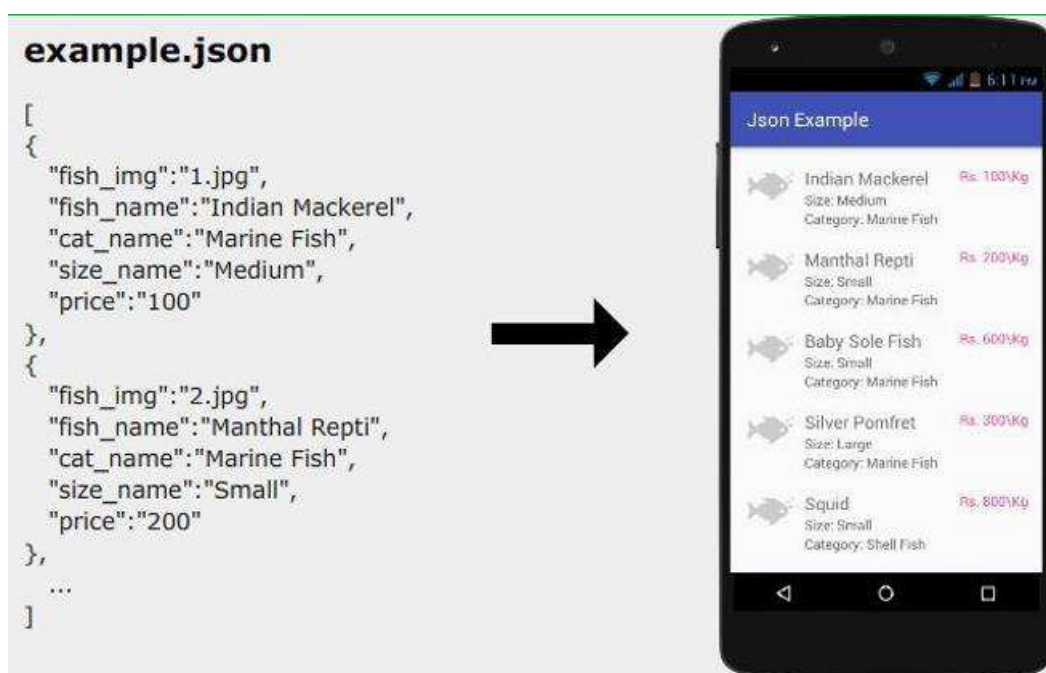


## HƯỚNG DẪN TẠO DỰ ÁN VỚI RESTful API TRONG ANDROID

Thao tác thêm, cập nhật và xóa

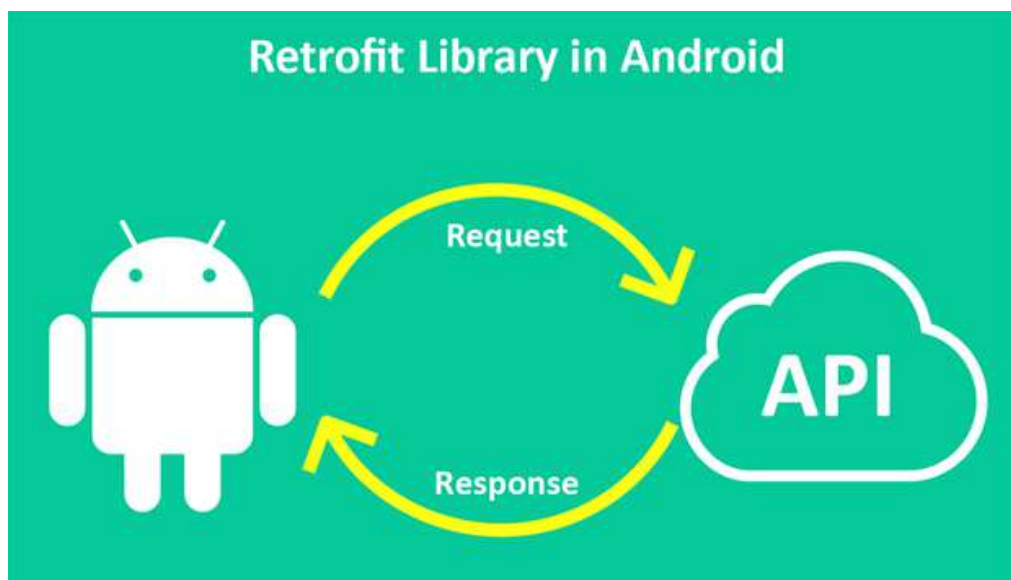


## Mục lục

1. Giới thiệu.....	3
2. Các bước thực hiện tương tác Restful API trong Android .....	3
3. Ví dụ mẫu.....	3
4. Tự làm .....	17

## 1. Giới thiệu

- Trong bài lab này chúng ta sẽ tiếp tục sử dụng Retrofit để tương tác với API. Cụ thể từ Android chúng ta sẽ thêm dữ liệu, cập nhật dữ liệu và xóa dữ liệu



## 2. Các bước thực hiện tương tác Restful API trong Android

2.1 Thêm dependencies

2.2 Tạo danh sách Model tương ứng với số lượng Entity

2.3 Tạo interface cung cấp các annotations cho từng phương thức GET, POST, PUT và DELETE

2.4 Tạo thể hiện của Retrofit để gửi yêu cầu (request) đến API

2.5 Thêm quyền truy cập Internet cho ứng dụng trong AndroidManifest.xml

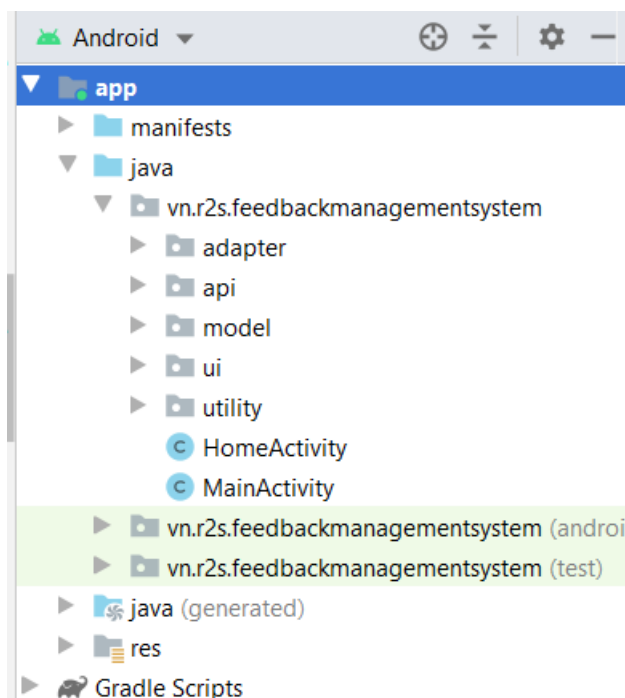
2.6 Tương tác API (Thêm, cập nhật và xóa dữ liệu)

➔ Chúng ta chỉ tập trung bước 2.6 vì những bước từ 2.1 -> 2.5 chúng ta đã thực hiện ở bài R2S\_Consuming RestfulAPI\_R\_Lab\_v1.0.docx

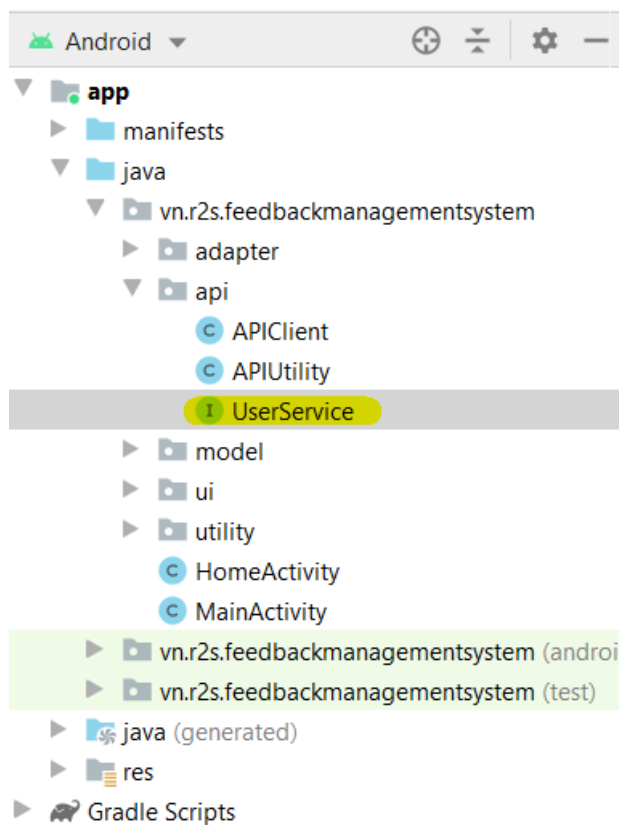
## 3. Ví dụ mẫu

### a) Mở Android Studio

- Mở project tên FeedbackManagementSystem sau khi đã hoàn tất ở bài **R2S\_Consuming RestfulAPI\_R\_Lab\_v1.0.docx**



- Thực hiện code cho **UserService**
  - Mở UserService.java



- Thay đổi code (chú ý phần đóng khung viền vàng, đó là phần các bạn phải thay đổi)

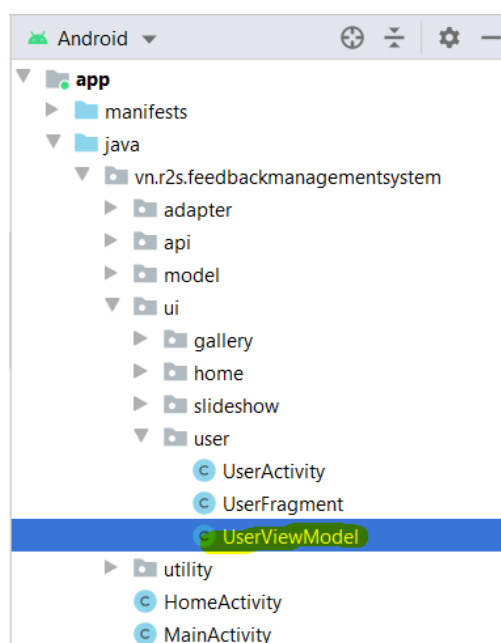
```

UserService.java
21  @GET("api/v1/users")
22  Call<ArrayList<User>> getUsers();
23
24  @GET("api/v1/users/{id}")
25  Call<User> getUserById(@Path("id") long id);
26
27  @POST("api/v1/users")
28  Call<User> postUser(@Body User user);
29
30  /**
31   * For example:
32   * @GET("/user/{username}?type={admin}")
33   * Here username is the path variable, and type is the query variable
34   * @GET("/user/{username}?type={admin}")
35   * void getUserOuth(@Path("username") String username, @Query("type") String type)
36   *
37   * @param id
38   * @param user
39   * @return
40   */
41  @PUT("api/v1/users/{id}")
42  Call<Void> updateUser(@Path("id") long id, @Body User user);
43
44  @DELETE("api/v1/user/{id}")
45  Call<Void> delete(@Path("id") long id);
46  }

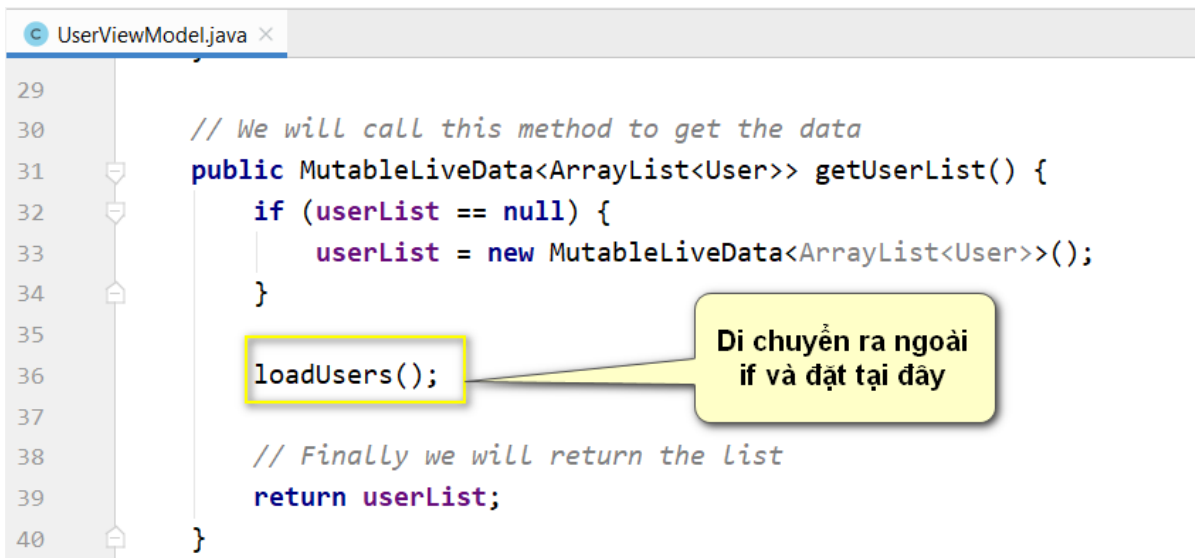
```

- Thay đổi code cho **UserViewModel**

- Mở UserViewModel.Java



- Thay đổi code cho **UserViewModel**
  - Thay đổi code cho phương thức **getUserList**



```

29
30 // We will call this method to get the data
31 public MutableLiveData<ArrayList<User>> getUserList() {
32     if (userList == null) {
33         userList = new MutableLiveData<ArrayList<User>>();
34     }
35     loadUsers();
36
37     // Finally we will return the list
38     return userList;
39
40 }
  
```

Di chuyển ra ngoài if và đặt tại đây

- Bổ sung phương thức **deleteUser**

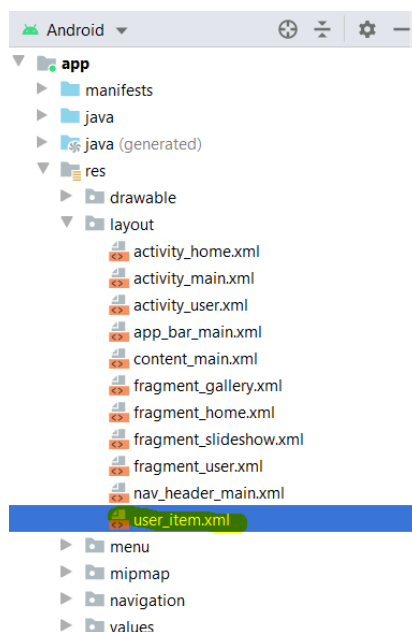


```

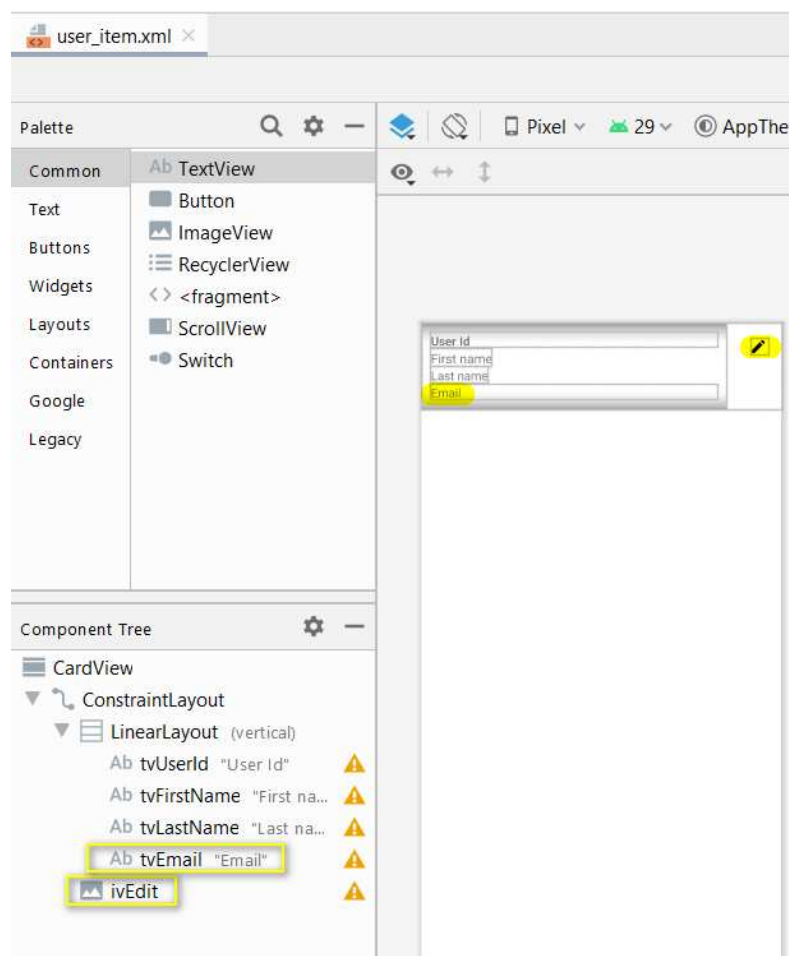
56 @Override
57 public void onFailure(Call<ArrayList<User>> call, Throwable t) {
58     Toast.makeText(getApplication().getBaseContext()
59         , text: "UserViewModel: " + t.getMessage(), Toast.LENGTH_LONG).show();
60 }
61 });
62 }
63
64 /**
65  * Delete an user
66  * @param key
67  */
68 public void deleteUser(long key) {
69     Call<Void> call = userService.delete(key);
70     call.enqueue(new Callback<Void>() {
71         @Override
72         public void onResponse(Call<Void> call, Response<Void> response) {
73             loadUsers();
74         }
75
76         @Override
77         public void onFailure(Call<Void> call, Throwable t) {
78             Toast.makeText(getApplication().getBaseContext()
79                 , text: "UserViewModel: " + t.getMessage(), Toast.LENGTH_LONG).show();
80         }
81     });
82 }
83 }
  
```

- Thay đổi layout **user\_item**

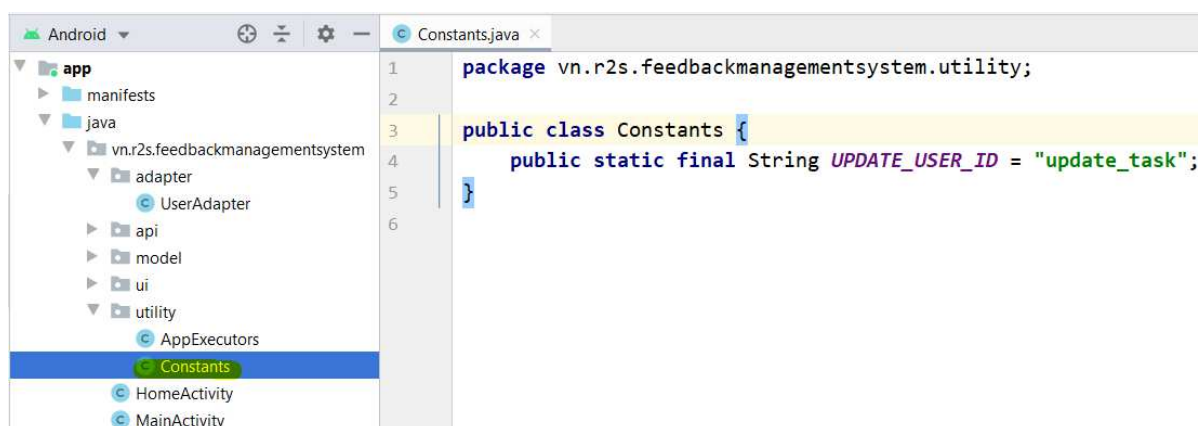
- Mở user\_item.xml



- Thêm một ImageView, đặt id là ivEdit và một TextView, đặt id là tvEmail

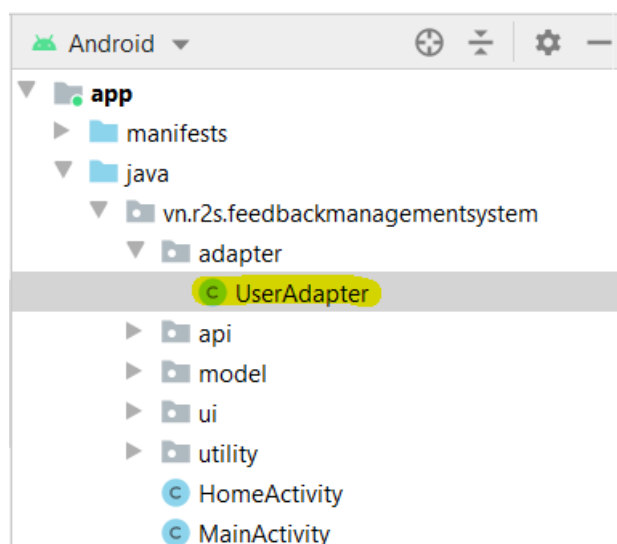


- Tạo một Java Class, đặt tên **Constants** và khai một hằng số



- Thay đổi code cho **UserAdapter**

- Mở UseAdapter.java



- Thay đổi code cho **UserAdapter**
  - Thay đổi code cho lớp ViewHolder (Chú ý phần đóng khung viền vàng)



```

UserAdapter.java
79 public TextView tvUserId;
80 public TextView tvFirstName;
81 public TextView tvLastName;
82 public TextView tvEmail;
83 public ImageView ivEdit;
84
85 // We also create a constructor that accepts the entire item row
86 // and does the view lookups to find each subview
87 public ViewHolder(@NonNull View itemView) {
88     super(itemView);
89
90     tvUserId = itemView.findViewById(R.id.tvUserId);
91     tvFirstName = itemView.findViewById(R.id.tvFirstName);
92     tvLastName = itemView.findViewById(R.id.tvLastName);
93     tvEmail = itemView.findViewById(R.id.tvEmail);
94
95     ivEdit = itemView.findViewById(R.id.ivEdit);
96     ivEdit.setOnClickListener(new View.OnClickListener() {
97         @Override
98         public void onClick(View v) {
99             long elementId = mUsers.get(getAdapterPosition()).getId();
100             Intent i = new Intent(context, UserActivity.class);
101             i.putExtra(Constants.UPDATE_USER_ID, elementId);
102             context.startActivity(i);
103         }
104     });
105 }
106

```

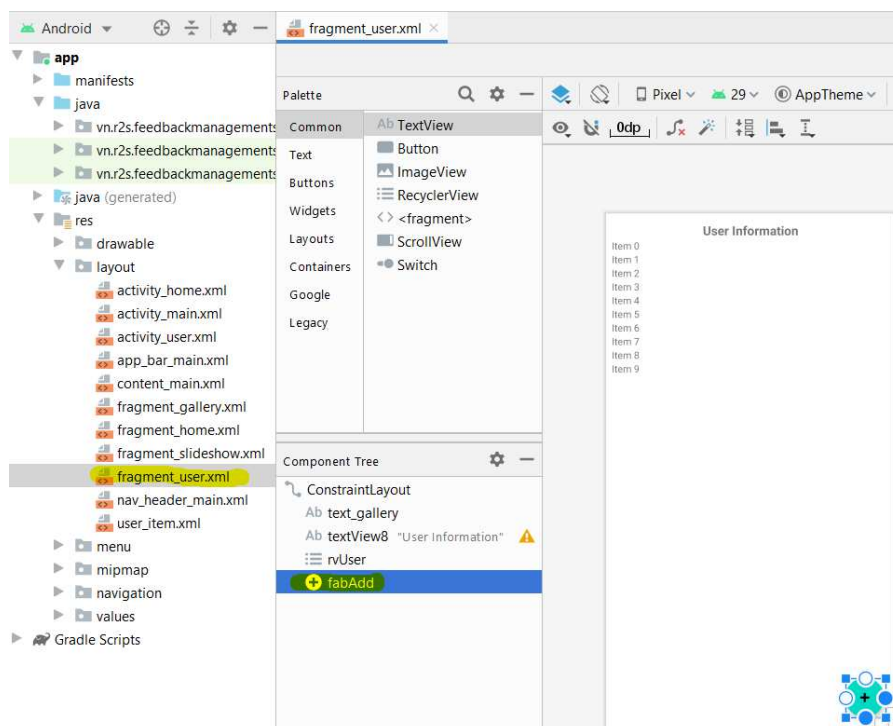
- Thay đổi code cho phương thức **onBindViewHolder**

```

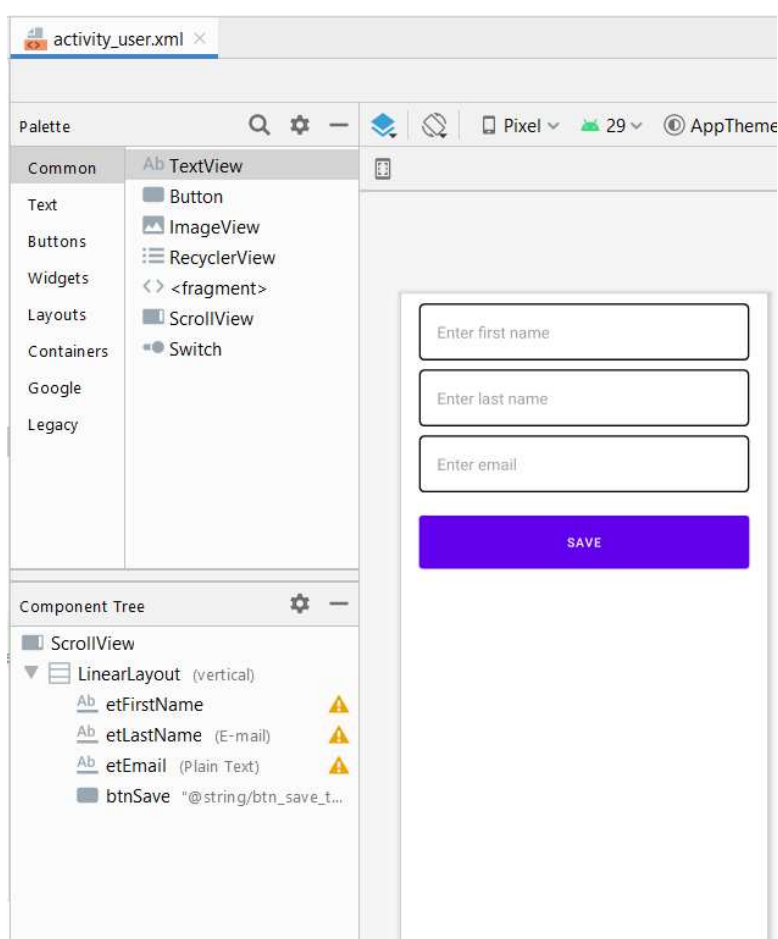
53 // Involves populating data into the item through holder
54 @Override
55 public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
56     // Get the data model based on position
57     User user = mUsers.get(position);
58
59     // Set item views based on your views and data model
60     holder.tvUserId.setText("UserId: " + user.getId());
61     holder.tvFirstName.setText("FirstName: " + user.getFirstName());
62     holder.tvLastName.setText("LastName: " + user.getLastName());
63     holder.tvEmail.setText("Email: " + user.getEmail());
64 }

```

- Thay đổi layout **fragment\_user** (Bổ sung một FloatingActionButton để thực hiện chức năng thêm)



- Tạo mới một Activity và đặt tên **UserActivity**
- Thiết kế layout



- Viết code

```

UserActivity.java x
23 public class UserActivity extends AppCompatActivity {
24
25     private EditText etFistName;
26     private EditText etLastName;
27     private EditText etEmail;
28     private Button btnSave;
29
30     private long mPersonId;
31     private Intent intent;
32     private UserService userService;
33
34     @Override
35     protected void onCreate(final Bundle savedInstanceState) {
36         super.onCreate(savedInstanceState);
37
38         setContentView(R.layout.activity_user);
39         getSupportActionBar().setDisplayHomeAsUpEnabled(true);
40
41         initView();
42
43         userService = APIUtility.getUserService();
44
45         intent = getIntent();
46
47         if (intent != null && intent.hasExtra(Constants.UPDATE_USER_ID)) {
48             btnSave.setText("Update");
49
50             mPersonId = intent.getLongExtra(Constants.UPDATE_USER_ID, defaultValue: -1);
51
52             Call<User> call = userService.getUserById(mPersonId);
53             call.enqueue(new Callback<User>() {
54                 @Override
55                 public void onResponse(Call<User> call, Response<User> response) {
56                     User user= response.body();
57                     populateUI(user);
58                 }
59
60                 @Override
61                 public void onFailure(Call<User> call, Throwable t) {
62                     Toast.makeText(getApplicationContext().getBaseContext()
63                         , text: "UserActivity: " + t.getMessage(), Toast.LENGTH_LONG).show();
64                 }
65             });
66         }
67     }
68
69     /**
70      * Populate UI
71      * @param user
72      */
73     private void populateUI(User user) {
74
75         // Check null
76         if (user == null) {
77             return;
78         }

```

```

79
80      // Display
81      etFistName.setText(user.getFirstName());
82      etLastName.setText(user.getLastName());
83      etEmail.setText(user.getEmail());
84
85  }
86
87  /**
88   * Init views
89   */
90  private void initView() {
91
92      etFistName = findViewById(R.id.etFirstName);
93      etLastName = findViewById(R.id.etLastName);
94      etEmail = findViewById(R.id.etEmail);
95
96      btnSave = findViewById(R.id.btnSave);
97      btnSave.setOnClickListener(new View.OnClickListener() {
98          @Override
99          public void onClick(View v) {
100              onSaveButtonClicked();
101          }
102      });
103  }
104
105  /**
106   * On save button clicked
107   */
108  public void onSaveButtonClicked() {
109      final User user = new User(etFistName.getText().toString()
110          , etLastName.getText().toString(),etEmail.getText().toString());
111
112      AppExecutors.getInstance().networkIO().execute(new Runnable() {
113          @Override
114          public void run() {
115              // Insert
116              if (!intent.hasExtra(Constants.UPDATE_USER_ID)) {
117                  Call<User> call = userService.postUser(user);
118                  call.enqueue(new Callback<User>() {
119                      @Override
120                      public void onResponse(Call<User> call, Response<User> response) {
121                          Toast.makeText( context: UserActivity.this, text: "Add user successful!"
122                              ,Toast.LENGTH_LONG).show();
123                      }
124
125                      @Override
126                      public void onFailure(Call<User> call, Throwable t) {
127                          Toast.makeText( context: UserActivity.this, t.getMessage()
128                              ,Toast.LENGTH_LONG).show();
129                      }
130                  });
131              // Update

```

```
132         } else {
133             Call<Void> call = userService.updateUser(mPersonId, user);
134             call.enqueue(new Callback<Void>() {
135                 @Override
136                 public void onResponse(Call<Void> call, Response<Void> response) {
137                     Toast.makeText( context: UserActivity.this, text: "Update user successful!"
138                                 ,Toast.LENGTH_LONG).show();
139                 }
140
141                 @Override
142                 public void onFailure(Call<Void> call, Throwable t) {
143                     Toast.makeText( context: UserActivity.this, t.getMessage()
144                                 ,Toast.LENGTH_LONG).show();
145                 }
146             });
147         }
148     }
149 }
150
151
152 @Override
153 public boolean onOptionsItemSelected(MenuItem item) {
154     switch (item.getItemId()) {
155         // Respond to the action bar's Up/Home button
156         case android.R.id.home:
157             onBackPressed();
158             return true;
159     }
160     return super.onOptionsItemSelected(item);
161 }
162
```

- Thay đổi code cho **UserFragment**
  - Thay đổi code cho phương thức onCreateView (Chú ý phần code đóng khung viền màu vàng)



```

UserFragment.java x
28 private UserViewModel userViewModel;
29 private RecyclerView recyclerView;
30 private ArrayList<User> data;
31 private UserAdapter adapter;
32 private FloatingActionButton faAdd;
33
34 public View onCreateView(@NonNull LayoutInflater inflater,
35                          ViewGroup container, Bundle savedInstanceState) {
36     userViewModel =
37         ViewModelProviders.of(fragment: this).get(UserViewModel.class);
38
39     View root = inflater.inflate(R.layout.fragment_user, container, attachToRoot: false);
40
41     recyclerView = root.findViewById(R.id.rvUser);
42     recyclerView.setHasFixedSize(true);
43     // Set Layout manager to position the items
44     recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
45
46     faAdd = root.findViewById(R.id.fabAdd);
47     faAdd.setOnClickListener(new View.OnClickListener() {
48         @Override
49         public void onClick(View v) {
50             startActivity(new Intent(getContext(), UserActivity.class));
51         }
52     });
53
54     data = new ArrayList<>();
55     adapter = new UserAdapter(getContext(), data);
56
57     recyclerView.setAdapter(adapter);
58
59     retrieveTasks();
60
61     new ItemTouchHelper(new ItemTouchHelper.SimpleCallback( dragDirs: 0,
62                                                         swipeDirs: ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT) {
63         @Override
64         public boolean onMove(RecyclerView recyclerView, RecyclerView.ViewHolder viewHolder,
65                               RecyclerView.ViewHolder target) {
66             return false;
67         }
68
69         // Called when a user swipes Left or right on a ViewHolder
70         @Override
71         public void onSwiped(final RecyclerView.ViewHolder viewHolder, int swipeDir) {
72             int position = viewHolder.getAdapterPosition();
73             List<User> tasks = adapter.getTasks();
74             User user = tasks.get(position);
75             userViewModel.deleteUser(user.getId());
76             retrieveTasks();
77         }
78     }).attachToRecyclerView(recyclerView);
79
80     return root;
81 }

```

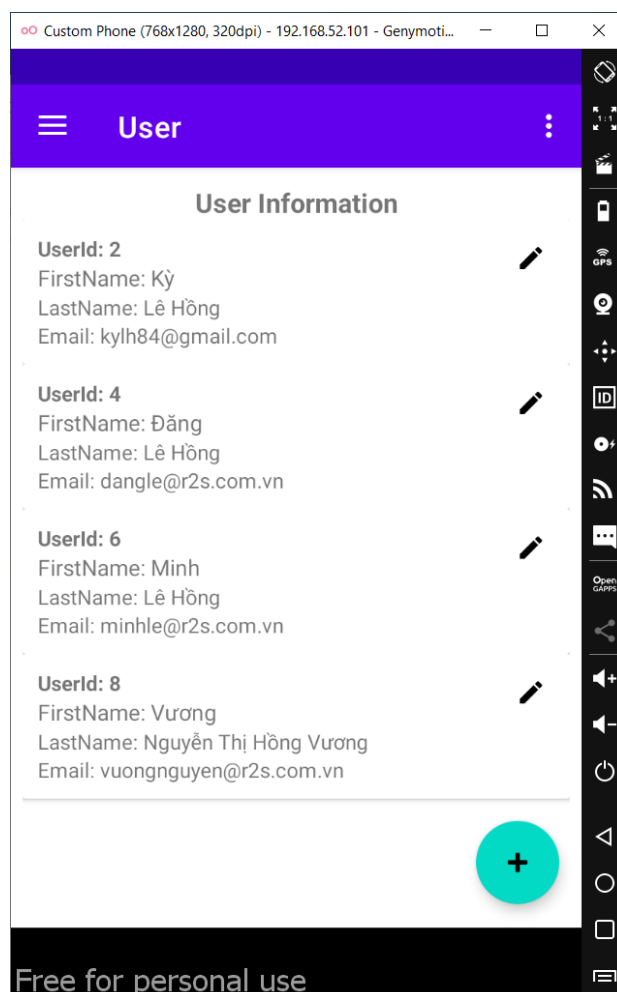
- Thêm phương thức **retrieveTask** và **onResume**

```

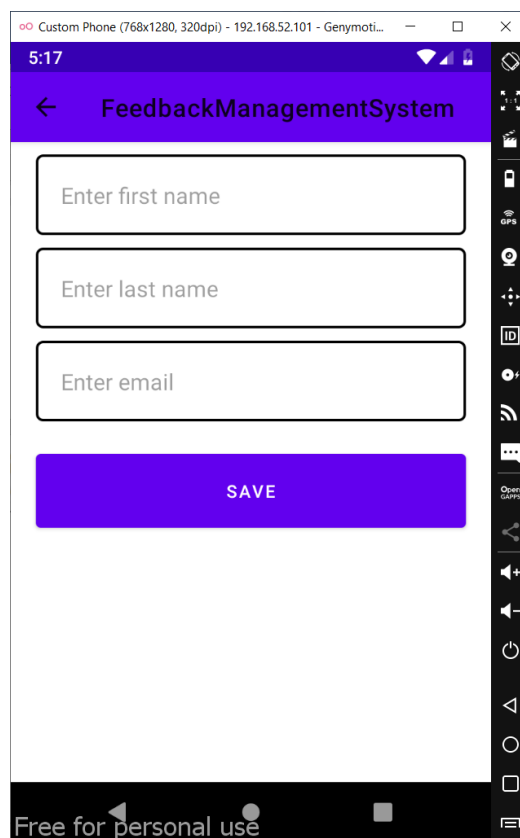
81
82  /**
83   * Retrieve users list
84   */
85   private void retrieveTasks() {
86       userModel.getUserList().observe(getViewLifecycleOwner(),
87           new Observer<ArrayList<User>>() {
88               @Override
89               public void onChanged(ArrayList<User> users) {
90                   // ReLoad data
91                   data.clear();
92                   data.addAll(users);
93                   adapter.notifyDataSetChanged();
94               }
95           });
96   }
97
98   @Override
99   public void onResume() {
100       super.onResume();
101       retrieveTasks();
102   }
103

```

- Build và run ứng dụng
  - Hiển thị danh sách user, cho phép cập nhật và thêm mới



- Màn hình thêm mới User



Custom Phone (768x1280, 320dpi) - 192.168.52.101 - Genymoti...

5:17

← FeedbackManagementSystem

Enter first name

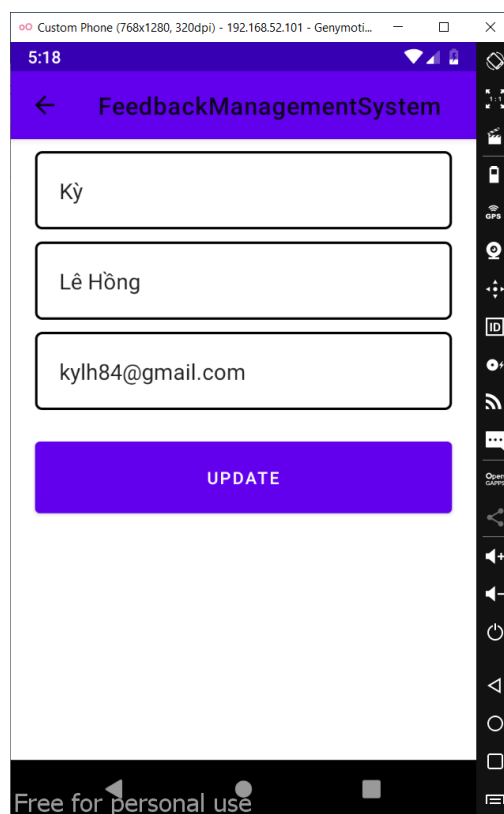
Enter last name

Enter email

SAVE

Free for personal use

- Màn hình cập nhật thông tin User



Custom Phone (768x1280, 320dpi) - 192.168.52.101 - Genymoti...

5:18

← FeedbackManagementSystem

Kỳ

Lê Hồng

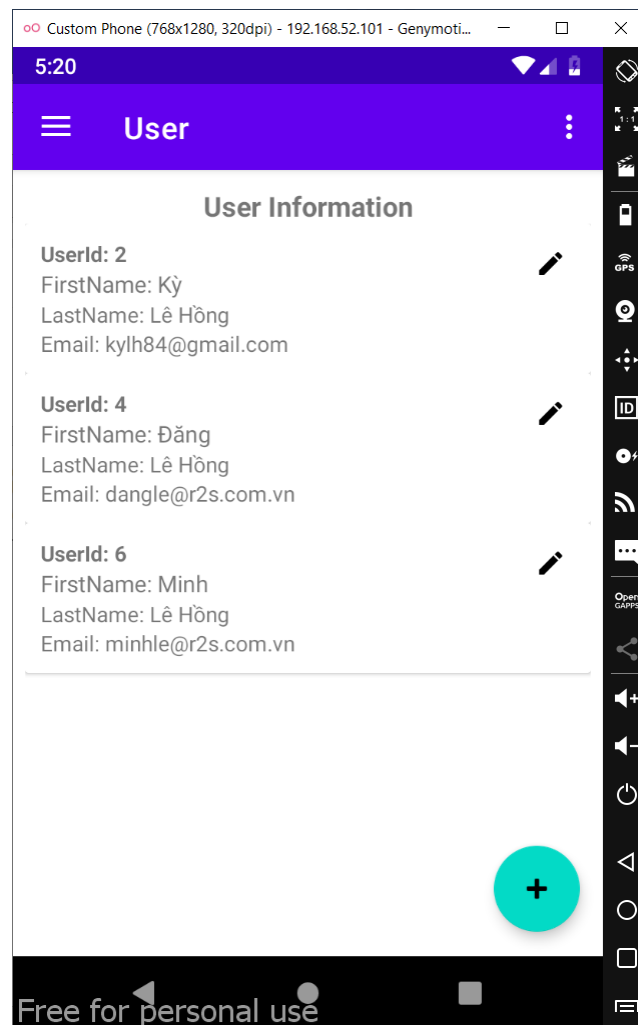
kylh84@gmail.com

UPDATE

Free for personal use



- Xóa một User bằng cách chạm, giữ và di chuyển sang trái/phải



#### 4. Tự làm

Vận dụng bài lab này để phát triển dự án Feedback Management System