Preliminares a la **comunicación Android con GLAMPs** – N[1]

Créditos

Abstract.

Lo directo, cuando el desarrollador Android quiere acceder a la red Elgg es usar **oAuth** [1] y **Account** [2]. Una vez que el cliente puede manejar los ***tokens*** **que el servidor reparte** [2.1] puede **convertir sus clases Java nativas a JSON** [3] y enviar **peticiones** síncronas [4.1] dentro del **Adaptador de sincronización** [5] o asíncronas [4.2] en cualquier punto de la aplicación. El **plugin webservices** [7] junto con algún otro plugin que extienda métodos responderán a esas peticiones enviando **arrays de objetos** JSON [8].

[1] **Write your own Android Authenticato**r: http://blog.udinic.com/2013/04/24/write-your-own-android-authenticator/
[2] **Android.Account dev pages**: https://developer.android.com/reference/android/accounts/Account.html
[2.1] **Never your asked?** http://stackoverflow.com/questions/7030694/**why-do-access-tokens-expire**
[2.2] **Enhanced plugin ApiAdmi**: https://elgg.org/plugins/1105480
[3] **A Java serialization/deserialization library that can convert Java Objects into JSON and back.** https://github.com/google/gson
[4.1] **Go salmon**: http://stackoverflow.com/questions/16904741/**can-i-do-a-synchronous-request-with-volley**
[4.2] **Asynchronous HTTP Requests in Android Using Volley** http://arnab.ch/blog/2013/08/asynchronous-http-requests-in-android-using-volley/
[5] **Write your own Android Sync Adapter**: http://blog.udinic.com/2013/07/24/write-your-own-android-sync-adapter/
[7] **Docs on Webservices plugin**: http://learn.elgg.org/es/stable/guides/web-services.html
[8] **Ver lista de métodos que expone un sitio elgg visitando su url**: /services/api/rest/json/?method=system.api.list

1 **No tiene valor técnico, es pieza artística. – nadievstodos.tumblr.com – Verano 2016 – Iberia**

Forward 1.

De acuerdo, en este punto, la **conexión de Adaptadores** [1] que c**arguen cursores a listas**, quizás alguna con *swang* a borbotones [2] que tenga opciones en botonera cargadas al vuelo completa una posible vía de interacción desde Android.

[1] **Crear ListView con cursor:** http://www.sgoliver.net/foro/viewtopic.php?t=233
[2] **A swipe menu for ListView:** https://github.com/baoyongzhang/SwipeMenuListView

Forward 2.

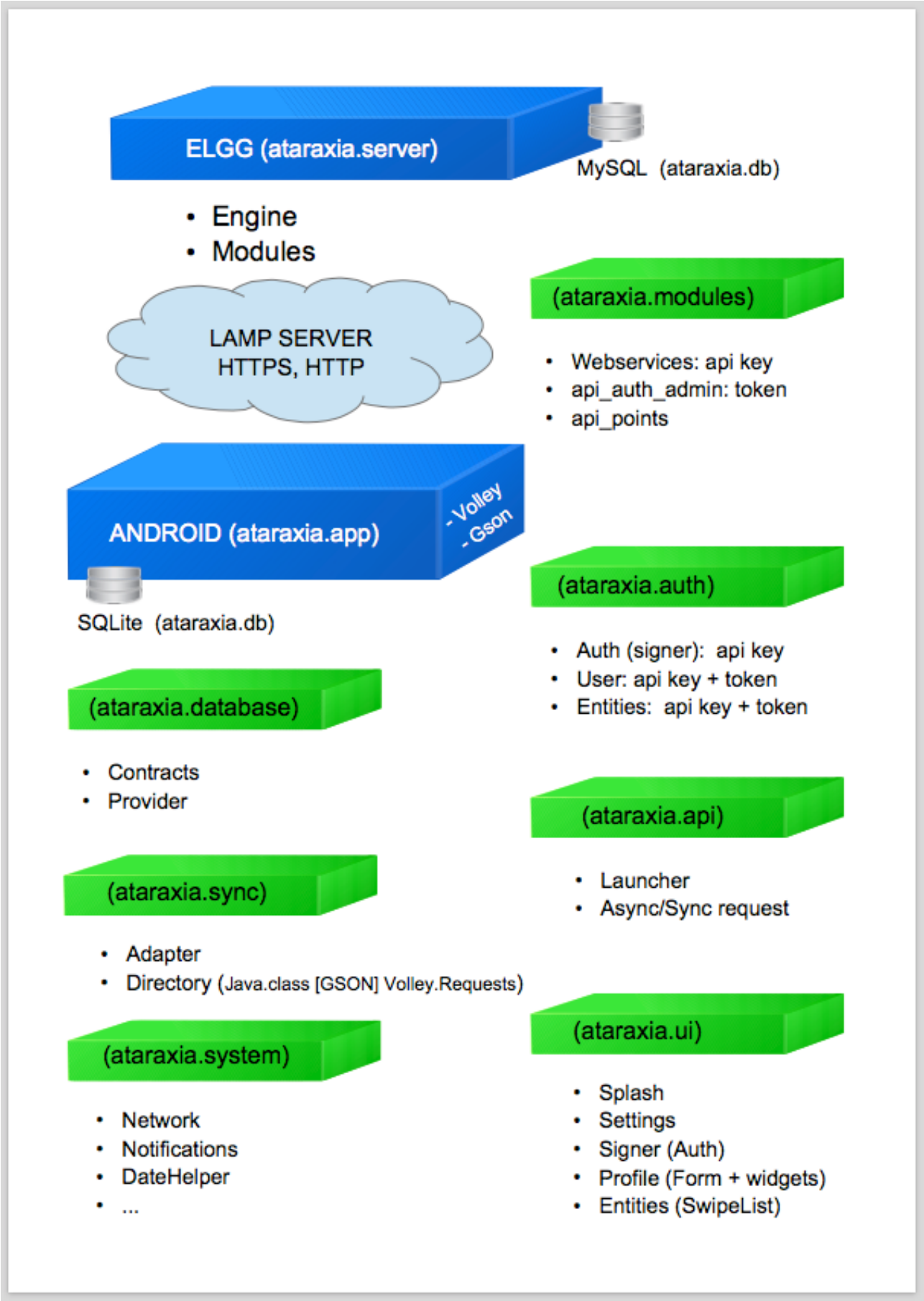
Pero, aunque lo anterior tiene todo el jugo y juego: el título de este micro ensayo sugiere algo, digamos, más allá.

Ensayando, se sabe que no se habla libre de error sino liberando el error del hablar, tomando notas y tal (...), sin querer interrumpir en el buen progreso de lo no ensayado sino planeado, etc.

Decimos: **MicroEnsayo Más Elgg-UX-browsers** y, porque todas y cada una de las palabras del título han sido escogidas atendiendo a criterios de peso específico semántico, tenemos gran parte del microensayo relatado.

(continuará) http://aleph1888.github.io/ataraxia_archive/

Schema


ELGG (ataraxia.server)
MySQL (ataraxia.db)
• Engine
• Modules
LAMP SERVER
HTTPS, HTTP
(ataraxia.modules)
• Webservices: api key
• api_auth_admin: token
• api_points
ANDROID (ataraxia.app)
- Volley
- Gson
SQLite (ataraxia.db)
(ataraxia.auth)
• Auth (signer): api key
• User: api key + token
• Entities: api key + token
(ataraxia.database)
• Contracts
• Provider
(ataraxia.api)
• Launcher
• Async/Sync request
(ataraxia.sync)
• Adapter
• Directory (Java.class [GSON] Volley.Requests)
(ataraxia.system)
• Network
• Notifications
• DateHelper
• ...
(ataraxia.ui)
• Splash
• Settings
• Signer (Auth)
• Profile (Form + widgets)
• Entities (SwipeList)

(contexto, antecedente)

MicroEnsayo Más Elgg-UX-browsers

Creada por aleph hace una hora Categorías: Filosofía expansion

Escritura automática en el autobús, aprovechando un trayecto de dos horas, desde el lugar en que construyo un boceto sobre lo expuesto hasta la casa de la persona que más q...

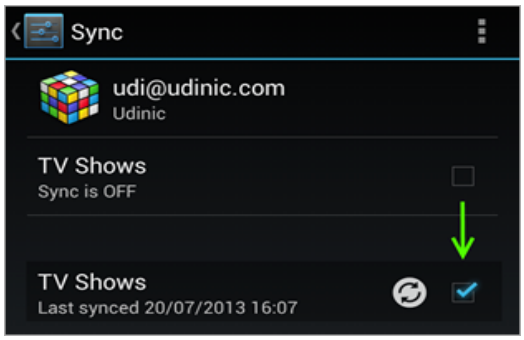
Entonces, sigamos donde lo dejamos, **¿dónde lo habíamos dejado?**

==========

Ahora el coder Android ha logrado poder jugar con el botoncito de sincronización [1], lo cual le pega tirones al servidor, performando la sincronización base[2]:

[1] **Programatically click the sync button**: http://stackoverflow.com/questions/4465765/how-to-code-a-sync-now-operation-on-android

[2] **Shit! This is happening:** http://stackoverflow.com/questions/34685069/android-syncadapter-onperformsync-not-called



Por dónde seguimos? Accediendo por la UI

==========

Miramos ahora por el otro lado, en lugar de mirar la parte de atrás, en la carga de datos entre un servicio alejado de la capa de usuario en el Android y un servicio REST remoto en un GLAMP, ahora miramos por la parte de adelante...

... la idea es hacer un pequeño puente entre el usuario *logueado* y una visita normal con el navegador. Es decir, usamos un **WebViewer [1]** que tenga un **WebClient** y un **WebChromeClient** [2] para cargar una cierta api de urls que se forman, como todo elgg-*coder* saber a partir de tres claves: *"owner, friends, all"*; y otros tantos verbos casi REST: *"add,..."* para manejar las distintas entidades [3].

[1] **Docs on WebView Class**: https://developer.android.com/guide/webapps/webview.html

[2] **Did you ask your self...** http://stackoverflow.com/questions/2835556/**whats-the-difference-between-setwebviewclient-vs-setwebchromeclient**

[3] **Docs on Entities Class** http://reference.elgg.org/entities_8php.html

Un conciso: loguear el navegador.

==========

Gracias al comando loadUrl [1] podemos lanzar peticiones a nuestro WebView.

Ahí podemos poner las url's de elgg [2] o lanzar comandos javascript [3].

[1] **how to load a url to webview in android?** http://stackoverflow.com/questions/11288611/how-to-load-a-url-to-webview-in-android

[2] **Docs on Encaminamiento** http://learn.elgg.org/es/stable/guides/routing.html

[3] **Best code examples for WebView loadUrl method** *(android.webkit.WebView.loadUrl)*
http://www.codota.com/android/methods/android.webkit.WebView/loadUrl

Entonces, parece lógico que, como todo *elgg-coder* sabe, si haces una petición a Elgg antes de estar validado serás redireccionado a la pantalla de autorización. Por tanto, **si usamos el evento onPageFinished [1] para comprobar si la url cargada se corresponde con la esperada**, la primera vez no corresponderá, porque habremos aterrizado en el formulario de autorización, **podremos enviar los datos al formulario vía loadUrl y presionar el botón de envío** teniendo el WebView en un estado de ***visibility*** *invisible* (que como todo *android-coder* sabe tiene al control en pantalla pero no visible, diferente de *gone* que lo extrae) hasta que cargue la validación y, de nuevo, en *onPageFinished* esta vez, tengamos la url que buscábamos y ya podemos pasar a *visible*.

[1] **Have you ever been in such...** http://stackoverflow.com/questions/6719814/**onpagefinished-never-called-webview**

Lo de arriba es *cool* pero ortodoxo usar good staff para estandarizar el proceso [1].

[1] **How to do brower level authorization instead of UI one...** http://stackoverflow.com/questions/2585055/using-webview-sethttpauthusernamepassword

One step beyond

==========

Después, ¿miramos de agregar una interfaz [2] en cada lado [3] para fluir mejor?

[2] **Diseño Android: Interfaces web con WebView** https://danielme.com/2013/02/14/diseno-android-interfaces-web-con-webview/

[3] **Docs on Javascript moduling AMD on Elgg** http://learn.elgg.org/es/stable/design/amd.html

(Continuará) http://aleph1888.github.io/ataraxia_archive/