Documentazione tecnica integrazione Junker – App satellite

Sommario

Architettura Junker – App satellite di sblocco contenitori

Scopo del documento

Scopo del documento è fornire le specifiche funzionali e architettura ad alto livello relative all'implementazione della logica di sblocco dei contenitori integrata tramite App satellite e Junker.

Ambito applicativo

All'interno dell'area "Servizi" dell'app Junker è possibile aggiungere in modo dinamico funzionalità aggiuntive.
Ognuno di questi servizi può essere una WebView o un link ad una app esterna a Junker.
In questa architettura la WebView sarà utilizzata per integrare un'area clienti (se presente) sviluppata dal fornitore o da Giunko. L'App satellite, anche questa sviluppata da fornitore o da Giunko, conterrà invece la logica necessaria per recuperare le chiavi di sblocco e per l'interazione col contenitore stesso.

Architettura e scenari d'uso

Scenario 1: utente si logga in area cliente

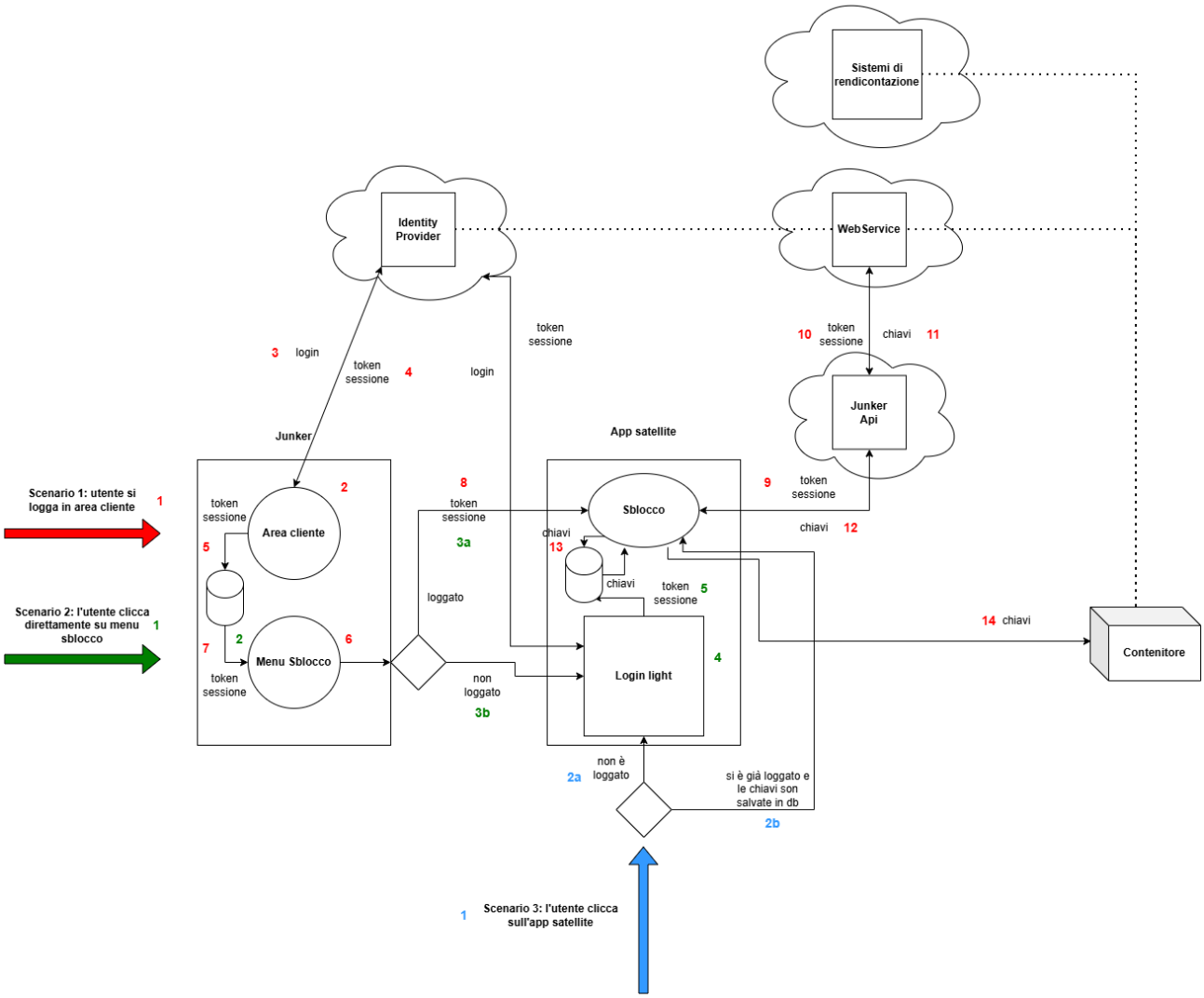
1. L'utente entra in Junker e trova configurato sia il servizio di Sblocco che quello di Area Cliente.
2. Clicca nella WebView che mostra l'Area Cliente
3. Inserisce le sue credenziali di accesso, variabile per fornitore, e si logga sul sistema che funge da identity provider
4. La WebView riceve un token di sessione o di login, sufficiente per il recupero delle chiavi
5. Tramite bridge JavaScript fornito da Giunko (vedi sezione Architettura Webview per dettagli), si salva il token sul DB in locale di Junker
6. Si clicca poi sull'Menu sblocco
7. Si recupera da DB in locale Junker il token salvato in precedenza
8. Tramite intent (vedi sezione Architettura Intent per dettagli) viene aperta l'App satellite prodotta da Junker o dal fornitore
9. L'App satellite si interfaccia o con le Junker API (layer non obbligatorio)
10. La richiesta di chiavi arriva al WebService del fornitore
11. Ottiene le chiavi di necessarie
12. Le chiavi arrivano fino in App satellite
13. Le chiavi vengono salvate su App satellite per uso futuro (ad esempio in caso di mancata connessione)
14. Le chiavi vengono propagate tramite BLE fino al contenitore tramite logica custom del fornitore delle elettroniche

Scenario 2: l'utente clicca direttamente su menu di sblocco

1. L'utente entra in Junker e trova configurato il servizio di Sblocco
2. Clicca nel menu di sblocco che legge sul DB locale Junker il token di sessione e, se presente, lo passa all'App satellite che viene aperta tramite intent (vedi sezione Architettura Intent per dettagli)
3. Se il token di sessione è presente, viene ricevuto dall'App satellite (step 3a) che lo utilizza come da flusso rosso, step 8. Nel caso non sia presente segue lo step 3b
4. Non avendo il token di sessione, sufficiente per recuperare le chiavi di sblocco, l'App satellite mostra una login all'utente che inserisce le sue credenziali e si logga sull'Identity provider. Questa login light può essere una semplice schermata di login o una WebView come su Junker
5. Ricevuto il token di sessione questo viene salvato sul DB locale dell'App satellite. Si procederà poi al recupero delle chiavi come da flusso rosso, step 9

Scenario 3: l'utente clicca sull'App satellite

1. L'utente apre direttamente l'App satellite
2. Se era loggato già in precedenza, sarà possibile recuperare il token di sessione sul DB locale (step 2.b) e procedere col flusso rosso, step 9 (o flusso rosso, step 13 nel caso le chiavi siano presenti e non scadute). Nel caso non si sia mai loggato si procede come flusso verde, step 4



Descrizione architettura con gli scenari descritti al punto precedente

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:
  - Piattaforma (Android/iOS)
  - Versione sistema operativo
  - ID univoco device
- **JunkerInfo** contiene informazioni sull'app Junker:
  - Versione e build number dell'app
  - Identificativo univoco (all'interno del DB Junker) del comune selezionato dall'utente
  - Lingua selezionata dall'utente
  - 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.