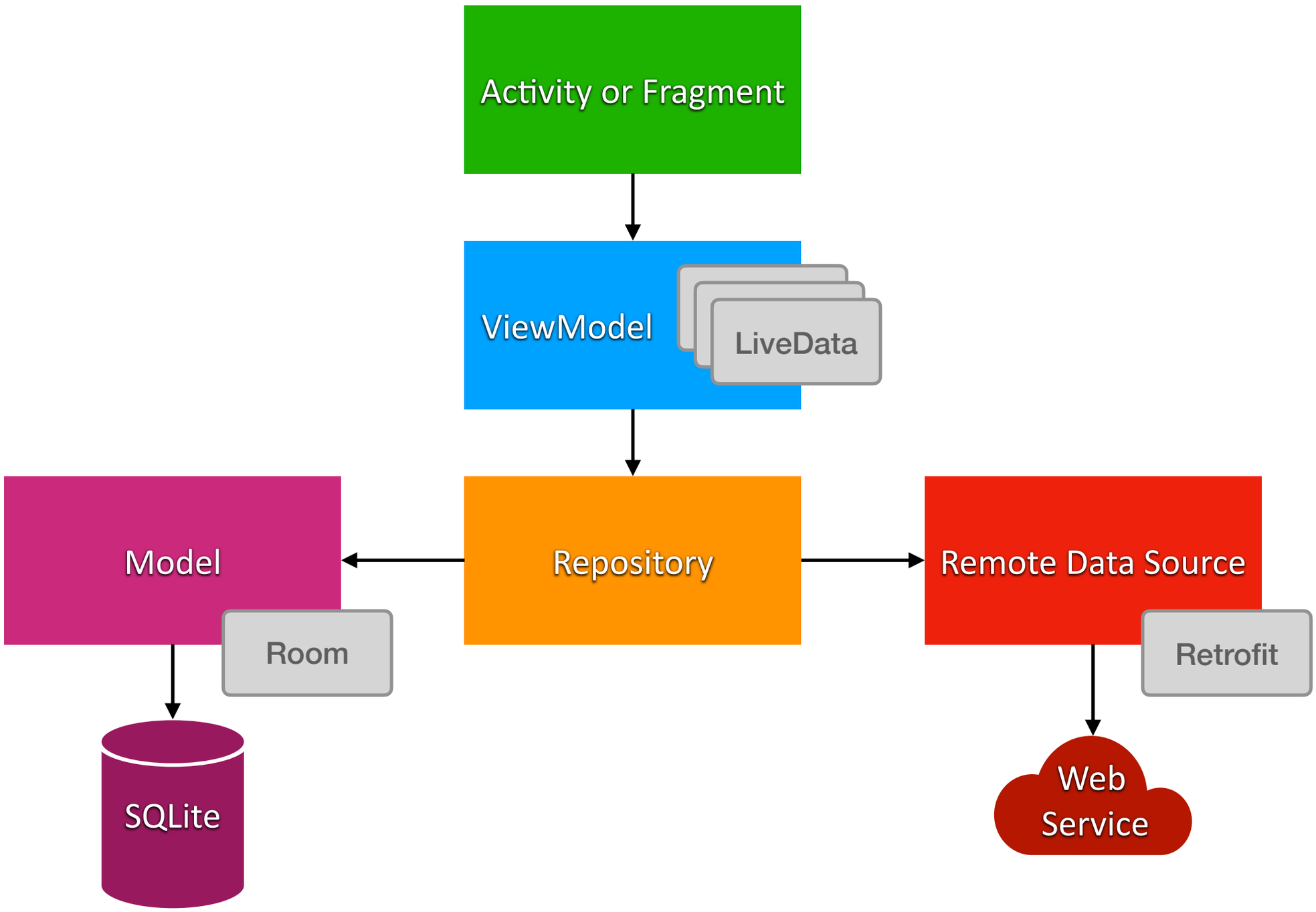
Retrofit App

based on the Retrofit Tutorial (Getting Started)

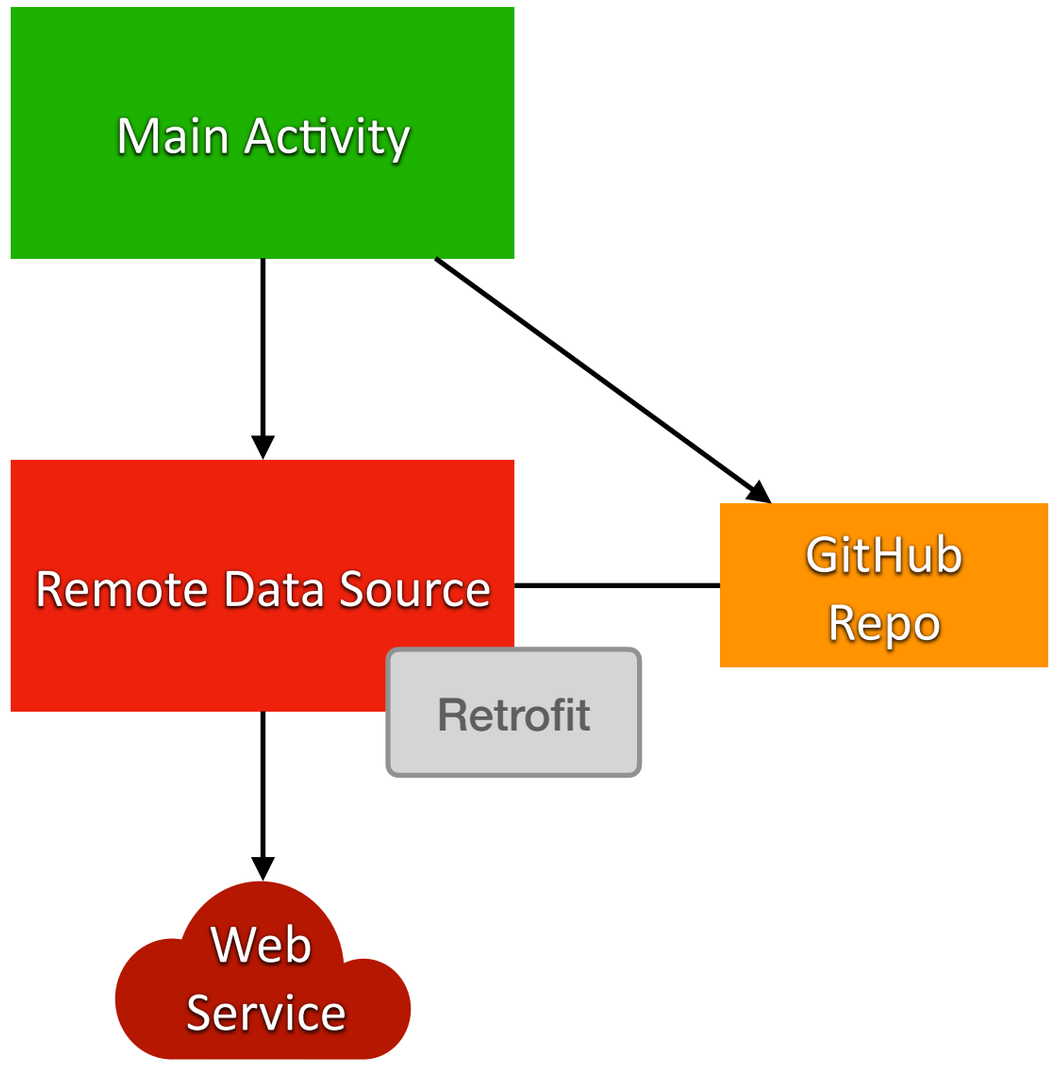
(https://www.youtube.com/watch?v=R4XU8yPzSx0&t=4s)

Repository Pattern



— https://developer.android.com/jetpack/guide

Our Retrofit App



— https://developer.android.com/jetpack/guide

REST API

6 Constraints

Web service APIs that satisfy these 6 constraints are said to be RESTful

- Client-Server
- Stateless
- Cacheable
- Uniform Interface
- Layered System
- Code on Demand

Separation of concerns between user interface and data storage

Requests from client to server contain all info needed for server to fulfill the request

In other words, server does *not* store session state!

— restfulapi.net

REST API

Resource Methods

Methods *loosely* correspond to database CRUD methods, but it depends on the API

In PhotoGallery, we only use GET (read)

- GET
- PUT
- POST
- DELETE
- (and a few more)

build.gradle

```
...

buildFeatures {
    viewBinding = true
}

...

dependencies {
    ...

    implementation 'com.squareup.retrofit2:retrofit:2.5.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.5.0'

    ...
}
```

We use view-bindings instead of findViewById

Add these to gradle to use retrofit and gson

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="edu.vt.cs.cs5254.retrofitapp">

    <uses-permission android:name="android.permission.INTERNET" />

    <application ...
    </application>

</manifest>
```

Add this in your manifest to access the internet

GitHubRepo

```
data class GitHubRepo(val name: String)
```

This is the model

This app shows a list of repository names

GitHubClient

```
import retrofit2.Call
import retrofit2.http.GET
import retrofit2.http.Path

interface GitHubClient {

    @GET("/users/{user}/repos")
    fun reposForUser(@Path("user") user: String) : Call<List<GitHubRepo>>
}
```

This is the interface for a request

Retrofit implements this for you based on the annotations

BNR would name this GitHubApi

GitHubClient

```
import retrofit2.Call
import retrofit2.http.GET
import retrofit2.http.Path

interface GitHubClient {

    @GET("/users/{user}/repos")
    fun reposForUser(@Path("user") user: String) : Call<List<GitHubRepo>>
}
```

This is not a full URL, it is *relative* to the the api

This is a GET request

"user" is a path parameter. It's passed-in value replaces {user} in the URL

Returning a Call ensures the request occurs on a separate thread (not the UI thread)

Returning List<GitHubRepo> works for some APIs, but it freezes the UI thread while we wait for a response

MainActivity

```
class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding
    private var repoAdapter: RepoAdapter? = RepoAdapter(emptyList())

    inner class RepoHolder...
    inner class RepoAdapter...

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        val view = binding.root
        setContentView(view)

        binding.repoRecyclerView.layoutManager = LinearLayoutManager(this)
        binding.repoRecyclerView.adapter = repoAdapter

        val retrofit: Retrofit = Retrofit.Builder()
            .baseUrl("https://api.github.com")
            .addConverterFactory(GsonConverterFactory.create())
            .build()

        val client = retrofit.create(GitHubClient::class.java)
        val call = client.reposForUser("square")

        call.enqueue...
    }
}
```

next slide

next slide + 1

Build a retrofit object using Retrofit.Builder

We must provide a base URL; the one in the client is relative

Gson converts JSON to Java/ Kotlin

Retrofit creates an implementation of the client

"square" becomes the user in the URL

MainActivity : inner classes for recycler-view

```
inner class RepoHolder(val itemBinding: ListItemRepoBinding) :
RecyclerView.ViewHolder(itemBinding.root) {
    private lateinit var repo: GitHubRepo

    fun bind(repo: GitHubRepo) {
        this.repo = repo
        itemBinding.repoName.setText(this.repo.name)
    }
}

inner class RepoAdapter(var repos: List<GitHubRepo>) : RecyclerView.Adapter<RepoHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int) : RepoHolder {
        val itemBinding = ListItemRepoBinding
            .inflate(LayoutInflater.from(parent.context), parent, false)
        return RepoHolder(itemBinding)
    }

    override fun getItemCount() = repos.size

    override fun onBindViewHolder(holder: RepoHolder, position: Int) {
        val repo = repos[position]
        holder.bind(repo)
    }
}
```

The holder pairs a repo-name text view from the item-list-repo layout (view) with the name of a git-hub repo object (model)

The adapter holds a list of git-hub-repo objects

A call can be executed synchronously with execute or asynchronously with enqueue

For a call executed synchronously, the developer must ensure it occurs on a separate thread; Android does not network activity on the UI thread

A Callback object is created as an anonymous class

MainActivity : onCreate : call.enqueue

```
call.enqueue(object : Callback<List<GitHubRepo>> {
    override fun onFailure(call: Call<List<GitHubRepo>>, t: Throwable) {
        Log.d(TAG, "Failed Request")
        Toast.makeText(this@MainActivity, "error", Toast.LENGTH_SHORT).show()
    }

    override fun onResponse(
        call: Call<List<GitHubRepo>>,
        response: Response<List<GitHubRepo>>
    ) {
        val repos = response.body() ?: emptyList()
        Log.d(TAG, "Successful request and response with ${repos.size} repos")
        binding.repoRecyclerView.adapter = RepoAdapter(repos)
    }

} )
```

If the request fails, onFailure is invoked

If it succeeds, invoke onResponse

The "Elvis" operator is used to ensure repos is never null

The list of repository names is passed to a new adapter and assigned to the recycler view

retrofit2.Call

enqueue

```
void enqueue (Callback<T> callback)
```

Asynchronously send the request and notify callback of its response or if an error occurred talking to the server, creating the request, or processing the response.

Parameters	
`callback`	**Callback**: The callback to be notified of a response or error.

retrofit2.Callback

Interface Callback<T>

Type Parameters:
`T – Successful response body type.`

`public interface` **`Callback<T>`**

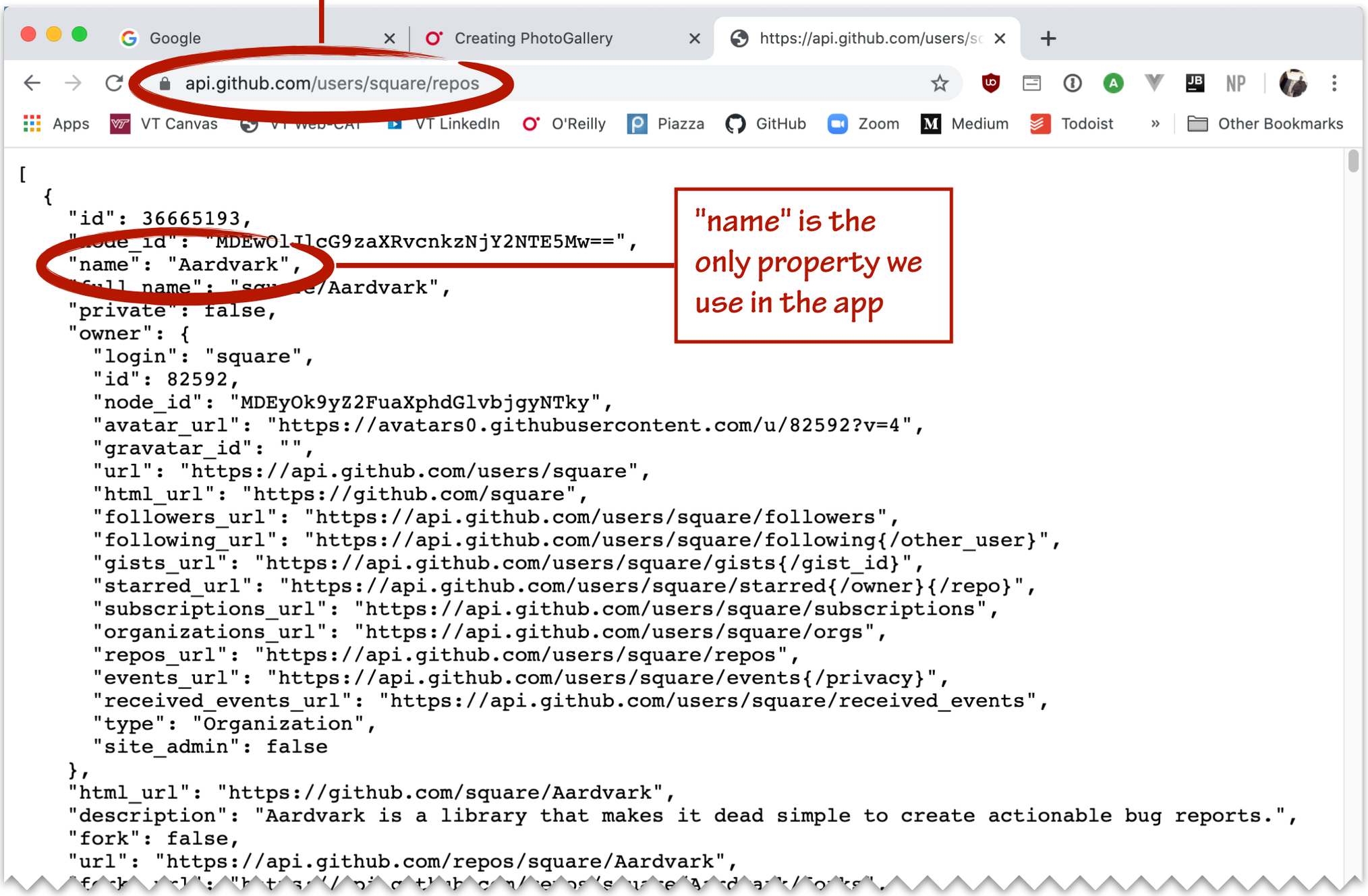
Communicates responses from a server or offline requests. One and only one method will be invoked in response to a given request.

Callback methods are executed using the Retrofit callback executor. When none is specified, the following defaults are used:

- Android: Callbacks are executed on the application's main (UI) thread.
- JVM: Callbacks are executed on the background thread which performed the request.

Modifier and Type	Method and Description
`void`	**`onFailure`**`(`**`Call`**`<T> call, `**`Throwable`**` t)` Invoked when a network exception occurred talking to the server or when an unexpected exception occurred creating the request or processing the response.
`void`	**`onResponse`**`(`**`Call`**`<T> call, `**`Response`**`<T> response)` Invoked for a received HTTP response.

JSON directly from address bar



"name" is the only property we use in the app

When we run the app, we get a list of all GitHub repo names for user "square", the developers of Retrofit