**ПРИЛОЖЕНИЕ Б**

**(обязательное)**

**Листинг программного кода**

package dev.logarithmus.p2pdroid.ui.activity

import android.annotation.SuppressLint

import android.app.Activity

import android.app.AlertDialog

import android.bluetooth.BluetoothAdapter

import android.content.ActivityNotFoundException

import android.content.Context

import android.content.Intent

import android.graphics.Bitmap

import android.graphics.drawable.Drawable

import android.os.Bundle

import android.os.Handler

import androidx.constraintlayout.widget.ConstraintLayout

import androidx.coordinatorlayout.widget.CoordinatorLayout

import androidx.cardview.widget.CardView

import androidx.recyclerview.widget.LinearLayoutManager

import androidx.recyclerview.widget.RecyclerView

import android.view.Menu

import android.view.MenuItem

import android.view.View

import android.view.ViewGroup

import android.view.animation.AnimationUtils

import android.widget.\*

import dev.logarithmus.p2pdroid.R

import dev.logarithmus.p2pdroid.ui.adapter.ChatAdapter

import dev.logarithmus.p2pdroid.ui.presenter.ChatPresenter

import dev.logarithmus.p2pdroid.ui.util.ScrollAwareBehavior

import dev.logarithmus.p2pdroid.ui.util.SimpleTextWatcher

import dev.logarithmus.p2pdroid.ui.view.ChatView

import dev.logarithmus.p2pdroid.ui.view.NotificationView

import dev.logarithmus.p2pdroid.ui.viewmodel.ChatMessageViewModel

import dev.logarithmus.p2pdroid.ui.widget.ActionView

import dev.logarithmus.p2pdroid.ui.widget.GoDownButton

import dev.logarithmus.p2pdroid.utils.\*

import com.squareup.picasso.Picasso

import com.squareup.picasso.Target

import com.timehop.stickyheadersrecyclerview.StickyRecyclerHeadersDecoration

import org.koin.android.ext.android.inject

import org.koin.core.parameter.parametersOf

import pl.aprilapps.easyphotopicker.DefaultCallback

import pl.aprilapps.easyphotopicker.EasyImage

import java.io.File

import java.lang.Exception

import java.util.\*

class ChatActivity : SkeletonActivity(), ChatView {

private val deviceAddress: String? by argument(EXTRA\_ADDRESS)

private val presenter: ChatPresenter by inject {

parametersOf(deviceAddress ?: "", this)

}

private val chatContainer: ConstraintLayout by bind(R.id.cl\_chat\_container)

private val actions: ActionView by bind(R.id.av\_actions)

private val chatList: RecyclerView by bind(R.id.rv\_chat)

private val messageField: EditText by bind(R.id.et\_message)

private val sendButtonsSwitcher: ViewSwitcher by bind(R.id.vs\_send\_buttons)

private val textSendingHolder: ViewGroup by bind(R.id.ll\_text\_sending\_holder)

private val imageSendingHolder: ViewGroup by bind(R.id.ll\_image\_sending\_holder)

private val transferringImagePreview: ImageView by bind(R.id.iv\_transferring\_image)

private val transferringImageSize: TextView by bind(R.id.tv\_file\_size)

private val transferringImageHeader: TextView by bind(R.id.tv\_sending\_image\_label)

private val transferringImageProgressLabel: TextView by bind(R.id.tv\_file\_sending\_percentage)

private val transferringImageProgressBar: ProgressBar by bind(R.id.pb\_transferring\_progress)

private val presharingContainer: CardView by bind(R.id.cv\_presharing\_image\_holder)

private val presharingImage: ImageView by bind(R.id.iv\_presharing\_image)

private val goDownButton: GoDownButton by bind(R.id.gdb\_go\_down)

private val chatLayoutManager = LinearLayoutManager(this).apply {

reverseLayout = true

}

private lateinit var scrollBehavior: ScrollAwareBehavior

private lateinit var chatAdapter: ChatAdapter

private val showAnimation by lazy {

AnimationUtils.loadAnimation(this, R.anim.anime\_fade\_slide\_in)

}

private val hideAnimation by lazy {

AnimationUtils.loadAnimation(this, R.anim.anime\_fade\_slide\_out).apply {

onEnd { presharingContainer.visibility = View.GONE }

}

}

private var disconnectedDialog: AlertDialog? = null

private var lostConnectionDialog: AlertDialog? = null

private val textWatcher = object : SimpleTextWatcher() {

private var previousText: String? = null

override fun afterTextChanged(text: String) {

if (previousText.isNullOrEmpty() && text.isNotEmpty()) {

sendButtonsSwitcher.showNext()

} else if (!previousText.isNullOrEmpty() && text.isEmpty()) {

sendButtonsSwitcher.showPrevious()

}

previousText = text

}

}

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_chat, ActivityType.CHILD\_ACTIVITY)

lifecycle.addObserver(presenter)

title = if (deviceAddress.isNullOrEmpty()) getString(R.string.app\_name) else deviceAddress

toolbar?.let {

it.subtitle = getString(R.string.chat\_\_not\_connected)

it.setTitleTextAppearance(this, R.style.ActionBar\_TitleTextStyle)

it.setSubtitleTextAppearance(this, R.style.ActionBar\_SubTitleTextStyle)

}

messageField.addTextChangedListener(textWatcher)

findViewById<ImageButton>(R.id.ib\_send).setOnClickListener {

presenter.sendMessage(messageField.text.toString().trim())

}

findViewById<ImageButton>(R.id.ib\_image).setOnClickListener {

presenter.performFilePicking()

}

findViewById<ImageButton>(R.id.ib\_cancel).setOnClickListener {

presenter.cancelFileTransfer()

}

findViewById<Button>(R.id.btn\_retry).setOnClickListener {

presharingContainer.startAnimation(hideAnimation)

presenter.proceedPresharing()

}

findViewById<Button>(R.id.btn\_cancel).setOnClickListener {

presharingContainer.startAnimation(hideAnimation)

presenter.cancelPresharing()

}

goDownButton.setOnClickListener {

chatLayoutManager.scrollToPosition(0)

scrollBehavior.hideChild()

goDownButton.setUnreadMessageNumber(0)

}

scrollBehavior = ScrollAwareBehavior(this).apply {

onHideListener = { goDownButton.setUnreadMessageNumber(0) }

}

val params = goDownButton.layoutParams as CoordinatorLayout.LayoutParams

params.behavior = scrollBehavior

goDownButton.requestLayout()

chatAdapter = ChatAdapter().apply {

imageClickListener = { view, message ->

ImagePreviewActivity.start(this@ChatActivity, view, message)

}

}

chatList.apply {

adapter = chatAdapter

layoutManager = chatLayoutManager

addItemDecoration(StickyRecyclerHeadersDecoration(chatAdapter))

addOnScrollListener(object : RecyclerView.OnScrollListener() {

override fun onScrollStateChanged(recyclerView: RecyclerView, scrollState: Int) {

val picasso = Picasso.get()

if (scrollState == RecyclerView.SCROLL\_STATE\_IDLE || scrollState == RecyclerView.SCROLL\_STATE\_DRAGGING) {

picasso.resumeTag(chatAdapter.picassoTag)

} else {

picasso.pauseTag(chatAdapter.picassoTag)

}

}

})

}

presenter.onViewCreated()

if (Intent.ACTION\_SEND == intent.action) {

val textToShare = intent.getStringExtra(EXTRA\_MESSAGE)

val fileToShare = intent.getStringExtra(EXTRA\_FILE\_PATH)

if (textToShare != null) {

messageField.setText(textToShare)

} else if (fileToShare != null) {

//FIXME

Handler().postDelayed({

presenter.sendFile(File(fileToShare))

}, 1000)

}

intent.action = Intent.ACTION\_VIEW

}

}

override fun setBackgroundColor(color: Int) {

chatContainer.setBackgroundColor(color)

}

override fun dismissMessageNotification() {

getNotificationManager()

.cancel(NotificationView.NOTIFICATION\_TAG\_MESSAGE, NotificationView.NOTIFICATION\_ID\_MESSAGE)

}

override fun showPartnerName(name: String, device: String) {

title = "$name ($device)"

}

override fun showStatusConnected() {

toolbar?.subtitle = getString(R.string.chat\_\_connected)

}

override fun showStatusNotConnected() {

toolbar?.subtitle = getString(R.string.chat\_\_not\_connected)

}

override fun showStatusPending() {

toolbar?.subtitle = getString(R.string.chat\_\_pending)

}

override fun showNotConnectedToSend() =

Toast.makeText(this, getString(R.string.chat\_\_not\_connected\_to\_send), Toast.LENGTH\_LONG).show()

override fun afterMessageSent() {

messageField.text = null

}

override fun showNotConnectedToThisDevice(currentDevice: String) {

actions.setActionsAndShow(getString(R.string.chat\_\_connected\_to\_another, currentDevice),

ActionView.Action(getString(R.string.chat\_\_connect)) { presenter.connectToDevice() },

null

)

}

override fun showNotConnectedToAnyDevice() {

actions.setActionsAndShow(getString(R.string.chat\_\_not\_connected\_to\_this\_device),

ActionView.Action(getString(R.string.chat\_\_connect)) { presenter.connectToDevice() },

null

)

}

override fun showWainingForOpponent() {

actions.setActionsAndShow(getString(R.string.chat\_\_waiting\_for\_device),

ActionView.Action(getString(R.string.general\_\_cancel)) { presenter.resetConnection() },

null

)

}

override fun showConnectionRequest(displayName: String, deviceName: String) {

actions.setActionsAndShow(getString(R.string.chat\_\_connection\_request, displayName, deviceName),

ActionView.Action(getString(R.string.general\_\_start\_chat)) { presenter.acceptConnection() },

ActionView.Action(getString(R.string.chat\_\_disconnect)) { presenter.rejectConnection() }

)

}

override fun showServiceDestroyed() = doIfStarted {

AlertDialog.Builder(this)

.setMessage(getString(R.string.general\_\_service\_lost))

.setPositiveButton(getString(R.string.general\_\_restart)) { \_, \_ -> presenter.prepareConnection() }

.setCancelable(false)

.show()

}

override fun hideActions() {

actions.visibility = View.GONE

}

override fun showMessagesHistory(messages: List<ChatMessageViewModel>) {

chatAdapter.messages = LinkedList(messages)

chatAdapter.notifyDataSetChanged()

}

override fun showReceivedMessage(message: ChatMessageViewModel) {

chatAdapter.messages.addFirst(message)

chatAdapter.notifyItemInserted(0)

if (!scrollBehavior.isChildShown()) {

chatLayoutManager.scrollToPosition(0)

} else {

goDownButton.setUnreadMessageNumber(goDownButton.getUnreadMessageNumber() + 1)

}

}

override fun showSentMessage(message: ChatMessageViewModel) {

chatAdapter.messages.addFirst(message)

chatAdapter.notifyItemInserted(0)

chatLayoutManager.scrollToPosition(0)

}

override fun showSendingMessageFailure() {

Toast.makeText(this, R.string.chat\_\_sending\_failed, Toast.LENGTH\_LONG).show()

}

override fun showRejectedConnection() = doIfStarted {

AlertDialog.Builder(this)

.setMessage(getString(R.string.chat\_\_connection\_rejected))

.setPositiveButton(getString(R.string.general\_\_ok), null)

.show()

}

override fun showBluetoothDisabled() {

actions.setActionsAndShow(getString(R.string.chat\_\_bluetooth\_is\_disabled),

ActionView.Action(getString(R.string.chat\_\_enable)) { presenter.enableBluetooth() },

null

)

}

override fun showLostConnection() = doIfStarted {

lostConnectionDialog = AlertDialog.Builder(this)

.setMessage(getString(R.string.chat\_\_connection\_lost))

.setPositiveButton(getString(R.string.chat\_\_reconnect)) { \_, \_ -> presenter.reconnect() }

.setNegativeButton(getString(R.string.general\_\_cancel), null)

.show()

}

override fun hideLostConnection() {

lostConnectionDialog?.dismiss()

lostConnectionDialog = null

}

override fun showDisconnected() = doIfStarted {

disconnectedDialog = AlertDialog.Builder(this)

.setMessage(getString(R.string.chat\_\_partner\_disconnected))

.setPositiveButton(getString(R.string.chat\_\_reconnect)) { \_, \_ -> presenter.reconnect() }

.setNegativeButton(getString(R.string.general\_\_cancel), null)

.show()

}

override fun hideDisconnected() {

disconnectedDialog?.dismiss()

disconnectedDialog = null

}

override fun showFailedConnection() = doIfStarted {

AlertDialog.Builder(this)

.setMessage(getString(R.string.chat\_\_unable\_to\_connect))

.setPositiveButton(getString(R.string.general\_\_try\_again)) { \_, \_ -> presenter.connectToDevice() }

.setNegativeButton(getString(R.string.general\_\_cancel), null)

.show()

}

override fun showNotValidMessage() {

Toast.makeText(this, getString(R.string.chat\_\_message\_cannot\_be\_empty), Toast.LENGTH\_SHORT).show()

}

override fun showDeviceIsNotAvailable() = doIfStarted {

AlertDialog.Builder(this)

.setMessage(getString(R.string.chat\_\_device\_is\_not\_available))

.setPositiveButton(getString(R.string.chat\_\_rescan)) { \_, \_ -> ScanActivity.start(this) }

.show()

}

override fun requestBluetoothEnabling() {

val enableBtIntent = Intent(BluetoothAdapter.ACTION\_REQUEST\_ENABLE)

try {

startActivityForResult(enableBtIntent, REQUEST\_ENABLE\_BLUETOOTH)

} catch (e: ActivityNotFoundException) {

showUnableToActivateBluetoothMessage()

}

}

private fun showUnableToActivateBluetoothMessage() = doIfStarted {

AlertDialog.Builder(this)

.setMessage(R.string.scan\_\_unable\_to\_activate)

.setPositiveButton(R.string.general\_\_ok, null)

.show()

}

override fun showBluetoothEnablingFailed() = doIfStarted {

AlertDialog.Builder(this)

.setMessage(getString(R.string.chat\_\_bluetooth\_required))

.setPositiveButton(getString(R.string.general\_\_ok), null)

.show()

}

override fun showBluetoothEnablingFailed() = doIfStarted {

AlertDialog.Builder(this)

.setMessage(getString(R.string.chat\_\_bluetooth\_required))

.setPositiveButton(getString(R.string.general\_\_ok), null)

.show()

}

override fun showImageTooBig(maxSize: Long) = doIfStarted {

AlertDialog.Builder(this)

.setMessage(getString(R.string.chat\_\_too\_big\_image, maxSize.toReadableFileSize()))

.setPositiveButton(getString(R.string.general\_\_ok), null)

.show()

}

override fun showImageNotExist() {

Toast.makeText(this, R.string.chat\_\_file\_not\_exist, Toast.LENGTH\_LONG).show()

}

override fun showPresharingImage(path: String) {

presharingContainer.visibility = View.VISIBLE

presharingContainer.startAnimation(showAnimation)

Picasso.get()

.load("file://$path")

.centerCrop()

.fit()

.into(presharingImage)

}

override fun openImagePicker() {

EasyImage.openChooserWithGallery(this, getString(R.string.chat\_\_choose\_image), 0)

}

override fun showImageTransferLayout(fileAddress: String?, fileSize: Long, transferType: ChatView.FileTransferType) {

textSendingHolder.visibility = View.GONE

imageSendingHolder.visibility = View.VISIBLE

transferringImageHeader.text = getString(if (transferType == ChatView.FileTransferType.SENDING)

R.string.chat\_\_sending\_image else R.string.chat\_\_receiving\_images)

Picasso.get()

.load("file://$fileAddress")

.into(object : Target {

override fun onPrepareLoad(placeHolderDrawable: Drawable?) {

transferringImagePreview.setImageResource(R.drawable.ic\_photo)

}

override fun onBitmapFailed(e: Exception?, errorDrawable: Drawable?) {

transferringImagePreview.setImageResource(R.drawable.ic\_photo)

}

override fun onBitmapLoaded(bitmap: Bitmap?, from: Picasso.LoadedFrom?) {

transferringImagePreview.setImageBitmap(bitmap)

}

})

transferringImageSize.text = fileSize.toReadableFileSize()

transferringImageProgressLabel.text = "0%"

//FIXME should work with Long

transferringImageProgressBar.progress = 0

transferringImageProgressBar.max = fileSize.toInt()

}

@SuppressLint("SetTextI18n")

override fun updateImageTransferProgress(transferredBytes: Long, totalBytes: Long) {

val percents = transferredBytes.toFloat() / totalBytes \* 100

transferringImageProgressLabel.text = "${Math.round(percents)}%"

//FIXME should work with Long

transferringImageProgressBar.progress = transferredBytes.toInt()

}

override fun hideImageTransferLayout() {

textSendingHolder.visibility = View.VISIBLE

imageSendingHolder.visibility = View.GONE

}

override fun showImageTransferCanceled() {

Toast.makeText(this, R.string.chat\_\_partner\_canceled\_image\_transfer, Toast.LENGTH\_LONG).sh

ow()

}

override fun showImageTransferFailure() {

Toast.makeText(this, R.string.chat\_\_problem\_during\_file\_transfer, Toast.LENGTH\_LONG).show(

)

}

override fun showReceiverUnableToReceiveImages() = doIfStarted {

AlertDialog.Builder(this)

.setMessage(R.string.chat\_\_partner\_unable\_to\_receive\_images)

.setPositiveButton(R.string.general\_\_ok, null)

.show()

}

override fun onCreateOptionsMenu(menu: Menu): Boolean {

menuInflater.inflate(R.menu.menu\_chat, menu)

return super.onCreateOptionsMenu(menu)

}

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {

super.onActivityResult(requestCode, resultCode, data)

if (requestCode == REQUEST\_ENABLE\_BLUETOOTH) {

if (resultCode == Activity.RESULT\_OK) {

presenter.onBluetoothEnabled()

} else {

presenter.onBluetoothEnablingFailed()

}

} else {

EasyImage.handleActivityResult(requestCode, resultCode, data, this, object : DefaultCallback() {

override fun onImagesPicked(imageFiles: MutableList<File>, source: EasyImage.ImageSource?, type: Int) {

if (imageFiles.isNotEmpty()) {

presenter.sendFile(imageFiles[0])

}

}

override fun onImagePickerError(e: Exception?, source: EasyImage.ImageSource?, type: Int) {

Toast.makeText(this@ChatActivity, R.string.chat\_\_unable\_to\_pick\_image, Toast.L

ENGTH\_LONG).show()

}

override fun onCanceled(source: EasyImage.ImageSource?, type: Int) {

if (source == EasyImage.ImageSource.CAMERA\_IMAGE) {

EasyImage

.lastlyTakenButCanceledPhoto(this@ChatActivity)

?.delete()

}

}

})

}

}

override fun onOptionsItemSelected(item: MenuItem): Boolean {

return when (item.itemId) {

R.id.action\_images -> {

ReceivedImagesActivity.start(this, deviceAddress)

true

}

R.id.action\_disconnect -> {

presenter.disconnect()

true

}

else -> super.onOptionsItemSelected(item)

}

}

}