Ingegneria Informatica A.A 2022/2023

Object Design Document

Team SPVM

Macaluso Alessandro
Picone Alessandro
Spedito Antonio
Venturella Giuseppe

INDICE:

Indice

Introduzione

Object design trade-offs

Durante la realizzazione del progetto, si è scelto di utilizzare un approccio di tipo modulare, atomizzando e raggruppando quanto più possibile tutte le caratteristiche del sistema. L'obiettivo che l'approccio si pone è quello di garantire robustezza in termini di facilità nella lettura, scrittura e riuso dei singoli moduli, talvolta permettendo, qualora fossero necessarie, eventuali modifiche in futuro, la riscrittura dei moduli o la semplice modifica ovviando quindi al problema della riscrittura totale dell'intero sistema (complicando progettazione e implementazione). Un altro dei problemi che la suddivisione in moduli permette di ovviare è proprio quello di isolare gli errori nei singoli moduli, in questo modo si potrà quindi agire sul singolo sottosistema e non sull'intero sistema. La struttura è di tipo Repository, basata su una struttura dati centrale (database) e un insieme di nodi disaccoppiati (sottosistemi) che possono interagire solo attraverso la Repository. I sottosistemi sono composti da un'interfaccia utente che comunica con il controllore sottostante, che ha il compito di far funzionare la parte logica del software (riferita al singolo sottosistema) e di gestire le richieste verso il sottosistema DBMSManager(MariaDB) (che gestisce le comunicazioni entranti verso la struttura dati centrale prevedendo inoltre eventuali errori).

Tecnologie utilizzate per le interfacce: JavaFX

Per la realizzazione delle interfacce si è scelto di utilizzare il framework JavaFX: una famiglia di software applicativi, basati sulla piattaforma Java, per la creazione di interfacce grafiche. JavaFX sfrutta documenti di tipo FXML, basati cioè sul linguaggio di markup FXML, derivato da XML e creato da Oracle Corporation per scrivere componenti utilizzando i tag predefiniti (da libreria). Si è scelto inoltre di utilizzare il Tool 'SceneBuilder' che sfrutta il meccanismo drag & drop per facilitare e velocizzare l'implementazione delle interfacce grafiche. Per le componenti di tipo "Calendar" si è deciso invece di implementare un'ulteriore libreria chiamata "CalendarFX" che aggiunge dei tag (e rispettive classi) al framework JavaFX.

Tecnologie utilizzate per la gestione e realizzazione del Database

Per lo sviluppo del Database e la relativa gestione (DBMS) si è scelto in primis di studiare ed analizzare nel dettaglio i requisiti principali del problema, successivamente abbiamo sviluppato un modello E-R (Entity-Relationship) per fissare quali fossero le entità e che tipo di relazioni avessero tra loro. Una volta realizzato e fissato il modello E-R, abbiamo deciso di implementarlo e di gestirlo mediante MariaDB. Il DBMS scelto si basa sulla gestione di Database di tipo relazionale. MariaDB nasce come fork della tecnologia MySQL. Nella fase implementativa abbiamo utilizzato JDBC (Java DataBase Connector), un connettore e un

Indice

driver per database che consente l'accesso e la gestione della persistenza dei dati mediante il linguaggio di programmazione Java, indipendentemente dal tipo di DBMS utilizzato.

Tecnologie utilizzate per l'invio delle Mail

Per garantire una comunicazione quanto più affidabile possibile, si è deciso di implementare un sistema per l'invio di Mail da parte dell'azienda. Abbiamo sfruttato la tecnologia JavaMail API: un package della Sun Microsystems che fornisce le classi necessarie per la gestione della posta elettronica in linguaggio Java. JavaMail supporta i principali protocolli di posta elettronica (POP/POP3, SMTP e IMAP). Nel nostro progetto era necessario l'utilizzo del protocollo SMTP per l'invio delle e-mail. Con javax.mail abbiamo implementato il package JavaMail e abbiamo configurato la nostra casella postale di riferimento (spvm@alessandromacaluso.it) con i parametri SMTP e con le credenziali di autenticazione forniti dal servizio postale Aruba. Una volta configurati i parametri necessari per l'autenticazione nel sistema Aruba, mediante protocollo SMTP, potremo inviare le nostre mail esplicitando l'indirizzo mail del destinatario, il corpo e l'oggetto del messaggio. Al dominio di riferimento appartengono inoltre le mail aziendali dei singoli impiegati ma a scopo dimostrativo si è scelto di utilizzare un'email personale.

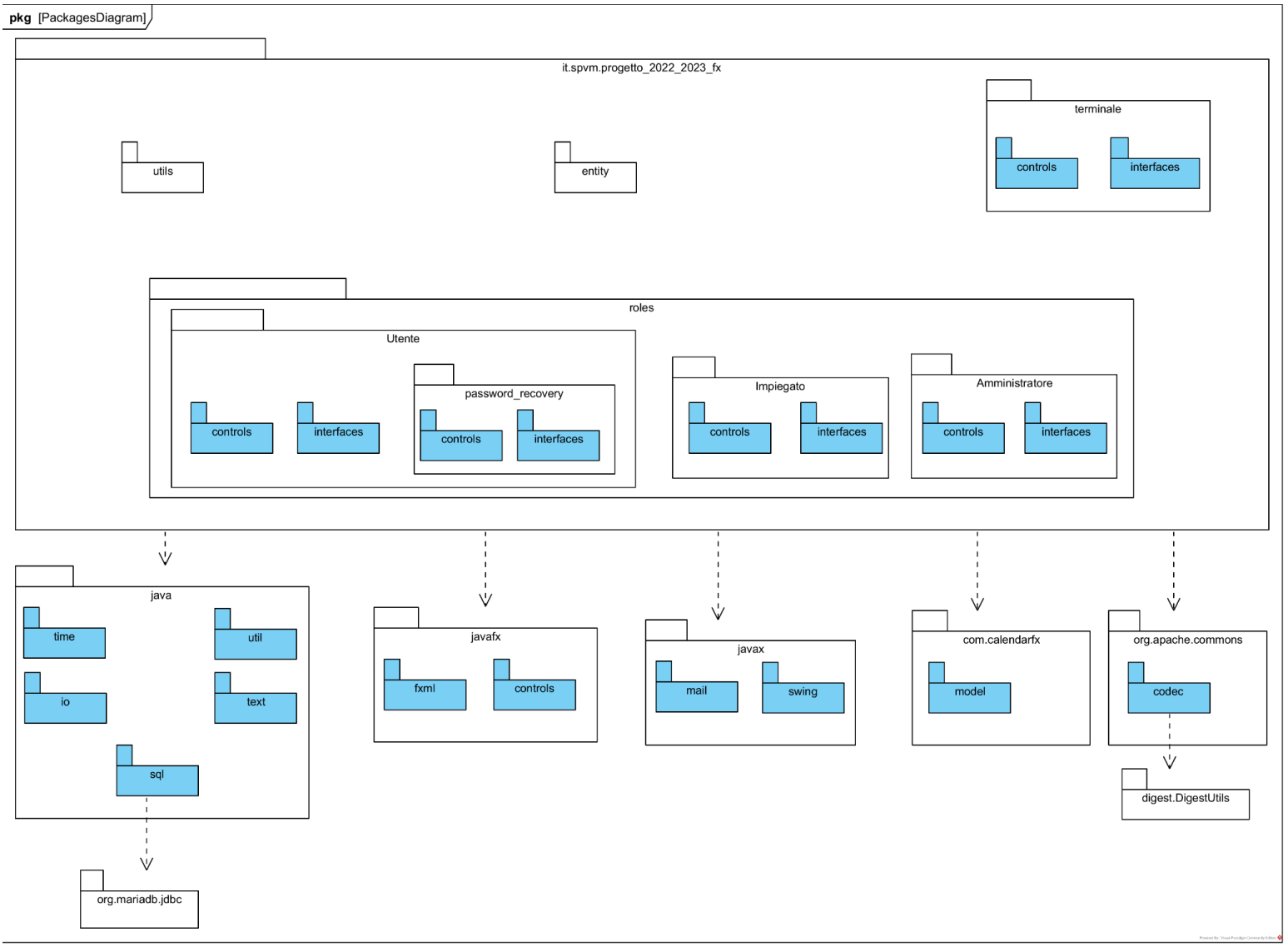
Tecnologie utilizzate per la gestione dei processi automatizzati

Per gestire l'insieme di tutti quei processi aziendali temporizzati e necessariamente automatizzati, abbiamo deciso di implementare i Thread: suddivisione di un processo in una o più istanze. Il ruolo principale del Thread (nel progetto) è quello di temporizzatore. Ogni secondo il sistema verifica se e quando avviare gli automatismi previsti durante la fase di progettazione. Gli automatismi progettuali sono: generazione dei turni, calcolo ed accredito degli stipendi, gestione delle comunicazioni di mancata firma, gestione della chiusura dei servizi e infine rilevazione di ingresso ed uscita per ogni impiegato. Il Thread rimane attivo 24/7 per garantire la puntualità degli automatismi.

Indice

Packages

Packages Diagram



Indice

it.spvm.progetto_2022_2023_fx

Rappresenta il package principale del progetto.

- **Roles**

  Contiene tutti i package relativi alla gestione dell'account, compreso il sistema di autenticazione.

  - **Utente**

    Contiene tutti quei pacchetti da installare sulle postazioni di tutti i dipendenti (indipendentemente dal ruolo occupato). Il sistema stesso sarà in grado di smistare il corrispettivo pannello in base all'autenticazione. "Interface" contiene tutte le classi associate alle interfacce utente realizzate con FXML, "controls" invece gestisce tutta la parte logica delle funzionalità dell'Utente generico.

  - **Utente.password_recovery**

    Contiene tutti quei pacchetti da installare sulle postazioni di tutti gli Utenti (indipendentemente dal ruolo occupato), in particolare "Interface" contiene tutte le classi associate alle interfacce utente realizzate con FXML. "Controls" invece gestisce tutta la parte logica del programma in merito al recupero password (step 1 e step 2).

  - **Impiegato**

    "Interface" contiene tutte le classi associate alle interfacce utente realizzate con FXML, "controls" invece gestisce tutta la parte logica delle funzionalità del ruolo "Impiegato".

  - **Amministratore**

    "Interface" contiene tutte le classi associate alle interfacce utente realizzate con FXML, "controls" invece gestisce tutta la parte logica delle funzionalità del ruolo "Amministratore".

- **terminale**

  Contiene tutti quei pacchetti da installare sui terminali di ingresso e uscita dell'azienda. Il pacchetto "Interface" contiene tutte le classi associate alle interfacce utente realizzate con FXML, "controls" invece gestisce tutta la parte logica di ingresso, uscita di ogni impiegato e include inoltre l'implementazione di tutti gli automatismi.

- **entity**

Indice

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

- **utils**
  Contiene classi wrapper per API, il pacchetto è specializzato per il sistema e quindi poco riutilizzabile in altri contesti.

java

Contiene i package e le librerie standard di Java.

- **time**
  API per le date, il tempo, gli istanti e le durate.
- **io**
  Fornisce input e output di sistema tramite flussi di dati, serializzazione e file system.
- **util**
  Costituisce l'insieme di tutte le strutture dati utilizzate dal sistema.
- **text**
  Fornisce classi e interfacce per la gestione di testo, date, numeri e messaggi.
- **sql**
  Fornisce l'API per l'accesso e l'elaborazione dei dati archiviati in un'origine dati (nel nostro caso database relazionale). Permette quindi di gestire la comunicazione con il DBMS e viene utilizzato nella classe DBMSManager per creare le connessioni ed eseguire query.

org.mariadb.jdbc

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

javafx

Package che gestisce la realizzazione di interfacce utente, in particolar modo gestisce sia la parte logica dell'interfaccia, sia la parte grafica.

- **fxml**
  Permette di caricare i file FXML, e gestisce l'associazione tra le variabili della classe associata e le rispettive componenti grafiche mediante Id.
- **controls**
  Contiene l'insieme delle classi che permettono di definire componenti fxml, in particolar modo possiamo non solo modificarne le singole proprietà ma anche creare degli eventi specifici utilizzando gli EventHandler dedicati [Event-Driven] (Ad esempio: Al click del mouse si scatena un evento).

Indice

javax

Contiene tutti quei pacchetti da installare sulle postazioni di tutti i dipendenti (indipendentemente dal ruolo occupato)

- **mail**
  Permette di sfruttare tutti i protocolli mail (POP/POP3, SMTP e IMAP), viene utilizzato in configurazione SMTP per tutte quelle comunicazioni che l'azienda deve fare nei confronti di un impiegato.

calendarfx

Contiene tutti quei pacchetti da installare sulle postazioni di tutti i dipendenti (indipendentemente dal ruolo occupato)

- **model**
  Contiene l'insieme delle classi dei modelli utilizzati da calendarFX per la realizzazione di calendari, consentendo un'ampia personalizzazione dei calendari. Utilizzato nella Dashboard di ogni account per consultare i turni trimestrali.

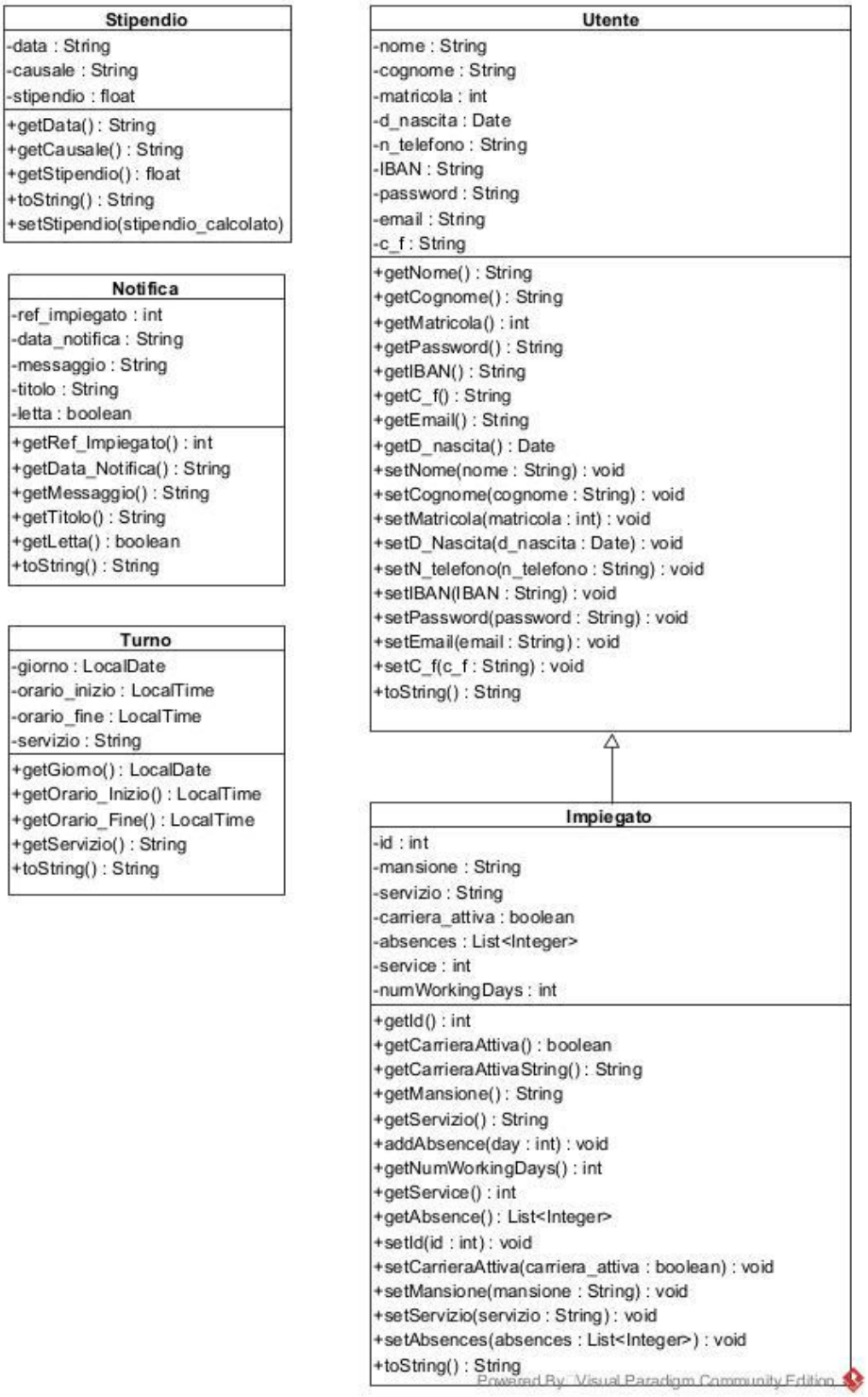
org.apache.commons.codec

Apache Commons Codec fornisce le implementazioni dei più comuni codificatori a Base 64. Viene utilizzato nel progetto il codec "digest.DigestUtils" che consente una crittografia della password in SHA1.

Indice

Object Design UML

Entity



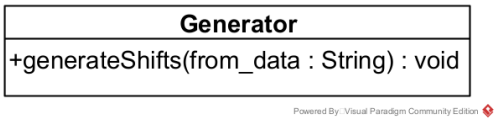
Indice

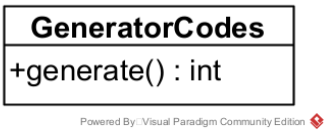
Utils

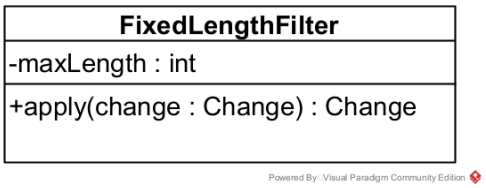
CustomCalendar
+createEntries(data_turno : LocalDate, start_turno : LocalTime, stop_turno : LocalTime, servizio : String) : void

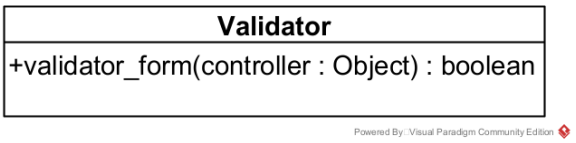
<<Boundary>> **DBMSManager**
-baseUrl : String = spvmproject.ddns.net -port : int = 3306 -user : String = unipa -pass : String = prog_2022_2023 -database : String = unipa -conn : Connection = null -impiegato : Impiegato
+startConnection() : void +closeConnection() : void +getConnection() : Connection +getCurrentImpiegato() : Impiegato +getImpiegatoByMatricola(matricola : int) : Impiegato +getImpiegatoWhere(matricola : int, password : String) : Impiegato +getImpiegatiMancataFirma(i_turno : String, f_turno : String) : ArrayList<Impiegato> +getImpiegatiMancataFirmaImpiegato(matricola : int, i_turno : String, f_turno : String) : boolean +queryGetCountImpiegati() : int +queryGetCountImpiegatiAdmin() : int +queryGetMatricolaImpiegato(email : String, nome : String, cognome : String) : Impiegato +queryGetPermessi() : Map<Impiegato, List<Integer>> +queryGetImpiegatiTurni() : Map<Integer, List<Turno>> +queryGetStipendioImpiegati() : Map<Integer, List<Stipendio>> +queryVerificaCredenzialiImpiegato(matricola : int, password : String) : boolean +queryVerificaIndirizzoEmail(email : String) : boolean +queryVerificaEsistenzaImpiegato(matricola : int) : boolean +queryVerificaEsistenzaImpiegato(matricola : int, nome : String, cognome : String) : boolean +queryVerificaEsistenzaImpiegatoAttivo(matricola : int) : boolean +queryVerificaFirma(matricola : int) : boolean +queryVerificaEsistenzaTurno(matricola : int, i_permesso : String, f_permesso : String) : boolean +queryVerificaEsistenzaPeriodoIncrementoAttivita(data_inizio : String, data_fine : String) : boolean +queryAggiornaPassword(matricola : int, nuovaPSW : String) : boolean +queryAggiornaNumeroTelefono(matricola : int, n_telefono : String) : boolean +queryAggiornaIBAN(matricola : int, IBAN : String) : boolean +queryAggiornaStipendio(matricola : int, stipendio : Stipendio) : boolean +queryAggiornaTurni(impiegati_ridistribuiti : Map<Impiegato, Turno>) : boolean +queryInserisciPermesso(data_inizio_permesso : String, data_fine_permesso : String, String : tipo_permesso, matricola : int) : boolean +queryInserisciPermessoAdmin(matricola : int, data_inizio_permesso : String, data_fine_permesso : String, tipo_permesso : String) : boolean +queryInserisciPresenzaRitardo(motivazione : String) : boolean +queryInserisciNotifica(matricola : int, messaggio : String, titolo : String) : boolean +queryInserisciRitardoImpiegato(matricola : int, motivazione : String) : boolean +queryInserisciIncrementoAttivita(data_inizio : String, data_fine : String) : boolean +queryInsertTurni(matricola : int, turno : Turno) : boolean +queryRecuperaPasswordImpiegato(email : String, password : String) : boolean +queryStipendioWhere(matricola : int) : ArrayList<Stipendio> +queryImpiegati() : ArrayList<Impiegato> +queryImpiegatiOrderBySurname() : ArrayList<Impiegato> +queryImpiegatiTurni() : Map<Impiegato, Turno> +queryImpiegatiServizio(servizio : String) : ArrayList<Integer> +queryImpiegatiAttivi() : ArrayList<Impiegato> +queryStipendiDipendenti() : ArrayList<Stipendio> +queryTurniImpiegato() : ArrayList<Turno> +queryTurniImpiegato(matricola : int) : ArrayList<Turno> +queryLicenziamentoImpiegato(ref_impiegato : int) : boolean +queryAssunzioneImpiegato(nome : String, cognome : String, d_nascita : String, n_telefono : String, servizio : String, iban : String, mail : String, cod_fisc : String, psw : String) : boolean +queryRilevazioneIngresso(matricola : int, nome : String, cognome : String) : boolean +queryRilevazioneUscita(matricola : int, nome : String, cognome : String) : boolean +queryRimuoviTurni(matricola : int, inizio_turno : String, fine_turno : String) : boolean +queryRichiesteMalattiaImpiegati() : Map<String, Integer> +queryUltimoTurno() : String +queryAggiornaNotifica(matricola : int, messaggio : String, titolo : String) +queryOreLavorative(matricola : int) : HashMap<Integer, Float> +queryVisualizzaNotifiche(matricola : int) : ArrayList<Notifica> +queryChiusuraServizio() : ArrayList<Integer> +queryNumeroImpiegatiServizi() : int +queryImpiegatiDisponibili() : List<Impiegato> +queryListaFerie(data_picker_from : DatePicker, data_picker_to : DatePicker) : List<Impiegato> +queryRimuoviFerie(matricola : int) : void +erroreComunicazioneDBMS(e : SQLException) : void

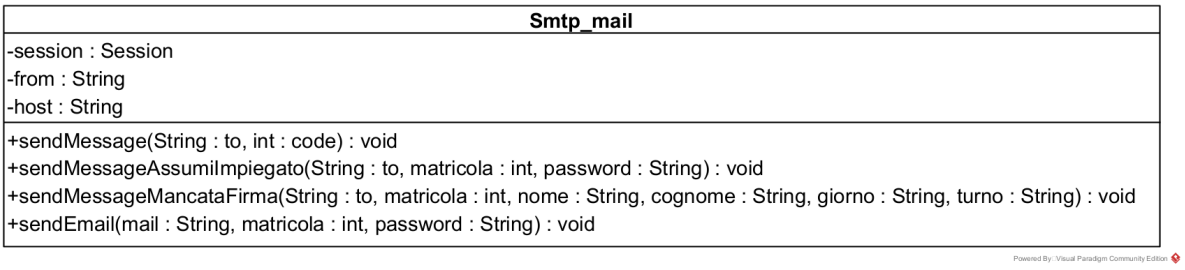
Indice

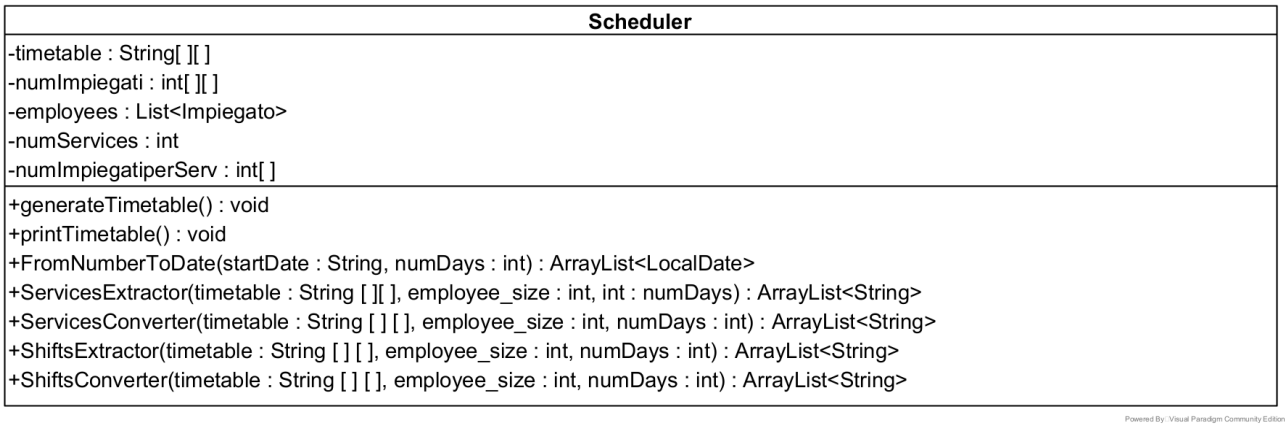

Generator
+generateShifts(from_data : String) : void
Powered By: Visual Paradigm Community Edition

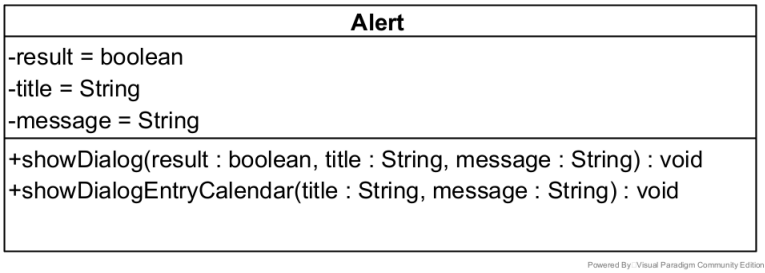

GeneratorCodes
+generate() : int
Powered By: Visual Paradigm Community Edition


FixedLengthFilter
-maxLength : int
+apply(change : Change) : Change
Powered By: Visual Paradigm Community Edition


Validator
+validator_form(controller : Object) : boolean
Powered By: Visual Paradigm Community Edition

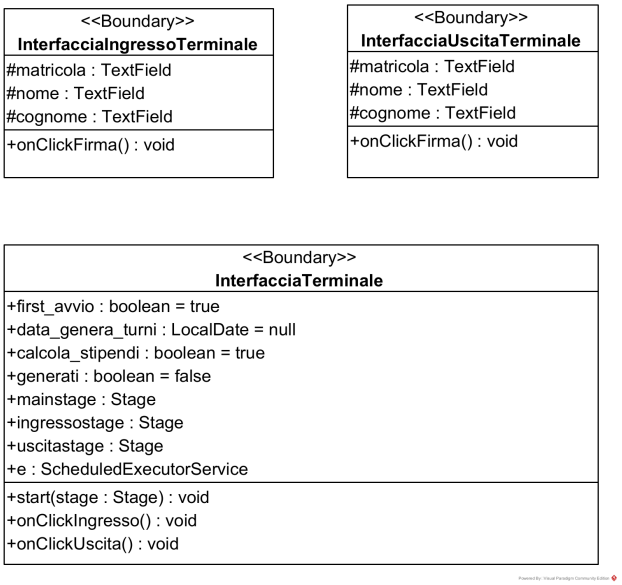

Smtp_mail
-session : Session
-from : String
-host : String
+sendMessage(String : to, int : code) : void
+sendMessageAssumiImpiegato(String : to, matricola : int, password : String) : void
+sendMessageMancataFirma(String : to, matricola : int, nome : String, cognome : String, giorno : String, turno : String) : void
+sendEmail(mail : String, matricola : int, password : String) : void
Powered By: Visual Paradigm Community Edition


Scheduler
-timetable : String[ ][ ]
-numImpiegati : int[ ][ ]
-employees : List<Impiegato>
-numServices : int
-numImpiegatiperServ : int[ ]
+generateTimetable() : void
+printTimetable() : void
+FromNumberToDate(startDate : String, numDays : int) : ArrayList<LocalDate>
+ServicesExtractor(timetable : String [ ][ ], employee_size : int, int : numDays) : ArrayList<String>
+ServicesConverter(timetable : String [ ] [ ], employee_size : int, numDays : int) : ArrayList<String>
+ShiftsExtractor(timetable : String [ ] [ ], employee_size : int, numDays : int) : ArrayList<String>
+ShiftsConverter(timetable : String [ ] [ ], employee_size : int, numDays : int) : ArrayList<String>
Powered By: Visual Paradigm Community Edition

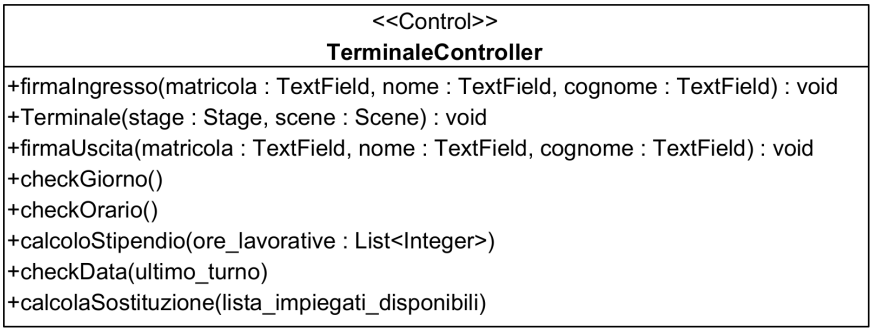

Alert
-result = boolean
-title = String
-message = String
+showDialog(result : boolean, title : String, message : String) : void
+showDialogEntryCalendar(title : String, message : String) : void
Powered By: Visual Paradigm Community Edition

Indice

Terminale.interfaces

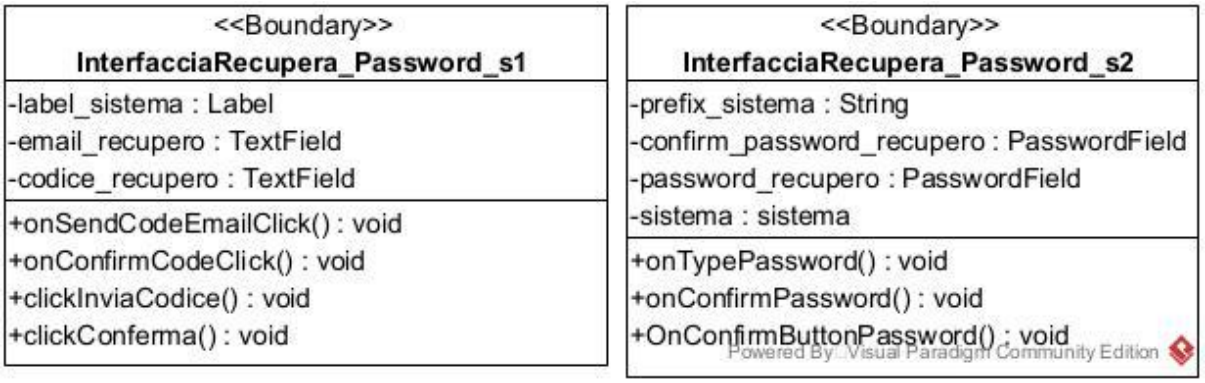


Terminale.controls

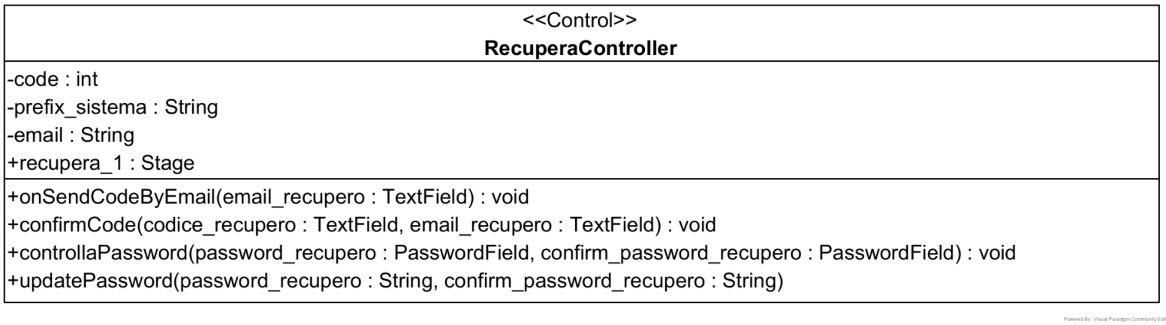


Indice

Roles.utente.password_recovery.interfaces

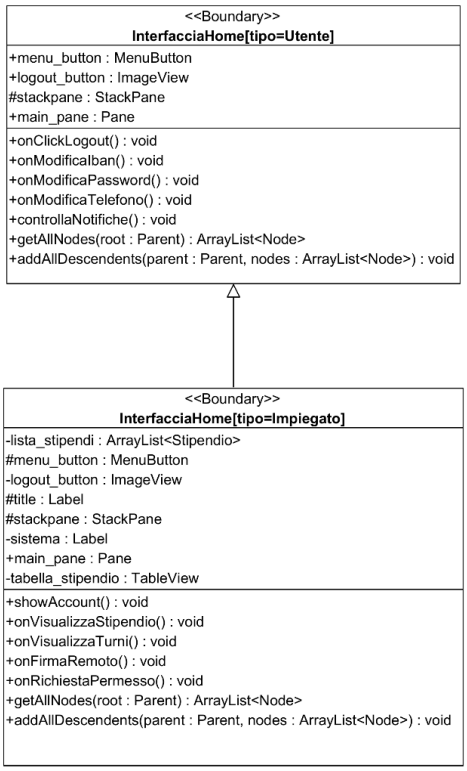


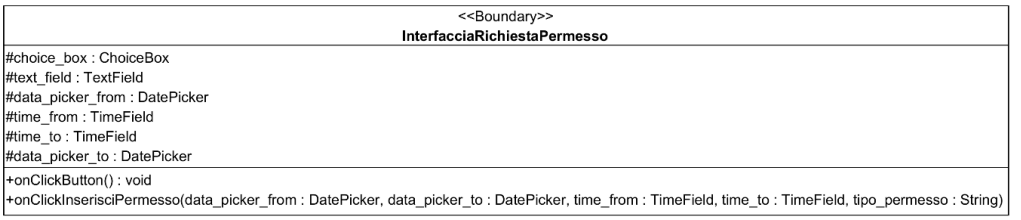
Roles.utente.password_recovery.controls

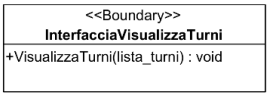


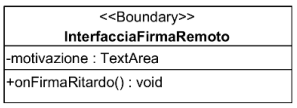
Indice

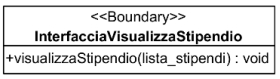
Roles.impiegato.interfaces



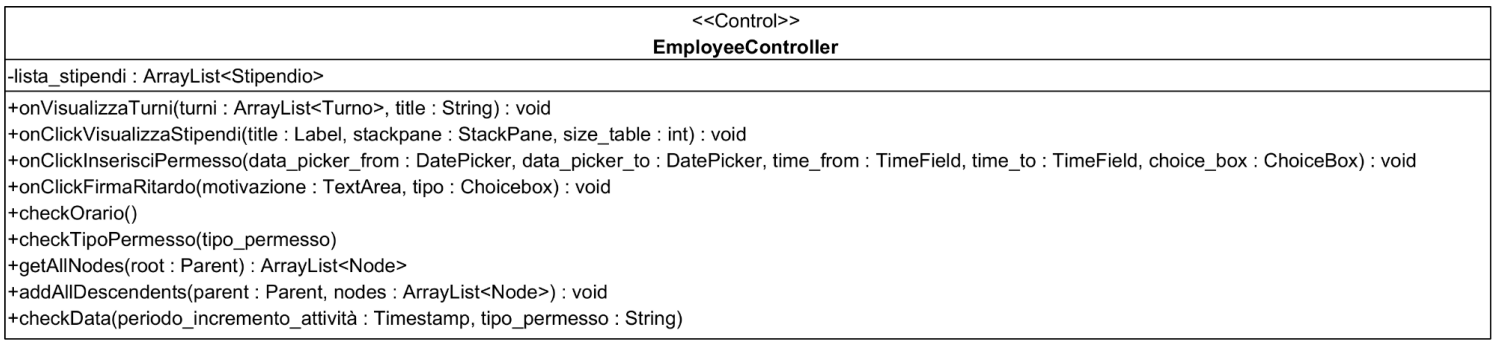






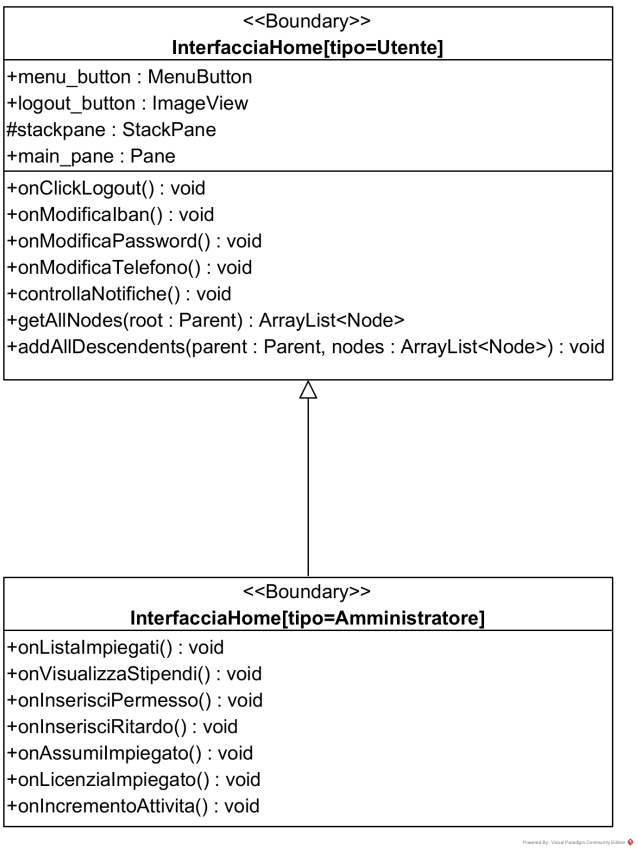


Roles.impiegato.controls

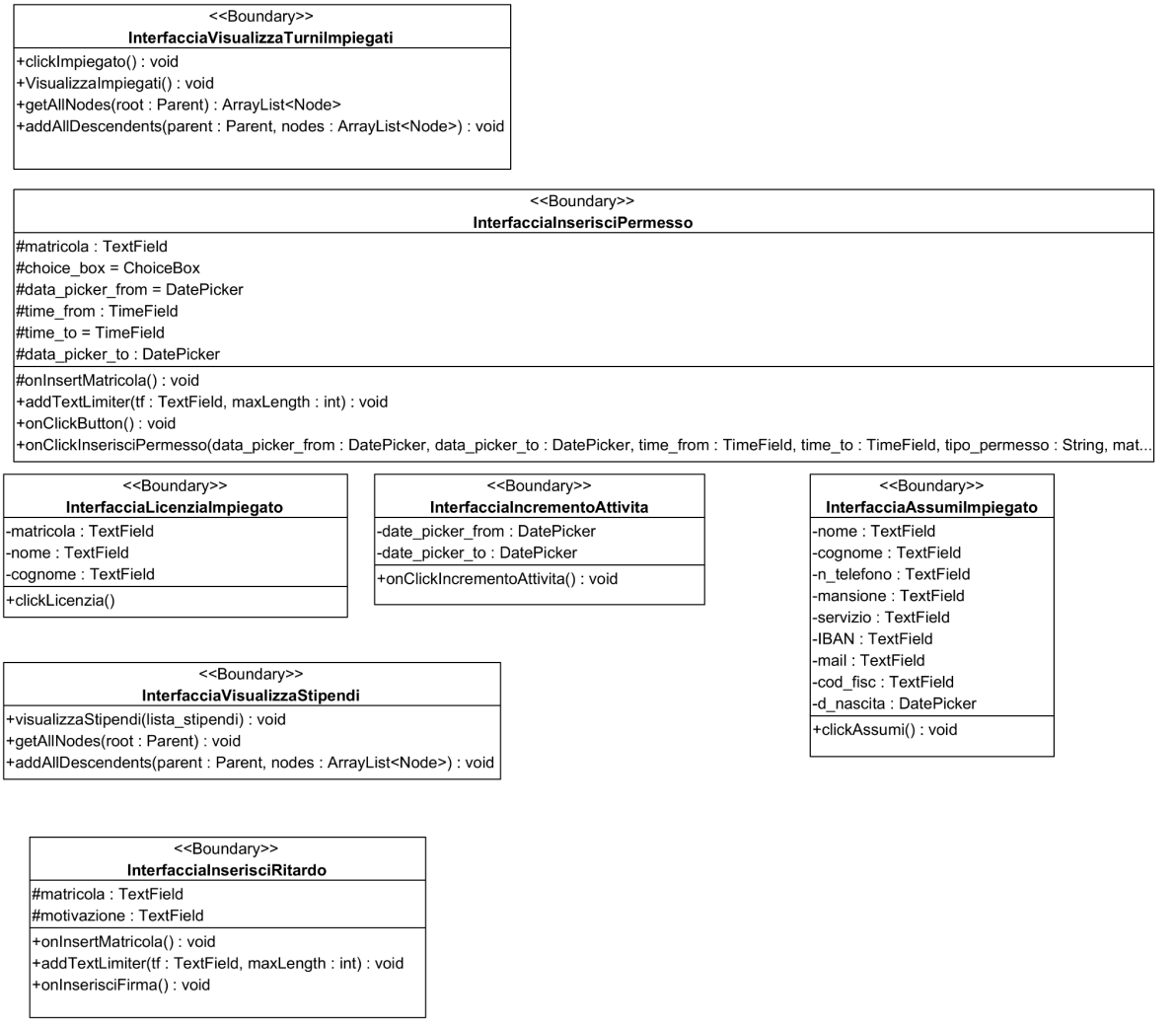


Indice

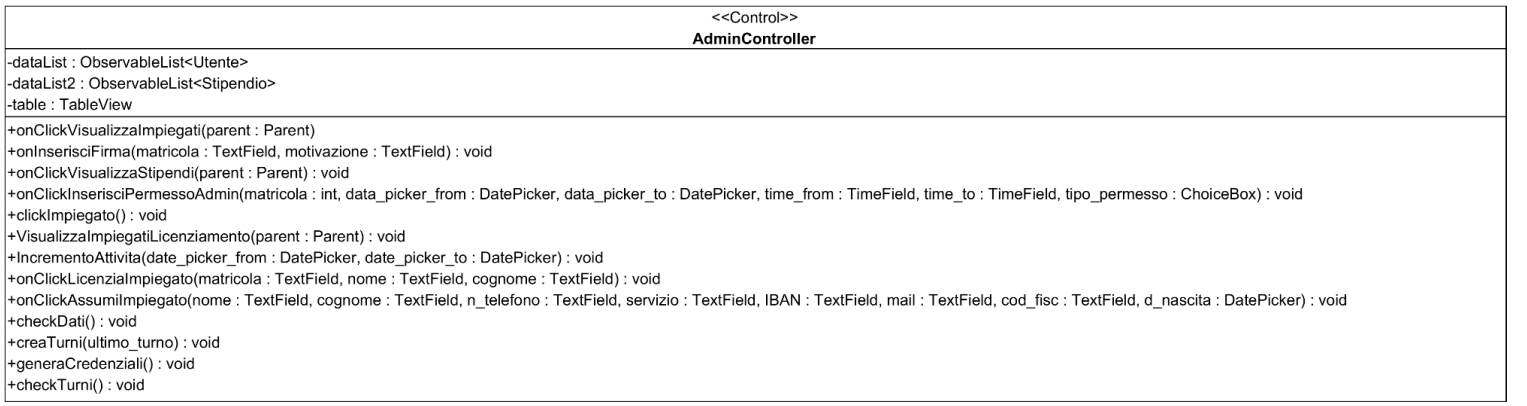
Roles.amministratore.interfaces



Indice

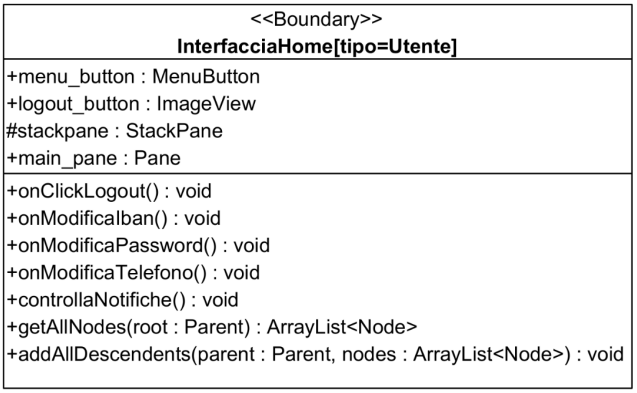


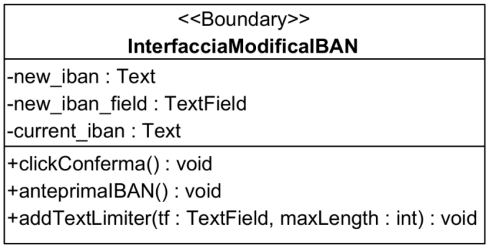
Roles.amministratore.controls

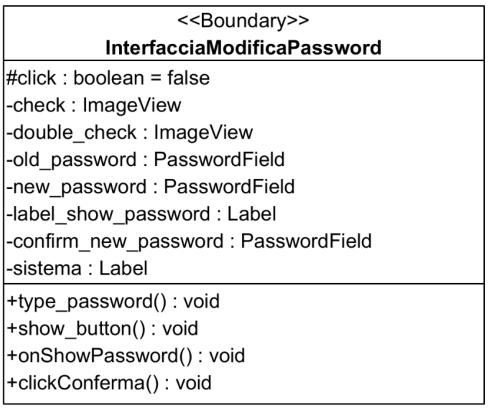


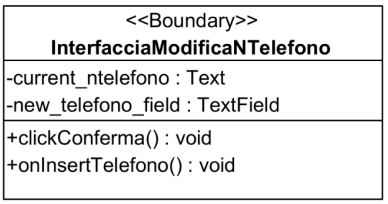
Indice

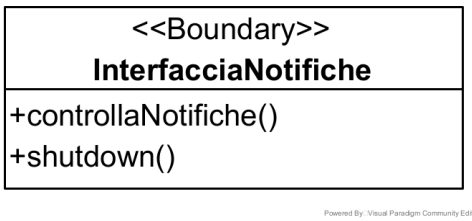
Roles.utente.interfaces



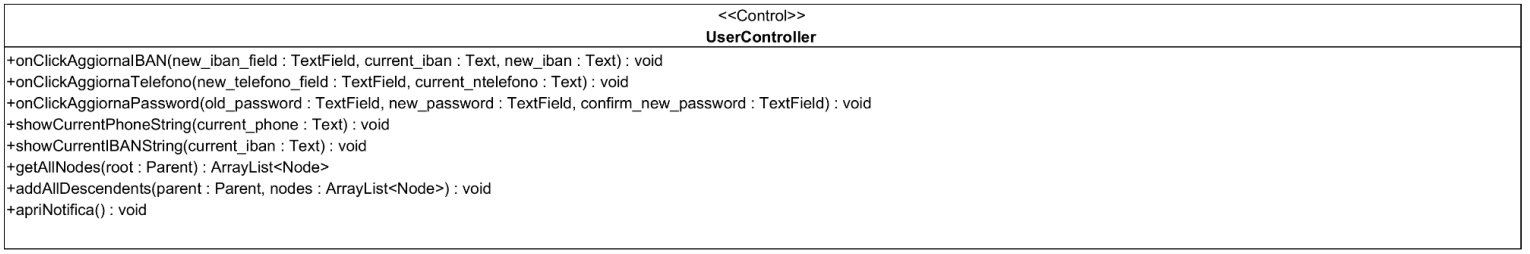






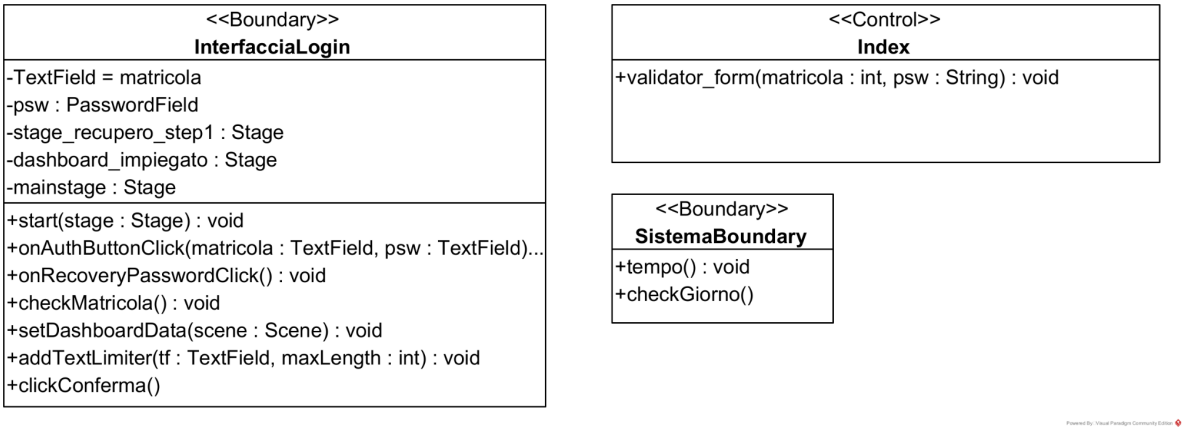


Roles.utente.controls



Indice

Roles



Indice