**Università degli Studi di Salerno  
Corso di Ingegneria del Software**

****

**urNotes  
ODD**

**Object Design Document**

****

Data: 18/12/2018

**Coordinatore del progetto:**

|  |  |
| --- | --- |
| Nome | Matricola |
| Francesco Parisi | 0512103582 |

**Partecipanti:**

|  |  |
| --- | --- |
| Nome | Matricola |
| Marco La Cortiglia | 0512103630 |
| Matteo Maiorano | 0512104578 |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Descrizione | Autore |
| 18/12/2018 | 0.1 | Aggiunta Packages,Sottosistemi e Revisione Organizzazione Package | Francesco Parisi |
| 29/12/2018 | 0.2 | Completamento e revisione documento | Tutti |
|  |  |  |  |

Indice

1. Introduzione ……………....................................................................................................................................................................4

1.1 Object design trade-offs……………........................................................................................................................................4

1.2 Linee guida per la documentazione dell'Interfaccia…………….....................................................................................5

1.3 Definizioni, Acronimi e Abbreviazioni……………................................................................................................................8

1.4 Riferimenti……………..................................................................................................................................................................8

2. Packages…………….............................................................................................................................................................................9

2.1 Package Bean……………...................................................................................................................................................... 10

2.2 Package Connection...................................................................................................................................................... 11

**2.3** Package Model..................................................................................................................................................................12

**2.4** Package View.....................................................................................................................................................................12

**2.4.1 Package Pubblico** ....................................................................................................................................................13

**2.4.2 Package Gestore**.......................................................................................................................................................14

**2.4.3 Package Studente**.....................................................................................................................................................15

**2.5** Package Account..............................................................................................................................................................24

**2.6** Package Documenti........................................................................................................................................................25

**2.7** Package Ordini................................................................................................................................................................. 28

3. Interfacce delle classi………….…………….…………………………………………………………………………………….…...………………………….…….30

Glossario………….…………….……………………………………………………….……………………………………………………..……………….……………….… 45

1. ***Introduzione***
   1. **Object design trade-offs**

**Comprensibilità vs costi**

Si preferisce avere un aumento dei costi per la documentazione, al fine di rendere il codice quanto più comprensibile possibile per i vari utenti: vale a dire per chi non è coinvolto nel progetto, per chi non ha lavorato in una parte in particolare, ma anche per chi lo ha fatto, in modo da agevolarne il mantenimento o un eventuale modifica e in generale per averne una comprensione migliore.

**Interfaccia vs Usabilità**

L’interfaccia grafica fa utilizzo di form e bottoni accompagnati da etichette con indicazioni brevi e concise, in modo da agevolare l’utilizzo del sistema all’utente finale, rendendo il tutto semplice, chiaro e intuitivo.

**Memoria vs efficienza**

La priorità viene data all’efficienza, poiché il sistema ha lo scopo di fornire all’utente finale una visione chiara e corretta della fase iniziale della richiesta di Tirocinio, quindi si deve avere la certezza che le operazioni effettuate, come la ricerca di offerte formative in base a determinati parametri, siano corrette. La memorizzazione dei dati ovviamente non viene a mancare, in quanto bisogna mantenere la tracciabilità di tutte le richieste ed è indispensabile il salvataggio dei profili utenti.

**Sicurezza vs efficienza**

La sicurezza è un aspetto molto importante del nostro sistema. L’autenticazione di un utente avviene, ovviamente dopo opportuna registrazione, attraverso username e password. Ogni utente può accedere solo a specifiche funzionalità del sistema, a seconda del tipo di account a cui fa capo.

* 1. **Linee guida per la documentazione dell'Interfaccia**

Ogni metodo ed ogni file devono essere preceduti da un commento, ovvero da una documentazione che riporti l’obiettivo che si vuole e deve raggiungere. Vi saranno inoltre, commenti e giustificazioni in merito a particolari decisioni o calcoli.

La prima lettera può essere sia maiuscola, come il nome delle classi in Java, che minuscola, come le proprietà di un oggetto in Java.

**Organizzazione dei file**

Ogni file deve:

o Essere sviluppato e correlato ad un’unica funzionalità che persegue. Ogni pagina deve essere implementata in file separato (Accedi, SalvaUtente, etc). All’interno dello stesso file potranno, tuttavia, essere inserite classi private associate alla classe pubblica.

o Essere diviso in più file se raggiunge una lunghezza tale da divenire difficile da leggere e comprendere.

Per quanto riguarda i nomi dei file, operazioni e variabili, si è scelto di utilizzare nomi evocativi, in lingua italiana o inglese, ma che in ogni caso risultano di facile comprensione e di immediata intuizione. I termini GET e SET, ad esempio, non sono stati tradotti in quanto la traduzione in italiano non regge il confronto. Di conseguenza viene inserito il rispettivo significato nella terminologia.

I file delle librerie usate e altre risorse utilizzate per lo sviluppo del processo verranno organizzati nelle varie cartelle.

In generale, i file hanno la seguente struttura:

**Commento iniziale**: un commento che specifichi il nome della classe.

**Package e Import**: le prime linee di codice specificano in quale package il file è contenuto e le righe successive possono contenere direttive di import per l’utilizzo di classi non contenute dal package corrente.

**Descrizione della classe**: la dichiarazione della classe è preceduta da un commento più o meno breve che descrive approfonditamente le funzionalità delle classi, le metodologie utilizzate, l’autore della classe e la versione. Il commento fa utilizzo delle annotazioni javadoc.

**Dichiarazione della classe (o dell’interfaccia**): gli elementi della classe devono essere dichiarati nel modo seguente:

1. La prima istruzione deve essere la dichiarazione della classe (o interfaccia) pubblica.

2. Le variabili sono dichiarate all’inizio della classe, seguendo il seguente ordine: prima le costanti (final, static), poi le variabili di classe(static), in seguito le variabili di istanza (public, protected, private).

3. I costruttori vengono dichiarati prima dei metodi.

4. Infine, vengono dichiarati i metodi, raggruppati in base alle funzionalità e non in base alla visibilità.

**Istruzioni**

Nel caso di istruzioni semplici, ogni linea deve contenere al massimo una sola istruzione. Mentre nel caso di istruzioni composte vanno rispettate le seguenti regole:

* Le istruzioni racchiuse all’interno di un blocco (esempio: for), devono essere indentate di un’unita all’interno dell’istruzione composta.
* La parentesi di apertura del blocco deve trovarsi alla fine della riga dell’istruzione composta.
* La parentesi di chiusura del blocco deve trovarsi allo stesso livello di indentazione dell’istruzione composta
* Le istruzioni composte formate da un’unica istruzione devono essere racchiuse da parentesi.

**Spazi bianchi**

Uno spazio bianco dovrebbe essere usato nelle seguenti condizioni:

* Dopo le virgole nelle liste di argomenti.
* Tutti gli operatori binari eccetto “. “(operatore punto) dovrebbero essere separati dai loro operandi tramite spazi. Gli spazi bianchi non dovrebbero mai separare gli operatori unari come l’operatore meno, l’incremento e il decremento.

**Convenzioni di nomi**

**Classi**

I nomi di classe devono essere sostantivi, con lettere minuscole e, sia la prima lettera del nome della classe sia la prima lettera di ogni parola interna, deve essere maiuscola. I nomi delle classi dovrebbero essere semplici, descrittivi e che rispettino il dominio applicativo. Usare parole intere evitando acronimi e abbreviazioni. Non dovrebbero essere usati underscore per legare nomi.

**Metodi**

I nomi dei metodi iniziano con una lettera minuscola (non sono consentiti caratteri speciali) e seguono la notazione a cammello. Dovranno essere semplici, descrittivi e che rispettino il dominio applicativo

**Variabili**

La scelta di un nome deve essere mnemonica e di facile intuizione, rispettando il dominio applicativo. Dovrebbero essere evitati nomi di variabili con un solo carattere.

**Pagine Html**

Le pagine HTML, statiche e dinamiche, devono essere aderenti allo standard HTML5.

o Nella prima riga, va dichiarato il tipo del documento

o Per i nomi degli elementi, HTML5 consente l’uso del minuscolo e del maiuscolo. Non è buona norma mischiare il minuscolo e il maiuscolo. Gli sviluppatori sono soliti usare il minuscolo, che risulta più “pulito”, è più semplice da scrivere.

o In HTML5 non bisogna chiudere obbligatoriamente tutti gli elementi. Per consentire una lettura più chiara e scorrevole, si preferisce chiudere tutti gli elementi.

o HTML5 consente l’uso del minuscolo e del maiuscolo per gli attributi, è consigliato utilizzare il minuscolo.

o In HTML5 non sono necessari i tag principali: <html>, <body>, <head>. Tuttavia si preferisce utilizzarli, per evitare problemi di validazione del documento.

o Usare una sintassi semplice per includere fogli di stile esterni.

**Fogli di stile CSS**

Per una più chiara lettura, si consiglia di scrivere le regole CSS su più righe, In generale:

o Aprire la parentesi graffa sulla stessa riga del selettore.

o Usare un solo spazio prima della parentesi graffa.

o Usare due punti più uno spazio tra una proprietà e il suo valore.

o Usare uno spazio dopo una virgola o dopo un punto e virgola (regola che vale in qualsiasi caso).

o Chiudere le parentesi graffe su una nuova riga, senza spazi.

* 1. **Definizioni, acronimi e abbreviazioni**

|  |  |
| --- | --- |
| ***urNotes*** | ***Nome del progetto*** |
| ***RAD*** | *Requirement Analysis Document* |
| ***SDD*** | System Design Document. |
| ***ODD*** | Object Design Document |
| **DB** | *Database* |
| **MySql** | Database Open Source basato sul linguaggio SQL, composto da un client a riga di comando e un server. |
| **DBMS** | Database Management System |
| ***SQL*** | Linguaggio standardizzato per database basati sul modello relazionale |
| ***CSS*** | Linguaggio usato per definire la formattazione di pagine web. |
| ***HTML5*** | Linguaggio di markup per la strutturazione delle pagine web. |

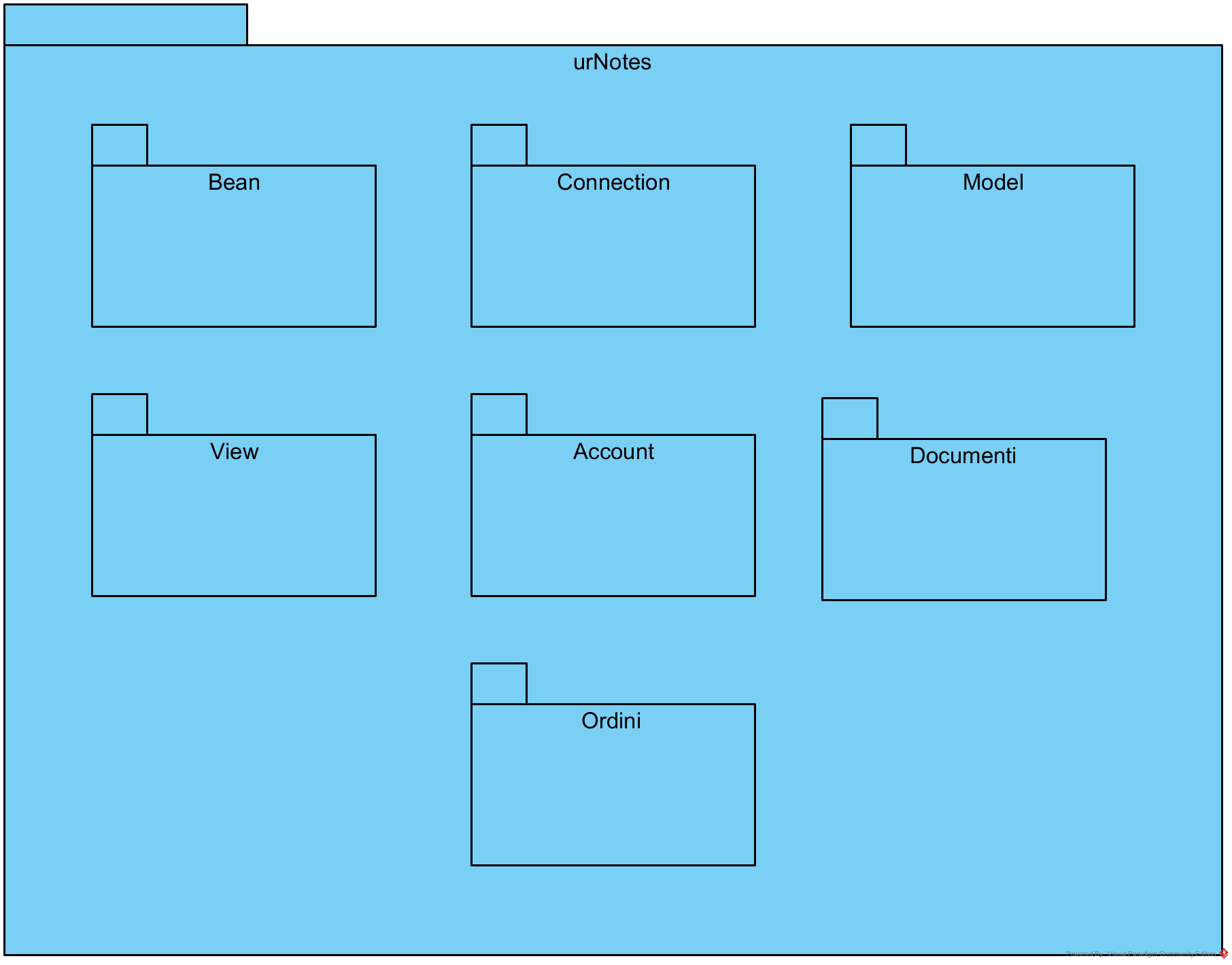
* 1. **Riferimenti**
* Slide del corso, presenti sulla piattaforma e-learning,
* B. Bruegge, A.H. Dutoit, Object Oriented Software Engineering – Using UML, Patterns and Java, Prentice Hall
* [*https://www.studocu.com/it*](https://www.studocu.com/it) *Piattaforma dedicata alla condivisione di vari documenti (Appunti, Libri, Riassunti ecc.)*
* Documenti RAD\_urNotes
* Documenti SDD\_urNotes

1. ***Packages***

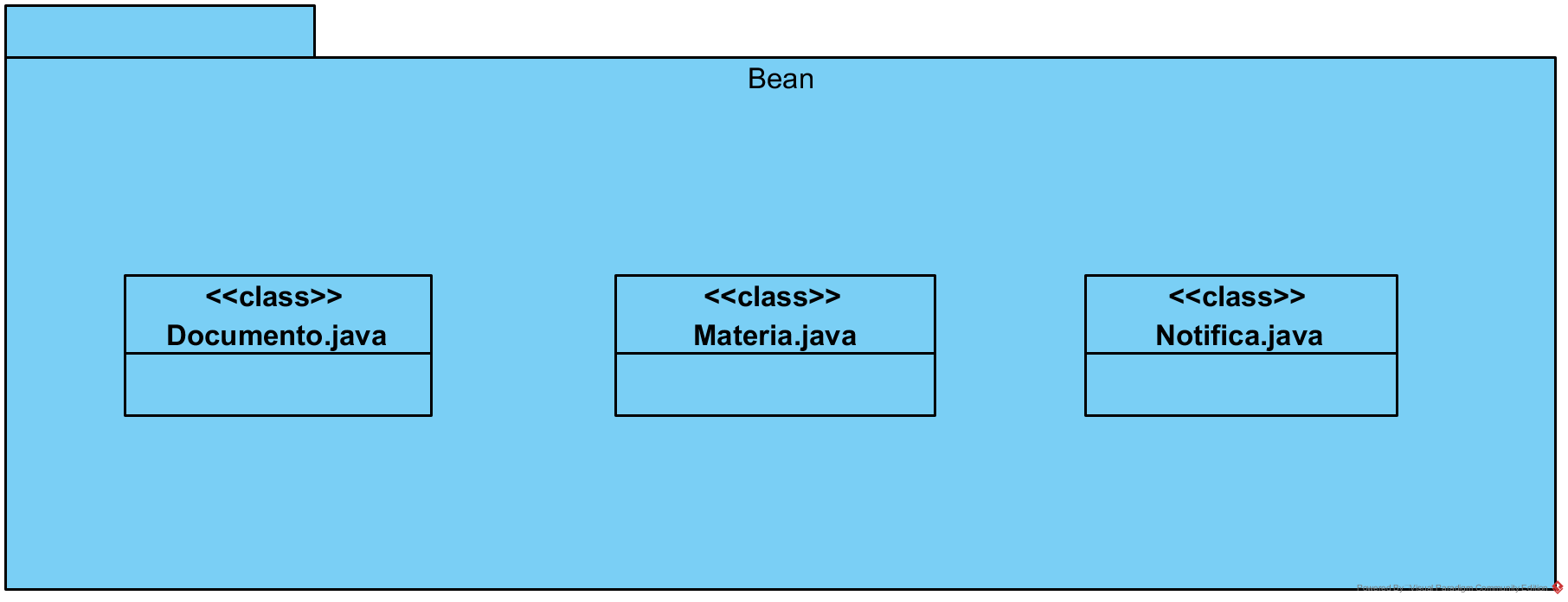
La gestione del nostro sistema è suddivisa in tre livelli (three-tier):

* Presentation Layer
* Application Logic Layer
* Storage Layer

Il package urNotes contiene sottopackage che a loro volta inglobano classi il cui scopo è quello di gestire le richieste dell’utente.

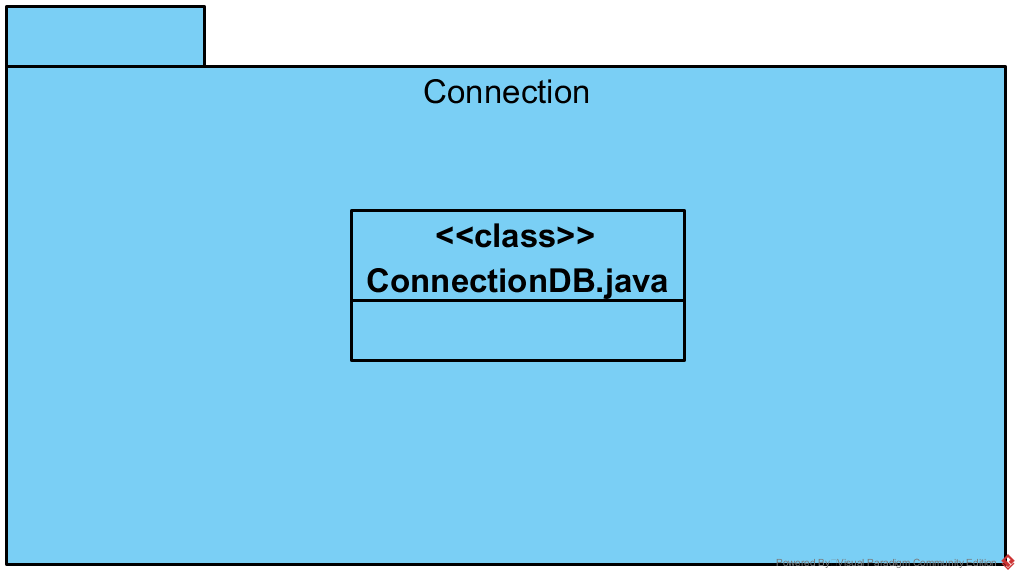


* 1. **Package Bean**



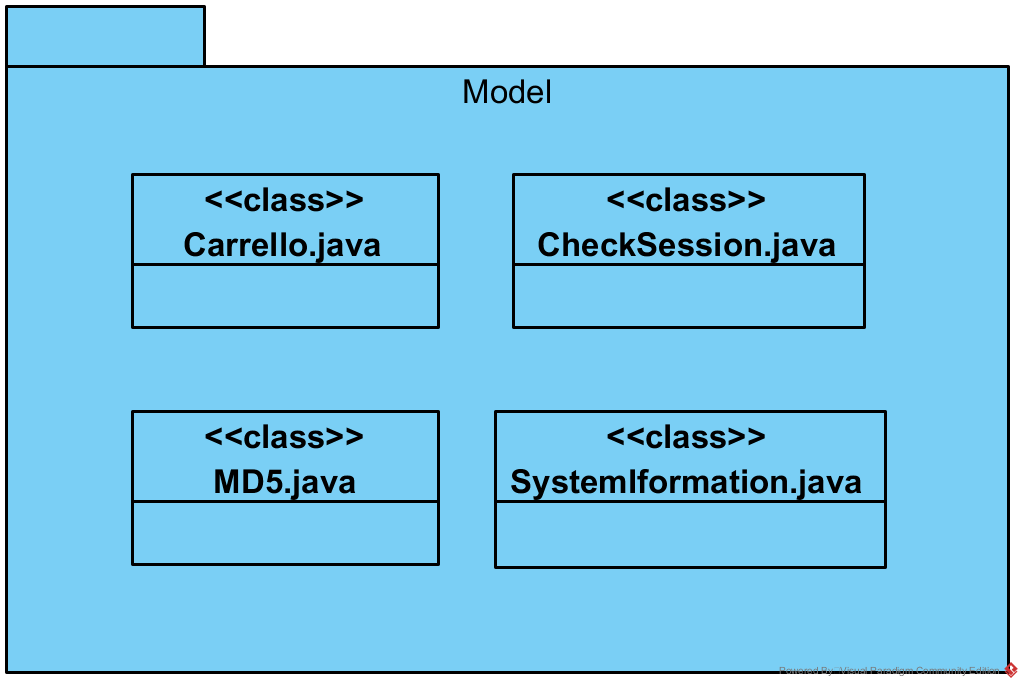
|  |  |
| --- | --- |
| Classe | Descrizione |
| Documento.java | Descrive un appunto/dispensa. |
| Materia.java | Descrive una materia. |
| Notifica.java | Descrive una notifica di pubblicazione di un documento. |

* 1. **Package Connection**



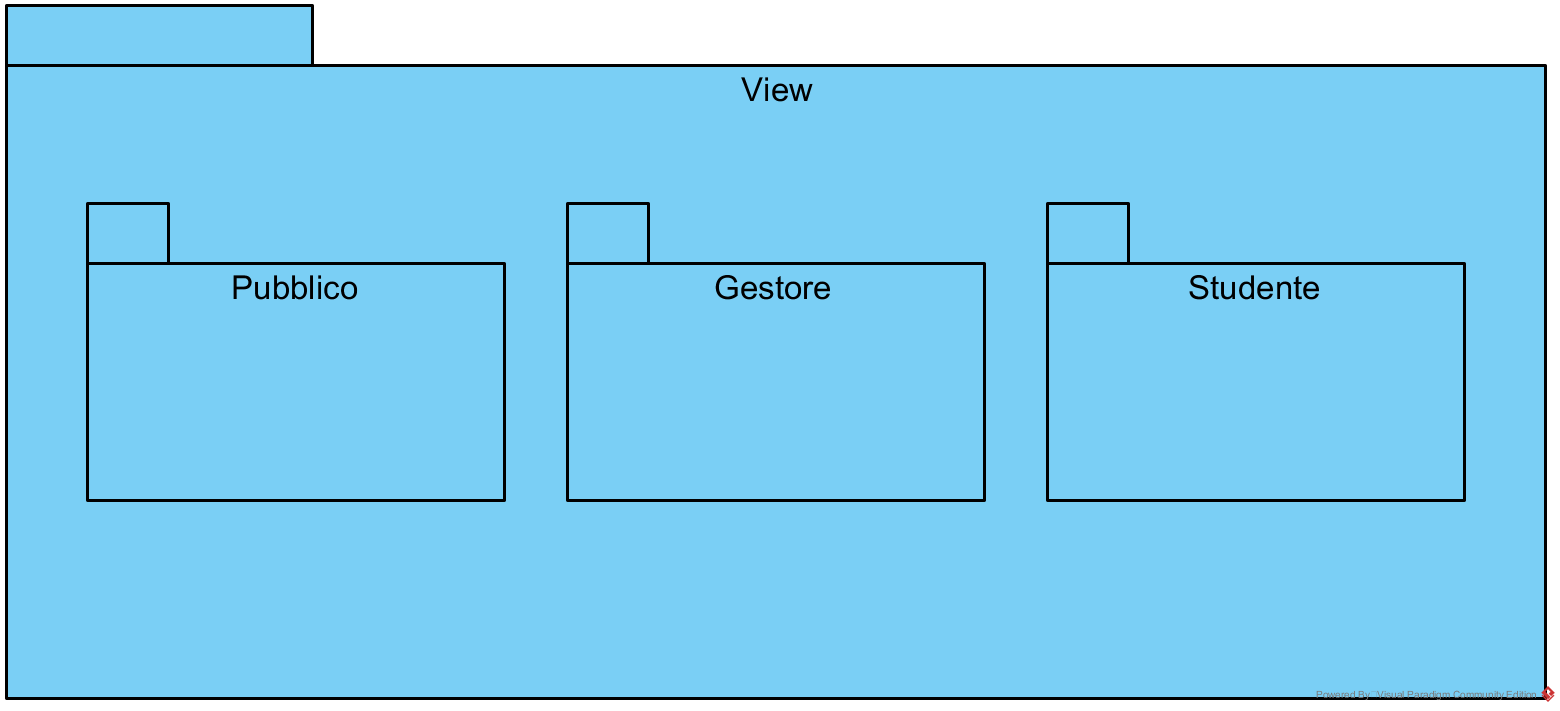
|  |  |
| --- | --- |
| Classe | Descrizione |
| ConnessioneDB.java | La classe che si occupa di gestire la connesione con il DataBase. |

* 1. **Package Model**

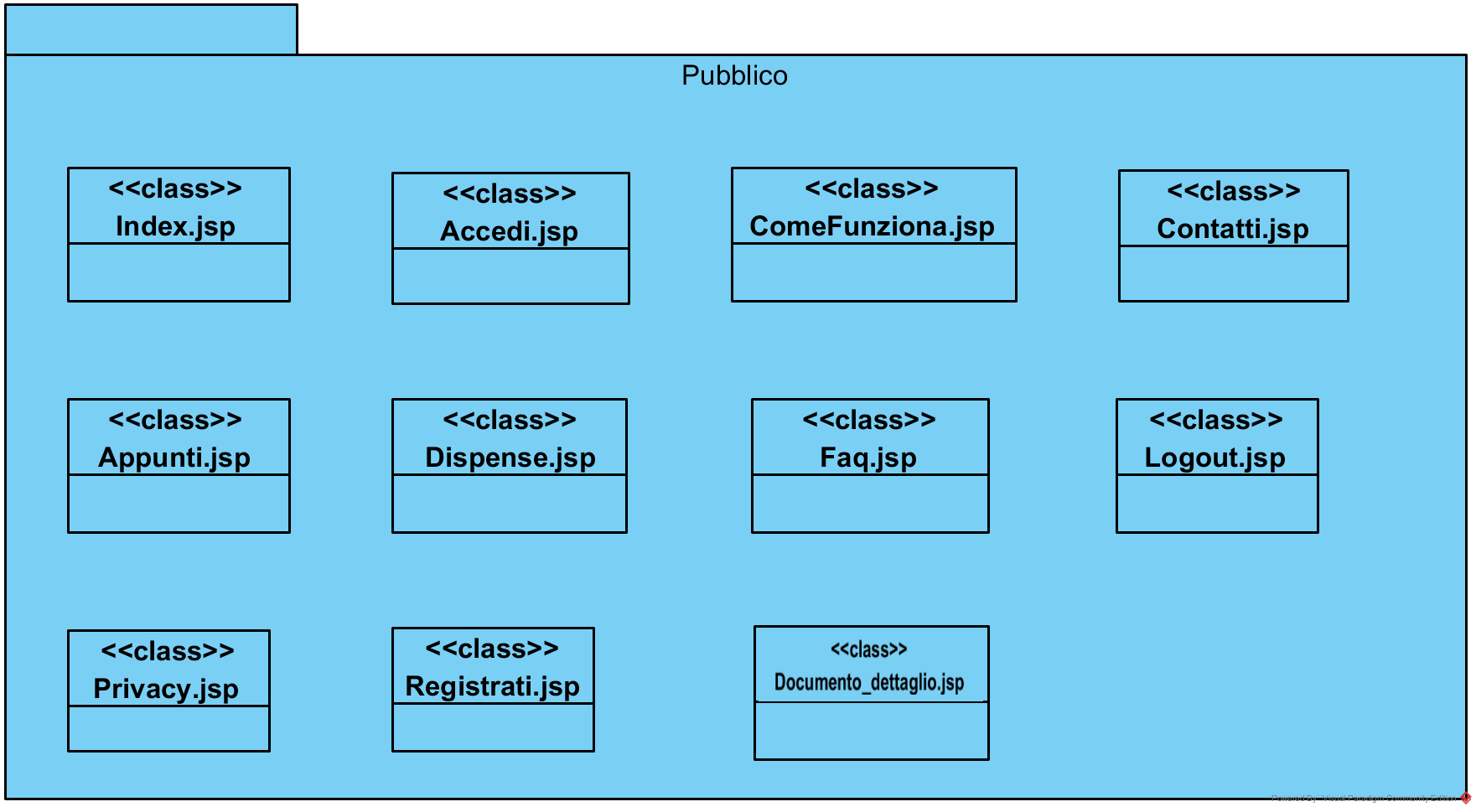


|  |  |
| --- | --- |
| Classe | Descrizione |
| Carrello.java | Descrive il carrello dello studente. |
| CheckSession.java | Descrive la sessione di un utente . |
| MD5.java | Descrive la crittografia scelta per proteggere la password di un utente . |
| SystemInformation.java | Descrive le informazioni di sistema. |

* 1. **Package View**

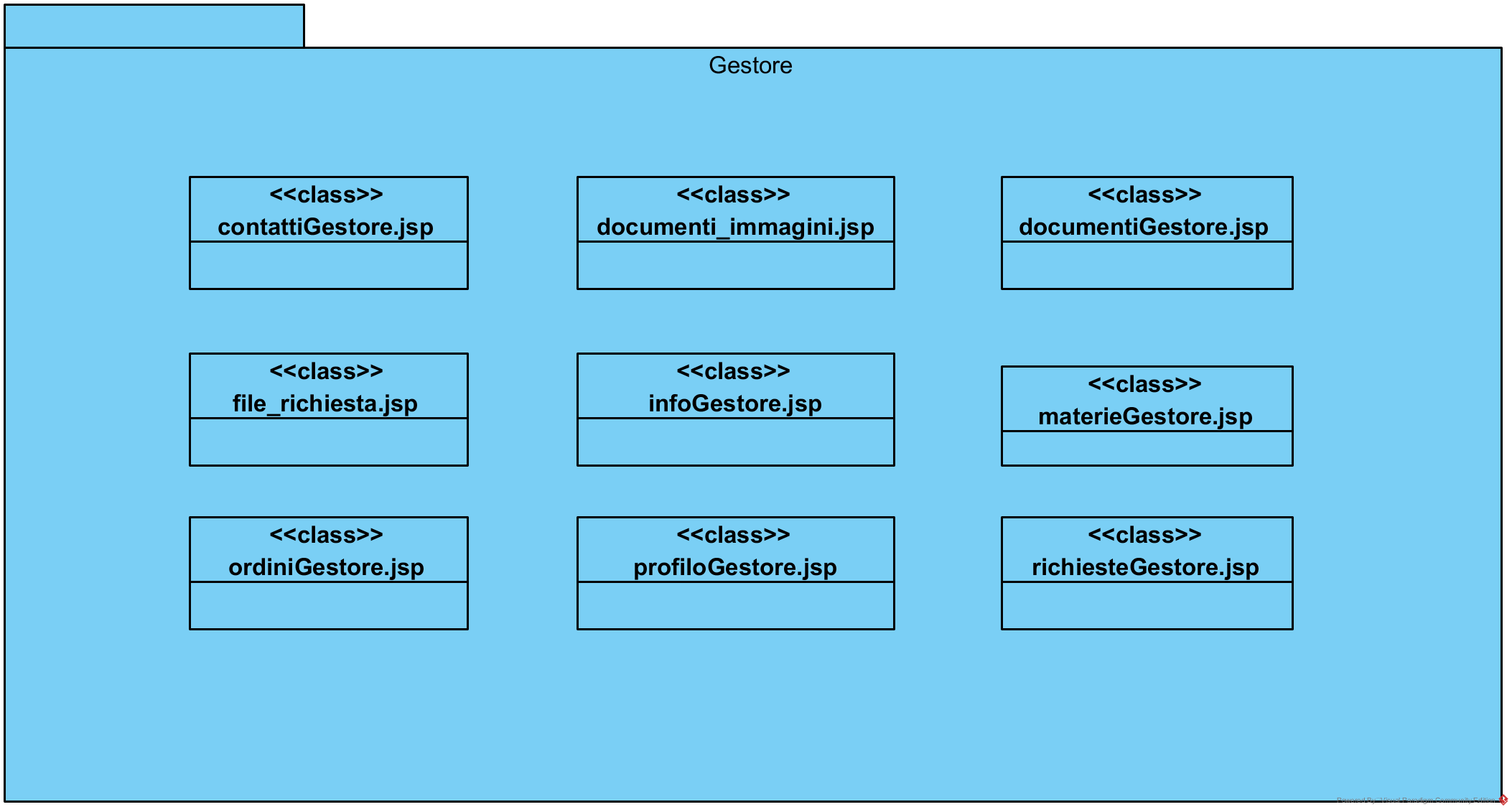


* + 1. **Package Pubblico**



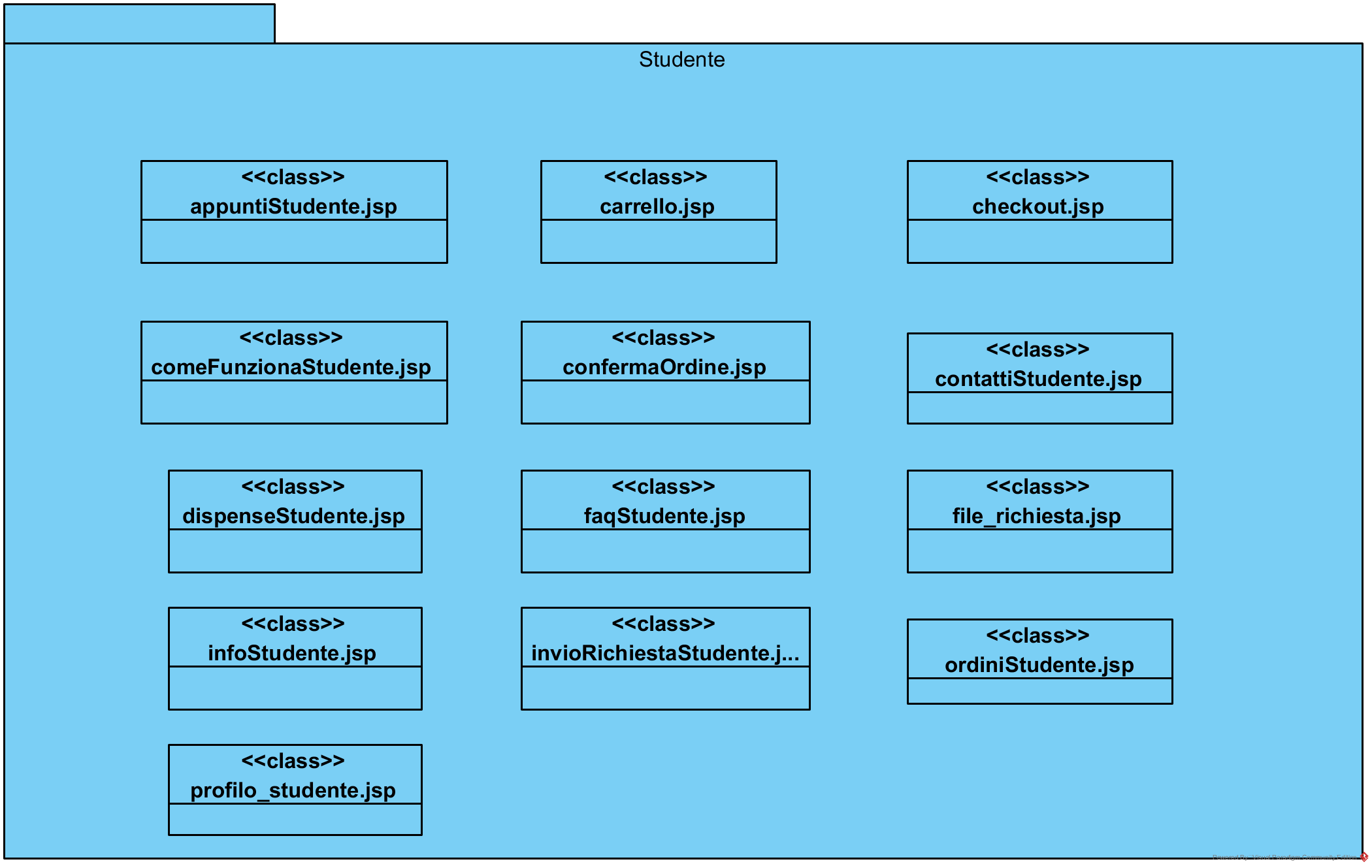
|  |  |
| --- | --- |
| Classe | Descrizione |
| Index.jsp | La view che mostra l’HomePage agli utenti non loggati al sistema. |
| Accedi.jsp | La view che mostra la form per accedere al sistema. |
| ComeFunziona.jsp | La view che spiega l’utilizzo del sistema. |
| Contatti.jsp | La view che mostra i contatti principali del sistema. |
| Appunti.jsp | La view che mostra il catalogo di appunti presenti nel sistema. |
| Dispense.jsp | La view che mostra il catalogo di dispense presenti nel sistema. |
| Faq.jsp | La view che mostra Domande&Risposte relative all’utilizzo del sistema. |
| Logout.jsp | La view che mostra l’uscita dell’utente dal sistema. |
| Privacy.jsp | La view che mostra le informazioni su come vengono gestiti i dati personali |
| Registrati.jsp | La view che mostra la form per la registrazione dello studente. |
| Documento\_dettaglio.jsp | La view che mostra la pagina relativa alle informazioni del documento scelto. |

* + 1. **Package Gestore**



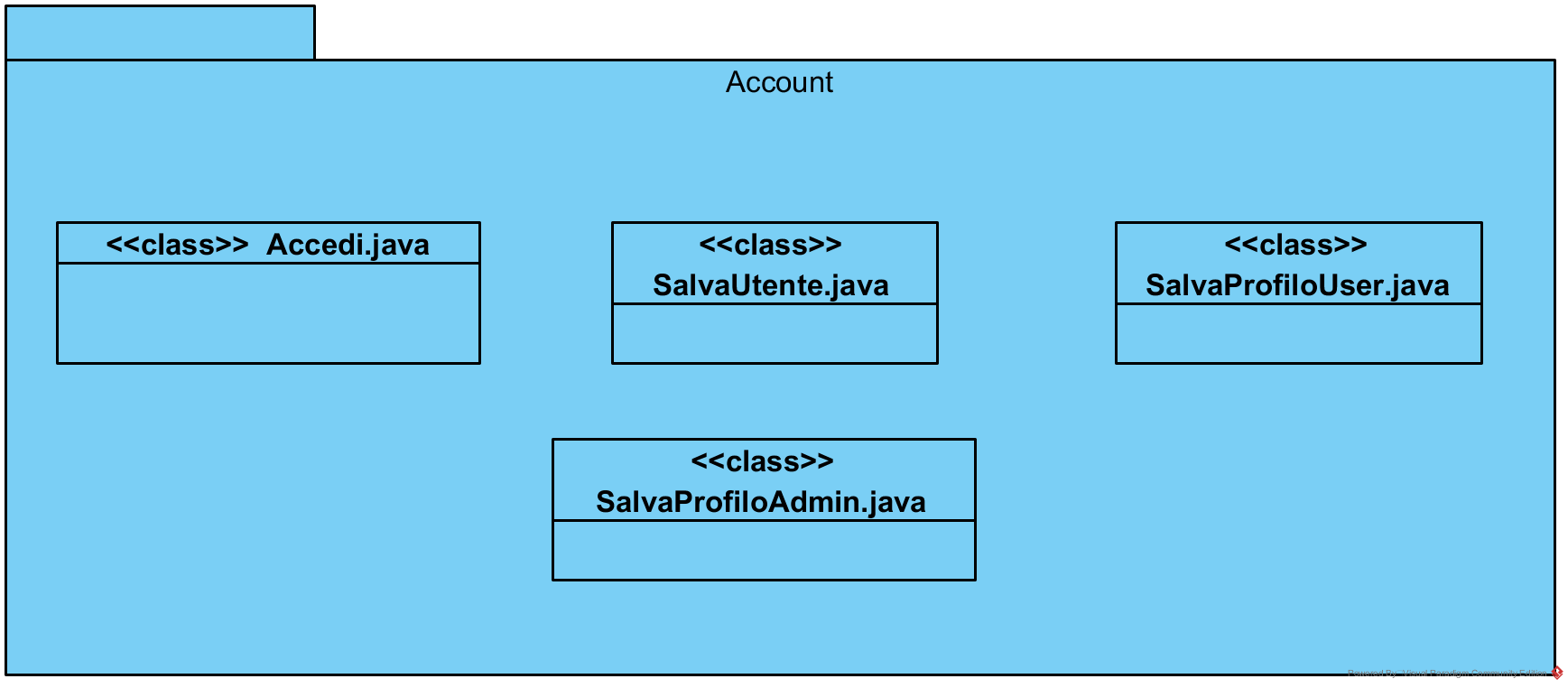
|  |  |
| --- | --- |
| Classe | Descrizione |
| contattiGestore.jsp | La view che mostra i contatti della piattaforma. |
| documenti\_immagini.jsp | La view che mostra la pagina dedicata all’aggiunta immagine di un documento. |
| documentiGestore.jsp | La view che mostra i documenti inseriti dal gestore e la possibilita di aggiunta,modifica prezzo,aggiunta immagine ed eliminazione di un documento. |
| file\_richiesta.jsp | La view che mostra la pagina dedicata all’apertura o eliminazione di un documento inviato da uno studente. |
| infoGestore.jsp | La view che mostra la pagina dedicata alla modifica dei dati personali. |
| materieGestore.jsp | La view che mostra la pagina dedicata all’aggiunta/eliminazione di una materia. |
| ordiniGestore.jsp | La view che mostra la pagina dedicata agli ordini effettuati dagli studenti e la possibilità di visualizzare tutti i dettagli. |
| profiloGestore.jsp | La view che mostra il profilo personale del gestore del sistema. |
| richiesteGestore.jsp | La view che mostra la pagina dedicata alle richieste |

* + 1. **Package Studente**



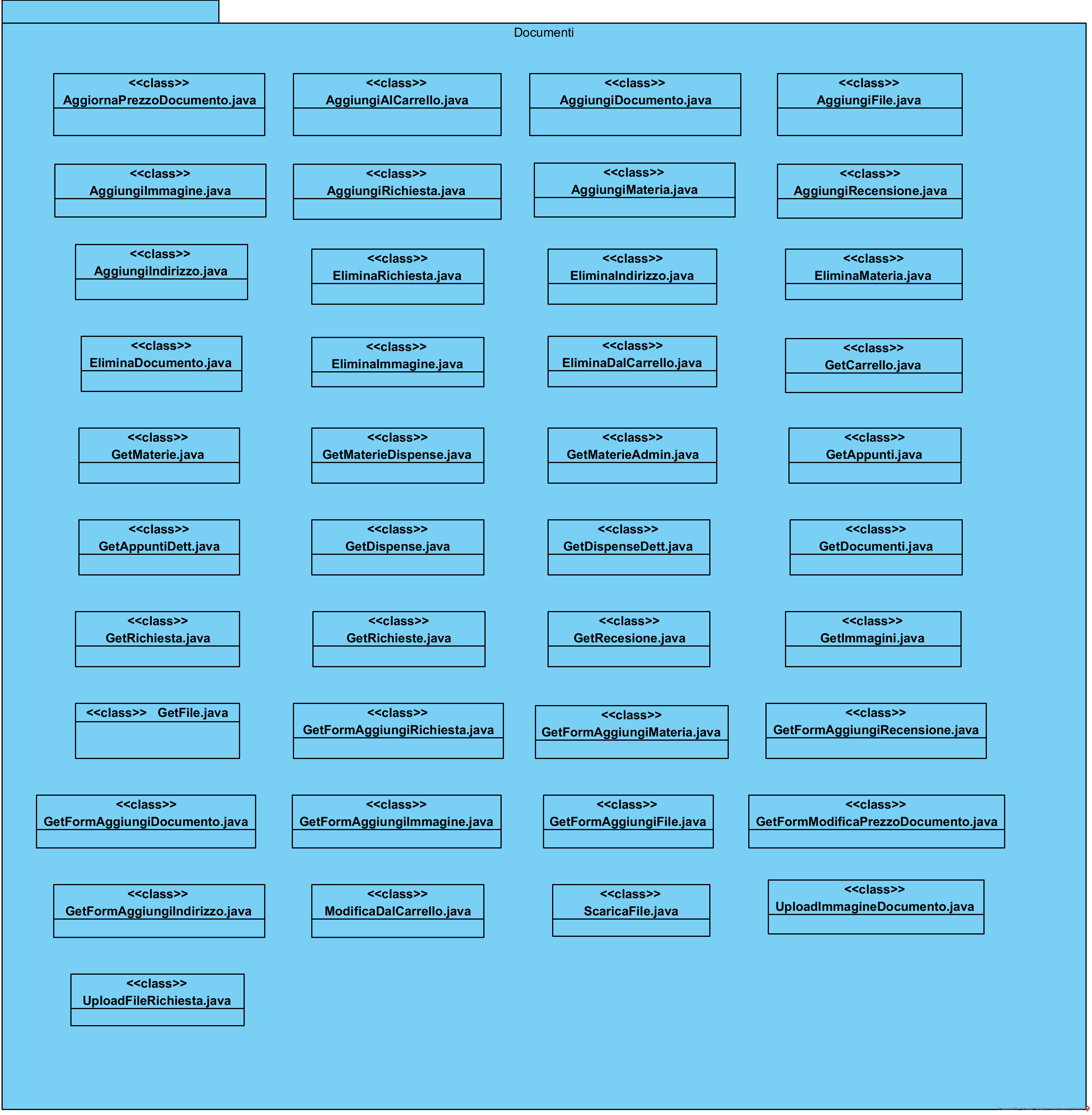
|  |  |
| --- | --- |
| Classe | Descrizione |
| carrello.jsp | La view che mostra il carrello dello studente. |
| checkout.jsp | La view che mostra la pagina relativa al finalizzazione dell’ordine. |
| ComeFunzionaStudente.jsp | La view che spiega l’utilizzo del sistema. |
| conferma\_ordine.jsp | La view che mostra i documenti da acquistare. |
| contattiStudente.jsp | La view che mostra i contatti della piattaforma. |
| appuntiStudente.jsp | La view che mostra gli appunti presenti sulla piattaforma. |
| dispenseStudente.jsp | La view che mostra le disense presenti sulla piattaforma. |
| faqStudente.jsp | La view che mostra Domande&Risposte relative all’utilizzo del sistema. |
| file\_richiesta.jsp | La view che mostra la pagina dedicata all’apertura o eliminazione di un documento. |
| infoStudente.jsp | La view che mostra la pagina dedicata alla modifica dei dati personali. |
| invioRichiestaStudente.jsp | La view che mostra la pagina dedicata all’invio della richiesta di pubblicazione documento. |
| ordiniStudente.jsp | La view che mostra gli ordini effettuati dallo studente. |
| profiloStudente.jsp | La view che mostra il profilo personale dello studente. |

* 1. **Package Account**



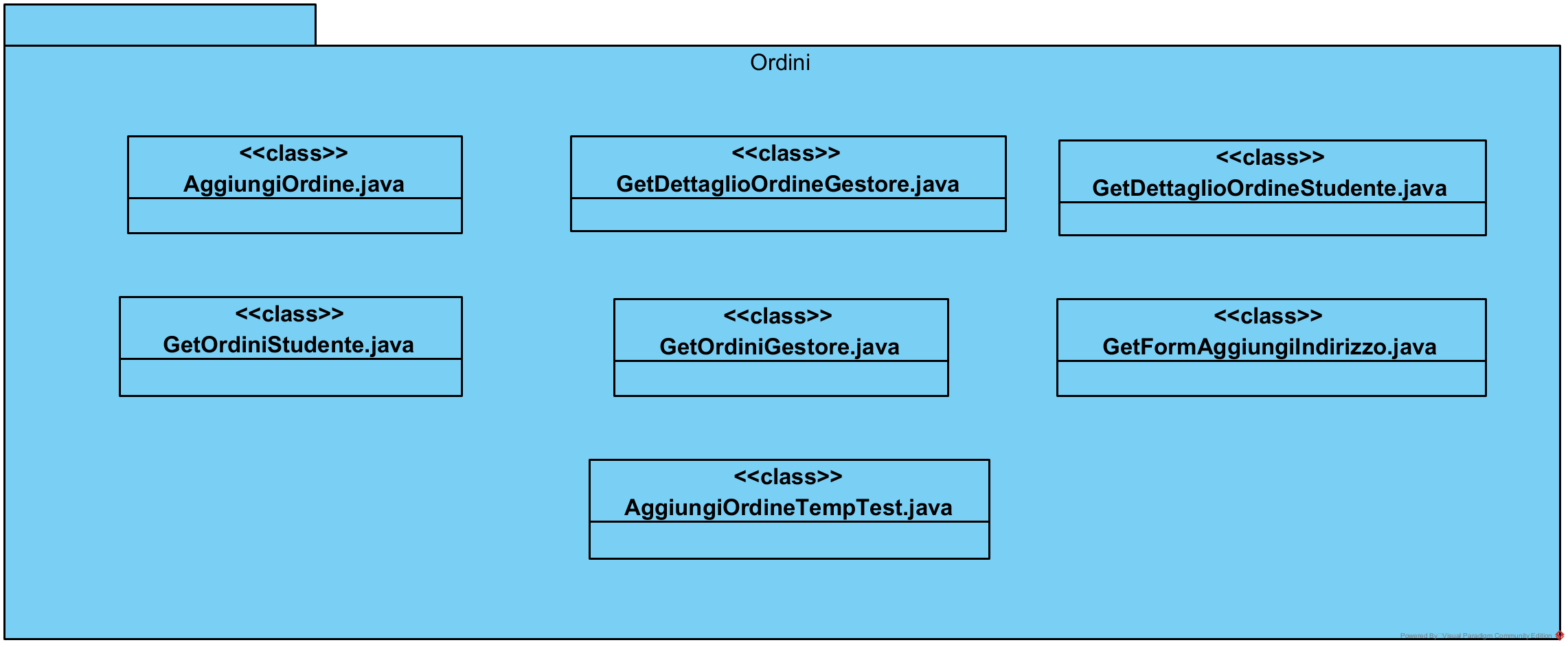
|  |  |
| --- | --- |
| Classe | Descrizione |
| Accedi.java | Servlet che permette la gestione del login. |
| SalvaUtente.java | Servlet che permette la gestione della registrazione dello studente. |
| SalvaProfiloUser.java | Servlet che permette la gestione delle modifiche delle info personali dello studente. |
| SalvaProfiloAdmin.java | Servlet che permette la gestione delle modifiche delle info personali del gestore. |

* 1. **Package Documenti**



|  |  |
| --- | --- |
| Classe | Descrizione |
| AggiornaPrezzoDocumentoGestore.java | Servlet che permette la gestione dell’aggiornamento del prezzo del documento. |
| AggiungiAlCarrello.java | Servlet che permette la gestione di aggiunta di un documento al carrello. |
| AggiungiDocumento.java | Servlet che permette la gestione di aggiunta di un documento. |
| AggiungiFile.java | Servlet che permette la gestione di aggiunta di un file (documento della richiesta). |
| AggiungiImmagine.java | Servlet che permette la gestione di aggiunta di un’ immagine. |
| AggiungiRichiesta.java | Servlet che permette la gestione di aggiunta di una richiesta. |
| AggiungiMateria.java | Servlet che permette la gestione di aggiunta di una materia. |
| AggiungiRecensione.java | Servlet che permette la gestione di aggiunta di una recensione da parte di uno studente. |
| AggiungiIndirizzo.java | Servlet che permette la gestione di aggiunta di un indirizzo da parte di uno studente. |
| EliminaRichiesta.java | Servlet che permette la gestione di rimozione di una richiesta. |
| EliminaIndirizzo.java | Servlet che permette la gestione di rimozione di un indirizzo. |
| EliminaMateria.java | Servlet che permette la gestione di rimozione di una materia. |
| EliminaDocumento.java | Servlet che permette la gestione di rimozione di un documento. |
| EliminaImmagine.java | Servlet che permette la gestione di rimozione di un’immagine. |
| EliminaFile.java | Servlet che permette la gestione di rimozione di un file. |
| EliminaDalCarrello.java | Servlet che permette la gestione di rimozione di un appunto dal carrello. |
| GetCarrello.java | Servlet che permette la gestione di visualizzazione del carrello. |
| GetMaterie.java | Servlet che permette la gestione di visualizzazione delle materie. |
| GetMaterieDispense.java | Servlet che permette la gestione di visualizzazione delle materie per le dispense. |
| GetMaterieAdmin.java | Servlet che permette la gestione di visualizzazione delle materie da parte dell’admin. |
| GetAppunti.java | Servlet che permette la gestione di visualizzazione degli appunti. |
| GetAppuntiDett.java | Servlet che permette la gestione di visualizzazione della pagina descrittiva di un appunto. |
| GetDispense.java | Servlet che permette la gestione di visualizzazione degli dispense. |
| GetDispenseDett.java | Servlet che permette la gestione di visualizzazione della pagina descrittiva di una dispensa. |
| GetDocumenti.java | Servlet che permette la gestione di visualizzazione dei documenti da parte del gestore. |
| GetRichiesta.java | Servlet che permette la gestione di visualizzazione delle richiesta. |
| GetRichieste.java | Servlet che permette la gestione di visualizzazione delle richieste. |
| GetRecesione.java | Servlet che permette la gestione di visualizzazione delle recensioni. |
| GetImmagini.java | Servlet che permette la gestione di visualizzazione di un’immagine. |
| GetFile.java | Servlet che permette la gestione di visualizzazione di un file (documento) |
| GetFormAggiungiRichiesta.java | Servlet che permette la gestione di visualizzazione del form per inviare una richiesta. |
| GetFormAggiungiMateria.java | Servlet che permette la gestione di visualizzazione del form per aggiungere una materia. |
| GetFormAggiungiRecensione.java | Servlet che permette la gestione di visualizzazione del form per aggiungere una recensione. |
| GetFormAggiungiDocumento.java | Servlet che permette la gestione di visualizzazione del form per aggiungere un documento. |
| GetFormAggiungiImmagine.java | Servlet che permette la gestione di visualizzazione del form per aggiungere un’immagine. |
| GetFormAggiungiFile.java | Servlet che permette la gestione di visualizzazione del form per aggiungere un file (documento .pdf/.docx). |
| GetFormModificaPrezzoDocumento.java | Servlet che permette la gestione di visualizzazione del form per modificare un documento. |
| GetFormAggiungiIndirizzo.java | Servlet che permette la gestione di visualizzazione del form per aggiungere un indirizzo |
| ModificaDalCarrello.java | Servlet che permette la gestione di modifica dei documenti dal carrello. |
| ScaricaFile.java | Servlet che permette l’apertura del file(documento della rixhiesta). |
| UploadImmagineDocumento.java | Servlet che permette la gestione dell’upload dell’immagine del documento. |
| UploadFileRichiesta.java | Servlet che permette la gestione dell’upload del file della richiesta. |

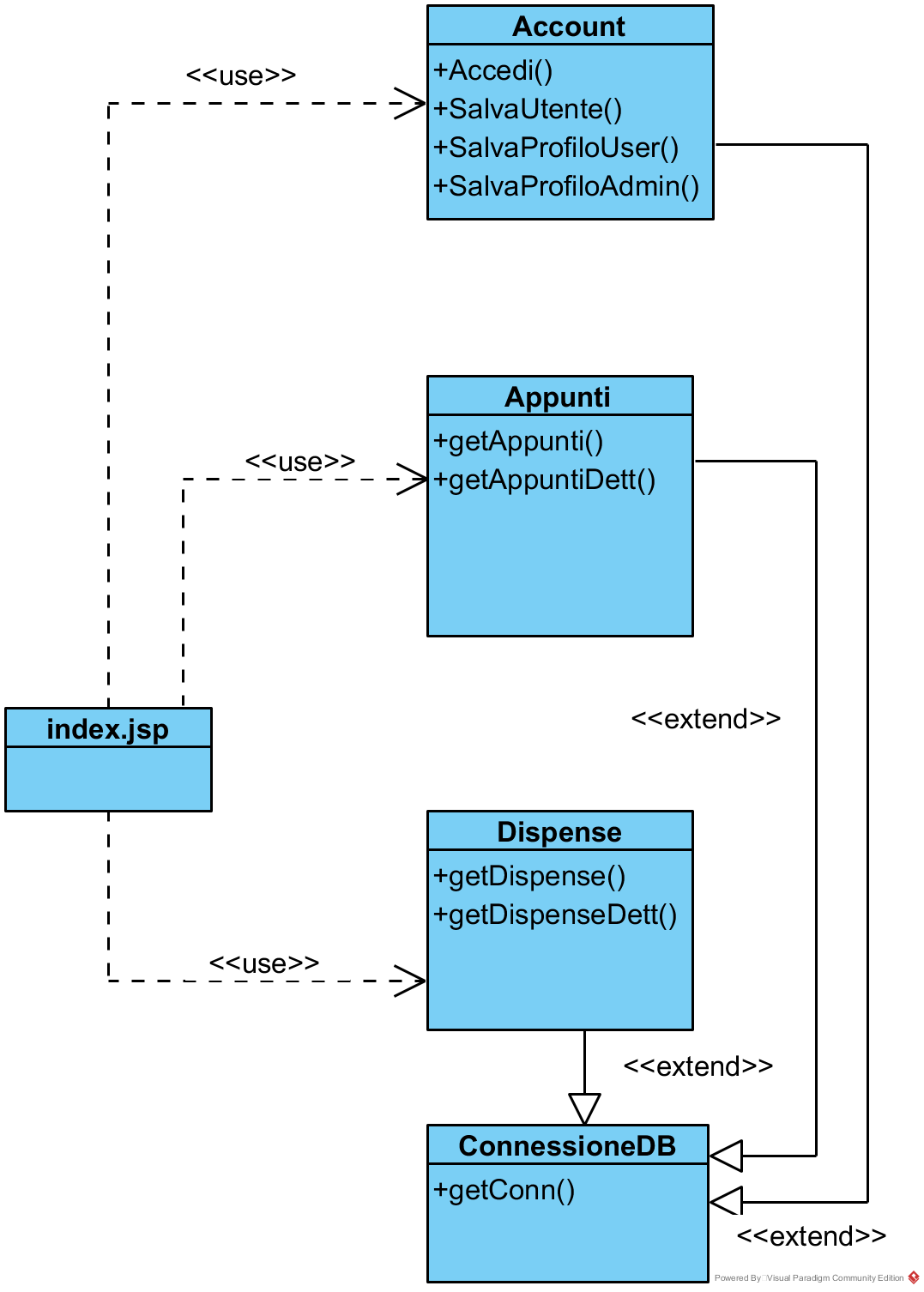
* 1. **Package Ordini**



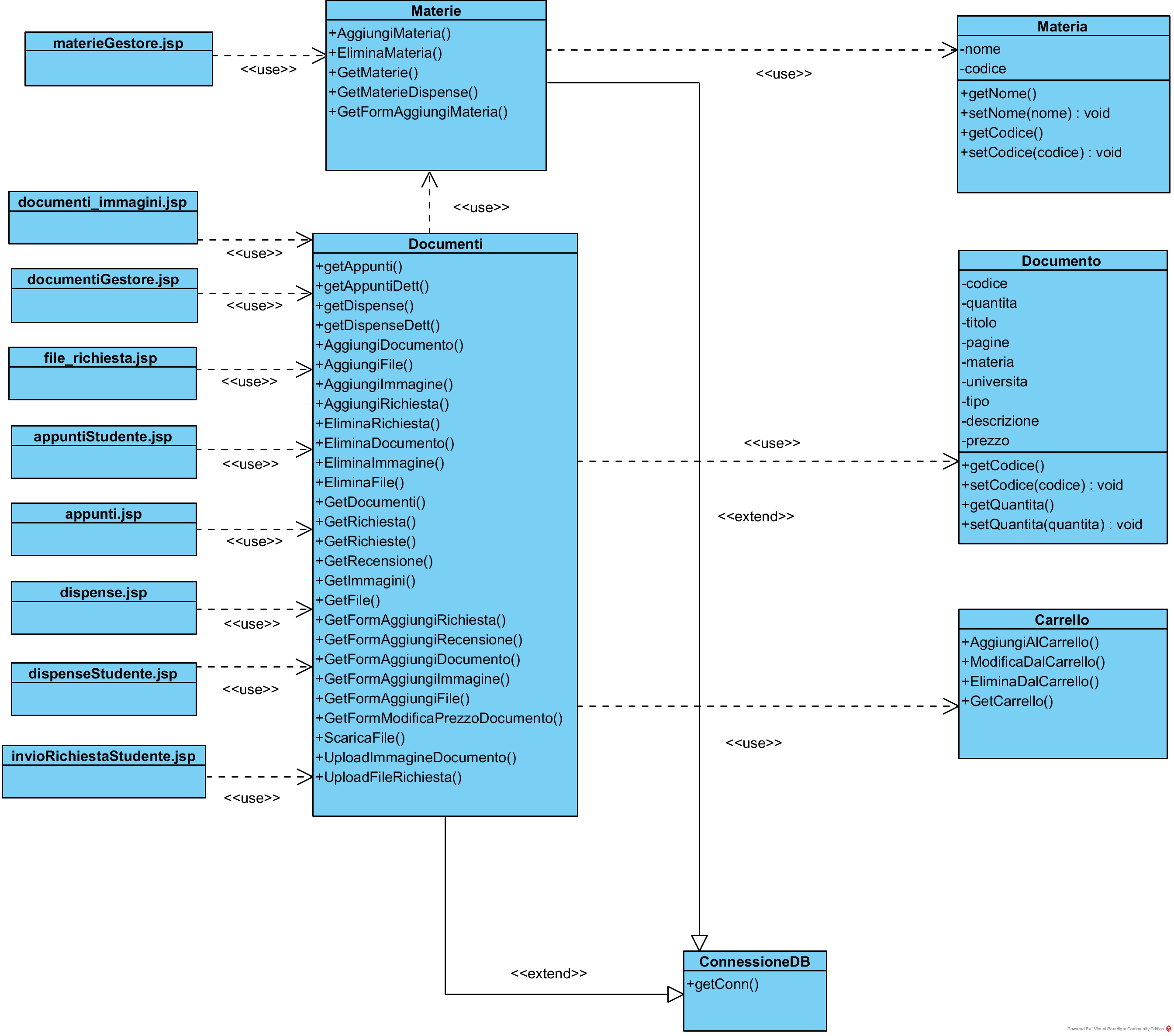
|  |  |
| --- | --- |
| Classe | Descrizione |
| AggiungiOrdine.java | Servlet che permette la gestione di aggiunta dell’ordine. |
| AggiungiOrdineTemp.java | Servlet che permette la gestione di aggiunta dell’ordine temporaneo. |
| GetDettaglioOrdineGestore.java | Servlet che permette la gestione di visualizzazione degli ordini da parte del gestore. |
| GetDettaglioOrdineStudente.java | Servlet che permette la gestione di visualizzazione degli ordini da parte dello studente. |
| GetOrdiniGestore.java | Servlet che permette la gestione di visualizzazione degli ordini da parte del gestore. |
| GetOrdiniStudente.java | Servlet che permette la gestione di visualizzazione degli ordini da parte dello studente. |

1. **Interfacce delle classi**

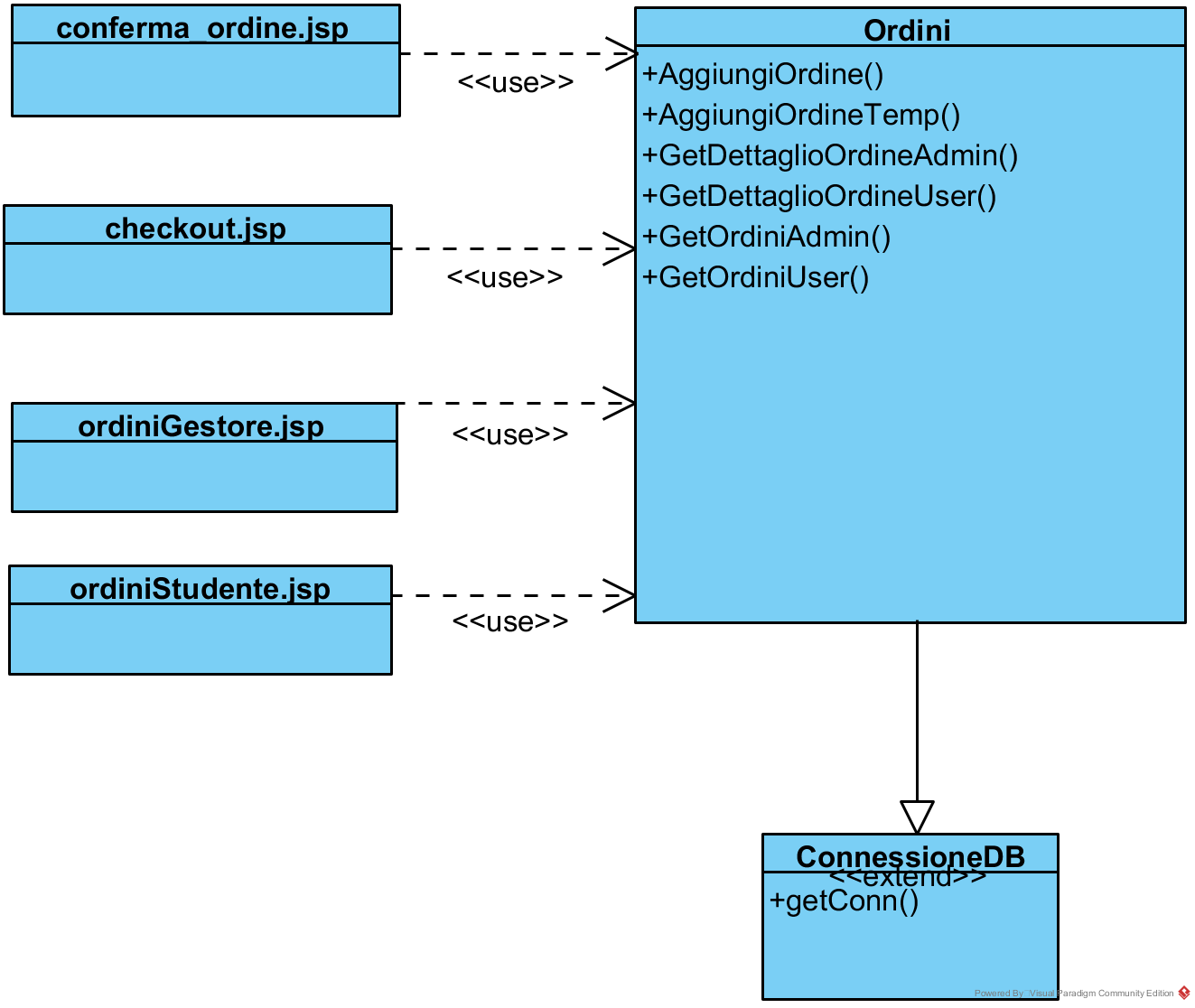
***Sottosistema Account***



***Sottosistema Documenti***



***Sottosistema Ordini***



***Glossario***

|  |  |
| --- | --- |
| **Sigla/Termine** | **Definizione** |
| urNotes | Nome del sistema che verrà sviluppato. |
| Servlet | Le Servlet sono oggetti scritti in linguaggio Java che operano all’interno di un server web (es. Tomcat) permettendo la creazione di applicazioni web. |
| Query | Interrogazione di un database per estrarre o aggiornare i dati che soddisfano un certo criterio di ricerca. |
| Sito web | Un insieme di pagine web correlate. |
| Layer | Utilizzato per indicare la suddivisione nell’architettura three-tier (tre strati). |
| DBMS | Sistema di gestione di basi di dati, cioè un sistema software progettato per consentire la creazione, la manipolazione e l’interrogazione efficiente di un database. |
| MySQL | È un Relational database management system (RDBMS) composto da un client a riga di comando e un server. |
| Apache Tomcat | È un application server nella forma di contenitore servlet open source. |