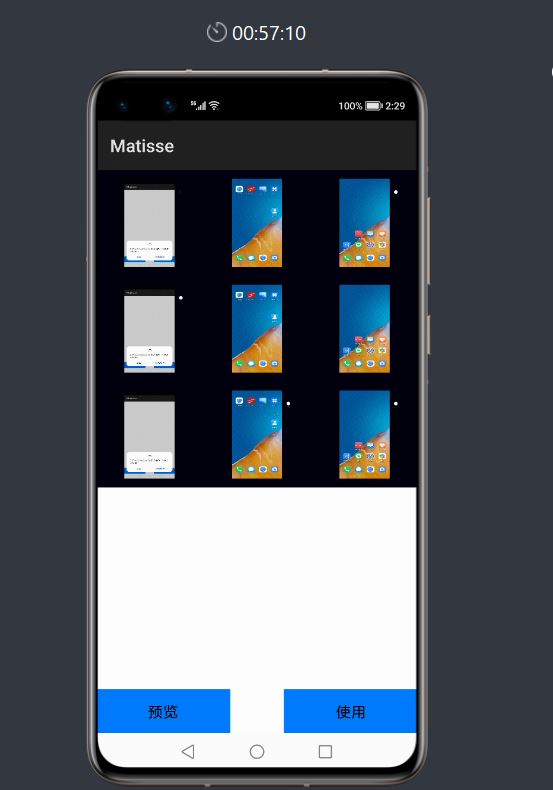
# Matisse组件功能介绍

## 功能介绍：

Matisse组件是一个图片选择框架，实现图片的选择及使用相机进行拍摄，最后将选择或拍摄的照片显示在ability中（目前初版还未实现所有功能）。

## 模拟器上运行效果：



# Matisse使用方法

## 新建工程，增加组件Har包依赖

在应用模块中添加HAR，只需要将Matisse.har复制到entry\libs目录下即可（由于build.gradle中已经依赖的libs目录下的\*.har，因此不需要在做修改）。

## 修改主页面代码成为请求方

在MainAbilitySlice中构造Intent以及包含Action的Operation对象，并调用startAbilityForResult()方法发起请求。然后重写onAbilityResult()回调方法，对请求结果进行处理。

private void pickPicture() {

Intent intent = new Intent();

Operation operation = new Intent.OperationBuilder()

.withAction("action.pick\_pic")

.withBundleName("com.example.matisse")

.withAbilityName("com.example.matisse.MatisseAbility")

.build();

intent.setOperation(operation);

startAbilityForResult(intent, REQ\_CODE\_PICK\_PIC);

}

@Override

protected void onAbilityResult(int requestCode, int resultCode, Intent resultData) {

switch (requestCode) {

case REQ\_CODE\_PICK\_PIC:

// Do something with result.

...

return;

default:

...

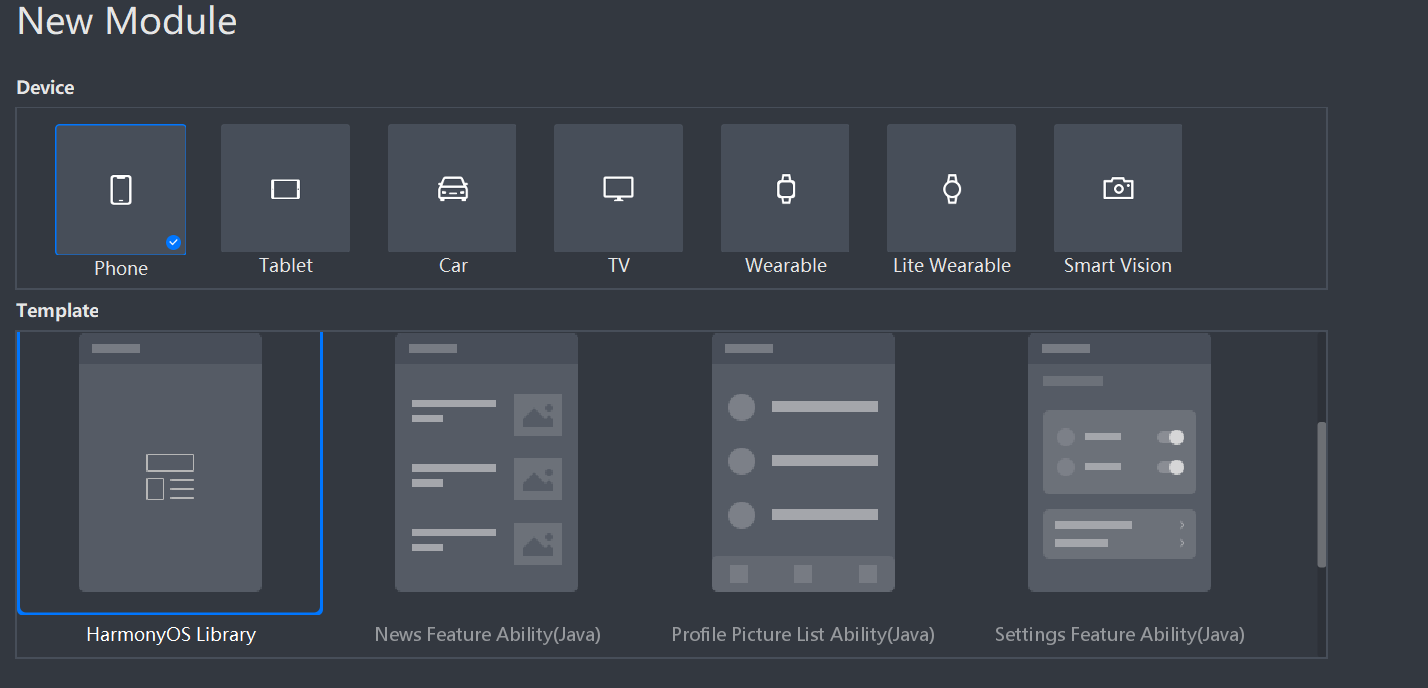
}

}

# Matisse开发实现

## 新建一个Module

新建一个Module，类型选择HarmonyOS Library，模块名为Matisse，如图



## 获得媒体库所有图片

新建一个MatisseAbility 在其slice的onStart方法中申请权限并配制config.json：

@Override  
public void onStart(Intent intent) {  
 super.onStart(intent);  
 super.setUIContent(ResourceTable.*Layout\_ability\_matisse*);  
 String[] per={"ohos.permission.READ\_USER\_STORAGE"};  
 requestPermissionsFromUser(per,0);  
  
}

为了获得媒体库所有图片（的URI），需要定义一个DataAbilityHelper，并设置resultSet，代码如下：

DataAbilityHelper helper = DataAbilityHelper.*creator*(this);  
try {  
 ResultSet resultSet = helper.query(AVStorage.Images.Media.*EXTERNAL\_DATA\_ABILITY\_URI*, null, null);  
 while (resultSet != null && resultSet.goToNextRow()) {  
 int mid = resultSet.getInt(resultSet.getColumnIndexForName(AVStorage.Images.Media.*ID*));  
 Uri uri = Uri.*appendEncodedPathToUri*(AVStorage.Images.Media.*EXTERNAL\_DATA\_ABILITY\_URI*, "" + mid);  
 FileDescriptor fileDescriptor = null;  
 fileDescriptor = helper.openFile(uri, "r");  
 ImageSource.DecodingOptions decodingOptions = new ImageSource.DecodingOptions();  
 decodingOptions.desiredSize = new Size(180, 320);  
 ImageSource imageSource = ImageSource.*create*(fileDescriptor, null);  
 PixelMap pixelMap = imageSource.createThumbnailPixelmap(decodingOptions, true);  
 uris.add(new UriModel(uri, pixelMap));  
  
 }  
} catch (DataAbilityRemoteException | FileNotFoundException e) {  
 e.printStackTrace();  
}

由于Listcontainer目前不支持grid布局，所以取得所有图片的uri后对结果进行分组：

if(uris.size()>3) {  
 List<List<UriModel>> newList = arraySplitUtil.splistList(uris, 3);  
  
 for (List<UriModel> uriModel : newList) {  
 nestedLists.add(new NestedList(uriModel));  
 }  
 }else {  
 nestedLists.add(new NestedList(uris));  
 }

## 新建Listconainer并使用嵌套方式实现相册视图：

listContainer = (ListContainer) findComponentById(ResourceTable.*Id\_galley*);  
UltimateProvider ultimateProvider=new UltimateProvider(nestedLists,getContext());  
listContainer.setItemProvider(ultimateProvider);

这里使用了三方件UltimateProvider减少代码量编写。

## 点击事件和回调处理

使用DependentLayout在图片布局右上方加上checkbox， 代码中id\_icon为图片:

checkbox=(Checkbox)findComponentById(ResourceTable.*Id\_checked*);  
 checkbox.setChecked(false);  
 icon.setClickedListener(new Component.ClickedListener() {  
 @Override  
 public void onClick(Component component) {  
 checkbox.toggle();  
 // getModel().getUri()  
  
 }  
 });  
  
checkbox.setCheckedStateChangedListener((view, state) -> {  
 if (state) {  
 MatisseAbilitySlice.*selectedSet*.add(getModel());  
 }else {  
 MatisseAbilitySlice.*selectedSet*.remove(getModel());  
 }  
});

回调：在config.json中增加"action.pick\_pic"并在MatisseAbility中添加如下路由：

addActionRoute("action.pick\_pic", MatisseAbilitySlice.class.getName());

新建使用按钮，并增加如下代码：

use.setClickedListener(new Component.ClickedListener() {  
 @Override  
 public void onClick(Component component) {  
 if (*selectedSet*.size()==0){  
 return;  
 }  
 else {  
 Intent resultIntent = new Intent();  
 resultIntent.setParam("model", (Serializable) *selectedSet*);  
 // Intent resultIntent = new Intent();  
 setResult(resultIntent);  
 }  
 }  
});

## 编译HAR包

利用Gradle可以将HarmonyOS Library库模块构建为HAR包，构建HAR包的方法如下：

在Gradle构建任务中，双击PackageDebugHar或PackageReleaseHar任务，构建Debug类型或Release类型的HAR。

待构建任务完成后，可以在工程目录中的Matisse> bulid > outputs > har目录中，获取生成的HAR包。

