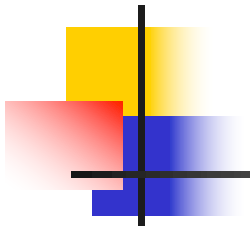
RESTful Web Services and Android



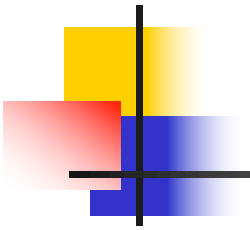
RESTFul Services: URIs

- We will modify our JobsTracker Android app to use a RESTful Web service, instead of an SQLite database.
- We will not spend much time on the actual service implementation, and focus on using a remote Web Service, instead.

RESTFul Services: URIs

Simple URI design for JobTracker app:

	GET	**PUT**	**POST**	**DELETE**
/joblead	Y	N	Y	N
/joblead/{id}	Y	Y	N	Y



RESTFul Services: URIs

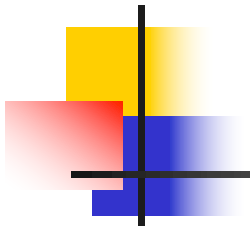
URI design:

- GET /joblead -- get all job leads
- POST /joblead -- create a new job lead
- GET /joblead/{id} -- get a job lead w/ {id}
- PUT /joblead/{id} -- update a job lead w/ {id}
- DELETE /joblead/{id} -- delete a job lead w/ {id}

JSON is used to represent job lead resources.

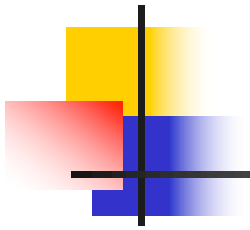
All job leads are returned as a JSON array.

XML representation could be easily added later.



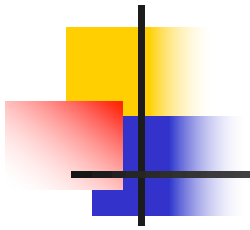
JSON

- JavaScript Object Notation (JSON) has gained considerable popularity
- It is a language-independent data interchange format (`www.json.org`)
- It originated with JavaScript, recently it has been used in numerous programming languages



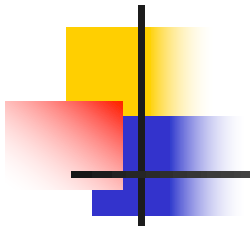
JSON

- JSON is frequently preferred to XML for data exchanges between communicating systems, for several reasons:
  - less verbose
  - easier parsing
  - better fit to programming language data structures, especially OO languages
  - offers arrays (XML does not)



JSON

- JSON offers two basic structures:
  - An object, which is an unordered set of name/value pairs
  - An array, which is an ordered collection of values



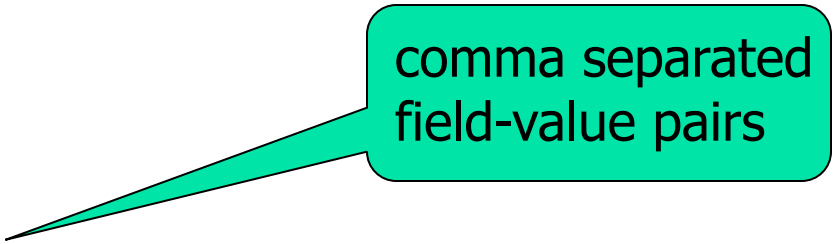
JSON

- A JSON object corresponds to similar data types in many programming languages, such as structures, objects, hash tables, etc.
- A JSON array corresponds to arrays, lists, vectors, etc., which are available in many programming languages

JSON: Object Example

- Example: a job lead represented in JSON

```
{
   ”companyName” : ”UPS”,
   ”phone” : ”(404) 334-1281”,
   ”url” : ”www.ups.com”,
   ”comments” : ”Good talk with Chuck (product manager)”
}
```



POJO and JSON Objects

```
public class JobLead {
    private String companyName;
    private String phone;
    private String url;
    private String comments;
    ...
}
```

:JobLead
companyName = ”UPS” phone = ”(404) 334-1281” url = ”www.ups.com” comments = ”Good talk with...”

JSON:

```
{
    ”companyName” : ”UPS”,
    ”phone” : ”(404) 334-1281”,
    ”url” : ”www.ups.com”,
    ”comments” : ”Good talk with Chuck (product manager)”
}
```

POJO and XML Representation

```
public class JobLead {
    private String companyName;
    private String phone;
    private String url;
    private String comments;
    ...
}
```

:JobLead
companyName = ”UPS” phone = ”(404) 334-1281” url = ”www.ups.com” comments = ”Good talk with...”

XML:

```
<jobLead>
    <companyName>UPS</companyName>
    <phone>(404) 334-1281</phone>
    <url>www.ups.com<url>
    <comments>Good talk with Chuck (product manager)</comments>
</jobLead>
```

JSON Arrays

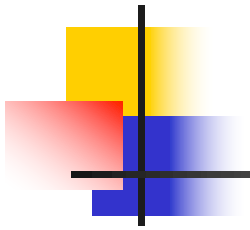
- Arrays in JSON:

  [”Home Depot, Inc.”, ”Google”, ”Amazon”, ”IBM”]

- Arrays can hold objects, as well:

```
[
  {  ”companyName” : ”UPS”,
     ”phone” : ”(404) 334-1281”,
     ”url” : ”www.ups.com”,
     ”comments” : ”Good exchange Chuck (product manager)”  }  ,

  {  ”companyName” : ”Google”,
     ”phone” : ”(800) 443-5578”,
     ”url” : ”www.google.com/careers”,
     ”comments” : ”I think I will get a second interview”  },
  …
]
```



JSON: Value Types

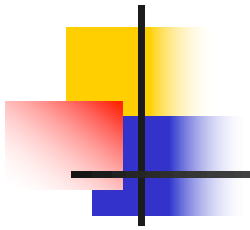
- JSON values can be:
  - string
  - number
  - object
  - array
  - “true” and “false”
  - “null”

JSON: Another Example

```
{
    "firstName" : "Mary",
    "lastName" : "Smith",
    "studentID" : "811123456",
    "major" : "Computer Science",
    "address" : { "street" : "123 Wide St.",
                  "city" : "Athens",
                  "state" : "Georgia",
                  "zip" : "30602" },
    "phones" : [ {"home" : "706-123-4567" },
                 {"cell" : "706-123-4567" },
                 {"work" : "706-123-4567" } ]
}
```

Nested object

Value which is an array of objects



RESTFul Services: Hands-on

- REST services (and SOAP) are intended to be invoked programmatically, e.g., from a Java program or from an Android app.
- JAX-RS libraries provide convenient way of making requests and receiving responses.
- JobsTracker app using a RESTful service is available on eLC (JobsTrackerRest)
- Java class (using JAX-RS) providing the service is available as an illustration, as well (JobLeadResource.java)

RESTFul Services: Hands-on

- Testing of REST services can be done from a Web browser with a suitable plugin.
- Examples include:
  - RESTer for Google Chrome and Firefox
  - Postman Interceptor for Google Chrome

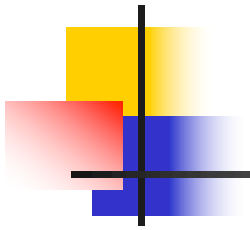
Need: Content-Type: application/json

{ "companyName":"NCR","phone":"404-667-9876","url":"www.ncr.com",
"comments":”Great company to work for in Atlanta, GA" }

Android and RESTFul Services

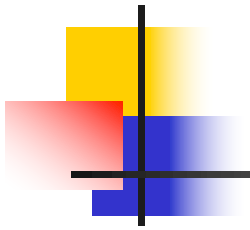
- It is not a good practice to directly interact with a relational database server from an app because:
  - database users typically must have user/password (credentials) to access the DB
  - an app would have to store the database server's credentials within the app
  - it would be feasible to *decompile* the app, recover the DB credentials, and perform malicious/illegal DB access/modifications
  - other reasons exist, too

Android and RESTFul Services



- Consequently, apps should interact with remote data sources via Web services.
- An app requires each user to register with the app and therefore be able to access the service
- Hence, only the service (backend) must know the DB credentials; access to the DB is controlled *inside* of the service.

Android and RESTFul Services



- It is possible to code your own service interactions in Android, using the regular Java/Kotlin networking API
- However, it is easier to use one of the available libraries for REST interactions:
  - Retrofit (http://square.github.io/retrofit/)
  - Volley (http://developer.android.com/training/volley/)
  - RESTDroid (http://github.com/PCreations/RESTDroid) (older library)
- We will use Retrofit in our examples

Android and RESTFul Services

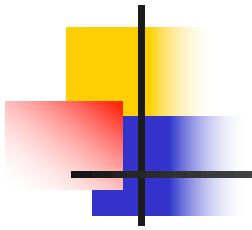
- Retrofit is a popular library to develop Android Apps that interact with RESTful services
- Setting up the Android Studio project:
  - Add to `AndroidManifest` file to allow network use:

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

  - Add these lines to `build.gradle (module: app)` to download the necessary libraries:

    ```
    implementation 'com.squareup.retrofit2:retrofit:2.9.0'
    implementation 'com.google.code.gson:gson:2.10.1'
    implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
    ```

Android and RESTFul Services



- Define the base URL of the service:

  String BASE_URL = "http://uml.cs.uga.edu:8080/jobtracker/rest/";

- The URIs designed for the service are concatenated to the BASE_URL. For example:

  http://uml.cs.uga.edu:8080/jobtracker/rest/joblead/112

  is the URI for a specific job lead (112)

Android and RESTFul Services

- Define the interface to access the service

```
private interface JobLeadsService {

    // Request method and URL specified in the annotation

    @GET("joblead")
    Call<List<JobLead>> retrieveAllJobLeads();

    @POST("joblead")
    Call<Void> storeJobLead( @Body JobLead jobLead );

    @PUT("joblead/{id}")
    Call<Void> updateJobLead( @Path("id") long itemId, @Body JobLead jobLead );

    @DELETE("joblead/{id}")
    Call<Void> deleteJobLead( @Path("id") long itemId );
}
```

Android and RESTFul Services

- Make the call to the service, as in the example:

```
List<JobLead> jobLeads = null;

// create the call object
Call<List<JobLead>> jobLeadsCall = service.retrieveAllJobLeads();
try {
    // execute the call and get the results
    jobLeads = jobLeadsCall.execute().body();
}
catch( IOException e ) {
    Log.e( DEBUG_TAG, e.toString() );
    return null;
}
```

Android and RESTFul Services

- Retrofit calls can be made asynchronously by overriding:

```
public void onResponse( Call<User> call, Response<User> response )
public void onFailure( Call<User> call, Throwable throwable )
```

- However, in our example app, we are using the `AsyncTask` class, unchanged from the previous version of the app, that was based on SQLite.

- The app is available on eLC in the Sample Apps folder; it uses a Web service deployed on uml.cs.uga.edu (which is behind the UGA firewall – use VPN when outside UGA net).

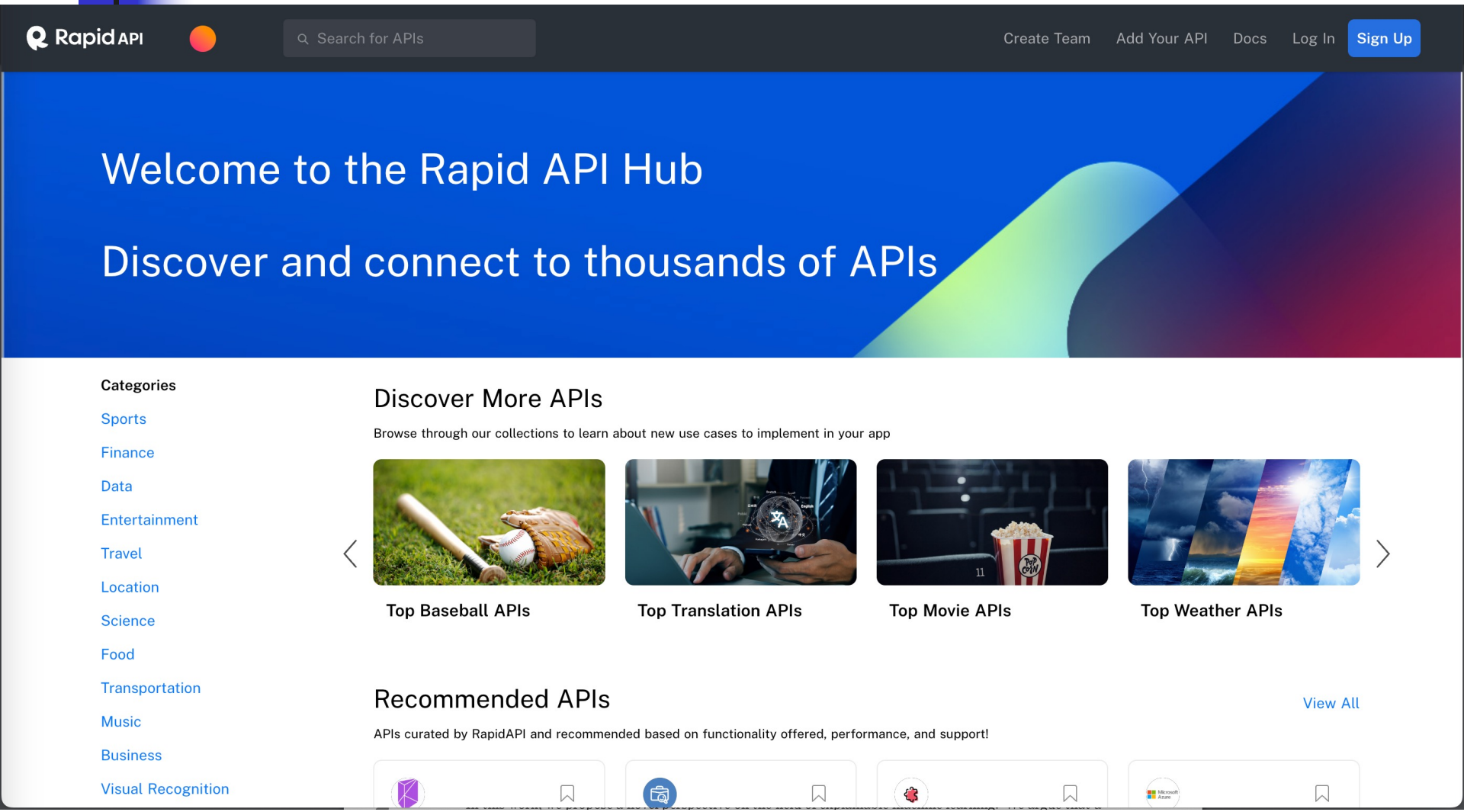
Using Available Web Services

- Literally, *thousands* of Web Services are available for use! How to find them – remember UDDI for SOAP Web Services?
- Many directories providing info about Web Services are available. One of the best is:

  http://rapidapi.com

  which currently lists well over 40,000 services; they are often referred to as Web APIs.
- Most of them are pay-only services, but some can be used free of charge, or when keeping a low usage (number of requests)

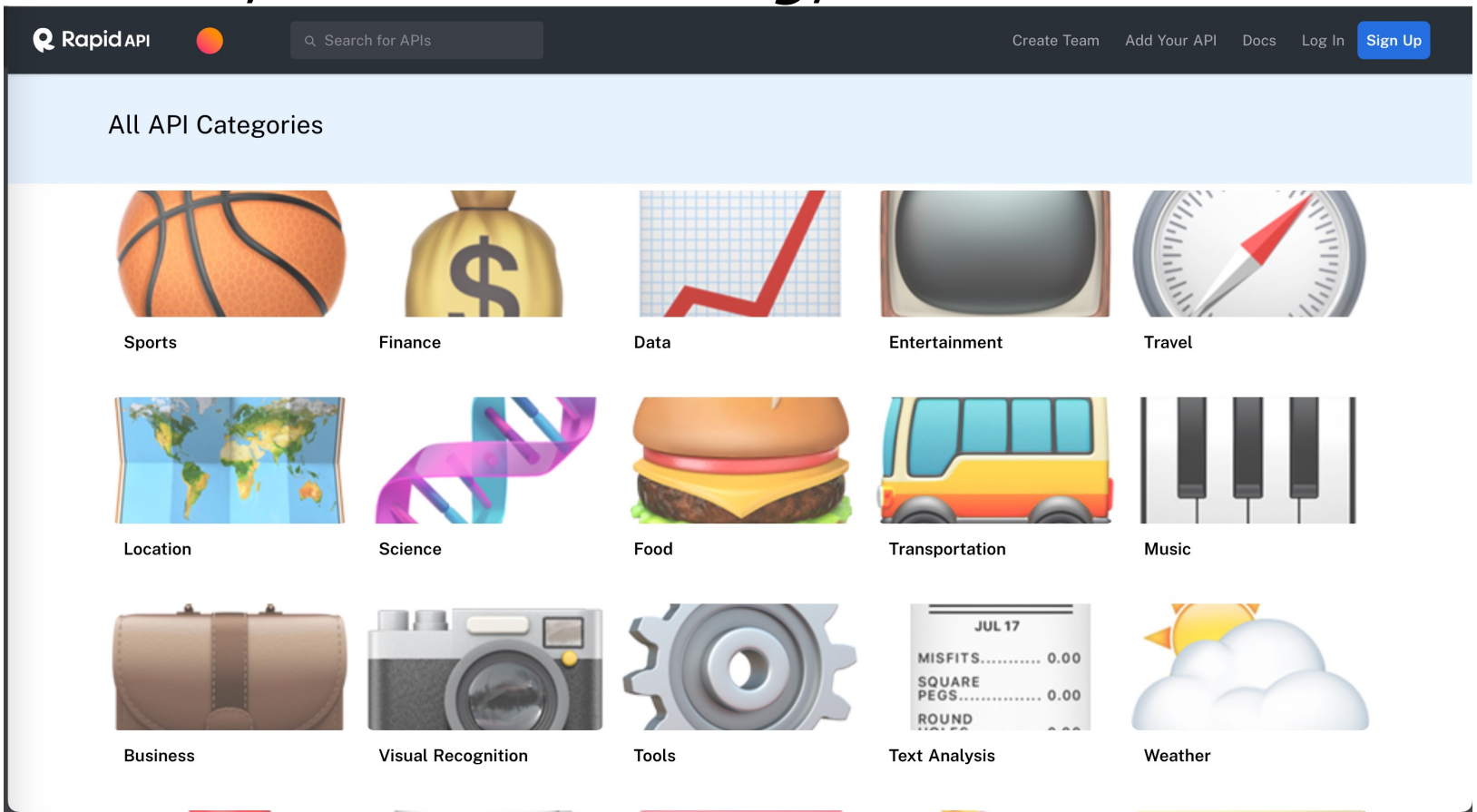
RapidAPI Hub



Hub at: rapidapi.com/hub Documentation at: https://docs.rapidapi.com/

Using Available Web Services

- Available Web Services provide a wide variety functions, ranging from *Advertising*, through *Finance*, *Machine Learning*, to *Weather*

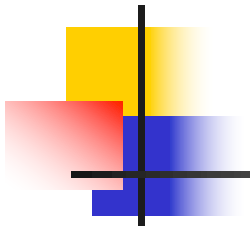


Rapidapi's categories

Using Available Web Services

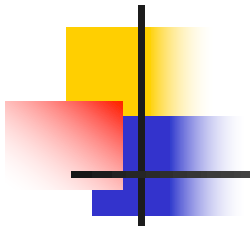
- Several Web services can be integrated within a single value-added Web Service; this is called **Web Service composition**
- For example, services for airline reservations, hotel booking, car rental, and even weather forecasting could be integrated to provide a single, comprehensive Traveler Web Service
- Even a single Web Service can be incorporated in a software system to provide a useful service or data

Using Available Web Services



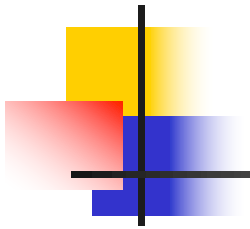
- RapidAPI can be helpful in integrating a Web API into your own application
- You can also create your own Web API, publish it, and hope to generate a revenue stream
- Of course, should it be a popular service, you would have to focus on high availability and many other issues

Using Available Web Services



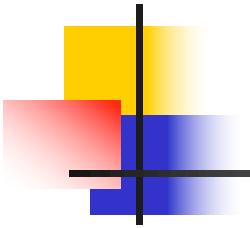
Some examples of Web APIs:

- Weather: https://openweathermap.org/api
- Hotels: https://connect.booking.com/
- Air travel: https://apiportal.delta.com/
- Translation: https://cloud.google.com/translate/docs/reference/rest
- GitHub: https://docs.github.com/en/rest



NewsFinder API

- We will use one of such Web Services to build a simple Android App to find interesting news
- The service we will utilize is News API v2, available at https://newsapi.org
- It is quite easy to build a simple app to find news items of interest and display a few of them for the user

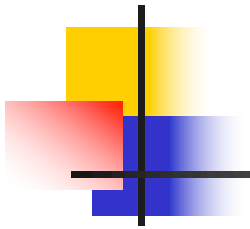


NewsFinder App

- As we already know, most Web Services are available to paying customers only
- To use them in an app, the app developer must provide a payment information or obtain a special contract/license for their use
- Often, the service provider issues the developer a special key (or some other authentication method) to enable processing of the client's requests

NewsFinder App

- Even free of charge services usually require a developer to register for service and obtain an API key
- It is the case with the News API (free for non-commercial use), so we must register and obtain such an API key
- News API, as other Web Service providers, offers documentation/user guides detailing the available requests (GET, POST, etc.), and message exchange formats, usually in XML or JSON



NewsFinder App

- In order to use a service, we need to:
  - decide which operations (methods + URIs) are needed for your app
  - create Java domain objects (POJOs) to send/receive data to the service
  - find out about the response codes for all operations
  - create a Retrofit interface to send operation requests to the service asynchronously
  - utilize the received data in your app

NewsFinder App

- The News API service endpoint is at:

  `https://newsapi.org`

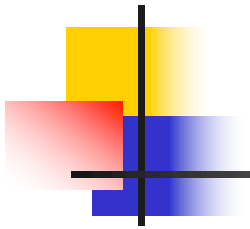
- In our NewsFinder app, we will be using only the GET method on the URI:

  `/v2/everything`

  which returns all news sources related to the search query sent to the service

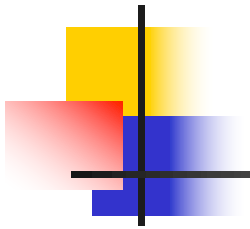
- The search query will be sent as the query parameter, along with the API key:

  `/v2/everything?q=covid-19&apiKey=`*`API_KEY`*



NewsFinder App

- The POJO classes are relatively simple
- Analyze the documentation describing the data formats required for communication with the service and create the needed Java classes
- For each separate JSON object, create a Java class; you can use any reasonable name
- You must have a default constructor and a set of setters and getters for all instance variables



NewsFinder App

- The Java class must have instance variable names that are **identical** to the field names in the JSON object
- Use suitable primitive datatypes (int, double, etc.) or String for literals included in the JSON objects; use Java lists for JSON arrays
- For each instance variable variableName, you must create the setter and getter of the form:
  - getVariableName()
  - setVariableName()

NewsFinder App

For example:

```
{
    "status": "ok",
    "totalResults": 3539,
    "articles": [
        {
            "source": {
            "id": "techcrunch",
            "name": "TechCrunch"
        },
        "author": "Romain Dillet",

        ...

}
```

```
public class SearchResult {
    private String status;
    private int totalResults;
    private List<Article> articles;
    …
    public String getStatus() …
    public void setStatus( String …
    public int getTotalResults() …
    public void setTotalResults( int …
}

public class Article {
    private Source source;
    private String author;
    …
}
```

NewsFinder App

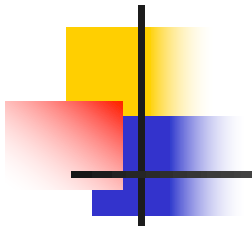
- The URI for the GET request must look like this:

  https://newsapi.org/v2/everything?q=query&apiKey=key

- So, the Retrofit interface for our NewsFinder is:

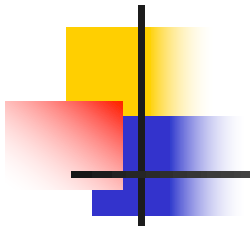
```
private interface NewsService {

    @GET("everything")
    Call<SearchResult> searchNews(
                            @Query("q") String query,
                            @Query("apiKey") String apiKey );
}
```



NewsFinder App

- The rest of the application is quite straightforward
- We use a RecyclerAdapter and CardView to display the news search results
- The News API supplies images, as well, which we are utilizing in our application



NewsFinder App

- The complete application is available on eLC
- **However, before you can use it, you must register for the News API service, obtain your own API key, and use it in the downloaded app**
- Copy your API key into the file

  `assets/api_key`

  within the News Finder project