Xây dựng ứng dụng mobile chụp ảnh từ camera và gửi về máy tính

# 1. Giới thiệu

* **Mục tiêu**: Ứng dụng cho phép người dùng chụp ảnh từ camera trên thiết bị Android và gửi hình ảnh đó về máy tính qua mạng.
* **Tính năng chính**:
  + Chụp ảnh từ camera của thiết bị Android.
  + Cho phép sử dụng Flash và chỉnh cỡ ảnh 3x4 và 9x16
  + Gửi ảnh đến máy tính qua socket.
  + Giao tiếp giữa ứng dụng Android và máy tính.
* **Các bước xây dựng dự án:**
  + **Bước 1**: Xây dựng giao diện ứng dụng để chụp ảnh và xem ảnh vừa chụp bằng camera, có lưu lại trên thư viện ảnh của máy.
  + **Bước 2**: Nghiên cứu và triển khai mã nguồn để thiết lập giao tiếp truyền dữ liệu giữa điện thoại và máy tính.
  + **Bước 3**: Lập trình để điện thoại có thể gửi dữ liệu đến máy tính, và lập trình máy tính để lưu trữ dữ liệu nhận được vào một file.

# 2. Yêu cầu hệ thống

* **Phần cứng**:
  + Thiết bị Android với hệ điều hành từ Android 10 trở lên.
  + Máy tính chạy hệ điều hành phù hợp với ứng dụng nhận ảnh.
* **Phần mềm**:
  + Android Studio.
  + Android SDK với các phiên bản tối thiểu 29.
  + Thư viện socket cho Android (nếu có).
  + Phần mềm máy tính nhận ảnh (ví dụ: ứng dụng socket server).

# 3. Thiết kế

## 3.1. Kiến trúc Tổng quan

### 3.1.1. Ứng dụng Android

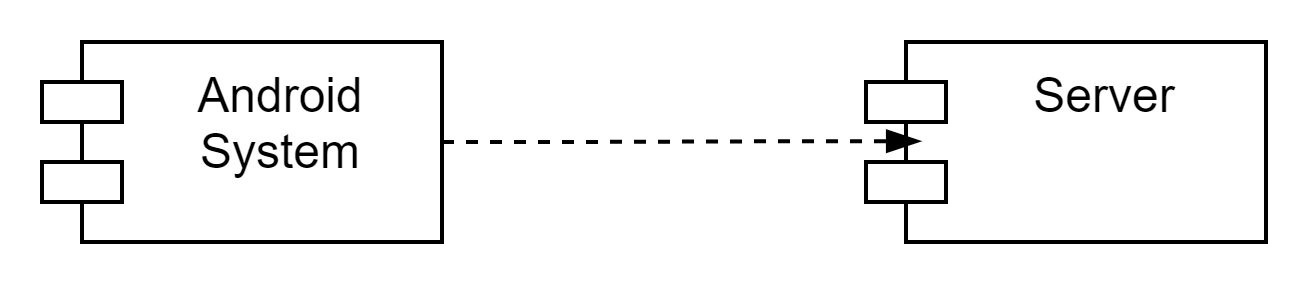
* **Chụp ảnh**:
  + **Camera API**: Sử dụng Camera API hoặc CameraX để chụp ảnh. Camera API cho phép truy cập và điều khiển camera của thiết bị, trong khi CameraX cung cấp API dễ sử dụng và tích hợp tốt với các thành phần khác của Android.
  + **Xử lý ảnh**: Sau khi chụp ảnh, ảnh sẽ được lưu vào bộ nhớ tạm thời hoặc bộ nhớ ngoài của thiết bị. Bạn có thể sử dụng các thư viện như Bitmap để xử lý ảnh trước khi gửi.
* **Gửi ảnh qua socket**:
  + **Socket Programming**: Sử dụng socket để thiết lập kết nối mạng giữa ứng dụng Android và máy tính. Trong Android, bạn có thể sử dụng lớp Socket từ gói java.net để thực hiện việc này.
  + **Dữ liệu ảnh**: Chuyển đổi ảnh thành định dạng byte (như JPEG) để gửi qua socket. Sử dụng ByteArrayOutputStream để chuyển đổi Bitmap thành byte array và gửi qua socket.

### 3.1.2. Máy tính

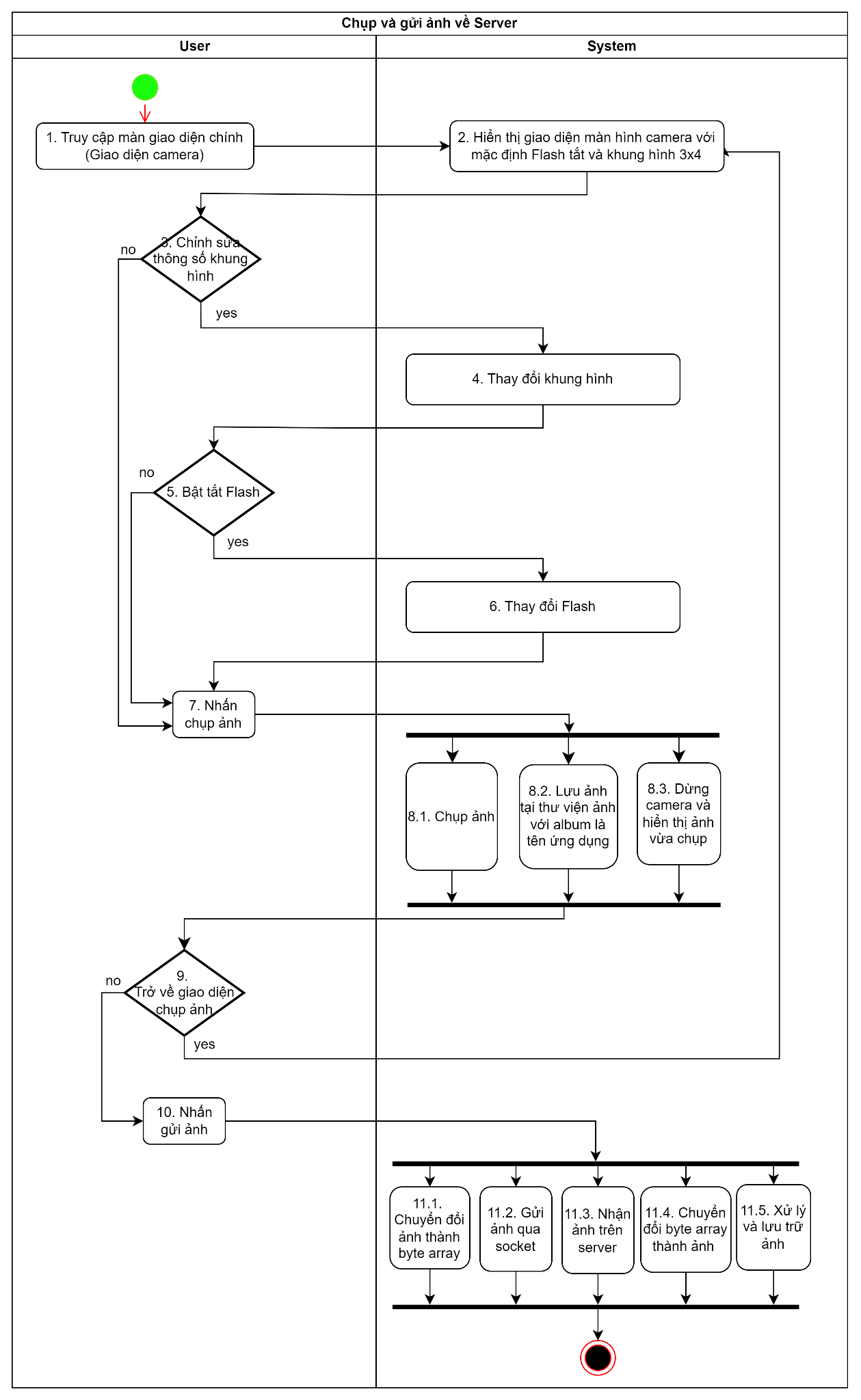
* **Nhận ảnh qua socket**:
  + **Socket Server**: Thiết lập một socket server để lắng nghe các kết nối từ ứng dụng Android. Bạn có thể sử dụng thư viện hoặc framework socket server phù hợp với ngôn ngữ lập trình mà máy tính đang chạy.
  + **Nhận và xử lý ảnh**: Nhận dữ liệu byte từ socket, sau đó chuyển đổi byte array trở lại thành ảnh (ví dụ: sử dụng ImageIO trong Java hoặc các thư viện tương tự trong các ngôn ngữ khác).
* **Lưu trữ hoặc xử lý ảnh**:
  + **Lưu trữ**: Lưu ảnh vào hệ thống tập tin hoặc cơ sở dữ liệu theo định dạng phù hợp.
  + **Xử lý**: Thực hiện các xử lý cần thiết trên ảnh như phân tích, chỉnh sửa hoặc xử lý hình ảnh.

## 3.2. Diagrams

### 3.2.1. Sơ đồ hệ thống



### 3.2.2. Sơ đồ luồng dữ liệu



**Giải thích**:

* **Chụp ảnh từ Camera**: Quá trình chụp và xử lý ảnh trên thiết bị Android.
* **Chuyển đổi ảnh thành byte array**: Chuyển đổi ảnh chụp thành định dạng byte để gửi qua socket.
* **Gửi dữ liệu qua socket**: Gửi byte array qua kết nối socket.
* **Nhận dữ liệu byte array**: Máy tính nhận dữ liệu từ ứng dụng Android qua socket.
* **Chuyển đổi byte array thành ảnh**: Chuyển đổi byte array trở lại thành ảnh.
* **Lưu trữ hoặc Xử lý ảnh**: Lưu ảnh vào hệ thống tập tin hoặc thực hiện xử lý ảnh.

## 3.3. Giao diện Người dùng

* **Màn hình chính**:
  + **Nút chụp ảnh**: Cho phép người dùng kích hoạt camera để chụp ảnh. Nút này có thể nằm ở vị trí nổi bật trên giao diện.
  + **Xem ảnh đã chụp**: Cung cấp một cách để người dùng xem ảnh đã chụp trước khi gửi.
  + **Flash**: Cho phép người dùng bật tắt đèn flash để chụp ảnh
  + **Điều chỉnh kích thước khung hình**: Người dùng có thể điều chỉnh khung 3x4 hoặc khung hình 9x16



# 4. Hướng dẫn chi tiết

* **Trên Android Studio**:
  + Cài đặt và cấu hình dự án trong Android Studio.
  + Các thư viện cần thiết và cách thêm vào dự án.
* **Trên máy tính**:
  + Hướng dẫn cài đặt phần mềm nhận ảnh (nếu có) và cấu hình để lắng nghe kết nối từ ứng dụng Android.

## 4.1. Ứng dụng Android

### 4.1.1. Cấu hình quyền và các thư viện cần thiết

#### 4.1.1.1. Manifest

Manifest là tệp tin cấu hình chính của một ứng dụng Android, xác định các quyền, tính năng, hoạt động, dịch vụ và các thành phần khác mà ứng dụng cần. Dưới đây là các quyền và cấu hình đã được khai báo trong tệp AndroidManifest.xml:

| <?xml version=**"1.0"** encoding=**"utf-8"**?> <manifest xmlns:android=**"http://schemas.android.com/apk/res/android"**  xmlns:tools=**"http://schemas.android.com/tools"**>   **<!-- Quyền truy cập Internet -->**  <uses-permission android:name=**"android.permission.INTERNET"** />    **<!-- Yêu cầu về camera -->**  <uses-feature  android:name=**"android.hardware.camera"**  android:required=**"false"** />  <uses-feature android:name=**"android.hardware.camera.autofocus"** />   **<!-- Quyền sử dụng camera -->**  <uses-permission android:name=**"android.permission.CAMERA"** />    **<!-- Quyền đọc/ghi bộ nhớ ngoài -->**  <uses-permission android:name=**"android.permission.WRITE\_EXTERNAL\_STORAGE"** />  <uses-permission android:name=**"android.permission.READ\_EXTERNAL\_STORAGE"** />   <application  android:allowBackup=**"true"**  android:dataExtractionRules=**"@xml/data\_extraction\_rules"**  android:fullBackupContent=**"@xml/backup\_rules"**  android:icon=**"@mipmap/ic\_launcher"**  android:label=**"@string/app\_name"**  android:requestLegacyExternalStorage=**"true"**  android:roundIcon=**"@mipmap/ic\_launcher\_round"**  android:supportsRtl=**"true"**  android:theme=**"@style/Theme.Camera"**  tools:targetApi=**"31"**>   **<!-- Cấu hình Activity chính -->**  <activity  android:name=**".CameraxActivity"**  android:exported=**"true"**>  <intent-filter>  <action android:name=**"android.intent.action.MAIN"** />  <category android:name=**"android.intent.category.LAUNCHER"** />  </intent-filter>  </activity>  </application>  </manifest> |
| --- |

#### 4.1.1.2. build.gradle.kts

1. **Plugins**

Phần này dùng để áp dụng plugin `androidApplication` từ file `libs.versions.toml`, nếu bạn đã cấu hình đúng cách. Đây là một cách hiện đại để quản lý các dependency và plugin.

| plugins {  alias(libs.plugins.androidApplication) } |
| --- |

1. **Cấu hình Android**

* **namespace**: Xác định tên gói (package) cho ứng dụng của bạn.
* **compileSdk**: Xác định phiên bản API của Android mà dự án sẽ biên dịch.
* **defaultConfig**: Chứa các thiết lập cấu hình cơ bản như `applicationId`, `minSdk`, `targetSdk`, `versionCode`, và `versionName`.
* **testInstrumentationRunne**`: Chỉ định runner cho các bài kiểm tra tự động.

| android {  namespace = **"work.ngangiang.camera"**  compileSdk = 34  defaultConfig {  applicationId = **"work.ngangiang.camera"**  minSdk = 29  targetSdk = 34  versionCode = 1  versionName = **"1.0"**  testInstrumentationRunner = **"androidx.test.runner.AndroidJUnitRunner"**  } } |
| --- |

1. **Loại bản dựng (Build Types)**

**release**: Định nghĩa cấu hình cho bản phát hành, ví dụ như `isMinifyEnabled` để tối ưu hóa mã và các file cấu hình ProGuard.

| buildTypes {  release {  isMinifyEnabled = **false**  proguardFiles(  getDefaultProguardFile(**"proguard-android-optimize.txt"**),  "proguard-rules.pro"  )  }  } |
| --- |

1. **Tùy chọn biên dịch (Compile Options)**

Đặt khả năng tương thích của nguồn và mục tiêu cho Java là phiên bản 1.8.

| compileOptions {  sourceCompatibility = JavaVersion.VERSION\_1\_8  targetCompatibility = JavaVersion.VERSION\_1\_8 } |
| --- |

1. **Dependencies (Phụ thuộc)**

* Đoạn mã này khai báo các thư viện mà dự án của bạn phụ thuộc vào, chẳng hạn như `appcompat`, `material`, `constraintlayout`, `uiautomator`, `monitor`, và các thư viện kiểm tra như `junit`, `espresso`.
* Thư viện CameraX từ AndroidX được sử dụng để tích hợp các tính năng liên quan đến camera.
* commons-net là một thư viện tiện ích mạng.

| dependencies {  implementation(libs.appcompat)  implementation(libs.material)  implementation(libs.activity)  implementation(libs.constraintlayout)  implementation(**"androidx.exifinterface:exifinterface:1.3.6"**)  implementation(libs.uiautomator)  implementation(libs.monitor)  testImplementation(libs.junit)  androidTestImplementation(libs.ext.junit)  androidTestImplementation(libs.espresso.core)  val camerax\_version = **"1.2.0"**  implementation (**"androidx.camera:camera-core:$camerax\_version"**)  implementation (**"androidx.camera:camera-camera2:$camerax\_version"**)  implementation (**"androidx.camera:camera-lifecycle:$camerax\_version"**)  implementation (**"androidx.camera:camera-view:1.2.0-alpha04"**)   implementation (**"commons-net:commons-net:3.8.0"**) } |
| --- |

### 4.1.2. FileTransferTask

#### 4.1.2.1. Tổng quan

**FileTransferTask** là một lớp được triển khai trong ứng dụng Android, thuộc package **work.ngangiang.camera.Services**. Lớp này mở rộng từ AsyncTask<Uri, Void, Boolean>, cho phép thực hiện tác vụ chuyển file từ thiết bị Android lên server thông qua mạng mà không làm ảnh hưởng đến giao diện người dùng (UI). Tác vụ này chủ yếu được sử dụng để gửi file hình ảnh.

#### 4.1.2.2. Thành phần và cấu trúc

1. **Thuộc tính**

* SERVER\_IP: Một hằng số String lưu trữ địa chỉ IP của server mà file sẽ được gửi đến.
* SERVER\_PORT: Một hằng số int chỉ định cổng của server.
* context: Biến kiểu Context, đại diện cho ngữ cảnh của ứng dụng Android hiện tại, cho phép truy cập các tài nguyên và dịch vụ của ứng dụng.

| **private** **static** **final** String SERVER\_IP = **"192.168.1.30"**; **private** **static** **final** **int** SERVER\_PORT = 100; **private** Context context; |
| --- |

1. **Constructor**

Constructor này nhận một đối tượng Context và gán cho thuộc tính context của lớp. Context được sử dụng để truy cập các tài nguyên và dịch vụ của hệ thống Android, ví dụ như ContentResolver để mở InputStream từ Uri.

| **public** FileTransferTask(Context context) {  **this**.context = context; } |
| --- |

1. **Phương thức doInBackground**

**Mô tả**: Phương thức này thực hiện tác vụ chính của FileTransferTask, chạy trên một luồng nền để không ảnh hưởng đến UI.

**Tham số**:

* uris: Một mảng các Uri đại diện cho các file cần gửi. Ở đây chỉ sử dụng Uri đầu tiên (uris[0]).

**Hoạt động**:

* **Mở kết nối Socket**: Tạo kết nối đến server thông qua địa chỉ IP và cổng đã định trước.
* **Mở InputStream**: Sử dụng ContentResolver để mở InputStream từ Uri của file. Nếu không thể mở InputStream, ghi lỗi vào log và trả về false.
* **Lấy tên file**: Sử dụng phương thức getFileName để lấy tên file từ Uri.
* **Gửi tên file**: Gửi tên file đến server, thêm ký tự xuống dòng (\n) để phân cách.
* **Gửi dữ liệu file**: Đọc dữ liệu từ BufferedInputStream và ghi vào OutputStream của socket.
* **Xử lý lỗi**: Bắt và ghi lại bất kỳ ngoại lệ nào phát sinh trong quá trình thực hiện, sau đó trả về false.

| @Override **protected** Boolean doInBackground(Uri... uris) {  Uri fileUri = uris[0];  **try** (  Socket socket = **new** Socket(SERVER\_IP, SERVER\_PORT);  InputStream inputStream = context.getContentResolver().openInputStream(fileUri);  BufferedInputStream bis = **new** BufferedInputStream(inputStream);  OutputStream os = socket.getOutputStream()  ) {  **if** (inputStream == **null**) {  Log.e(**"FileTransferTask"**, **"InputStream is null."**);  **return** **false**;  }   String fileName = getFileName(fileUri);  os.write((fileName + **"\n"**).getBytes());  **byte**[] buffer = **new** **byte**[4096];  **int** bytesRead;  **while** ((bytesRead = bis.read(buffer)) != -1) {  os.write(buffer, 0, bytesRead);  }  os.flush();  **return** **true**;  } **catch** (Exception e) {  e.printStackTrace();  **return** **false**;  } } |
| --- |

1. **Phương thức onPostExecute**

**Mô tả:** Được gọi sau khi doInBackground hoàn thành và chạy trên luồng giao diện người dùng (UI thread).

**Tham số:**

result: Kết quả của doInBackground, là true nếu gửi file thành công, false nếu thất bại.

**Hoạt động: Hiển thị một thông báo Toast để thông báo cho người dùng về kết quả của việc gửi file.**

| @Override **protected** **void** onPostExecute(Boolean result) {  **if** (result) {  Toast.makeText(context, **"File ảnh đã được gửi thành công!"**, Toast.LENGTH\_LONG).show();  } **else** {  Toast.makeText(context, **"Không thể gửi file ảnh."**, Toast.LENGTH\_LONG).show();  } } |
| --- |

1. **Phương thức getFileName**

**Mô tả:** Phương thức này dùng để lấy tên file từ Uri bằng cách truy vấn ContentResolver.

**Tham số:** uri: Uri của file.

**Hoạt động:**

Sử dụng ContentResolver để truy vấn thông tin của file.

Sử dụng Cursor để lấy tên file từ cột DISPLAY\_NAME trong MediaStore.

Trả về tên file, nếu không thể lấy được thì trả về tên mặc định là "unknown.jpg".

| **private** String getFileName(Uri uri) {  ContentResolver contentResolver = context.getContentResolver();  String[] projection = {android.provider.MediaStore.Images.Media.DISPLAY\_NAME};  **try** (Cursor cursor = contentResolver.query(uri, projection, **null**, **null**, **null**)) {  **if** (cursor != **null** && cursor.moveToFirst()) {  **int** nameIndex = cursor.getColumnIndexOrThrow(android.provider.MediaStore.Images.Media.DISPLAY\_NAME);  **return** cursor.getString(nameIndex);  }  }  **return** **"unknown.jpg"**; *// Tên mặc định nếu không thể xác định* } |
| --- |

#### 4.1.2.3. Cách sử dụng

Để sử dụng FileTransferTask, bạn cần tạo một thể hiện của lớp này và gọi phương thức execute với Uri của file cần gửi. Ví dụ:

| Uri fileUri = ...; FileTransferTask fileTransferTask = **new** FileTransferTask(context); fileTransferTask.execute(fileUri); |
| --- |

Hoặc

| **new** FileTransferTask(context).execute(fileUri); |
| --- |

### 4.1.3. CameraXActivity

Lớp CameraxActivity tích hợp nhiều tính năng để cung cấp trải nghiệm chụp ảnh trên Android. Nó cho phép người dùng thay đổi tỷ lệ khung hình, bật/tắt flash, và thực hiện các thao tác với camera một cách dễ dàng thông qua giao diện người dùng trực quan. Bằng cách sử dụng CameraX, ứng dụng có thể tận dụng các khả năng hiện đại của camera với cấu hình đơn giản và hiệu quả.

#### 4.1.3.1. Giao diện

**Layout gốc**: ConstraintLayout - Đây là loại layout mạnh mẽ, linh hoạt, cho phép bạn định vị các view dựa trên các ràng buộc (constraints) với các view khác hoặc với chính layout đó.

**Các thành phần chính**:

* PreviewView: Dùng để hiển thị preview từ camera.
* LinearLayout: Chứa các nút điều khiển như flash, lựa chọn tỷ lệ khung hình.
* Các nút điều khiển như ImageButton cho các chức năng như chụp ảnh, đóng và gửi ảnh.

| <?xml version=**"1.0"** encoding=**"utf-8"**?> <**androidx.constraintlayout.widget.ConstraintLayout** xmlns:android=**"http://schemas.android.com/apk/res/android"**  xmlns:app=**"http://schemas.android.com/apk/res-auto"**  xmlns:tools=**"http://schemas.android.com/tools"**  android:id=**"@+id/main"**  android:layout\_width=**"match\_parent"**  android:layout\_height=**"match\_parent"**  android:background=**"@color/black"**  tools:context=**".CameraxActivity"**>   <**androidx.camera.view.PreviewView**  android:id=**"@+id/viewFinder"**  android:layout\_width=**"0dp"**  android:layout\_height=**"0dp"**  android:layout\_marginBottom=**"10dp"**  app:layout\_constraintBottom\_toTopOf=**"@+id/btnCapture"**  app:layout\_constraintEnd\_toEndOf=**"parent"**  app:layout\_constraintHorizontal\_bias=**"1.0"**  app:layout\_constraintStart\_toStartOf=**"parent"**  app:layout\_constraintTop\_toBottomOf=**"@+id/linearLayout"**>  </**androidx.camera.view.PreviewView**>   <**LinearLayout**  android:id=**"@+id/linearLayout"**  android:layout\_width=**"match\_parent"**  android:layout\_height=**"wrap\_content"**  android:gravity=**"center\_vertical"**  android:weightSum=**"3"**  app:layout\_constraintTop\_toTopOf=**"parent"**>   <**ImageButton**  android:id=**"@+id/btnFlash"**  android:layout\_width=**"20dp"**  android:layout\_height=**"match\_parent"**  android:layout\_alignParentEnd=**"true"**  android:layout\_weight=**"1"**  android:background=**"?attr/selectableItemBackgroundBorderless"**  android:padding=**"10dp"**  android:scaleType=**"fitCenter"**  android:src=**"@drawable/ic\_flash\_off"**  app:layout\_constraintStart\_toStartOf=**"parent"**  app:layout\_constraintTop\_toTopOf=**"parent"** />   <**TextView**  android:id=**"@+id/btnAspectRatio3\_4"**  android:layout\_width=**"wrap\_content"**  android:layout\_height=**"wrap\_content"**  android:layout\_weight=**"1"**  android:paddingVertical=**"10dp"**  android:text=**"3:4"**  android:textAlignment=**"center"**  android:textColor=**"#fff"**  android:textStyle=**"bold"** />   <**TextView**  android:id=**"@+id/btnAspectRatio9\_16"**  android:layout\_width=**"wrap\_content"**  android:layout\_height=**"wrap\_content"**  android:layout\_weight=**"1"**  android:paddingVertical=**"10dp"**  android:text=**"9:16"**  android:textAlignment=**"center"**  android:textColor=**"#fff"**  android:textStyle=**"bold"** />   </**LinearLayout**>   <**ImageButton**  android:id=**"@+id/btnClose"**  android:layout\_width=**"80dp"**  android:layout\_height=**"80dp"**  android:layout\_alignParentEnd=**"true"**  android:layout\_marginStart=**"10dp"**  android:layout\_marginBottom=**"10dp"**  android:background=**"?attr/selectableItemBackgroundBorderless"**  android:padding=**"20dp"**  android:scaleType=**"fitCenter"**  android:src=**"@drawable/ic\_close"**  android:visibility=**"gone"**  app:layout\_constraintBottom\_toBottomOf=**"parent"**  app:layout\_constraintStart\_toStartOf=**"parent"** />   <**ImageButton**  android:id=**"@+id/btnSend"**  android:layout\_width=**"80dp"**  android:layout\_height=**"80dp"**  android:layout\_alignParentEnd=**"true"**  android:layout\_marginBottom=**"10dp"**  android:background=**"?attr/selectableItemBackgroundBorderless"**  android:padding=**"6dp"**  android:scaleType=**"fitCenter"**  android:src=**"@drawable/ic\_send"**  android:visibility=**"gone"**  app:layout\_constraintBottom\_toBottomOf=**"parent"**  app:layout\_constraintEnd\_toEndOf=**"parent"**  app:layout\_constraintStart\_toStartOf=**"parent"** />   <**ImageButton**  android:id=**"@+id/btnCapture"**  android:layout\_width=**"70dp"**  android:layout\_height=**"70dp"**  android:layout\_alignParentEnd=**"true"**  android:layout\_marginBottom=**"10dp"**  android:background=**"?attr/selectableItemBackgroundBorderless"**  android:scaleType=**"fitCenter"**  android:src=**"@drawable/btncapture"**  android:visibility=**"visible"**  app:layout\_constraintBottom\_toBottomOf=**"parent"**  app:layout\_constraintEnd\_toEndOf=**"parent"**  app:layout\_constraintStart\_toStartOf=**"parent"** /> </**androidx.constraintlayout.widget.ConstraintLayout**> |
| --- |

#### 4.1.3.2. Mã nguồn

**Lớp CameraxActivity** kế thừa từ AppCompatActivity và cung cấp các chức năng chụp ảnh với CameraX, một thư viện Android cung cấp các API để tương tác với camera của thiết bị.

**Khai báo thư viện**

Trong lớp CameraxActivity, các phần import đóng vai trò rất quan trọng trong việc cung cấp các chức năng cần thiết để thực hiện nhiệm vụ của ứng dụng. Dưới đây là bảng chi tiết mô tả các thư viện và lớp được nhập vào, cùng với chức năng cụ thể của chúng:

| **Import** | **Chức năng** |
| --- | --- |
| **android.Manifest** | Cung cấp các hằng số cần thiết để kiểm tra và yêu cầu các quyền truy cập như camera và bộ nhớ, điều quan trọng để ứng dụng có thể hoạt động đúng cách. |
| **android.annotation.SuppressLint** | Giúp bỏ qua các cảnh báo từ trình biên dịch về hành vi không an toàn hoặc không khuyến khích, giúp giảm bớt các cảnh báo không cần thiết trong quá trình phát triển. |
| **android.content.ContentValues** | Dùng để lưu trữ các giá trị cột cho cơ sở dữ liệu hoặc các ContentProvider, ví dụ, lưu trữ thông tin về ảnh khi lưu trữ vào bộ sưu tập ảnh của thiết bị. |
| **android.content.pm.PackageManager** | Cung cấp các phương thức để kiểm tra các quyền truy cập và thông tin về các ứng dụng cài đặt, rất cần thiết để quản lý quyền truy cập trong ứng dụng. |
| **android.net.Uri** | Đại diện cho các địa chỉ URI, cho phép chỉ định và làm việc với địa chỉ của dữ liệu, như URI của ảnh đã chụp từ camera. |
| **android.os.Build** | Cung cấp thông tin về phiên bản hệ điều hành Android, giúp kiểm tra phiên bản hệ điều hành để yêu cầu các quyền hoặc thực hiện các hành động cụ thể. |
| **android.os.Bundle** | Được sử dụng để lưu trữ và truyền dữ liệu giữa các component của ứng dụng, như lưu trữ trạng thái khi Activity được tạo lại sau khi bị hủy. |
| **android.os.Environment** | Cung cấp các công cụ để truy cập vào các thư mục đặc biệt trong hệ thống file của thiết bị, như thư mục chứa hình ảnh. |
| **android.os.StrictMode** | Dùng để theo dõi và kiểm tra các hành vi không mong muốn trong ứng dụng, như thao tác mạng trên thread chính, và giúp đảm bảo hiệu suất và độ ổn định. |
| **android.provider.MediaStore** | Cung cấp quyền truy cập vào bộ sưu tập media trên thiết bị, rất cần thiết để lưu trữ và truy xuất các file hình ảnh. |
| **android.util.Log** | Công cụ ghi lại thông tin log, cảnh báo và lỗi, giúp theo dõi hoạt động của ứng dụng và gỡ lỗi trong quá trình phát triển. |
| **android.view.ScaleGestureDetector** | Cung cấp các công cụ để phát hiện và xử lý các cử chỉ zoom từ người dùng, như pinch-to-zoom, để cải thiện trải nghiệm người dùng. |
| **android.widget.Button** | Đại diện cho các nút bấm trong giao diện người dùng, cho phép người dùng thực hiện các hành động trong ứng dụng. |
| **android.widget.ImageButton** | Đại diện cho các nút bấm hình ảnh, cho phép người dùng thực hiện các hành động thông qua các nút bấm có hình ảnh. |
| **android.widget.LinearLayout** | Layout container cho phép sắp xếp các thành phần con theo chiều ngang hoặc dọc, giúp tổ chức giao diện người dùng. |
| **android.widget.Toast** | Hiển thị các thông báo ngắn gọn và tạm thời cho người dùng, như thông báo thành công hoặc lỗi. |
| **androidx.activity.EdgeToEdge** | Cung cấp công cụ để làm việc với các insets của hệ thống, cho phép nội dung giao diện người dùng tràn viền màn hình. |
| **androidx.annotation.NonNull** | Đánh dấu các tham số hoặc giá trị không được phép là null, giúp đảm bảo tính toàn vẹn của dữ liệu và giảm lỗi runtime. |
| **androidx.appcompat.app.AppCompatActivity** | Lớp cơ sở cho các activity hỗ trợ ActionBar của AppCompat, cung cấp tính tương thích ngược với các phiên bản Android cũ hơn. |
| **androidx.camera.core.AspectRatio** | Cung cấp các hằng số cho các tỷ lệ khung hình của camera, như 4:3 và 16:9, cho phép chọn tỷ lệ khung hình phù hợp với nhu cầu của ứng dụng. |
| **androidx.camera.core.Camera** | Đại diện cho camera đang được sử dụng, cho phép thực hiện các hoạt động như chụp ảnh và điều khiển các tính năng của camera. |
| **androidx.camera.core.CameraControl** | Cung cấp các công cụ để điều khiển các chức năng của camera, như zoom, flash và focus, giúp tinh chỉnh các thiết lập camera. |
| **androidx.camera.core.CameraInfo** | Cung cấp thông tin về trạng thái hiện tại của camera, như tỷ lệ zoom và chế độ flash, để phản hồi chính xác với người dùng. |
| **androidx.camera.core.CameraSelector** | Công cụ để chọn camera, ví dụ như camera trước hoặc sau, giúp thay đổi nguồn camera tùy theo nhu cầu của ứng dụng. |
| **androidx.camera.core.ImageCapture** | Công cụ chụp ảnh, cho phép chụp ảnh và lưu trữ vào bộ nhớ thiết bị hoặc thực hiện các thao tác xử lý ảnh. |
| **androidx.camera.core.ImageCaptureException** | Ngoại lệ ném ra khi có lỗi xảy ra trong quá trình chụp ảnh, giúp xử lý lỗi và thông báo cho người dùng. |
| **androidx.camera.core.Preview** | Công cụ để hiển thị preview từ camera trên giao diện người dùng, cho phép người dùng xem trước hình ảnh hoặc video trước khi chụp. |
| **androidx.camera.lifecycle.ProcessCameraProvider** | Cung cấp các công cụ để liên kết camera với vòng đời của activity hoặc fragment, đảm bảo camera được quản lý đúng cách theo vòng đời của ứng dụng. |
| **androidx.camera.view.PreviewView** | View dùng để hiển thị preview từ camera trên giao diện người dùng, cho phép người dùng xem trước nội dung camera. |
| **androidx.core.app.ActivityCompat** | Cung cấp các phương thức tiện ích để kiểm tra và yêu cầu quyền truy cập vào các tính năng hệ thống, như quyền camera và bộ nhớ. |
| **androidx.core.content.ContextCompat** | Cung cấp các phương thức tiện ích cho ngữ cảnh và các thành phần khác của ứng dụng, như kiểm tra quyền truy cập và truy xuất thông tin. |
| **androidx.core.graphics.Insets** | Đại diện cho các insets của hệ thống xung quanh nội dung UI, giúp quản lý và điều chỉnh bố cục giao diện người dùng. |
| **androidx.core.view.ViewCompat** | Cung cấp các phương thức tiện ích cho các view, giúp duy trì khả năng tương thích ngược với các phiên bản Android cũ hơn. |
| **androidx.core.view.WindowInsetsCompat** | Đại diện cho các insets của cửa sổ, bao gồm các phần như thanh trạng thái và thanh điều hướng, để điều chỉnh giao diện người dùng cho phù hợp. |
| **com.google.common.util.concurrent.ListenableFuture** | Cung cấp các công cụ làm việc với các đối tượng Future, cho phép lắng nghe và xử lý kết quả khi Future hoàn thành, đặc biệt hữu ích cho các tác vụ bất đồng bộ. |
| **java.util.Objects** | Cung cấp các phương thức tiện ích cho việc làm việc với đối tượng, như kiểm tra null, so sánh, và lấy hash code, giúp cải thiện mã nguồn và giảm lỗi. |
| **java.util.concurrent.ExecutionException** | Ngoại lệ ném ra khi tính toán không đồng bộ gặp lỗi, giúp xử lý các lỗi liên quan đến các tác vụ bất đồng bộ. |
| **work.ngangiang.camera.Services.FileTransferTask** | Lớp định nghĩa FileTransferTask để thực hiện nhiệm vụ gửi file ảnh, giúp chuyển ảnh từ thiết bị đến server. |

| **package** work.ngangiang.camera;  **import** android.Manifest; **import** android.annotation.SuppressLint; **import** android.content.ContentValues; **import** android.content.pm.PackageManager; **import** android.net.Uri; **import** android.os.Build; **import** android.os.Bundle; **import** android.os.Environment; **import** android.os.StrictMode; **import** android.provider.MediaStore; **import** android.util.Log; **import** android.view.ScaleGestureDetector; **import** android.widget.Button; **import** android.widget.ImageButton; **import** android.widget.LinearLayout; **import** android.widget.Toast;  **import** androidx.activity.EdgeToEdge; **import** androidx.annotation.NonNull; **import** androidx.appcompat.app.AppCompatActivity; **import** androidx.camera.core.AspectRatio; **import** androidx.camera.core.Camera; **import** androidx.camera.core.CameraControl; **import** androidx.camera.core.CameraInfo; **import** androidx.camera.core.CameraSelector; **import** androidx.camera.core.ImageCapture; **import** androidx.camera.core.ImageCaptureException; **import** androidx.camera.core.Preview; **import** androidx.camera.lifecycle.ProcessCameraProvider; **import** androidx.camera.view.PreviewView; **import** androidx.core.app.ActivityCompat; **import** androidx.core.content.ContextCompat; **import** androidx.core.graphics.Insets; **import** androidx.core.view.ViewCompat; **import** androidx.core.view.WindowInsetsCompat;  **import** com.google.common.util.concurrent.ListenableFuture;  **import** java.util.Objects; **import** java.util.concurrent.ExecutionException;  **import** work.ngangiang.camera.Services.FileTransferTask; |
| --- |

**Biến thành viên**

| **private** **static** **final** String TAG = **"CameraxActivity"**; **private** **static** **final** **int** REQUEST\_PERMISSIONE = 100; **private** PreviewView previewView; **private** ImageCapture imageCapture; **private** CameraControl cameraControl; **private** CameraInfo cameraInfo; **private** ImageButton btnCapture, btnFlash, btnSend, btnClose; **private** **int** currentAspectRatio = AspectRatio.RATIO\_4\_3; **private** ProcessCameraProvider cameraProvider; **private** Uri imageUri; |
| --- |

* TAG**:** Chuỗi hằng số dùng để log thông tin trong lớp này.
* REQUEST\_PERMISSIONE: Mã yêu cầu dùng khi yêu cầu các quyền truy cập từ người dùng.
* previewView: Thành phần UI để hiển thị hình ảnh từ camera.
* imageCapture: Đối tượng dùng để chụp ảnh từ camera.
* cameraControl: Đối tượng dùng để điều khiển các chức năng của camera như zoom, focus, flash.
* cameraInfo: Cung cấp thông tin về trạng thái của camera.
* Các nút điều khiển trong giao diện người dùng, bao gồm:
  + btnCapture: Nút để chụp ảnh.
  + btnFlash: Nút để bật/tắt flash.
  + btnSend: Nút để gửi ảnh sau khi chụp.
  + btnClose: Nút để đóng hoặc hủy bỏ thao tác.
* currentAspectRatio: Tỷ lệ khung hình hiện tại của ảnh được chụp. Mặc định là 4:3.
* cameraProvider: Đối tượng cung cấp các thành phần camera để liên kết với lifecycle của activity hoặc fragment.
* imageUri: URI của ảnh đã được chụp và lưu trữ.

**Phương thức**

**1. onCreate(Bundle savedInstanceState)**

* Gọi phương thức **super.onCreate()** để kế thừa cấu trúc và hành vi cơ bản từ AppCompatActivity.
* Sử dụng EdgeToEdge để hiển thị UI tràn viền.
* Gọi **setContentView(R.layout.activity\_camerax)** để thiết lập giao diện người dùng từ file XML.
* Thiết lập xử lý insets cho giao diện để đảm bảo các phần tử UI không bị che khuất bởi hệ thống.
* Gọi **initViews()** để khởi tạo các view.
* Gọi **requestAppPermissions()** để yêu cầu các quyền cần thiết.
* Sử dụng **ProcessCameraProvider** để khởi tạo và liên kết camera với vòng đời của activity.
* Đăng ký các sự kiện click cho các nút điều khiển.

| @Override **protected** **void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  EdgeToEdge.enable(**this**);  setContentView(R.layout.activity\_camerax);   ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {  Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());  v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);  **return** insets;  });   initViews();  requestAppPermissions();   ListenableFuture<ProcessCameraProvider> cameraProviderFuture = ProcessCameraProvider.getInstance(**this**);  cameraProviderFuture.addListener(() -> {  **try** {  cameraProvider = cameraProviderFuture.get();  bindPreview();  } **catch** (ExecutionException | InterruptedException e) {  Log.e(TAG, **"Failed to get camera provider"**, e);  }  }, ContextCompat.getMainExecutor(**this**));   btnCapture.setOnClickListener(v -> takePhoto());  btnFlash.setOnClickListener(v -> toggleFlash());  btnClose.setOnClickListener(v -> {  bindPreview();  btnCapture.setVisibility(Button.VISIBLE);  btnSend.setVisibility(Button.GONE);  btnClose.setVisibility(Button.GONE);  findViewById(R.id.linearLayout).setVisibility(LinearLayout.VISIBLE);  });  btnSend.setOnClickListener(v -> {  **if** (imageUri != **null**) {  **new** FileTransferTask(CameraxActivity.**this**).execute(imageUri);  bindPreview();  btnCapture.setVisibility(Button.VISIBLE);  btnSend.setVisibility(Button.GONE);  btnClose.setVisibility(Button.GONE);  findViewById(R.id.linearLayout).setVisibility(LinearLayout.VISIBLE);  }  });   StrictMode.ThreadPolicy policy = **new** StrictMode.ThreadPolicy.Builder().permitAll().build();  StrictMode.setThreadPolicy(policy); } |
| --- |

**2. initViews()**

* Khởi tạo các thành phần UI từ layout XML và gán chúng vào các biến thành viên.
* Thiết lập các sự kiện click cho nút **btnAspectRatio3\_4** và **btnAspectRatio9\_16** để thay đổi tỷ lệ khung hình.

| **private** **void** initViews() {  previewView = findViewById(R.id.viewFinder);  btnCapture = findViewById(R.id.btnCapture);  btnFlash = findViewById(R.id.btnFlash);  btnSend = findViewById(R.id.btnSend);  btnClose = findViewById(R.id.btnClose);   findViewById(R.id.btnAspectRatio3\_4).setOnClickListener(v -> setAspectRatio(AspectRatio.RATIO\_4\_3));  findViewById(R.id.btnAspectRatio9\_16).setOnClickListener(v -> setAspectRatio(AspectRatio.RATIO\_16\_9)); } |
| --- |

**3. requestAppPermissions()**

Kiểm tra phiên bản Android và yêu cầu các quyền truy cập (đọc/ghi bộ nhớ và truy cập camera) nếu chưa được cấp.

| **private** **void** requestAppPermissions() {  **if** (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.LOLLIPOP &&  (!hasReadPermissions() || !hasWritePermissions() || !hasCameraPermissions())) {  ActivityCompat.requestPermissions(**this**,  **new** String[]{  Manifest.permission.READ\_EXTERNAL\_STORAGE,  Manifest.permission.WRITE\_EXTERNAL\_STORAGE,  Manifest.permission.CAMERA  }, REQUEST\_PERMISSIONE);  } } |
| --- |

**4. hasCameraPermissions()**

Kiểm tra quyền truy cập camera.

| **private** **boolean** hasCameraPermissions() {  **return** ActivityCompat.checkSelfPermission(**this**, Manifest.permission.CAMERA) == PackageManager.PERMISSION\_GRANTED; } |
| --- |

**5. hasReadPermissions()**

Kiểm tra quyền đọc bộ nhớ ngoài.

| **private** **boolean** hasReadPermissions() {  **return** ContextCompat.checkSelfPermission(getBaseContext(), Manifest.permission.READ\_EXTERNAL\_STORAGE) == PackageManager.PERMISSION\_GRANTED; } |
| --- |

6. **hasWritePermissions()**

Kiểm tra quyền ghi bộ nhớ ngoài.

| **private** **boolean** hasWritePermissions() {  **return** ContextCompat.checkSelfPermission(getBaseContext(), Manifest.permission.WRITE\_EXTERNAL\_STORAGE) == PackageManager.PERMISSION\_GRANTED; } |
| --- |

7. **bindPreview()**

* Khởi tạo và cấu hình đối tượng **Preview** để hiển thị preview từ camera.
* Khởi tạo **CameraSelector** để chọn camera phía sau.
* Khởi tạo **ImageCapture** để hỗ trợ chụp ảnh với tỷ lệ khung hình được chọn.
* Liên kết các thành phần với vòng đời của activity qua **cameraProvider.bindToLifecycle**.
* Lưu thông tin và điều khiển của camera vào các biến **cameraControl** và **cameraInfo**.
* Kích hoạt tính năng **pinch-to-zoom** thông qua **enablePinchToZoom()**.

| **private** **void** bindPreview() {  **if** (cameraProvider == **null**) **return**;   Preview preview = **new** Preview.Builder()  .setTargetAspectRatio(currentAspectRatio)  .build();  preview.setSurfaceProvider(previewView.getSurfaceProvider());   CameraSelector cameraSelector = **new** CameraSelector.Builder()  .requireLensFacing(CameraSelector.LENS\_FACING\_BACK)  .build();   imageCapture = **new** ImageCapture.Builder()  .setTargetAspectRatio(currentAspectRatio)  .build();   cameraProvider.unbindAll();  Camera camera = cameraProvider.bindToLifecycle(**this**, cameraSelector, preview, imageCapture);  cameraControl = camera.getCameraControl();  cameraInfo = camera.getCameraInfo();   enablePinchToZoom(); } |
| --- |

**8. setAspectRatio(int aspectRatio)**

Cập nhật tỷ lệ khung hình cho ảnh và gọi lại bindPreview() để áp dụng thay đổi.

| **private** **void** setAspectRatio(**int** aspectRatio) {  currentAspectRatio = aspectRatio;  bindPreview(); } |
| --- |

**9. enablePinchToZoom()**

* Sử dụng **ScaleGestureDetector** để phát hiện cử chỉ **pinch-to-zoom** từ người dùng.
* Điều chỉnh tỷ lệ zoom của camera thông qua **cameraControl.setZoomRatio.**
* Thiết lập **PreviewView** để nhận các cử chỉ touch và kích hoạt zoom khi phát hiện cử chỉ pinch.

| @SuppressLint(**"ClickableViewAccessibility"**) **private** **void** enablePinchToZoom() {  ScaleGestureDetector scaleGestureDetector = **new** ScaleGestureDetector(**this**, **new** ScaleGestureDetector.SimpleOnScaleGestureListener() {  @Override  **public** **boolean** onScale(@NonNull ScaleGestureDetector detector) {  **float** currentZoomRatio = Objects.requireNonNull(cameraInfo.getZoomState().getValue()).getZoomRatio();  **float** delta = detector.getScaleFactor();  cameraControl.setZoomRatio(currentZoomRatio \* delta);  **return** **true**;  }  });   previewView.setScaleType(PreviewView.ScaleType.FIT\_CENTER);   previewView.setOnTouchListener((v, event) -> {  scaleGestureDetector.onTouchEvent(event);  **return** **true**;  }); } |
| --- |

**10. takePhoto()**

* Tạo ContentValues để lưu trữ thông tin về file ảnh, bao gồm tên file và định dạng MIME.
* Khởi tạo ImageCapture.OutputFileOptions để xác định nơi lưu ảnh.
* Sử dụng imageCapture.takePicture() để chụp và lưu ảnh, xử lý callback OnImageSavedCallback để nhận thông báo khi ảnh được lưu thành công hoặc gặp lỗi.
* Nếu ảnh được lưu thành công, URI của ảnh được lưu vào imageUri và gọi viewSend() để chuẩn bị giao diện gửi ảnh.
* Trong trường hợp lỗi, log lỗi và hiển thị thông báo cho người dùng.

| **private** **void** takePhoto() {  ContentValues contentValues = **new** ContentValues();  contentValues.put(MediaStore.MediaColumns.DISPLAY\_NAME, **"photo\_"** + System.currentTimeMillis() + **".jpg"**);  contentValues.put(MediaStore.MediaColumns.MIME\_TYPE, **"image/jpeg"**);  contentValues.put(MediaStore.Images.Media.RELATIVE\_PATH, Environment.DIRECTORY\_PICTURES + **"/CameraX"**);   ImageCapture.OutputFileOptions outputOptions =  **new** ImageCapture.OutputFileOptions.Builder(  getContentResolver(),  MediaStore.Images.Media.EXTERNAL\_CONTENT\_URI,  contentValues  ).build();   imageCapture.takePicture(outputOptions, ContextCompat.getMainExecutor(**this**), **new** ImageCapture.OnImageSavedCallback() {  @Override  **public** **void** onImageSaved(@NonNull ImageCapture.OutputFileResults outputFileResults) {  imageUri = outputFileResults.getSavedUri();   **if** (imageUri != **null**) {  String msg = **"Photo capture succeeded: "** + imageUri.toString();  Log.d(TAG, msg);  } **else** {  Log.e(TAG, **"Photo capture succeeded but URI is null"**);  }   **if** (cameraProvider != **null**) {  cameraProvider.unbindAll();  }  viewSend();  }   @Override  **public** **void** onError(@NonNull ImageCaptureException exception) {  String msg = **"Photo capture failed: "** + exception.getMessage();  Log.e(TAG, msg);  Toast.makeText(getApplicationContext(), msg, Toast.LENGTH\_SHORT).show();  }  }); } |
| --- |

**11. toggleFlash()**

* Chuyển đổi giữa chế độ bật và tắt flash. Nếu flash đang tắt, chuyển sang bật và ngược lại.
* Cập nhật biểu tượng của btnFlash để phản ánh trạng thái hiện tại của flash.

| **private** **void** toggleFlash() {  **int** flashMode = imageCapture.getFlashMode() == ImageCapture.FLASH\_MODE\_OFF  ? ImageCapture.FLASH\_MODE\_ON  : ImageCapture.FLASH\_MODE\_OFF;   imageCapture.setFlashMode(flashMode);   btnFlash.setImageResource(flashMode == ImageCapture.FLASH\_MODE\_ON  ? R.drawable.ic\_flash\_on  : R.drawable.ic\_flash\_off); } |
| --- |

**12. viewSend()**

* Ẩn nút btnCapture và hiện các nút btnSend và btnClose.
* Ẩn thanh điều khiển tỷ lệ khung hình.
* Đặt lại biểu tượng của btnFlash về trạng thái tắt.

| **private** **void** viewSend() {  btnCapture.setVisibility(Button.GONE);  btnSend.setVisibility(Button.VISIBLE);  btnClose.setVisibility(Button.VISIBLE);  findViewById(R.id.linearLayout).setVisibility(LinearLayout.GONE);   btnFlash.setImageResource(R.drawable.ic\_flash\_off); } |
| --- |

## 4.2. Server

### 4.2.1. Giới Thiệu

Lớp FileServer là một ứng dụng Java đơn giản hoạt động như một máy chủ (server) để nhận và lưu trữ các file gửi từ client qua mạng. Lớp này sử dụng socket để giao tiếp với client, nhận tên file và dữ liệu file, sau đó lưu trữ file vào thư mục Local.

### 4.2.2. Cấu Trúc Lớp

Lớp FileServer bao gồm các phần chính sau:

1. **Khai Báo Thư Viện**: Các thư viện cần thiết để thực hiện các chức năng I/O và mạng.
2. **Định Nghĩa Cổng và Thư Mục**: Định nghĩa cổng để lắng nghe kết nối và thư mục lưu trữ file nhận được.
3. **Phương Thức main**: Điểm vào chính của chương trình, nơi khởi tạo server và xử lý kết nối client.
4. **Phương Thức getFileName**: Đọc tên file từ client.

#### 4.2.2.1. Chi Tiết Mã Nguồn

**Khai Báo Thư Viện**

* **java.io.\***: Cung cấp các lớp để thực hiện các hoạt động đầu vào/đầu ra (I/O), bao gồm đọc và ghi dữ liệu từ và đến các luồng (streams).
* **java.net.ServerSocket**: Tạo server socket để lắng nghe kết nối từ client.
* **java.net.Socket**: Đại diện cho kết nối socket giữa client và server.
* **java.net.InetAddress**: Cung cấp phương thức để lấy địa chỉ IP của máy tính.
* **java.net.UnknownHostException**: Ngoại lệ được ném ra khi không thể xác định địa chỉ IP của máy.

**Định Nghĩa Cổng và Thư Mục**

* **Cổng**: Lớp FileServer sử dụng cổng 100 để lắng nghe kết nối từ client. Cổng này được định nghĩa bằng hằng số PORT trong lớp.
* **Thư Mục**: Thư mục để lưu trữ các file nhận được từ client được đặt tên là FileReceived. Nếu thư mục này không tồn tại, chương trình sẽ tạo nó.

**Phương Thức main**

Phương thức main là điểm vào chính của chương trình. Nó thực hiện các bước sau:

1. Hiển Thị Đường Dẫn Thư Mục Làm Việc Hiện Tại:
2. Tạo Thư Mục FileReceived Nếu Nó Không Tồn Tại:
   * Thư mục này sẽ chứa các file nhận được từ client. Nếu không thể tạo thư mục, chương trình sẽ dừng lại.
3. Lấy Địa Chỉ IP Của Máy Tính:
   * Địa chỉ IP này sẽ được in ra màn hình và có thể được sử dụng bởi client để kết nối với server.
4. Khởi Tạo ServerSocket Để Lắng Nghe Kết Nối:
   * Server sẽ lắng nghe các kết nối đến cổng 100. Khi có kết nối từ client, server sẽ chấp nhận kết nối và xử lý việc nhận dữ liệu file từ client.

**Phương Thức getFileName**

Phương thức này đọc tên file từ BufferedReader và trả về tên file. Tên file này sẽ được sử dụng để lưu dữ liệu file vào đĩa.

**Xử Lý Kết Nối Client**

Khi có kết nối từ client, server thực hiện các bước sau:

1. **Nhận Tên File Từ Client**:
   * Sử dụng BufferedReader để đọc tên file được gửi từ client.
2. **Nhận Dữ Liệu File**:
   * Dữ liệu file được đọc từ InputStream và ghi vào file mới trong thư mục FileReceived.
3. **Đóng Kết Nối Với Client**:
   * Sau khi hoàn thành việc nhận file, server đóng kết nối với client.

#### 4.2.2.2. Chi tiết mã nguồn

**FileServer.java**

| **import** java.io.\*; **import** java.net.ServerSocket; **import** java.net.Socket;  **import** java.net.InetAddress; **import** java.net.UnknownHostException;  **public** **class** FileServer {  **private** **static** **final** **int** PORT = 100;   **public** **static** **void** main(String[] args) {  System.out.println(**"Thư mục làm việc hiện tại: "** + **new** File(**"."**).getAbsolutePath());  File receivedDir = **new** File(**"FileReceived"**);  **if** (!receivedDir.exists()) {  **if** (receivedDir.mkdirs()) {  System.out.println(**"Thư mục FileReceived đã được tạo."**);  } **else** {  System.err.println(**"Không thể tạo thư mục FileReceived."**);  **return**;  }  }   **try** {  InetAddress ip = InetAddress.getLocalHost();  System.out.println(**"Địa chỉ IP của máy: "** + ip.getHostAddress() + **":"** + PORT);  } **catch** (UnknownHostException e) {  System.err.println(**"Không thể lấy địa chỉ IP: "** + e.getMessage());  }   **try** (ServerSocket serverSocket = **new** ServerSocket(PORT)) {   **while** (**true**) {  Socket clientSocket = serverSocket.accept();  System.out.println(**"Đã kết nối với client: "** + clientSocket.getInetAddress().getHostAddress());   **try** (BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(clientSocket.getInputStream()));  InputStream inputStream = clientSocket.getInputStream();  FileOutputStream fileOutputStream = **new** FileOutputStream(**new** File(receivedDir, getFileName(reader)))) {  **byte**[] buffer = **new** **byte**[4096];  **int** bytesRead;  **while** ((bytesRead = inputStream.read(buffer)) != -1) {  fileOutputStream.write(buffer, 0, bytesRead);  }  System.out.println(**"Ảnh đã được nhận thành công và lưu vào thư mục FileReceived!"**);  } **catch** (Exception e) {  System.err.println(**"Lỗi trong khi nhận ảnh: "** + e.getMessage());  }  clientSocket.close();  }  } **catch** (Exception e) {  System.err.println(**"Lỗi khi khởi động server: "** + e.getMessage());  }  }   **private** **static** String getFileName(BufferedReader reader) **throws** IOException {  **return** reader.readLine();  } } |
| --- |

#### 4.2.2.3. Chạy Chương Trình

1. **Biên Dịch Lớp**: Sử dụng lệnh javac FileServer.java để biên dịch lớp.
2. **Chạy Server**: Sử dụng lệnh java FileServer để khởi chạy server.
3. **Kết Nối Client**: Đảm bảo client kết nối đến địa chỉ IP và cổng (xxx.xxx.xxx.xxx:100) mà server đang lắng nghe (cổng 100).