

# Documentazione tecnica integrazione Junker

## Sommario

Architettura WebView .....	2
Scopo del documento .....	2
Ambito applicativo .....	2
Implementazione tecnica.....	2
Informazioni sul device dell'utente .....	2
Navigazione .....	2
Utilizzo di funzioni native .....	3
Architettura Intent .....	4
Scopo del documento .....	4
Ambito applicativo .....	4
Implementazione tecnica.....	4
Cos'è un intent .....	4
Configurazione del richiedente.....	4
Configurazione del ricevente .....	5

# Architettura WebView

## Scopo del documento

Scopo del documento è fornire le specifiche tecniche relative all'implementazione della WebView da integrare all'interno dell'app Junker, per fornire i servizi del portale clienti agli utenti dei comuni coperti dal servizio.

## Ambito applicativo

All'interno del menu "Servizi" dell'app Junker è possibile aggiungere in modo dinamico funzionalità aggiuntive. Ognuno di questi servizi è identificato da un'icona, un nome e dallo URL che verrà aperto all'interno della WebView presente in app.

Per implementare la persistenza della sessione utente si può memorizzare un token di autenticazione/autorizzazione in un cookie chiamato `ClientToken`.

## Implementazione tecnica

### Informazioni sul device dell'utente

Junker passa alle applicazioni web implementate tramite WebView le informazioni sul device dell'utente in due cookie distinti:

- **DeviceInfo** contiene informazioni sul device, ovvero:
  - o Piattaforma (Android/iOS)
  - o Versione sistema operativo
  - o ID univoco device
- **JunkerInfo** contiene informazioni sull'app Junker:
  - o Versione e build number dell'app
  - o Identificativo univoco (all'interno del DB Junker) del comune selezionato dall'utente
  - o Lingua selezionata dall'utente
  - o Identificativo di registrazione sul Notification Hub per l'invio di notifiche push

Il valore dei cookie è in formato base64, che una volta decodificato fornisce un JSON con uno schema ben definito.

### Navigazione

La pagina web dovrà consentire all'utente di navigare indietro secondo il normale funzionamento della history html/javascript, in quanto il tasto back del device viene tradotto nell'equivalente funzione `history.back()` di javascript. Per questo motivo è consigliato evitare per quanto possibile i postback, che generano degli eventi di *form resubmission* che interrompono il normale flusso di navigazione utente.

Per le interazioni client-server è consigliato utilizzare AJAX, dando sempre evidenza all'utente dello svolgimento di operazioni in corso tramite messaggi e loader.



È possibile (anzi consigliato) usare le history API, che sono supportate da tutte le implementazioni recenti delle WebView.

### Utilizzo di funzioni native

Junker mette a disposizione alcuni metodi javascript per interagire con l'app nativa:

- `window.AndroidBridge.androidTakePicture()`: consente di caricare una foto dalla galleria del device
- `window.AndroidBridge.androidTakePictureCamera()`: consente di scattare una foto con la camera del device
- `window.AndroidBridge.androidOpenExternal(url_b64)`: consente di aprire un link web nel browser di sistema; l'url deve essere passato in base64
- `window.AndroidBridge.androidBackToApp()`: consente di chiudere la WebView e tornare alla lista dei servizi

*Altri metodi sono in sviluppo per lo scambio del token di sessione ed altre informazioni con Junker, in questo momento non disponibili.*

**N.B.** Nonostante il nome dell'oggetto `AndroidBridge`, le funzioni sono le stesse e sono disponibili sia per Android che per iOS.

# Architettura Intent

## Scopo del documento

Scopo del documento è fornire le specifiche tecniche relative integrazione di app terze nell'ecosistema Junker su device mobile, implementando, se necessario, scambio di dati.

## Ambito applicativo

All'interno del menu "Servizi" dell'app Junker è possibile aggiungere in modo dinamico funzionalità aggiuntive. Ognuno di questi servizi è identificato da un'icona, un nome e dallo URL intent che verrà aperto all'interno del contesto del device in uso.

Per implementare lo scambio di dati, è possibile specificare parametri da utilizzare durante la chiamata intent.

## Implementazione tecnica

### Cos'è un intent

Intent è un meccanismo di interazione tra app che permette ad una app di richiedere azioni ad un'altra e/o scambiare dati.

Intent è disponibile sia su piattaforma Android che iOS e necessita di configurazione sia sull'app richiedente che su quella ricevente.

### Configurazione del richiedente

Un esempio di configurazione in uscita su Android manifest (specifico per piattaforma Android), necessaria per lanciare intent:

```
<queries>
  <intent>
    <action android:name="android.intent.action.VIEW" />
    <data android:scheme="http" />
  </intent>
</queries>
```

In questa maniera sarà disponibile una chiamata intent ad una azione di tipo Action.VIEW nello schema http. Il codice potrà richiedere una attività di visualizzazione, ad esempio (c#/android):

```
Intent i = new Intent(Intent.ActionView , "http://google.com" );
Android.App.Application.Context.StartActivity(i);
```

Il sistema operativo, ricevuta la richiesta andrà a cercare tra tutte le app installate quella che è registrata allo schema *http* e fornisce una action *view*.



## Configurazione del ricevente

La app che risponderà per lo schema http e action view avrà una configurazione simile alla seguente (Android):

```
<intent-filter>
  <action android:name="android.intent.action.VIEW" />
  <category android:name="android.intent.category.DEFAULT" />
  <category android:name="android.intent.category.BROWSABLE" />
  <data android:scheme="http" />
</intent-filter>
```

E' possibile anche differenziare diverse 'rotte' intent all'interno della stessa app ricevente aggiungendo un path che differenzia diverse azioni sulla stessa app:

```
<data android:scheme="appSatellite"
      android:host="connect" />
```

Questa configurazione lato ricevente, permetterà di accettare intent del tipo:

appSatellite://connect

Se alla chiamata saranno appesi parametri come in una query string classica su browser, il ricevente potrà leggerli, ad esempio:

appSatellite://connect?token=djaoih2890dd92ndd

Nel caso dell'architettura di sblocco Junker – App satellite sarà quindi necessario configurare correttamente entrambe le app in maniera che possano dialogare correttamente.