Laurea Triennale in informatica-Università di Salerno
Corso di *Ingegneria del Software*- Prof.ssa F. Ferrucci

ODD
Object Design Document

Riferimento	
Versione	1.0
Data	15/12/2018
Destinatario	Prof.ssa F. Ferrucci
Presentato da	Daniele De Vinco, Donatella Cioffi, Federica Ungherese, Luigi Di Palma
Approvato da	

Revision History

Data	Versione	Descrizione	Autori
15/12/2018	1.0	Prima stesura	Daniele De Vinco, Donatella Cioffi, Federica Ungherese, Luigi Di Palma
14/01/2019	1.1	Revisione finale. Sostituzione di package e design pattern, modifica del cap.3	Daniele De Vinco, Donatella Cioffi, Federica Ungherese, Luigi Di Palma

Sommario

1. Introduzione

1.1 Trade-off

Funzionalità vs. Usabilità:
Il nostro sistema mira ad una maggiore usabilità rispetto alle funzionalità perché è stato ideato per garantire un utilizzo il più semplice possibile da parte degli utenti. Proprio per questo l'app risulterà avere un'interfaccia chiara e intuitiva.

Costi vs. Robustezza:
Il nostro sistema predilige una maggiore robustezza rispetto ai costi perché sarà in grado comportarsi in modo ragionevole qualora si presentino alcune situazioni impreviste. Non sono consentiti errori sull'inserimento di dati affinché il database possa essere robusto e privo di inconsistenze.

Efficienza vs. Portabilità
Il sistema ideato predilige efficienza alla portabilità perché è stato ideato per sfruttare un numero minimo di risorse del dispositivo da cui è utilizzato. Essendo un sistema molto semplice mira a adoperare solo le risorse strettamente necessarie.

Costi vs. Riusabilità
Il sistema predilige la riusabilità rispetto ai costi perché sarà possibile in futuro riutilizzare tale progetto, o parte di esso, per inserirlo all'interno di uno nuovo. Qualora vi sia un progetto dove vi sarà presente una sorta di forum sarà possibile incorporalo al suo interno.

1.2 Componenti off-the-shelf

Per il progetto abbiamo deciso di utilizzare una piattaforma sviluppata da Google, Firebase, per gestire il lato server. Firebase consente di gestire l'autenticazione degli utenti in maniera sicura e fornisce la possibilità di istanziare un database disponibile 24/7.

Il database utilizza dati JSON che possono essere condivisi in tempo reale da tutti gli utenti.

1.3 Linee guida per la documentazione dell'interfaccia

In ogni file sarà presente, relativamente a ciascun metodo, una documentazione realizzata mediante Javadoc.

La convenzione adottata sui nomi di metodi, variabili e file contenuti nei packages è la Camel Notation.

1.3.1 Organizzazione dei file

Ogni file deve essere:

- Correlato ad un'unica funzionalità che persegue, di modo da garantire coerenza;
- Diviso in più file se la funzionalità relativa a tale file risulta tanto complessa e lunga da non consente una buona leggibilità del codice;
- La convenzione adottata riguardo i nomi dei file, delle operazioni e delle variabili è quella di avere nomi evocativi ed in lingua inglese, fatta eccezione per gli oggetti Entity individuati in fase di analisi dei requisiti ed indicati nel Class Diagram.
- Organizzare in una cartella i file delle librerie usate e le altre risorse scaricate necessarie per lo sviluppo del progetto.

1.3.2 Spostamento di linee

Quando un'espressione supera la lunghezza della linea, occorre spezzarla secondo i seguenti principi generali:

- Interrompere la linea dopo una virgola;
- Interrompere la linea prima di un operatore;
- Preferire interruzioni di alto livello rispetto ad interruzioni di basso livello (interrompere laddove non si interrompe un discorso logico, valido soprattutto per le formule);
- Allineare la nuova linea con l'inizio dell'espressione nella linea precedente;
- Se le regole precedenti rendono il codice più confuso o il codice è troppo spostato verso il margine destro, utilizzare solo otto spazi di indentazione.

1.3.3 Indentazione

- L'indentazione deve essere effettuata con un TAB da due spazi, qualunque sia il linguaggio usato per la produzione di codice, ed ogni istruzione deve essere opportunamente indentata.

Esempio forma corretta:

```
<LinearLayout>
        <Button>
        </Button>
</LinearLayout>
```

1.3.4 Inizializzazione

- Inizializzare le variabili locali nel punto in cui sono state dichiarate a meno che il loro valore iniziale non dipenda da un calcolo che occorre eseguire prima.

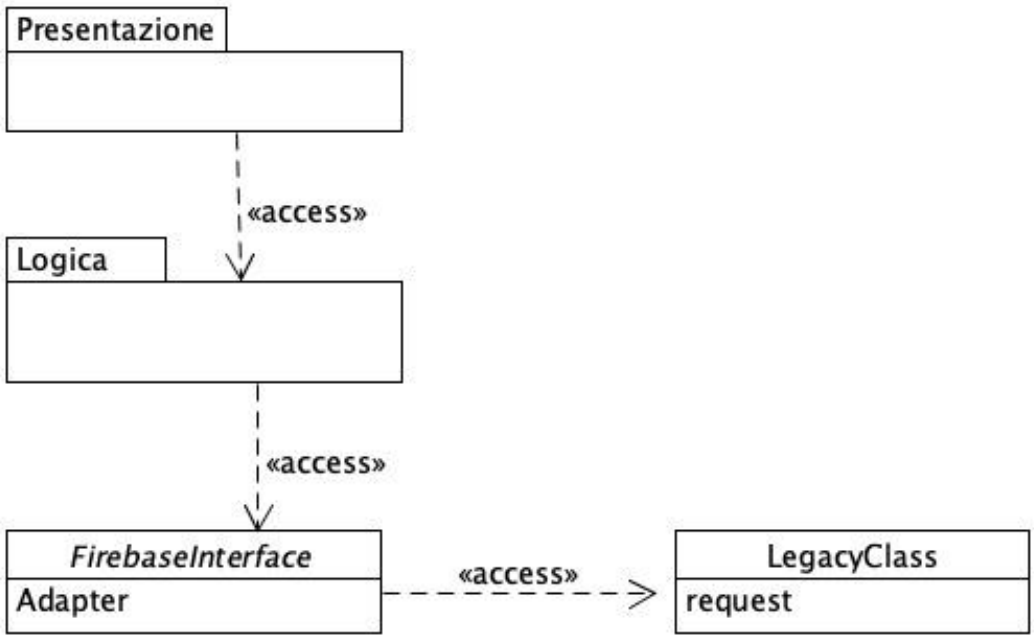
1.3.5 Posizione

- Mettere le dichiarazioni all'inizio dei blocchi. Non aspettare di dichiarare le variabili al loro primo uso: può confondere il programmatore inesperto ed impedire la portabilità del codice dentro lo scope.
- Evitare dichiarazioni locali che nascondono dichiarazioni a più alto livello.

1.3.6 Parentesi

- A prescindere dalle istruzioni che seguono un IF è necessario riportare il blocco di istruzioni tra parentesi graffe.
- Ogni tag di apertura deve essere necessariamente seguito dall'apposito tag di chiusura.
- Per quanto riguarda l'inserimento di numeri o di valori costanti, si adotterà la convenzione di non usare una codifica fissa (hard coding), ma di associare sempre il valore ad una variabile o semplicemente definire una macro che può essere richiamata da eventi ed essere parametrizzata.

1.4 Design Pattern



1.5 Definizioni, Acronimi e abbreviazioni

Erasmus+: progetto gestito da UNISA per studiare all'estero.

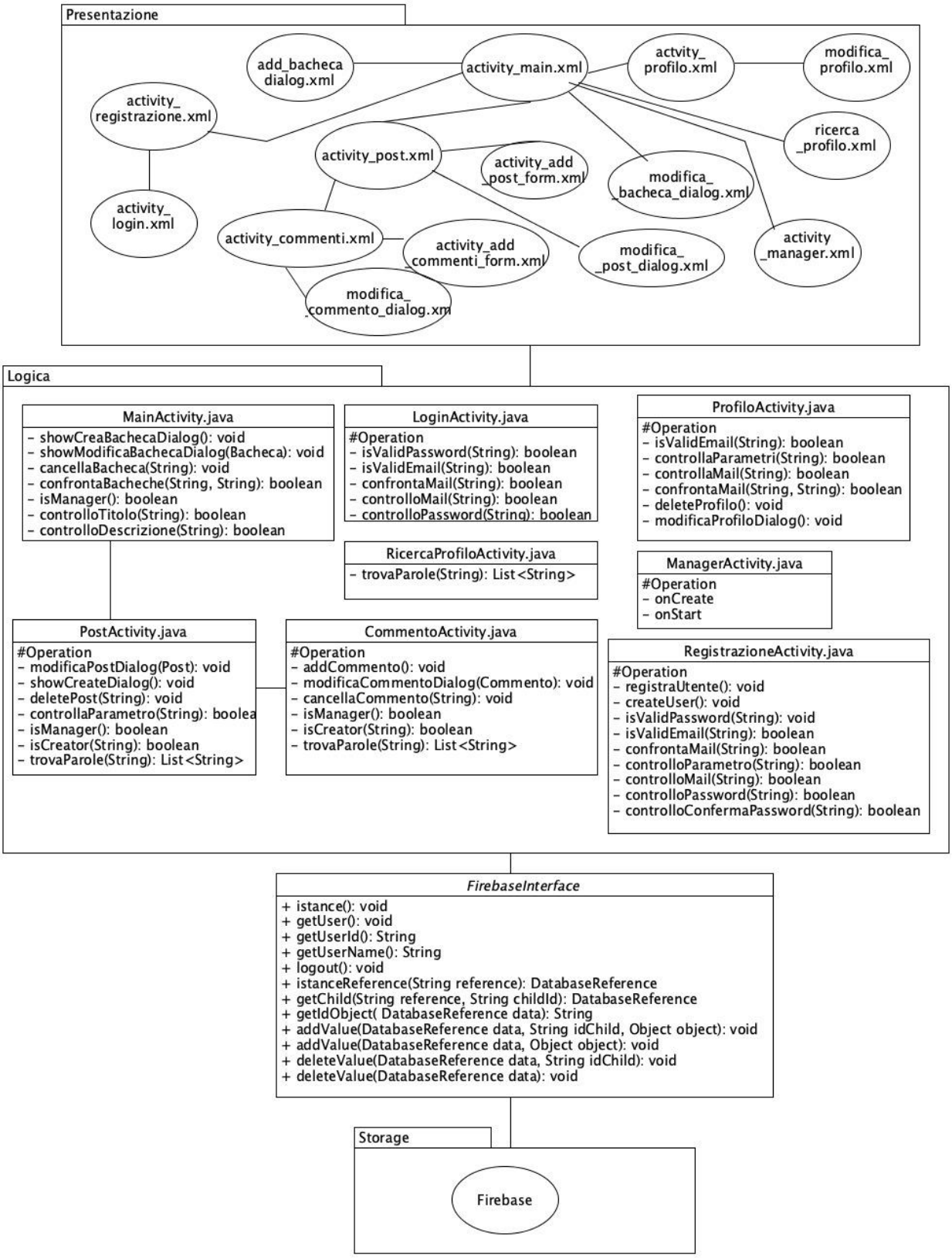
UNISA: Università degli studi di Salerno.

Firebase: è una piattaforma di sviluppo per web e mobile.

1.6 Riferimenti

- *B. BRUGGE, A.H. DUTOIT, OBJECT ORIENTED SOFTWARE ENGINEERING – USING UML, PATTERNS AND JAVA, PRENTICE HALL, 3D EDITION, 2009*
- *C. GHEZZI, D. MANDRIOLI, M. JAZAYERI, INGEGNERIA DEL SOFTWARE – FONDAMENTI E PRINCIPI, PRENTICE HALL, 2004*
- *SOMMERVILLE, SOFTWARE ENGINEERING, ADDISON WESLEY*
- *Materiale didattico reperibile su http://elearning.informatica.unisa.it/el-platform/ nella sezione del corso di Ingegneria del Software (professoressa Filomena Ferrucci)*

2. Packages


Presentazione
add_bacheca dialog.xml
activity_main.xml
actvity_ profilo.xml
modifica_ profilo.xml
activity_ registrazione.xml
ricerca _profilo.xml
activity_post.xml
activity_add _post_form.xml
modifica_ _bacheca_dialog.xml
activity_ login.xml
activity_commenti.xml
activity_add _commenti_form.xml
modifica_ _post_dialog.xml
activity _manager.xml
modifica_ _commento_dialog.xml
Logica
MainActivity.java
- showCreaBachecaDialog(): void
- showModificaBachecaDialog(Bacheca): void
- cancellaBacheca(String): void
- confrontaBacheche(String, String): boolean
- isManager(): boolean
- controlloTitolo(String): boolean
- controlloDescrizione(String): boolean
LoginActivity.java
#Operation
- isValidPassword(String): boolean
- isValidEmail(String): boolean
- confrontaMail(String): boolean
- controlloMail(String): boolean
- controlloPassword(String): boolean
ProfiloActivity.java
#Operation
- isValidEmail(String): boolean
- controllaParametri(String): boolean
- controllaMail(String): boolean
- confrontaMail(String, String): boolean
- deleteProfilo(): void
- modificaProfiloDialog(): void
RicercaProfiloActivity.java
- trovaParole(String): List<String>
ManagerActivity.java
#Operation
- onCreate
- onStart
PostActivity.java
#Operation
- modificaPostDialog(Post): void
- showCreateDialog(): void
- deletePost(String): void
- controllaParametro(String): boolea
- isManager(): boolean
- isCreator(String): boolean
- trovaParole(String): List<String>
CommentoActivity.java
#Operation
- addCommento(): void
- modificaCommentoDialog(Commento): void
- cancellaCommento(String): void
- isManager(): boolean
- isCreator(String): boolean
- trovaParole(String): List<String>
RegistrazioneActivity.java
#Operation
- registraUtente(): void
- createUser(): void
- isValidPassword(String): void
- isValidEmail(String): boolean
- confrontaMail(String): boolean
- controlloParametro(String): boolean
- controlloMail(String): boolean
- controlloPassword(String): boolean
- controlloConfermaPassword(String): boolean
FirebaseInterface
+ istance(): void
+ getUser(): void
+ getUserId(): String
+ getUserName(): String
+ logout(): void
+ istanceReference(String reference): DatabaseReference
+ getChild(String reference, String childId): DatabaseReference
+ getIdObject(DatabaseReference data): String
+ addValue(DatabaseReference data, String idChild, Object object): void
+ addValue(DatabaseReference data, Object object): void
+ deleteValue(DatabaseReference data, String idChild): void
+ deleteValue(DatabaseReference data): void
Storage
Firebase

3. Interfacce delle classi

Nome classe	Bacheca
Descrizione	Tale classe identifica una bacheca e mantiene le informazioni ed i post ad essa associate.
Pre-condizione	**context** Bacheca:: Bacheca(id, title, description, author, authorId, data); **pre:** title != null && description != null && id != null && author != null && authorId != null && data != null
Post-condizione	
Invarianti	**context** Bacheca **inv:** title != null && description != null && author != null && id != null && author != null && authorId != null && data != null

Nome classe	Post
Descrizione	Tale classe contiene le informazioni relative ad un intervento di un utente nel sistema e l'elenco dei commenti ad esso associati.
Pre-condizione	**context** Post:: Post(id, title, description, author, authorId, date); **pre:** title != null && description != null && author != null && id != null && author != null && authorId != null && data != null
Post-condizione	
Invarianti	**context** Post **inv:** title != null && description != null && author != null&& author != null && id != null && author != null && authorId != null && data != null

Nome classe	Commento
Descrizione	Tale classe rappresenta i commenti fatti dagli utenti in relazione ad uno specifico post.
Pre-condizione	**context** Commento:: Commento(id,description, author, authorId, date); **pre:** description != null&& author != null && id != null && author != null &&

	authorId != null && data != null
Post-condizione	
Invarianti	**context** Commento **inv** description != null && author != null && author != null && id != null && author != null && authorId != null && data != null

Nome classe	**Utente**
Descrizione	La classe rappresenta le informazioni fondamentali alla gestione dell'utente registrato nel sistema.
Pre-condizione	**context** Utente:: Utente(id, nome, cognome, sesso, dataDiNascita, email, password, ruolo); **pre**: email!=null && password!=null && id != null && nome != null && cognome != null && sesso != null && dataDiNascita != null
Post-condizione	
Invarianti	**context** Utente **inv:** email!=null && password!=null && id != null && nome != null && cognome != null && sesso != null && dataDiNascita != null