Worklist Client

# Introducción

existe un worklist server (WLS) con la capacidad de notificar a clientes workitems, tomarlos y resolverlos.

# Objetivo

Desarrollar un worklist client (WLC) html5, css3, javascript (Sencha Touch)

# Alcance

en principio que el WLC (Client-Side):

* pueda verse en un web browser mobile (sencha touch + profiles [s3+galaxy tab2]).
  + smartphone: sencha touch x default
  + tablet: ejemplo: no utilizar todo el ancho de la pantalla para los textbox

ver propuesta: <https://docs.google.com/a/2innovateit.com/drawings/d/18EYM5AwqrD3Wk2GBCoMO4z-9oPhPFqliVsDAiguL0Hw/edit>

* pueda ser utilizado en un contenedor phonegap (android, ios)
  + prototipar en chrome - primer sprint
  + incluir en phonegap - siguiente sprint
* que consuma el WLS vía WS REST (json)
* Que se dibuje en el cliente las views necesarias para el caso de negocio a implementar (IMPORTANTE)
* el workitem (solicitud de crédito) no se crea desde la app, sino que ya viene creado.

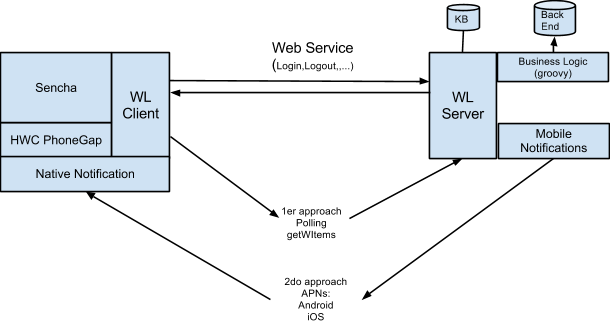
## alcances proxima etapa

* notificaciones nativas
  + <http://aws.amazon.com/sns/>
* uso offline de la app
* creación de solicitudes de crédito
  + un nuevo perfil de vendededor
  + re/utilizar la view existente para poder realizar la creación de un workitem
* implementación dinámica de la-las views necesarias para cada WorkItem
  + con el objetivo de crear pantallas simples reutilizables e independiente de los dispositivos (por ejemplo en html5)

# 

# 

# Solución - Arquitectura



# 

# 

# Caso de negocio a implementar

Gestión de aprobaciones de crédito.

## Actores de negocio:

- vendedores: crean las solicitudes de crédito. (puede ser implementado con un demonio, no es parte del alcance del primer sprint)

- oficiales de backend: aprueban o rechazan los créditos.

## Atributos de una solicitud de crédito

* Datos de Persona
  + ID, LastName, FirstName, Address, Phone, eMail,
* Datos Financieros: Gross Income (mensuales gross),
* Datos del Credito solicitado:
  + Type: Loan, Credit
  + Amount
  + Months (en meses)
  + Rate (en %)
* Others: textbox tipo comment

datos ejemplo de una solicitud:

[{"ID":"26200300",

"LastName":"Barilari",

"FirstName":"Juan",

"Address":"St.Martin 123",

"Phone":"54955555555",

"eMail":"[juan.barilari@2innovateit.com](mailto:juan.barilari@2innovateit.com)",

“InquiryDate”:”2013-09-03”,

"GrossIncome":"20000",

"ProductType":"Loan",

"Amount":"250000",

"Months":"12",

"Rate":"9.45",

"Others":"{“xx”:”xx” }"

}]

## Estados y ciclo:

1. para aprobar

2.1 aprobado

2.2 desaprobado

La validación del paso de un estado a otro es responsabilidad de WL Server, en esta primer etapa no existirán restricciones.

Los estados siguientes de un workitem esta especificado en el WL Server, en esta primera seran los que arriba se nombraron y no se pedira al WLServer cuales son los siguientes estados.

## Actores / Sistemas:

- BackEnd: almacena las solicitudes de crédito.

- WLServer: encola solicitudes de aprobación y notifica al cliente, gestiona las tareas (tomar, liberar, crear, notificar)

- WLClient: login, logout, obtiene la lista de las tareas disponibles, toma una tarea, muestra sus atributos y realiza acciones.

# Notificaciones cliente-servidor

**1er approach**:el cliente realiza ejecuta un mecanismo de pooling sobre el WLS getWorkItems(usuario)

timer configurable de 5 segundos. (conf.js + conf.json)

**2do approach**: web socket client, tipo push.

# 

# 

# Interfaces expuestas por WLS(Marcelo)

Web Service Rest esta disponible en la siguiente dirección:

<http://200.40.234.146:8181/WorkListFrontRest/WLRestApplication/>

Los web methods retornan siempre un objeto WorkListResponse en formato json, el cual esta compuesto por:

* + un entero errorCode, donde 0 = exito y cualquier otro valor indica un error.
  + una cadena msg, donde se describe el error si corresponde.
  + un objeto data, en formato JSon con la información relevante de cada operación

Los WS rest estan accesibles mediante GET pero se debe de poder resolver la invocación del lado del cliente mediante POST por motivos de seguridad y largo en los mensajes (x ejemplo esto es una limitante a nivel de los browser's: explorer, firefox, chrome)

Caso para probar invocación por POST http://200.40.234.146:8181/WorkListFrontRest/WLRestApplication/echo recibe un parametro message y la respuesta es "Echo = message"

Se deben encodear los caracteres para poder pasar mensajes de tipo json en data y metadata

Encoding de los caracteres:

%22 es =

%7B es {

%7D es }

%5B es [

%5D es ]

1. **Crear WI**

**Ejemplo**  http://200.40.234.146:8181/WorkListFrontRest/WLRestApplication/createDirectWorkItem?JPCallback=WLResponse&system=MOBILITYDEMO&userId=jmarcos@android&workItemType=SCT1&metadata=%7B%7D&data=%7B%22vnotitype%22:%22Queue%22,%22vqueue%22:%22Officer%20BackEnd%22,%22dni%22:%22123456789%22,%22apellido%22:%22Casta%C3%B1a%22,%22nombre%22:%22Cacho%22,%22monto%22:%2225000%22,%22periodo%22:%2209/2013-09/2015%22,%22tasa%22:%2223,5%22,%22scoring%22:%229%22%7D

Para el ejemplo anterior los valores sin encodear son los siguientes

* + metadata={} // se guarda en texto plano,locks,queue,priority
  + data={"**vnotitype**":"Queue","**vqueue**":"Officer BackEnd","dni":"123456789","apellido":"Castaña","nombre":"Cacho","monto":"25000","periodo":"09/2013-09/2015","tasa":"23,5","scoring":"9"} // datos del negocio, se almacena ZIPPADA

1. **Evento sobre WI**

**Ejemplo**

http://200.40.234.146:8181/WorkListFrontRest/WLRestApplication/notifyEventDirectWorlList?JPCallback=WLResponse&system=MOBILITYDEMO&userId=jmarcos@android&workItemId=**IDWORKITEM**&eventType=Take&metadata=%7B%7D&relevantData=%7B%7D

IDWORKITEM debe ser reemplazado con el id que se retorna en la creación

Para el ejemplo anterior los valores sin encodear son los siguientes

* + metadata = {} // x ejemplo para enviar a otra cola
  + data = {} // se podria cambiar algún dato de negocio
  + eventType = ToBeApproved -> Take -> Approve o Reject

respuesta a evaluar:

* exito,cuando errorcode=0
  + {"errorCode":0,"msg":"","data":"True"}
* error,cuando errorcode<>0, ejemplo:
  + {"errorCode":-1,"msg":"problema XXX","data":"False"}

1. **Listado de WI para un usuario**

**Ejemplo request:**

[http://200.40.234.146:8181/WorkListFrontRest/WLRestApplication/getAsignedWorkItems?JPCallback=WLResponse&system=MOBILITYDEMO&userId=jmarcos@android](http://200.40.234.146:8181/WorkListFrontRest/WLRestApplication/getAsignedWorkItems?system=MOBILITYDEMO&userId=jmarcos@android)

Parser JSON: <http://json.parser.online.fr/>

Encoder: <http://meyerweb.com/eric/tools/dencoder/>

**Ejemplo rta:**

{"errorCode":0,"msg":"","data":[{"id":1,"idWi":"ddb48c7a-b9c1-4abf-9e4d-09c3f531a586","system":"MOBILITYDEMO","workItemType":"SCT1","workItemVers":"1.0","stateBussines":"ToBeApproved","locked":false,"locker":"jmarcos@android","lockerSystemId":"","metaData":"{\"queues\":{\"queue|officer backend\":{\"notysettype\":\"Queue\",\"notypriority\":1,\"notsystemid\":\"\",\"notysetid\":\"Officer BackEnd\",\"notylifetime\":60000,\"notytype\":1}}}","context":"H4sIAAAAAAAAACXLUQ6CMAwG4KuYPmMcw6nwqPHZeIQ6ii7CitsgMcZDeQYvZsGn9v/+9gWj5+TSsyeo4DzQQJDB+JiXCk5N4yyFxR7t/ehrqWrvxHNdrM1muytFsKe2dTULHzAm/H5Q1HN3CTSbvbFAxz5NN9oopST3FBzPX6pcaZUXy/800iWMOJ0W2ZSi5eD8VaCE9w+LBFCxsQAAAA==","lastProcessDate":"2013-08-29","events":null,"eventsRetrivedFromDb":[],"wasMaterializedFromDB":true,"mustMaterialize":false,"dispatched":true,"finalized":false,"clientMachine":null,"coordinatorMachine":null,"procesedEvents":[],"allProperties":null}]}

context: {

"id":1,

"customer":

{

"dni": "3434",

"surname": "Barilari",

"name": "Juan",

"address": "Address",

"telephone": "12345678",

"email": "jb@gma.com"

},

"income": 5353,

"credit":

{

"type":"credit",

"amount":5000,

"period":12,

"rate": 10

}

"date":"20/02/2012",

"subject":"test-1",

"state":"pending"

}

Para el ejemplo anterior los valores sin encodear son los siguientes

* + metadata={}
  + data={}

1. **Tomar WI**

**Ejemplo**

http://200.40.234.146:8181/WorkListFrontRest/WLRestApplication/getWorkItem?JPCallback=WLResponse&system=MOBILITYDEMO&userId=jmarcos@android&workItemId=**IDWORKITEM**

IDWORKITEM debe ser reemplazado con el id que se retorna en la creación

Para el ejemplo enterior los valores sin encodear son los siguientes

metadata={}

data={}

1. **Login**

Ejemplo

http://200.40.234.146:8181/WorkListFrontRest/WLRestApplication/login?JPCallback=WLResponse&user=jmarcos@android&pass=1234&rol=Officer BackEnd&system=MOBILITYDEMO&protocol=HTTP&callback=http://notificationServer:8080/NotifyReceptor&metadata=%7B%7D

Para el ejemplo anterior los valores sin encodear son los siguientes

metadata={}

1. **Logout**

Ejemplo

http://200.40.234.146:8181/WorkListFrontRest/WLRestApplication/logout?JPCallback=WLResponse&userId=jmarcos@android&system=MOBILITYDEMO&protocol=HTTP&callback=http://notificationServer:8080/NotifyReceptor

Para el ejemplo anterior los valores sin encodear son los siguientes

metadata={}

1. **Cancelar WI**

**Ejemplo**

http://200.40.234.146:8181/

Dudas (Marcelo):

Valores posibles:

**systemID**: corresponde al identificador del sistema externo.

**userId:** identificador del usuario si se requiere registrar el usuario por múltiples dispositivos se debe de usar la siguiente nomenclatura userId@dispositivo

**workItemType, eventType:** estos valores dependen de la definición del caso concreto.

JMJB: si, tienes las siguientes transiciones sin restricciones ni reglas.:

- **inicial**: Para Aprobar (el dataentry carga una solicitud de crédito, por ahora es un demonio que crea al workitem)

- **siguientes**: (el usuario de esta app)

- Aprobado

- Desaprobado



Para lo cual se deben de definir los estados, las acciones a tomar en cada estado, los eventos y las reglas (si corresponde) para los cambios de estado.

**Roles:** Lista de roles ¿hay web method? no habria que crearlo.

**ErrorCode + ErrorDescription:** no se contempló para la versión actual

**Metadata, relevantData, Data:** estos campos se encuentran en formato JSON y tienen sentido en el contexto de la evaluación de las reglas y acciones, como también en la notificación desde la WorkList hacia los clientes

**Especificación WorkItem:**

idWi - dentificador del work item

systemId - identificador del sistema (que usa la WL)

workItemType - tipo de workItem

workItemVers - version tipo de workItem

stateBussines - estado del work item

locked - indica si esta locked

locker - usuario que lo tiene locked

lockerSystemId -

metaData - metadatos en formato json

context - datos de contexto

# 

# 

# Views necesarias

P1- **Login**

* Usuario (text)
* Contraseña (text)
* Rol (combo)
* Mensaje de error (label)
* btnLogin: dispara WMethod Login
  + login correcto: muestra view P2
  + login incorrecto: muestra ALERT con rta del WMethod

P2- **Grid de workitems** + Boton de **Logout**.

* Grilla
  + Icono (tipo),
    - ..\images\workitemType.png
    - tener en cuenta las resoluciones y separa el repositorio de img de la lógica de obtención de img según resolución.
  + Fecha
    - formato de fecha, tal vez usar el conf.json
    - utilizar campo “inquiry date” de la solicitud
  + Subject
    - en la tablet seguro que entran más caracteres...
    - en smartphone menos y luego “...”
    - persona: [Apellido] - [Nombre] - [monto]
  + Estado: del workitem
  + Acción (una acción por defecto, ver relevant data), disparada por un TAP en el WI
    - ejecuta WMethod (X) que pide que el cliente tome el WI y muestra la P3

P3- **Workitem** (caso de aprobacion de creditos)

mostrar lista de atributos (label + textbox)

acciones del WI

btnAprobar: envía WebMethod (X) y si todo OK vuelve a P2

btnDesaprobar: envia WebMethod (X) y si todo OK vuelve a P2

[o combo con opciones Aprobar & Desaprobar + btn Guardar]

btnVolver: vuelve a la P2

**Notas:**

* P1 y P2 son genéricas para cualquier implementación de WorkList Client.
  + entregar concepto de marca blanca (CSS)
  + Mejora futura: look & feel especial para cada cliente
* La P3 es específica para el caso de uso, es el primer sprint utilizaremos.
* La idea en un sprint futuro es poseer una base de conocimientos con la especificación de la o las views necesarias para atender cada tipo de workitem y cada uno de los distintos dispositivos.
* screens a contemplar: samsung s3, galaxy note 2 10’
* para cada consumo de WS REST chequear que el conjunto de datos:
  + errorCode=0, caso contrario la app devuelve un ALERT(errorDescription)
  + errorDescription=’’.

# Procesos

una vez logueado, el proceso de obtener los WI para cada usuarios será un pooling desde el cliente hacia el servidor.

# 

# 

# Notas

detalle y set de pruebasd con REST

<https://docs.google.com/a/2innovateit.com/spreadsheet/ccc?key=0AnxvibseGmYhdFVjWUdCbFlEdTFkUG04eEJCeHNNNXc#gid=0>

prototipo

<http://ksencha.99k.org/app.html?deviceType=Phone>

[http://ksencha.99k.org/app.html?deviceType=](http://ksencha.99k.org/app.html?deviceType=Phone)Tablet

Parser

<http://json.parser.online.fr/>