

## LAB 5

### MỤC TIÊU

Kết thúc bài thực hành sinh viên có khả năng:

- ✓ Sử dụng Retrofit GET/POST API

### NỘI DUNG

#### BÀI 1: LẤY DANH SÁCH DISTRIBUTOR HIỆN THỊ LÊN RECYCLE VIEW

**Yêu cầu:** Xây giao diện android như hình



## Bước 1: Viết API (mở file `api.js`)

```
router.get('/get-list-distributor', async (req, res) => {
  try {
    // Lấy danh sách theo thứ tự distributors mới nhất
    const data = await Distributors.find().sort({createdAt: -1});
    if(data)
    {
      // Trả về danh sách
      res.json({
        "status" : 200,
        "messenger" : "thành công",
        "data" : data
      })
    }
    else
    {
      // Nếu thêm không thành công result null, thông báo không thành công
      res.json({
        "status" : 400 ,
        "messenger" : "Lỗi, không thành công",
        "data" : []
      })
    }
  } catch (error) {
    console.log(error);
  }
});
```

**Bước 2:** Tại project Android tạo một package **model**. Tại package model tạo object đặt tên là **Response**

```
Response.java x
1  package com.example.myapplication.model;
2
3  public class Response<T> {
4      private int status;
5      private String messenger;
6      //T là kiểu Generic
7      private T data;
8
9      public Response() {
10     }
11
12     public Response(int status, String messenger, T data) {
13         this.status = status;
14         this.messenger = messenger;
15         this.data = data;
16     }
17
18     public int getStatus() {
19         return status;
20     }
21
22     public void setStatus(int status) {
23         this.status = status;
24     }
25
26     public String getMessenger() {
27         return messenger;
28     }
29
30     public void setMessenger(String messenger) {
31         this.messenger = messenger;
32     }
33
34     public T getData() {
35         return data;
36     }
37
38     public void setData(T data) {
39         this.data = data;
40     }
41 }
```

### Bước 3 : Tạo object Distributor

```
package com.example.myapplication.model;

import com.google.gson.annotations.SerializedName;

public class Distributor {

    //Có thể dùng Annotations của gson để đổi tên cho các trường nhận vào
    //Ví dụ trường _id nhận từ api, thay vì đặt tên trường trong object là _id
    //Có thể đặt là id và thêm vào Annotations @SerializedName("_id")
    @SerializedName("_id")
    private String id;
    private String name, createdAt, updatedAt;

    public Distributor() {
    }

    public Distributor(String id, String name, String createdAt, String updatedAt) {
        this.id = id;
        this.name = name;
        this.createdAt = createdAt;
        this.updatedAt = updatedAt;
    }

    public String get_id() {
        return id;
    }

    public void set_id(String id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getCreatedAt() {
        return createdAt;
    }
}
```

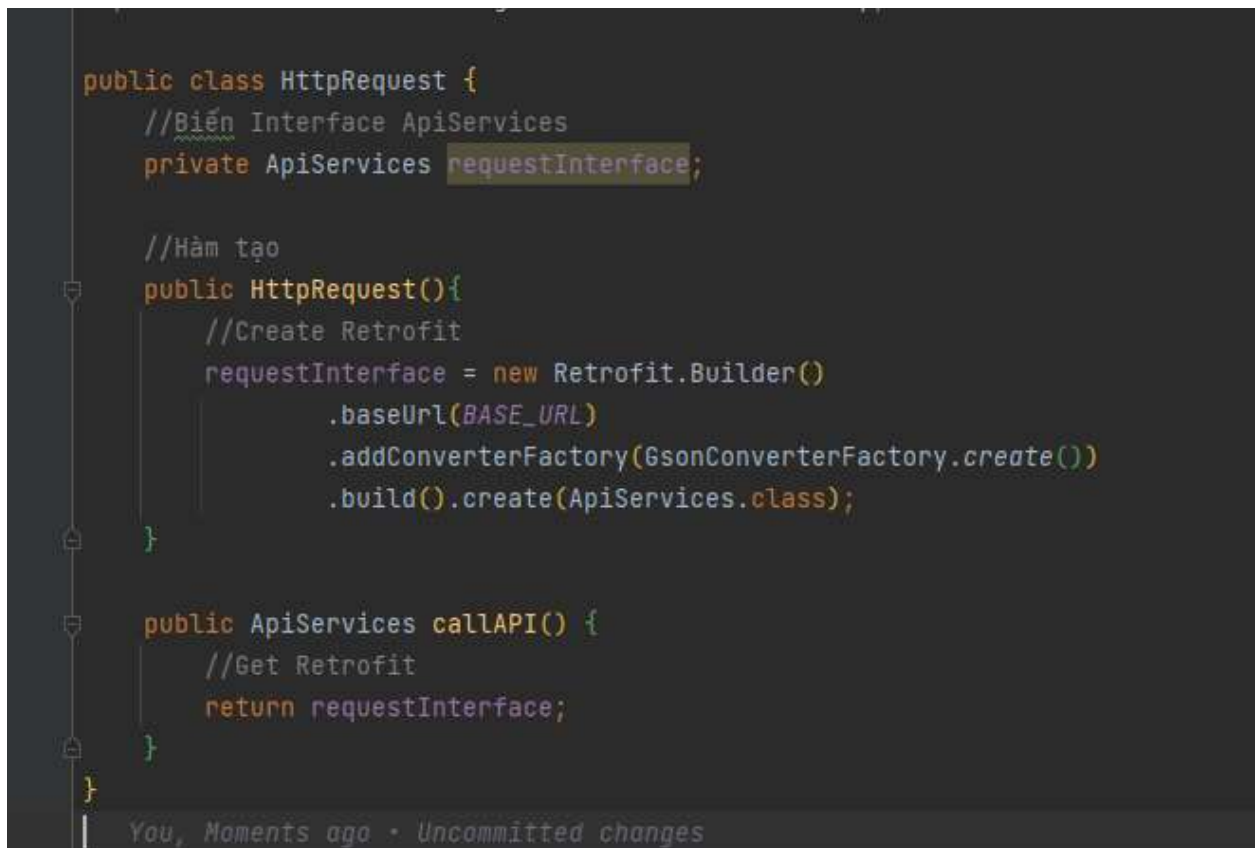
**Bước 4: Tạo package `services`. Tại package `services` tạo 1 interface `ApiServices`**



```

18 public interface ApiService {
19     //Và đây là address của api localhost thay thành ip 88.8.8.8
20     //URL với máy tính là http://88.8.8.8
21     //Base URL là url của api
22     public static String BASE_URL = "http://88.8.8.8:8080/api/";
23
24     //Annotation gọi các method GET và url phân ra api
25     //Base URL + gọi("get-list-distributor") + http://88.8.8.8:8080/api/get-list-distributor
26     @GET("get-list-distributor")
27     Call<Response<ArrayList<Distributor>>> getListDistributor();
28     //Gọi các api của api
29 }
    
```

**Bước 5: Tại package `services` tạo 1 object `HttpRequest`**



```

public class HttpRequest {
    //Biến Interface ApiService
    private ApiService requestInterface;

    //Hàm tạo
    public HttpRequest(){
        //Create Retrofit
        requestInterface = new Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .build().create(ApiServices.class);
    }

    public ApiService callAPI() {
        //Get Retrofit
        return requestInterface;
    }
}
    
```

You, Moments ago • Uncommitted changes

## Bước 6: Call API

```

public class MainActivity extends AppCompatActivity {
    private HttpRequest httpRequest;
    private RecyclerView recycle_distributors;
    private RecyclerView.Adapter recycle_item_distributors_adapter;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        recycle_distributors = findViewById(R.id.recycle_distributors);
        //Khai tạo services Request
        httpRequest = new HttpRequest();
        //Thay thì call API
        httpRequest.callAPI()
            .getListDistributor() //Phương thức API cần thực thi
            .enqueue(getDistributorAPI()); // Xử lý bất đồng bộ
    }

    private void getData(ArrayList<Distributor> ds) {
        adapter = new RecyclerView.Adapter<ViewHolder>() {
            @Override
            public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {
                //Lấy data
                ArrayList<Distributor> ds = response.body().getData();
                //Set dữ liệu lên recycle
                recycle_item_distributors_adapter.notifyDataSetChanged();
                recycle_item_distributors_adapter.notifyDataSetChanged();
            }
        };
        recycle_distributors.setAdapter(adapter);
    }

    Callback<Response<ArrayList<Distributor>>> getDistributorAPI = new Callback<Response<ArrayList<Distributor>>>() {
        @Override
        public void onResponse(Call<Response<ArrayList<Distributor>>> call, retrofit2.Response<Response<ArrayList<Distributor>>> response) {
            //Khi call thành công sẽ chạy vào hàm này
            if(response.isSuccessful()) {
                //check status code
                if(response.body().getStatus() == 200) {
                    //Lấy data
                    ArrayList<Distributor> ds = response.body().getData();
                    //Set dữ liệu lên recycle
                    recycle_item_distributors_adapter.notifyDataSetChanged();
                    //Toast ra thông tin từ Messenger
                    Toast.makeText(MainActivity.this, response.body().getMessenger(), Toast.LENGTH_SHORT).show();
                }
            }
        }
    };

    @Override
    public void onFailure(Call<Response<ArrayList<Distributor>>> call, Throwable t) {
        //Khi call thất bại sẽ chạy vào hàm này
        Log.d("Log", "====> onFailure: " + t.getMessage());
    }
}

```

## BÀI 2: TÌM KIẾM DISTRIBUTOR THEO TÊN

**Yêu cầu:** Xây giao diện android như hình. Khi nhấn biểu tượng tìm kiếm trên bàn phím sẽ bắt đầu call api tìm kiếm. Thêm thuộc tính imeOptions vào ô **EditText**

```
android:imeOptions="actionSearch"
```



## Bước 1: Viết API (app.js)

```
router.get('/search-distributor', async (req, res) => {
  try {
    const key = req.query.key; // Nhận từ query
    // Lấy danh sách theo thứ tự distributors mới nhất
    const data = await Distributors.find({name: { "$regex": key, "$options": "i" }})
      .sort({createdAt: -1});
    if(data)
    {
      // Trả về danh sách
      res.json({
        "status" : 200,
        "messenger" : "thành công",
        "data" : data
      })
    }
  } else
  {
    // Nếu thêm không thành công result null, thông báo không thành công
    res.json({
      "status" : 400 ,
      "messenger" : "Lỗi, thành công",
      "data" : []
    })
  }
} catch (error) {
  console.log(error);
}
});
```



## Bước 2: Thêm phương thức call API interface **ApiServices**

```
//Call gọi API trả về của api>  
  
@GET("search-distributor")  
Call<Response<ArrayList<Distributor>>> searchDistributor(@Query("key") String key);  
  
}
```

## Bước 3: Cài API

```
//Bạn cần phải cài đặt thư viện Retrofit và OkHttp  
edtTimkiem.setOnEditorActionListener(new TextView.OnEditorActionListener() {  
    @Override  
    public boolean onEditorAction(TextView textView, int i, KeyEvent keyEvent) {  
        if(i == EditorInfo.IME_ACTION_SEARCH)  
        {  
            //Lấy từ khóa từ ô tìm kiếm  
            String key = edtTimkiem.getText().toString();  
  
            httpRequest.callAPI()  
                .searchDistributor(key) //Phương thức API cần thực thi  
                .enqueue(getDistributorAPI); // Xử lý bất đồng bộ  
            //Vì giá trị trả về vẫn là một list Distributor  
            //nên có thể sử dụng lại Callback của getListDistributor() You, 2 minutes d  
            return true;  
        }  
        return false;  
    }  
});
```

Ta được kết quả:



### BÀI 3: ADD DISTRIBUTOR SAU ĐÓ RELOAD LẠI API DANH SÁCH

**Yêu cầu :** Xây giao diện có 1 edittext và một nút thêm

**\*Giao diện tham khảo**



## Bước 1: Thêm phương thức call API interface **ApiServices**

```
@POST("add-distributor")
Call<Response<Distributor>> addDistributor(@Body Distributor distributor);
```

## Bước 2: Call API

```
btn_add.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String name = edt_diglog_name.getText().toString();
        Distributor distributor = new Distributor();
        distributor.setName(name);
        httpRequest.callAPI()
            .addDistributor(distributor) //Phương thức API cần thực thi
            .enqueue(responseDistributorAPI); // Xử lý bất đồng bộ
        alertDialog.dismiss();
    }
});
```

## Tạo Callback **responseDistributorAPI**

```
Callback<Response<Distributor>> responseDistributorAPI = new Callback<Response<Distributor>>() {
    @Override
    public void onResponse(Call<Response<Distributor>> call, retrofit2.Response<Response<Distributor>> response) {
        if(response.isSuccessful())
        {
            //check status code
            if(response.body().getStatus() == 200)
            {
                //Call lại api danh sách
                httpRequest.callAPI()
                    .getListDistributor() //Phương thức API cần thực thi
                    .enqueue(getDistributorAPI);
                Toast.makeText(context, MainActivity.this, response.body().getMessenger(), Toast.LENGTH_SHORT).show();
            }
        }
    }

    @Override
    public void onFailure(Call<Response<Distributor>> call, Throwable t) {
        Log.d("tag: >>> getListDistributor", "tag: onFailure: " + t.getMessage());
    }
};
```

#### BÀI 4: XÓA DISTRIBUTOR

##### Bước 1: Viết API (app.js)

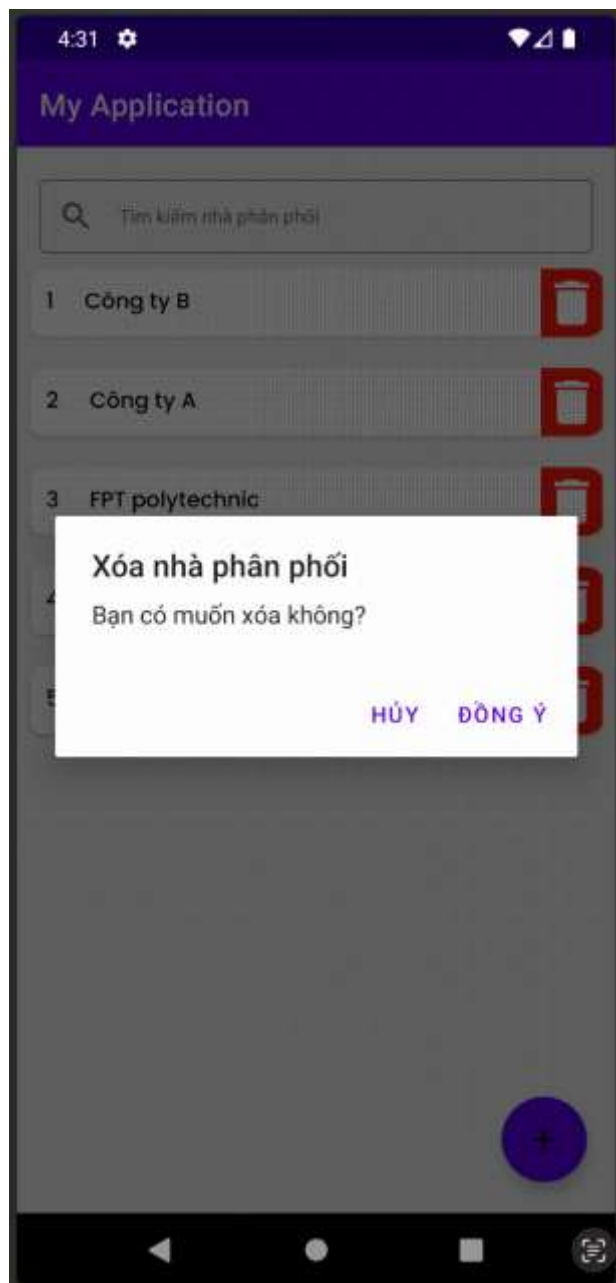
```
router.delete('/delete-distributor-by-id/:id', async (req, res) => {
  try {
    const {id} = req.params
    const result = await Distributors.findByIdAndDelete(id);
    if(result)
    {
      //Nếu xóa thành công sẽ trả về thông item đã xóa
      res.json({
        "status" : 200,
        "messenger" : "Xóa thành công",
        "data" : result
      })
    }
    else
    {
      res.json({
        "status" : 400 ,
        "messenger" : "Lỗi, Xóa không thành công",
        "data" : []
      })
    }
  } catch (error) {
    console.log(error);
  }
})
```

##### Bước 2: Thêm phương thức call API interface **ApiServices**

```
//Param url sẽ bỏ vào {}
@DELETE("delete-distributor-by-id/{id}")
Call<Response<Distributor>> deleteDistributorById(@Part("id") String id);
```

### Bước 3: Call API

```
public void Delete(String id) {  
    httpRequest.callAPI()  
        .deleteDistributorById(id) //Phương thức API cần thực thi  
        .enqueue(responseDistributorAPI);  
    //Vi delete và add trả về cùng kiểu dữ liệu nên có thể sử dụng lại Callback responseDistributorAPI  
}
```



## BÀI 5: UPDATE DISTRIBUTOR

### Bước 1: Viết API (app.js)

```
router.put('/update-distributor-by-id/:id', async (req, res) => {
  try {
    const {id} = req.params
    const data = req.body; // Lấy dữ liệu từ body
    const result = await Distributors.findByIdAndUpdate(id, {name : data.name})
    if(result)
    {
      // Nếu thêm thành công result !null trả về dữ liệu
      res.json({
        "status" : 200,
        "messenger" : "Thêm thành công",
        "data" : result
      })
    }
    else
    {
      // Nếu thêm không thành công result null, thông báo không thành công
      res.json({
        "status" : 400 ,
        "messenger" : "Lỗi, thêm không thành công",
        "data" : null
      })
    }
  } catch (error) {
    console.log(error);
  }
});
```

### Bước 2: Thêm phương thức call API interface ApiService

```
@PUT("update-distributor-by-id/{id}")
Call<Response<Distributor>> updateDistributorById(@Path("id") String id, @Body Distributor distributor);
}
```

### Bước 3: Call API

```
distributor.setName(name); //set lại dữ liệu cần update  
httpRequest.callAPI()  
    .updateDistributorById(id, distributor) //Phương thức API cần thực thi  
    .enqueue(responseDistributorAPI); // Xử lý bất đồng bộ
```

#### \*\*\* YÊU CẦU NỘP BÀI:

Sv nén file bao gồm các yêu cầu đã thực hiện trên, nộp lms đúng thời gian quy định của giảng viên. Không nộp bài coi như không có điểm.

--- Hết ---