CLIMATE MONITORING

MANUALE TECNICO

Sviluppato e ideato da:

- **Pichierri Nicola**
- **Scolaro Francesco**
- **Tagliabue Luca**
- **Xulio Elezi**

Università degli studi dell'Insubria

Laboratorio Interdisciplinare B - anno accademico 2023/2024

Sommario

Struttura Classi e Complessita' :

- Accessoimpl;
  - → eseguiLogin(String operator_id, String password);
  - → eseguiRegistrazione(String nome_cognome, String codice_fiscale, String mail, String operator_id, String password);
- AccessoRMI;
  - → eseguiLogin(String operator_id, String password);
  - → eseguiRegistrazione(String nome_cognome, String codice_fiscale, String mail, String operator_id, String password);
- AccessoServer;
  - → main(String[] args);
- CentroMonitoraggioImpl;
  - → registraCentroAree(String nomecentro, String indirizzo, String operator_id);
  - → visualizzaCentriMonitoraggio(String operator_id);
- CentroMonitoraggioRMI;
  - → registraCentroAree(String nomecentro, String indirizzo, String operator_id);
  - → visualizzaCentriMonitoraggio(String operatore);
- ClientUI;
  - → Main():
  - → createHomePanel();
  - → createCoordinateSearchPanel();
  - → createDenominationSearchPanel();
  - → createLoginPanel();
  - → createRegisterPanel();
  - → createWelcomePanel();
  - → createClimateParamsPanel();
  - → createInsertCenterPanel();
  - → isValidEmail(String email);
  - → isValidPassword(String password);
  - → updateWelcomePanel();
  - → createStyledButton(String text);

- ParametriClimaticiImpl;
  - → ParametriClimaticiImpl();
  - → inserisciParametriClimatici(String nomecentro, String geoname, String vento, String umidita, String pressione, String temperatura, String precipitazioni, String altitudineghiacciai, String massaghiacciai);
  - → esisteGeoname(String geoname);

- ParametriClimaticiRMI;
  - → inserisciParametriClimatici(String nomecentro, String geoname, String vento, String umidita, String pressione, String temperatura, String precipitazioni, String altitudineGhiacciai, String massaGhiacciai)

- RicercaImpl;
  - → RicercaImpl()
  - → ricercaPerDenominazione(String denominazione, String stato);
  - → ricercaPerCoordinate(String coordinate)
  - → getParametriClimatici(String geoname);
  - →

- RIcercaRMI;
  - → ricercaPerDenominazione(String denominazione, String stato);
  - → ricercaPerCoordinate(String coordinate);
  - → getParametriClimatici(String geoname);

Basi di dati

Il progetto è basato su Java 17 e postgreSQL ed è stato testato su windows 10 e 11. Struttura Il programma ha una struttura molto semplice in quanto si serve di poche classi e non utilizza classi esterne per funzionare, rendendo il programma leggero e dal facile utilizzo.

Nel file inviato sono presenti gli uml citati in questo manuale e un class diagram che sintetizza tutto il progetto.

AccessoImpl

La classe AccessoImpl estende UnicastRemoteObject e implementa l'interfaccia remota AccessoRMI. Ciò consente a questa classe di essere un'implementazione di un servizio accessibile via RMI per gestire operazioni di accesso remoto come il login e la registrazione.

- Costanti di Connessione al Database

    → DB_URL: Contiene l'URL del database PostgreSQL a cui la classe si connette.
    → USER: Il nome utente del database.
    → PASS: La password per accedere al database.

♦ Costruttore AccessoImpl

Il costruttore richiama super() per esportare l'oggetto remoto, rendendolo accessibile via RMI.

♦ Metodo eseguiLogin

Questo metodo accetta due parametri, l'operator_id e la password, e verifica se esiste un utente corrispondente nel database. Utilizza una query SQL preparata per controllare se le credenziali inserite sono corrette:

→ La query cerca una corrispondenza in una tabella chiamata operatoriregistrati.
→ Se viene trovato un record corrispondente (l'utente esiste e la password è corretta), restituisce true; altrimenti, restituisce false.

Memoria per gli oggetti remoti:

→ Vengono creati quattro oggetti, ciascuno dei quali occupa uno spazio fisso di memoria. Anche qui, il numero di oggetti è fisso (quattro), quindi lo spazio complessivo è costante: O(1).

Per progettare questa classe abbiamo realizzato:

- Un activity Diagram;
- Un sequence Diagram;
- Uno state Diagram;
- Un class Diagram.

AccessoRMI

AccessoRMI definisce un'interfaccia Java per Remote Method Invocation (RMI). RMI è una tecnologia Java che permette a un'applicazione di invocare metodi su oggetti situati in una macchina remota come se fossero locali.

Ecco una spiegazione breve di ciascuna parte

Importazioni:

- `Import java.rmi.Remote;` e `import java.rmi.RemoteException;`: Questi importano le classi necessarie per creare un'interfaccia remota e gestire le eccezioni legate alla comunicazione remota.
- `import java.util.List;` (anche se non viene usata nel codice fornito): Importa la classe `List` dalla libreria Java, che è utile per la manipolazione di collezioni di oggetti.

Definizione dell'interfaccia:

- `public interface AccessoRMI extends Remote`: Questa linea definisce un'interfaccia pubblica chiamata `AccessoRMI` che estende l'interfaccia `Remote`. Per essere utilizzata in RMI, un'interfaccia deve estendere `Remote`.

Metodi dichiarati:

- `boolean eseguiLogin(String operator_id, String password) throws RemoteException;`: Questo metodo permette di effettuare il login, prendendo come

parametri l'ID dell'operatore e la password, e restituisce un valore booleano per indicare il successo o il fallimento dell'operazione. Può lanciare un'eccezione `RemoteException`, che gestisce i problemi legati alla comunicazione remota.

- `boolean eseguiRegistrazione(String nome_cognome, String codice_fiscale, String mail, String operator_id, String password) throws RemoteException;`. Questo metodo gestisce la registrazione di un nuovo utente. Prende come parametri il nome e cognome, il codice fiscale, l'e-mail, l'ID dell'operatore e la password, restituendo un booleano che indica se la registrazione è andata a buon fine o meno. Anche questo metodo può lanciare un'eccezione `RemoteException`.

Per progettare questa classe abbiamo realizzato:

- Un activity Diagram;
- Un sequence Diagram;
- Uno use case Diagram;

AccessoServer

Il codice contenuto in AccessoServer fornisce un'implementazione base di un server RMI (Remote Method Invocation) in Java e si occupa delle seguenti operazioni:

Importazioni:

- `java.rmi.Naming`: Per il binding e la ricerca degli oggetti remoti.
- `java.rmi.registry.LocateRegistry`: Per creare e gestire il registro RMI.

Classe `AccessoServer`:

- È la classe principale responsabile dell'avvio e della configurazione del server RMI.

Metodo `main`:

→ **Creazione e Avvio del Registro RMI**: Usa `LocateRegistry.createRegistry(1099)` per avviare il registro RMI

sulla porta predefinita 1099. Questo registro è essenziale per il funzionamento di RMI poiché gestisce il binding degli oggetti remoti.

→ **Creazione e Registrazione degli Oggetti Remoti**:

- **`AccessoRMI`**: Un oggetto `AccessoImpl` viene creato e registrato con il nome "rmi://localhost/AccessoRMI". Gli oggetti remoti sono accessibili attraverso questo nome.

- **`RicercaRMI`**: Un oggetto `RicercaImpl` viene creato e registrato con il nome "rmi://localhost/RicercaRMI".

- **`CentroMonitoraggioRMI`**: Un oggetto `CentroMonitoraggioImpl` viene creato e registrato con il nome "rmi://localhost/CentroMonitoraggioRMI".

- **`ParametriClimaticiRMI`**: Un oggetto `ParametriClimaticiImpl` viene creato e registrato con il nome "rmi://localhost/ParametriClimaticiRMI".

Ogni oggetto remoto è associato a un nome univoco nel registro RMI per consentire ai client di localizzarlo e invocare i metodi remoti.

→ **Gestione delle Eccezioni**: Se si verifica un'eccezione durante la creazione del registro o il binding degli oggetti, viene catturata e il suo stack trace viene stampato, fornendo informazioni sul problema che si è verificato.

Per progettare questa classe abbiamo realizzato:

- Un activity Diagram;
- Un sequence Diagram;
- Uno state Diagram;

CentroMonitoraggioImpl

La classe `CentroMonitoraggioImpl` permette ai client di interagire con un database per gestire i centri di monitoraggio.

Dichiarazione della Classe:

- `CentroMonitoraggioImpl` estende `UnicastRemoteObject`, che è un modo standard per esportare oggetti remoti in Java RMI.
- Implementa l'interfaccia `CentroMonitoraggioRMI`, che definisce i metodi remoti disponibili per i client.

Costanti:

- DB_URL, USER e PASS definiscono i dettagli di connessione per il database PostgreSQL.

Costruttore:

- Il costruttore chiama `super()` per inizializzare `UnicastRemoteObject`, che è necessario per il funzionamento di RMI.

Metodi:

- **`registraCentroAree`**:
  - **Scopo**: Registra un nuovo centro di monitoraggio nel database.
  - **Funzionamento**:
    - Stabilisce una connessione al database utilizzando i dettagli forniti.
    - Prepara una dichiarazione SQL per inserire i dati del centro di monitoraggio (`nomecentro`, `indirizzo`, `operator_id`).
    - Esegue l'operazione di inserimento.
    - In caso di errore, l'eccezione viene trasformata in una `RemoteException`.
- **`visualizzaCentriMonitoraggio`**:
  - **Scopo**: Recupera e visualizza i centri di monitoraggio associati a un operatore specifico.
  - **Funzionamento**:
    - Stabilisce una connessione al database.
    - Prepara una dichiarazione SQL per selezionare i dati dei centri di monitoraggio basati su `operator_id`.

- Esegue la query e itera sui risultati.
- Per ogni centro, raccoglie le informazioni (nome, indirizzo, geonames) e le aggiunge a una lista.
- Ritorna la lista di centri di monitoraggio al client.

Per progettare questa classe abbiamo realizzato:

- Un sequence Diagram.

CentroMonitoraggioRMI

CentroMonitoraggioRMI crea un'interfaccia RMI (Remote Method Invocation) in Java chiamata CentroMonitoraggioRMI. Questa interfaccia stabilisce un contratto per i metodi che possono essere invocati remotamente da un client.

- registraCentroAree:
  - Parametri:
    - nomecentro (String): Il nome del centro di monitoraggio.
    - indirizzo (String): L'indirizzo del centro di monitoraggio.
    - operator_id (String): L'ID dell'operatore che gestisce il centro.
  - Ritorna: Una lista di array di stringhe (List<String[]>) che rappresentano i dettagli del centro registrato.
  - Eccezioni: Lancia RemoteException in caso di errore nella comunicazione remota.

  Questo metodo è destinato a registrare un nuovo centro di monitoraggio e restituire una rappresentazione dei dettagli registrati.

- visualizzaCentriMonitoraggio:
  - Parametri:
    - operatore (String): L'ID dell'operatore per il quale si vogliono visualizzare i centri.
  - Ritorna: Una lista di array di stringhe (List<String[]>) contenente i dettagli dei centri di monitoraggio associati all'operatore.
  - Eccezioni: Lancia RemoteException in caso di errore nella comunicazione remota.

Questo metodo restituisce i centri di monitoraggio registrati per un determinato operatore.

Per progettare questa classe abbiamo realizzato:

- Uno use case Diagram;
- Un sequence Diagram;

ClientUI

ClientUI si occfupa di gestire l'interfaccia utente (UI) per un client che interagisce con un sistema basato su RMI (Remote Method Invocation). L'interfaccia grafica è costruita usando Swing e utilizza diversi pannelli per gestire login, registrazione, ricerca, e altre funzionalità.

Ecco una breve descrizione di alcuni metodi chiave:

- Main(): Il metodo main avvia l'applicazione client con interfaccia grafica e connessione RMI. Ecco i passaggi principali:
  - → Imposta il Look and Feel: Viene applicato il look and feel di sistema per l'interfaccia.
  - → Connessione ai servizi RMI: Si tentano quattro connessioni ai servizi remoti (AccessoRMI, RicercaRMI, ParametriClimaticiRMI, CentroMonitoraggioRMI). In caso di errore, viene mostrato un messaggio e l'applicazione termina.
  - → Inizializza il layout: Viene configurato un CardLayout per gestire diversi pannelli dell'interfaccia.
  - → Aggiunta dei pannelli: Pannelli come Home, CoordinateSearch, DenominationSearch, e altri vengono creati e aggiunti al layout.
  - → Creazione del frame principale: Viene creato un JFrame, impostato un'icona, aggiunto il pannello principale, e visualizzata l'interfaccia.In sintesi, il metodo gestisce la configurazione dell'interfaccia e le connessioni ai servizi remoti.

- createHomePanel(): Il metodo createHomePanel crea e configura il pannello principale dell'applicazione. Ecco i principali passaggi:
  - → Creazione del Pannello: Crea un JPanel con un layout a griglia (GridLayout) che dispone 5 righe e 1 colonna, con spaziatura tra le righe.
  - → Impostazione dello Sfondo: Imposta il colore di sfondo del pannello su un azzurro chiaro.

→ Aggiunta di Componenti:
    - Label di Benvenuto: Aggiunge un JLabel al centro con il testo "Benvenuto!", in grassetto e con font grande.
    - Pulsanti: Crea e aggiunge quattro JButton con etichette per la ricerca per coordinate, ricerca per denominazione, login, e registrazione. Usa il metodo createStyledButton per uno stile personalizzato.

→ Gestione degli Eventi: Assegna azioni a ciascun pulsante per cambiare il pannello visualizzato utilizzando cardLayout.

→ Restituzione del Pannello: Ritorna il pannello configurato.

- createCoordinateSearchPanel(): Il metodo createCoordinateSearchPanel crea e configura un pannello per la ricerca basata su coordinate. Ecco una sintesi delle sue principali funzioni:
    → Creazione del Pannello: Crea un JPanel con un layout GridBagLayout e uno sfondo azzurro chiaro.
    → Aggiunta dei Componenti:
        - Label: Aggiunge un JLabel con il testo "Inserisci le coordinate:".
        - Campo di Testo: Aggiunge un JTextField per l'inserimento delle coordinate.
        - Pulsanti: Crea e aggiunge due pulsanti: uno per avviare la ricerca (Cerca) e uno per tornare indietro (Indietro).
    → Gestione degli Eventi: Layout dei Componenti: Posiziona i componenti sul pannello utilizzando le constraint di GridBagLayout.
        - Pulsante di Ricerca: Quando cliccato, verifica se il campo di testo non è vuoto e, in caso positivo, effettua una ricerca delle coordinate usando ricercaRMI. Mostra i risultati in un JTextArea all'interno di un JScrollPane se ci sono risultati; altrimenti, mostra un messaggio informativo o di errore.

        - Pulsante di Indietro: Cambia il pannello visualizzato a "Home" utilizzando cardLayout.
    → Layout dei Componenti: Posiziona i componenti sul pannello utilizzando le constraint di GridBagLayout.

- createDenominationSearchPanel(): Il metodo createDenominationSearchPanel crea e configura un pannello per la ricerca basata su denominazione. Ecco una sintesi delle sue funzioni principali:
    - → Creazione del Pannello: Crea un JPanel con layout GridBagLayout e uno sfondo azzurro chiaro.
    - → Aggiunta dei Componenti:
        - Label: Aggiunge due JLabel, uno per "Inserisci la città" e uno per "Inserisci lo stato".
        - Campi di Testo: Aggiunge due JTextField per inserire la città e lo stato.
        - Pulsanti: Crea e aggiunge due pulsanti: "Cerca" e "Indietro".
    - → Gestione degli Eventi:
        - Pulsante di Ricerca: Quando cliccato, verifica se almeno uno dei campi di testo non è vuoto e, in caso positivo, effettua una ricerca per denominazione usando ricercaRMI. Mostra i risultati in un JTextArea all'interno di un JScrollPane se ci sono risultati; altrimenti, mostra un messaggio informativo o di errore.
        - Pulsante di Indietro: Cambia il pannello visualizzato a "Home" utilizzando cardLayout.
    - → Layout dei Componenti:
        - Posiziona le etichette e i campi di testo utilizzando GridBagConstraints.
        - Crea un JPanel inferiore con un FlowLayout per i pulsanti e lo aggiunge al pannello principale.

- createLoginPanel(): Il metodo createLoginPanel crea e configura un pannello per il login dell'utente. Ecco una sintesi delle sue funzioni principali:
    - → Creazione del Pannello: Crea un JPanel con layout GridBagLayout e uno sfondo azzurro chiaro.
    - → Aggiunta dei Componenti:
        - Etichette e Campi di Testo: Aggiunge etichette e campi di testo per l'inserimento di username e password, utilizzando GridBagConstraints per la posizione e l'allineamento.
        - Pulsanti: Crea due pulsanti, uno per il login e uno per tornare alla schermata principale ("Home").
    - → Gestione degli Eventi:
        - Pulsante di Login: Verifica che entrambi i campi non siano vuoti e, se sono compilati, tenta di eseguire il login chiamando il metodo

eseguiLogin del servizio RMI. Se il login ha successo, imposta l'utente corrente, cambia il pannello visualizzato a "Welcome" e aggiorna il pannello di benvenuto. In caso contrario, mostra un messaggio di errore.
    - Pulsante di Indietro: Cambia il pannello visualizzato a "Home".

- createRegisterPanel(): Il metodo createRegisterPanel crea un pannello per la registrazione di un nuovo utente. Ecco una sintesi delle sue funzioni principali:
    - → Creazione del Pannello: Imposta un JPanel con layout GridBagLayout e sfondo azzurro chiaro.
    - → Aggiunta dei Componenti:
        - Campi di Input: Aggiunge etichette e campi di testo per inserire nome e cognome, codice fiscale, email, nome utente e password.
        - Pulsanti: Aggiunge due pulsanti, uno per completare la registrazione e uno per tornare alla schermata principale ("Home").
    - → Gestione degli Eventi:
        - Pulsante di Registrazione: Verifica che tutti i campi siano compilati e validi (email e password). Se i dati sono validi, tenta di registrare l'utente chiamando il metodo eseguiRegistrazione del servizio RMI. Mostra un messaggio di successo o di errore a seconda del risultato della registrazione.
        - Pulsante di Indietro: Cambia il pannello visualizzato a "Home".

- CreateWelcomePanel(): Il metodo createWelcomePanel crea un pannello di benvenuto con le seguenti caratteristiche:
    - → Creazione del Pannello: Configura un JPanel con layout GridLayout e sfondo azzurro chiaro.
    - → Aggiunta dei Componenti:
        - Etichetta di Benvenuto: Visualizza un messaggio di benvenuto al centro del pannello.
        - Pulsanti: Aggiunge quattro pulsanti:
            - "Registra Centro Aree": Mostra il pannello per registrare un centro.

- "Visualizza Centri di Monitoraggio": Recupera e mostra un elenco di centri di monitoraggio tramite una finestra di dialogo.
- "Inserisci Parametri Climatici": Mostra il pannello per inserire parametri climatici.
- "Indietro": Torna al pannello di login.

→ Gestione degli Eventi:

- Pulsante di Registrazione: Cambia al pannello di registrazione dei centri.
- Pulsante di Visualizzazione: Mostra una finestra di dialogo con un elenco dei centri di monitoraggio registrati, se presenti.
- Pulsante di Inserimento Parametri: Passa al pannello di inserimento dei parametri climatici.
- Pulsante Indietro: Torna alla schermata di login.

- CreateClimateParamsPanel: Il metodo createClimateParamsPanel crea un pannello per l'inserimento dei parametri climatici con le seguenti caratteristiche:

→ Creazione del Pannello: Configura un JPanel con layout GridBagLayout e sfondo azzurro chiaro.

→ Aggiunta dei Componenti:

- Campi di Testo: Aggiunge nove JTextField per l'inserimento di parametri climatici, ciascuno con un'etichetta corrispondente (Nome Centro, Geoname, Vento, Umidità, Pressione, Temperatura, Precipitazioni, Altitudine Ghiacciai, Massa Ghiacciai).
- Pulsanti: Aggiunge due pulsanti:
  - "Inserisci Parametri": Inserisce i parametri climatici nel sistema.
  - "Indietro": Torna al pannello di benvenuto.

→ Gestione degli Eventi:

- Pulsante Inserisci Parametri: Raccoglie i dati dai campi di testo, verifica che tutti i campi siano compilati e chiama un metodo RMI per inserire i parametri climatici. Mostra messaggi di successo o errore in base all'esito dell'operazione.
- Pulsante Indietro: Passa al pannello di benvenuto.

- createInsertCenterPanel(): Il metodo createInsertCenterPanel crea un pannello per l'inserimento di un nuovo centro di monitoraggio con le seguenti caratteristiche:

→ Creazione del Pannello: Configura un JPanel con layout GridBagLayout e sfondo azzurro chiaro.

→ Aggiunta dei Componenti:
    - Campi di Testo: Aggiunge tre JTextField per l'inserimento delle informazioni del centro:
        - Nome Centro
        - Indirizzo
        - ID Operatore
    - Pulsanti:
        - "Inserisci Centro": Aggiunge un centro di monitoraggio utilizzando un metodo RMI e mostra un messaggio di successo o errore.
        - "Indietro": Torna al pannello di benvenuto.

→ Gestione degli Eventi:
    - Pulsante Inserisci Centro: Raccoglie i dati dai campi di testo, verifica che tutti i campi siano compilati e chiama il metodo RMI per registrare il centro. Mostra messaggi di successo o errore basati sull'esito dell'inserimento.
    - Pulsante Indietro: Passa al pannello di benvenuto.

- isValidEmail(): Il metodo isValidEmail verifica se una stringa di testo rappresenta un'email valida. Utilizza un'espressione regolare per controllare se l'email segue un formato comune, che include caratteri alfanumerici, +, _, ., - prima di un simbolo @, seguito da un dominio. Restituisce true se l'email è valida e false altrimenti.
- isValidPassword(): Il metodo isValidPassword verifica se una password soddisfa determinati requisiti di sicurezza. Controlla che la password sia lunga almeno 8 caratteri e contenga almeno una lettera maiuscola, un numero e un carattere speciale (come simboli o punteggiatura). Restituisce true se tutti questi criteri sono soddisfatti e false altrimenti.
- updateWelcomePanel(): Il metodo updateWelcomePanel aggiorna il messaggio di benvenuto sul pannello di benvenuto. Se il pannello di benvenuto (welcomePanel) non è nullo, il metodo modifica il testo del primo JLabel del pannello per includere un saluto personalizzato con il nome utente corrente, oppure "Utente" se il nome utente non è disponibile.

- createStyledButton(): Il metodo createStyledButton crea un pulsante con uno stile personalizzato. Imposta il colore di sfondo su blu, il testo su nero, e utilizza il font Arial in grassetto da 14 punti. Rimuove il contorno di focus, aggiunge spazio interno al pulsante, e cambia il cursore a forma di mano quando il pulsante viene passato sopra.

Complessita'

- createLoginPanel():
  - → Compl. Temporale: Costante O(1) poiché il numero di operazioni non dipende dalla dimensione dell'input. La creazione e configurazione dei componenti GUI è un'operazione fissa.
  - → Compl. Spaziale: Costante O(1) in termini di spazio aggiuntivo, poiché i componenti sono creati e aggiunti in modo fisso.
- createRegisterPanel():
  - → Compl. Temporale: Costante O(1). Anche qui, il numero di operazioni non cambia con la dimensione dell'input.
  - → Compl. Spaziale: Costante O(1), simile a createLoginPanel().
- createWelcomePanel():
  - → Compl. Temporale: Costante O(1). Le operazioni per creare e configurare i componenti sono fisse.
  - → Compl. Spaziale: Costante O(1).
- createClimateParamsPanel():
  - → Compl. Temporale: Costante O(1). Anche in questo caso, l'operazione di creazione e configurazione è fissa.
  - → Compl. Spaziale: Costante O(1).
- createInsertCenterPanel():
  - → Compl. Temporale: Costante O(1). Come gli altri metodi di creazione dei pannelli, l'operazione è fissa.
  - → Compl. Spaziale: Costante O(1).
- isValidEmail(String email):
  - → Compl. Temporale: Lineare O(n), dove n è la lunghezza della stringa email. La regex deve scansionare ogni carattere della stringa.
  - → Compl. Spaziale: Costante O(1), poiché la regex non richiede spazio aggiuntivo significativo.
- isValidPassword(String password):
  - → Compl. Temporale: Lineare O(n), dove n è la lunghezza della stringa password. La regex deve scansionare ogni carattere della stringa.
  - → Compl. Spaziale: Costante O(1), simile a isValidEmail.
- updateWelcomePanel():
  - → Compl. Temporale: Costante O(1), poiché si tratta solo di aggiornare un'etichetta esistente.

→ Compl. Spaziale: Costante O(1).

- ❖ createStyledButton(String text):
  - → Compl. Temporale: Costante O(1), poiché la creazione e configurazione di un pulsante è fissa e non dipende dalla dimensione dell'input.
  - → Compl. Spaziale: Costante O(1), in quanto solo uno stile di pulsante viene creato e configurato.

Complessità Complessiva

Compl. Temporale: La maggior parte dei metodi ha una complessità temporale costante, tranne isValidEmail e isValidPassword, che sono lineari rispetto alla lunghezza della stringa di input. Tuttavia, dato che questi metodi sono chiamati raramente rispetto alla creazione di pannelli, l'impatto sulla complessità complessiva è minimo.

Compl. Spaziale: Tutti i metodi hanno una complessità spaziale costante, eccetto per i metodi che manipolano stringhe, che richiedono spazio aggiuntivo per l'input.

Per progettare questa classe abbiamo realizzato:

- Un sequence diagram

ParametriClimaticiImpl

La classe ParametriClimaticiImpl è un'implementazione del servizio RMI che consente di gestire parametri climatici associati a centri di monitoraggio e località geografiche. Le principali funzionalità includono:

- Inserimento dei parametri climatici: Verifica se esistono il centro di monitoraggio e il geoname nel database, e se presenti, inserisce i parametri climatici forniti.
- Verifica del nome del centro: Metodo privato per controllare se un centro di monitoraggio esiste nel database.
- Verifica del geoname: Metodo privato per controllare se una località geografica esiste nel database.

Per progettare questa classe abbiamo realizzato:

- Un sequence diagram;
- Un activity diagram;

InserisciParametriClimaticiRMI

inserisciParametriClimatici(): Questo metodo permette di inserire i parametri climatici di un luogo (nome geografico) e memorizzare informazioni dettagliate come vento, umidità, pressione, temperatura, precipitazioni, altitudine e massa dei ghiacciai. Viene utilizzato principalmente per il monitoraggio e la gestione dei dati climatici da parte di un centro specializzato.

- → Ritorno: Restituisce un valore booleano (true se l'operazione è andata a buon fine, false altrimenti).
- → Eccezioni: Può lanciare un'eccezione RemoteException se ci sono problemi con la comunicazione remota.

Per progettare questa classe abbiamo realizzato:

- Un sequence diagram;

RicercaImpl

La classe RicercaImpl implementa l'interfaccia RicercaRMI per fornire servizi di ricerca remota tramite Java RMI (Remote Method Invocation). Le principali funzionalità della classe includono:

- Ricerca per Denominazione e Stato:
  - → Metodo ricercaPerDenominazione: esegue una query SQL per trovare informazioni geografiche (nome, nome ASCII, paese, coordinate) relative ad una denominazione (nome) e un paese specifico.

- Ricerca per Coordinate:

→ Metodo ricercaPerCoordinate: consente di trovare informazioni geografiche (nome, nome ASCII, paese, coordinate) in base alle coordinate fornite.

- Recupero Parametri Climatici:
  → Metodo getParametriClimatici: esegue una query SQL per recuperare i parametri climatici (vento, umidità, pressione, temperatura, precipitazioni, altitudine e massa dei ghiacciai) relativi a un determinato luogo (geoname).

Complessità

I metodi di questa classe dipendono principalmente dalle operazioni di query e dalle dimensioni delle tabelle nel database. La complessità di ogni metodo segue lo schema O(n + m), dove:

- n è il numero di record nella tabella su cui viene eseguita la query (complessità di ricerca).
- m è il numero di record restituiti dalla query (complessità di recupero).

Per progettare questa classe abbiamo realizzato:

- Un sequence Diagram;
- Un activity Diagram;
- Un use case Diagram.

RicercaRMI

L'interfaccia RicercaRMI definisce un set di operazioni per la ricerca remota di informazioni geografiche e climatiche. Le sue principali funzionalità includono:

→ Ricerca per denominazione: Consente di cercare luoghi specifici in base al nome e allo stato in cui si trovano.
→ Ricerca per coordinate: Consente di cercare luoghi utilizzando le coordinate geografiche.
→ Recupero dei parametri climatici: Restituisce i parametri climatici di un luogo in base al suo nome geografico.

Per progettare questa classe abbiamo realizzato:

- Un sequence diagram;

```
CREATE TABLE geonamesandcoordinates (
   geoname VARCHAR(255),PRIMARY KEY,
   name VARCHAR(255),
   ascii VARCHAR(255),
   country_code VARCHAR(255),
   country_name VARCHAR(255),
   coordinates VARCHAR(255)
);
```

Questa query SQL crea una tabella chiamata geonamesandcoordinates nel database. La tabella è strutturata per contenere informazioni relative a nomi geografici e le loro coordinate. Di seguito è una spiegazione dettagliata di ciascun elemento della query:

Spiegazione dei Componenti

1.CREATE TABLE geonamesandcoordinates:

- Questo comando SQL crea una nuova tabella chiamata geonamesandcoordinates nel database.

2.geoname VARCHAR(255)PRIMARY KEY:

- Questa colonna chiamata geoname è di tipo VARCHAR(255) e funge da chiave primaria della tabella.
- La chiave primaria (PRIMARY KEY) deve essere unica per ogni riga e non può contenere valori nulli. È utilizzata per identificare in modo univoco ogni record nella tabella.

3.name VARCHAR(255):

- Colonna name di tipo VARCHAR con una lunghezza massima di 255 caratteri.
- Questa colonna è progettata per contenere il nome geografico.

4.ascii VARCHAR(255):

- Colonna ascii di tipo VARCHAR con una lunghezza massima di 255 caratteri.
- È presumibilmente usata per memorizzare una versione ASCII del nome, che può essere utile per la ricerca o l'ordinamento in contesti dove i caratteri speciali non sono supportati.

5.country_code VARCHAR(255):

- Colonna country_code di tipo VARCHAR con una lunghezza massima di 255 caratteri.
- Questa colonna è utilizzata per memorizzare il codice del paese (tipicamente un codice ISO a 2 o 3 lettere).

6. country_name VARCHAR(255):

- Colonna country_name di tipo VARCHAR con una lunghezza massima di 255 caratteri.
- Questa colonna memorizza il nome del paese.

7. coordinates VARCHAR(255):

- Colonna coordinates di tipo VARCHAR con una lunghezza massima di 255 caratteri.
- Probabilmente utilizzata per memorizzare le coordinate geografiche, come latitudine e longitudine, in formato testo (ad esempio, "40.7128,-74.0060").

```
CREATE TABLE operatoriregistrati (

    nome_cognome VARCHAR(255),

    codice_fiscale VARCHAR(255),

    mail VARCHAR(255),
```

```
    operator_id VARCHAR(255) PRIMARY KEY,
    password VARCHAR(255)
);
```

Questa query SQL crea una tabella chiamata operatoriregistrati nel database. La tabella è progettata per contenere le informazioni relative agli operatori registrati, come il nome, il codice fiscale, l'indirizzo email, l'ID dell'operatore e la password. Ecco una spiegazione dettagliata di ciascun componente della query:

Spiegazione dei Componenti

1. CREATE TABLE operatoriregistrati:
   - Questo comando SQL crea una nuova tabella chiamata operatoriregistrati nel database.
2. nome_cognome VARCHAR(255):
   - Colonna nome_cognome di tipo VARCHAR con una lunghezza massima di 255 caratteri.
   - Questa colonna è utilizzata per memorizzare il nome e il cognome dell'operatore.
3. codice_fiscale VARCHAR(255):
   - Colonna codice_fiscale di tipo VARCHAR con una lunghezza massima di 255 caratteri.
   - Memorizza il codice fiscale dell'operatore. In Italia, il codice fiscale è generalmente di 16 caratteri, quindi potresti ottimizzare la lunghezza di questa colonna.
4. mail VARCHAR(255):
   - Colonna mail di tipo VARCHAR con una lunghezza massima di 255 caratteri.
   - Questa colonna è progettata per memorizzare l'indirizzo email dell'operatore.
5. operator_id VARCHAR(255) PRIMARY KEY:
   - Colonna operator_id di tipo VARCHAR con una lunghezza massima di 255 caratteri, e funge da chiave primaria della tabella.
   - La chiave primaria (PRIMARY KEY) deve essere unica per ogni riga e non può contenere valori nulli. È utilizzata per identificare in modo univoco ogni operatore registrato nella tabella.

6. password VARCHAR(255):

- Colonna password di tipo VARCHAR con una lunghezza massima di 255 caratteri.
- Memorizza la password dell'operatore. Per motivi di sicurezza, le password dovrebbero essere memorizzate in forma cifrata (hashata) e non in chiaro.

```
CREATE TABLE parametriclimatici (
    nomecentro VARCHAR(255),
    geoname VARCHAR(255),
    vento VARCHAR(255),
    umidita VARCHAR(255),
    pressione VARCHAR(255),
    temperatura VARCHAR(255),
    precipitazioni VARCHAR(255),
    altitudinegiacciai VARCHAR(255),
    massaghiacciai VARCHAR(255),
    PRIMARY KEY (nomecentro, geoname),
    FOREIGN KEY (nomecentro) REFERENCES centrimonitoraggio(nomecentro),
    FOREIGN KEY (geoname) REFERENCES geonamesandcoordinates(geoname)
);
```

Questa query SQL crea una tabella chiamata parametriclimatici progettata per memorizzare parametri climatici associati a centri di monitoraggio e a specifiche posizioni geografiche. La tabella utilizza chiavi primarie composte e chiavi esterne per collegare i dati ad altre tabelle. Ecco la spiegazione dettagliata della query:

Spiegazione dei Componenti

1.CREATE TABLE parametriclimatici:

- Questo comando crea una nuova tabella chiamata parametriclimatici nel database.

2.nomecentro VARCHAR(255):

- Colonna nomecentro di tipo VARCHAR con una lunghezza massima di 255 caratteri.
- Questa colonna memorizza il nome del centro di monitoraggio.

3.geoname VARCHAR(255):

- Colonna geoname di tipo VARCHAR con una lunghezza massima di 255 caratteri.
- Questa colonna memorizza il nome geografico o un identificatore associato alla posizione geografica.

4.vento, umidita, pressione, temperatura, precipitazioni, altitudinegiacciai, massaghiacciai VARCHAR(255):

- Queste colonne memorizzano vari parametri climatici come il vento, l'umidità, la pressione, la temperatura, le precipitazioni, l'altitudine dei ghiacciai, e la massa dei ghiacciai. Tutte sono definite con tipo VARCHAR(255), ma potrebbero essere ottimizzate se si conoscono formati specifici (numeri decimali, interi, ecc.).

5. PRIMARY KEY (nomecentro, geoname):

- Definisce una chiave primaria composta utilizzando le colonne nomecentro e geoname.
- Questa chiave primaria composta assicura che ogni combinazione di nomecentro e geoname sia unica nella tabella, evitando duplicati.

6. FOREIGN KEY (nomecentro) REFERENCES centrimonitoraggio(nomecentro):

- Questa chiave esterna impone un vincolo di integrità referenziale tra nomecentro in parametriclimatici e nomecentro nella tabella centrimonitoraggio.
- Assicura che ogni nomecentro in parametriclimatici esista anche in centrimonitoraggio.

7. FOREIGN KEY (geoname) REFERENCES geonamesandcoordinates(geoname):

- Questa chiave esterna impone un vincolo di integrità referenziale tra geoname in parametriclimatici e geoname nella tabella geonamesandcoordinates.

- Assicura che ogni geoname in parametriclimatici esista anche in geonamesandcoordinates.

```
CREATE TABLE centrimonitoraggio (
    nomecentro VARCHAR(255),
    indirizzo VARCHAR(255),
    operator_id VARCHAR(255)
);
```

Spiegazione dei Componenti

1.CREATE TABLE centrimonitoraggio:

- Questo comando SQL crea una nuova tabella chiamata centrimonitoraggio nel database.

2.nomecentro VARCHAR(255):

- Colonna nomecentro di tipo VARCHAR con una lunghezza massima di 255 caratteri.
- Questa colonna è progettata per contenere il nome del centro di monitoraggio. Potrebbe essere un identificativo unico, quindi potrebbe essere utile definire questa colonna come chiave primaria o con un vincolo di unicità se ogni nome deve essere unico.

3.indirizzo VARCHAR(255):

- Colonna indirizzo di tipo VARCHAR con una lunghezza massima di 255 caratteri,
- Questa colonna memorizza l'indirizzo fisico del centro di monitoraggio.

4. operator_id VARCHAR(255):

- Colonna operator_id di tipo VARCHAR con una lunghezza massima di 255 caratteri.
- Questa colonna è utilizzata per memorizzare l'ID dell'operatore associato al centro di monitoraggio.

Sarebbe utile definire questa colonna come chiave esterna che fa riferimento all'ID degli operatori nella tabella operatoriregistrati.