Climate Monitoring

Manuale Tecnico

Galimberti Riccardo
Paredi Giacomo
Radice Lorenzo

Università degli Studi dell'Insubria
Como
Laurea Triennale in Informatica

Versione: 1.0.0
Data Rilascio: 20/01/2024

Indice

Introduzione

Descrizione

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

Funzionamento dell'applicazione

Gli utenti che potranno utilizzare l'applicazione sono divisi in due categorie: operatori autorizzati e comuni cittadini. Gli operatori autorizzati possono, dopo essersi registrati e aver effettuato il login, inserire dei parametri climatici associati a delle aree di interesse. I comuni cittadini possono ricercare un'area di loro interesse e visualizzare eventuali parametri climatici associati a tale area.

Licenza

Condizioni di utilizzo

Il programma è distribuito sotto Licenza GPLv3: GNU GPL versione 3 **https://www.gnu.org/licenses/gpl-3.0.html**.

Software di terze parti

Il software di terze parti incluso nel codice sorgente ha licenze compatibili con la licenza del programma.

- OpenCSV 5.5.2 **Apache License Version 2.0**
- **Commons Lang** 3.1 **Apache License Version 2.0**

Requisiti di sistema

Il software è sviluppato e testato su sistema operativo GNU/Linux 6.3.11, Windows 10 e Windows 11 con Java 17 tuttavia non si esclude la compatibilità con altri sistemi e versioni.

Requisiti minimi

Sistema Operativo	GNU/Linux	Windows	MacOS	Altro
Versione	Kernel 5.4	8	10.13.6	N/D
Architettura	64 bit			
Java	17			
Memoria di archiviazione	15 MB			
Scheda video	Non richiesta			

Requisiti consigliati

Sistema Operativo	GNU/Linux	Windows	MacOS	Altro
Versione	Kernel 6.6	11	14.2	N/D
Architettura	64 bit			
Java	17			
Memoria di archiviazione	50 MB			
Scheda video	Non richiesta			

Sistemi non supportati direttamente

Lo sviluppo del software non supporta direttamente sistemi diversi da GNU/Linux, Windows e MacOS.
Non sono supportati direttamente: BSD, Solaris e altri sistemi; tuttavia, non è né esclusa la compatibilità. Vedere le apposite tabelle per verificare la compatibilità.

Sistemi non supportati

Non sono supportati:

- sistemi diversi da quelli a 64 bit.
- server
- dispositivi mobili
- dispositivi che non supportano Java 17.

Software esterno utilizzato

Il programma fa uso di software esterno. Il software di terze parti è incluso nel programma al suo scaricamento e **non sono richieste azioni aggiuntive da parte dell'utente**. Le librerie esterne sono presenti nella cartella **lib**.

OpenCSV

La libreria OpenCSV 5.5.2 è utilizzata all'interno del programma per la gestione dei file CSV.

Commons Lang 3

La libreria Commons Lang 3.1 è richiesta per il corretto funzionamento della libreria OpenCSV.

Installazione

Ottenimento del programma

Il programma, sia eseguibile che sorgente, è disponibile sulla piattaforma Github al link **https://github.com/ozneroL541/Climate_Monitoring**.
È occasionalmente possibile ottenere il programma direttamente dagli sviluppatori.

Non utilizzare software proveniente da altre fonti.

Installazione

Una volta scaricato il programma non è richiesta alcuna procedura di installazione.
Se il programma è stato fornito tramite una cartella compressa estrarne il contenuto.

Avvio dell'applicazione

Per avviare l'applicazione, tramite linea di comando, entrare nella cartella principale **Climate_Monitoring** e lanciare il seguente comando:

```
java -jar bin/ClimateMonitor.jar
```

Eseguire sempre il programma mentre ci si trova nella cartella dove è presente la cartella **resources** e, qualora fosse presente, la cartella **data**.

Sebbene sconsigliato, è possibile spostare il file **ClimateMonitor.jar** in qualsiasi altra cartella, purché venga sempre eseguito, utilizzando il percorso corretto, dalla cartella dove sono presenti le cartelle sopra citate.

Troubleshooting

Nel caso il comando indicato non esegua il programma assicurarsi che il file **ClimateMonitor.jar** si trovi all'interno della cartella **bin** e di trovarsi nella cartella corretta nel momento in cui si esegue il programma.

Nel caso non ci sia la cartella **resources** o un file contenuto in essa è consigliato riscaricarla. La mancanza di questi file non preclude il corretto funzionamento del programma; tuttavia, non permette di eseguire alcuni controlli in fase di inserimento dei dati e pertanto ne è altamente consigliato l'utilizzo.

Struttura del programma

Classi

All'interno della cartella **src** è contenuto il codice sorgente del programma. Tutte le classi sono state scritte interamente dagli sviluppatori.

```
src
├── climatemonitoring
│   └── ClimateMonitor.java
├── common
│   ├── CommonMethods.java
│   ├── CSV_Utilities.java
│   ├── Distance.java
│   ├── InputScanner.java
│   └── Research.java
├── geographicarea
│   ├── Coordinates.java
│   └── GeographicArea.java
├── header
│   └── Header.java
├── maxpq
│   └── MaxPQ.java
├── menu
│   └── MainMenu.java
├── monitoringcentre
│   └── MonitoringCentre.java
├── parameters
│   ├── Parameters.java
│   └── Table.java
└── users
    ├── AutorizedOperator.java
    ├── MenuCentre.java
    ├── MenuOperator.java
    └── User.java
```

Librerie Esterne

All'interno della cartella **lib** sono presenti le librerie esterne, in formato JAR, necessarie al funzionamento del programma.

```
lib
├── commons-lang3-3.1.jar
└── opencsv-5.5.2.jar
```

Risorse

All'interno della cartella **resources** sono presenti file per il controllo dei dati inseriti. La loro presenza non è necessaria per il corretto funzionamento del programma, tuttavia è migliorativa.

I file in questa cartella vengono utilizzati in fase di esecuzione, ma non vengono modificati.

```
resources
├── comuni-localita-cap-italia.csv
└── iso-3166-coutries.csv
```

Dati

La cartella **data** viene generata (se non presente) in fase di esecuzione e i suoi file vengono creati e modificati dall'utente attraverso l'interfaccia del programma.

```
data
├── CentroMonitoraggio.dati.csv
├── CoordinateMonitoraggio.dati.csv
├── OperatoriRegistrati.dati.csv
└── ParametriClimatici.dati.csv
```

Nel file **CentroMonitoraggio.dati.csv** vengono memorizzate le informazioni relative ai centri nel formato: `"Nome,Via,Civico,CAP,Comune,Provincia,Aree"`. Le aree sono salvate in un'unica cella del file separate da un trattino `"-"`.

Nel file **CoordinateMonitoraggio.dati.csv** vengono memorizzate le informazioni relative alle aree di interesse nel formato: `"Geoname ID,Name,ASCII Name,Country Code,Country Name,Coordinates"`.

Nel file **OperatoriRegistrati.dati.csv** vengono memorizzate le informazioni relative agli operatori che si sono registrati nel formato: `"Matricola,Nome,Cognome,Codice Fiscale,Indirizzo e-mail,Password,Centro di Monitoraggio"`.

Nel file **ParametriClimatici.dati.csv** vengono memorizzate le informazioni relative ai parametri climatici nel formato: `"Geoname ID,Data,Centro,Vento,Umidità,Pressione,Temperatura,Precipitazioni,Altitudine dei ghiacciai,Massa dei ghiacciai,Note Vento,Note Umidità,Note Pressione,Note Temperatura,Note Precipitazioni,Note Altitudine dei ghiacciai,Note Massa dei ghiacciai"`.

Altri file

I seguenti file, contenuti nella cartella principale **Climate_Monitoring**, non necessari per il funzionamento del programma.

```
automatic.sh
autori.txt
```

```
LICENSE.txt
README.md
readme.txt
```

Il file **automatic.sh** permette di compilare e documentare il programma con maggiore facilità. Il file è scritto interamente dagli sviluppatori ed è sottoposto alle stesse condizioni di licenza. Per maggiori informazioni consultare uno dei file **Readme**.

Il file **autori.txt** contiene i nomi e le matricole universitarie degli autori.

Il file **LICENSE.txt** contiene la licenza, per esteso, del programma.

I file **README.md** e **readme.txt** contengono una breve descrizione sull'utilizzo del programma.

Classi principali

ClimateMonitor

Contiene il *main* del programma. Questa classe crea un oggetto della classe *MainMenu*, il quale creerà il menu principale del programma.

MainMenu

Un oggetto della classe *MainMenu* rappresenta il menu di principale dell'applicazione.

Common

Nel package *Common* si trovano classi di utilità generale sfruttate da tutte le classi del programma.

CSV_Utilities

Raccolta di metodi statici utili per la gestione dei file CSV.

- `addArraytoCSV(File file, String[] linecells, String header)`

Aggiunge un array di stringhe rappresentanti le celle alla fine del file. Il metodo trasforma, effettuando tutti i controlli e le modifiche necessarie, l'array di stringhe in un'unica stringa di celle divise dalla virgola per poterla inserire in un file CSV, dopodiché procede con l'inserimento di quest'ultima.

Complessità
T = O(1)

- `addCellAtEndOfLine(File file, String string, int line)`

Aggiunge una stringa alla fine di una riga di un file CSV considerandola come il contenuto dell'ultima cella di tale riga. Per poter inserire la riga essa viene formatta e il file dove la si vuole scrivere viene rinominato con un nome temporaneo. Una volta rinominato il file viene scritto il contenuto dello stesso in un nuovo file col nome che il primo aveva in precedenza fino a che non si arriva alla riga di cui è richiesta la modifica e ne si aggiunge la cella. Il programma procede copiando la seconda parte del file ed eliminando l'originale.

Complessità
T = θ(n) n = righe del file

CommonMethods

Raccolta di vari metodi statici utilizzati da più classi.

Distance

Classe di appoggio per la ricerca delle coordinate. La classe rappresenta la distanza di un'area geografica dalle coordinate cercate e la riga dove la coordinata presente ed estende la classe *Comparable<Distance>*. Due oggetti *Distance* vengono comparati in base alla distanza.

- `toLines( MaxPQ<Distance> a2 )`

Il metodo estrae un heap di oggetti *Distance* da un oggetto *MaxPQ<Distance>*. Viene ritornato un array di interi che contiene le righe dove sono contenuti gli oggetti cercati in ordine dal più vicino al più lontano (gli interi non sono ordinati).

InputScanner

Classe contenente unicamente la costante statica e pubblica *INPUT_SCANNER* che è un oggetto *Scanner* utilizzato per leggere dallo stream di standard input. La costante permette di leggere da *stdin* da tutte le classi del programma evitando conflitti.

Research

Classe contenente gli algoritmi di ricerca per i dati nei file CSV.

- `AllStringInCol(File file, int col, String str)`
- `AllStringInCol_notCaseS(File file, int col, String str)`

Questi metodi ricercano una stringa in una determinata colonna e restituisce ogni riga in cui occorre. Il primo metodo è case sensitive mentre il secondo non lo è. La ricerca scorre tutto il file per cercare ogni occorrenza.

Complessità
T = θ(n) n = righe del file

- `areInSameLine(File file, int col1, int col2, String str1, String str2)`

Controlla se esiste una linea in cui c'è sia la prima stringa che la seconda nelle rispettive colonne.

Complessità
T = O(n) n = righe del file

- `CoordinatesAdvancedV3(File file, int col, double[] c)`

Metodo di ricerca pensato per la ricerca di aree tramite coordinate geografiche.

Restituisce tutte le linee che contengono le coordinate più vicine a quella passata in argomento. L'array è restituito con le celle in ordine di vicinanza.

Durante la ricerca, per ogni area, viene calcolata la distanza tra le coordinate dell'area e di quelle passate come argomento con la formula di Haversine. Una volta calcolata la distanza, se è inferiore a 3 000 km (dato che il punto più distante dalla terraferma è a 2 700 km), viene salvata, insieme alla linea corrispondente, nella classe *Distance* e inserita in una coda a massima priorità nella classe *MaxPQ* che contiene le m aree più vicine. Tramite un heapsort le varie distanze vengono riordinate e tramite il metodo ***toLines*** della classe *Distance* viene restituito l'array delle righe corrispondenti alle aree di cui si è precedentemente calcolata la distanza.

Complessità
T = O(n×log(m)) n = righe del file
S = O(m) m = lunghezza della coda a massima priorità

Per precauzione si è deciso di fissare la lunghezza *m* della coda a massima priorità a 10 nonostante non siano stati riscontrati rallentamenti o errori rispetto a una lista ordinata di lunghezza variabile (utilizzata nella versione 2 del metodo, che, per il medesimo motivo, si è deciso di limitare alla stessa lunghezza).

Essendo la lunghezza *m* fissata la complessità diviene T = O(n×log(m)) e S = O(1).

- `getColArray(File file, int col)`
- `getColNoRepetition(File file, int col)`

I metodi restituiscono tutte le celle appartenenti alla colonna selezionata. Il primo metodo accetta ripetizioni, il secondo no.

Complessità
T = θ(n)

- `getRecord(File file, int line)`
- `getRecordByData(File file, int col, String str)`
- `getRecordByTwoDatas(File file, int col1, String str1, int col2, String str2)`

I metodi ritornano un array di stringhe contenente le celle della riga cercata. Il primo metodo cerca la linea in argomento. Il secondo metodo cerca la prima linea che ha la stringa passata come argomento nella colonna indicata. Il terzo metodo cerca la prima linea che contiene le due stringhe passate come argomento nelle rispettive colonne.

Complessità
T = O(n) n = righe del file

- `isStringInCol(File file, int col, String str)`

Controlla che esista almeno una linea in cui è presente la stringa passata come argomento nella colonna indicata.

Complessità
T = O(n) n = righe del file

- `OneStringInCol(File file, int col, String str)`

Questo metodo ricerca una stringa in una determinata colonna e restituisce la riga corrispondente alla sua prima occorrenza. In caso non venga trovata un'occorrenza restituisce -1. In caso di errore restituisce un numero inferiore a -1.

Complessità
T = O(n) n = righe del file

MaxPQ

La classe rappresenta una coda ad accesso prioritario generica. La classe viene sfruttata dal programma per effettuare un heapsort durante la ricerca per coordinate.

Classi rappresentanti dati

Le seguenti classi forniscono metodi atti alla gestione dei dati presenti nei file CSV nella cartella **data**.

User

La classe *User* rappresenta i comuni cittadini che, non essendo essi registrati, non hanno un corrispettivo file CSV. I metodi presenti permettono di registrarsi come operatore autorizzato, effettuare l'accesso come tale, cercare aree di interesse e visualizzare i parametri climatici associati a ciascuna area di esse.

- `cercaAreaGeografica()`

Questo metodo permette di ricercare delle aree geografiche secondo diversi parametri di ricerca.
I parametri disponibili sono:

0 - "Geoname ID"
1 - "Nome Unicode"
2 - "Nome ASCII"
3 - "Codice Nazione"
4 - "Nome Nazione"
5 - "Coordinate"
10 - "Nome Generico"

La ricerca, in base al parametro scelto, verrà effettuata, tramite la classe *GeographicArea,* sfruttando differenti metodi offerti della classe *Research***.**

- `visualizzaAreaGeografica()`

Questo metodo permette di visualizzare le informazioni relative alle aree geografiche. Il metodo richiede alla classe *Parameters* tutte le aree presenti nel file **ParametriClimatici.dati.csv** e permette all'utente di visualizzarne i dati.

- `registrazione()`

Questo metodo permette di registrarsi come operatore autorizzato. Verrà chiesto all'utente di inserire alcuni dati che verranno, se la registrazione va a buon fine, salvati nel file **OperatoriRegistrati.dati.csv**. Durante la fase di registrazione vengono effettuati controlli sui dati inseriti e si previene l'esistenza di due utenze con il medesimo codice fiscale del quale viene inoltre controllata la validità.

AutorizedOperator

AutorizedOperator è una classe che rappresenta gli operatori autorizzati.

La classe estende *User* e, così facendo, eredita tutti i metodi disponibili per un utente non autorizzato, aggiungendo però i metodi disponibili solo per gli utenti autorizzati. Agli operatori autorizzati è permesso in aggiunta di creare una o più aree di interesse, creare centri di monitoraggio con l'elenco delle aree di interesse e inserire i valori dei parametri climatici per un'area di interesse.

AutorizedOperator ha come campi user-id, password, nome, cognome, codice fiscale, indirizzo di posta elettronica, e centro monitoraggio di afferenza (se presente). I campi vengono salvati, in fase di registrazione, nel file **OperatoriRegistrati.dati.csv**; dal quale vengono letti in fase di login.

- `inserisciParametriClimatici()`

Questo metodo permette agli operatori autorizzati di inserire i parametri climatici relativi ad un'area geografica. Un operatore può aggiungere parametri solamente alle aree associate al proprio centro di appartenenza. I parametri vengono richiesti e inseriti mediante l'uso della classe *Parameters*; la quale provvede al loro inserimento nel file **ParametriClimatici.dati.csv**.

MonitoringCentre

MonitoringCentre rappresenta i centri di monitoraggio. I suoi campi sono: nome del centro, indirizzo e aree di interesse afferenti. I dati dei centri sono salvati in **CentroMonitoraggio.dati.csv**.

- `registraCentroAree()`

Questo metodo permette di creare un nuovo centro di monitoraggio. Se la creazione va a buon fine il nuovo centro viene salvato nel file **CentroMonitoraggio.dati.csv**. Ogni campo del centro è salvato in una cella differente del file CSV, eccetto per le aree di interesse, i cui Geoname ID, siccome non possono essere modificate, sono salvati nella medesima cella distanziati da un separatore per praticità.

GeographicArea

GeographicArea rappresenta le aree di interesse. I suoi campi sono: Geoname ID, nome, nome in formato ASCII, codice nazione, nome nazione e coordinate. I dati delle coordinate sono salvati in **CoordinateMonitoraggio.dati.csv**.

Parameters

Parameters rappresenta la tabella dei parametri, mediante la classe *Table*, associata all'ID dell'area geografica, alla data di inserimento dei parametri e al centro di monitoraggio presso cui la si inserisce. Tutti queste informazioni vengono inserite nel file **ParametriClimatici.dati.csv**.

Table

Table rappresenta i parametri inseriti da un operatore autorizzato e le rispettive note.

Bibliografia

Calcolo distanza coordinate

Haversine formula - Wikipedia

https://www.matematicamente.it/forum/viewtopic.php?t=186924#:~:text=distanza%20%3D%20sqrt((x2%20%2D%20x1,z2%20%2D%20z1)%5E2)%3B

Country list

https://github.com/lukes/ISO-3166-Countries-with-Regional-Codes/blob/master/all/all.csv

Codice Fiscale

https://it.wikipedia.org/wiki/Codice_fiscale

CSV Writer-Reader

https://www.baeldung.com/opencsv

Lista Comuni

https://rosariociaglia.altervista.org/database-delle-localita-comuni-e-cap-ditalia-csv-excel-free-download/

Nomi con la virgola

https://www.ilsole24ore.com/art/l-esperto-risponde-doppio-nome-documenti-solo-se-non-c-e-virgola-AETbwlhD

Poste Italiane - Indirizzi

https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiWp9zu1tCCAxWvUqQEHTEMDhIQFnoECCAQAQ&url=https%3A%2F%2Fwww.poste.it%2Fstandard_composizione_indirizzi.pdf&usg=AOvVaw1bRMYlIBzrYuypLKyeYtHs&opi=89978449

Punto più distante dalla terraferma 2700km

https://www.passioneastronomia.it/punto-nemo-larea-piu-remota-della-terra