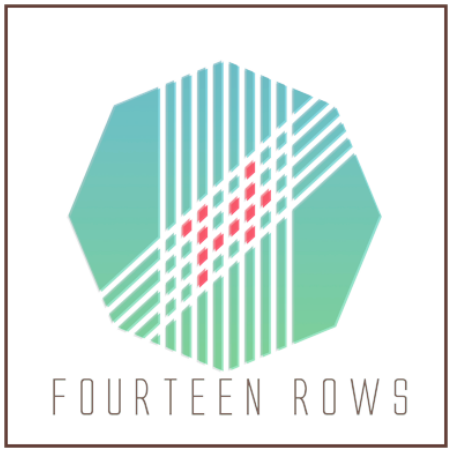
Fourteen Rows

Progetto P2PCS



Manuale dello Sviluppatore

Versione 1.0.0

Approvatore	**Bianchin Daniele**
Verificatori	**Romito Sara** **Brunetta Alessio**
Redattori	**Lanza Davide** **Munaro Cristiano** **Sartori Elisa**
Uso	**Esterno**
Destinatari	**Prof. Vardanega Tullio** **Prof. Cardin Riccardo** **Fourteen Rows**

Diario delle modifiche

Versione	Data	Descrizione	Nominativo	Ruolo
1.0.0	2019-07-02	Approvazione versione finale RA	Bianchin Daniele	*Approvatore*
0.9.2	2019-07-01	Verifica lessicale, grammaticale, logica	Brunetta Alessio Romito Sara	*Verificatore*
0.9.1	2019-06-27	Modifica diagrammi ed aggiunta elementi mancanti	Lanza Davide Brunetta Alessio Romito Sara	*Redattore*
0.9.0	2019-06-07	Approvazione versione finale RQ	Bianchin Daniele	*Approvatore*
0.1.0	2019-06-05	Verifica lessicale, grammaticale, logica	Sartori Elisa Romito Sara	*Verificatore*
0.0.5	2019-06-03	Stesura §5 Estensione delle funzionalità	Lanza Davide	*Redattore*
0.0.4	2019-06-02	Stesura §4 Framework e librerie esterne	Lanza Davide	*Redattore*
0.0.3	2019-06-01	Stesura §3 Architettura del software	Munaro Cristiano	*Redattore*
0.0.2	2019-05-30	Stesura §1 Introduzione e §2 Requisiti per lo sviluppo	Lanza Davide	*Redattore*
0.0.1	2019-05-30	Stesura del template e dello scheletro del documento	Lanza Davide	*Redattore*

Indice

Elenco delle figure

1 Introduzione

1.1 Scopo del documento

Lo scopo di questo documento è fornire le informazioni e le indicazioni necessarie per estendere, correggere e migliorare P2PCS, l'applicazione sviluppata dal gruppo Fourteen Rows. In questo documento si trovano indicazioni riguardo:

- Le tecnologie usate dagli sviluppatori;
- Il *software*$_{\text{G}}$ e gli strumenti usati;
- L'architettura *software*$_{\text{G}}$;
- I diagrammi della struttura del prodotto;
- L'estensione delle funzionalità.

1.2 Scopo del prodotto

Lo scopo del prodotto è lo sviluppo della struttura di un'applicazione *Android*$_{\text{G}}$ di *car sharing*$_{\text{G}}$ per l'azienda GaiaGo che implementi almeno cinque *game mechanics*$_{\text{G}}$ del *framework*$_{\text{G}}$ *Octalysis*$_{\text{G}}$ per implementare il concetto della *Gamification*$_{\text{G}}$.

1.3 Glossario

Per chiarire eventuali termini che potrebbero suscitare ambiguità è stato inserito in appendice un *Glossario* contenente tutti i termini necessari per la comprensione di questo documento. Le parole con un riferimento al *Glossario* verranno sempre indicate con una G a pedice. Esempio: *parola*$_{\text{G}}$ Se una parola appartente al *Glossario* è presente nel titolo di una sezione o sottosezione, questa non viene indicata con la G a pedice.

2 Riferimenti

- **Normativi**:
  - **Capitolato**: www.math.unipd.it/~tullio/IS-1/2018/Progetto/C5.pdf;
  - **ISO/IEC 12207:1995**: www.iso.org/standard/21208.html;
  - **ISO 8601**: www.iso.org/iso-8601-date-and-time-format.html;
  - *Norme di Progetto - v1.0.0.*
- **Informativi**:
  - *Piano di Progetto - v1.0.0*;
  - *Piano di Qualifica - v1.0.0.*

3 Requisiti per lo sviluppo

In questa sezione sono illustrati i requisiti *hardware*$_G$ e *software*$_G$ che gli sviluppatori dovranno soddisfare per poter attuare modifiche al prodotto.

3.1 Ambiente di sviluppo

Android Studio rappresenta l'*IDE*$_G$ scelto per lo sviluppo del prodotto. È basato su IntelliJ IDEA, un *IDE*$_G$ ricco di strumenti atti all'analizzare, configurare ed effettuare il debug software senza dover installare componenti esterne. IntelliSense aiuta inoltre a scrivere codice corretto rapidamente. Android Studio viene utilizzato per la stesura del codice in *Kotlin*$_G$ e dei *layout*$_G$ in *XML*$_G$.

È reperibile al seguente sito: https://developer.android.com/studio ed è disponibile per i seguenti sistemi operativi:

- Windows 7 o superiore;
- Mac OS X 10.10 o superiore;
- Linux;
- Chrome OS.

Per i sistemi operativi basati su Linux, è richiesta la libreria GNU C (glibc) 2.19 o superiore.

I requisiti *hardware*$_G$ sono:

- **RAM**: minimo 4GB, raccomandati 8GB;
- **Hard drive**: minimo 2GB, raccomandati 4GB;
- **Risoluzione**: minima 1280 x 800.

Per i sistemi operativi Chrome OS, è richiesto anche una CPU Intel i5 o superiore.

3.2 Database

Firebase$_G$ rappresenta il database scelto per lo sviluppo del prodotto. È accessibile via internet per mezzo di un browser. Le versioni più recenti supportate sono:

- **Google Chrome**: dalla versione 68
- **Firefox**: dalla versione 64

La documentazione ufficiale è disponibile al seguente link: https://firebase.google.com/docs/guides.

3.3 Altro

Sono necessari anche i seguenti strumenti:

- **Android Oreo 8.0**: per poter eseguire correttamente P2PCS è necessario essere a disposizione di uno smartphone Android con una versione del sistema operativo maggiore o uguale a 8.0;
- **Git**: P2PCS risiede su un *repository*$_G$ sulla piattaforma *GitHub*$_G$;

È necessaria la conoscenza dei seguenti linguaggi;

- **Kotlin**: linguaggio di programmazione utilizzato per la stesura del corpo del prodotto. La documentazione ufficiale è disponibile al seguente link: https://kotlinlang.org/docs/reference/android-overview.html;
- **XML**: linguaggio di markup utilizzato per la stesura dei *layout*$_G$ del prodotto. La documentazione è disponibile al seguente link: https://developer.android.com/guide/topics/ui/declaring-layout.

3.4 Installazione dell'ambiente di sviluppo

Presupponendo che l'utente abbia già a disposizione un sistema operativo coerente con quanto descritto sopra, andremo ora a definire il processo con il quale uno sviluppatore può installare e configurare il proprio ambiente di sviluppo.

- Scaricare ed installare *Git*$_G$;
- Scaricare ed installare *Android Studio*$_G$;
- Al primo avvio di *Android Studio*$_G$ verrà richiesto di personalizzare la propria configurazione;
- Accedere alle impostazioni da File -> Settings e navigare nelle impostazioni del *SDK*$_G$, qui spuntare, se non già spuntate, le caselle adiacenti a:
  - Android 8.0 (Oreo);
  - Android SDK Tools;
  - Android SDK Platform Tools;
  - Android SDK Build-Tools 29.
- Clonare tramite *Android Studio*$_G$ il repository, reperibile al seguente indirizzo: https://github.com/FourteenRows/P2PCS;
- Aprire il progetto appena clonato;
- Attendere che *Gradle*$_G$ termini lo scaricamento delle dipendenze e l'autoconfigurazione delle stesse.

Per eseguire l'applicazione è necessario inoltre essere in possesso di un dispositivo *Android*$_G$ o un emulatore con *Android*$_G$ 8.0 o successivo. Per usare un dispositivo fisico è obbligatorio abilitare la modalità di debug e connetterlo tramite cavo USB al computer. Per usare un emulatore invece abilitare dalle impostazioni dell'*SDK*$_G$ "Android Emulator" e configurarlo seguendo la procedura automatica con *Android*$_G$ 8.0.

4 Architettura del software

I vari path all'interno del progetto si possono dividere in quattro sezioni principali:

- **/app/src/main/java/com/fourteenrows/p2pcs/**: contenente il codice sorgente dell'applicativo;
- **/app/src/res/**: contenente i *layout*$_G$ e le risorse utilizzate, quali stringhe, colori, icone ecc;
- **/app/src/androidTest/java/com/fourteenrows/p2pcs/**: contenente i test UI dell'applicativo;
- **/app/src/test/java/com/fourteenrows/p2pcs/**: contenente i test di unità.

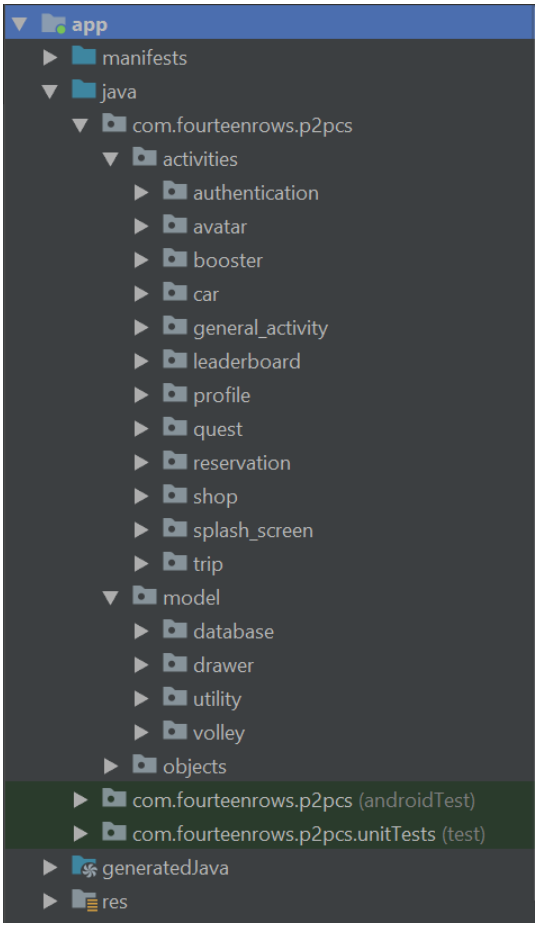


Figura 1: Struttura dei package

5 Pattern architetturale

Data la natura del prodotto, il gruppo ha scelto di adottare il pattern architetturale MVP. Questo, permette di realizzare efficacemente un prodotto mantenendo separati grafica e logica, facilitando così la creazione dei test, lo sviluppo delle varie componenti e l'estensione delle stesse. L'MVP si compone di tre parti:

- **Model**: contiene i dati che l'applicazione può elaborare. Mette inoltre a disposizione metodi per leggere e modificare tali dati. Il Model nell'applicazione P2PCS si trova all'interno del package "model" appunto. Si compone di varie parti:
  - **database**: contiene i metodi per il dialogo tra l'applicazione ed il database;
  - **drawer**: contiene l'istanza del menu dell'applicazione;
  - **utility**: contiene tre file principali:
    - **ModelDates**: insieme di metodi riguardanti operazioni sulle date;
    - **ModelUtils**: insieme di metodi a supporto dell'applicazione;
    - **ModelValidator**: insieme di metodi atti alla validazione dei dati.
  - **Volley**: libreria che permette l'esecuzione di richieste HTTP, utilizzata per l'interlacciamente tra l'applicazione e le API di Google Maps.
- **View**: rappresenta l'interfaccia utente. Ogni view (attività) disegna sul dispositivo una schermata che permette all'utente di interagire con l'applicazione; La view nell'applicazione P2PCS si trova:
  - All'interno del package "res" per quanto riguarda i layout, i colori, le stringhe, le icone ecc;
  - In ogni file che termina con la parola "Activity" all'interno del package activities.

  Si compone di varie parti:
  - **login**: attività che permette agli utenti di autenticarsi all'interno dell'app, resettare la propria password e richiedere una nuova registrazione;
  - **registrazione**: attività che permette ad un nuovo utente di registrarsi;
  - **avatar**: attività che permette all'utente la personalizzazione del proprio avatar;
  - **booster**: attività che permette all'utente l'attivazione dei booster posseduti;
  - **car**: attività che permette all'utente la gestione delle proprie auto;
  - **general**: attività base dalla quale tutte le altre derivano, contenente metodi utilizzati in tutte le schermate, quali l'inizializzazione del menu, la visualizzazione di un *alert*$_{\text{G}}$, ecc;
  - **leaderboard**: attività che permette all'utente la visualizzazione della classifica degli utenti;
  - **profile**: attività che permette all'utente di gestire il proprio profilo;
  - **quest**: attività che permette all'utente di gestire le proprie missioni attive;
  - **reservation**: attività che permette all'utente di gestire le proprie prenotazioni;
  - **shop**: attività che permette all'utente di acquistare degli oggetti;

- **splash_screen**: la prima attività che viene visualizzata, rappresenta la schermata di caricamento dell'applicazione;
- **trip**: attività che permette all'utente di gestire i propri viaggi.

- **Presenter**: è il mediatore ed elaboratore dei dati tra la View ed il Model. Esso si occupa di gestire l'input dell'utente, scegliere la strategia corretta per elaborarlo, visualizzare gli output dei processi ed interagire con il Model. Il Presenter nell'applicazione P2PCS si trova in ogni file che termina con la parola "Presenter" all'interno del package activities.

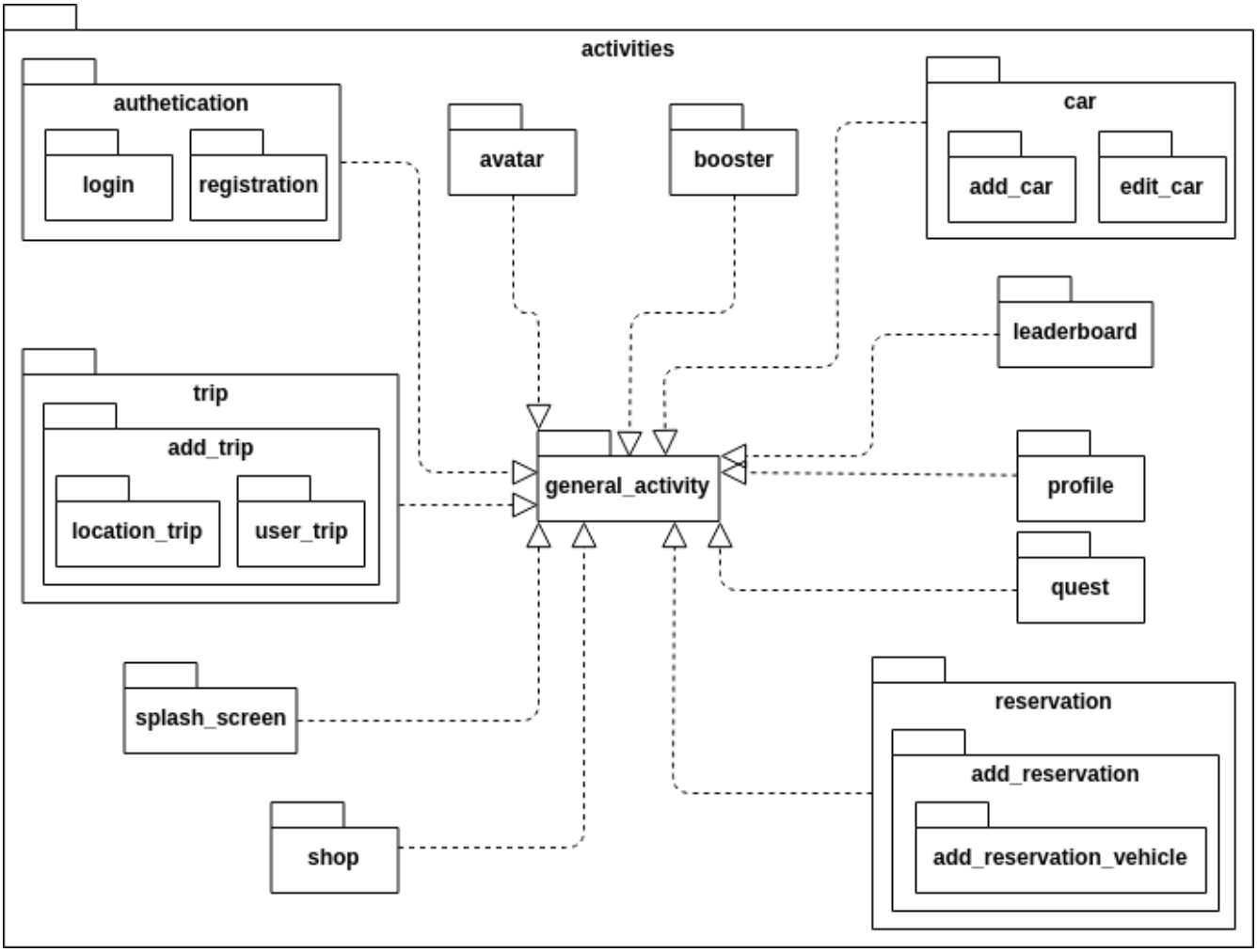


Figura 2: Diagramma dei package delle attività

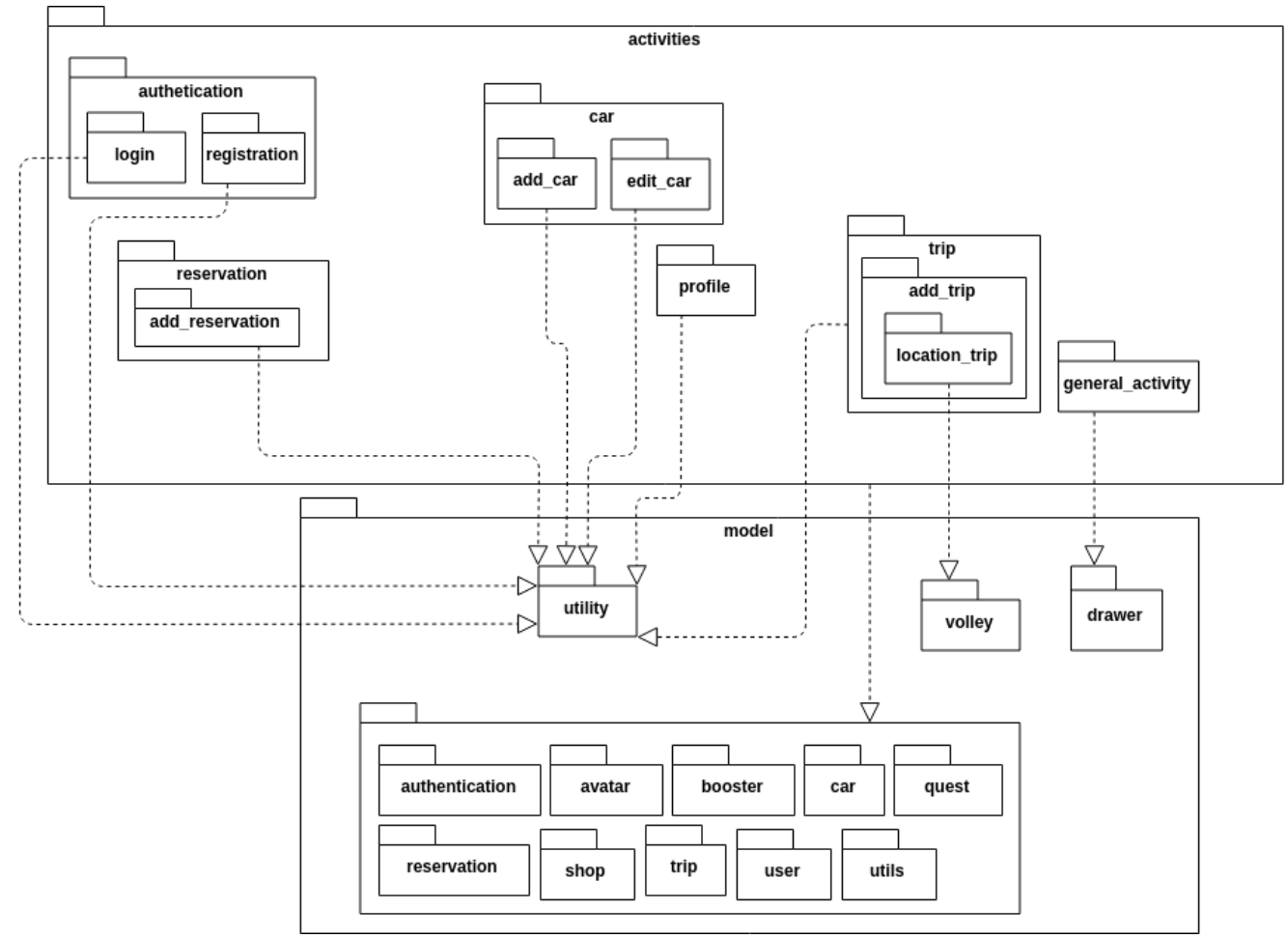


Figura 3: Diagramma del collegamento tra i package delle attività e quelli del model

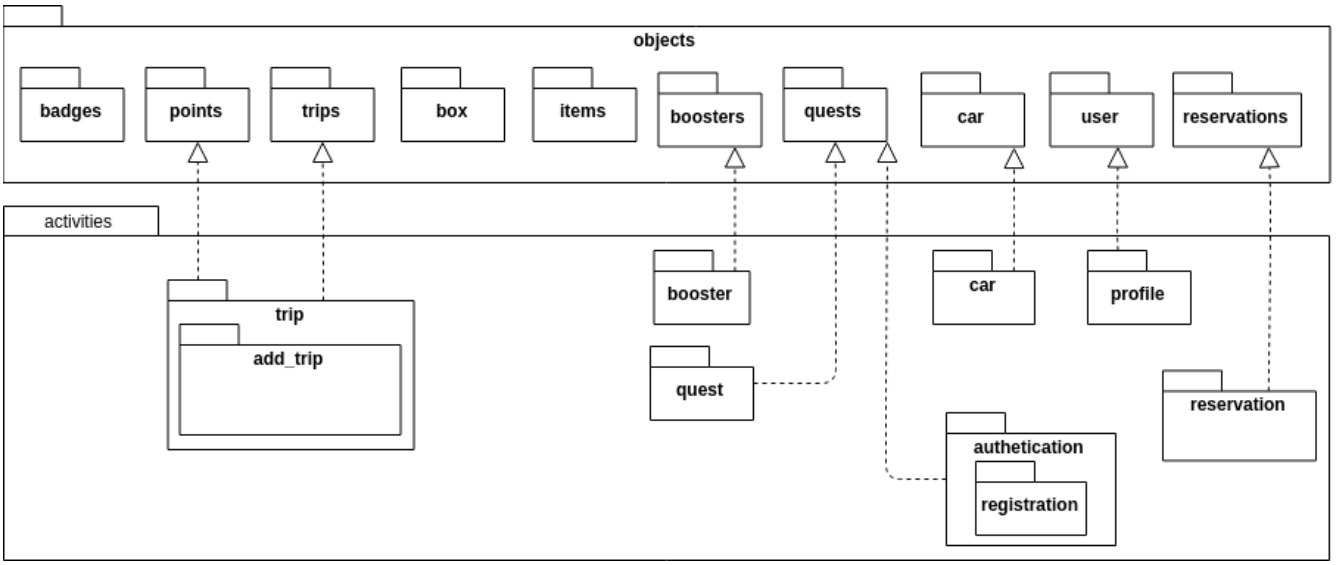


Figura 4: Diagramma del collegamento tra i package delle attività e quelli degli objects

5.1 Esempio attività

Viene qui descritta un attività di esempio: Profile.

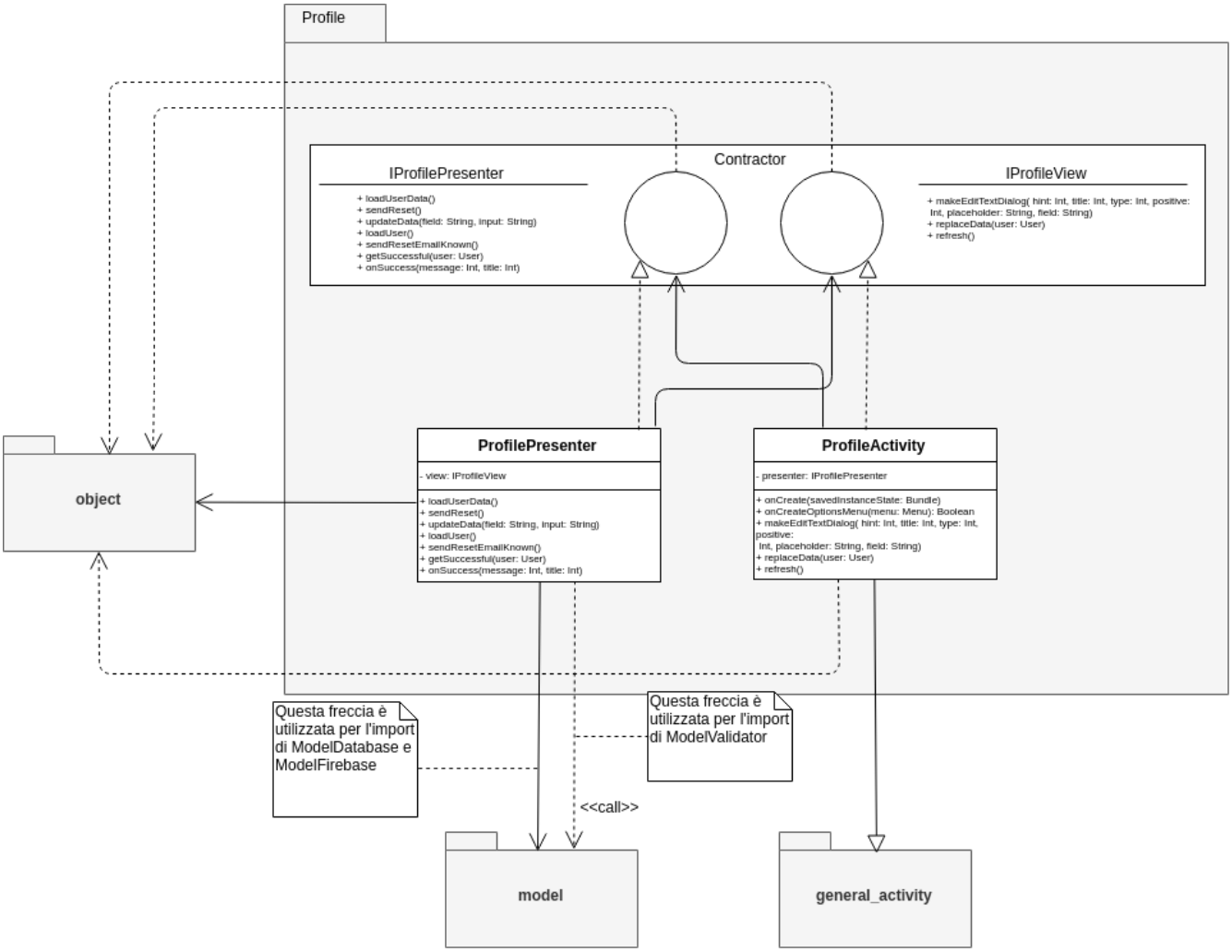


Figura 5: UML profile

In Figura 3 si vede come l'attività estenda la classe astratta GeneralActivity, un attività base con metodi ricorrenti in tutte le attività del prodotto.

La View in questo caso è ProfileActivity che implementa l'interfaccia IProfileView, all'avvio della attività viene creata un'istanza di ProfilePresenter con un riferimento a ProfileActivity di tipo IProfileView, consentendo la comunicazione tra Presenter e View.

Allo stesso modo, alla creazione di ProfilePresenter viene istanziato ModelFirebase per permettere al Presenter di comunicare con esso.

Alla pressione di un bottone, ProfileActivity notifica a ProfilePresenter l'evento, così il Presenter decide come agire.

A questo punto, nel caso ce ne sia bisogno, ProfilePresenter comunica con il Model per modificare lo stato del Database e se necessario, il Presenter comunica alla View il nuovo stato dei dati, richiedendo un aggiornamento di quanto mostrato all'utente.

6 Framework e librerie esterne

In questa sezione verranno descritte tutte le tecnologie esterne che sono state scelte e implementate nello sviluppo del prodotto.

6.1 Codifica

- **Android GIF Drawable**: libreria utilizzata per poter visualizzare la GIF presente nella splash screen. Per ulteriori informazioni: https://github.com/koral--/android-gif-drawable;
- **Directions API**: libreria utilizzata per calcolare la distanza a livello stradale tra i due punti inseriti durante la fase di inserimento di un nuovo viaggio. Per ulteriori informazioni: https://developers.google.com/maps/documentation/directions/start;
- **Material Drawer**: libreria utilizzata per avere un drawer persistente, evitando duplicazioni di codice. Per ulteriori informazioni: https://github.com/mikepenz/MaterialDrawer;
- **Places API**: libreria utilizzata per poter trovare punti il cui nome è uguale o analogo a quello inserito dall'utente durante la fase di inserimento di un punto di arrivo o di partenza di un viaggio. Per ulteriori informazioni: https://developers.google.com/places/web-service/intro.

6.2 Test

- **Espresso**: libreria utilizzata per la stesura di test UI: https://developer.android.com/training/testing/espresso;
- **JUnit**: libreria utilizzata per la stesura di test di unità sui metodi del prodotto: https://junit.org/junit4/;

7 Estensione delle funzionalità

Nel caso in cui si volesse estendere l'applicazione P2PCS, i passi da svolgere sono pochi e semplici:

- Creare un nuovo package con il nome della nuova sezione da aggiungere. Questo passo non è obbligatorio ma consigliato;
- Creare un'interfaccia per la View, una per il Presenter ed una per un CompleteListener, nel caso vi sia necessità. Questo passo non è obbligatorio ma consigliato;
- Creare un file per la View e uno per il Presenter. Il file della View deve ereditare da GeneralActivity e contenere un'istanza del Presenter, mentre quest'ultimo deve contenere un'istanza di View e Model;
- Aggiungere la View appena creata al file AndroidManifest.xml e in un punto desiderato nel metodo loadActivityLinks, seguendo l'impostazione delle altre, nel file DrawerSingleton.kt, in modo che la nuova schermata sia presente nel menu del drawer;
- All'interno dell'onCreate, chiamare il metodo initializeDrawer().

Ora si è liberi di aggiungere qualsiasi tipo di contenuto si voglia all'applicazione P2PCS.

A Glossario

A

- **Alert**: messaggio d'errore o di avvertimento che viene visualizzato quando l'utente compie un'azione imprevista;
- **Android**: sistema operativo per dispositivi mobili sviluppato da Google.

C

- **Car sharing**: servizio di condivisione di auto che permette agli utilizzatori di guadagnare dalla propria auto che risulterebbe, altrimenti, inutilizzata.

F

- **Framework**: architettura logica di supporto con cui un software può essere progettato e realizzato, spesso facilitandone lo sviluppo.

G

- **Game Mechanichs**: meccaniche di gioco facenti parte del *framework*$_{\text{G}}$ *Octalysis*$_{\text{G}}$;
- **Gamification**: applicazione dei principi di game-design in un contesto dove l'applicazione da sviluppare non è un videogame.

L

- **Layout**: la disposizione sulla pagina degli elementi che costituiscono un bozzetto.

O

- **Octalysis**: *framework*$_{\text{G}}$ utilizzato per lo sviluppo della *Gamification*$_{\text{G}}$.

S

- **Software**: insieme di processi che possono essere impiegati in un sistema di elaborazione dei dati.