TEAM

COMMITTED

Università degli Studi di Padova

Definizione di prodotto V1.0

Informazioni sul documento

Nome documento	Definizione di prodotto
Data documento	GG/MM/AAAA
Redattori	Giorgio Maggiolo
Verificatori	xxxxxx
Approvazione	xxxxxx
Uso documento	Interno
Lista distribuzione	• *Team Committed* • *Prof. Tullio Vardanega*

Sommario

xxxxxxxxxxx

Diario delle modifiche

Modifica	Autore	Data	Versione
Redatto il capitolo 2	Giorgio Maggiolo	2012/02/16	V0.2
Creati i sorgenti per il file. Redatto il capitolo 1	Giorgio Maggiolo	2012/02/15	V0.1

Indice

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 compoenenti.

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 improvvisate.

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, v 3.0** (allegato `Norme_di_Progetto_V3.0.pdf`)
- **Analisi dei Requisiti, v 3.0** (allegato `Analisi_dei_Requisiti_V3.0.pdf`)

1.4.2 Riferimenti informativi

- **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

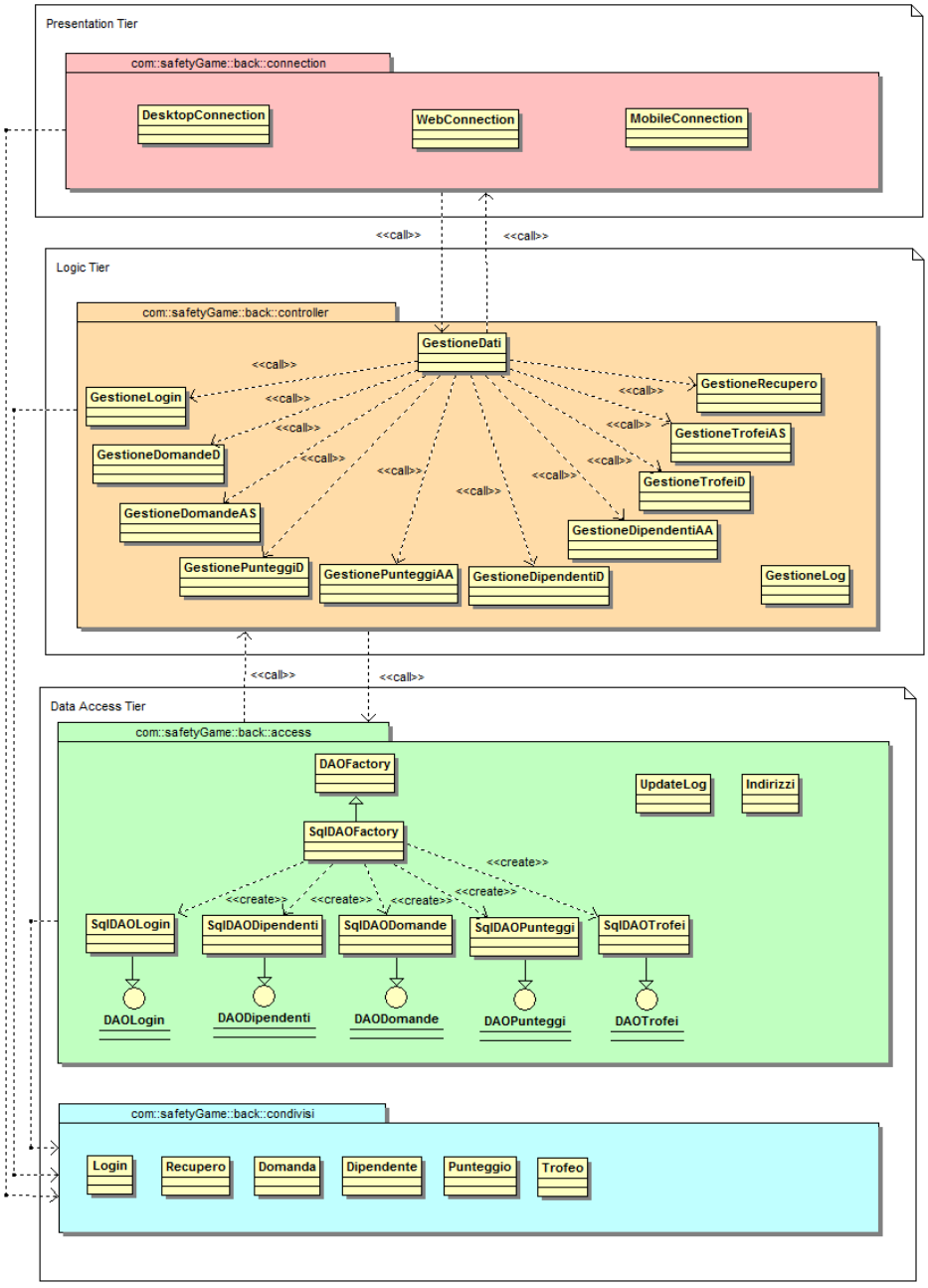

Figure 1: *Back-end, diagramma dei package*

3.1 Package com.safetyGame.back.connection

Figure 2: *Package* ***com.safetyGame.back.connection*** *e le sue relazioni*

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
Interfaccie 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*

Il seguente diagramma delle classi fornisce una panoramica di tutte le classi che compongono il package.

Figure 3: *Diagramma delle classi*

3.1.1 com.safetyGame.back.connection.WebConnection

Tipo, obiettivo e funzione del componente: la classe si occupa di gestire le richieste sia del *Front-end Web* che del *Front-end Desktop*

Relazioni d'uso di altre componenti: utilizza il package **com.safetyGame.back.controller** per inoltrare le richieste provenienti dai *Front-end Web* e *Desktop* ai controlli logici del *Back-End*

Interfaccie 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 *Front-End Web* e *Desktop*

Attività svolte e dati trattati: fa da connettore fra i *Front-End Web* e *Desktop* e il back-end, divenendo parte del componente *Presenter* del design pattern *MVP*

Attributi

- GestioneDati dati

Attributo necessario per mettere in collegamento la classe con il *Controller-Tier* per la gestione delle richieste dei *Front-End Web* e *Desktop*

Metodi

+ WebConnection(GestioneDati dati)

+ Dipendente getDati(String username)

+ boolean login(String username, String password)

+ Punteggio getStat(String username)

+ Badge[] getBadge(String username)

+ void modPass(String pass, String username)

+ void modMail(String mail, String username)

+ void resetPass(String username, String codfis, String mail)

+ Domanda mostraDomanda(String username)

+ void posticipa(String username)

+ boolean rispondi(String username, String risposta)

+ void logout(String username)

+ Domanda[] getElencoDomande(boolean interne)

+ void aggiungiDomanda(int id)

+ void cancellaDomanda(int id)

+ **Dipendente[] getElencoDipendenti()**

+ **void setTrofei(String dipendente, int n)**

+ **boolean aggiungiDipendente(String nome, String cognome, String codfis, String mail, String**

+ **boolean cancellaDipendente(String username)**

+ **void modNome(String username, String nome)**

+ **void modCognome(String username, String cognome)**

+ **void modCodFis(String username, String codfis)**

+**void modUsername(String usernameOld, String username)**

+ **void modImpiego(String username, String impiego)**

3.1.2 com.safetyGame.back.connection.MobileConnection

Tipo, obiettivo e funzione del componente: la classe si occupa di gestire le richieste del *Front-End Mobile*

Relazioni d'uso di altre componenti: utilizza il package **com.safetyGame.back.controller** per inoltrare le richieste provenienti dal *Front-end Mobile* ai controlli logici del *Back-End*

Interfaccie 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 il *Front-End Mobile*

Attività svolte e dati trattati: fa da connettore fra il *Front-End Mobile* e il back-end, divenendo parte del componente *Presenter* del design pattern *MVP*

Attributi

Metodi

3.2 Package com.safetyGame.back.controller

Figure 4: *Package* ***com.safetyGame.back.controller*** *e le sue relazioni*

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
Interfaccie 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*

Il seguente diagramma delle classi fornisce una panoramica di tutte le classi che compongono il package.

Figure 5: *Diagramma delle classi*

3.2.1 com.safetyGame.back.controller.GestioneDati

Tipo, obiettivo e funzione del componente: la classe si occupa di gestire le richieste sia del *Front-end Web* che del *Front-end Desktop*

Relazioni d'uso di altre componenti: utilizza il package **com.safetyGame.back.-controller** per inoltrare le richieste provenienti dai *Front-end Web* e *Desktop* ai controlli logici del *Back-End*

Interfaccie 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 *Front-End Web* e *Desktop*

Attività svolte e dati trattati: fa da connettore fra i *Front-End Web* e *Desktop* e il back-end, divenendo parte del componente *Presenter* del design pattern *MVP*

Attributi

Metodi

+ **Badge[] getBadge(int ultimi, Dipendente d)**

+ **Badge[] getBadges()**

+ **Badge[] getMyBadges(Login l)**

+ **bool recupero(Recupero r)**

+ **boolean checkLogin(Login login)**

+ **boolean modificaDipendente(Dipendente vecchio, Dipendente nuovo)**

+ **Dipendente[] getDipendenti()**

+ **Domanda[] getListaCalderone()**

+**Domanda[] getListaDomande()**

+ **Domanda getDomanda(Login l)**

+ **Domanda getDomandaD(Login l)**

+ **int getDistanzaD(Login l, Badge b)**

+ **Punteggio[] getStatAzienda()**

+ **Punteggio[] getStatDipendente(Dipendente d)**

+ **Punteggio[] getStatDomanda(Domanda d)**

+ **Punteggio[] getStatistiche(Login l)**

+ **Punteggio[] getStatisticheAzienda()**
+ **Punteggio getDistanzaPunti(Login l, Badge b)**
+ **void addDipendente(Dipendente d)**
+ **void addDomande(int[] idDom)**
+ **void creaBadges(Badge b)**
+ **void eliminaDipendente(Dipendente d)**
+ **void eliminaDomande(int[] idDom)**
+ **void scriviDomPost(Login l, Domanda d)**
+ **void scriviDomProp(Login l, Domanda d)**
+ **void scriviDomRic(Login l, Domanda d)**
+ **void scriviDomRisp(Login l, Domanda d)**
+ **void scriviLogin(Login l)**
+ **void scriviLogout(Login l)**
+ **void scriviModEmailD(Dipendente d)**
+ **void scriviModPassD(Dipendente d)**
+ **void scriviOttenimentoBadge(Dipendente d, Badge b)**
+ **void setRisposta(Login l, Risposta r)**
+ **void setRisposta(Login l, Risposta r)**

3.2.2 com.safetyGame.back.controller.GestioneLogin

Tipo, obiettivo e funzione del componente: la classe si occupa di gestire le richieste sia del *Front-end Web* che del *Front-end Desktop*

Relazioni d'uso di altre componenti: utilizza il package **com.safetyGame.back.-controller** per inoltrare le richieste provenienti dai *Front-end Web* e *Desktop* ai controlli logici del *Back-End*

Interfaccie 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 *Front-End Web* e *Desktop*

Attività svolte e dati trattati: fa da connettore fra i *Front-End Web* e *Desktop* e il back-end, divenendo parte del componente *Presenter* del design pattern *MVP*

Attributi

- DAOLogin daoLogin

- GestioneLog gestioneLog

Metodi

+ GestioneLogin(DAOLogin d, GestioneLog g)

+ boolean checkLogin(Login login)

3.2.3 com.safetyGame.back.controller.GestioneDomandeD

Tipo, obiettivo e funzione del componente: la classe si occupa di gestire le richieste sia del *Front-end Web* che del *Front-end Desktop*

Relazioni d'uso di altre componenti: utilizza il package **com.safetyGame.back.-controller** per inoltrare le richieste provenienti dai *Front-end Web* e *Desktop* ai controlli logici del *Back-End*

Interfaccie 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 *Front-End Web* e *Desktop*

Attività svolte e dati trattati: fa da connettore fra i *Front-End Web* e *Desktop* e il back-end, divenendo parte del componente *Presenter* del design pattern *MVP*

Attributi

- DAODomande daoDomande

- GestionePunteggiD gestionePunteggiD

- GestioneLog gestioneLog

Metodi

+ GestioneDomandeD(DAODomande d, GestionePunteggiD gp, GestioneLog gl)

+ Domanda getDomanda(Login l)

+ void setRisposta(Login l, Risposta r)

3.2.4 com.safetyGame.back.controller.GestioneDomandeAS

Tipo, obiettivo e funzione del componente: la classe si occupa di gestire le richieste sia del *Front-end Web* che del *Front-end Desktop*

Relazioni d'uso di altre componenti: utilizza il package **com.safetyGame.back.-controller** per inoltrare le richieste provenienti dai *Front-end Web* e *Desktop* ai controlli logici del *Back-End*

Interfaccie 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 *Front-End Web* e *Desktop*

Attività svolte e dati trattati: fa da connettore fra i *Front-End Web* e *Desktop* e il back-end, divenendo parte del componente *Presenter* del design pattern *MVP*

Attributi

- DAODomande daDomande

Metodi

+ GestioneDomandeAS(DAODomande d)

+ Domanda[] getListaDomande()

+ Domanda[] getListaCalderone()

+ void addDomande(int[] idDom)

+ void eliminaDomande(int[] idDom)

3.2.5 com.safetyGame.back.controller.GestionePunteggiD

Tipo, obiettivo e funzione del componente: la classe si occupa di gestire le richieste sia del *Front-end Web* che del *Front-end Desktop*

Relazioni d'uso di altre componenti: utilizza il package **com.safetyGame.back.-controller** per inoltrare le richieste provenienti dai *Front-end Web* e *Desktop* ai controlli logici del *Back-End*

Interfaccie 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 *Front-End Web* e *Desktop*

Attività svolte e dati trattati: fa da connettore fra i *Front-End Web* e *Desktop* e il back-end, divenendo parte del componente *Presenter* del design pattern *MVP*

Attributi

- DAOPunteggi daoPunteggi

- DAOBadge daoBadge

Metodi

+ GestionePunteggiD(DAOPunteggi dp, DAOBadge db)

+ Punteggio[] getStatisticheAzienda()

+ Punteggio[] getStatistiche(Login l)

3.2.6 com.safetyGame.back.controller.GestionePunteggiAA

Tipo, obiettivo e funzione del componente: la classe si occupa di gestire le richieste sia del *Front-end Web* che del *Front-end Desktop*

Relazioni d'uso di altre componenti: utilizza il package **com.safetyGame.back.-controller** per inoltrare le richieste provenienti dai *Front-end Web* e *Desktop* ai controlli logici del *Back-End*

Interfaccie 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 *Front-End Web* e *Desktop*

Attività svolte e dati trattati: fa da connettore fra i *Front-End Web* e *Desktop* e il back-end, divenendo parte del componente *Presenter* del design pattern *MVP*

Attributi

- DAOPunteggi daoPunteggi

- DAOBadge daoBadge

Metodi

+ GestionePunteggiAA(DAOPunteggi dp, DAOBadge db)

+ Punteggio[] getStatAzienda()

+ Punteggio[] getStatDipendente(Dipendente d)

+ Punteggio[] getStatDomanda(Domanda d)

+ Badge[] getBadge(int ultimi, Dipendente d)

3.2.7 com.safetyGame.back.controller.GestioneDipendentiD

Tipo, obiettivo e funzione del componente: la classe si occupa di gestire le richieste sia del *Front-end Web* che del *Front-end Desktop*

Relazioni d'uso di altre componenti: utilizza il package **com.safetyGame.back.-controller** per inoltrare le richieste provenienti dai *Front-end Web* e *Desktop* ai controlli logici del *Back-End*

Interfaccie 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 *Front-End Web* e *Desktop*

Attività svolte e dati trattati: fa da connettore fra i *Front-End Web* e *Desktop* e il back-end, divenendo parte del componente *Presenter* del design pattern *MVP*

Attributi

- DAODomande daoDomande

- GestionePunteggiD gestionePunteggiD

- GestioneLog gestioneLog

Metodi

+ GestioneDomandeD(DAODomande dd, GestionePunteggiD gpd, GestioneLog gl)

+ Domanda getDomandaD(Login l)

+ void setRisposta(Login l, Risposta r)

3.2.8 com.safetyGame.back.controller.GestioneDipendentiAA

Tipo, obiettivo e funzione del componente: la classe si occupa di gestire le richieste sia del *Front-end Web* che del *Front-end Desktop*

Relazioni d'uso di altre componenti: utilizza il package **com.safetyGame.back.-controller** per inoltrare le richieste provenienti dai *Front-end Web* e *Desktop* ai controlli logici del *Back-End*

Interfaccie 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 *Front-End Web* e *Desktop*

Attività svolte e dati trattati: fa da connettore fra i *Front-End Web* e *Desktop* e il back-end, divenendo parte del componente *Presenter* del design pattern *MVP*

Attributi

- DAODipendenti daoDipendenti

Metodi

+ GestioneDipendentiAA(DAODipendenti dp)

+ Dipendente[] getDipendenti()

+ void addDipendente(Dipendente d)

+ void eliminaDipendente(Dipendente d)

+ boolean modificaDipendente(Dipendente vecchio, Dipendente nuovo)

3.2.9 com.safetyGame.back.controller.GestioneBadgeD

Tipo, obiettivo e funzione del componente: la classe si occupa di gestire le richieste sia del *Front-end Web* che del *Front-end Desktop*

Relazioni d'uso di altre componenti: utilizza il package **com.safetyGame.back.-controller** per inoltrare le richieste provenienti dai *Front-end Web* e *Desktop* ai controlli logici del *Back-End*

Interfaccie 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 *Front-End Web* e *Desktop*

Attività svolte e dati trattati: fa da connettore fra i *Front-End Web* e *Desktop* e il back-end, divenendo parte del componente *Presenter* del design pattern *MVP*

Attributi

- DAOBadge daoBadge

Metodi

+ GestioneBadgeD(DAOBadge db)

+ Badge[] getBadges()

+ Punteggio getDistanzaPunti(Login l, Badge b)

+ Badge[] getMyBadges(Login l)

+ int getDistanzaD(Login l, Badge b)

3.2.10 com.safetyGame.back.controller.GestioneBadgeAS

Tipo, obiettivo e funzione del componente: la classe si occupa di gestire le richieste sia del *Front-end Web* che del *Front-end Desktop*

Relazioni d'uso di altre componenti: utilizza il package **com.safetyGame.back.-controller** per inoltrare le richieste provenienti dai *Front-end Web* e *Desktop* ai controlli logici del *Back-End*

Interfaccie 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 *Front-End Web* e *Desktop*

Attività svolte e dati trattati: fa da connettore fra i *Front-End Web* e *Desktop* e il back-end, divenendo parte del componente *Presenter* del design pattern *MVP*

Attributi

- DAOBadge daoBadge

Metodi

+ GestioneBadgeAS(DAOBadge db)

+ Badge[] getBadges()

+ void creaBadges(Badge b)

3.2.11 com.safetyGame.back.controller.GestioneRecupero

Tipo, obiettivo e funzione del componente: la classe si occupa di gestire le richieste sia del *Front-end Web* che del *Front-end Desktop*

Relazioni d'uso di altre componenti: utilizza il package **com.safetyGame.back.-controller** per inoltrare le richieste provenienti dai *Front-end Web* e *Desktop* ai controlli logici del *Back-End*

Interfaccie 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 *Front-End Web* e *Desktop*

Attività svolte e dati trattati: fa da connettore fra i *Front-End Web* e *Desktop* e il back-end, divenendo parte del componente *Presenter* del design pattern *MVP*

Attributi

- String email

- DAODipendente daoDipendente

Metodi

+ GestioneRecupero(String email, DAODipendente dd)

+ bool recupero(Recupero r)

3.2.12 com.safetyGame.back.controller.GestioneLog

Tipo, obiettivo e funzione del componente: la classe si occupa di gestire le richieste sia del *Front-end Web* che del *Front-end Desktop*

Relazioni d'uso di altre componenti: utilizza il package **com.safetyGame.back.-controller** per inoltrare le richieste provenienti dai *Front-end Web* e *Desktop* ai controlli logici del *Back-End*

Interfaccie 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 *Front-End Web* e *Desktop*

Attività svolte e dati trattati: fa da connettore fra i *Front-End Web* e *Desktop* e il back-end, divenendo parte del componente *Presenter* del design pattern *MVP*

Attributi

- UpdateLog updateLog

Metodi

+ GestioneLog(UpdateLog ul)

+ void scriviLogin(Login l)

+ void scriviLogout(Login l)

+ void scriviDomRic(Login l, Domanda d)

+ void scriviDomProp(Login l, Domanda d)

+ void scriviDomPost(Login l, Domanda d)

+ void scriviDomRisp(Login l, Domanda d)

+ void scriviModPassD(Dipendente d)

+ void scriviModEmailD(Dipendente d)

+ void scriviOttenimentoBadge(Dipendente d, Badge b)

3.3 Package com.safetyGame.back.access

Tipo, obiettivo e funzione del componente: la classe si occupa di gestire le richieste sia del *Front-end Web* che del *Front-end Desktop*

Relazioni d'uso di altre componenti: utilizza il package **com.safetyGame.back.-controller** per inoltrare le richieste provenienti dai *Front-end Web* e *Desktop* ai controlli logici del *Back-End*

Interfaccie 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 *Front-End Web* e *Desktop*

Attività svolte e dati trattati: fa da connettore fra i *Front-End Web* e *Desktop* e il back-end, divenendo parte del componente *Presenter* del design pattern *MVP*

Il seguente diagramma delle classi fornisce una panoramica di tutte le classi che compongono il package.

Figure 6: *Diagramma delle classi*

3.3.1 Interfaccia com.safetyGame.back.access.DAODipendenti

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:

Metodi

+ **void passD(String pass, String username)**

+ **void passA(String pass, String username)**

+ **void mailD(String mail, String username)**

+ **Punteggio getStat(String username)**

+ **Dipendente[] elencoDipendenti()**

+ **boolean aggiungiDipendente(String nome, String cognome, String codfis, String mail, String**

+ **boolean cancellaDipendente(String username)**

+ **void modNome(String username, String nome)**

+ **void modCognome(String username, String cognome)**

+ **void modCodFis(String username, String codfis)**

+ **void modUsername(Stirng usernameOld, String username)**

+ **void modImpiego(String username, String impiego)**

+ **void trofei(String username, int n)**

+ **void reset(String username, String codfis, String mail)**

3.3.2 com.safetyGame.back.access.SqlDAODipendenti

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:

Attributi

Metodi

+ **void passD(String pass, String username)**

+ **void passA(String pass, String username)**

+ **void mailD(String mail, String username)**

+ **Punteggio getStat(String username)**

+ **Dipendente[] elencoDipendenti()**

+ **boolean aggiungiDipendente(String nome, String cognome, String codfis, String mail, String**

+ **boolean cancellaDipendente(String username)**

+ **void modNome(String username, String nome)**

+ **void modCognome(String username, String cognome)**

+ **void modCodFis(String username, String codfis)**

+ **void modUsername(Stirng usernameOld, String username)**

+ **void modImpiego(String username, String impiego)**

+ **void trofei(String username, int n)**

+ **void reset(String username, String codfis, String mail)**

3.3.3 Interfaccia com.safetyGame.back.access.DAODomande

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:

Metodi

+ **Domanda getDomanda(Dipendente d)**

Metodo che recupera i campi che compongono un oggetto Domanda tramite il suo id.

+ **boolean posticipa(Dipendente d, Domanda dom);**

+ **boolean rispondi(Dipendente d, Domanda dom);**

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

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

+ **boolean addDomanda(Domanda d);**

+ **boolean remDomanda(Domanda d);**

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

3.3.4 com.safetyGame.back.access.SqlDAODomande

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:

Attributi

- Indirizzo serverDomande;

- Indirizzo serverAzienda;

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 d)

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 d, 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 d, Domanda dom)
+ ArrayList<Domanda> domandeA()
+ ArrayList<Domanda> domande(Dipendente d, Domanda dom)
+ boolean addDomanda(Domanda d)

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

3.3.5 Interfaccia com.safetyGame.back.access.DAOFactory

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.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.

3.3.6 com.safetyGame.back.access.SqlDAOFactory

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

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

+ **SqlDAOFactory()**
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.

+ **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.

3.3.7 Interfaccia com.safetyGame.back.access.DAOLogin

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

Metodi

+ boolean loginAmministratore(Login l)

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

+ boolean loginDipendente(Login l)

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

3.3.8 com.safetyGame.back.access.SqlDAOLogin

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.acecss.SqlDAOFactory`

Inoltre vengono utilizzate le seguenti classi:

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

Attributi

- Indirizzo serverAzienda

Metodi

+ SqlDAOLogin(Indirizzo azienda)

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

+ boolean loginAmministratore(Login l)

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 l)

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**.

3.3.9 Interfaccia com.safetyGame.back.access.DAOPunteggi

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.SqlDAOPunteggi`
- `back.access.DAOFactory`
- `back.access.SqlDAOFactory`
- `back.controller.GestioneDomandeD`
- `back.controller.GesteionePunteggiAA`
- `back.controller.GestionePunteggiD`

Metodi

+ Punteggio getStat(Dipendente d)

Questo metodo dovrà ritornare un oggetto di tipo Punteggio con informazioni relative al solo Dipendente in oggetto.

+ 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).

3.3.10 com.safetyGame.back.access.SqlDAOPunteggi

Funzione

Implementa back.access.DAOPunteggi.
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;

- Indirizzo serverDomande;

Metodi

+ SqlDAOPunteggi(Indirizzo azienda, Indirizzo domande)
Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

+ Punteggio getStat(Dipendente d)
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.

3.3.11 com.safetyGame.back.access.UpdateLog

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`

Attributi

- PrintWriter out

Metodi

+ **UpdateLog(String s)**
Costruttore che dovrà impostare il valore degli attributi secondo il valore dei parametri passati.

+ **void finalize()**
Distruttore della classe: dovrà chiudere lo stream al file di log

+ **void scrivi(String s)**
Aggiunge una nuova riga al file di log

+ **void scriviChiudi(String s)**
Aggiunge una nuova riga al file e chiude lo stream al file di log

3.3.12 com.safetyGame.back.access.Indirizzo

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.SqlDAODipententi`
- `back.access.SqlDAODomande`
- `back.access.SqlDAOFactory`
- `back.access.SqlDAOLogin`
- `back.access.SqlDAOPunteggi`

Attributi

- Connection conn

- Statement connettore

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:

(?,?....,?)

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 ritoranre **false**, altrimenti **true**.

+ boolean cancellaRiga(String tabella, String controlli)

Elimina una tupla di valori all'interno della tabella indicata, seleziondola 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)

Prevela 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 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*

Interfaccie 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**

3.4.1 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.convidisi.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 d, Punteggio p)**

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)**

3.4.2 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)**

3.4.3 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.GestioneDipendentiD`
- `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 ammAA
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(Stirng 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)**

3.4.4 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.SqlDAOPunteggi`
- `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'Amminstratore 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)**

3.4.5 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.SqlDAODipendenti`
- `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)**

3.4.6 com.safetyGame.back.condivisi.Punteggio

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 dipendenti 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)

3.4.7 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.SqlDAODipendenti`
- `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

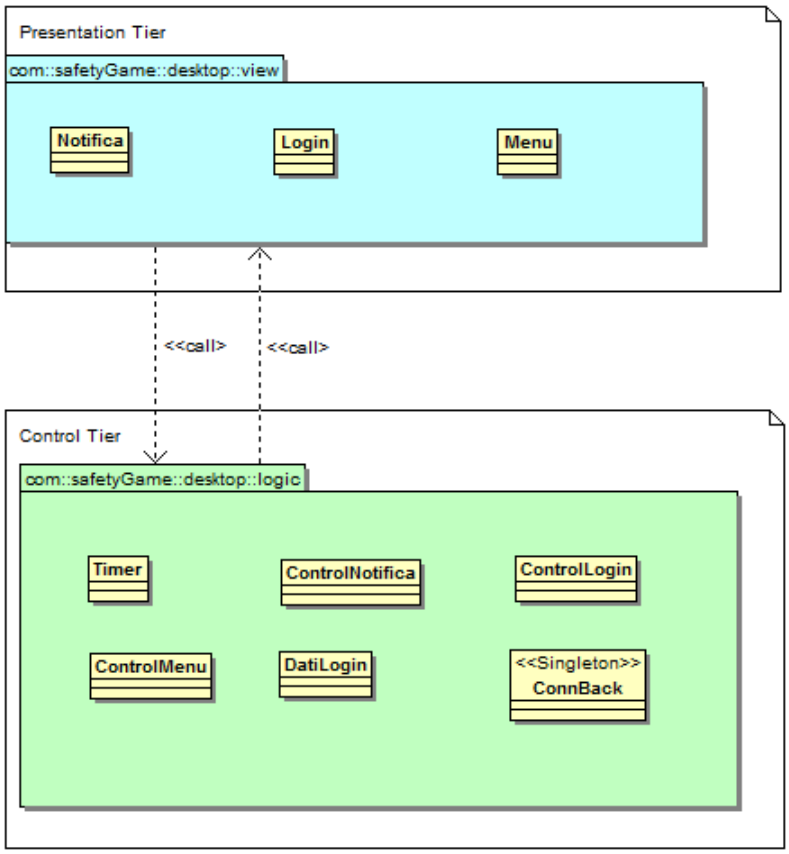

Figure 7: *Front-end Desktop, diagramma dei package*

5 Specifica Front-end Web

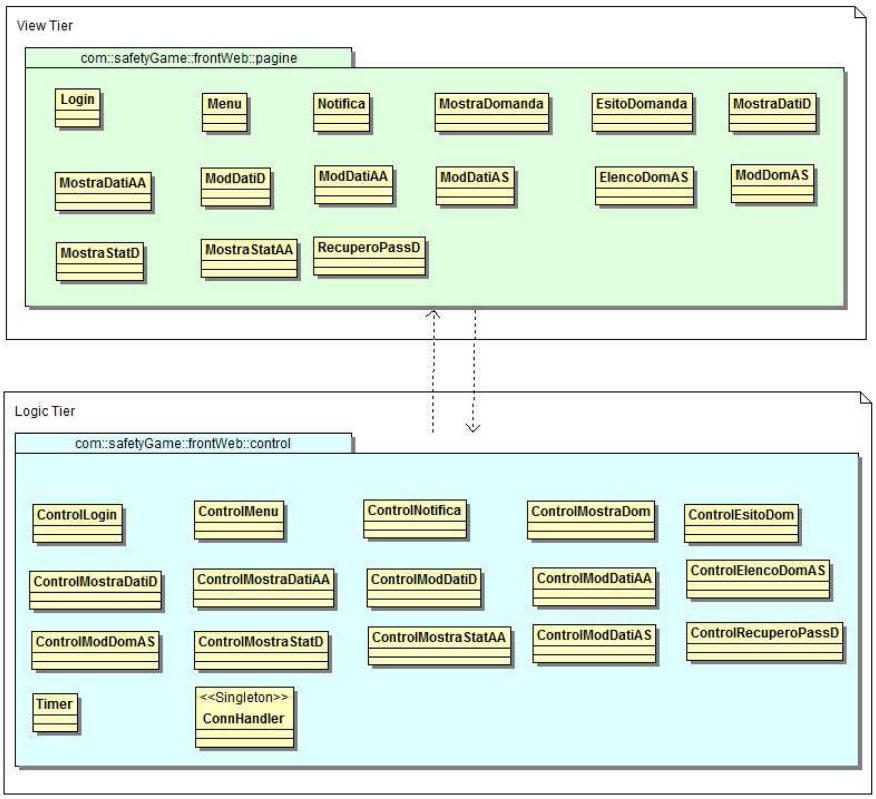

Figure 8: *Front-end Web, Diagramma dei Package*

6 Specifica Front-end Mobile

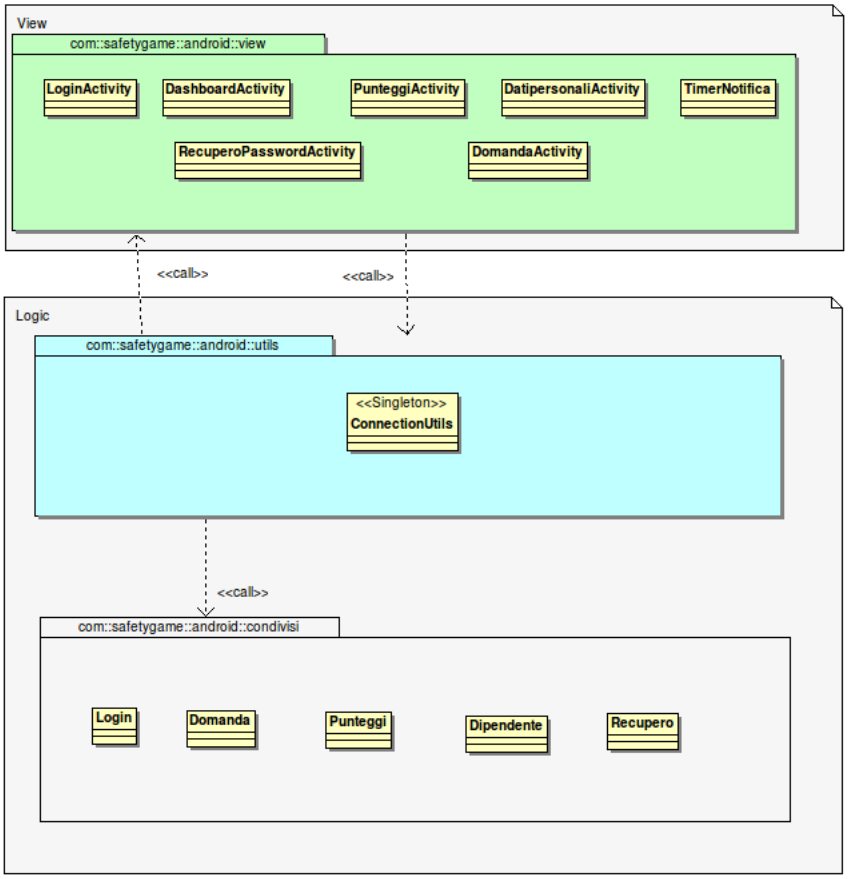

Figure 9: *Front-end Mobile, diagramma dei package*

6.1 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 TimenNotifica è parte di questo componente nonostante estenda la classe Java Android "Service".
Relazione d'uso di altre componenti: Vengono utilizzate funzioni del package Utils.
Interfaccie 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.LoginActivity

Tipo, obiettivo e funzionamento del componente: Consente ad un Dipendente di effettuare il login.
Relazione d'uso di altre componenti: Viene richiamato ConnectionUtils per inviare i dati al server, inoltre consente di arrivare alle classi DashboardActivity e RecuperoPasswordActivity. Oggetti di tipo Login sono utilizzati per trasferire i dati.
Interfaccie con e relazioni d'uso da altre componenti: Riceve dati da ConnectionUtils.
Attività svolte e dati trattati: Raccoglie i dati che il Dipendente Autenticato inserisce ed invia l'apposita richiesta di login al back-end.
Attributi:
- Context context

Metodi:
+ void onCreate(Bundle savedInstanceState)

6.1.2 com.safetyGame.mobile.View.DashboardActivity

Tipo, obiettivo e funzionamento del componente: Permette di raggiungere le altre Activity del programma.
Relazione d'uso di altre componenti: Ha la possibilità di richiamare le classi PunteggioActivity, DomandeActivity, DatiActivity.
Interfaccie con e relazioni d'uso da altre componenti: Viene chiamato da LoginActivity.
Attività svolte e dati trattati: Intercetta gli input e lancia altre Activity.
Attributi:
- Context context

Metodi:
+ void onCreate(Bundle savedInstanceState)

6.1.3 com.safetyGame.mobile.View.PunteggiActivity

Tipo, obiettivo e funzionamento del componente: Permette ad un Dipendente Autenticato di visualizzare i vari punteggi e trofei.
Relazione d'uso di altre componenti: ConnectionUtils è usato per richiedere dati al server, Punteggi invece è usato per trasferire i dati.
Interfaccie con e relazioni d'uso da altre componenti: Viene chiamato da DashboardActivity e riceve i dati da ConnectionUtils.
Attività svolte e dati trattati: Permette la visualizzazione dei trofei del DIpendente Autenticato.
Attributi:
- Context context

Metodi:
+ void onCreate(Bundle savedInstanceState)
+ boolean onOptionsItemSelected(MenuItem item)

6.1.4 com.safetyGame.mobile.View.DatiActivity

Tipo, obiettivo e funzionamento del componente: Permette ad un Dipendente Autenticato di visionare e modificare i dati personali personali.
Relazione d'uso di altre componenti: ConnectionUtils è usato per richiedere ed inviare modifiche riguardanti i dati personali del Dipendente Autenticato**.** Usa inoltre oggetti di tipo Dipendente per trasferire i dati.
Interfaccie con e relazioni d'uso da altre componenti: Chiamato da DashboardActivity e riceve dati richiesti da ConnectionUtils.
Attività svolte e dati trattati: Visualizzazione e modifica dei dati personali di un DipendenteAutenticato.
Attributi:
- Context context

Metodi:
+ void onCreate(Bundle savedInstanceState)
+ boolean onOptionsItemSelected(MenuItem item)

6.1.5 com.safetyGame.mobile.View.DomandaActivity

Tipo, obiettivo e funzionamento del componente: Mette a disposizione del Dipendente Autenticato la possibilità di visualizzare, richiedere e rispondere alle domande.
Relazione d'uso di altre componenti: ConnectionUtils viene utilizzato per richiedere dati al server. Domanda è usato per trasferire dati.
Interfaccie con e relazioni d'uso da altre componenti: Chiamato da DashboardActivity o TimerNotifica e riceve dati da ConnectionUtils.
Attività svolte e dati trattati: Consente ad un Dipendente Autenticato di richiedere, visualizzare e rispondere a domande.
Attributi:
- Context context

Metodi:
+ void onCreate(Bundle savedInstanceState)
+ boolean onOptionsItemSelected(MenuItem item)
+ void onActivityResult(int requestCode, int resultCode, Intent intent)

6.1.6 com.safetyGame.mobile.View.RecuperoPasswordActivity

Tipo, obiettivo e funzionamento del componente: Permette il recupero della password di un Dipendente**.**
Relazione d'uso di altre componenti: ConnectionUtils è utilizzato per l'invio dei dati al server.
Interfaccie con e relazioni d'uso da altre componenti: Chiamato da LoginActivity e riceve dati da ConnectionUtils.
Attività svolte e dati trattati: Effettua la raccolta dei dati del Dipendente per richiedere il recupero della password.
Attributi:
Metodi:

6.1.7 com.safetyGame.mobile.View.TimerNotifica

Tipo, obiettivo e funzionamento del componente: Servizio di temporizzazioen che propone al Dipendente Autenticato una domanda secondo quanto impostato tramite le notifiche standard di Android.
Relazione d'uso di altre componenti: Viene chiamata la classe DomandaActivity.
Interfaccie con e relazioni d'uso da altre componenti: Nessuna.
Attività svolte e dati trattati: Un timer, in accordo con quanto impostato di default nel dispositivo Android in cui sarà installato il software, farà comparire una notifica standard di Android che permette di visualizzare una nuova domanda.
Attributi:
- long mStartTime
- Handler mHandler = new Handler()
- NotificationManager mNotificationManager
- Notification notification
- SaredPreferences prefs

Metodi:
+ int onStartCommand(Intent intent, int flags, int startId)
+ IBinder onBind(Intent arg0)

6.1.8 com.safetyGame.mobile.View.DashboardLayout

Tipo, obiettivo e funzionamento del componente: Classe che aiuta la visualizzazione della Dashboard all'avvio dell'applicazione.
Relazione d'uso di altre componenti: Nessuna**.**
Interfaccie con e relazioni d'uso da altre componenti: Viene usata per costruire il layout "main.xml".
Attività svolte e dati trattati:
Attributi:
- static final int UNEVEN_GRID_PENALTY_MULTIPLIER = 10
- UNEVEN_GRID_PENALTY_MULTIPLIER = 1
- int mMaxChildWidth = 0
- int mMaxChildHeight = 0

Metodi:
void onMeasure(int widthMeasureSpec, int heightMeasureSpec)
void onMeasure(int widthMeasureSpec, int heightMeasureSpec)

6.2 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.
Interfaccie 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

Tipo, obiettivo e funzionamento del componente: Classe che estende BroadcastReceiver e viene automaticamente chiamata al completamento del boot del dispositivo.
Relazione d'uso di altre componenti: Fa partire il service com.safetygame.mobile.View.TimerNotifica
.
Interfaccie con e relazioni d'uso da altre componenti: Nessuna
Attività svolte e dati trattati: La classe fa avviare il servizio in bacground che propone le domande.
Variabili:
Nessuna
Metodi:
+ onReceive(Context context, Intent arg1)

6.2.2 com.safetyGame.mobile.Utils.ConnectionUtils

Tipo, obiettivo e funzionamento del componente: 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.
Relazione d'uso di altre componenti: Richiama gli appositi metodi del back-end in relazione alle richieste della View.
Interfaccie con e relazioni d'uso da altre componenti: Le Activity richiamano la classe che invia o riceve dati al back-end.
Attività svolte e dati trattati: La classe si occupa di effettuare richieste HTTP al server e ricevere le risposte tramite pagine XML.
Variabili:
Nessuna
Metodi:
- static Element rootXML(HttpResponse response)
+ static String parseXML(Element root, String stringa, int position)
+ static Object HttpCreateClient(String url, List<NameValuePair> nameValuePairs)

6.2.3 com.safetyGame.mobile.Utils.IntentIntegrator

Tipo, obiettivo e funzionamento del componente: Classe che aiuta l'integrazione dell'applicazione Safety Game con l'applicazione Barrcode Scanner.
Relazione d'uso di altre componenti: Nessuna**.**
Interfaccie con e relazioni d'uso da altre componenti: Viene richiamata dall'Activity.
Attività svolte e dati trattati:
Variabili:
+ 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 = list("UPC_A", "UPC_E", "EAN_8", "EAN_13", "CODE_39", "CODE_93", "CODE_128", "ITF", "RSS_14", "RSS_EXPANDED")
+ static final Collection<String> QR_CODE_TYPES = Collections.singleton("QR_CODE");
+ static final Collection<String> DATA_MATRIX_TYPES = Collections.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:
+ 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(
+ void setTargetApplications(Collection<String> targetApplications)
+ void setSingleTargetApplication(String targetApplication)
+ AlertDialog initiateScan()
+ AlertDialog initiateScan(Collection<String> desiredBarcodeFormats)
# void startActivityForResult(Intent intent, int code)
- AlertDialog showDownloadDialog()
- String findTargetAppPackage(Intent intent)
+ static IntentResult parseActivityResult(int requestCode, int resultCode, Intent intent)
+ void shareText(CharSequence text)
- private static Collection<String> list(String... values)

6.2.4 com.safetyGame.mobile.Utils.IntentResult

Tipo, obiettivo e funzionamento del componente: Classe che aiuta a gestire i risultati dello scan del barcode.
Relazione d'uso di altre componenti: Nessuna
Interfaccie con e relazioni d'uso da altre componenti: Restituisce i dati all'Activity
Attività svolte e dati trattati:
Variabili:
- final String contents
- final String formatName
- final byte[] rawBytes
- final Integer orientation
- final String errorCorrectionLevel

Metodi:
+ String getContents()
+ String getFormatName()
+ byte[] getRawBytes()
+ Integer getOrientation()
+ String getErrorCorrectionLevel()
+ String toString()

6.3 Package com.safetyGame.mobile.condivisi

Tipo, obiettivo e funzionamento del componente: Package che raggruppa tutti i tipi di dato non nativo usati dall'applicazione.
Relazione d'uso di altre componenti:
Interfaccie con e relazioni d'uso da altre componenti:
Attività svolte e dati trattati:

6.3.1 com.safetyGame.mobile.condivisi.Dati

Tipo, obiettivo e funzionamento del componente: Package che raggruppa tutti i tipi di dato non nativo usati dall'applicazione.
Relazione d'uso di altre componenti:
Interfaccie con e relazioni d'uso da altre componenti:
Attività svolte e dati trattati:
Variabili:
- String nome
- String cognome

Metodi:
+ String getNome()
+ String getCognome()

6.3.2 com.safetyGame.mobile.condivisi.Domanda

Tipo, obiettivo e funzionamento del componente: Package che raggruppa tutti i tipi di dato non nativo usati dall'applicazione.
Relazione d'uso di altre componenti:
Interfaccie con e relazioni d'uso da altre componenti:
Attività svolte e dati trattati:
Variabili:
- String type
- String title
- String testo
- String[] risposte

Metodi:
+ String getType()
+ String getTitle()
+ String getTesto()
+ String[] getRisposte()

6.3.3 com.safetyGame.mobile.condivisi.Punteggi

Tipo, obiettivo e funzionamento del componente: Package che raggruppa tutti i tipi di dato non nativo usati dall'applicazione.
Relazione d'uso di altre componenti:
Interfaccie con e relazioni d'uso da altre componenti:
Attività svolte e dati trattati:
Variabili:
- String rispostedate
- String rispostecorrette
- String risposteerrate
- String punti
- String[] badges

Metodi:
+ String getRispostedate()
+ String getRispostecorrette()
+ String getRisposteerrate()
+ String getPunti()
+ String[] getBadges()

6.3.4 com.safetyGame.mobile.condivisi.Quest

Tipo, obiettivo e funzionamento del componente: Package che raggruppa tutti i tipi di dato non nativo usati dall'applicazione.
Relazione d'uso di altre componenti:
Interfaccie con e relazioni d'uso da altre componenti:
Attività svolte e dati trattati:
Variabili:
- String title
- String testo

Metodi:
+ String getTitle()
+ String getTesto()