PET FINDER

ANDROID & JAVA

Contributors
Soyoun S. Lim and **Derya O. Kurin**

INDEX

1. PET FINDER

Petfinder is an internet company that operates the largest online pet adoption website serving all of North America. The company reports that it currently lists "more than 315,000 adoptable pets from nearly 14,000 animal shelters and rescue groups." A commercial enterprise founded in 1996, it is now owned by Nestlé Purina PetCare Company and reports that it has facilitated more than 22 million pet adoptions as of 2013. Most of the pets listed on Petfinder are dogs and cats, but they list all types of animals available from shelters and rescue groups, from small fish, reptiles and birds to horses and livestock. (retrieved from wikipedia.org)

https://www.petfinder.com/developers/v2/docs/#using-the-api

2. Resources

UI : user interface
https://www.figma.com/file/fUi6K4HGPZjiNO1agkLFqs/PET-FINDER?node-id=0%3A1

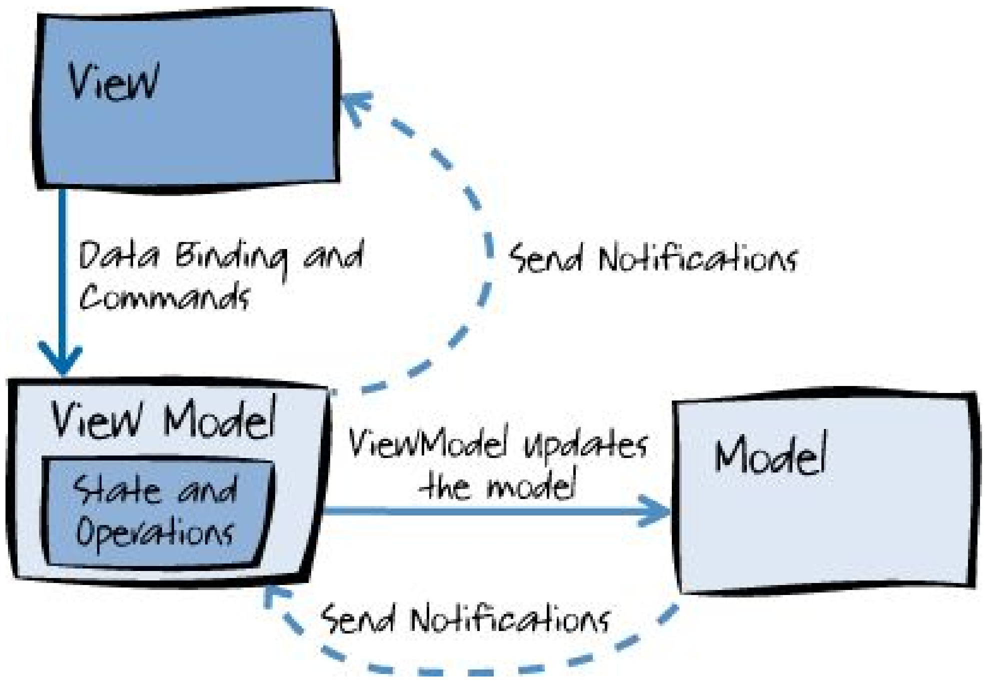
API: **https://www.petfinder.com/developers/v2/docs**
Authentication: Token based authentication
https://android.jlelse.eu/token-authorization-with-retrofit-android-oauth-2-0-747995c79720

Create account from pet finder and get key and secret
Api_key
Secret
https://www.petfinder.com/user/developer-settings/

Libraries use:
Retrofit: https://square.github.io/retrofit/
Http Logger: https://github.com/square/okhttp/tree/master/okhttp-logging-interceptor
Toasty: https://github.com/GrenderG/Toasty
Glide: https://bumptech.github.io/glide/doc/generatedapi.html

Flow of activities:
MainActivity > ResultActivity > DetailActivity > RequestActivity

3. ***Project Structure :*** **depicting MVVM architecture android**

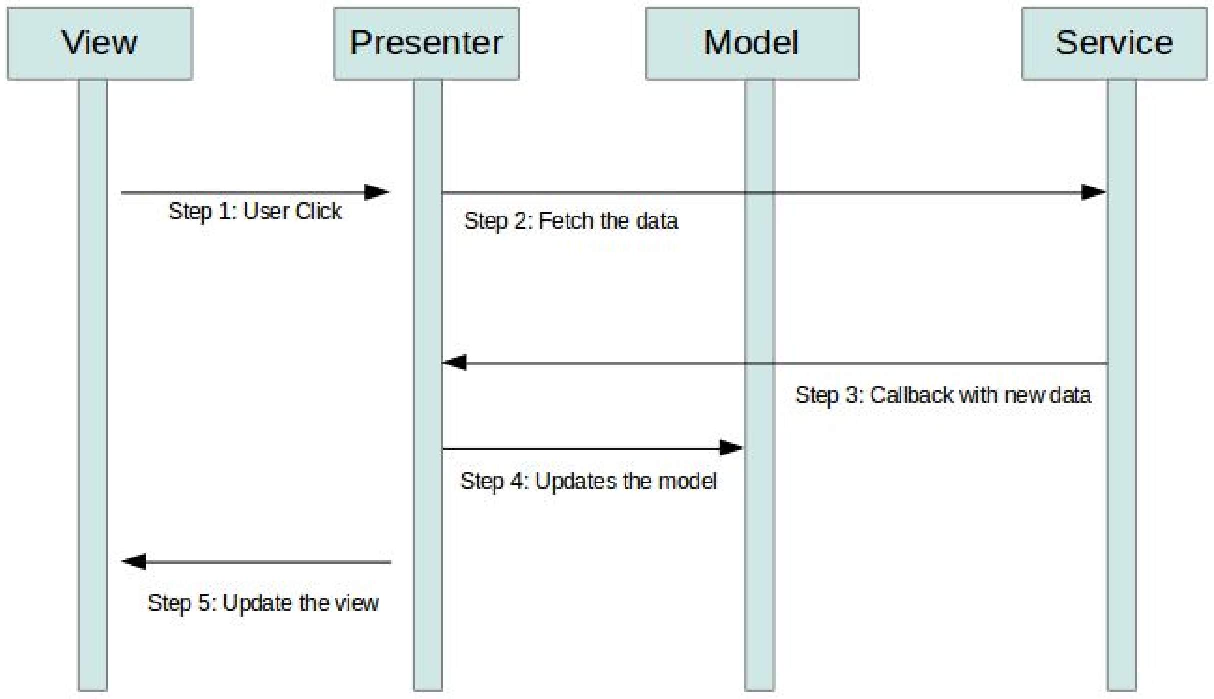


MVVM (Model – View – ViewModel) is an architectural design pattern for software development. MVVM architecture in android provides a better way of dealing with the components by making them loosely coupled.
In MVVM every components has its own task to be furnished so they are faster as well.

MVVM architecture android :

Model	**Model deals with repository related components i.e., data may be from local database or remote api's. Data is accessed from model and is passed on to View-Model for further processing business logic.**
View	**View's are the UI related components which mostly deal with Activity's Fragment's and their layout files, all together constitute to View.**
View-Model	**The main business logic in the application is carried out in this module. Data is received from Model and processed accordingly and emits / exposes data which is observed by View.**

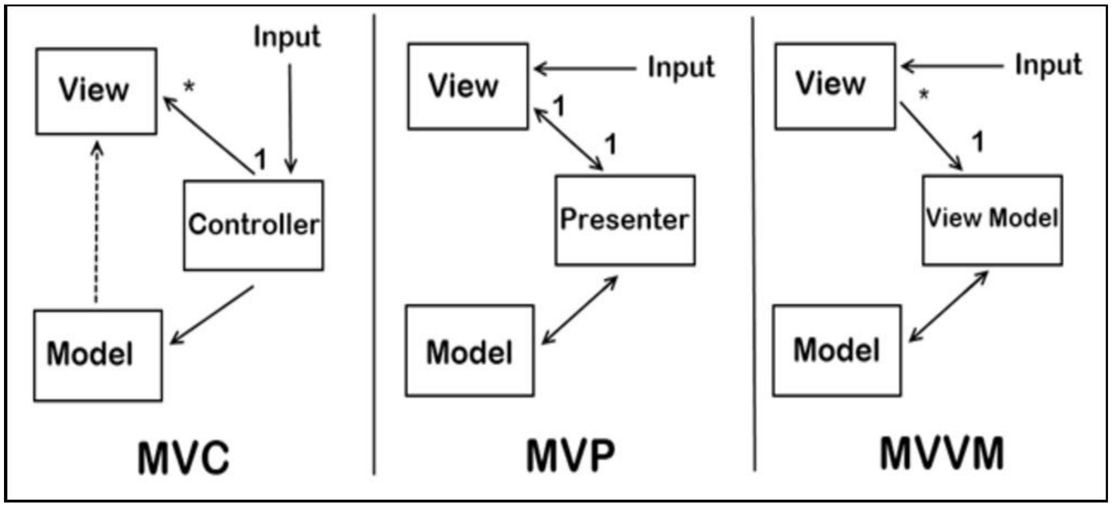
Data Binding: Another key component with this pattern which makes the handling / accessing of UI components much faster, also avoids boilerplate code(repetition).
Data binding plays a vital role in app performance as initializing components and setting data to them with huge number of UI components is a time taking task**.**



Advantages :

- MVVM breaks the tight coupling between UI and business logic makes it easier for unit testing.
- Design, development, data related components are purely independent so multi developers working on the project will have a good scope to work on individual components.
- Using observers ViewModel is independent of the View.
- Individual components make code re-usability and add up to app performance.
- Maintenance of code is much simple, following a appropriate project structure.
- Advanced components like LiveData add up to MVVM architecture providing *lifecycle* aware app transactions which reduces app crashes.
- Memory leaks, app crashes are handled in this pattern.
- Also the most happening scenario's like outdated UI i.e., Activity or fragment not available at UI update after api calls also handled by making use of life cycle aware components.

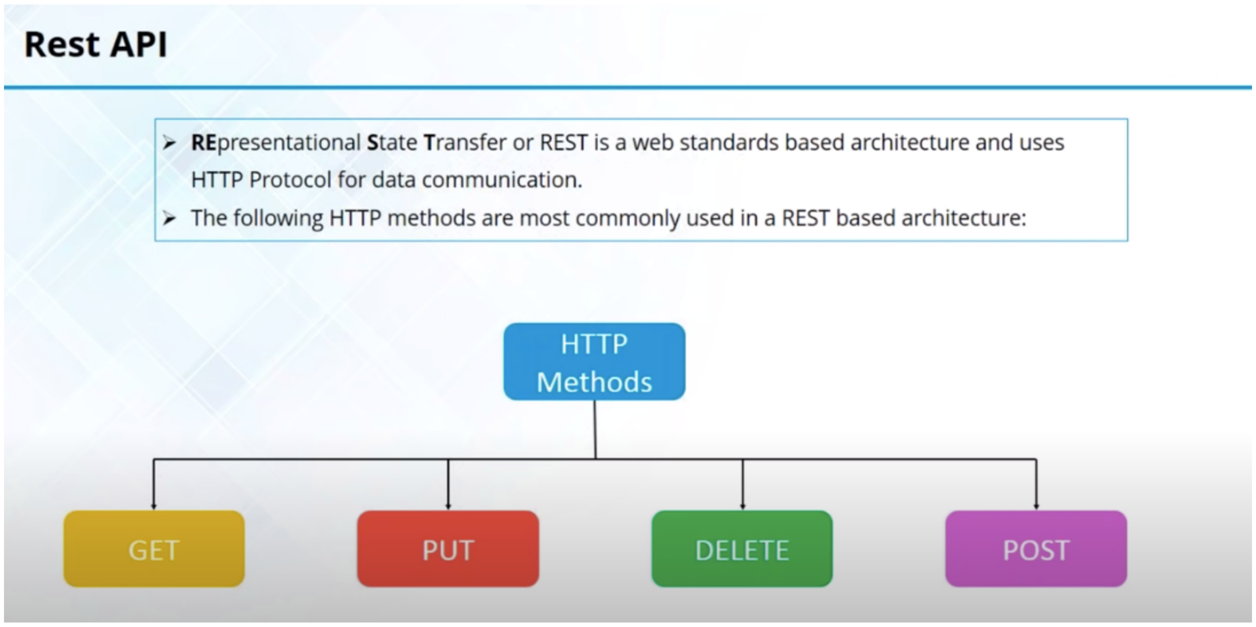
LiveData is an observable data holder that is lifecycle aware. Inside your View you can add an observer to the LiveData so that when the LiveData is updated, your View (for example an Activity or Fragment) is notified so that you can update your user interface with the new data. The ViewModel is generally used to update the LiveData, however LiveData can also be modified in the repository on certain events such as handling an API response.



http://www.androidcoding.in/2020/05/05/android-livedata-tutorial/

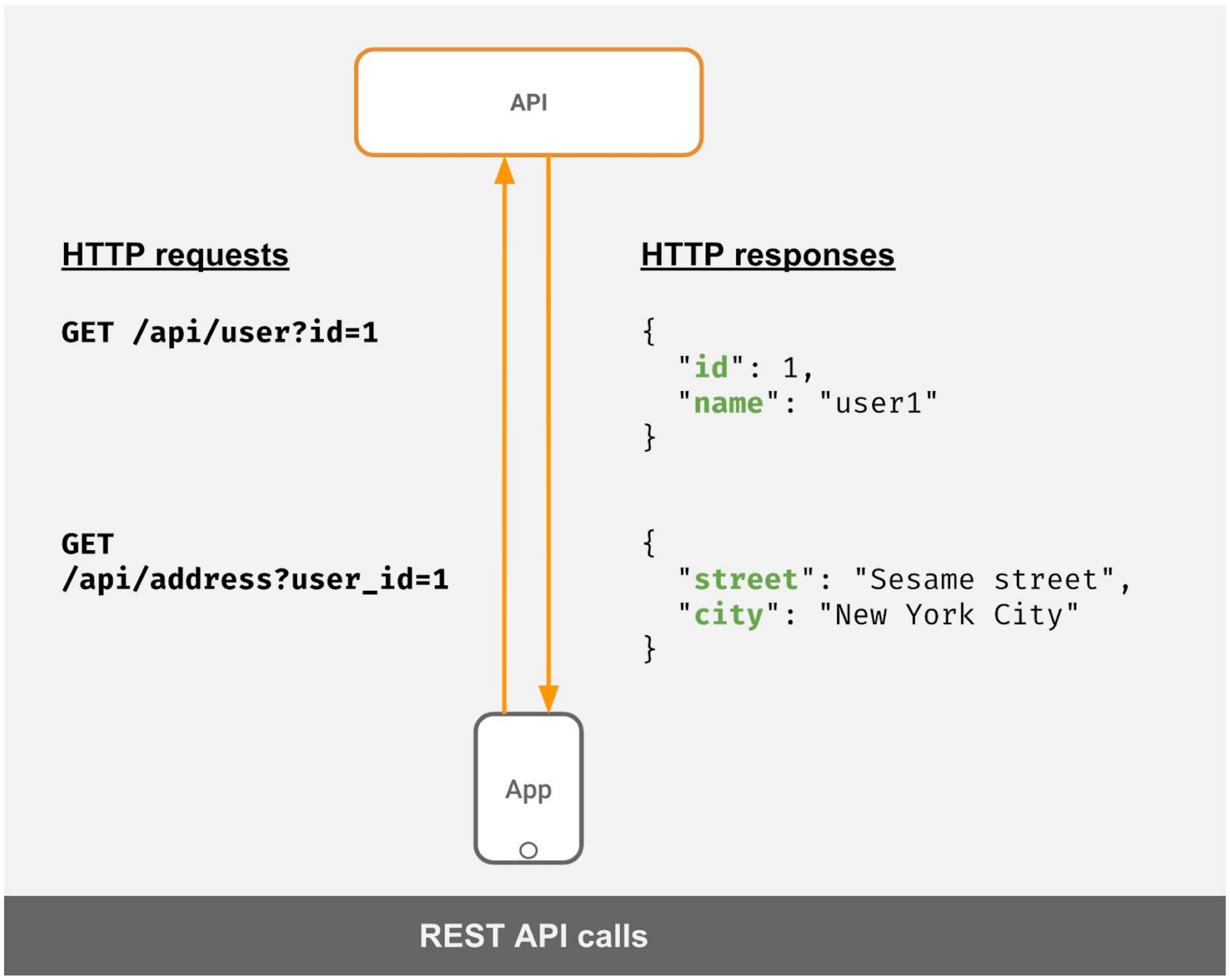
4. REST API

An API is like a cog that allows two different systems to interact.
*(Image from **Brent 2.0**, **spinning gears**,*
***CC BY-ND 2.0**.)*



Rest API

- REpresentational State Transfer or REST is a web standards based architecture and uses HTTP Protocol for data communication.
- The following HTTP methods are most commonly used in a REST based architecture:



HTTP requests

```
GET /api/user?id=1

GET
/api/address?user_id=1
```

HTTP responses

```
{
  "id": 1,
  "name": "user1"
}

{
  "street": "Sesame street",
  "city": "New York City"
}
```

https://idratherbewriting.com/learnapidoc/docapis_what_is_a_rest_api.html

5. **Retrofit** VS **Volley**

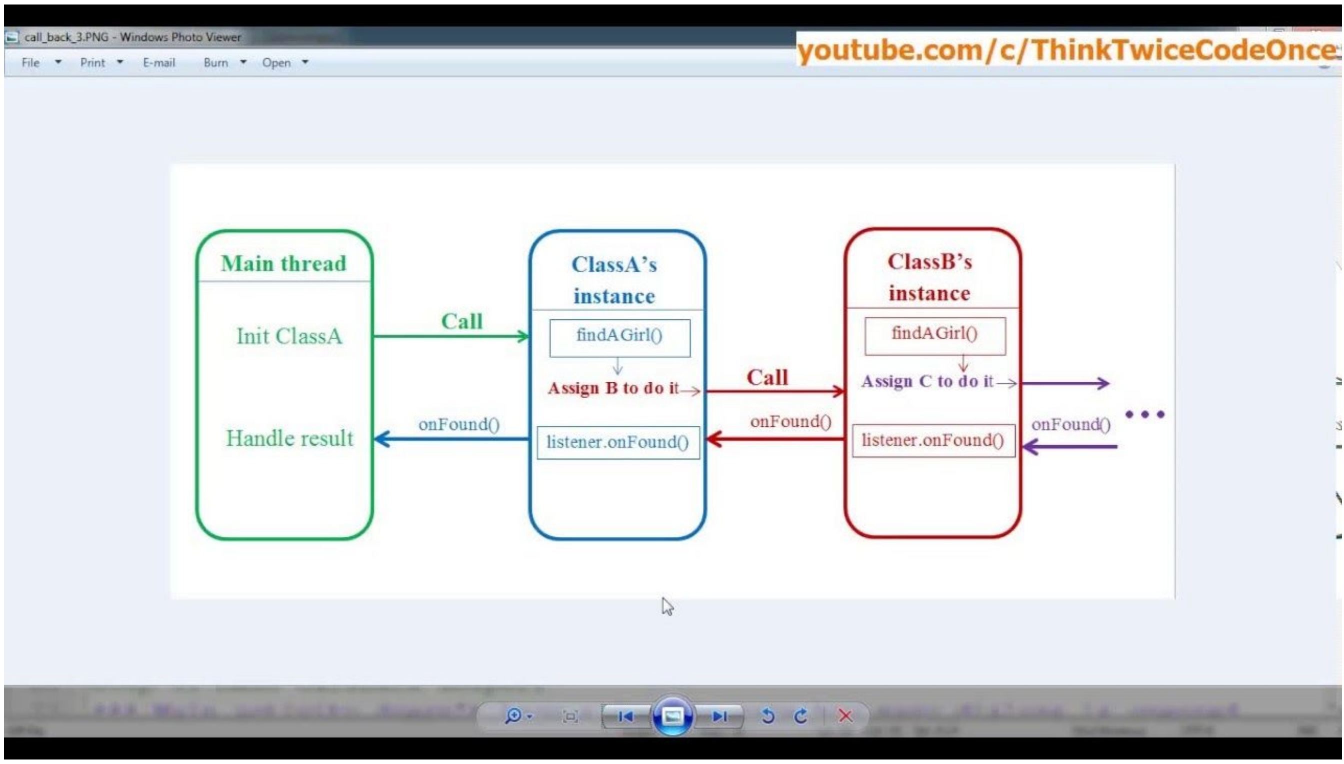
a **With Volley, you specify the entire endpoint** (destination) **dynamically (parameters and all) at the time of making the API call. By default, Volley returns** **JSONObject** **or a** **JSONArray** **depending on the type of request.**

RetroFit Wins

In the end, we decided to go with Retrofit for our application. Not only is it ridiculously fast, but it meshes quite well with our existing architecture. We were able to make a parent *Callback Interface* that automatically handles the error function, caching, and pagination with little to no effort for our APIs. In order to merge in Retrofit, we have to rename our variables to make our models GSON compliant, write a few simple interfaces.

Reference
http://instructure.github.io/blog/2013/12/09/volley-vs-retrofit/

- **What is Callback**



Interface world Java + Android, Level 1: Callback method !

*** Callback method:**
- "Don't call me, I'll call you".
- Main thread starts another thread to do something, when the worker thread finishes, it calls the result back to the main thread.

*** Java implement:**
Step 1. Create a class as a worker.
Step 2. Create a callback interface.
Step 3. Worker class do something and call result back.
Step 4. Make callback deeper.
- Main thread doesn't know + doesn't care how and what thread actually did the work, it just handles the result, that's all.

*** Android implement:**
Step 1. Create MainActivity + a button to show custom dialog with 3 options.
Step 2. Create a custom dialog.
Step 3. Create a callback interface.
Step 4. Integrate callback and handle result.
Step 5. Make callback deeper.

- Main activity doesn't know + doesn't care how many dialogs is opened to let user select, it just handle the final result.

Lecture https://www.youtube.com/watch?v=OTNuxe_ihtM
Github https://github.com/think-twice-code-once/interface_world

7. JSON PARSE TO SPINNER

8. ANDROID RETROFIT FETCH JSON AND DISPLAY IN GRIDVIEW

STEPs:
add retrofit dependency in android
retrofit request body and response body
perform get request using retrofit
load image in android using glide

Reference

Document
https://larntech.net/android-retrofit-fetch-json-and-display-in-gridview-gridview-with-image-and-text-using-retrofit/

Video lecture https://www.youtube.com/watch?v=Yps0G5M4ilA

Steps to Insert data to database using retrofit

- Create a new project.
- Add retrofit, Gson, HttpLoggingInterceptor and Glide library dependencies.
- Create ApiClient which will return retrofit.
- Create ApiInterface where we will define all of endpoints.
- Create a ImagesResponse class that will handle the request body.
- Add GridView in your activity
- Add layout to hold GridView items.
- Create Adapter class to pass data into GridView.
- Listen to click.

Send Data from the Form
Using Retrofit Library || Post Data Using Retrofit Http Library

http://www.androidcoding.in/2018/02/10/android-tutorial-retrofit-library/

Glide Library

https://github.com/codepath/android_guides/wiki/Displaying-images-with-the-Glide-library

10. ZIP CODE SEARCH

Glide

is a fast and efficient open source media management and image loading framework for Android that wraps media decoding, memory and disk caching, and resource pooling into a simple and easy to use interface.

Glide supports fetching, decoding, and displaying video stills, images, and animated GIFs. Glide includes a flexible API that allows developers to plug in to almost any network stack. By default Glide uses a custom `HttpUrlConnection` based stack, but also includes utility libraries plug in to Google's Volley project or Square's OkHttp library instead.

Glide's primary focus is on making scrolling any kind of a list of images as smooth and fast as possible, but Glide is also effective for almost any case where you need to fetch, resize, and display a remote image.

https://github.com/bumptech/glide

Model

In the model, we defined classes as response bodies and used inner classes with some methods to be used by the Android Activities to fetch the data from the response body.

These model classes need to implement Serializable since we needed this interface to convert from JSon to GSon and pass the response body between activities through Intents.

Potential improvement areas in the Pet Finder App:

We could improve the usability of our app by expanding the OrganizationResponse model class to get the Organization email data and use it in the Request activity to send the adoption request form to the corresponding animal shelter. We could ideally add more searching criteria such as breed, gender of the pet etc.