LAB 7

MỤC TIÊU

Kết thúc bài thực hành sinh viên có khả năng:

- ✓ Biết sử dụng QueryMap.
- ✓ Load more trong ứng dụng

NỘI DUNG

BÀI 1: THỰC HIỆN CHỨC NĂNG LOAD MORE

Bước 1: Code API (**api.js**)

```
router.get('/get-page-fruit',async (req,res) => {
    //Auten
    const authHeader = req.headers['authorization']
    const token = authHeader && authHeader.split(' ')[1]
    if (token == null) return res.sendStatus(401)
    let payload;
    JWT.verify(token, SECRETKEY , (err,_payload) => {
        if(err instanceof JWT.TokenExpiredError) return res.sendStatus(401)
        if (err) return res.sendStatus(403)
        payload = _payload;
      })
    let perPage = 2; // số lượng sản phẩm xuất hiện trên 1 page
    let page = req.query.page || 1; //Page truyền lên
    let skip = (perPage * page) - perPage; // Phân trang
    let count = await Fruits.find().count();// Lấy tổng số phần tử
    try {
        const data = await Fruits.find()
                                 .populate('id_distributor')
                                 .skip(skip)
                                 .limit(perPage)
        res.json({      You, 1 second ago • Uncommitted changes
            "status" : 200,
            "messenger" : "Danh sách fruit",
            "data" : {
                "data" : data,
                "currentPage" : Number(page),
                "totalPage" : Math.ceil(count/perPage)
            }
        })
    } catch (error) {
        console.log(error);
    }
})
```

Bước 2: Tạo model **Page**

Page.java | ApiServices.java | HomeActivity.java

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

public class Page <T> {
    private T data;
    private int currentPage,totalPage;

    public Page() {
    }

    public Page(T data, int currentPage, int totalPage) {
        this.data = data;
        this.currentPage = currentPage;
        this.totalPage = totalPage;
    }

    public T getData() {
        return data;
    }

    public void setData(T data) {
        this.data = data;
    }

    public int getCurrentPage() {
        return currentPage;
    }

    public void setCurrentPage(int currentPage) {
        this.currentPage = currentPage;
    }

    public int getTotalPage() {
        return totalPage;
    }

    public void setTotalPage(int totalPage) {
        this.totalPage = totalPage;
    }
}
```

Bước 3: Thêm phương thức call API interface **ApiServices**

```java
@GET("get-page-fruit")
Call<Response<Page<ArrayList<Fruit>>>> getPageFruit(@Header("Authorization") String token,@Query("page") int page);
```

Bước 4: Thêm **ProgressBar** vào layout, bọc ngoài là một **main layout** là một **NestedScrollView**

```
<androidx.core.widget.NestedScrollView
    android:id="@+id/nestScrollView"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textInputLayout">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/recycle_fruits"
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="1"
            android:nestedScrollingEnabled="false" />

        <ProgressBar
            android:id="@+id/loadmore"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:indeterminate="true"
            android:indeterminateTint="@color/purple_500"
            android:indeterminateTintMode="src_atop"
            android:visibility="gone" />
    </LinearLayout>
</androidx.core.widget.NestedScrollView>
```

Bước 5: Thêm các biến toàn cục

```
private ProgressBar loadmore;
private ArrayList<Fruit> ds = new ArrayList<>();
private int page = 1;
private int totalPage = 0;
private NestedScrollView nestScrollView;
```

Bước 6: Thay đổi đoạn code call **API**

```
@Override
protected void onResume() {
    super.onResume();
    httpRequest.callAPI().getPageFruit("Bearer " + token,page).enqueue(getListFruitResponse);
}

Callback<Response<Page<ArrayList<Fruit>>>> getListFruitResponse = new Callback<Response<Page<ArrayList<Fruit>>>>() {
    @Override
    public void onResponse(Call<Response<Page<ArrayList<Fruit>>>> call, retrofit2.Response<Response<Page<ArrayList<Fruit>>>> response) {
        if(response.isSuccessful())
        {
            //check status code
            if(response.body().getStatus() == 200)
            {
                //Get total Page
                totalPage = response.body().getData().getTotalPage();
                //Lấy data
                ArrayList<Fruit> _ds = response.body().getData().getData();
                //Set dữ liệu lên recycle
                getData(_ds);
                //Toast ra thông tin từ Messenger
            }
        }
    }

    @Override
    public void onFailure(Call<Response<Page<ArrayList<Fruit>>>> call, Throwable t) {
        Log.d( tag: "=== getListFruit", msg: "onFailure: " + t.getMessage());
    }
};
```

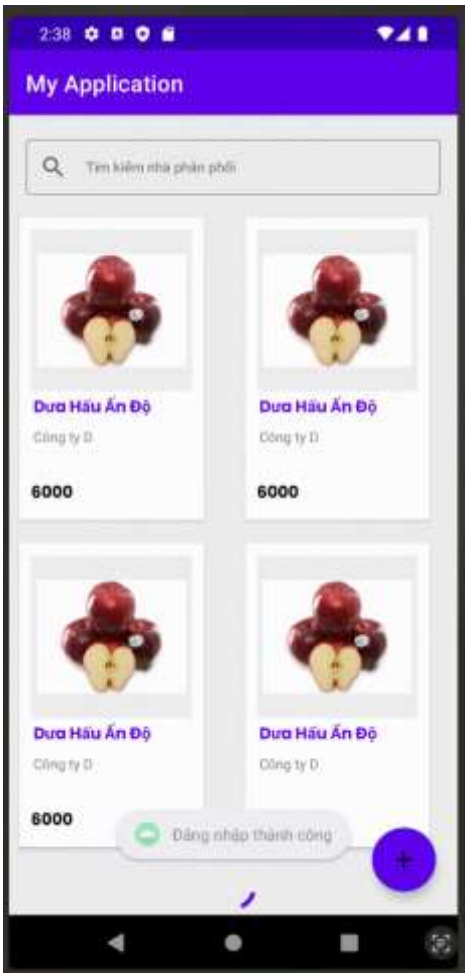
Bước 7: Thay đổi code trong hàm **GetData()**

```
private void getData(ArrayList<Fruit> _ds)
{
    //Kiểm tra nếu process load more chạy thì chỉ cần add thêm fruits vào list
    if(loadmore.getVisibility() == View.VISIBLE)
    {
        //Do chạy ở local nên tốt độ mạng tốt
        //nên sẽ thêm 1 đoạn code deplay ( deplay 1s)
        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                //
                adapter.notifyItemInserted( position: ds.size() - 1);
                loadmore.setVisibility(View.GONE);
                ds.addAll(_ds);
                //Thông báo adapter dữ liệu thay đổi
                adapter.notifyDataSetChanged();
            }
        }, delayMillis: 1000);
        return;
    }
    ds.addAll(_ds);
    adapter = new Recycle_Item_Fruits(ds, context: this);
    recycle_fruits.setLayoutManager(new GridLayoutManager( context: this, spanCount: 2));
    recycle_fruits.setAdapter(adapter);
}
```

Bước 8: Bắt sự kiện khi lướt đến item cuối cùng sẽ Call API và load thêm sản phẩm

Thông qua **NestedScrollView**

```
nestScrollView.setOnScrollChangeListener(new NestedScrollView.OnScrollChangeListener() {
    @Override
    public void onScrollChange(@NonNull NestedScrollView v, int scrollX, int scrollY, int oldScrollX, int oldScrollY) {
        if (scrollY == v.getChildAt( index: 0).getMeasuredHeight() - v.getMeasuredHeight()) {
            if(totalPage == page) return;
            if (loadmore.getVisibility() == View.GONE) {
                loadmore.setVisibility(View.VISIBLE);
                page++; // Tăng page
                //Call API
                httpRequest.callAPI().getPageFruit( token: "Bearer " + token,page).enqueue(getListFruitResponse);
            }
        }
    }
});
```



BÀI 2: FILTER FRUIT QUA TÊN, GIÁ LỚN HOẶC GIÁ NHẬP VÀO, SẮP XẾP THEO GIÁ TĂNG HOẶC GIẢM DẦN

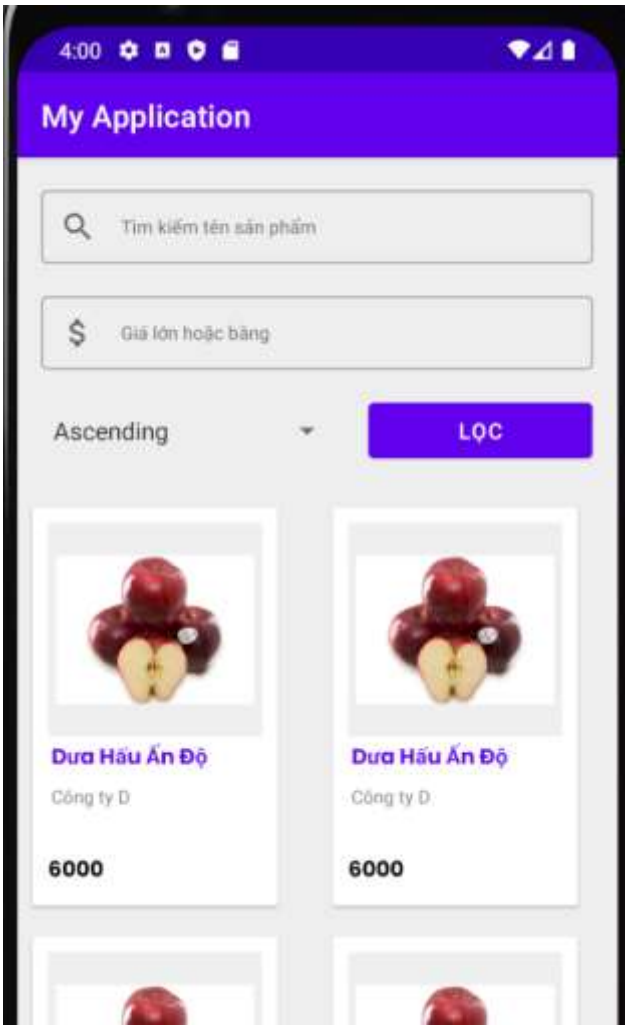
Bước 1: Sửa code API

```
router.get('/get-page-fruit',async (req,res) => {
    //Auten
    const authHeader = req.headers['authorization']
    const token = authHeader && authHeader.split(' ')[1]
    if (token == null) return res.sendStatus(401)
    let payload;
    JWT.verify(token, SECRETKEY , (err,_payload) => {
        if(err instanceof JWT.TokenExpiredError) return res.sendStatus(401)
        if (err) return res.sendStatus(403)
        payload = _payload;
      })
    let perPage = 6; // số lượng sản phẩm xuất hiện trên 1 page
    let page = req.query.page || 1; //Page truyền lên
    let skip = (perPage * page) - perPage; // Phân trang
    let count = await Fruits.find().count();// Lấy tổng số phần tử
    //filtering
    //Lọc theo tên
    const name = { "$regex": req.query.name ?? "", "$options": "i" }
    //Lọc giá lớn hơn hoặc bằng giá truyền vào
    const price = {$gte: req.query.price ?? 0}
    //Lọc sắp xếp theo giá
    const sort = {price: req.query.sort ?? 1}
    try {
        const data = await Fruits.find({name: name, price: price})
                                 .populate('id_distributor')      You, 4 minutes ag
                                 .sort(sort)
                                 .skip(skip)
                                 .limit(perPage)
        res.json({
            "status" : 200,
            "messenger" : "Danh sách fruit",
            "data" : {
                "data" : data,
                "currentPage" : Number(page),
                "totalPage" : Math.ceil(count/perPage)
            }
        })
    } catch (error) {
        console.log(error);
    }
})
```

Bước 2: Thay phương thức call API interface **ApiServices**

```
@GET("get-page-fruit")
Call<Response<Page<ArrayList<Fruit>>>> getPageFruit(@Header("Authorization") String token,
                                                    @QueryMap Map<String,String> stringMap);
```

Bước 3: Sửa giao diện như hình



Bước 4: Setup spinner lọc giá

***Tại file string.xml**

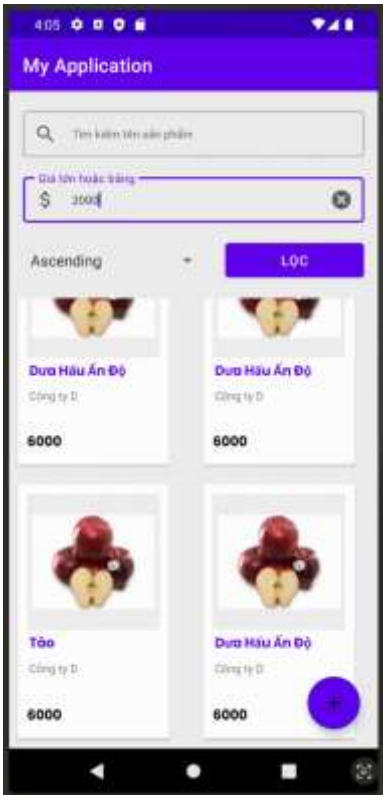
```xml
<resources>
    <string name="app_name">My Application</string>
    <string-array name="spinner_price">
        <item>Ascending</item>
        <item>Decrease</item>
    </string-array>
</resources>   Kyna2k, 8/18/2023 8:38 PM • Initial commit
```

```java
ArrayAdapter<CharSequence> spinnerAdapter = ArrayAdapter.createFromResource( context: this,
        R.array.spinner_price, android.R.layout.simple_spinner_item);
spinner.setAdapter(spinnerAdapter);
spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
        CharSequence value = (CharSequence) adapterView.getAdapter().getItem(i);
        if(value.toString().equals("Ascending"))
        {
            //Biến sort toàn cục
            //Tăng dần
            sort = "1";
        }else if(value.toString().equals("Decrease"))
        {
            //Giảm dần
            sort = "-1";
        }
    }

    @Override
    public void onNothingSelected(AdapterView<?> adapterView) {

    }
});
spinner.setSelection(1);
```

Bước 5: Call API

```
@Override
protected void onResume() {
    super.onResume();
    //filter mặc định
    Map<String,String> map = getMapFilter(page, _name: "", _price: "0", _sort: "-1");
    httpRequest.callAPI() ApiServices
            .getPageFruit( token: "Bearer " + token,map) Call<Response<Page<ArrayList<Fruit>>>>
            .enqueue(getListFruitRespone);
}
//Hàm Call API theo filter   You, Moments ago • Uncommitted changes
private void FilterFruit()
{
    String _name = edttimkiem.getText().toString().equals("") ? "" : edttimkiem.getText().toString();
    String _price = edtgia.getText().toString().equals("") ? "0" :  edtgia.getText().toString();
    String _sort = sort.equals("") ? "-1" : sort;
    Map<String,String> map = getMapFilter(page,_name,_price,_sort);
    httpRequest.callAPI().getPageFruit( token: "Bearer " + token,map).enqueue(getListFruitRespone);
}
//Hàm setup MapQuery
private Map<String,String> getMapFilter(int _page,String _name,String _price,String _sort)
{
    Map<String,String> map = new HashMap<>();
    map.put( k: "page",String.valueOf(_page));
    map.put( k: "name", _name);
    map.put( k: "price", _price);
    map.put( k: "sort", _sort);
    return map;
}
```

```
nestScrollView.setOnScrollChangeListener(new NestedScrollView.OnScrollChangeListener() {
    @Override
    public void onScrollChange(@NonNull NestedScrollView v, int scrollX, int scrollY, int oldScrollX, int oldScrollY) {
        if (scrollY == v.getChildAt( index: 0).getMeasuredHeight() - v.getMeasuredHeight()) {
            if(totalPage == page) return;
            if (loadmore.getVisibility() == View.GONE) {
                loadmore.setVisibility(View.VISIBLE);
                page++; // Tăng page
                //Call API
                //load more theo filter
                FilterFruit();
            }
        }
    }
});
btn_loc.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //đưa page về 1
        page = 1;
        //Nếu lọc sẽ clean list
        ds.clear();
        adapter.notifyDataSetChanged();
        FilterFruit();
    }
});
```



BÀI 3: XỬ LÝ INTERCEPT AUTHORIZATION

Bước 1: Tại file **HttpRequest.** Tạo thêm hàm một constructor

```
public HttpRequest(String token){
    //Tạo intercept
    OkHttpClient.Builder httpClient = new OkHttpClient.Builder();
    httpClient.addInterceptor(new Interceptor() {
        @NonNull
        @Override
        public Response intercept(Chain chain) throws IOException {
            Request request = chain.request().newBuilder().addHeader( name: "Authorization",  value: "Bearer " + token).build();
            return chain.proceed(request);
        }
    });
    //Create Retrofit
    requestInterface = new Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .client(httpClient.build())
            .build().create(ApiServices.class);   You, Moments ago • Uncommitted changes
}
```

Bước 2: Bỏ Header tại các interface method

```
    @GET("get-list-fruit")
    Call<Response<ArrayList<Fruit>>> getListFruit();
    //@Header("Authorization") là token ta aut ta cần truyền lên để có thể lấy dữ liệu

    @GET("get-page-fruit")
    Call<Response<Page<ArrayList<Fruit>>>> getPageFruit(@QueryMap Map<String,String> stringMap);
}
```

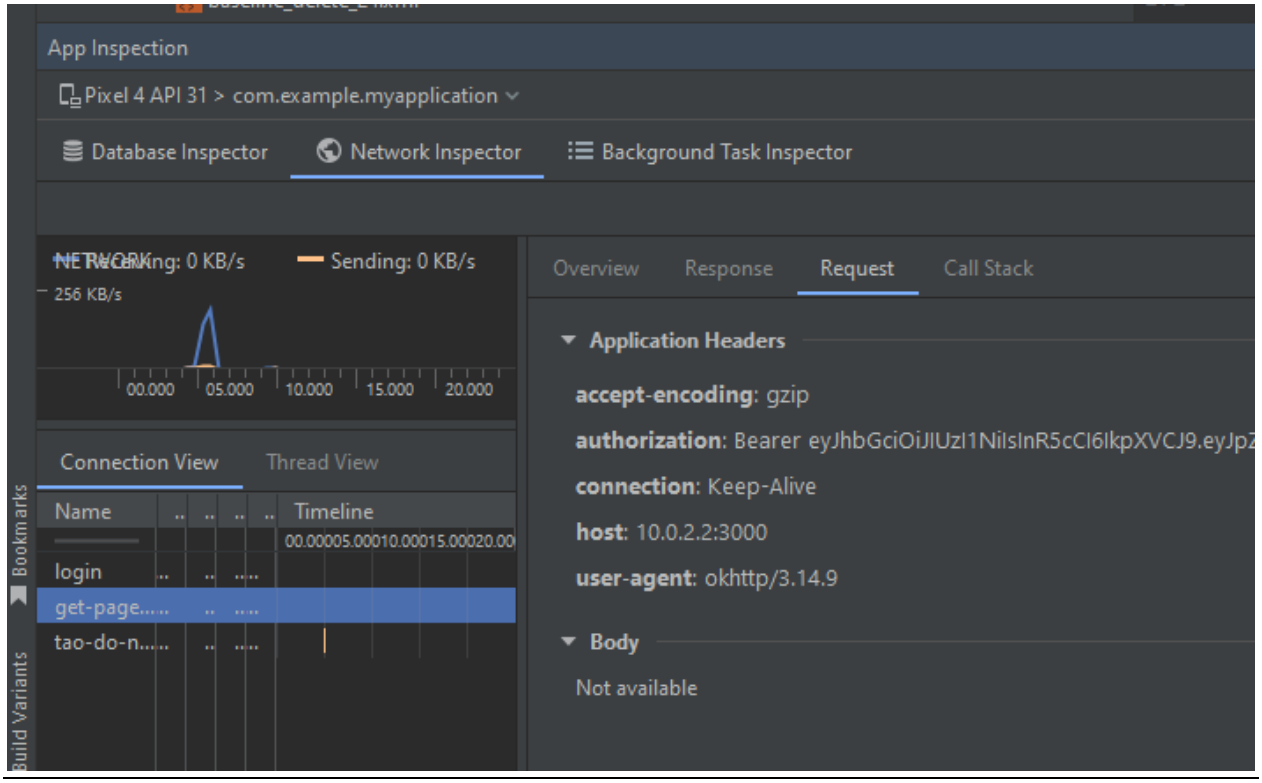
Bước 3: Thay đổi hàm khởi tạo của **HttpRequest** (nếu API đó cần **Authorization** sẽ thay hàm tạo cần truyền tham số token vào, còn không cần **Authorization** thì không cần truyền)

```
        btn_loc = findViewById(R.id.btn_loc);
        sharedPreferences = getSharedPreferences( name: "INFO",MODE_PRIVATE);
        //Lấy token từ sharedPreferences
        token = sharedPreferences.getString( s: "token", s1: "");
        httpRequest = new HttpRequest(token);
```

***Code thực thi gọi API sau khi đổi hàm tạo**

```
@Override
protected void onResume() {
    super.onResume();
    //filter mặc định
    Map<String,String> map = getMapFilter(page, _name: "", _price: "0", _sort: "-1");
    httpRequest.callAPI() ApiServices
            .getPageFruit(map) Call<Response<Page<ArrayList<Fruit>>>>
            .enqueue(getListFruitRespone);
}
//Hàm Call API theo filter
private void FilterFruit()
{
    String _name = edttimkiem.getText().toString().equals("") ? "" : edttimkiem.getText().toString();
    String _price = edtgia.getText().toString().equals("") ? "0" :  edtgia.getText().toString();
    String _sort = sort.equals("") ? "-1" : sort;
    Map<String,String> map = getMapFilter(page,_name,_price,_sort);
    httpRequest.callAPI().getPageFruit(map).enqueue(getListFruitRespone);
}
```

API nhận được **Authorization**



BÀI 4: GET CHI TIẾT FRUIT

***Giao diện tùy ý**

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

```
@GET("get-fruit-by-id/{id}")
Call<Response<Fruit>> getFruitById(@Path("id") String id);
```

Bước 2: Call **API**

```
httpRequest.callAPI().getFruitById(id).enqueue(new Callback<Response<Fruit>>() {
    @Override
    public void onResponse(Call<Response<Fruit>> call, retrofit2.Response<Response<Fruit>> response) {

    }

    @Override
    public void onFailure(Call<Response<Fruit>> call, Throwable t) {

    }
});
```

***** 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 ---