# CLIPS

Communication & Localization with Indoor Positioning Systems

---

## Università di Padova

## Manuale Sviluppatore 1.00


leaf.gruppo@gmail.com

| | |
|---|---|
| **Versione** | 1.00 |
| **Data Redazione** | 2016-05-08 |
| **Redazione** | Eduard Bicego |
| **Verifica** | |
| **Approvazione** | Davide Castello |
| **Uso** | Esterno |
| **Distribuzione** | Prof. Vardanega Tullio |
| | Prof. Cardin Riccardo |
| | Miriade S.p.A. |

# Diario delle modifiche

| Versione | Data | Autore | Ruolo | Descrizione |
|---|---|---|---|---|
| 0.03 | 2016-05-04 | Marco Zanella | Progettista | Aggiunti contenuti sezione Strumenti di sviluppo |
| 0.02 | 2016-05-03 | Eduard Bicego | Amministratore | Ristrutturato documento |
| 0.01 | 2016-05-01 | Eduard Bicego | Amministratore | Aggiunta struttura documento |

# Indice

# Elenco delle figure

# 1 Introduzione

## 1.1 Panoramica generale

CLIPS è un'applicazione che offre funzionalità riguardanti la navigazione guidata all'interno di edifici, attraverso l'utilizzo di dispositivi mobile Android e dei beacon.
Tale applicazione è stata svilupparta per avere la possibilità di accedere alle informazioni per raggiungere una specifica area di interesse di un edificio offrendo indicazione testuali, sonore e visuali. Questo manuale ha lo scopo di illustrare le parti che compongono tale applicazione e il modo in tale applciazione è stata sviluppata.

## 1.2 Riferimenti utili

- Javadoc AltBeacon:
  https://altbeacon.github.io/android-beacon-library/javadoc/;
- Javadoc Clips:
  http://leafswe.github.io/clips/;
- Javadoc Dagger:
  http://google.github.io/dagger/api/2.0/;
- Javadoc Gson:
  http://google-gson.googlecode.com/svn/trunk/gson/docs/javadocs/index.html;
- Javadoc Picasso:
  https://square.github.io/picasso/2.x/picasso/;

# 2 Strumenti di sviluppo

## 2.1 SDK

Framework di sviluppo per applicazioni Android.

- Vesione SDK 24.4.1;
- Versione build tools 23.0.3;
- Versione target SDK 23;
- Versione minima SDK 19.

## 2.2 JDK

Insieme di strumenti di sviluppo per applicazioni Java.

- Vesione oraclejdk8.

## 2.3 Gradle

Sistema open source per l'automazione dello sviluppo basato su Groovy.

- Vesione 2.1.0.

### 2.3.1 Configurazione file gradle

Nel file gradle vengono dichiarate tutte le dipendenze da risolvere per poter testare, compilare ed eseguire l'applicazione. Di seguito sono riportate le dipendenze dichiarate per l'utilizzo dell'applicativo.

- Dichiarazione dei plugin utilizzati

```
apply plugin: 'com.android.application'
apply plugin: 'com.neenbedankt.android-apt'
```

- Dichiarazione dipendenze per gli script di build

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
```

```
        classpath 'com.android.tools.build:gradle:2.1.0'
    }
}
```

- Dichiarazione della configurazione di Android utilizzata

```
android {
    compileSdkVersion 23
    buildToolsVersion '23.0.3'

    defaultConfig {
        applicationId "com.leaf.clips"
        minSdkVersion 19
        targetSdkVersion 23
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled true
            shrinkResources true
            proguardFiles getDefaultProguardFile
              ('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
    defaultConfig {
        testInstrumentationRunner "android.support.test.
          runner.AndroidJUnitRunner"
    }
}
```

- Dichiarazione delle dipendenze verso pacchetti esterni utilizzati
  - Dipendenze da componenti Android

```
compile 'com.android.support:appcompat-v7:23.3.0'
androidTestCompile 'com.android.support:
  appcompat-v7:23.3.0'
androidTestCompile 'com.android.support:
  support-annotations:23.3.0'

compile 'com.android.support:design:23.3.0'
```

```
androidTestCompile 'com.android.support:design:23.3.0'
compile 'com.android.support:support-v4:23.3.0'
androidTestCompile 'com.android.support:support-v4
    :23.3.0'
```

– Dipendenze verso Dagger

```
compile 'com.google.dagger:dagger:2.0'
provided 'com.google.dagger:dagger-compiler:2.0'
provided 'org.glassfish:javax.annotation:10.0-b28'
```

– Dipendenze verso JGraphT

```
compile 'org.jgrapht:jgrapht-core:0.9.1'
```

– Dipendenze verso Gson

```
compile 'com.google.code.gson:gson:2.6.2'
```

– Dipendenze verso JUnit(test)

```
testCompile 'junit:junit:4.12'
androidTestCompile 'com.android.support:
  support-annotations:23.1.1'
androidTestCompile 'com.android.support.test:rules:0.5'
androidTestCompile 'com.android.support.test:runner:0.5
    '
```

– Dipendenze verso Mockito(test)

```
testCompile 'org.mockito:mockito-core:1.10.19'
```

– Dipendenze verso Espresso(test)

```
androidTestCompile 'com.android.support.test.espresso:
  espresso-core:2.2.2'
androidTestCompile 'com.android.support.test.espresso:
  espresso-web:2.2.2'
androidTestCompile ('com.android.support.test.espresso:
  espresso-contrib:2.2.2') {
    exclude group: 'com.android.support',
      module: 'appcompat'
    exclude group: 'com.android.support',
      module: 'support-v4'
    exclude module: 'recyclerview-v7'
}
androidTestCompile 'com.android.support.test.espresso:
  espresso-idling-resource:2.2.2'
```

– Dipendenze verso FindBugs(analisi statica)

```
compile 'com.google.code.findbugs:jsr305:2.0.1'
```

# 3 Componenti esterne

## 3.1 SQLite

Libreria che implementa un DBMS SQL transazionale senza la necessità di un server. Viene utilizzata per salvare e gestire le mappe scaricate e installate nel dispositivo e il relativo contenuto.

- Versione utilizzata 3.9.2

## 3.2 AltBeacon

Libreria che permette ai sistemi operativi mobile di interfacciarsi ai Beacon, offrendo molteplici funzionalità. Viene utilizzata per permettere la comunicazione tra l'applicativo Android e i Beacon.

- Versione utilizzata 2.02

## 3.3 JGraphT

Libreria Java che fornisce funzionalità matematiche per modellare grafi. Viene utilizzata per la rappresentazione delle mappe e per il calcolo dei percorsi.

- Versione utilizzata 0.9.1

## 3.4 Gson

Libreria Java che fornisce funzionalità per la gestione di oggetti JSON. Tale libreria è utilizzata la gestione del download delle mappe da remoto.

- Versione utilizzata 2.6.2

## 3.5 Dagger

Libreria Android utilizzata per effettuare la dependency injection. Viene utilizzata per la creazione dei singleton.

- Versione utilizzata 2.0

## 3.6 Picasso

Libreria per la gestione delle immagini in remoto. Viene utilizzata per scaricare le immagini utilizzate durante la navigazione.

- Versione utilizzata 2.5.2

# 4 Funzionalità

Nella presente sezione vengono spiegate nel dettaglio le componenti dell'applicazione e il loro scopo, presentate per funzionalità offerte dall'applicazione. Ogni funzionalità viene prima descritta in una sottosezione *"Panoramica"* e successivamente in sottosezioni che approfondiscono gli aspetti che compongono la funzionalità.

Le funzionalità offerte dall'applicazione e descritte in seguito sono:

- Localizzazione Utente 4;
- Gestione preferenze **??**;
- Gestione delle mappe **??**;
- Navigazione **??**;
- Area sviluppatore **??**.

## 4.1 Localizzazione utente

### 4.1.1 Panoramica

L'applicazione offre la funzionalità di localizzare l'utente all'interno di un edifico in cui risiedono i beacon riconosciuti dall'applicazione e di mostrare semplici informazioni sull'edifico.

La localizzazione utente avviene seguendo le seguenti fasi:

1. l'utente avvia l'applicazione;
2. l'applicazione avvia il monitoring per poter rilevare i beacon circostanti;
3. l'applicazione reperisce l'identificativo major;
4. l'applicazione si accerta che i beacon rilevati siano pertinenti all'applicazione attraverso un confronto tra major rilevato e major dei beacon nel database locale;
5. se il beacon è riconosciuto ed esiste un match nel database locale:
   - viene costruito il grafico dal database;
   - vengono mostrate all'utente semplici informazioni sull'edificio;
6. se il beacon è riconosciuto e non esiste un match nel database locale:
   - viene segnalato all'utente che la mappa dell'edificio non è scaricata nel device;
   - l'utente se lo desidera è reindirizzato alla gestione delle mappe;
7. se il beacon non è riconosciuto:
   - viene ignorato.

## 4.2 Interfaccia grafica

### Componenti interne

- Package:

  ...

- Interfacce e classi:

  ...

### Componenti esterne

- Interfacce e classi SDK:

  ...

??? Immagine dell'applicazione nella home vuota
??? Immagine dell'applicazione nella home con le informazioni dell'edificio
??? Immagini dell'applicazione nella home vuota con messaggio di avviso mappa non disponibile nel device

## 4.3 Presenter

### Componenti interne

- Package:

  ...

- Interfacce e classi:

  ...

### Componenti esterne

- Interfacce e classi SDK:

  ...

  - `onCreate()`
  - `onDestroy()`
  - ...

### 4.3.1 Rilevamento beacon

**Componenti interne**

- Package:

  com.leaf.clips.model;

  com.leaf.clips.model.beacon;

- Interfacce e classi:

  BeaconManagerAdapter, MyBeacon, MyBeaconImp, MyDistanceCalculator, BeaconManagerAdapter.LocalBinder, BeaconRanger;

**Componenti esterne**

- Interfacce e classi AltBeacon:

  BeaconManager, BootstrapNotifier, BeaconConsumer, RangeNotifier, Regin, BeaconParser, DistanceCalculator, Beacon.

- Interfacce e classi JDK:

  PriorityQueue;

- Interfacce e classi SDK:

  Intent, LocalBroadcastManager, Service, Binder, LocalBroadcastManager, IBinder;

La classe BeaconManagerAdapter estende un bind Service (vedi appendice ) ed ha il compito di effettuare il ranging e il monitoring dei beacon circostanti. Il ranging è l'operazione svolta in background per riconoscere i beacon circostanti senza eccessiva precisione mentre il monitoring è l'operazione che segue in cui i beacon vengono invece rilevati con tutte le informazioni e con più precisamente. La comunicazione dei beacon rilevati dal com.leaf.clips.model avviene attraverso l'uso degli oggetti MyBeacon inviati tramite Intent per cui serializzati. Gli Intent vengono recuperati tramite BroadcastReceiver implementato in altre classi.

## 4.4 Costruzione grafo

### Componenti interne

- Package:

  com.leaf.clips.model

  com.leaf.clips.model.dataaccess

  com.leaf.clips.model.dataaccess.service

  com.leaf.clips.model.dataaccess.dao

  com.leaf.clips.model.navigator.graph

  com.leaf.clips.model.navigator.graph.edge

  com.leaf.clips.model.navigator.graph.vertex

  com.leaf.clips.model.navigator.graph.area

  com.leaf.clips.model.navigator.graph.navigationinformation

- Interfacce e classi:

  MapGraph,

### Componenti esterne

- Interfacce e classi JDK:

  ???

- Interfacce e classi JGraphT:

  ???

- Interfacce e classi SDK:

  ???

## 4.5 Gestione preferenze

Settings utilizzo

## 4.6 Gestione delle mappe

## 4.7 Navigazione

### 4.7.1 Panoramica

La funzionalità di navigazione è resa disponibile dalle componenti dei package:

- `navigator`;
- `beacon`;
- `compass`;
- `dataaccess`;
- `setting`.

Esse permettono di guidare l'utente all'interno di un edificio. La navigazione è gestita attraverso queste fasi:

1. l'utente interagendo con l'interfaccia grafica avvia la navigazione;
2. la business logic dell'applicazione costruisce un grafo;
3. alle componenti di `navigator` viene passato il grafo;
4. viene calcolato il percorso utilizzando la libraria `JgraphT`;
5. vengono restituite le informazioni necessarie per guidare l'utente verso la destinazione da lui scelta;
6. l'interfaccia mostra all'utente le informazioni.

### 4.7.2 Interfaccia grafica

## 4.8 Presenter

### 4.8.1 Costruzione grafo

Componenti coinvolte:

**Package:** `dataacess`, `dao`, `service`;

**Classi:** ;

**Componenti esterne:** .

La costruzione del grafo avviene MapGraph
dal database parti del grafo spiegare package graph

### 4.8.2 Calcolo percorso

### 4.8.3 Bussola

**Componenti coinvolte**

**Package:** `compass`;

**Classi:** `Compass`.

**Componenti esterne**

**Classi SDK:** `SensorManager`, `Sensor`, `SensorEventListener`.

### 4.8.4 Navigazione

**Componenti interne**

**Package:** `navigator`, `graph`, `edge`, `vertex`, `area`;

**Classi:** .

**Componenti esterne**

**Classi JGraphT:** `SimpleDirectedWeightedGraph`, `DijkstraPathFinder`, `DefaultWeightedEdge`;

**Classi JDK:** `Exception`.

## 4.9 Area sviluppatore

# A Fondamenti di Android

## A.1 Ciclo di vita di un'applicazione

## A.2 Activity

### A.2.1 Ciclo di vita

## A.3 Service

Tipologie di service

unbind service

## A.4 Ciclo di vita

# 5 Struttura degli oggetti JSON per il download delle mappe

La struttura di seguito proposta ricalca la struttura data agli oggetti JSON scaricati dal database remoto per l'installazione di mappe in locale. Nel caso in cui si voglia cambiare tale struttura si consiglia di estendere le classi con prefisso **Remote** e suffisso **Dao** presenti nel package **dao**.

## 5.1 Esempio di oggetto JSON rappresentante una mappa


```
{
  "building" : {
    "id" : 1,
    "uuid" : "f7826da6-4fa2-4e98-8024-bc5b71e0893e",
    "major" : 666,
    "name" : "Torre Archimede"
    "description" : "Edificio del Dipartimento di Matematica",
    "openingHours" : "08:00-19:00",
    "address" : "Via Trieste 63, 35121, Padova (PD)",
    "mapVersion" : "1.0",
    "mapVersion" : "5.2 KB"
  },
  "rois" : [ {
    "id" : 1,
    "uuid" : "f7826da6-4fa2-4e98-8024-bc5b71e0893e",
    "major" : 666,
    "minor" : 1001
  }, {
    "id" : 2,
    "uuid" : "f7826da6-4fa2-4e98-8024-bc5b71e0893e",
    "major" : 666,
    "minor" : 1002
  }, {
    "id" : 3,
    "uuid" : "f7826da6-4fa2-4e98-8024-bc5b71e0893e",
    "major" : 666,
    "minor" : 1003
  } ],
  "categories" : [ {
    "id" : 2,
    "description" : "Aule"
  }, {
    "id" : 1,
    "description" : "Bagni"
  } ],
```

```
  "pois" : [ {
    "id" : 1,
    "name" : "2BC60",
    "description" : "Aula 2BC60",
    "categoryId" : 2
  }, {
    "id" : 2,
    "name" : "Bagno femminile",
    "description" : "Bagno femminile",
    "categoryId" : 1
  } ],
  "roipois" : [ {
    "roiid" : 1,
    "poiid" : 1
  }, {
    "roiid" : 2,
    "poiid" : 2
  } ],
  "edgeTypes" : [ {
    "id" : 1,
    "description" : "Default"
  } ],
  "edges" : [ {
    "id" : 1,
    "startROI" : 1,
    "endROI" : 2,
    "distance" : 50,
    "coordinate" : "23",
    "typeId" : 1,
    "action" : "Alla fine del corridoio troverai
      il bagno femminile.",
    "longDescription" : "Esci da aula 2BC60,
      prosegui nel corridoio e in fondo a
      sinistra troverai il bagno femminile"
  } ],
  "photos" : [ {
    "id" : 1,
    "edgeId" : 1,
    "url" : "URL della prima foto"
  }, {
    "id" : 2,
    "edgeId" : 1,
    "url" : "URL della seconda foto"
  } ]
}
```

## 5.2 Descrizione oggetto building

```
...
  "building" : {
    "id" : 1,
    "uuid" : "f7826da6-4fa2-4e98-8024-bc5b71e0893e",
    "major" : 666,
    "description" : "Edificio del Dipartimento di Matematica",
    "openingHours" : "08:00-19:00",
    "address" : "Via Trieste 63, 35121, Padova (PD)",
    "mapVersion" : "1.0",
    "mapSize" : "5.2 KB"
  }
...
```

Tale oggetto è utilizzato per raccogliere le informazioni generali riguardanti un edificio e la sua mappa. In particolare:

- **id**: Rappresenta l'identificativo numerico univoco dell'oggetto;
- **uuid** Rappresenta l'identificativo UUID uguale per tutti i beacon sfruttati dall'applicativo;
- **major** Rappresenta l'identificativo Major uguale per tutti i beacon appartenenti ad uno stesso edificio;
- **name** Rappresenta il nome dell'edificio;
- **description** Rappresenta una descrizione dell'edificio. In questa parte si consiglia di spiegare la tipologia di edificio e per cosa tale edificio è utilizzato;
- **openingHours** Rappresenta l'orario di apertura e chiusura dell'edificio;
- **address** Rappresenta l'indirizzo dell'edificio;
- **mapVersion** Rappresenta la versione della mappa;
- **mapSize** Rappresenta la dimensione della mappa.

## 5.3 Descrizione oggetto rois

```
...
  "rois" : [ {
    "id" : 1,
    "uuid" : "f7826da6-4fa2-4e98-8024-bc5b71e0893e",
    "major" : 666,
    "minor" : 1001
  },
  ...
  ],
...
```

Tale oggetto è utilizzato per rappresentare tutti le Region Of Interest di un certo edificio. Ogni oggetto all'interno all'interno di tale array rappresenta una specifica Region Of Interest. In particolare:

- **id** Rappresenta l'identificativo numerico univoco dell'oggetto;
- **uuid** Rappresenta l'identificativo UUID uguale per tutti i beacon sfruttati dall'applicativo;
- **major** Rappresenta l'identificativo Major uguale per tutti i beacon appartenenti ad uno stesso edificio;
- **minor** Rappresenta l'identificativo univoco di un certo beacon all'interno di un edificio.

## 5.4 Descrizione oggetto pois

```
...
  "pois" : [ {
  "id" : 1,
  "name" : "2BC60",
  "description" : "Aula 2BC60",
  "categoryId" : 2
    },
  ...
  ],
...
```

Tale oggetto è utilizzato per rappresentare tutti i Point Of Interest di un certo edificio. Ogni oggetto all'interno all'interno di tale array rappresenta uno specifico Point Of Interest. In particolare:

- **id** Rappresenta l'identificativo numerico univoco dell'oggetto;
- **name** Rappresenta il nome associato ad uno specifico Point Of Interest;
- **description** Rappresenta una descrizione associata ad un Point Of Interest. Si consiglia di mettere in tale descrizione la funzione di tale Point Of Interest;
- **categoryId** Rappresenta l'identificativo associato alla categoria di appartenenza del Point Of Interest.

## 5.5 Descrizione oggetto roipois

```
...

  "roipois" : [ {
  "roiid" : 1,
  "poiid" : 1
  },
  ...
  ],
...
```

Tale oggetto è utilizzato per rappresentare i collegamenti tra Region Of Interest e Point Of Interest in un certo edificio. Ogni oggetto all'interno all'interno di tale array rappresenta uno specifico collegamento. In particolare:

- **roiid** Rappresenta l'identificativo numerico univoco di una Region Of Interest;
- **poiid** Rappresenta l'identificativo numerico univoco di un Point Of Interest.

## 5.6 Descrizione oggetto edges

```
...
  "edges" : [ {
    "id" : 1,
```

```
    "startROI" : 1,
    "endROI" : 2,
    "distance" : 50,
    "coordinate" : "23",
    "typeId" : 1,
    "action" : "Alla fine del corridoio troverai
      il bagno femminile.",
    "longDescription" : "Esci da aula 2BC60,
      prosegui nel corridoio e in fondo a
      sinistra troverai il bagno femminile"
  },
  ...
  ]
...
```

Tale oggetto è utilizzato per rappresentare tutti gli archi che collegano Region Of Interest nel grafo che rappresenta un edificio. Ogni oggetto all'interno all'interno di tale array rappresenta uno specifico arco. In particolare:

- **id** Rappresenta l'identificativo numerico univoco dell'oggetto;
- **startROI** Rappresenta la Region Of Interest di partenza dell'arco;
- **endROI** Rappresenta la Region Of Interest di arrivo dell'arco;
- **distance** Rappresenta lunghezza dell'arco;
- **coordinate** Rappresenta l'ampiezza dell'arco che ha per lati l'arco e il collegamento tra la Region Of Interest di partenza e il nord polare;
- **typeId** Rappresenta l'identificativo associato al tipo di appartenenza dell'arco';
- **action** Rappresenta una descrizione basilare delle azioni da compiere per superare l'arco;
- **description** Rappresenta una descrizione dettagliata delle azioni da compiere per superare l'arco.

## 5.7 Descrizione oggetto edgeTypes

```
...
  "edgeTypes" : [ {
    "id" : 1,
    "description" : "Default"
  },
  ...
  ],
...
```

Tale oggetto è utilizzato per rappresentare tutti i tipi di arco che possono essere presenti all'interno di un edificio. Ogni oggetto all'interno all'interno di tale array rappresenta uno specifico tipo di arco. In particolare:

- **id** Rappresenta l'identificativo numerico univoco di un tipo;
- **description** Rappresenta una descrizione testuale del tipo di arco.

## 5.8 Descrizione oggetto photos

```
...
  "photos" : [ {
    "id" : 1,
    "edgeId" : 1,
    "url" : "www.imageurl.com"
  },
  ...
  ],
...
```

Tale oggetto è utilizzato per rappresentare i link alle immagini utili alla navigazione. Ogni oggetto all'interno all'interno di tale array rappresenta il collegamento ad una specifica immagine collegata ad uno specifico arco. In particolare:

- **id** Rappresenta l'identificativo numerico univoco dell'oggetto;
- **edgeId** Rappresenta l'identificativo numerico univoco della Region Of Interest a cui l'immagine è collegata;
- **url** Rappresenta l'URL a cui è possibile recuperare l'immagine.