# Assignment #3: Relazione

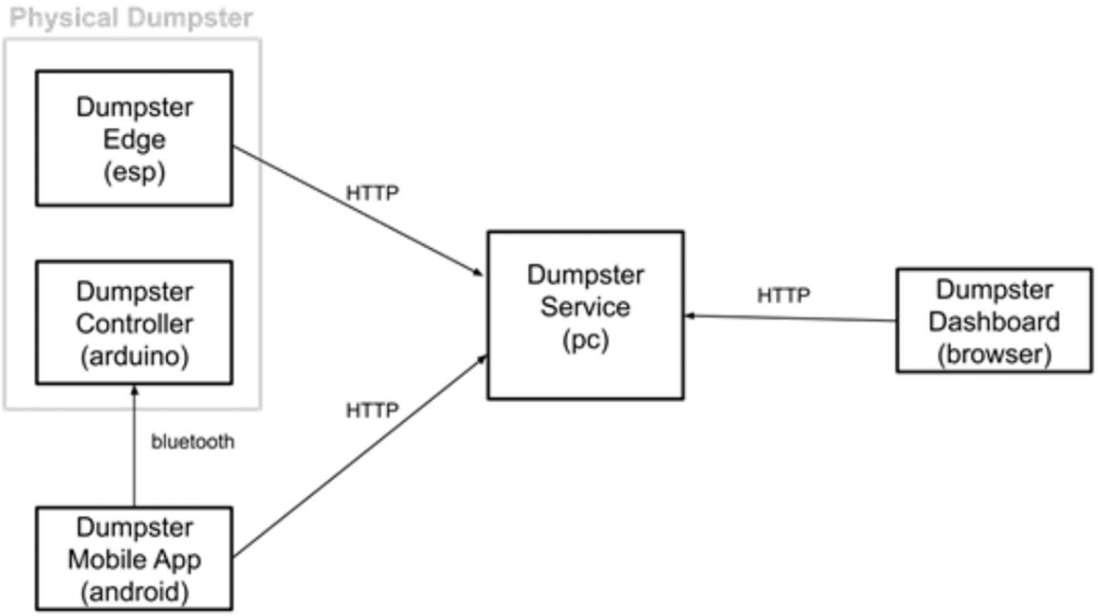
CORSO DI SISTEMI EMBEDDED E IOT

GIACHIN MARCO, BAIARDI MARTINA, LOMBARDINI ALESSANDRO

# Sommario

# 1. Introduzione

Si vuole realizzare un sistema IoT che implementi una versione semplificata di uno *smart dumpster*, ovvero un cassonetto dei rifiuti "intelligente". Il sistema complessivamente è costituito da 5 sotto-sistemi:




# 2. Dumpster Controller

## 2.1 Introduzione

Il Dumpster Controller (DC) è il sistema embedded che controlla il cassonetto e interagisce, mediante tecnologia bluetooth, con la Mobile App (MA) del sistema. Un utente munito di tale applicazione ha la possibilità di recarsi presso il DC ed effettuare un deposito di rifiuti. Al momento della richiesta di deposito il DC apre il cassonetto, illumina il led relativo al tipo di rifiuto selezionato dall'utente e chiude il cassonetto nel momento in cui scade il tempo concesso per il deposito.

I messaggi bluetooth che possono essere ricevuti dal DC da parte della MA sono di tre tipi:

1. *typeOfTrash*, inviato per richiedere l'inizio di un nuovo deposito. Questo messaggio contiene, in particolare:
    I. Il token
    II. Il tipo di rifiuto scelto dall'utente
2. *requestOfTime*, inviato per richiedere più tempo per depositare il rifiuto.
3. *finishDeposit*, inviato per richiedere che il deposito venga terminato (prima che il tempo a disposizione dell'utente si esaurisca).

## 2.2 Implementazione

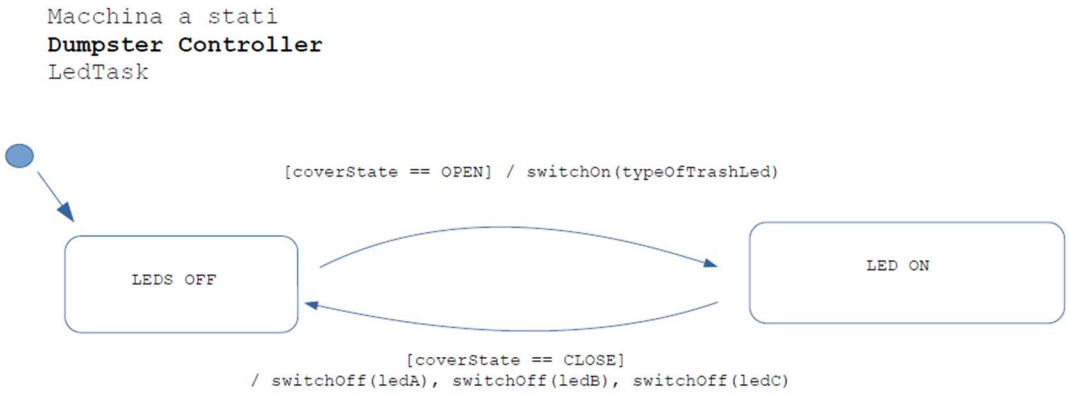
Il DC è implementato con architettura a task sincroni. Per mantenere coordinati i task è presente un oggetto condiviso chiamato *State* che mantiene lo stato attuale del DC. La classe *State* mantiene le seguenti variabili:

- *coverStatus*, stato dello sportello del DC (aperto o chiuso).
- *typeOfTrash*, tipo di rifiuto che l'utente desidera depositare.
- *timeStartCountDown*, istante di tempo in cui parte il conto alla rovescia per chiudere il portello. Il suo valore viene scritto nel momento in cui il DC riceve il messaggio inerente al tipo di rifiuto che l'utente vuole depositare.
- *countDownValue*, tempo per cui lo sportello deve rimanere aperto; comprende la somma del tempo concesso inizialmente e di tutte proroghe effettuate successivamente.
- *timeExpired*, variabile booleana che permette di sapere se il countdown è scaduto.
- *TEMPO_CONCESSO_A_PRESCINDERE*, costante di valore 90 che rappresenta la quantità di tempo concesso quando viene richiesto un deposito.
- *TEMPO_AGGIUNTO_SE_RICHIESTO*, costante di valore 30 che rappresenta la quantità di tempo che viene aggiunto quando l'utente richiede più tempo

Di seguito vengono analizzati i singoli Task.

### 2.2.1 Led Task

Gestisce l'accensione e lo spegnimento dei led del DC. Ad ogni tick, questo task si occupa di controllare in quale stato è lo sportello del cassonetto; se è aperto significa che è stata effettuata una richiesta di deposito e deve quindi essere illuminato il led relativo al rifiuto selezionato. Se è chiuso tutti i led devono rimanere spenti.




### 2.2.2 Bluetooth Task

Consente l'invio e la ricezione di messaggi attraverso il modulo Bluetooth, tramite la libreria *SoftwareSerial.h* (il cui scopo è quello di emulare via software una comunicazione seriale su due pin differenti da quelli standard, ovvero PIN0 e PIN1)

Ad ogni tick il task controlla se ci sono dei messaggi arrivati sulla seriale prelevandoli e processandoli. Controlla inoltre la variabile *timeExpired*, la quale indica se il tempo concesso all'utente per depositare il rifiuto è terminato. In caso affermativo invia un messaggio alla MA per indicare che il deposito è concluso.

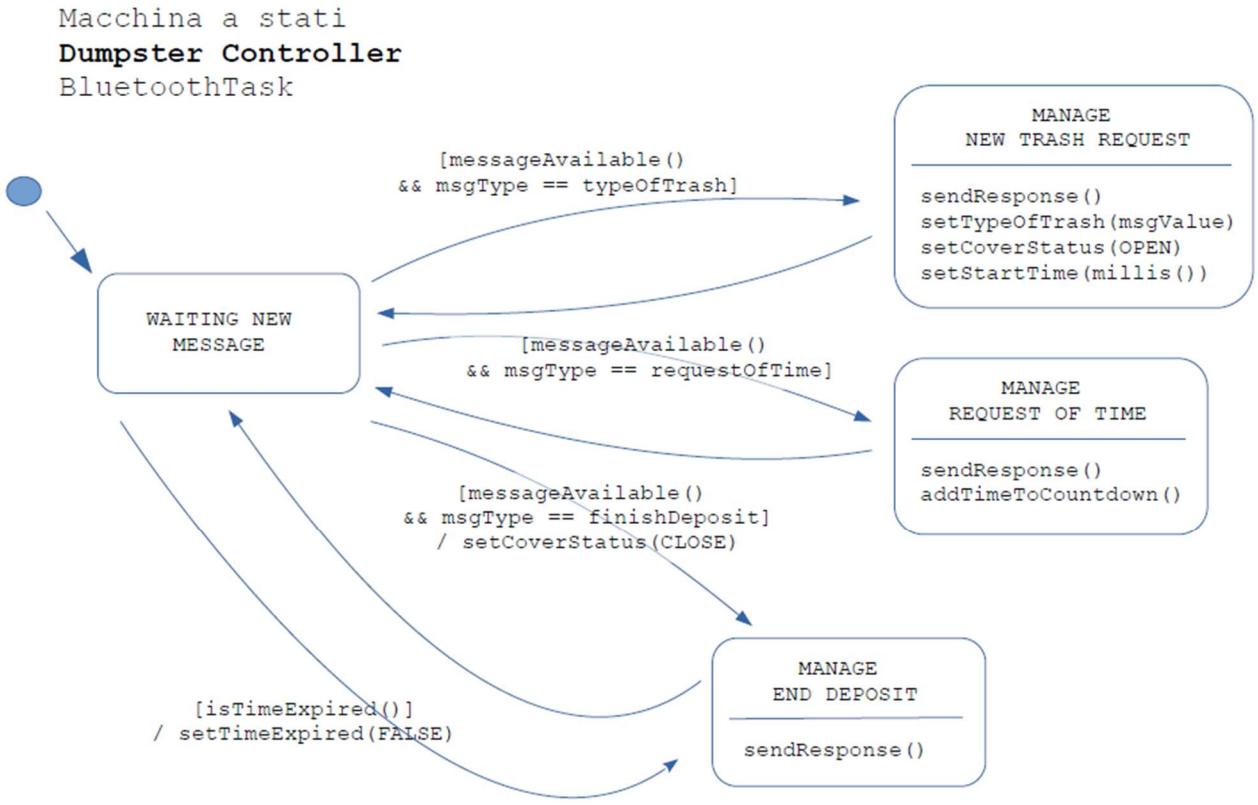
Quando il task controlla il contenuto del messaggio, prima di tutto verifica quale tipo di messaggio è arrivato:

- Se di tipo *typeOfTrash*:
    a. effettua un controllo sul token
    b. controlla quale tipologia di rifiuto l'utente vuole depositare
    c. salva un timestamp per indicare l'istante di ricezione della richiesta
    d. indica che lo sportello deve essere aperto
    e. invia un messaggio di risposta per confermare all'utente che la richiesta è stata presa in carico

- Se di tipo *requestOfTime*:
    a. aggiunge la costante *TEMPO_AGGIUNTO_SE_RICHIESTO* al countdown
    b. invia una risposta all'utente per comunicargli la buona riuscita dell'operazione

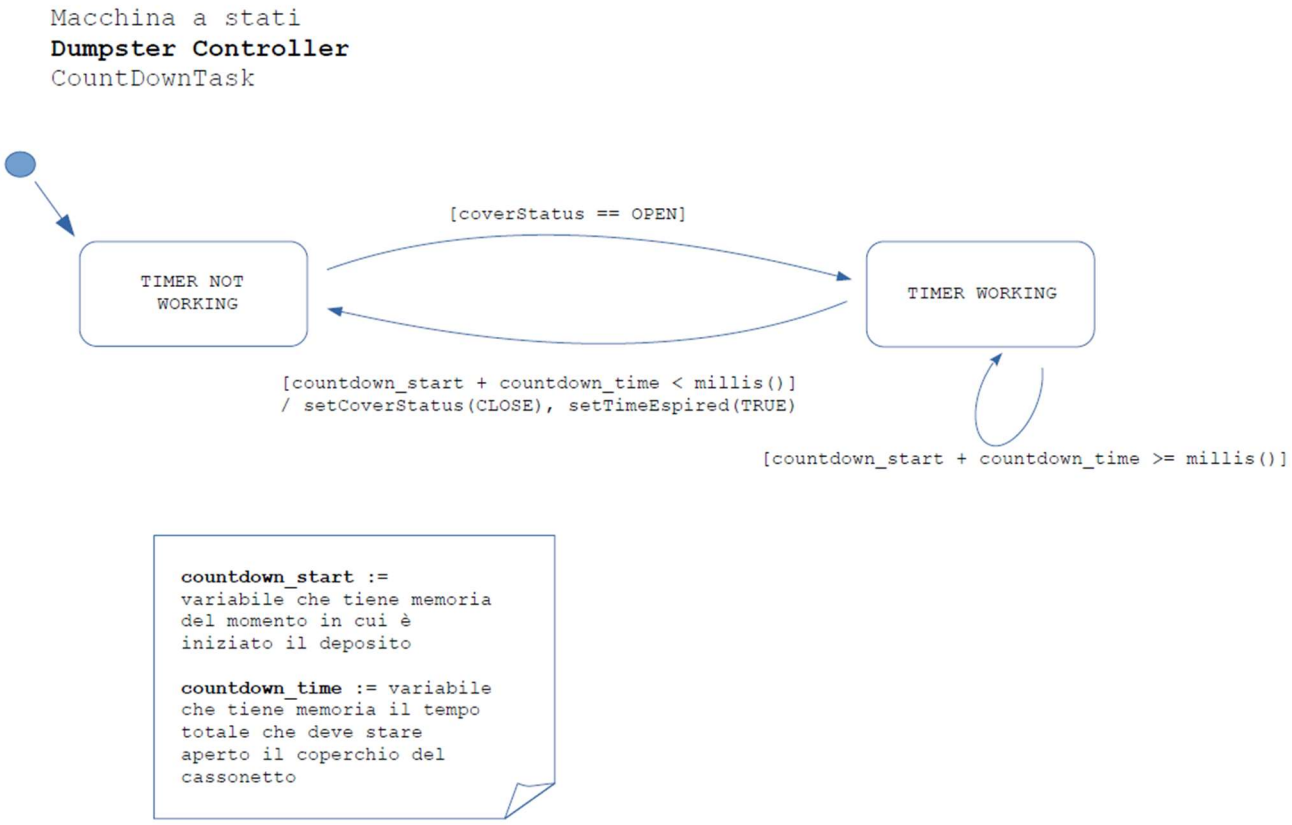
Nella nostra implementazione l'utente non decide liberamente il tempo concessogli per depositare il rifiuto, questo viene assegnato tramite una costante (da noi impostata a 90 secondi).
Ad esempio: un utente effettua una richiesta di deposito e, come valore standard, ha a propria disposizione 90s per completare il deposito. Se l'utente si rende conto che non sono sufficienti può effettuare una ulteriore richiesta affinché gli venga concesso più tempo; il DC aggiungerà 30s, permettendo allo sportello del cassonetto di rimanere aperto in totale 120s. Tale richiesta può essere fatta un numero illimitato di volte.

- Se di tipo *finishDeposit*
    a. modifica lo stato del DC in chiuso
    b. invia un messaggio di conferma alla MA



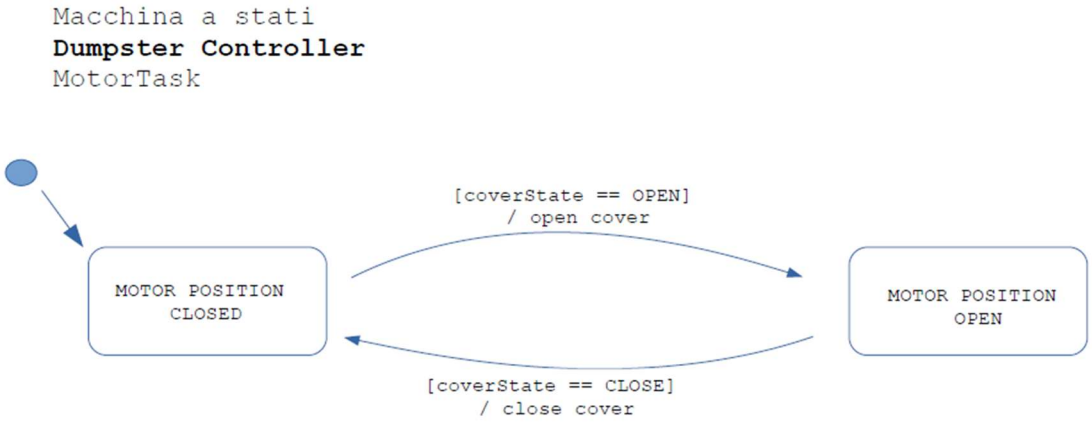
### 2.2.3 Countdown task

Si occupa di gestire il tempo concesso all'utente per depositare il rifiuto. Ad ogni tick il task controlla se lo sportello del cassonetto è aperto: in tal caso controlla se la differenza tra i due timestamp (rispettivamente di fine e inizio deposito) supera il valore della variabile *countDownValue*. In caso affermativo viene impostato a chiuso lo stato dello sportello del cassonetto e viene posta a *true* la variabile booleana che permetterà di far capire al Bluetooth task che deve comunicare alla MA che il tempo è scaduto.




### 2.2.4 Motor Task

Si occupa dell'apertura e chiusura dello sportello del cassonetto nel momento opportuno. Ad ogni tick controlla se lo stato del cassonetto è cambiato rispetto all'ultimo controllo. Qualora sia variato in aperto, il task del motore si occuperà di aumentare l'angolo del servo (che simula lo sportello); nel caso in cui sia variato in chiuso si occuperà di effettuare il movimento inverso.



# 3. Dumpster Edge

## 3.1 Introduzione

Il Dumpster Edge (DE) è il componente che si occupa di mantenerne lo stato attuale del sistema. Non comunica direttamente con il DC, ma solamente con il Dumpster Service (DS) mediante HTTP. La sua funzione secondaria è la lettura di un potenziometro, il cui valore indica il peso del rifiuto depositato dall' utente.

Gli stati in cui può trovarsi sono due:

- AVAILABLE, quando il cassonetto contiene un peso totale di rifiuti inferiore alla capacità massima. In questa situazione è consentito al DS di rilasciare un nuovo token per poter depositare nuovi rifiuti.
- NOT AVAILABLE, stato che indica che il cassonetto è pieno; in questa situazione gli utenti non hanno la possibilità di effettuare depositi.

Gli stati sono indicati dalla presenza di due led, rispettivamente di colore verde e rosso.

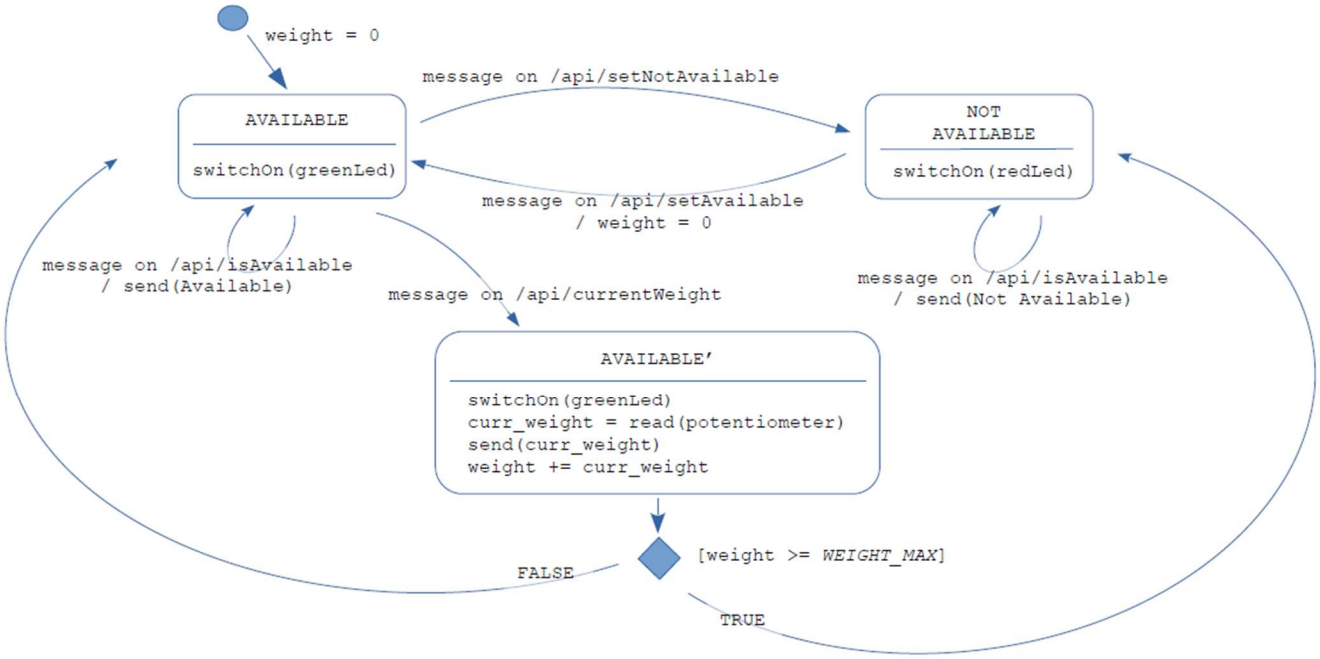
## 3.2 Implementazione

Il DE utilizza la libreria ESP8266WebServer per offrire un servizio web server. Questo componente del sistema è in grado di gestire le richieste HTTP provenienti dal DS controllando la presenza di eventuali richieste pendenti.

Il DE mette a disposizione 4 API alle quali il DS può rivolgersi:

1. */api/isAvailable*, informa se il cassonetto può accettare un nuovo deposito.
2. */api/setAvailable*, comando con il quale il cassonetto viene svuotato dal suo contenuto e ne viene cambiato lo stato in AVAILABLE.
3. */api/set*NotAvailable, comando con il quale viene cambiato lo stato del cassonetto in NOT AVAILABLE
4. */api/currentWeight*, comando con il quale viene chiesto il valore del potenziometro. Quando il DE riceve una richiesta su quest'API è consapevole che un deposito è stato terminato con successo e quindi somma il peso inviato al Dumpster Service al peso dei rifiuti già presenti nel cassonetto.

Macchina a stati
**Dumpster Edge**




# 4. Dumpster Mobile App

## 4.1 Introduzione

È la parte del sistema in esecuzione sui dispositivi mobili degli utenti. Il ruolo di questa componente è consentire all'utente di interagire con il cassonetto al fine di effettuare un deposito di un rifiuto. Per effettuare le sue mansioni interagisce con due parti del sistema: il DC, mediante tecnologia Bluetooth, e il DS, mediante HTTP. In ordine, le operazioni che l'utente può richiedere mediante l'interfaccia dell'applicazione sono:

- Richiesta del token, che può andare a buon fine oppure fallire
- Connessione con il cassonetto, anche questa può andare a buon fine oppure fallire.
- Selezione del tipo di rifiuto, a cui consegue l'apertura del cassonetto e l'inizio del countdown per effettuare il deposito

Esistono poi due ulteriori operazioni, non obbligatorie per il conseguimento del deposito, che sono:

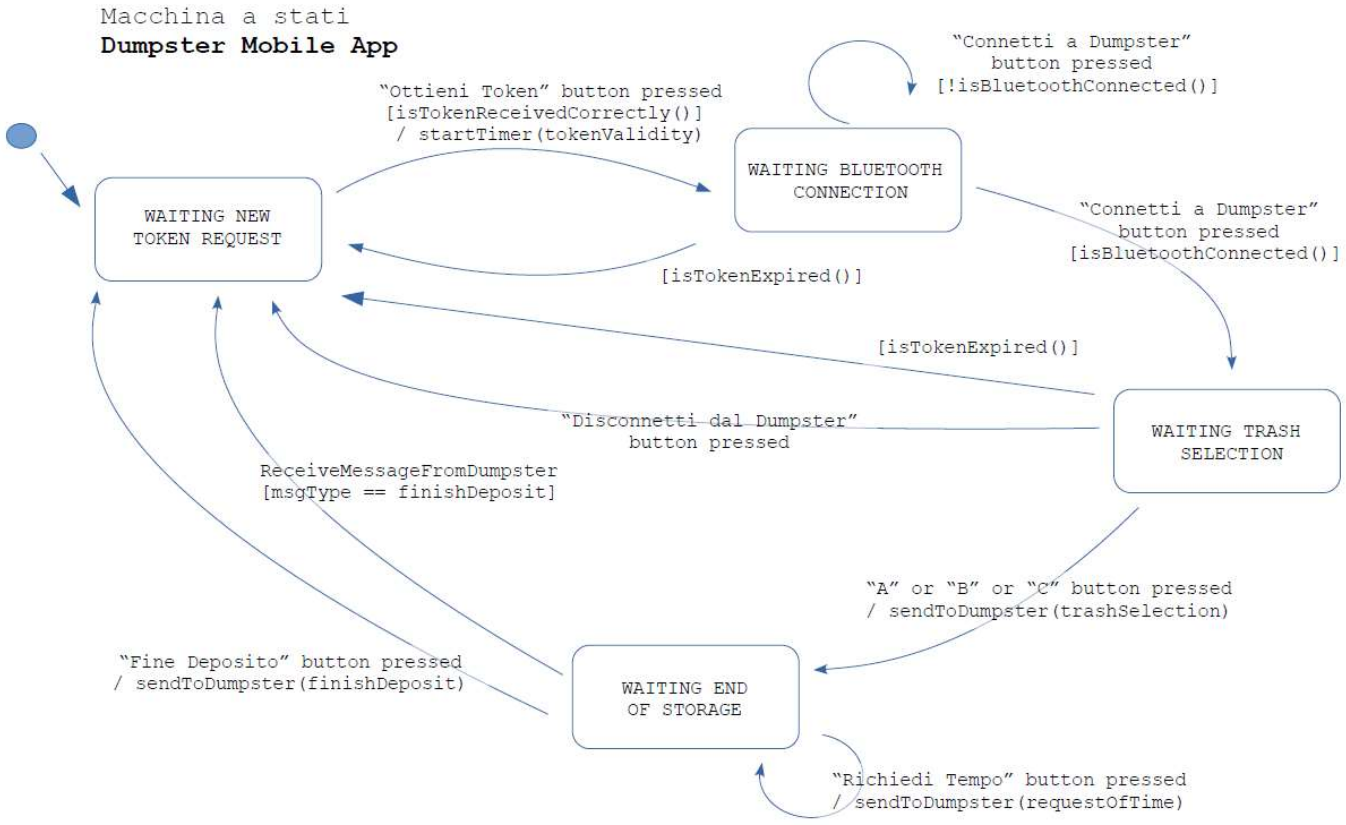
- Richiesta di ulteriore tempo
- Conclusione del deposito prima del termine del countdown

## 4.2 Implementazione

L'applicazione, al momento della richiesta del token, invia una richiesta al DS il quale, se il dumpster è nello stato *available*, restituisce un valore univoco come token. All'interno del messaggio inviato al dispositivo è presente anche un valore numerico, che indica la validità, in secondi, del token. Il token ha una validità che si aggira attorno ai 30 secondi, dopo i quali l'utente che ne è entrato in possesso non è più abilitato ad interagire con il DC. Se entro questo lasso temporale l'utente non seleziona un tipo di rifiuto l'interfaccia viene resettata. Lo scadere del tempo viene realizzato mediante un meccanismo di countdown che viene avviato al momento della ricezione del messaggio HTTP contenente il token.

Una volta scelto il tipo di rifiuto la valenza temporale del token perde rilevanza e l'utente, oltre ai novanta secondi concessi inizialmente, può richiedere più volte altri trenta secondi da aggiungersi a quelli iniziali. Al termine del deposito, sia per lo scadere del tempo che per richiesta esplicita, l'applicazione comunica al DS il successo dell'operazione, indicando la tipologia di rifiuto depositato.

Poiché Android non dà alcuna garanzia che il nome del dispositivo bluetooth (in questo caso il modulo HC-06) sia trasmesso subito allo smartphone, per avere la certezza di connettersi al dispositivo corretto è stato utilizzato il MAC, di cui abbiamo invece certezza di ricezione. È dato per scontato che i dispositivi BT siano stati preventivamente accoppiati.



# 5. Dumpster Dashboard

## 5.1 Introduzione

La dashboard è l'elemento del sistema che svolge le funzioni di visualizzazione e controllo, permettendo all'utente di controllare in tempo reale lo stato del cassonetto e il suo utilizzo nel tempo. Questa consiste in una pagina HTML con logica scritta in Javascript e il cui stile è realizzato mediante la libreria Bootstrap. Per la visualizzazione dei dati si è deciso di sfruttare la libreria grafica Graph.js.

Le funzionalità esposte dalla dashboard sono:

- Visualizzazione dello stato attuale del cassonetto
- Visualizzazione dell'andamento dell'utilizzo del cassonetto negli ultimi N giorni
- Modifica da parte dell'utente dello stato del cassonetto da *available* a *notAvailable* e viceversa

## 5.2 Implementazione

All'apertura della dashboard sono mostrati lo stato attuale del cassonetto e l'andamento dell'utilizzo negli ultimi sette giorni. I valori vengono aggiornati periodicamente mediante una opportuna API esposta dal Dumpster Service, alla quale viene rivolta una chiamata HTTP di tipo Ajax ogni duecento millisecondi.

Lo scopo di tale chiamata è quello di richiedere al service due informazioni fondamentali:

- Stato attuale del dumpster: *available* o *notAvailable,* numero depositi e quantità corrente
- Lista dei depositi compiuti negli ultimi N giorni (selezionabile mediante interfaccia).

È possibile modificare lo stato attuale del cassonetto mediante la pressione di un pulsante che consente, dato il valore attuale, di rendere il sistema *available* o *notAvailable*. Quando il cassonetto passa dallo stato *notAvailable* ad *available* viene svuotato di tutti i suoi rifiuti. La dashboard è l'unico punto del sistema in cui l'utente è in grado di modificare a suo piacimento lo stato del cassonetto e quindi, di conseguenza, anche di svuotarlo.

# 6. Dumpster Service

## 6.1 Introduzione

Il Dumpster Service ((DS) è la parte centrale dell'applicazione e ha il compito di interconnettere le varie parti del sistema. In particolare: Mobile App, Dumpster Edge e Dumpster Dashboard. Il DS espone diverse API che possono essere richiamate dalle varie componenti del sistema, è in costante attesa di nuove chiamate HTTP.

Per realizzare le API e permettere al DS di essere sempre contattabile si è scelto di utilizzare Node.js, il quale permette di eseguire codice Javascript lato server. Per la realizzazione delle API e delle chiamate HTTP sono stati usati due moduli propri di Node.js, quali:

- *express*, un framework pensato per applicazioni web che permette di mettere il server in ascolto su una porta consentendogli di essere contattato da applicazioni client mediante la chiamata di specifiche API (tutte con un formato */api/nome_api*).
- *xmlhttprequest*, un oggetto Javascript che permette di realizzare richieste HTTP

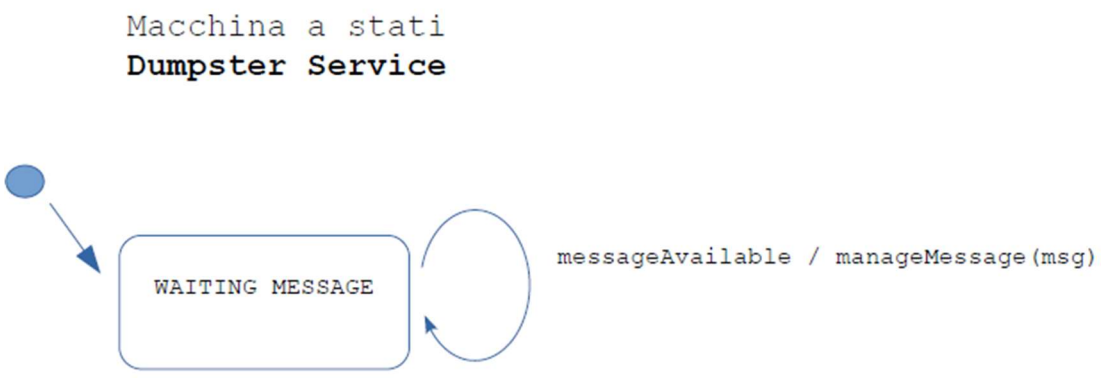
## 6.2 Implementazione

Il Dumpster Service mette a disposizione numerose API che le diverse componenti del sistema con cui esso comunica possono contattare.

Periodicamente la dashboard richiama una di queste API per potersi aggiornare. Il DS genera una risposta che consiste in un oggetto JSON che contiene lo stato corrente del cassonetto (disponibilità, numero depositi e quantità corrente) e i dati relativi ai depositi degli ultimi N giorni. È proprio mediante queste informazioni che la Dashboard realizza la propria interfaccia.

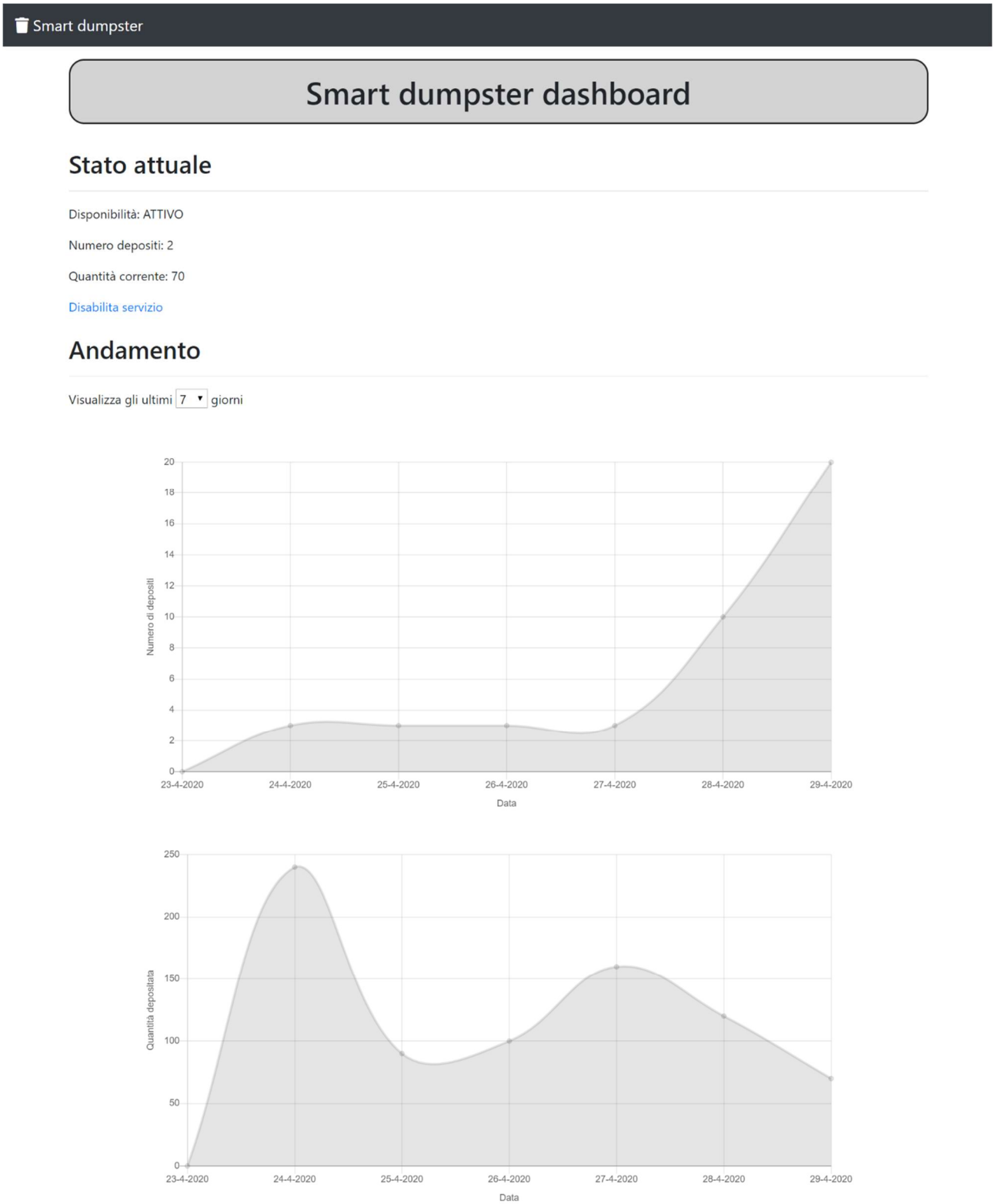
Il DS espone anche due API che la dashboard può sfruttare per modificare lo stato del cassonetto. Ricevuta la chiamata il DS si preoccupa di interagire con l'ESP per richiedergli la modifica. Viene esposta in questa ottica anche un'API che può essere chiamata dall'ESP stesso, che permette a quest'ultimo di comunicare il proprio stato, senza che questo sia stato modificato da parte dell'utente mediante la dashboard. Questa chiamata avviene nel caso in cui un deposito porti il valore del peso oltre la soglia, momento in cui il dumpster passa dallo stato *available* a quello *notAvailable*.

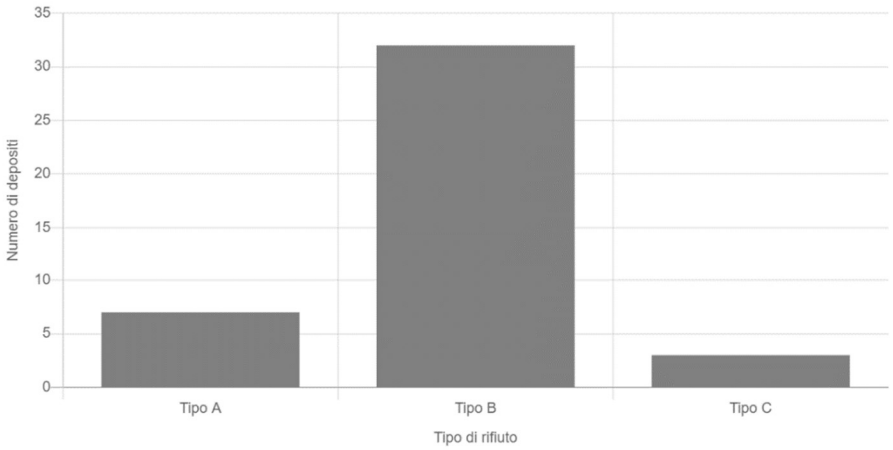
Per quanto riguarda la comunicazione con la MA sono disponibile diverse API dedicate. Una viene utilizzata per consentire alla MA di richiedere un token. In tale situazione il DS controlla lo stato attuale del cassonetto e, nel caso sia *available,* il token viene correttamente inviato. In caso contrario viene negata l'autorizzazione. Al termine del deposito la MA invia ad un'altra API tutte le informazioni legate al deposito. Il DS si premunirà di effettuare una chiamata HTTP verso l'ESP per richiedere l'effettivo peso del rifiuto, aggiornando quindi i dati locali in funzione di tale valore. Il DS memorizza ciascun deposito all'interno del file *story.json*.

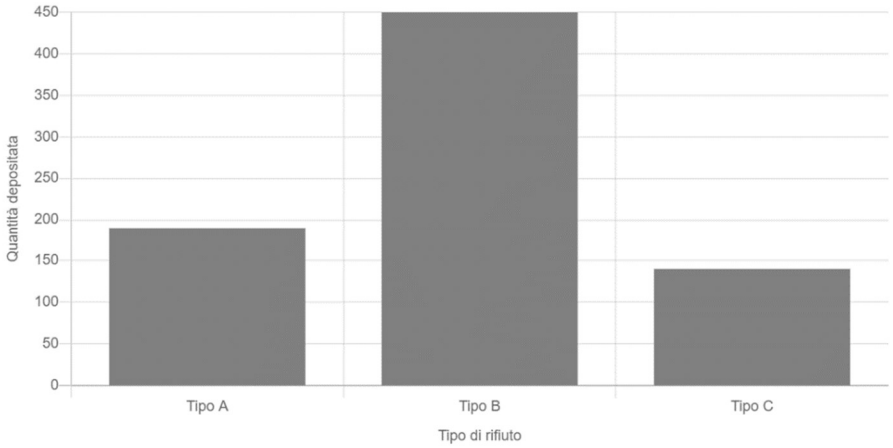


# 7. Galleria

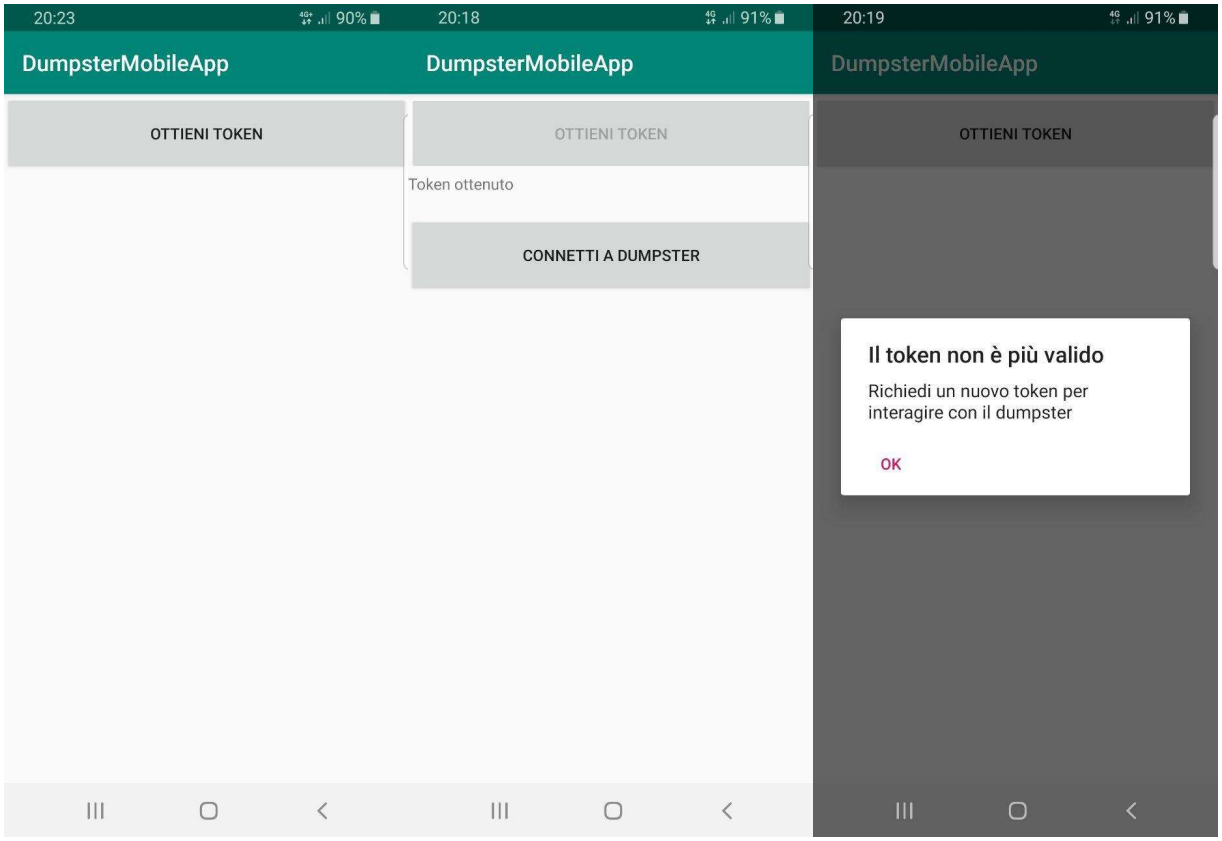
## 7.1 Dumpster Dashboard

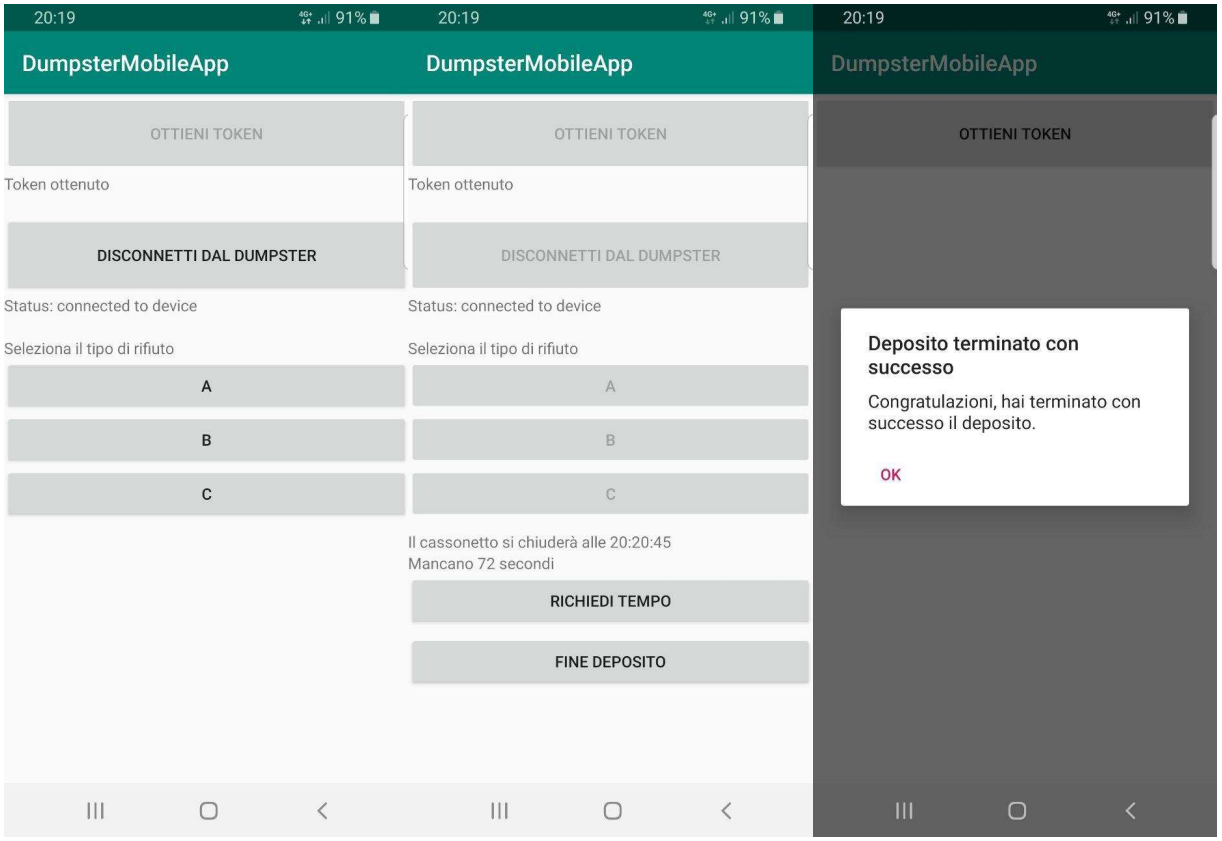



35
30
25
20
15
10
5
0
Numero di depositi
Tipo A
Tipo B
Tipo C
Tipo di rifiuto


450
400
350
300
250
200
150
100
50
0
Quantità depositata
Tipo A
Tipo B
Tipo C
Tipo di rifiuto

## 7.2 Dumpster Mobile App



## 7.3 Schema del circuito