Help

Anno 2022/23

Object Design Document

A cura degli studenti:

- Ambrogi Federico Ennio
- Di Fede Andrea
- Mercurio Matteo
- Puccio Gabriele

HELP

Sommario

Introduzione

Scelte di implementazione e linee guida

Per la realizzazione del sistema è optato per la realizzazione di un'architettura di tipo Repository. L'architettura Repository è basata sulla comunicazione tra sottosistemi indipendenti e un sistema centralizzato chiamato Repository(deposito), non altro che una banca dati (DBMS). Più dettagliatamente, la comunicazione tra i vari sottosistemi può avvenire in due approcci: il primo approccio, garantisce come i dati possono essere immagazzinati in un database centralizzato ove tutti i vari sottosistemi possono consultare liberamente. Il secondo approccio prevede come ogni sottosistema possiede un proprio database e l'accesso a tali dati avviene tramite la comunicazione con altri sottosistemi. I vantaggi nell'utilizzare questo tipo di architettura sono il meccanismo di scambio di dati efficiente, scalabilità dei vari sottosistemi, architettura di tipo centralizzata soprattutto nell'ambito della sicurezza e gestione degli errori e inoltre ogni sottosistema non deve preoccuparsi di come i dati siano stati prodotti e/o elaborati.

Per la realizzazione di questo software si è deciso di utilizzare Eclipse IDE come ambiente di sviluppo, sia per la facilità e conoscenza pregressa del software ma anche per la facilità di gestione delle librerie. In aggiunta alle librerie proprie di Java per la gestione e implementazione dell'interfaccia grafica, ovvero Java Swing, è stato utilizzato un ulteriore tool noto come Window Builder.

Per quanto riguarda il servizio di hosting del software si è stato utilizzato XAMPP, non altro che una distribuzione Apache per creare, gestire oppure fare servizi di hosting per tecnologie open source, contente Apache HTTP Server e DBMS MariaDB.

Per interfacciarsi al database, si è usufruito di DBMS MariaDB fornito da XAMPP attraverso l'utilizzo dell'interfaccia PHPMyAdmin, ovvero un sistema che consente di gestire un database relazionale MySQL attraverso un qualsiasi browser come Chrome, Firefox etc.

Per quanto concerne l'invio dell'e-mail per il recupero della password, è stato deciso di utilizzare un indirizzo di posta elettronica fornito da Google denominato "recuperocredenziali.@gmail.com". Dopo aver configurato le impostazioni per questo server, viene avviata una sessione in cui l'indirizzo email mittente scelto effettua l'autenticazione in background e successivamente invia l'e-mail al destinatario specificato nel campo "email" durante il processo di recupero delle credenziali.

Mentre per quanto riguarda la creazione dei report (mensili, trimestrali, annuali) è stato deciso di utilizzare una libreria di java chiamata "BufferedWriter" compresa nel pacchetto "java.io", che permette di generare dei file passando una stringa formatta in modo

apposito per essere utilizzata dal buffer di output creato con la libreria definita in precedenza.

Packages

Per avere una migliore organizzazione dal punto di vista di sviluppo e organizzazione si è deciso di raggruppare le varie sezioni del codice in packages ognuno con una funzione ben specifica. A seguire l'elenco dei package utilizzati:

Package	Descrizione
Pannello di Controllo	Contiene tutti i pacchetti da installare sul sistema dell'Admin UE (Amministratore di sistema). Le interfacce sono state realizzare mediante il tool di supporto grafico Window Builder in supporto alle librerie di Java Swing, e le control sua appartenenza vengono utilizzate per svolgere diverse mansioni: • Gestire i Nuclei Familiari • Gestione dei Poli • Gestione delle Diocesi • Gestione della lista dei viveri • Visualizzare Errori • Creare dei Report • Visualizzare una previsione • Visualizzare i carichi presenti nel magazzino • Conferma la ricezione dei carichi, con Conferma Carico • Inoltrare i carichi
Polo	Contiene tutti i pacchetti da installare nel sistema del responsabile del polo. Le interfacce sono state realizzare mediante il tool di supporto grafico Window Builder in supporto alle librerie di Java Swing, e le control sua appartenenza vengono utilizzate per svolgere diverse mansioni: • Gestire i Nuclei Familiare • Segnalare Errori • Creare dei Report • Conferma la ricezione dei carichi, con Conferma Carico • Visualizzare i carichi presenti nel magazzino • Distribuire Cibo ai Nuclei Familiari
	Contiene tutti i pacchetti da installare nel sistema del responsabile della diocesi. Le interfacce sono state realizzare

Diocesi	mediante il tool di supporto grafico Window Builder in supporto alle librerie di Java Swing, e le control sua appartenenza vengono utilizzate per svolgere diverse mansioni: • Visualizzare i Poli • Visualizzare i carichi presenti nel magazzino • Inoltrare i carichi • Conferma la ricezione dei carichi, con Conferma Carico • Creare dei Report
Azienda	Contiene tutti i pacchetti da installare nel sistema del responsabile dell'azienda. Le interfacce sono state realizzare mediante il tool di supporto grafico Window Builder in supporto alle librerie di Java Swing, e le control sua appartenenza vengono utilizzate per svolgere diverse mansioni: • Distribuzione dei lotti di cibo all'UE • Gestire il catalogo • Gestire le richieste generate dal sistema
Autenticazione	Contiene tutti i pacchetti necessari affinchè l'utente posso effettuare l'autenticazione al sistema. Le interfacce sono state realizzare mediante il tool di supporto grafico Window Builder in supporto alle librerie di Java Swing, e le control sua appartenenza vengono utilizzate per svolgere diverse mansioni: • Autenticazione al sistema (Login) • Registrazione al sistema • Recupero delle credenziali tramite e-mail
Utils	Il pacchetto è specializzato per il sistema e quindi poco riutilizzabile in altri contesti. Più dettagliatamente contiene il pacchetto che si occupa della distribuzione dei carichi ai vari livelli del sistema (UE-Diocesi-Polo).
Entity	Contiene le classi che creano e modellano le entità di cui è necessaria la rappresentazione nel sistema.
Common	Contiene le classi che si occupano di gestire le interrogazioni con il DBMS e ulteriori classi come Account e la creazione dei report gestita ai vari livelli del sistema.
Help	Contiene la classe principale per l'avvio del sistema (Main).

Exceptions	Contiene tutte le classi che gestiscono le varie eccezioni che vengono sollevate in fase di run-time. Esempio: eccezioni generate per l'incoerenza delle date.
Icon	Contiene tutte le icone presenti nel sistema che vengono richiamate durante la fase di avvio del sistema.

Java Swing

È una libreria di Java utilizzata per la creazione di interfacce utente Desktop Java. Java Swing offre una serie di componenti grafichi (Componets), come pulsanti, caselle di testo, tabelle e molti altri, che possono essere utilizzati per le creazioni di interfacce interattive.

Java

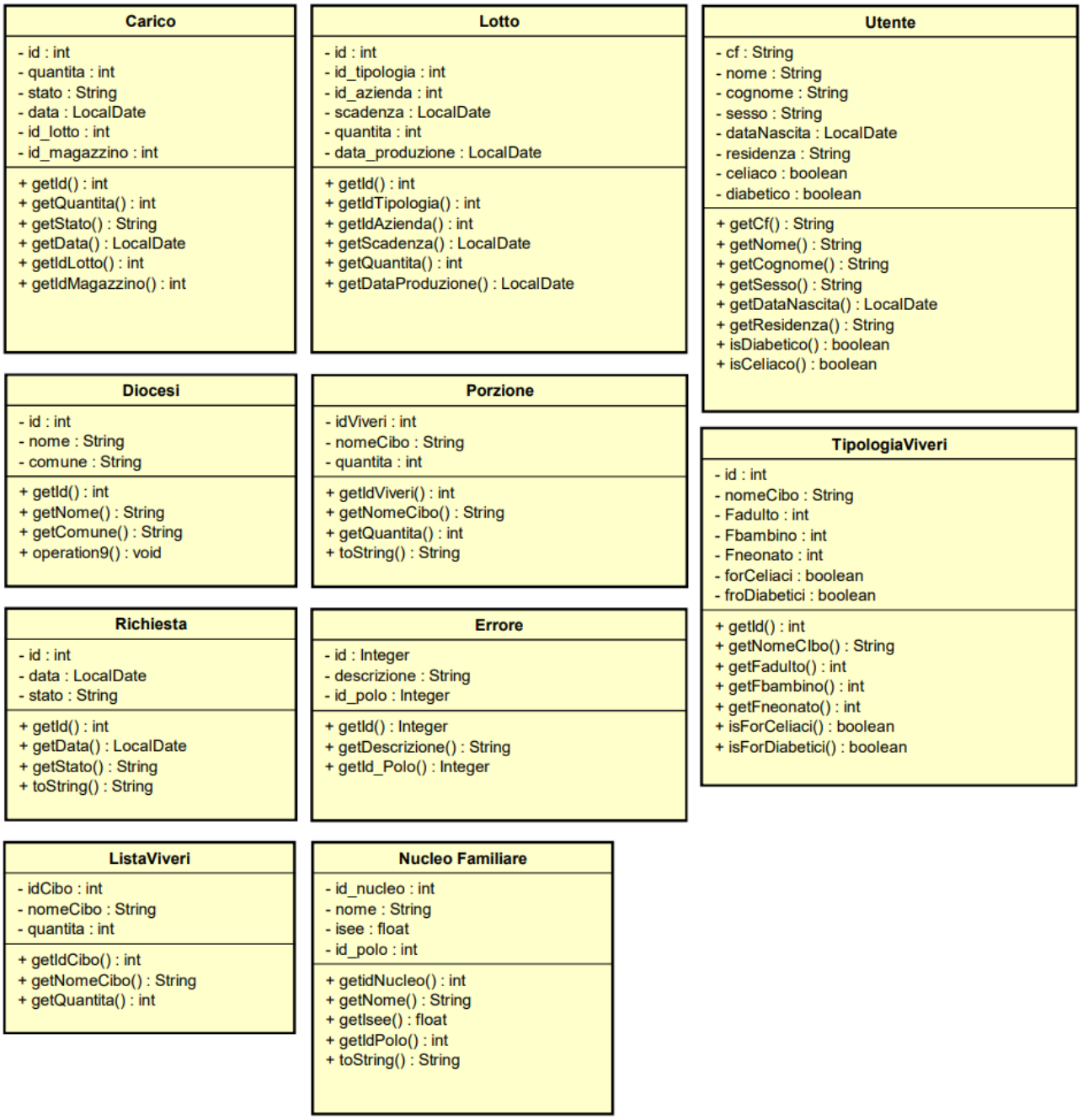
Contiene i package e le librerie standard di Java

- **MySQL-Connector:** È un driver Java che consente di comunicare con il DBMS MariaDB. Il file "mysql-connector.jar" contiene tutte le primitive necessarie per instaurare la comunicazione con il DB e inoltre tutte le primitive per gestire le interrogazioni (Query)per la manipolazione dei dati.
- **SQL:**
  - È un package utilizzato per gestire le comunicazioni con il DBMS, in particolare si occupa di gestire le query che permettono l'elaborazione delle varie informazioni estrapolate dalle interfacce grafiche (GUI) durante le fasi di inserimento, modifica ed eliminazione.
- **MAIL:**
  - È un package utilizzato per gestire le comunicazioni tra utente ed host gmail, in particolare viene utilizzato per inviare un'email contente un link per il reset della password.
- **Utils:**
  - Contiene le strutture dati utilizzate per la realizzazione del sistema.
- **Time:**
  - API per le date, il tempo, gli istanti e le durate.
- **JDatePicker:**
  - Libreria che fornisce un componente grafico per la selezione delle date per le applicazioni.
- **Swing:**
  - Fornisce un'implementazione di base con l'interfaccia JTable con l'aggiunta di ulteriori modelli che permettono di manipolare e formattare correttamente i dati presenti sulle tabelle.
- **AWT:**

- È una libreria standard di java che si occupa di gestire tutto ciò che riguarda le interfacce grafiche, come ad esempio gestire eventi, la creazione di finestre, controlli e molto altro.

Object Design UML

Entity

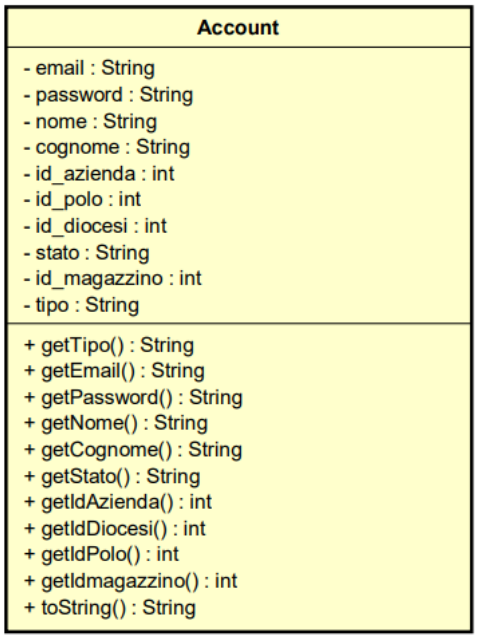


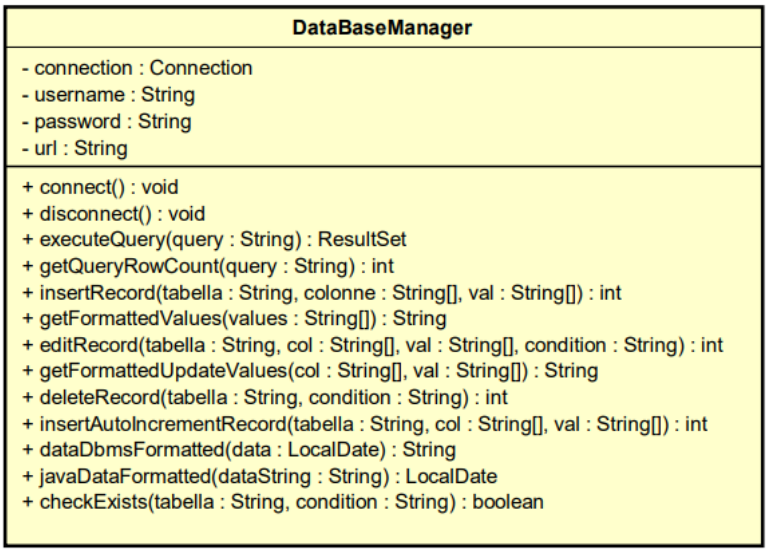
Utils

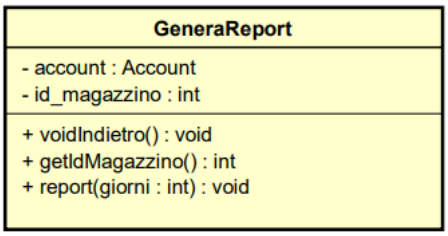
DIstributoreManager

+ createCaricoFromLotto(id_lotto : int, id_magazzino : int) : int
+ createCarico(c : Carico) : int
+ getEveryUtente() : ArrayList<Utente>
+ getEveryUtentePolo(id_polo : Integer) : ArrayList<Utente>
+ getEveryUtenteDiocesi(id_diocesi : Integer) : ArrayList<Utente>
+ getNneonati(utenti : ArrayList<Utente>) : int
+ getNbambini(bambini : ArrayList<Utente>) : int
+ getNadulti(adulti : ArrayList<Utente>) : int
+ getTipologiaViveri() : ArrayList<TipologiaViveri>
+ getPrevisioneGlobale() : ArrayList<Porzione>
+ getPrevisionePolo(id_polo : Integer) : ArrayList<Porzione>
+ getEveryCarico(id_magazzino : Integer) : ArrayList<Carico>
+ getEveryCarico(id_magazzino : Integer, stato : String) : ArrayList<Carico>
+ getCarico(id_magazzino : Integer) : ArrayList<Carico>
+ sendaCarico(id_carico : Integer, quantita : Integer, id_magazzinoDestinazione : Integer) : void
+ getCaricoRimanente(id_carico : Integer) : int
+ confermaSpedizione(id_carico : int) : void
+ getIdMagazzino(account : Account) : int
+ findTipologiaViveri(id_tipologia : int) : TipologiaViveri
+ getPersoneAdatteDiocesi(id_diocesi : int, id_tipologiaViveri : int) : ArrayList<Utente>
+ getPersoneAdatte(listaUtenti : ArrayList<Utente>, id_tipologiaViveri : int) : ArrayList<Utente>
+ doesMagazzinohave(id_magazzino : int, id_lotto : int) : boolean
+ getTipologiaCarico(id_carico : int) : int
+ getIdLottoFromCarico(id_carico : int) : int
+ getIdMagazzinoDiocesi(id_diocesi : int) : int
+ getIdMagazzinoPolo(id_polo : int) : int
+ distribuisciDiocesi(id_diocesi : int, id_carico : int) : void
+ getCaricoForDiocesi(id_diocesi : int) : ArrayList<Carico>
+ distribuisciPolo(id_polo : int, id_carico : int) : void
+ getCarichiForPoli(id_polo : int) : ArrayList<Carico>
+ disribuisciFamiglia(id_nucleo : Integer, id_carico : int) : void
+ getEveryUtenteFamiglia(id_nucleo : int) : ArrayList<Utente>
+ getCarichiForFamiglia(id_famiglia : int) : ArrayList<Carico>

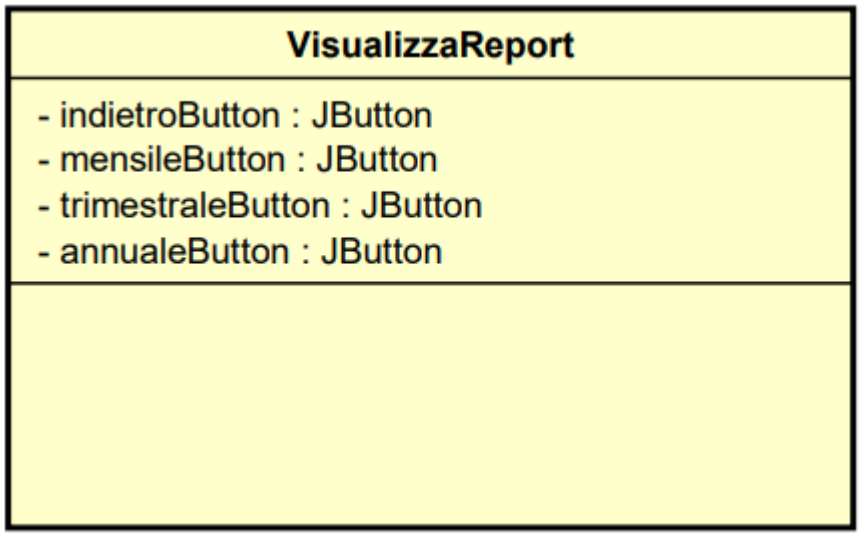
Common.Control







Common.Interface



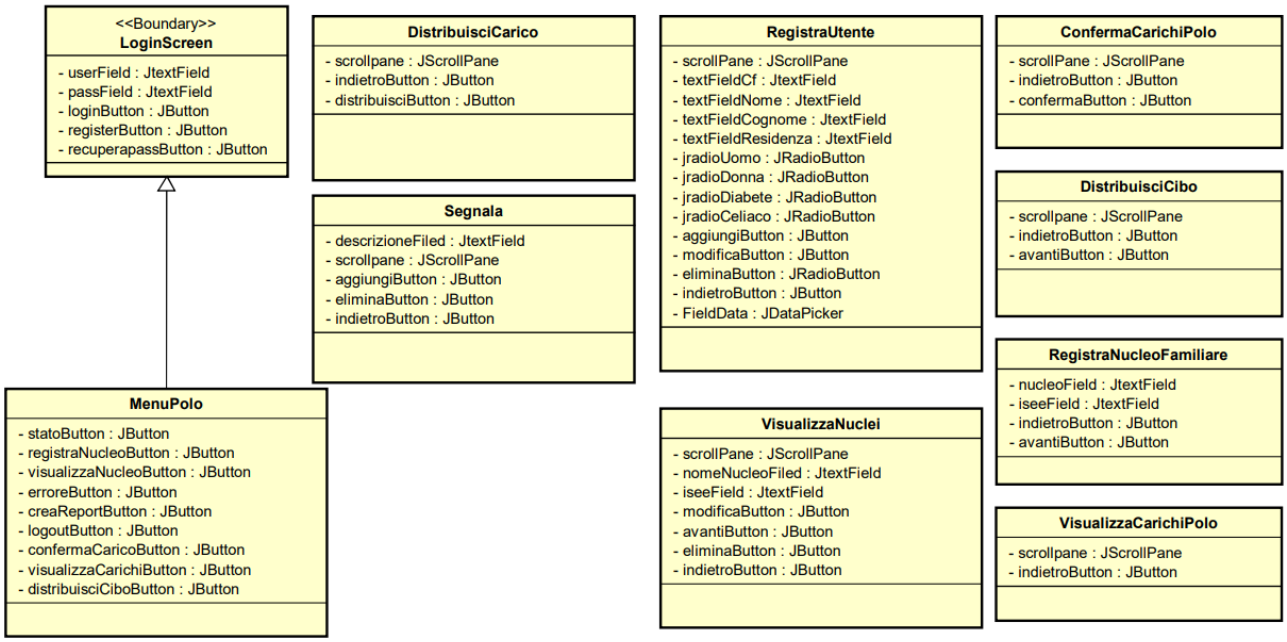
Polo.Control

PoloManager

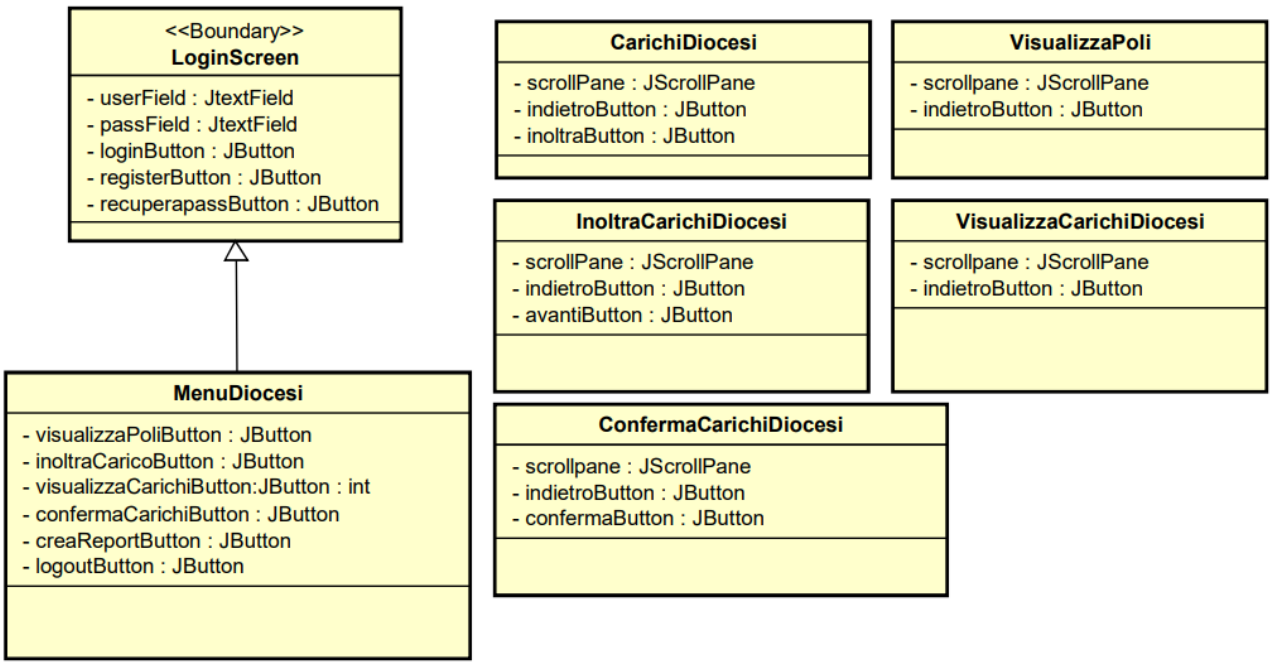
- id : Integer
- nome : String
- citta : String
- id_diocesi : Integer
- stato : String

+ getId() : Integer
+ getNome() : String
+ getCitta() : String
+ getId_Diocesi() : String
+ getStato() : String
+ createNFamialiare(nome : String, isee : String, id_polo : Integer) : int
+ createUtente(cf : String, nome : String, cognome : String, sesso : String, residenza : String, datanascita : LocalDate, id_nucleo : String) : Utente
+ updateUtente(cf : String, nome : String, cognome : String, sesso : String, residenza : String, datanascita : LocalDate, id_nucleo : String) : Utente
+ deleteUtente(cf : String) : void
+ getUtenti(id_nucleo : String) : ArrayList<Utente>
+ getNuclei(id_polo : String) : ArrayList<NucleoFamiliare>
+ deleteFromNucleo(id : Integer) : void
+ updateNucleoFamiliare(id : Integer, nome : String, isee : float) : void
+ deleteFromPolo(id : Integer) : void
+ addDisturbo(cf : String, id_disturbo : String) : void
+ getDIsturbiString(cf : String) : String
+ getAllNuceli() : ArrayList<NucleoFamiliare>
+ checkFutureDate(data : LocalDate) : void
+ createErrore(descrizione : String, id_polo : Integer) : int
+ getErrori(id_polo : Integer) : ArrayList<Errore>
+ deleteFromErrore(id : Integer) : void
+ getStato(id_polo : Integer) : String
+ setStato(id_polo : Integer, stato : String) : void
+ getListaFromDiocesi(id_diocesi : Integer) : ArrayList<PoloManager>
+ checkPolo(id_polo : Integer) : boolean

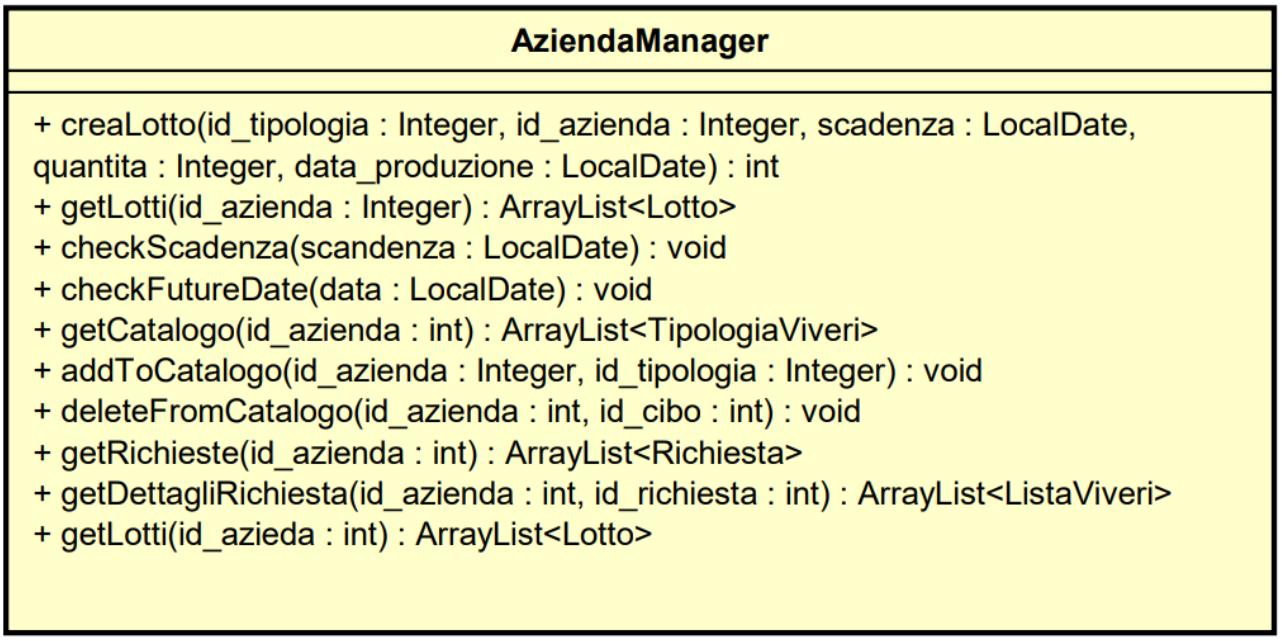
Polo.Interface



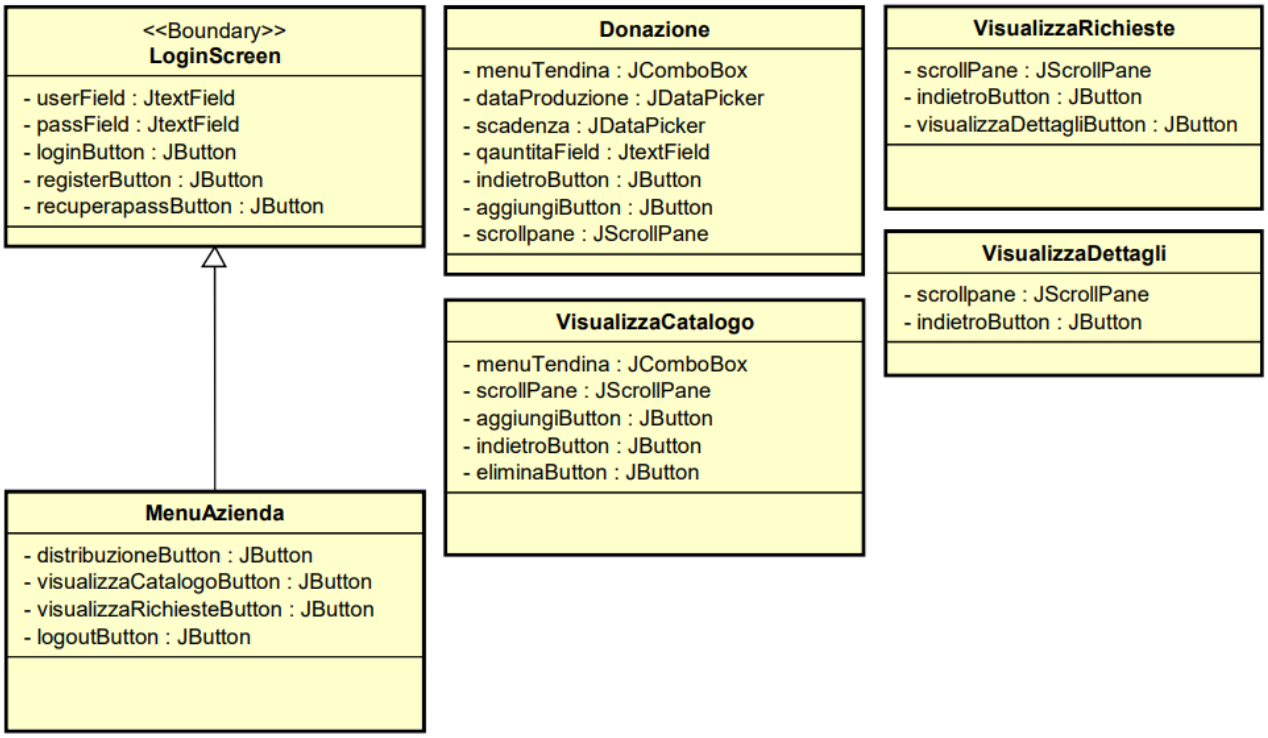
Diocesi.Interface



Azienda.Control



Azienda.Interface

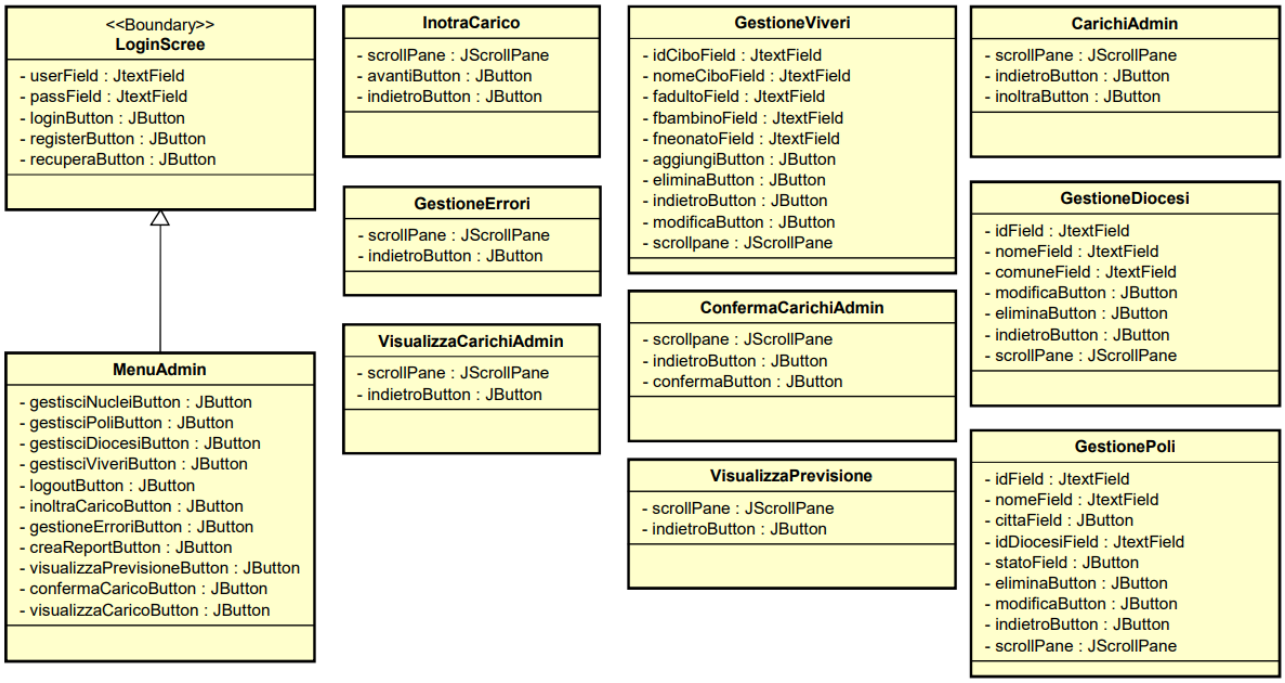


Pannello di Controllo.Control

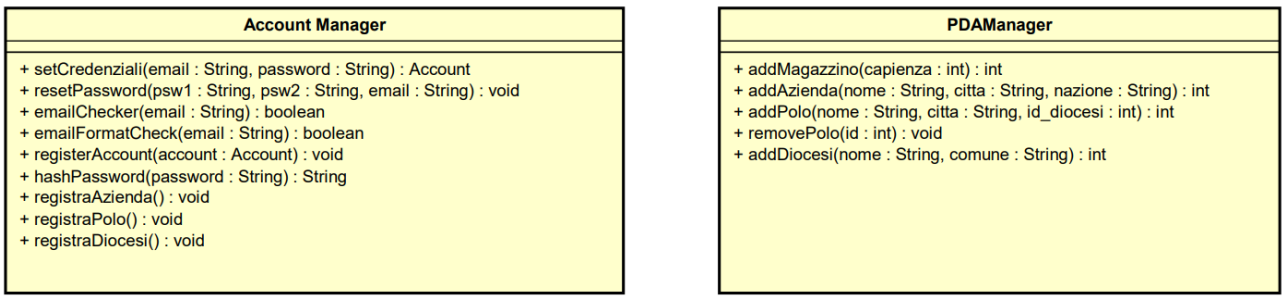
PannelloDiControlloManager

+ creaCibo(id : Integer, nomeCibo : String, fadulto : Integer, fbambino : Integer, fneonato : Integer) : int
+ getListaViveri(id_azienda : int) : ArrayList<TipologiaViveri>
+ deleteFromTipologiaViveri(id_cibo : Integer) : void
+ updateListaViveri(id : Integer, nomeCibo : String, fadulto : Integer, fbambino : Integer, fneonato : Integer) : void
+ getListaDiocesi(id : int) : ArrayList<Diocesi>
+ updateListaDiocesi(id : Integer, nome : String, comune : String) : void
+ deleteFromDiocesi(id : Integer) : void
+ getListaPoli() : ArrayList<PoloManager>
+ deleteFromPolo(id : Integer) : void
+ updateListaPoli(id : Integer, nome : String, citta : String, id_diocesi : Integer, stato : String) : void
+ getListaErrori() : ArrayList<Errore>
+ creaRichiesta() : void

Pannello di Controllo.Interface



Autenticazione.Control



Autenticazione.Interface