UNIVERSITÀ DEGLI STUDI DI TRENTO

Dipartimento di Ingegneria e Scienza dell'Informazione

Corso di Laurea in
Informatica

Progetto Ingegneria del Software

Sistema di monitoraggio ambientale

D4 G19

Anno accademico 2021/2022

Indice

Scopo del documento

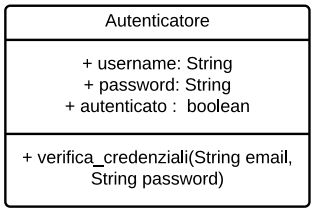
Il presente documento riporta la definizione dell'architettura del progetto "Sistema monitoraggio ambientale" usando il diagramma delle classi in Unified Modeling Language (UML). Viene inoltre introdotto il linguaggio Object Constraint Language (OCL) per esprimere vincoli che altrimenti non sarebbe possibili esporre mediante UML in modo formale e preciso. Per rappresentare l'architettura del sistema viene utilizzato il diagramma delle classi in linguaggio UML che dovranno essere implementate a livello di codice; viene inoltre spiegata la logica dietro il corretto funzionamento del sistema.

1 | Diagramma delle classi

Nel presente capitolo vengono presentate le classi previste nell'ambito del progetto "Sistema di monitoraggio ambientale". Ogni attore e ogni sistema esterno presenti nel diagramma di contesto diventano una o più classi.

1.1 Autenticatore

Nel nostro sistema, come è possibile osservare nel component diagram, è presente un elemento incaricato di gestire l'autenticazione di un amministratore. Questa funzionalità è stata modellata con la classe `Autenticatore`. L'autenticazione viene effettuata controllando i dati inseriti dall'utente attraverso la funzione `verifica_credenziali(String email,String password)`. Questa funzione quindi chiamerà al suo interno `Interfaccia_database.get_password(String email)` e confronterà il suo valore di ritorno con quello inserito dall'utente. Di seguito la classe con i suoi attributi e metodi.

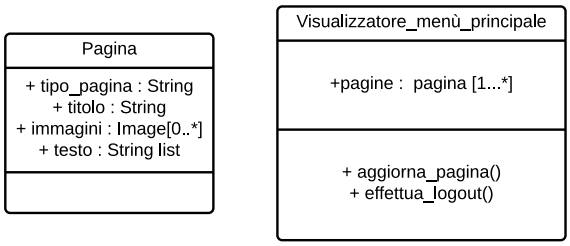


1.2 Visualizzatore menù principale

Analizzando il diagramma di contesto del sistema si nota la presenza di due attori: "Utente non autenticato" e "Amministratore". L'utente non autenticato è in grado di ricevere delle notifiche di allarme in caso di allerta ambientale e può ottenere informazioni sulla flora e la fauna. L'amministratore, invece, è in grado di effettuare funzionalità più complesse, ossia ottenere lo storico di animali specifici in un determinato lasso di tempo, ottenere statistiche sulla valutazione dei rischi e gestire il parco.

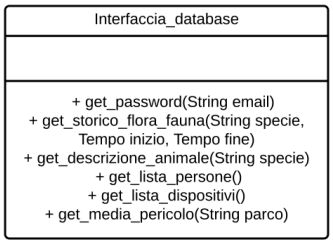
Il componente "visualizzatore menù principale" presentato nel diagramma dei componenti viene implementato utilizzando una classe con il medesimo nome. Esso permette all'utente di svolgere le attività secondo il livello di accesso che esso possiede secondo il principio del "Need to Know". Il metodo `aggiorna_pagina()` permette all'applicazione di fornire la pagina richiesta comunicando con `Interfaccia_database` ed il `Generatore_istogrammi`, secondo il livello di accesso possesso dall'utente.

Le pagine vengono fornite creando un'istanza della classe `Pagina` in base alla richiesta dell'utente. Per esempio, nel caso in cui venga richiesto una pagina contenente la descrizione di una determinata specie, essa conterrà immagini di animali di quella specie, la loro descrizione e le posizioni in tempo reale degli animali sulla mappa del parco.



1.3 Interfaccia Database

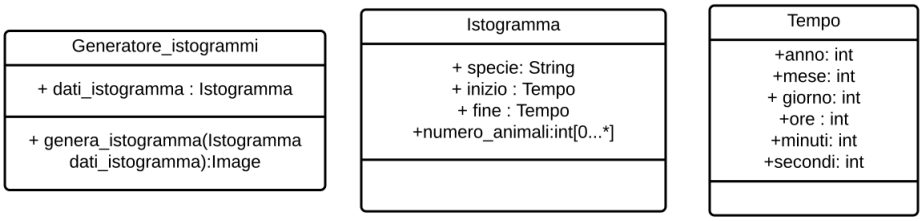
Il componente "Interfaccia Base di Dati" è necessario per il corretto funzionamento del sistema in quanto gli permette di dialogare con il database esterno. La classe `Interfaccia_database` implementa tali funzionalità e permette ai componenti interni di ottenere i dati richiesti. Di seguito viene esposta la classe con i suoi attributi ed i suoi metodi.



1.4 Generatore Istogrammi

Analizzando il componente "Generatore istogrammi", necessario per creare i grafici sullo storico di un animale, abbiamo identificato 3 classi `Generatore_istogramma`, `Istogramma`, `Tempo` con attributi e funzioni specifici.

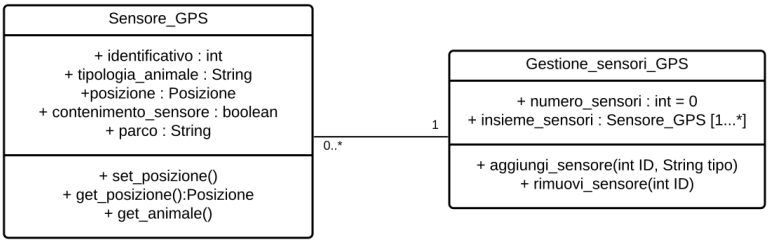
In particolare la classe `Generatore_istogramma` è in grado di creare un'immagine partendo da un dato di tipo `Istogramma`, il quale racchiude tutti i dati necessari richiesti dall'utente.



1.5 Gestione sensori GPS

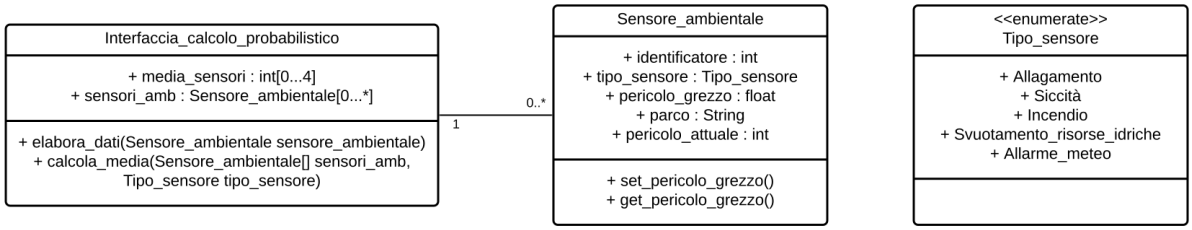
Come si può vedere nel diagramma di contesto, il sistema dovrà comunicare con il Sistema GPS. Questo è stato reso possibile realizzando la classe `Sensore_GPS` e la classe `gestione_sensori_GPS`:

- la prima è caratterizzata da degli attributi che identificano univocamente a quale animale è assegnato, mentre le funzioni permettono principalmente di monitorare la posizione di esso. Nel caso in cui un animale esca dal suo parco di appartenenza, il sensore è in grado di capire che si trova fuori dalla zona a lui assegnata, e l'attributo `contenimento_sensore` cambia valore in false. Questa informazione verrà poi percepita da un'istanza della classe `interfaccia_notifiche` per inviare le dovute segnalazioni.
- la seconda invece memorizza tutti i sensori e il numero totale di essi, implementando anche metodi per aggiungere o rimuovere unità dal sistema.



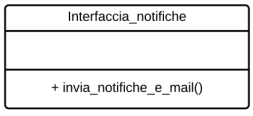
1.6 Gestione sensori ambientali

Nel diagramma di contesto sono presenti due sistemi esterni chiamati Sensori e Sistema calcolo probabilistico. Le entità di cui sopra sono state modellate in questo class diagram attraverso le classi `Sensore ambientale` e `Interfaccia calcolo probabilistico`. Gli attributi appartenenti alla classe `Sensore_ambientale` sono utili per identificarli univocamente all'interno di un parco e per descriverne la loro natura (per descrivere il loro compito all'interno del sistema). Infatti è stato creato il nuovo tipo `Tipo_sensore` che descrive se il sensore è designato ad acquisire informazioni su allagamento, siccità, incendio, termine delle risorse idriche o meteo. Inoltre questa classe permette di registrare i valori grezzi e raffinati del pericolo della rispettiva categoria. La classe Interfaccia calcolo probabilistico invece permette di memorizzare la totalità dei sensori nella `lista sensori_amb` e anche la media delle percentuali di pericolo per ogni categoria di pericolo. Parlando dei metodi invece è possibile notare che `elabora_dati(Sensore_ambientale sensore_ambientale)` prende in input un sensore e restituisce un intero che andrà a popolare l'attributo `pericolo_attuale` di quel sensore. Invece la funzione `calcola_media(Sensore_ambientale[] sensori_amb, Tipo_sensore tipo_sensore)` prende in input l'array di tutti di sensori e il tipo sensore del quale fare la media con l'obiettivo di popolare l'array delle medie.

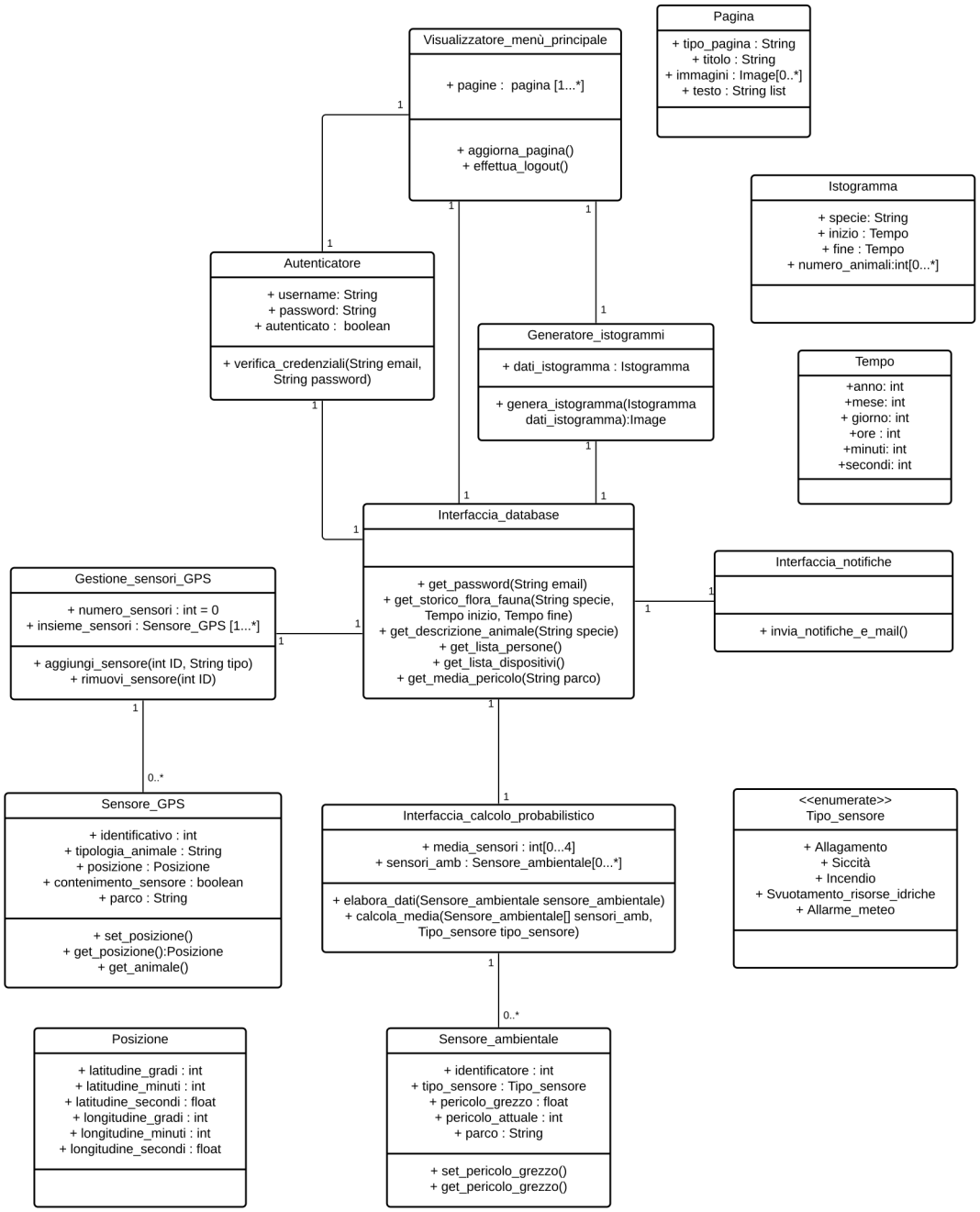


1.7 Interfaccia notifiche

Questa classe è stata modellata per interfacciarsi al Sistema invio mail/notifiche. In questo progetto abbiamo deciso di implementare un singolo metodo che al suo interno chiamerà le funzioni `Interfaccia_database.get_lista_persone()` e `Interfaccia_database.get_lista_dispositivi()` per sapere a quali dispositivi inviare la notifica e a quali mail inviare le segnalazioni. Inoltre, chiamando la funzione `Interfaccia_database.get_media_pericolo(String parco)`, il sistema sarà in grado di capire il livello di pericolo (vedi documenti precedenti) sapendo le percentuali restituite dalla funzione appena chiamata.



1.8 Diagramma delle classi complessivo



2 | OCL (Object Constraint Language)

In questo capitolo viene utilizzato il linguaggio Object Constraint Language (OCL) per descrivere formalmente dei vincoli necessari al corretto funzionamento del sistema. Questo linguaggio è utilizzato perchè non ci sono altri modi per descrivere tali limitazioni nel contesto UML.

2.1 Verifica credenziali

Per permettere all'amministratore di effettuare il login è necessario che esso non lo abbia già effettuato. L'attributo `autenticato` nella classe `Autenticatore` deve quindi essere impostato a `false` prima della chiamata della funzione. Viene proposta questa condizione mediante precondizione in OCL:

```
context
Autenticatore::verifica_credenziali()
pre:Autenticatore.autenticato==false
```

2.2 Effettua logout

L'amministratore che desidera effettuare il logout deve necessariamente aver effettuato il login, inoltre dopo aver effettuato il logout esso deve essere in grado di effettuare nuovamente il login. Di seguito viene espresso mediante OCL utilizzando una pre-condizione ed una post-condizione.

```
context
Visualizzatore_menù_principale::effetta_logout()
pre:Autenticatore.autenticato==true

context
Visualizzatore_menù_principale::effettua_logout()
post:Autenticatore.autenticato==false
```

2.3 Generazione istogramma

Per poter generare l'istogramma è necessario che il lasso temporale selezionato esista, ossia che l'attributo inizio abbia un valore minore di fine. Essendo `Tempo` un valore non primitivo bisogna confrontare nel seguente ordine anno, mese, giorno, ore, minuti e secondi nel caso anche i minuti avessero lo stesso valore.

Un'altra condizione necessaria per richiedere la generazione dell'istogramma è che l'utente sia autenticato, ossia che l'attributo `autenticato` nella classe `Autenticatore` sia impostato a `True`.

I seguenti due vincoli sono espressi mediante OCL utilizzando un'invariante e una precondizione con il seguente codice:

```
context Generatore_istogrammi inv:
inizio<fine

context:
Generatore_istogrammi::genera_istogramma()
pre:Autenticatore.autenticato==true
```

3 Diagramma delle classi con codice OCL

Riportiamo infine il diagramma delle classi con tutte le classi fino ad ora presentate ed il codice OCL individuato.