CLIMATE

MONITORING

Manuale Tecnico

Università degli Studi dell'Insubria – Laurea Triennale in Informatica
Progetto Laboratorio A: Climate Monitoring
Sviluppato da: Federico, Tognali, 748247, VA
Sviluppato da: Gianpaolo, Torino, 746577, VA

SOMMARIO

INTRODUZIONE

Climate Monitoring e' un'applicazione che consiste in un sistema di monitoraggio di parametri climatici fornito da centri di monitoraggio sul territorio internazionale in grado di rendere disponibili, ad operatori ambientali e comuni cittadini, i dati relativi alla propria zona di interesse.

Funzionamento Generale Dell' Applicazione

L'applicazione usa i dati forniti da una tipologia di file :

- **File dati** : file di testo (formato *.dati) che contiene una sequenza di dati specificati di seguito :

    - **Coordinate Monitoraggio :** Contiene tutti i dati riguardanti vari paesi.

      Formattazione :

        - Denominazione Ufficiale
        - Nome stato abbreviato
        - Nome Stato
        - Latitudine
        - Longitudine

        - *Cirò,IT,Italy,39.38297, 17.06377*

    - **Operatori Registrati :** Contiene i dati riguardanti tutti gli operatori già registrati o futuri utenti che si vorranno registrare .

      Formattazione :

        - Nome
        - Cognome
        - Codice Fiscale
        - Indirizzo Di Posta Elettronica
        - Userid
        - Password Per Accedere Al Sistema
        - Centro Di Monitoraggio Di Afferenza (Se Presente, Altrimenti Vedi Creazione Centri Monitoraggio)

        - *Mario;Rossi;RSSMRA00A01B123C;mariorossi@test.com;mrossi;password;1*

    - **Centro Monitoraggio :** Contiene i centri di monitoraggio registrati o che verranno registrati dall utente

Formattazione:

- Numero Centro Monitoraggio
- Nome Centro Monitoraggio
- Indirizzo fisico (via/piazza, numero civico, cap, comune, provincia)
- Elenco aree di interesse di cui l'operatore intende inserire i parametri climatici

- *1;Centro delle belle arie ;Via Roma,10,21010,Roma;Roma*

- **Parametri Climatici** : Continente i parametri per una zona geografica inseriti dall'utente (registrato) .

  Formattazione

  - Numero centro di monitoraggio di appartenenza
  - Area di interesse
  - Data di rilevazione
  - Tipologia di parametro
  - Valore parametro climatico
  - Nota (se inserite)

  - *1;Roma;22/04/2023 18:17:05;Precipitazioni;5;Precipitazioni molto forti*

Struttura generale del sistema di classi

Il progetto e’ stutturato fondamentalmente in 1 ramo di classi le quali sono indipendenti l'una dall'altra salvo quelle che richiedono un login dall’ utente:

- **Core classes :**
    - **CentroAree** :
        - Creata per gestire tutte i centroAree
    - **Climate**
        - Classe Main
    - **Login**
        - Classe login usata per accedere a funzioni speciali nel programma
    - **ParametriClimatici :**
        - Classe per i parametri climatici
    - **Registration :**
        - Classe per gestire la registrazioni di un utente

CLASSES

Verranno presentate ora le core classes nel dettaglio :

- **CentroAree :**
    - **CentroAree**
        - Funzione :
            - *CentroAree(int id, String nome, String indirizzo, String citta)*
        - Costruttore per creare l'oggetto
    - **writeCentri**
        - Funzione :
            - *writeCentri(LinkedList<CentroAree> list)*
        - Funzione che passata la lista fa un check se la lista é vuota, procede a verificare se il file é stato creato e utilizza il bufferedWriter con il fileWriter per scrivere nel file
    - **readCentri**
        - Funzione :
            - *readCentri()*
        - Funzione che crea una linked list, controlla se il file é stato creato oppure no, e legge il file utilizzando il bufferedReader e il fileReader e ritorna la linked list
    - **insertCentro**
        - Funzione :
            - *insertCentro(CentroAree centro)*
        - Funzione che gestisce la lettura e la scrittura utilizzando readCentro e writeCentro
    - **cercaCentro**
        - Funzione :
            - *cercaCentro(int id)*
        - Funzione che cerca centri nel file utilizzato come check per vedere se il centro esiste o no

- **ClimateMonitor**
    - **arrayResponseCreate**
        - Funzione :
            - *arrayResponseCreate(String success, String message)*
        - Funzione per gestire l'hashmap
    - **CercaAreaGeografica:**
        - Funzione :
            - *cercaAreaGeografica(String nomeArea, int scelta)*
            - *cercaAreaGeografica(String lat, String lon, int scelta)*
        - Funzione per cercare l'area geografica tramite nome o latitudine e longitudine, gestita con una scelta
    - **CreateFile:**
        - Funzione :
            - *createFile(String filePath)*
        - funzione per gestire la creazione del file e della sua cartella data
    - **retryWhenEmpty:**
        - Funzione :
            - *retryWhenEmpty(String message, Scanner scanner)*
        - funzione usata per riprovare ogni qualvolta venga lasciato il campo vuoto
    - **tryScannerInt**
        - Funzione :
            - *tryScannerInt(Scanner scanner)*
        - Funzione usata come check nel caso l'utente immetta al posto di un numero una stringa e provochi un eccezione riprova all'infinito

- **readClimateCoordinates**

    - Funzione :
        - *readClimateCoordinates()*
    - Funzione che crea una linked list, controlla se il file è stato creato oppure no, e legge il file utilizzando il bufferedReader e il fileReader e ritorna la linked list

- **visualizzaAreaGeografica:**

    - Funzione :
        - *visualizzaAreaGeografica(String nomeArea)*
    - Funzione che dopo aver letto i parametri climatici crea una media e fa vedere le note associate all'area passata

- **insertCentroArea:**

    - Funzione :
        - *insertCentroArea(int idCentro, boolean autenticato)*
    - Funzione che inserisce un centro area se l'utente durante la registrazione usa un idCentro non trovato nel file

- **menuLogin:**

    - Funzione :
        - *menuLogin(Login login)*
    - Primo menu che vede l'utente una volta avviato il programma

- **menuUtente:**

    - Funzione :
        - *menuUtente(boolean logged)*
    - Visualizza il secondo menù, se l'utente è loggato vengono visualizzate le opzioni aggiuntive

- **Login**
    - **Login:**
        - Funzioni :
            - *Login()*
            - *Login(String userid, String password)*
        - Costruttori per la login class (quello vuoto serve per invocare dopo la funzione login)
    - **isAutenticato:**
        - Funzione :
            - *isAutenticato()*
        - Funzione che returna se l'utente è loggato o no
    - **login:**
        - Funzione :
            - *login()*
        - Funzione che legge il file dei operatori registrati e se trova nella stessa riga sia userid che password corrispondenti assegna true alla variable autenticato

- **ParametriClimatici**
    - **ParametriClimatici:**
        - Funzione :
            - *ParametriClimatici(int id_centro, String areaInteresse, String dataRilevazione, int climate, int score, String notes)*
        - Costruttore per creare l'oggetto
    - **arrayResponseCreate:**
        - Funzione :
            - *arrayResponseCreate(String success, String message)*
        - Funzione che crea un hashmap e formatta il success e il message
    - **dateNow**
        - Funzione :
            - *dateNow()*
        - Funzione che genera l'ora e la data attuale utilizza il formato "dd/MM/yyyy HH:mm:ss"
    - **readParametri**
        - Funzione :
            - *readParametri()*
        - Funzione che crea una linked list, controlla se il file é stato creato oppure no, e legge il file utilizzando il bufferedReader e il fileReader e ritorna la linked list

- **writeParametri**

    - Funzione :
        - *writeParametri(LinkedList<ParametriClimatici> list)*
    - Funzione che passata la lista fa un check se la lista é vuota, procede a verificare se il file é stato creato e utilizza il bufferedWriter con il fileWriter per scrivere nel file

- **inserisciParametriClimatici:**

    - Funzione :
        - *inserisciParametriClimatici(ParametriClimatici parametriClimatici)*
    - Funzione che gestisce la lettura e la scrittura utilizzando readParametri e writeParametri

- **Registration:**
    - **Registration**
        - Funzione :
            - *Registration(String nome, String cognome, String codiceFiscale, String email, String userid, String password, int centroMonitoraggio)*
        - Costruttore per creare l'oggetto:
    - **readRegistrati:**
        - Funzione :
            - *LinkedList<Registration> readRegistrati()*
        - Funzione che crea una linked list, controlla se il file è stato creato oppure no, e legge il file utilizzando il bufferedReader e il fileReader e ritorna la linked list
    - **writeRegistrati:**
        - Funzione :
            - *writeRegistrati(LinkedList<Registration> list)*
        - Funzione che passata la lista fa un check se la lista è vuota, procede a verificare se il file è stato creato e utilizza il bufferedWriter con il fileWriter per scrivere nel file
    - **registrazione:**
        - Funzione :
            - *registrazione(Registration utenteRegistrato)*
        - Funzione che gestisce la lettura e la scrittura utilizzando readRegistrati e writeRegistrati e controlla se lo userId é giá presente nel file

MANTENIMENTO DELLA SESSIONE DURANTE IL PROGRAMMA:

Abbiamo utilizzato una classe Apposita per il login in modo da poterla estendere qualsiasi volta avessimo una classe che dovesse usufruire dell'essere autenticato per eseguire una funzione

Viene passato nel menú principale l'oggetto login in modo da poter verificare se l'utente é stato autenticato o no