

# UNIVERSITÀ DEGLI STUDI DI PALERMO FACOLTÀ DI INGEGNERIA

# LAUREA TRIENNALE IN INGEGNERIA INFORMATICA



AUTORI

Mirko Stefano Cirivello Maria Chiara Ciuffoli Flavio Giarrusso

Alessia Pizzuto

DOCENTE

Dott.sa Seidita Valeria

lr	ntroduzione	2
	Object Design Trade-offs	2
	Linee guida per la documentazione delle interfacce	2
	Linee guida per la documentazione dei DBMS	2
	Linee guida per la documentazione dell'invio Mail	3
Ρ	ackages	3
	Utils	3
	Commons	3
	Entity	3
	Com.org.mariadb.jdbc	. 4
	javafx	. 4
	java	. 4
Object design UML		5
	Utils	. 5
	Commons	. 6
	Entity	7
	Interfacce Gestione Organizzazione evento	. 8
	Control Gestione Organizzazione Evento:	. 8
	Interfacce Gestione versione finale e stampa	9
	Control Gestione Versione Finale e Stampa	9
	Control Peer review	10
	Interface Peer review	11
	Control Sottomissione Articoli	12
	Interface sottomissione Articoli	12
	Control Gestione Notifiche	13
	Control Gestione utenti	13
	Interface Gestione utenti	14

# 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

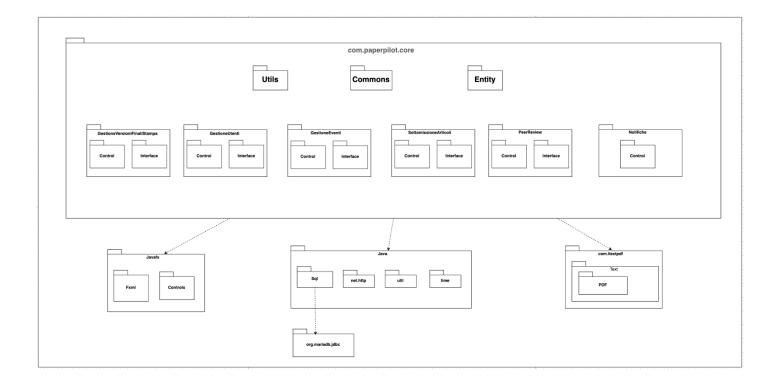
Per la gestione della persistenza dei dati all'interno del sistema "PaperPilot", si è scelto di adottare un modello relazionale. Questa decisione è motivata dalla necessità di gestire in modo efficiente e strutturato le relazioni complesse tra le diverse entità del sistema, quali articoli, autori, revisori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

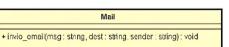
# Utils

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 + chieol\_orano(): LocalTime + chiedi\_dete\_corrente(): LocalDate + confronte\_drario() boolean + confronta date(): boolean



+ quary richiasta indirizz mail revisori o sotto-revisori(nome\_evento : int) : list + quary richiasta lista versioni finali articoli(nome\_evento : string) . Ist

query richieste articolo(nome\_articolo : string) : string

#### Utlls

- → creaLoader(path: string) | FXMLLoader + creanterfaccia(loader: FXMLLoader, w : int, n : int, stage : Stage)
- cambieinterfaccia(path string, stage: Stage, calback NULL, w∵int, h∵int)
   croaPannelloConforma(msg∶stnng)
- creaPannelloErrore(msg:string) void

# <<Boundary>> DBMSdaemon - connessioneDBMS : boolean + connect() boolean + query richiesta graduatorie per cui lutente è chair(dati\_utente : string) : list + quary richiesta cortenuto graduatona (nome\_graduatona : string) . list + query invio decisione al sistema (articolo string) : void + quary richiosta graduatorio per cui lutento è chair e relative recensioni e rate(dati\_ucente ; string) ; list + query richiesta recensioni e rate(articote suring). list + query richiesta recensioni e rate(articote suring). list + query richiesta liste membri PC degli eventi per cui l'utente è chair(dati\_utente string) list + query venfice presenza all'interno della lista(dati\_utente string, maili string, nome\_evento string): booken + query aggiorna lista membri PC(deti\_utono: string, mail: string, nome\_evento: string): void + query richiesta data di scadenza invio articoli(nome\_evento: string): date + query inchiesta maid del chair(norme\_evento\_string) - saine + query ottieni liste essegnezione enticoli-revisori(nome\_evento\_string) - list + query inchiesta insta revison disponitoli(nome\_evento\_string) - list + query revisore de agglungere(erricolo\_string\_revisore - string\_lists\_essegnezioni - list) - void + query richiesta liste articoli(daf\_utente\_string) - list + query richiesta soto-revisori(nome\_evento - string\_lists\_essegnezioni - list) + query assegnazione sotto-revisore all'articolo(sotto\_revisore : string, articolo : string) : void + query richiesta data scadenza invio revisioni(nome\_evento : string) : Dete query richiesta articoli e recensioni[nome\_evento\_string): list query richiesta mail del chain(nome\_evento: string): string query ottieni lista membri PC eventi precedenti(dati\_urente : string): list + query inclaim teat information in Construction and precental processing in the query inserimento ellegate, dad personal nome evento, informazioni aggiuntive ed D(documento: string, nome\_evento: string, nome\_evento: string, connome\_autoro: string, and: string, areo\_compotenza: string, D: int): void + query inchiesta indirezz mail de membri PC(liste\_membri: list, nome\_evento: string): list + query inchiesta dalla scaderiza invio versioni final (nome\_evento: string): Dute + query richiesta indirizzi mail autori articol acceltati(nome\_evento : string) : l'st + query richiesta lista titol articoli assegnati(dati\_uterxe : string) : list + query obtanti data scadenza per invio revisione (nome\_articolo: string): Date + query richiesta articolo e informazioni relativo ad esso (nome\_articolo: string): list + quary invior revisione(nome\_articolo : string, review; string, reae : int) : vold + query richiesta articoli accettati che attendono una versione finale(nome\_autore : string, cognome\_autore : string, nome\_evento : string) : i st + 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 deprey find allocol find equations acting, recently and string, rome\_evento : string, nome\_evento : string) : list duery timuovi revisore(nome\_articolo : string nome\_revisore : string, cognome\_revisore : string) : vold duery timuovi revisore(nome\_articolo : string nome\_revisore : string, cognome\_revisore : string) : list duery tichiesta articoli accettab, recensioni articoli nome evento e mail degli autori(nome\_evento : string) : list + query richiesta articoli infiutati, 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) i list query richiesta ID e indirizzo mail (nome\_articolo string, nome\_evento string) ; string) + query richiesta indinzzo mail editore(nomo\_evento : im) : string + query richiesta graduatoria per cui lutenta a chair(dat\_utento : string) : list + query richiesta contenuto graduatoria(nome\_graduatoria: string) . list + query rimozione articolo(nome\_articola: int, nome\_graduatoria: int): vold + query richiesta liste membri PC degli eventi per cui l'utente è chaindatt\_utente : string). Ilist • query rimozione membro(nome\_liste\_membri : string nome\_membro : string, cagname\_membro : string) void + query sahataggio lista(nome\_lista : string) · vold + query richiesta data scaderza invo document(nome\_evento : string) : Date

+ query creazione evento(norme\_evento: string, num\_min\_revisore: int. num\_max\_anticoli: int, range\_valuazzione int, data\_scad\_invio-articoli: Date, data\_scad\_invio\_revisioni Date, data\_scad\_invio\_versioni\_finali Date) void

# **Entity**

# pkgentity

utente
- nomeUtente : string
- cognomeUtente : string
- cognomeUtente : string
- maiUtente : string
- areaCompetenza : string
- areaCompetenza : string
- ildUtente int
- ildEvento : int
- ildEvento : int
- ildEvento : int
- Ruolo : string
- Password : string
- setNomeUtente(nomeUtente : string) : void
- getNomeUtente(cognome : string) : void
- getSomeOmeUtente(cognome : string) : void
- getCognomeUtente(cognome : string) : void
- getCognomeUtente(cognome : string) : void
- getCognomeUtente(cognome : string) : void
- getMaiUtente() : string
- setAreaCompetenza() : string
- ceateFromDB(: ResultSet) : utente
- setdUtente() : int)
- getdUtente() : string
- setdUtento() : string
- setdUtento() : string
- setdUtento() : string
- setGuolo() : string
- setGuolo() : string
- setGuolo() : string
- setPassword() : string

# assegnazione automatica -nomeArticolo:string -nomeArticolo:string -nomeRevisore:string -cognomeRevisore:string -cognomeRevisore:string -topomeRevisore:string -topomeRevisore:string -topomeRevisore:string -topomeRevisore:string -topomeRevisore:string -topomeRevisore:string -topomeRevisore:string -topomeRevisore:int -toldArete:int -toldArete:int -toldArete:int -toldArete:int -toldArete:int -settomeArticolo() string -setomeArticolo() string -setomeRevisore() string -set

evento

nomeEvento string
numMnRevsoriPerArticolo int
NumMsArticoliPerGevisore int
rangeValutazione int
dataScaderazini vo Articoli Date
dataScaderazini vo Versioni Finali : Date
idiSevanto int
idiAssegnazione int
idiAssegnazione int
idiAssegnazione int
idiAssegnazione int
idiGradutationa int
setti int
setti
setti int
setti
setti int
setti
setti int
setti
setti
setti int
setti
set

graduatoria automatica

nomeArticolo isting

nomeRevisore string

cognomeRevisore string

nomeEvento string

nomeScotorevisore isting

nomeScotorevisore isting

cognomeSotorevisore isting

- PunteggioFinale int

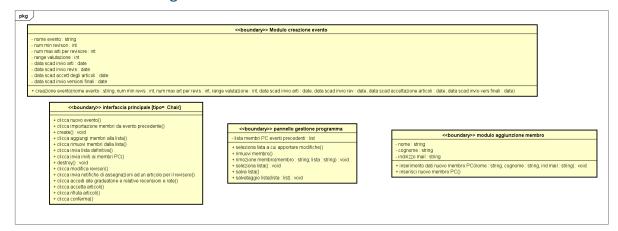
totoloGraduatonaEvento isting

- IdEvento int

- IddArticol Int

- Idd

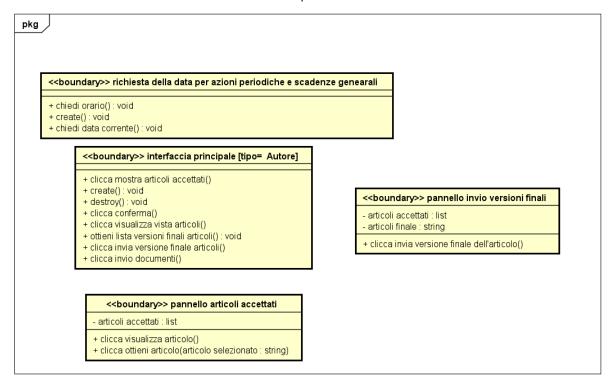
# Interfacce Gestione Organizzazione evento



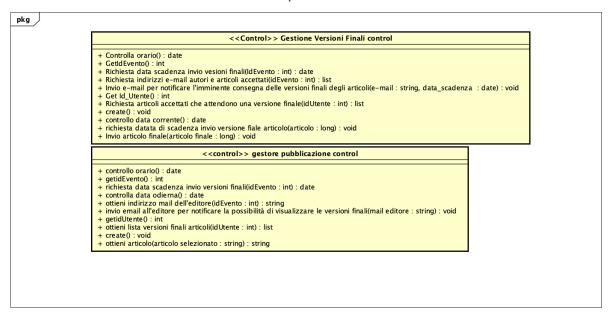
# Control Gestione Organizzazione Evento:



# Interfacce Gestione versione finale e stampa



# Control Gestione Versione Finale e Stampa



# Control Peer review

#### pkg peer review

#### <<control>> Gestore Tempistica peer review

- + getidEvento() : void + richiesta data scadenza invio revisioni(idEvento : int) : date + Invio mail di promemoria() : void

#### <<control>> gestore invio revisioni

- + getidUtente(): int

- + dela data odierna(): void + richiesta lista titoli articoli assegnati(idUtente : int) : list + ottieni data scadenza per invio revisione articolo(titolo articolo : string) : date + ottieni articolo e informazioni reltive ad esso(titolo articolo : string) : list
- + create(): void
- + invio revisione(contenuto revisione : string, rate : int, stato revisone : string) : void

#### <<control>> gestore graduatoria

- + controlla orario(): void
- + getidEvento() : int + richiesta data scadenza invio revisioni(idEvento : int) : date

- + richiesta data stadenza inwo revision(dezvento : int) : date + richiesta articoli e recension(idEvento : int) : list + generazione graduatoria() : void + create() : void + richiesta mail del chair (idEvento : int) : string + invio notifica di creazione della graduatoria al chair() : void

#### <<control>> gestore recensioni

- + getidUtente(): int
- + richiesta graduatoria per cui l'utente è chair e relative recensioni e rate(idUtente : int) : list + create() : void

- + create(): void
  + richiesta recensioni e rate(articolo: string): list
  + richiesta graduatoria per cui l'utente è chair(idUtente: int): list
  + richiesta contenuto graduatoria(graduatoria: int): list
  + invia decisione al sistema (articolo da accettare: string): void
  + rimozione articolo(articolo: string): void
  + richiesta articoli accettati, recensioni articoli,nome evento e mail degli autori(idUtente: int): list
  + richiesta articoli rifiutati, recensioni articoli,nome evento e mail degli autori(idUtente: int): list
  + articoli: idiutati recensioni evento indirizzo mail autori(): list

- + articoli riflutati, recensioni, evento, indirizzo mail autore(): list
  + richiesta data di scadenza invio versioni finali articoli(nomeEvento: string): date
  + invio mail agli autori degli articoli riflutati (articoli riflutati,recensioni,evento,indirizzo mail autore))(): void
  + invio mail agli autori degli articoli accettati (articoli accettati,recensioni,evento,indirizzo mail autore,data di scadenza)(): void

# Interface Peer review

## pkg

#### <<boundary>> richiesta dell'orario per notifiche imminenti

+ chiedi orario() : void + create(): void

#### <<box></box<br/>revisore]</br>

- + cliccaAssegnaSottorevisore()
- + cliccaAccettaArticolo()
- + cliccaRctettaArticolo() + cliccaRifiutaArticolo() + cliccaInviaRevisioni() + create(): void

#### <<bod><<br/>soundary>> visualizzatore articoli

- lista titoli articoli assegnati : list
- + clicca revisiona()
- + ottieni data scadenza per invio revisione(titolo articolo : string) : date

#### <<box></box<br/>ondary>> modulo valutazione articolo</br>

- contenuto articolo : string campo testo recensione : string
- rate : int
- + compilazione campi revisione(testo recensione : string, rate : int) : void + clicca invia revisione() : void

#### <<box></box><br/>richiesta della data per azioni periodiche e scadenze genearali

- + chiedi orario() : void + create() : void

#### <<box><<br/>boundary>> pannello invia notifica all'autore

+ clicca invia notifiche di accettazione e rifiuto articolo() + create() : void

### <<box></box> interfaccia principale [tipo= Chair]

- clicca nuovo evento()
   clicca importazione membri da evento precedente()
   create() · void
   clicca aggiungi membri alla lista()
   clicca rimuovi membri dalla lista()
   clicca rimuovi membri dalla lista()
   clicca rimuia lista definitiva()
   clicca rimuia inviti ai membri PC()
   destroy() · void
   clicca modifica revisori()
   clicca invia notifiche di assegnazioni ad un articolo per il revisore()
   clicca caccid alle graduatorie e relative recensioni e rate()
- + clicca accedi alle graduatorie e relative recensioni e rate() + clicca accetta articoli()

- + clicca rifiuta articoli() + clicca conferma()

# <<bod></body>

- + cliccaAccettaArticolo(): void
- + cliccaRifiutaArticolo(): void + cliccalnviaRevisioni(): void

#### <<box><<br/>boundary>> pannello feedback revisori

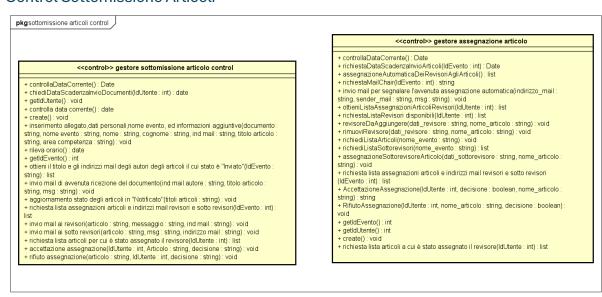
- contenuto graduatoria : list
- revisioni : string
- recensioni : string rate : int nome graduatoria : string

- nome graduatoria : string
  + selezione articolo(): void
  + clicca vedi recensioni, rate e rate complessivo()
  + richiesta recensioni e rate(articolo: string): list
  + clicca visualizza graduatoria()
  + richiesta contenuto graduatoria()
  + richiesta contenuto graduatoria(nome graduatoria: string): list
  + clicca accetta articolo(articolo selezionato: string)
  + inivia decisione al sistema(articolo da accettare: string): void
  + clicca rifuta articolo(articolo selezionato: string): void
  + clicca conferma decisioni()
  + rimozione articolo(articolo selezionato: string): void

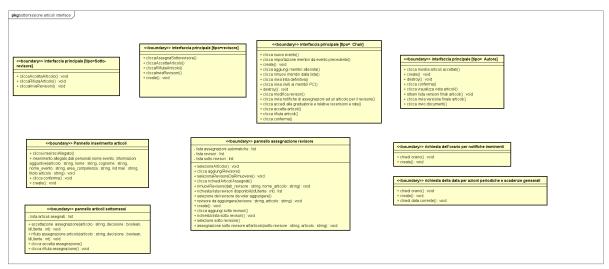
#### <<box></box<br/>revisore]

- + cliccaAssegnaSottorevisore() + cliccaAccettaArticolo() + cliccaRifiutaArticolo()
- + cliccalnviaRevisioni() + create() : void

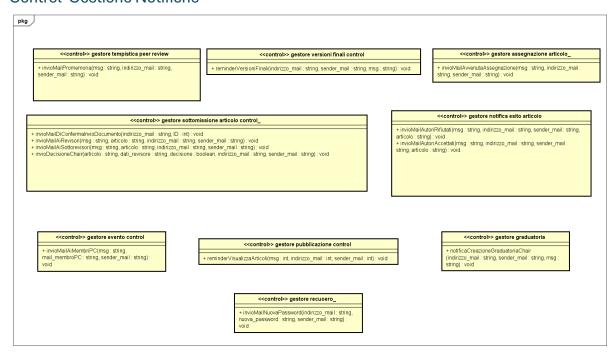
# Control Sottomissione Articoli



# Interface sottomissione Articoli

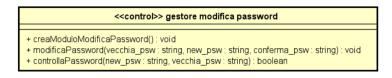


# Control Gestione Notifiche



# Control Gestione utenti

pkg gestione account control ODD



# - nome : string - cognome : string - email : string - password : string - password : string - areeCompetenza : string + creaModuloRegistrazione() : void + registraAccount(nome : string, cognome : string, email : string, password : string, conferma\_password : string, aree\_Competenza : string) : void + controllaValiditaPsw(password : string, conferma\_password : string) : boolean + controllaMail(mail : string) : boolean

# + creaModuloRecupero() : void + verificaMail(mail : string) : void + generaNuovaPassword() : string

<<control>> gestore recupero

+ verificaCredenziali(email: string, password: string): void
+ crealnterfacciaPrincipale(): void

# Interface Gestione utenti

pkggestione utenti interface

# <<bod><<br/>boundary>>Modulo Login

- campoEmail : string
- campoPassword : string
- + cliccaAccedi() : void + cliccaRegistrati() : void
- + cliccaRecuperaCredenziali(): void

# <<box><<br/>boundary>> modulo registrazione

- campoNome : int
- campoCognome : int
- campoMail: int
- campoPassword : int
- campoConfermaPassword : int
- tastoRegistrati : int
- campoAnnulla : int
- campoAreaDiCompetenza : int
- + cliccaRegistrati(): void
- + cliccaConferma(): void

# <<body><br/><<br/>boundary>> modulo recupero

- campoEmail: string
- + cliccaConferma(): void

# <<bod><<bod><br/>dary>> modulo modifica password</br>

- campoPasswordVecchia: int
- campoNuovaPassword : int
- campoConfermaPassword : int
- + cliccaSalva(): void