2022/2023

MANUALE TECNICO

|  |  |
| --- | --- |
| RICCI MARCO  RIZZI EDOARDO  ALBERTO STAGNO  DENIS DI NAPOLI | 755656  754618  755404  755340 |

Climate Monitoring

UNIVERSITà DEGLI STUDI DELL’INSUBRIA LAUREA TRIENNALE IN INFORMATICA

Sommario

[Introduzione 2](#_Toc137743430)

[Librerie esterne 2](#_Toc137743431)

[Codice fiscale 2](#_Toc137743432)

[JCalendar 2](#_Toc137743433)

[Struttura del sistema 2](#_Toc137743434)

[Classi Logiche 2](#_Toc137743435)

[Classi Grafiche 2](#_Toc137743436)

[Classi logiche 3](#_Toc137743437)

[ParserCSV 3](#_Toc137743438)

[Organizzazione dei file 3](#_Toc137743439)

[JAreaInteresse 3](#_Toc137743440)

[JCoordinate 3](#_Toc137743441)

[JLuogo 3](#_Toc137743442)

[JPrevisioni 3](#_Toc137743443)

[JUser 3](#_Toc137743444)

[Classi grafiche 3](#_Toc137743445)

[admin\_panel 3](#_Toc137743446)

[creaStazione 3](#_Toc137743447)

[homepage 3](#_Toc137743448)

[infoStazione 3](#_Toc137743449)

[login 3](#_Toc137743450)

[mostraPrevisioni 3](#_Toc137743451)

[Bibliografia 4](#_Toc137743452)

# Introduzione

Il progetto “*Climate Monitoring*” è un sistema di monitoraggio di parametri climatici fornito da centri di monitoraggio sul territorio italiano; utilizzabile sia dal punto di vista di un comune cittadino, che potrà ricercare l’area d’interesse o la stazione metereologica da lui desiderata consultando poi le previsioni inserite, e sia dal punto di vista di un operatore ambientale che potrà oltre a visionare le previsioni anche modificarle e aggiungerle.

Il progetto è sviluppato in Java 17 su sistemi windows 10 e 11, e testato sia sugli stessi che sul sistema MacOS --.

Quest’ultimo è stato svolto per il corso “Laboratorio interdisciplinare A” nell’anno scolastico 2022/2023 per il corso di laurea informatica dell’Università degli Studi dell’Insubria.

## Librerie esterne

### Codice fiscale

La libreria “*codice-fiscale-java-master*” ci permette tramite i metodi in essa contenuti di calcolare il codice fiscale di una persona, di cui ci vengono forniti i dati personali necessari per il calcolo. Questa libreria viene utilizzata nella parte di registrazione di un utente.2

### JCalendar

JCalendar, è una libreria che abbiamo aggiunto per migliorare l’aspetto grafico del lavoro, la libreria permette di selezionare una data tramite un calendario o dei combobox. Al suo interno troviamo altri oggetti come JDayChooser, JMonthChooser e JYearChooser.

## Struttura del sistema

Le classi del progetto si suddividono in due categorie; quelle relative all’interfaccia grafica (che chiameremo “*classi grafiche*”) e quelle che elaborano i diversi dati (che chiameremo “*classi logiche*”)

### Classi Logiche

* ParserCSV
* JAreaInteresse
* JCoordinate
* JLuogo
* JPrevisioni
* JUser

### Classi Grafiche

* admin\_panel
* creaStazione
* homepage
* infoStazione
* login
* mostraPrevisioni
* registrazione

Presenteremo adesso le diverse classi sia quelle “Logiche” che quelle “Grafiche”, descrivendone il loro compito e i metodi principali della classe.

# Classi logiche

## ParserCSV

La classe ParserCSV è la classe cardine del progetto, il cui scopo è quello di gestire la scrittura e la lettura dei dati presenti nei file. Fornendo poi i dati elaborati in liste alle diverse classi grafiche per la visualizzazione a schermo.

### Organizzazione dei file (attributi classe)

Per la gestione del dataset abbiamo deciso di utilizzare diversi file CSV. I file sono sei ed ognuno

memorizza uno specifico tipo di dato, e sono suddivisi nel seguente modo:

|  |  |  |
| --- | --- | --- |
| Nome del file | Descrizione | Dati contenuti |
| utenti.csv | File su cui sono presenti i dati di accesso delle persone registrate, ogni utente sarà associato ad un operatore | nome, cognome, pass, cf, id\_stazione, codiceOperatore |
| operatori.csv | File su cui sono presenti i dati relativi all’operatore ambientale | codiceOperatore |
| areedinteresse.csv | File in cui sono presenti le aree d’interesse inserite dagli operatori, ogni area d’interesse sarà associata ad una stazione metereologica | id\_area, geoname\_id, nome |
| previsioni.csv | File in cui sono presenti le previsioni inserite dagli operatori | data, id\_centro, id\_area, username, vVento, pUmidita, pressione, temperatura, precipitazioni, aGhiacciai, mGhiacciai |
| stazioni.csv | File in cui sono presenti tutte le stazioni metereologiche inserite degli operatori | geoname\_id, citta, cod\_nazione, nazione, coordinate |
| nazioni.csv | File in cui sono presenti tutte le nazioni | --- |

La classe contiene come attributi i vari percorsi relativi dei file che sono contenuti nella cartella “*dati*” all’interno di quella del progetto.

### Metodi Principali della classe

|  |  |  |  |
| --- | --- | --- | --- |
| Nome metodo | Parametri | Descrizione | Complessità |
| aggiungiAreaInteresse | Integer geoname\_id:  String nome | aggiunge una nuova area di interesse al file delle aree di interesse | O(n + m ), dipende dalla dimensione del file delle aree di interesse |
| aggiungiPrevisione | JPrevisione previsione | aggiunge una nuova previsione al file delle previsioni | O(n + m), dipende dalla dimensione del file delle previsioni |
| cercaPerArea | String citta | cerca le aree di interesse in base al nome della città fornita come parametro | O(n + m), dipende dalla dimensione del file delle aree di interesse |
| cercaPerStazione | String citta  JCoordinate coordinata  Integer Distanza | cerca i luoghi (stazioni) in base alla città o alle coordinate fornite come parametri | O(n + m), dipende principalmente dalla dimensione del file delle stazioni |
| checkCodiceOperatore |  | verifica se un determinato codice operatore esiste nel file degli operatori | O(n + m + k), dipende dalla dimensione del file degli operatori e dalla lunghezza del codice operatore |
| creaListaPrevisioni | Integer id\_area | legge le previsioni dal file delle previsioni e crea un'ArrayList di oggetti JPrevisioni corrispondenti all'area specificata | O(n + m), dipende dalla dimensione del file delle previsioni |
| creaListaPrevisioniByDate | Integer id\_area  String strData | legge le previsioni dal file delle previsioni e crea un'ArrayList di oggetti JPrevisioni corrispondenti all'area specificata e alla data specificata | dipende principalmente dalla dimensione del file delle previsioni e può essere approssimata a O(n + m |
| creaListaPrevisioniUniqueDate |  | legge le previsioni dal file delle previsioni e crea un'ArrayList di oggetti JPrevisioni corrispondenti all'area specificata, con date univoche | O(n + m + k), dipende dalla dimensione del file delle previsioni e dalla dimensione della lista l di previsioni univoche |
| creaListaStazioni |  | Crea una lista di oggetti JLuogo leggendo le linee da un file e inizializzando un nuovo oggetto JLuogo per ogni linea letta. | O(n + m), dove n è la dimensione del file e m è il numero di linee lette |
| creaStazione |  | Crea una nuova riga nel file delle stazioni con i dati specificati. | O(k), dipende dalla dimensione della riga |
| creaUtetenteLoggato |  | cerca un utente nel file degli utenti corrispondente al nome utente e alla password forniti e restituisce un oggetto JUser rappresentante l'utente loggato. | O(n + m), dipende dalla dimensione del file degli utenti |
| esisteUtente |  | verifica se esiste un utente nel file degli utenti corrispondente al nome utente e alla password specificati | O(n + m), dipende dalla dimensione del file degli utenti |
| getAllAreaInteresse |  | restituisce un elenco di oggetti JAreaInteresse corrispondenti a un determinato ID di Geoname | O(n + m), dipende dalla dimensione del file delle aree di interesse |
| getNazioni |  | legge le linee da un file e crea una lista di array di stringhe rappresentanti le nazioni | O(n + m), dipende dalla dimensione del file delle nazioni |
| getNomeStazioneByGeonameId |  | restituisce il nome della stazione corrispondente a un determinato ID di Geoname | O(n + m), dipende dalla dimensione del file delle aree di interesse |
| getStazione |  | Cerca un oggetto JLuogo nella lista di stazioni utilizzando un ID specifico. | O(n + m + k) |
| registraUtente |  | registra un nuovo utente nel file degli utenti | O(n + m + k), dipende dalla dimensione del file degli utenti |

## JAreaInteresse

JAreaIntresse è la classe che gestisce le diverse aree di interesse.

Gli attributi della classe sono:

|  |  |  |
| --- | --- | --- |
| Tipo | Nome attributo | Descrizione |
| Integer | id\_area | Codice identificativo dell'area d'interesse |
| Integer | geoname\_id | Codice della stazione metereologica a cui fa appartenenza l'area d'interesse. |
| String | nome | Nome dell’area d’interesse |

Nella classe non sono presenti metodi particolari oltre i classici operatori, get/set e il toString.

## JCoordinate

La classe JCoordinate e di conseguenza l’omonimo oggetto, ha lo scopo di facilitare la gestione dei punti cardinale utilizzate varie volte nel corso del progetto.

Gli attributi della classe sono:

|  |  |  |
| --- | --- | --- |
| Tipo | Nome attributo | Descrizione |
| Float | lat | Coordinata geografica che indica la distanza dall’equatore al punto in cui ti trovi |
| Float | lon | Coordinata geografica che indica la distanza dal meridiano di Greenwhic al punto in cui ti trovi |

I metodi principali della classe sono

|  |  |  |  |
| --- | --- | --- | --- |
| Nome metodo | Parametri | Descrizione | Complessità |
| distanzaDa | JCoordinate altro:  secondo oggetto da cui si misura la distanza | Serve per calcolare la distanza tra la coordinata che richiama il metodo e quella passata tramite parametro | O(1) |
| sonoCoordinate | String txt:  stringa che contiene le possibili coordinate | controlla che la stringa passata sia effettivamente una coordinata | O(n), qui la complessità varia a seconda della lunghezza di txt. |

## JLuogo

JLuogo è la classe che si occupa della gestione delle stazioni metereologiche.

Gli attributi della classe sono:

|  |  |  |
| --- | --- | --- |
| Tipo | Nome attributo | Descrizione |
| Integer | geoname\_id | Codice identificativo della cittá dove si trova la stazione metereologica. |
| String | nome | Nome della cittá dove si trova la stazione metereologica |
| String | country\_code | Codice della nazione dove si trova la stazione metereologica |
| String | country | Nome della nazione dove si trova la stazione metereologica |
| JCoordinate | coordinate | Coordinate dove si trova la stazione metereologica |

Nella classe non sono presenti metodi particolari oltre i classici operatori, get/set e il toString.

## JPrevisioni

JPrevisioni è la classe che si occupa della creazione e gestione delle previsioni inserite dagli operatori ambientali.

Gli attributi della classe sono:

|  |  |  |
| --- | --- | --- |
| Tipo | Nome attributo | Descrizione |
| Integer | id\_area | Codice identificativo dell'area d'interesse |
| Integer | id\_centro | Codice della stazione metereologica a cui fa appartenenza l'area d'interesse. |
| Date | data | Indica la data in cui è stata inserita la previsione |
| String | username | Indica l’operatore che ha inserito la previsione |
| Integer | vVento | Velocità del vento (km/h), suddivisa in fasce |
| Integer | pUmidita | Percentuale di Umidità, suddivisa in fasce |
| Integer | pressione | Livello della pressione espressa in hPa, suddivisa in fasce |
| Integer | temperatura | Livello della temperatura, espressa in °C, suddivisa in fasce |
| Integer | precipitazioni | Quantità di pioggia caduta, espressa in mm, espressa in fasce |
| Integer | aGhiacciai | Altitudine dei ghiacciai espressa in m, suddivisa in fasce |
| Integer | mGhiacciai | Massa dei ghiacciai espressa in kg, suddivisa in fasce |
| String | nVento | Note, facoltative, che aggiungo informazioni al valore vVento |
| String | nUmidita | Note, facoltative, che aggiungo informazioni al valore pUmidita |
| String | nPressione | Note, facoltative, che aggiungo informazioni al valore pressione |
| String | nTemperatura | Note, facoltative, che aggiungo informazioni al valore temperatura |
| String | nPrecipitazioni | Note, facoltative, che aggiungo informazioni al valore precipitazioni |
| String | nAGhiacciai | Note, facoltative, che aggiungo informazioni al valore aGhiacciai |
| String | nMGhiacciai | Note, facoltative, che aggiungo informazioni al valore mGhiacciai |

Nella classe non sono presenti metodi particolari oltre i classici operatori, get/set e il toString.

## JUser

JUser è la classe che si occupa della gestione degli operatori ambientali.

Gli attributi della classe sono:

|  |  |  |
| --- | --- | --- |
| Tipo | Nome attributo | Descrizione |
| String | username | Indica il nome con cui l’operatore inserirà le previsioni |
| String | nome | Nome dell’utente |
| String | cognome | Cognome dell’utente |
| String | id | Codice univoco che identifica l’user |
| String | password | sequenza di caratteri alfanumerici utilizzata per accedere in modo esclusivo |
| String | cf | Codice fiscale dell’utente |
| String | mail | Mail dell’utente |
| Integer | geoname\_id | Indica la stazione metereologica a cui è associato l'operatore |
| int | id\_areaselezionata | Indica l'ultima area di interesse di cui l'operatore ha aggiunto una previsione |

Nella classe non sono presenti metodi particolari oltre i classici operatori, get/set e il toString.

# Classi grafiche

## admin\_panel

admin\_panel è la classe che si occupa della finestra personale di un operatore ambientale, questa pagina la si raggiunge dopo aver effettuato il login

## creaStazione

p

## homepage

p

## infoStazione

p

## login

p

## mostraPrevisioni

p

## regisistrazione

# Bibliografia

codice-fiscale-java-master - matteocastagnaro - <https://github.com/matteocastagnaro/codice-fiscale-java/blob/master/jar/codice-fiscale.jar?raw=true>

JCalendar - Kai Tödter - <https://toedter.com/jcalendar/>