Laboratorio interdisciplinare A

Climate Monitoring

Un sistema di monitoraggio di parametri climatici

CATTANEO LUCA

FICARA PAOLO  
MAURI ANDREA

Indice

[Introduzione 2](#_Toc168605302)

[Librerie esterne utilizzate 2](#_Toc168605303)

[ codice-fiscale-java-master 2](#_Toc168605304)

[STRUTTURA DEL DATABASE 2](#_Toc168605305)

[TABELLE PRINCIPALI 2](#_Toc168605306)

[parametriclimatici 2](#_Toc168605307)

[coordinatemonitoraggio 3](#_Toc168605308)

[centromonitoraggio 3](#_Toc168605309)

[operatoreregistrato 3](#_Toc168605310)

[lavora 3](#_Toc168605311)

[RELAZIONI 3](#_Toc168605312)

[parametriclimatici - coordinatemonitoraggio: 3](#_Toc168605313)

[parametriclimatici - centromonitoraggio: 3](#_Toc168605314)

[operatoreregistrato - centromonitoraggio: 3](#_Toc168605315)

[lavora - coordinatemonitoraggio e centromonitoraggio: 4](#_Toc168605316)

[Struttura generale del sistema di classi 4](#_Toc168605317)

[Gestione dei dati e rappresentazioni oggetti 4](#_Toc168605318)

[Gestione dell’interfaccia grafica 4](#_Toc168605319)

[Classi per la gestione dei dati 4](#_Toc168605320)

[InterestingAreas 4](#_Toc168605321)

[Metodi principali della classe: 4](#_Toc168605322)

[MonitoringStation 5](#_Toc168605323)

[User 5](#_Toc168605324)

[Metodi principali della classe: 5](#_Toc168605325)

[Forecast 6](#_Toc168605326)

[Metodi della classe: 6](#_Toc168605327)

[FileManager 6](#_Toc168605328)

[Metodi principali della classe: 6](#_Toc168605329)

[DatiCondivisi 8](#_Toc168605330)

[Metodi principali della classe: 8](#_Toc168605331)

[Classi per la gestione dell’interfaccia grafica 9](#_Toc168605332)

[Menu(classe main) 9](#_Toc168605333)

[Eccezioni 10](#_Toc168605334)

[Login 10](#_Toc168605335)

[Complessità 10](#_Toc168605336)

[Register 10](#_Toc168605337)

[AddNotes 11](#_Toc168605338)

[Complessità 11](#_Toc168605339)

[CreateMonitoringStation 11](#_Toc168605340)

[Complessità 12](#_Toc168605341)

[MenuOperatore 12](#_Toc168605342)

[Complessità 13](#_Toc168605343)

[SearchResult 13](#_Toc168605344)

# 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.

1. 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.

1. 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.

1. 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.

1. 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.

1. 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.

1. 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.

# Menu(classe main)

Menu è la finestra main dell’applicazione in quanto presenta tutte le operazioni che un utente può compiere:

* Ricerca di una città:
  + Con una JTextField viene prelevato il teste generato dall’utente che indica un’area di interesse o delle coordinate in cui va fatta la ricerca. Per questa casella di testo è presente un metodo che gestisce l’evento della pressione del tasto “Enter” che richiama l’evento del bottone;
  + Una ListBox per mostrare le città all’interno dell’area di interesse denominata le quali possono essere selezionate per mostrare le rilevazioni fatte su quel paese;
  + Un bottone denominato “Cerca” riordina la lista della classe DatiCondivisi per migliorare la ricerca e poi viene effettuata tramite il metodo “cercaAreaGeografica” della classe DatiCondivisi e viene immesso come parametro proprio la JTextField dove l’utente scrive l’area di interesse poi rende visibili la ListBox se il metodo “cercaAreaGeografica” restituisce un vettore non vuoto.
* Login/Logout:
  + Il login viene effettuato con un bottone che rende visibile la finestra modale per l’apposita funzione.
  + Il logout, come la login, viene eseguito con un bottone che imposta l’operatore della classe DatiCondivisi a null, rende nascosti i bottoni per immettere una rilevazione, della medesima logout e rende visibili quelli della registrazione e della login.
* Registrazione:
  + La registrazione viene effettuata come la login cioè un bottone apposito apre la finestra modale adibita all’operazione.
* Torna al menù operatore:
  + Un bottone, mostrato solo se fatta la login da parte di un operatore, che rende visibili la finestra modale dove vengono inserite le rilevazioni;

### Eccezioni

Ogni metodo lancia delle eccezioni di parsing o di input/output solo se ci sono errori nelle informazione nei file.

# Login

La login page è una finestra modale in cui un operatore accede ed entra ufficialmente in sessione.

* JTextFiels:
  + Email: casella di testo dove digitare la mail dell’operatore immessa in fase di registrazione
  + Password: password abbinata alla mail immessa dall’operatore in fase di registrazione
* JButton:
  + Accedi: se entrambi i campi sono stati impostati correttamente viene effettuata una ricerca all’interno della lista utenti della classe DatiCondivisi e una volta trovato quello interessato, viene impostato l’operatore del singleton come quello trovato e viene resa visibile la finestra modale per l’inserimento delle rilevazioni;
  + Cancel: ritorna alla finestra del menu principale

### Complessità

Il bottone accedi ha una complessità di O(n) nel caso peggiore dove n è il numero di utenti già registrati.

# Register

La Register page è la finestra modale che effettua la registrazione per un nuovo operatore fornendo le credenziali necessarie ai suoi fini.

* JTextField
  + Nome
  + Cognome
  + Provincia
  + Data di nascita
  + Comune di nascita
  + Email
  + Nickname
  + Password
* JRadioButton
  + Sesso
* JComboBox
  + Centro di monitoraggio di afferenza: comboBox creata dinamicamente all’avvio della finestra e viene riempita grazie alla lista delle stazioni di monitoraggio della classe DatiCondivisi
* JButton
  + Registrati: se ogni campo viene compilato correttamente allora dati i campi: nome, cognome, data di nascita, comune di nascita e sesso viene generato il codice fiscale proprio della persona e se la ComboBox ha un elemento selezionato allora viene scritto su file l’operatore appena registrato e viene aperta la finestra del menù principale, se non ha un elemento selezionato si apre un’altra finestra modale in cui viene creato un nuovo centro di monitoraggio.
  + Cancel: ritorna al menù principale

# AddNotes

La AddNotes page viene aperta nel momento in cui si vuole inserire una nuova misurazione. La pagina permette di aggiungere una nota per ogni parametro della misurazione. Infine premendo il bottone “Inserisci la rilevazione” viene inserita la rilevazione con le reciproche note.

* JTextField
  + Vento
  + Umidità
  + Pressione
  + Temperatura
  + Precipitazioni
  + Altitudine dei ghiacciai
  + Massa dei ghiacciai
* JButton
  + Inserisci la rilevazione: i campi non sono obbligatori, se vengono completati le note vengono associate ai dati inseriti in precedenza, altrimenti la nota rimane vuota. Fatto ciò si viene riportati al menu operatore.

### Complessità

Il bottone “Inserisci la rilevazione” ha una complessità di O(n) nel caso peggiore dove vengono inseriti tutti e sette i campi.

# CreateMonitoringStation

Questa pagina viene aperta quando durante la fase di registrazione l’utente non seleziona un centro di monitoraggio di afferenza. In questa pagina l’utente può dare un nome al suo centro di afferenza e associarci una o più aree di interesse. Infine, quando viene premuto il bottone registrati si chiude la pagina attuale e si torna al menù principale con l’accesso già effettuato.

* JTextField
  + Nome
  + Indirizzo
* JComboBox
  + Elenco aree,
* JButton
  + Aggiungi area, questo bottone permette di aggiungere l’area selezionata nel JComboBox al centro di afferenza dell’utente e successivamente svuotare la JComboBox
  + Registrati, se è stata selezionata almeno un area di interesse da associare al centro di afferenza dell’utente si termina la registrazione e si torna al menù principale con l’accesso già fatto
  + Chiudi, una volta premuto questo bottone si ritorna al menù principale annullando la registrazione e i dati inseriti in precedenza

### Complessità

Il bottone “Inserisci la rilevazione” ha una complessità di O(n+m), dove n è la dimensione delle liste "areas" e m è la lunghezza della stringa "partialInfo", nel caso peggiore dove un utente associa al suo centro di afferenza tutte le aree di interesse.

# MenuOperatore

Questa pagina viene aperta una volta che un operatore effettua l’accesso dal menù principale. Sulla sinistra ha la possibilità di selezionare una delle aree di interesse che fanno parte del suo centro di afferenza e ha la possibilità di inserire una nuova misurazione tramite gli appositi slider. Nella parte destra ha una tabella contenente tutte le misurazioni da lui inserite.

* JComboBox
  + Area di interesse
* JSlider
  + Vento
  + Umidità
  + Pressione
  + Temperatura
  + Precipitazioni
  + Altitudine dei ghiacciai
  + Massa dei ghiacciai
* JTable
  + Misurazioni, cliccando su ogni cella verrà visualizzata la nota relativa alla rilevazione presente. Il pop-up viene mostrato indipendentemente dalla presenza di note, in contenuto varierà in base alla stringa contenuta nel file.
* JButton
  + Inserisci una nuova rilevazione, se nella JComboBox è stata selezionata un’area di interesse vengono presi i valori degli JSlider e viene aperta la pagina per aggiungere le note dopo la quale viene inserita la misurazione.
  + Torna alla home, il bottone permette di tornare al Menù principale mantenendo l’accesso da operatore
  + Logout, il bottone permette di tornare al Menù principale effettuato il logout

## Complessità

Le complessità più alte sono O(n) dei metodi createComboMonitoringStation() e refreshTable(String area, String stazione). Sulla base di queste stime approssimative, possiamo dire che la complessità è O(n) dove n è il numero di elementi nella lista temp o monitoringStations.

# SearchResult

SearchResult è una finestra modale che mostra i risultati di ricerca in una tabella di tutte le rilevazioni di quella determinata area di interesse.

Tramite un metodo “refreshTable” viene inizializzata la tabella e la lista di Forecast attributo privato della classe che rappresenta tutte le rilevazioni di quella determinata area di interesse.

* JButton
  + Torna alla home: bottone che chiude la finestra corrente e rende visibili quella del menù principale
* JTable
  + Evento MouseClicked: in caso venga cliccato un campo della tabella di una determinata rilevazione, viene mostrata la nota annessa a quel campo di quella previsione selezionata grazie ad uno switch che in base alla colonna selezionata, preleva la nota.

Complessità

L’unico metodo che potrebbe peggiorare le prestazioni è il “refreshTable” in quanto scorre tutta la lista di Forecast che è quella di tutte le rilevazioni fatte fino a quel momento e inizializza sia la tabella sia la lista attributo, quindi il tutto ha complessità in tempo Θ(n) dove n è il numero di rilevazioni.