



UNIVERSITÀ DEGLI STUDI DI PALERMO
DIPARTIMENTO DI INGEGNERIA
CORSO DI LAUREA TRIENNALE IN INGEGNERIA INFORMATICA

Paper Review

Object Design Document

Studenti

Leonardo Giovanni Caiezza
Diego Corona
Luca Gaetani
Daniele Orazio Susino

Docente

Prof.ssa Valeria Seidita

Anno accademico

2024 - 2025

Indice

1 Introduzione	2
1.1 Prestazione VS costi	2
1.2 Sviluppo Front-End	2
1.2.1 Librerie grafiche	2
1.3 Sviluppo Back-End	2
1.3.1 Librerie Back-End	3
1.4 Tempi di risposta	3
1.5 Linee guida per la documentazione delle interfacce	3
1.6 Linee guida per la documentazione del DBMS	3
1.7 Linee guida per la documentazione della LLM	3
2 Packages	4
2.1 com.paperreview	4
2.1.1 Common	4
2.1.2 Entities	4
2.1.3 Gestione Account	4
2.1.4 Gestione Notifiche	5
2.1.5 Gestione Revisioni	5
2.1.6 Gestione Conferenza	5
2.1.7 Gestione Paper Definitivi	5
2.1.8 Presentazione Articolo	5
2.2 Java	6
2.3 MYSQL-Connector-Java(JDBC)	6
2.4 JavaFX	6
2.5 com.paperreviewServer	6
2.5.1 Gestione Notifiche	6
3 Object Design UML	7
3.1 Gestione Account	7
3.2 Gestione Conferenza	8
3.3 Gestione Notifiche	9
3.4 Gestione Paper Definitivi	9
3.5 Gestione Revisioni	10
3.6 Presentazione Articolo	11
3.7 Common	13
3.8 Entity	14
4 Copyright e diritto d'autore	15

1 Introduzione

1.1 Prestazione VS costi

Per la realizzazione del software sono state utilizzate esclusivamente **librerie Open Source**, minimizzando il tempo ed i costi di sviluppo, permettendo contemporaneamente di ottenere prestazioni più che soddisfacenti.

Inoltre è stato usato un server dotato di una scheda video NVIDIA abilitata CUDA per l'elaborazione del pacchetto riguardante l'LLM.

I **Large Language Models** richiedono una discreta potenza di calcolo per gestire la complessità delle operazioni di deep learning durante l'inferenza. Le GPU Nvidia di ultima generazione sono dotate di CUDA cores che accelerano queste operazioni.

Le **librerie CUDA** sono essenziali per garantire una potenza computazionale adeguata.

1.2 Sviluppo Front-End

Le interfacce del software, grazie alla facilità d'utilizzo e al layout semplice e minimale, permettono all'utente la migliore interazione possibile.

Quando è stato possibile, tutti gli errori ed i messaggi rivolti all'utente sono stati resi non bloccanti per permettere una migliore user experience.

L'interfaccia grafica rivolta all'utente è sviluppata con **JavaFX** e **l'uso di stili CSS**, questa combinazione permette di avere un design moderno ed un'applicazione dinamica.

- **JavaFX (v. 17.0.6):** Tecnologia software che, combinata con Java, consente la creazione e la distribuzione di applicazioni all'avanguardia con contenuti avanzati, audio e video.
Questa è basata su un modello di tipo **MVC (Model-View-Controller)** e per questo è stata adattata al nostro modello **ECB (Entity-Control-Boundary)** adottando le seguenti classi:
 - **Boundary:** si occupa di visualizzare a video l'interfaccia grafica.
 - **Control:** si occupa di gestire i listener delle boundary e di effettuare richieste alle entity.
 - **Entity:** memorizza i dati relativi all'entità assegnata.

1.2.1 Librerie grafiche

- **javafx-controls (v. 17.0.6):** Fornisce i controlli di base per JavaFX, inclusi pulsanti, etichette, caselle di testo, ecc.
- **cssfx (v. 11.4.0):** Libreria di sviluppo per applicare stili CSS senza dover riavviare il programma.
- **javafx-fxml (v. 17.0.6):** Consente di caricare interfacce grafiche definite in FXML.
- **controlsFx (v. 11.2.1):** Utilizzata per il sistema di popup delle notifiche.
- **formsfx-core (v. 11.6.0):** Libreria per la creazione di form con validazione in JavaFX.
- **ikonli-javafx (v. 12.3.1):** Utilizzata per visualizzare le icone nelle UI.
- **ikonli-fontawesome5-pack (v. 12.3.1):** Raccolta di icone FontAwesome per JavaFX.

1.3 Sviluppo Back-End

Lo sviluppo lato server è stato affrontato quasi esclusivamente con **MYSQL e JDBC**.

Lo sviluppo ha avuto come obiettivo principale la possibilità di **utilizzo del software da qualsiasi dispositivo** di tipo PC (Computer Fisso, Laptop, ecc), ciò si traduce nella **responsività dell'interfaccia** a seconda della dimensione dello schermo in cui è visualizzata.

Non è stato implementato un sistema di cache in quanto le uniche informazioni che possono essere salvate al suo interno sono le informazioni riguardanti l'utente e queste vengono salvate nella **memoria locale del client**. Le altre informazioni devono essere sempre reperite dal database per non avere casi di concorrenza non gestita o informazioni non aggiornate.

1.3.1 Librerie Back-End

Anche per il back-end sono state usate **librerie Open-Source** per minimizzare i tempi ed i costi di sviluppo.

- **dotenv-java (v. 3.0.0)**: Carica variabili d'ambiente da un file .env.
- **jakarta.mail (v. 2.0.1)**: Utilizzata per inviare e ricevere email tramite il protocollo SMTP.
- **bcrypt (v. 0.9.0)**: Libreria per eseguire l'hashing delle password usando l'algoritmo bcrypt.
- **mariadb-java-client (v. 3.3.3)**: Driver JDBC per la connessione al database MariaDB.
- **quartz (v. 2.3.2)**: Libreria per lo scheduling di job in Java, utilizzata per eseguire compiti pianificati a intervalli regolari o in momenti specifici.
- **jansi (v. 2.4.0)**: Abilita il supporto ai colori ANSI nella console, utile per migliorare la leggibilità e l'estetica dell'output.
- **slf4j-simple (v. 1.7.36)**: Implementazione semplice di SLF4J (Simple Logging Facade for Java), utile per evitare warning e semplificare la gestione dei log.

1.4 Tempi di risposta

I tempi di risposta tra server e software **sono rapidi** e praticamente assenti. Tuttavia, con l'aumento dei dati all'interno del Database, si verifica un incremento dei relativi tempi di attesa.

È importante specificare che **i tempi di elaborazione del modulo LLM** (Large Language Model) **non sono prevedibili**. Poiché l'elaborazione dipende da diversi fattori, tra cui la complessità della richiesta e la quantità di dati da processare, i tempi possono variare notevolmente.

1.5 Linee guida per la documentazione delle interfacce

Per la realizzazione delle interfacce si è deciso di utilizzare **JavaFX insieme a stili CSS**, esso permette di realizzare interfacce grafiche immediate, essenziali ed efficaci, permettendo all'utilizzatore un facile utilizzo del software proprio per la sua **intuitività**.

1.6 Linee guida per la documentazione del DBMS

Per il sistema è stato scelto **MariaDB**, un DBMS derivato da MySQL. La scelta di MariaDB è motivata principalmente dalla sua **natura open source** e dall'**assenza di costi aggiuntivi**, garantendo così una soluzione economica e flessibile per la gestione del database.

Per l'interazione tra il sistema e il DBMS si è usato **MySQL-Connector-Java** (Java DataBase Connectivity – JDBC), con relativo driver per MariaDB.

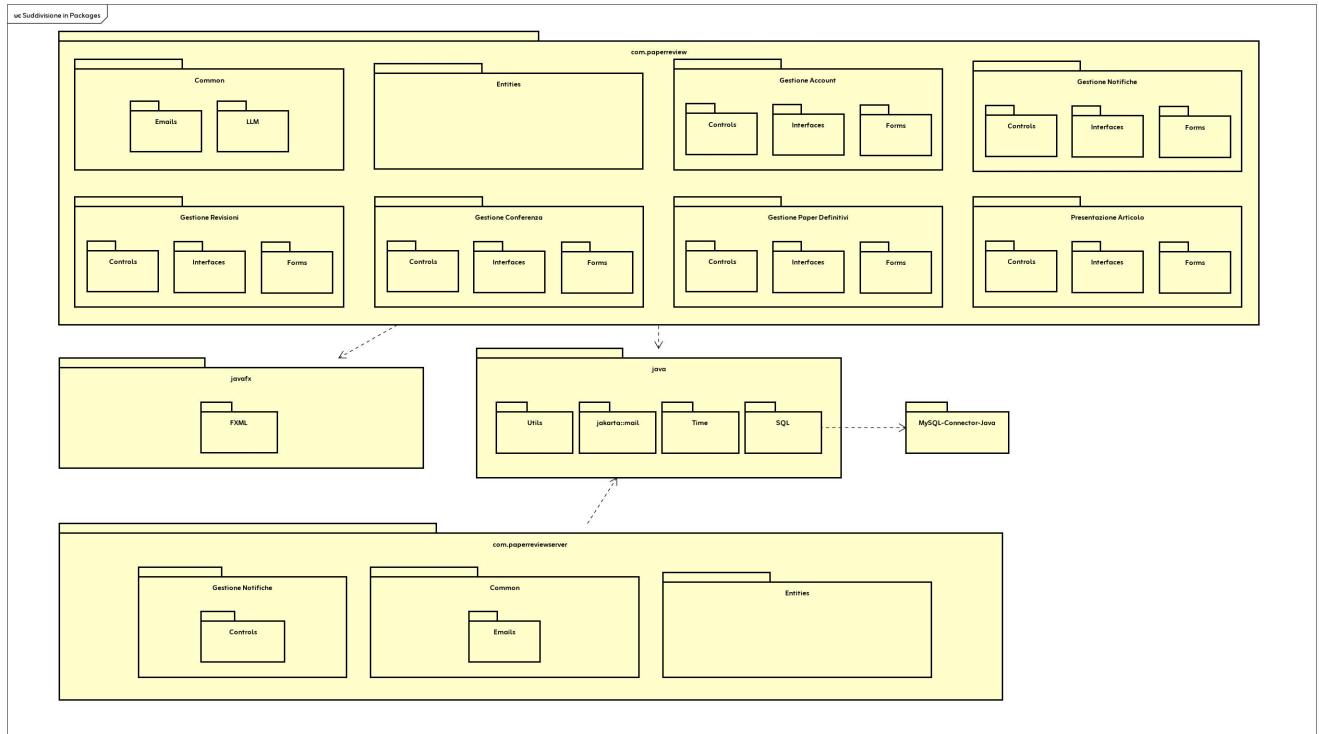
I **driver JDBC** permettono di mantenere la **persistenza dei dati** e di interagire con il database attraverso Java, offrendo così un'interfaccia standard per l'accesso e la gestione delle basi di dati.

1.7 Linee guida per la documentazione della LLM

Per l'analisi semantica dei documenti **PDF** e l'assegnazione automatica di **parole chiave** è stato usato un servizio esterno basato su **modelli linguistici (LLM)**. La comunicazione con tale servizio avviene tramite richieste **HTTP multipart**, costruite secondo lo standard **multipart/form-data**, consentendo l'invio congiunto del file **PDF** e della lista di **parole chiave**.

Questa scelta progettuale consente al sistema di rimanere **modulare e scalabile**, esternalizzando il carico computazionale dell'**analisi testuale** a un servizio specializzato.

2 Packages



2.1 com.paperreview

Il package dell'applicazione **client**.

2.1.1 Common

Questo package contiene logiche e **funzionalità condivise** e trasversali a tutto il progetto. Al suo interno si trovano componenti utilizzate in diversi punti dell'applicazione, tra cui:

- **Emails**: gestione della comunicazione via posta elettronica.
- API: contiene le classi wrapper che implementano le **API** di comunicazione con la **LLM**.
- **DBMSBoundary**: gestisce la connessione al database.
- **DotenvUtil**: consente di leggere le variabili d'ambiente (es. credenziali, rotte API), mantenendole fuori dal codice sorgente.
- **PasswordUtil**: genera password casuali e verifica la validità delle credenziali fornite dall'utente confrontandole con quelle archiviate.
- **UserContext**: mantiene l'istanza dell'utente autenticato, memorizzando i suoi dati persistenti durante la sessione.

2.1.2 Entities

Contiene le entità del dominio dell'applicazione, ovvero le classi che modellano i principali oggetti del sistema e che vengono popolati con i dati provenienti dal database.

2.1.3 Gestione Account

Vi risiedono i pacchetti necessari che devono essere installati sul sistema client.

Nello specifico sono presenti:

- Interfaces: Contiene i file FXML delle GUI
- Controls: Contiene le classi collegate alle specifiche GUI, gestendo così la logica dell'applicazione.

- Forms: La parte della GUI dove l'utente deve inserire dei dati.

Questo pacchetto permette agli utenti di effettuare il login, di registrarsi, di autenticarsi, di recuperare le credenziali, effettuare il logout e modificare la propria password.

2.1.4 Gestione Notifiche

Vi risiedono i pacchetti necessari che devono essere installati sul sistema client.

Nello specifico sono presenti:

- Interfaces: Contiene i file FXML delle GUI
- Controls: Contiene le classi collegate alle specifiche GUI, gestendo così la logica dell'applicazione.
- Forms: La parte della GUI dove l'utente deve inserire dei dati.

Questo pacchetto contiene i sistemi non automatici che permettono agli utenti di ricevere notifiche ed inviti. Permette anche di partecipare ad una conferenza mediante un codice invito. Inoltre, si occupa della gestione dei sottosistemi per l'archiviazione e la visualizzazione delle notifiche e degli inviti.

2.1.5 Gestione Revisioni

Vi risiedono i pacchetti necessari che devono essere installati sul sistema client.

Nello specifico sono presenti:

- Interfaces: Contiene i file FXML delle GUI
- Controls: Contiene le classi collegate alle specifiche GUI, gestendo così la logica dell'applicazione.
- Forms: La parte della GUI dove l'utente deve inserire dei dati.

Questo pacchetto contiene i sistemi che permettono ai revisori ed ai sotto-revisori di presentare e modificare revisioni fino alla relativa scadenza e di segnalare un possibile plagio.

2.1.6 Gestione Conferenza

Vi risiedono i pacchetti necessari che devono essere installati sul sistema client.

Nello specifico sono presenti:

- Interfaces: Contiene i file FXML delle GUI
- Controls: Contiene le classi collegate alle specifiche GUI, gestendo così la logica dell'applicazione.
- Forms: La parte della GUI dove l'utente deve inserire dei dati.

Questo pacchetto contiene i sistemi che permettono ai chair di creare le conferenze e gestirne ogni aspetto: dalle impostazioni agli inviti dei collaboratori.

2.1.7 Gestione Paper Definitivi

Vi risiedono i pacchetti necessari che devono essere installati sul sistema client.

Nello specifico sono presenti:

- Interfaces: Contiene i file FXML delle GUI
- Controls: Contiene le classi collegate alle specifiche GUI, gestendo così la logica dell'applicazione.
- Forms: La parte della GUI dove l'utente deve inserire dei dati.

Questo pacchetto contiene i sistemi che permettono ad un editore di scaricare i paper, richiedere delle correzioni agli autori e pubblicare i proceeding. Permettono all'autore di pubblicare le versioni finale dei propri paper.

2.1.8 Presentazione Articolo

Vi risiedono i pacchetti necessari che devono essere installati sul sistema client.

Nello specifico sono presenti:

- Interfaces: Contiene i file FXML delle GUI
- Controls: Contiene le classi collegate alle specifiche GUI, gestendo così la logica dell'applicazione.
- Forms: La parte della GUI dove l'utente deve inserire dei dati.

Questo pacchetto contiene i sottosistemi che permettono agli autori di presentare e modificare la propria sottomissione. Selezionando, volendo automaticamente, le parole chiave dei paper.

2.2 Java

Include tutte le librerie e i package standard di Java. In particolare, sono presenti:

- **Utils**: Contiene le strutture dati utilizzate nel sistema.
- **SQL**: Gestisce le comunicazioni con i DBMS, in particolare tramite la classe `DBMSBoundary` per connettersi ed eseguire query.
- **Time**: API per la gestione di date e orari.
- **Jakarta.mail**: API per la gestione delle comunicazioni SMTP, utilizzata per l'invio delle email, ad esempio nel recupero delle password.

2.3 MYSQL-Connector-Java(JDBC)

Il JDBC (Java Database Connectivity) contiene i driver che consentono alle applicazioni Java di interagire con i database.

In questo caso, stiamo utilizzando il driver JDBC di **MariaDB**, che permette di connettersi a un database MariaDB e di eseguire operazioni come query e aggiornamenti.

2.4 JavaFX

JavaFX è un framework per creare interfacce utente in Java, ed in questo package sono presenti i file FXML per definire la struttura grafica dell'applicazione.

2.5 com.paperreviewServer

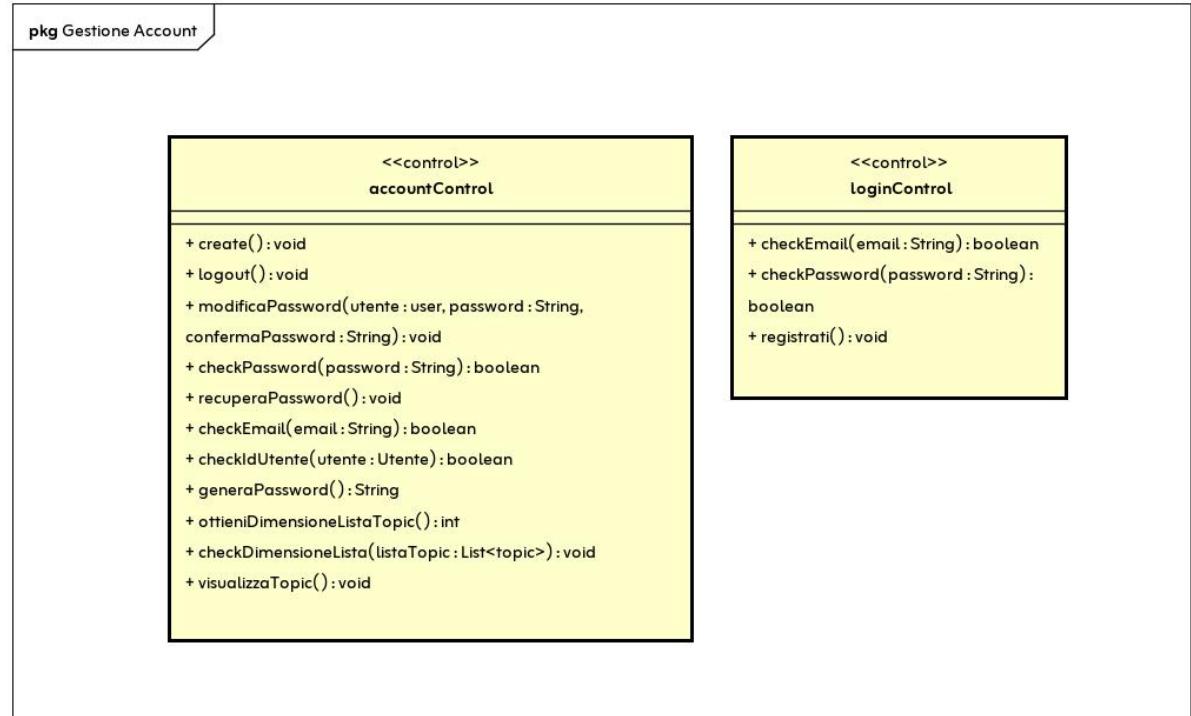
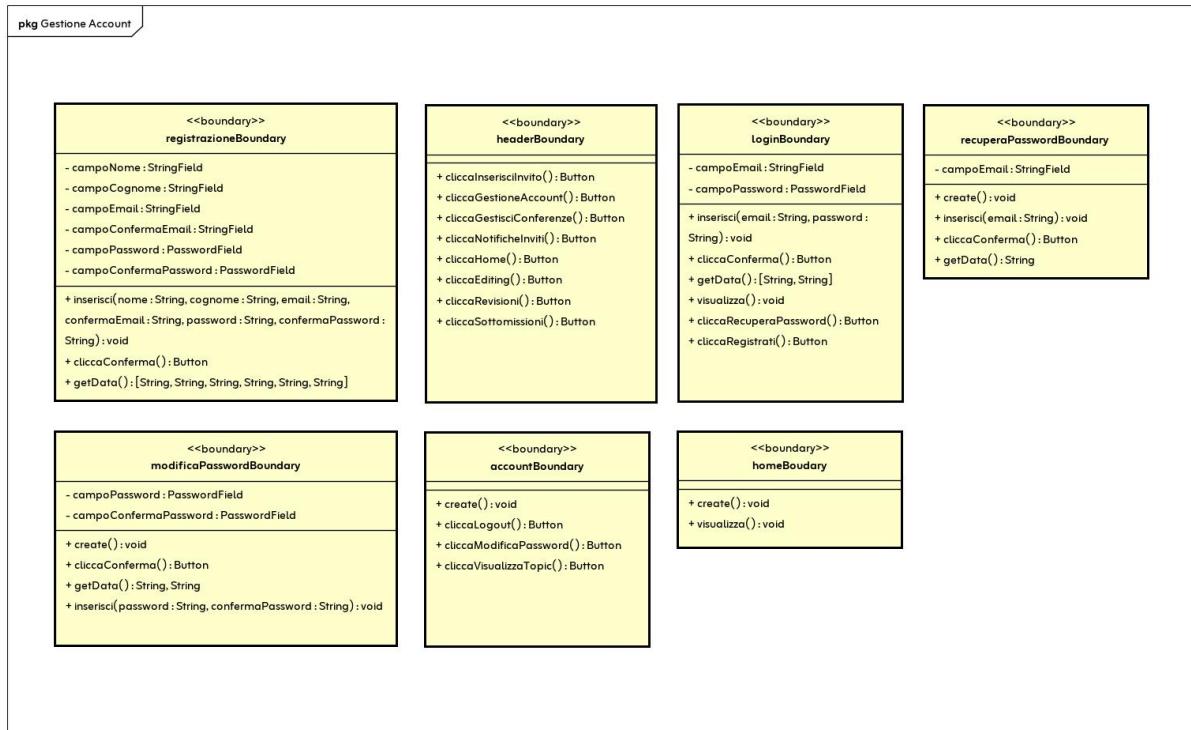
Package dell'applicazione **server**.

2.5.1 Gestione Notifiche

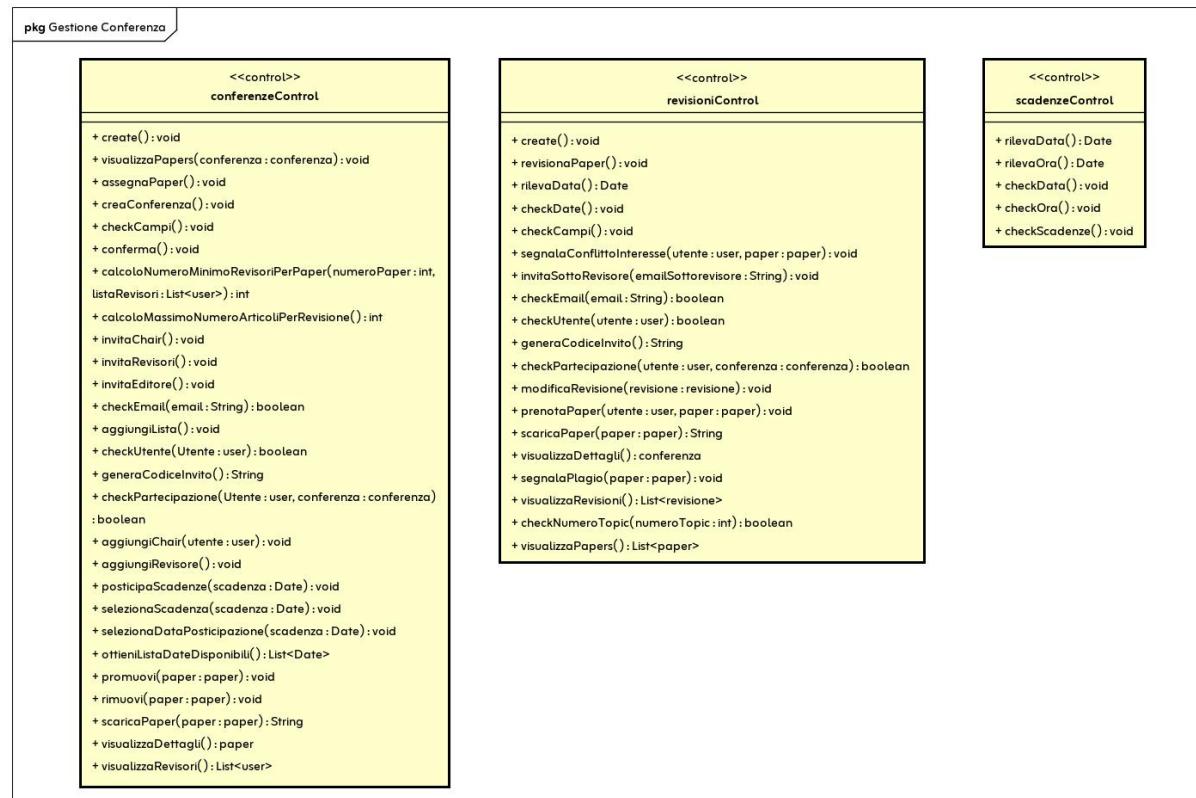
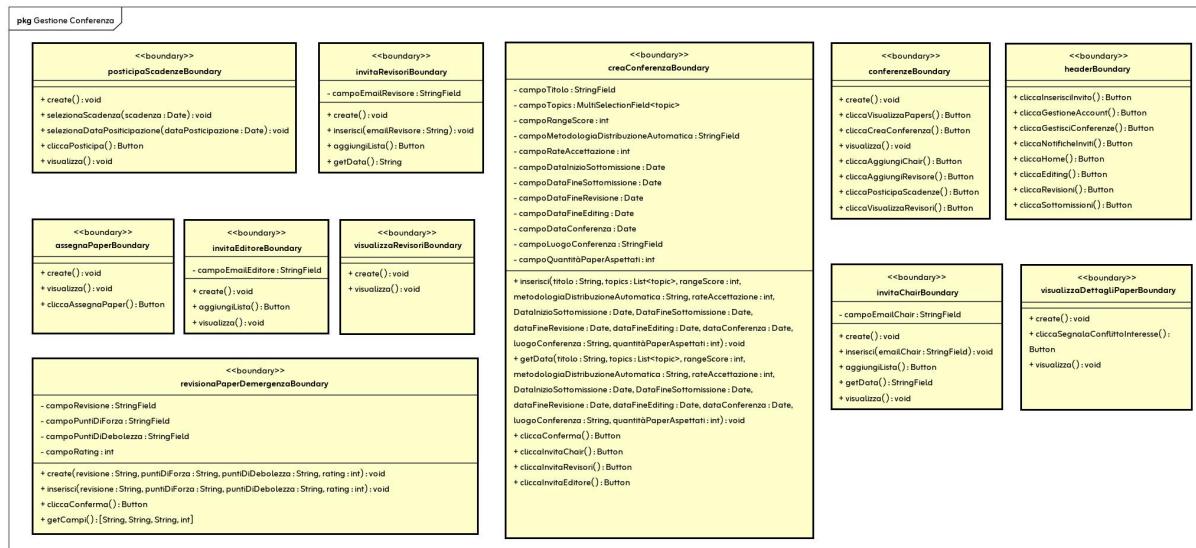
Contiene le funzionalità automatiche di invio notifiche del sistema, ad esempio per notificare di un'imminente scadenza.

3 Object Design UML

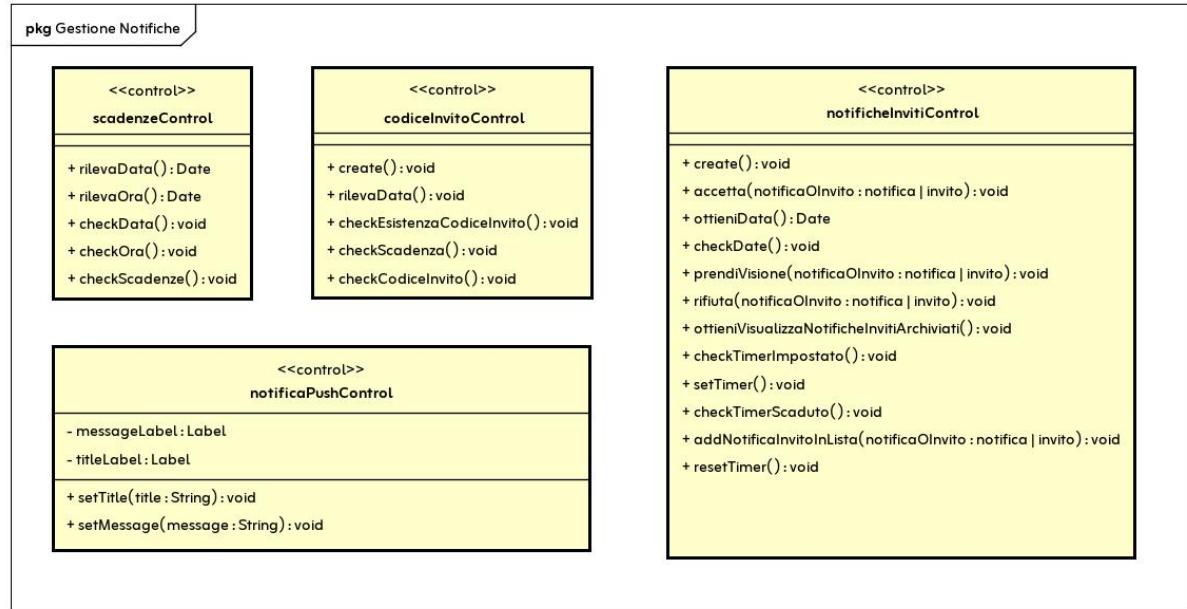
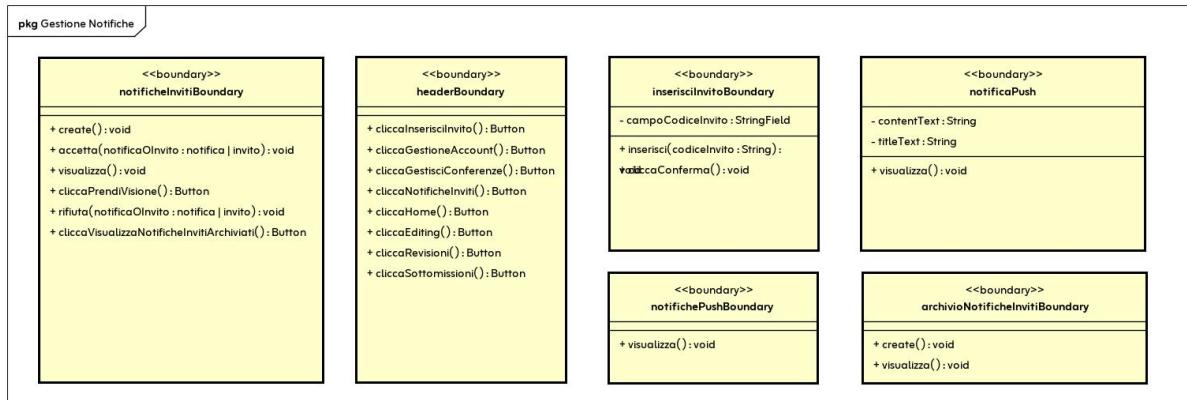
3.1 Gestione Account



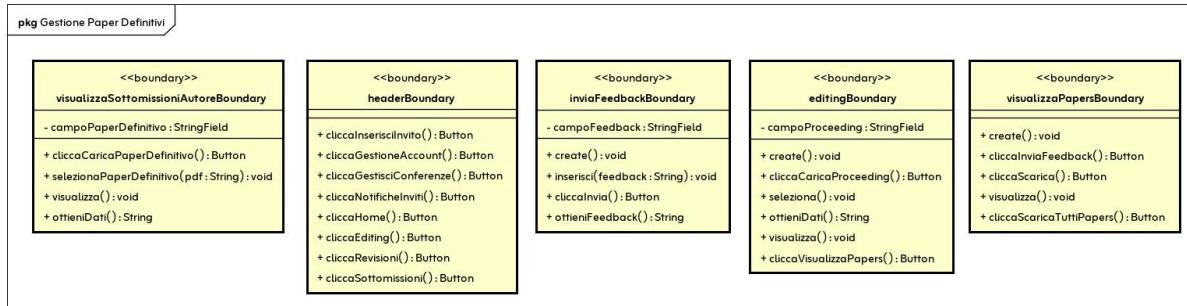
3.2 Gestione Conferenza

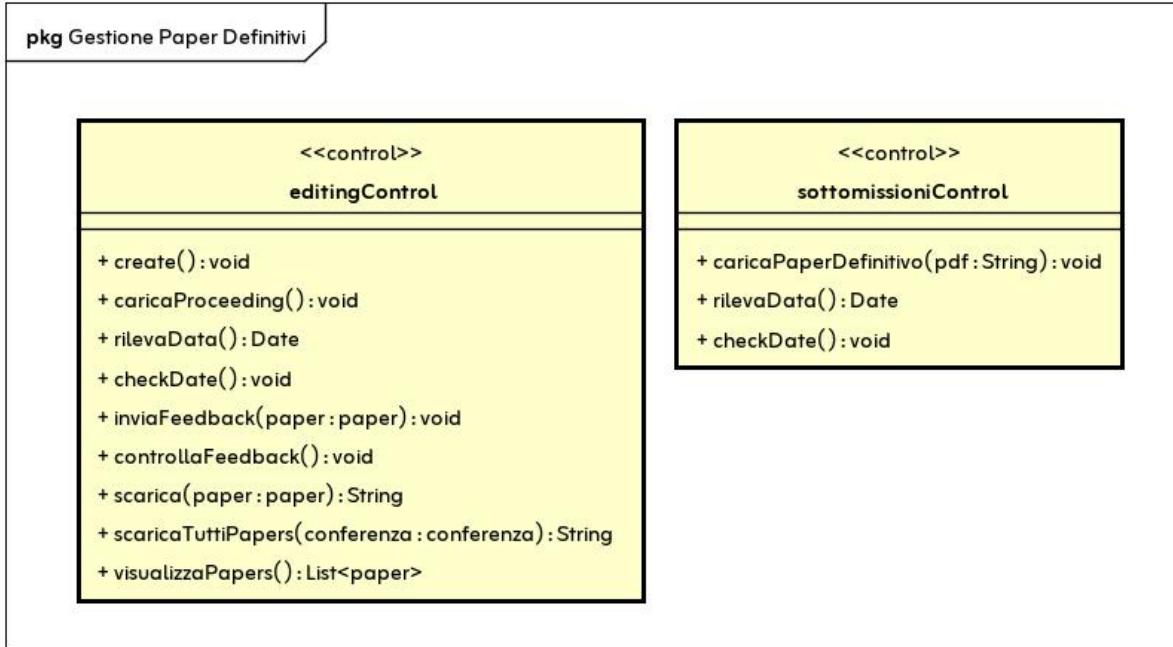


3.3 Gestione Notifiche

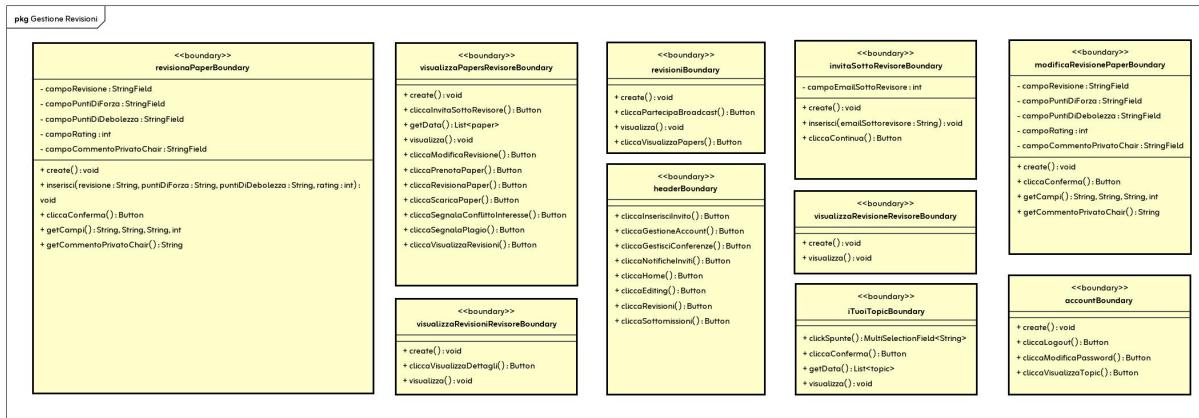


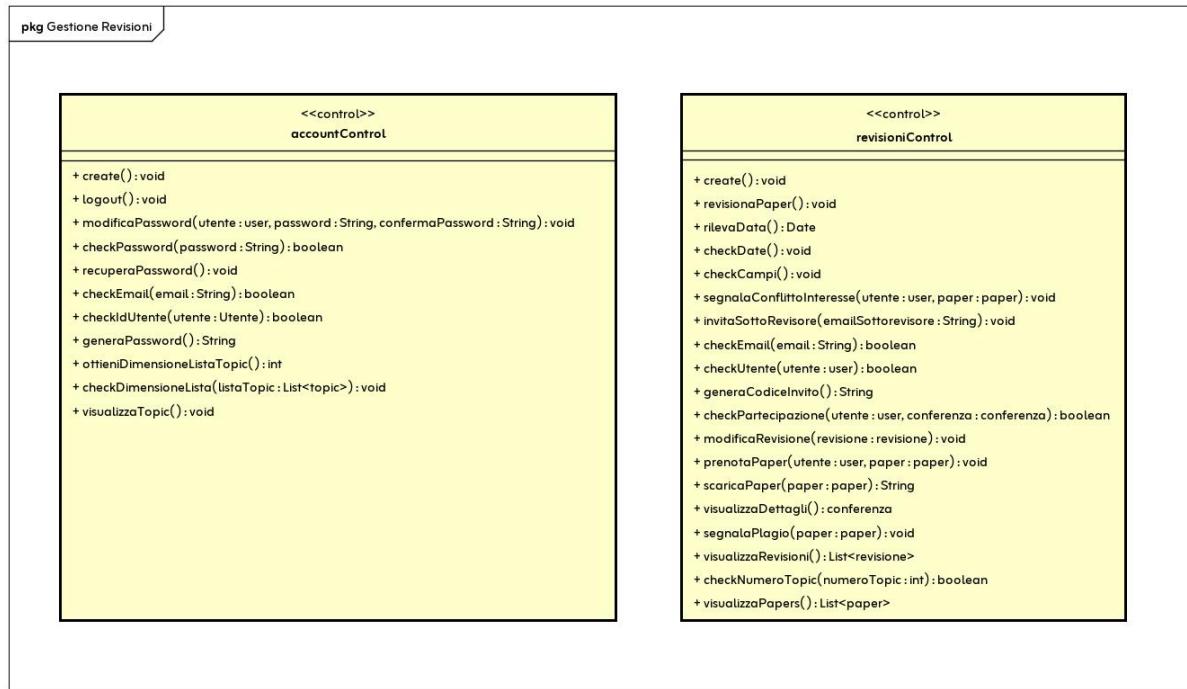
3.4 Gestione Paper Definitivi



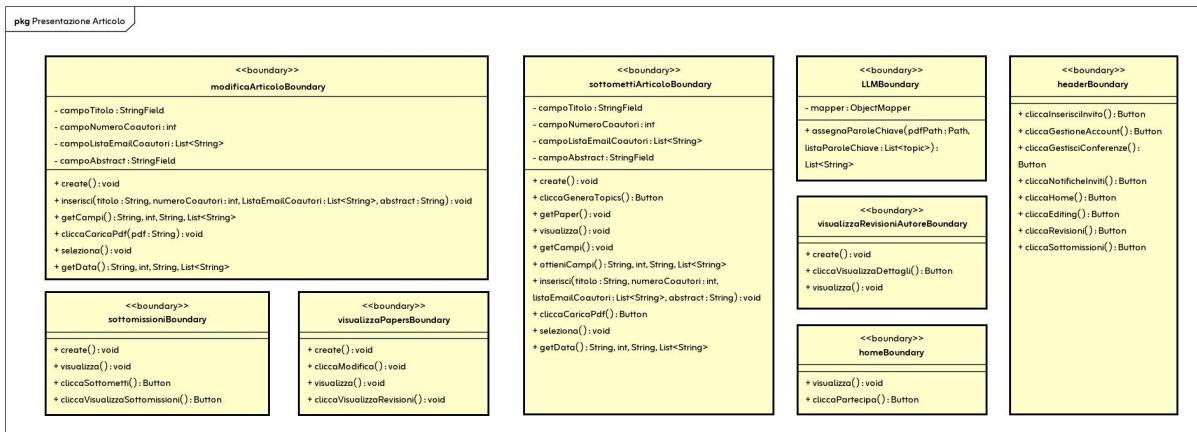


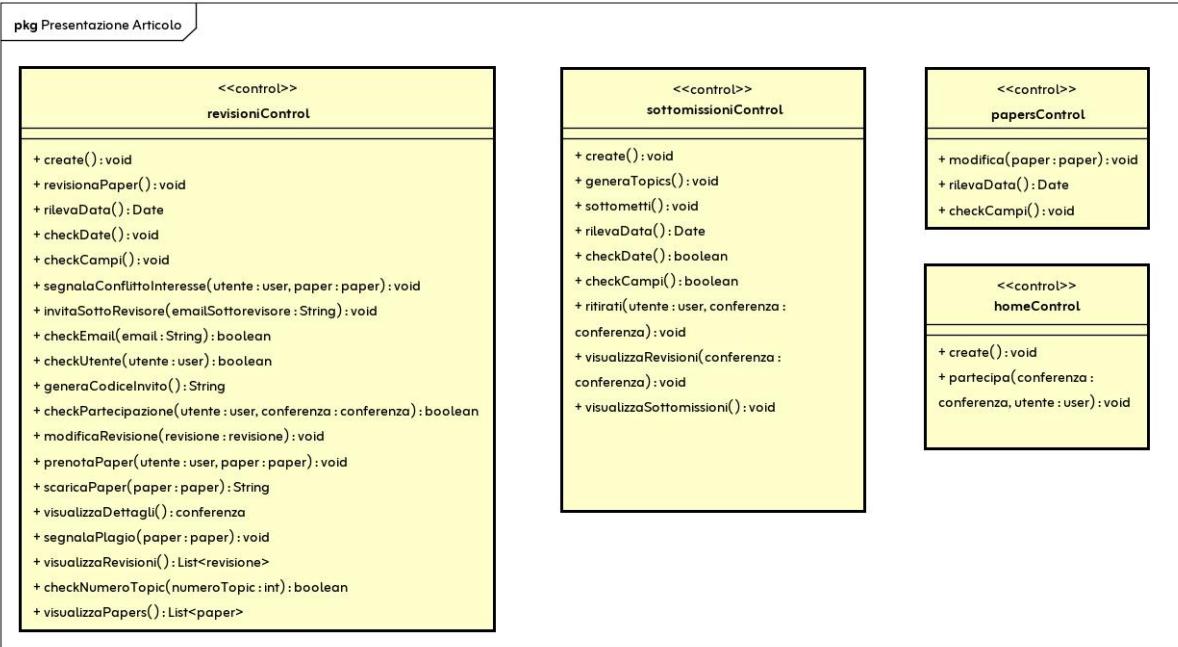
3.5 Gestione Revisioni



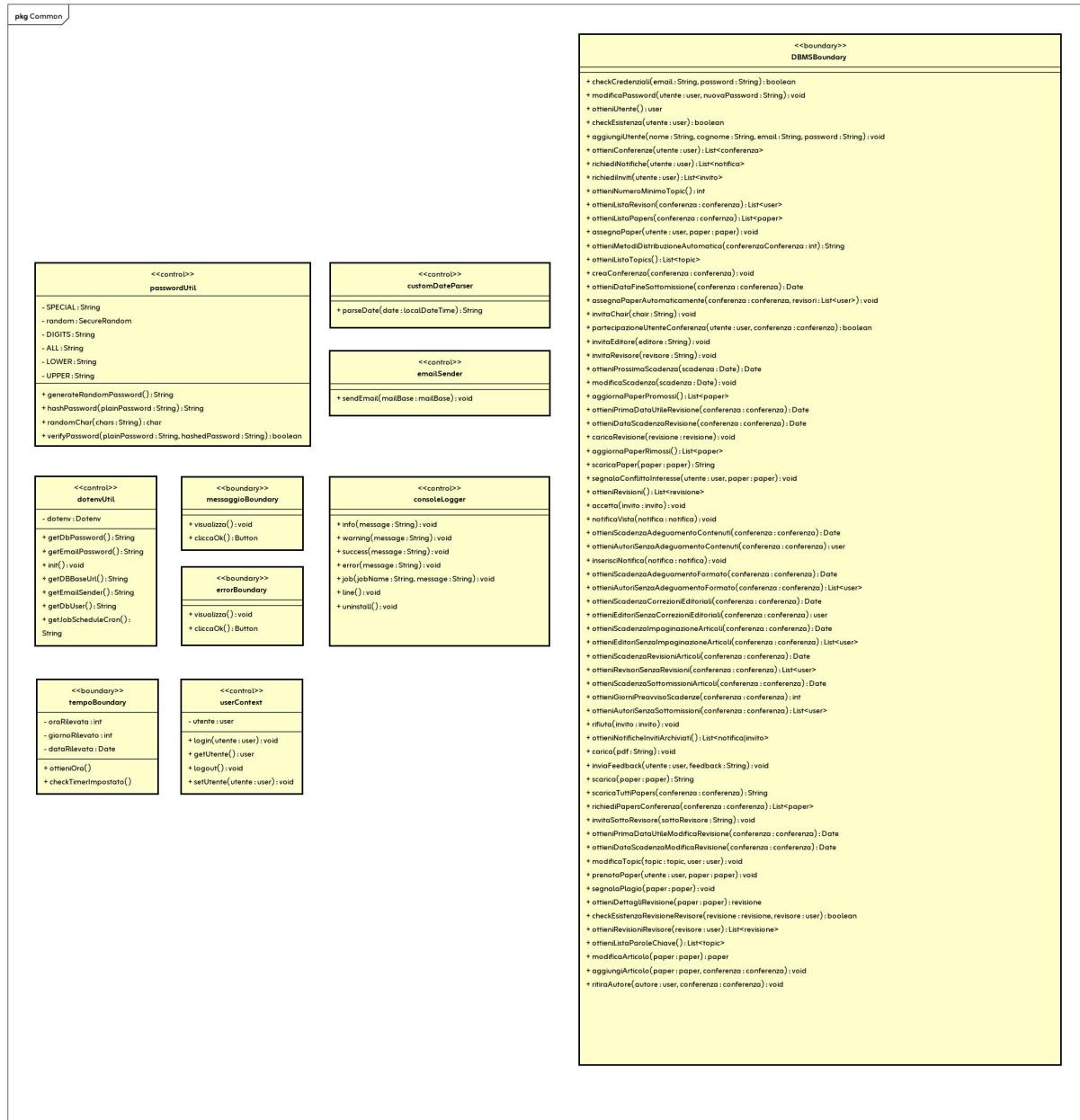


3.6 Presentazione Articolo

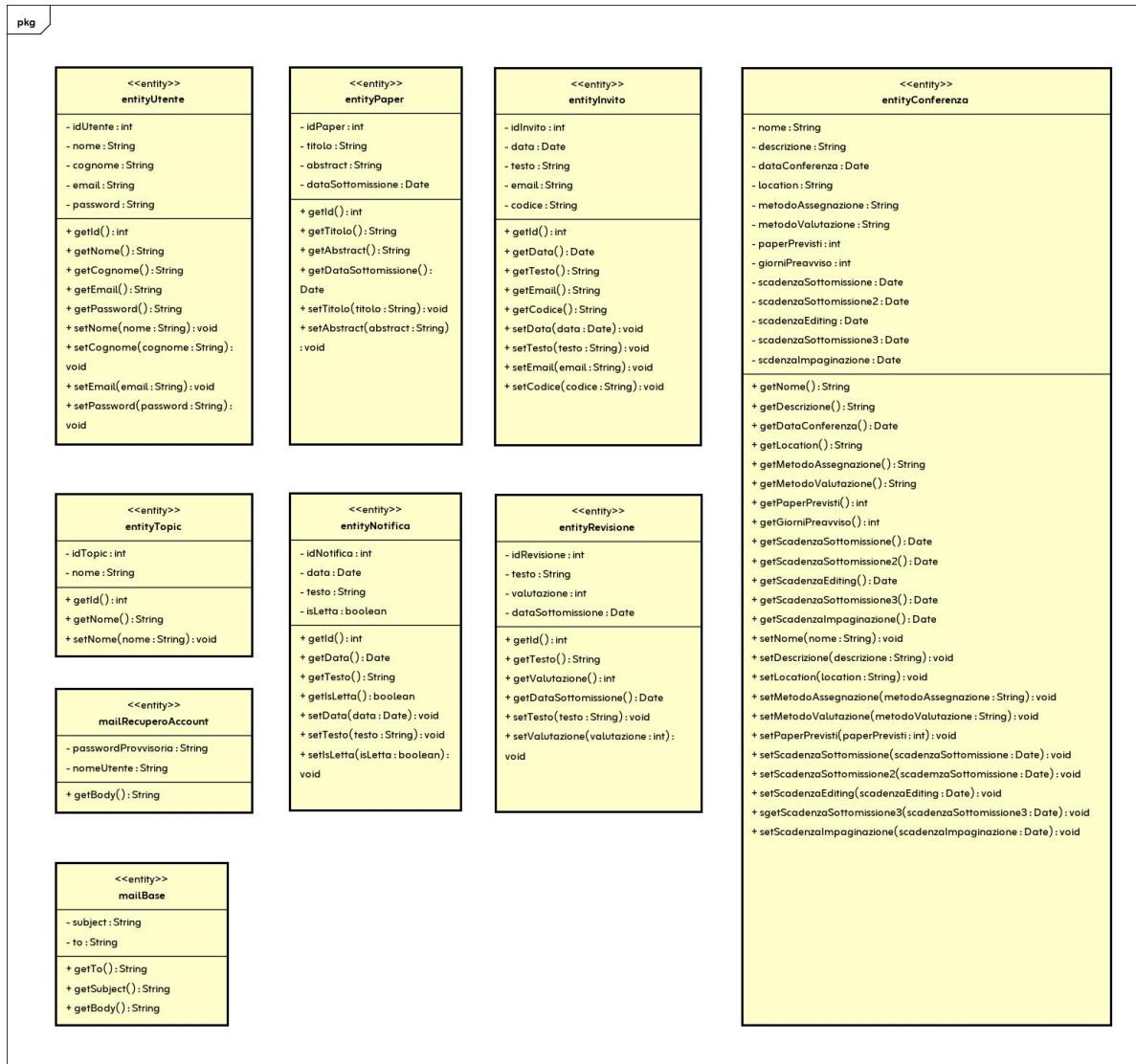




3.7 Common



3.8 Entity



4 Copyright e diritto d'autore

La presente **documentazione** è protetta dalle **leggi sul diritto d'autore**. Nessuna parte di questo documento può essere riprodotta, distribuita o trasmessa in alcuna forma o con alcun mezzo, elettronico o meccanico, inclusa la fotocopia, la registrazione o altri sistemi di memorizzazione o recupero di informazioni, senza il **previo consenso scritto degli autori**.



PaperReview © 2025

Leonardo Giovanni Coazza

Diego Corona

Duccio Gaetano

Daniela D'Urso