



TEAM COMMITTED

UNIVERSITÀ DEGLI STUDI DI PADOVA

Definizione di prodotto V2.0



Informazioni sul documento

Nome documento	Definizione di prodotto
Data documento	2012/07/19
Redattori	<ul style="list-style-type: none">• Marco Begolo• Gabriele Facchin• Alessandro Cornaglia• Massimo Dalla Pietá• Lorenzo Braghetto• Giacomo Quadrio• Giorgio Maggioli
Verificatori	<ul style="list-style-type: none">• Marco Begolo• Lorenzo Braghetto• Giacomo Quadrio
Approvazione	<ul style="list-style-type: none">• Giorgio Maggioli• Gabriele Facchin
Uso documento	Esterno
Lista distribuzione	<ul style="list-style-type: none">• <i>Team Committed</i>• <i>Prof. Tullio Vardanega</i>

Sommario

Il presente documento intende descrivere in modo dettagliato tutte le componenti del *Progetto SafetyGame* e i criteri con le quali interagiscono fra loro. Per ogni componente sono illustrati gli attributi con il loro significato e i metodi con il loro comportamento



Diario delle modifiche

Modifica	Autore	Data	Versione
<i>Approvato</i>	Gabriele Facchin	2012/07/19	V2.0
<i>Verificato documento. In attesa di approvazione</i>	Marco Begolo	2012/07/18	V1.17
<i>Aggiornate immagini dei vari Front-End e Back-End. In attesa di verifica.</i>	Massimo dalla Pieta	2012/07/16	V1.16
<i>Terminato aggiornamento capitolo 5 Specifica Front-End Web 42</i>	Massimo dalla Pieta	2012/07/14	V1.15
<i>Inizio aggiornamento capitolo 5 Specifica Front-End Web</i>	Giacomo Quadrio	2012/07/12	V1.14
<i>Terminato aggiornamento capitolo 6 Specifica Front-End Mobile</i>	Giacomo Quadrio	2012/07/08	V1.13
<i>Inizio aggiornamento capitolo 6 Specifica Front-End Mobile</i>	Giacomo Quadrio	2012/07/06	V1.12
<i>Rilevati alcuni errori di forma e corretti.</i>	Massimo dalla pieta	2012/07/05	V1.11
<i>Modificati schemi Front-End Desktop.</i>	Alessandro Cornaglia	2012/07/04	V1.10
<i>Terminato aggiornamento capitolo 4 Specifica Front-End Desktop.</i>	Giorgio Maggiolo	2012/07/03	V1.9
<i>Modifica descrizione di alcune classi del capitolo 4.</i>	Giorgio Maggiolo	2012/07/03	V1.8
<i>Inizio aggiornamento capitolo 4 Specifica Front-End Desktop</i>	Giorgio Maggiolo	2012/07/02	V1.7
<i>Terminato aggiornamento capitolo 3 Specifica Back-End</i>	Alessandro Cornaglia	2012/06/27	V1.6
<i>Modificate classi relative al Back-End</i>	Massimo dalla Pieta	2012/06/29	V1.5
<i>Inizio aggiornamento capitolo 3 Specifica Back-End</i>	Alessandro Cornaglia	2012/06/29	V1.4
<i>Terminata correzione del documento Definizione di prodotto</i>	Giorgio Maggiolo	2012/06/28	V1.3
<i>Correzione delle immagini secondo indicazioni del documento di valutazione.</i>	Giacomo Quadrio	2012/06/28	V1.2
<i>Inizio correzioni Definizione di prodotto secondo indicazioni riportate nel documento di valutazione.</i>	Giorgio Maggiolo	2012/06/27	V1.1
<i>Approvato</i>	Giorgio Maggiolo	2012/06/15	V1.0
<i>Verifica ultimata. In attesa di approvazione.</i>	Giacomo Quadrio	2012/06/10	V0.23
<i>Inizio verifica</i>	Marco Begolo	2012/06/10	V0.22
<i>Finita redazione capitolo 8. In attesa di verifica</i>	Alessandro Cornaglia	2012/06/05	V0.21

<i>Finita redazione capitolo 7</i>	Gabriele Facchin	2012/05/26	V0.20
<i>Finita redazione capitolo 3</i>	Alessandro Cornaglia	2012/05/22	V0.19
<i>Redazione capitolo 3: finito back.controller. Inizio back.access</i>	Alessandro Cornaglia	2012/05/20	V0.18
<i>Finita redazione capitolo 5</i>	Marco Begolo	2012/05/16	V0.17
<i>Inizio redazione capitolo 8</i>	Alessandro Cornaglia	2012/05/15	V0.16
<i>Inizio redazione capitolo 7</i>	Gabriele Facchin	2012/05/13	V0.15
<i>Redazione capitolo 3: finito package back.access e iniziato back.controller</i>	Alessandro Cornaglia	2012/05/12	V0.14
<i>Redazione capitolo 5: finito layer pagine. Inizio controllo</i>	Marco Begolo	2012/05/10	V0.13
<i>Iniziata redazione capitolo 5</i>	Massimo Dalla Pietá	2012/05/07	V0.12
<i>Redazione capitolo 3: iniziato package back.access</i>	Marco Begolo	2012/05/03	V0.11
<i>Finita redazione capitolo 6</i>	Lorenzo Braghetto	2012/05/06	V0.10
<i>Iniziata redazione capitolo 6</i>	Lorenzo Braghetto	2012/05/03	V0.9
<i>Continua redazione capitolo 3. Finito il package back.condivisi</i>	Marco Begolo	2012/05/02	V0.8
<i>Finita redazione capitolo 4</i>	Giacomo Quadrio	2012/05/02	V0.7
<i>Continua redazione capitolo 4. Inserito package desktop.condivisi e desktop.logic</i>	Massimo Dalla Pietá	2012/02/28	V0.6
<i>Continua redazione capitolo 3. Inserito parte del package back.condivisi</i>	Giacomo Quadrio	2012/02/26	V0.5
<i>Iniziata redazione capitolo 4</i>	Massimo Dalla Pietá	2012/02/20	V0.4
<i>Iniziata redazione capitolo 3</i>	Lorenzo Braghetto	2012/02/17	V0.3
<i>Redatto il capitolo 2</i>	Massimo Dalla Pietá	2012/02/16	V0.2
<i>Creati i sorgenti per il file. Redatto il capitolo 1</i>	Marco Begolo	2012/02/15	V0.1

Indice

1	Introduzione	8
1.1	Scopo del documento	8
1.2	Scopo del prodotto	8
1.3	Ambiguità	8
1.4	Riferimenti	8
1.4.1	Riferimenti normativi	8
1.4.2	Riferimenti informativi	8
2	Standard di progetto	10
2.1	Standard di progettazione architetturale	10
2.2	Standard di documentazione del codice	10
2.3	Standard di denominazione di entità e relazioni	10
2.4	Standard di programmazione	10
2.5	Strumenti di lavoro	10
3	Specifiche Back-end	11
3.1	Package com.safetyGame.back	12
3.1.1	com.safetyGame.back.Inizializzatore	13
3.2	Package com.safetyGame.back.connection	18
3.2.1	com.safetyGame.back.connection.WebConnection	19
3.2.2	com.safetyGame.back.connection.ApplicazioniConnection	24
3.3	Package com.safetyGame.back.controller	27
3.3.1	com.safetyGame.back.controller.GestioneBadgeAS	28
3.3.2	com.safetyGame.back.controller.GestioneBadgeD	29
3.3.3	com.safetyGame.back.controller.GestioneDati	31
3.3.4	com.safetyGame.back.controller.GestioneDipendentiAA	37
3.3.5	com.safetyGame.back.controller.GestioneDipendentiD	39
3.3.6	com.safetyGame.back.controller.GestioneDomandeAS	41
3.3.7	com.safetyGame.back.controller.GestioneDomandeD	43
3.3.8	com.safetyGame.back.controller.GestioneLog	46
3.3.9	com.safetyGame.back.controller.GestioneLogin	51
3.3.10	com.safetyGame.back.controller.GestionePunteggiAA	53
3.3.11	com.safetyGame.back.controller.GestionePunteggiD	55
3.3.12	com.safetyGame.back.controller.GestioneRecupero	57
3.4	Package com.safetyGame.back.access	60
3.4.1	Interfaccia com.safetyGame.back.access.DAOBadge	61
3.4.2	com.safetyGame.back.access.SqlDAOBadge	62
3.4.3	Interfaccia com.safetyGame.back.access.DAODipendenti	64
3.4.4	com.safetyGame.back.access.SqlDAODipendenti	67
3.4.5	Interfaccia com.safetyGame.back.access.DAODomande	70
3.4.6	com.safetyGame.back.access.SqlDAODomande	72
3.4.7	Interfaccia com.safetyGame.back.access.DAOFactory	75
3.4.8	com.safetyGame.back.access.SqlDAOFactory	77
3.4.9	Interfaccia com.safetyGame.back.access.DAOLogin	79

3.4.10	com.safetyGame.back.access.SqlDAOLogin	80
3.4.11	Interfaccia com.safetyGame.back.access.DAOPunteggi . . .	82
3.4.12	com.safetyGame.back.access.SqlDAOPunteggi	83
3.4.13	com.safetyGame.back.access.Indirizzo	85
3.4.14	com.safetyGame.back.access.UpdateLog	88
3.5	Package com.safetyGame.back.condivisi	90
3.5.1	com.safetyGame.back.condivisi.Badge	91
3.5.2	com.safetyGame.back.condivisi.DataOra	93
3.5.3	com.safetyGame.back.condivisi.Dipendente	96
3.5.4	com.safetyGame.back.condivisi.Domanda	101
3.5.5	com.safetyGame.back.condivisi.Login	105
3.5.6	com.safetyGame.back.condivisi.Punteggio	107
3.5.7	com.safetyGame.back.condivisi.Recupero	110
4	Specifiche Front-end Desktop	112
4.1	Package com.safetyGame.desktop.view	113
4.1.1	com.safetyGame.desktop.view.Error	114
4.1.2	com.safetyGame.desktop.view.Login	115
4.1.3	com.safetyGame.desktop.view.Menu	118
4.1.4	com.safetyGame.desktop.view.Notifica	121
4.1.5	com.safetyGame.desktop.view.Richiesta	123
4.2	Package com.safetyGame.desktop.logic	124
4.2.1	com.safetyGame.desktop.view.Browser	125
4.2.2	com.safetyGame.desktop.view.ConnBack	127
4.2.3	com.safetyGame.desktop.logic.ControlLogin	130
4.2.4	com.safetyGame.desktop.logic.ControlMenu	131
4.2.5	com.safetyGame.desktop.logic.ControlNotifica	133
4.2.6	com.safetyGame.desktop.logic.DatiLogin	134
4.2.7	com.safetyGame.desktop.logic.Parser	135
4.2.8	com.safetyGame.desktop.logic.Timer	137
4.3	Package com.safetyGame.desktop.condivisi	139
4.3.1	com.safetyGame.desktop.condivisi.Login	140
5	Specifiche Front-end Web	141
5.1	Package com.safetyGame.frontWeb.pagine	142
5.1.1	com.safetyGame.frontWeb.pagine.Login	142
5.1.2	com.safetyGame.frontWeb.pagine.Menu	142
5.1.3	com.safetyGame.frontWeb.pagine.Notifica	142
5.1.4	com.safetyGame.frontWeb.pagine.MostraDomanda	142
5.1.5	com.safetyGame.frontWeb.pagine.MostraDatiD	143
5.1.6	com.safetyGame.frontWeb.pagine.MostraDatiAA	143
5.1.7	com.safetyGame.frontWeb.pagine.MostraStatD	143
5.1.8	com.safetyGame.frontWeb.pagine.ModDatiD	143
5.1.9	com.safetyGame.frontWeb.pagine.MostraStatAA	143
5.1.10	com.safetyGame.frontWeb.pagine.ModDatiAA	143
5.1.11	com.safetyGame.frontWeb.pagine.RecuperoPass	143

5.1.12	com.safetyGame.frontWeb.pagine.ModDatiAS	143
5.1.13	com.safetyGame.frontWeb.pagine.ElencoDomAS	144
5.1.14	com.safetyGame.frontWeb.pagine.ModDomAS	144
5.2	Package com.safetyGame.frontWeb.control	145
5.2.1	com.safetyGame.frontWeb.control.ControlLogin	145
5.2.2	com.safetyGame.frontWeb.control.ControlMostraDatiD . .	145
5.2.3	com.safetyGame.frontWeb.control.ControlModDomAS . .	145
5.2.4	com.safetyGame.frontWeb.control.Timer	145
5.2.5	com.safetyGame.frontWeb.control.ControlMenu	145
5.2.6	com.safetyGame.frontWeb.control.ControlMostraStatD . .	146
5.2.7	com.safetyGame.frontWeb.control.ControlNotifica	146
5.2.8	com.safetyGame.frontWeb.control.ControlModDatiD . . .	146
5.2.9	com.safetyGame.frontWeb.control.ControlMostraDom . .	146
5.2.10	com.safetyGame.frontWeb.control.ControlModDatiAA . .	146
5.2.11	com.safetyGame.frontWeb.control.ControlModDatiAS . .	146
5.2.12	com.safetyGame.frontWeb.control.ControlElencoDomAS .	146
5.2.13	com.safetyGame.frontWeb.control.ControlRecuperoPass .	146
5.3	Descrizione di dettaglio delle pagine JSP	147
5.3.1	addTrofeo.jsp	147
5.3.2	admin_page.jsp	147
5.3.3	aggiungiDomande.jsp	147
5.3.4	checkAggiungiDipendente.jsp	147
5.3.5	checkAggiungiDomande.jsp	147
5.3.6	checkDelDipendente.jsp	148
5.3.7	checkLogin.jsp	148
5.3.8	checkModDipendente.jsp	148
5.3.9	checkModEmailD.jsp	148
5.3.10	checkModPassA.jsp	148
5.3.11	checkModPassD.jsp	148
5.3.12	checkRecuperoPass.jsp	149
5.3.13	checkRisposta.jsp	149
5.3.14	eliminaDipendente.jsp	149
5.3.15	forzaCambioPass.jsp	149
5.3.16	forzaCambioPassA.jsp	149
5.3.17	gestioneDipendenti.jsp	149
5.3.18	getCookies.jsp	150
5.3.19	index.jsp	150
5.3.20	login.jsp	150
5.3.21	logout.jsp	150
5.3.22	modEmailD.jsp	150
5.3.23	modificaDipendente.jsp	150
5.3.24	modPassA.jsp	150
5.3.25	modPassD.jsp	151
5.3.26	nuovaDomanda.jsp	151
5.3.27	recuperaPass.jsp	151
5.3.28	rimuoviDomande.jsp	151

5.3.29	subTrofeo.jsp	151
5.3.30	user_page.jsp	151
5.3.31	visualizzaDatiA.jsp	151
5.3.32	visualizzaDatiD.jsp	152
5.3.33	visualizzaDipendente.jsp	152
6	Specifiche Front-end Mobile	153
6.1	Package com.safetyGame.mobile.View	154
6.1.1	com.safetyGame.mobile.View.DashboardActivity	155
6.1.2	com.safetyGame.mobile.View.DashboardLayout	156
6.1.3	com.safetyGame.mobile.View.DatiActivity	158
6.1.3.1	com.safetyGame.mobile.View.DatiActivity.DatiTask159	
6.1.3.2	com.safetyGame.mobile.View.DatiActivity.InviaDatiTask160	
6.1.4	com.safetyGame.mobile.View.DomandaActivity	161
6.1.4.1	com.safetyGame.mobile.View.DomandaActivity.DomandaTask162	
6.1.5	com.safetyGame.mobile.View.LoginActivity	163
6.1.5.1	com.safetyGame.mobile.View.LoginActivity.LoginTask165	
6.1.6	com.safetyGame.mobile.View.PunteggiActivity	167
6.1.6.1	com.safetyGame.mobile.View.PunteggiActivity.PunteggiTask168	
6.1.7	com.safetyGame.mobile.View.TimerNotifica	169
6.1.7.1	com.safetyGame.mobile.View.TimerNotifica.mUpdateTimeTask170	
6.2	Package com.safetyGame.mobile.Utils	171
6.2.1	com.safetyGame.mobile.Utils.BootReceiver	172
6.2.2	com.safetyGame.mobile.Utils.ConnectionUtils	173
6.2.3	com.safetyGame.mobile.Utils.IntentIntegrator	174
6.2.4	com.safetyGame.mobile.Utils.IntentResult	177
6.3	Package com.safetyGame.mobile.condivisi	179
6.3.1	com.safetyGame.mobile.condivisi.Dati	180
6.3.2	com.safetyGame.mobile.condivisi.Domanda	181
6.3.3	com.safetyGame.mobile.condivisi.Punteggi	183
7	Diagrammi di sequenza	185
7.1	Login mobile	186
7.2	Aggiunta domanda da parte di un Amministratore Sicurezza	188
8	Tracciamento	190
8.1	Packages naming	190
8.2	Tracciamento componenti-requisiti	190
8.2.1	Desktop	190
8.2.2	Mobile	192
8.2.3	Back End	195
8.3	Tracciamento requisiti-componenti	204

1 Introduzione

1.1 Scopo del documento

Il documento di *Definizione di Prodotto* descrive tutte le componenti del sistema e il modo in cui esse collaborano. Per ogni componente vengono illustrati gli attributi con il loro significato e i medoti con il loro comportamento.

Gli aspetti più delicati sono rappresentati anche attraverso diagrammi di sequenza, in modo da mostrare in modo chiaro l'ordine delle operazioni e l'interazione delle componenti.

Lo scopo principale del documento è quello di fornire ai programmatori una solida e precisa guida alla codifica, in modo che essi possano lavorare in modo autonomo attenendosi alle scelte progettuali ed evitando soluzioni personali ed improvvise.

1.2 Scopo del prodotto

Il prodotto denominato **SafetyGame** si propone di fornire uno strumento informatico per la gestione delle pratiche di sicurezza sul lavoro in modo dinamico, evitando corsi di formazione che spesso si dimostrano inutili per la poca attenzione prestata dai partecipanti.

Lo strumento si basa sul concetto di **gamification** che comporta competizione tra i dipendenti all'interno delle aziende creando un sano interesse per un argomento delicato come la sicurezza sul luogo di lavoro.

Il sistema è pensato sia per lavoratori che hanno una postazione fissa dotata di **PC**, sia per quelli che hanno la necessità di spostarsi e che quindi sono forniti di **dispositivi mobili**. Ad essi verranno poste periodicamente domande, di varia tipologia, le cui risposte comporteranno l'assegnazione di un punteggio generando una classifica aziendale.

1.3 Ambiguità

Al fine di evitare ogni ambiguità relativa al linguaggio e ai termini utilizzati nei documenti formali, il glossario viene incluso nel file **Glossario-V3.0.pdf**, dove vengono definiti e descritti i termini marcati da una sottolineatura.

1.4 Riferimenti

1.4.1 Riferimenti normativi

- **Norme di Progetto, V4.0** (allegato **Norme_di_Progetto_V4.0.pdf**)
- **Analisi dei Requisiti, V3.0** (allegato **Analisi_dei_Requisiti_V3.0.pdf**)

1.4.2 Riferimenti informativi

- **Specifiche Tecniche, V2.0** (allegato **Specifiche_Tecniche_V2.0.pdf**)



- **Elementi di Gamification, V1.0** (allegato Elementi_Gamification_V1.0.pdf)

2 Standard di progetto

2.1 Standard di progettazione architetturale

Per gli standard di progettazione architetturale, si veda il documento *Specifica Tecnica - V 2.0*.

2.2 Standard di documentazione del codice

La documentazione del codice sarà prodotta come Javadoc. Per ulteriori informazioni si vedano le *Norme di Progetto - V 3.0*.

2.3 Standard di denominazione di entità e relazioni

Entità e relazioni devono avere nomi chiari, coincisi ed esplicativi. Per ulteriori informazioni si faccia riferimento alle *Norme di Progetto - V 3.0*.

2.4 Standard di programmazione

Per gli standard di programmazione si vedano le *Norme di Progetto - V 3.0*.

2.5 Strumenti di lavoro

Per gli strumenti di lavoro si faccia riferimento alle *Norme di Progetto - V 3.0*.

3 Specifica Back-end

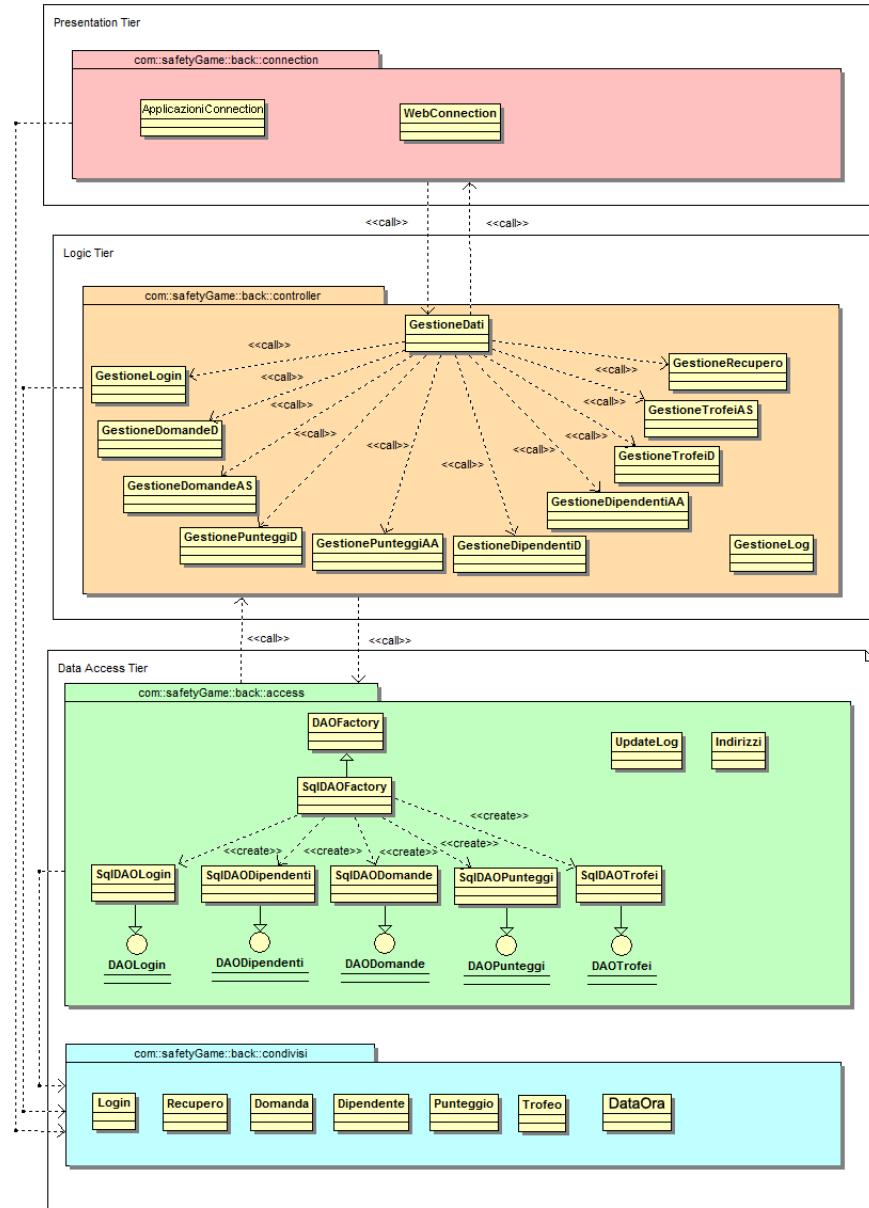


Figura 1: *Back-end, diagramma dei package*

3.1 Package com.safetyGame.back

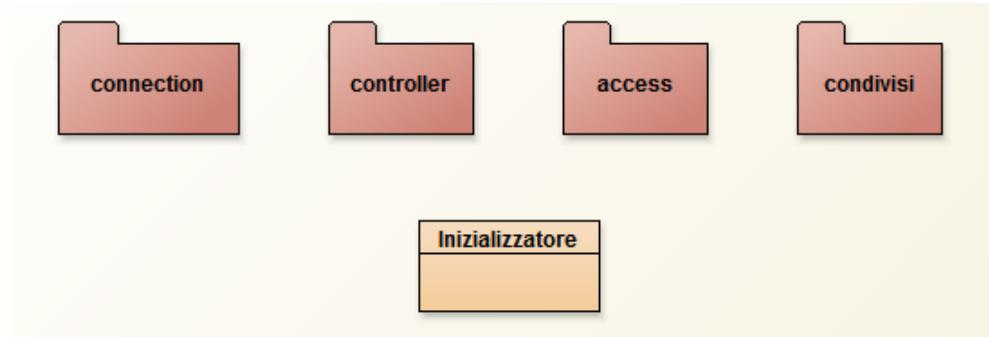


Figura 2: *Package com.safetyGame.back*

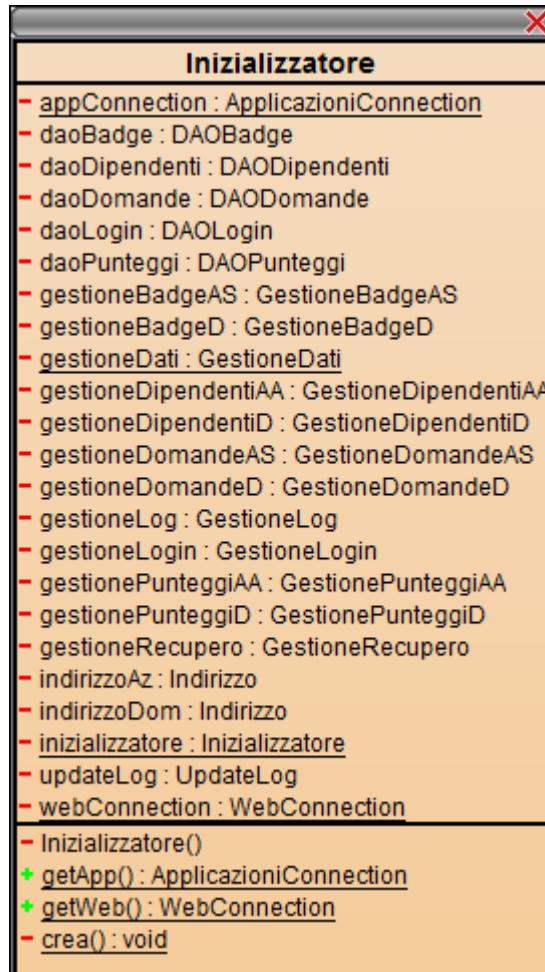
Tipo, obiettivo e funzione del componente: contiene gli oggetti deputati alla creazione dell'intero back-end

Relazioni d'uso di altre componenti: crea le classi dei package com.safetyGame.back.connection, com.safetyGame.back.controller, com.safetyGame.-back.access

Interfacce con e relazioni d'uso da altre componenti: viene utilizzato dal Front-end web per recuperare il puntatore alla classe back.connection.WebConnection e dal Front-end Mobile per recuperare il puntatore alla classe back.connection.ApplicationConnection

Attività svolte e dati trattati: fa da istanziatore del back-end

3.1.1 com.safetyGame.back.Inizializzatore



Funzione

Questa classe si occuperà di istanziare l'intero Back-end e di fornire i riferimenti alle classi `back.connection.WebConnection` e `back.connection.ApplicationConnection` agli opportuni richiedenti

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- `web.View.checkLogin`
- `web.View.checkRisposta`

- web.View.nuovaDomanda
- web.View.userPage

Inoltre utilizzerà le seguenti classi:

- back.connection.WebConnection
- back.connection.ApplicationConnection
- back.controller.GestioneBadgeAS
- back.controller.GestioneBadgeD
- back.controller.GestioneDati
- back.controller.GestioneDipendentiD
- back.controller.GestioneDomandeAS
- back.controller.GestioneDomandeD
- back.controller.GestioneLog
- back.controller.GestioneLogin
- back.controller.GestionePunteggiAA
- back.controller.GestionePunteggiD
- back.controller.GestioneRecupero
- back.access.DAOBadge
- back.access.DAODipendenti
- back.access.DAODomande
- back.access.DAOFactory
- back.access.DAOLogin
- back.access.DAOPunteggi
- back.access.Indirizzo
- back.access.SqlDAOBadge
- back.access.SqlDAODipendenti
- back.access.SqlDAODomande
- back.access.SqlDAOFactory
- back.access.SqlDAOLogin
- back.access.SqlDAOPunteggi
- back.access.UpdateLog

Attributi

- Indirizzo indirizzoAz

Riferimento all'istanza della classe *Indirizzo* contenente i riferimenti al database aziendale

- Indirizzo indirizzoDom

Riferimento all'istanza della classe *Indirizzo* contenente i riferimenti al database delle domande

- DAODipendenti daoDipendenti

Riferimento all'istanza della classe *DAODipendenti*, opportunamente istanziato alla corretta tipologia di database utilizzato

- DAOPunteggi daoPunteggi

Riferimento all'istanza della classe *DAOPunteggi*, opportunamente istanziato alla corretta tipologia di database utilizzato

- DAOLogin daoLogin

Riferimento all'istanza della classe *DAOLogin*, opportunamente istanziato alla corretta tipologia di database utilizzato

- UpdateLog updateLog

Riferimento all'istanza della classe *UpdateLog*

- DAODomande daoDomande

Riferimento all'istanza della classe *DAODomande*, opportunamente istanziato alla corretta tipologia di database utilizzato

- DAOBadge daoBadge

Riferimento all'istanza della classe *DAOBadge*, opportunamente istanziato alla corretta tipologia di database utilizzato

- GestioneRecupero gestioneRecupero

Riferimento all'istanza della classe *GestioneRecupero*

- GestionePunteggiD gestionePunteggiD

Riferimento all'istanza della classe *GestionePunteggiD*

- GestionePunteggiAA gestionePunteggiAA

Riferimento all'istanza della classe *GestionePunteggiAA*

- GestioneLog gestioneLog

Riferimento all'istanza della classe *GestioneLog*

- GestioneLogin gestioneLogin

Riferimento all'istanza della classe *GestioneLogin*

- **GestioneBadgeD gestioneBadgeD**
Riferimento all'istanza della classe *GestioneBadgeD*
- **GestioneDomandeD gestioneDomandeD**
Riferimento all'istanza della classe *GestioneDomandeD*
- **GestioneDomandeAS gestioneDomandeAS**
Riferimento all'istanza della classe *GestioneDomandeAS*
- **GestioneDipendentiD gestioneDipendentiD**
Riferimento all'istanza della classe *GestioneDipendentiD*
- **GestioneDipendentiAA gestioneDipendentiAA**
Riferimento all'istanza della classe *GestioneDipendentiAA*
- **GestioneBadgeAS gestioneBadgeAS**
Riferimento all'istanza della classe *GestioneBadgeAS*
- **GestioneDati gestioneDati**
Riferimento all'istanza della classe *GestioneDati*
- **static WebConnection webConnection = null**
Riferimento statico all'istanza della classe *WebConnection*
- **static ApplicazioniConnection appConnection = null**
Riferimento statico all'istanza della classe *appConnection*
- **static Inizializzatore inizializzatore = null**
Riferimento statico all'istanza della classe *Inizializzatore*

Metodi

- **Inizializzatore()**

Questo costruttore dovrà creare l'intero back-end. Per una corretta creazione del back-end, si dovrà seguire la seguente lista di creazione¹:

- *indirizzoAz:Indirizzo*
- *indirizzoDom:Indirizzo*
- *daoDipendenti:DAOPunteggi*
- *daoLogin:DAOLogin*

¹La lista è ordinata, per cui non è consentito la modifica dell'ordine di creazione; la lista è in formato <<NOME_VARIABILE>>:<<TIPO_VARIABILE>>



- updateLog:UpdateLog
- daoDomande:DAODomande
- daoBadge:DAOBadge
- gestioneRecupero:GestioneRecupero
- gestionePunteggiD:GestionePunteggiD
- gestionePunteggiAA:GestionePunteggiAA
- gestioneLog:GestioneLog
- gestioneLogin:GestioneLogin
- gestioneBadgeD:GestioneBadgeD
- gestioneDomandeD:GestioneDomandeD
- gestioneDomandeAS:GestioneDomandeAS
- gestioneDipendentiD:GestioneDipendentiD
- gestioneDipendentiAA:GestioneDipendentiAA
- gestioneBadgeAS:GestioneBadgeAS
- gestioneDati:GestioneDati
- webConnection:WebConnection
- appConnection:ApplicazioniConnection

+ **WebConnection getWeb()**

Dovrà controllare che la variabile statica *inizializzatore* sia già stata inizializzata: se non lo è stata, chiama la funzione crea(). Quindi ritorna il riferimento all'oggetto statico WebConnection

+ **static ApplicazioniConnection getApp()**

Dovrà controllare che la variabile statica *inizializzatore* sia già stata inizializzata: se non lo è stata, chiama la funzione crea(). Quindi ritorna il riferimento all'oggetto statico ApplicazioniConnection

- **static synchronized void crea()**

Dovrà controllare che la variabile statica *inizializzatore* sia già stata inizializzata: se non lo è stata, la inizializza creando una nuova istanza di inizializzatore e assegnandola alla variabile.

3.2 Package com.safetyGame.back.connection

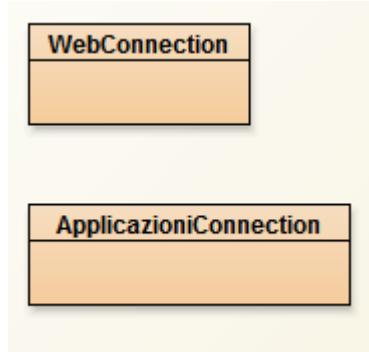


Figura 3: *Package com.safetyGame.back.connection*

Tipo, obiettivo e funzione del componente: contiene gli oggetti deputati all'interfacciamento dei vari front-end con il back-end

Relazioni d'uso di altre componenti: utilizza la classe **com.safetyGame.back.controller.GestioneDati** per inoltrare le richieste che provengono dai vari front-end verso gli strati inferiori del back-end

Interfacce con e relazioni d'uso da altre componenti: viene utilizzato dalla classe **com.safetyGame.back.controller.GestioneDati** per inoltrare le elaborazioni compiute dagli strati inferiori del back-end verso i vari front-end

Attività svolte e dati trattati: fa da connettore fra i vari front-end e il back-end, divenendo il componente *Presenter* del design pattern *MVP*

3.2.1 com.safetyGame.back.connection.WebConnection



Funzione

Questa classe si occuperà di presentare al front-end web tutte le funzionalità decise in fase di Progettazione Architetturale a cui ha accesso

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.Inizializzatore
- web.*

Inoltre utilizzerà le seguenti classi:

- back.controller.GestioneDati

Attributi

- GestioneDati dati



Riferimento alla classe façade *GestioneDati*

Metodi

+ **WebConnection(GestioneDati gestDati)**

Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

+ **boolean loginDip(String username, String password)**

Metodo per il login dei dipendenti.

Dovrà creare un oggetto *Login* a partire dai parametri passati e quindi chiamare la funzione della classe Façade deputata alla gestione del login.

+ **boolean loginAdmin(String username, String password, boolean tipo)**

Metodo per il login degli utenti amministratori.

Dovrà creare un oggetto *Login* a partire dai parametri passati e quindi chiamare la funzione della classe Façade deputata alla gestione del login.

+ **Dipendente getDati(Login login)**

Metodo che consente di reperire le informazioni di un dipendente a partire dal suo login.

Dovrà chiamare la funzione della classe Façade deputata al recupero dei dati di un *Dipendente*

+ **Dipendente getDatiA(Login login)**

Metodo che consente di reperire le informazioni di un amministratore a partire dal suo login.

Dovrà chiamare la funzione della classe Façade deputata al recupero dei dati di un *Amministratore*

+ **Punteggio getPunteggio(Login login)**

Metodo che consente di recuperare le statistiche di un determinato dipendente.

Dovrà chiamare la funzione della classe Façade deputata al recupero del punteggio di un *Dipendente*

+ **Punteggio getStat(Login login)**

Metodo che consente di recuperare le statistiche globali

Dovrà chiamare la funzione della classe Façade deputata al recupero delle statistiche dell'intera azienda

+ **ArrayList<Badge> getBadge(Login login, int num)**

Metodo per ottenere i dati dei badge per un dato utente

Dovrà chiamare la funzione della classe Façade deputata al recupero dei primi n *Badge* guadagnati da un *Dipendente*



+ **boolean modPassD(Dipendente dip)**

Metodo che consente la modifica della password da parte di un dipendente

Dovrà chiamare la funzione della classe Façade deputata alla modifica della password di un *Dipendente*

+ **boolean modPassA(Dipendente dip)**

Metodo che consente la modifica della password da parte di un amministratore

Dovrà chiamare la funzione della classe Façade deputata alla modifica della password di un *Amministratore*

+ **boolean modMail(Dipendente dip, String mail)**

Metodo che consente la modifica della mail da parte di un dipendente

Dovrà chiamare la funzione della classe Façade deputata alla modifica dell'indirizzo email di un *Dipendente*

+ **boolean resetPassD(Recupero recupero)**

Metodo che consente la rigenerazione della password per un dipendente

Dovrà chiamare la funzione della classe Façade deputata alla rigenerazione della password di un *Dipendente*

+ **boolean resetPassA(Recupero recupero)**

Metodo che consente la rigenerazione della password per un amministratore

Dovrà chiamare la funzione della classe Façade deputata alla modifica della password di un *Amministratore*

+ **Domanda mostraDomanda(Login login)**

Metodo che consente di recuperare una domanda

Dovrà chiamare la funzione della classe Façade deputata al recupero di una nuova domanda da sottoporre ad un *Dipendente*

+ **boolean setRisposta(Login login, Domanda risposta)**

Metodo che si occupa di comunicare la risposta data

Dovrà chiamare la funzione della classe Façade deputata al controllo della correttezza della risposta data ad una *Domanda* da un *Dipendente*

+ **boolean posticipa(Login login, Domanda domPost)**

Metodo per posticipare una domanda

Dovrà chiamare la funzione della classe Façade deputata al posticipo della *Domanda* sottoposta ad un *Dipendente*

+ **void logoutD(Login login)**

Metodo per segnalare al sistema il logout di un dipendente.

Dovrà chiamare la funzione della classe Façade deputata alla gestione della richiesta di logout da parte di un utente

+ void logoutA(Login login)

Metodo per segnalare al sistema il logout di un amministratore.

Dovrà chiamare la funzione della classe Façade deputata alla gestione della richiesta di logout da parte di un utente

+ ArrayList<Domanda> getElencoDomande()

Metodo per ottenere la lista di tutte le domande

Dovrà chiamare la funzione della classe Façade deputata alla generazione dell'elenco di tutte le domande selezionate dall'Amministratore Sicurezza per l'azienda

+ boolean aggiungiDomanda(Domanda dom)

Metodo per inserire una domanda dal server domande al server dell'azienda

Dovrà chiamare la funzione della classe Façade deputata all'inserimento di una nuova domanda all'interno dell'elenco delle domande selezionate dall'Amministratore Sicurezza per l'azienda

+ boolean cancellaDomanda(Domanda dom)

Metodo per eliminare una domanda dal server dell'azienda

Dovrà chiamare la funzione della classe Façade deputata alla rimozione di una domanda all'interno dell'elenco delle domande selezionate dall'Amministratore Sicurezza per l'azienda

+ ArrayList<Dipendente> getElencoDipendenti()

Metodo per ottenere i dati dei dipendenti dell'azienda

Dovrà chiamare la funzione della classe Façade deputata al recupero dell'elenco di tutti i *Dipendenti* registrati nel sistema

+ boolean setTrofei(Dipendente dip, int num)

Metodo per modificare il numero di trofei di un dipendente

Dovrà chiamare la funzione della classe Façade deputata alla modifica del numero di trofei di un *Dipendente*

+ boolean aggiungiDipendente(Dipendente dip)

Metodo per aggiungere un dipendente

Dovrà chiamare la funzione della classe Façade deputata all'aggiunta di un *Dipendente* all'interno del database

+ boolean cancellaDipendente(Dipendente dip)

Metodo per eliminare un Dipendente

Dovrà chiamare la funzione della classe Façade deputata alla rimozione di un *Dipendente* all'interno del database

+ boolean modDipendente(Dipendente dipOld, Dipendente dipNew)

Metodo per modificare i dati di un dipendente



Dovrà chiamare la funzione della classe Façade deputata alla modifica di un *Dipendente* all'interno del database

+ **ArrayList<Badge> getBadgesAS()**

Metodo per ottenere tutti i badge

Dovrà chiamare la funzione della classe Façade deputata al recupero di tutti i *Badge* inseriti nel database

+ **boolean assegnaBadge(Domanda dom, Login login)**

Metodo per assegnare un badge

Dovrà chiamare la funzione della classe Façade deputata all'assegnazione di un *Badge* ad un *Dipendente*

+ **ArrayList<Dipendente> getPunteggi()**

Metodo per ottenere i punteggi medi dell'azienda e i punteggi di tutti i dipendenti

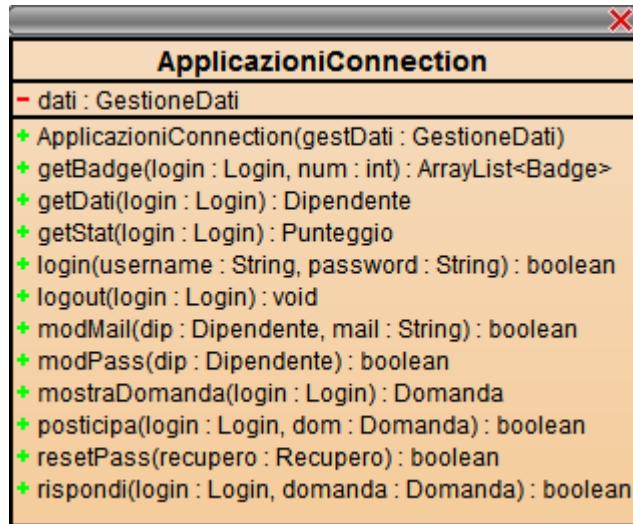
Dovrà chiamare la funzione della classe Façade deputata al recupero dei punteggi di tutti i *Dipendenti*, oltre che ai punteggi medi dell'azienda.

+ **ArrayList<String> getElencoRuoli()**

Metodo per recuperare la lista dei ruoli aziendali.

Dovrà chiamare la funzione della classe Façade deputata al recupero dei ruoli presenti all'interno dell'azienda.

3.2.2 com.safetyGame.back.connection.ApplicazioniConnection



Funzione

Questa classe si occuperà di presentare al front-end mobile e desktop tutte le funzionalità decise in fase di Progettazione Architetturale a cui ha accesso

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.Inizializzatore
- web.API.*

Inoltre utilizzerà le seguenti classi:

- back.controller.GestioneDati

Attributi

- GestioneDati dati

Riferimento alla classe façade *GestioneDati*

Metodi

+ ApplicazioniConnection(GestioneDati gestDati)

Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

+ boolean login(String username, String password)

Metodo per il login.

Dovrà creare un oggetto *Login* a partire dai parametri passati e quindi chiamare la funzione della classe Façade deputata alla gestione del login. Poiché l'uso del front-end mobile e desktop è riservato ai *Dipendenti*, questo metodo dovrà chiamare il metodo per il login dei *Dipendenti*

+ Dipendente getDati(Login login)

Metodo che consente di reperire le informazioni di un dipendente a partire dal suo login.

Dovrà chiamare la funzione della classe Façade deputata al recupero dei dati di un *Dipendente*

+ Punteggio getStat(Login login)

Metodo che consente di recuperare le statistiche globali

Dovrà chiamare la funzione della classe Façade deputata al recupero delle statistiche dell'intera azienda

+ void modPass(Dipendente dip)

Metodo che consente la modifica della password da parte di un dipendente

Dovrà chiamare la funzione della classe Façade deputata alla modifica della password di un *Dipendente*

+ void modMail(Dipendente dip, String mail)

Metodo che consente la modifica della mail da parte di un dipendente

Dovrà chiamare la funzione della classe Façade deputata alla modifica dell'indirizzo email di un *Dipendente*

+ void resetPass(Recupero recupero)

Metodo che consente la rigenerazione della password per un dipendente

Dovrà chiamare la funzione della classe Façade deputata alla rigenerazione della password di un *Dipendente*, visto che l'uso del front-end mobile e desktop è riservato a questi ultimi.

+ Domanda mostraDomanda(Login login)

Metodo che consente di recuperare una domanda

Dovrà chiamare la funzione della classe Façade deputata al recupero di una nuova domanda da sottoporre ad un *Dipendente*

+ void posticipa(Login login, Domanda dom)

Metodo per posticipare una domanda

Dovrà chiamare la funzione della classe Façade deputata al posticipo della *Domanda* sottoposta ad un *Dipendente*

+ boolean rispondi(Login login, Domanda domanda)



Metodo che si occupa di comunicare la risposta data

Dovrà chiamare la funzione della classe Façade deputata al controllo della correttezza della risposta data ad una *Domanda* da un *Dipendente*

+ **void logoutD(Login login)**

Metodo per segnalare al sistema il logout di un dipendente

Dovrà chiamare la funzione della classe Façade deputata alla gestione della richiesta di logout da parte di un utente

+ **void logoutD(Login login)**

Metodo per segnalare al sistema il logout di un amministratore

Dovrà chiamare la funzione della classe Façade deputata alla gestione della richiesta di logout da parte di un utente

+ **ArrayList<Badge> getBadge(Login login, int num)**

Metodo per ottenere i dati dei badge per un dato utente

Dovrà chiamare la funzione della classe Façade deputata al recupero dei primi n *Badge* guadagnati da un *Dipendente*

3.3 Package com.safetyGame.back.controller

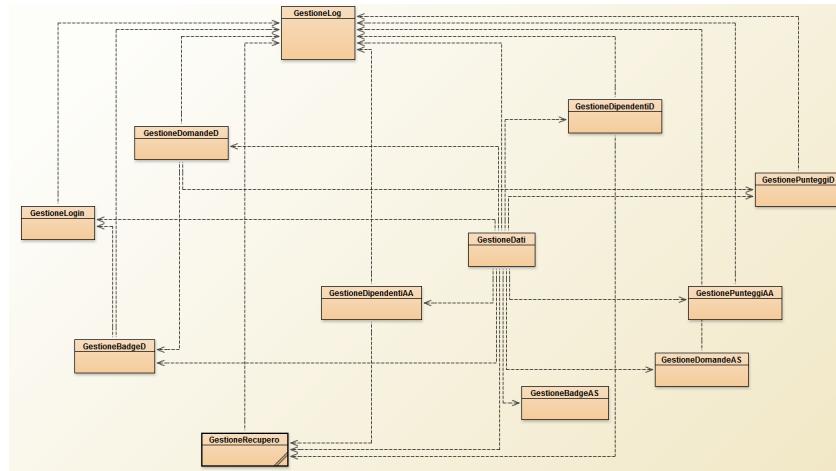


Figura 4: *Package com.safetyGame.back.controller*

Tipo, obiettivo e funzione del componente: contiene gli oggetti deputati al controllo dei dati immessi nei vari front-end e del loro eventuale indirizzamento verso l'*Access-Tier* per la memorizzazione. Inoltre si occupa di chiedere all'*Access-Tier* di recuperare dati richiesti e di poi inviarli al *Presentation-Tier* affinché vengano spediti al front-end di competenza

Relazioni d'uso di altre componenti: utilizza la classe `com.safetyGame.back.access` per memorizzare i dati da lui elaborati

Interfacce con e relazioni d'uso da altre componenti: viene utilizzato dal package `com.safetyGame.back.connection` per elaborare i dati in arrivo ed in invio da e per i vari front-end

Attività svolte e dati trattati²: è deputato alla computazione su tutti i dati da e per i front-end

²Attenzione: dal diagramma si può notare un forte accoppiamento fra le classi del package. Questo è dovuto alla presenza di una classe che implementa il Design Pattern Façade, che implica quindi un alto grado di accoppiamento [vedere Specifica Tecnica in proposito]

3.3.1 com.safetyGame.back.controller.GestioneBadgeAS



Funzione

Questa classe si occuperà di gestire le operazioni che l'Amministratore Sicurezza può effettuare sui Badge. In particolare potrà recuperare la lista di tutti i badge disponibili all'interno dell'azienda.

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.controller.GestioneDati

Inoltre utilizzerà le seguenti classi:

- back.access.DAOBadge
- back.condivisi.Badge

Attributi

- DAOBadge **accessBadge**

Riferimento all'istanza della classe *DAOBadge*, istanziato nella corretta classe concreta secondo il database utilizzato

Metodi

+ GestioneBadgeAS(DAOBadge **accessBadge**)

Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

+ ArrayList<Badge> **getBadgesAS()**

Metodo per ottenere tutti i badge possibili.

Dovrà interrogare l'*Access-Tier* affinché ritorni un array con gli oggetti *Badge* che rappresenteranno tutti i badge presenti nel database.

3.3.2 com.safetyGame.back.controller.GestioneBadgeD



Funzione

Questa classe si occuperà di gestire le operazioni che i Dipendenti possono effettuare sui propri Badge. In particolare:

- **Ottenere** la lista dei badge fino ad ora acquisiti
- **Controllare** che, rispondendo ad una domanda, si abbia diritto ad acquisire uno o più nuovi badge

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.controller.GestioneDati
- back.controller.GestioneDomandeD

Inoltre utilizzerà le seguenti classi:

- back.access.DAOBadge
- back.access.DAODipendenti
- back.access.DAODomande
- back.controller.GestioneLog
- back.controller.GestioneLogin
- back.condivisi.Badge
- back.condivisi.Login
- back.condivisi.Dipendente
- back.condivisi.Domanda

Attributi

- DAOBadge accessBadge

Riferimento all'istanza della classe *DAOBadge*, istanziato nella corretta classe concreta secondo il database utilizzato

- DAODipendenti accessDip

Riferimento all'istanza della classe *DAODipendenti*, istanziato nella corretta classe concreta secondo il database utilizzato

- DAODomande accessDom

Riferimento all'istanza della classe *DAODomande*, istanziato nella corretta classe concreta secondo il database utilizzato

- GestioneLog gestLog

Riferimento all'istanza della classe *GestioneLog*

- GestioneLogin gestLogin

Riferimento all'istanza della classe *GestioneLogin*

Metodi

+ GestioneBadgeD(DAOBadge accessBadge, DAODipendenti accesSip, DAODomande accessDom, GestioneLog gestlog, GestioneLogin gestlogin)

Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

+ ArrayList<Badge> getBadgeD(Login login, int num)

Metodo per ottenere i dati dei badge per un dato utente.

Dovrà verificare che esista un *Dipendente* collegato alla *Login* passata come parametro. Se esiste, allora dovrà recuperare l'oggetto *Dipendente* e la lista dei suoi badge, quindi seleziona i primi *n* Badge e li restituisce. Nel caso non esistesse il *Dipendente* o non avesse alcun *Badge*, la funzione dovrà ritornare **null**.

+ boolean assegnaBadge(Domanda domanda, Login login)

Metodo per controllare se l'utente ha soddisfatto dei requisiti per ottenere un badge.

Dovrà verificare che esista un *Dipendente* collegato alla *Login* passata come parametro. Se esiste, allora dovrà controllare tutti i possibili *Badge* assegnabili al *Dipendente* e, nel caso ne trovasse almeno uno, dovrà chiamare la funzione deputata all'assegnazione di tale *Badge* nella classe *DAOBadge* e chiamare la classe *GestioneLog* deputata alla scrittura sul file di log di tale assegnazione.

Se dovesse aver assegnato almeno un *Badge*, la funzione dovrà ritornare **true**, altrimenti **false**.



3.3.3 com.safetyGame.back.controller.GestioneDati



Funzione

Questa classe è la componente principale del Design Pattern Façade. Contiene tutti i prototipi dei metodi pubblici delle altre classi contenute nel package e la funzione principale della classe *GestioneDati* è quella di reindirizzare le chiamate alle classi competenti.

RelazionRiferimento all’istanza della classe *GestioneLogi* d’uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.connection.ApplicazioniConnection
 - back.connection.WebConnection

Inoltre utilizzerà le seguenti classi:

- back.controller.GestioneRecupero
 - back.controller.GestioneLogin
 - back.controller.GestioneDomandeD
 - back.controller.GestioneDomandeAS
 - back.controller.GestioneDipendentiD
 - back.controller.GestioneDipendentiAA
 - back.controller.GestioneBadgeD
 - back.controller.GestioneBadgeAS
 - back.controller.GestionePunteggiD
 - back.controller.GestionePunteggiAA

- back.condivisi.Badge
- back.condivisi.Login
- back.condivisi.Dipendente

Attributi

- **GestioneRecupero gestioneRecupero**
Riferimento all'istanza della classe *GestioneRecupero*
- **GestioneLogin gestioneLogin**
Riferimento all'istanza della classe *GestioneLogin*
- **GestioneDomandeD gestioneDomandeD**
Riferimento all'istanza della classe *GestioneDomandeD*
- **GestioneDomandeAS gestioneDomandeAS**
Riferimento all'istanza della classe *GestioneDomandeAS*
- **GestioneDipendentiD gestioneDipendentiD**
Riferimento all'istanza della classe *GestioneDipendentiD*
- **GestioneDipendentiAA gestioneDipendentiAA**
Riferimento all'istanza della classe *GestioneDipendentiAA*
- **GestioneBadgeD gestioneBadgeD**
Riferimento all'istanza della classe *GestioneBadgeD*
- **GestioneBadgeAS gestioneBadgeAS**
Riferimento all'istanza della classe *GestioneBadgeAS*
- **GestionePunteggiD gestionePunteggiD**
Riferimento all'istanza della classe *GestionePunteggiD*
- **GestionePunteggiAA gestionePunteggiAA**
Riferimento all'istanza della classe *GestionePunteggiAA*

Metodi

- + GestioneDati(GestioneRecupero gestRec, GestioneLogin gestLogin, GestioneDomandeD gestDomD, GestioneDomandeAS gestDomAS, GestioneDipendentiD gestDipD, GestioneDipendentiAA gestDipAA, GestioneBadgeD gestBadgeD, GestioneBadgeAS gestBadgeAS, GestionePunteggiD gestPuntD, GestionePunteggiAA gestPuntAA)

Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

+ **ArrayList<Badge> getBadgesAS()**

Metodo per ottenere tutti i badge possibili.

Dovrà interrogare la classe *GestioneBadgeAS* affinché ritorni un array con gli oggetti *Badge* che rappresenteranno tutti i badge presenti nel database.

+ **ArrayList<Badge> getBadgeD(Login login, int numeroBadge)**

Metodo per ottenere i dati delle badge per un dato utente.

Dovrà interrogare la classe *GestioneBadgeD* affinché ritorni un array con gli oggetti *Badge* che rappresenteranno tutti i badge guadagnati da un *Dipendente* rappresentato dall'oggetto *Login*.

+ **boolean assegnaBadge(Domanda domanda, Login login)**

Metodo per controllare se l'utente ha soddisfatto dei requisiti per ottenere un badge.

Dovrà interrogare la classe *GestioneBadgeD* affinché assegni un nuovo badge al *Dipendente* data la *Domanda* in oggetto.

+ **ArrayList<Dipendente> getElencoDipendenti()**

Metodo per ottenere i dati dei dipendenti dell'azienda.

Dovrà interrogare la classe *GestioneDipendentiAA* affinché ritorni un array contenente oggetti *Dipendente* rappresentazione di tutti i Dipendenti presenti in database

+ **boolean aggiungiDipendente(Dipendente Dip)**

Metodo per aggiungere un dipendente.

Dovrà chiamare la funzione contenuta in *GestioneDipendentiAA* atta ad inserire il *Dipendente* in oggetto all'interno del database.

+ **boolean cancellaDipendente(Dipendente Dip)**

Metodo per eliminare un dipendente.

Dovrà chiamare la funzione contenuta in *GestioneDipendentiAA* atta a eliminare il *Dipendente* in oggetto all'interno del database.

+ **boolean modDipendente(Dipendente newDip, Dipendente oldDip)**

Metodo per modificare i dati di un dipendente.

Dovrà chiamare la funzione contenuta in *GestioneDipendentiAA* atta a modificare i dati del *Dipendente oldDip* con quelli contenuti in *newDip* all'interno del database.

+ **Dipendente getDati(Login login)**

Metodo che consente di reperire le informazioni di un dipendente a partire dal suo login.

Dovrà chiamare la funzione contenuta in *GestioneDipendentiD* atta a reperire i dati del *Dipendente*, identificato dall'oggetto *Login* passato come parametro, all'interno del database.

+ **Dipendente getDatiA(Login login)**

Metodo che consente di reperire le informazioni di un amministratore a partire dal suo login.

Dovrà chiamare la funzione contenuta in *GestioneDipendentiAA* atta a reperire i dati di un *Amministratore*, identificato dall'oggetto *Login* passato come parametro, all'interno del database.

+ **boolean modificaPass(Dipendente dip)**

Metodo che consente la modifica della password da parte di un dipendente.

Dovrà chiamare la funzione contenuta in *GestioneDipendentiD* atta a modificare la password del *Dipendente* all'interno del database.

+ **boolean modificaEmail(Dipendente dip, String nEmail)**

Metodo che consente la modifica della mail da parte di un dipendente.

Dovrà chiamare la funzione contenuta in *GestioneDipendentiD* atta a modificare l'indirizzo email del *Dipendente* all'interno del database.

+ **ArrayList<Domanda> getElencoDomande()**

Metodo per ottenere la lista di tutte le domande.

Dovrà interrogare la classe *GestioneDomandeAS* affinché ritorni un array contenente oggetti *Domanda* rappresentazione di tutte le Domande scelte dall'Amministratore Sicurezza all'interno del database centrale per l'azienda.

+ **boolean addDomanda(Domanda Dom)**

Metodo per inserire una domanda dal server domande al server dell'azienda.

Dovrà chiamare la funzione contenuta in *GestioneDomandeAS* atta ad inserire una nuova *Domanda* fra quelle proponibili ai *Dipendenti* dell'azienda.

+ **boolean remDomanda(Domanda Dom)**

Metodo per eliminare una domanda dal server dell'azienda.

Dovrà chiamare la funzione contenuta in *GestioneDomandeAS* atta a rimuovere una *Domanda* fra quelle proponibili ai *Dipendenti* dell'azienda.

+ **Domanda getDomandaD(Login login)**

Metodo che consente di recuperare una domanda.

Dovrà chiamare la funzione contenuta in *GestioneDomandeD* atta a recuperare una domanda a cui il *Dipendente*, identificato dall'oggetto *Login*, o non ha ancora visualizzato, o ha chiesto di posticiparla.

+ **boolean setRisposta(Login login, Domanda risposta)**

Metodo che si occupa di controllare la risposta data da un dipendente ad una domanda e tenta di scrivere tali informazioni sul database. Se la risposta è corretta assegna il punteggio al dipendente.

Dovrà chiamare la funzione contenuta in *GestioneDomandeD* atta a controllare che la risposta data dal *Dipendente*, identificato dall'oggetto *Login* una domanda e tenta di scrivere tali informazioni sul database. Se la risposta è corretta assegna il punteggio al dipendente.

+ **boolean loginAdmin(Login login, boolean tipo)**

Metodo per il login degli utenti amministratori.

Dovrà “passare la chiamata” al metodo deputato alla gestione dei tentativi di login da parte degli *Amministratori* nella classe *GestioneLogin*.

+ **boolean loginUser(Login login)**

Metodo per il login dei Dipendenti.

Dovrà “passare la chiamata” al metodo deputato alla gestione dei tentativi di login da parte dei *Dipendenti* nella classe *GestioneLogin*.

+ **ArrayList<Dipendente> getPunteggi()**

Metodo per ottenere i punteggi medi dell'azienda e i punteggi di tutti i dipendenti.

Dovrà “passare la chiamata” al metodo deputato al recupero di tutti i punteggi dei *Dipendenti* nella classe *GestionePunteggiAA*.

+ **boolean setTrofei(Dipendente dip, int numTrofei)**

Metodo per modificare i trofei di un dipendente.

Dovrà “passare la chiamata” al metodo deputato alla modifica del numero di trofei di un *Dipendente* nella classe *GestionePunteggiAA*.

+ **Punteggio getStatisticheD(Login login)**

Metodo che consente di recuperare le statistiche di un determinato dipendente.

Dovrà “passare la chiamata” al metodo deputato al recupero delle statistiche di un *Dipendente* nella classe *GestionePunteggiD*.

+ **boolean posticipa(Login login, Domanda dom)**

Metodo che si occupa di controllare quando una domanda viene posticipata.

Dovrà “passare la chiamata” al metodo deputato alla posticipazione di una domanda sottoposta ad un *Dipendente* nella classe *GestionePunteggiD*.

+ **void logoutD(Login login)**

Metodo per segnalare al sistema il logout di un dipendente.

Dovrà “passare la chiamata” al metodo deputato alla gestione dei logout nella classe *GestioneLogin*.

+ **void logoutA(Login login)**

Metodo per segnalare al sistema il logout di un amministratore.

Dovrà “passare la chiamata” al metodo deputato alla gestione dei logout nella classe *GestioneLogin*.

+ **Punteggio getStatisticheGlob(Login login)**

Metodo che consente di recuperare le statistiche globali.

Dovrà “passare la chiamata” al metodo deputato al recupero delle statistiche dell’intera azienda all’interno di *GestionePunteggiD* a partire dal *Dipendente* identificato tramite l’oggetto *Login* passato come parametro.

+ **boolean recuperoD(Recupero dip)**

Metodo che consente ad un Dipendente di resettare la propria password.

Dovrà “passare la chiamata” al metodo deputato al reset della password di un *Dipendente* all’interno della classe *GestioneRecupero*

+ **boolean recuperoA(Recupero amm)**

Metodo che consente ad un amministratore di resettare la propria password.

Dovrà “passare la chiamata” al metodo deputato al reset della password di un *Amministratore* all’interno della classe *GestioneRecupero*.

+ **boolean modPassA(Dipendente admin)**

Metodo per modificare la password di un amministratore

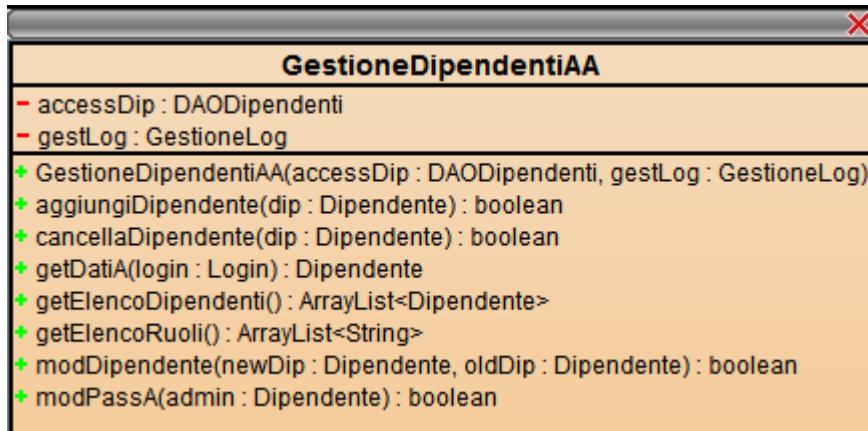
Dovrà chiamare la funzione contenuta in *GestioneDipendentiD* atta a modificare la password del *Dipendente* all’interno del database.

+ **ArrayList<String> getElencoRuoli()**

Metodo che consente di recuperare la lista dei ruoli aziendali.

Dovrà chiamare la funzione contenuta in *GestioneDipendentiAA* deputata al recupero di tutti i ruoli assegnabili ad un *Dipendente* inseriti all’interno del database.

3.3.4 com.safetyGame.back.controller.GestioneDipendentiAA



Funzione

Questa classe si occuperà di gestire tutte le operazioni che un Amministratore Azienda potrà effettuare per gestire gli account sia Dipendente che Amministratore

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.controller.GestioneDati

Inoltre utilizzerà le seguenti classi:

- back.access.DAODipendenti
- back.condivisi.Dipendente
- back.controller.GestioneLog

Attributi

- DAODipendenti accessDip

Riferimento all'istanza della classe *DAODipendenti*, opportunamente istanziato alla corretta tipologia di database utilizzato

- GestioneLog gestLog

Riferimento all'istanza della classe *GestioneLog*

Metodi

+ **GestioneDipendentiAA(DAO_{Dipendenti} accessDip, GestioneLog gestLog)**

Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

+ **ArrayList<Dipendente> getElencoDipendenti()**

Metodo per ottenere i dati dei dipendenti dell'azienda.

Dovrà interrogare l'*Access-Tier* affinché restituisca un array contenente tutti gli oggetti *Dipendente* relativi ai Dipendenti registrati nel sistema.

+ **boolean aggiungiDipendente(Dipendente Dip)**

Metodo per aggiungere un dipendente.

Dovrà interrogare l'*Access-Tier* affinché inserisca il contenuto dell'oggetto *Dipendente* passato come parametro all'interno del database.

+ **boolean cancellaDipendente(Dipendente Dip)**

Metodo per eliminare un dipendente.

Dovrà interrogare l'*Access-Tier* affinché elimini il *Dipendente* passato come parametro dall'interno del database.

+ **boolean modDipendente(Dipendente newDip, Dipendente oldDip)**

Metodo per modificare i dati di un dipendente.

Dovrà interrogare l'*Access-Tier* affinché modifichi i dati del *Dipendente* all'interno del database.

+ **boolean modPassA(Dipendente admin)**

Metodo per modificare la password di un amministratore.

Dovrà interrogare l'*Access-Tier* affinché modifichi i dati dell'*Amministratore* all'interno del database.

+ **Dipendente getDatiA(Login login)**

Metodo che consente di reperire le informazioni di un amministratore da parte dal suo login.

Dovrà interrogare l'*Access-Tier* affinchè reperisca le informazioni di un amministratore inserite all'interno del database.

+ **ArrayList<String> getElencoRuoli()**

Metodo per recuperare l'elenco dei ruoli di un'azienda.

Dovrà interrogare l'*Access-Tier* affinché restituisca un *ArrayList<String>* contenente tutti i ruoli inseriti in database, quindi dovrà restituirlo al chiamante.

3.3.5 com.safetyGame.back.controller.GestioneDipendentiD

GestioneDipendentiD	
-	daoDipendenti : DAODipendenti
-	gestioneLog : GestioneLog
-	gestioneRecupero : GestioneRecupero
+	GestioneDipendentiD(daoDipendenti : DAODipendenti, gestioneLog : GestioneLog)
+	GestioneDipendentiD()
+	getDaoDipendenti() : DAODipendenti
+	getDatiD(login : Login) : Dipendente
+	getGestioneLog() : GestioneLog
+	modificaEmail(dip : Dipendente, nEmail : String) : boolean
+	modificaPass(dip : Dipendente) : boolean
+	setDaoDipendenti(daoDip : DAODipendenti) : void
+	setGestioneLog(gestioneLog : GestioneLog) : void

Funzione

Questa classe si occuperà di gestire tutte le operazioni che un Dipendente può effettuare sul suo account.

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.controller.GestioneDati

Inoltre utilizzerà le seguenti classi:

- back.access.DAODipendenti
- back.controller.GestioneLog
- back.condivisi.Login
- back.condivisi.Dipendente

Attributi

- DAODipendenti daoDipendenti

Riferimento all'istanza della classe *DAODipendenti*, opportunamente istanziato alla corretta tipologia di database utilizzato

- GestioneLog gestioneLog

Riferimento all'istanza della classe *GestioneLog*

- GestioneRecupero gestioneRecupero

Riferimento all'istanza della classe *GestioneRecupero*

Metodi

+ **GestioneDipendentiD(DAO_{Dipendenti} daoDipendenti, GestioneLog gestioneLog)**

Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

+ **Dipendente getDatiD(Login login)**

Metodo che consente di reperire le informazioni di un dipendente a partire dal suo login.

Dovrà interrogare l'*Access-Tier* affinché gli ritorni un oggetto *Dipendente* costruito a partire dal *Login* passato come parametro.

+ **boolean modificaPass(Dipendente dip)**

Metodo che consente la modifica della password da parte di un dipendente.

Dovrà interrogare l'*Access-Tier* affinché modifichi la password del *Dipendente* con quella contenuta all'interno dell'oggetto *Dipendente* passato come parametro. Se non ci sono problemi, dovrà chiamare la classe deputata alla gestione del file di log affinché scriva che il *Dipendente* ha modificato la sua password.

+ **boolean modificaEmail(Dipendente dip, String nEmail)**

Metodo che consente la modifica della mail da parte di un dipendente.

Dovrà interrogare l'*Access-Tier* affinché modifichi la mail del *Dipendente* con la stringa passata in oggetto. Quindi, se non ci sono problemi, dovrà chiamare la classe deputata alla gestione del file di log affinché scriva che il *Dipendente* ha modificato il suo indirizzo email.

Vengono quindi definiti i metodi getter/setter per i vari attributi della classe. In particolare:

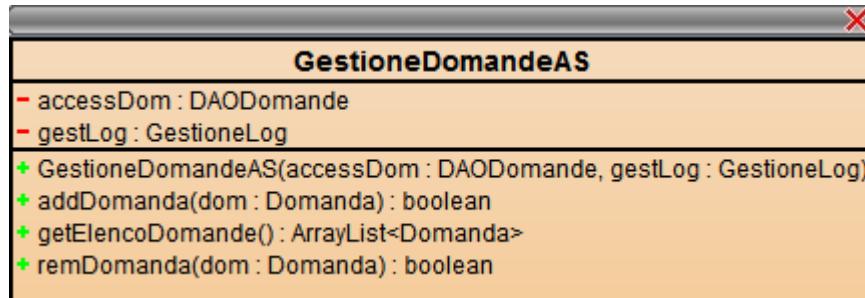
+ **DAO_{Dipendenti} getDaoDipendenti()**

+ **void setDaoDipendenti(DAO_{Dipendenti} daoDip)**

+ **GestioneLog getGestioneLog()**

+ **void setGestioneLog(GestioneLog gestioneLog)**

3.3.6 com.safetyGame.back.controller.GestioneDomandeAS



Funzione

Questa classe si occuperà di gestire tutte le operazioni che un Amministratore Sicurezza può effettuare sull'elenco delle domande.

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.controller.GestioneDati

Inoltre utilizzerà le seguenti classi:

- back.access.DAODomande
- back.condivisi.Domanda
- back.controller.GestioneLog

Attributi

- DAODomande accessDom

Riferimento all'istanza della classe *DAODipendenti*, opportunamente istanziato alla corretta tipologia di database utilizzato

- GestioneLog gestLog

Riferimento all'istanza della classe *GestioneLog*

Metodi

+ GestioneDomandeAS(DAODomande accessDom, GestioneLog gestLog)

Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

+ ArrayList<Domanda> getElencoDomande()



Metodo per ottenere la lista di tutte le domande.

Dovrà chiamare l'*Access-Tier* affinché gli restituisca un array di tutte le domande selezionate fino ad ora dall'*Amministratore* fra quelle disponibili nel database centrale.

+ **boolean addDomanda(Domanda Dom)**

Metodo per inserire una domanda dal server domande al server dell'azienda.

Dovrà chiamare l'*Access-Tier* affinché inserisca la *Domanda* all'interno delle domande selezionate dall'*Amministratore*.

+ **boolean remDomanda(Domanda Dom)**

Metodo per eliminare una domanda dal server dell'azienda.

Dovrà chiamare l'*Access-Tier* affinché elimini la *Domanda* dalle domande selezionate dall'*Amministratore*.

3.3.7 com.safetyGame.back.controller.GestioneDomandeD



Funzione

Questa classe si occuperà di gestire tutte le operazioni che i Dipendenti possono effettuare sulle domande, in particolare:

- **Rispondere** ad una domanda
- **Ottenere** una nuova domanda
- **Posticipare** una domanda

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- `back.controller.GestioneDati`

Inoltre utilizzerà le seguenti classi:

- `back.access.DAODomande`
- `back.access.DAOPunteggi`
- `back.access.DAODependentI`
- `back.controller.GestionePunteggiD`
- `back.controller.GestioneLog`
- `back.controller.GestioneBadgeD`
- `back.condivisi.Login`
- `back.condivisi.Dipendente`
- `back.condivisi.Domanda`

Attributi

- DAODomande daoDomande

Riferimento all'istanza della classe *DAODomande*, opportunamente istanziato alla corretta tipologia di database utilizzato

- DAOPunteggi daoPunteggi

Riferimento all'istanza della classe *DAOPunteggi*, opportunamente istanziato alla corretta tipologia di database utilizzato

- DAODipendenti daoDipendenti

Riferimento all'istanza della classe *DAODipendenti*, opportunamente istanziato alla corretta tipologia di database utilizzato

- GestionePunteggiD gestionePunteggiD

Riferimento all'istanza della classe *GestionePunteggiD*

- GestioneLog gestioneLog

Riferimento all'istanza della classe *GestioneLog*

- GestioneBadgeD gestioneBadge

Riferimento all'istanza della classe *GestioneBadgeD*

Metodi

+ GestioneDomandeD(DAODomande daoDom, DAOPunteggi daoPunt, DAODipendenti daoDip, GestionePunteggiD gestPuntD, GestioneLog gestLog, GestioneBadgeD gestBadgeD)

Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

+ Domanda getDomandaD(Login login)

Metodo che consente di recuperare una domanda.

Dovrà prelevare l'utente caratterizzato dalla *Login* passata come parametro e, nel caso in cui venga trovato, dovrà interrogare il database in modo tale da fornire una *Domanda* candidata ad avere risposta da parte del *Dipendente*. Quindi, nel caso venga trovata una *Domanda*, dovrà chiamare l'*Access-Tier* affinché scriva nel database e la classe deputata alla gestione dei log affinché scriva nel file di log che è stata sottoposta una nuova *Domanda* al *Dipendente*.

+ boolean setRisposta(Login login, Domanda risposta)

Metodo che si occupa di controllare la risposta data da un dipendente ad una domanda e tenta di scrivere tali informazioni sul database. Se la risposta è corretta assegna il punteggio al dipendente.

Dovrà prelevare l'utente caratterizzato dalla *Login* passata come parametro e, nel caso in cui venga trovato, dovrà interrogare il database in modo tale che

scriva che alla *Domanda* è stata data risposta. Quindi, nel caso in cui sia stata data risposta corretta, dovrà controllare che nel database esista un *Badge* assegnabile³. Infine dovrà scrivere nel file di log che il *Dipendente* ha risposto alla *Domanda*.

+ **boolean posticipa(Login login, Domanda dom)**

Metodo che si occupa di controllare quando una domanda viene posticipata.

Dovrà prelevare l'utente caratterizzato dalla *Login* passata come parametro e, nel caso in cui venga trovato, dovrà interrogare il database in modo tale che scriva che la *Domanda* è stata posticipata. Nel caso non ci siano problemi, dovrà quindi scrivere nel file di log che la *Domanda* è stata posticipata dal *Dipendente*.

Vengono quindi definiti i metodi getter/setter per i vari attributi della classe. In particolare:

+ **DAODomande getDaoDomande()**

+ **void setDaoDomande(DAODomande daoDom)**

+ **GestionePunteggiID getGestionePunteggiID()**

+ **void setGestionePunteggiID(GestionePunteggiID gestionePunteggiID)**

+ **GestioneLog getGestioneLog()**

+ **void setGestioneLog(GestioneLog gestioneLog)**

+ **DAOPunteggi getDaoPunteggi()**

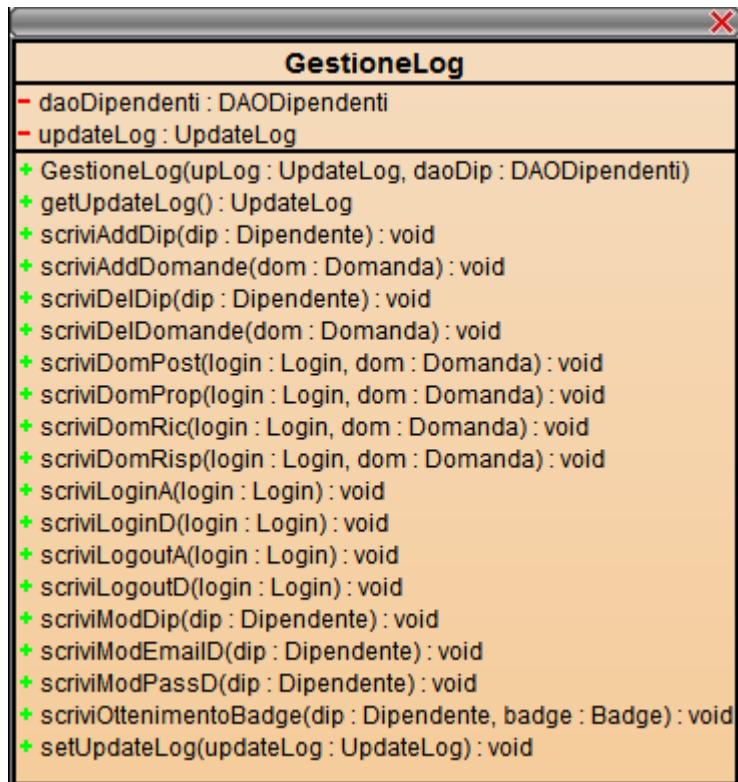
+ **void setDaoPunteggi(DAOPunteggi daoPunteggi)**

+ **DAODipendenti getDaoDipendenti()**

+ **void setDaoDipendenti(DAODipendenti daoDipendenti)**

³Ad esempio se si è raggiunto un certo numero di risposte corrette

3.3.8 com.safetyGame.back.controller.GestioneLog



Funzione

Questa classe si occuperà di creare le stringhe che poi andranno inserite nelle tabelle dei log all'interno del database aziendale.

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.controller.GestioneBadgeD
- back.controller.GestioneDati
- back.controller.GestioneDipendentiD
- back.controller.GestioneDomandeD

Inoltre utilizzerà le seguenti classi:

- back.access.UpdateLog
- back.access.DAO^{Dipendenti}



- back.condivisi.Login
- back.condivisi.Domanda
- back.condivisi.Dipendente

Attributi

- UpdateLog updateLog

Riferimento all'istanza della classe *UpdateLog*.

- DAO Dipendenti daoDipendenti

Riferimento all'istanza della classe *DAODipendenti*, opportunamente istanziato alla corretta tipologia di database utilizzato

Metodi

+ GestioneLog(UpdateLog upLog, DAO Dipendenti daoDip)

Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

+ void scriviLoginD(Login login)

Metodo che si occupa di inviare alla classe *UpdateLog* la stringa da inserire nel file di log dopo un login effettuato correttamente.

Dovrà chiamare l'*Access-Tier* affinchè scriva all'interno della tabella *LogLoginD* che un dipendente ha effettuato login. Dovrà inoltre specificare la data e l'ora in cui l'operazione è avvenuta.

+ void scriviLoginA(Login login)

Metodo che si occupa di inviare alla classe *UpdateLog* la stringa da inserire nel file di log dopo un login effettuato correttamente.

Dovrà chiamare l'*Access-Tier* affinchè scriva all'interno della tabella *LogLoginA* che un amministratore ha effettuato login. Dovrà inoltre specificare la data e l'ora in cui l'operazione è avvenuta.

+ void scriviLogoutD(Login login)

Metodo che si occupa di inviare alla classe *UpdateLog* la stringa da inserire nel file di log dopo un logout effettuato correttamente

Dovrà chiamare l'*Access-Tier* affinchè scriva all'interno della tabella *LogLogoutD* che un dipendente ha effettuato logout. Dovrà inoltre specificare la data e l'ora in cui l'operazione è avvenuta.

+ void scriviLogoutA(Login login)

Metodo che si occupa di inviare alla classe *UpdateLog* la stringa da inserire nel file di log dopo un logout effettuato correttamente

Dovrà chiamare l'*Access-Tier* affinchè scriva all'interno della tabella *LogLogoutA* che un amministratore ha effettuato logout. Dovrà inoltre specificare la



data e l'ora in cui l'operazione è avvenuta.

+ void scriviDomRic(Login login, Domanda dom)

Metodo che si occupa di inviare alla classe UpdateLog la stringa da inserire nel file di log dopo che una domanda viene ricevuta da un Dipendente.

Dovrà chiamare l'*Access-Tier* affinchè scriva all'interno della tabella *LogRicevuta* che un dipendente ha ricevuto una nuova domanda. Dovrà inoltre specificare la data e l'ora in cui l'operazione è avvenuta e l'id della domanda ricevuta.

+ void scriviDomProp(Login login, Domanda dom)

Metodo che si occupa di inviare alla classe UpdateLog la stringa da inserire nel file di log dopo che una domanda viene proposta ad un Dipendente

Dovrà chiamare l'*Access-Tier* affinchè scriva all'interno della tabella *LogProposta* che ad un dipendente è stata proposta una nuova domanda. Dovrà inoltre specificare la data e l'ora in cui l'operazione è avvenuta e l'id della domanda proposta.

+ void scriviDomPost(Login login, Domanda dom)

Metodo che si occupa di inviare alla classe UpdateLog la stringa da inserire nel file di log dopo che una domanda è stata posticipata da un Dipendente

Dovrà chiamare l'*Access-Tier* affinchè scriva all'interno della tabella *LogPosticipa* che un dipendente ha posticipato una domanda. Dovrà inoltre specificare la data e l'ora in cui l'operazione è avvenuta e l'id della domanda posticipata.

+ void scriviDomRisp(Login login, Domanda dom)

Metodo che si occupa di inviare alla classe UpdateLog la stringa da inserire nel file di log dopo che un Dipendente risponde ad una Domanda

Dovrà chiamare l'*Access-Tier* affinchè scriva all'interno della tabella *LogRisposta* che un dipendente ha risposto una domanda. Dovrà inoltre specificare la data e l'ora in cui l'operazione è avvenuta e l'id della domanda a cui si è data risposta.

+ void scriviModPassD(Dipendente dip)

Metodo che si occupa di inviare alla classe UpdateLog la stringa da inserire nel file di log dopo che un dipendente modifica la propria password

Dovrà chiamare l'*Access-Tier* affinchè scriva all'interno della tabella *LogModificaDip* che un dipendente ha modificato la propria password. Dovrà inoltre specificare la data e l'ora in cui l'operazione è avvenuta e la tipologia di modifica.

+ void scriviModEmailD(Dipendente dip)

Metodo che si occupa di inviare alla classe UpdateLog la stringa da inserire nel file di log dopo che un dipendente modifica la propria email

Dovrà chiamare l'*Access-Tier* affinchè scriva all'interno della tabella *LogModificaDip* che un dipendente ha modificato il proprio indirizzo email. Dovrà inoltre specificare la data e l'ora in cui l'operazione è avvenuta e la tipologia di modifica.

+ void scriviOttenimentoBadge(Dipendente dip, Badge badge)

Metodo che si occupa di inviare alla classe UpdateLog la stringa da inserire nel file di log dopo che un dipendente ottiene un badge

Dovrà chiamare l'*Access-Tier* affinchè scriva all'interno della tabella *OttenimentoBadge* che un dipendente ha ottenuto un nuovo badge. Dovrà inoltre specificare la data e l'ora in cui l'operazione è avvenuta e l'id del badge.

+ void scriviAddDip(Dipendente dip)

Metodo che si occupa di inviare alla classe UpdateLog la stringa da inserire nel file di log dopo che l'AA aggiunge un dipendente

Dovrà chiamare l'*Access-Tier* affinchè scriva all'interno della tabella *Mod-Dipendente* che un nuovo dipendente è stato aggiunto. Dovrà inoltre specificare la data e l'ora in cui l'operazione è avvenuta e la tipologia di operazione.

+ void scriviDelDip(Dipendente dip)

Metodo che si occupa di inviare alla classe UpdateLog la stringa da inserire nel file di log dopo che l'AA rimuove un dipendente

Dovrà chiamare l'*Access-Tier* affinchè scriva all'interno della tabella *Mod-Dipendente* che un dipendente è stato rimosso. Dovrà inoltre specificare la data e l'ora in cui l'operazione è avvenuta e la tipologia di operazione.

+ void scriviModDip(Dipendente dip)

Metodo che si occupa di inviare alla classe UpdateLog la stringa da inserire nel file di log dopo che l'AA modifica un dipendente

Dovrà chiamare l'*Access-Tier* affinchè scriva all'interno della tabella *Mod-Dipendente* che le informazioni di un dipendente sono state modificate. Dovrà inoltre specificare la data e l'ora in cui l'operazione è avvenuta e la tipologia di operazione.

+ void scriviAddDomande(Domanda [] dom)

Metodo che si occupa di inviare alla classe UpdateLog la stringa da inserire nel file di log dopo che l'AS aggiunge una o più domande

Dovrà chiamare l'*Access-Tier* affinchè scriva all'interno della tabella *AddR-
emDomanda* che una nuova domanda è stata inserita nella lista delle domande sottoponibili ai dipendenti. Dovrà inoltre specificare la data e l'ora in cui l'ope-
razione è avvenuta e la tipologia di operazione.

+ void scriviDelDomande(Domanda [] dom)

Metodo che si occupa di inviare alla classe UpdateLog la stringa da inserire nel file di log dopo che l'AS rimuove una o più domande

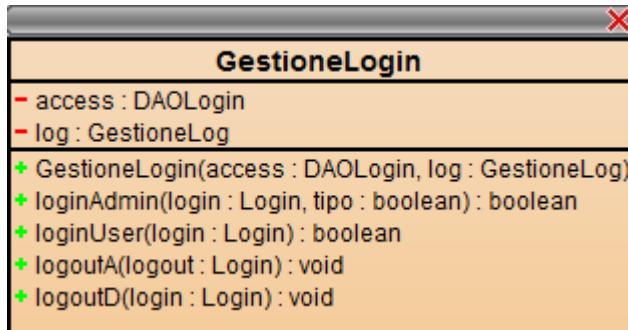
Dovrà chiamare l'*Access-Tier* affinchè scriva all'interno della tabella *AddR-
emDomanda* che una domanda è stata rimossa dalla lista delle domande sottopo-
nibili ai dipendenti. Dovrà inoltre specificare la data e l'ora in cui l'operazione è avvenuta e la tipologia di operazione.



Vengono quindi definiti i metodi getter/setter per i vari attributi della classe. In particolare:

- + **UpdateLog getUpdateLog()**
- + **void setUpdateLog(UpdateLog updateLog)**

3.3.9 com.safetyGame.back.controller.GestioneLogin



Funzione

Questa classe si occuperà di gestire tutte le richieste di login e logout, sia da parte di Dipendenti che di Amministratori (Azienda o Sicurezza che sia)

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.controller.GestioneBadgeD
- back.controller.GestioneDati

Inoltre utilizza le seguenti classi:

- back.access.DAOLogin
- back.controller.GestioneLog
- back.condivisi.Login

Attributi

- DAOLogin access

Riferimento all'istanza della classe *DAOLogin*, opportunamente istanziato alla corretta tipologia di database utilizzato

- GestioneLog log

Riferimento all'istanza della classe *GestioneLog*

Metodi

+ GestioneLogin(DAOLogin access, GestioneLog log)

Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

+ boolean loginAdmin(Login login, boolean tipo)

Metodo per controllare la correttezza dei dati di un tentativo di autenticazione nel sistema come Amministratore

+ boolean loginUser(Login login)

Metodo per controllare la correttezza dei dati di un tentativo di autenticazione nel sistema come Dipendente

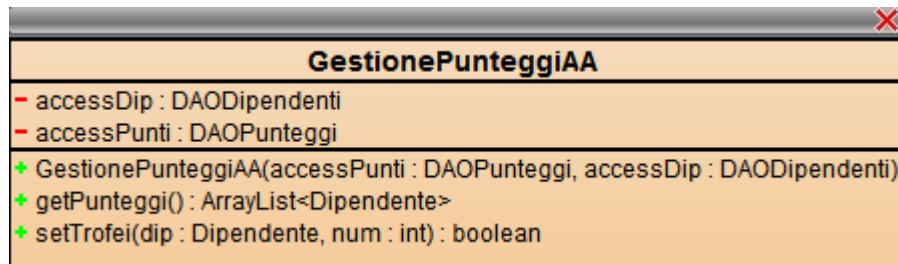
+ void logoutD(Login login)

Metodo per segnalare al sistema il logout di un dipendente

+ void logoutA(Login login)

Metodo per segnalare al sistema il logout di un amministratore

3.3.10 com.safetyGame.back.controller.GestionePunteggiAA



Funzione

Questa classe si occuperà di gestire tutti i dati riguardanti i punteggi, in particolare fornendo metodi per recuperare i punteggi di tutti i Dipendenti iscritti al sistema o per poter modificare il numero di trofei assegnati ad un utente.

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.controller.GestioneDati

Inoltre utilizza le seguenti classi:

- back.access.DAO_Punteggi
- back.access.DAODipendenti
- back.condivisi.Dipendente
- back.condivisi.Punteggio

Attributi

- DAO_Punteggi accessPunti

Riferimento all'istanza della classe *DAOPunteggi*, opportunamente istanziato alla corretta tipologia di database utilizzato

- DAODipendenti accessDip

Riferimento all'istanza della classe *DAODipendenti*, opportunamente istanziato alla corretta tipologia di database utilizzato

Metodi

- + GestionePunteggiAA(DAO_Punteggi accessPunti, DAODipendenti accessDip)

Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

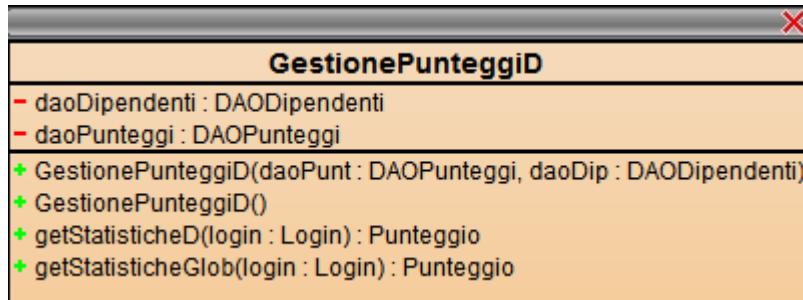
+ **ArrayList<Dipendente> getPunteggi()**

Metodo per ottenere i punteggi medi dell'azienda e i punteggi di tutti i dipendenti. Dovrà costruire un array di oggetti *Dipendente* con al loro interno i dati di ogni Dipendente (per poterlo identificare) e al loro interno un oggetto *Punteggio* con sia i punteggi del Dipendente che dell'intera azienda.

+ **boolean setTrofei(Dipendente dip, int num)**

Metodo per modificare il numero di trofei di un dipendente

3.3.11 com.safetyGame.back.controller.GestionePunteggiD



Funzione

Questa classe si occuperà di gestire tutti i dati riguardanti i Badge, sia per quanto riguarda il loro recupero dal database, che una loro eventuale modifica. Fornisce l'interfaccia minima necessaria a tutte le classi derivate che dovranno offrire questo tipo di servizio

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.controller.GestioneDati
- back.controller.GestioneDomandeD

Inoltre utilizza le seguenti classi:

- back.access.DAOBadge
- back.access.DAODipendenti
- back.access.DAO_Punteggi
- back.condivisi.Login
- back.condivisi.Punteggio

Attributi

- DAOPunteggi daoPunteggi

Riferimento all'istanza della classe *DAOPunteggi*, opportunamente istanziato alla corretta tipologia di database utilizzato

- DAODipendenti daoDipendenti

Riferimento all'istanza della classe *DAODipendenti*, opportunamente istanziato alla corretta tipologia di database utilizzato

Metodi

- + **GestionePunteggiD(DAOPunteggi daoPunt, DAO Dipendenti daoDip)**

Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

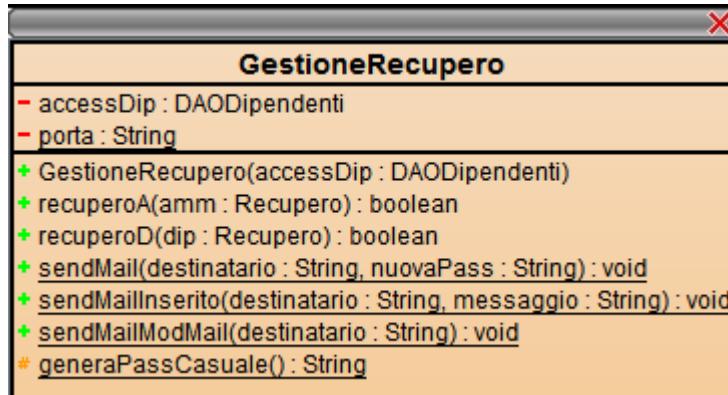
- + **Punteggio getStatisticheD(Login login)**

Metodo che consente di recuperare le statistiche di un Dipendente identificato dal *Login*.

- + **Punteggio getStatisticheGlob(Login login)**

Metodo che consente di recuperare le statistiche dell'intera azienda e del Dipendente identificato dal *Login*.

3.3.12 com.safetyGame.back.controller.GestioneRecupero



Funzione

Questa classe si occuperà di gestire le richieste di recupero delle password.

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.controller.GestioneDati

Inoltre verranno utilizzate le seguenti classi:

- back.access.DAODipendenti
- back.controller.GestioneLog

Attributi

- DAODipendenti accessDip

Riferimento all'istanza della classe *DAODipendenti*, opportunamente istanziato alla corretta tipologia di database utilizzato

- static String porta

Stringa che dovrà contenere la porta di comunicazione del back-end con il server SMTP tramite il quale inviare le email

Metodi

+ GestioneRecupero(DAODipendenti accessDip)

Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

+ boolean recuperоД(Recupero dip)



Metodo che controlla la validità dei dati inseriti dall'utente e, nel caso questi siano validi, genera una nuova password, andando ad inserirla nel database.

Dovrà prima di tutto controllare che i dati inseriti siano corretti, chiamando l'*Access-Tier* affinché verifichi che *codice fiscale* e indirizzo *mail* appartengano effettivamente all'account identificato dall'*username*. Se i controlli non dovessero segnalare incongruenze, allora il metodo dovrà generare una nuova password e modificare quella che è attualmente presente nel database.

+ **boolean recuperoA(Recupero amm)**

Metodo che consente ad un amministratore di resettare la propria password.

Dovrà generare la password casuale, quindi dovrà chiamare l'*Access-Tier* affinché modifichi la riga riferita ai dati di accesso dell'Amministratore.

+ **static void sendMail(String destinatario, String messaggio)**

Metodo che si occupa di inviare una mail all'utente che ha richiesto il ripristino della password.

Dovrà creare una connessione con il server SMTP indicato nella configurazione interna al metodo, quindi tentare di inviare il messaggio. Nel caso ci sia un errore, dovrà lanciare un'eccezione di tipo **RuntimeException**.

+ **static void sendMailInserito(String destinatario, String messaggio)**

Metodo che si occupa di inviare una mail all'utente che è stato inserito all'interno del sistema.

Dovrà creare una connessione con il server SMTP indicato nella configurazione interna al metodo, quindi tentare di inviare il messaggio. Nel caso ci sia un errore, dovrà lanciare un'eccezione di tipo **RuntimeException**.

+ **static void sendMailModMail(String destinatario)**

Metodo che si occupa di inviare una mail all'utente che ha richiesto la modifica del proprio indirizzo email.

Dovrà creare una connessione con il server SMTP indicato nella configurazione interna al metodo, quindi tentare di inviare il messaggio. Nel caso ci sia un errore, dovrà lanciare un'eccezione di tipo **RuntimeException**.

static String generaPassCasuale()

Metodo che genera una nuova password di 16 caratteri di lunghezza. Questa password dovrà contenere almeno un carattere maiuscolo, uno minuscolo, un numero e un carattere non alfa-numerico contenuto fra questi:

- @
- #
- *
- +
- ?

- ^
- %
- &
- /
- \$
- !
- +
- -

3.4 Package com.safetyGame.back.access

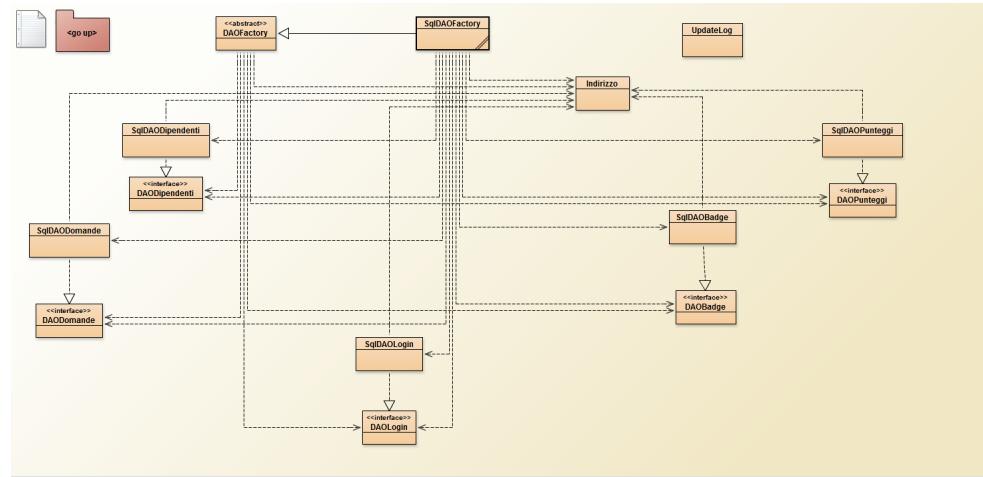


Figura 5: Package *com.safetyGame.back.access*

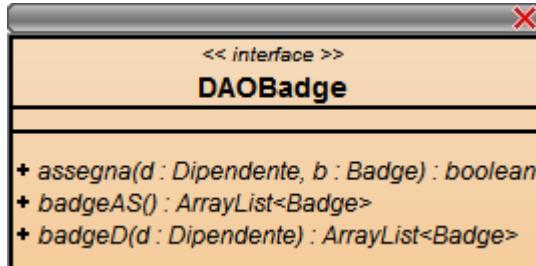
Tipo, obiettivo e funzione del componente: il package contiene tutte le classi che si occupano di eseguire le operazioni che il *Controller-Tier* chiede che vengano eseguite sui database aziendali e centrali.

Relazioni d'uso di altre componenti: utilizza il package `com.safetyGame.back.condivisi` per trasmettere le informazioni trovate all'interno dei database al *Controller-Tier*

Interfacce con e relazioni d'uso da altre componenti: viene utilizzato dal package **com.safetyGame.back.controller** per ricevere tutti i dati dalla database di cui ha bisogno per le sue elaborazioni e per essere mostrati agli utenti tramite i *Front-end*

Attività svolte e dati trattati: fa da connettore fra il *Back-end* e i database, divenendo parte del componente *Model* del design pattern *MVP*

3.4.1 Interfaccia com.safetyGame.back.access.DAOBadge



Funzione

Questa classe si occuperà di gestire tutti i dati riguardanti i Badge, sia per quanto riguarda il loro recupero dal database, che una loro eventuale modifica. Fornisce l’interfaccia minima necessaria a tutte le classi derivate che dovranno offrire questo tipo di servizio

Relazioni d’uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.access.SqlDAOBadge
- back.access.DAOFactory
- back.access.SqlDAOFactory
- back.controller.GestioneBadgeD
- back.controller.GestioneBadgeAS
- back.controller.GestioneDipendentiD

Inoltre utilizzerà le seguenti classi:

- back.condivisi.Dipendente
- back.condivisi.Badge

Metodi

- + **ArrayList<Badge> badgeD(Dipendente dip)**
Metodo che prende le badge ottenute da un Dipendente dal database
- + **ArrayList<Badge> badgeAS()**
Metodo che preleva tutti i badge esistenti nel database
- + **boolean assegna(Dipendente dip, Badge badge)**
Metodo che assegna una Badge ad un Dipendente

3.4.2 com.safetyGame.back.access.SqlDAOBadge



Funzione

Questa classe si occuperà di gestire tutti i dati riguardanti i Badge, sia per quanto riguarda il loro recupero dal database, che una loro eventuale modifica.

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.access.SqlDAOFactory

Inoltre utilizzerà le seguenti classi:

- back.access.Indirizzo
- back.condivisi.Dipendente
- back.condivisi.Badge

Attributi

- Indirizzo serverAzienda

Riferimento all'istanza della classe *Indirizzo* contenente i riferimenti al server aziendale

Metodi

+ SqIDAOBadge(Indirizzo azienda)

Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

+ ArrayList<Badge> badgeD(Dipendente dip)

Metodo che prende i badge ottenute da un Dipendente dal database.

Nel caso in cui la query dovesse ritornare un oggetto vuoto (ovvero la cui dimensione interna è uguale a zero), la funzione dovrà ritornare **null**.

+ ArrayList<Badge> badgeAS()

Metodo che preleva tutti i badge esistenti nel database.

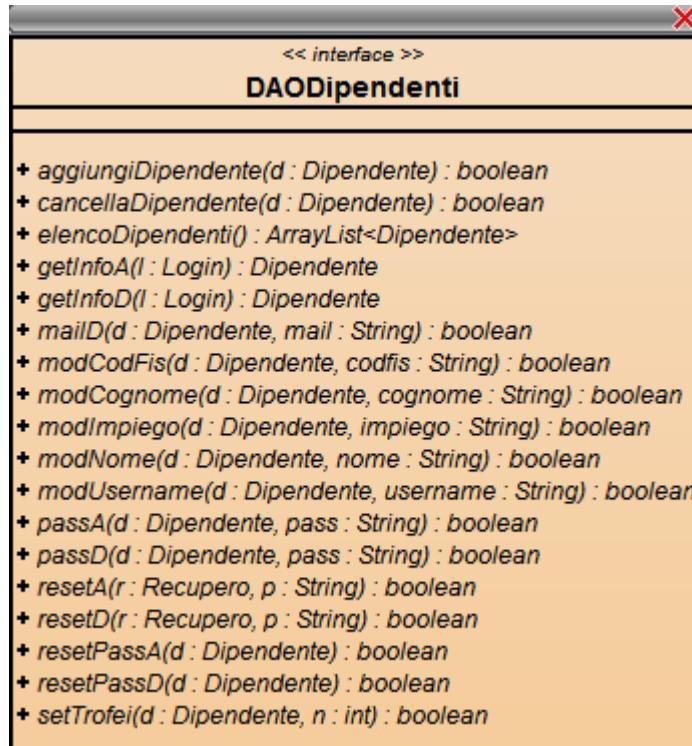
Nel caso in cui la query dovesse ritornare un oggetto vuoto (ovvero la cui dimensione interna è uguale a zero), la funzione dovrà ritornare **null**.

+ boolean assegna(Dipendente dip, Badge badge)

Metodo che assegna un Badge ad un Dipendente.

Dovrà passare dei parametri alla classe deputata all'interazione con il database. Tali parametri dovranno includere l'id del dipendente e del badge assegnato

3.4.3 Interfaccia com.safetyGame.back.access.DAODipendenti



Funzione

Questa classe si occuperà di gestire tutti i dati riguardanti i Dipendenti. Fornisce l'interfaccia minima necessaria a tutte le classi derivate che dovranno offrire questo tipo di servizio

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.access.SqlDAODipendenti
- back.access.DAOFactory
- back.access.SqlDAOFactory
- back.controller.GestioneBadgeD
- back.controller.GestioneDipendentiAA
- back.controller.GestioneDipendentiD
- back.controller.GestioneDomandeD

- back.controller.GestionePunteggiaA
- back.controller.GestionePunteggiD
- back.controller.GestioneRecupero

Inoltre utilizzerà le seguenti classi:

- back.condivisi.Login
- back.condivisi.Dipendente
- back.condivisi.Recupero

Metodi

+ Dipendente getInfoD(Login login)

Metodo che prende le informazioni di un Dipendente dal database e le inserisce all'interno di un oggetto Dipendente.

+ Dipendente getInfoA(Login login)

Metodo che prende le informazioni di un Amministratore dal database e le inserisce all'interno di un oggetto Dipendente.

+ boolean resetPassD(Dipendente dip)

Metodo che resetta il campo password modificata di un Dipendente nel database.

+ boolean resetPassA(Dipendente dip)

Metodo che resetta il campo password modificata di un Amministratore nel database.

+ boolean passD(Dipendente dip, String pass)

Metodo che modifica la password di un Dipendente all'interno del Database secondo la stringa *pass*.

+ boolean passA(Dipendente dip, String pass)

Metodo che setta il campo password (e il campo data pass) di un Amministratore

+ boolean mailD(Dipendente dip, String mail)

Metodo che setta il campo mail di un Dipendente

+ ArrayList<Dipendente> elencoDipendenti()

Metodo che ritorna l'elenco dei Dipendenti dell'Azienda

+ boolean aggiungiDipendente(Dipendente dip)

Metodo che aggiunge un Dipendente nel database

+ **boolean cancellaDipendente(Dipendente dip)**

Metodo che cancella un Dipendente dal database

+ **boolean modNome(Dipendente dip, String nome)**

Metodo che modifica il nome di un Dipendente nel database

+ **boolean modCognome(Dipendente dip, String cognome)**

Metodo che modifica il nome di un Dipendente nel database

+ **boolean modCodFis(Dipendente dip, String codfis)**

Metodo che modifica il codice fiscale di un Dipendente nel database

+ **boolean modUsername(Dipendente dip, String username)**

Metodo che modifica lo username di un Dipendente nel database

+ **boolean modImpiego(Dipendente dip, String impiego)**

Metodo che modifica l'impiego di un Dipendente nel database

+ **boolean setTrofei(Dipendente dip, int numTrofei)**

Metodo che modifica il numero di trofei di un Dipendente nel database

+ **boolean resetD(Recupero rec, String pass)**

Metodo che resetta la password (casuale) di un Dipendente

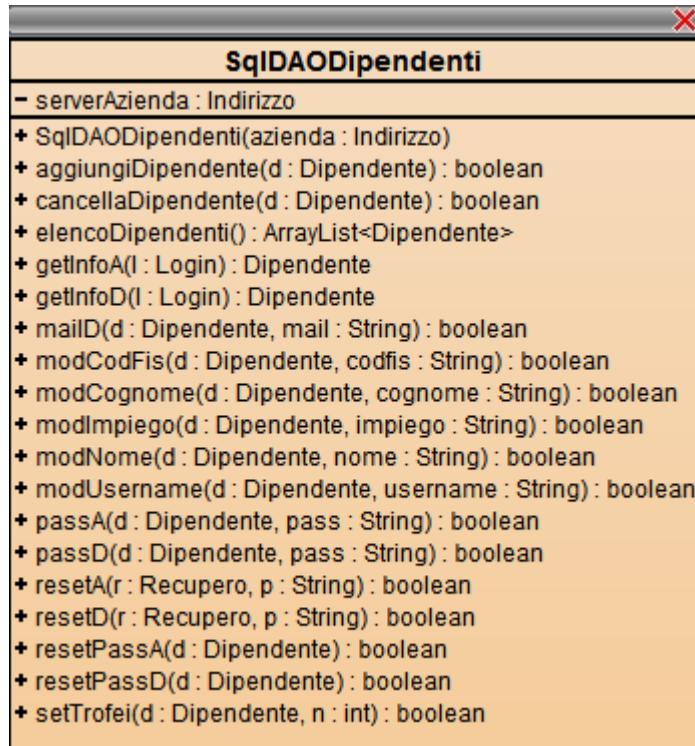
+ **boolean resetA(Recupero rec, String pass)**

Metodo che resetta la password (casuale) di un Amministratore

+ **ArrayList<String> getElencoRuoli()**

Metodo che consente di recuperare l'elenco dei ruoli aziendali

3.4.4 com.safetyGame.back.access.SqlDAODipendenti



Funzione

Questa classe si occuperà di gestire tutti i dati riguardanti i Dipendenti.

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.access.SqlDAOFactory

Inoltre utilizzerà le seguenti classi:

- back.access.Indirizzo
- back.condivisi.Login
- back.condivisi.Dipendente
- back.condivisi.Recupero

Attributi

- Indirizzo serverAzienda

Riferimento all'istanza della classe *Indirizzo* contenente i riferimenti al server aziendale

Metodi

+ SqlDAODipendenti(Indirizzo azienda)

Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

+ Dipendente getInfoD(Login login)

Metodo che prende le informazioni di un Dipendente dal database e le inserisce all'interno di un oggetto Dipendente.

+ Dipendente getInfoA(Login login)

Metodo che prende le informazioni di un Amministratore dal database e le inserisce all'interno di un oggetto Dipendente.

+ boolean resetPassD(Dipendente dip)

Metodo che resetta il campo password modificata di un Dipendente nel database.

+ boolean resetPassA(Dipendente dip)

Metodo che resetta il campo password modificata di un Amministratore nel database.

+ boolean passD(Dipendente dip, String pass)

Metodo che modifica la password di un Dipendente all'interno del Database secondo la stringa *pass*.

+ boolean passA(Dipendente dip, String pass)

Metodo che setta il campo password (e il campo data pass) di un Amministratore

+ boolean mailD(Dipendente dip, String mail)

Metodo che setta il campo mail di un Dipendente

+ ArrayList<Dipendente> elencoDipendenti()

Metodo che ritorna l'elenco dei Dipendenti dell'Azienda

+ boolean aggiungiDipendente(Dipendente dip)

Metodo che aggiunge un Dipendente nel database

+ boolean cancellaDipendente(Dipendente dip)



Metodo che cancella un Dipendente nel database

+ **boolean modNome(Dipendente dip, String nome)**

Metodo che modifica il nome di un Dipendente nel database

+ **boolean modCognome(Dipendente dip, String cognome)**

Metodo che modifica il nome di un Dipendente nel database

+ **boolean modCodFis(Dipendente dip, String codfis)**

Metodo che modifica il codice fiscale di un Dipendente nel database

+ **boolean modUsername(Dipendente dip, String username)**

Metodo che modifica lo username di un Dipendente nel database

+ **boolean modImpiego(Dipendente dip, String impiego)**

Metodo che modifica l'impiego di un Dipendente nel database

+ **boolean setTrofei(Dipendente dip, int numTrofei)**

Metodo che modifica il numero di trofei di un Dipendente nel database

+ **boolean resetD(Recupero rec, String pass)**

Metodo che resetta la password (casuale) di un Dipendente

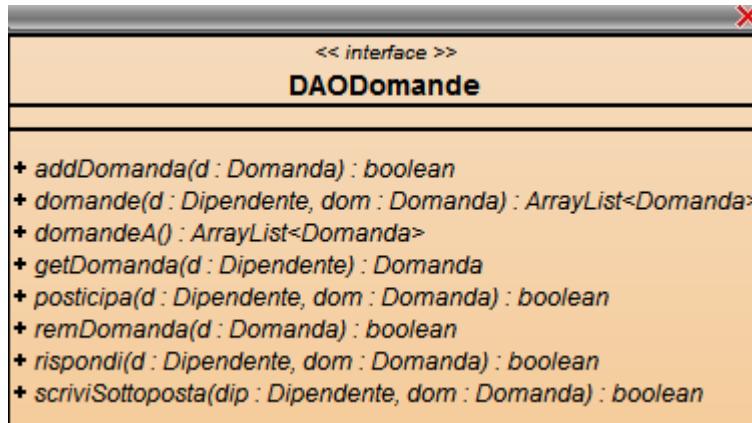
+ **boolean resetA(Recupero rec, String pass)**

Metodo che resetta la password (casuale) di un Amministratore

+ **ArrayList<String> getElencoRuoli()**

Metodo che restituisce la lista dei ruoli aziendali

3.4.5 Interfaccia com.safetyGame.back.access.DAODomande



Funzione

Questa classe si occuperà di gestire tutti i dati riguardanti le domande. Fornisce l'interfaccia minima necessaria a tutte le classi derivate che dovranno offrire questo tipo di servizio

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.access.DAOFactory
- back.access.SqlDAODomande
- back.access.SqlDAOFactory
- back.controller.GestioneBadgeD
- back.controller.GestioneDomandeAS
- back.controller.GestioneDomandeD

Inoltre utilizzerà le seguenti classi:

- back.condivisi.Dipendente
- back.condivisi.Domanda

Metodi

+ Domanda getDomanda(Dipendente dip)

Metodo che, dato un Dipendente, fornisce un oggetto Domanda che conterrà una domanda a cui il Dipendente o non ha risposto (possibilmente avendo



chiesto di posticiparla), o ha risposto erroneamente o non ha mai visto.

+ **boolean posticipa(Dipendente dip, Domanda dom)**

Metodo che dovrà posticipare la Domanda sottoposta al Dipendente, in modo tale che questa possa essere riproposta più tardi allo stesso.

+ **boolean rispondi(Dipendente dip, Domanda dom)**

Metodo che permette di verificare la correttezza di una risposta data da un Dipendente ad una Domanda.

+ **ArrayList<Domanda> domandeA()**

Metodo che ritorna un array contenente tutte le Domande che l'Amministratore Sicurezza ha scelto per l'azienda.

+ **ArrayList<Domanda> domande(Dipendente dip, Domanda dom)**

Metodo che ritorna un array contenente tutte le Domande a cui un dipendente ha dato risposta.

+ **boolean addDomanda(Domanda dom)**

Metodo che aggiunge una Domanda al database aziendale, in modo tale che questa possa essere sottoposta ai Dipendenti

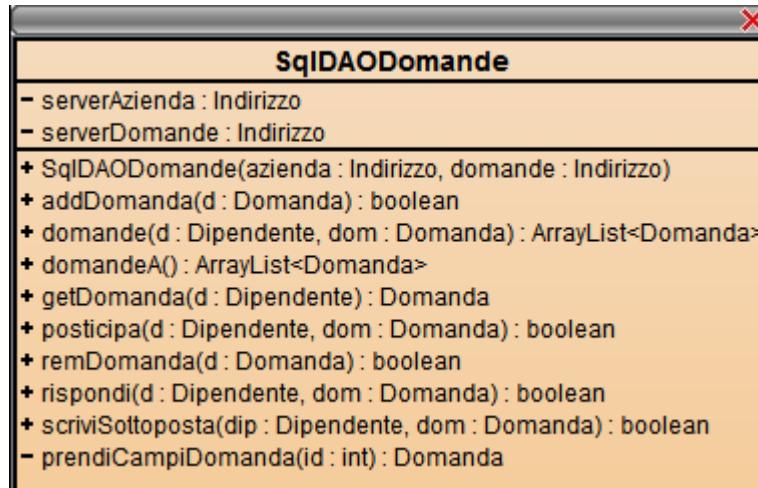
+ **boolean remDomanda(Domanda dom)**

Metodo che rimuove una Domanda dal database aziendale, in modo tale che questa non possa più essere sottoposta ai Dipendenti.

+ **boolean scriviSottoposta(Dipendente dip, Domanda dom)**

Metodo che scrive sul database che una Domanda è stata sottoposta ad un Dipendente.

3.4.6 com.safetyGame.back.access.SqlDAODomande



Funzione

Questa classe si occuperà di gestire tutti i dati riguardanti le domande.

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.access.SqlDAOFactory

Inoltre le seguenti classi sono utilizzate:

- back.access.Indirizzo
- back.condivisi.Dipendente
- back.condivisi.Domanda

Attributi

- Indirizzo serverDomande

Riferimento all'istanza della classe *Indirizzo* contenente i riferimenti al server delle domande

- Indirizzo serverAzienda

Riferimento all'istanza della classe *Indirizzo* contenente i riferimenti al server aziendale

Metodi

- + **SqlDAODomande(Indirizzo azienda, Indirizzo domande)**
Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.
- **Domanda prendiCampiDomanda(int id)**
Metodo che recupera i campi che compongono un oggetto Domanda tramite il suo id.
Dovrà innanzitutto recuperare la domanda dal server centrale, quindi creare l'oggetto *Domanda* e inserire al suo interno i dati recuperati. Nel caso una delle query dovesse lanciare un'eccezione, la funzione dovrà ritornare un valore **null**.
- + **Domanda getDomanda(Dipendente dip)**
Metodo che, dato un Dipendente, fornisce un oggetto Domanda che conterrà una domanda a cui il Dipendente o non ha risposto (possibilmente avendo chiesto di posticiparla), o ha risposto erroneamente o non ha mai visto.
Tale metodo dovrà controllare in primis che non esistano domande a cui il Dipendente non ha ancora dato risposta. Se questa condizione dovesse essere verificata, la funzione dovrà prelevare una nuova domanda da sottoporre al Dipendente combinando le domande a cui ha risposto erroneamente con quelle a cui non ha mai dato risposta e che non gli sono state sottoposte. Quindi ritornerà l'oggetto passato dalla funzione *prendiCampiDomanda(int id)*
- + **boolean posticipa(Dipendente dip, Domanda dom)**
Metodo che dovrà posticipare la Domanda sottoposta al Dipendente, in modo tale che questa possa essere riproposta più tardi allo stesso.
- + **boolean rispondi(Dipendente dip, Domanda dom)**
Metodo che permette di verificare la correttezza di una risposta data da un Dipendente ad una Domanda.
Dovrà confrontare la risposta data con l'id della risposta corretta e, nel caso coincidessero, assegnare i punti previsti al Dipendente. Quindi dovrà segnare in database sia che è stata data risposta, che l'esito, provvedendo eventualmente ad aggiornare il punteggio del Dipendente.
- + **ArrayList<Domanda> domandeA()**
Metodo che ritorna un array contenente tutte le Domande che l'Amministratore Sicurezza ha scelto per l'azienda.
Questo metodo dovrà recuperare gli indici di tutte le domande dal database aziendale, quindi comporre gli oggetti Domanda recuperando le informazioni dal database centrale. Infine inserirle dentro un array, quindi restituirlo alla funzione chiamante.
- + **ArrayList<Domanda> domande(Dipendente dip, Domanda dom)**

Metodo che ritorna un array contenente tutte le Domande a cui un dipendente ha dato risposta.

Dovrà associare al Dipendente tutte le possibili domande e da queste estrarre quelle a cui ha dato risposta.

+ **boolean addDomanda(Domanda dom)**

Metodo che aggiunge una Domanda al database aziendale, in modo tale che questa possa essere sottoposta ai Dipendenti

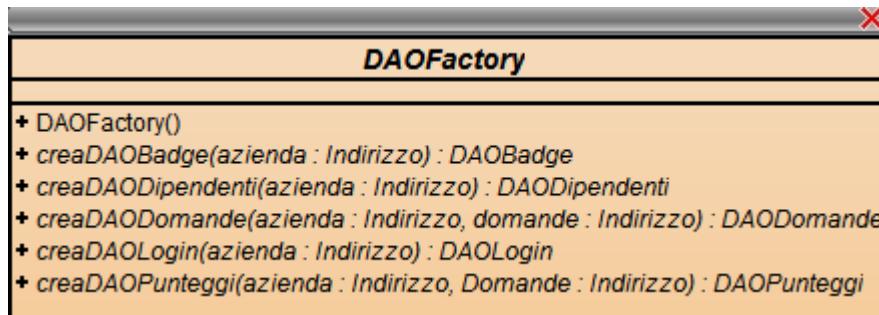
+ **boolean remDomanda(Domanda dom)**

Metodo che rimuove una Domanda dal database aziendale, in modo tale che questa non possa più essere sottoposta ai Dipendenti.

+ **boolean scriviSottoposta(Dipendente dip, Domanda dom)**

Metodo che scrive sul database che una Domanda è stata sottoposta ad un Dipendente.

3.4.7 Interfaccia com.safetyGame.back.access.DAOFactory



Funzione

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.access.SqlDAOFactory
- back.controller.GestioneDipendentiD
- back.controller.GestioneDomandeD

Inoltre le seguenti classi vengono utilizzate:

- back.access.DAOLogin
- back.access.DAODipendenti
- back.access.DAODomande
- back.access.DAOBadge
- back.access.DAOPunteggi
- back.access.Indirizzo

Metodi

+ DAOLogin creaDAOLogin(Indirizzo azienda)

Metodo che dovrà restituire un oggetto di sottotipo di DAOLogin; più precisamente dovrà esser il sottotipo corretto in relazione alla tipologia di database utilizzato.

+ DAODipendenti creaDAODipendenti(Indirizzo azienda)

Metodo che dovrà restituire un oggetto di sottotipo di DAODipendenti; più precisamente dovrà esser il sottotipo corretto in relazione alla tipologia di database utilizzato.

+ **DAODomande creaDAODomande (Indirizzo azienda, Indirizzo domande)**

Metodo che dovrà restituire un oggetto di sottotipo di DAODomande; più precisamente dovrà esser il sottotipo corretto in relazione alla tipologia di database utilizzato.

+ **DAOBadge creaDAOBadge(Indirizzo azienda)**

Metodo che dovrà restituire un oggetto di sottotipo di DAOBadge; più precisamente dovrà esser il sottotipo corretto in relazione alla tipologia di database utilizzato.

+ **DAOPunteggi creaDAOPunteggi(Indirizzo azienda, Indirizzo domande)**

Metodo che dovrà restituire un oggetto di sottotipo di DAOLogin; più precisamente dovrà esser il sottotipo corretto in relazione alla tipologia di database utilizzato.

3.4.8 com.safetyGame.back.access.SqlDAOFactory



Funzione

Questa classe istanzia i corretti oggetti afferenti alla gestione di un database Sql in base alle interfacce definite dal Design Pattern DAO⁴

Relazioni d'uso con altri moduli

Le seguenti classe verranno utilizzate:

- back.access.SqlDAOLogin
- back.access.SqlDAODipendenti
- back.access.SqlDAODomande
- back.access.SqlDAOBadge
- back.access.SqlDAOPunteggi
- back.access.Indirizzo

Attributi

Metodi

+ **SqIDAOFactory()**

Costruttore della classe.

+ **DAOLogin creaDAOLogin(Indirizzo azienda)**

Metodo che dovrà restituire un oggetto di tipo SqlDAOLogin, visto che la classe riguarda Sql.

+ **DAODipendenti creaDAODipendenti(Indirizzo azienda)**

Metodo che dovrà restituire un oggetto di tipo SqlDAODipendenti, visto che la classe riguarda Sql.

⁴Data Access Object

+ **DAODomande creaDAODomande(Indirizzo azienda, Indirizzo domande)**

Metodo che dovrà restituire un oggetto di tipo SqlDAODomande, visto che la classe riguarda Sql.

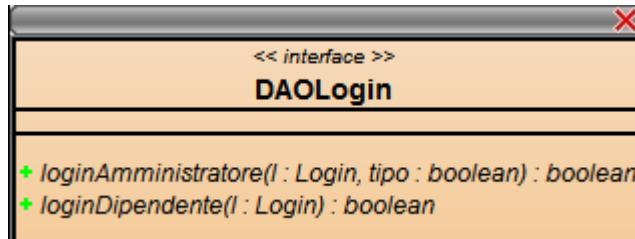
+ **DAOBadge creaDAOBadge(Indirizzo azienda)**

Metodo che dovrà restituire un oggetto di tipo SqlDAOBadge, visto che la classe riguarda Sql.

+ **DAOPunteggi creaDAOPunteggi(Indirizzo azienda, Indirizzo domande)**

Metodo che dovrà restituire un oggetto di tipo SqlDAOPunteggi, visto che la classe riguarda Sql.

3.4.9 Interfaccia com.safetyGame.back.access.DAOLogin



Funzione

Questa classe si occuperà di gestire tutti i dati riguardanti i tentativi di autenticazione all'interno del sistema. Fornisce l'interfaccia minima necessaria a tutte le classi derivate che dovranno offrire questo tipo di servizio

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.access.DAOFactory
- back.access.SqlDAOFactory
- back.access.SqlDAOLogin
- back.controller.GestioneLogin

Metodi

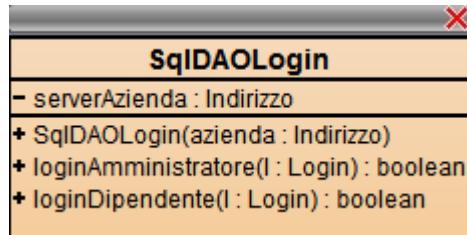
+ boolean loginAmministratore(Login login, boolean tipo)

Metodo che controllerà la correttezza dei dati inseriti da un utente per connettersi al sistema come amministratore.

+ boolean loginDipendente(Login login)

Metodo che controllerà la correttezza dei dati inseriti da un utente per connettersi al sistema come dipendente.

3.4.10 com.safetyGame.back.access.SqlDAOLogin



Funzione

Questa classe si occuperà di gestire tutti i dati riguardanti i tentativi di autenticazione all'interno del sistema.

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.access.SqlDAOFactory

Inoltre vengono utilizzate le seguenti classi:

- back.access.Indirizzo
- back.condivisi.Login

Attributi

- Indirizzo serverAzienda

Riferimento all'istanza della classe *Indirizzo* contenente i riferimenti al server aziendale

Metodi

+ SqIDAOLogin(Indirizzo azienda)

Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

+ boolean loginAmministratore(Login login, boolean tipo)

Metodo che controllerà la correttezza dei dati inseriti da un utente per connettersi al sistema come amministratore. Dovrà eseguire una query sul database aziendale cercando i dati contenuti nell'oggetto *Login* all'interno della tabella contenente le credenziali d'accesso degli amministratori e, nel caso non vengano lanciate eccezioni, dovrà ritornare **true**. Altrimenti **false**.

+ boolean loginDipendente(Login login)



Metodo che controllerà la correttezza dei dati inseriti da un utente per connettersi al sistema come dipendente. Dovrà eseguire una query sul database aziendale cercando i dati contenuti nell'oggetto Login all'interno della tabella contenente le credenziali d'accesso dei Dipendenti e, nel caso non vengano lanciate eccezioni, dovrà ritornare **true**. Altrimenti **false**.

3.4.11 Interfaccia com.safetyGame.back.access.DAO_Punteggi



Funzione

Questa classe si occuperà di recuperare i dati riguardanti i Punteggi dei Dipendenti e dell’Azienda. Fornisce l’interfaccia minima necessaria a tutte le classi derivate che dovranno offrire questo tipo di servizio

Relazioni d’uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.access.SqlDAO_Punteggi
- back.access.DAOFactory
- back.access.SqlDAOFactory
- back.controller.GestioneDomandeD
- back.controller.GestionePunteggiAA
- back.controller.GestionePunteggiD

Metodi

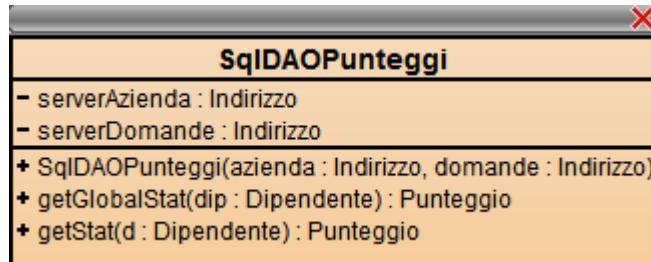
+ Punteggio getStat(Dipendente dip)

Metodo che dovrà ritornare un oggetto di tipo Punteggio con informazioni relative al solo Dipendente in oggetto.

+ Punteggio getGlobalStat(Dipendente dip)

Metodo che dovrà ritornare un oggetto di tipo Punteggio contenente le informazioni sia del Dipendente in oggetto, che di quei Dipendenti immediatamente prossimi a lui nella classifica dei punteggi, oltre che alle statistiche dell’azienda (come punteggio medio o numero di risposte corrette totali).

3.4.12 com.safetyGame.back.access.SqlDAOPunteggi



Funzione

Implementa back.access.DAO⁺Punteggi.

Questa classe si occuperà di recuperare i dati riguardanti i Punteggi dei Dipendenti e dell’Azienda da un database Sql.

Relazioni d’uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.access.SqlDAOFactory

Inoltre utilizzerà le seguenti classi:

- back.access.Indirizzo
- back.condivisi.Dipendente
- back.condivisi.Punteggio

Attributi

- Indirizzo serverAzienda

Riferimento all’istanza della classe *Indirizzo* contenente i riferimenti al server aziendale

- Indirizzo serverDomande

Riferimento all’istanza della classe *Indirizzo* contenente i riferimenti al server delle domande

Metodi

+ SqIDAO⁺Punteggi(Indirizzo azienda, Indirizzo domande)

Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

+ Punteggio getStat(Dipendente dip)

Questo metodo dovrà ritornare un oggetto di tipo Punteggio con informazioni relative al solo Dipendente in oggetto. Per cui dovrà effettuare una query al database aziendale per recuperare il punteggio che il Dipendente ha ottenuto fino a quel momento e quindi ritornare l'oggetto Punteggio creato a partire da quanto risulta nel database.

+ **Punteggio getGlobalStat(Dipendente dip)**

Questo metodo dovrà ritornare un oggetto di tipo Punteggio contenente le informazioni sia del Dipendente in oggetto, che di quei Dipendenti immediatamente prossimi a lui nella classifica dei punteggi, oltre che alle statistiche dell'azienda (come punteggio medio o numero di risposte corrette totali). Dovrà interrogare ogni tabella che contiene questo tipo di informazioni e, una volta recuperate, incapsularle all'interno di un oggetto Punteggio, che dovrà restituire alla funzione chiamante.

3.4.13 com.safetyGame.back.access.Indirizzo

Indirizzo	
- conn : Connection	
- connettore : Statement	
+ Indirizzo(database : String, utente : String, password : String)	
+ cancellaRiga(tabella : String, controlli : String) : boolean	
+ finalize() : void	
+ inserisciRiga(tabella : String, colonne : String, valori : String[]) : boolean	
+ modificaRiga(tabella : String, colonnevalori : String, controlli : String) : boolean	
+ selezione(tabella : String, colonne : String, controlli : String, extra : String) : ResultSet	

Funzione

Questa classe contiene tutti i metodi per l'esecuzione di query all'interno dei database (azienda e centrale)

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.access.SqlDAOBadge
- back.access.SqlDAODipendenti
- back.access.SqlDAODomande
- back.access.SqlDAOFactory
- back.access.SqlDAOLogin
- back.access.SqlDAOPunteggi

Attributi

- Connection conn

Riferimento alla connessione al database

- Statement connettore

Riferimento alla risorsa impiegata per eseguire query sul database

Metodi

+ Indirizzo(String database, String utente, String password)

Costruttore che imposta il valore degli attributi secondo il valore dei parametri passati.

+ void finalize()

Distruttore della classe Indirizzo: prova a chiudere la connessione al database quando l'oggetto viene distrutto. Nel caso di lancio di eccezioni, dovrà stampare a video un messaggio d'errore

+ boolean inserisciRiga(String tabella, String colonne, String [] valori)

Crea, a partire dall'array *valori*, una stringa (che chiameremo *val*) che contiene tutti i valori contenuti. Dovrà avere la seguente struttura:

$$(\?, \?, \dots, \?)$$

ripetendo tanti punti interrogativi quanti sono gli elementi dentro all'array *valori*. Quindi crea la query da eseguire sul database utilizzando:

- **tabella:** nome della tabella su cui eseguire la query
- **colonne:** colonne dove andranno inseriti i dati. Al contenuto di questa variabile dovrà essere applicata una funzione per eliminare eventuali caratteri di spaziatura
- **val**

Quindi i punti interrogativi saranno sostituiti dai valori contenuti nell'array *valori*. Se la funzione di esecuzione della query non lancerà alcuna eccezione, questo metodo si concluderà ritornando **true**. Altrimenti dovrà ritornare **false**.

+ boolean modificaRiga(String tabella, String colonnevalori, String controlli)

Modifica una tupla di valori all'interno della tabella indicata, selezionandola secondo i parametri di controllo indicati. Nel caso l'esecuzione della query ritornasse un'eccezione, la funzione dovrà ritornare **false**, altrimenti **true**.

+ boolean cancellaRiga(String tabella, String controlli)

Elimina una tupla di valori all'interno della tabella indicata, selezionandola secondo i parametri di controllo indicati. Se la stringa *controlli* dovesse risultare vuota dopo aver applicato una funzione per eliminare eventuali caratteri di spaziatura, la funzione dovrà ritornare **false**, così come se l'esecuzione della query sul database dovesse sollevare un'eccezione. Altrimenti la funzione dovrà ritornare **true**.

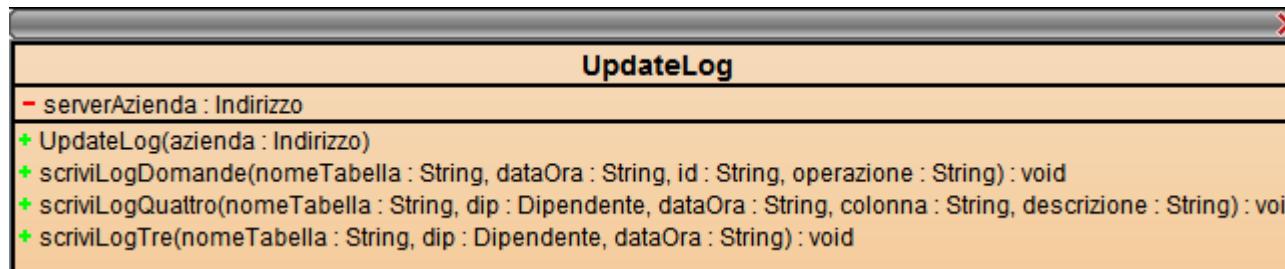
+ ResultSet selezione(String tabella, String colonne, String controlli, String extra)

Preleva una tupla di valori all'interno della tabella indicata, selezionandola secondo i parametri di controllo indicati. Dovrà eseguire una query sulla *tabella* indicata, prelevando i valori all'interno delle *colonne* secondo le condizioni indicate in *controlli* e *extra*.



Nel caso dovesse essere sollevata un'eccezione, la funzione dovrà ritornare **null**. Altrimenti dovrà ritornare quanto ritornato dalla query.

3.4.14 com.safetyGame.back.access.UpdateLog



The diagram shows a UML class named **UpdateLog**. It has one private attribute, **serverAzienda : Indirizzo**, indicated by a minus sign before the attribute name. It has four public methods: **UpdateLog(azienda : Indirizzo)**, **scriviLogDomande(nomeTabella : String, dataOra : String, id : String, operazione : String) : void**, **scriviLogQuattro(nomeTabella : String, dip : Dipendente, dataOra : String, colonna : String, descrizione : String) : void**, and **scriviLogTre(nomeTabella : String, dip : Dipendente, dataOra : String) : void**.

Funzione

Questa classe fungerà da contenitore per i dati riguardanti i Badge che sono stati inseriti all'interno del database aziendale

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- `back.controller.GestioneLog`

Inoltre utilizzerà le seguenti classi:

- `back.access.Indirizzo`

Attributi

- **private Indirizzo serverAzienda**

Riferimento all'istanza della classe `Indirizzo` contenente i riferimenti al server aziendale

Metodi

+ **UpdateLog(Indirizzo azienda)**

Costruttore che imposta il valore degli attributi secondo il valore dei parametri passati.

+ **void scriviLogTre(String nomeTabella, Dipendente dip, String dataOra)**

Metodo che scrive un log di login o logout nelle relative tabelle.

Dovrà prelevare i parametri che gli sono stati passati, riordinarli e chiamare la classe `Indirizzo` affinchè inserisca una nuova riga nel database aziendale.

+ **void scriviLogQuattro(String nomeTabella, Dipendente dip, String dataOra, String colonna, String descrizione)**

Metodo che scrive un log specifico nella tabella data.

Dovrà prelevare i parametri che gli sono stati passati, riordinarli e chiamare la classe Indirizzo affinchè inserisca una nuova riga nel database aziendale.

3.5 Package com.safetyGame.back.condivisi

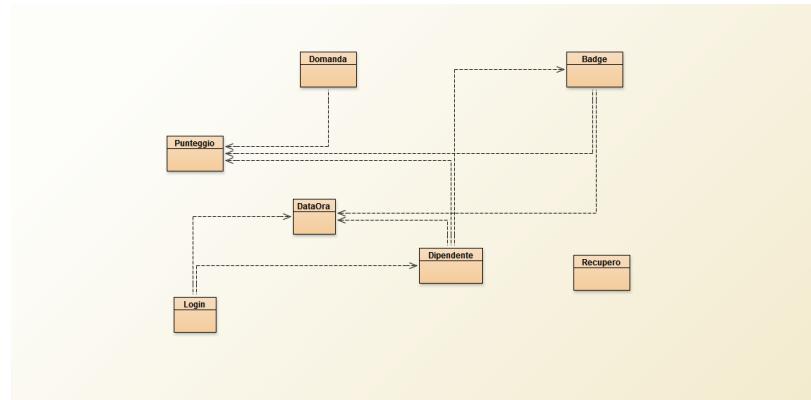


Figura 6: *Package com.safetyGame.back.condivisi*

Tipo, obiettivo e funzione del componente: il package contiene le classi che conterranno le informazioni che dovranno essere condivise fra i vari strati del *Back-end*

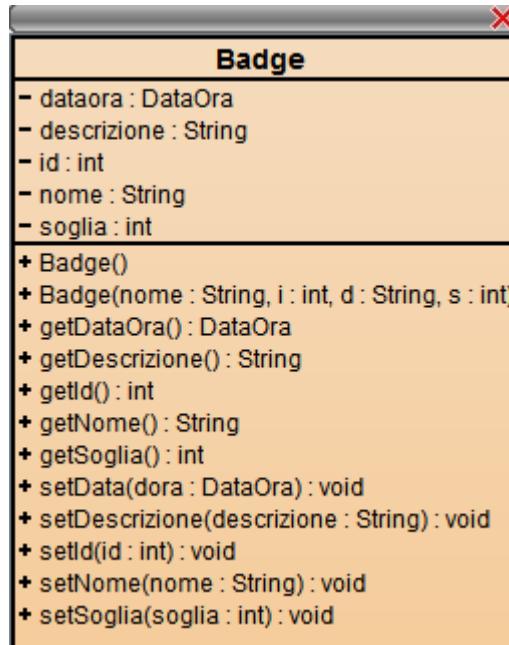
Relazioni d'uso di altre componenti: non utilizza alcun componente del *Back-end*

Interfacci con e relazioni d'uso da altre componenti: viene utilizzato da tutte le classi dei package *controller*, *access* e *connection* per trasmettere informazioni complesse

Attività svolte e dati trattati: serve a trasferire le informazioni che dovranno essere condivise fra i vari package che compongono il *back-end*. In particolare si occuperà di contenere le informazioni riguardanti:

- Badge
- Dipendenti
- Domande
- Login
- Punteggi
- Richieste di recupero password

3.5.1 com.safetyGame.back.condivisi.Badge



Funzione

Questa classe fungerà da contenitore per i dati riguardanti i Badge che sono stati inseriti all'interno del database aziendale

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.access.SqlDAOBadge
- back.condivisi.Dipendente
- back.connection.WebConnection
- back.controller.GestioneBadgeAS
- back.controller.GestioneBadgeD
- back.controller.GestioneDati
- back.controller.GestioneLog



Attributi

- String nome

Nome del badge

- String descrizione

Descrizione del badge (ex. la motivazione per cui si è guadagnato questo badge)

- Punteggio soglia

Indica il quantitativo di punti necessari all'acquisizione del badge

Metodi

+ Badge(String nome, String descr, Punteggio punt)

Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

+ Badge()

Costruttore di default che dovrà inizializzare ogni attributo a **NULL**

Vengono quindi definiti i metodi getter/setter per i vari attributi della classe. In particolare:

+ String getNome()

+ void setNome(String nome)

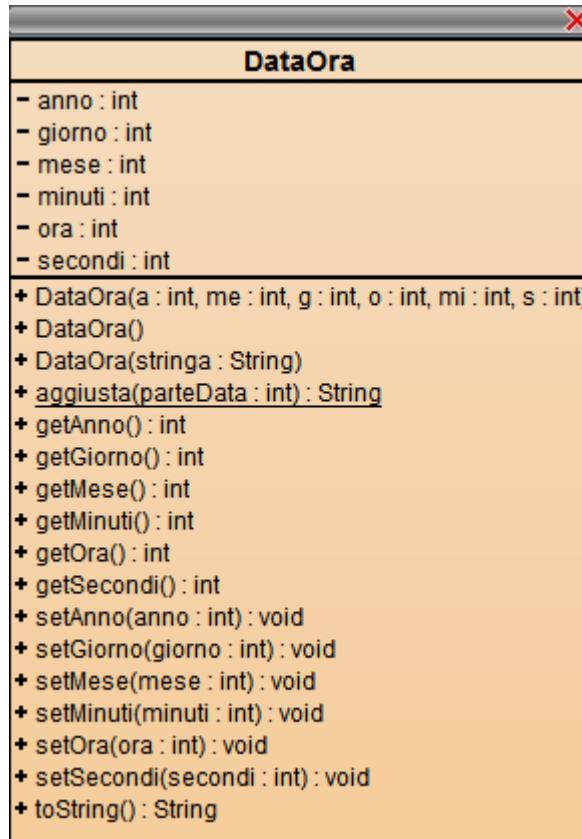
+ String getDescrizione()

+ void setDescrizione(String descrizione)

+ Punteggio getSoglia()

+ void setSoglia(Punteggio soglia)

3.5.2 com.safetyGame.back.condivisi.DataOra



Funzione

Questa classe fungerà da contenitore per le date e le ore

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.access.SqlDAOBadge
- back.access.SqlDAODipendenti
- back.condivisi.badge
- back.condivisi.Dipendente
- back.condivisi.Login
- back.controller.GestioneLog

Attributi

- int anno
- int mese
- int giorno
- int ora
- int minuti
- int secondi

Metodi

+ DataOra(int a, int me, int g, int o, int mi, int s)

Costruttore che imposta gli attributi dell'oggetto alla data e all'ora specificata

+ DataOra()

Costruttore di default che crea l'oggetto con la data e l'ora corrente

+ static String aggiusta(int parteData)

Funzione statica che deve restituire la parte della data (ex. giorno) scritta con due cifre (ex. se si è al giorno “2”, questa funzione deve restituire una stringa con scritto “02”)

+ String toString()

Restituisce una stringa dell'oggetto nel seguente formato:

AAAA/MM/GG HH:MM:SS

Vengono quindi definiti i metodi getter/setter per i vari attributi della classe. In particolare:

- + int getAnno()
- + void setAnno(int anno)
- + int getMese()
- + void setMese(int mese)
- + int getGiorno()
- + void setGiorno(int giorno)

```
+ int getOra()  
+ void setOra(int ora)  
+ int getMinuti()  
+ void setMinuti(int minuti)  
+ int getSecondi()  
+ void setSecondi(int secondi)
```

3.5.3 com.safetyGame.back.condivisi.Dipendente



Funzione

Questa classe fungerà da contenitore per i dati riguardanti i Dipendenti che sono stati inseriti all'interno del database aziendale

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.access.SqlDAOBadge
- back.access.SqlDAODipendenti
- back.access.SqlDAODomande
- back.access.SqlDAOLogin
- back.access.SqlDAOPunteggi
- back.condivisi.Domanda

- back.condivisi.Login
- back.condivisi.Punteggio
- back.condivisi.Recupero
- back.connection.ApplicazioniConnection
- back.connection.WebConnection
- back.controller.GestioneBadgeD
- back.controller.GestioneDati
- back.controller.GestioneDipendentiAA
- back.controller.GestioneDipendentiID
- back.controller.GestioneDomandeD
- back.controller.GestioneLog
- back.controller.GestioneLogin
- back.controller.GestionePunteggiAA
- back.controller.GestionePunteggiD

Attributi

- int id

Numero univoco che identifica l'utente all'interno del database

- String codFiscale

Contiene il codice fiscale dell'utente

- ArrayList<Badge>

Array contenente tutti i badge guadagnati da un utente

- Punteggio punteggio

Contiene alcune statistiche riguardanti le risposte date e i punti guadagnati dall'utente

- String nome

Contiene il nome dell'utente

- String cognome

Contiene il cognome dell'utente

- String email

Contiene l'indirizzo email dell'utente

- **String nickname**

Contiene l'username assegnato all'utente

- **String password**

Contiene la password dell'utente

- **String ruolo**

Contiene il ruolo che l'utente ha all'interno dell'azienda

- **boolean ammA**

True = L'utente è un amministratore azienda; False = L'utente è un amministratore sicurezza

- **DataOra dataModPass**

Contiene la data e l'ora dell'ultima modifica alla password

- **String nuovaPass**

Contiene la password che dovrà essere scritta nel database

- **int trofei**

Contiene il numero di trofei guadagnati

Metodi

+ **Dipendente()**

+ **Dipendente(int id, String cf, String n, String c, String e, String nn, String p, String r, int pu, String np, int trf)**

+ **Dipendente(boolean aA, DataOra dmp, String np, String mail, String nick, String pass, String codfisc, int i)**

Vengono quindi definiti i metodi getter/setter per i vari attributi della classe. In particolare:

+ **int getId()**

+ **void setId(int id)**

+ **String getCodFiscale()**

+ **void setCodFiscale(String codFiscale)**

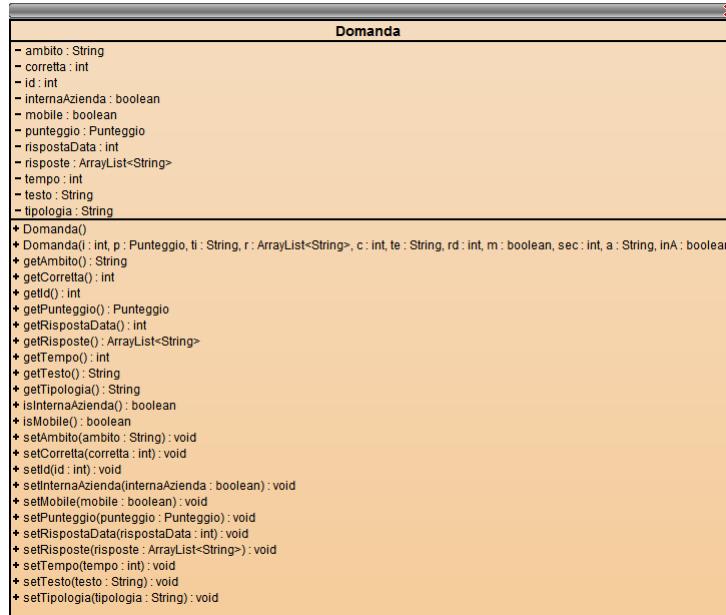
+ **ArrayList<Badge> getBadges()**

```
+ void setBadges(ArrayList<Badge> badges)
+ void addBadge(Badge badge)
+ Punteggio getPunteggio()
+ void setPunteggio(Punteggio punteggio)
+ String getNome()
+ void setNome(String nome)
+ String getCognome()
+ void setCognome(String cognome)
+ String getEmail()
+ void setEmail(String email)
+ String getNickname()
+ void setNickname(String nickname)
+ String getPassword()
+ void setPassword(String password)
+ String getRuolo()
+ void setRuolo(String ruolo)
+ String toStringID()
+ boolean isAmmAA()
+ void setAmmAA(boolean ammAA)
+ DataOra getDataModPass()
+ void SetDataModPass(DataOra dataModPass)
+ String getNuovaPass()
+ void setNuovaPass(String nuovaPass)
```



```
+ int getTrofei()  
+ void setTrofei(int trofei)
```

3.5.4 com.safetyGame.back.condivisi.Domanda



Funzione

Questa classe fungerà da contenitore per le domande che dovranno essere trasmesse ai vari front-end e le relative risposte che da questi verranno indirizzate al back-end

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.access.SqlDAODomande
- back.access.SqlDAO_Punteggio
- back.condivisi.Punteggio
- back.connection.ApplicazioniConnection
- back.connection.WebConnection
- back.controller.GestioneBadgeD
- back.controller.GestioneDati
- back.controller.GestioneDomandeAS
- back.controller.GestioneDomandeD

- back.controller.GestioneLog
- mobile.condivisi.Domanda
- mobile.Utils.ConnectionUtils
- mobile.View.DashboardActivity
- mobile.View.DatiActivity
- mobile.View.DomandaActivity
- mobile.View.PunteggiActivity
- mobile.View.TimerNotifica

Attributi

- int id

Numero univoco che identifica la domanda all'interno del database

- Punteggio punteggio

Punteggio attribuito alla domanda

- String tipologia

Identifica la tipologia della domanda (ex. Risposta multipla, risposta aperta, ecc...)

- ArrayList<String> risposte

Conterrà tutte le possibili risposte

- int corretta

Identifica all'interno della lista *risposte* quella corretta

- String testo

Contiene il testo della domanda

- int rispostaData

Identifica qual'è stata la risposta selezionata da un dipendente. rispostaData=-1 se non è ancora stata selezionata alcuna risposta

- boolean mobile

Identifica se la domanda è stata proposta o dovrà essere proposta sarà destinata ad un dispositivo mobile

- int tempo

Identifica il tempo permesso per rispondere alla domanda. tempo=-1 se non è stato assegnato un parametro temporale alla domanda (ovvero si potrà rispondere alla domanda impiegando quanto "tempo si vuole")

- String ambito

Identifica il “settore” di appartenenza della domanda (per esempio se la domanda è attinente alle norme anti-incendio oppure a quelle anti-infortunistica)

- boolean internaAzienda

Indica se la domanda può essere proposta ai Dipendenti in quanto selezionata dall’Amministratore Sicurezza

Metodi

+ Domanda(int i, Punteggio p, String ti, ArrayList<String> r, int c, String te, int rd, boolean m, int sec, String a, boolean inA)

Costruttore con parametri. Dovrà inizializzare i campi dati con le variabili passate

Vengono quindi definiti i metodi getter/setter per i vari attributi della classe. In particolare:

+ int getId()

+ void setId(int id)

+ Punteggio getPunteggio()

+ void setPunteggio(Punteggio punteggio)

+ String getTipologia()

+ void setTipologia(String tipologia)

+ ArrayList<String> getRisposte()

+ void setRisposte(ArrayList<String> risposte)

+ int getCorretta()

+ void setCorretta(int corretta)

+ String getTesto()

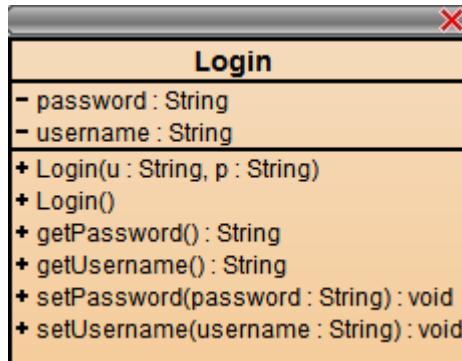
+ void setTesto(String testo)

+ int getRispostaData()

+ void setRispostaData(int rispostaData)

```
+ boolean isMobile()  
+ void setMobile(boolean mobile)  
+ int getTempo()  
+ void setTempo(int tempo)  
+ String getAmbito()  
+ void setAmbito(String ambito)  
+ boolean isInternaAzienda()  
+ void setInternaAzienda(boolean internaAzienda)
```

3.5.5 com.safetyGame.back.condivisi.Login



Funzione

Questa classe fungerà da contenitore per i dati riguardanti i tentativi di autenticarsi all'interno della piattaforma

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.access.SqlDAODependenti
- back.access.SqlDAOLogin
- back.connection.ApplicazioniConnection
- back.connection.WebConnection
- back.controller.GestioneBadgeD
- back.controller.GestioneDati
- back.controller.GestioneDipendentiD
- back.controller.GestioneDomandeD
- back.controller.GestioneLog
- back.controller.GestioneLogin
- back.controller.GestionePunteggiD
- desktop.logic.ControlLogin
- desktop.logic.DatiLogin
- desktop.view.Login

- mobile.Utils.ConnectionUtils
- mobile.Utils.IntentIntegrator
- mobile.View.LoginActivity
- mobile.View.TimerNotifica

Attributi

- String username

Username dell'account che tenta di fare login

- String password

Password dell'account che tenta di fare login

Metodi

+ Login()

Costruttore di default senza parametri. I campi dati dovranno essere inizializzati con valori di default

+ Login(String u, String p)

Costruttore con parametri. Dovrà inizializzare i campi dati con le variabili passate

Vengono quindi definiti i metodi getter/setter per i vari attributi della classe. In particolare:

+ String getUsername()

+ void setUsername(String username)

+ String getPassword()

+ void setPassword(String password)

3.5.6 com.safetyGame.back.condivisi.Punteggio

Punteggio	
- mediaPuntiAzienda : double	
- nDomRisp : int	
- nRispCorr : int	
- punti : int	
- puntiPrec : int	
- puntiSuc : int	
+ Punteggio()	
+ Punteggio(punti : int)	
+ getMediaPuntiAzienda() : double	
+ getPunti() : int	
+ getPuntiPrec() : int	
+ getPuntiSuc() : int	
+ getnDomRisp() : int	
+ getnRispCorr() : int	
+ setMediaPuntiAzienda(mediaPuntiAzienda : double) : void	
+ setPunti(punti : int) : void	
+ setPuntiPrec(puntiPrec : int) : void	
+ setPuntiSuc(puntiSuc : int) : void	
+ setnDomRisp(nDomRisp : int) : void	
+ setnRispCorr(nRispCorr : int) : void	

Funzione

Questa classe fungerà da contenitore per i dati riguardanti i punteggi sia dei dipendenti che dell'intera azienda.

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.access.SqlDAODomande
- back.access.SqlDAOPunteggi
- back.condivisi.Dipendente
- back.condivisi.Domanda
- back.connection.ApplicazioniConnection
- back.connection.WebConnection
- back.controller.GestioneDati

- back.controller.GestioneBadgeD
- back.controller.GestionePunteggiAA
- back.controller.GestionePunteggiD

Attributi

- int punti

Indica i punti accumulati da un dipendente O i punti accumulati da tutti i dipendenti dell'azienda

- double mediaPuntiAzienda

Indica il punteggio medio di tutti i dipendenti dell'azienda

- int puntiPrec

Indica il numero di punti ottenuti dal Dipendente direttamente successivo al Dipendente “proprietario” all'interno della classifica dei punteggi (ovvero colui che ha meno punti del proprietario di questo oggetto)

- int puntiSuc

Indica il numero di punti ottenuti dal Dipendente direttamente precedente al Dipendente “proprietario” all'interno della classifica dei punteggi (ovvero colui che ha più punti del proprietario di questo oggetto)

- int nDomRisp

Indica il numero di domande a cui o il Dipendente o tutti i Dipendenti dell'azienda hanno dato risposta

- int nRispCorr

Indica il numero di domande a cui o il Dipendente o tutti i Dipendenti dell'azienda hanno dato risposta corretta

Metodi

+Punteggio()

Costruttore di default senza parametri. I campi dati dovranno essere inizializzati con valori di default

+ Punteggio(int punti)

Costruttore con parametri. Dovrà inizializzare i campi dati con le variabili passate

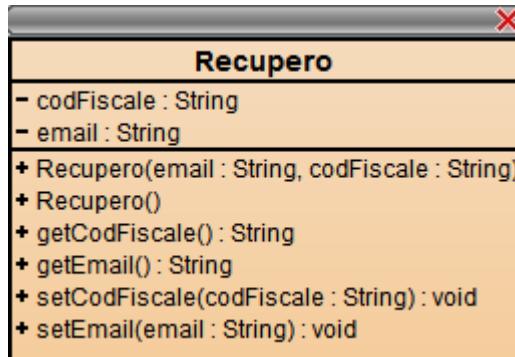
Vengono quindi definiti i metodi getter/setter per i vari attributi della classe. In particolare:

+ int getPunti()



+ void setPunti(int punti)

3.5.7 com.safetyGame.back.condivisi.Recupero



Funzione

Questa classe fungerà da contenitore per i dati riguardanti le richieste di recupero password.

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- back.access.SqlDAO Dipendenti
- back.connection.WebConnection

Attributi

- String email

Indirizzo email a cui si dovrà inviare la mail con la password

- String codFiscale

Codice fiscale dell'account a cui è collegato anche la mail

Metodi

+ Recupero(String email, String codFiscale)

Costruttore con parametri. Dovrà inizializzare i campi dati con le variabili passate

+ Recupero()

Costruttore di default senza parametri. I campi dati dovranno essere inizializzati con valori di default

Vengono quindi definiti i metodi getter/setter per i vari attributi della classe. In particolare:



```
+ String getEmail()  
+ void setEmail(String email)  
+ String getCodFiscale()  
+ void setCodFiscale(String codFiscale)
```

4 Specifica Front-end Desktop

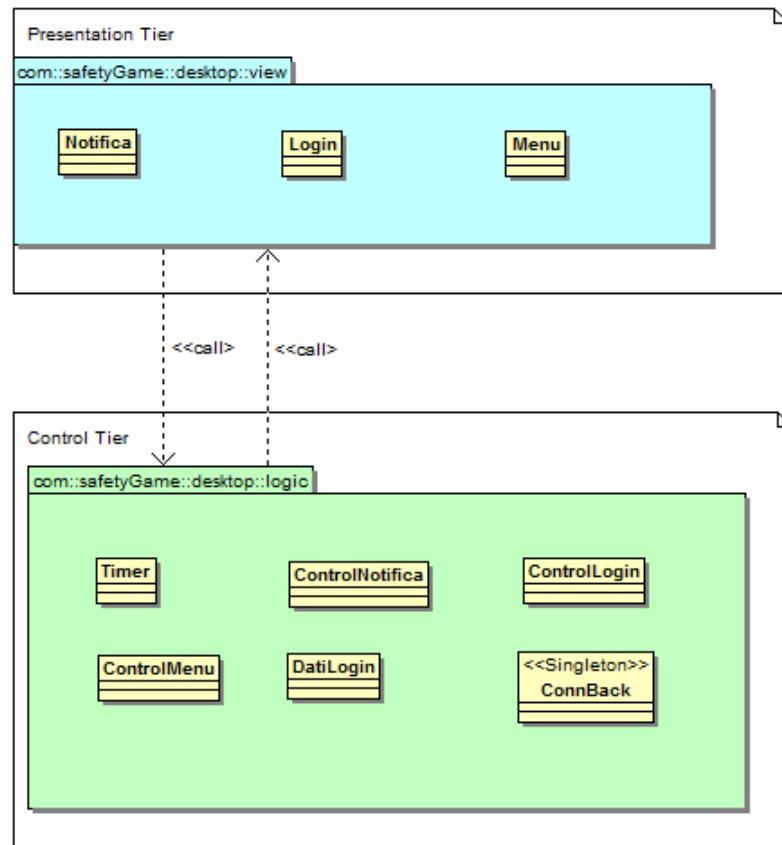


Figura 7: *Front-end Desktop, diagramma dei package*

4.1 Package com.safetyGame.desktop.view

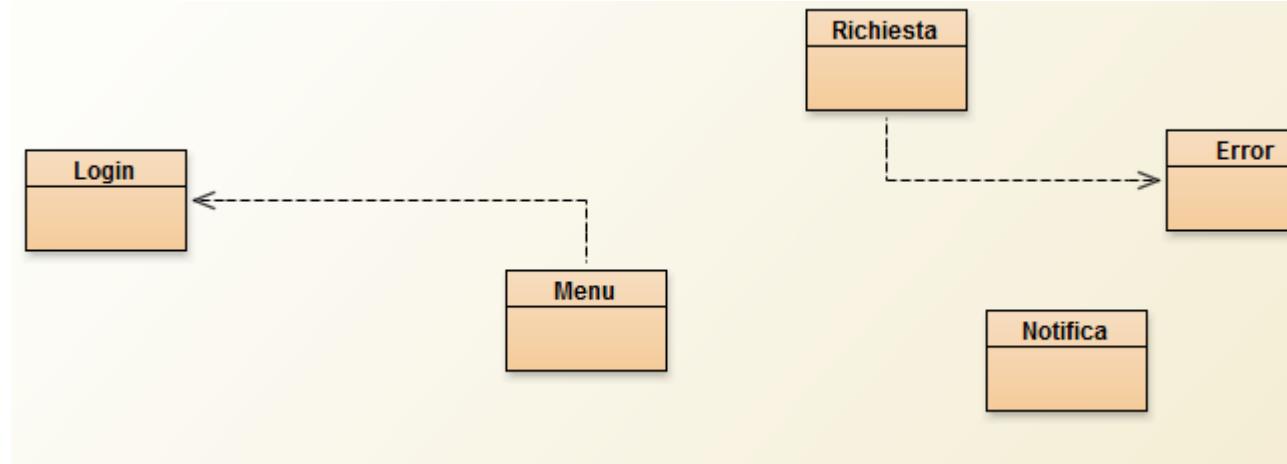


Figure 8: Package `com.safetyGame.desktop.view`

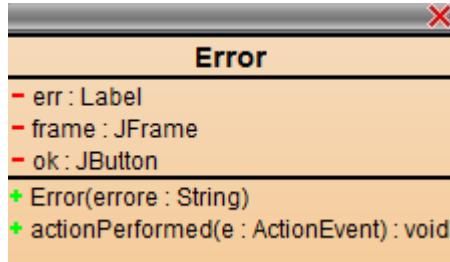
Tipo, obiettivo e funzione del componente: contiene le classi che istanzieranno gli oggetti per la grafica dell'applicativo

Relazioni d'uso di altre componenti: utilizza le classi contenute nel package `com.safetyGame.desktop.logic` per attivare le funzioni mostrate a video e quelle nel package `com.safetyGame.desktop.condivisi` per lo scambio dei dati tra le componenti e il Back-end.

Interfacce con e relazioni d'uso da altre componenti: viene utilizzato dal package `com.safetyGame.desktop.logic` per mostrare le modifiche elaborate

Attività svolte e dati trattati: funge da GUI dell'applicativo desktop per i Dipendenti

4.1.1 com.safetyGame.desktop.view.Error



Funzione

Questa classe si occuperà di presentare al Dipendente un popup nel caso di errori

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata e utilizzerà le seguenti classi:

- desktop.logic.ConnBack
- desktop.view.Richiesta

Attributi

- JFrame frame

Riferimento alla schermata visualizzata

- JButton ok

Riferimento al pulsante “OK”

- Label err

Stringa contenente l'errore da mostrare

Metodi

+ Error(String errore)

Costruttore della classe Error.

Dovrà creare una nuova JFrame con titolo “*Errore*”, un pulsante per chiudere la finestra con scritto “OK” e all’interno dovrà essere contenuta la stringa passata come parametro.

+ void actionPerformed(ActionEvent e)

Metodo che gestisce le azioni che i pulsanti intraprenderanno.

Dovrà controllare l’origine di una *ActionEvent* e, nel caso provenisse dal *JButton* ok, dovrà rendere invisibile l’oggetto.

4.1.2 com.safetyGame.desktop.view.Login



Funzione

Questa classe si occuperà di presentare al Dipendente un popup che gli permetta di effettuare il login nel sistema

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata e utilizzerà le seguenti classi:

- desktop.logic.ControlLogin
- desktop.view.Menu

Attributi

- JFrame frame

Riferimento alla schermata di login

- JFrame frame_recupero



Riferimento alla schermata per il recupero della password

- JButton ok

Riferimento al pulsante “OK” contenuto nella schermata di login

- JButton annulla

Riferimento al pulsante “Annulla” contenuto nella schermata di login

- JButton okr

Riferimento al pulsante “OK” contenuto nella schermata del recupero password

- JButton annular

Riferimento al pulsante “Annulla” contenuto nella schermata del recupero password

- JButton passdim

Riferimento al pulsante “Password dimenticata” contenuto nella schermata di login

- Label userl

Riferimento alla label “Username” associata al campo testuale per inserire l’username contenuto nella schermata di login

- Label passl

Riferimento alla label “Password” associata al campo testuale per inserire la password contenuto nella schermata di login

- Label errore

Riferimento alla label per mostrare un eventuale messaggio di errore contenuto nella schermata di login

- Label errorer

Riferimento alla label per mostrare un eventuale messaggio di errore contenuto nella schermata di recupero password

- Label codfisl

Riferimento alla label associata al campo testuale per inserire il codice fiscale contenuto nella schermata di recupero password

- Label maill

Riferimento alla label associata al campo testuale per inserire la email contenuta nella schermata di recupero password

- TextField mail



Riferimento all'area di testo per inserire la email contenuto nella schermata di recupero password

- **TextField codfis**

Riferimento all'area di testo per inserire il codice fiscale contenuto nella schermata di recupero password

- **TextField username**

Riferimento all'area di testo per inserire lo username contenuto nella schermata di login

- **JPasswordField password**

Riferimento all'area di testo mascherata per inserire la password contenuta nella schermata di login

- **ControlLogin controller**

Riferimento al controllore del Login

Metodi

+ **Login(ControlLogin controllore)**

costruttore che dovrà inizializzare gli attributi

- **void creaSubFrameRecupero()**

metodo che crea una sotto frame per richiedere il recupero password

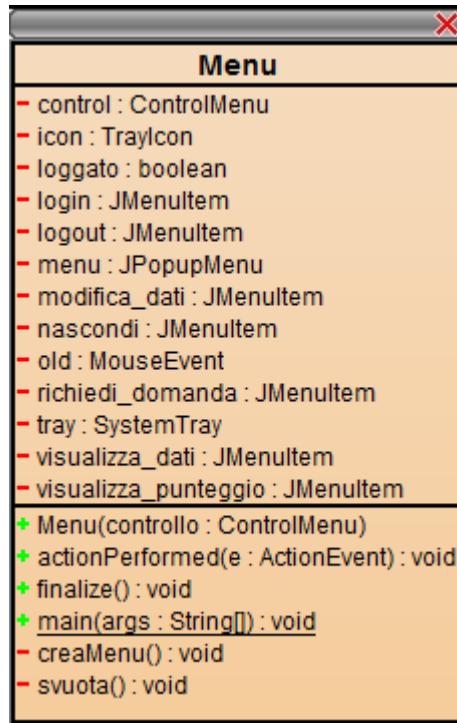
+ **void actionPerformed(ActionEvent e)**

metodo che dovrà gestire in modo corretto gli eventi generati dai pulsanti della grafica

+ **boolean isVisible()**

metodo che ritorna la visibilità della finestra principale

4.1.3 com.safetyGame.desktop.view.Menu



Funzione

Questa classe si occuperà di presentare al Dipendente e al Dipendente Autenticato tutta la serie di menù della System Tray che gli permetteranno di utilizzare le varie funzioni previste

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata e utilizzerà le seguenti classi:

- desktop.view.Login
- desktop.logic.ControlMenu

Attributi

- **boolean loggato=false**
booleano che indica se un dipendente ha effettuato il login
- **ControlMenu control**



Riferimento al controllore del Menu

- **SystemTray tray**

Riferimento alla icona di sistema nella barra delle applicazioni

- **TrayIcon icon**

Riferimento all'immagine dell'icona di sistema

- **JPopupMenu menu**

Riferimento al menu di popup

- **JMenuItem login**

Riferimento alla voce "login" del menu di popup

- **JMenuItem richiedi_domanda**

Riferimento alla voce "domanda" del menu di popup

- **JMenuItem visualizza_punteggio**

Riferimento alla voce "visualizza punteggio" del menu di popup

- **JMenuItem visualizza_dati**

Riferimento alla voce "visualizza dati" del menu di popup

- **JMenuItem modifica_dati**

Riferimento alla voce "modifica dati" del menu di popup

- **JMenuItem logout**

Riferimento alla voce "logout" del menu di popup

- **JMenuItem nascondi**

Riferimento alla voce "nascondi" del menu di popup

- **MouseEvent old**

Riferimento al precedente evento del Mouse

Metodi

+ **Menu()**

costruttore che dovrà inizializzare gli attributi

- **void svuota()**

metodo che svuota le voci del menu

- **void creaMenu()**



metodo che dovrà creare i menu della System Tray

+ **void finalize()**

metodo che chiuderà correttamente l'applicazione eliminando l'icona di sistema

+ **void actionPerformed(ActionEvent e)**

metodo che gestisce le azioni delle voci del menu

+ **static void main(String args [])**

metodo che inizializza il front end

4.1.4 com.safetyGame.desktop.view.Notifica



Funzione

Questa classe si occuperà di presentare al Dipendente Autenticato un popup che gli permetta di accettare o posticipare la visualizzazione di una domanda

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata e utilizzerà le seguenti classi:

- desktop.logic.ControlNotifica
- desktop.logic.ConnBack

Attributi

- JFrame frame

Riferimento alla schermata della notifica

- JButton si

Riferimento alla voce “accetta” della notifica

- JButton posticipa

Riferimento alla voce “posticipa” della notifica

- Label testo

Riferimento alla label per mostrare il testo della proposta di nuova domanda

- ControlNotifica controller

Riferimento al controllore della Notifica

Metodi

+ Notifica(ControlNotifica controllore)

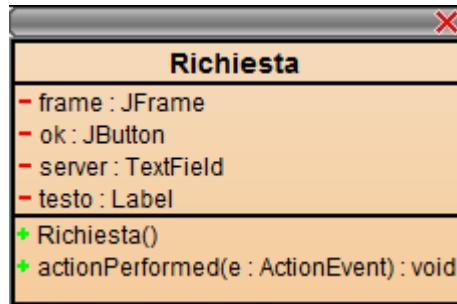


costruttore che dovrà inizializzare gli attributi

+ **void actionPerformed(ActionEvent e)**

metodo che dovrà gestire in modo corretto gli eventi generati dai pulsanti della grafica

4.1.5 com.safetyGame.desktop.view.Richiesta



Funzione

Questa classe si occuperà di presentare all'Utilizzatore una finestra per inserire l'indirizzo del server del Back end

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata e utilizzerà le seguenti classi:

- desktop.logic.ConnBack

Attributi

- JFrame frame

Riferimento alla schermata della notifica

- JButton ok

Riferimento alla voce "ok" della richiesta di indirizzo del server

- Label testo

Riferimento alla label per mostrare la richiesta di inserimento di indirizzo del server

- TextField server

Riferimento all'area di testo per inserire l'indirizzo del server

Metodi

+ Richiesta()

costruttore che dovrà inizializzare gli attributi

+ void actionPerformed(ActionEvent e)

metodo che dovrà gestire in modo corretto gli eventi generati dai pulsanti della grafica

4.2 Package com.safetyGame.desktop.logic

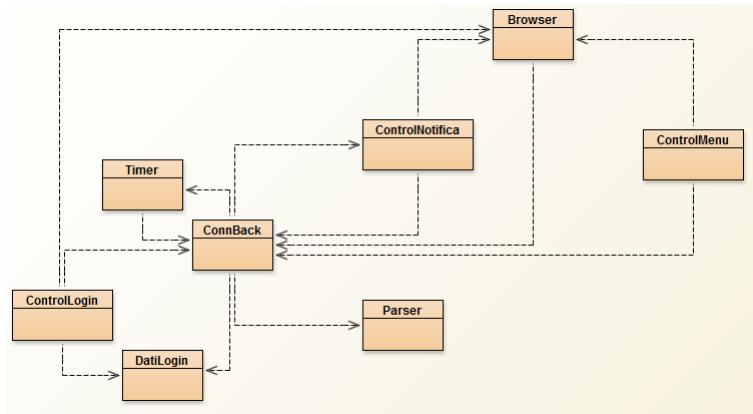


Figure 9: Package `com.safetyGame.desktop.logic`

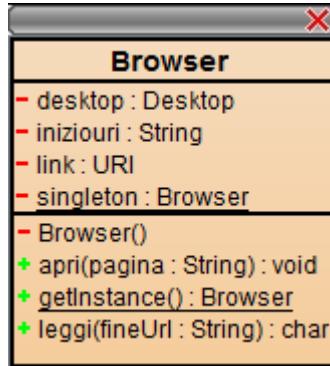
Tipo, obiettivo e funzione del componente: contiene le classi che effettueranno i controlli dei vari componenti e faranno in modo che l'applicazione si sviluppi nel modo corretto

Relazioni d'uso di altre componenti: utilizza le classi contenute nel package **com.safetyGame.desktop.view** per mostrare correttamente i dati elaborati e quelle nel package **com.safetyGame.desktop.condivisi** per lo scambio dei dati tra le componenti e il Back-end. Utilizza inoltre il Front-end web per aprire le funzioni una volta che il Dipendente è autenticato e comunica direttamente con Back-end per le funzioni di login, logout e recupero password.

Interfacce con e relazioni d'uso da altre componenti: viene utilizzato dal package `com.safetyGame.desktop.view` per elaborare le funzioni richieste dal Dipendente

Attività svolte e dati trattati: utilizza i dati che gli arrivano dal Backend e li elabora per mostrare al Dipendente la corretta scelta di opzioni del menu. Gestisce il timer per la notifica della proposta di una nuova domanda e gestisce l'interfacciamento con il web per la maggior parte delle funzioni a disposizione del Dipendente Autenticato

4.2.1 com.safetyGame.desktop.view.Browser



Funzione

Questa classe si occuperà di gestire il browser di sistema.

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata e utilizzerà le seguenti classi:

- desktop.logic.ControlNotifica
- desktop.logic.ControlMenu
- desktop.logic.ControlLogin
- desktop.logic.ConnBack

Attributi

- static Browser singleton

Riferimento alla classe Browser

- Desktop desktop

Riferimento al gestore del sistema per aprire le pagine web

- URI link

Riferimento all'uri completo della pagina web

- String iniziouri

Riferimento all'inizio dell'uri

Metodi

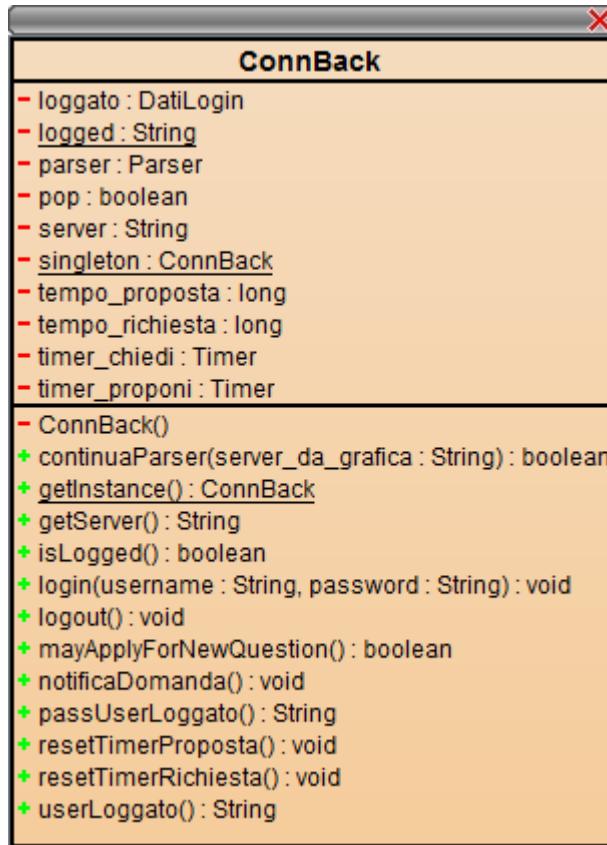
- Browser()



costruttore che dovrà inizializzare gli attributi

- + **static Browser getInstance()**
metodo statico che ritorna la corretta istanza della classe Browser
- + **void apri(String pagina)**
metodo che apre la pagina web nel browser
- + **char leggi(String fineUrl)**
metodo che apre una pagina web in lettura

4.2.2 com.safetyGame.desktop.view.ConnBack



Funzione

Questa classe si occuperà di gestire i timer per le domande, gestire il Dipendente autenticato e centralizzare l'esecuzione dell'applicazione

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata e utilizzerà le seguenti classi:

- desktop.logic.ControlNotifica
- desktop.logic.ControlMenu
- desktop.logic.ControlLogin
- desktop.logic.DatiLogin
- desktop.logic.Timer

- desktop.logic.Parser
- desktop.logic.Browser
- desktop.view.Error
- desktop.view.Richiesta

Attributi

- **static ConnBack singleton = null**

Riferimento alla classe ConnBack

- **DatiLogin loggato**

Riferimento alla classe DatiLogin, indica il Dipendente loggato

- **String server**

Riferimento alla server del Back-end

- **static String logged**

Riferimento allo username dell'oggetto DatiLogin

- **Parser parser**

Riferimento alla classe Parser

- **boolean pop**

Boleano che indica se è stata notificata una domanda

- **long tempo_proposta**

Long che indica quanto tempo deve passare prima di proporre una nuova domanda

- **long tempo_richiesta**

Long che indica quanto tempo deve passare prima di poter richiedere una nuova domanda

- **Timer timer_chiedi**

Riferimento alla classe Timer che scandisce il tempo per inibire la richiesta di una nuova domanda

- **Timer timer_proponi**

Riferimento alla classe Timer che scandisce il tempo per la proposta di una nuova domanda



Metodi

- **ConnBack()**

costruttore che dovrà inizializzare gli attributi

+ **static ConnBack getInstance()**

metodo statico che ritorna la corretta istanza della classe ConnBack

+ **boolean continuaParser(String server_da_grafica)**

metodo che fa continuare il costruttore ottenendo l'indirizzo del server dalla grafica e salvandolo

+ **String getServer()**

metodo statico che ritorna la stringa del server

+ **boolean login(String username, String password)**

metodo statico che imposta le variabili interne dell'applicazione al login

+ **void logout()**

metodo statico che imposta le variabili in modo corretto all'uscita del Dipendente

+ **boolean mayApplyForNewQuestion()**

metodo che indica se un Dipendente può o meno venire sottoposto ad una domanda sotto la sua richiesta

+ **boolean isLoggedIn()**

metodo che indica se Dipendente è loggato

+ **void notificaDomanda()**

metodo statico che crea la notifica di una nuova domanda

+ **String passUserLoggato()**

metodo che ritorna la password del dipendente loggato

+ **String userLoggato()**

metodo che ritorna lo username del dipendente loggato

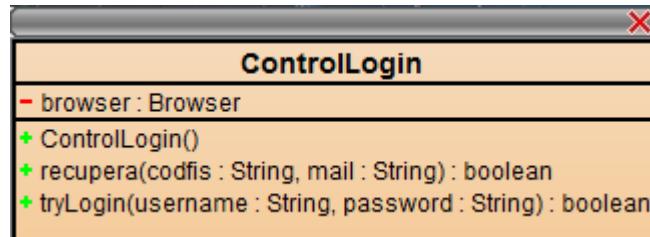
+ **void resetTimerProposta()**

Metodo che resetta il timer della domanda proposta dall'applicazione

+ **void resetTimerRichiesta()**

Metodo che resetta il timer della domanda richiesta dal dipendente

4.2.3 com.safetyGame.desktop.logic.ControlLogin



Funzione

Questa classe gestirà le informazioni inserite nell'oggetto grafico Login e si occuperà di mandare l'autenticazione al Back-end

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata e utilizzerà le seguenti classi:

- desktop.view.Login
- desktop.logic.ConnBack
- desktop.logic.Browser
- desktop.logic.DatiLogin

Attributi

- Browser browser

Riferimento al gestore del browser

Metodi

+ControlLogin()

costruttore che dovrà inizializzare gli attributi con i parametri passati per riferimento

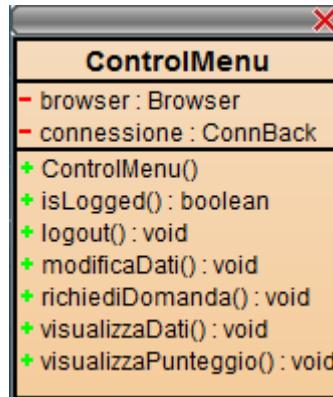
+void recupera(String codfis, String mail)

metodo richiamerà il browser per impostare la richiesta di nuova password

+void tryLogin(String username, String password)

metodo che richiamerà il browser per tentare di effettuare il login

4.2.4 com.safetyGame.desktop.logic.ControlMenu



Funzione

Questa classe si occuperà di gestire tutte le funzionalità che la System Tray offre

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata e utilizzerà le seguenti classi:

- desktop.view.Menu
- desktop.logic.Browser
- desktop.logic.ConnBack

Attributi

- ConnBack connessione

Riferimento al gestore della connessione

- Browser browser

Riferimento al gestore del browser

Metodi

+ ControlMenu()

costruttore che dovrà inizializzare gli attributi

+ boolean isLoggedIn()

metodo che indica se un Dipendente ha effettuato il login

+ void logout()

metodo che richiamerà il browser per aprire il Front-end web per effettuare il logout

+void richiediDomanda()

metodo che richiamerà il browser per aprire il Front-end web con la domanda richiesta, sempre se il controllo sul timer è corretto

+void visualizzaPunteggio()

metodo che richiamerà il browser per aprire il Front-end web con il punteggio del Dipendente

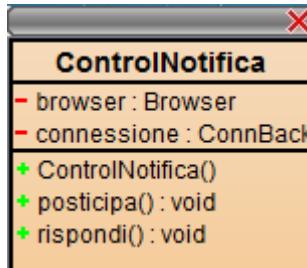
+void visualizzaDati()

metodo che richiamerà il browser per aprire il Front-end web con i dati personali del Dipendente

+void modificaDati()

metodo che richiamerà il browser per aprire il Front-end web con la pagina di modifica dei dati personali

4.2.5 com.safetyGame.desktop.logic.ControlNotifica



Funzione

Questa classe gestirà le informazioni date dall'oggetto grafico Notifica e si occuperà di mandare le corrette informazioni al Back-end eventualmente richiamando il web. Si occuperà inoltre di gestire il timer per sapere quando effettuare il popup.

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata e utilizzerà le seguenti classi:

- desktop.view.Notifica
- desktop.logic.ConnBack
- desktop.logic.Browser

Attributi

- ConnBack connessione

Riferimento al gestore della connessione

- Browser browser

Riferimento al gestore del browser

Metodi

+ControlNotifica()

costruttore che dovrà inizializzare gli attributi con i parametri passati per riferimento

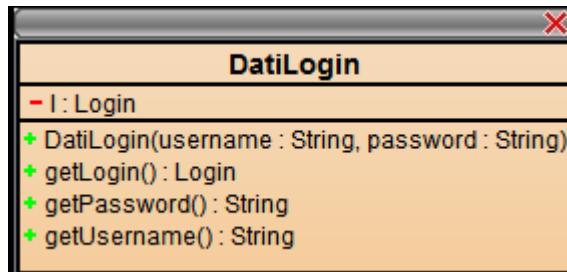
+void posticipa()

metodo che richiamerà il browser e gli farà posticipare la domanda

+void rispondi()

metodo che richiamerà il browser per aprire il Front-end web con la domanda proposta

4.2.6 com.safetyGame.desktop.logic.DatiLogin



Funzione

Questa classe wrapper si occupa di ritornare in maniera più semplice e corretta i valori contenuti nella classe desktop.condivisi.Login

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata e utilizzerà le seguenti classi:

- desktop.condivisi.Login
- desktop.logic.ControlLogin
- desktop.logic.ControlMenu
- desktop.logic.ConnBack
- desktop.view.Menu
- desktop.view.Login

Attributi

- Login I

riferimento all'oggetto condiviso Login

Metodi

- + **DatiLogin(String username, String password)**
costruttore che dovrà inizializzare gli attributi
- + **String getUsername()**
metodo che ritorna lo username
- + **String getPassword()**
metodo che ritorna la password
- + **Login getLogin()**
metodo che ritorna l'oggetto Login da passare al Back-end

4.2.7 com.safetyGame.desktop.logic.Parser



Funzione

Questa classe si occuperà di recuperare i dati del server dell'azienda dal file di testo dedicato. Se il file non esiste, viene creato e viene richiesto all'Utilizzatore di inserire un indirizzo

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata e utilizzerà le seguenti classi:

- desktop.logic.ConnBack

Attributi

- boolean aperto

indica se il file in lettura è aperto o meno

- File file

Riferimento al flusso di dati verso il file

- FileReader in

Riferimento allo stream in input

Metodi

+ Parser()

costruttore che dovrà inizializzare gli attributi e provare ad aprire il file in lettura

+ String leggi()

metodo che servirà a leggere la stringa contenente l'indirizzo del database dell'azienda

+boolean scrivi(String server)

metodo che scriverà sul file appena creato l'indirizzo dell'azienda così da non doverlo richiedere ad ogni avvio del programma

- void apri()

metodo che apre il file in lettura

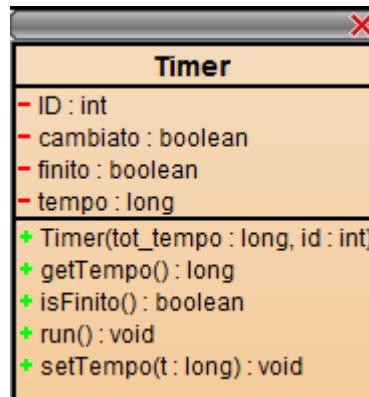
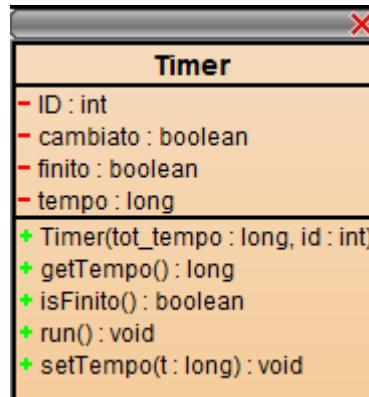
+boolean isOpen()

metodo che indica se il file è aperto in lettura o no

+void finalize()

metodo che chiude correttamente il flusso di dati verso il file

4.2.8 com.safetyGame.desktop.logic.Timer



Funzione

Questa classe si occuperà di tenere il tempo per proporre una domanda e per abilitare l'opzione di richiedere una domanda. Per semplificare estenderà la classe Thread

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata e utilizzerà le seguenti classi:

- desktop.logic.ControlMenu
- desktop.logic.ControlNotifica
- desktop.logic.ConnBack

Attributi

- **long tempo**

tempo in millisecondi

- **boolean cambiato**

booleano che indica se il tempo è stato cambiato durante l'attesa

- **int ID**

intero che identifica il numero del thread Timer

- **boolean finito**

booleano che indica se il tempo è finito

Metodi

+ **Timer(long tot_tempo, int id)**

costruttore che dovrà inizializzare il tempo in millisecondi

+**void run()**

metodo che gestirà il tempo di attesa prima di mettersi in attesa passiva. Il metodo mette in attesa il thread per il tempo specificato, una volta che ha finito, imposta la variabile finito a true e rimane in attesa del "prossimo giro"

+**void setTempo(int t)**

metodo che permette di resettare il tempo dell'oggetto, così da evitare sprechi di memoria per creare altri thread

+**boolean getFinito()**

metodo che ritorna lo stato del termine di un ciclo di attesa

+**long getTempo()**

metodo che ritorna il tempo con cui è stato inizializzato l'oggetto

4.3 Package com.safetyGame.desktop.condivisi

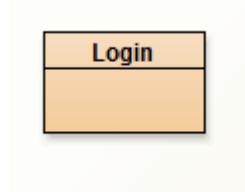


Figure 10: Package **com.safetyGame.desktop.condivisi**

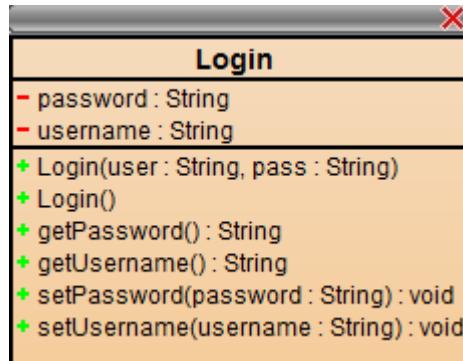
Tipo, obiettivo e funzione del componente: contiene parte delle classi contenute nel package **com.safetyGame.back.condivisi** del Back-end per garantire completa compatibilità nello scambio di informazioni.

Relazioni d'uso di altre componenti: non utilizza alcuna altra classe.

Interfacce con e relazioni d'uso da altre componenti: viene utilizzato dal package **com.safetyGame.desktop.logic** per comunicare con il back-end

Attività svolte e dati trattati: gestisce correttamente i dati ricevuti e inviati al Back-end

4.3.1 com.safetyGame.desktop.condivisi.Login



Funzione

La funzione di questa classe è identica a quella della classe contenuta in **com.safetyGame.back.condivisi.Login**, per ogni dubbio si faccia riferimento alla documentazione di quella classe

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata e utilizzerà le seguenti classi:

- **desktop.logic.DatiLogin**

Attributi

Tutti gli attributi di questa classe sono identici a quelli della classe contenuta in **com.safetyGame.back.condivisi.Login**, per ogni dubbio si faccia riferimento alla documentazione di quella classe

Metodi

Tutti i metodi di questa classe sono identici a quelli della classe contenuta in **com.safetyGame.back.condivisi.Login**, per ogni dubbio si faccia riferimento alla documentazione di quella classe

5 Specifica Front-end Web

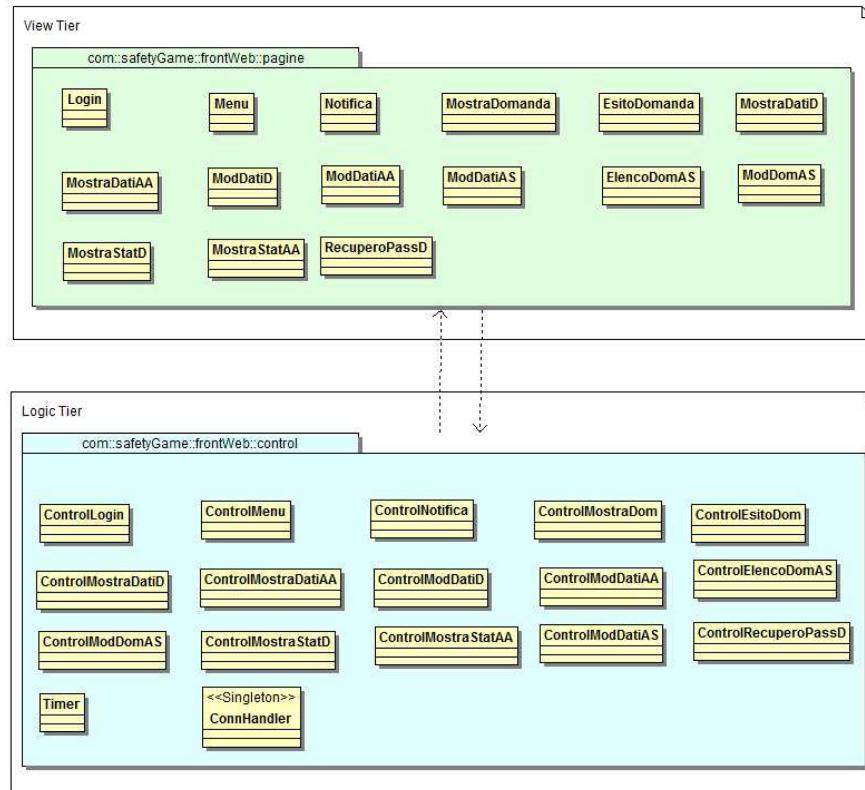


Figure 11: *Front-end Web, Diagramma dei Package*

Il package `com.safetyGame.frontWeb`, diversamente dagli altri, è stato sviluppato tramite tecnologia jsp. La forte caratterizzazione web-oriented di questa tecnologia ha impedito lo sviluppo dei vari componenti in modo standard, come avvenuto per l'intero back-end o gli altri front-end. Infatti, ad ogni componente non corrisponde una classe, nonostante sia possibile creare le proprie classi personali in Java per poi poterle utilizzare nella propria applicazione jsp. I vari componenti dei due package (Package `com.safetyGame.frontWeb.pagine` e Package `com.safetyGame.frontWeb.control`) sono stati implementati tramite pagine `.jsp`, le quali possono svolgere le funzionalità di uno o più componenti del package. Un esempio può essere il componente `Menu`; questo componente risulta essere unico nella rappresentazione dei package, tuttavia viene implementato in due pagine jsp differenti, a seconda che si tratti del menu relativo

a Dipendete o ad Amministratore. Viceversa, una stessa pagina jsp può concorrere all'implementazione di più di un componente del package. Un esempio sono i componenti MostraDatiD e MostraStatD, i cui compiti vengono assolti da una stessa pagina (`user_panel.jsp`).

5.1 Package com.safetyGame.frontWeb.pagine

Tipo, obiettivo e funzionamento del componente: Contiene le pagine `.jsp` che si occupano di fornire un'interfaccia utente. Queste pagine servono a dare all'utente gli strumenti per utilizzare il software e possono sia richiedere delle informazioni dall'utente per poi comunicarle al package `com.safetyGame.frontWeb.control`, sia ricevere informazioni dal back-end per poi mostrarle all'utente.

Relazione d'uso di altre componenti: Vengono utilizzate funzioni del package `com.safetyGame.connection`.

Interfaccie con e relazioni d'uso da altre componenti: In alcuni casi le informazioni vengono trasmesse ad altre pagine che fanno parte del package `com.safetyGame.frontWeb.control`

Attività svolte e dati trattati: Definisce l'aspetto dell'applicazione web definendone l'interfaccia con cui l'utente dovrà interagire.

5.1.1 com.safetyGame.frontWeb.pagine.Login

Funzione: Fornisce ad un Utente l'interfaccia grafica per poter effettuare il login.

File JSP che implementano il componente: `login.jsp`

5.1.2 com.safetyGame.frontWeb.pagine.Menu

Funzione: Fornisce ad un Utente l'interfaccia grafica per scegliere fra le possibili attività che può effettuare per interagire con il sistema Safety-Game.

File JSP che implementano il componente: `user_page.jsp`, `index.jsp`, `html/menu.html`,

5.1.3 com.safetyGame.frontWeb.pagine.Notifica

Funzione: Mostra ad un Dipendente autenticato la notifica di una nuova domanda disponibile a cui rispondere.

File JSP che implementano il componente: `js/countdown.js`

5.1.4 com.safetyGame.frontWeb.pagine.MostraDomanda

Funzione: Si occupa di far visualizzare al Dipendente la domanda appena richiesta.

File JSP che implementano il componente: `nuovaDomanda.jsp`

5.1.5 com.safetyGame.frontWeb.pagine.MostraDatiD

Funzione: Si occupa di far visualizzare le informazioni di un determinato dipendente.

File JSP che implementano il componente: user_page.jsp, gestioneDipendenti.jsp, visualizzaDatiD.jsp, visualizzaDipendente.jsp

5.1.6 com.safetyGame.frontWeb.pagine.MostraDatiAA

Funzione: Si occupa di far visualizzare i dati relativi ad un Amministratore Azienda.

File JSP che implementano il componente: visualizzaDatiA.jsp

5.1.7 com.safetyGame.frontWeb.pagine.MostraStatD

Funzione: Si occupa di far visualizzare le statistiche relative ad un Dipendente

File JSP che implementano il componente: user_page.jsp, visualizzaDatiD.jsp, visualizzaDipendente.jsp

5.1.8 com.safetyGame.frontWeb.pagine.ModDatiID

Funzione: Si occupa di fornire l'interfaccia per modificare i dati di un Dipendente e per inserire un nuovo Dipendente

File JSP che implementano il componente: aggiungiDipendente.jsp

5.1.9 com.safetyGame.frontWeb.pagine.MostraStatAA

Funzione: Si occupa di far visualizzare le statistiche di un'Azienda

File JSP che implementano il componente: al momento nessuna pagina implementa questo componente.

5.1.10 com.safetyGame.frontWeb.pagine.ModDatiAA

Funzione: Fornisce l'interfaccia per permettere di modificare i dati di un Amministratore Azienda

File JSP che implementano il componente: modPassA.jsp

5.1.11 com.safetyGame.frontWeb.pagine.RecuperoPass

Funzione: Fornisce all'Utente l'interfaccia per poter effettuare il recupero della propria password

File JSP che implementano il componente: recuperaPass.jsp

5.1.12 com.safetyGame.frontWeb.pagine.ModDatiAS

Funzione: Fornisce l'interfaccia per permettere di modificare i dati di un Amministratore Sicurezza

File JSP che implementano il componente: modPassA.jsp



5.1.13 com.safetyGame.frontWeb.pagine.ElencoDomAS

Funzione: Fornisce l'interfaccia ad un Amministratore Sicurezza per poter visualizzare le possibili domande importabili

File JSP che implementano il componente: aggiungiDomande.jsp

5.1.14 com.safetyGame.frontWeb.pagine.ModDomAS

Funzione: Fornisce l'interfaccia ad un Amministratore Sicurezza per poter selezionare le possibili domande importabili

File JSP che implementano il componente: aggiungiDomande.jsp, rimuoviDomande.jsp



5.2 Package com.safetyGame.frontWeb.control

Tipo, obiettivo e funzionamento del componente: Contiene le pagine .jsp che si occupano di scambiare dati fra l'Interfaccia utente (Package com.safetyGame.frontWeb.pagine) ed il back-end. Queste pagine non hanno alcun compito di far visualizzare informazioni agli utenti (se non brevi messaggi di conferma); tuttavia si occupano di creare una connessione con il back-end e di ottenere dati dall'utente verso il back-end e viceversa.

Interfaccie con e relazioni d'uso da altre componenti: i componenti di questo package possono sia ricevere informazioni (passate tramite metodo "post") dalle pagine appartenenti al package com.safetyGame.frontWeb.pagine sia inviare, richiedere e ricevere dati con il package com.safetyGame.back.connection.

Attività svolte e dati trattati: Implementa le funzionalità dell'applicazione web richiamando funzioni ed oggetti dal back-end. Spesso utilizza le classi del package com.safetyGame.back.condivisi per riuscire ad interpretare oggetti forniti dal back-end o per creare dati da inviare allo stesso.

5.2.1 com.safetyGame.frontWeb.control.ControlLogin

Funzione: Si occupa di gestire i dati di login inseriti dall'Utente e di richiedere al back-end la loro consistenza. Si occupa inoltre di inizializzare dei cookie per mantenere le informazioni riguardanti un Utente durante la navigazione.

File JSP che implementano il componente: checkLogin.jsp

5.2.2 com.safetyGame.frontWeb.control.ControlMostraDatiID

Funzione: Si occupa di richiedere i dati di un Dipendente al back-end e di fornirli al componente MostraDatiID

File JSP che implementano il componente: user_page.jsp

5.2.3 com.safetyGame.frontWeb.control.ControlModDomAS

Funzione: Si occupa di gestire la richiesta di rimozione e inserimento di nuove domande da parte di un Amministratore Sicurezza

File JSP che implementano il componente: checkAggiungiDomande.jsp

5.2.4 com.safetyGame.frontWeb.control.Timer

Funzione: Si occupa di mantenere il controllo del tempo per assegnare ad un Dipendente una nuova domanda.

File JSP che implementano il componente: js/countdown.js

5.2.5 com.safetyGame.frontWeb.control.ControlMenu

Funzione: Si occupa di riconoscere il tipo di Utente e di fornirgli il menu con le funzioni che gli compeono.

File JSP che implementano il componente: admin_page.jsp, user_page.jsp

5.2.6 com.safetyGame.frontWeb.control.ControlMostraStatD

Funzione: Si occupa di recuperare dal back-end le statistiche relative ad un dato utente.

File JSP che implementano il componente: user_page.jsp

5.2.7 com.safetyGame.frontWeb.control.ControlNotifica

Funzione: Si occupa di comunicare all'interfaccia utente la disponibilità di una nuova domanda comunicando periodicamente con il back-end

File JSP che implementano il componente: js/countdown.js

5.2.8 com.safetyGame.frontWeb.control.ControlModDatiD

Funzione: Si occupa di ricevere dall'interfaccia utente (quindi dall'utente) i dati relativi alla modifica o all'inserimento di un nuovo Dipendente

File JSP che implementano il componente: checkAggiungiDipendente.jsp

5.2.9 com.safetyGame.frontWeb.control.ControlMostraDom

Funzione: Si occupa di recuperare una nuova domanda per un Dipendente e di comunicarla all'interfaccia utente.

File JSP che implementano il componente: nuovaDomanda.jsp

5.2.10 com.safetyGame.frontWeb.control.ControlModDatiAA

Funzione: Si occupa di ricevere dall'interfaccia utente (quindi dall'utente) i dati relativi alla modifica di un Amministratore Azienda.

File JSP che implementano il componente: modPassA.jsp

5.2.11 com.safetyGame.frontWeb.control.ControlModDatiAS

Funzione: Si occupa di ricevere dall'interfaccia utente (quindi dall'utente) i dati relativi alla modifica di un Amministratore Sicurezza.

File JSP che implementano il componente: modPassA.jsp

5.2.12 com.safetyGame.frontWeb.control.ControlElencoDomAS

Funzione: Si occupa di recuperare l'elenco delle domande disponibili dal back-end e di comunicarlo all'interfaccia utente.

File JSP che implementano il componente: checkAggiungiDomande.jsp, aggiungiDomande.jsp, rimuoviDomande.jsp

5.2.13 com.safetyGame.frontWeb.control.ControlRecuperoPass

Funzione: SI occupa di inoltrare la richiesta ed i dati utente per il recupero password al back-end.

File JSP che implementano il componente: checkRecuperoPass.jsp

5.3 Descrizione di dettaglio delle pagine JSP

Questa sezione descrive nel dettaglio il funzionamento e le caratteristiche delle pagine jsp che implementano il package com.safetyGame.frontWeb.

5.3.1 addTrofeo.jsp

Questa pagina riceve tramite request method GET l'id del dipendente cui bisogna aggiungere un trofeo. Tramite id recupera il l'oggetto dipendente dal back-end, quindi ne aumenta il numero di Trofei di 1. E' raggiungibile ed utilizzabile esclusivamente da un Amministratore Azienda.

5.3.2 admin_page.jsp

Questa pagina si occupa di fornire le informazioni di base sull'account agli amministratori (siano essi Amministratori Azienda o Sicurezza). Tramite cookie riconosce la differenza fra Amministratore Aienda e Amministratore Sicurezza e offre loro due menù differenti.

5.3.3 aggiungiDomande.jsp

Questa pagina fornisce ad un Amministratore Sicurezza l'interfaccia per inserire nuove domande nel database aziendale. Dopo aver recuperato dal back-end l'elenco delle domande inseribili le inserisce in un form tabellare, dal quale l'utente potrà selezionare una o più domande da inserire. Infine, il form invia (tramite POST) alla pagina checkAggiungiDomande.jsp la lista degli id delle domande da inserire.

5.3.4 checkAggiungiDipendente.jsp

Questa pagina riceve (tramite POST) dalla pagina gestioneDipendenti.jsp una 5-upla di elementi (nome, cognome, codice fiscale, email e ruolo) necessari alla creazione di un nuovo dipendente. Dopo aver creato un nuovo oggetto Dipendente con tali dati, invia una richiesta di inserimento al back-end. Prima di inviare tale richiesta effettua un controllo (tramite regular expression) sulla validità dell'indirizzo email (il quale, se errato, genera errore nel componente Java-mail).

5.3.5 checkAggiungiDomande.jsp

Questa pagina, diversamente da quanto faccia intendere il nome, si occupa sia dell'inserimento che della rimozione delle domande dal database aziendale. Riceve (tramite POST) dalle pagine aggiungiDomande.jsp o rimuoviDomande.jsp l'elenco delle domande da spostare. Quindi, recupera tali domande tramite back-end e tramite la funzione isInternaAzienda riconosce se deve aggiungere o rimuovere domande dal database aziendale.

5.3.6 checkDelDipendente.jsp

Questa pagina riceve (tramite GET) dalla pagina eliminaDipendente.jsp l'id del dipendente da eliminare. Tramite id ottiene l'oggetto Dipendente da eliminare, quindi fa richiesta al back-end di rimuoverlo.

5.3.7 checkLogin.jsp

Questa pagina riceve (tramite POST) una tripla di valori (username, password e ambito) dalla pagina login.jsp. Con il valore Ambito riconosce se effettuare un login Amministratore o un login Dipendente, quindi richiama la funzione corretta del back-end passando i valori username e password. Il back-end comunica il successo o fallimento dell'operazione. In caso di successo la pagina salva questa tripla di valori in 3 cookie differenti, quindi reindirizza alla pagina index.jsp. In caso di fallimento segnale errore.

5.3.8 checkModDipendente.jsp

Questa pagina riceve (tramite POST) dalla pagina modificaDipendente.jsp una 6-upla di valori da assegnare il dipendente. Questi valori vengono quindi assegnati ad un oggetto Dipendente, il quale viene comunicato al back-end per sostituirlo all'oggetto Dipendente precedente.

5.3.9 checkModEmailD.jsp

Questa pagina riceve (tramite POST) dalla pagina modEmailD.jsp una coppia di valori (nuova email e password), quindi li passa al back-end per richiedere il cambio password. Prima della richiesta al back-end viene effettuato un controllo sulla validità dell'indirizzo email.

5.3.10 checkModPassA.jsp

Questa pagina riceve (tramite POST) dalla pagina modPassA.jsp una tripla di valori (nuova password, conferma nuova password e vecchia password) sui quali esegue una serie di controlli di validità (coerenza delle prime due password, correttezza della vecchia password). Se i controlli vanno a buon fine richiede al back-end il cambiamento di password per quell'Amministratore.

5.3.11 checkModPassD.jsp

Questa pagina riceve (tramite POST) dalla pagina modPassD.jsp una tripla di valori (nuova password, conferma nuova password e vecchia password) sui quali esegue una serie di controlli di validità (coerenza delle prime due password, correttezza della vecchia password). Se i controlli vanno a buon fine richiede al back-end il cambiamento di password per quel Dipendente.

5.3.12 checkRecuperoPass.jsp

Questa pagina riceve (tramite POST) dalla pagina recuperaPass.jsp una tripla di valori (email, codice fiscale e ambito). Utilizza il valore di ambito per differenziare il recupero password da un Amministratore ad un Dipendente. Dopo aver individuato la funzione corretta la richiama passando i parametri email e codice fiscale.

5.3.13 checkRisposta.jsp

Questa pagina riceve (tramite POST) dalla pagina nuovaDomanda.jsp il valore della risposta ottenuta per poi comunicarlo al back-end.

5.3.14 eliminaDipendente.jsp

Questa pagina riceve (tramite GET) dalla pagina gestioneDipendenti l'id del dipendente da rimuovere. Recupera dal back-end le informazioni relative a quel dipendente, quindi le mostra all'utente per chiedere conferma sull'eliminazione. In caso di conferma richiama la pagina checkDelDipendente.jsp, passando come parametro GET proprio l'id che lei stessa ha ricevuto.

5.3.15 forzaCambioPass.jsp

Questa pagina contiene solo poche righe di codice e non viene mai utilizzata da sola, ma sempre inclusa in altre pagine. Recupera dal back-end le informazioni relative all'utente che vi accede; nel caso in cui l'utente abbia effettuato l'accesso con la password fornita (tramite email) dal sistema impedisce l'accesso a tutte le pagine che non siano la pagina per il cambio password.

Specifica per i Dipendenti.

5.3.16 forzaCambioPassA.jsp

Ha le stesse funzioni della pagina precedente, ma specifiche per gli amministratori.

5.3.17 gestioneDipendenti.jsp

Questa pagina fornisce all'Amministratore Azienda l'interfaccia per visualizzare, modificare, eliminare o aggiungere un nuovo dipendente.

Un form tabellare permette di visualizzare l'elenco dei dipendenti, quindi accedere alla loro visualizzazione, modifica o rimozione (richiamando, rispettivamente, le pagine visualizzaDipendente.jsp, modificaDipendente.jsp e eliminaDipendente.jsp) passando l'id del dipendente tramite GET.

Un form classico, invece, permette di raccogliere le informazioni relative ad un nuovo dipendente ed inviarle (tramite POST) alla pagina aggiungiDipendente.jsp

5.3.18 getCookies.jsp

Questa pagina non viene mai eseguita autonomamente, ma viene sempre inclusa in altre pagine. Serve a recuperare i cookie salvati ed inserirli in variabili differenti. In caso di cookie mancanti effettua un redirect alla pagina login.jsp.

5.3.19 index.jsp

Questa pagina non comunica con il backend. Si occupa esclusivamente di analizzare i cookie e, tramite essi, riconoscere che tipo di utente sia collegato, quindi effettua un redirect alla user_page.jsp per i Dipendente, admin_page.jsp per gli amministratori e login.jsp in caso in cui non sia stato effettuato alcun login.

5.3.20 login.jsp

Fornisce l'interfaccia e il form per poter effettuare il login. Passa (tramite POST) una tripla di parametri (username, password e ambito) alla pagina checkLogin.jsp.

5.3.21 logout.jsp

Questa pagina dealloca tutti i possibili cookie creati dal resto dell'applicazione. Infine, a seconda che l'utente che effettua il logout sia un Amministratore o un Dipendente, richiama la funzione logoutD() o logoutA() del back-end per permettere la gestione dei log.

5.3.22 modEmailD.jsp

Questa pagina fornisce l'interfaccia per la modifica dell'email di un Dipendente. Richiede una coppia di valori (nuova email e passworrd) che passa (tramite POST) alla pagina checkModEmailD.

5.3.23 modificaDipendente.jsp

Questa pagina fornisce l'interfaccia per permettere ad un Amministratore Azienda di modificare i dati di un certo dipendente. Tramite GET ottiene l'ide dell'utente da modificare. Dopodichè recupera le informazioni dal back-end e le inserisce in un form (come valori di default). Tale form comunicherà tali valori (siano essi modificati o meno) alla pagina checkModDipendente.jsp

5.3.24 modPassA.jsp

Questa pagina fornisce l'interfaccia per la modifica della password da parte di un Amministratore. Questi dovrà compilare un form, il quale (tramite POST) invierà una tripla di valori (nuova password, conferma nuova password e vecchia password) alla pagina checkModPassA.jsp.

5.3.25 modPassD.jsp

Questa pagina fornisce l'interfaccia per la modifica della password da parte di un Dipendente. Questi dovrà compilare un form, il quale (tramite POST) invierà una tripla di valori (nuova password, conferma nuova password e vecchia password) alla pagina checkModPassD.jsp.

5.3.26 nuovaDomanda.jsp

Questa pagina si occupa di richiedere al back-end una nuova domanda. Ogni Dipendente può scegliere quando richiedere una nuova domanda. Nel caso in cui non sia passato abbastanza tempo dall'ultima domanda la pagina mostra un messaggio di errore. Diversamente, il back-end comunicherà la successiva domanda da somministrare al Dipendente. Nel caso in cui non siano disponibili ulteriori domande verrà visualizzato un messaggio di errore.

5.3.27 recuperaPass.jsp

Questa pagina è l'unica (assieme a login.jsp) raggiungibile da un utente non autenticato. Fornisce il form per l'inserimento dei dati necessari al recupero password (la tripla di valori indirizzo email, codice fiscale e ambito) che vengono poi inviati (tramite POST) alla pagina checkRecuperoPass.jsp

5.3.28 rimuoviDomande.jsp

Questa pagina fornisce ad un Amministratore Sicurezza l'interfaccia per rimuovere domande presenti nel database aziendale. Dopo aver recuperato dal back-end l'elenco delle domande presenti le inserisce in un form tabellare, dal quale l'utente potrà selezionare una o più domande da rimuovere. Infine, il form invia (tramite POST) alla pagina checkAggiungiDomande.jsp la lista degli id delle domande da rimuovere.

5.3.29 subTrofeo.jsp

Questa pagina riceve tramite request method GET l'id del dipendente cui bisogna rimuovere un trofeo. Tramite id recupera il l'oggetto dipendente dal back-end, quindi ne riduce il numero di Trofei di 1. E' raggiungibile ed utilizzabile esclusivamente da un Amministratore Azienda.

5.3.30 user_page.jsp

Questa pagina si occupa di fornire le informazioni di base sull'account ai Dipendenti e le statistiche personali (punteggio, Trofei e Badge).

5.3.31 visualizzaDatiA.jsp

Questa pagina si occupa di recuperare dal back-end le informazioni relative all'Amministratore che ha effettuato l'accesso e di mostrarle a video.



5.3.32 visualizzaDatiD.jsp

Questa pagina si occupa di recuperare dal back-end le informazioni relative al Dipendente che ha effettuato l'accesso e di mostrarle a video. A fondo pagina è disponibile un piccolo menù, il quale sarà utilizzato dal dipendente per accedere alle pagine di modifica email e password.

5.3.33 visualizzaDipendente.jsp

Questa pagina si occupa di mostrare ad un Amministratore Azienda le informazioni relative ad un certo Dipendente. Tramite GET riceve l'id del Dipendente interessato, ne recupera i dati dal back-end e li mostra a video. Infine permette (con un piccolo menù accanto la visualizzazione dei trofei) di aggiungere o rimuovere trofei al dipendente.

6 Specifica Front-end Mobile

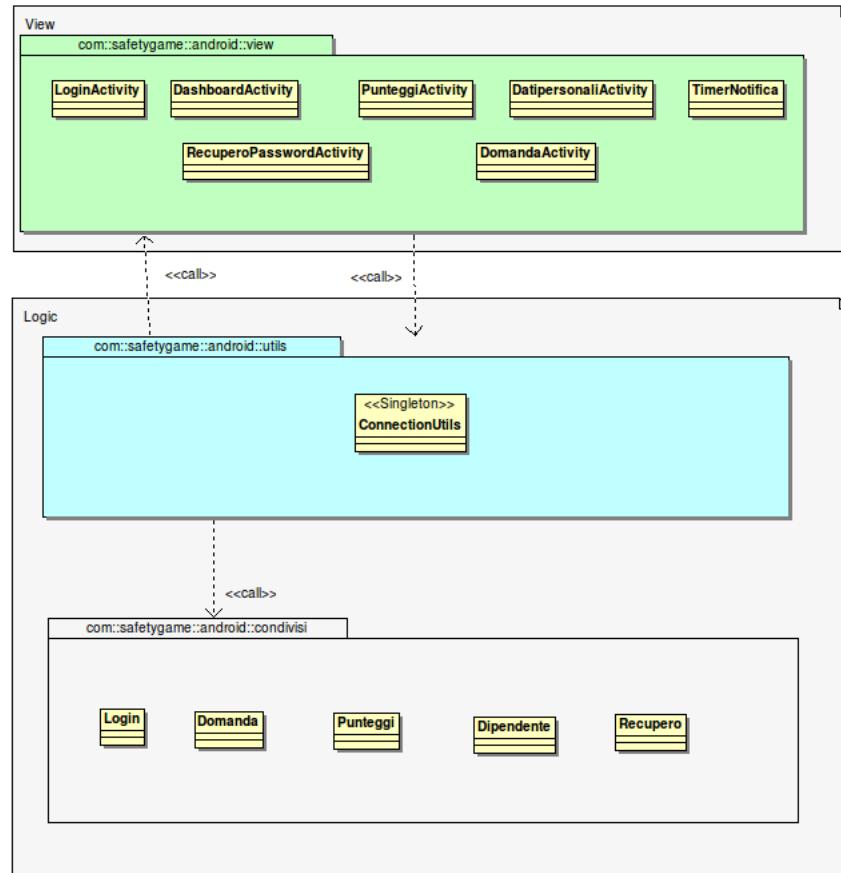


Figura 12: *Front-end Mobile, diagramma dei package*

6.1 Package com.safetyGame.mobile.View

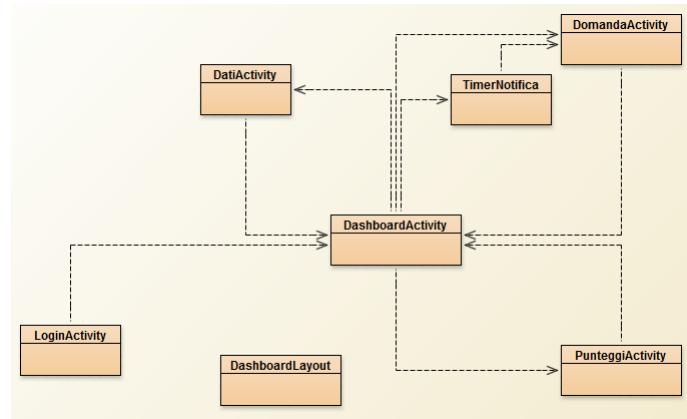


Figura 13: *Package com.safetyGame.mobile.View*

Tipo, obiettivo e funzionamento del componente: Contiene le classi Java che estendono quelle Android “Activity” e che quindi sono inerenti la grafica dell’applicazione. La classe TimerNotifica è parte di questo componente nonostante estenda la classe Java Android “Service”.

Relazione d’uso di altre componenti: Vengono utilizzate funzioni del package Utils.

Interfacce con e relazioni d’uso da altre componenti: Nel momento dell’arrivo dei dati Utils notifica View.

Attività svolte e dati trattati: Definisce l’aspetto dell’applicazione Android definendone l’interfaccia con cui l’utente dovrà interagire.

6.1.1 com.safetyGame.mobile.View.DashboardActivity

DashboardActivity	
-	context : Context
+	onCreate(savedInstanceState : bundle) : void

Funzione

Permette di raggiungere le altre Activity del programma. Dovrà intercettare gli input dell'utente e lanciare le altre Activity.

Relazioni d'uso con altri moduli

Questa classe utilizzerà le seguenti classi:

- mobile.View.DomandaActivity
- mobile.View.DatiActivity
- mobile.View.PunteggiActivity

Attributi:

- Context context

Metodi

+ void onCreate(Bundle savedInstanceState)

Metodo chiamato alla creazione della classe.

Dovrà prendere il controllo della grafica e disegnare al suo interno i pulsanti per accedere alle seguenti funzionalità:

- Risposta domanda
- Esecuzione quest
- Visualizzazione propri punteggi
- Visualizzazione propri dati

Quindi dovranno essere ridefiniti le azioni di pressione dei pulsanti affinché creino gli oggetti deputati all'azione richiesta.

6.1.2 com.safetyGame.mobile.View.DashboardLayout

DashboardLayout
- <u>UNEVEN_GRID_PENALTY_MULTIPLIER : int = 10</u>
- mMaxChildWidth : int = 0
- mMaxChildHeight : int = 0
onMeasure(in widthMeasureSpec : int, in heightMeasureSpec : int) : void
+ onLayout(in changed : boolean, in l : int, in t : int, in r : int, in b : int) : void

Funzione

Classe che aiuta la visualizzazione della Dashboard all'avvio dell'applicazione.

Relazioni d'uso con altri moduli

Questa classe utilizzerà le seguenti classi:

- mobile.View.PunteggiActivity
- mobile.View.DomandaActivity
- mobile.View.DatiActivity

Questa classe verrà utilizzata dalle seguenti classi:

- mobile.View.LoginActivity

Attributi

- static final int UNEVEN_GRID_PENALTY_MULTIPLIER = 10
- int mMaxChildWidth = 0
- int mMaxChildHeight = 0

Metodi

+ DashboardLayout(Context context)

Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

+ DashboardLayout(Context context, AttributeSet attrs)

Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

+ DashboardLayout(Context context, AttributeSet attrs, int defStyle)

Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

void onMeasure(int widthMeasureSpec, int heightMeasureSpec)

Metodo utilizzato per calcolare la dimensione ideale che la grafica dovrebbe avere a seconda del suo contenuto.

Deve controllare le dimensioni del widget interno più grande, le ricontrolla per essere sicuro che siano quelle, quindi imposta gli attributi *nMaxChildWidth* e *mMaxChildHeight* con quella dimensione.

void onMeasure(int widthMeasureSpec, int heightMeasureSpec)

Metodo per determinare la grandezza ideale del layout che dovrebbe impiegare⁵.

Dovrà provare calcolando ogni possibile combinazione di righe e colonne, per poi decidere in base al rapporto fra la dimensione degli elementi presenti e lo spazio disponibile rimasto quale sia il miglior rapporto.

⁵Layout calcolato in colonne e righe, come in una tabella

6.1.3 com.safetyGame.mobile.View.DatiActivity

DatiActivity	
-	context : Context
+	onCreate(in savedInstanceState : bundle) : void
+	onOptionsItemSelected(in item : MenuItem) : boolean

Funzione

Permette ad un Dipendente Autenticato di visionare e modificare i dati personali personali.

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- mobile.View.DashboardActivity

Inoltre utilizzerà le seguenti classi:

- mobile.Utils.ConnectionsUtils

Attributi

- Context context

Metodi:

- + void **onCreate(Bundle savedInstanceState)**

Metodo invocato alla creazione dell'oggetto *DatiActivity*.

Dovrà creare un'istanza della classe *DatiTask* e quindi mandarla in esecuzione

- + boolean **onOptionsItemSelected(MenuItem item)**

Metodo invocato quando si preme un pulsante del menù.

Dovrà controllare quale pulsante è stato premuto tramite il suo id, quindi eseguire l'azione associata.



6.1.3.1 com.safetyGame.mobile.View.DatiActivity.DatiTask

Funzione

Consente ad un Dipendente di effettuare il login di un Dipendente, recuperando i dati immessi dal Dipendente e inviandoli al Back-End per la verifica

Relazioni d'uso con altri moduli

Questa classe utilizzerà le seguenti classi:

- mobile.Utils.ConnectionUtils
- mobile.View.DatiActivity

Inoltre verrà utilizzata dalle seguenti classi:

- mobile.View.DatiActivity

Attributi

```
# ProgressDialog dialog  
  
# EditText user  
  
# EditText passw
```

Metodi

void onPreExecute()

Metodo che viene eseguito prima di mostrare la grafica.

Dovrà mostrare all'utente una *ProgressDialog* con scritto sotto “Loading. Please wait...” fino a quando non è stata creata la vista dei campi dati.

String doInBackground(Object... params).

Metodo che crea la grafica deputata a raccogliere le credenziali d'accesso al sistema.

Dovrà inserire all'interno dell'interfaccia grafica i due campi dove poter inserire username e password, scrivendone al loro interno l'indicazione di cosa dovrà essere inserito. Quindi, una volta inseriti, dovrà controllare che i campi siano stati compilati; una volta compilati, dovrà interrogare il package mobile.Utils affinché comunichi con il Back-End per controllare la correttezza dei dati inseriti.

void onPostExecute(String status)

Metodo che si occupa di segnalare all'utente la correttezza o meno del login, mostrando a schermo un messaggio d'errore o facendo sì che il gestore della grafica mostri tutte le opzioni che erano nascoste all'utente che non abbia fatto accesso all'applicazione.

Dovrà rimuovere dalla grafica il pannello creato con la funzione doInBackground(...) e controllare che la stringa passata come parametro non sia vuota e che contenga “OK”. Se una di queste due condizioni non dovesse essere verificata, dovrà far comparire un messaggio d’errore all’utente. Altrimenti dovrà notificare al gestore della grafica che il Dipendente è stato identificato, quindi mostrare al Dipendente le opzioni che prima non erano visibili.

6.1.3.2 com.safetyGame.mobile.View.DatiActivity.InviaDatiTask

Funzione

Consente ad un Dipendente di effettuare il cambio password

Relazioni d’uso con altri moduli

Questa classe utilizzerà le seguenti classi:

- mobile.Utils.ConnectionUtils
- mobile.View.DatiActivity

Inoltre verrà utilizzata dalle seguenti classi:

- mobile.View.DatiActivity

Attributi

```
# ProgressDialog dialog
```

Metodi

```
# void onPreExecute()
```

Metodo che viene eseguito prima di mostrare la grafica.

Dovrà mostrare all’utente una *ProgressDialog* con scritto sotto “Loading. Please wait...” fino a quando non è stata creata la vista dei campi dati.

```
# String doInBackground(Object... params).
```

Metodo che controlla che la nuova password e la conferma della nuova password siano uguali e che invia la richiesta al server.

```
# void onPostExecute(String status)
```

Metodo che si occupa di segnalare all’utente la correttezza o meno della richiesta

6.1.4 com.safetyGame.mobile.View.DomandaActivity

DomandaActivity	
-	context : Context
+	onCreate(in savedInstanceState : bundle) : void
+	onActivityResult(in requestCode : int, in resultCode : int, in intent : Intent) : void
+	onOptionsItemSelected(in item : MenuItem) : boolean

Funzione

Mette a disposizione del Dipendente Autenticato la possibilità di visualizzare, richiedere e rispondere alle domande.

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

- mobile.View.DashboardActivity
- mobile.Utils.TimerNotifica

Attributi:

- Context context

Metodi:

- + void **onCreate(Bundle savedInstanceState)**

Metodo chiamato alla creazione della classe.

Dovrà controllare la tipologia di azione richiesta (Domanda o Quest) e chiamare la sottoclassificazione corrispondente

- + boolean **onOptionsItemSelected(MenuItem item)**

Metodo invocato quando si preme un pulsante del menù.

Dovrà controllare quale pulsante è stato premuto tramite il suo id, quindi eseguire l'azione associata.

- void **TimerMethod()**

Metodo che conta quanti secondi l'utente ci mette a rispondere nel caso di una domanda di tipo isMobile()



6.1.4.1 com.safetyGame.mobile.View.DomandaActivity.DomandaTask

Funzione

Consente ad un Dipendente di rispondere ad una Domanda

Relazioni d'uso con altri moduli

Questa classe utilizzerà le seguenti classi:

- mobile.Utils.ConnectionUtils

Attributi

```
# ProgressDialog dialog  
# EditText user  
# EditText passw
```

Metodi

void onPreExecute()

Metodo che viene eseguito prima di mostrare la grafica.

Dovrà mostrare all'utente una *ProgressDialog* con scritto sotto “Loading. Please wait...” fino a quando non è stata creata la vista dei campi dati.

String doInBackground(Object... params).

Metodo che recupera dal Back-End la domanda che dovrà essere sottoposta al Dipendente.

void onPostExecute(String status)

Metodo che crea la grafica deputata a mostrare la domanda e a ricevere la risposta che il Dipendente ha dato alla suddetta domanda

Dovrà inserire all'interno dell'interfaccia grafica il testo della domanda e le possibili risposte con a fianco un pulsante che permetterà di selezionarne solo una alla volta. Inoltre dovrà essere presente un pulsante che permetterà di inviare la risposta al Back-End per verificare che questa sia corretta

6.1.5 com.safetyGame.mobile.View.LoginActivity

LoginActivity	
-	context : Context
+	onCreate(savedInstanceState : bundle) : void

Funzione

Classe di gestione della parte grafica dell'attività di login, gestendo l'ingresso alla pagina che permette l'inserimento delle credenziali d'accesso e contenendo la classe deputata alla visualizzazione dei campi dati per l'immissione

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata da:

- mobile.Utils.ConnectionUtils

Questa classe utilizzerà le seguenti classi:

- mobile.View.DomandaActivity.LoginTask (*classe privata interna alla classe LoginActivity*)
- mobile.Utils.ConnectionUtils
- mobile.View.DashboardActivity
- mobile.View.RecuperoPasswordActivity
- mobile.Condivisi.Login

Attributi

- Context context

Metodi

- + void **onCreate(Bundle savedInstanceState)**

Metodo per la creazione dell'oggetto LoginActivity. Questo oggetto verrà creato all'avvio dell'applicazione.

Dovrà creare un pulsante che permetta l'accesso alla funzionalità di login, collegando l'azione di pressione del pulsante alla creazione di un oggetto *LoginTask* che permetterà l'inserimento delle credenziali d'accesso al sistema. Il pulsante così creato dovrà poi essere inserito all'interno della grafica dell'applicazione.



- **boolean checkServer()**

Metodo che serve per vedere se l'app è stata lanciata la prima volta

6.1.5.1 com.safetyGame.mobile.View.LoginActivity.LoginTask

Funzione

Consente ad un Dipendente di effettuare il login di un Dipendente, recuperando i dati immessi dal Dipendente e inviandoli al Back-End per la verifica

Relazioni d'uso con altri moduli

La classe verrà utilizzata dalle seguenti classi:

- mobile.Utils LoginActivity

Questa classe utilizzerà le seguenti classi:

- mobile.Utils ConnectionUtils

Attributi

```
# ProgressDialog dialog  
  
# EditText user  
  
# EditText passw
```

Metodi

void onPreExecute()

Metodo che viene eseguito prima di mostrare la grafica.

Dovrà mostrare all'utente una *ProgressDialog* con scritto sotto “Loading. Please wait...” fino a quando non è stata creata la vista dei campi dati.

String doInBackground(Object... params).

Metodo che crea la grafica deputata a raccogliere le credenziali d'accesso al sistema.

Dovrà inserire all'interno dell'interfaccia grafica i due campi dove poter inserire username e password, scrivendone al loro interno l'indicazione di cosa dovrà essere inserito. Quindi, una volta inseriti, dovrà controllare che i campi siano stati compilati; una volta compilati, dovrà interrogare il package mobile.Utils affinché comunichi con il Back-End per controllare la correttezza dei dati inseriti.

void onPostExecute(String status)

Metodo che si occupa di segnalare all'utente la correttezza o meno del login, mostrando a schermo un messaggio d'errore o facendo sì che il gestore della grafica mostri tutte le opzioni che erano nascoste all'utente che non abbia fatto accesso all'applicazione.

Dovrà rimuovere dalla grafica il pannello creato con la funzione doInBackground(...) e controllare che la stringa passata come parametro non sia vuota



e che contenga “OK”. Se una di queste due condizioni non dovesse essere verificata, dovrà far comparire un messaggio d’errore all’utente. Altrimenti dovrà notificare al gestore della grafica che il Dipendente è stato identificato, quindi mostrare al Dipendente le opzioni che prima non erano visibili.

6.1.6 com.safetyGame.mobile.View.PunteggiActivity

PunteggioActivity	
-	context : Context
+	onCreate(in savedInstanceState : Bundle) : void
+	onOptionsItemSelected(in item : MenuItem) : boolean

Funzione

Permette ad un Dipendente Autenticato di visualizzare i vari punteggi e trofei.

Relazioni d'uso con altri moduli

Questa classe utilizzerà le seguenti classi:

- mobile.Utils.ConnectionUtils
- mobile.condivisi.Punteggio

Questa classe verrà utilizzata dalle seguenti classi:

- mobile.View.Dashboard
- mobile.Utils.ConnectionUtils

Attributi

- Context context

Metodi:

+ void **onCreate(Bundle savedInstanceState)**

Metodo invocato alla selezione dell'azione di visualizzazione dei punteggi.
Dovrà creare un oggetto anonimo di tipo *PunteggiTask* ed eseguirlo.

+ boolean **onOptionsItemSelected(MenuItem item)**

Metodo invocato quando si preme un pulsante del menù.
Dovrà controllare quale pulsante è stato premuto tramite il suo id, quindi eseguire l'azione associata.

6.1.6.1 com.safetyGame.mobile.View.PunteggiActivity.PunteggiTask

Funzione

Permette ad un Dipendente Autenticato di visualizzare i vari punteggi e trofei.

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

Attributi

- Context context

Metodi:

void onCreate(Bundle savedInstanceState)

Metodo che viene eseguito prima di mostrare la grafica.

Dovrà mostrare all'utente una *ProgressDialog* con scritto sotto "Loading. Please wait..." fino a quando non è stata creata la vista dei campi dati.

Punteggi doInBackground(Object... params)

Metodo che interroga il Back-End per recuperare i punteggi del Dipendente.

Dovrà mandare una richiesta al Back-End interrogando la pagina deputata a restituire un oggetto di tipo *Punteggi* e restituirlo al chiamante

void onPostExecute(Punteggi punteggi)

Metodo per la visualizzazione dei punteggi.

Dovrà far visualizzare le statistiche del Dipendente mostrando:

- Risposte date
- Risposte corrette
- Risposte errate
- Punteggio ottenuto

Nel caso ci siano problemi, dovrà far visualizzare un messaggio d'errore.

6.1.7 com.safetyGame.mobile.View.TimerNotifica

TimerNotifica	
-	mStartTime : long
-	mHandler : Handler = new Handler()
-	mNotificationManager : NotificationManager
-	notification : Notification
-	prefs : Sharedpreferences
-	mUpdateTimeTask : Runnable = new Runnable()
+	onStartCommand(in intent : Intent, in flags : int, in startId : int) : int
+	onBind(in arg0 : Intent) : IBinder

Funzione

Servizio di temporizzazione che propone al Dipendente Autenticato una domanda secondo quanto impostato tramite le notifiche standard di Android.

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

Attributi:

- long mStartTime
- Handler mHandler = new Handler()
- NotificationManager mNotificationManager
- Notification notification
- SharedPreferences prefs

Metodi:

- + int onStartCommand(Intent intent, int flags, int startId)
Metodo lanciato dal sistema ad ogni accensione dell'applicazione e permette di poter eseguire il controllo sulle notifiche.

Dovrà creare il set di istruzioni che il sistema dovrà eseguire ogni volta che deve notificare ad un Dipendente la presenza di una nuova domanda

- + IBinder onBind(Intent arg0)



6.1.7.1 com.safetyGame.mobile.View.TimerNotifica.mUpdateTimeTask

Funzione

Questa classe si occuperà di segnalare al sistema operativo che deve segnalare al Dipendente la presenza di una nuova domanda

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

Attributi

Metodi

+ void run()

Metodo che controlla la presenza di nuove domande dopo aver atteso un numero preciso di minuti.

6.2 Package com.safetyGame.mobile.Utils

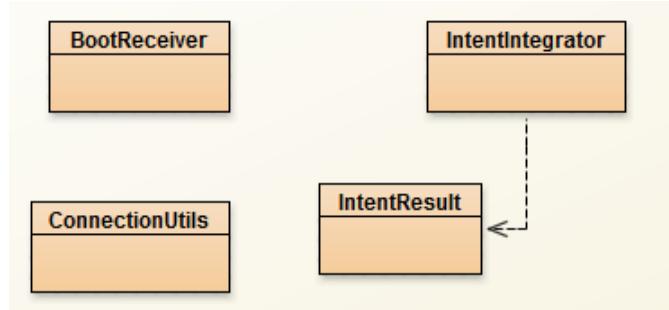


Figura 14: *Package com.safetyGame.mobile.Utils*

Tipo, obiettivo e funzionamento del componente: Vi appartengono le classi che comunicano con le API del server. Queste classi ricevono i dati e notificano la View.

Relazione d'uso di altre componenti: Comunica con il Back-End e quando i dati sono pronti invia notifiche al componente View.

Interfacce con e relazioni d'uso da altre componenti: Viene utilizzato dalla View.

Attività svolte e dati trattati: Invia richieste HTTP al server, il quale gli risponderà inviando i dati richiesti attraverso un XML. In seguito ne estrarrà i dati e li renderà disponibili alla View.

6.2.1 com.safetyGame.mobile.Utils.BootReceiver

BootReceiver
+ onReceive(in context : context, in arg1 : intent) : void

Funzione

Classe che estende BroadcastReceiver e viene automaticamente chiamata al completamento del boot del dispositivo.

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

Attributi

Metodi

+ onReceive(Context context, Intent arg1)

Metodo chiamato all'inizializzazione della classe, dovrà connettere al contatore interno al sistema operativo il timer delle notifiche.

6.2.2 com.safetyGame.mobile.Utils.ConnectionUtils

ConnectionUtils
- <code>rootXML(in response : HttpResponse) : Element</code>
+ <code>HttpCreateClient(in url : string, in nameValuePairs : list<NameValuePair>) : Object</code>
+ <code>parseXML(in root : Element, in stringa : string, in position : int) : string</code>

Funzione

La classe è utilizzata per effettuare connessioni fisiche tra il front-end Mobile ed il Back-End, ricevere dati, effettuare parser e creare gli opportuni oggetti da notificare alla View.

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

Attributi

Metodi:

+ static Object `HttpCreateClient(String url, List<NameValuePair> nameValuePairs)`

Metodo che serve a chiamare le funzioni del Back-End tramite connessione Http.

Dovrà creare un oggetto che permetta la connessione http, quindi mandare una richiesta POST all'*url* indicato. Nel caso non ci siano problemi, dovrà controllare l'indirizzo che ha contattato e, in base a questo, creare un oggetto che verrà passato al chiamante.

- static Element `rootXML(HttpResponse response)`

Restituirà il nodo radice di un file XML ricevuto tramite Http.

Dovrà estrarre la radice del file XML e poi restituirla alla funzione chiamante.

+ static String `parseXML(Element root, String stringa, int position)`

Metodo che restituirà il contenuto del *position* nodo di categoria stringa appartenente all'albero identificato da *root*



6.2.3 com.safetyGame.mobile.Utils.IntentIntegrator

```
IntentIntegrator
+ TAG : string = IntentIntegrator.class.getSimpleName()
+ BS_PACKAGE : string = "com.google.zxing.client.android"
+ REQUEST_CODE : int = 0
+ DEFAULT_TITLE : string = "Install Barcode Scanner?"
+ DEFAULT_MESSAGE : string = "This application requires Barcode Scanner. Would you like to install it?"
+ DEFAULT_YES : string = "Yes"
+ DEFAULT_NO : string = "No"
+ PRODUCT_CODE_TYPES : Collection<String> = list("UPC_A", "UPC_E", "EAN_8", "EAN_13", "RSS_14")
+ ONE_D_CODE_TYPES : Collection<String> = list("UPC_A", "UPC_E", "EAN_8", "EAN_13", "CODE_39", "CODE_93", "CODE_128", "ITF", "RSS_14", "RSS_EXPANDED")
+ QR_CODE_TYPES : Collection<String> = Collections.emptyList()
+ DATA_MATRIX_TYPES : Collection<String> = Collections.singleton("DATA_MATRIX")
+ ALL_CODE_TYPES : Collection<String> = null
+ TARGET_ALL_APPLICATIONS : Collection<String> = Collections.singletonList("com.safetygame.safetygame")
+ TARGET_ALL_KNOWN : Collection<String> = list(BS_PACKAGE, "com.safetygame.safetygame", "com.safetygame.safetygame.simple")
+ title : string
+ message : string
+ buttonYes : string
+ buttonNo : string
+ targetApplications : Collection<String>
+ setTitle : void
+ setTitleByDinTitle : int : void
+ getButtonYes : string
+ getMessage : string
+ setMessage(message : String) : void
+ setMessageByDin(messageId : int) : void
+ getButtonYesIs : string
+ setButtonYesIsYes : string
+ setButtonYesIsNo : string
+ getButtonNo : string
+ setButtonNoIs : string
+ setButtonNoIsNo : string
+ setButtonNoIsYes : string
+ getTargetApplications() : Collection<String>
+ setTargetApplications(targetApplications : Collection<String>) : void
+ startActivityForResult(intent : Intent, requestCode : int) : IntentResult
+ initiateScan() : AlertDialog
# static final Collection<String> PRODUCT_CODE_TYPES = Collections.emptyList()
- onActivityResult(intent : Intent, requestCode : int, resultCode : int) : void
- findTargetAppPackage(intent : Intent) : string
- showDownloadDialog() : AlertDialog
- parseActivityResult(requestCode : int, resultCode : int, intent : Intent) : IntentResult
- shareActivityResult(resultCode : int, intent : Intent)
- isIntentValid(string... : Collection<String>)
```

Funzione

Classe che aiuta l'integrazione dell'applicazione Safety Game con l'applicazione Barcode Scanner. Poiché è una libreria esterna, i metodi non sono specificati, ma vengono solo elencati.

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

Attributi

```
+ static final int REQUEST_CODE = 0x0000c0de
- static final String TAG = IntentIntegrator.class.getSimpleName()
+ static final String DEFAULT_TITLE = "Install Barcode Scanner?"
+ static final String DEFAULT_MESSAGE = "This application requires Barcode Scanner. Would you like to install it?"
+ static final String DEFAULT_YES = "Yes"
+ static final String DEFAULT_NO = "No"
- static final String BS_PACKAGE = "com.google.zxing.client.android"
+ static final Collection<String> PRODUCT_CODE_TYPES = list("UPC_A", "UPC_E", "EAN_8", "EAN_13", "RSS_14")
```

```
+ static final Collection<String> ONE_D_CODE_TYPES = li-  
st("UPC_A", "UPC_E", "EAN_8", "EAN_13", "CODE_39", "CO-  
DE_93", "CODE_128", "ITF", "RSS_14", "RSS_EXPANDED")  
  
+ static final Collection<String> QR_CODE_TYPES = Collec-  
tions.singleton("QR_CODE");  
  
+ static final Collection<String> DATA_MATRIX_TYPES = Col-  
lections.singleton("DATA_MATRIX")  
  
+ static final Collection<String> ALL_CODE_TYPES = null;  
  
+ static final Collection<String> TARGET_BARCODE_SCANNER_ONLY  
= Collections.singleton(BS_PACKAGE)  
+ static final Collection<String> TARGET_ALL_KNOWN = list(  
BS_PACKAGE, "com.srowen.bs.android", "com.srowen.bs.android.simple")  
  
- final Activity activity  
  
- String title  
  
- String message  
  
- String buttonYes  
  
- String buttonNo  
  
- Collection<String> targetApplications
```

Metodi

```
+ IntentIntegrator(Activity activity)  
  
+ AlertDialog initiateScan()  
  
+ AlertDialog initiateScan(Collection<String> desiredBarcodeFor-  
mats)  
  
# void startActivityForResult(Intent intent, int code)  
  
- AlertDialog showDownloadDialog()  
  
- String findTargetAppPackage(Intent intent)  
  
+ static IntentResult parseActivityResult(int requestCode, int re-
```

```
sultCode, Intent intent)

+ void shareText(CharSequence text)

- private static Collection<String> list(String... values)

+ String getTitle()

+ void setTitle(String title)

+ void setTitleByID(int titleID)

+ String getMessage()

+ void setMessage(String message)

+ void setMessageByID(int messageID)

+ String getButtonYes()

+ void setButtonYes(String buttonYes)

+ void setButtonYesByID(int buttonYesID)

+ String getButtonNo()

+ void setButtonNo(String buttonNo)

+ void setButtonNoByID(int buttonNoID)

+ Collection<String> getTargetApplications(Collection<String> targetApplications)

+ void setTargetApplications(Collection<String> targetApplications)

+ void setSingleTargetApplication(String targetApplication)
```

6.2.4 com.safetyGame.mobile.Utils.IntentResult

IntentResult	
-	contents : string
-	formatName : string
-	orientation : Integer
-	rawBytes : byte[]
-	errorCorrectionLevel : string
+	getContents() : string
+	getFormatName() : string
+	getRawBytes() : byte[]
+	getOrientation() : Integer
+	getErrorCorrectionLevel() : string
+	toString() : string

Funzione

Classe che aiuta a gestire i risultati dello scan del barcode. Poiché è una libreria esterna, i metodi non sono specificati, ma vengono solo elencati.

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

Attributi

- final String contents
- final String formatName
- final byte[] rawBytes
- final Integer orientation
- final String errorCorrectionLevel

Metodi

- + IntentResult(String contents, String formatName, byte[] rawBytes, Integer orientation, String errorCorrectionLevel)



- + **String getContents()**
- + **String getFormatName()**
- + **byte[] getRawBytes()**
- + **Integer getOrientation()**
- + **String getErrorCorrectionLevel()**
- + **String toString()**

6.3 Package com.safetyGame.mobile.condivisi

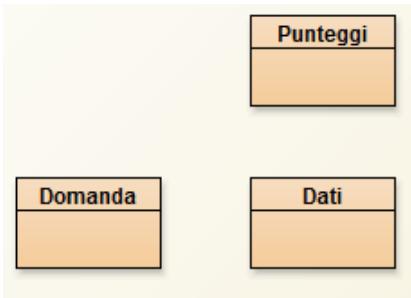


Figura 15: *Package com.safetyGame.mobile.condivisi*

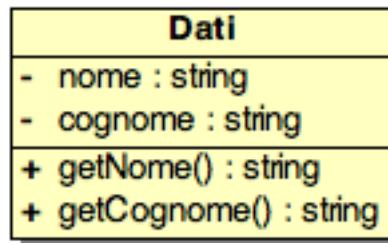
Tipo, obiettivo e funzionamento del componente: Package che raggruppa tutti i tipi di dato non nativi usati dall'applicazione.

Relazione d'uso di altre componenti:

Interfacce con e relazioni d'uso da altre componenti:

Attività svolte e dati trattati:

6.3.1 com.safetyGame.mobile.condivisi.Dati



Funzione

Questa classe si contenere i dati dei Dipendenti

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

Attributi

- String nome
- String cognome

Metodi

public Dati(String nome, String cognome)

Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

Vengono quindi definiti i metodi getter/setter per i vari attributi della classe. In particolare:

- + String getNome()
- + String getCognome()

6.3.2 com.safetyGame.mobile.condivisi.Domanda

Domanda	
-	type : string
-	title : string
-	testo : string
-	risposte : string[]
+	getType() : string
+	getTitle() : string
+	getTesto() : string
+	getRisposte() : string

Funzione

Questa classe fungerà da contenitore per le domande.

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

Attributi

- private id
- String type
- String testo
- String[] risposte
- boolean mobile
- int tempo
- int punteggio
- int corretta
- int numeroRisposte

Metodi

```
public Domanda(int id, String type, String testo, int punteggio, int corretta)
```

Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

```
public Domanda(int id, String type, String testo, String[] risposte, int numeroRisposte, int punteggio, int corretta, int tempo, boolean mobile)
```

Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

Vengono quindi definiti i metodi getter/setter per i vari attributi della classe. In particolare:

- + int getId()
- + String getType()
- + String getTesto()
- + String[] getRisposte()
- + int getPunteggio()
- + int getCorretta()
- + int getNumR()
- + boolean isMobile()
- + int getTempo()

6.3.3 com.safetyGame.mobile.condivisi.Punteggi

Punteggi	
-	rispostedate : string
-	rispostecorrette : string
-	risposteerrate : string
-	punti : string
-	badges : string[]
+	getRisposteDate() : string
+	getRispostecorrette() : string
+	getRisposteerrate() : string
+	getPunti() : string
+	getBadge() : string[]

Funzione

Questa classe fungerà da contenitore per i dati riguardanti i punteggi dei dipendenti

Relazioni d'uso con altri moduli

Questa classe verrà utilizzata dalle seguenti classi:

Attributi

- String **rispostedate**
- String **rispostecorrette**
- String **risposteerrate**
- String **punti**
- String[] **badges**

Metodi

+ **Punteggi(String rispostedate, String rispostecorrette, String risposteerrate, String punti, String[] badge, int numeroBadge)**

Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

Vengono quindi definiti i metodi getter/setter per i vari attributi della classe. In particolare:

- + **String getRispostedate()**
- + **String getRispostecorrette()**
- + **String getRisposteerrate()**
- + **String getPunti()**
- + **String[] getBadges()**

7 Diagrammi di sequenza

In questa sezione sono descritte in maniera più approfondita alcune importanti operazioni previste dal sistema facendo uso dei diagrammi di sequenza. Questi diagrammi sono stati introdotti solo a questo punto del documento in quanto, al loro interno, si fa riferimento a metodi di varie classi, che dovevano essere quindi prima definiti.

Lo scopo è di mostrare come i metodi delle classi dei vari package interagiscano per ottenere il risultato richiesto. Sarà dunque possibile osservare come il flusso delle operazioni nasca dalle varie View e raggiunga il Model, per poi invertirsi⁶.

Analizzeremo in particolare le seguenti azioni:

- **Login da dispositivo mobile**
- **Aggiunta domanda da parte di un Amministratore Sicurezza**

⁶Dove si parlerà di interfacce (ex. *DAOLogin*) si intenderà la sua implementazione in una classe concreta. Per semplicità continueremo a riferirci all'interfaccia, tenendo a mente però che in realtà verranno chiamati metodi e/o campi dati sulle classi concrete

7.1 Login mobile

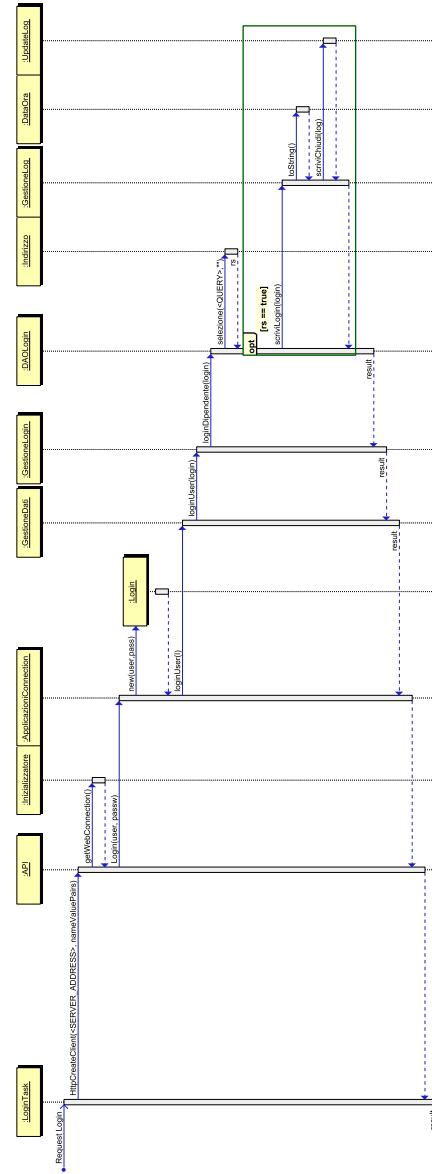


Figura 16: Diagramma di sequenza relativo al login di un Dipendente da dispositivo mobile

- L'utente inserisce all'interno dei campi dati della grafica le sue credenziali d'accesso che verranno passate alla classe **LoginTask**
- **LoginTask** dovrà avviare una comunicazione *HTTP* con le pagine *JSP* deputate alla ricezione delle richieste del *Front-end Mobile*. In particolare dovrà comunicare con la pagina deputata alla ricezione di richieste riguardanti le domande
- Le **API** richiederanno alla classe **Inizializzatore** un oggetto di tipo **ApplicazioniConnection**, sul quale poi si invocherà il metodo *Login*, passando come parametri l'username e password come *Stringhe*
- **ApplicazioniConnection** dovrà generare un oggetto di tipo **Login** a partire dalle informazioni trasmesse dal *Front-End*, quindi invocherà sulla classe Façade **GestioneDati** il metodo *loginUser*
- **GestioneDati** dovrà inoltrare la chiamata alla classe deputata alla gestione delle operazioni di login/logout, **GestioneLogin**, invocando su di essa il metodo *loginUser*
- **GestioneLogin** chiamerà **DAOLogin** invocando il metodo *loginDipendente*.
- **DAOLogin** dovrà interrogare il database attraverso la classe **Indirizzo** per verificare che all'interno del database esista un dipendente con le credenziali d'accesso inserite. Se esiste, dovrà ritornare **true**, altrimenti **false**. Questo valore dovrà propagarsi fino al *Front-end Mobile*, il quale dovrà far visualizzare o la schermata principale o un messaggio d'errore.

7.2 Aggiunta domanda da parte di un Amministratore Sicurezza

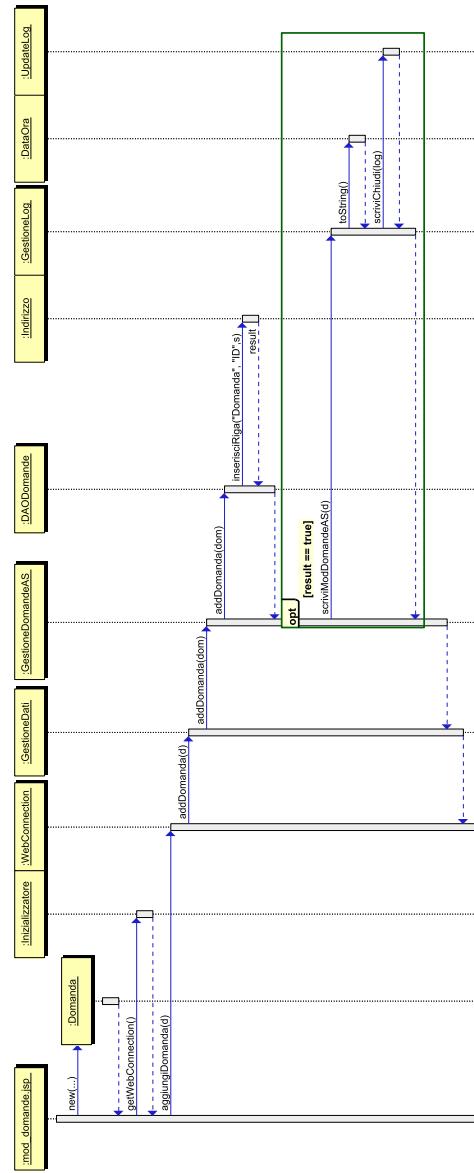


Figura 17: Diagramma di sequenza relativo all'aggiunta di una domanda da parte di un Amministratore Sicurezza

- L'Amministratore Sicurezza aggiungerà una domanda all'elenco di domande disponibili per l'azienda prelevata dal database centrale all'interno della pagina **admin_page.jsp**. Quest'ultima creerà un oggetto di tipo **Domanda** e recupererà dall'oggetto **Inizializzatore** il puntatore alla classe **WebConnection**. Quindi invocherà su di essa il metodo **aggiungiDomanda**
- **WebConnection** quindi redirigerà la chiamata verso la classe **GestioneDati** invocando il metodo **addDomanda**
- **GestioneDati**, a sua volta, invocherà il metodo **addDomanda** della classe **GestioneDomandeAS**, il quale chiamerà l'oggetto **DAODomande** invocando il metodo **addDomanda**
- **DAODomanda** invocherà sull'oggetto **Indirizzo** il metodo **inserisciRiga** e, nel caso non ci siano errori, scrivere nel file di log opportuno tramite l'invocazione del metodo **scriviAddDomande** della classe **GestioneLog** la quale invocherà la classe **UpdateLog** chiamandone il metodo **scriviChiudi**

8 Tracciamento

8.1 Packages naming

La seguente tabella indica la nomenclatura utilizzata nelle tabelle successive per quanto riguarda i package.

Nome package	Sigla
safetygame	SG
safetygame.back	BK
safetygame.back.access	BK.A
safetygame.back.condivisi	BK.COND
safetygame.back.connection	BK.CONN
safetygame.back.controller	BK.CONT
safetygame.desktop	DT
safetygame.desktop.condivisi	DT.COND
safetygame.desktop.logic	DT.L
safetygame.desktop.view	DT.V
safetygame.mobile	MB
safetygame.mobile.condivisi	MB.COND
safetygame.mobile.Utils	MB.U
safetygame.mobile.View	MB.V
safetygame.web	WB
safetygame.web.pagine	WB.P

8.2 Tracciamento componenti-requisiti

8.2.1 Desktop

Package	Classe	Requisiti
DT.V	Notifica	RFOB 6 RFOB 6.1 RFOB 13 RQOB 1 RQOB 3 RQOP 3.1 RQOP 3.2
DT.V	Login	RFOB 4 RFOB 5 RFOB 5.1 RQOB 1 RQOB 3 RQOP 3.1 RQOP 3.2



DT.V	Menu	RFOB 7 RFOB 7.1 RFOB 10 RFOB 12 RFOB 15 RFOB 16 RQOB 1 RQOB 3 RQOP 3.1 RQOP 3.2
DT.L	Timer	RFOB 6 RFOB 7.1 RQOB 1 RQD 5 RVOB 1.1 RVOB 1.1.1
DT.L	ControlNotifica	RFOB 6 RFOB 6.1 RQOB 1 RQD 5 RVOB 1.1 RVOB 1.1.1
DT.L	ControlLogin	RFOB 5 RQOB 1 RQD 5 RVOB 1.1 RVOB 1.1.1
DT.L	ControlMenu	RFOB 7 RFOB 10 RFOB 12 RFOB 15 RFOB 16 RQOB 1 RQD 5 RVOB 1.1 RVOB 1.1.1
DT.L	DatiLogin	RFOB 5 RQOB 1 RQD 5 RVOB 1.1 RVOB 1.1.1
DT.L	ConnBack	RFOB 1 RPOB 1 RQOB 1 RQD 5

		RVOB 1.1 RVOB 1.1.1
DT.L	Parser	RVOB 2
DT.COND	Domanda	RFOB 8 RFOB 9 RFOB 9.1.1 RFD 9.1.2 RFOP 9.1.3 RFD 9.2 RFD 9.2.1 RFOB 14.1 RFOB 27.1 RFOB 28.1 RFOB 28.2
DT.COND	Punteggio	RFOB 15.1 RFOB 23.1
DT.COND	Login	RFOB 5 RFOB 17 RFOB 26

8.2.2 Mobile

Package	Classe	Requisiti
MB.V	DatiTask	RFOB 12 RFOB 12.1 RQOB 1 RQOB 3 RQOP 3.1 RQOP 3.2 RVOB 1.2 RPOB 1
MB.V	LoginTask	RFOB 4 RFOB 5 RFOB 5.1 RQOB 1 RQOB 3 RQOP 3.1 RQOP 3.2 RVOB 1.2 RPOB 1
MB.V	DomandaTask	RFOB 7 RFOB 7.1 RFOB 8 RFOB 9 RFOB 9.1.1

		RFD 9.1.2 RFOP 9.1.3 RFD 9.2 RFD 9.2.1 RQOB 1 RQOB 3 RQOP 3.1 RQOP 3.2 RVOB 1.2 RPOB 1
MB.V	DomandaActivity.QuestTask	RFD 9.1.2
MB.V	PunteggiTask	RFOB 15 RFOB 15.1 RQOB 1 RQOB 3 RQOP 3.1 RQOP 3.2 RVOB 1.2 RPOB 1
MB.V	UpdateTimeTask	RFOB 6 RFOB 6.1 RQOB 1 RQOB 3 RQOP 3.1 RQOP 3.2 RVOB 1.2 RPOB 1
MB.V	DashboardLayout	RQOB 3
MB.V	LoginActivity	RFOB 4 RFOB 4.1 RFOB 4.1.1 RFOB 5 RFOB 5.1 RQOB 1 RQOB 3 RQOP 3.1 RQOP 3.2 RVOB 1.2 RPOB 1
MB.V	DashBoardActivity	RFOB 7 RFOB 7.1 RFOB 10 RFOB 12 RFOB 15 RFOB 16

		RQOB 1 RQOB 3 RQOP 3.1 RQOP 3.2 RVOB 1.2 RPOB 1
MB.V	PunteggiActivity	RFOB 15 RFOB 15.1 RQOB 1 RQOB 3 RQOP 3.1 RQOP 3.2 RVOB 1.2 RPOB 1
MB.V	DatiActivity	RFOB 12 RFOB 12.1 RQOB 1 RQOB 3 RQOP 3.1 RQOP 3.2 RVOB 1.2 RPOB 1
MB.V	DomandaActivity	RFOB 7 RFOB 7.1 RFOB 8 RFOB 9 RFOB 9.1.1 RFD 9.1.2 RFOP 9.1.3 RFD 9.2 RFD 9.2.1 RQOB 1 RQOB 3 RQOP 3.1 RQOP 3.2 RVOB 1.2 RPOB 1
MB.V	TimerNotifica	RFOB 6 RFOB 6.1 RQOB 1 RQOB 3 RQOP 3.1 RQOP 3.2 RVOB 1.2 RPOB 1

MB.U	BootReceiver	RFOB 16
MB.U	IntentIntegrator	RFOB 9.1 ROFB 9.1.2
MB.U	IntentResult	RFOB 9.1 ROFB 9.1.2
MB.U	ConnectionUtils	RFOB 1 RPOB 1 RPOB 3 RQD 5 RVOB 1
MB.COND	Dati	RFOB 5 RFOB 10 RFOB 12 RFOB 12.1
MB.COND	Domanda	RFOB 8 RFOB 9 RFOB 9.1 RFOB 9.1.1 RFD 9.1.2 RFOP 9.1.3 RFD 9.2 RFD 9.2.1 RFOB 14 RFOB 14.1
MB.COND	Punteggi	RFOB 15 RFOB 15.1
MB.COND	Quest	RFD 9.1.2

8.2.3 Back End

Package	Classe	Requisiti
BK.CONN	WebConnection	RFD 2 RFD 2.1 RFOB 4 RFOB 4.1 RFOB 4.1.1 RFOB 5 RFOB 5.1 RFOB 6 RFOB 6.1 RFOB 7 RFOB 7.1 RFOB 8 RFOB 9

		RFOB 9.1.1 RFD 9.1.2 RFOP 9.1.3 RFD 9.2 RFD 9.2.1 RFOB 10 RFOB 11 RFOB 11.1 RFOB 11.2 RFOB 12 RFOB 12.1 RFOB 15 RFOB 15.1 RFOB 16 RFOB 17 RFOB 18 ROFB 19 ROFB 19.1 ROFB 19.2 RFOB 20 RFOB 21 RFOB 21.1 RFOB 22 RFOB 22.1 RFOB 23 RFOB 23.1 RFOB 24 RFOB 26 RFOB 27 RFOB 27.1 RFOB 28 RFOB 28.1 RFOB 28.2 RFOB 30 RPOB 1 RPOB 2 RPOB 3 RQOB 1 RVOB 1 RVOB 2
BK.CONN	ApplicazioniConnection	RFD 2 RFD 2.1 RFOB 4 RFOB 4.1 RFOB 4.1.1

		RFOB 5 RFOB 5.1 RFOB 6 RFOB 6.1 RFOB 7 RFOB 7.1 RFOB 8 RFOB 9 RFOB 9.1.1 RFD 9.1.2 RFOP 9.1.3 RFD 9.2 RFD 9.2.1 RFOB 10 RFOB 11 RFOB 11.1 RFOB 11.2 RFOB 12 RFOB 12.1 RFOB 15 RFOB 15.1 RFOB 16 RPOB 1 RPOB 2 RQOB 1 RVOB 1 RVOB 2
BK.CONT	GestioneLog	RFOB 3 RFOB 3.1
BK.CONT	GestioneLogin	RFOB 5 RFOB 5.1 RFOB 17 RFOB 26
BK.CONT	GestioneDomandeD	RFOB 6 RFOB 6.1 RFOB 7 RFOB 7.1 RFOB 8 RFOB 9 RFOB 9.1.1 RFD 9.1.2 RFOP 9.1.3 RFD 9.2 RFD 9.2.1
BK.CONT	GestioneDomandeAS	RFOB 27



		RFOB 27.1 RFOB 28 RFOB 28.1 RFOB 28.2
BK.CONT	GestionePunteggiD	RFOB 15 RFOB 15.1
BK.CONT	GestionePunteggiAA	RFOB 23 RFOB 23.1 RFOB 24
BK.CONT	GestioneDipendentiD	RFOB 10 RFOB 11 RFOB 11.1 RFOB 11.2 RFOB 12 RFOB 12.1 RFOB 21 RFOB 21.1
BK.CONT	GestioneDipendentiAA	RFOB 11 RFOB 11.1 RFOB 18 ROFB 19 ROFB 19.1 ROFB 19.2 ROFB 19.2.1 ROFB 19.2.2 RFOB 20 RFOB 22 RFOB 22.1
BK.CONT	GestioneBadgeD	RFD 2 RFD 2.1 RFOB 15 RFOB 15.1
BK.CONT	GestioneBadgeAS	RFOB 23 RFOB 23.1
BK.CONT	GestioneRecupero	RFOB 4 RFOB 4.1 RFOB 4.1.1
BK.CONT	GestioneDati	RFD 2 RFD 2.1 RFOB 4 RFOB 4.1 RFOB 4.1.1 RFOB 5 RFOB 5.1 RFOB 6

		RFOB 6.1 RFOB 7 RFOB 7.1 RFOB 8 RFOB 9 RFOB 9.1.1 RFD 9.1.2 RFOP 9.1.3 RFD 9.2 RFD 9.2.1 RFOB 10 RFOB 11 RFOB 11.1 RFOB 11.2 RFOB 12 RFOB 12.1 RFOB 15 RFOB 15.1 RFOB 17 RFOB 18 ROFB 19 ROFB 19.1 ROFB 19.2 RFOB 20 RFOB 21 RFOB 21.1 RFOB 22 RFOB 22.1 RFOB 23 RFOB 23.1 RFOB 26 RFOB 27 RFOB 27.1 RFOB 28 RFOB 28.1 RFOB 28.2
BK.A	DAOFactory	RPOB 2 RQOB 1 RVOB 1 RVOB 2
BK.A	SqldAOFactory	RPOB 2 RQOB 1 RVOB 1 RVOB 2
BK.A	SqldAOLogin	RFOB 5



		RFOB 17 RFOB 26
BK.A	SqldaODipendenti	RFOB 11 RFOB 12 RFOB 12.1 RFOB 14 RFOB 14.1 RFOB 15.1 RFOB 20 RFOB 21 RFOB 21.1 RFOB 22 RFOB 22.1 RFOB 24
BK.A	SqldaODomande	RFOB 8 RFOB 9 RFOB 9.1.1 RFD 9.1.2 RFOP 9.1.3 RFD 9.2 RFD 9.2.1 RFOB 14.1 RFOB 27.1 RFOB 28.1 RFOB 28.2
BK.A	SqldaOPunteggi	RFOB 15.1 RFOB 23.1
BK.A	SqldaOBadge	RFOB 15.1 RFOB 23.1
BK.A	DAOLogin	RFOB 5 RFOB 17 RFOB 26
BK.A	DAODipendenti	RFOB 11 RFOB 12 RFOB 12.1 RFOB 14 RFOB 14.1 RFOB 15.1 RFOB 20 RFOB 21 RFOB 21.1 RFOB 22 RFOB 22.1 RFOB 24
BK.A	DAODomande	RFOB 8

		RFOB 9 RFOB 9.1.1 RFD 9.1.2 RFOP 9.1.3 RFD 9.2 RFD 9.2.1 RFOB 14.1 RFOB 27.1 RFOB 28.1 RFOB 28.2
BK.A	DAOPunteggi	RFOB 15.1 RFOB 23.1
BK.A	DAOBadge	RFOB 15.1 RFOB 23.1
BK.A	Indirizzi	RFOB 1 RFOB 1.1 RPOB 1 RPOB 2 RVOB 2
BK.A	UpdateLog	RFOB 3 RFOB 3.1
BK.COND	Login	RFOB 5 RFOB 17 RFOB 26
BK.COND	Recupero	RFOB 4 RFOB 4.1 RFOB 4.1.1
BK.COND	Dipendente	RFOB 11 RFOB 12 RFOB 12.1 RFOB 14 RFOB 14.1 RFOB 15.1 RFOB 21 RFOB 21.1 RFOB 22 RFOB 22.1
BK.COND	Domanda	RFOB 8 RFOB 9 RFOB 9.1.1 RFD 9.1.2 RFOP 9.1.3 RFD 9.2 RFD 9.2.1 RFOB 14.1

		RFOB 27.1 RFOB 28.1 RFOB 28.2
BK.COND	Punteggio	RFOB 15.1 RFOB 23.1
BK.COND	Badge	RFOB 15.1 RFOB 23.1 RFOP 29 RFOP 29.1
BK.COND	DataOra	RFOB 3.1
BK	Inizializzatore	RFOB 1 RFOB 1.1 RFOB 3 RFOB 3.1 RFOB 4 RFOB 4.1 RFOB 4.1.1 RFOB 5 RFOB 5.1 RFOB 6 RFOB 6.1 RFOB 7 RFOB 7.1 RFOB 8 RFOB 9 RFOB 9.1.1 RFD 9.1.2 RFOP 9.1.3 RFD 9.2 RFD 9.2.1 RFOB 10 RFOB 11 RFOB 11.1 RFOB 11.2 RFOB 12 RFOB 12.1 RFOB 14 RFOB 14.1 RFOB 15 RFOB 15.1 RFOB 16 RFOB 17 RFOB 18 ROFB 19 ROFB 19.1

	ROFB 19.2 RFOB 20 RFOB 21 RFOB 21.1 RFOB 22 RFOB 22.1 RFOB 23 RFOB 23.1 RFOB 24 RFOB 25 RFOB 26 RFOB 27 RFOB 27.1 RFOB 28 RFOB 28.1 RFOB 28.2 RFOP 29 RFOP 29.1 RFOB 30 RPOB 1 RPOB 2 RQOB 1 RQOB 2 RQOB 3 RQD 5 RVOB 1 RVOB 1.1.1 RVOB 2
--	--

8.3 Tracciamento requisiti-componenti

Requisito	Package	Classe
RFOB 1	DT.L MB.U BK.A BK	ConnBack ConnectionUtils Indirizzi Inizializzatore
RFOB 1.1	BK.A BK	Indirizzi Inizializzatore
RFD 2	BK.CONN BK.CONN BK.CONT BK.CONT	WebConnection ApplicazioniConnection GestioneBadgeD GestioneDati
RFD 2.1	BK.CONN BK.CONN BK.CONT BK.CONT	WebConnection ApplicazioniConnection GestioneBadgeD GestioneDati
RFD 2.1.1		
RFOB 3	BK.CONT BK.A BK	GestioneLog UpdateLog Inizializzatore
RFOB 3.1	BK.CONT BK.A BK.COND BK	GestioneLog UpdateLog DataOra Inizializzatore
RFOB 4	DT.V MB.V MB.V BK.CONN BK.CONN BK.CONT BK.CONT BK.COND BK	Login LoginTask LoginActivity WebConnection ApplicazioniConnection GestioneRecupero GestioneDati Recupero Inizializzatore
RFOB 4.1	MB.V BK.CONN BK.CONN BK.CONT BK.CONT BK.COND BK	LoginActivity WebConnection ApplicazioniConnection GestioneRecupero GestioneDati Recupero Inizializzatore
RFOB 4.1.1	MB.V BK.CONN BK.CONN BK.CONT	LoginActivity WebConnection ApplicazioniConnection GestioneRecupero



	BK.CONT BK.COND BK	GestioneDati Recupero Inizializzatore
RFOB 5	DT.V DT.L DT.L DT.COND MB.V MB.V MB.COND BK.CONN BK.CONN BK.CONT BK.CONT BK.A BK.A BK.COND BK	Login ControlLogin DatiLogin Login LoginTask LoginActivity Dati WebConnection ApplicazioniConnection GestioneLogin GestioneDati SqlDAOLogin DAOLogin Login Inizializzatore
RFOB 5.1	DT.V DT.V MB.V MB.V BK.CONN BK.CONN BK.CONT BK.CONT BK	Notifica Login LoginTask LoginActivity WebConnection ApplicazioniConnection GestioneLogin GestioneDati Inizializzatore
RFOB 6	DT.V DT.L DT.L MB.V MB.V BK.CONN BK.CONN BK.CONT BK.CONT BK	Notifica Timer ControlNotifica UpdateTimeTask TimerNotifica WebConnection ApplicazioniConnection GestioneDomandeD GestioneDati Inizializzatore
RFOB 6.1	DT.L MB.V MB.V BK.CONN BK.CONN BK.CONT BK.CONT BK	ControlNotifica UpdateTimeTask TimerNotifica WebConnection ApplicazioniConnection GestioneDomandeD GestioneDati Inizializzatore

RFOB 7	DT.V DT.L MB.V MB.V MB.V BK.CONN BK.CONN BK.CONT BK.CONT BK	Menu ControlMenu DomandaActivity DashBoardActivity DomandaActivity WebConnection ApplicazioniConnection GestioneDomandeD GestioneDati Inizializzatore
RFOB 7.1	DT.V DT.L MB.V MB.V MB.V BK.CONN BK.CONN BK.CONT BK.CONT BK	Menu Timer DomandaActivity DashBoardActivity DomandaActivity WebConnection ApplicazioniConnection GestioneDomandeD GestioneDati Inizializzatore
RFOB 8	DT.COND MB.V MB.V MB.COND BK.CONN BK.CONN BK.CONT BK.CONT BK.A BK.A BK.COND BK	Domanda DomandaActivity DomandaActivity Domanda WebConnection ApplicazioniConnection GestioneDomandeD GestioneDati SqlDAODomande DAODomande Domanda Inizializzatore
RFOB 9	DT.COND MB.V MB.V MB.COND BK.CONN BK.CONN BK.CONT BK.CONT BK.A BK.A BK.COND BK	Domanda DomandaActivity DomandaActivity Domanda WebConnection ApplicazioniConnection GestioneDomandeD GestioneDati SqlDAODomande DAODomande Domanda Inizializzatore
RFOB 9.1	MB.U	IntentIntegrator



	MB.U MB.COND	IntentResult Domanda
RFOB 9.1.1	DT.COND MB.V MB.V MB.COND BK.CONN BK.CONN BK.CONT BK.CONT BK.A BK.A BK.COND BK	Domanda DomandaActivity DomandaActivity Domanda WebConnection ApplicazioniConnection GestioneDomandeD GestioneDati SqlDAODomande DAODomande Domanda Inizializzatore
RFD 9.1.2	DT.COND MB.V MB.V MB.V MB.U MB.U MB.COND MB.COND BK.CONN BK.CONN BK.CONT BK.CONT BK.A BK.A BK.COND BK	Domanda DomandaActivity.QuestTask DomandaActivity DomandaActivity IntentIntegrator IntentResult Domanda Quest WebConnection ApplicazioniConnection GestioneDomandeD GestioneDati SqlDAODomande DAODomande Domanda Inizializzatore
RFOP 9.1.3	DT.COND MB.V MB.V MB.COND BK.CONN BK.CONN BK.CONT BK.CONT BK.A BK.A BK.COND BK	Domanda DomandaActivity DomandaActivity Domanda WebConnection ApplicazioniConnection GestioneDomandeD GestioneDati SqlDAODomande DAODomande Domanda Inizializzatore
RFD 9.2	DT.COND MB.V MB.V	Domanda DomandaActivity DomandaActivity

	MB.COND BK.CONN BK.CONN BK.CONT BK.CONT BK.A BK.A BK.COND BK	Domanda WebConnection ApplicazioniConnection GestioneDomandeD GestioneDati SqlDAODomande DAODomande Domanda Inizializzatore
RFD 9.2.1	DT.COND MB.V MB.V MB.COND BK.CONN BK.CONN BK.CONT BK.CONT BK.A BK.A BK.COND BK	Domanda DomandaActivity DomandaActivity Domanda WebConnection ApplicazioniConnection GestioneDomandeD GestioneDati SqlDAODomande DAODomande Domanda Inizializzatore
RFOB 10	DT.V DT.L MB.V MB.COND BK.CONN BK.CONN BK.CONT BK.CONT BK	Menu ControlMenu DashBoardActivity Dati WebConnection ApplicazioniConnection GestioneDipendentiD GestioneDati Inizializzatore
RFOB 11	BK.CONN BK.CONN BK.CONT BK.CONT BK.CONT BK.A BK.A BK.COND BK	WebConnection ApplicazioniConnection GestioneDipendentiD GestioneDipendentiAA GestioneDati SqlDAODependenti DAODependenti Dipendente Inizializzatore
RFOB 11.1	BK.CONN BK.CONN BK.CONT BK.CONT BK.CONT BK	WebConnection ApplicazioniConnection GestioneDipendentiD GestioneDipendentiAA GestioneDati Inizializzatore

RFOB 11.2	BK.CONN BK.CONN BK.CONT BK.CONT BK	WebConnection ApplicazioniConnection GestioneDipendentiD GestioneDati Inizializzatore
RFOB 12	DT.V DT.L MB.V MB.V MB.V MB.COND BK.CONN BK.CONN BK.CONT BK.CONT BK.A BK.A BK.COND BK	Menu ControlMenu DatiTask DashBoardActivity DatiActivity Dati WebConnection ApplicazioniConnection GestioneDipendentiD GestioneDati SqlDAODipendenti DAO Dipendenti Dipendente Inizializzatore
RFOB 12.1	MB.V MB.V MB.COND BK.CONN BK.CONN BK.CONT BK.CONT BK.A BK.A BK.COND BK	DatiTask DatiActivity Dati WebConnection ApplicazioniConnection GestioneDipendentiD GestioneDati SqlDAODipendenti DAO Dipendenti Dipendente Inizializzatore
RFOB 13	DT.V	Notifica
RFOB 14	MB.COND BK.A BK.A BK.COND BK	Domanda SqlDAODipendenti DAO Dipendenti Dipendente Inizializzatore
RFOB 14.1	DT.COND MB.COND BK.A BK.A BK.A BK.COND BK.COND	Domanda Domanda SqlDAODipendenti SqlDAO Domande DAO Dipendenti DAO Domande Dipendente Domanda



	BK	Inizializzatore
RFOB 15	DT.V DT.L MB.V MB.V MB.V MB.COND BK.CONN BK.CONN BK.CONT BK.CONT BK.CONT BK	Menu ControlMenu PunteggiTask DashBoardActivity PunteggiActivity Punteggi WebConnection ApplicazioniConnection GestionePunteggiD GestioneBadgeD GestioneDati Inizializzatore
RFOB 15.1	DT.COND MB.V MB.V MB.COND BK.CONN BK.CONN BK.CONT BK.CONT BK.CONT BK.A BK.A BK.A BK.A BK.A BK.COND BK.COND BK.COND BK	Punteggio PunteggiTask PunteggiActivity Punteggi WebConnection ApplicazioniConnection GestionePunteggiD GestioneBadgeD GestioneDati SqlDAO dipendenti SqlDAO Badge DAO Dipendenti DAO Punteggi DAO Badge Dipendente Punteggio Badge Inizializzatore
RFOB 16	DT.V DT.L MB.V MB.U BK.CONN BK.CONN BK	Menu ControlMenu DashBoardActivity BootReceiver WebConnection ApplicazioniConnection Inizializzatore
RFOB 17	DT.COND BK.CONN BK.CONT BK.CONT BK.A BK.A BK.COND	Login WebConnection GestioneLogin GestioneDati SqlDAO Login DAO Login Login

	BK	Inizializzatore
RFOB 18	BK.CONN BK.CONT BK.CONT BK	WebConnection GestioneDipendentiAA GestioneDati Inizializzatore
RFOB 19	BK.CONN BK.CONT BK.CONT BK	WebConnection GestioneDipendentiAA GestioneDati Inizializzatore
RFOB 19.1	BK.CONN BK.CONT BK.CONT BK	WebConnection GestioneDipendentiAA GestioneDati Inizializzatore
RFOB 19.2	BK.CONN BK.CONT BK.CONT BK	WebConnection GestioneDipendentiAA GestioneDati Inizializzatore
RFOB 19.2.1	BK.CONT	GestioneDipendentiAA
RFOB 19.2.2	BK.CONT	GestioneDipendentiAA
RFOB 19.2.3		
RFOB 20	BK.CONN BK.CONT BK.CONT BK.A BK.A BK	WebConnection GestioneDipendentiAA GestioneDati SqlDAODipendenti DAODipendenti Inizializzatore
RFOB 21	BK.CONN BK.CONT BK.CONT BK.A BK.COND BK	WebConnection GestioneDipendentiID GestioneDati DAODipendenti Dipendente Inizializzatore
RFOB 21.1	BK.CONN BK.CONT BK.CONT BK.A BK.COND BK	WebConnection GestioneDipendentiID GestioneDati DAODipendenti Dipendente Inizializzatore
RFOB 22	BK.CONN BK.CONT BK.CONT BK.A BK.A BK.COND BK	WebConnection GestioneDipendentiAA GestioneDati SqlDAODipendenti DAODipendenti Dipendente Inizializzatore



RFOB 22.1	BK.CONN BK.CONT BK.CONT BK.A BK.A BK.COND BK	WebConnection GestioneDipendentiAA GestioneDati SqlDAODipendenti DAO Dipendenti Dipendente Inizializzatore
RFOB 23	BK.CONN BK.CONT BK.CONT BK.CONT BK.A BK	WebConnection GestionePunteggiAA GestioneBadgeAS GestioneDati SqlDAOBadge Inizializzatore
RFOB 23.1	DT.COND BK.CONN BK.CONT BK.CONT BK.A BK.A BK.COND BK.COND BK	Punteggio WebConnection GestionePunteggiAA GestioneBadgeAS DAO Punteggi DAO Badge Punteggio Badge Inizializzatore
RFOB 24	BK.CONN BK.CONT BK.A BK.A BK	WebConnection GestionePunteggiAA SqlDAODipendenti DAO Dipendenti Inizializzatore
RFOB 25	BK	Inizializzatore
RFOB 26	DT.COND BK.CONN BK.CONT BK.CONT BK.A BK.A BK.COND BK	Login WebConnection GestioneLogin GestioneDati SqlDAO Login DAO Login Login Inizializzatore
RFOB 27	BK.CONN BK.CONT BK.CONT BK	WebConnection GestioneDomandeAS GestioneDati Inizializzatore
RFOB 27.1	DT.COND BK.CONN BK.CONT BK.CONT BK.A	Domanda WebConnection GestioneDomandeAS GestioneDati SqlDAO Domande



	BK.A BK.COND BK	DAODomande Domanda Inizializzatore
RFOB 28	BK.CONN BK.CONT BK.CONT BK	WebConnection GestioneDomandeAS GestioneDati Inizializzatore
RFOB 28.1	DT.COND BK.CONN BK.CONT BK.CONT BK.A BK.A BK.COND BK	Domanda WebConnection GestioneDomandeAS GestioneDati SqlDAODomande DAODomande Domanda Inizializzatore
RFOB 28.2	DT.COND BK.CONN BK.CONT BK.CONT BK.A BK.A BK.COND BK	Domanda WebConnection GestioneDomandeAS GestioneDati SqlDAODomande DAODomande Domanda Inizializzatore
RFOP 29	BK.COND BK	Badge Inizializzatore
RFOP 29.1	BK.COND BK	Badge Inizializzatore
RFOB 30	BK.CONN BK	WebConnection Inizializzatore
RFOB 31		
RFOB 32		
RFOB 33		
RFOB 34		
RFOP 35		
RPOB 1	DT.L MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V	ConnBack DatiTask LoginTask DomandaActivity PunteggiTask UpdateTimeTask LoginActivity DashBoardActivity PunteggiActivity DatiActivity DomandaActivity



	MB.V MB.U BK.CONN BK.CONN BK.A BK	TimerNotifica ConnectionUtils WebConnection ApplicazioniConnection Indirizzi Inizializzatore
RPOB 2	BK.CONN BK.CONN BK.A BK.A BK.A BK	WebConnection ApplicazioniConnection DAOFactory SqlDAOFactory Indirizzi Inizializzatore
RPOB 3	MB.U BK.CONN	ConnectionUtils WebConnection
RQOB 1	DT.V DT.V DT.V DT.L DT.L DT.L DT.L DT.L MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V BK.CONN BK.CONN BK.A BK.A BK	Notifica Login Menu Timer ControlNotifica ControlLogin ControlMenu DatiLogin ConnBack DatiTask LoginTask DomandaActivity PunteggiTask UpdateTimeTask LoginActivity DashBoardActivity PunteggiActivity DatiActivity DomandaActivity TimerNotifica WebConnection ApplicazioniConnection DAOFactory SqlDAOFactory Inizializzatore
RQOB 2	BK	Inizializzatore
RQOB 2.1		
RQOB 2.2		
RQOB 2.3		
RQOB 2.4		
RQOB 3	DT.V	Notifica



	DT.V DT.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V BK	Login Menu DatiTask LoginTask DomandaActivity PunteggiTask UpdateTimeTask DashboardLayout LoginActivity DashBoardActivity PunteggiActivity DatiActivity DomandaActivity TimerNotifica Inizializzatore
RQOP 3.1	DT.V DT.V DT.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V	Notifica Login Menu DatiTask LoginTask DomandaActivity PunteggiTask UpdateTimeTask LoginActivity DashBoardActivity PunteggiActivity DatiActivity DomandaActivity TimerNotifica
RQOP 3.2	DT.V DT.V DT.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V	Notifica Login Menu DatiTask LoginTask DomandaActivity PunteggiTask UpdateTimeTask LoginActivity DashBoardActivity PunteggiActivity DatiActivity DomandaActivity TimerNotifica
RQOB 4		
RQD 5	DT.L	Timer



	DT.L DT.L DT.L DT.L DT.L MB.U BK	ControlNotifica ControlLogin ControlMenu DatiLogin ConnBack ConnectionUtils Inizializzatore
RQD 6		
RQD 6.1		
RQD 6.2		
RQOP 7		
RVOB 1	MB.U BK.CONN BK.CONN BK.A BK.A BK	ConnectionUtils WebConnection ApplicazioniConnection DAOFactory SqlDAOFactory Inizializzatore
RVOB 1.1	DT.L DT.L DT.L DT.L DT.L DT.L	Timer ControlNotifica ControlLogin ControlMenu DatiLogin ConnBack
RVOB 1.1.1	DT.L DT.L DT.L DT.L DT.L DT.L BK	Timer ControlNotifica ControlLogin ControlMenu DatiLogin ConnBack Inizializzatore
RVOB 1.2	MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V MB.V	DatiTask LoginTask DomandaActivity PunteggiTask UpdateTimeTask LoginActivity DashBoardActivity PunteggiActivity DatiActivity DomandaActivity TimerNotifica
RVOB 2	DT.L BK.CONN BK.CONN BK.A	Parser WebConnection ApplicazioniConnection DAOFactory



	BK.A BK.A BK	SqI DAOFactory Indirizzi Inizializzatore
--	--------------------	--