UNIVERSITÀ
DEGLI STUDI
DI PALERMO

Object Design Document

A.A 2023/2024

Nome del progetto: OnPoint

Baiamonte Gabriele

Pennavaria Gaetano

Ribisi Simone

Rubino Riccardo

Indice

1 Introduzione

1.1 Compromessi nella progettazione degli oggetti

Per la realizzazione del Sistema è stato scelto un approccio modulare per promuovere la scalabilità, facilitare l'implementazione e semplificare la gestione di eventuali problemi. In particolare, è stata adottata un'architettura di tipo Repository, che assicura il disaccoppiamento dei sottosistemi. Questi sottosistemi possono comunicare solo con il sottosistema di Storage, ma non tra di loro. I sottosistemi sono costituiti da un'interfaccia utente che interagisce esclusivamente con il controllore sottostante. Questo controllore contiene la logica del programma e gestisce le richieste indirizzate al DBMSDaemon, che si occupa delle comunicazioni con il nodo di Storage. L'interfaccia utente, il controllore e il DBMSDaemon di ciascun sottosistema si trovano sullo stesso nodo, mentre il sottosistema di Storage è situato su un nodo separato.

1.2 Nozioni per la documentazione delle interfacce

Per quanto concerne le interfacce grafiche, si è optato per l'utilizzo del pacchetto di librerie JavaFX, che ci ha consentito di creare le interfacce grafiche mediante documenti di markup FXML. JavaFX è un pacchetto di librerie basato su componenti, il che permette un'elevata riusabilità del codice e una maggiore leggibilità durante la fase di scrittura.

In particolare, per lo sviluppo delle interfacce grafiche del Sistema, è stato impiegato JavaFX Scene Builder, un tool che consente la creazione di documenti FXML attraverso un sistema di drag-and-drop.

1.3 Nozioni per la documentazione dei DBMS

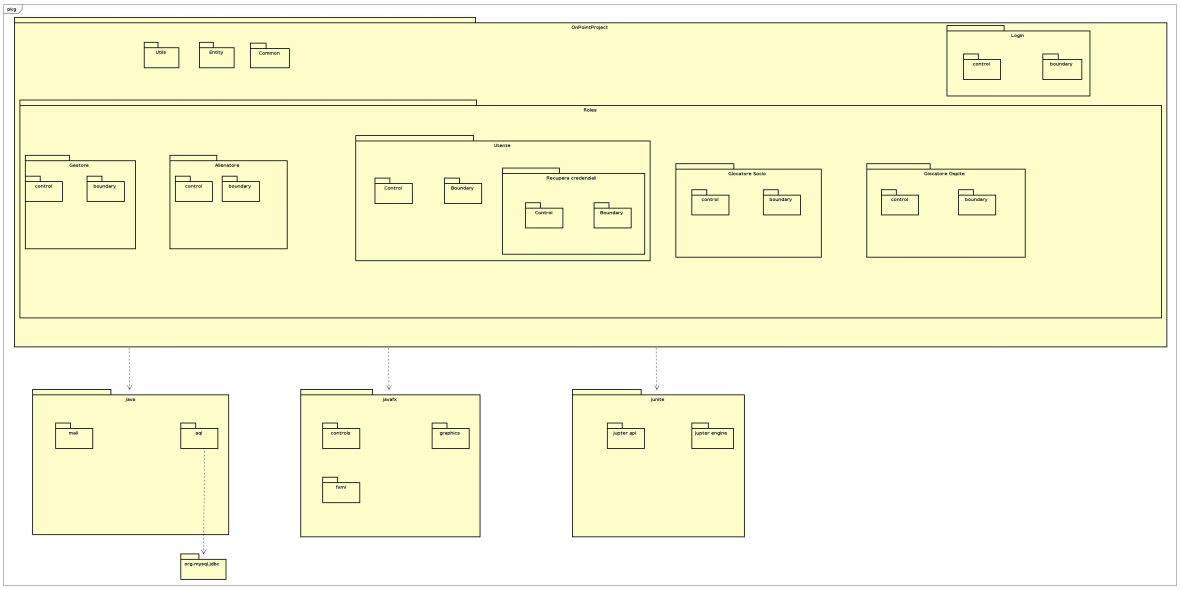
Per quanto riguarda la gestione dei database, abbiamo optato per l'utilizzo di MySQL come sistema di gestione di database (DBMS). Per la comunicazione tra il nostro sistema e il database, abbiamo scelto di adottare MySQL Connector, che è un driver JDBC di tipo 4. La designazione "tipo 4" indica che il driver JDBC è una pura implementazione di Java del protocollo MySQL e non dipende dalle librerie del client MySQL. JDBC è un connettore e

un driver per database che consente l'accesso e la gestione della persistenza dei dati su basi di dati da qualsiasi programma scritto in Java, indipendentemente dal tipo di DBMS utilizzato.

1.4 Nozioni per la documentazione dell'invio mail

Per quanto concerne l'invio della posta elettronica abbiamo optato per utilizzare la mail fornita da libero. Per l'invio delle mail, come nel caso d'uso Recupera Credenziali in cui inviamo la password che corrisponde all'account che ha richiesto di recuperare le credenziali, abbiamo usato l'API di javamail, che rende facile ed agevole la scrittura di codice Java per gestire l'invio automatico di e-mail.

2 Packages



3 On point project

Rappresenta il package principale del progetto.

3.1 Roles

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

3.1.1 Utente

Contiene tutti quei pacchetti da installare sulle postazioni delle varie tipologie di Utenti (quali allenatori, gestore, giocatori soci e giocatori ospiti). Il sistema stesso sarà in grado di smistare alla corrispettiva dash board in base all'autenticazione. La "Boundary" contiene le varie classi che sono associate alle interfacce utente realizzate con FXML, mentre "controls" gestisce tutta la parte logica delle funzionalità dei vari tipi di utente.

3.2 Utente.Recupero credenziali

Contiene tutti quei pacchetti da installare sulle postazioni di tutti gli Utenti (indipendentemente dal ruolo occupato), la "Boundary" contiene le varie classi che sono associate alle interfacce utente realizzate con FXML, mentre "controls" gestisce tutta la parte logica che riguarda il recupero delle credenziali dell'utente .

3.3 Gestore

"Boundary" contiene tutte le classi associate alle interfacce utente realizzate con FXML, mentre "controls" si occupa di tutta la parte logica delle funzionalità del ruolo "Gestore".

3.4 Allenatore

"Boundary" contiene tutte le classi associate alle interfacce utente realizzate con FXML, mentre "controls" si occupa di tutta la parte logica delle funzionalità del ruolo "Allenatore".

3.5 Giocatore Socio

“Boundary” contiene tutte le classi associate alle interfacce utente realizzate con FXML, mentre “controls” si occupa di tutta la parte logica delle funzionalità del ruolo “Giocatore Socio”.

3.6 Giocatore Ospite

“Boundary” contiene tutte le classi associate alle interfacce utente realizzate con FXML, mentre “controls” si occupa di tutta la parte logica delle funzionalità del ruolo “Giocatore Ospite”.

3.7 Login

Contiene le classi che servono per effettuare il login, differenziando la tipologia di utente.

3.8 Entity

Contiene le classi che servono per modellare l’entità di cui è necessaria la rappresentazione nel sistema

3.9 Utils

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

3.10 Java

Contiene i package e le librerie standard di Java:

3.10.1 Mail

La libreria JavaMail offre un sistema versatile e compatibile con diverse piattaforme e protocolli per lo sviluppo di applicazioni di posta e messaggistica. Consente di utilizzare vari protocolli di posta elettronica come POP/POP3, SMTP e IMAP.

3.10.2 Sql

L'API fornisce un modo standardizzato per accedere e manipolare i dati memorizzati in una sorgente dati, come ad esempio un database relazionale. Questo significa poter gestire la comunicazione con il sistema di gestione del database (DBMS), creare connessioni e eseguire interrogazioni senza dover scrivere codice specifico per ogni singolo DBMS utilizzato.

3.11 Javafx

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

3.11.1 Controls

JavaFX Controls rappresenta un insieme completo di strumenti grafici integrati nella piattaforma JavaFX, progettati appositamente per costruire interfacce utente sofisticate e coinvolgenti nelle applicazioni destinate al desktop. Questi controlli consentono agli sviluppatori di incorporare elementi interattivi, come pulsanti e altro ancora, senza dover partire da zero. Ogni controllo è dotato di una serie di opzioni di personalizzazione, tra cui proprietà, metodi ed eventi, che permettono di adattare l'aspetto e il comportamento per soddisfare le esigenze specifiche dell'applicazione. Grazie alla vasta gamma di controlli disponibili e alla loro flessibilità, JavaFX semplifica notevolmente lo sviluppo di interfacce utente moderne e intuitive per le applicazioni desktop Java.

3.11.2 Graphics

JavaFX Graphics costituisce un insieme di strumenti integrati nell'ambito della piattaforma JavaFX, finalizzati alla creazione e alla gestione di elementi grafici avanzati nelle applicazioni

destinate al desktop. Questo set di funzionalità abilita gli sviluppatori a disegnare forme, incorporare immagini, testo e altri elementi direttamente sulla scena dell'applicazione. Ogni aspetto di queste funzionalità è altamente personalizzabile, consentendo agli sviluppatori di creare esperienze visive coinvolgenti e perfettamente adattate alle necessità specifiche dell'applicazione.

3.11.3 Fxml

JavaFX FXML è un linguaggio di marcatura progettato appositamente per definire l'aspetto e la struttura delle interfacce utente nelle applicazioni JavaFX. Questo linguaggio consente agli sviluppatori di specificare in modo dichiarativo, e separato dalla logica di programmazione, la disposizione dei componenti grafici, i loro collegamenti agli eventi e altre proprietà visive utilizzando un formato basato su XML. Questo approccio promuove una suddivisione chiara dei compiti tra i programmatori, che si occupano della logica applicativa, e i progettisti, responsabili dell'estetica e dell'usabilità dell'interfaccia utente. Grazie a JavaFX FXML, le applicazioni possono godere di una struttura più organizzata e di una manutenibilità migliorata, facilitando l'aggiornamento e la riutilizzazione dei componenti grafici.

3.12 JUnite

JUnit si presenta come un framework essenziale per i test unitari in Java, fornendo agli sviluppatori la possibilità di creare e lanciare test automatici per controllare il funzionamento di porzioni specifiche di codice, come metodi o classi. Questo strumento agevola notevolmente il processo di verifica delle funzionalità del software, consentendo una validazione rapida ed efficiente delle singole unità di codice.

3.13 Jupyter engine

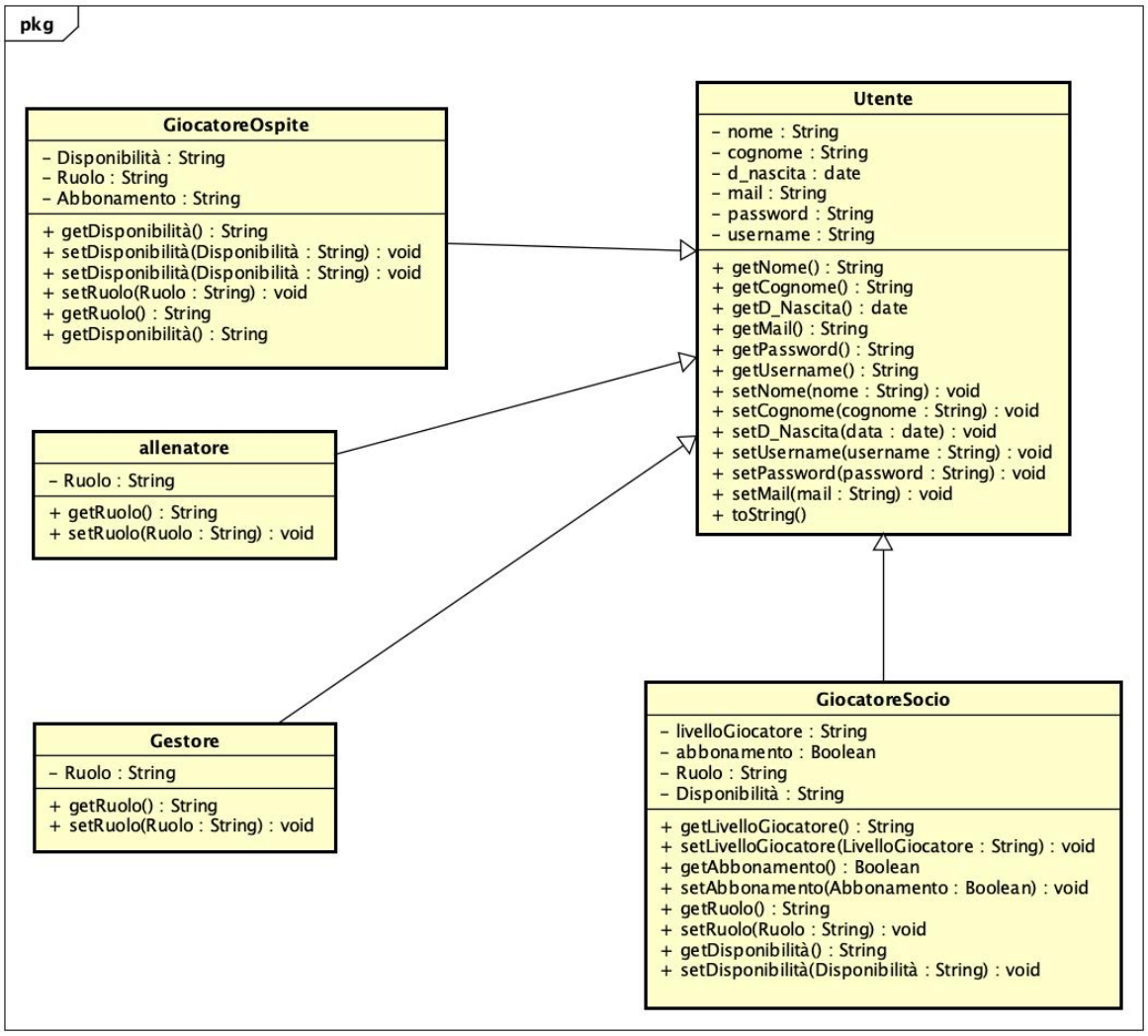
JUnit Jupiter Engine Maven Dependency è una dipendenza Maven per integrare il motore JUnit Jupiter nei progetti Java. JUnit è un framework di test unitari per il linguaggio di programmazione Java. Il motore JUnit Jupiter è la parte di JUnit 5 che supporta la nuova

programmazione model-driven e fornisce un'API più ricca e flessibile rispetto alle versioni precedenti di JUnit.

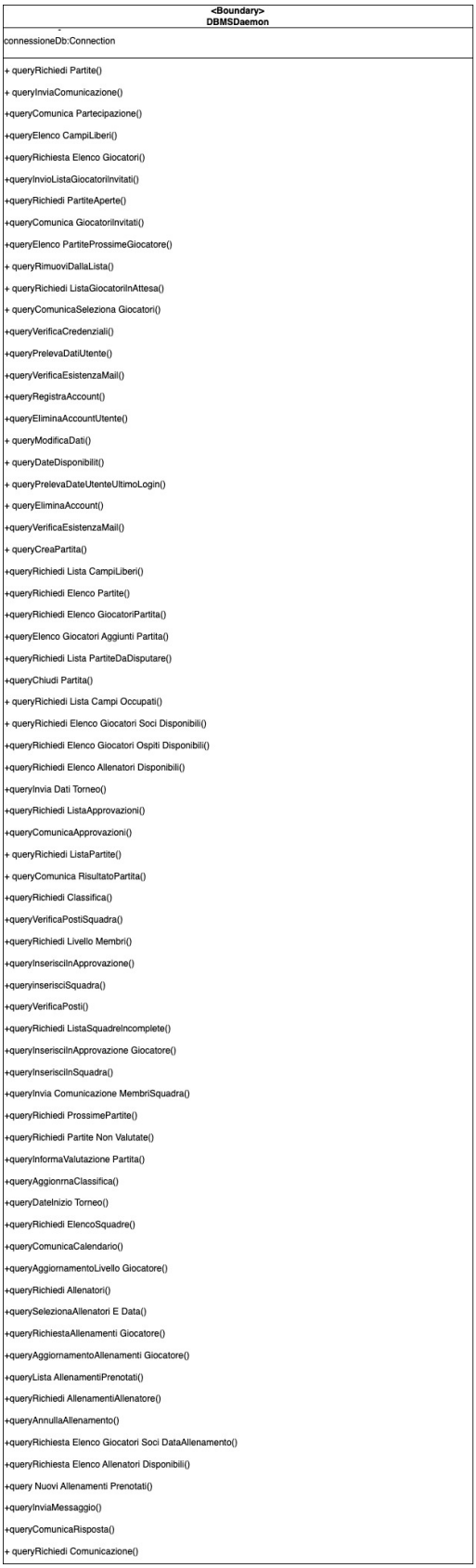
Integrando la dipendenza Maven del motore JUnit Jupiter nel progetto, è possibile utilizzare le funzionalità avanzate di JUnit 5 per scrivere e eseguire test unitari in modo più efficace e flessibile. Ciò include l'uso di annotazioni come @Test, @BeforeEach, @AfterEach, @BeforeAll, @AfterAll e molte altre per definire il comportamento dei test e organizzare il codice di inizializzazione e pulizia.

4 Object Design UML

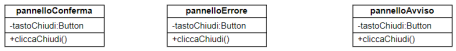
4.1 Entity



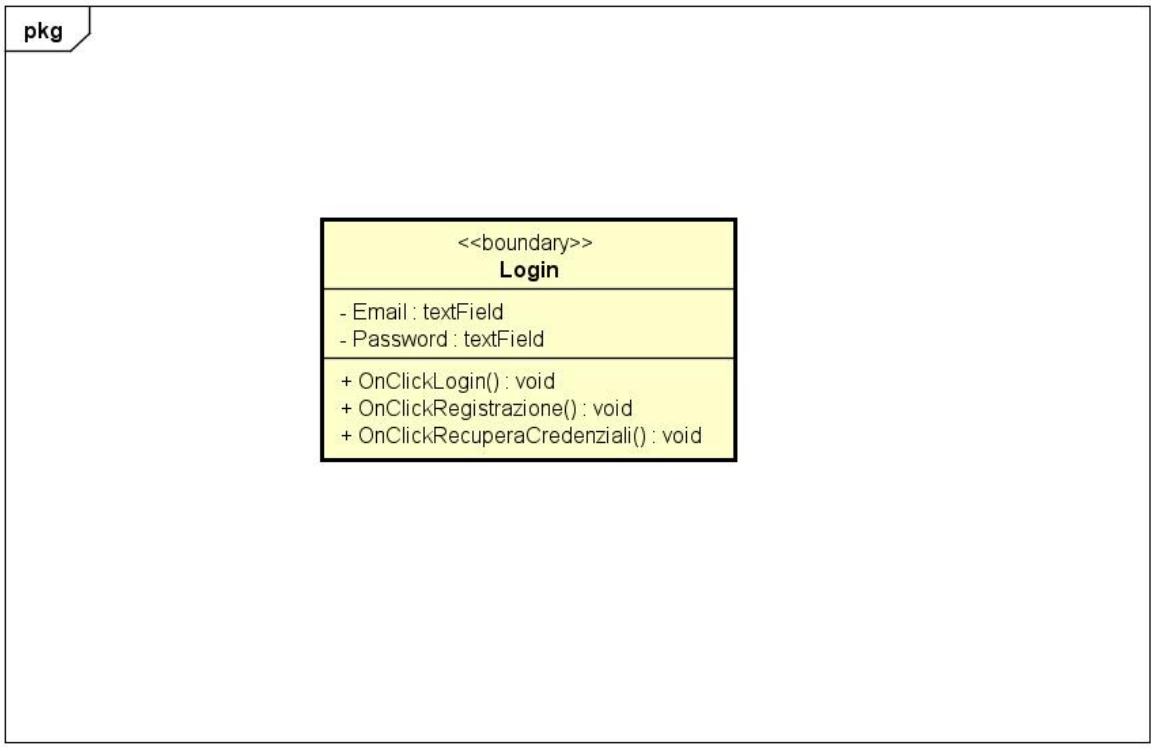
4.2 Utils


<Boundary>
DBMSDaemon
connessioneDb:Connection
+ queryRichiedi Partite()
+ queryInviaComunicazione()
+queryComunica Partecipazione()
+queryElenco CampiLiberi()
+queryRichiesta Elenco Giocatori()
+queryInvioListaGiocatoriInvitati()
+queryRichiedi PartiteAperte()
+queryComunica GiocatoriInvitati()
+queryElenco PartiteProssimeGiocatore()
+ queryRimuoviDallaLista()
+ queryRichiedi ListaGiocatoriInAttesa()
+ queryComunicaSeleziona Giocatori()
+queryVerificaCredenziali()
+queryPrelevaDatiUtente()
+queryVerificaEsistenzaMail()
+queryRegistraAccount()
+queryEliminaAccountUtente()
+ queryModificaDati()
+ queryDateDisponibilit()
+ queryPrelevaDateUtenteUltimoLogin()
+ queryEliminaAccount()
+queryVerificaEsistenzaMail()
+ queryCreaPartita()
+queryRichiedi Lista CampiLiberi()
+queryRichiedi Elenco Partite()
+queryRichiedi Elenco GiocatoriPartita()
+queryElenco Giocatori Aggiunti Partita()
+queryRichiedi Lista PartiteDaDisputare()
+queryChiudi Partita()
+ queryRichiedi Lista Campi Occupati()
+ queryRichiedi Elenco Giocatori Soci Disponibili()
+queryRichiedi Elenco Giocatori Ospiti Disponibili()
+queryRichiedi Elenco Allenatori Disponibili()
+queryInvia Dati Torneo()
+queryRichiedi ListaApprovazioni()
+queryComunicaApprovazioni()
+ queryRichiedi ListaPartite()
+ queryComunica RisultatoPartita()
+queryRichiedi Classifica()
+queryVerificaPostiSquadra()
+queryRichiedi Livello Membri()
+queryInserisciInApprovazione()
+queryinserisciSquadra()
+queryVerificaPosti()
+queryRichiedi ListaSquadreIncomplete()
+queryInserisciInApprovazione Giocatore()
+queryInserisciInSquadra()
+queryInvia Comunicazione MembriSquadra()
+queryRichiedi ProssimePartite()
+queryRichiedi Partite Non Valutate()
+queryInformaValutazione Partita()
+queryAggionrnaClassifica()
+queryDateInizio Torneo()
+queryRichiedi ElencoSquadre()
+queryComunicaCalendario()
+queryAggiornamentoLivello Giocatore()
+queryRichiedi Allenatori()
+querySelezionaAllenatori E Data()
+queryRichiestaAllenamenti Giocatore()
+queryAggiornamentoAllenamenti Giocatore()
+queryLista AllenamentiPrenotati()
+queryRichiedi AllenamentiAllenatore()
+queryAnnullaAllenamento()
+queryRichiesta Elenco Giocatori Soci DataAllenamento()
+queryRichiesta Elenco Allenatori Disponibili()
+query Nuovi Allenamenti Prenotati()
+queryInviaMessaggio()
+queryComunicaRisposta()
+ queryRichiedi Comunicazione()

4.3 Commons

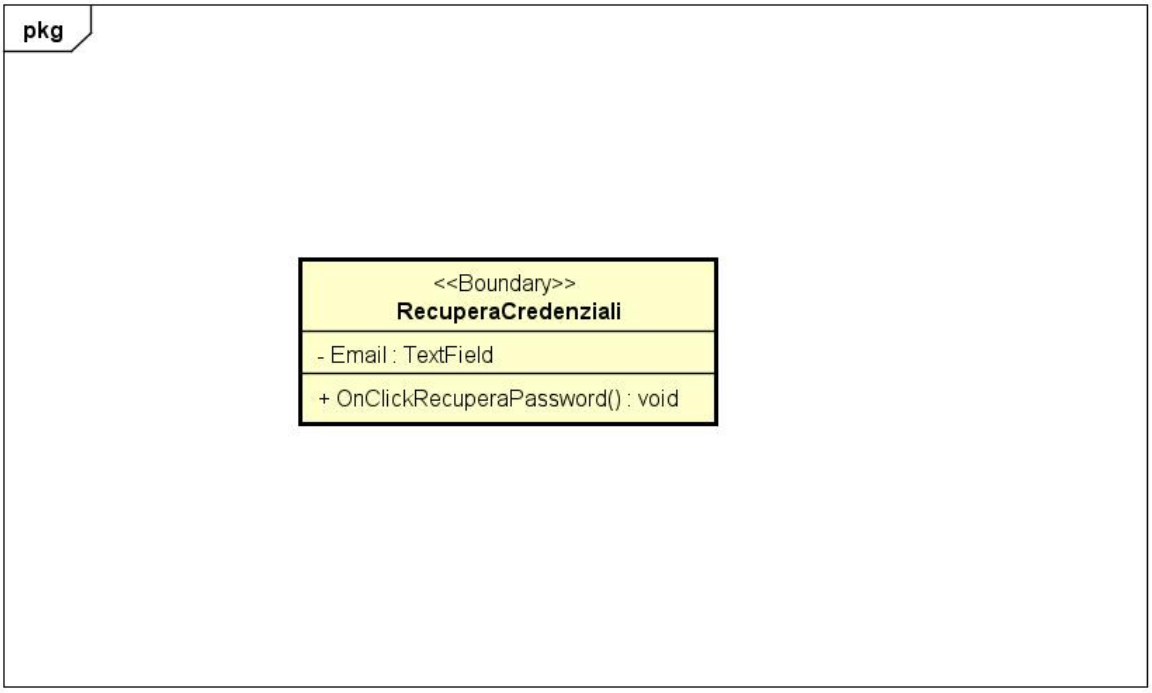


4.4 LOGIN.INTERFACES

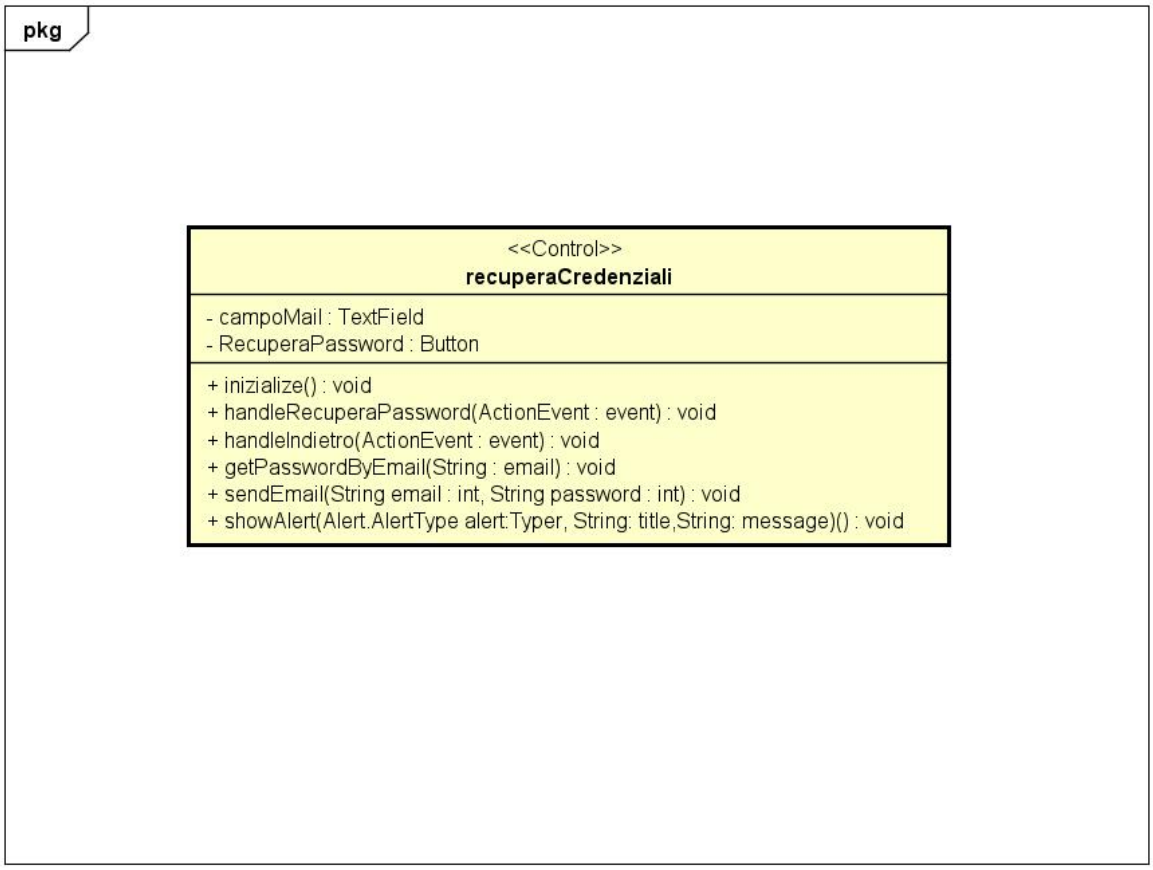


4.5 LOGIN.CONTROLS

4.6 RecuperaCredenziali.INTERFACES



4.7 RecuperaCredenziali.CONTROLS



4.8 Roles.Gestore.Interfaces

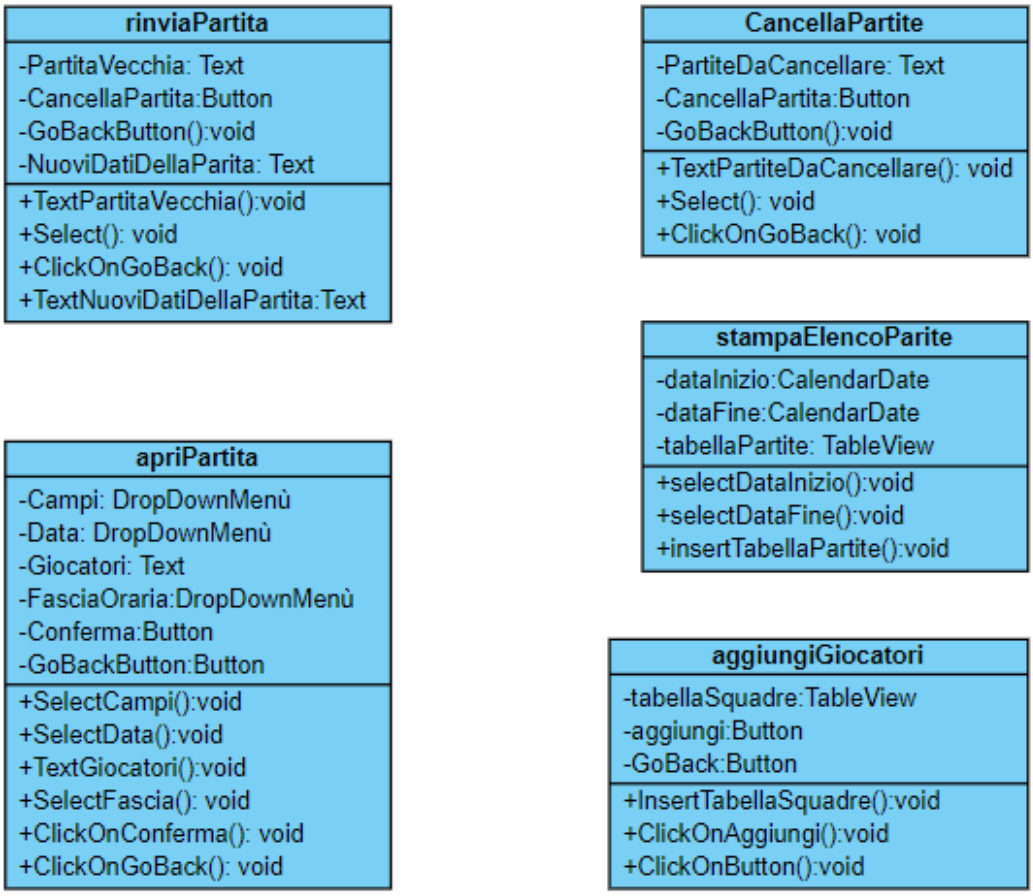
PortaleModificaDatiGestore
-modificareDati:Button -backButton:ImageView -logoutButton:ImageView
+clickOnModificareDati():void +handleBackButton(ActionEvent:event):void +handleLogoutButton(ActionEvent:event):void

ModificaDatiGestore
-emailModifica:TextField -passwordModifica:TextField -inviaConferma:Button
+clickOnInviaconferma():void +TextOnEmailModifica():void +TextOnPasswordModifica():void

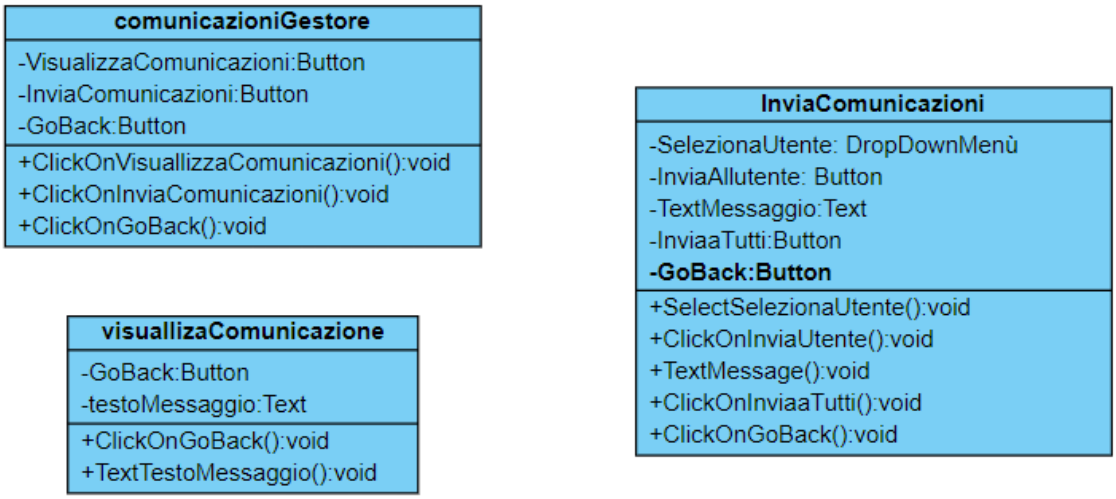
RegistraAllenatoreGestore
-emailAllenatore:TextField -passwordAllenatore:TextField -nomeAlleantore:TextField -cognomeAllenatore:TextField -usernameAllenatore:TextField -dataDiNascita:TextField -registra:Button -backButton:ImageView
+clickOnRegistra():void +TextOnEmailAllenatore():void +TextOnPasswordAllenatore():void +clickOnBackButton():void +TextOnNomeAllenatore():void +TextOnCognomeAllenatore():void +TextOnUsernameAlleantore():void +TextOnDataDiNascitaAllenatore():vodi

HomepageGestore
-buttonGestionePartite:Button -buttonRegistraAccountAllenatore:Button -buttonComunicazioni:Button -buttonTornei:Button -buttonGestioneUtente:Button -backButton:ImageView
+clickOnGestionePartite():void +clickOnRegistraAccountAllenatore():void +clickOnComunicazioni():void +clickOnBackButton():void +clickOnTornei():void +clickOnPortaleUtente():void

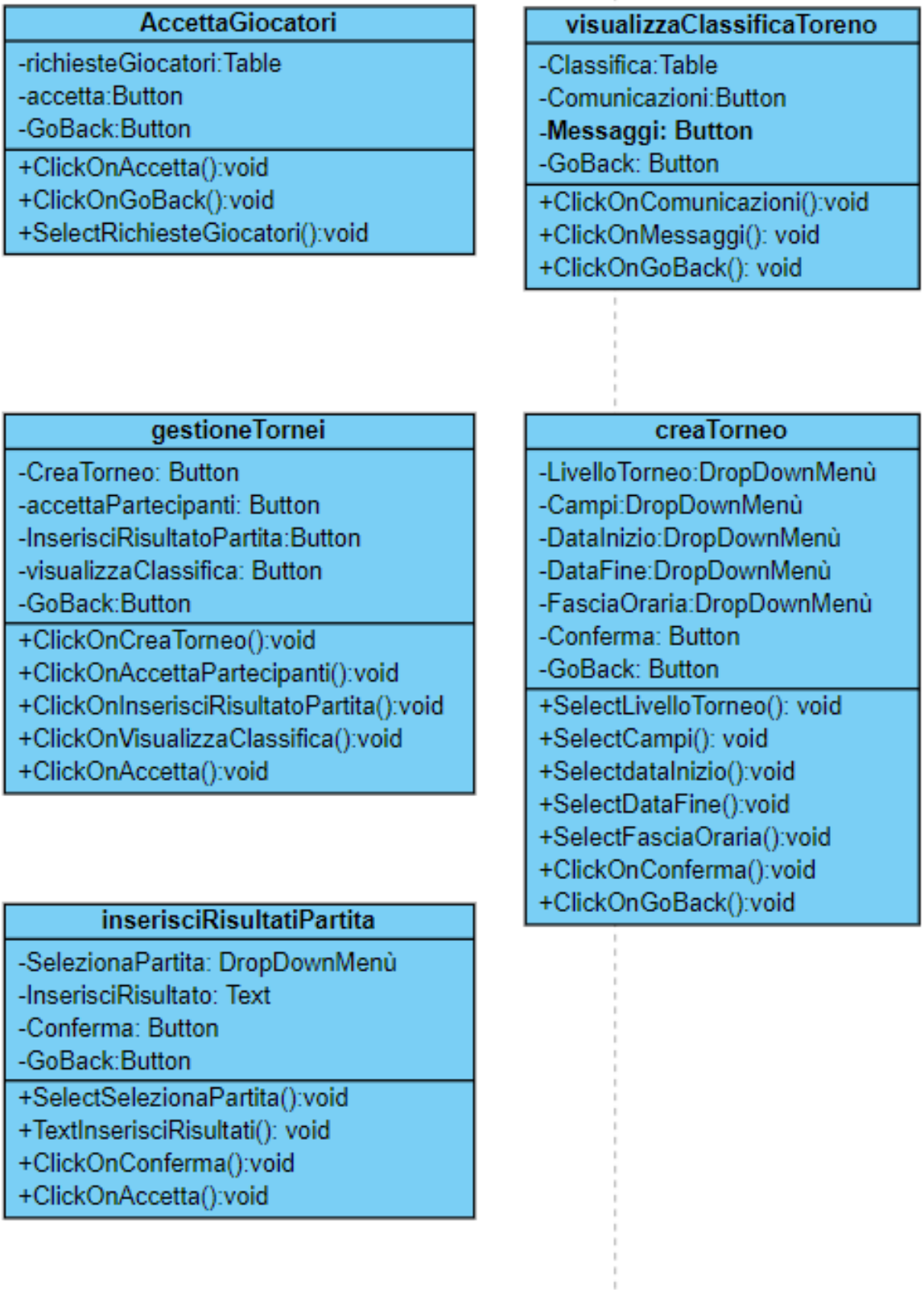
4.8.1 Roles.Gestore.Interfaces.Partite



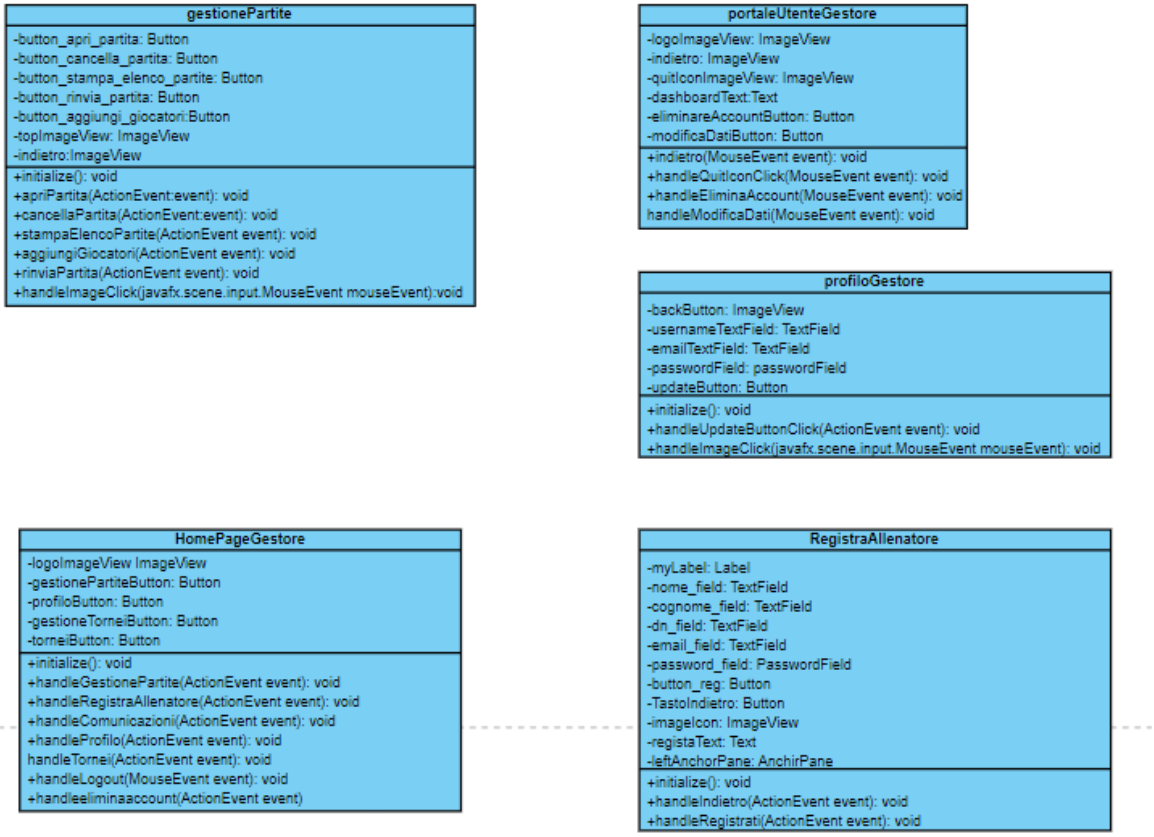
4.8.2 Roles.Gestore.Interfaces.Comunicazione



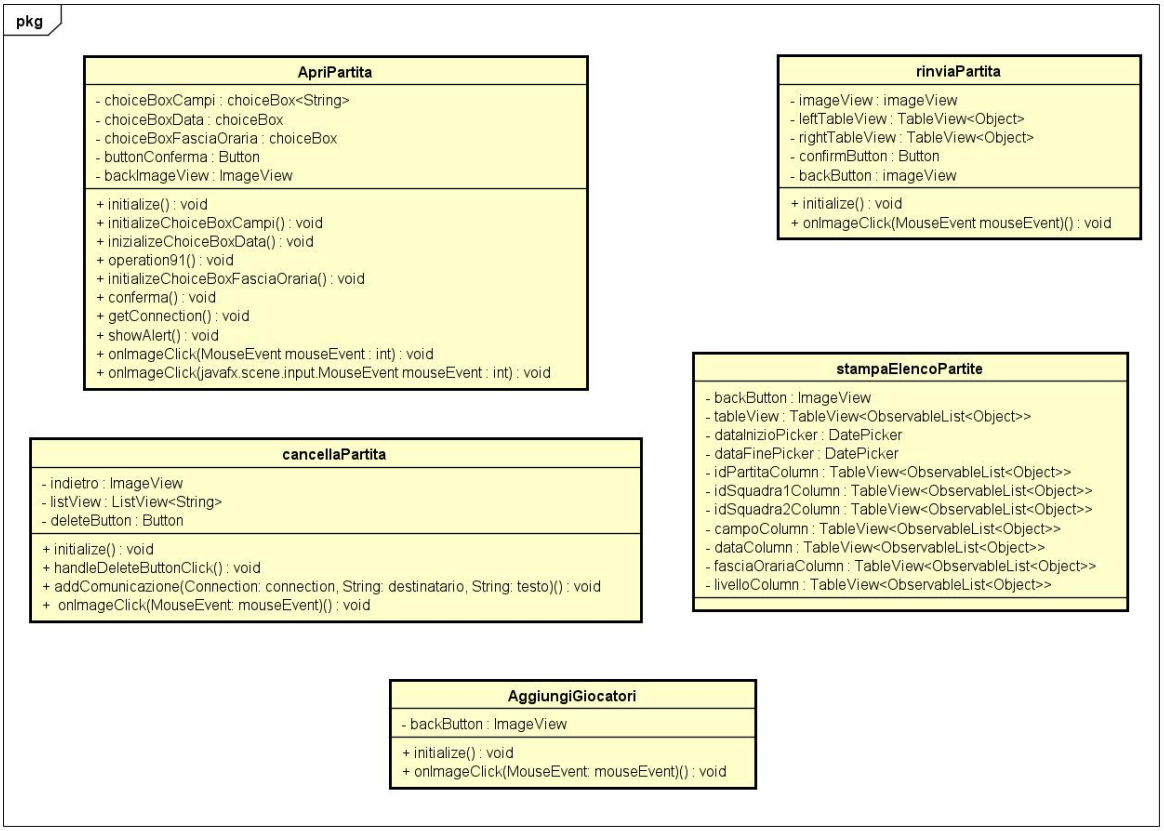
Roles.Gestore.Interfaces.Torneo



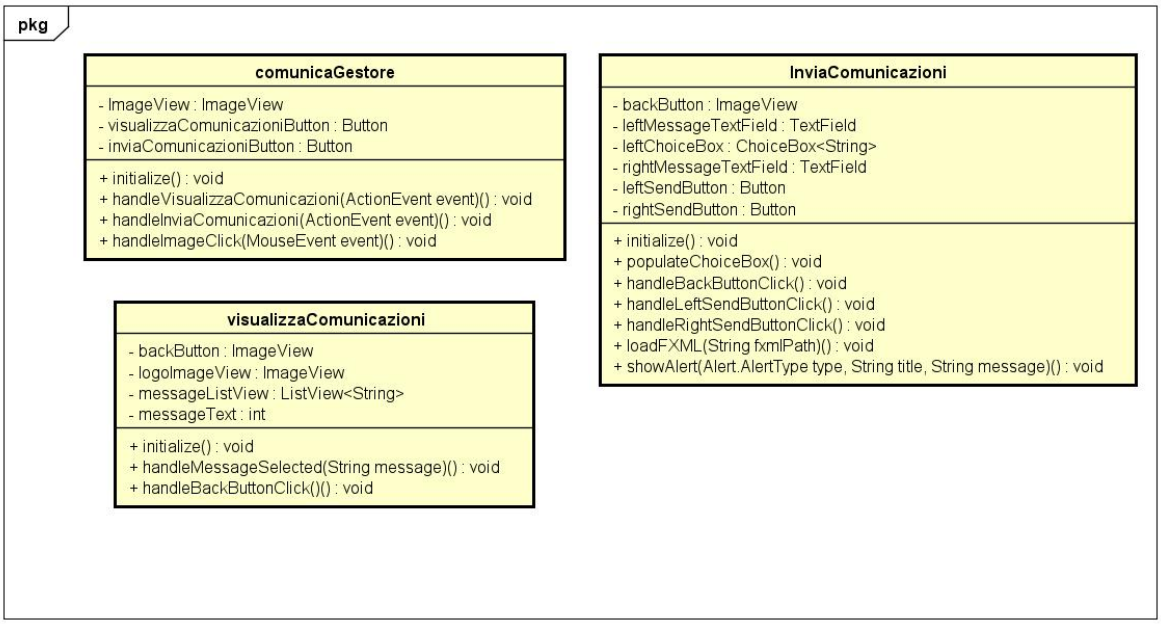
4.9 Roles.Gestore.Control



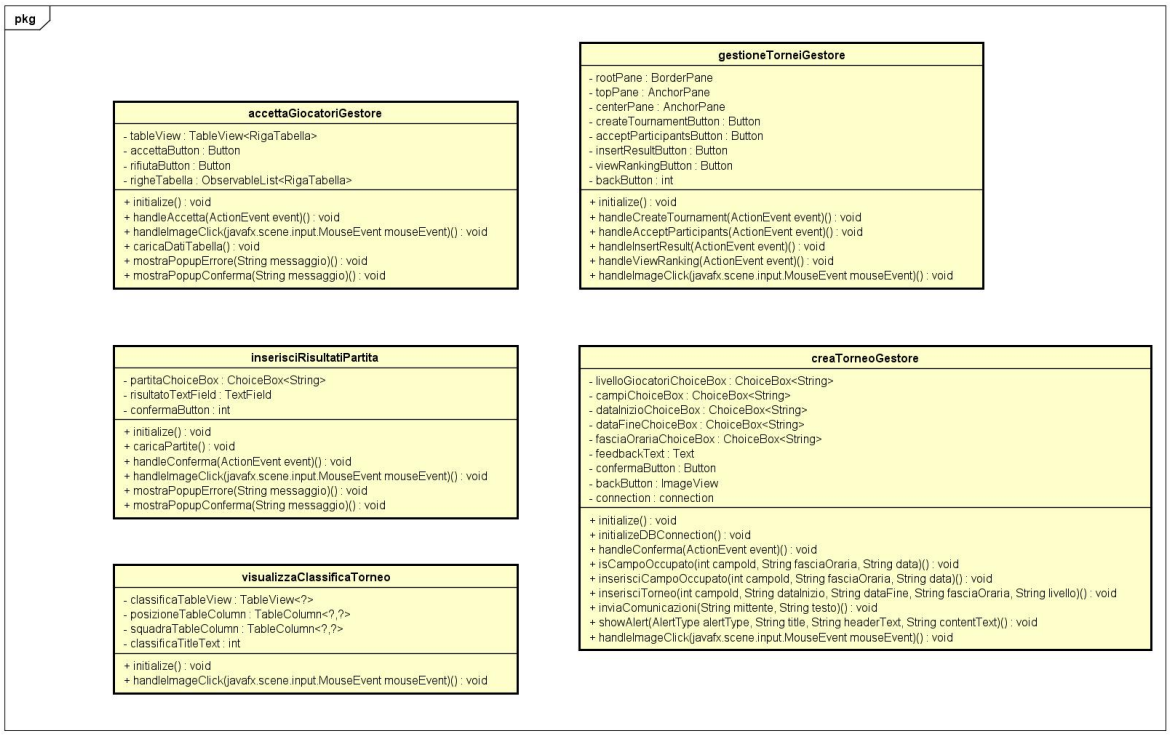
4.9.1 Roles.Gestore.Control.Partite



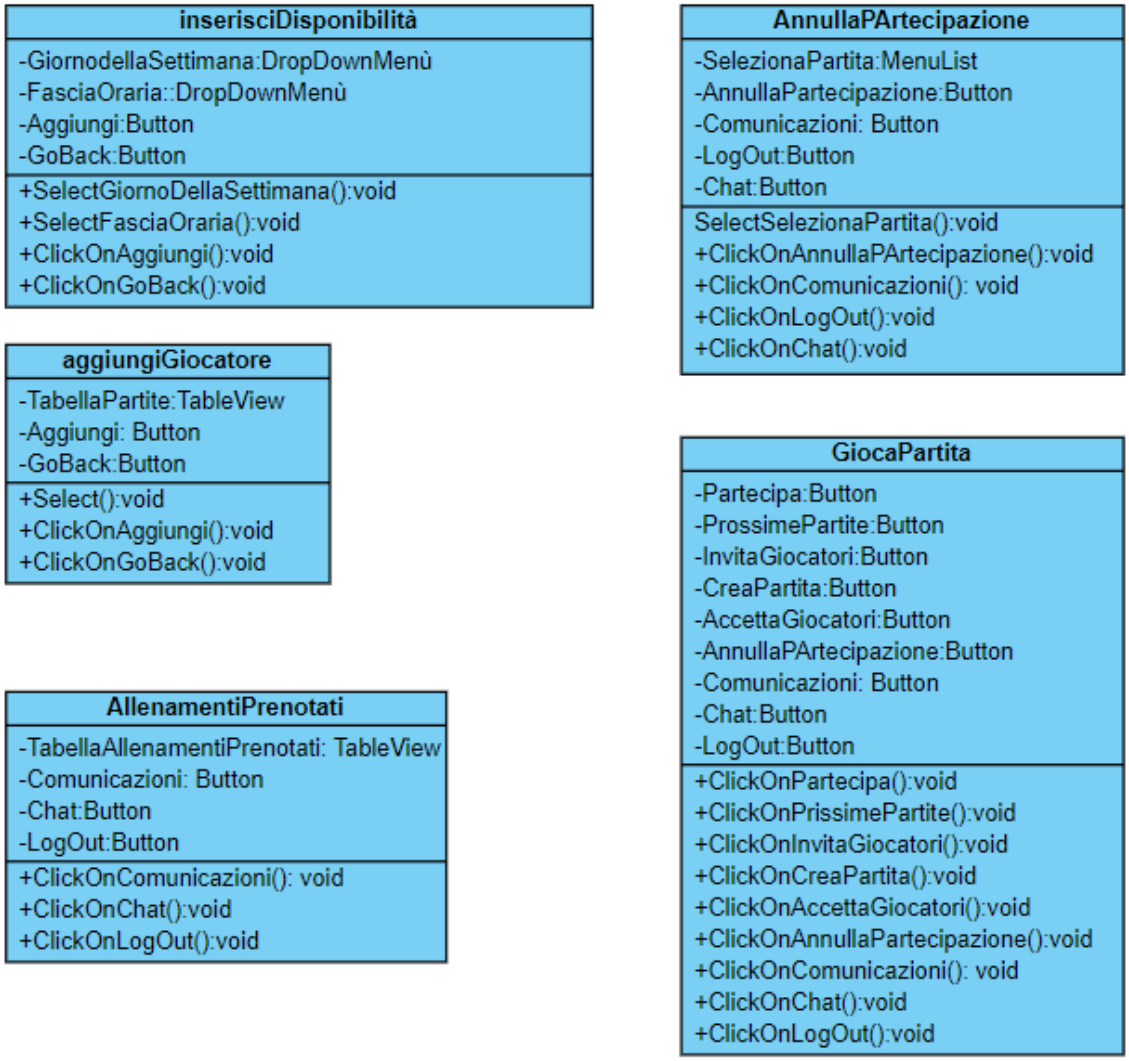
4.9.2 Roles.Gestore.Control.Comunicazioni

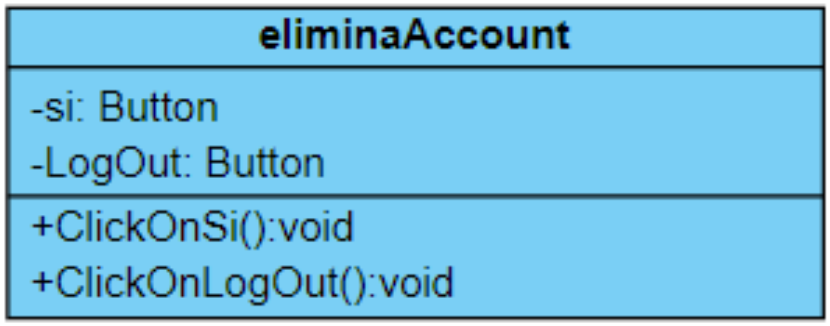


4.9.3 Roles.Gestore.Control.Torneo

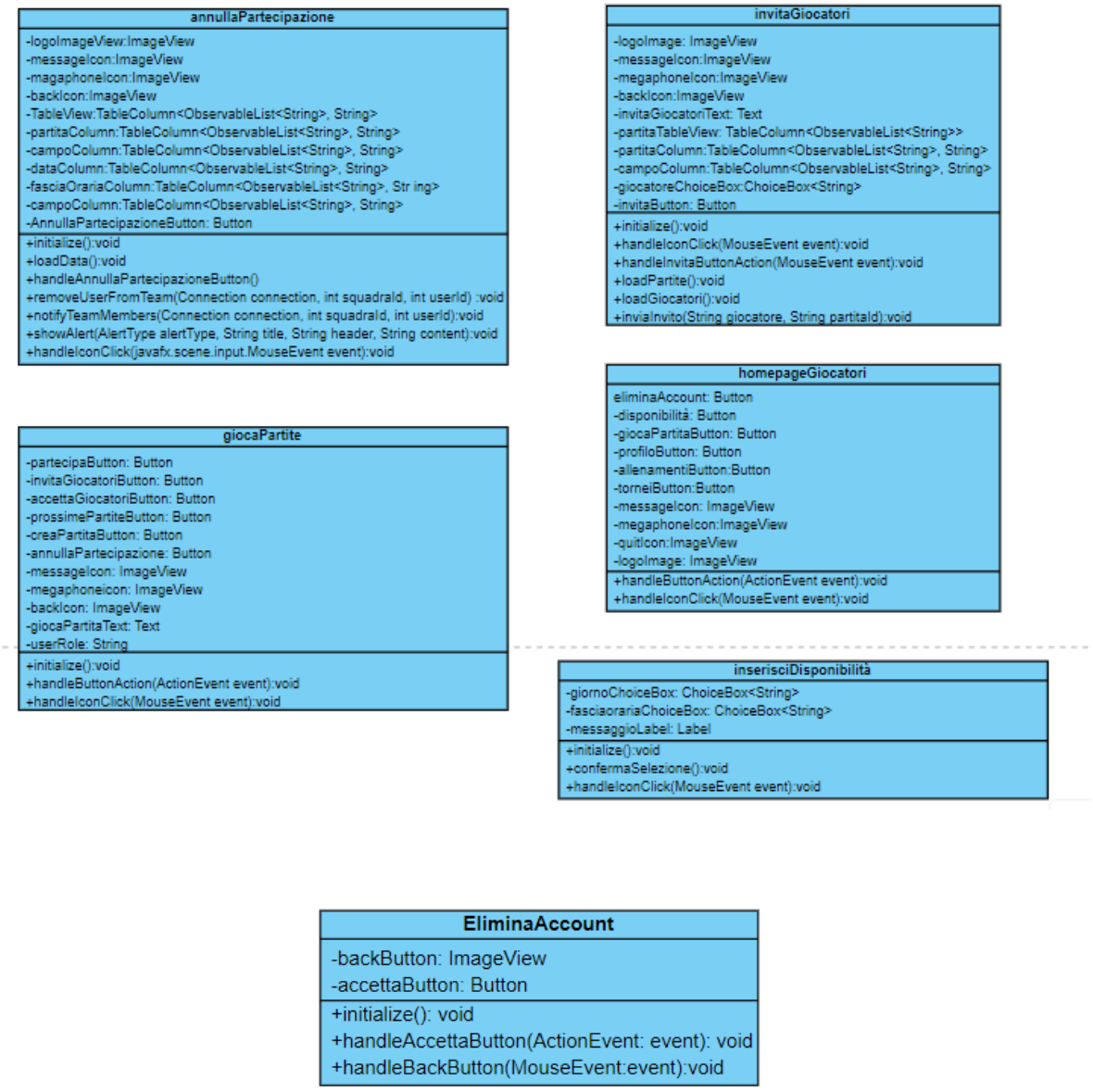


4.10 Roles.GiocatoreOspite.Interfaces

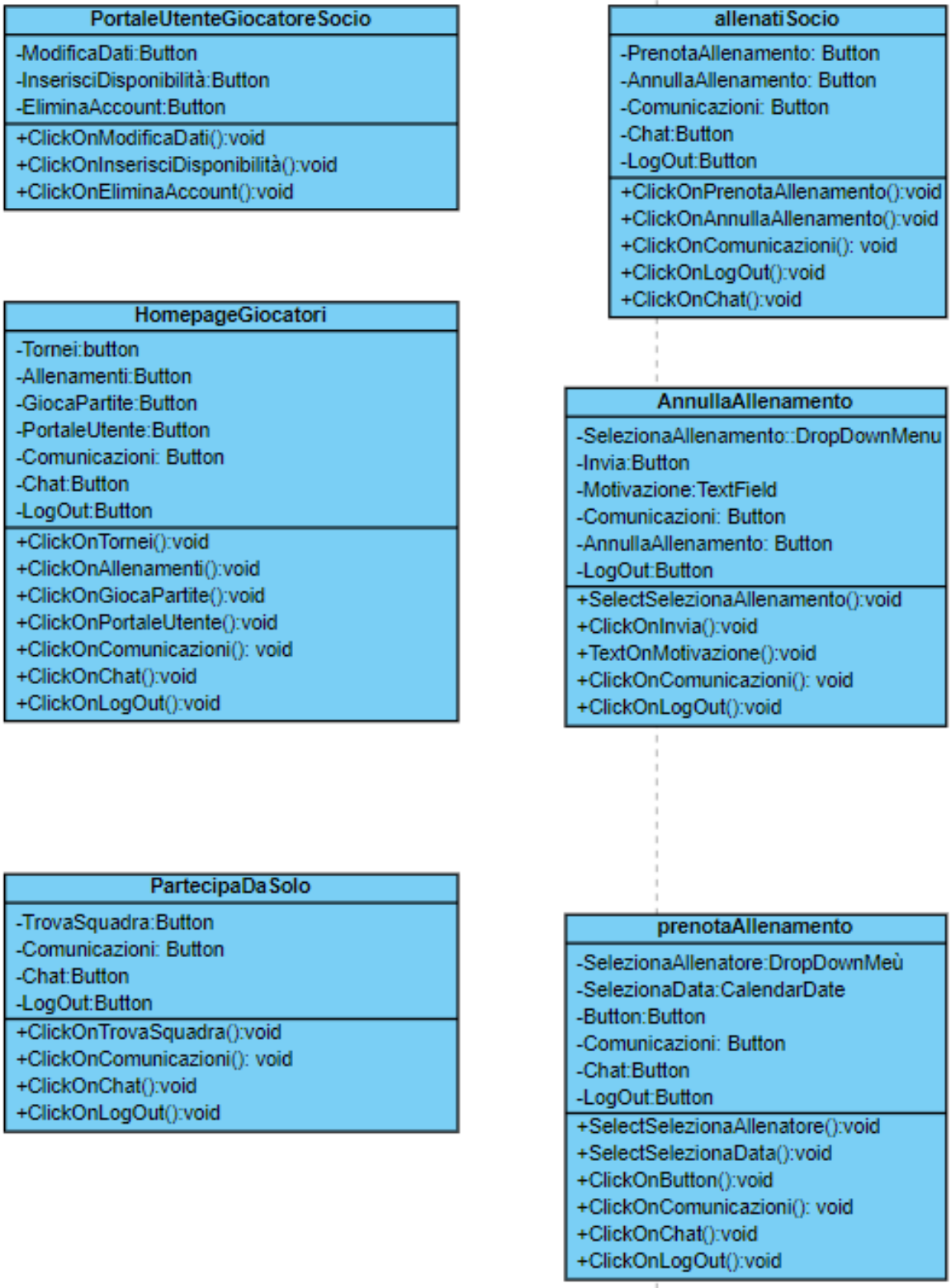


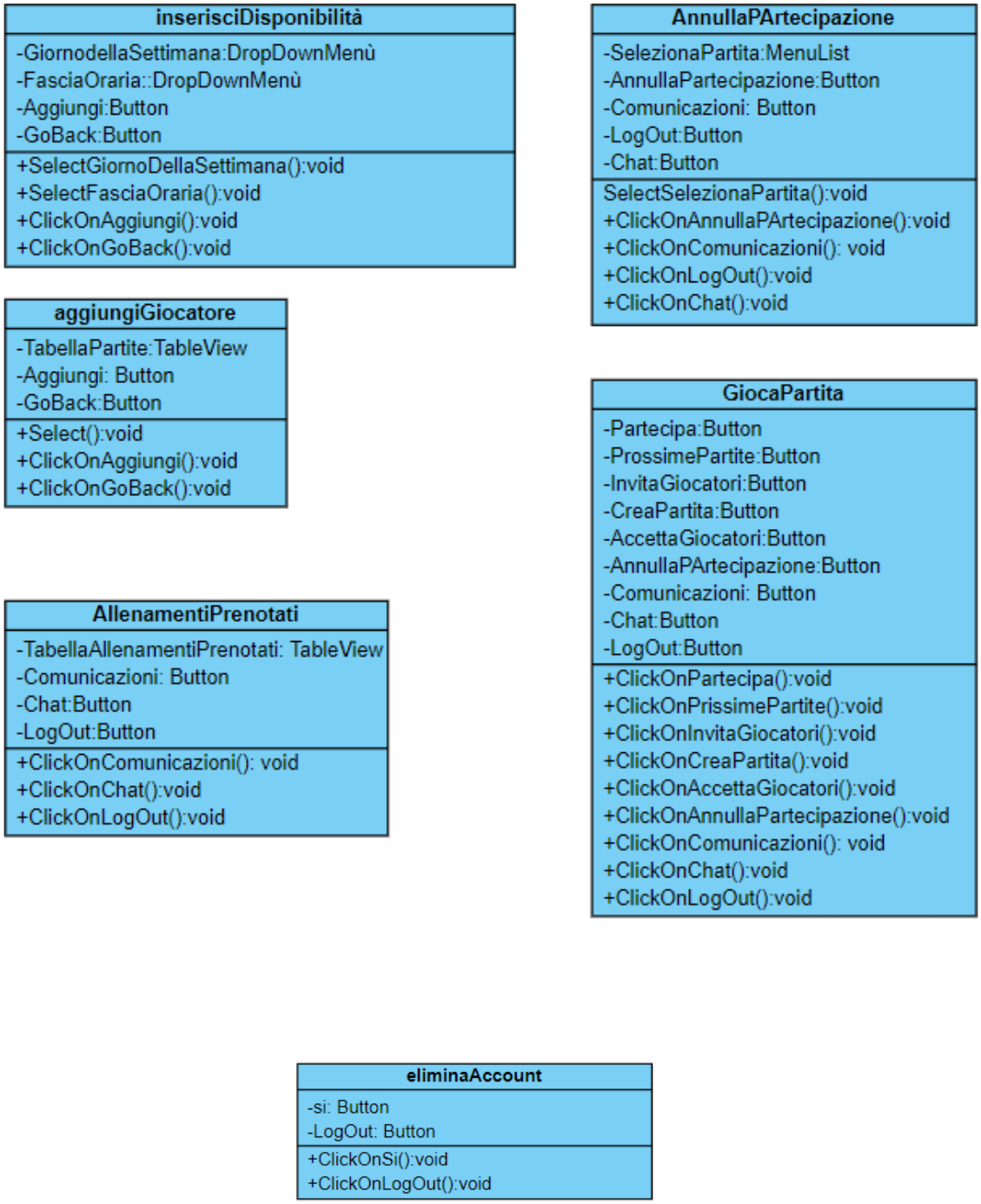

eliminaAccount
-si: Button
-LogOut: Button
+ClickOnSi():void
+ClickOnLogOut():void

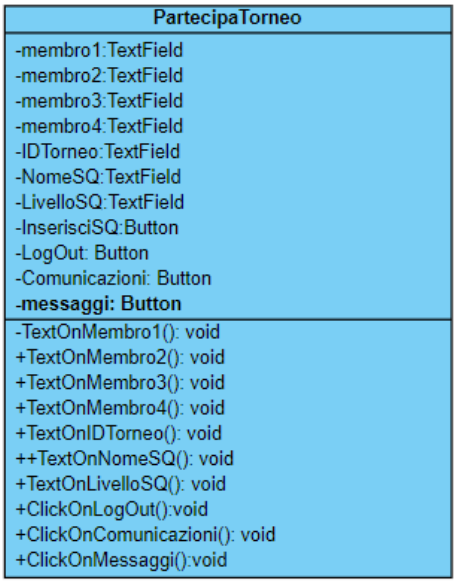
4.11 Roles.GiocatorOspite.Controls

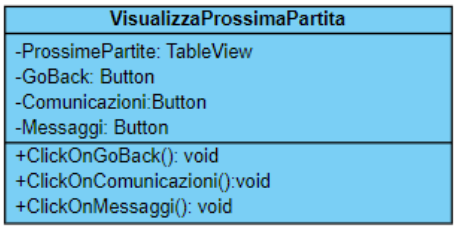


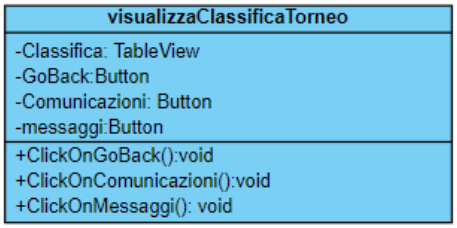
4.12 Roles.GiocatoreSocio.Interfaces



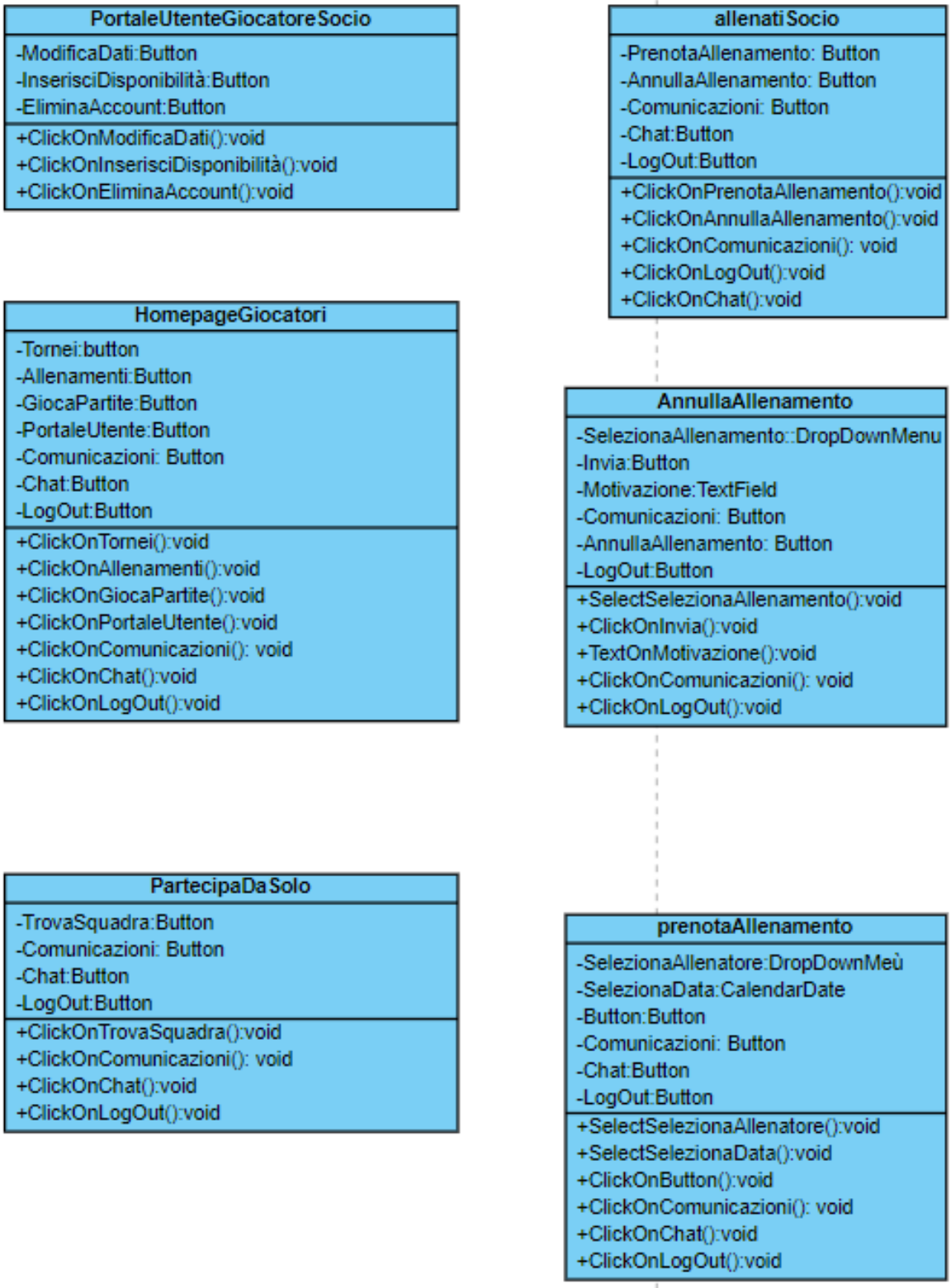

inserisciDisponibilità
-GiornodellaSettimana:DropDownMenù
-FasciaOraria::DropDownMenù
-Aggiungi:Button
-GoBack:Button
+SelectGiornoDellaSettimana():void
+SelectFasciaOraria():void
+ClickOnAggiungi():void
+ClickOnGoBack():void
AnnullaPArtecipazione
-SelezionaPartita:MenuList
-AnnullaPartecipazione:Button
-Comunicazioni: Button
-LogOut:Button
-Chat:Button
SelectSelezionaPartita():void
+ClickOnAnnullaPArtecipazione():void
+ClickOnComunicazioni(): void
+ClickOnLogOut():void
+ClickOnChat():void
aggiungiGiocatore
-TabellaPartite:TableView
-Aggiungi: Button
-GoBack:Button
+Select():void
+ClickOnAggiungi():void
+ClickOnGoBack():void
GiocaPartita
-Partecipa:Button
-ProssimePartite:Button
-InvitaGiocatori:Button
-CreaPartita:Button
-AccettaGiocatori:Button
-AnnullaPArtecipazione:Button
-Comunicazioni: Button
-Chat:Button
-LogOut:Button
+ClickOnPartecipa():void
+ClickOnPrissimePartite():void
+ClickOnInvitaGiocatori():void
+ClickOnCreaPartita():void
+ClickOnAccettaGiocatori():void
+ClickOnAnnullaPartecipazione():void
+ClickOnComunicazioni(): void
+ClickOnChat():void
+ClickOnLogOut():void
AllenamentiPrenotati
-TabellaAllenamentiPrenotati: TableView
-Comunicazioni: Button
-Chat:Button
-LogOut:Button
+ClickOnComunicazioni(): void
+ClickOnChat():void
+ClickOnLogOut():void
eliminaAccount
-si: Button
-LogOut: Button
+ClickOnSi():void
+ClickOnLogOut():void

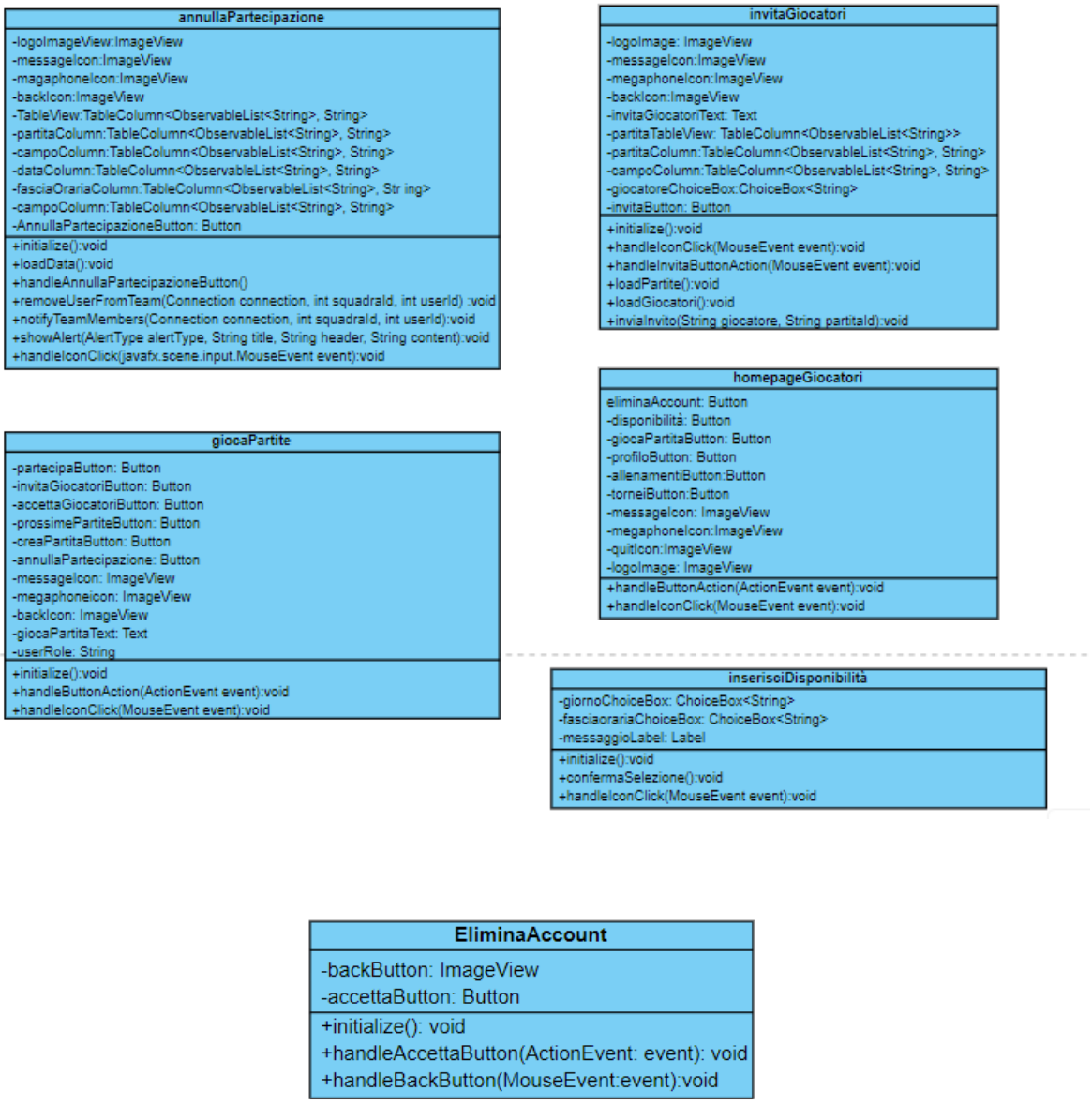

PartecipaTorneo
-membro1:TextField
-membro2:TextField
-membro3:TextField
-membro4:TextField
-IDTorneo:TextField
-NomeSQ:TextField
-LivelloSQ:TextField
-InserisciSQ:Button
-LogOut: Button
-Comunicazioni: Button
-messaggi: Button
-TextOnMembro1(): void
+TextOnMembro2(): void
+TextOnMembro3(): void
+TextOnMembro4(): void
+TextOnIDTorneo(): void
++TextOnNomeSQ(): void
+TextOnLivelloSQ(): void
+ClickOnLogOut():void
+ClickOnComunicazioni(): void
+ClickOnMessaggi():void


VisualizzaProssimaPartita
-ProssimePartite: TableView
-GoBack: Button
-Comunicazioni:Button
-Messaggi: Button
+ClickOnGoBack(): void
+ClickOnComunicazioni():void
+ClickOnMessaggi(): void

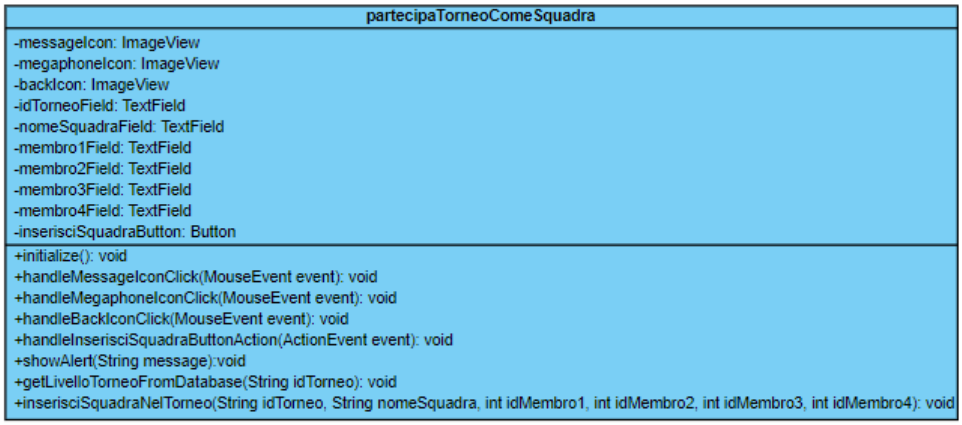

visualizzaClassificaTorneo
-Classifica: TableView
-GoBack:Button
-Comunicazioni: Button
-messaggi:Button
+ClickOnGoBack():void
+ClickOnComunicazioni():void
+ClickOnMessaggi(): void

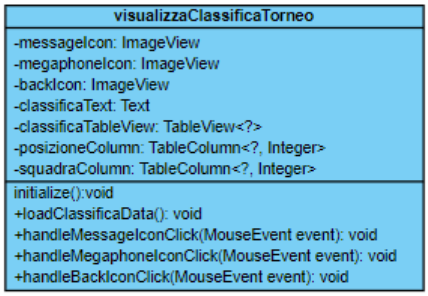
4.13 Roles.GiocatoreSocio.Controls

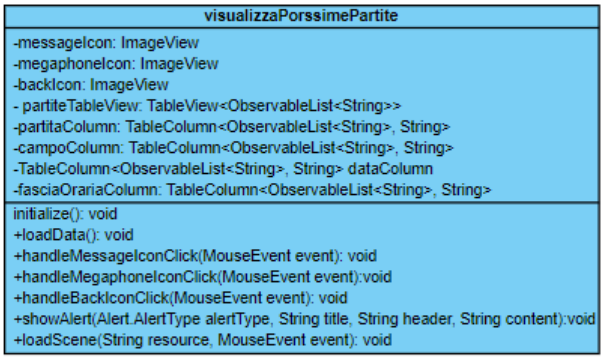



annullaPartecipazione
-logoImageView:ImageView
-messageIcon:ImageView
-magaphoneIcon:ImageView
-backIcon:ImageView
-TableView:TableColumn<ObservableList<String>, String>
-partitaColumn:TableColumn<ObservableList<String>, String>
-campoColumn:TableColumn<ObservableList<String>, String>
-dataColumn:TableColumn<ObservableList<String>, String>
-fasciaOrariaColumn:TableColumn<ObservableList<String>, Str ing>
-campoColumn:TableColumn<ObservableList<String>, String>
-AnnullaPartecipazioneButton: Button
+initialize():void
+loadData():void
+handleAnnullaPartecipazioneButton()
+removeUserFromTeam(Connection connection, int squadraId, int userId) :void
+notifyTeamMembers(Connection connection, int squadraId, int userId):void
+showAlert(AlertType alertType, String title, String header, String content):void
+handleIconClick(javafx.scene.input.MouseEvent event):void
invitaGiocatori
-logoImage: ImageView
-messageIcon:ImageView
-megaphoneIcon:ImageView
-backIcon:ImageView
-invitaGiocatoriText: Text
-partitaTableView: TableColumn<ObservableList<String>>
-partitaColumn:TableColumn<ObservableList<String>, String>
-campoColumn:TableColumn<ObservableList<String>, String>
-giocatoreChoiceBox:ChoiceBox<String>
-invitaButton: Button
+initialize():void
+handleIconClick(MouseEvent event):void
+handleInvitaButtonAction(MouseEvent event):void
+loadPartite():void
+loadGiocatori():void
+inviaInvito(String giocatore, String partitaId):void
homepageGiocatori
eliminaAccount: Button
-disponibilità: Button
-giocaPartitaButton: Button
-profiloButton: Button
-allenamentiButton:Button
-torneiButton:Button
-messageIcon: ImageView
-megaphoneIcon:ImageView
-quitIcon:ImageView
-logoImage: ImageView
+handleButtonAction(ActionEvent event):void
+handleIconClick(MouseEvent event):void
giocaPartite
-partecipaButton: Button
-invitaGiocatoriButton: Button
-accettaGiocatoriButton: Button
-prossimePartiteButton: Button
-creaPartitaButton: Button
-annullaPartecipazione: Button
-messageIcon: ImageView
-megaphoneIcon: ImageView
-backIcon: ImageView
-giocaPartitaText: Text
-userRole: String
+initialize():void
+handleButtonAction(ActionEvent event):void
+handleIconClick(MouseEvent event):void
inserisciDisponibilità
-giornoChoiceBox: ChoiceBox<String>
-fasciaorariaChoiceBox: ChoiceBox<String>
-messaggioLabel: Label
+initialize():void
+confermaSelezione():void
+handleIconClick(MouseEvent event):void

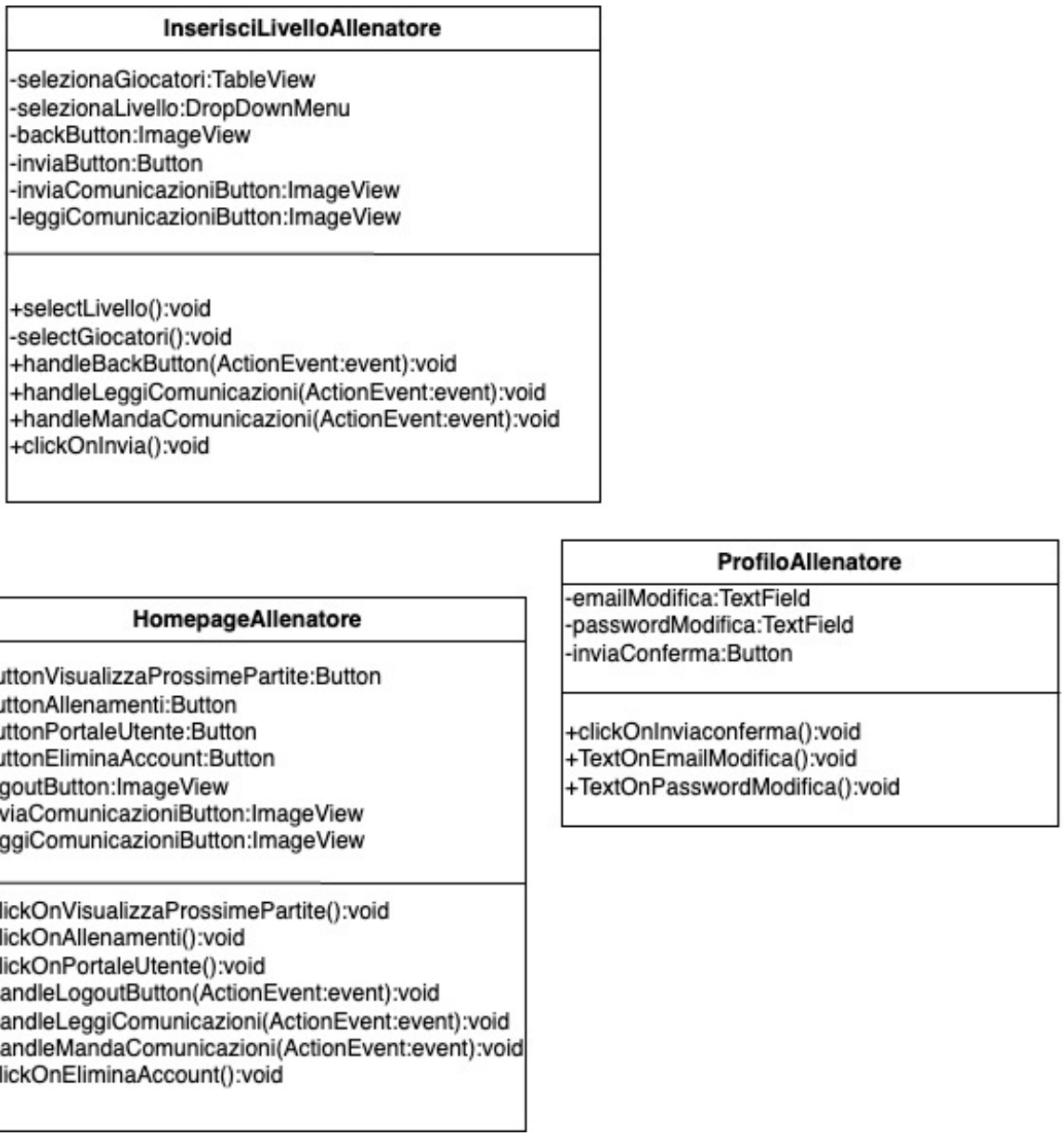
EliminaAccount
-backButton: ImageView
-accettaButton: Button
+initialize(): void
+handleAccettaButton(ActionEvent: event): void
+handleBackButton(MouseEvent:event):void

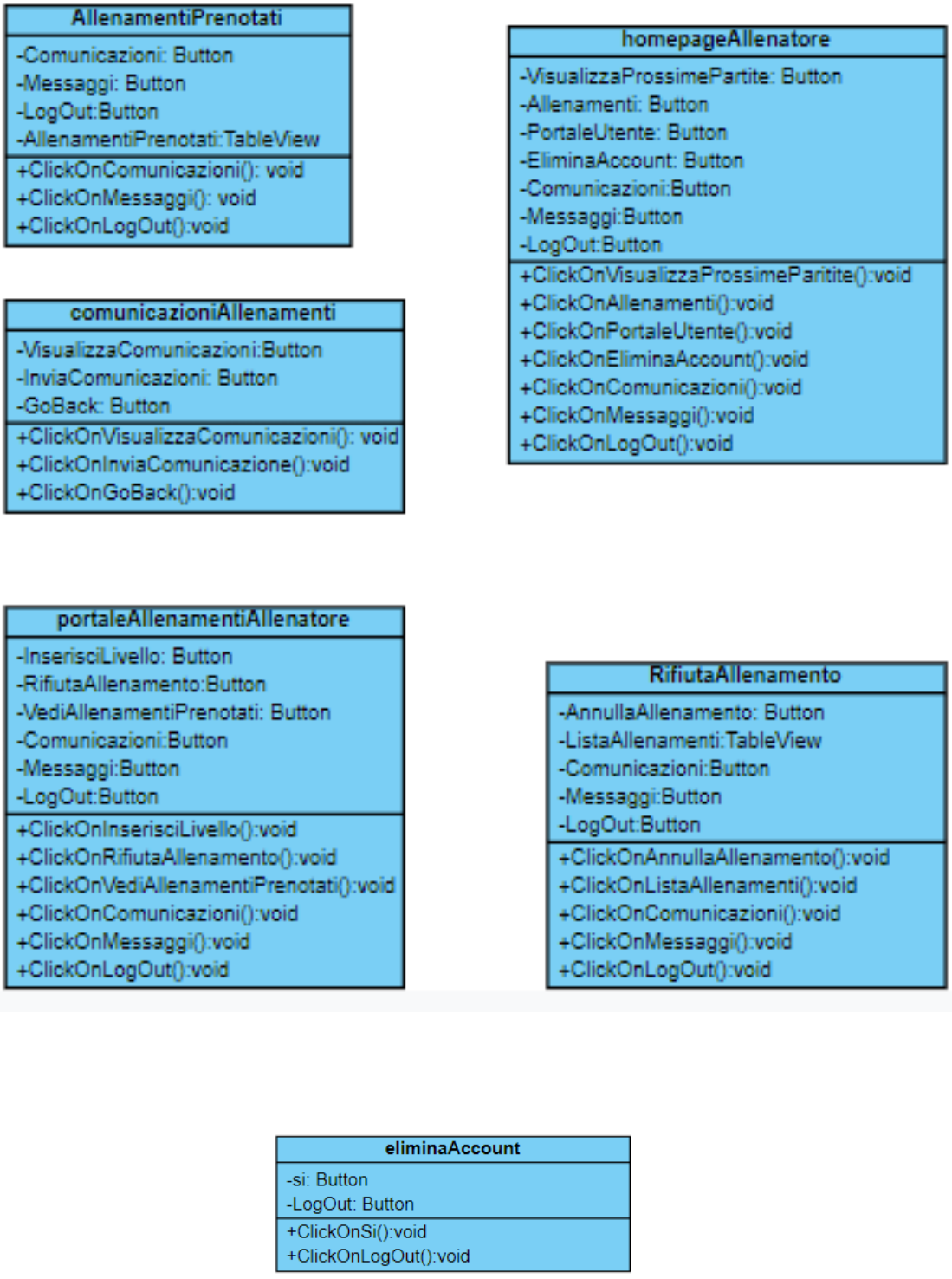

partecipaTorneoComeSquadra
-messageIcon: ImageView
-megaphoneIcon: ImageView
-backIcon: ImageView
-idTorneoField: TextField
-nomeSquadraField: TextField
-membro1Field: TextField
-membro2Field: TextField
-membro3Field: TextField
-membro4Field: TextField
-inserisciSquadraButton: Button
+initialize(): void
+handleMessageIconClick(MouseEvent event): void
+handleMegaphoneIconClick(MouseEvent event): void
+handleBackIconClick(MouseEvent event): void
+handleInserisciSquadraButtonAction(ActionEvent event): void
+showAlert(String message):void
+getLivelloTorneoFromDatabase(String idTorneo): void
+inserisciSquadraNelTorneo(String idTorneo, String nomeSquadra, int idMembro1, int idMembro2, int idMembro3, int idMembro4): void


visualizzaClassificaTorneo
-messageIcon: ImageView
-megaphoneIcon: ImageView
-backIcon: ImageView
-classificaText: Text
-classificaTableView: TableView<?>
-posizioneColumn: TableColumn<?, Integer>
-squadraColumn: TableColumn<?, Integer>
initialize():void
+loadClassificaData(): void
+handleMessageIconClick(MouseEvent event): void
+handleMegaphoneIconClick(MouseEvent event): void
+handleBackIconClick(MouseEvent event): void

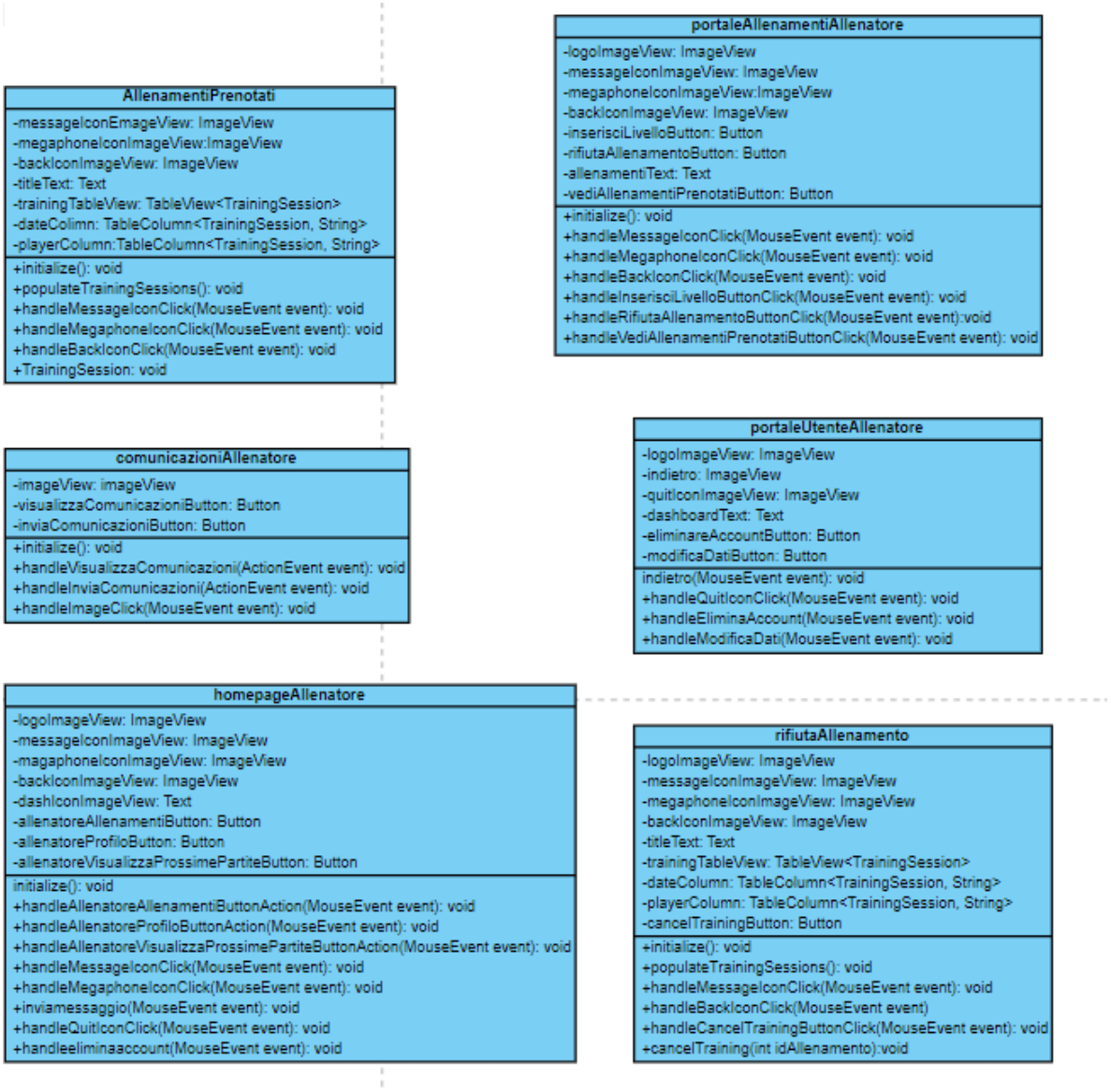

visualizzaPorssimePartite
-messageIcon: ImageView
-megaphoneIcon: ImageView
-backIcon: ImageView
- partiteTableView: TableView<ObservableList<String>>
-partitaColumn: TableColumn<ObservableList<String>, String>
-campoColumn: TableColumn<ObservableList<String>, String>
-TableColumn<ObservableList<String>, String> dataColumn
-fasciaOrariaColumn: TableColumn<ObservableList<String>, String>
initialize(): void
+loadData(): void
+handleMessageIconClick(MouseEvent event): void
+handleMegaphoneIconClick(MouseEvent event):void
+handleBackIconClick(MouseEvent event): void
+showAlert(Alert.AlertType alertType, String title, String header, String content):void
+loadScene(String resource, MouseEvent event): void

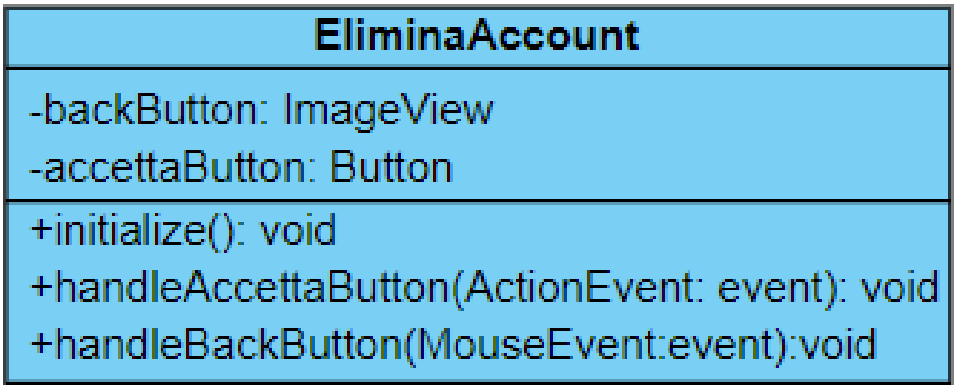
4.14 Roles.Allenatore.Interfaces

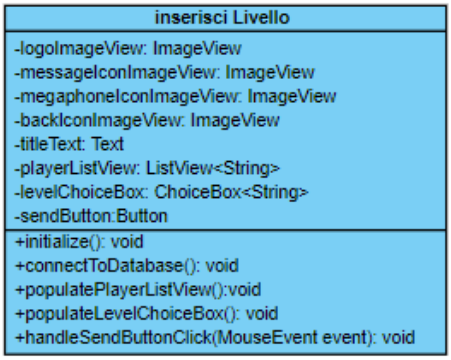

InserisciLivelloAllenatore
-selezionaGiocatori:TableView
-selezionaLivello:DropDownMenu
-backButton:ImageView
-inviaButton:Button
-inviaComunicazioniButton:ImageView
-leggiComunicazioniButton:ImageView
+selectLivello():void
-selectGiocatori():void
+handleBackButton(ActionEvent:event):void
+handleLeggiComunicazioni(ActionEvent:event):void
+handleMandaComunicazioni(ActionEvent:event):void
+clickOnInvia():void
HomepageAllenatore
-buttonVisualizzaProssimePartite:Button
-buttonAllenamenti:Button
-buttonPortaleUtente:Button
-buttonEliminaAccount:Button
-logoutButton:ImageView
-inviaComunicazioniButton:ImageView
-leggiComunicazioniButton:ImageView
+clickOnVisualizzaProssimePartite():void
+clickOnAllenamenti():void
+clickOnPortaleUtente():void
+handleLogoutButton(ActionEvent:event):void
+handleLeggiComunicazioni(ActionEvent:event):void
+handleMandaComunicazioni(ActionEvent:event):void
+clickOnEliminaAccount():void
ProfiloAllenatore
-emailModifica:TextField
-passwordModifica:TextField
-inviaConferma:Button
+clickOnInviaconferma():void
+TextOnEmailModifica():void
+TextOnPasswordModifica():void

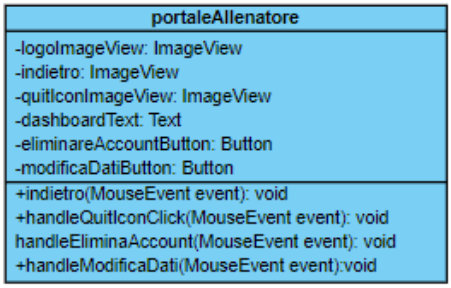

AllenamentiPrenotati
-Comunicazioni: Button
-Messaggi: Button
-LogOut:Button
-AllenamentiPrenotati:TableView
+ClickOnComunicazioni(): void
+ClickOnMessaggi(): void
+ClickOnLogOut():void
homepageAllenatore
-VisualizzaProssimePartite: Button
-Allenamenti: Button
-PortaleUtente: Button
-EliminaAccount: Button
-Comunicazioni:Button
-Messaggi:Button
-LogOut:Button
+ClickOnVisualizzaProssimeParitite():void
+ClickOnAllenamenti():void
+ClickOnPortaleUtente():void
+ClickOnEliminaAccount():void
+ClickOnComunicazioni():void
+ClickOnMessaggi():void
+ClickOnLogOut():void
comunicazioniAllenamenti
-VisualizzaComunicazioni:Button
-InviaComunicazioni: Button
-GoBack: Button
+ClickOnVisualizzaComunicazioni(): void
+ClickOnInviaComunicazione():void
+ClickOnGoBack():void
portaleAllenamentiAllenatore
-InserisciLivello: Button
-RifiutaAllenamento:Button
-VediAllenamentiPrenotati: Button
-Comunicazioni:Button
-Messaggi:Button
-LogOut:Button
+ClickOnInserisciLivello():void
+ClickOnRifiutaAllenamento():void
+ClickOnVediAllenamentiPrenotati():void
+ClickOnComunicazioni():void
+ClickOnMessaggi():void
+ClickOnLogOut():void
RifiutaAllenamento
-AnnullaAllenamento: Button
-ListaAllenamenti:TableView
-Comunicazioni:Button
-Messaggi:Button
-LogOut:Button
+ClickOnAnnullaAllenamento():void
+ClickOnListaAllenamenti():void
+ClickOnComunicazioni():void
+ClickOnMessaggi():void
+ClickOnLogOut():void
eliminaAccount
-si: Button
-LogOut: Button
+ClickOnSi():void
+ClickOnLogOut():void

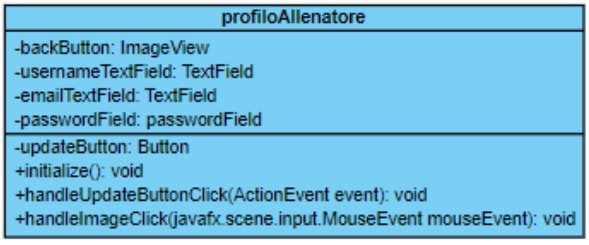
4.15 Roles.Allenatore.Controls




EliminaAccount
-backButton: ImageView
-accettaButton: Button
+initialize(): void
+handleAccettaButton(ActionEvent: event): void
+handleBackButton(MouseEvent:event):void


inserisci Livello
-logoImageView: ImageView
-messageIconImageView: ImageView
-megaphoneIconImageView: ImageView
-backIconImageView: ImageView
-titleText: Text
-playerListView: ListView<String>
-levelChoiceBox: ChoiceBox<String>
-sendButton:Button
+initialize(): void
+connectToDatabase(): void
+populatePlayerListView():void
+populateLevelChoiceBox(): void
+handleSendButtonClick(MouseEvent event): void


portaleAllenatore
-logoImageView: ImageView
-indietro: ImageView
-quitIconImageView: ImageView
-dashboardText: Text
-eliminareAccountButton: Button
-modificaDatiButton: Button
+indietro(MouseEvent event): void
+handleQuitIconClick(MouseEvent event): void
handleEliminaAccount(MouseEvent event): void
+handleModificaDati(MouseEvent event):void


profiloAllenatore
-backButton: ImageView
-usernameTextField: TextField
-emailTextField: TextField
-passwordField: passwordField
-updateButton: Button
+initialize(): void
+handleUpdateButtonClick(ActionEvent event): void
+handleImageClick(javafx.scene.input.MouseEvent mouseEvent): void