Visualizza i risultati della ricerca. Ogni riga rappresenta una previsione meteorologica e ogni colonna rappresenta un attributo della previsione (data, ora, vento, umidità, pressione, temperatura, precipitazioni, altitudine dei ghiacciai, massa dei ghiacciai).

- **Pulsanti (JButton)**: Il pulsante “Torna alla home” permette all’utente di chiudere la finestra dei risultati di ricerca e tornare al menu principale.

Classi per la gestione del sistema distribuito

ServerMain

La classe ServerMain estende UnicastRemoteObject e implementa l’interfaccia ServerInterface. Questa classe rappresenta il server principale dell’applicazione.

Attributi della classe

- PORT: Costante che rappresenta la porta su cui il server è in ascolto. Tipo: int.

Metodi Principali della Classe

1. main(String[] args): Avvia il server e lo rende disponibile per le connessioni remote. Complessità: O(1).
2. readUser(): Legge la lista degli utenti dal database. Complessità: O(n), dove n è il numero di utenti.
3. readAreas(): Legge la lista delle aree di interesse dal database. Complessità: O(n), dove n è il numero di aree di interesse.
4. readStation(): Legge la lista delle stazioni di monitoraggio dal database. Complessità: O(n), dove n è il numero di stazioni di monitoraggio.
5. readForecast(): Legge la lista delle previsioni dal database. Complessità: O(n), dove n è il numero di previsioni.
6. sortAreas(): Ordina in modo crescente le aree di interesse. Complessità: O(n log n), dove n è il numero di aree di interesse.
7. cercaAreaGeografica(String a): Cerca un’area geografica nel database e restituisce un array di stringhe contenenti le città nell’area d’interesse. Complessità: O(n), dove n è il numero di aree di interesse.
8. existForecast(String a): Controlla se esistono previsioni per un’area specifica. Complessità: O(n), dove n è il numero di previsioni.
9. refresh(): Aggiorna i dati condivisi. Complessità: O(n), dove n è la somma del numero di utenti, previsioni e stazioni di monitoraggio.
10. writeForecast(Forecast f): Scrive una previsione nel database. Complessità: O(1).
11. writeUser(User u): Scrive un utente nel database. Complessità: O(1).
12. writeStation(MonitoringStation ms, List<String> areas): Scrive una stazione di monitoraggio nel database. Complessità: O(1).
13. convertNameToId(String name): Converte il nome di un’area nel suo ID corrispondente. Complessità: O(1).

14. normalizeStrings(String s): Normalizza una stringa rimuovendo i caratteri diacritici. Complessità: O(1).

Nota La complessità degli accessi e delle modifiche agli attributi è O(1), in quanto si tratta di operazioni dirette sugli attributi dell'oggetto. Le eccezioni gestite
includono ClassNotFoundException, SQLException e RemoteException

ClientHandler

La classe ClientHandler estende UnicastRemoteObject e gestisce la connessione del client al server. Questa classe è progettata come un singleton per garantire che esista una sola istanza di ClientHandler.

Attributi della classe

- PORT: Costante che rappresenta la porta su cui il client si connette al server. Tipo: int.
- stub: Riferimento all'interfaccia del server per l'invocazione dei metodi remoti. Tipo: ServerInterface.
- instance: Singleton che rappresenta l'istanza unica della classe. Tipo: ClientHandler.

Metodi Principali della Classe

1. ClientHandler(): Costruttore della classe. Crea una connessione remota con il server. Complessità: O(1).
2. getStub(): Restituisce il riferimento all'interfaccia del server. Complessità: O(1).
3. getInstance(): Restituisce l'unica istanza della classe ClientHandler, creando l'istanza se non esiste. Complessità: O(1).

Nota La complessità degli accessi e delle modifiche agli attributi è O(1), in quanto si tratta di operazioni dirette sugli attributi dell'oggetto. Le eccezioni gestite
includono RemoteException, NotBoundException e InterruptedException.