# MicroAppEngine

MicroApp è un’applicazione per sistemi operativi Android che permette di progettare ed eseguire delle micro applicazioni personalizzate. Tramite un’editor grafico, l’utente può comporre come un puzzle l’applicazione desiderata, la quale sarà poi costruita e lanciata in esecuzione dall’engine di MicroApp. La composizione di una microapp fa affidamento sul concetto di componente: questa rappresenta una singola funzionalità tipicamente offerta dal sistema, come chiamare un contatto, scattare una foto o inviare una mail; inoltre, per svolgere la propria funzione, una componente può richiedere dei dati in input, elaborarli e generare dei nuovi dati in output. Una volta terminata la composizione visuale della microapp, l’engine raccoglie le informazioni relative alle componenti utilizzate e al flusso di dati che intercorre tra loro, per poi occuparsi di ordinarle in maniera tale che ciascuna componente abbia a disposizione tutti i dati di cui ha bisogno prima di essere lanciata.

L’obiettivo del presente progetto è quello di ristrutturare l’engine di MicroApp, rappresentato dal software MicroAppEngine. Questo, oltre ad eseguire le funzioni appena descritte, è stato esteso nel tempo con tante funzionalità differenti, come l’inserimento di servizi web, la gestione dei flussi di controllo nell’esecuzione delle componenti e la composizione di interfacce grafiche personalizzate. Nell’ambito della presente ristrutturazione, tuttavia, sono state prese in considerazione soltanto le entità principali del software, opportunamente isolate delle estensioni a cui l’engine è stato sottoposto nel tempo. Pertanto, la versione di MicroAppEngine presa in esame corrisponde a quella realizzata nella sua versione iniziale.

## Analisi dei requisiti

La progettazione iniziale di MicroAppEngine comprende i seguenti requisiti funzionali:

* *Scelta del file di configurazione*: è necessario permettere all’utente di scegliere quale microapp eseguire tra quelle realizzate in precedenza tramite l’editor; la scelta può essere effettuata selezionando uno dei file di deploy tra quelli disponibili, ognuno dei quali contiene informazioni sul flusso di esecuzione delle proprie componenti e sui dati scambiati tra loro; il file di deploy, in formato xml, deve essere sottoposto ad un controllo per verificarne la correttezza.
* *Avvio dell’applicazione*: il file scelto deve poter essere avviato solo su esplicita azione dell’utente che, al termine dell’esecuzione della microapp, può eventualmente selezionare un nuovo file di deploy dall’elenco per avviare una nuova microapp.
* *Vincoli sul flusso di esecuzione*: le componenti derivate dal file di deploy devono essere ordinate in maniera sequenziale in maniera tale che la componente posta in alto a sinistra nell’editor risulti quella iniziale e ogni componente, prima di essere lanciata, possa aver ricevuto tutti gli input di cui aveva bisogno.
* *Funzionamento delle componenti*: ogni componente deve offrire una determinata funzionalità, la quale può variare in funzione dei dati ricevuti in input dalle altre componenti; di conseguenza, ciascuna componente può comportarsi in maniera differente e può decidere di interagire con l’utente, visualizzare semplicemente dei dati, essere eseguita in background oppure interagire con il sistema intercettandone gli eventi; i dati eventualmente generati dall’esecuzione della componente devono essere correttamente trasmessi alle componenti destinatarie, così come definito dal flusso di esecuzione.
* *Navigabilità della microapp*: in qualunque punto dell’esecuzione, deve essere offerta all’utente sia la possibilità di procedere alla componente successiva che quella di tornare alle componenti eseguite in precedenza, reimpostando eventualmente tutti i dati necessari e annullando o sovrascrivendo le operazioni effettuate.
* *Tipi di dato e molteplicità*: ciascuna microapp deve poter trattare i seguenti tipi di dato:
  + stringa, per descrivere una sequenza di caratteri;
  + immagine, per descrivere un’immagine acquisita da una qualsiasi fonte;
  + contatto, per descrivere l’informazione riguardante una persona;
  + posizione, per descrivere le coordinate geografiche relative a un dato luogo;

inoltre, i dati prodotti da una componente possono essere di tipo singolo, se consistono di una sola istanza, o di tipo multiplo, se consistono di un numero arbitrario di istanze; di conseguenza, le componenti devono essere in grado di poter gestire entrambe le tipologie di dati.

* *Tipi di componenti*: l’applicazione deve prevedere i seguenti tipi di componenti:
  + SelectContact, per selezionare un contatto dalla rubrica;
  + PreviewContact, per visualizzare i dettagli di un contatto;
  + Call, per chiamare un contatto;
  + SendMessage, per inviare un sms a un contatto;
  + Send, per inviare una mail a un contatto;
  + Take, per scattare una foto;
  + Location, per raccogliere e visualizzare la posizione corrente;
  + Map, per visualizzare una posizione sulla mappa.
* *Estendibilità dell’applicazione*: per favorire l’aggiunta di nuove componenti e di nuovi tipi di dati, deve essere possibile estendibile l’applicazione nella maniera più semplice e naturale possibile per gli sviluppatori.

## Entità

Il sistema è composto dalle seguenti entità fondamentali:

* *MAActivity*, che rappresenta le schermate che si presentano all’utente permettendo a quest’ultimo di interagire con il sistema e definisce la logica di base delle schermate delle diverse componenti, compresi i pulsanti di navigazione Prev e Next;
* *MAComponent*, che descrive una generica componente, contiene le informazioni acquisite dal file di deploy e viene specializzata in modo da descrivere informazioni riguardanti le specifiche componenti;
* *MicroAppGenerator*, che mantiene le componenti ricavate dal file di deploy e gestisce il flusso di dati durante l’esecuzione, dopo aver delegato la loro creazione e il loro ordinamento alle altre entità preposte, come *DeployParser*, *ComponentCreator* e *DepthFirstOrder*.
* *GenericData*, che modella uno o più dati di un certo tipo e li incapsula in apposite strutture dati come *DataCollection* e *DataContainer*, così da rendere possibile lo scambio di dati tra le componenti.

# Reverse Engineering

Al fine di realizzare una ristrutturazione su MicroAppEngine, si è proceduto in primo luogo ad effettuare delle operazioni di reengineering sul sistema al fine di ricavarne dei diagrammi che fossero d’aiuto per la comprensione del software.

## Diagramma delle classi

Di seguito viene illustrato il diagramma delle classi di MicroAppEngine.

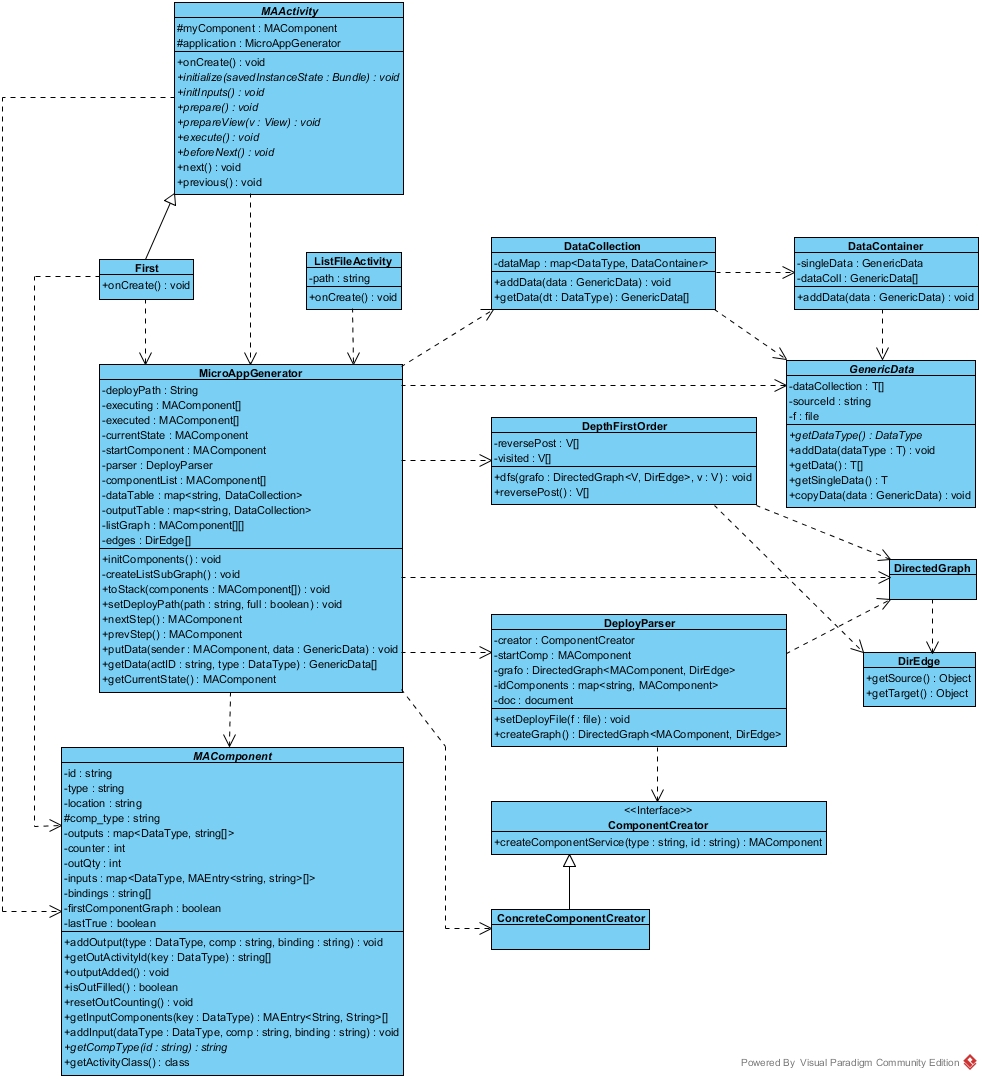


Figura Diagramma delle classi di MicroAppEngine

Le classi MAActivity e MAComponent vengono estese da diverse sottoclassi, corrispondenti ai tipi di componenti previsti. Queste sono illustrate di seguito.

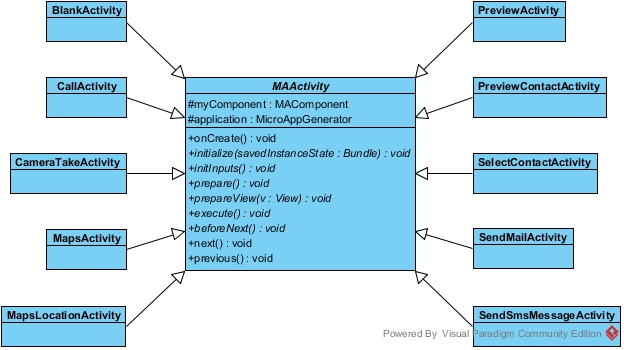


Figura Diagramma delle sottoclassi di MAActivity

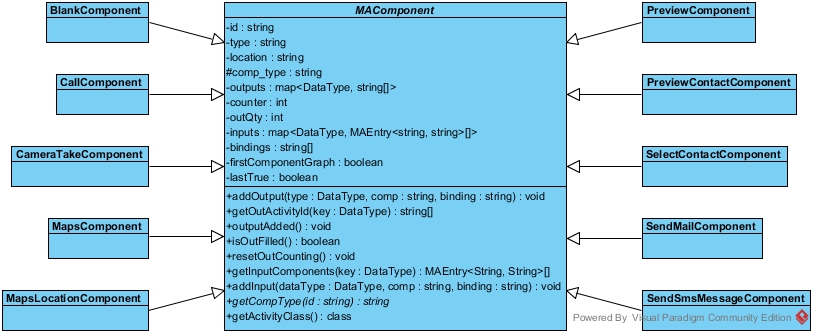


Figura Diagramma delle sottoclassi di MAComponent

## Sequence Diagram

Di seguito viene illustrato il diagramma di sequenza di una tipica esecuzione di MicroAppEngine; in particolare, l’utente avvia MicroApp e, selezionando uno dei file di deploy disponibili, lancia una microapp che consiste di due componenti di tipo Blank.

