Alma Mater Studiorum Università di Bologna
Campus di Cesena
C.d.L. in Ingegneria e Scienze Informatiche

RUSCO-1

Relazione sull'elaborato
per il corso di Sistemi Multimediali
a.a. 2014/2015

Realizzato da:

Nicola Giancecchi
M. 0000653628
nicola.giancecchi@studio.unibo.it

Aldo Junior Simoncini
M. 0000652120
aldojunior.simoncini@studio.unibo.it

Filippo Pagani
M. 0000654230
filippo.pagani@studio.unibo.it

Dominio del problema

Sviluppo di una web app che realizza un serious game sul riciclaggio dei rifiuti.
L'utente fotografa un rifiuto, il sistema chiede il tipo o offre un aiuto a determinare il tipo; se l'utente chiede aiuto, il sistema si collega a Google Image e prova a classificare automaticamente l'immagine. Se non ci riesce aspetta che l'utente selezioni manualmente.
In base al tipo individuato dall'utente, il sistema visualizza il percorso verso il bidone più vicino.
Se l'utente raggiunge la posizione del bidone, accumulala un punto. In funzione dei punti si guadagnano badge e si riduce l'impronta ecologica.
Il rifiuto successivo, per generare punti deve essere o di tipo diverso o in un luogo diverso.
Un test funzionante comprende la georeferenziazione di alcuni bidoni, anche virtuali, in modo che si possano ottenere premi.

Analisi del progetto

Framework Apache Cordova
Si è deciso di realizzare la web app tramite il framework Apache Cordova.
Tale framework permette di sviluppare applicazioni web che però si possono appoggiare a componenti dello smartphone, come ad esempio il file system, la fotocamera o il GPS. Cordova può generare, in seguito, applicazioni native per iOS, Android, Windows, Blackberry, Amazon FireOS e Firefox OS contenenti esclusivamente una webview che carica il codice HTML, CSS e Javascript della pagina. Il pregio principale di questa piattaforma è che è possibile sviluppare applicazioni multi piattaforma in una sola volta; il difetto è che l'applicazione non è reattiva come quella nativa e possono esserci incompatibilità con i plugin che possono funzionare per una piattaforma e per l'altra non completamente . Ciò anche per le caratteristiche dei sistemi operativi: su iOS, ad esempio, la UIWebView si basa su una versione di Safari priva dell'engine Javascript Nitro, il che rende l'applicazione più lenta.

Prototipo
Il prototipo realizzato permette di eseguire tutti i compiti richiesti. Si può infatti cercare il nome di un rifiuto in un database e/o farsi aiutare scattando una foto ed utilizzando CloudSight. Successivamente è possibile raggiungere il bidone più vicino e guadagnare punti, sbloccando quindi i badge.

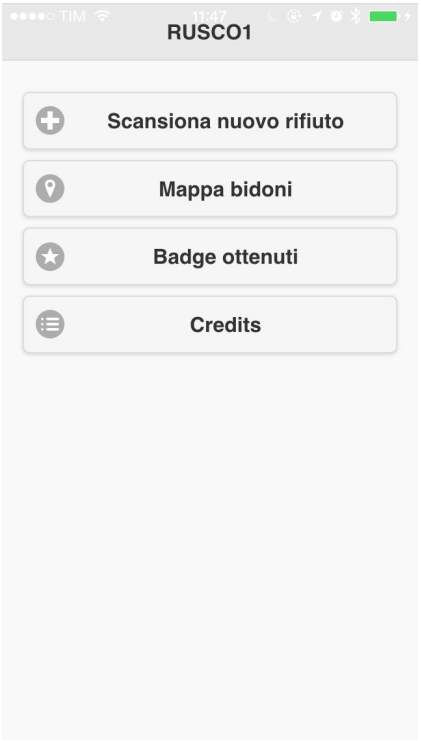
Suddivisione dei compiti

Task svolti da Nicola Giancecchi

I principali task svolti sono stati:

- creazione grafica delle schermate

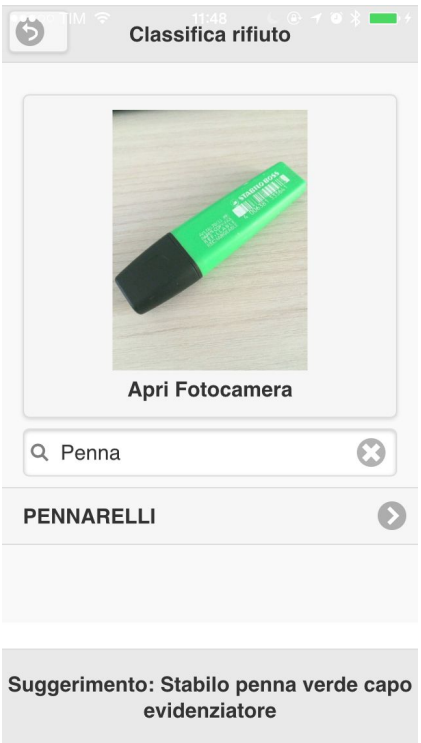
- connessione a webservice di CloudSight
- creazione lista rifiuti e selezione del rifiuto
- creazione lista punti di raccolta



Grafica delle schermate
In primo luogo sono state create le singole schermate. Esse sono molto semplici e seguono lo stile di default fornito da JQuery Mobile.
La home è caratterizzata da quattro scelte: scansiona nuovo rifiuto, mappa bidoni, badge e credits.

Facendo tap su "Scansiona nuovo rifiuto", si aprirà una nuova schermata dove in alto è presente il tasto "Apri Fotocamera" e subito dopo la barra di ricerca. Una volta fatto tap su "Apri Fotocamera" verrà aperta la fotocamera di sistema per scattare la foto da inviare a CloudSight. Una volta scattata la foto, nella barra in basso verrà mostrato l'avanzamento dell'upload e, alla fine, il risultato della ricerca. Cercando nella barra di ricerca si potrà quindi scegliere tra un'ampia lista di rifiuti. Tale lista è formattata in JSON e viene scaricata da un web server.
Facendo tap su un rifiuto si aprirà la mappa verso il bidone più vicino. Una volta raggiunto il bidone verranno assegnati i punti.

Tornando alla home, invece, abbiamo la Mappa Bidoni, dove è presente una cartina con le isole ecologiche più vicine, e i Badge ottenuti, ovvero una lista di badge guadagnati durante l'utilizzo dell'applicazione.



Connessione al webservice di CloudSight
Per il riconoscimento del rifiuto sono state utilizzate le API di CloudSight (www.cloudsightapi.com) , un servizio realizzato appositamente dai creatori di CamFind, un'applicazione social che permette di dare una descrizione agli oggetti fotografati. Altre API sono state prese in considerazione, ma erano tutte meno convenienti (CloudSight mette a disposizione 1000 request gratuitamente) e meno accurate. Google Immagini, inoltre, non fornisce la Visual Search tra le proprie API.
Come intuibile dalla documentazione di CloudSight (https://cloudsight.readme.io), il webservice è composto solamente da due chiamate HTTPS.
La prima, `image_request`, permette di inviare l'immagine fotografata al server. La chiamata è effettuata in POST e richiede come parametri minimi la lingua in cui restituire i risultati e, naturalmente, l'immagine. E' necessario inoltre

specificare nell'header la API key come campo "Authorization".
Una volta inviata l'immagine al server, è necessario effettuare un polling continuo verso l'endpoint `image_responses` in GET. Per ogni chiamata effettuata viene restituito lo stato corrente della richiesta: not completed, completed, not found, skipped e timeout. Nel caso che il sistema riesca a descrivere la foto, viene restituita la stringa descrittiva e mostrata a video.

Lista rifiuti
E' stata inoltre implementata una lista di rifiuti con i relativi bidoni. La lista viene scaricata online dalla repository di Github tramite RawGit (file rifiuti.json).
Essa è strutturata in formato JSON. Ogni record del file è composto dall'ID del rifiuto, il nome e il bidone di riferimento.
La chiamata viene effettuata tramite il metodo `getJSON` che è asincrono. In caso di successo l'array ricevuto viene iterato e viene aggiunto un elemento `<li>` alla lista non ordinata `<ul>` presente nella schermata di selezione del rifiuto. Il filtraggio in base ai dati cercati viene effettuato automaticamente da JQuery Mobile.

Lista punti di raccolta
E' stata ricavata, sempre dal database dell'app San Marino Differenzia (disponibile su iOS e Android), la lista dei punti di raccolta della Repubblica di San Marino in formato JSON. La lista viene scaricata online dalla repository di Github tramite RawGi (file puntiraccolta.json).. Ogni record comprende: ID, latitudine, longitudine e nome della via.
Tali dati vengono poi utilizzati per geolocalizzazione e georeferenziazione, parti sviluppate da Filippo.

Task svolti da Filippo Pagani

Come precedentemente elencato nei requisiti richiesti dall'applicazione, si è proceduto nello sviluppo delle funzionalità riguardanti il GPS e la localizzazione. Queste funzionalità in particolare riguardano:

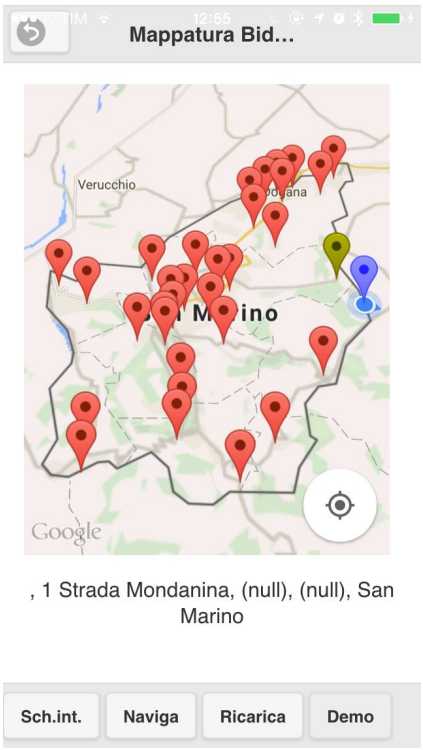
- La Georeferenziazione dei bidoni sulla mappa.
- La Geolocalizzazione dell'utente.
- "Geocoding and Reverse – Geocoding".
- Il calcolo del bidone più vicino rispetto la posizione dell'utente (nell'ordine dei metri).
- La creazione del percorso e la navigazione al bidone più vicino rispetto la posizione dell'utente.

Per garantire queste funzionalità è stato utilizzato il plugin di Cordova plugin.google.maps (`phonegap-googlemaps-plugin`) che permette l'utilizzo di una parte delle funzionalità tipiche delle librerie di Google Maps. Per utilizzare le API di Google Maps all'interno dell'applicazione è necessario avere una "API_KEY" generata nella Console Api Developers di Google (https://console.developers.google.com/ . E' necessario avere almeno un account di tipo free) specificamente per una data applicazione e piattaforma di riferimento. Nel nostro caso

abbiamo sviluppato questa applicazioni cross-platform per piattaforma Android e iOS; abbiamo quindi generato 2 API_KEY specifiche per questi due sistemi mobili. Una volta generate le due chiavi è sufficiente digitare Il comando

```
cordova plugin add plugin.google.maps --variable
API_KEY_FOR_ANDROID="YOUR_ANDROID_API_KEY_IS_HERE" --variable
API_KEY_FOR_IOS="YOUR_IOS_API_KEY_IS_HERE”
```

nel terminale per aggiungere il plugin al progetto ed utilizzarlo a livello di codice javascript nel file “index.js”. Nel caso in cui si sbaglino le impostazioni nella Console Api Developers o nell'inserimento della API_KEY si presenterebbe il problema “Blank Map” all'interno dell'applicativo. Il plugin permette un utilizzo molto semplificato delle funzionalità riguardanti le mappe e la Geolocalizzazione.



Creazione mappa, geolocalizzazione e georeferenziazione bidoni

Per creare la mappa nell'interfaccia grafica è sufficiente chiamare il metodo `plugin.google.maps.Map.getMap(mapDiv)` all'interno dell'evento `deviceready`. Se le impostazioni nella Console Api Developers sono corrette, la chiave inserita al momento dell'aggiunta del plugin è corretta, la connessione internet e il gps sono attivi, allora la nostra mappa comparirà all'interno del `<div>` specificato nel metodo sopra elencato. La geolocalizzazione della posizione può esser fatta in due modi:

- Attraverso `plugin.google.maps` con il metodo `map.getMyLocation(onSuccess, onError);`
- Attraverso le API Cordova riguardanti la Geolocalizzazione, utilizzando il metodo `navigator.geolocation.getCurrentPosition(onSuccess, onError)` o il metodo `navigator.geolocation.watchPosition(onSuccess, onError, { timeout: 30000 }).` La differenza tra i due è che il secondo controlla la posizione ciclicamente ogni tot millisecondi impostati nel parametro “timeout”.

In Rusco1 abbiamo utilizzato la seconda opzione.

Il database utilizzato contenente le coordinate geografiche dei bidoni viene fornito da San Marino Differenzia. Vengono mostrati nella mappa attraverso i marker creati e aggiunti con il seguente metodo:

```
map.addMarker( {
'position' : markNearLatLng,
'title' : markNearTitle,
'icon' : 'yellow'
}//new marker, function(marker) //call when user tap marker{
```

```
      marker.showInfoWindow();
 });
```

Si è optato per dare ai marker un significato secondo questa legenda:

- Marker Giallo: bidone più vicino all'utente.
- Marker Blu: posizione attuale dell'utente.
- Marker Rosso: bidone.

Geocoding e reverse geocoding

Questi due processi corrispondono rispettivamente all'operazione di traduzione dei nomi dei luoghi (vie, strade,ecc) in coordinate geografiche (latitudine, longitudine) e viceversa. Nell'applicativo viene utilizzato il processo di reverse geocoding, ovvero date le coordinate geografiche, vogliamo sapere il nome della via in cui ci troviamo. Quest'operazione viene effettuata grazie al seguente metodo:

```
plugin.google.maps.Geocoder.geocode(request, function(results){});
```

passando la variabile *request* inizializzata con l'oggetto latLng che contiene latidudine e longitudine del punto in cui vogliamo sapere il nome della via, nel nostro caso il punto preciso in cui ci troviamo:

```
var request = {
  'position': location.latLng
};
```

Calcolo del bidone più vicino rispetto la posizione dell'utente

Nelle API native di google maps sono presenti vari metodi per calcolare la distanza in metri tra due punti. Purtroppo nel plugin utilizzato non sono presenti queste funzionalità perciò si è optato per l'utilizzo della formula di Haversine. Questa formula, date le coordinate geografiche (latitudine, longitudine), calcola la distanza in metri tra due punti nella mappa. Si esegue il metodo iterativamente cercando la distanza minore tra dove si trova l'utente e il bidone più vicino. Una volta individuato il bidone più vicino si aggiunge il rispettivo marker di colore giallo.

Metodo che utilizza la formula di Haversine per calcolare la distanza in metri tra due punti:

```
//Haversine formula for calculate distance between marker points
distanceBetweenTwoMarker: function (latP1, longP1, latP2, longP2){
    var rad = function(x) {
      return x * Math.PI / 180;
    };
    var getDistance = function() {
      var R = 6378137; // Earth's mean radius in meter
      var dLat = rad(latP2 - latP1);
```

```
      var dLong = rad(longP2 - longP1);
      var a = Math.sin(dLat / 2) * Math.sin(dLat / 2) +
        Math.cos(rad(latP1)) * Math.cos(rad(latP2)) *
        Math.sin(dLong / 2) * Math.sin(dLong / 2);
      var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
      var d = R * c;
      return d; // returns the distance in meter
    };
    return getDistance();
}
```

Creazione percorso e navigazione al bidone più vicino rispetto la posizione dell'utente

Un ulteriore funzionalità non permessa dal plugin per le mappe e quella di disegnare un percorso da un marker all'altro. In questo caso però viene data la possibilità di richiamare l app predefinita del dispositivo per la navigazione (Navigatore Google Maps in Android, Mappe in iOS). E' sufficiente richiamare il seguente metodo:

```
plugin.google.maps.external.launchNavigation({
  "from": myPositionMark,
  "to": nearTrashMark
});
```

dove myPositionMark e nearTrashMark sono due oggetti latLng che contengono latitudine e longitudine dei due punti di interesse, quello della posizione attuale dell'utente e quello del bidone più vicino.

Task svolti da Aldo Junior Simoncini

Come richiesto dalle specifiche di elaborato, all'utilità di un applicazione che aiuti l'utente nella raccolta differenziata doveva essere associata una logica di gioco.
L'utente guadagna punti e riconoscimenti riciclando e smaltendo correttamente i rifiuti, un'idea di gioco molto semplice che può essere sicuramente sviluppata e arricchita.
Una sola regola gestisce il punteggio: per guadagnare punti, il rifiuto gettato nel bidone indicato deve essere di tipologia differente dal precedente oppure, il rifiuto successivo deve essere gettato in un bidone differente da quello precedentemente utilizzato.

Le seguenti fasi hanno caratterizzato lo sviluppo:

- Creazione della schermata "Badge ottenuti"
- Funzione specifica per l'assegnamento dei punteggi
- Funzione per il caricamento dei dati salvati

Creazione della schermata "Badge ottenuti"

"Badge ottenuti" raccoglie semplicemente i punti guadagnati dall'utente suddivisi per tipologia di rifiuto, più una voce sottostante ad indicare il punteggio totale.
La tabella presenta tre colonne:

- Tipologia, elenca le tipologie di rifiuti trattate
- Quantità, visualizza il numero di rifiuti per ogni tipologia che hanno generato un punteggio
- Badge, visualizza eventuali badges di riconoscimento per la specifica tipologia. L'utente guadagna un badge per la tipologia se getta correttamente almeno 5 rifiuti appartenenti alla stessa tipologia

Funzione specifica per l'assegnamento dei punteggi

La schermata "Badge ottenuti" è per il momento "priva di vita", non è altro che dell'HTML con qualche istruzione di CSS. E' stata quindi sviluppata la funzione *salvaPunti: function(tipologia).* La suddetta funzione, viene chiamata quando l'utente raggiunge un bidone tra quelli mappati, se e solo se supera i controlli della funzione *onSuccess(position)*.
Se il bidone raggiunto è diverso dal precedente o se il rifiuto è di una tipologia differente, allora viene chiamato il metodo *salvaPunti()* che presa in input la stringa rappresentante la tipologia di rifiuto scelto dall'utente salva il punteggio guadagnato e lo notifica con un alert.
Considerate le poche informazioni da salvare, abbiamo optato per la soluzione più semplice e immediata utilizzando il *Web Storage*.
Abbiamo utilizzato questa tecnologia di storage per salvare coppie chiave-valore che descrivevano rispettivamente la tipologia del rifiuto e il numero di rifiuti correttamente gettati.
La stessa scelta è stata fatta per memorizzare in modo permanente le informazioni riguardanti le coordinate dell'ultimo bidone utilizzato e la tipologia dell'ultimo rifiuto utilizzato, in modo da poter controllare che le condizioni del gioco vengano rispettate nel metodo onSuccess().
Sempre nel metodo *salvaPunti()*, viene effettuato un controllo sul numero di rifiuti gettati nella corrente tipologia, se il numero corrisponde a 5, allora viene assegnato un badge di riconoscimento altrimenti viene solo notificato l'assegnamento del punto.

Funzione per il caricamento dei dati salvati

I dati salvati dall'applicazione, ovvero punteggi per tipologie, ultimo bidone e ultimo rifiuto, vengono letti nella funzione appositamente costruita *carica()*.
La suddetta funzione ha il compito di leggere dal file tutte le informazioni per utilizzarle:

- nel riempimento della tabella nella schermata "Badge ottenuti"

- nell'assegnamento dei rispettivi valori alle variabili *tipoRifiutoPrecedente* e *bidonePrecedente*