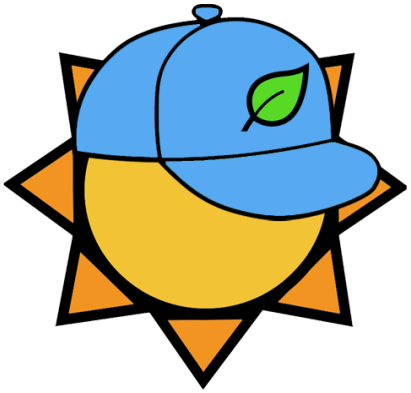
ODD

Progetto



WeatherStyle

Versione	0.6
Data	01/02/2023
Destinatario	Professore Carmine Gravino.
Presentato da	Aurucci Raffaele, Miglino Annalaura, Palmieri Angelo, Zammarrelli Francesco Giuseppe.

Revision History

Data	Versione	Descrizione	Autore
19/01/2023	0.1	Prima stesura	Tutto il team
21/01/2023	0.2	Aggiunti package	Aurucci Raffaele Francesco Giuseppe Zammarelli
27/01/2023	0.3	Aggiunte classi, interfacce per gestione suggerimenti e gestione citta + bridge pattern gestione suggerimenti	Aurucci Raffaele
27/01/2023	0.3	Aggiunta specifica delle interfacce per il package gestione ambiente	Palmieri Angelo
27/01/2023	0.3	Aggiunta specifica delle interfacce per il package gestione guardaroba	Miglino Annalaura
30/01/2023	0.4	Aggiunto adapter pattern e class diagram delle interfacce utilizzate	Aurucci Raffaele
30/01/2023	0.5	Aggiunta specifica delle interfacce per il package gestionemeteo e gestioneutente	Francesco Giuseppe Zammarrelli
31/01/2023	0.6	Revisione Completa	Tutto il team

1. Introduzione

WeatherStyle ha come obiettivo principale quello di mostrare le previsioni meteo di qualsiasi città del mondo e, sulla base di queste, suggerire agli utenti dei capi d'abbigliamento ritenuti più adatti da indossare.

1.1.Linee Guida per la Scrittura del Codice

Le linee guida seguite per la scrittura del codice seguono la convenzione java nota come **Sun Java Coding Conventions** e, in particolare, è stato fatto uso del tool **CheckStyle** per rispettare tale convenzione.

Inoltre, per lo sviluppo sono state definite le seguenti linee guida:

- Le classi dei package control devono iniziare con un verbo e avere il suffisso **Servlet**
- Le interfacce dei package logic.service devono avere il suffisso **LogicService;**
- Le classi che implementano le interfacce in logic.service e storage.service devono avere il suffisso **LogicImpl;**
- Le interfacce dei package storage.service devono avere il suffisso **Service;**
- Le interfacce dei package storage.dao devono avere il suffisso **DAOInterface**;
- Le classi che implementano le interfacce nei package storage.dao devono avere il suffisso **DAOImpl**;
- Le classi devono essere espresse al singolare.

1.2.Definizione, acronimi e abbreviazioni

Package: raggruppamento di classi e interfacce;
Design pattern: template di soluzioni a problemi ricorrenti impiegati per ottenere riuso e flessibilità;
Interfaccia: insieme di signature di operazioni offerte dalle classi che le implementano;
Javadoc: sistema di documentazione realizzato da Java in formato HTML;

1.3.Riferimenti e Link Utili

- Statement of Work (SOW)
- Requirements Analysis Document (RAD)
- System Design Document (SDD)
- Testing Plan
- Test Case Specification
- Test Incident Report
- Test Summary Report
- Matrice di tracciabilità

2. Packages

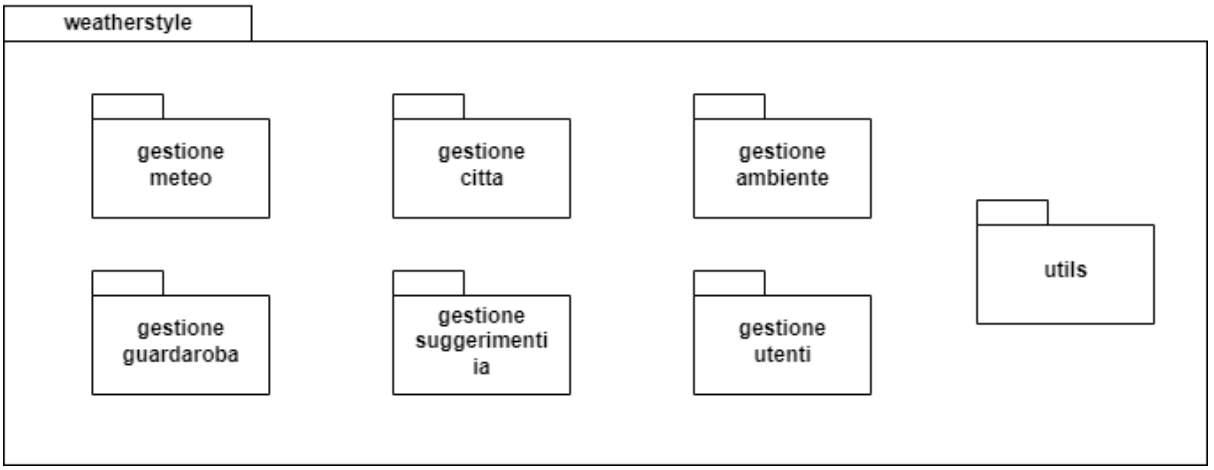
Nella presente sezione viene mostrato il package principale di WeatherStyle, composto a sua volta da ulteriori package.

Ogni package al suo interno sarà strutturato nel modo seguente:

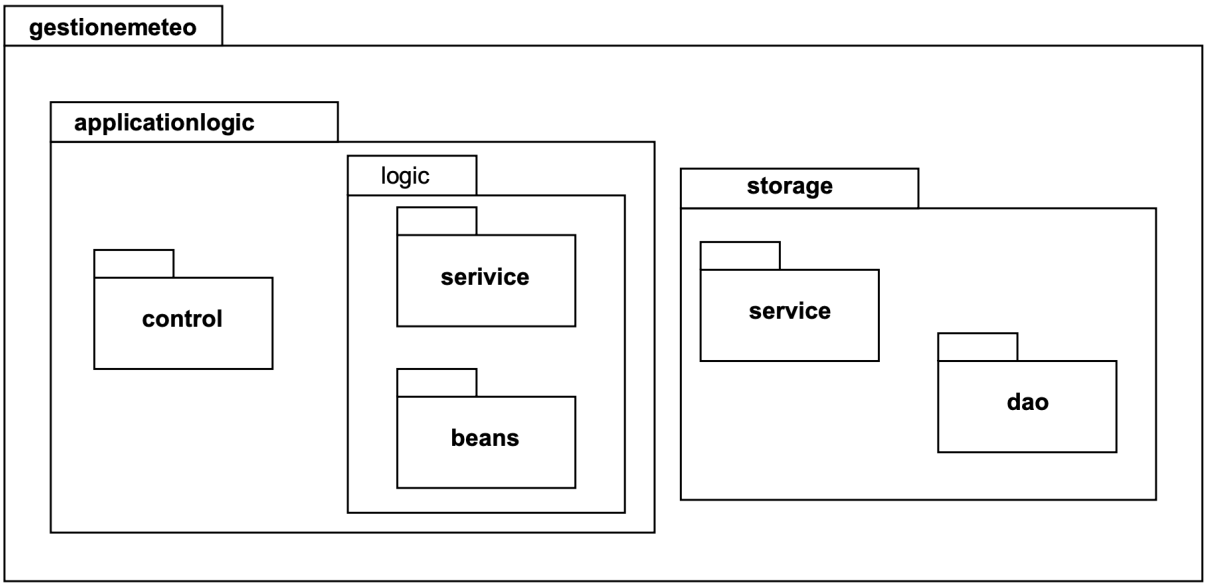
- un **package applicationlogic** contenente:
    - il **package control** per le classi servlet.
    - il **package logic** contenente:
        - il **package service** per le classi che gestiscono la logica di business.
        - il **package beans** per le classi che rappresentano le componenti base del nostro sistema.
- un **package storage** contenente:
    - il **package dao** per l'interfaccia al database.

Per i **package gestionemeteo** e **gestionecitta,** all'interno del **package storage** vedranno l'aggiunta del **package service**, contenente le classi per lo scambio dati con le API.
Infine, solo per il package **gestionesuggerimentiia,** il **package applicationlogic.logic** conterrà, in aggiunta, il **package algorithms** contenente la logica algoritmica per i suggerimenti intelligenti.

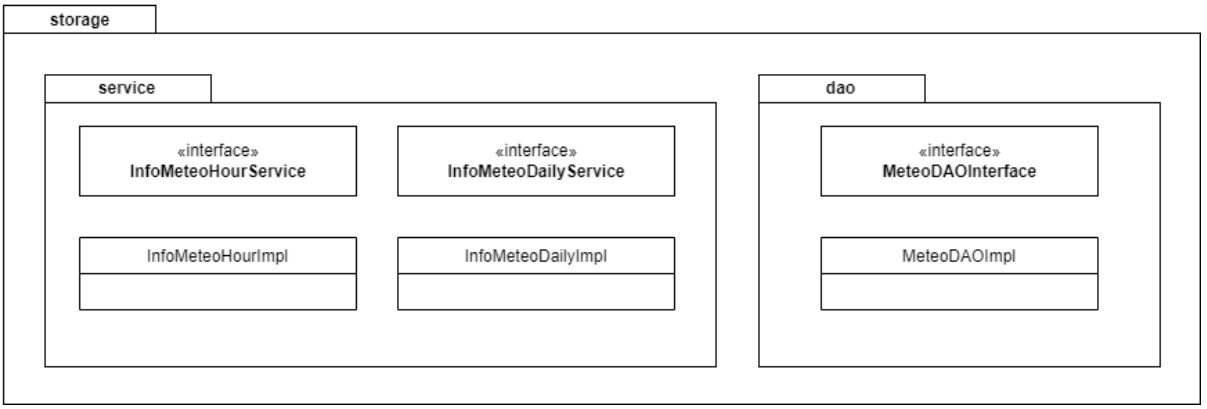
Package weatherstyle



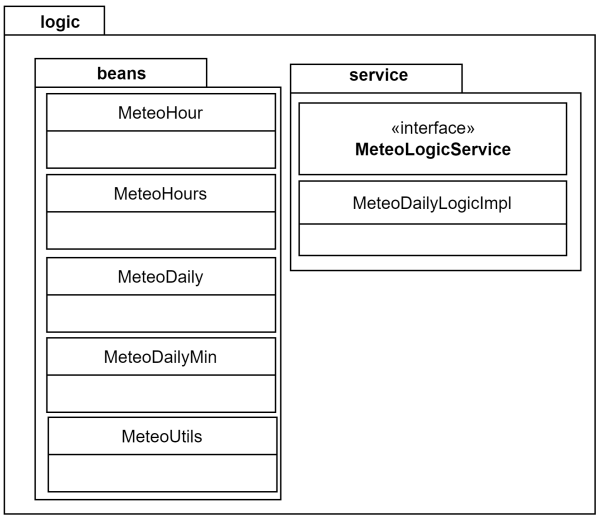
Package gestionemeteo



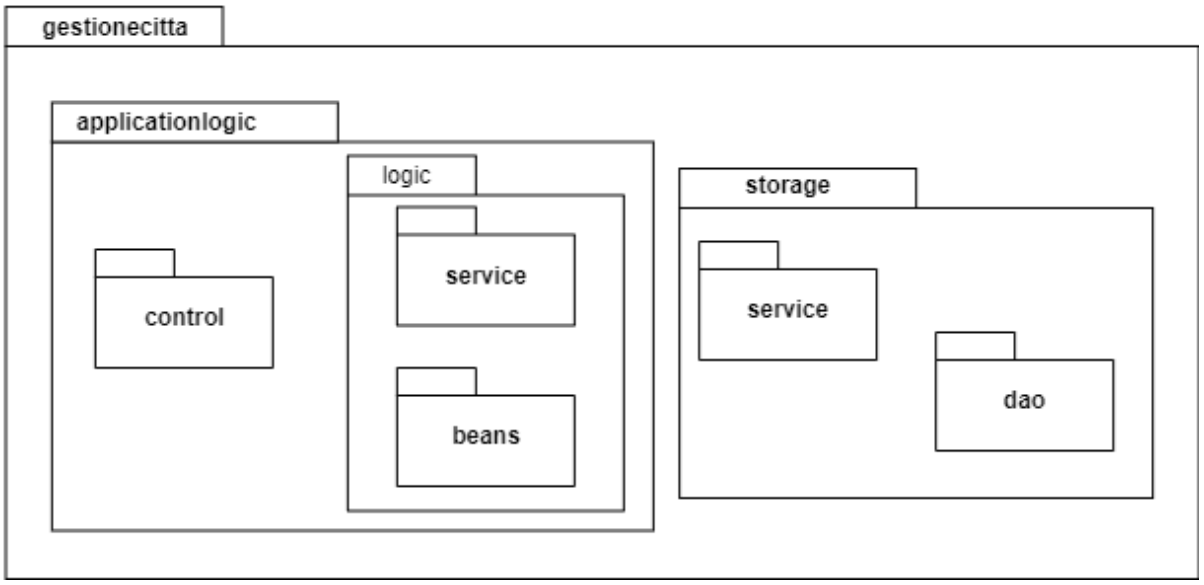
storage



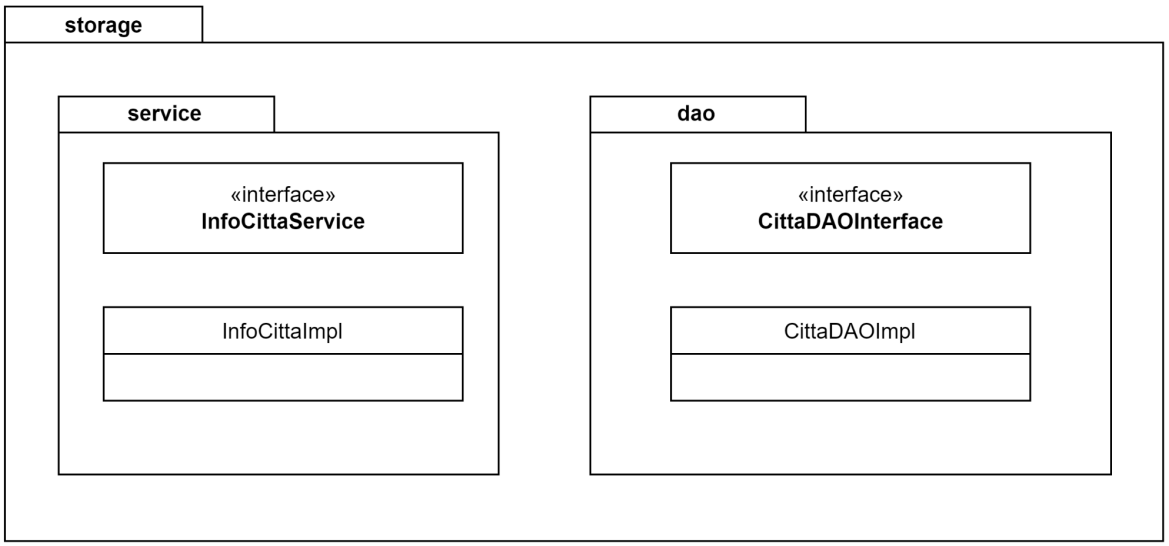
application.logic



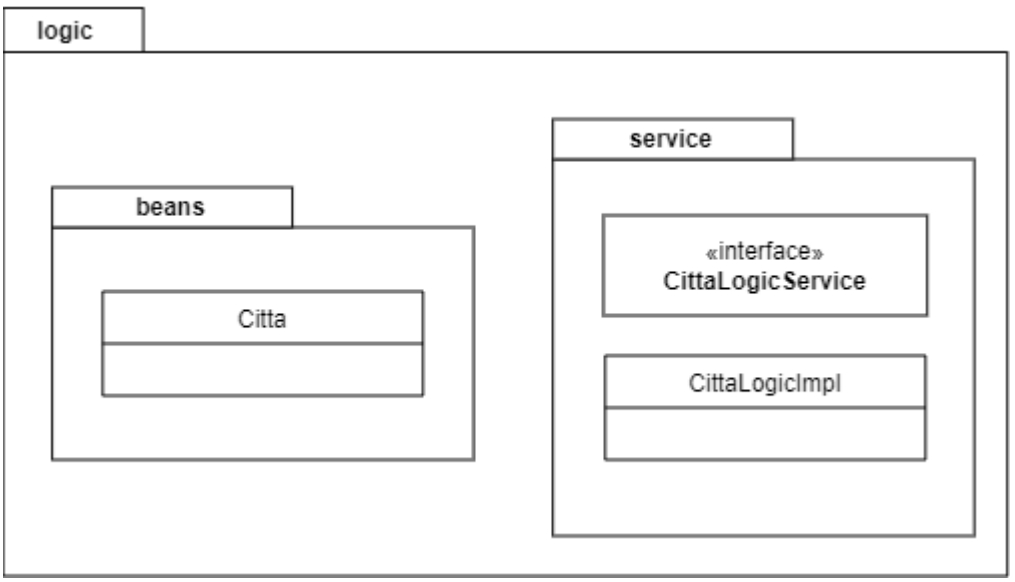
Package gestionecitta



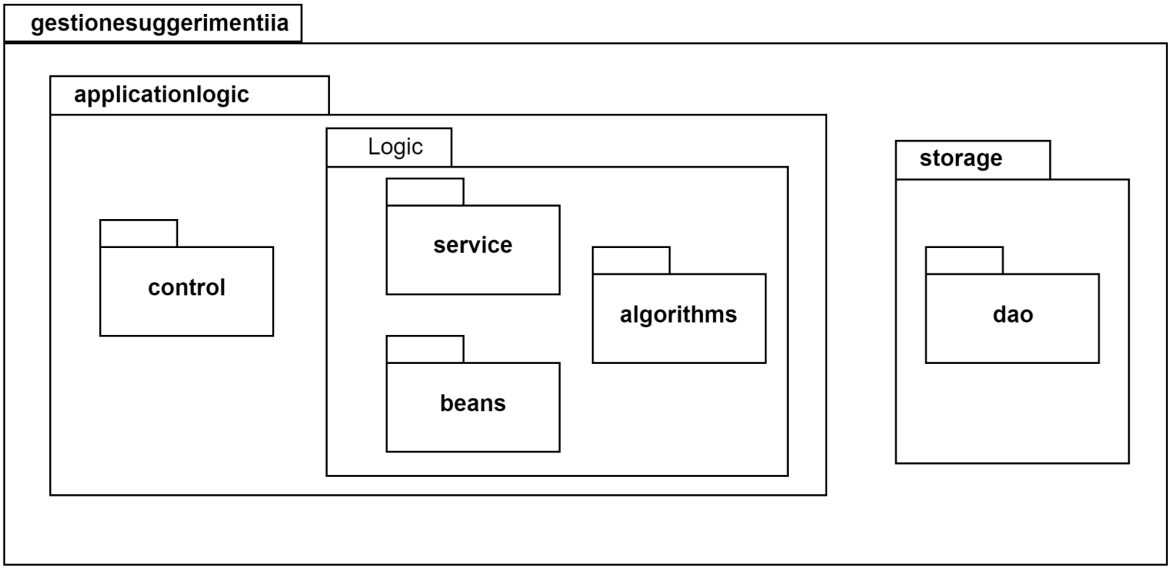
storage



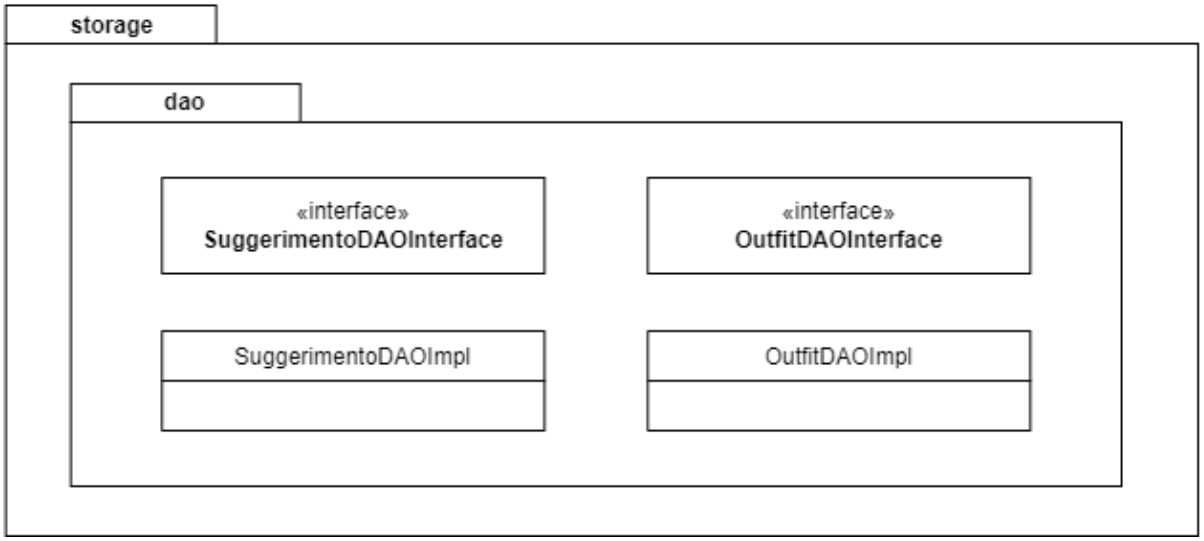
applicationlogic.logic



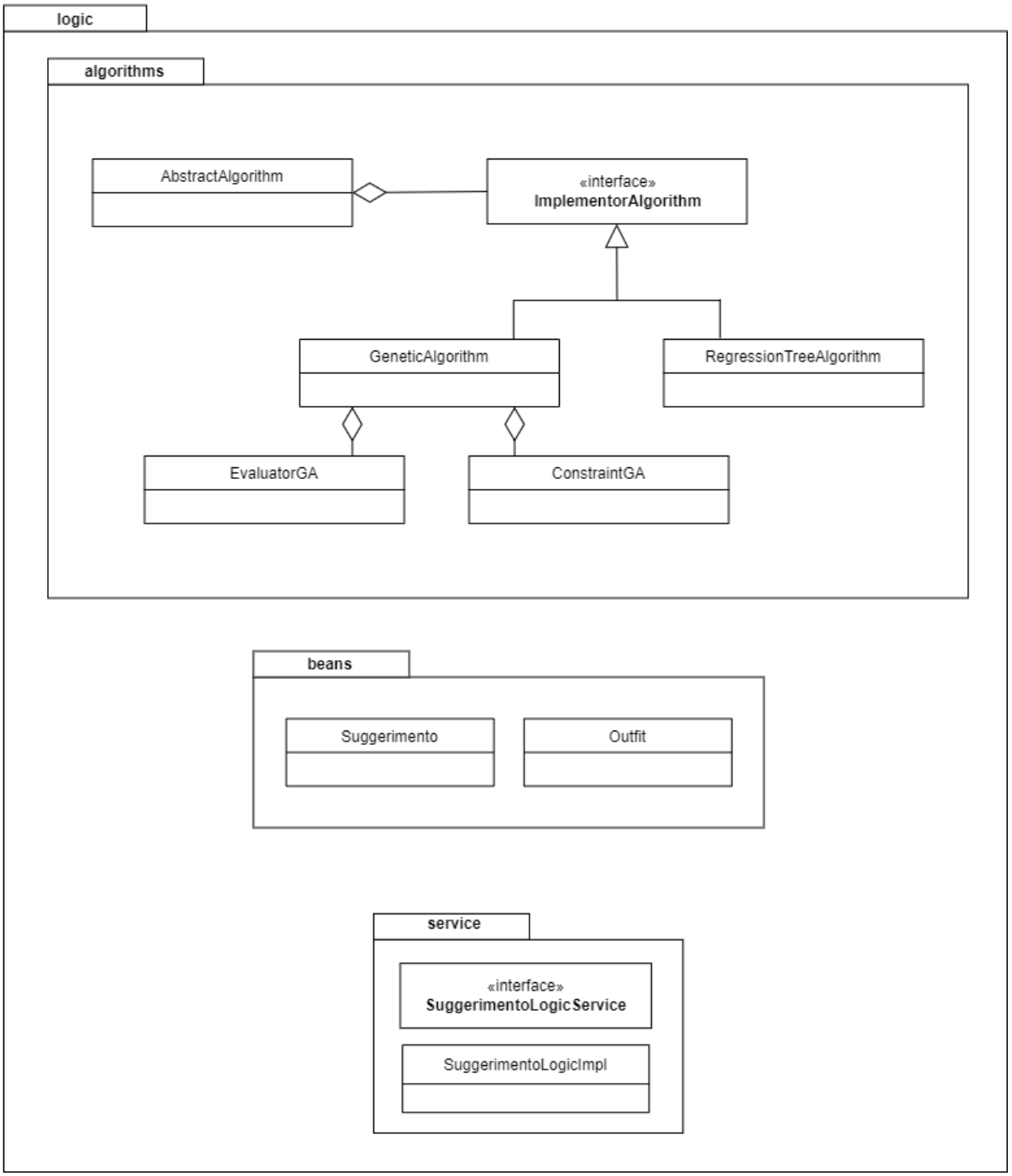
Package gestionesuggerimentiia



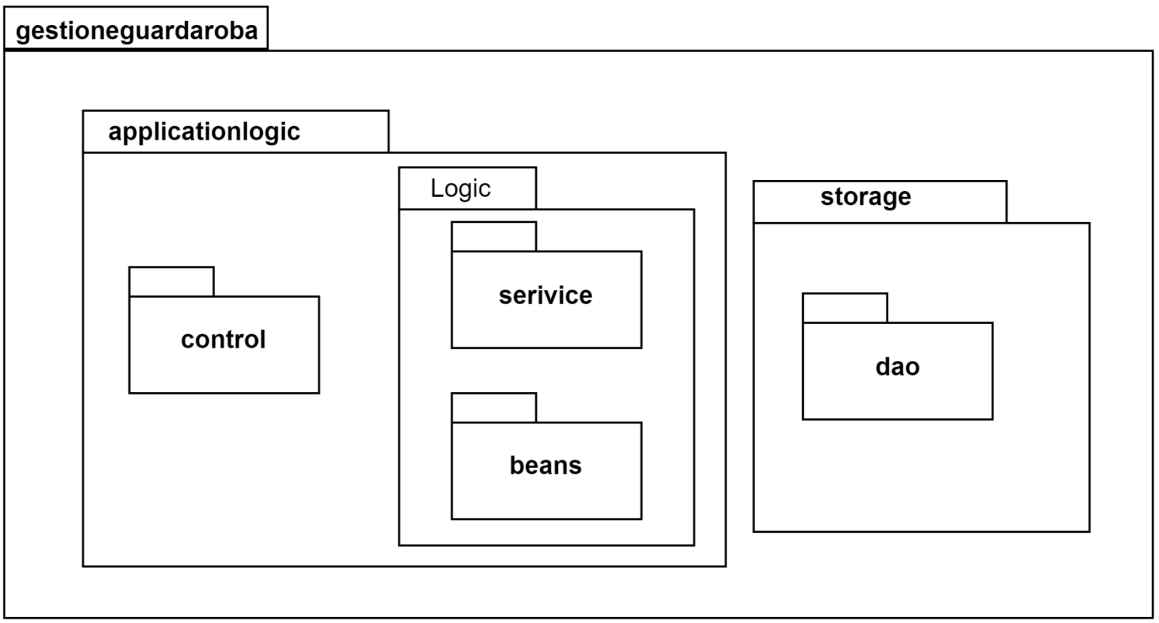
storage



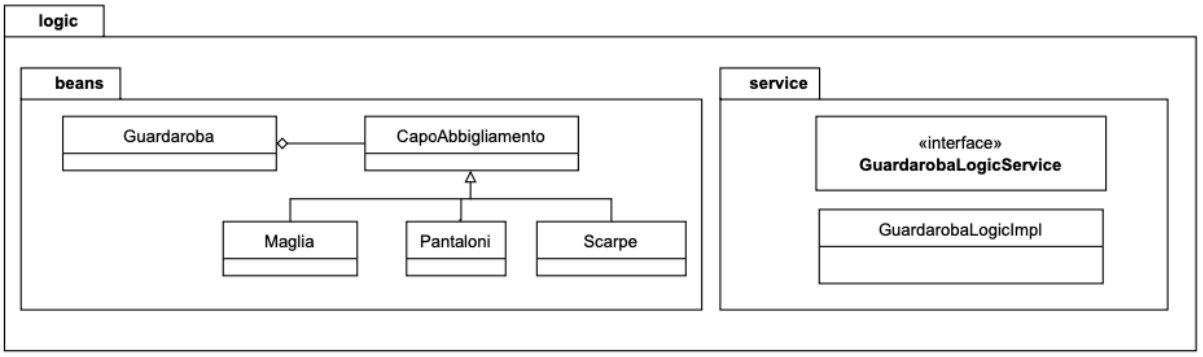
applicationlogic.logic



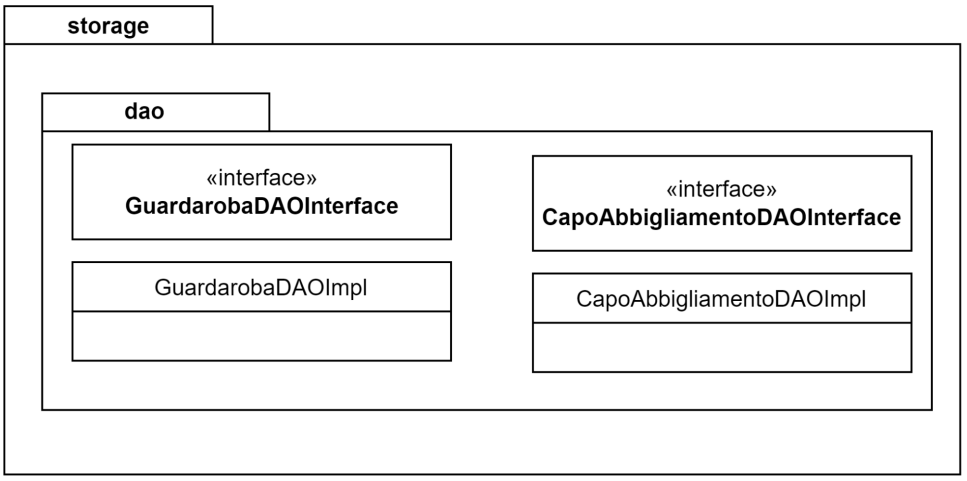
Package gestioneguardaroba



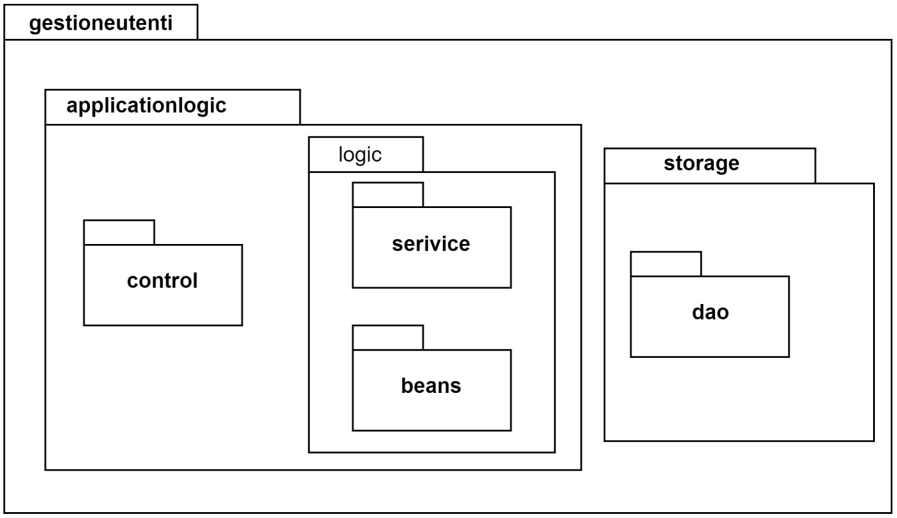
application.logic



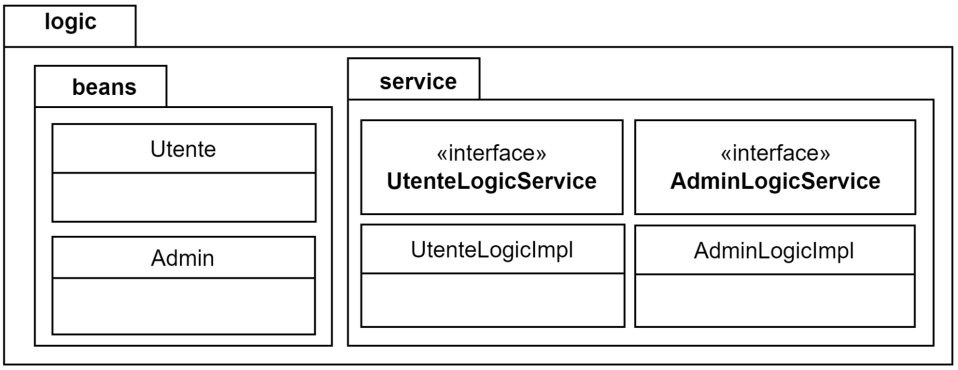
storage



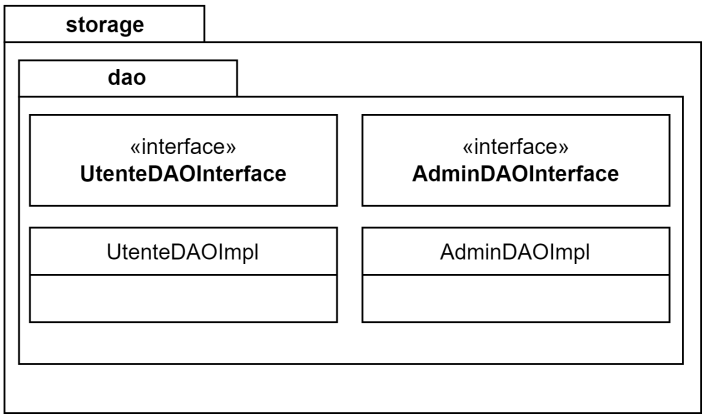
Package gestioneutenti



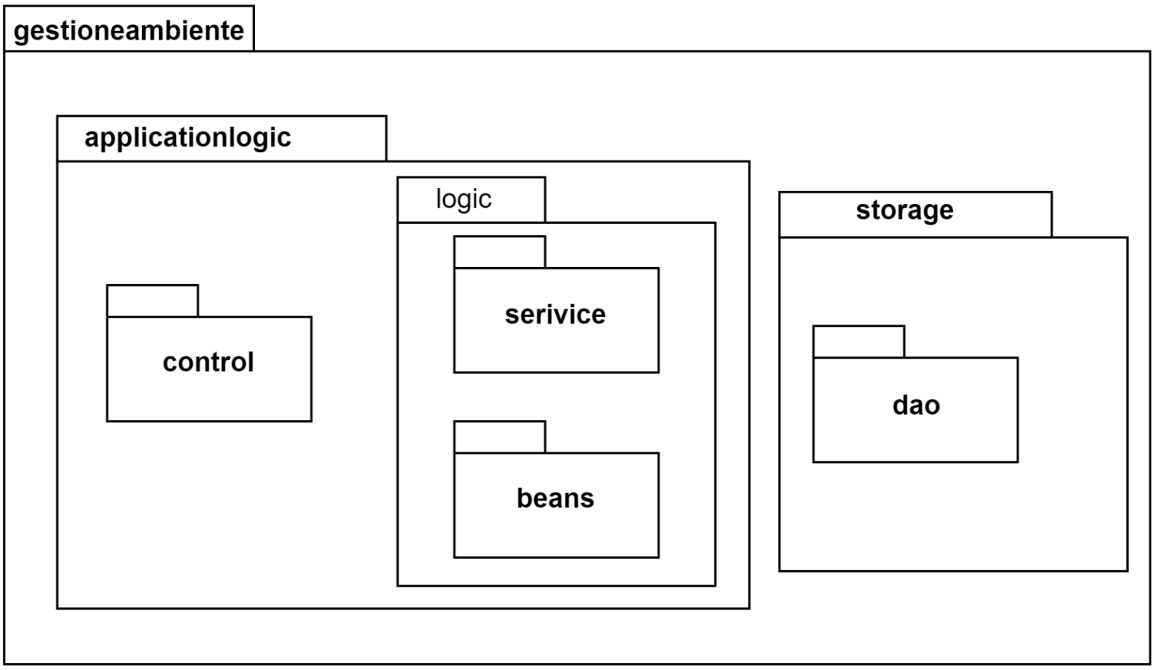
application.logic



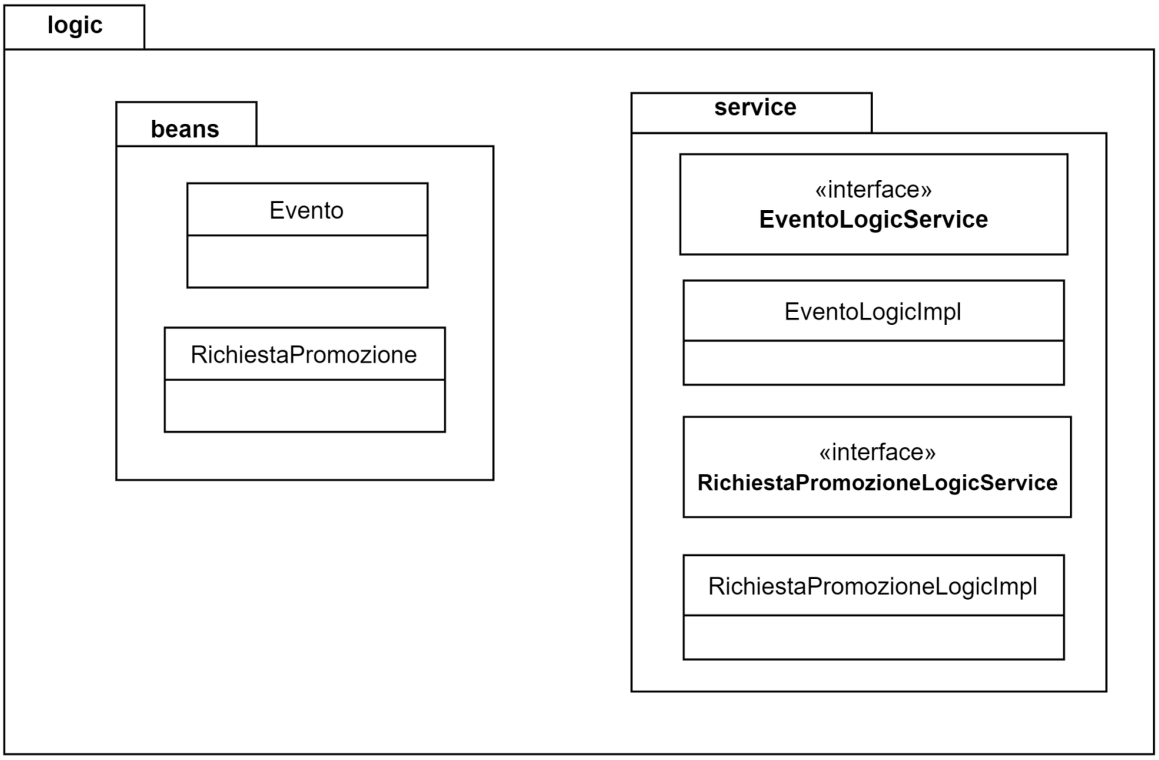
storage



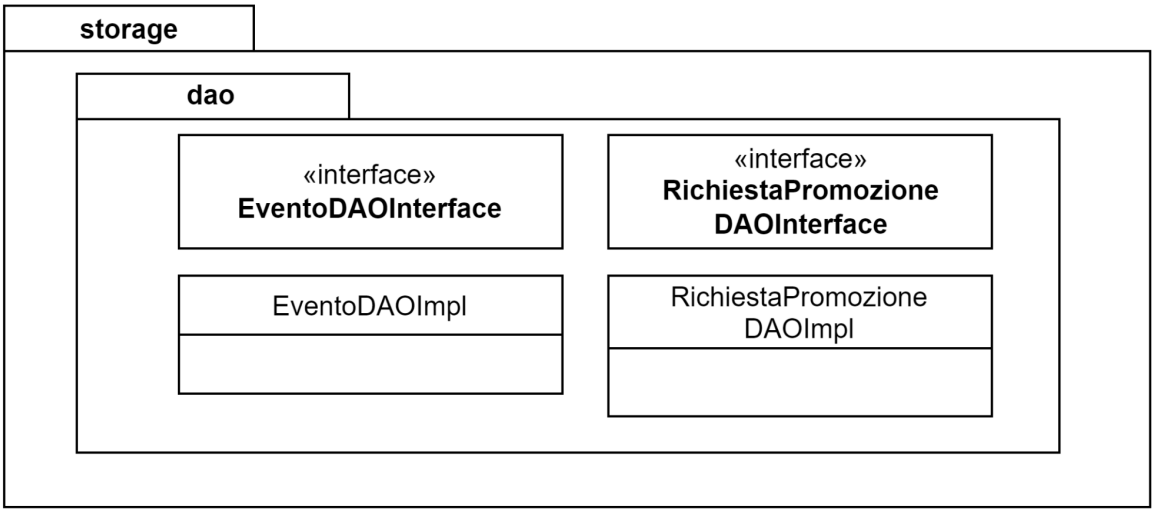
Package gestioneambiente



application.logic



storage


storage
dao
«interface»
EventoDAOInterface
«interface»
RichiestaPromozione
DAOInterface
EventoDAOImpl
RichiestaPromozione
DAOImpl

3. Class Interfaces

In questa sezione saranno presentate le interfacce dei vari package, non saranno presenti le interfacce relative ai **package logic.controller** e ai **package model.beans.**
Per le altre è possibile fare riferimento alla JavaDoc disponibile al seguente link.

3.1 Package gestionemeteo

- **weatherstyle.gestionemeteo.storage.service.*InfoMeteoHourImpl***

Nome classe	*InfoMeteoHourImpl*
Descrizione	Questa classe permette di gestire le operazioni relative alle informazioni meteo orarie interfacciandosi con una API.
Metodi	+getInfoMeteoHourByDay(LocalDate day, Citta citta):MeteoHours +getInfoMeteoHourByRangeOfDays(LocalDate init, LocalDate end, Citta citta):List<MeteoHours>
Invariante classe	/

Nome metodo	+getInfoMeteoHourByDay(LocalDate day, Citta citta):MeteoHours
Descrizione	Metodo che restituisce le informazioni meteo orarie di una data città in un dato giorno.
Pre-condizione	**context:** InfoMeteoHourImpl::getInfoMeteoHourByDay(LocalDate day, Citta citta) **pre**: citta != null && (LocalDateTime.now().equals(day) || LocalDateTime.now().before(day))
Post-condizione	**/**
Nome metodo	+getInfoMeteoHourByRangeOfDays(LocalDate init, LocalDate end, Citta citta):List<MeteoHours>
Descrizione	Metodo che restituisce le informazioni meteo orarie di una data città dei giorni da init a end.
Pre-condizione	**context:** InfoMeteoHourImpl::getInfoMeteoHourByRangeOfDays(LocalDate day, Citta citta) **pre**: citta != null && (LocalDateTime.now().equals(init) || LocalDateTime.now().before(init)) && init.before(end)
Post-condizione	**/**

- **weatherstyle.gestionemeteo.storage.service.*InfoMeteoDailyImpl***

Nome classe	*InfoMeteoDailyImpl*
Descrizione	Questa classe permette di gestire le operazioni relative alle informazioni meteo giornaliere interfacciandosi con una API.
Metodi	+getInfoMeteoDailyByDay(LocalDate day, Citta citta): MeteoDaily +getInfoMeteoDailyByRangeOfDays(LocalDate init, LocalDate end, Citta citta): List<MeteoDaily>
Invariante classe	/

Nome metodo	+getInfoMeteoDailyByDay(LocalDate day, Citta citta):MeteoDaily
Descrizione	Metodo che restituisce le informazioni meteo giornaliere di una data città in un dato giorno.
Pre-condizione	**context**: InfoMeteoDailyImpl::getInfoMeteoDailyByDay (LocalDate day, Citta citta) **pre**: citta != null && (LocalDateTime.now().equals(day) || LocalDateTime.now().before(day))
Post-condizione	**/**
Nome metodo	+getInfoMeteoDailyByRangeOfDays(LocalDate end init, LocalDate end, Citta citta):List<MeteoDaily>
Descrizione	Metodo che restituisce le informazioni meteo giornaliere di una data città dei giorni da init a end.
Pre-condizione	**context:** InfoMeteoDailyImpl::getInfoMeteoDailyByRangeOfDays(LocalDate day, Citta citta) **pre**: citta != null && (LocalDateTime.now().equals(init) || LocalDateTime.now().before(init)) && init.before(end)
Post-condizione	**/**

- **weatherstyle.gestionemeteo.storage.dao.MeteoDAOImpl**

Nome classe	*MeteoDAOImpl*
Descrizione	Questa classe contiene le operazioni per gestire le informazioni meteo persistenti.
Metodi	+doSaveMeteo(MeteoDailyMin meteo, int idSuggerimento): boolean +doRetrieveMeteoBySuggerimentoID(int idSuggerimento): MeteoDailyMin
Invariante di classe	/

Nome metodo	+doSaveMeteo(MeteoDailyMin meteo, int idSuggerimento): boolean
Descrizione	Metodo che permette di salvare il meteo relativo a un suggerimento
Pre-condizione	**context**: MeteoDAOImpl::doSaveMeteo(MeteoDailyMin meteo, int idSuggerimento): boolean **pre**: meteo != null && getMeteoBySuggerimentoID(idSuggerimento) == null
Post-condizione	**context:** MeteoDAOImpl::doSaveMeteo(MeteoDailyMin meteo, int idSuggerimento): boolean **post:** getMeteoBySuggerimentoID(idSuggerimento) != null
Nome metodo	+doRetrieveMeteoBySuggerimentoID(int idSuggerimento): MeteoDailyMin
Descrizione	Metodo che permette di ottenere info meteo giornaliere relative a un suggerimento con ID idSuggerimento
Pre-condizione	**context**: MeteoDAOImpl::doRetrieveMeteoBySuggerimentoID(int idSuggerimento): MeteoDailyMin **pre**: idSuggerimento > 0
Post-condizione	/

- **weatherstyle.gestionemeteo.logic.service.MeteoLogicServiceImpl**

Nome classe	*MeteoLogicServiceImpl*
Descrizione	Questa classe contiene le operazioni per gestire le informazioni relative al meteo.
Metodi	+getMeteoHours(Citta citta):List<MeteoHours> +getMeteoDaily(Citta citta): List<MeteoDaily>
Invariante di classe	/

Nome metodo	+getMeteoHours(Citta citta):List<MeteoHours>
Descrizione	Metodo che permette di ottenere il meteo orario di una determinata città. Le date delle previsioni sono quella del giorno in cui viene effettuata la chiamata e i due giorni successivi
Pre-condizione	**context**: MeteoLogicSerivceImpl::getMeteoHours():List<MeteoHours> **pre**: citta!= null
Post-condizione	/
Nome metodo	+getMeteoDaily(Citta citta): List<MeteoDaily>
Descrizione	Metodo che permette di ottenere il meteo giornaliero di una determinata città. Le date delle previsioni sono quella del giorno in cui viene effettuata la chiamata e i due giorni successivi
Pre-condizione	**context**: MeteoLogicSerivceImpl::getMeteoDaily():List<MeteoDaily> **pre**: citta != null
Post-condizione	/

3.2 Package gestionecitta

- **weatherstyle.gestionecitta.storage.service.InfoCittaImpl**

Nome classe	*InfoCittaImpl*
Descrizione	Questa classe permette di ottenere informazioni sulle città ricercate dall'Utente interfacciandosi con una API.
Metodi	+getCittaByName(String name): List<Citta>
Invariante di classe	/

Nome metodo	+getCittaByName(String name): List<Citta>
Descrizione	Questo metodo permette di ottenere una lista di città a partire dalla stringa passata in input al metodo.
Pre-condizione	**context:** CittaService::getCittaByName(String name) **pre:** name.length()>=1
Post-condizione	/

- **weatherstyle.gestionecitta.storage.dao.CittaDAOImpl**

Nome classe	*CittaDAOImpl*
Descrizione	Questa classe contiene le operazioni che permettono di gestire le informazioni persistenti relative ad una città.
Metodi	+doSaveCitta(Citta citta): boolean +doRetrieveCittaBySuggerimentoID(int idSuggerimento): Citta +doRetrieveCittaPreferiteByUtenteID(int idUtente): List<Citta> +doSaveCittaByUtenteID(int idUtente): boolean -doRetrieveCittaByLatLon(Citta citta): boolean
Invariante di classe	/

Nome metodo	+doSaveCitta(Citta citta): boolean
Descrizione	Questo metodo permette di salvare una città nel database.
Pre-condizione	/
Post-condizione	/
Nome metodo	+doRetrieveCittaBySuggerimentoID(int idSuggerimento): Citta
Descrizione	Questo metodo permette di ottenere una città precedentemente salvata rispetto all'id del suggerimento a cui si riferisce.
Pre-condizione	/
Post-condizione	/
Nome metodo	+doRetrieveCittaPreferiteByUtenteID(int idUtente): Citta
Descrizione	Questo metodo permette di ottenere la lista di citta preferite da un utente.
Pre-condizione	/
Post-condizione	/
Nome metodo	+doSaveCittaByUtenteID(int idUtente): Citta
Descrizione	Questo metodo permette di salvare una città preferita da un Utente.
Pre-condizione	/
Post-condizione	/
Nome metodo	-doRetrieveCittaByLatLon(int idSuggerimento): Citta
Descrizione	Questo metodo permette di verificare se una città è già presente nel DB tramite latitudine e langitudine.
Pre-condizione	/
Post-condizione	/

- **weatherstyle.gestionecitta.applicationlogic.logic.service.CittaLogicImpl**

Nome classe	*CittaLogicImpl*
Descrizione	Questa classe contiene i metodi fondamentali per la logica di business.
Metodi	+ottieniCittaByName(String name): List<Citta> +ottieniJsonDaCitta(List<Citta> cittaList): String
Invariante di classe	/

Nome metodo	+ottieniCittaByName(String name): List<Citta>
Descrizione	Questo metodo permette di ottenere la lista di citta preferite da un utente.
Pre-condizione	**context:** CittaLogicImpl::ottieniCittaByName(String name) **pre:** name != null && name.lenght() >= 1
Post-condizione	/
Nome metodo	+ottieniJsonDaCitta(List<Citta> cittaList): String
Descrizione	Questo metodo permette di ottenere una stringa JSON a partire da una lista di città.
Pre-condizione	/
Post-condizione	/

3.3 Package gestionesuggerimentiia

- **weatherstyle.applicationlogic.logic.service.SuggerimentoLogicImpl**

Nome classe	*SuggerimentoLogicImpl*
Descrizione	Questa classe contiene i metodi fondamentali per la logica di business.
Metodi	+salvaSuggerimento(Suggerimento suggerimento): boolean +ottieniCronologiaSuggerimentiUtente(Integer idUtente): List<Suggerimento> +ottieniSuggerimentiCapi(ImplementorAlgorithm<T> impl, List<T> capi, MeteoDailyMin meteoDaily): List<T>
Invariante di classe	/

Nome metodo	+salvaSuggerimento(Suggerimento suggerimento): boolean
Descrizione	Questo metodo permette di salvare un suggerimento nel DB.
Pre-condizione	**context:** SuggerimentoLogicImpl::salvaSuggerimento(Suggeriemnto suggerimento) **pre:** suggerimento != null && suggerimento.getCitta() != null && suggerimento.getMeteo() != null && suggerimento.getOutfit() != null && suggerimento.getUtente() != null
Post-condizione	/
Nome metodo	+ottieniCronologiaSuggerimentiUtente(Integer idUtente): List<Suggerimento>
Descrizione	Questo metodo permette di ottenere la lista di suggerimenti richiesti dall'utente nel corso dell'utilizzo del sistema.
Pre-condizione	**context**: SuggerimentoLogicImpl::ottieniCittaByName(String name) **pre:** name != null && name.lenght() >= 1
Post-condizione	/
Nome metodo	+ottieniSuggerimentiCapi(ImplementorAlgorithm<T> impl, List<T> listaCapi, MeteoDailyMin meteoDaily): List<T>
Descrizione	Questo metodo permette di ottenere la lista di capi suggeriti dal sistema rispetto alla lista di capi di tipo T.
Pre-condizione	**context:** SuggerimentoLogicImpl::ottieniSuggerimentiCapi(ImplementorAlgorithm<T> impl, List<T> capi, MeteoDaily meteoDaily) **pre:** impl != null && meteoDaily != null && listaCapi.lenght() > 3
Post-condizione	/

Nome metodo	+ottieniOutfitDaSuggerimento(Suggerimento suggerimento): Outfit
Descrizione	Questo metodo permette di ottenere un Outfit creato dall'utente dal database dato il suggerimento associato.
Pre-condizione	**context:** SuggerimentoLogicImpl::ottieniOutfitDaSuggerimento(Suggerimento suggerimento) **pre:** suggerimento.getId()!=null
Post-condizione	**/**

- **weatherstyle.gestionesuggerimentiia.storage.dao.OutfitDAOImpl**

Nome classe	*OutfitDAOImpl*
Descrizione	Questa classe contiene le operazioni che permettono di gestire le informazioni persistenti relative agli outfit di un Utente.
Metodi	+doSaveOutfit(Outfit outfit): boolean +doRetrieveOutfitBySuggerimentoID(int suggerimentoID): Outfit -doRetrieveMagliaByOutfitID(int outfitID): Maglia -doRetrievePantaloneByOutfitID(int outfitID): Pantalone -doRetrieveScarpeByOutfitID(int outfitID): Scarpe -doSaveComporreCapoAbbigliamentoByOutfitID(int capoAbbigliamentoID, int outfitID): boolean -doRetrieveComporreOutfitByCapoAbbigliamentoID(int capoAbbigliamentoID)
Invariante di classe	/

Nome metodo	+doSaveOutfit(Outfit outfit): int
Descrizione	Questo metodo permette di salvare un outfit nel database.
Pre-condizione	/
Post-condizione	/
Nome metodo	+doRetrieveOutfitBySuggerimentoID(int suggerimentoID): Outfit
Descrizione	Questo metodo permette di ottenere un Outfit data la chiave di un suggerimento
Pre-condizione	/
Post-condizione	/
Nome metodo	-settingCapiInOutfit(Outfit outfit): void
Descrizione	Questo metodo permette di inserire in un outfit i capi d'abbigliamento di cui è composto recuperandoli dal DB.
Pre-condizione	/
Post-condizione	/
Nome metodo	-doSaveComporreCapoAbbigliamentoByOutfitID(int capoAbbigliamentoID, int outfitID): boolean
Descrizione	Questo metodo permette di associare un capo d'abbigliamento a un outfit nel DB.
Pre-condizione	/
Post-condizione	/

Nome metodo	-doRetrieveComporreOutfitByCapoAbbigliamentoID(int capoAbbigliamentoID)
Descrizione	Questo metodo permette di verificare se già esiste un capo d’abbigliamento associato a un outfit.
Pre-condizione	/
Post-condizione	/

- **weatherstyle.gestionesuggerimentiia.storage.dao.SuggerimentoDAOImpl**

Nome Classe	*SuggerimentoDAOImpl*
Descrizione	Questa classe contiene le operazioni che permettono di gestire le informazioni riguardanti i suggerimenti sui capi d'abbigliamento più adatti.
Metodi	+doSaveSuggerimento(Suggerimento suggerimento): boolean +doRetrieveCronologiaSuggerimentiByUtenteID(int utenteID): List<Suggerimento>
Invariante Classe	/

Nome metodo	+doSaveSuggerimento(Suggerimento suggerimento): boolean
Descrizione	Questo metodo permette di salvare un suggerimento nel database.
Pre-condizione	**/**
Post-condizione	**/**
Nome metodo	+doRetrieveCronologiaSuggerimentiByUtenteID(int utenteID): List<Suggerimento>
Descrizione	Questo metodo permette di ottenere una lista di tutti i suggerimenti dato un certo utente.
Pre-condizione	**/**
Post-condizione	**/**

- **weatherstyle.gestionesuggerimentiia.applicationlogic.logic.algorithms.AbstractAlgorithm**

Nome classe	*AbstractAlgorithm<T extends CapoAbbigliamento>*
Descrizione	Questa classe permette al client di utilizzare diverse istanze di una determinata implementazione.
Metodi	+getImplementorML():ImplementorAlgorithm<T> +getImplementorGA():ImplementorAlgorithm<T>
Invariante di classe	/

Nome metodo	+getImplementorAlgorithm(): ImplementorAlgorithm<T>
Descrizione	Questo metodo permette di restituire un'istanza di ImplementorAlgorithm.
Pre-condizione	/
Post-condizione	/

- **weatherstyle.gestionesuggerimentiia.applicationlogic.logic.algorithms.ImplementorAlgorithm**

Nome interfaccia	*ImplementorAlgorithm<T extends CapoAbbigliamento>*
Descrizione	Questa interfaccia permette di definire un'interfaccia comune alle diverse implementazioni degli algoritmi di intelligenza artificiale.
Metodi	+getBestThreeCapoAbbigliamento(List<T> lista, MeteoDaily meteoDaily): List<T>
Invariante di classe	/

- **weatherstyle.gestionesuggerimentiia.applicationlogic.logic.algorithms.Genetic Algorithm**

Nome classe	*GeneticAlgorithm<T extends CapoAbbigliamento>*
Descrizione	Questa classe permette di fare il setup di un algoritmo genetico e restituisce i tre capi migliori di tipo T.
Metodi	+getBestThreeCapoAbbigliamento(List<T> lista, MeteoDaily meteoDaily): List<T> -eval(Genotype<IntegerGene> genotype): int
Invariante di classe	/

Nome metodo	+getBestThreeCapoAbbigliamento(List<T> lista, MeteoDailyMin meteoDaily): List<T>
Descrizione	Questo metodo restituisce una lista dei tre capi d'abbigliamento migliori di tipo T.
Pre-condizione	**context:** GeneticAlgorithm::getBestThreeClothes() **pre:** lista.size() >= 3 && meteoDaily != null
Post-condizione	/
Nome metodo	-eval(Genotype<IntegerGene> genotype): int
Descrizione	Questo metodo viene utilizzato internamente alla classe durante il setup dell'algoritmo genetico, rappresenta la funzione di fitness.
Pre-condizione	/
Post-condizione	/

- **weatherstyle.gestionesuggerimentiia.applicationlogic.logic.algorithms.RegressionTreeAlgorithm**

Nome classe	*RegressionTreeAlgorithm<T extends CapoAbbigliamento>*
Descrizione	Questa classe permette di fare il setup del modello di machine learning che lavora su capi d'abbigliamento di tipo T.
Metodi	+getBestThreeCapoAbbigliamento(List<T> lista, MeteoDaily meteoDaily): List<T> -testModelWithTenFoldsCrossValidation():void -createClassBalancer():Filter -classifyInstances(): List<ScoreCapoAbbigliamento>
Invariante di classe	/

Nome metodo	+getBestThreeCapoAbbigliamento(List<T> lista, MeteoDailyMin meteoDaily): List<T>
Descrizione	Questo metodo restituisce una lista dei tre capi d'abbigliamento migliori di tipo T.
Pre-condizione	**context:** RegressionTreeAlgorithm::getBestThreeClothes() **pre:** lista.size() >= 3 && meteoDaily != null
Post-condizione	/
Nome metodo	-testModelWithTenFoldsCrossValidation():void
Descrizione	Questo metodo permette di testare e addestrare il modello.
Pre-condizione	/
Post-condizione	/
Nome metodo	-createClassBalancer(Instances trainingSet):Filter
Descrizione	Questo metodo crea un filtro che permette di fare oversampling e undersampling sui dati di training.
Pre-condizione	/
Post-condizione	/
Nome metodo	-classifyInstance(T capoAbbigliamento, MeteoDailyMin meteoDaily): ScoreCapoAbbigliamento
Descrizione	Questo metodo classifica l'istanza di tipo T passata in input al metodo insieme alle informazioni meteo e restituisce un'istanza di un oggetto che contiene il capo d'abbigliamento passato e il punteggio assegnato dal modello.
Pre-condizione	/

Post-condizione	/

- **weatherstyle.gestionesuggerimentiia.applicationlogic.logic.algorithms.ConstraintGA**

Nome classe	*ConstraintGA*
Descrizione	Classe utilizzata per impostare il vincolo in GeneticGA
Metodi	+test(Phenotype<IntegerGene, Integer> phenotype): boolean +repair(Phenotype<IntegerGene, Integer> phenotype, long l): Phenotype<IntegerGene, Integer>
Invariante di classe	/

Nome metodo	+test(Phenotype<IntegerGene, Integer> phenotype):boolean
Descrizione	Questo metodo verifica che phenotype rispetti il vincolo. Restituisce true se il vincolo è rispettato, false altrimenti.
Pre-condizione	/
Post-condizione	/
Nome metodo	+repair(Phenotype<IntegerGene, Integer> phenotype, long l):Phenotype<IntegerGene, Integer>
Descrizione	Questo metodo restituisce un nuovo Phenotype<IntegerGene, Integer> che rispetti il vincolo.
Pre-condizione	/
Post-condizione	/

- **weatherstyle.gestionesuggerimentiia.applicationlogic.logic.algorithms.EvaluatorGA**

Nome classe	*EvaluatorGA*
Descrizione	Classe utilizzata da GeneticAlgorithm per valutare le istanze e assegnare loro un punteggio.
Metodi	+valuta(CapoAbbigliamento capo, MeteoDailyMin meteoDaily): int -valutazioneScarpe(CapoAbbigliamento capo, MeteoDailyMin meteoDaily): int -valutazioneMagliaOrPantaloni(CapoAbbigliamento capo, MeteoDailyMin meteoDaily): int -searchRange(int temperaturaPercepita): int -valutazioneColore(CapoAbbigliamento capo, MeteoDailyMin meteoDaily): int -valutazioneLunghezza(CapoAbbigliamento capo, MeteoDailyMin meteoDaily): int -valutazioneStagionePrevisione(CapoAbbigliamento capo, MeteoDaily meteoDailyMin): int -valutazionePioggia(Scarpe scarpe, MeteoDailyMin meteoDaily): int -valutazioneTipoScarpe(Scarpe scarpe, MeteoDailyMin meteoDaily): int -initTable(): void
Invariante di classe	/

Nome metodo	+valuta(CapoAbbigliamento capo, MeteoDailyMin meteoDaily): int
Descrizione	Questo metodo serve a valutare le istanze di capi d'abbigliamento.
Pre-condizione	/
Post-condizione	/
Nome metodo	-valutazioneScarpe(CapoAbbigliamento capo, MeteoDailyMin meteoDaily): int
Descrizione	Questo metodo permette di valutare le scarpe.
Pre-condizione	/
Post-condizione	/
Nome metodo	-valutazioneMagliaOrPantaloni(CapoAbbigliamento capo, MeteoDailyMin meteoDaily): int
Descrizione	Questo metodo permette di valutare le maglie o i pantaloni a seconda del tipo di capo.
Pre-condizione	/
Post-condizione	/

Nome metodo	-searchRange(int temperaturaPercepita): int
Descrizione	Questo metodo permette di ottenere l'indice da una lista di tabelle hash con punteggi definiti rispetto ai range di temperatura.
Pre-condizione	/
Post-condizione	/
Nome metodo	-valutazioneColore(CapoAbbigliamento capo, MeteoDailyMin meteoDaily): int
Descrizione	Questo metodo permette di valutare il colore di un capo d'abbigliamento.
Pre-condizione	/
Post-condizione	/
Nome metodo	-valutazioneTemperatura(CapoAbbigliamento capo, MeteoDailyMin meteoDaily): int
Descrizione	Questo metodo permette di valutare un capo d'abbigliamento rispetto alla temperatura.
Pre-condizione	/
Post-condizione	/
Nome metodo	-valutazioneLunghezza(CapoAbbigliamento capo, MeteoDailyMin meteoDaily): int
Descrizione	Questo metodo permette di valutare la lunghezza delle maniche di una maglia o di un pantalone.
Pre-condizione	/
Post-condizione	/
Nome metodo	-valutazioneStagionePrevisione(CapoAbbigliamento capo, MeteoDailyMin meteoDaily): int
Descrizione	Questo metodo permette di valutare un capo d'abbigliamento rispetto alla stagione della previsione.
Pre-condizione	/
Post-condizione	/
Nome metodo	-valutazionePioggia(Scarpe scarpe, MeteoDailyMin meteoDaily): int
Descrizione	Questo metodo permette di valutare le scarpe rispetto alla pioggia.
Pre-condizione	/

Post-condizione	/
Nome metodo	-valutazioneTipoScarpe(Scarpe scarpe, MeteoDailyMin meteoDaily): int
Descrizione	Questo metodo permette di valutare le scarpe rispetto alla loro tipologia.
Pre-condizione	/
Post-condizione	/
Nome metodo	-initTable(): void
Descrizione	Inizializza delle tabelle hash per definire i punteggi necessari nelle varie valutazioni.
Pre-condizione	/
Post-condizione	/

3.4 Package gestioneambiente

- **weatherstyle.gestioneambiente.strorage.dao.EventoDAOImpl**

Nome Classe	*EventoDAOImpl*
Descrizione	Questa classe contiene le operazioni che permettono di gestire le informazioni persistenti riguardanti gli eventi per salvaguardare l'ambiente.
Metodi	+doRetrieveById(int idEvento): Evento +doSaveEvento(Evento evento): boolean +doRetrieveAfterCurrentDate(): List<Evento> +doRetrieveAll(): List<Evento>
Invariante Classe	/

Nome metodo	+doRetrieveById(int idEvento): Evento
Descrizione	Questo metodo restituisce un Evento dato il suo id, se esiste, altrimenti restituisce null.
Pre-condizione	/
Post-condizione	/
Nome metodo	+doSaveEvento(Evento evento): boolean
Descrizione	Questo metodo permette di salvare un evento nel database. Se va a buon fine allora restituisce true, altrimenti false.
Pre-condizione	**context:** EventoDAOImpl::doSaveEvento(Evento Evento) **pre:** evento != null
Post-condizione	**context:** EventoDAOImpl::doSaveEvento(Evento Evento) **post:** doRetrieveById(evento.getId()) != null
Nome metodo	+doRetrieveAfterCurrentDate(): List<Evento>
Descrizione	Questo metodo restituisce una lista di quegli eventi che si svolgono in un momento successivo rispetto alla data e l'ora in cui questo metodo viene invocato.
Pre-condizione	/
Post-condizione	/
Nome metodo	+doRetrieveAll(): List<Evento>
Descrizione	Questo metodo restituisce una lista di tutti gli eventi salvati nel database.
Pre-condizione	/
Post-condizione	/

- **weatherstyle.gestioneambiente.applicationlogic.service.EventoLogicImpl**

Nome Classe	*EventoLogicImpl*
Descrizione	Questa classe contiene le operazioni che permettono di gestire la logica di business riguardanti gli eventi per salvaguardare l'ambiente.
Metodi	+ottieniEventoPerId(int idEvento): Evento +salvaEvento(Evento evento): boolean +ottieniListaEventiFuturi(): List<Evento> +ottieniListaEventi(): List<Evento>

Invariante Classe	/

Nome metodo	+ottieniEventoPerId(int idEvento): Evento
Descrizione	Questo metodo restituisce un Evento dato il suo id, se esiste, altrimenti restituisce null.
Pre-condizione	/
Post-condizione	/
Nome metodo	+salvaEvento(Evento evento): boolean
Descrizione	Questo metodo permette di salvare un evento. Se il metodo va a buon fine restituisce true, altrimenti false.
Pre-condizione	**context:** EventoLogicServiceI::salvaEvento(Evento Evento) **pre:** evento != null
Post-condizione	**context:** EventoLogicService::salvaEvento(Evento Evento) **post:** ottieniEventoPerId(evento.getId()) != null
Nome metodo	+ottieniListaEventiFuturi(): List<Evento>
Descrizione	Questo metodo restituisce una lista di quegli eventi che si svolgono in un momento successivo alla data e l'ora corrente in cui questo metodo viene invocato.
Pre-condizione	/
Post-condizione	/
Nome metodo	+ottieniListaEventi(): List<Evento>
Descrizione	Questo metodo restituisce una lista di tutti gli eventi salvati precedentemente.
Pre-condizione	/
Post-condizione	/

- **weatherstyle.gestioneambiente.storage.dao.RichiestaPromozioneDAOImpl**

Nome Classe	*RichiestaPromozioneDAOImpl*
Descrizione	Questa classe contiene le operazioni che permettono di gestire le informazioni persistenti riguardanti le richieste di promozione da Utente ad Ecologista.

Metodi	+doSaveRichiestaPromozione(RichiestaPromozione richiestaPromozione): boolean +doRetrieveAll(): List<RichiestaPromozione> +doUpdateStato(RichiestaPromozione richiestaPromozione, String nuovoStato, Admin admin): boolean +doRetrieveRichiestaPromozioneByStato(String stato): List<RichiestaPromozione> +doRetrieveByIdUtente(int idUtente): RichiestaPromozione +doRetrieveById(int idRichiestaPromozione): RichiestaPromozione
Invariante Classe	/

Nome metodo	+doSaveRichiestaPromozione(RichiestaPromozione richiestaPromozione): boolean
Descrizione	Questo metodo permette di salvare una richiesta di promozioni ad Ecologista nel database. Se il metodo va a buon fine restituisce true, altrimenti false.
Pre-condizione	**context:** RichiestaPromozioneDAOImpl::doSaveRichiestaPromozione (RichiestaPromozione richiestaPromozione) **pre:** richiestaPromozione != null
Post-condizione	**context:** RichiestaPromozioneDAOImpl:: doSaveRichiestaPromozione(RichiestaPromozione richiestaPromozione) **post:** doRetrieveById(richiestaPromozione.getId()) != null
Nome metodo	+doRetrieveAll(): List<RichiestaPromozione>
Descrizione	Questo metodo restituisce una lista di tutte le richieste di promozione salvate nel database.
Pre-condizione	**/**
Post-condizione	**/**
Nome metodo	+doUpdateStato(RichiestaPromozione richiestaPromozione, String nuovoStato, Admin admin): boolean
Descrizione	Questo metodo aggiorna lo stato di una richiesta di promozione fornita in input in nuovoStato. E imposta l'id dell'amministratore che ha effettuato l'aggiornamento dello stato. Se il metodo va a buon fine restituisce true, altrimenti false.

Pre-condizione	**context:** RichiestaPromozioneDAOImpl:: doUpdateStatoById(RichiestaPromozione richiestaPromozione, String nuovoStato, Admin admin) **pre:** richiestaPromozione != null && nuovoStato != null && admin != null
Post-condizione	**context:** RichiestaPromozioneDAOImpl:: doUpdateStatoById(RichiestaPromozione richiestaPromozione, String nuovoStato, Admin admin) **post:** richiestaPromozione.getStato().equals(nuovoStato) && richiestaPromozione.getAdmin() == admin
Nome metodo	+doRetrieveRichiestaPromozioneByStato(String stato): List<RichiestaPromozione>
Descrizione	Questo metodo restituisce una lista di richieste di promozione che si trovano in un determinato stato.
Pre-condizione	**context:** RichiestaPromozioneDAOImpl:: doRetrieveRichiestaPromozioneByStato(String stato) **pre:** stato != null
Post-condizione	/
Nome metodo	+doRetrieveByIdUtente(int idUtente): RichiestaPromozione
Descrizione	Questo metodo restituisce una richiesta di promozione dato l'id dell'utente che l'ha richiesta. Se quell'id è associato ad un utente che non ha richiesto la promozione ad Ecologista oppure non esiste, allora restituirà null.
Pre-condizione	/
Post-condizione	/
Nome metodo	+doRetrieveById(int idRichiestaPromozione): RIchiestaPromozione
Descrizione	Questo metodo restituisce una richiesta di promozione dato il suo id, restituisce invece null se l'id non esiste.
Pre-condizione	/
Post-condizione	/

- **weatherstyle.gestioneambiente.applicationlogic.service.RichiestaPromozioneLogicImpl**

Nome Classe	*RichiestaPromozioneLogicImpl*

Descrizione	Questa classe contiene le operazioni che permettono di gestire la logica di business riguardante le richieste di promozione da Utente ad Ecologista.
Metodi	+ottieniRichiestaPromozionePerIDUtente(int idUtente): RichiestaPromozione +salvaRichiestaPromozione(RichiestaPromozione richiestaPromozione): boolean +ottieniListaRichiestaPromozione(): List<RichiestaPromozione> +ottieniListaRichiestePromozionePerStato(String stato): List<RichiestaPromozione> +aggiornaStatoRichiestaPromozione(RichiestaPromozione richiestaPromozione, String nuovoStato, Admin admin): boolean +ottieniRIchiestaPromozionePerId(int idRichiestaPromozione): RichiestaPromozione
Invariante Classe	/

Nome metodo	+ottieniRichiestaPromozionePerIDUtente(int idUtente): RichiestaPromozione
Descrizione	Questo metodo restituisce la richiesta di promozione dato l'id dell'utente, se l'utente non ha mai richiesto la promozione ad ecologista allora restituirà null.
Pre-condizione	**/**
Post-condizione	**/**
Nome metodo	+salvaRichiestaPromozione(RichiestaPromozione richiestaPromozione): boolean
Descrizione	Questo metodo permette di salvare una richiesta di promozione da utente a ecologista. Se il metodo va a buon fine restituisce true, altrimenti false.
Pre-condizione	**context:** RichiestaPromozioneLogicService::salvaRichiestaPromozione (RichiestaPromozione richiestaPromozione) **pre:** richiestaPromozione != null
Post-condizione	**context:** RichiestaPromozioneLogicService:: salvaRichiestaPromozione(RichiestaPromozione richiestaPromozione) **post:**

	ottieniRichiestaPromozionePerId(richiestaPromozione.getId()) != null
Nome metodo	+ottieniListaRichiestaPromozione(): List<RichiestaPromozione>
Descrizione	Questo metodo restituisce una lista che contiene tutte le richieste di promozione effettuate dagli utenti.
Pre-condizione	/
Post-condizione	/
Nome metodo	+ottieniListaRichiestePromozionePerStato(String stato): List<RichiestaPromozione>
Descrizione	Questo metodo restituisce una lista che contiene tutte le richieste di promozione che si trovano in un determinato stato.
Pre-condizione	**context:** RichiestaPromozioneLogicService:: ottieniListaRichiestePromozionePerStato(String stato) **pre:** stato != null
Post-condizione	/
Nome metodo	+aggiornaStatoRichiestaPromozione(RichiestaPromozione richiestaPromozione, String nuovoStato, Admin admin): boolean
Descrizione	Questo metodo permette di aggiornare lo stato di una determinata richiesta di promozione in nuovoStato. Inoltre aggiorna l'id dell'amministratore che ha eseguito l'aggiornamento. Se l'operazione va a buon fine restituisce true, altrimenti false.
Pre-condizione	**context:** RichiestaPromozioneLogicService:: aggiornaStatoRichiestaPromozione(RichiestaPromozione richiestaPromozione, String nuovoStato, Admin admin) **pre:** richiestaPromozione != null && nuovoStato != null && admin != null
Post-condizione	**context:** RichiestaPromozioneLogicService:: aggiornaStatoRichiestaPromozione(RichiestaPromozione richiestaPromozione, String nuovoStato, Admin admin) **post:** richiestaPromozione.getStato().equals(nuovoStato) && richiestaPromozione.getAdmin() == admin
Nome metodo	+ottieniRIchiestaPromozionePerId(int idRichiestaPromozione): RichiestaPromozione
Descrizione	Questo metodo restituisce una richiesta di promozione dato il suo id. Se non esiste restituisce null.
Pre-condizione	/

Post-condizione	/

3.5 Package gestioneutente

- **weatherstyle.gestioneutenti.storage.dao.UtenteDAOImpl**

Nome Classe	*UtenteDAOImpl*
Descrizione	Questa classe contiene le operazioni che permettono di gestire le informazioni dell'Utente per l'interazione con il DB
Metodi	+doSaveUtente(Utente utente):boolean +doRetrieveUtenteByID(int id):Utente +doRetrieveUtenteByEmailAndPassword(String email, String password):Utente

	+doExistsEmail(String email):boolean
Invariante Classe	/

Nome metodo	+doSaveUtente(Utente utente):boolean
Descrizione	Metodo che permette di salvare un utente
Pre-condizione	**context:** UtenteDAO::doSaveUtente(Utente utente) **pre**: utente != null && doExistsEmail(utente.getEmail()) == false
Post-condizione	**/**
Nome metodo	+doRetrieveUtenteByID(int id):Utente
Descrizione	Metodo che permette di ottenere un Utente tramite il suo id.
Pre-condizione	**context:** UtenteDAO::doRetrieveUtenteByID(int id) **pre**: id > 0
Post-condizione	**/**
Nome metodo	+doRetrieveUtenteByEmailAndPassword(String email, String password):Utente
Descrizione	Metodo che permette di ottenere un Utente tramite email e password.
Pre-condizione	**context:** UtenteDAO::doRetrieveUtenteByEmailAndPassword(String email, String password) **pre**: email != null && password != null && email.matches("^[a-z0-9\\._]+@[a-z]+\\.[a-z]{2,3}$")
Post-condizione	**/**
Nome metodo	+doExistsEmail(String email):boolean
Descrizione	Metodo che permette di ottenere un Utente tramite il suo id.
Pre-condizione	**context:** UtenteDAO:doExistsEmail(String email) **pre**: email != null&& email.matches("^[a-z0-9\\._]+@[a-z]+\\.[a-z]{2,3}$")
Post-condizione	**/**

- **weatherstyle.gestioneutenti.logic.service.UtenteLogicServiceImpl**

Nome Classe	*UtenteLogicServiceImpl*
Descrizione	Questa classe contiene le operazioni che permettono di gestire le informazioni dell'Utente
Metodi	+registraUtente(Utente utente):boolean +loginUtente(String email, String password):Utente +promuoviUtenteAdEcologista():Ecologista +existsUtente(String email):boolean
Invariante Classe	/

Nome metodo	+registraUtente(Utente utente):boolean
Descrizione	Questo metodo permette di registrare un utente sul database
Pre-condizione	**context:** UtenteLogicServiceImpl::registraUtente(Utente utente) **pre:** utente != null && existsUtente(utente.getEmail()) == false
Post-condizione	**context:** UtenteLogicServiceImpl::registraUtente(Utente utente) **pre:** existsUtente(utente.getEmail()) == true
Nome metodo	+loginUtente(String email, String password):Utente
Descrizione	Questo metodo permette di verificare che email password siano quelli di un Utente
Pre-condizione	**context:** UtenteLogicServiceImpl::loginUtente(String email, String password) **pre:** email != null && password != null
Post-condizione	**/**
Nome metodo	+existsUtente(String email):boolean
Descrizione	Questo metodo permette di verificare se esiste o meno un utente con email *email*
Pre-condizione	**context:** UtenteLogicServiceImpl::existsUtente(String email) **pre:** email!= null
Post-condizione	**/**

- **weatherstyle.gestioneutenti.logic.service.AdminLogicServiceImpl**

Nome Classe	**AdminLogicServiceImpl**
Descrizione	Questa classe contiene le operazioni che permettono di gestire le informazioni dell'Admin
Metodi	+loginAdmin(String email,String password):Admin
Invariante Classe	/

Nome metodo	+loginAdmin(String email,String password):Admin
Descrizione	Questo metodo permette di verificare che email password siano quelli di un Admin
Pre-condizione	**context:** AdminLogicServiceImpl::loginAdmin(String email,String password) **pre:** email!= null
Post-condizione	**/**

- **weatherstyle.gestioneutenti.storage.dao.AdminDAOImpl**

Nome Classe	*AdminDAOImpl*
Descrizione	Questa classe contiene le operazioni che permettono di gestire le informazioni di un amministratore.
Metodi	+doRetrieveAdminByEmailAndPassword(String email, String password):Admin
Invariante Classe	/

3.6 Package gestioneguardaroba

- **weatherstyle.gestioneguardaroba.storage.dao.GuardarobaDAOImpl**

Nome Classe	*GuardarobaDAOImpl*
Descrizione	Questa classe contiene le operazioni che permettono di gestire le informazioni del Guardaroba per l'interazione con il database.
Metodi	+doRetrieveGuardarobaById (int id): Guardaroba +doSaveGuardaroba (int idUtente): boolean +doRetrieveNumeroCapi (int idUtente): int +doSaveNumeroCapi (int idUtente, int numero): boolean
Invariante Classe	/

Nome metodo	+doRetrieveGuardarobaById (int id): Guardaroba
Descrizione	Questo metodo permette di ottenere un guardaroba a partire dal suo ID.
Pre-condizione	**context:** GuardarobaDAOImpl::doRetrieveGuardarobaById(int id) **pre:** id > 0
Post-condizione	**/**
Nome metodo	+doSaveGuardaroba (int idUtente): boolean
Descrizione	Questo metodo permette di salvare un guardaroba nel database. Se il metodo va a buon fine restituisce true, altrimenti false.
Pre-condizione	**context:** GuardarobaDAOImpl::doSaveGuardaroba(int idUtente) **pre:** idUtente > 0
Post-condizione	**/**
Nome metodo	+doRetrieveNumeroCapi (int idUtente): int
Descrizione	Questo metodo permette di ottenere il numero di capi d'abbigliamento presenti nel guardaroba dell'utente con l'ID idUtente.
Pre-condizione	**context**: GuardarobaDAOImpl::doRetrieveNumeroCapi (int idUtente) **pre:** idUtente > 0
Post-condizione	**/**
Nome metodo	+doSaveNumeroCapi (int idUtente, int numero): boolean

Descrizione	Questo metodo permette di salvare il numero di capi d'abbigliamento presenti nel guardaroba dell'utente con l'ID idUtente. Se il metodo va a buon fine restituisce true, altrimenti false.
Pre-condizione	**context:** GuardarobaDAOImpl::doSaveNumeroCapi (int idUtente, int numero) **pre:** idUtente > 0 && numero > 0
Post-condizione	**/**

- **weatherstyle.gestioneguardaroba.storage.dao.CapoAbbigliamentoDAOImpl**

Nome Classe	*CapoAbbigliamentoDAOImpl*
Descrizione	Questa classe contiene le operazioni che permettono di gestire le informazioni dei capi d'abbigliamento per l'interazione con il database.
Metodi	+doRetrieveCapoById (int idCapoAbbigliamento): CapoAbbigliamento +doRetrieveMagliaByIdCapoAbbigliamento (int idCapoAbbigliamento): Maglia +doRetrievePantaloniByIdCapoAbbigliamento (int idCapoAbbigliamento): Pantaloni +doRetrieveScarpeByIdCapoAbbigliamento (int idCapoAbbigliamento): Scarpe +doSaveMaglia (Maglia m, int idGuardaroba): boolean +doSavePantaloni (Pantaloni p, int idGuardaroba): boolean +doSaveScarpe (Scarpe s, int idGuardaroba): boolean +doRetrieveMaglieByIdGuardaroba (int idG): List <Maglia> +doRetrievePantaloniByIdGuardaroba (int idG): List <Pantaloni> +doRetrieveScarpeByIdGuardaroba (int dG): List<Scarpe>
Invariante Classe	/

Nome metodo	+doRetrieveCapoById (int idCapoAbbigliamento): CapoAbbigliamento
Descrizione	Questo metodo permette di ottenere un capo d'abbigliamento dal database in base al proprio ID.

Pre-condizione	**context:** CapoAbbigliamentoDAOImpl::doRetrieveCapoById (int idCapoAbbigliamento) **pre:** idCapoAbbigliamento > 0
Post-condizione	**/**
Nome metodo	+doRetrieveMagliaByIdCapoAbbigliamento (int idCapoAbbigliamento): Maglia
Descrizione	Questo metodo permette di ottenere una maglia in base al proprio ID.
Pre-condizione	**context**: GuardarobaDAOImpl::doRetrieveMagliaByIdCapoAbbigliamento (int idCapoAbbigliamento) **pre**: idCapoAbbigliamento > 0
Post-condizione	/
Nome metodo	+doRetrievePantaloniByIdCapoAbbigliamento (int idCapoAbbigliamento): Pantaloni
Descrizione	Questo metodo permette di ottenere dei pantaloni in base al proprio ID.
Pre-condizione	**context:** GuardarobaDAOImpl::doRetrievePantaloniByIdCapoAbbigliamento (int idCapoAbbigliamento) **pre:** idCapoAbbigliamento > 0
Post-condizione	/
Nome metodo	+doRetrieveScarpeByIdCapoAbbigliamento (int idCapoAbbigliamento): Scarpe
Descrizione	Questo metodo permette di ottenere delle scarpe in base al proprio ID.
Pre-condizione	**context:** GuardarobaDAOImpl::doRetrieveScarpeByIdCapoAbbigliamento (int idCapoAbbigliamento) **pre:** idCapoAbbigliamento > 0
Post-condizione	/
Nome metodo	+doSaveMaglia (Maglia m, int idGuardaroba): boolean
Descrizione	Questo metodo permette di salvare una maglia in un guardaroba. Se il metodo va a buon fine restituisce true, altrimenti false.

Pre-condizione	**context**: GuardarobaDAOImpl::doSaveMaglia(Maglia m, int idGuardaroba) **pre:** m != null && idGuardaroba > 0
Post-condizione	/
Nome metodo	+doSavePantaloni (Pantaloni p, int idGuardaroba): boolean
Descrizione	Questo metodo permette di salvare dei pantaloni in un guardaroba. Se il metodo va a buon fine restituisce true, altrimenti false.
Pre-condizione	**context:** GuardarobaDAOImpl::doSavePantaloni(Pantaloni p, int idGuardaroba) **pre:** p != null && idGuardaroba > 0
Post-condizione	/
Nome metodo	+doSaveScarpe (Scarpe s, int idGuardaroba): boolean
Descrizione	Questo metodo permette di salvare delle scarpe in un guardaroba. Se il metodo va a buon fine restituisce true, altrimenti false.
Pre-condizione	**context**: GuardarobaDAOImpl::doSaveScarpe(Scarpe s, int idGuardaroba) **pre:** s != null && idGuardaroba > 0
Post-condizione	/
Nome metodo	+doRetrieveMaglieByIdGuardaroba (int idG): List <Maglia>
Descrizione	Questo metodo permette di ottenere la lista di maglie presenti nel guardaroba.
Pre-condizione	**context:** GuardarobaDAOImpl::doRetrieveMaglieByIdGuardaroba (int dG) **pre:** idG > 0
Post-condizione	/
Nome metodo	+doRetrievePantaloniByIdGuardaroba (int idG): List <Pantaloni>
Descrizione	Questo metodo permette di ottenere la lista di pantaloni presenti nel guardaroba.
Pre-condizione	**context**: GuardarobaDAOImpl::doRetrievePantaloniByIdGuardaroba (int dG) **pre:** idG > 0
Post-condizione	/

Nome metodo	+doRetrieveScarpeByIdGuardaroba (int idG): List <Scarpe>
Descrizione	Questo metodo permette di ottenere la lista di scarpe presenti nel guardaroba.
Pre-condizione	**context:** GuardarobaDAOImpl::doRetrieveScarpeByIdGuardaroba (int dG) **pre:** idG > 0
Post-condizione	/

- **weatherstyle.gestioneguardaroba.applicationlogic.logic.service.GuardarobaLogicImpl**

Nome Classe	*GuardarobaLogicImpl*
Descrizione	Questa classe contiene le operazioni che permettono di gestire le informazioni dei capi d'abbigliamento contenuti nel guardaroba.
Metodi	+salvaMaglia (Maglia m, int idGuardaroba) throws ErrorParameterException: boolean +salvaPantaloni (Pantaloni p, int idGuardaroba) throws ErrorParameterException: boolean +salvaScarpe (Scarpe s, int idGuardaroba) throws ErrorParameterException: boolean +getMaglie (int idUtente): List<Maglia> +getPantaloni (int idUtente): List<Pantaloni> +getScarpe (int idUtente): List<Scarpe) +getAll (int idUtente): List<CapoAbbigliamento> +aggiornaNumeroCapi (int idUtente): boolean
Invariante Classe	/

Nome metodo	+salvaMaglia (Maglia m, int idGuardaroba) throws ErrorParameterException: boolean
Descrizione	Questo metodo permette di salvare una maglia nel guardaroba. Restituisce true se l'esecuzione del metodo è andata a buon fine, false altrimenti.

Pre-condizione	**context**: GuardarobaLogicImpl::salvaMaglia (Maglia m, int idGuardaroba) **pre:** m != null && idGuardaroba > 0
Post-condizione	/
Nome metodo	+salvaPantaloni (Pantaloni p, int idGuardaroba) throws ErrorParameterException: boolean
Descrizione	Questo metodo permette di salvare dei pantaloni nel guardaroba. Restituisce true se l'esecuzione del metodo è andata a buon fine, false altrimenti.
Pre-condizione	**context**: GuardarobaLogicImpl::salvaPantaloni (Panaloni p, int idGuardaroba) **pre**: p != null && idGuardaroba > 0
Post-condizione	/
Nome metodo	+salvaScarpe (Scarpe s, int idGuardaroba) throws ErrorParameterException: boolean
Descrizione	Questo metodo permette di salvare delle scarpe nel guardaroba. Restituisce true se l'esecuzione del metodo è andata a buon fine, false altrimenti.
Pre-condizione	**context**: GuardarobaLogicImpl::salvaScarpe (Scarpe s, int idGuardaroba) **pre:** s != null && idGuardaroba > 0
Post-condizione	/
Nome metodo	+getMaglie (int idUtente): List<Maglia>
Descrizione	Questo metodo permette di ottenere la lista delle maglie presenti nel guardaroba.
Pre-condizione	**context:** GuardarobaLogicImpl::getMaglie (int idUtente) **pre:** idUtente > 0
Post-condizione	/
Nome metodo	+getPantaloni (int idUtente): List<Pantaloni>
Descrizione	Questo metodo permette di ottenere la lista dei pantaloni presenti nel guardaroba.
Pre-condizione	**context:** GuardarobaLogicImpl::getPantaloni (int idUtente) **pre:** idUtente > 0
Post-condizione	/

Nome metodo	+getScarpe (int idUtente): List<Scarpe>
Descrizione	Questo metodo permette di ottenere la lista delle scarpe presenti nel guardaroba.
Pre-condizione	**context:** GuardarobaLogicImpl::getScarpe (int idUtente) **pre:** idUtente > 0
Post-condizione	/
Nome metodo	+getAll (int idUtente): List<CapoAbbigliamento>
Descrizione	Questo metodo permette di ottenere la lista di tutti i capi d'abbigliamento presenti nel guardaroba.
Pre-condizione	**context:** GuardarobaLogicImpl::getAll (int idUtente) **pre:** idUtente > 0
Post-condizione	/
Nome metodo	+aggiornaNumeroCapi (int idUtente): boolean
Descrizione	Questo metodo permette di aggiornare il numero di capi d'abbigliamento presenti nel guardaroba. Restituisce true se l'esecuzione del metodo è andato a buon fine, false altrimenti.
Pre-condizione	**context**: GuardarobaLogicImpl::aggiornaNumeroCapi (int idUtente) **pre:** idUtente > 0
Post-condizione	/

4. Class Diagram Ristrutturato

Il Class Diagram ristrutturato comprendente gli oggetti soluzione sono disponibili al seguente link: class diagram ristrutturato
Per leggibilità si è preferito non inserire le classi servlet, ma solo le classi dei package logic e storage delle diverse gestioni, escludendo il package utils.

5. Elementi di Riuso

Nella seguente sezione discuteremo dei design pattern che utilizzeremo per adottare il riuso, in particolare saranno usati due design pattern:

- Adapter Pattern
- Bridge Pattern

5.1. Design Pattern Usati

Bridge Pattern

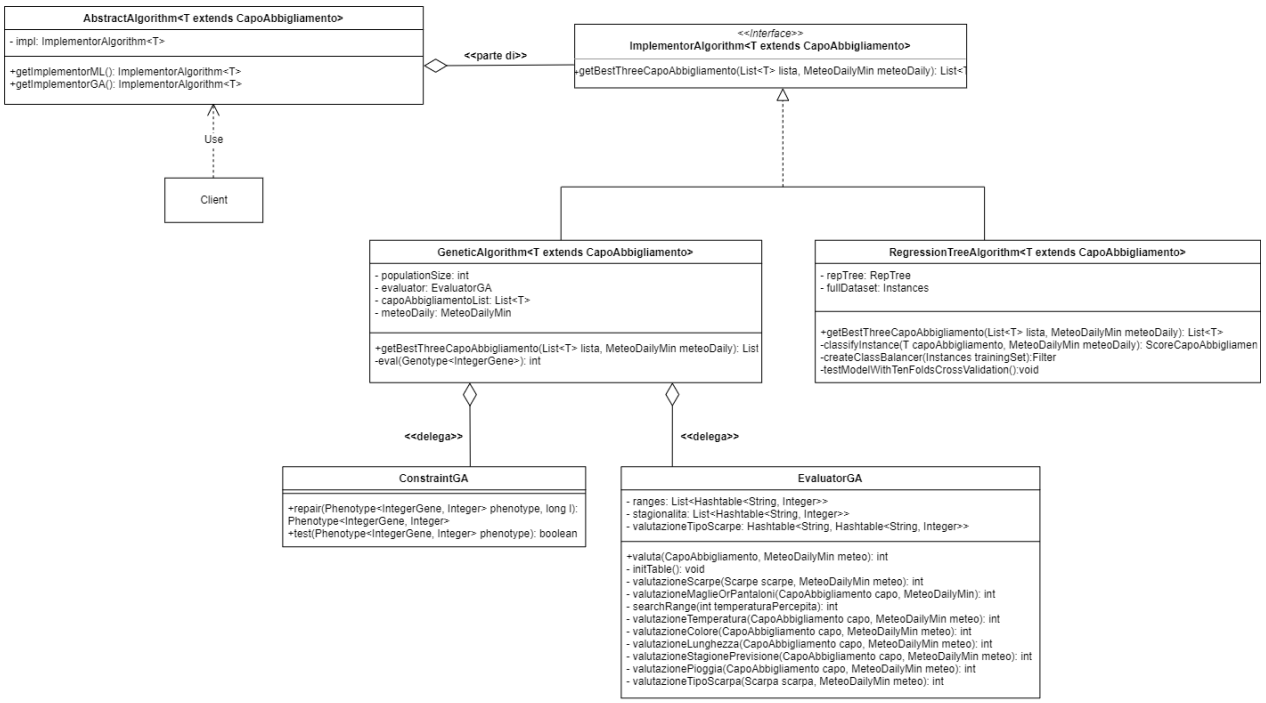
Il Bridge Pattern è un design pattern strutturale che si adatta bene in situazioni in cui si vogliono utilizzare diverse implementazioni che condividono un interfaccia comune.
Principalmente viene adottato in fase di sviluppo quando non sono ancora state realizzate o raffinate completamente determinate componenti.

Visto che il sistema WeatherStyle utilizza delle componenti di intelligenza artificiale, in particolare, algoritmi genetici e machine learning, e visto che queste hanno ampi margini di miglioramento, vogliamo avere la possibilità di incrementare le possibili soluzioni.
Si è deciso quindi di realizzare un'interfaccia comune **ImplementorAlgorithm** che permetta al Client di utilizzare allo stesso modo una o più delle possibili implementazioni.

Si è realizzata la classe **AbstractAlgorithm** con la quale il Client può chiamare i metodi **getImplementorGA()** o **getImplementorML()** per ottenere un'istanza di tipo **ImplementorAlgorithm**

L'istanza **ImplementorAlgorithm** è parte di **AbstractAlgorithm**.

Di seguito il class diagram in cui è possibile avere visione della modalità in cui è stato adattato il design pattern:



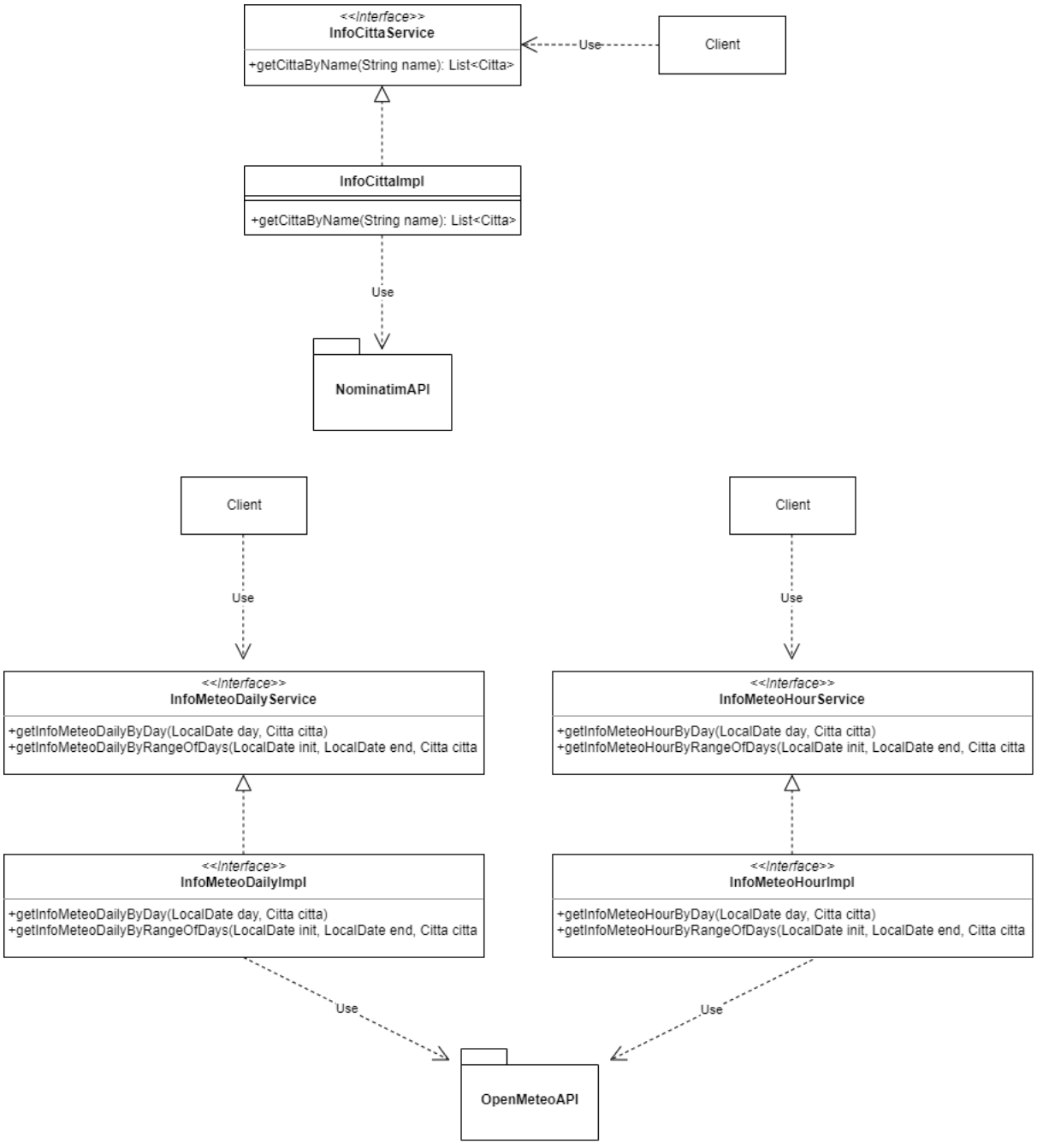
Adapter pattern

L'adapter pattern è un design pattern strutturale utilizzato quando si ha necessità di far comunicare un sistema in sviluppo con un altro sistema già esistente, in modo tale che i dati restituiti siano di un formato compatibile con l'applicativo che si sta sviluppando.

All'interno del sistema sono previste due interfacce adapter per far comunicare il Client con l' l'API relativa alle città e l'API relativa alle informazioni meteorologiche.

I dati restituiti da entrambi le API rispettano il formato JSON, nel nostro sistema convertiamo queste informazioni in oggetti utilizzabili.

Di seguito i class diagram relativi all'utilizzo del design pattern per entrambe le interfacce sviluppate:



5.2.Componenti Terze parti

Alcune parti del sistema sono state realizzate utilizzando componenti di terze parti. Di seguito un elenco che le riassume:

- Nominatim API : API per ottenere informazioni sulle città.
- Open-meteo API : API per ottenere informazioni meteorologiche.
- Bootstrap: framework per la realizzazione di pagine web.
- Jenetics : framework java per la realizzazione di Algoritmi Genetici.
- Weka: framework java per l'addestramento di modelli di Machine Learning.

Per il testing sono stati utilizzati:

- JUnit : Testing d'unità.
- Selenium : Testing di sistema.

Per verificare il rispetto dei vincoli imposti al paragrafo 1.1 è utilizzato il plugin maven Checkstyle, inoltre, per automatizzare l'applicazione di alcune regole imposte dalla convezione è stato utilizzato il plugin maven Rewrite.