# SOCIAL WFM ACEA (versione mockata)

## Descrizione dell'applicazione

### Rete e servizi

L'applicazione si interfaccia in Http con i seguenti sistemi:

- SAP Mobile Platform
- SAP JAM
- Web Services sviluppati in Python

### SAP Mobile Platform

Middleware che espone i servizi OData on-premise per:

- Login (prima fase)
- Ordini di lavoro (home page)
- Servizi SAP HR - Employee Self Service

**NOTA**: I servizi sopra indicati sono raggiungibili soltanto da rete APN TIM opportunamente configurata sui tablet di ACEA. Può essere utile per lo sviluppatore impostare un tablet in tethering e condividere la rete APN con il PC per poter testare i servizi dal proprio ambiente di sviluppo (es. client REST, test automatici di integrazione ecc.)

### SAP JAM

Servizi Cloud di SAP JAM per le seguenti funzionalità:

- Login (seconda fase)
- Notifiche Social
- Contatti
- Gruppi
- Bacheca personale
- Post social su impianti e oggetti tecnici di SAP PM

Tutti i servizi vengono aceduti in OAuth1.0. Nella fase di login relativa al social Network l'utente autorizza il Social WFM ad utilizzare il suo profilo JAM (OAuth).

**NOTA**: L'accesso a questi servizi è garantito da qualunque rete con accesso ad Internet pubblica. Non è necessario essere in rete APN

### Web Services sviluppati in Python

Evolutive successive hanno previsto l'integrazione dei servizi di Survey e Concorsi all'interno dell'applicazione.

**NOTA**: L'accesso a questi servizi è garantito da qualunque rete con accesso ad Internet pubblica. Non è necessario essere in rete APN

## Struttura del codice

### Networking

La libreria utilizzata per il networking è **Koush-Ion**. Il package dove risiede tutta la logica di networking è **it.acea.android.socialwfm.http** e classi principali al suo interno sono

- **HttpClientRequest.java**: Contiene gli endpoint per i servizi di SAP JAM, ordini di lavoro, survey e concorsi; nonché la logica di strutturazione degli header e gestione dei cookies
- **EssClient.java**: Contiene gli endpoint di accesso ai servizi di SAP HR - Employee Self Service;nonché la logica di strutturazione degli header e gestione dei cookies per queste nuove chiamate

All'interno del package sono presenti tutte le classi di modello costruite a partire dalle response Http.

### Database

Il database utilizzato è **Realm**. E' stato utilizzato nella prima fase di progetto per il caching degli ordini di lavoro e delle notifiche dal social network. Nella seconda fase è stato poi abbandonato perchè ritenuto inutile ai fini della funzionalità.

### Background Service

L'applicazione utilizza un service di background **it.acea.android.socialwfm.service.UpdateSchedulerJob** per leggere in polling gli ordini di lavoro dalla mobile platform e le notifiche da SAP JAM. Se nuovi oggetti sono disponibili vengono scatenati degli eventi nell'applicazione che informano i componenti interessati. I componenti che reagisono a tali eventi sono quelli della home page: Mappa e badge notifiche

### Activity principali

Tutti i componenti di interfaccia grafica sono stati inseriti nel package **it.acea.android.socialwfm.app.ui**.

Le activities principali al suo interno sono:

- **SplashActivity.java**: All'accesso dell'utente verifica nelle shared preferences che l'utente sia correttamente registrato sui sistemi (SMP registration code, token OAuth di SAP JAM)
- **MainActivity.java**: Contiene la mappa con gli ordini di lavoro e gestisce la toolbar con notifiche, settings e ricerca
- **AuthenticatedWFMUser.java**: Effettua il login sui sistemi on premise (mobile platform). Questo è il primo step di login
- **OAuthActivity.java**: Permette all'utente di loggarsi su JAM ed eseguire l'OAuth. Questo è il secondo step di login
- **SecondOAuthActivity.java**: Continua il processo iniziato da OAuthActivity
- **SearchActivity.java**: Implementa la ricerca nel social network
- **ConcorsiActivity.java**: Mostra una webview che permette all'utente di partecipare ad un concorso
- **SurveyActivity.java**: Mostra una webview che permette all'utente di partecipare ad una survey

## CLASSI MODIFICATE:

- GiustificativiCalendarioFragment.java
- GiustificativiListFragment.java
- GiustificativoEditDialog.java
- TimbratureFragment.java
- TimbraturaMotivo.java
- ReperibilitaFragment.java
- CudCedolinoFragment.java
- QuadraturaFragment.java
- TimbraturaTipo.java
- CartellinoOrologioFragment.java
- MonteOreFragment.java
- CudCedolinoAdapter.java
- ResponseManager.java
- FragmentProfileInfo.java
- FragmentMyWorkContainer.java
- MyWorkRecycleAdapter.java
- FragmentMyWorkList.java
- FragmentMyWorkDetail.java
- SAPMobilePlatformFactory.java
- MockOdl.java
- ProfileFragment.java

## FILE MODIFICATI:

- AndroidManifest.xml
- Build.gradle (app)
- Build.gradle (social_wfm_android)
- Local.properties
- Gradle-wrapper.properties

## Descrizione delle modifiche

**ProfileFragment**

- Modifica del metodo **fetchPersonalData()** per ottenere i dati personali dell'utente senza alcuna chiamata (creazione dell'oggetto DatiPersonali con valori fittizi) al fine di mostrarli nella sezione "Il mio profilo".

**MockOdl**

- Versione mockata del RealmObject "Odl" (Ordini di Lavoro): utilizzata in una prima fase, è stata poi abbandonata nella versione finale dell'app.

**SAPMobilePlatformFactory**

- Modifica del metodo **retrieveOdl()** utilizzato per recuperare gli ordini di lavoro dal sistema SAP Mobile Platform: vengono evitate richieste HTTP al sistema cosi da evitare eccezioni legate a problemi di connessione col sistema (la lista degli ordini di lavoro viene comunque costruita con il metodo **getActualOdlList()** e con le modifiche effettuate nelle classi descritte successivamente).

**FragmentMyWorkDetail**

- Modifica del metodo **loadOdlData()** utilizzato per recuperare i singoli ordini di lavoro (oggetti "Odl") dal database Realm: gli ordini di lavoro incompleti (cioè quelli che presentano dei campi vuoti) vengono popolati sulla base del primo ordine di lavoro nella lista (il quale è "completo").

**FragmentMyWorkList**

- Modificata in una prima fase per gestire gli ordini di lavoro creati in maniera fittizia (oggetti "MockOdl"), è stata riportata alla sua versione originale con gli accorgimenti nel Fragment **FragmentMyWorkDetail**.

**MyWorkRecycleAdapter**

- Modificata in una prima fase per gestire gli ordini di lavoro creati in maniera fittizia (oggetti "MockOdl"), è stata riportata alla sua versione originale con gli accorgimenti nel Fragment **FragmentMyWorkDetail**.

**FragmentMyWorkContainer**

- Modifica del metodo **doInBackground()**, il quale esegue l'accesso ai record del database Realm al fine di costruire la lista degli oggetti "Odl": gli ordini di lavoro incompleti vengono popolati sulla base del primo ordine di lavoro recuperato dal database.

**FragmentProfileInfo**

- Modifica del metodo **onViewCreated()** al fine di generare le informazioni del profilo JAM dell'utente in maniera fittizia (informazioni relative all'indirizzo ed al contatto telefonico).

**ResponseManager**

- Modifica del costruttore al fine di evitare il lancio di eccezioni relative a risposte HTTP assenti o che presentano un errore;
- Modifica del metodo **logCrashliticsIfNecessary()** per non riportare un errore a Crashlitics in base al codice di ritorno di una HTTP response.

**CudCedolinoAdapter**

- Modifica del metodo **getItemCount()** per evitare eccezioni di tipo NullPointerException nel caso in cui l'attributo "list" risulti null.

**MonteOreFragment**

- Modifica del metodo **fetchMonteOreTipo()** per ottenere la lista di oggetti "MonteOreTipo" senza effettuare richieste HTTP via ResponseManager;
- Modifica del metodo **fetchMonteOre()** per ottenere la lista di oggetti "MonteOre" senza effettuare richieste HTTP via ResponseManager.

**CartellinoOrologioFragment**

- Modifica del metodo **fetchDataList()** per ottenere la lista di oggetti "CartellinoOrologio" senza effettuare richieste HTTP via ResponseManager.

**TimbraturaTipo**

- Aggiunta dei setter methods.

**QuadraturaFragment**

- Modifica del metodo **fetchDataList()** per ottenere la lista di oggetti "Quadratura" senza effettuare richieste HTTP via ResponseManager.

**CudCedolinoFragment**

- Modifica del metodo **fetchDataList()** per ottenere la lista di oggetti "CudCedolino" senza effettuare richieste HTTP via ResponseManager.

**ReperibilitaFragment**

- Modifica del metodo **fetchData()** per ottenere la lista di oggetti "Reperibilita" senza effettuare richieste HTTP via ResponseManager.

**TimbraturaMotivo**

- Aggiunta dei setter methods.

**TimbratureFragment**

- Modifica del metodo **fetchTimbrature()** per ottenere l'oggetto "Timbratura" senza effettuare richieste HTTP via ResponseManager;
- Modifica del metodo **fetchTimbraturaTipo()** per ottenere la lista di oggetti "TimbraturaTipo" senza effettuare richieste HTTP via ResponseManager;
- Modifica del metodo **fetchTimbraturaMotivo()** per ottenere la lista di oggetti "TimbraturaMotivo" senza effettuare richieste HTTP via ResponseManager;
- Modifica del metodo **showTimbratureMessaggi()** per ottenere la lista di oggetti "TimbraturaMessaggio" senza effettuare richieste HTTP via ResponseManager.

**GiustificativoEditDialog**

- Modifica del metodo **setAction()** per ottenere la lista di oggetti “GiustificativiType” senza effettuare richieste ai servizi di SAP HR -ESS;
- Modifica del metodo **sendData()** utilizzato per creare giustificativi utilizzando i servizi SAP HR – ESS (Employee Self Service), i quali sono raggiungibili solo da rete APN TIM opportunamente configurata sui tablet di ACEA. Gli oggetti “Giustificativi” sono salvati in locale attraverso le SharedPreferences Android: gli attributi dei vari oggetti creati sono salvati in strutture di tipo LinkedHashSet<String> e sono ordinati con l’ausilio di una variabile intera “ordering_var”, cosi da poter creare correttamente la lista di oggetti “Giustificativi” nel momento in cui questa è richiesta.

**GiustificativiListFragment**

- Modifica del metodo **getData()** per ottenere le liste di oggetti “GiustificativiType” e “Giustificativi” senza effettuare richieste ai servizi di SAP HR –ESS. I valori fittizi per creare e popolare gli oggetti “Giustificativi” sono recuperati dalle SharedPreferences, come descritto sopra.

**GiustificativiCalendarioFragment**

- Modifica del metodo **getData()** per ottenere la lista di oggetti “GiustificativiCalendar” senza effettuare richieste ai servizi di SAP HR –ESS. Gli oggetti “GiustificativiCalendar” sono creati sulla base delle informazioni relative ai giustificativi memorizzate nelle SharedPreferences;
- Modifica del metodo **loadDataFromRealm()** per ottenere la lista di oggetti “GiustificativiCalendar” senza effettuare richieste ai servizi di SAP HR –ESS. Gli oggetti “GiustificativiCalendar” sono creati sulla base delle informazioni relative ai giustificativi memorizzate nelle SharedPreferences.