```
    implementation 'androidx.core:core-ktx:1.7.0'
    implementation 'androidx.appcompat:appcompat:1.5.1'
    implementation 'com.google.android.material:material:1.7.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.4'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.0'

//асинхрон
    implementation "androidx.lifecycle:lifecycle-runtime-ktx:2.5.1"

//для пост гет запросов
    implementation 'com.squareup.retrofit2:retrofit:2.9.0'
    implementation 'com.squareup.moshi:moshi:1.13.0'
    implementation("com.squareup.retrofit2:converter-moshi:2.4.0")
    kapt("com.squareup.moshi:moshi-kotlin-codegen:1.13.0")
    implementation("com.squareup.moshi:moshi-kotlin:1.9.1")

//база данных
    implementation 'androidx.room:room-runtime:2.4.3'
    implementation 'androidx.room:room-ktx:2.4.3'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    kapt 'androidx.room:room-compiler:2.4.3'
    implementation 'androidx.room:room-rxjava2:2.4.3'
    implementation 'com.google.android.gms:play-services-maps:17.0.1'

//картинки с url
    implementation ("com.github.bumptech.glide:glide:4.11.0") {
        exclude group: "com.android.support"
    }
```

```
id 'kotlin-kapt'
```

```
viewBinding.enabled = true
```

SPLASH SCREEN

АктивитиОтдельное

```
class SplashScreen : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        val intent = Intent(this, Main::class.java)
        startActivity(intent)
        finish()
    }
}
```

в манифесте
внутри созданого активити

```
android:exported="true"
android:theme="@style/SplashhScreen">
<intent-filter>
    <action android:name="android.intent.action.MAIN" />

    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

в values

splash_screen.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="SplashhScreen"
parent="Theme.MaterialComponents.DayNight.NoActionBar.Bridge">
        <item name="android:windowBackground">@drawable/splash_screen</item>
    </style>
</resources>
```

Манифест

Вверху

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.INTERNET"/>
```

Внутри

```
android:name=".App" //Если будет бд
android:usesCleartextTraffic="true"
```

байдинги

вверху

```
private lateinit var binding: ActivityMainBinding
```

внутри

```
binding = "название лайута".inflate(layoutInflater)
setContentView(binding.root)
```

интент

```
val intent= Intent(this,"Название активити"::class.java)
startActivity(intent)
finish()
```

JSON

```
@JsonClass(generateAdapter = true)
data class Fellings(
    @Json(name = "success")
    val success: Boolean,
    @Json(name = "data")
    val data: List<FData>,
)

@JsonClass(generateAdapter = true)
data class FData(
    @Json(name = "id")
    val id: Int,
    @Json(name = "title")
    val title: String,
    @Json(name = "image")
    val image: String,
    @Json(name = "position")
    val position: Int,
)
```

GET POST запросы

```
data = ApiInterface.quFelApi.feelings().body()!!.data
```

```
const val API_URL = "http://mskko2021.mad.hakta.pro/api/"

object ApiInterface {
    private val retrofit = Retrofit.Builder()
        .baseUrl(API_URL)
        .addConverterFactory(MoshiConverterFactory.create()).build()

    val userApi: UserApi = retrofit.create(UserApi::class.java)
    val quFelApi: QuFelApi = retrofit.create(QuFelApi::class.java)
}

interface UserApi {
    @POST("user/login")
    suspend fun login(@Body user: UserPost): Response<User>
}

interface QuFelApi {
    @GET("quotes")
    suspend fun quotes(): Response<Quotes>

    @GET("feelings")
    suspend fun feelings(): Response<Fellings>
}
```

```
val user = UserPost(binding.editTextTextPersonName.text.toString(),
binding.editTextTextPassword.text.toString())

lifecycleScope.launch {
    val result = ApiInterface.userApi.login(user)

    if (result.isSuccessful) {
        Log.d("asd1", result.body().toString())
    }
}
```

База данных

```
val ImgDao = (this.application as App).db.ImgDao() //обратиться к базе
```

```
@Database(entities = [ImagesBase::class], version = 1)
abstract class AppDataBase: RoomDatabase() {
    abstract fun ImgDao():ImgDao
}
```

```
@Dao
interface ImgDao {
    @Query("SELECT * FROM ImagesBase")
    suspend fun getAllImages(): List<ImagesBase>

    @Insert(entity = ImagesBase::class, onConflict =
OnConflictStrategy.IGNORE)
    suspend fun insertImage(img: ImagesBase)

    @Query("DELETE FROM ImagesBase WHERE image_id=:id")
    suspend fun deleteImage(id: Int)

    @Query("SELECT image_url FROM ImagesBase ORDER BY image_id DESC LIMIT 1")
    suspend fun getLastImage(): String
}
```

```
@Entity(tableName = "ImagesBase")
data class ImagesBase(
    @PrimaryKey(autoGenerate = true)
    val image_id: Int?,

    @ColumnInfo(name = "image_url")
    val image_url: String,
)
```

```
class App: Application() {
    lateinit var db: AppDataBase

    override fun onCreate() {
        super.onCreate()

        db = Room.databaseBuilder(
            applicationContext,
            AppDataBase::class.java,"db"
        ).build()
    }
}
```

Вставить картинку URL

```
Glide.with(this@"Название активити").load("url картинки").into("куда
(imageVIew)")
```

Кеш

```
val shared = getSharedPreferences("asd", AppCompatActivity.MODE_PRIVATE)
val editor = shared.edit()
editor.putInt("UserId", 1)
editor.apply()
shared.getString("константное значение", "знач если ничего не получил")
```

Свайпер

создаем лайоут

создаем скрипт адаптер

(заполнить)

```
binding.recil.adapter = FeelingsAdapter(data)
binding.recil.layoutManager = GridLayoutManager(this@Login, 2)
```

адаптер

```
class FeelingsAdapter(private val fellings: List<FData>) :
RecyclerView.Adapter<MyViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
MyViewHolder {
        val binding =
OneItemBinding.inflate(LayoutInflater.from(parent.context))
        return MyViewHolder(binding)
    }

    override fun onBindViewHolder(holder: MyViewHolder, position: Int) {
        val fellings = fellings[position]

        holder.binding.textView.text = fellings.title
    }

    override fun getItemCount(): Int {
        return fellings.size
    }
}

class MyViewHolder(val binding: OneItemBinding):
RecyclerView.ViewHolder(binding.root)
```

(есть orientation)

Чтобы добавить функции

При вызове адаптера

```
val adapter = AdapterFeeling(list,{it->click(it)},{it->onlong(it)})
```

в параметрах адаптера

```
val click:(Data)->Unit,val click1:(Data)->Unit
```

вызов функции

```
imgFealing.setOnClickListener {
    click(item)
}
```

```
textSize -> fontSIze
textAligment -> text aligin
bold -> text weight
textAllCaps -> все с больших
backgroundTint="#7C9A92" -> цвет кнопки

закруглить
CardView -> cardCorner
```

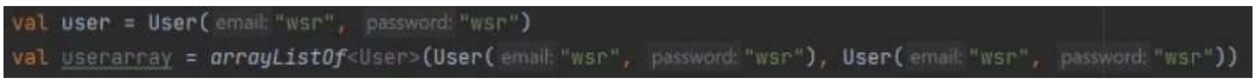
плагины

- json to kotlin
- kopilot

скрыть верхнюю плашку
res/values/themes/temes.xml

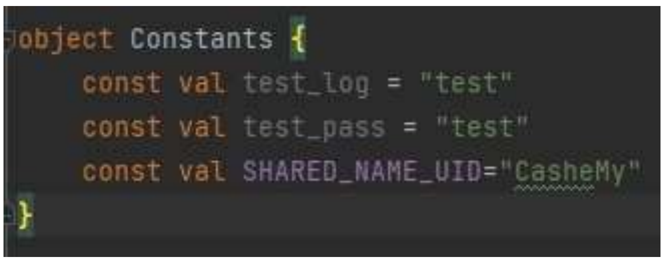
```
<style name="Theme.Podgotovka"
parent="Theme.MaterialComponents.DayNight.NoActionBar">
```

массив объектов



```
val user = User( email: "wsr",  password: "wsr")
val userarray = arrayListOf<User>(User( email: "wsr",  password: "wsr"), User( email: "wsr",  password: "wsr"))
```

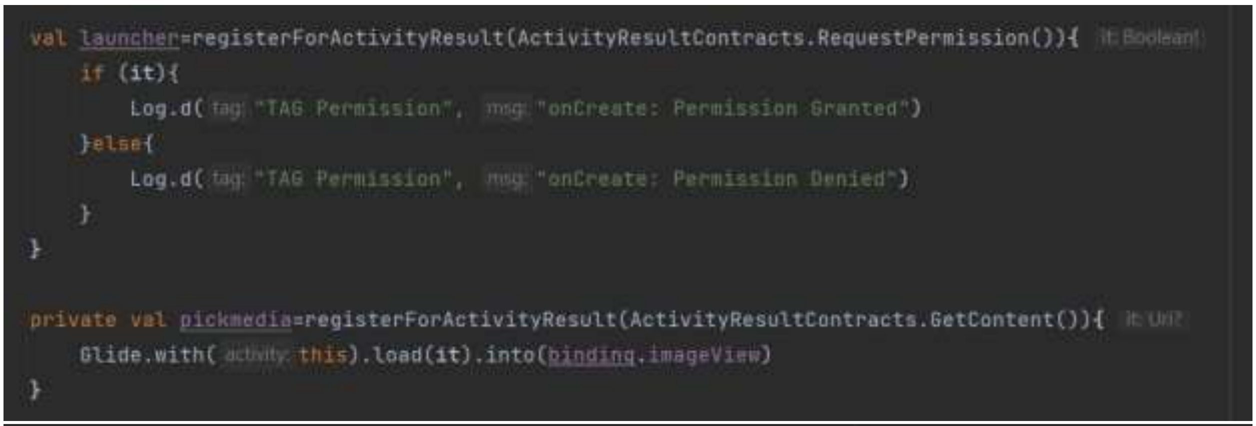
Константы



```
object Constants {
    const val test_log = "test"
    const val test_pass = "test"
    const val SHARED_NAME_UID="CasheMy"
}
```

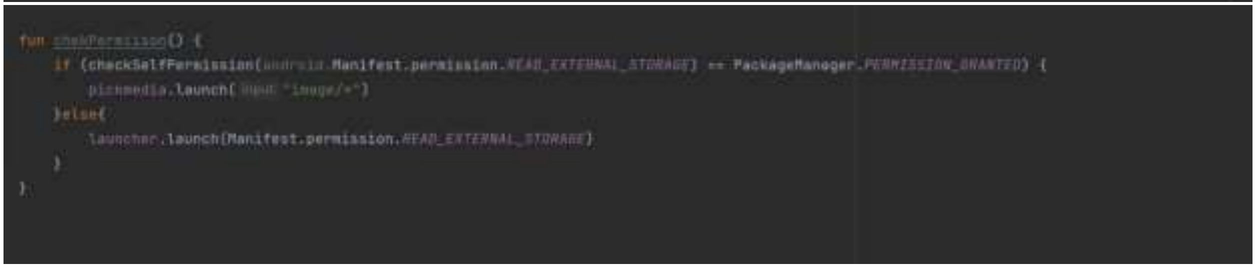
получить картинку из телефона

вызывать checkPermission



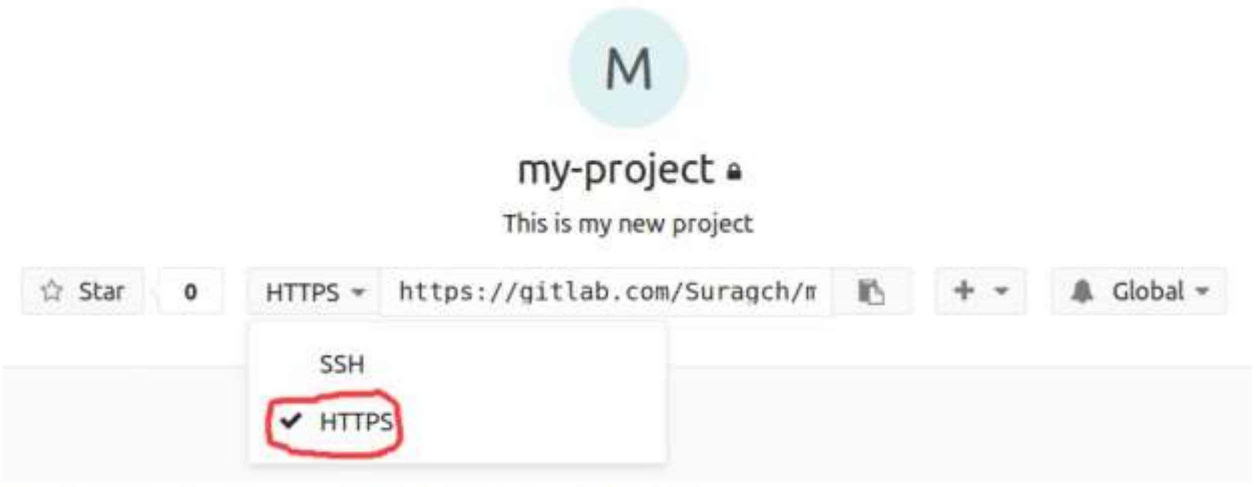
```kotlin
val launcher=registerForActivityResult(ActivityResultContracts.RequestPermission()){ it: Boolean!
    if (it){
        Log.d( tag: "TAG Permission",  msg: "onCreate: Permission Granted")
    }else{
        Log.d( tag: "TAG Permission",  msg: "onCreate: Permission Denied")
    }
}

private val pickmedia=registerForActivityResult(ActivityResultContracts.GetContent()){ it: Uri?
    Glide.with( activity: this).load(it).into(binding.imageView)
}
```



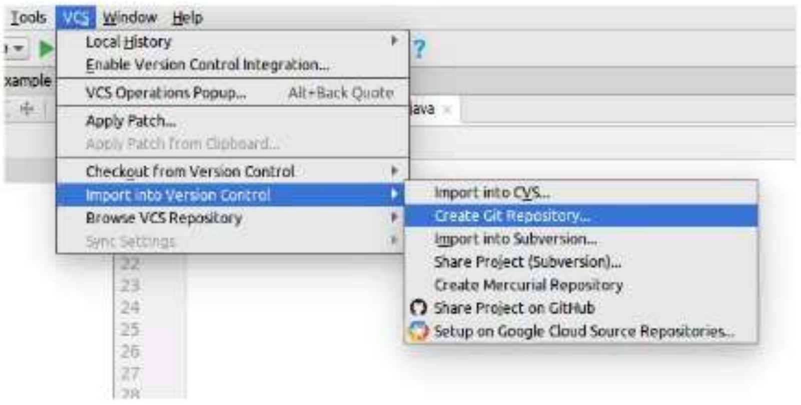
```kotlin
fun chekPermission() {
    if (checkSelfPermission(android.Manifest.permission.READ_EXTERNAL_STORAGE) == PackageManager.PERMISSION_GRANTED) {
        pickmedia.launch( input: "image/*")
    }else{
        launcher.launch(Manifest.permission.READ_EXTERNAL_STORAGE)
    }
}
```

git



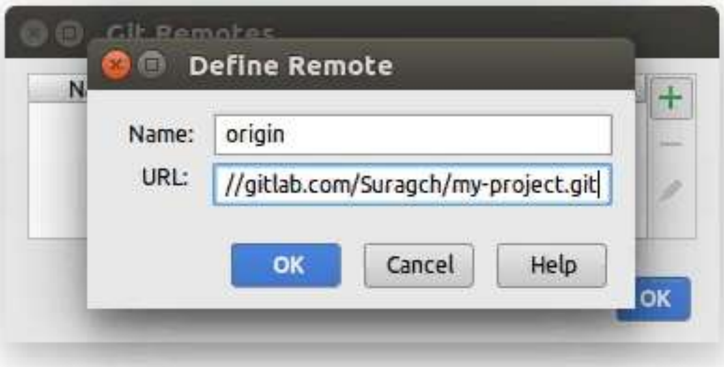
2. Create a Git repository in Android Studio

In the Android Studio menu go to **VCS** > **Import into Version Control** > **Create Git Repository...**


Tools
VCS
Window
Help
Local History
Enable Version Control Integration...
VCS Operations Popup...
Alt+Back Quote
Apply Patch...
Apply Patch from Clipboard...
Checkout from Version Control
Import into Version Control
Browse VCS Repository
Sync Settings
Import into CVS...
Create Git Repository...
Import into Subversion...
Share Project (Subversion)...
Create Mercurial Repository
Share Project on GitHub
Setup on Google Cloud Source Repositories...

3. Add remote

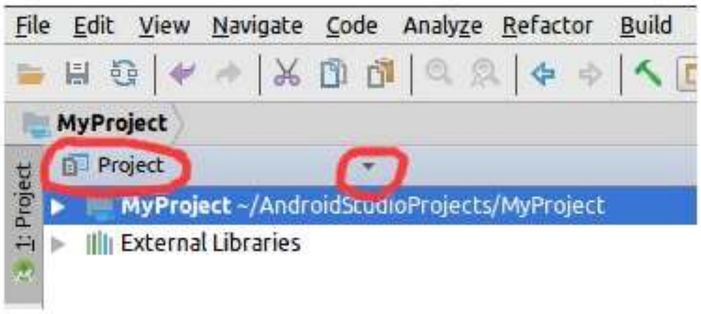
Go to **VCS** > **Git** > **Remotes....** Then paste in the https address you got from GitLab in step one.



Name: origin
URL: //gitlab.com/Suragch/my-project.git

4. Add, commit, and push your files

Make sure you have the top level of the project selected. If you are in the Android view you can switch it to the Project view.



- *Add:* Go to **VCS** > **Git** > **Add**.
- *Commit:* After adding, do **VCS** > **Git** > **Commit Directory**. (You will need to write a commit message, something like `initial commit`.)
- *Push:* Finally, go to **VCS** > **Git** > **Push**.

Иконка

res/mipmap -> удалить все
res/drawable -> удалить ic_launcer
res/mipmap -> правая кнопка мыши -> image asset -> finish
(в манифесте в icon менять, если не работает)

Шрифты

Создать папку font в res
загрузить шрифты и создать файл my_font.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<font-family xmlns:android="http://schemas.android.com/apk/res/android">
<font android:fontStyle="normal" android:fontWeight="400"
android:font="@font/alegreya_regular">
</font>

<font android:fontStyle="normal" android:fontWeight="600"
android:font="@font/alegreya_bold">
</font>

<font android:fontStyle="normal" android:fontWeight="800"
android:font="@font/alegreya_extrabold">
</font>

<font android:fontStyle="normal" android:fontWeight="500"
android:font="@font/alegreya_medium">
</font>
</font-family>
```

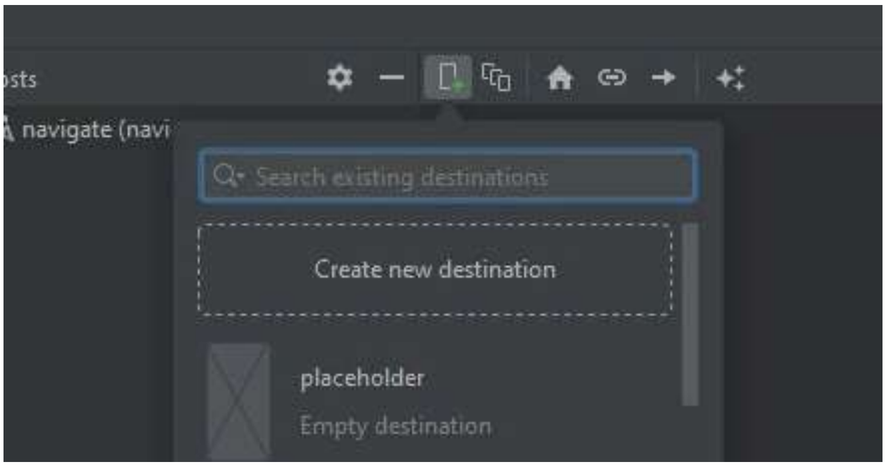
Потом в values/themes/themes.xml добавить

```xml
<item name="fontFamily">@font/my_font</item>
```

Bottom navigation

Создать фрагмент -> Fragment -> framgmetn with viewmodels

Создать файл в res -> new Resource file, type поменять на Navigation
в самом файле нажать на кнопку сверху



добавить id к фрагментам

Создать файл в res -> new Resource file, type поменять на Menu

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
<item
    android:id="@+id/blankFragment"
    android:icon="@drawable/logo"
    android:title="Меню" />
<item
    android:id="@+id/optionFragment"
    android:icon="@drawable/ic_baseline_360_24"
    android:title="Опции" />
<item
    android:id="@+id/joapFragment"
    android:icon="@drawable/logo"
    android:title="Жопа" />
</menu>
```

Далее куда мы хотим добавить навигацию
добавляем элемент include и вставляем navigation
и bottomNavigation (в него вставялем

```
app:menu="@menu/bottom_menu"
```

В коде активити вставить

```
val navHost=supportFragmentManager.findFragmentById(R.id."id navigation") as
NavHostFragment
val navController=navHost.navController
binding.bottomNavigationView.setupWithNavController(navController)
```

Модалка

Отдельный класс

```
class MyDialogFragment : DialogFragment() {
    override fun onCreateDialog(savedInstanceState: Bundle?): Dialog {
        val builder: AlertDialog.Builder =
            AlertDialog.Builder(activity).setTitle("Ваня")
                .setPositiveButton("Да"){
                        dialog,id->
                    parentFragmentManager.setFragmentResult("key",
bundleOf("key_bundle" to true))
                    dismiss()
                }.setNegativeButton("Нет"){
                        dialog,id->
                    parentFragmentManager.setFragmentResult("key",
bundleOf("key_bundle" to false))
                    dismiss()
                }

        return builder.create()
    }
}
```

вызов модалки

```
val dialogFragment=MyDialogFragment()
dialogFragment.show(childFragmentManager,"dialog")
childFragmentManager.setFragmentResultListener("key",this){
        it,bundle->
    val x= bundle.get("key_bundle")
    Log.d("TAG Dialog",x.toString())
}
```