

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ODD  Object Design Document CoralloSmart   |  |  | | --- | --- | | Riferimento |  | | Versione | 0.3 | | Data | 7/11/2022 | | Destinatario |  | | Presentato da | Tutto il Team | | Approvato da |  | |

RevisionHistory

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versione** | **Descrizione** | **Autori** |
| 8/12/2022 | 0.1 | Prima stesura | AG, EP, FP, GA, GG |
| 12/12/2022 | 0.2 | Stesura COTS | AG, EP, FP, GA, GG |
| 15/12/2022 | 0.3 | Stesura Object Design Trade offs, linee guida documentazione, “Descrizione, Acronimi e Abbreviazioni”, “Riferimenti” | AG, EP, FP, GA, GG |
| 17/12/2022 | 0.3 | Revisione | Tutto il team |

# Composizione Team e Acronimi

|  |  |  |
| --- | --- | --- |
| **Nome e Cognome** | **Matricola** | **Acronimo** |
| Attilio Gismondi | 0512105896 | AG |
| Ersilio Petretta | 0512110513 | EP |
| Francesco Perillo | 0512106817 | FP |
| Giuseppe Adinolfi | 0512111935 | GA |
| Graziano Giuseffi | 0512104852 | GG |

Sommario

[RevisionHistory 2](#_Toc122883288)

[Composizione Team e Acronimi 3](#_Toc122883289)

[1. Introduzione 5](#_Toc122883290)

[1.1. Componenti Off-The-Shelf 5](#_Toc122883291)

[1.2. Object Design Trade-Offs 8](#_Toc122883292)

[1.3. Linee guida per la documentazione 9](#_Toc122883293)

[1.4. Definizioni, Acronimi e Abbreviazioni 12](#_Toc122883294)

[1.5. Riferimenti 12](#_Toc122883295)

[1.6. JavaDoc 12](#_Toc122883296)

[2. Packages 13](#_Toc122883297)

[3. Interfaccia delle classi 13](#_Toc122883298)

[4. Design Pattern con Class Diagram 13](#_Toc122883299)

[4.1. Factory Pattern 13](#_Toc122883300)

[4.2. Decomposizione in sottosistemi 13](#_Toc122883301)

[5. Glossario 13](#_Toc122883302)

1. Introduzione
   1. Componenti Off-The-Shelf
      1. SQL

Structured Query Language (SQL) è un linguaggio di interrogazione (query) utilizzato per creare, modificare e gestire i dati in un database relazionale. Si tratta nello specifico di un linguaggio specifico di dominio (DSL) usato per comunicare con i sistemi di gestione di database relazionali (RDBMS).

Le operazioni fondamentali per la lettura e la manipolazione dei dati in SQL sono di quattro tipologie:

* Create: inserimento di nuovi dati;
* Read: lettura dei dati presenti nel database;
* Update: aggiornamento, modifica di dati già presenti nel database;
* Delete: cancellazione di dati presenti nel database.

Tali operazioni vengono effettuato tramite l’utilizzo di costrutti di programmazione denominati “query”.

* + 1. jQuery

jQuery è una libreria di funzioni JavaScript, cross-browser destinata ad applicazioni web, che si propone come obiettivo quello di semplificare la programmazione lato client delle pagine HTML. Il framework consente ai Designer di sviluppare pagine web con funzionalità interattive in maniera semplice. L’obiettivo è gestire al meglio gli aspetti grafici e strutturali come posizione di elementi, effetto di click su immagini, manipolazione del Document Object Model e quant’altro ancora, mantenendo la compatibilità tra browser diversi e standardizzando gli oggetti messi a disposizione dall’ interprete JavaScript del browser.

* + 1. MySQL

MySQL è il sistema di gestione di database relazionali open source (RDBMS) di livello enterprise più popolare al mondo che viene utilizzato su Facebook, Google, Adobe, Alcatel Lucent e Zappos e da molti siti Web/applicazioni online.

È sviluppato, distribuito e supportato da Oracle Corporation. È un database relazionale multipiattaforma, potente, flessibile ed estensibile basato sul linguaggio standardizzato SQL (Structured Query Language) utilizzato per creare e manipolare i database.

Il software del database MySQL è open source, utilizza la GPL (GNU General Public License).

MySQL viene utilizzato per un'ampia gamma di scopi, inclusi database web (l'uso più comune), data warehousing, e-commerce e applicazioni di registrazione.

* + 1. Bootstrap

Bootstrap è una libreria di componenti front end e un toolkit open source. Viene usato per creare pagine web reattive e mobile-first con HTML, CSS e JavaScript. Bootstrap usa tecnologie SASS, flexbox (sistema di griglie flessibili) e mixin (classi che contengono metodi utilizzabili da altre classi, senza che esista un rapporto genitore-figlio tra queste due classi).

* + 1. Apache Tomcat

Apache Tomcat (o semplicemente Tomcat) è un server web (nella forma di contenitore servlet) open source sviluppato dalla Apache Software Foundation. Implementa le specifiche JavaServer Pages (JSP) e servlet, fornendo quindi una piattaforma software per l'esecuzione di applicazioni web sviluppate in linguaggio Java. La sua distribuzione standard include anche le funzionalità di web server tradizionale, che corrispondono al prodotto Apache.

* + 1. Apache Commons E-mail

Commons E-mail mira a fornire un'API per l'invio di e-mail. È basato sull'API Java Mail, che ha il compito di semplificare la gestione dell’invio e della ricezione di e-mail.

Alcune delle classi di posta fornite sono le seguenti:

* SimpleE-mail: questa classe viene utilizzata per inviare e-mail basate su testo.
* MultiPartE-mail: questa classe viene utilizzata per inviare messaggi in più parti. Ciò consente un messaggio di testo con allegati inline o attached.
* HtmlE-mail - Questa classe viene utilizzata per inviare e-mail in formato HTML. Ha tutte le funzionalità di MultiPartE-mail che consente di aggiungere facilmente gli allegati. Supporta anche immagini incorporate.
* ImageHtmlE-mail: questa classe viene utilizzata per inviare e-mail in formato HTML con immagini incorporate. Ha tutte le funzionalità di HtmlE-mail ma trasforma tutti i riferimenti alle immagini in immagini inline.
* E-mailAttachment - Questa è una semplice classe contenitore per consentire una facile gestione degli allegati. È per l'uso con istanze di MultiPartE-mail e HtmlE-mail.
  + 1. JSP

Le JSP (Java Server Pages) sono molto simili dal punto di vista concettuale alle Servlet ma differiscono da queste nella struttura. Infatti, una pagina JSP oltre a poter contenere al suo interno del codice Java, ha la caratteristica di essere fortemente basata anche sul codice HTML puro (o anche mescolato con JavaScript). Una pagina JSP, nella piattaforma J2EE rappresenta un componente dello user interface logic layer (o presentation layer).

* + 1. JSTL

JavaServer Pages Standard Tag Library (JSTL) è una libreria inclusa come componente della piattaforma software di sviluppo per applicazioni web Java EE. È un'estensione di JSP ed incorpora un insieme di tag HTML definiti tramite file XML e programmati in linguaggio Java.

Il vantaggio della creazione di tag standard, analogamente a quanto avviene nelle funzioni di libreria, è quello di definire dei comportamenti univoci dello stesso tag in contesti diversi eliminando la necessità di definire degli scriptlet (applet fatte con un linguaggio di scripting) all'interno delle pagine.

* + 1. Project Lombok

Lombok è una libreria di tipo Annotation processor (APT) ovvero durante la compilazione del progetto esegue l’interpretazione delle cosiddette annotation (@NomeAnnotazione) dichiarate a livello di classe. In fase di compilazione Lombok esegue delle operazioni e genera in automatico il codice aggiuntivo riducendo di gran lunga la verbosità del linguaggio.

* 1. Object Design Trade-Offs

|  |  |
| --- | --- |
| **Trade Off** | **Descrizione** |
| Performance VS Dependability | Diamo precedenza alle Performance, perché riteniamo sia più importante fornire un uso performante del sito (DG\_1, DG\_2) rispetto a garantire l’uso del sito ad un numero elevato di utenti (DG\_9) |

|  |  |
| --- | --- |
| **Trade Off** | **Descrizione** |
| Maintenance VS End User Criteria | Diamo precedenza alla categoria Mainteneance, perché se durante lo sviluppo per motivi di tempo non dovessimo riuscire ad implementare tutto il necessario (DG\_4), preferiamo aver completato tutte le funzioni necessarie, a discapito di avere magari un’interfaccia utente meno raffinata (DG\_11) |

|  |  |
| --- | --- |
| **Trade Off** | **Descrizione** |
| Dependability VS End User Criteria | Diamo precedenza alla Dependability piuttosto che agli End user criteria perché, riteniamo sia più importante che il sito sia solido e possa ad esempio gestire un grande carico di utenti (DG\_9 Utenti contemporaneamente) piuttosto che avere un’interfaccia particolarmente raffinita ed intuitiva (DG\_11 Navigazione) |

* 1. Linee guida per la documentazione

In questa sezione vengono definite le linee guida che ogni team member rispetterà per la leggibilità del codice. Per la formattazione dei file XML, HTML, CSS e JS si userà il formatter di IntelliJ di default, mentre per i file Java si seguiranno le convenzioni della Sun di Java. Tra i cui aspetti fondamentali evidenziamo:

1.3.1 Nomi dei file

• Le classi devono avere nomi al singolare.

• I nomi dei file sorgente Java devono essere uguali al nome della classe in testa alla gerarchia.

• I nomi dei file di configurazione XML devono rispettare i requisiti del framework o tool che li

usa, come il pom.xml di Maven.

• I nomi dei file generati da compilatori, sistemi di build, o altri tool non devono essere modificati.

• I nomi delle classi di test di unità devono avere il suffisso “Test”, (es. classTest.java).

• I nomi delle classi di test di integrazione devono avere il suffisso “IT”, (es. classIT.java).

• In tutti gli altri casi, i nomi dei file devono rappresentare bene il contenuto e devono contenere

solo lettere minuscole, cifre ed underscore per connettere le diverse parole.

1.3.2 Dichiarazioni

Ogni dichiarazione di variabile locale può definire più di una variabile, mentre ogni dichiarazione di

variabile di istanza deve definire solo una variabile. Le variabili d’istanza devono essere private.

Ad una dichiarazione di variabile locale deve seguire l’inizializzazione nella stessa linea oppure nelle due righe seguenti.

1.3.3 Nomeclatura

Di seguito sono mostrati i vincoli di nomi delle componenti software del sistema:

• Package: solo lettere in lowerCamelCase

• Classi: solo lettere in UpperCamelCase

• Metodi: solo lettere in lowerCamelCase, devono contenere nel nome solo verbi e nomi degli attributi della classe

• Costanti: solo lettere e underscore in CONSTANT\_CASE

• Variabili: solo lettere in lowerCamelCase

• Parametri: solo lettere in lowerCamelCase. In particolare, dello stesso nome delle relative variabili

1.3.4 Convenzioni

• Le condizioni d’errore lanciano delle eccezioni e non valori di ritorno.

• Le collections ed i containers hanno un metodo iterator () che ritorna un Iterator.

• Gli iterator ritornati dai metodi iterator () sono robusti alla rimozione di elementi.

• Uso del for-each loop quando bisogna iterare per intero una collezione iterabile.

• Per gli if usare sempre le graffe, anche con singoli statement.

• Gli if sui booleani non devono prevedere un confronto diretto coi valori true e false (== true o == false).

1.3.5 Documentazione del codice

I commenti di documentazione saranno nel formato di Javadoc; verrà scritto un commento doc per gli elementi del codice.

1.3.6 Pagine HTML

Le pagine HTML, sia in forma statica che dinamica, devono essere conformi allo standard HTML 5. Inoltre, il codice HTML statico deve utilizzare l'indentazione, per facilitare la lettura, secondo le seguenti regole:

* Un'indentazione consiste in una tabulazione;
* Ogni tag deve avere un'indentazione maggiore del tag che lo contiene;
* Ogni tag di chiusura deve avere lo stesso livello di indentazione del corrispondente tag di apertura;
* I tag di commento devono seguire le stesse regole che si applicano ai tag normali.

1.3.7 Fogli di stile CSS

I fogli di stile (CSS) devono seguire le seguenti convenzioni:

* Tutti gli stili non inline devono essere collocati in fogli di stile separati.
* Ogni foglio di stile deve essere iniziato da un commento analogo a quello presente nei file Java.
* Ogni regola CSS deve essere formattata come segue:

1. I selettori della regola si trovano a livello 0 di indentazione, uno per riga;
2. L'ultimo selettore della regola è seguito da parentesi graffa aperta ({);
3. Le proprietà che costituiscono la regola sono listate una per riga e sono indentate rispetto ai selettori;
4. La regola è terminata da una parentesi graffa chiusa (}), collocata da sola su una riga;

1.3.8 Database SQL

I nomi delle tabelle devono seguire le seguenti regole:

* Devono essere costituiti da sole lettere maiuscole;
* Il nome deve essere un sostantivo singolare tratto dal dominio del problema ed esplicativo del contenuto.

I nomi dei campi devono seguire le seguenti regole:

* Devono essere costituiti da sole lettere minuscole;
* Se il nome è costituito da più parole, è previsto l’uso di underscore (\_);
  1. Definizioni, Acronimi e Abbreviazioni

|  |  |
| --- | --- |
| **Acronimo/Abbreviazione** | **Descrizione** |
| JS | JavaScript |
| JSP | Java Server Page |
| HTML | HyperText Markup Language |
| CSS | Cascading Style Sheet |
| SQL | Structured Query Language |
| COTS | Component Off The Shelf |
| DOM | Domain Object Model |

* 1. Riferimenti
* B. Bruegge, A.H. Dutoit – Object Oriented Software Engineering
* Sommerville, A. Wesley – Software Engineering
* C. Ghezzi, M. Jazayeri, D. Mandrioli – Ingegneria del Software
* Documentazione del Progetto
* Code Conventions for the JavaTM Programming Language
  1. JavaDoc

Per una maggiore comprensione e una più semplice lettura, è qui presente il link alla documentazione JavaDoc che contiene la descrizione di tutti i packages e di tutte le classi utilizzate.

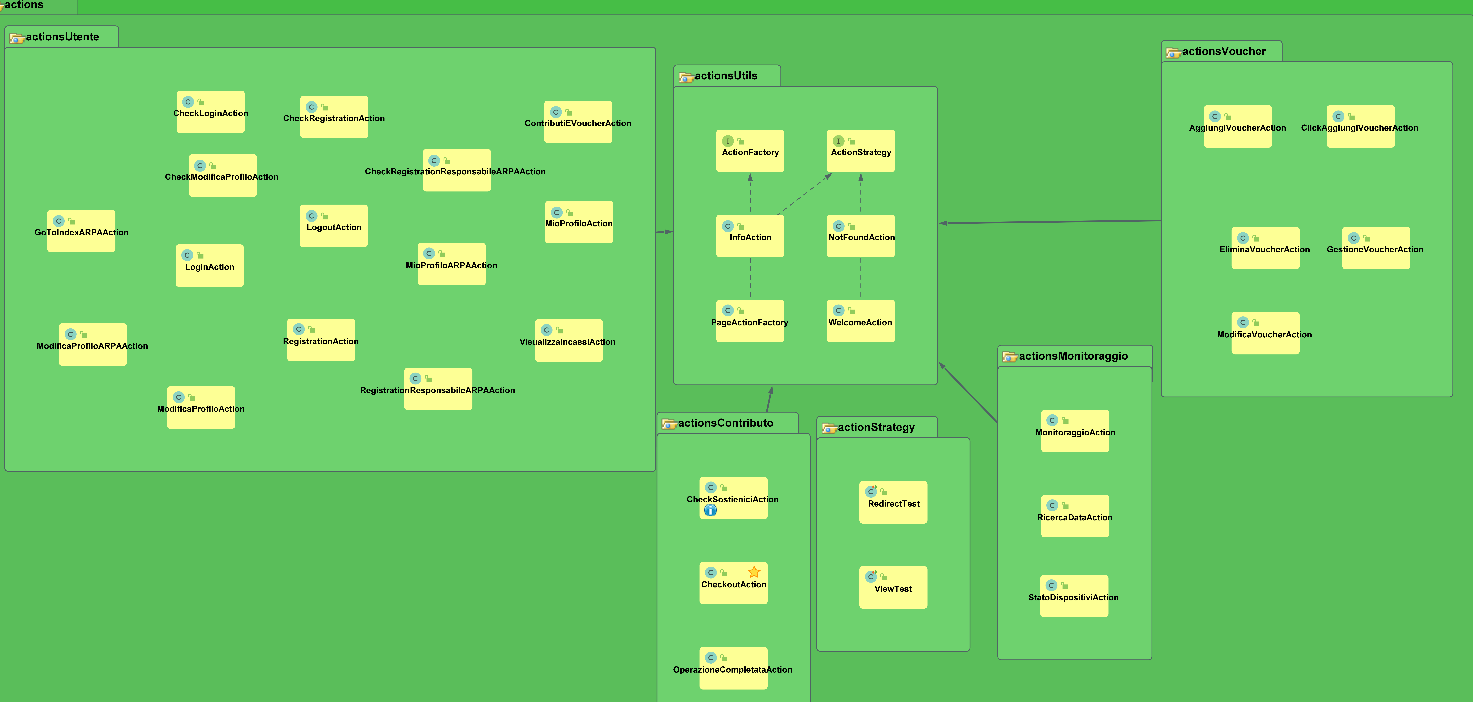
1. Packages

Il sistema CoralloSmart è così composto:



* 1. Actions

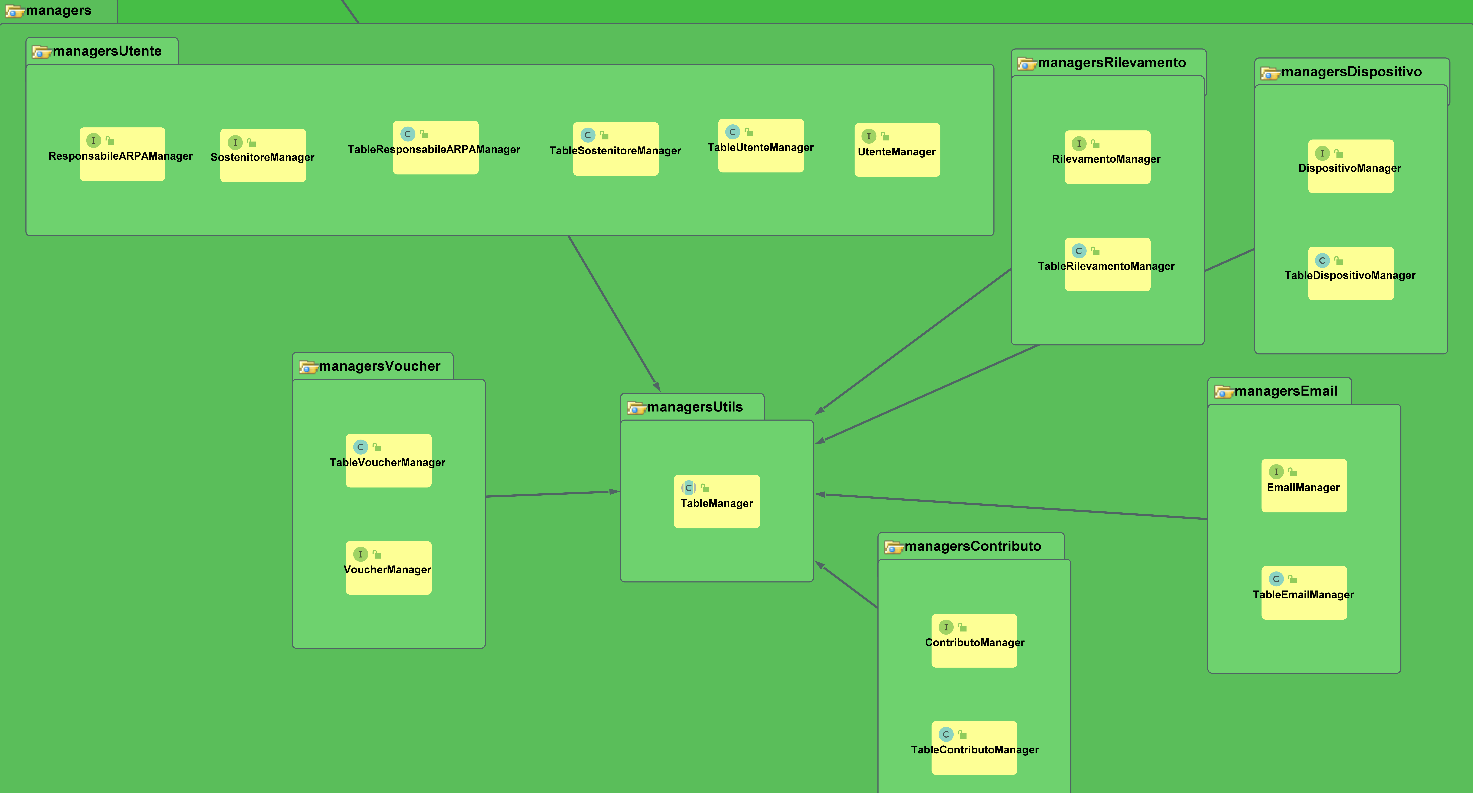
Il package **actions** raggruppa tutte le azioni che vengono eseguite prima di mostrare una pagina, rappresenta quelli che sono i control in un modello three tier.



Esso contiene:

* Package **ActionUtente**: contiene le action relative agli utenti.
* Package **ActionVoucher**: contiene le action relative ai voucher.
* Package **ActionContributo**: contiene le action relative al versamento di un contributo.
* Package **ActionMonitoraggio**: contiene le action relative al monitoraggio da parte di un responsabile ARPA.
* Package **ActionUtils**: contiene classi di supporto.
  1. Managers

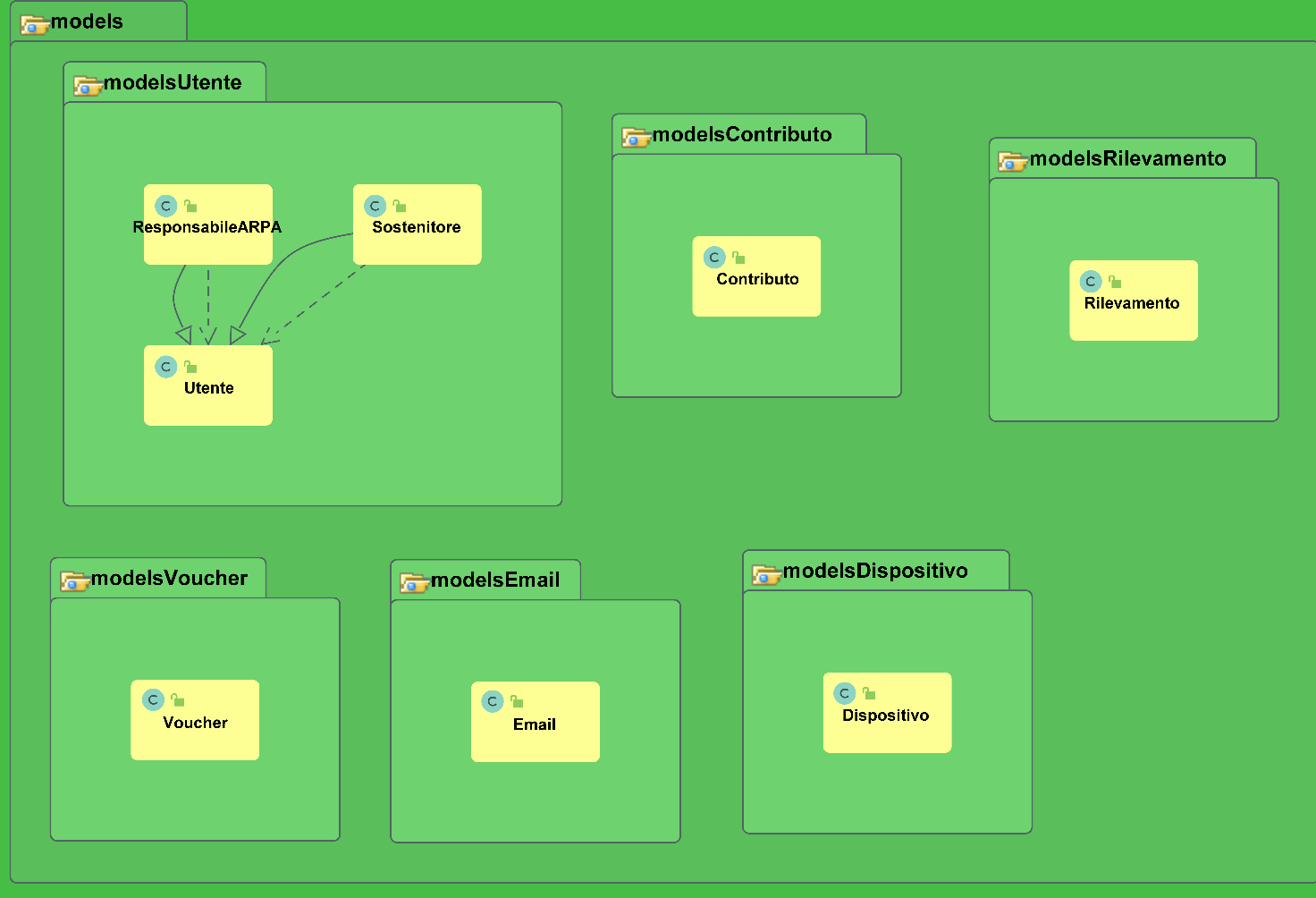
Il package **managers** contiene tutti i managers, i quali consento di effettuare operazioni sui model.



Esso Contiene:

* Package **managersUtils**: contiene classi di supporto.
* Package **managersUtente**: contiene le classi che consentono di interfacciarsi con il DB.
* Package **managersRilevamneto**: contiene le classi che consentono di interfacciarsi con il DB.
* Package **managersDispositivo**: contiene le classi che consentono di interfacciarsi con il DB.
* Package **managersContributo**: contiene le classi che consentono di interfacciarsi con il DB.
* Package **managersVoucher**: contiene le classi che consentono di interfacciarsi con il DB.
* Package **managersEmail**: contiene le classi che consentono di interfacciarsi con il DB.
  1. Models

Il package **models** contiene tutti i model.



Esso contiene:

* Package **modelsContributo**: contiene il model Contributo.
* Package **modelsRilevamento**: contiene il model Rilevamneto .
* Package **modelsVoucher**: contiene il model Voucher.
* Package **modelsEmail**: contiene il model E-mail.
* Package **modelsUtente**: contiene i models Utente, ResponsabileARPA e Sostenitore.
  1. Router

Il package **router** contiene la servlet che consente di mostrare una pagina.

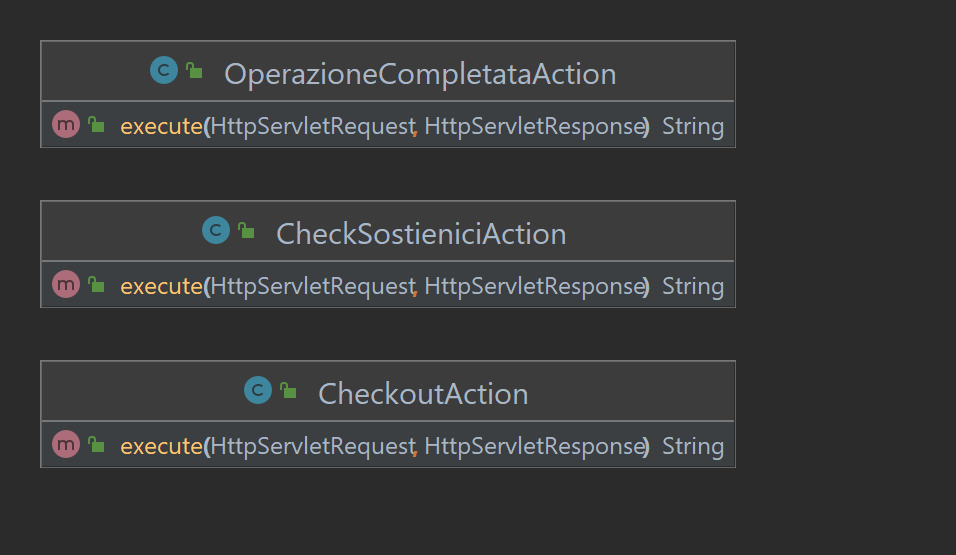


Esso contiene:

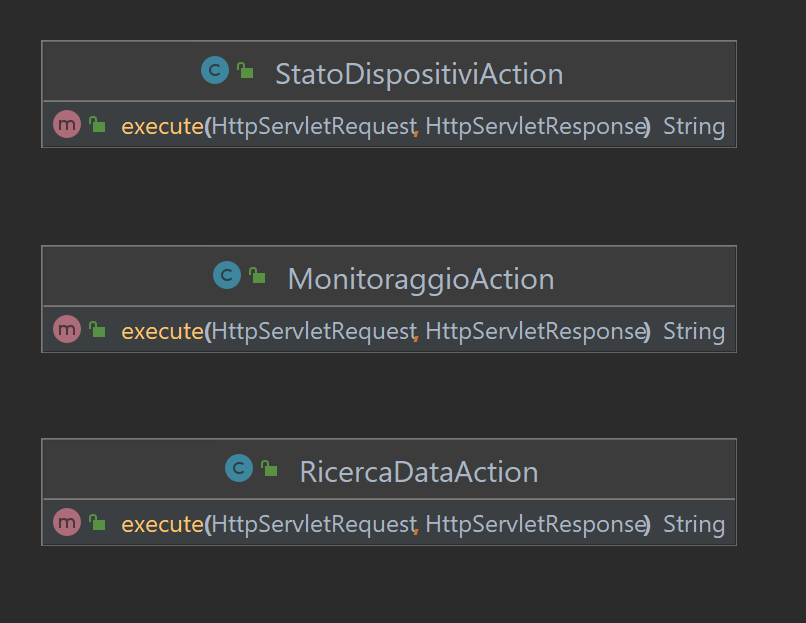
* **HttpMethod**: classe che contiene gli enum GET, POST.
* **Router**: classe che contiene tutte le informazioni su cosa mostrare, con che metodo e quando.
* **FrontController**: servlet che mostra la pagina richiesta grazie alle informazioni del router.
  1. Views

Il package **views** contiene tutte le views da mostrare.

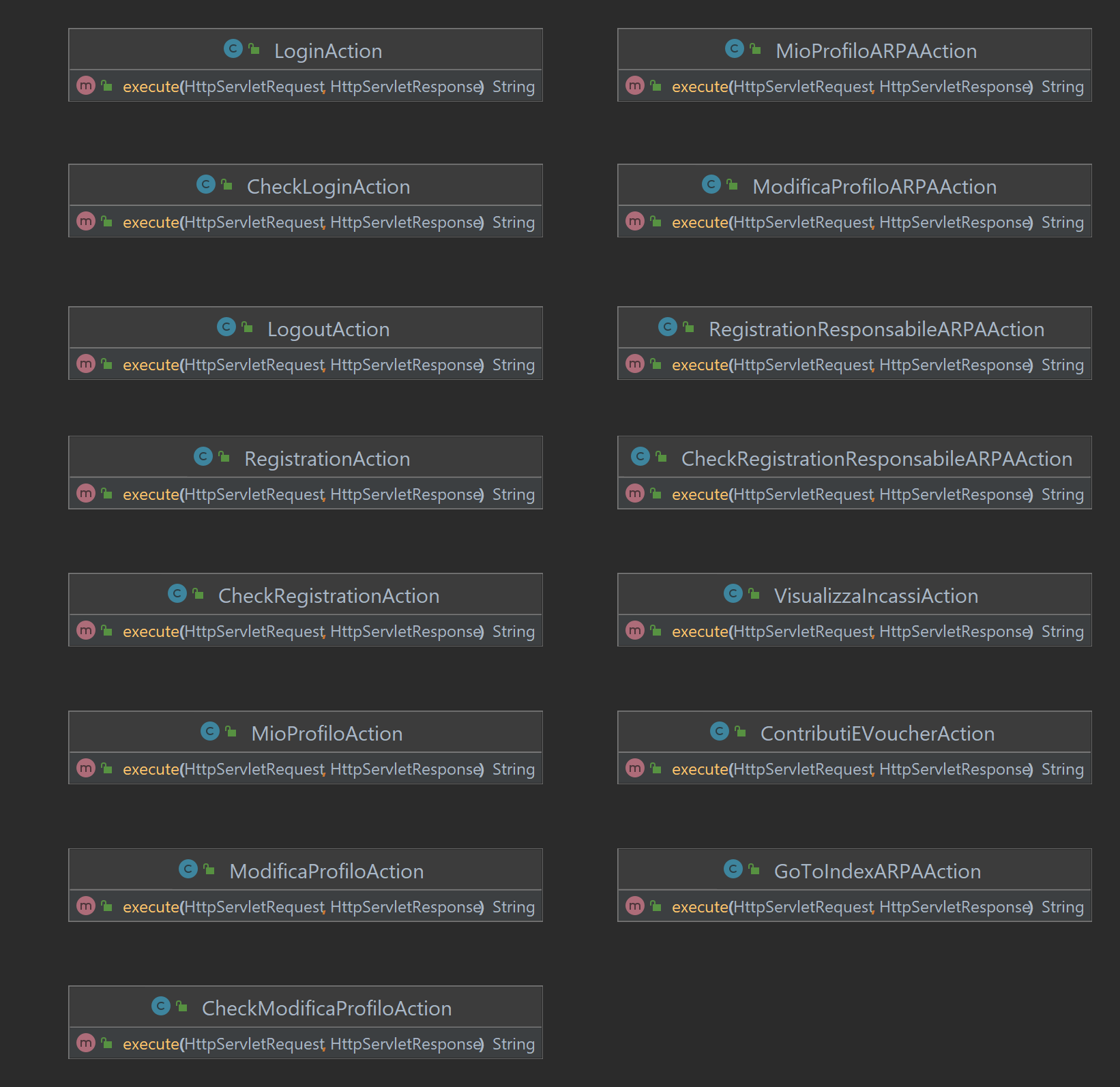
1. Interfaccia delle classi
   1. Contributo



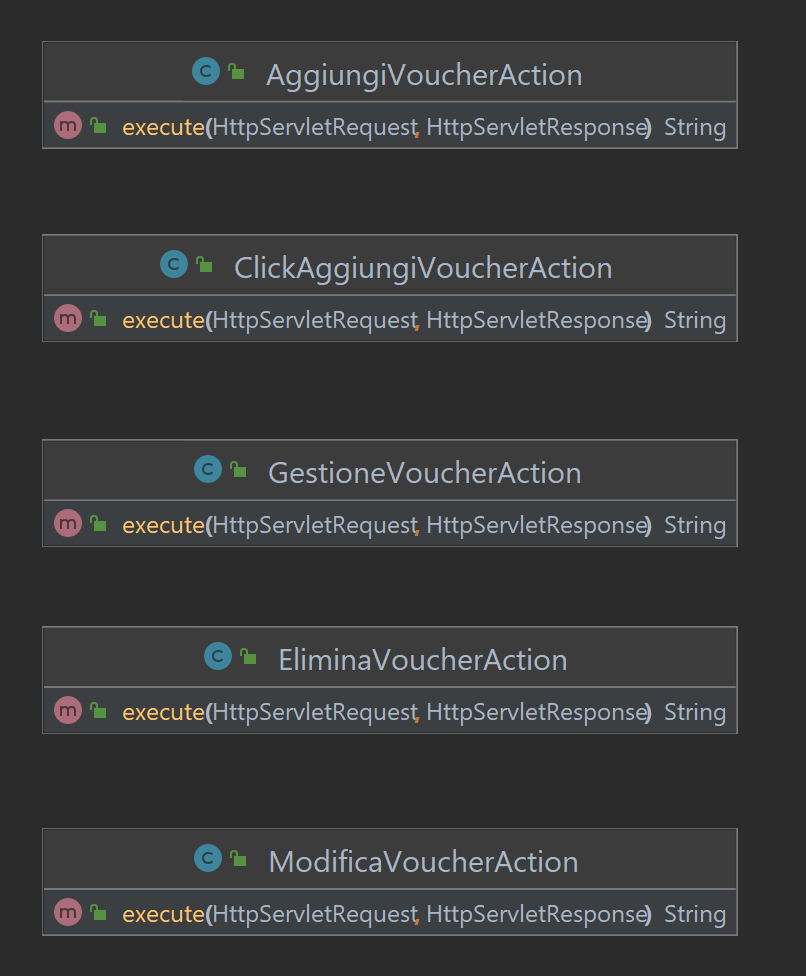
* 1. Monitoraggio



* 1. Utente



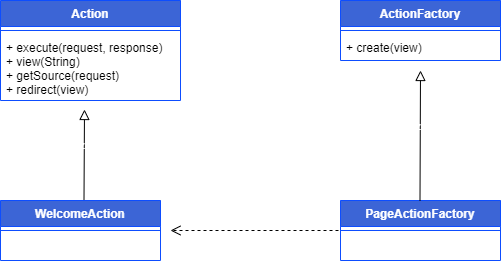
* 1. Voucher



Implementano tutti la stessa interfaccia ovvero ActionStrategy

1. Design Pattern con Class Diagram
   1. Factory Pattern

Si tratta di un pattern creazionale basato su classi e viene utilizzato per creare degli oggetti senza conoscerne i dettagli ma delegando un Creator (nel nostro caso l’ActionFactory) che, in base alle informazioni ricevute, saprà quale oggetto restituire.



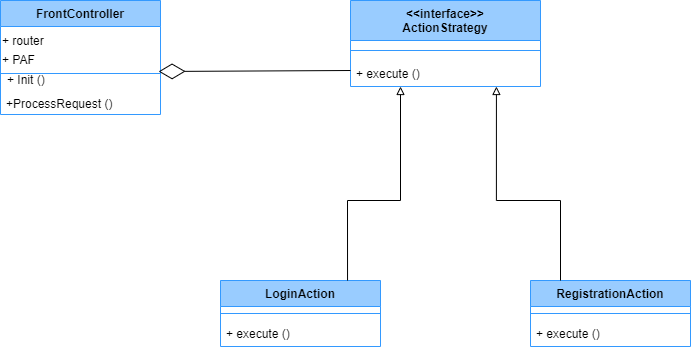
Questo pattern è composto dai seguenti partecipanti:

1. **ActionFactory**: dichiara la Factory che avrà il compito di ritornare l’oggetto appropriato.
2. **PageActionFactory**: effettua l’overwrite del metodo della Factory al fine di ritornare l’implementazione dell’oggetto.
3. **Action**: definisce l’interfaccia dell’oggetto che deve essere creato dalla Factory.
4. **WelcomeAction:** implementa l’oggetto in base ai metodi definiti dall’interfaccia Action.
   1. Strategy Pattern

É un modello di progettazione comportamentale che consente di definire una famiglia di algoritmi, isolando un algoritmo al di fuori di una classe, per far sì che quest’ultima possa variare dinamicamente il suo comportamento, rendendo così gli algoritmi intercambiabili a runtime.

Grazie allo Strategy Pattern è possibile utilizzare una qualsiasi implementazione di un algoritmo (che viene genericamente chiamata Strategy o Strategia), scegliendo fra quelle disponibili, che si rende più opportuna in un determinato contesto, in quanto tutte le implementazioni facenti parte della stessa famiglia hanno interfaccia comune.

Il tutto si traduce nel seguente diagramma delle classi:



Più nel dettaglio:

* *ActionStrategy*: dichiara l’interfaccia della nostra classe di algoritmi. Interfaccia che viene utilizzata dalle Action per invocare un algoritmo concreto
* *LoginAction* e *CheckLoginAction* sono i nostri algoritmi concreti, ovvero implementano uno specifico algoritmo che espone l’interfaccia Strategy
* *FrontController*, classe di contesto che invoca la ActionStrategy (sotto richiesta dei suoi client).

1. Glossario

|  |  |
| --- | --- |
| Termine | Descrizione |
| Design Pattern | È un concetto che può essere definito "una soluzione progettuale generale ad un problema ricorrente". Si tratta di una descrizione o modello logico da applicare per la risoluzione di un problema che può presentarsi in diverse situazioni durante le fasi di progettazione e sviluppo del software, ancor prima della definizione dell'algoritmo risolutivo |
| COTS | L’espressione “componente COTS” si riferisce a componenti hardware e software disponibili sul mercato per l'acquisto da parte di aziende di sviluppo interessate a utilizzarli nei loro progetti. |