ODD Object Design Document

Dashboard Dipartimento Informatica

Antonio Aurimma
Napolitano Maddalena
Peluso Maurizio
Scavone Fancesca

Data	Versione	Cambiamenti	Autori
27/11/2017	Draft 0.1	Prima stesura	Tutti
30/11/2017	Draft 0.2	Completamente e stesura delle linee guida per la documentazione	Scavone Francesca
1/11/2017	Draft 0.3	Stesura ed individuazione delle interfacce delle classi	Auriemma Antonio, Peluso Maurizio
2/11/2017	Draft 0.4	Individuazione e stesura dei Design Pattern	Peluso Maurizio, Napolitano Maddalena
4/11/2017	Draft 0.5	Individuazione e prima stesura dei Packages	Peluso Maurizio
7/11/2017	Draft 0.6	Revisione delle linee guida	Napolitano Maddalena
8/11/2017	Draft 0.7	Revisione e aggiornamento dei packages	Peluso Maurizio, Napolitano Maddalena
11/11/2017	V_1.0	Revisione documento	Tutti
15/12/2017	V_1.5	Aggiornamento e verifica di coesione	Tutti
26/01/2018	V_2.0	Revisione finale documento	Tutti

Sommario

1. Introduzione

1.1. Object design trade-offs

Sicurezza vs efficienza

La sicurezza rappresenta uno degli aspetti cardine del sistema, come specificato anche nei requisiti non funzionali presenti nel RAD. Per tale ragione sarà data molta importanza ai dati personali degli studenti e di tutte le altre figure che si approcciano al nostro sistema. L'autenticazione di un utente all'interno del sistema avviene fornendo username e password, così facendo l'utente potrà accedere solo a specifiche funzionalità a seconda del tipo di account a cui fa riferimento.

Tempo di risposta vs spazio di memoria

Quando all'interno del sistema vengono a mancare i requisiti di tempo di risposta, si vedrà necessario utilizzare maggiore memoria per velocizzare il sistema.

Memoria vs efficienza

La priorità viene data all'efficienza in quanto il nostro sistema è basato sull'invio di un modulo che deve raggiungere le diverse figure coinvolte per essere validato, per tale ragione si deve avere la certezza che le operazioni effettuate e i risultati siano corretti. La memorizzazione dei dati, però, non viene a mancare in quanto si necessità di mantenere la tracciabilità di tutte le operazioni ed è indispensabile il salvataggio dei profili utente.

Interfaccia vs usabilità

L'interfaccia grafica verrà realizzata in modo da risultare molto semplice, chiara e concisa. Ogni dettaglio sarà curato per essere di facile comprensione: i form o i pulsanti verranno disposti in modo tale da rendere semplice l'utilizzo del sistema da parte dell'utente finale.

Comprensibilità vs costi

Si dà priorità alla comprensibilità piuttosto che ai costi al fine di rendere il codice comprensibile anche a coloro che non sono direttamente coinvolti nel progetto o per coloro che, nonostante lo siano, non hanno lavorato a particolari punti dell'implementazione. Saranno presenti dei commenti all'interno del codice al fine di semplificarne la comprensione, andando di conseguenza ad agevolare anche il processo di manutenzione del sistema rendendolo il lavoro di modifica più agevole.

1.2 Linee guida per la documentazione dell'interfaccia

Di seguito il link a Google Java Style Guide:

http://checkstyle.sourceforge.net/reports/google-java-style-20170228.html#s4.8.2.1-variables-per-declaration

1.3 Design Pattern

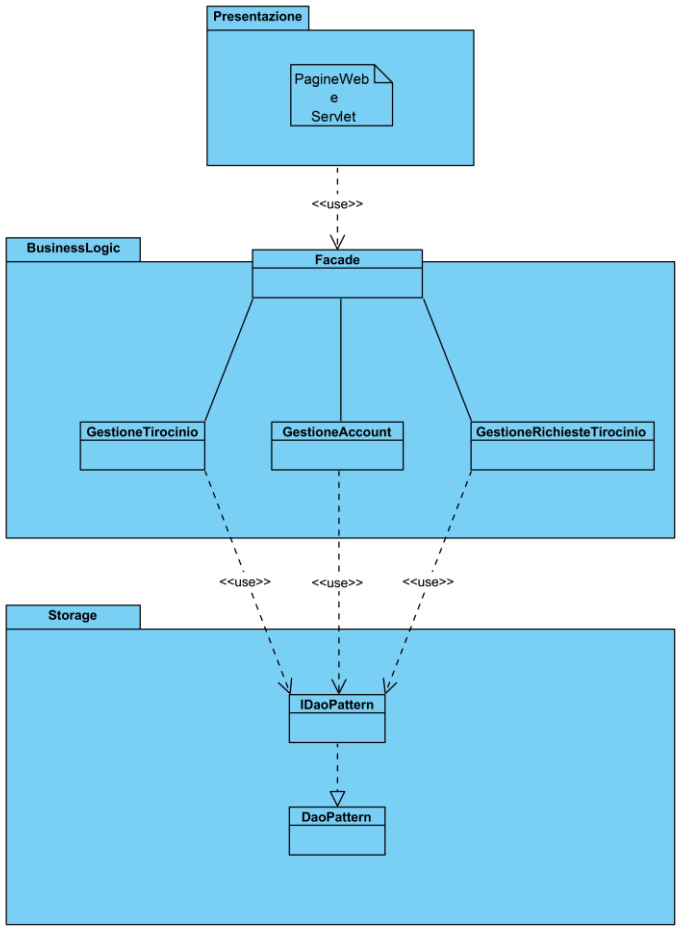
Facade Pattern

Il facade pattern permette ad un'unica interfaccia di accedere a quegli oggetti che compongono un dato sottosistema. Questo tipo di pattern può essere utilizzato da tutti i sottosistemi di un software. Il sistema MTO utilizza il facade pattern per definire un'unica interfaccia per i vari sottosistemi che, permettono all'utente di interagire, attraverso l'interfaccia, con le funzionalità del sistema. Tali funzionalità verranno percepite e viste come un unico sistema. Quindi, il client interagisce con linterfaccia del pattern facade, IFacade, e vengono delegate le operazioni delle classi dei sottosistemi con cui interagisce.

DAO Pattern

Il DAO (Data Access Object) è un pattern architetturale per la gestione dei dati persistenti. Viene usata principalmente in applicazioni web, per stratificare e isolare l'accesso ad una tabella tramite query (poste all'interno dei metodi della classe nello storage layer) creando un maggiore livello di astrazione ed una più facile manutenibilità. I metodi del DAO con le rispettive query dentro verranno così richiamati dalle classi della business logic. Il vantaggio relativo all'uso del DAO è dunque il mantenimento di una rigida separazione tra le componenti di un'applicazione.

Di seguito la rappresentazione grafica dei pattern utilizzati:



1.4 Definizioni, acronimi e abbreviazioni

M.T.O. = Moduli di Tirocinio Online, nome del sistema.
RAD = Requirement Analysis Document.
SDD = System Design Document.
UNISA = Università degli Studi di Salerno.
User-friendly = software di facile utilizzo anche per chi non è esperto.
UML = è un metodo per descrivere l'architettura di un sistema in dettaglio.
File = contenitore di informazioni/dati in formato digitale.

1.5 Riferimenti

- Prentice Object Oriented Software Engineering Using UML Patterns and Java
- Ghezzi Jazayeri Mandrioli - Ingegneria del Software
- RAD_Completo_V1.3
- SDD_V1.0

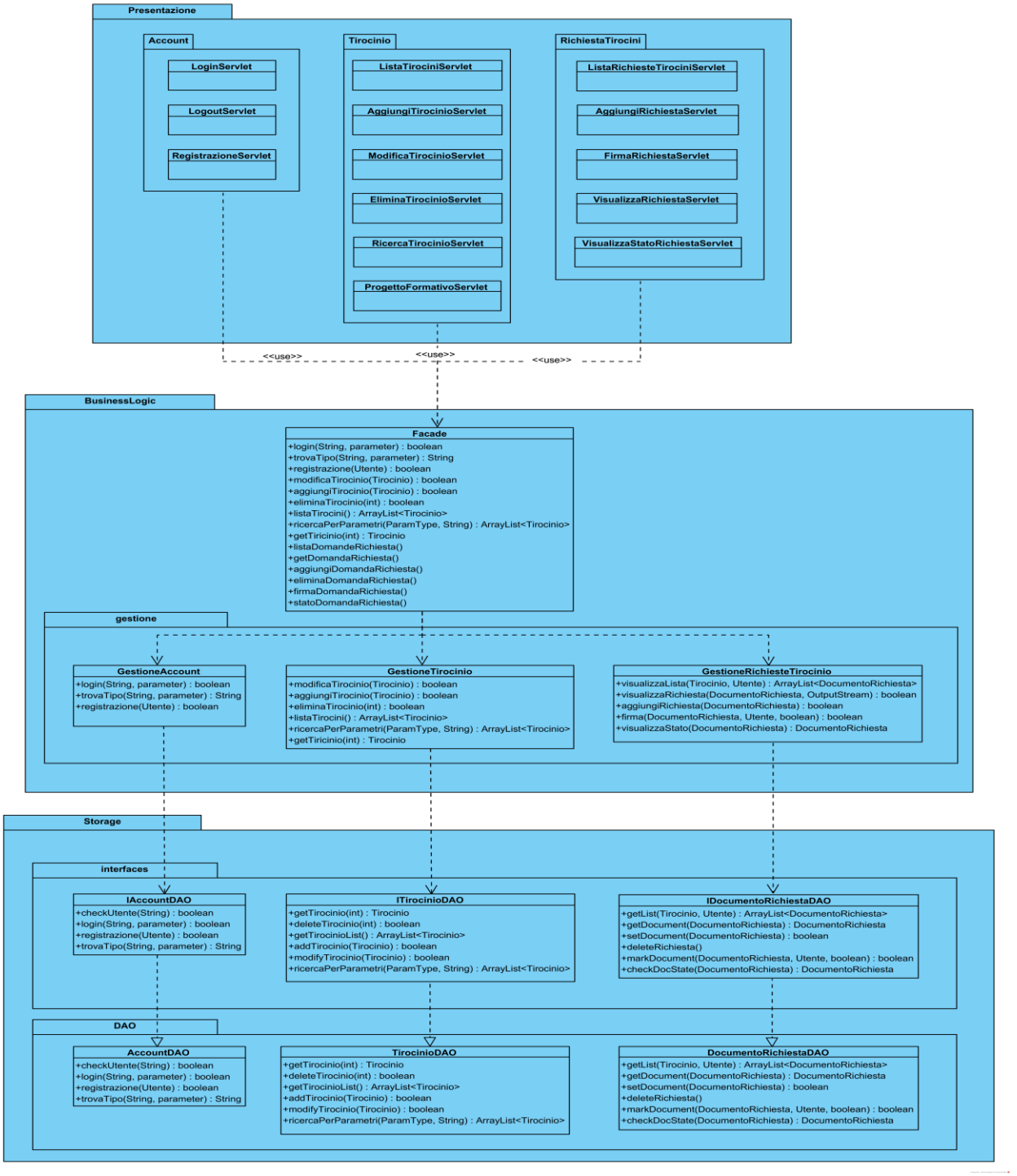
2 Packages

In questa sezione del documento verranno mostrati i vari packages e l'organizzazione in file del codice che verrà scritto per sviluppare il sistema.
Il sistema MTO, in base al punto 3.2.2 "Decomposizione in sottosistemi" presente nel documento SDD, è costituito da 3 layer, basato sull'architettura "Three Trial".
Il sistema uilizza sia facade pattern che il bridge pattern; per questo motivo è stato introdotto un unico pacake chiamato Common.
Il package Common ha al suo interno le classi che identificano le tabelle del database e che verranno utilizzate da tutti gli altri pacchetti.
Gli altri packages hanno il nome dei layer di cui si compone il sistema, come specificato nel documento SDD; e ognuno è composto da altri sottopacchetti sono:

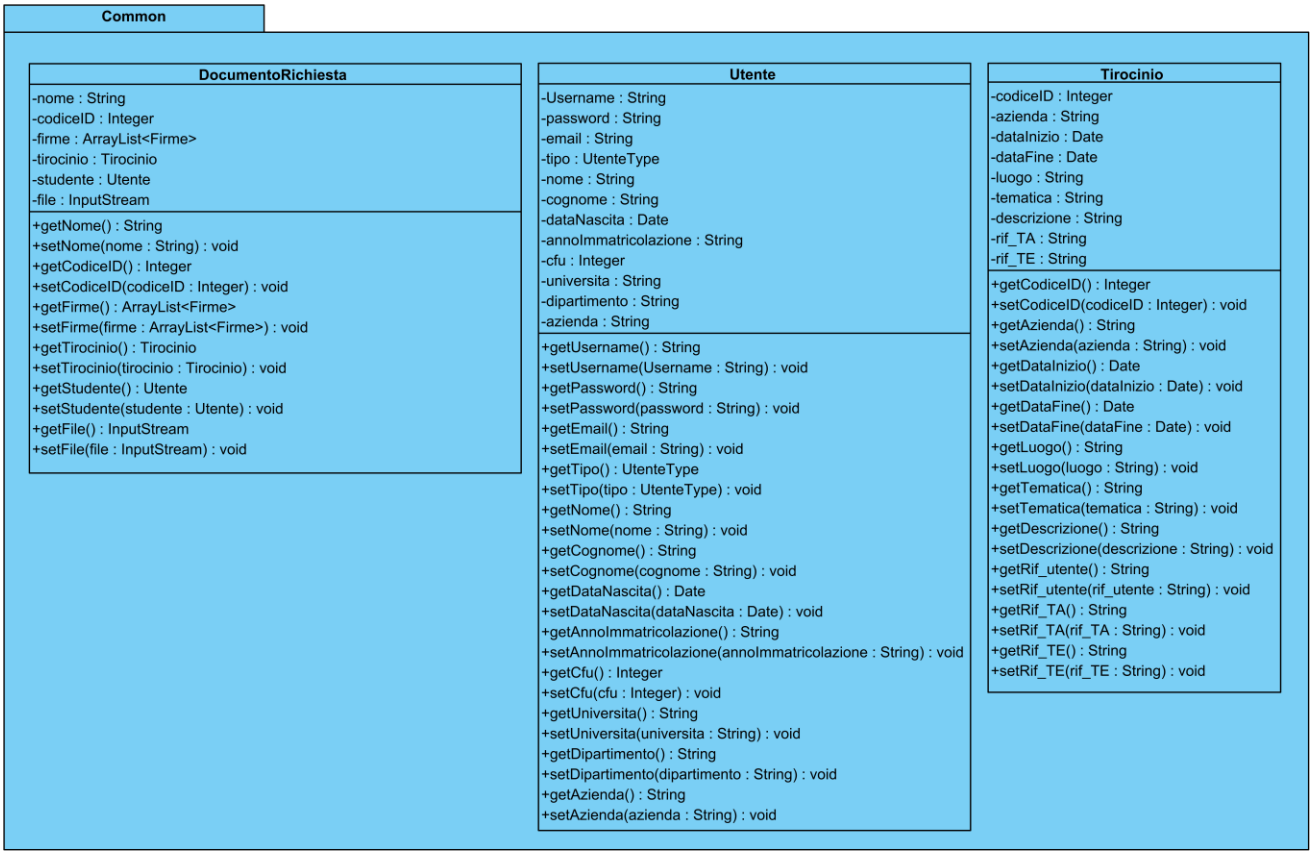
- Presentazione
    - Account
    - Tirocinio
    - RichiestaTirocini
- BusinessLogic
    - Gestioni
- Storage
    - Interfaces
    - DAO

In questi pacchetti ci sono tre classi : Utente, Tirocinio e DocumentoRichiesta che, verranno utilizzate da tutti elementi che si trovano nel pacchetto a cui appartengono; ecco perché nella rappresentazione grafica non vi è alcun collegamento.

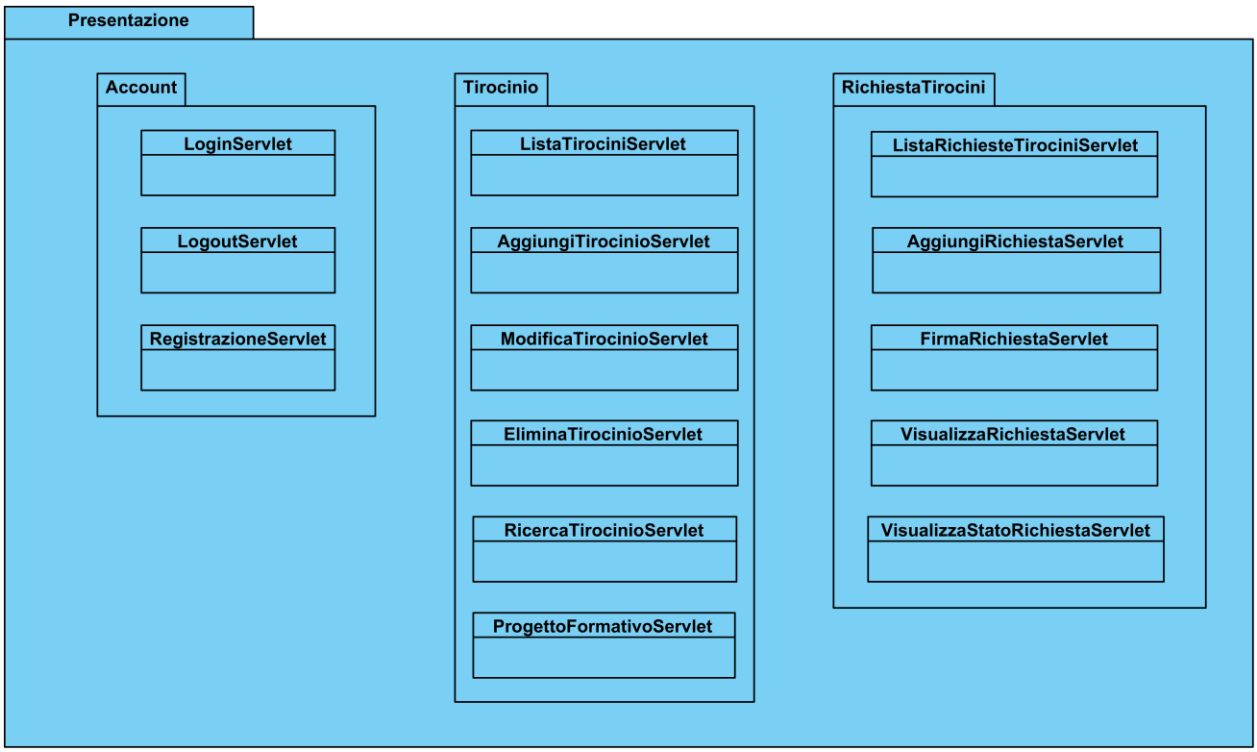
Ecco il modello grafico relativo al Package Generale :


Presentazione
Account
LoginServlet
LogoutServlet
RegistrazioneServlet
Tirocinio
ListaTirociniServlet
AggiungiTirocinioServlet
ModificaTirocinioServlet
EliminaTirocinioServlet
RicercaTirocinioServlet
ProgettoFormativoServlet
RichiestaTirocini
ListaRichiesteTirociniServlet
AggiungiRichiestaServlet
FirmaRichiestaServlet
VisualizzaRichiestaServlet
VisualizzaStatoRichiestaServlet
<<use>>
<<use>>
<<use>>
BusinessLogic
Facade
+login(String, parameter) : boolean
+trovaTipo(String, parameter) : String
+registrazione(Utente) : boolean
+modificaTirocinio(Tirocinio) : boolean
+aggiungiTirocinio(Tirocinio) : boolean
+eliminaTirocinio(int) : boolean
+listaTirocini() : ArrayList<Tirocinio>
+ricercaPerParametri(ParamType, String) : ArrayList<Tirocinio>
+getTiricinio(int) : Tirocinio
+listaDomandeRichiesta()
+getDomandaRichiesta()
+aggiungiDomandaRichiesta()
+eliminaDomandaRichiesta()
+firmaDomandaRichiesta()
+statoDomandaRichiesta()
gestione
GestioneAccount
+login(String, parameter) : boolean
+trovaTipo(String, parameter) : String
+registrazione(Utente) : boolean
GestioneTirocinio
+modificaTirocinio(Tirocinio) : boolean
+aggiungiTirocinio(Tirocinio) : boolean
+eliminaTirocinio(int) : boolean
+listaTirocini() : ArrayList<Tirocinio>
+ricercaPerParametri(ParamType, String) : ArrayList<Tirocinio>
+getTiricinio(int) : Tirocinio
GestioneRichiesteTirocinio
+visualizzaLista(Tirocinio, Utente) : ArrayList<DocumentoRichiesta>
+visualizzaRichiesta(DocumentoRichiesta, OutputStream) : boolean
+aggiungiRichiesta(DocumentoRichiesta) : boolean
+firma(DocumentoRichiesta, Utente, boolean) : boolean
+visualizzaStato(DocumentoRichiesta) : DocumentoRichiesta
Storage
interfaces
IAccountDAO
+checkUtente(String) : boolean
+login(String, parameter) : boolean
+registrazione(Utente) : boolean
+trovaTipo(String, parameter) : String
ITirocinioDAO
+getTirocinio(int) : Tirocinio
+deleteTirocinio(int) : boolean
+getTirocinioList() : ArrayList<Tirocinio>
+addTirocinio(Tirocinio) : boolean
+modifyTirocinio(Tirocinio) : boolean
+ricercaPerParametri(ParamType, String) : ArrayList<Tirocinio>
IDocumentoRichiestaDAO
+getList(Tirocinio, Utente) : ArrayList<DocumentoRichiesta>
+getDocument(DocumentoRichiesta) : DocumentoRichiesta
+setDocument(DocumentoRichiesta) : boolean
+deleteRichiesta()
+markDocument(DocumentoRichiesta, Utente, boolean) : boolean
+checkDocState(DocumentoRichiesta) : DocumentoRichiesta
DAO
AccountDAO
+checkUtente(String) : boolean
+login(String, parameter) : boolean
+registrazione(Utente) : boolean
+trovaTipo(String, parameter) : String
TirocinioDAO
+getTirocinio(int) : Tirocinio
+deleteTirocinio(int) : boolean
+getTirocinioList() : ArrayList<Tirocinio>
+addTirocinio(Tirocinio) : boolean
+modifyTirocinio(Tirocinio) : boolean
+ricercaPerParametri(ParamType, String) : ArrayList<Tirocinio>
DocumentoRichiestaDAO
+getList(Tirocinio, Utente) : ArrayList<DocumentoRichiesta>
+getDocument(DocumentoRichiesta) : DocumentoRichiesta
+setDocument(DocumentoRichiesta) : boolean
+deleteRichiesta()
+markDocument(DocumentoRichiesta, Utente, boolean) : boolean
+checkDocState(DocumentoRichiesta) : DocumentoRichiesta

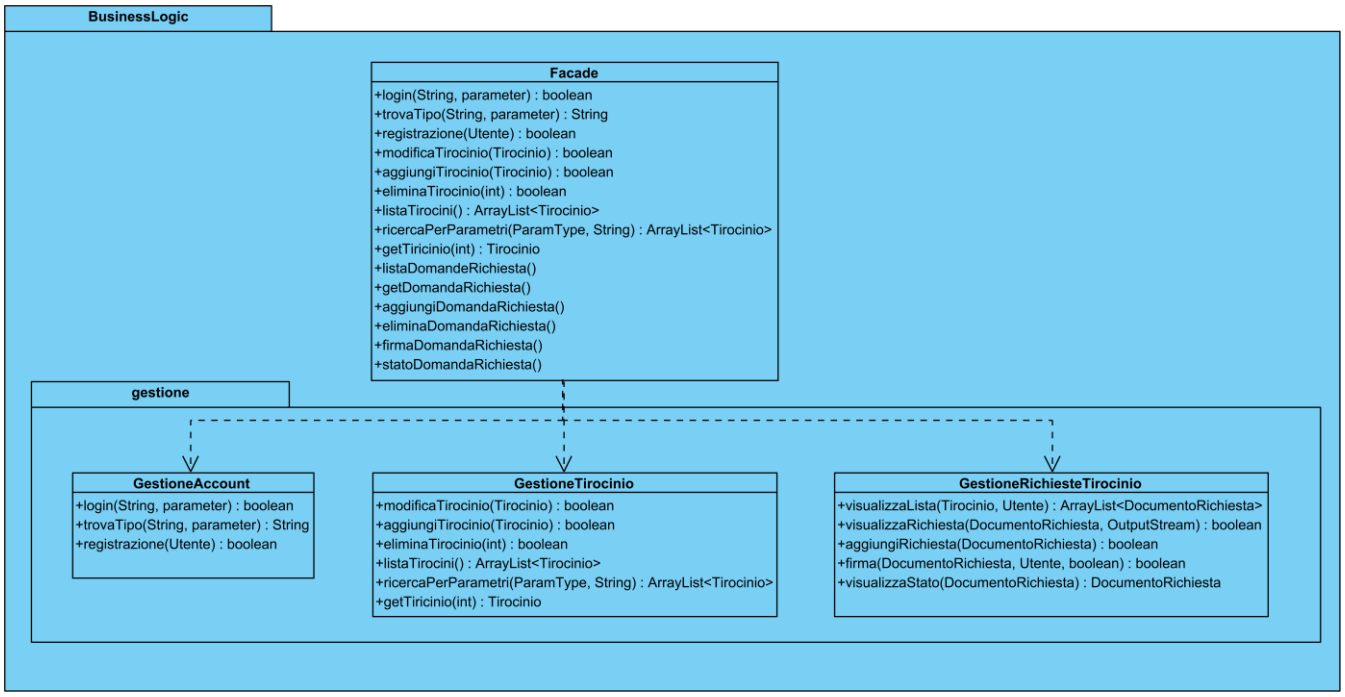
Ecco il modello grafico relativo al pacchetto Common:



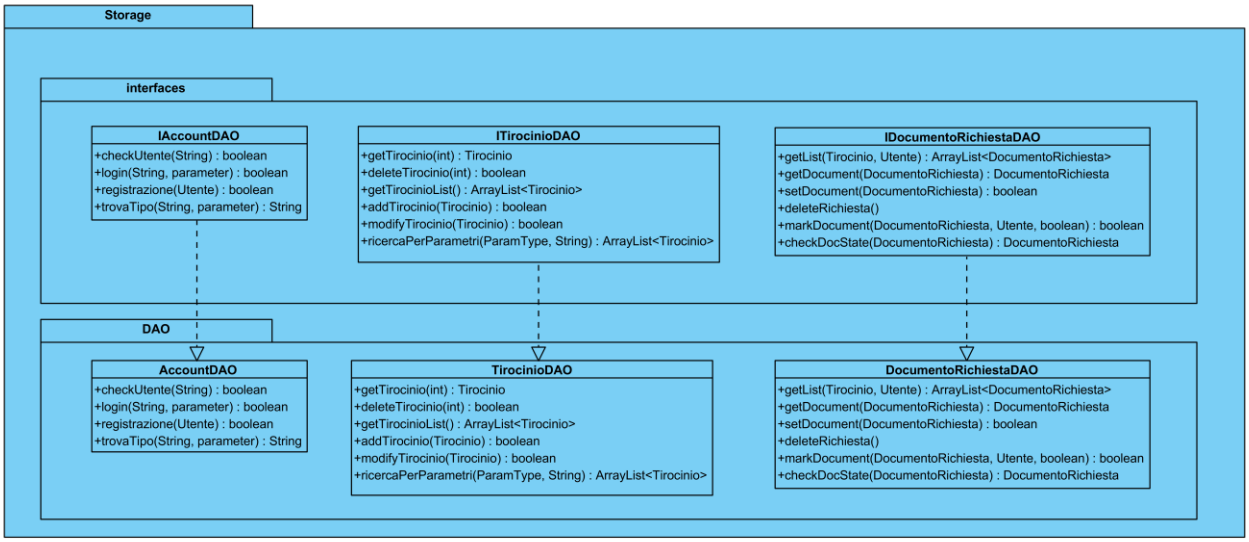
Ecco il modello grafico relativo al pacchetto di Presentazione:



Ecco il modello grafico relativo al pacchetto di BusinessLogic:



Ecco il modello grafico relativo al pacchetto dello Storage:



3 Interfacce delle classi

Questo paragrafo descrive le classi e le interfacce pubbliche utilizzate:

Nome Classe	GestioneAccount
Descrizione	Questa classe gestisce gli account, permettendo l'accesso, l'uscita, e la registrazione di un nuovo account
Pre-condizioni	**context** GestioneAccount: Boolean logIn(username, password)**pre**: username!=null && password!=null **context** GestioneAccount: Boolean registrazione(Utente); **pre:** Utente.username != null && Utente.password != null && Utente.email != null && Utente.nome != null && Utente.cognome != null && Utente.dataDiNascita != null && Utente.annoDiImmatricolazione && Utente.cfu != null
Post-condizione	
Invarianti	

Nome Classe	GestioneTirocinio
Descrizione	Questa classe gestisce I tirocini, permettendone l'aggiunta, la modifica e l'eliminazione.
Pre-condizioni	**context** GestioneTirocinio: Boolean aggiungiTirconio(Tirocinio)**pre** Tirocinio.azienda != null && Tirocinio.dataInizio != null && Tirocinio.dataFine != null && Tirocinio.luogo != nul && Tirocinio.tematica != null && Tirocinio.descrizione != null **context** GestioneTirocinio: Boolean modificaTirocinio(Tirocinio);) **pre**: Tirocinio.azienda != null && Tirocinio.dataInizio != null && Tirocinio.dataFine != null && Tirocinio.luogo != nul && Tirocinio.tematica != null && Tirocinio.descrizione != null **context** GestioneTirocinio: Boolean eliminaTirocinio(Integer id)**pre:** id != null && id > 0
Post-condizione	
Invarianti	

Nome Classe	GestioneRichiesteDiTirocinio
Descrizione	Questa classe gestisce le richieste di Tirocinio, permettendo il caricamento dei documenti, la visualizzazione dell'elenco delle richieste, dello stato delle pratiche, dei documenti e dell'elenco dei tirocini, e la possibilità di stabilire lo stato delle richieste
Pre-condizioni	**context** GestioneRichiesteDiTirocinio: Boolean caricaDocumento(DocumentoRichiesta)**pre** DocumentoRichiesta != null **context** GestioneRichiesteDiTirocinio: Boolean visualizzaElencoRichieste(Integer id)**pre:** id != null && id > 0 **context** GestioneAccount: Boolean visualizzaStatoPratica(Integer id)**pre:** id != null && id > 0 **context** GestioneAccount: Boolean visualizzaDocumenti(Integer id)**pre:** id != null && id > 0 **context** GestioneAccount: Boolean visualizzaElencoTirocini(Tipo, String) **pre**: Tipo != null && String != null **context** GestioneAccount: Boolean setStatoRichiesta(Integer id, Boolean valore) pre: id != null && id > 0 && valore != null
Post-condizione	
Invarianti	

4 Glossario

M.T.O. = Moduli di Tirocinio Online, nome del sistema.
RAD = Requirement Analysis Document.
SDD = System Design Document.
UNISA = Università degli Studi di Salerno.
UML = è un metodo per descrivere l'architettura di un sistema in dettaglio.
File = contenitore di informazioni/dati in formato digitale.