UNIVERSITÀ DEGLI STUDI DI PALERMO

FACOLTÀ DI INGEGNERIA

LAUREA TRIENNALE IN INGEGNERIA INFORMATICA

PAPERPILOT

AUTORI

Mirko Stefano Cirivello

Maria Chiara Ciuffoli

Flavio Giarrusso

Alessia Pizzuto

DOCENTE

Dott.sa Seidita Valeria

Introduzione

Object Design Trade-offs

Per la realizzazione del software **PaperPilot** è stata utilizzata un'architettura di tipo **Repository**, che garantisce il disaccoppiamento dei sottosistemi, i quali possono comunicare solo con il sottosistema di Storage, ma non tra loro. A loro volta i sottosistemi sono composti da un'interfaccia utente che comunica solo con il controllore sottostante, che contiene la logica del programma e che gestisce le richieste da rivolgere al sottosistema DBMSDaemon, che gestisce le comunicazioni al nodo di Storage. L'interfaccia utente, il controllore e il DBMSDaemon di ogni sottosistema risiedono sullo stesso nodo, mentre il sottosistema di Storage risiede su un nodo diverso.

Linee guida per la documentazione delle interfacce

Per quanto riguarda le interfacce grafiche si è scelto di usare il pacchetto di librerie JavaFX, che ci ha permesso di realizzare le interfacce grafiche attraverso documenti di marcatura FXML. JavaFX è un pacchetto di librerie component-based, ciò permette quindi di avere una grande riusabilità del codice, oltre a una maggiore leggibilità in fase di scrittura. Più in particolare, per la realizzazione delle interfacce grafiche del Sistema è stato usato JavaFX Scene Builder, un tool per la creazione di documenti di marcatura FXML che sfrutta un sistema drag-and-drop

Linee guida per la documentazione dei DBMS

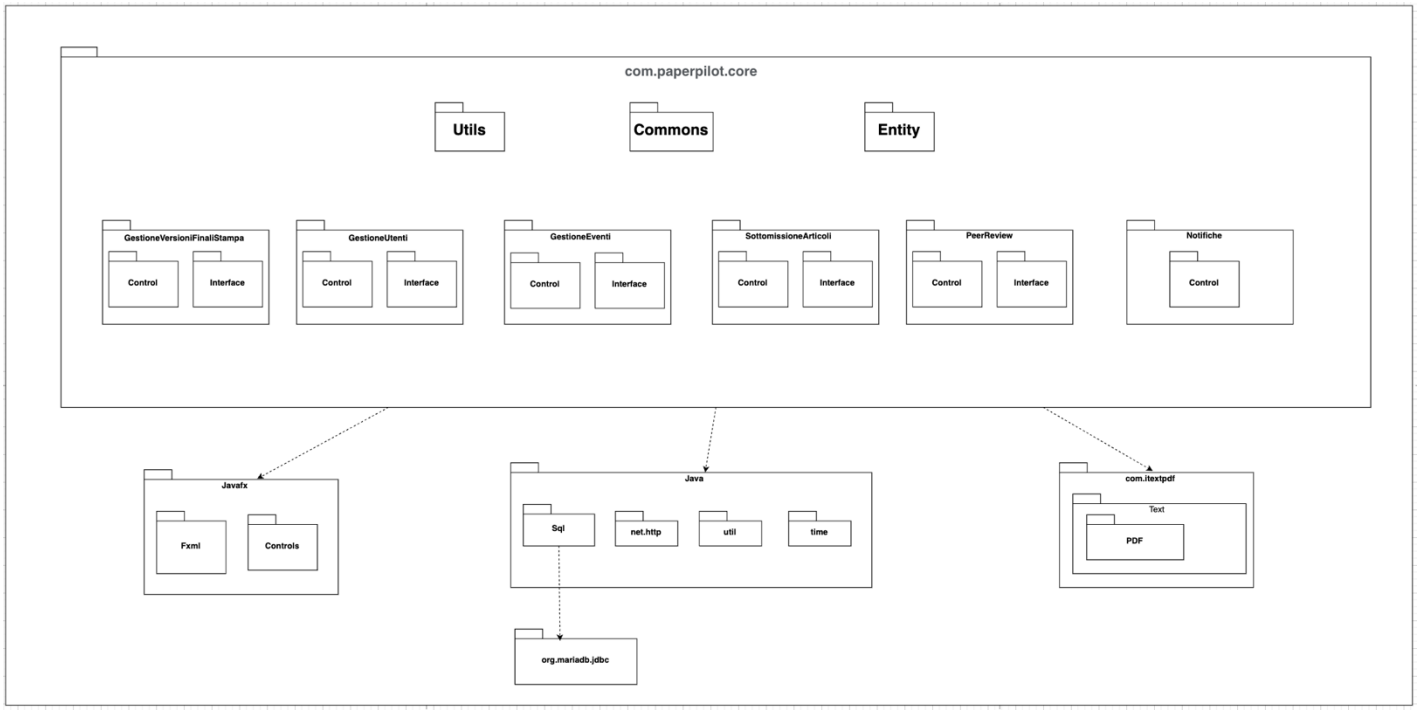
Per la gestione della persistenza dei dati all'interno del sistema "PaperPilot", si è scelto di adottare un modello relazionale. Questa decisione è motivata dalla necessità di gestire in modo efficiente e strutturato le relazioni complesse tra le diverse entità del sistema, quali articoli, autori, revisori e recensioni.

In particolare, il sistema di gestione di database relazionale (RDBMS) selezionato è **MariaDB**. MariaDB, essendo un fork di MySQL, offre robustezza, affidabilità e prestazioni elevate, che sono cruciali per un'applicazione che gestirà un volume crescente di dati e transazioni.

Per la comunicazione tra il sistema "PaperPilot" e il DBMS, si farà uso di **JDBC (Java DataBase Connectivity)**. JDBC rappresenta un framework standard in Java per l'accesso ai database. La sua scelta garantisce un'interfaccia uniforme e indipendente dal tipo specifico di DBMS utilizzato, permettendo al sistema di interagire con il database in modo flessibile. Questo approccio disaccoppia la logica applicativa dai dettagli implementativi del database, facilitando eventuali migrazioni future o l'integrazione con diverse fonti dati.

Linee guida per la documentazione dell'invio Mail

Per quanto riguarda le Mail, si è scelto di usare un indirizzo di posta elettronica fornito da Google ed un servizio offerto dallo stesso per l'invio, chiamato Apps Script. Apps Script permette l'interfacciamento con la casella di posta attraverso una richiesta HTTPS GET. In questa richiesta GET vanno inclusi alcuni parametri necessari per l'invio della mail, in particolare è necessario l'indirizzo di destinazione ed il corpo del messaggio.



Packages

com.ogc.paperpilotcode

Il package principale dell'applicazione

Utils

Contiene classi Wrapper per API, il pacchetto è specializzato per il sistema e quindi poco riutilizzabile in altri contesti.

Commons

Contiene classi comuni ai sottosistemi vedi SDD].

Entity

Contiene le classi che modellano le entità di cui è necessaria la rappresentazione nel sistema.

- **Gestione Utenti:** Questo sottosistema è dedicato alla gestione completa degli account utente all'interno del sistema PaperPilot. Contiene le classi e la logica per gestire l'autenticazione, consentire la registrazione di nuovi utenti, supportare il recupero delle

credenziali smarrite e permettere la modifica delle password esistenti. Interagisce con il Livello di Controllo per la ricezione delle richieste e con il Livello di Persistenza per l'accesso e la manipolazione dei dati utente.

- **Gestione Eventi:** Questo modulo è incaricato di tutte le operazioni relative alla creazione e configurazione degli eventi o delle conferenze nel sistema. Contiene la logica per gestire i membri del comitato di programma (PC), inclusa l'importazione di elenchi e l'invio degli inviti. Le sue classi comunicano con il Livello di Controllo per la logica operativa e possono utilizzare il Servizio Notifiche per le comunicazioni automatiche.
- **Sottomissione Articoli :** Questo sottosistema gestisce l'intero flusso di invio dei documenti da parte degli autori. Contiene la logica applicativa per la gestione delle scadenze di invio, la ricezione effettiva dei documenti e l'invio delle notifiche di avvenuta ricezione agli autori. Le sue classi dipendono dal Livello di Controllo per l'esecuzione delle operazioni e dal Servizio Notifiche per le comunicazioni.
- **Peer Review:** Questo sottosistema è responsabile della gestione dell'intero processo di revisione degli articoli. Contiene le classi e la logica per l'assegnazione dei revisori agli articoli, inclusa la gestione del processo di invio delle revisioni stesse da parte dei revisori. Le sue classi operano sotto il coordinamento del Livello di Controllo e si affidano al Servizio Notifiche per l'invio di comunicazioni relative alle assegnazioni e alle scadenze di revisione.
- **Gestione Versioni Finali:** Questo modulo è responsabile della raccolta delle versioni definitive degli articoli che sono stati accettati per la pubblicazione. Contiene la logica per raccogliere tali versioni e per il processo di invio finale all'editore. Le sue classi operano sotto l'egida del Livello di Controllo, utilizzano il Servizio Notifiche per l'invio di promemoria.
- **Notifiche :** Questo modulo è responsabile della raccolta e invio delle notifiche.

Com.org.mariadb.jdbc

Contiene i driver per le comunicazioni con il DBMS MariaDB utilizzato per la realizzazione del sistema

javafx

Il package è utilizzato per la realizzazione e la gestione delle interfacce
grafiche utente.

- **controls**
  Definisce i controlli Ul, i grafici e le skin che sono disponibili per il toolkit JavaFX Ul.
- **fxml**
  Permette di caricare i file FXML, e gestisce il binding tra le variabili della classe associata e le rispettive componenti grafiche (tramite id ed EventHandler)

java

Contiene i package e le librerie standard di Java

- **sql**
  Il package è utilizzato per gestire le comunicazioni con i DBMS, viene utilizzato nella classe DBMSDaemon per la creazione di connessioni e l'esecuzione di query.
- **net.http**
  API per le comunicazioni http, utilizzata per inviare richieste a una
- Google Application responsabile dell'invio di e-mail
- **utils**
  Contiene le strutture dati utilizzate per la realizzazione del sistema
- **time**
  API per le date, il tempo, gli istanti e le durate

com.itextpdf

Libreria per la creazione e la manipolazione di pdf, utilizzata per la generazione dei pdf di ricevuta dei colli da parte dell'impiegato

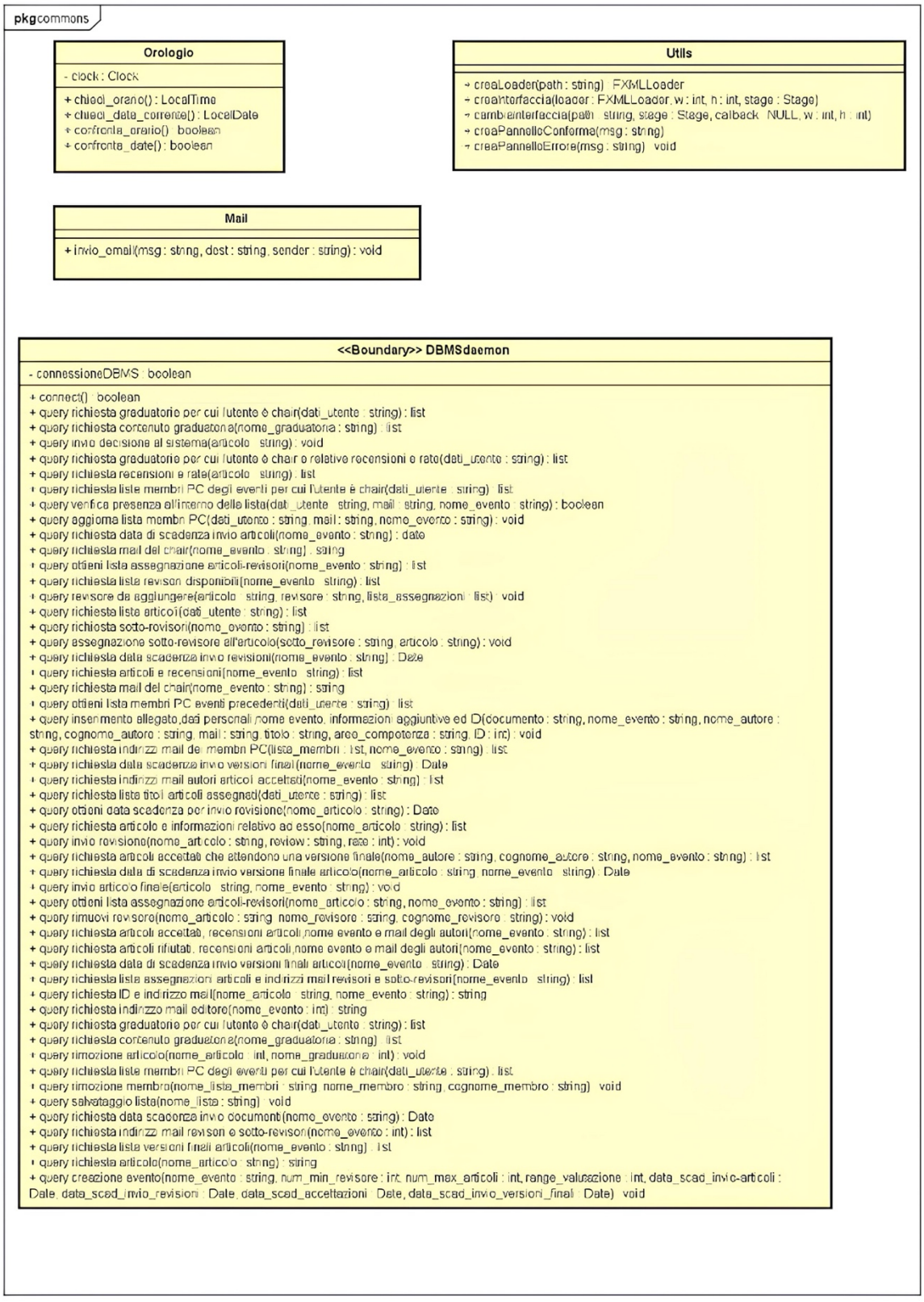
- **text**
  Package per il testo contenuto nel pdf
- **pdf**
  Package per la gestione del documento pdf

Object design UML

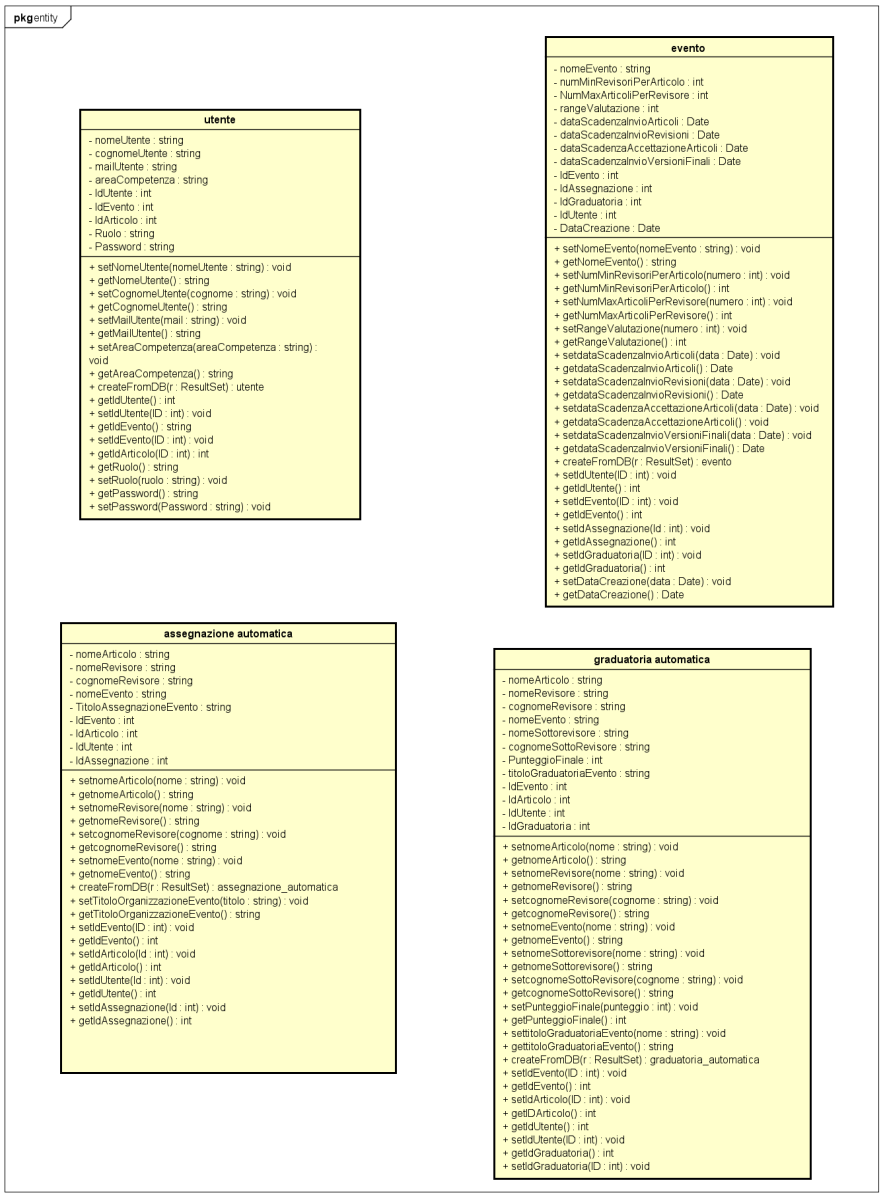
Utils

Nel progetto sviluppato, non essendo stata prevista la fase di creazione di codice, si è optato per non includere le "utils", ritenendo che la loro menzione avrebbe costituito un errore.

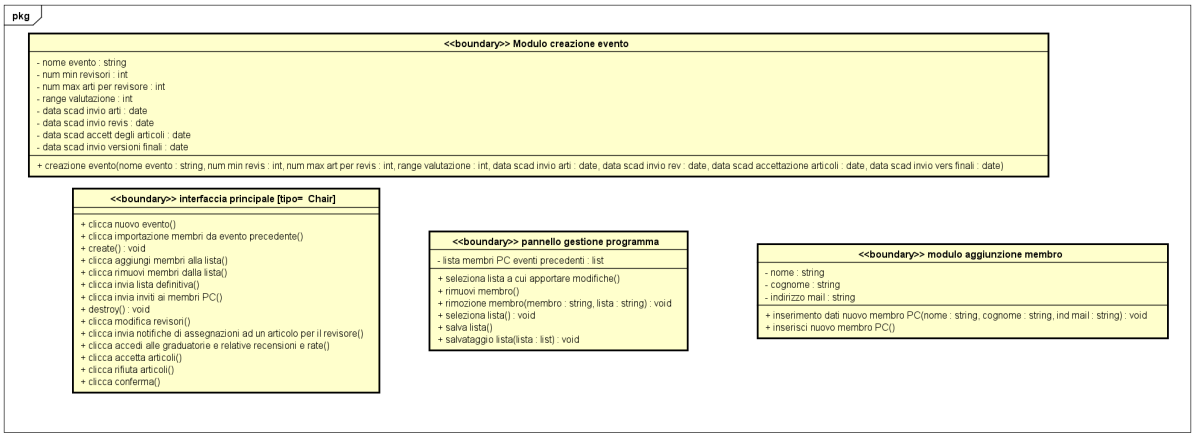
Commons


pkgcommons
Orologio
- clock : Clock
+ chiedi_orario() : LocalTime
+ chiedi_data_corrente() : LocalDate
+ confronta_orario() : boolean
+ confronta_date() : boolean
Utils
+ creaLoader(path : string) : FXMLLoader
+ creaInterfaccia(loader : FXMLLoader, w : int, h : int, stage : Stage)
+ cambiaInterfaccia(path : string, stage : Stage, callback : NULL, w : int, h : int)
+ creaPannelloConferma(msg : string)
+ creaPannelloErrore(msg : string) : void
Mail
+ invio_email(msg : string, dest : string, sender : string) : void
<<Boundary>> DBMSdaemon
- connessioneDBMS : boolean
+ connect() : boolean
+ query richiesta graduatorie per cui l'utente è chair(dati_utente : string) : list
+ query richiesta contenuto graduatoria(nome_graduatoria : string) : list
+ query invio decisione al sistema(articolo : string) : void
+ query richiesta graduatorie per cui l'utente è chair e relative recensioni e rate(dati_utente : string) : list
+ query richiesta recensioni e rate(articolo : string) : list
+ query richiesta liste membri PC degli eventi per cui l'utente è chair(dati_utente : string) : list
+ query verifica presenza all'interno delle lista(dati_utente : string, mail : string, nome_evento : string) : boolean
+ query aggiorna lista membri PC(dati_utente : string, mail : string, nome_evento : string) : void
+ query richiesta data di scadenza invio articoli(nome_evento : string) : date
+ query richiesta mail del chair(nome_evento : string) : string
+ query ottieni lista assegnazione articoli-revisori(nome_evento : string) : list
+ query richiesta lista revisori disponibili(nome_evento : string) : list
+ query revisore da aggiungere(articolo : string, revisore : string, lista_assegnazioni : list) : void
+ query richiesta liste articoli(dati_utente : string) : list
+ query richiesta sotto-revisori(nome_evento : string) : list
+ query assegnazione sotto-revisore all'articolo(sotto_revisore : string, articolo : string) : void
+ query richiesta data scadenza invio revisioni(nome_evento : string) : Date
+ query richiesta articoli e recensioni(nome_evento : string) : list
+ query richiesta mail del chair(nome_evento : string) : string
+ query ottieni lista membri PC eventi precedenti(dati_utente : string) : list
+ query inserimento allegato,dati personali,nome evento, informazioni aggiuntive ed ID(documento : string, nome_evento : string, nome_autore : string, cognome_autore : string, mail : string, titolo : string, area_competenza : string, ID : int) : void
+ query richiesta indirizzi mail dei membri PC(lista_membri : list, nome_evento : string) : list
+ query richiesta data scadenza invio versioni finali(nome_evento : string) : Date
+ query richiesta indirizzi mail autori articoli accettati(nome_evento : string) : list
+ query richiesta liste titoli articoli assegnati(dati_utente : string) : list
+ query ottieni data scadenza per invio revisione(nome_articolo : string) : Date
+ query richiesta articolo e informazioni relativo ad esso(nome_articolo : string) : list
+ query invio revisione(nome_articolo : string, review : string, rate : int) : void
+ query richiesta articoli accettati che attendono una versione finale(nome_autore : string, cognome_autore : string, nome_evento : string) : list
+ query richiesta data di scadenza invio versione finale articolo(nome_articolo : string, nome_evento : string) : Date
+ query invio articolo finale(articolo : string, nome_evento : string) : void
+ query ottieni lista assegnazione articoli-revisori(nome_articolo : string, nome_evento : string) : list
+ query rimuovi revisore(nome_articolo : string, nome_revisore : string, cognome_revisore : string) : void
+ query richiesta articoli accettati, recensioni articoli,nome evento e mail degli autori(nome_evento : string) : list
+ query richiesta articoli rifiutati, recensioni articoli,nome evento e mail degli autori(nome_evento : string) : list
+ query richiesta data di scadenza invio versioni finali articoli(nome_evento : string) : Date
+ query richiesta lista assegnazioni articoli e indirizzi mail revisori e sotto-revisori(nome_evento : string) : list
+ query richiesta ID e indirizzo mail(nome_articolo : string, nome_evento : string) : string
+ query richiesta indirizzo mail editore(nome_evento : int) : string
+ query richiesta graduatorie per cui l'utente è chair(dati_utente : string) : list
+ query richiesta contenuto graduatoria(nome_graduatoria : string) : list
+ query rimozione articolo(nome_articolo : int, nome_graduatoria : int) : void
+ query richiesta liste membri PC degli eventi per cui l'utente è chair(dati_utente : string) : list
+ query rimozione membro(nome_lista_membri : string, nome_membro : string, cognome_membro : string) : void
+ query salvataggio lista(nome_lista : string) : void
+ query richiesta data scadenza invio documenti(nome_evento : string) : Date
+ query richiesta indirizzi mail revisori e sotto-revisori(nome_evento : int) : list
+ query richiesta lista versioni finali articoli(nome_evento : string) : list
+ query richiesta articolo(nome_articolo : string) : string
+ query creazione evento(nome_evento : string, num_min_revisore : int, num_max_articoli : int, range_valutazione : int, data_scad_invio-articoli : Date, data_scad_invio_revisioni : Date, data_scad_accettazioni : Date, data_scad_invio_versioni_finali : Date) : void

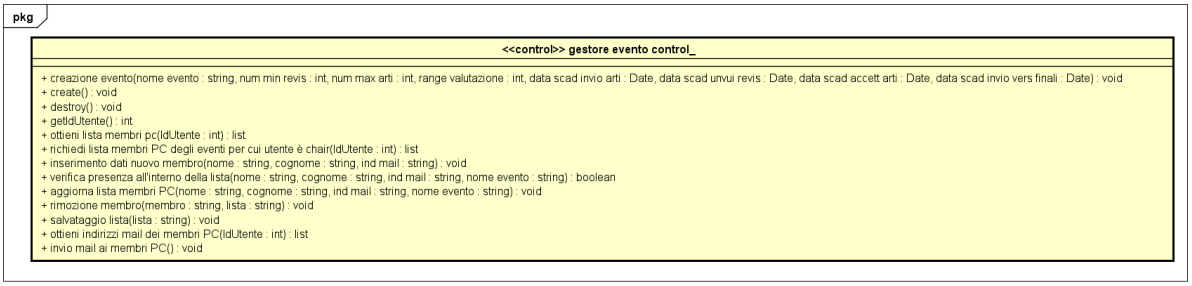
Entity


pkgentity
utente
- nomeUtente : string
- cognomeUtente : string
- mailUtente : string
- areaCompetenza : string
- IdUtente : int
- IdEvento : int
- IdArticolo : int
- Ruolo : string
- Password : string
+ setNomeUtente(nomeUtente : string) : void
+ getNomeUtente() : string
+ setCognomeUtente(cognome : string) : void
+ getCognomeUtente() : string
+ setMailUtente(mail : string) : void
+ getMailUtente() : string
+ setAreaCompetenza(areaCompetenza : string) : void
+ getAreaCompetenza() : string
+ createFromDB(r : ResultSet) : utente
+ getIdUtente() : int
+ setIdUtente(ID : int) : void
+ getIdEvento() : string
+ setIdEvento(ID : int) : void
+ getIdArticolo(ID : int) : int
+ getRuolo() : string
+ setRuolo(ruolo : string) : void
+ getPassword() : string
+ setPassword(Password : string) : void
evento
- nomeEvento : string
- numMinRevisoriPerArticolo : int
- NumMaxArticoliPerRevisore : int
- rangeValutazione : int
- dataScadenzaInvioArticoli : Date
- dataScadenzaInvioRevisioni : Date
- dataScadenzaAccettazioneArticoli : Date
- dataScadenzaInvioVersioniFinali : Date
- IdEvento : int
- IdAssegnazione : int
- IdGraduatoria : int
- IdUtente : int
- DataCreazione : Date
+ setNomeEvento(nomeEvento : string) : void
+ getNomeEvento() : string
+ setNumMinRevisoriPerArticolo(numero : int) : void
+ getNumMinRevisoriPerArticolo() : int
+ setNumMaxArticoliPerRevisore(numero : int) : void
+ getNumMaxArticoliPerRevisore() : int
+ setRangeValutazione(numero : int) : void
+ getRangeValutazione() : int
+ setdataScadenzaInvioArticoli(data : Date) : void
+ getdataScadenzaInvioArticoli() : Date
+ setdataScadenzaInvioRevisioni(data : Date) : void
+ getdataScadenzaInvioRevisioni() : Date
+ setdataScadenzaAccettazioneArticoli(data : Date) : void
+ getdataScadenzaAccettazioneArticoli() : void
+ setdataScadenzaInvioVersioniFinali(data : Date) : void
+ getdataScadenzaInvioVersioniFinali() : Date
+ createFromDB(r : ResultSet) : evento
+ setIdUtente(ID : int) : void
+ getIdUtente() : int
+ setIdEvento(ID : int) : void
+ getIdEvento() : int
+ setIdAssegnazione(Id : int) : void
+ getIdAssegnazione() : int
+ setIdGraduatoria(ID : int) : void
+ getIdGraduatoria() : int
+ setDataCreazione(data : Date) : void
+ getDataCreazione() : Date
assegnazione automatica
- nomeArticolo : string
- nomeRevisore : string
- cognomeRevisore : string
- nomeEvento : string
- TitoloAssegnazioneEvento : string
- IdEvento : int
- IdArticolo : int
- IdUtente : int
- IdAssegnazione : int
+ setnomeArticolo(nome : string) : void
+ getnomeArticolo() : string
+ setnomeRevisore(nome : string) : void
+ getnomeRevisore() : string
+ setcognomeRevisore(cognome : string) : void
+ getcognomeRevisore() : string
+ setnomeEvento(nome : string) : void
+ getnomeEvento() : string
+ createFromDB(r : ResultSet) : assegnazione_automatica
+ setTitoloOrganizzazioneEvento(titolo : string) : void
+ getTitoloOrganizzazioneEvento() : string
+ setIdEvento(ID : int) : void
+ getIdEvento() : int
+ setIdArticolo(Id : int) : void
+ getIdArticolo() : int
+ setIdUtente(Id : int) : void
+ getIdUtente() : int
+ setIdAssegnazione(Id : int) : void
+ getIdAssegnazione() : int
graduatoria automatica
- nomeArticolo : string
- nomeRevisore : string
- cognomeRevisore : string
- nomeEvento : string
- nomeSottorevisore : string
- cognomeSottoRevisore : string
- PunteggioFinale : int
- titoloGraduatoriaEvento : string
- IdEvento : int
- IdArticolo : int
- IdUtente : int
- IdGraduatoria : int
+ setnomeArticolo(nome : string) : void
+ getnomeArticolo() : string
+ setnomeRevisore(nome : string) : void
+ getnomeRevisore() : string
+ setcognomeRevisore(cognome : string) : void
+ getcognomeRevisore() : string
+ setnomeEvento(nome : string) : void
+ getnomeEvento() : string
+ setnomeSottorevisore(nome : string) : void
+ getnomeSottorevisore() : string
+ setcognomeSottoRevisore(cognome : string) : void
+ getcognomeSottoRevisore() : string
+ setPunteggioFinale(punteggio : int) : void
+ getPunteggioFinale() : int
+ settitoloGraduatoriaEvento(nome : string) : void
+ gettitoloGraduatoriaEvento() : string
+ createFromDB(r : ResultSet) : graduatoria_automatica
+ setIdEvento(ID : int) : void
+ getIdEvento() : int
+ setIdArticolo(ID : int) : void
+ getIDArticolo() : int
+ getIdUtente() : int
+ setIdUtente(ID : int) : void
+ getIdGraduatoria() : int
+ setIdGraduatoria(ID : int) : void

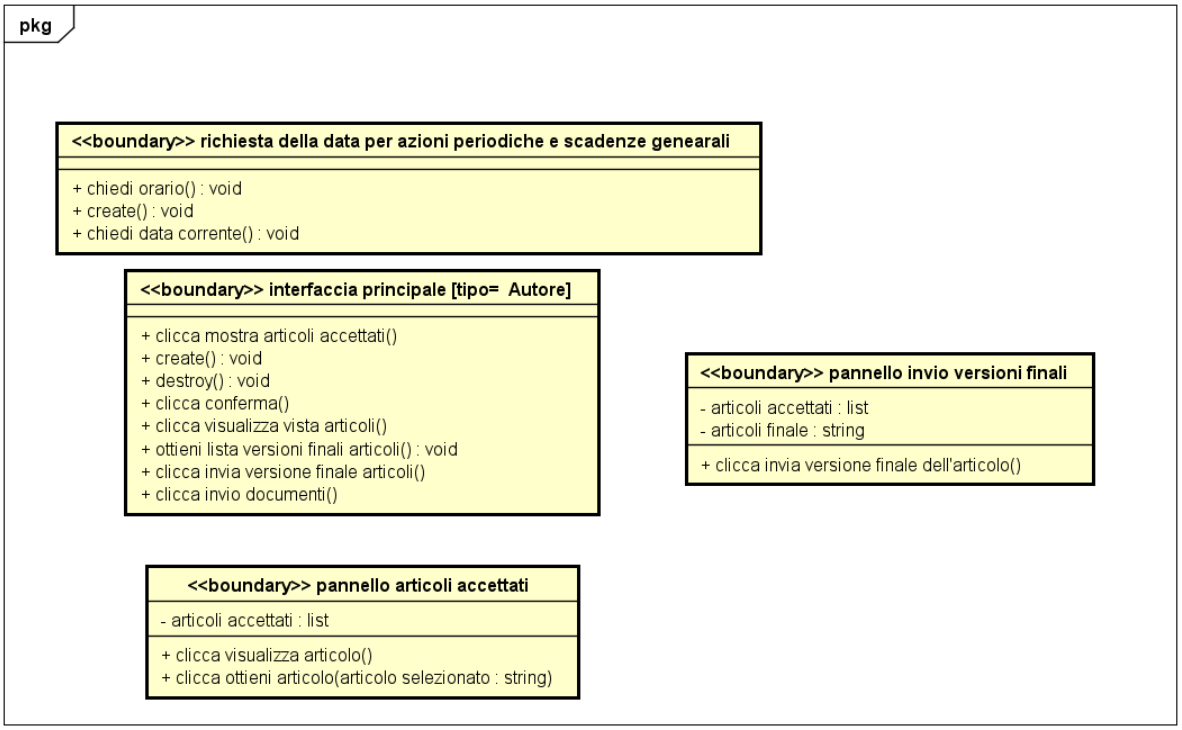
Interfacce Gestione Organizzazione evento



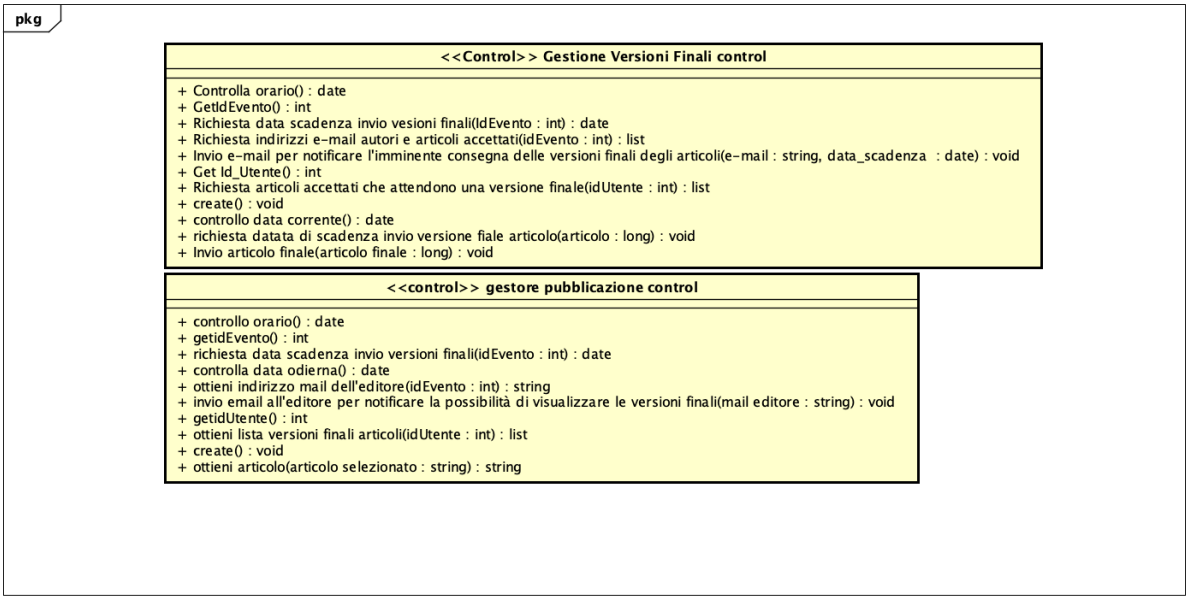
Control Gestione Organizzazione Evento:



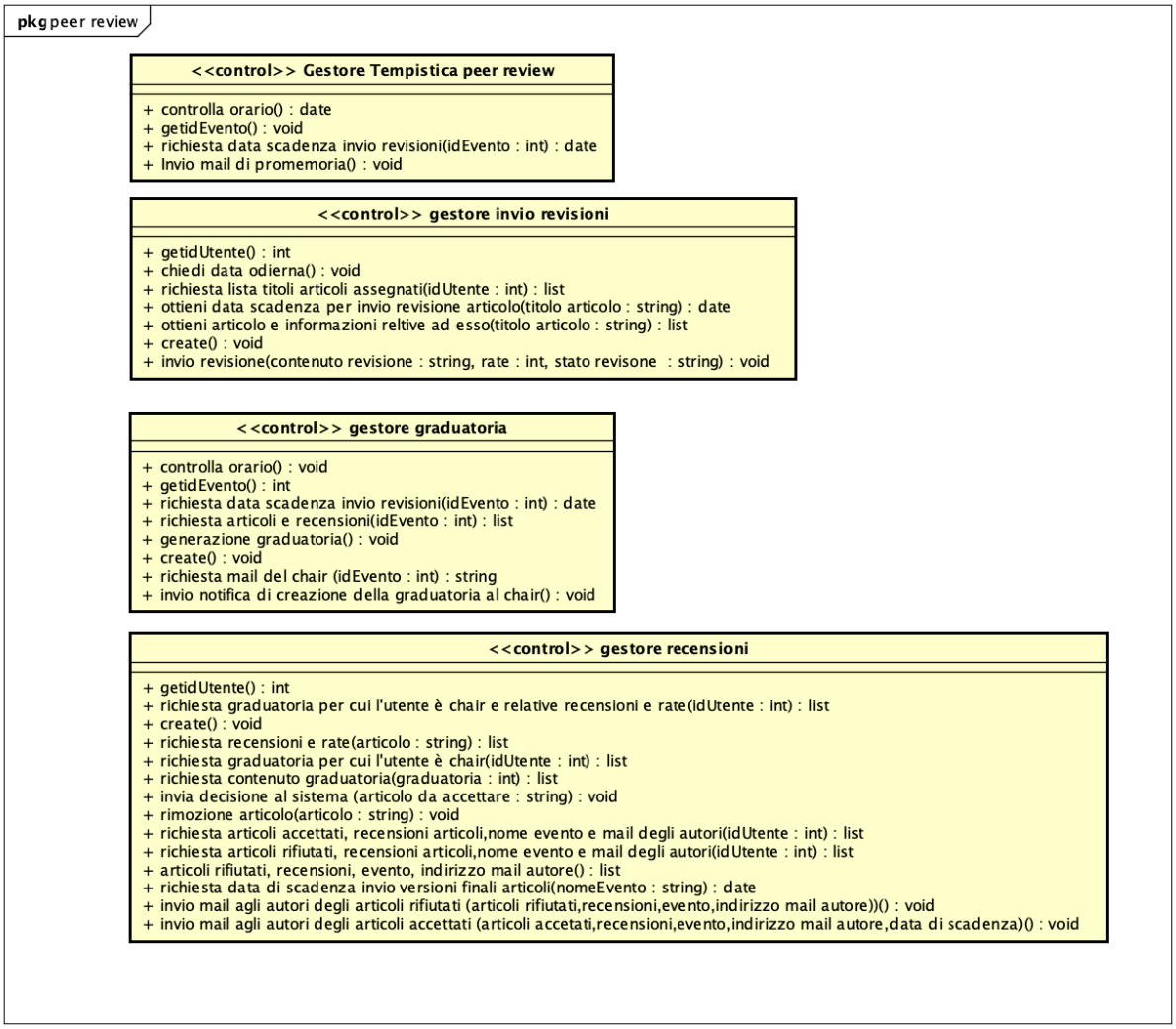
Interfacce Gestione versione finale e stampa



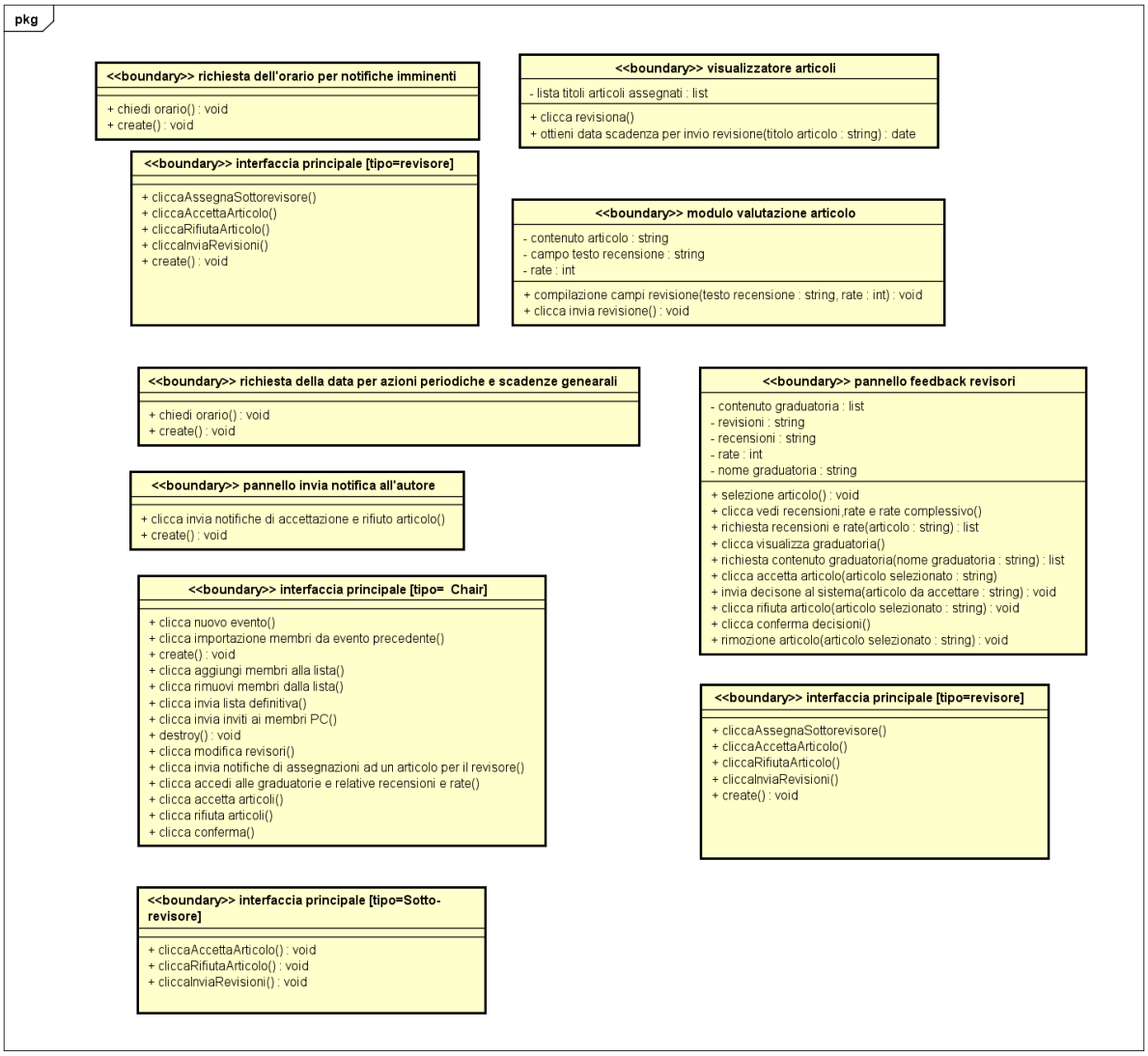
Control Gestione Versione Finale e Stampa



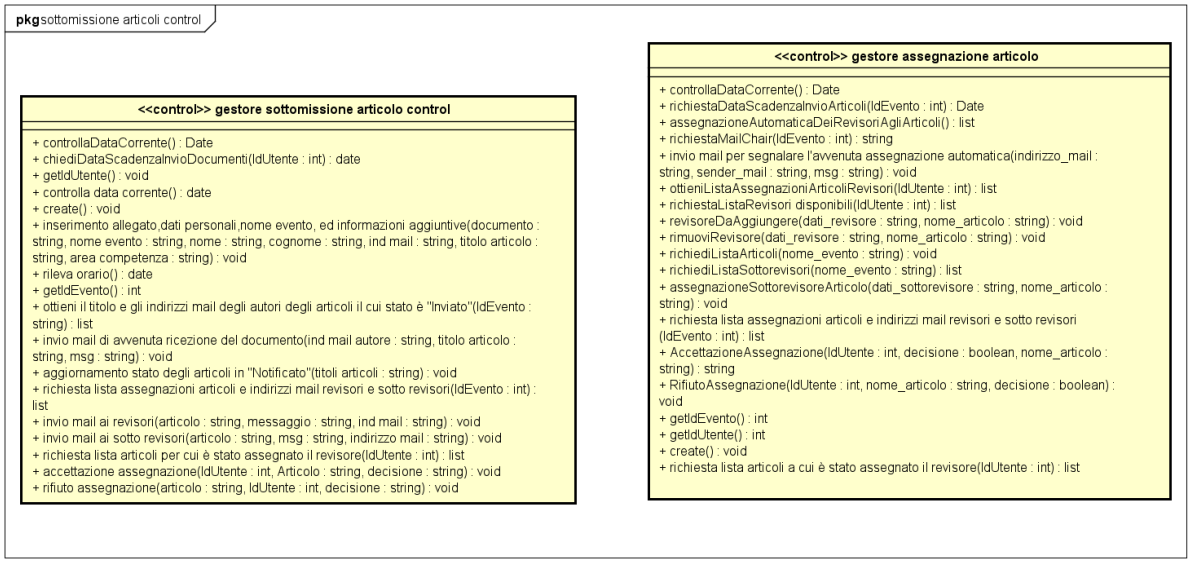
Control Peer review



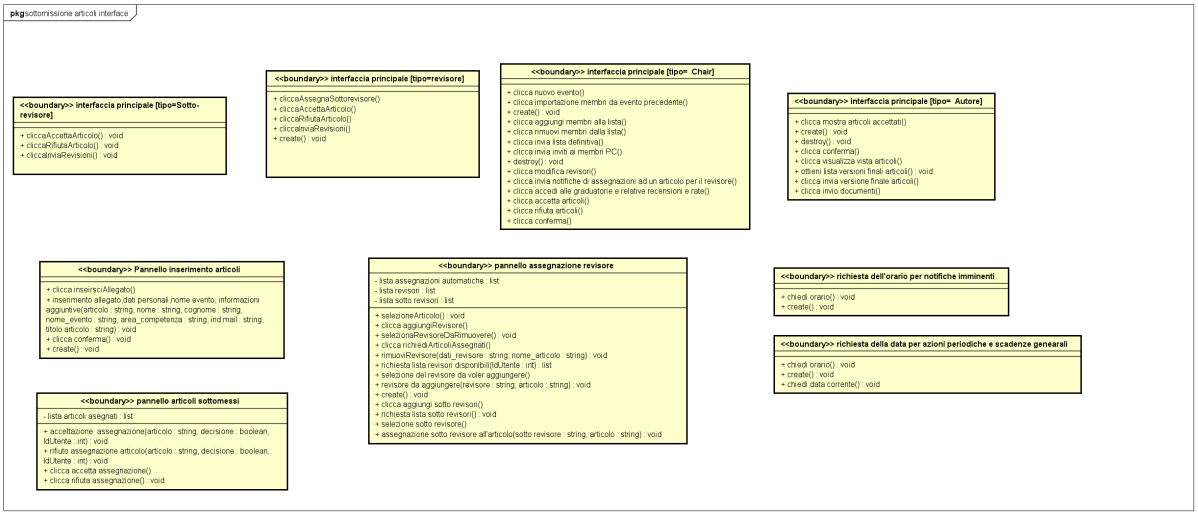
Interface Peer review



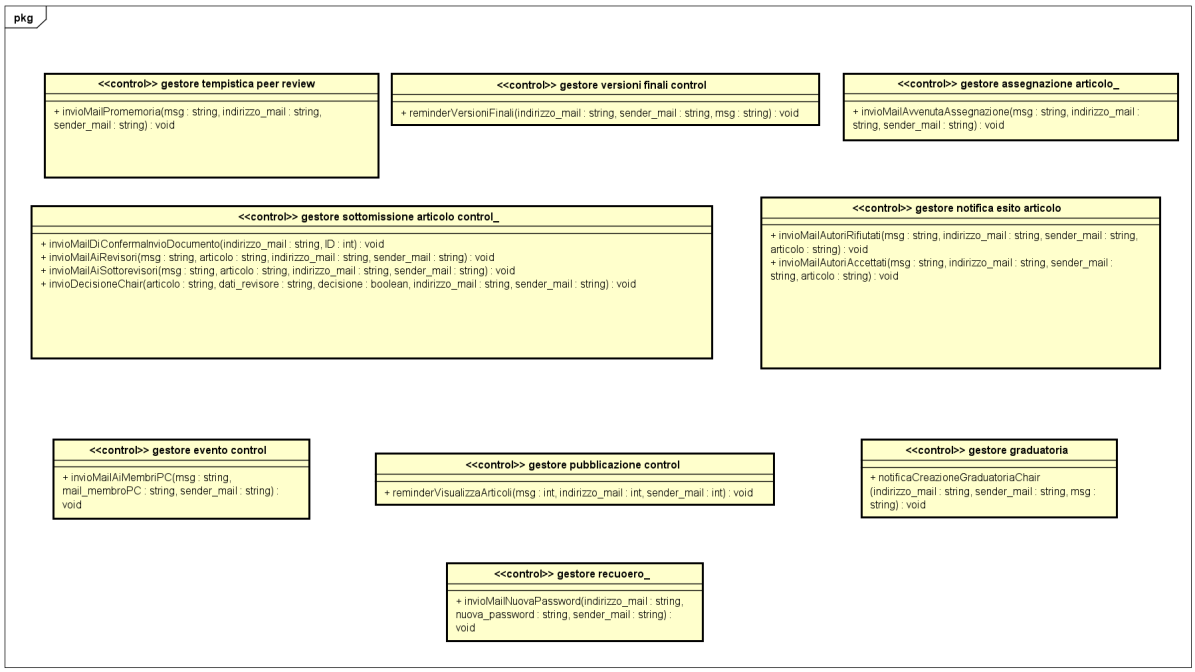
Control Sottomissione Articoli



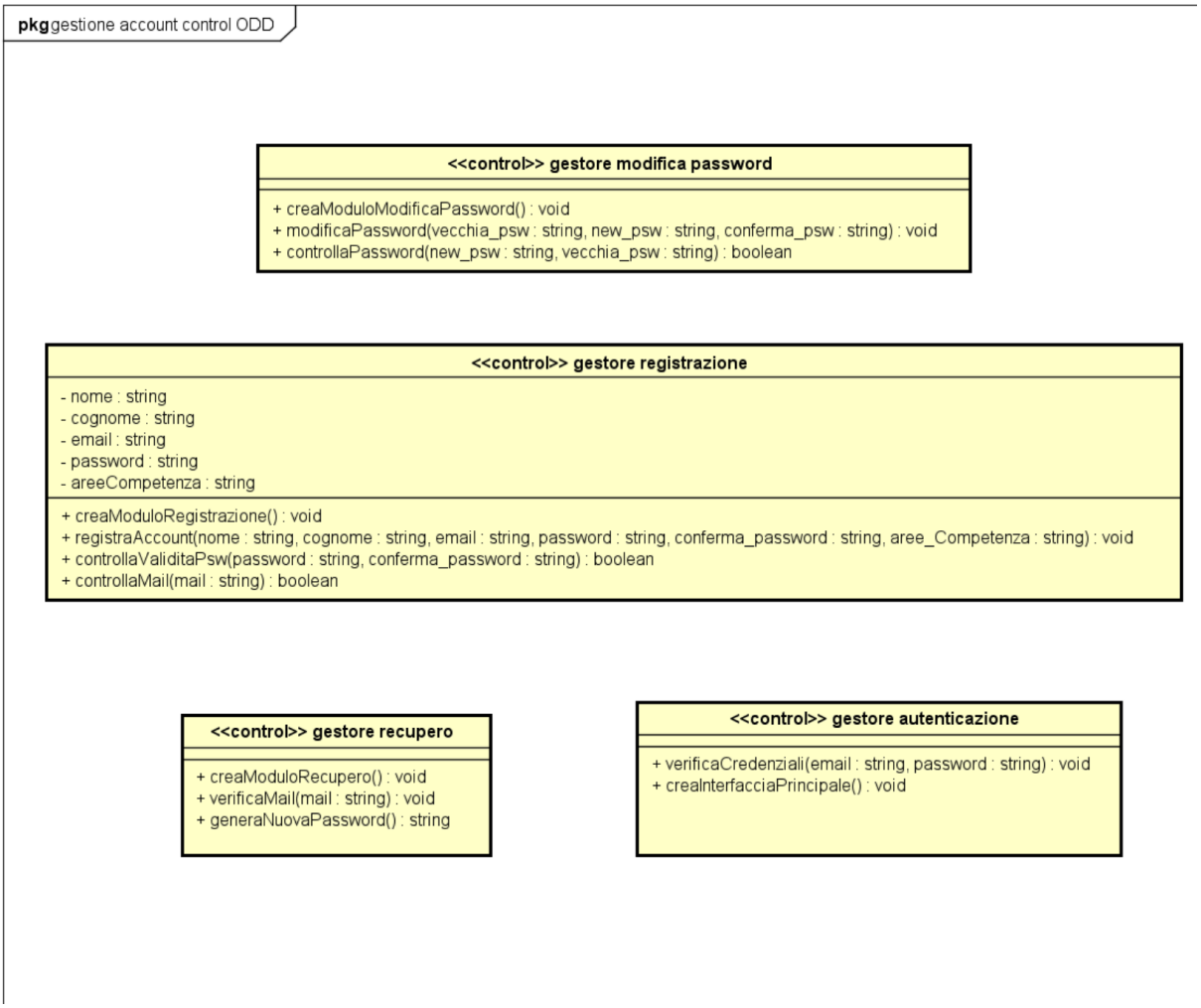
Interface sottomissione Articoli



Control Gestione Notifiche



Control Gestione utenti



Interface Gestione utenti