GIUSEPPE GARIPOLI

Senior Software Engineer

53 bld Raymond Poincaré, 06160 Antibes (France)

phone: +33 6 60 51 10 80 email: garipolig@gmail.com

www.linkedin.com/in/ggaripoli

Micro Blog Android Application

https://github.com/garipolig/AndroidBlogAppREST

1. Intro

The goal of this project is to implement an **Android Application** that communicates with a Web Server to retrieve information about *authors*, *posts* and *comments*.

The application is using *Google Volley* as HTTP library to communicate with the Server and to perform the caching and retry mechanisms.

2. Usage

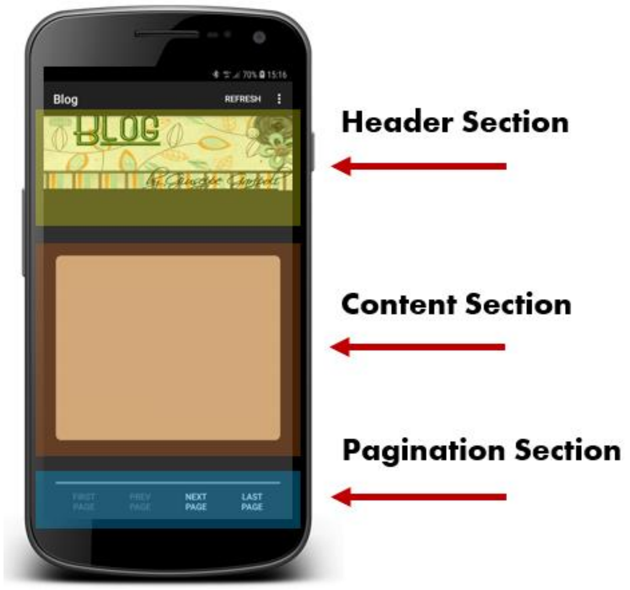
When the user starts the application, the first screen shows the list of authors available on the Blog.

Once a specific author is selected, the user is redirected to another page listing all the posts related to that author, with the addition of further information about the author, such as photo, email, address.

Finally, once a specific post is selected, the user is redirected to the final page which is listing all the comments related to that post, with the addition of further information about the post itself such as picture, date, title, content.

3. Layout and Transitions

An effort has been done to communize the different pages, trying to create a modular layout consisting in a base page, which is extendible depending on the needs.



The **MainActivity** is the Application entry point, taking care of listing the authors available on the Blog:

- *Header Section*: contains the cover image of the Blog.
- *Content Section*: shows the list of authors.
- *Pagination Section*: contains the buttons to move through the different pages of authors.

The **PostsActivity** is the next page, taking care of listing the posts related to a given author:

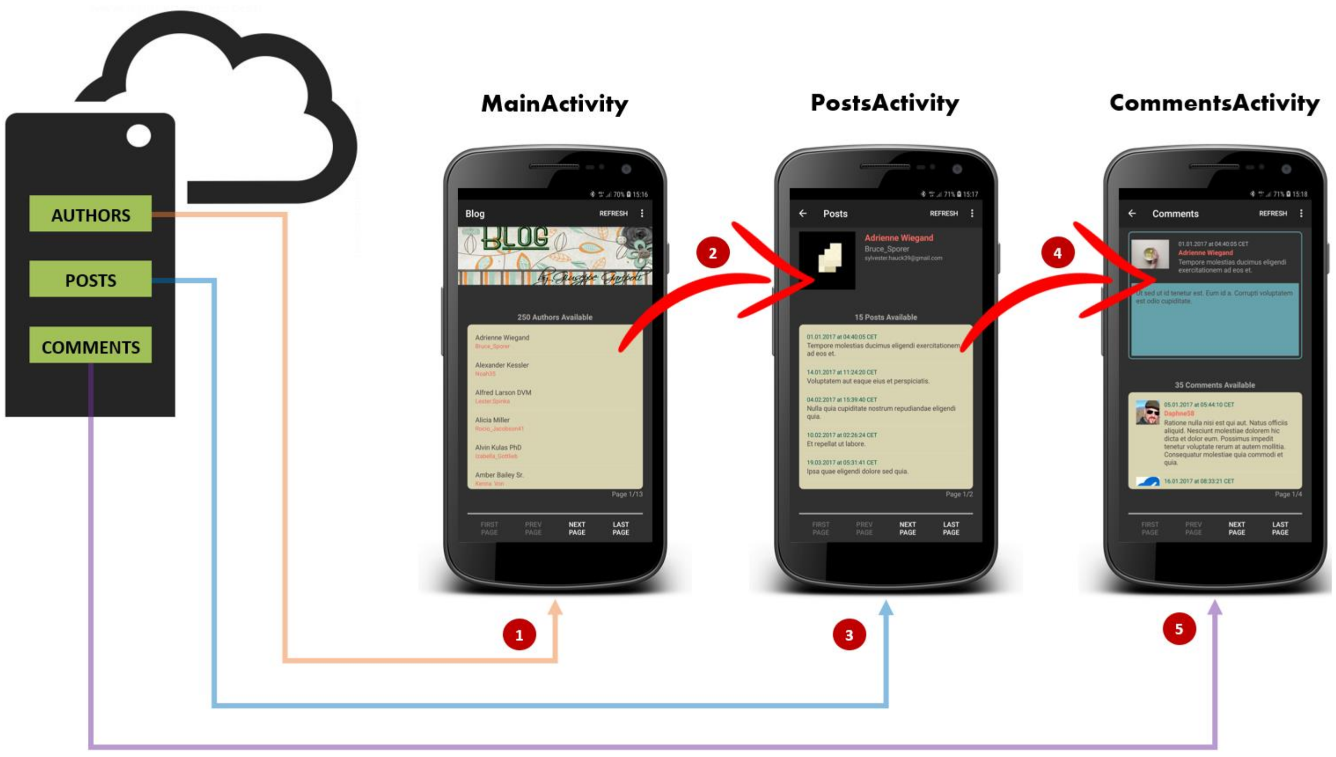
- *Header Section*: contains the details about the author.
- *Content Section*: shows the list of posts.
- *Pagination Section*: contains the buttons to move through the different pages of posts.

The **CommentsActivity** is the last page, taking care of listing the comments related to a given post:

- *Header Section*: contains the details about the post.
- *Content Section*: shows the list of comments.
- *Pagination Section*: contain the buttons to move through the different pages of comments.

The three Activities are inheriting from a common **BaseActivity** which is hosting most part of the code.

Here's a sequence diagram, showing the page transitions and when the server is interrogated:

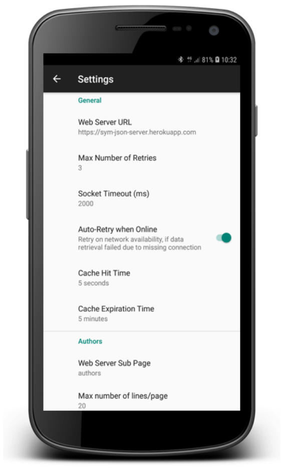


Note: An extension of the base Android class which is handling the table content has been done, to be able to display for each row a picture and a text.

4. Settings

The application is fully configurable, nothing is hardcoded. This is possible thanks to the **SettingsActivity**.

The possible configurable parameters are shown in the picture below:



General

Web Server URL
https://sym-json-server.herokuapp.com

Max Number of Retries
3

Socket Timeout (ms)
2000

Auto-Retry when Online
Retry on network availability, if data retrieval failed due to missing connection

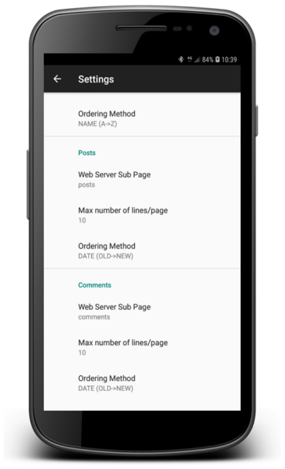
Cache Hit Time
5 seconds

Cache Expiration Time
5 minutes

Authors

Web Server Sub Page
authors

Max number of lines/page
20



Ordering Method
NAME (A->Z)

Posts

Web Server Sub Page
posts

Max number of lines/page
10

Ordering Method
DATE (OLD->NEW)

Comments

Web Server Sub Page
comments

Max number of lines/page
10

Ordering Method
DATE (OLD->NEW)

The user can change at any time those parameters.

A specific action will be performed to allow this request to be immediately effective (e.g. change the number of authors displayed on the table or the alphabetical order).

5. Communication with the Server

The application is using *Google Volley* as HTTP library to communicate with the Server and to perform the caching and retry mechanisms.

Google Volley offers the following benefits:

- Automatic scheduling of the network request (a queue is used)
- Response caching
- Cancellation under demand of the enqueued requests not yet sent to the Server
- Easy customization of cache/retry mechanisms

The default implementation of the *Google Volley* classes has been extended, to be able to:

- Configure the caching mechanism
- Retrieve the *Header* present in the Server response (not parsed by default)

The *Cache Hit Time* and the *Cache Expiration Time* are configurable via the Settings.

Once a request is done, the Server won't be interrogated if the response to that request is already present on the cache and not yet expired.

The cached entry will be given back to the application. This is valid for both texts and images.

Note: the singleton pattern has been used for the class which is interacting with the Server using the *Google Volley* APIs, to have a unique entry point for all the Activities.

6. Pagination Mechanism

Since the information on the Web Server can grows indefinitely, we cannot afford to load the full list of authors/comments/posts on the UI in one-shot.

Instead all the requests sent to the server will make usage of the *_page* and *_limit* parameters.

The *_limit* parameter is configurable through the Settings.

Here's what happens each time the user enters in a new page (Activity):

- The current Activity will start asking to the Web Server the page 1 (default initial request).

- The Server will answer with a JSON response containing the content to show on the table and a header containing useful information such as:

    1. *Total Number of Items* present in server (regarding the information asked: authors/posts/comments). This value has been used together with the current page number to show a page counter at the bottom of the table (e.g. page 2/4).

    2. *Lin*k Section, containing the URL needed to ask for the first/previous/next/last pages. This information is used to associate an action to the *Pagination Buttons* at the bottom of the page.

To summarize, it's only the first request which is hardcoded to *_page=1* once an Activity starts, but starting from that moment, once the buttons are associated to the correct action, everything will be handled dynamically, depending on which button the user is clicking. At each click of a button a new page will be loaded and all the buttons will be updated accordingly with the new associated URLs.

Furthermore, the buttons will be in a disabled/enabled state, depending on which action is possible to perform in a given page (e.g. the *First Page* button is disabled if we are already at the first page).

7. Refresh Mechanism

A refresh button has been added in the Application taskbar: once clicked, the cache will be deleted, and a fresh information will be retrieved from the Server.

Furthermore, an additional feature has been implemented which is performing an *auto-retry* of the last request that has failed because of missing connection (no network available on the phone).

Once back to *connected* state, the last request will be automatically re-sent. This is something that can be enabled/disabled through the Settings.

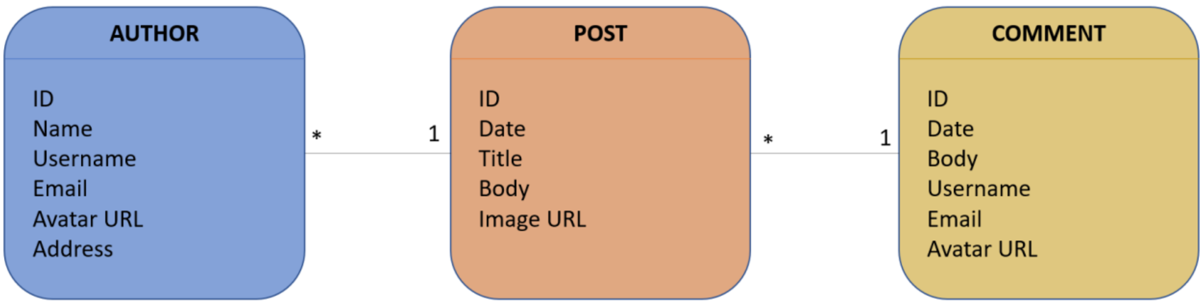
8. Minimization of the Server requests

Each activity is performing a specific request to the Web Server, related to the type of information that can handle.

For example, once we move from the **MainActivity** to the **PostsActivity**, in addition of the list of posts that must be retrieved from the Server, also the *Header* section must be populated with the details about the specific author. But this information is not asked to the Server by the **PostsActivity**, but it's instead passed by the **MainActivity** through intent, during the page transition. This allow us to minimize the number of requests sent to the Server.

The same occurs when moving from the **PostsActivity** to the **CommentsActivity**: the information about the specific posts is passed via intent.

Three *parcelable* objects have been created for this purpose, which are reflecting the structure of the Server database (same attributes):



The Post is including as attribute full Author object instead of a simple Author ID.

The Comment is including as attribute a full Post object instead of a simple Post ID.

A method *isValid()* has been associated to each object, to simplify the validation made by each Activity.

Note that the attributes are not all mandatory to make the object valid (e.g. *avatar image* not needed).

Each time there is an invalid information, it's skipped and not shown on the table.

9. Cancellation Mechanism

The cancellation mechanism of *Google Volley* has been used.

Each Activity can cancel all the previous requests (which are tagged with its *Class Name*) that have not been sent yet to the Server.

This is useful when for example the user is clicking very fast on the Pagination Buttons, without waiting for the Server response. The current click action will cancel all the previous ones.

10. Error Handling

Google Volley is not performing by default any retry mechanism in case of error when communicating with the Server.

This behavior has been changed by adding the possibility to decide the *Maximum Number of Retries* and the *Socket Timeout* in the Settings page.

Once a failure occurs after all the retries, the Application is notified so that an error message can be displayed to the user.

In addition, the *Pagination Buttons* are disabled in case of error.

No further retries are automatically performed, unless the user manually clicks on the Refresh Button.