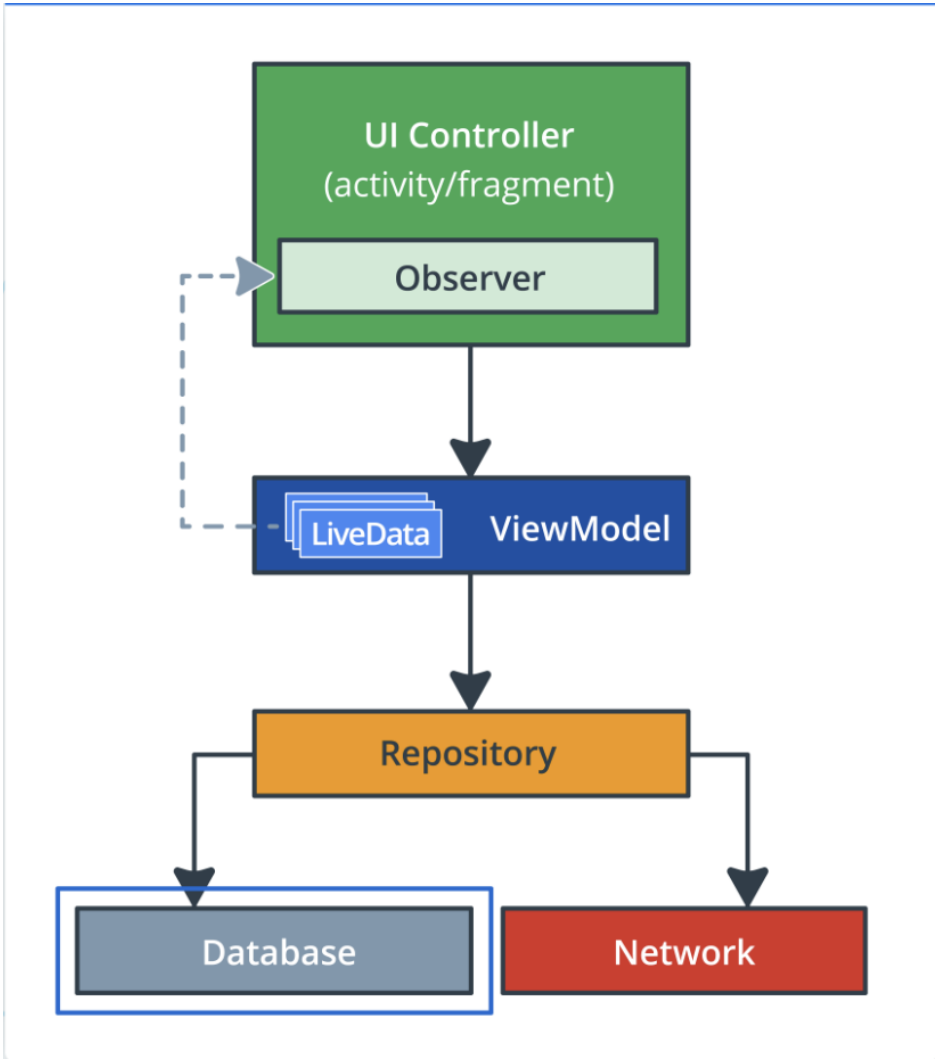
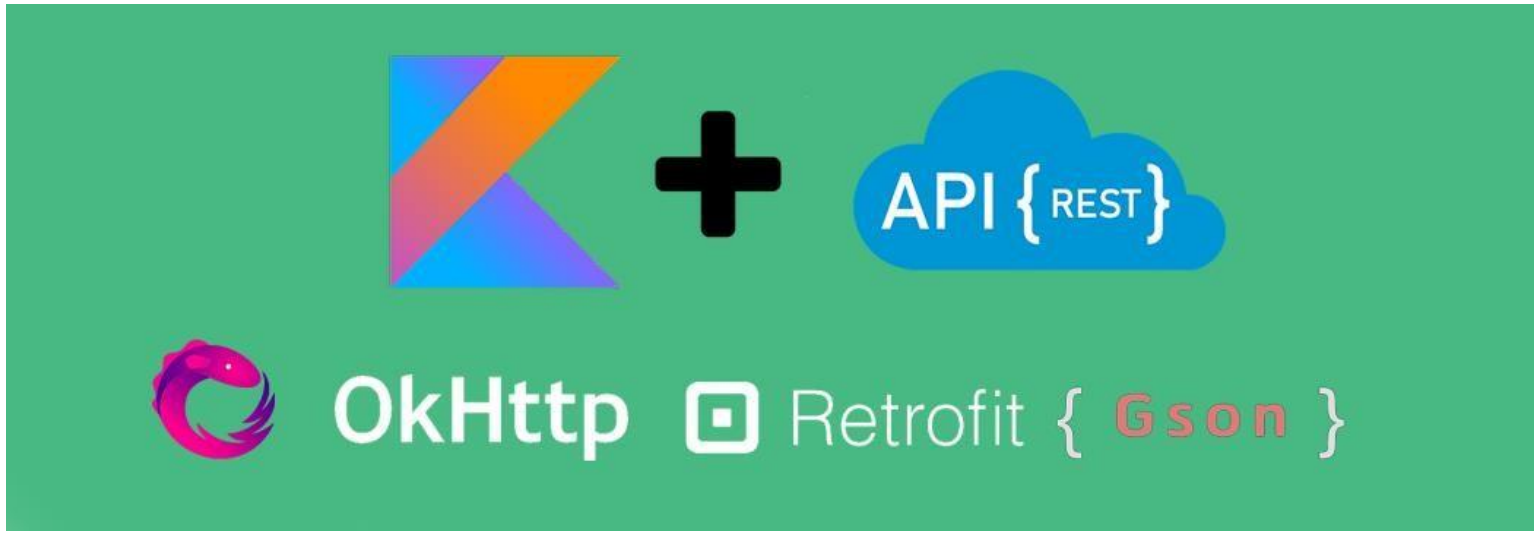
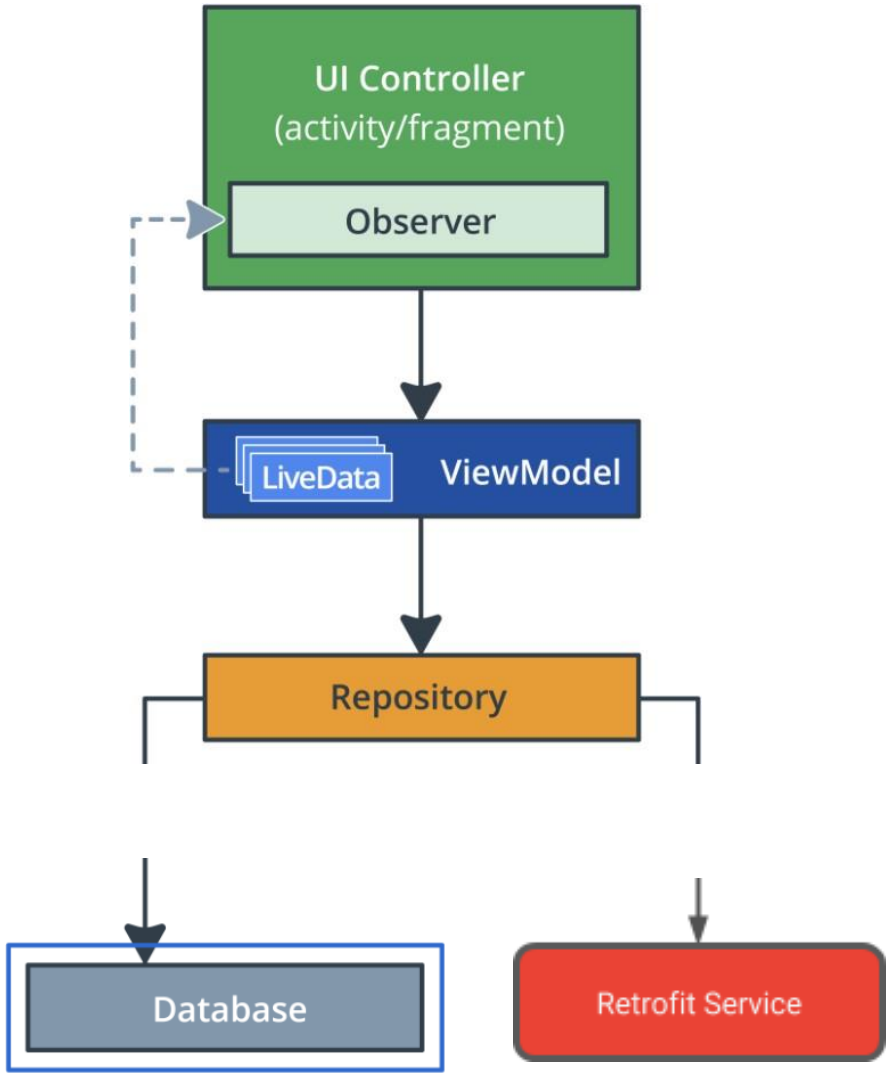
Získanie údajov z Internetu

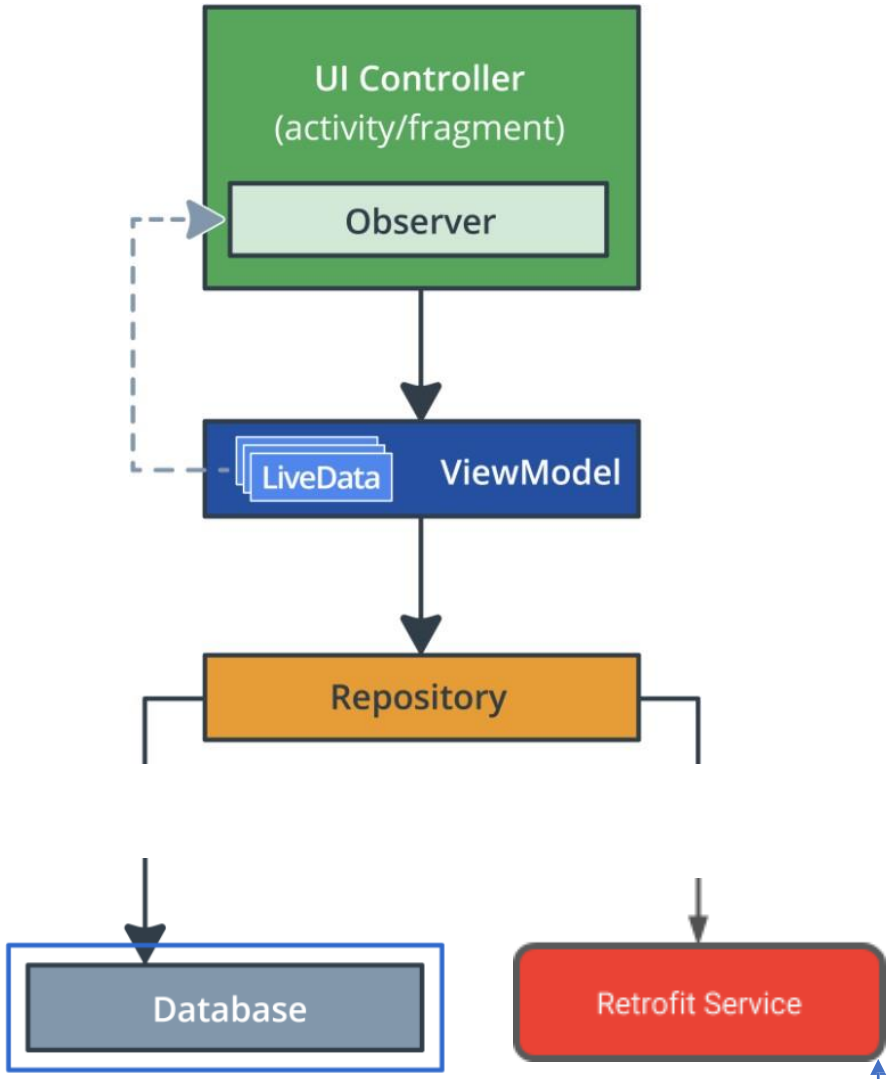


Získanie údajov z Internetu

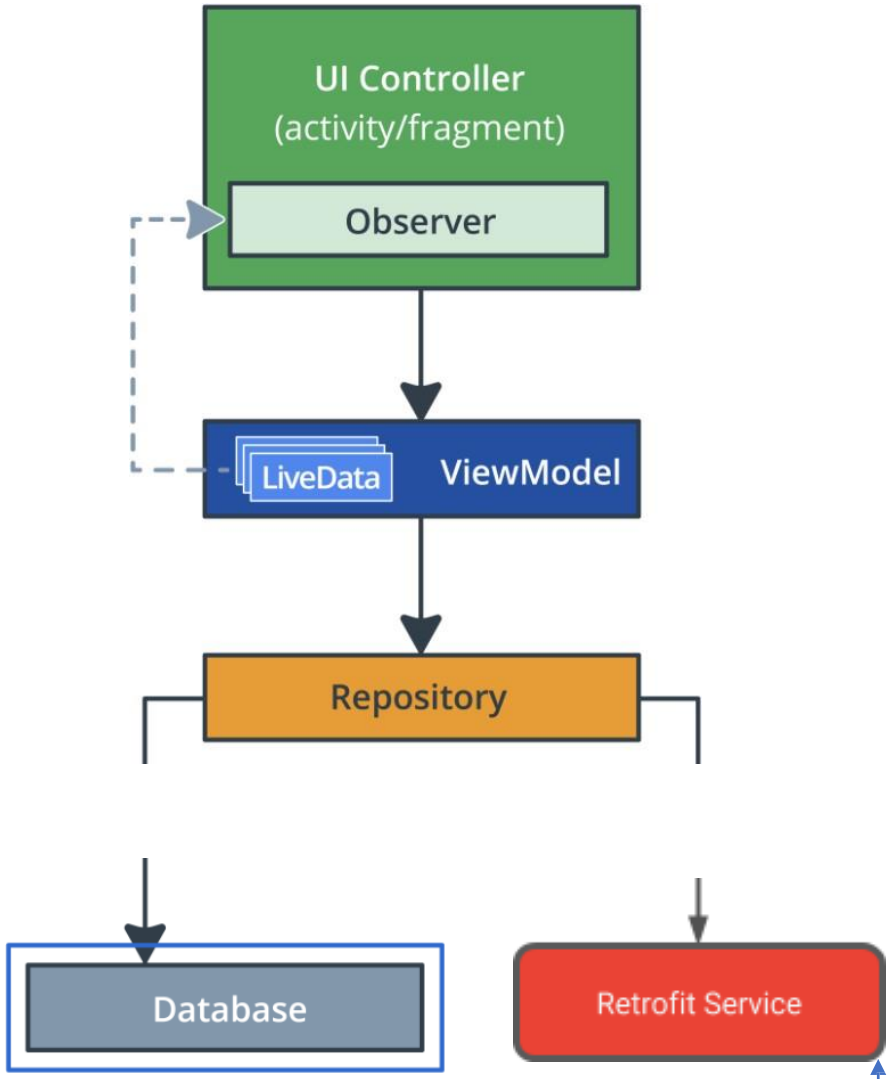
```
//Webservice
implementation 'com.squareup.retrofit2:retrofit:2.6.0'
implementation 'com.squareup.retrofit2:converter-gson:2.6.0'
implementation 'com.google.code.gson:gson:2.8.5'
```




UI Controller
(activity/fragment)
Observer
LiveData
ViewModel
Repository
Database
Retrofit Service

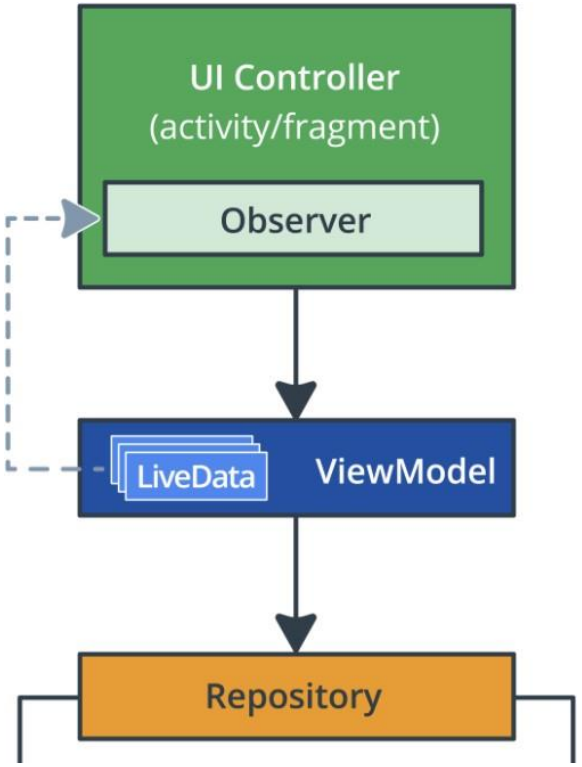



```
@GET("realestate")
    suspend fun getProperties():
Response<List<MarsResponse>>
```




```
@Insert(onConflict = OnConflictStrategy.REPLACE)
suspend fun insertImages(items:List<MarsItem>)

@Query("SELECT * FROM images")
fun getImages(): LiveData<List<MarsItem>>
```

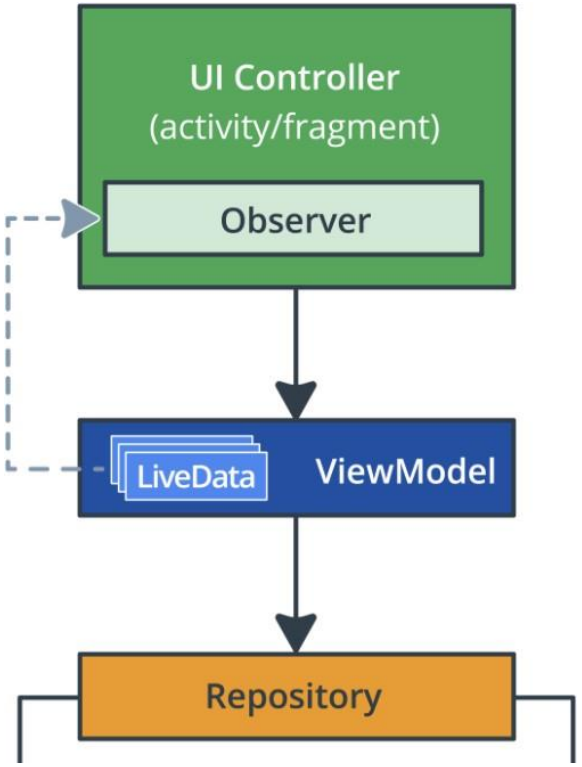
```
@GET("realestate")
    suspend fun getProperties():
Response<List<MarsResponse>>
```




```
suspend fun insertImages(items: List<MarsItem>)
{ dao.insertImages(items)}

fun getImages(): LiveData<List<MarsItem>> = dao.getImages()
```


```
@Insert(onConflict = OnConflictStrategy.REPLACE)
suspend fun insertImages(items:List<MarsItem>)

@Query("SELECT * FROM images")
fun getImages(): LiveData<List<MarsItem>>
```

```
@GET("realestate")
    suspend fun getProperties():
Response<List<MarsResponse>>
```

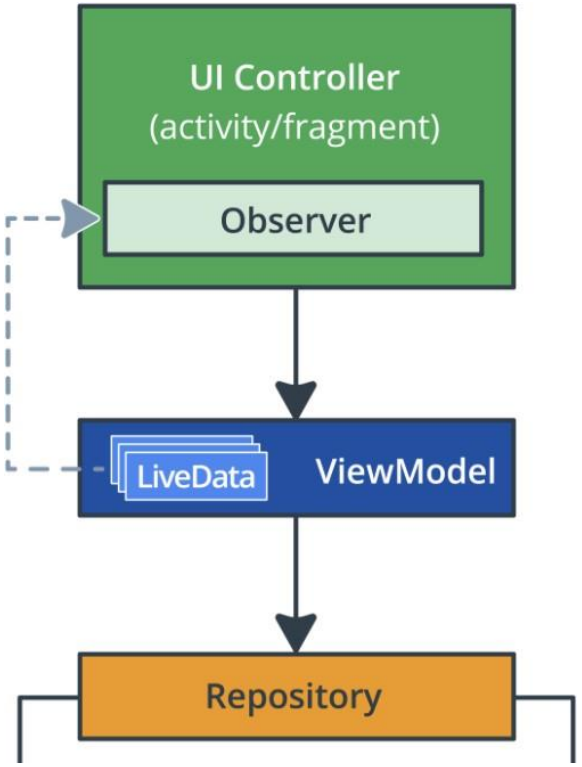



```
suspend fun insertImages(items: List<MarsItem>)
{ dao.insertImages(items)}

fun getImages(): LiveData<List<MarsItem>> = dao.getImages()
```


```
@Insert(onConflict = OnConflictStrategy.REPLACE)
suspend fun insertImages(items:List<MarsItem>)

@Query("SELECT * FROM images")
fun getImages(): LiveData<List<MarsItem>>
```

```
fun getMars(): LiveData<List<MarsItem>> = cache.getImages()
suspend fun loadMars(
        onError: (error: String) -> Unit)


{...
cache.insertImages(it.map { item ->
     MarsItem(item.price, item.id, item.type, item.img_src)
    })
 ...}
```

```
@GET("realestate")
    suspend fun getProperties():
Response<List<MarsResponse>>
```



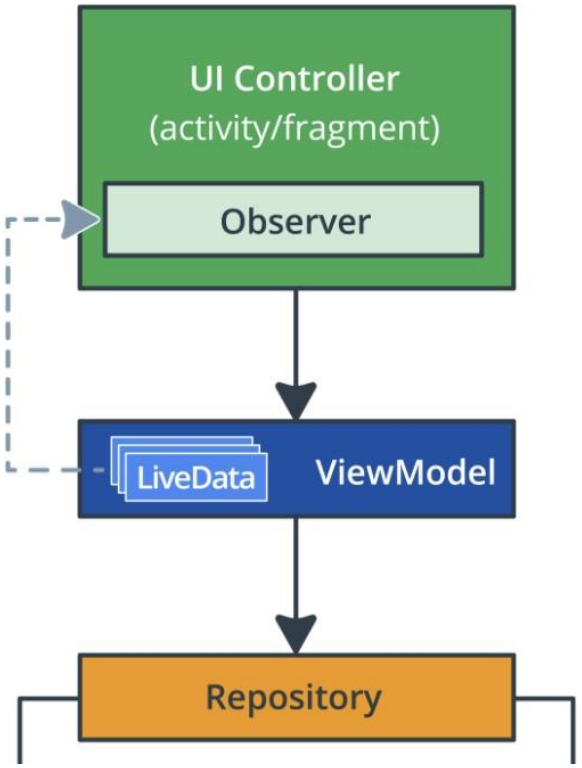

```
val images: LiveData<List<MarsItem>>
    get() = repository.getMars()

fun loadMars() {
    viewModelScope.launch {
        repository.loadMars { error.postValue(it) }
    }
}
```

```
fun getMars(): LiveData<List<MarsItem>> = cache.getImages()
suspend fun loadMars(
        onError: (error: String) -> Unit)

{...
cache.insertImages(it.map { item ->
     MarsItem(item.price, item.id, item.type, item.img_src)
    })
 ...}
```

```
suspend fun insertImages(items: List<MarsItem>)
{ dao.insertImages(items)}

fun getImages(): LiveData<List<MarsItem>> = dao.getImages()
```


```
@Insert(onConflict = OnConflictStrategy.REPLACE)
suspend fun insertImages(items:List<MarsItem>)

@Query("SELECT * FROM images")
fun getImages(): LiveData<List<MarsItem>>
```

```
@GET("realestate")
    suspend fun getProperties():
Response<List<MarsResponse>>
```

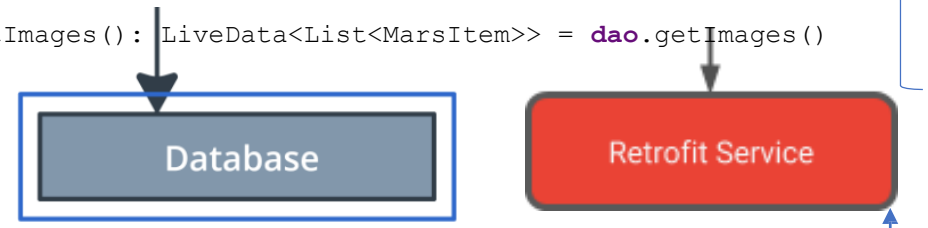



```
marsViewModel.images.observe(this) { adapter.data = it }

marsViewModel.error.observe(this){
          Toast.makeText(context,it,Toast.LENGTH_SHORT).show()
}
```

```
val images: LiveData<List<MarsItem>>
    get() = repository.getMars()

fun loadMars() {
    viewModelScope.launch {
        repository.loadMars { error.postValue(it) }
    }
}
```

```
suspend fun insertImages(items: List<MarsItem>)
{ dao.insertImages(items)}

fun getImages(): LiveData<List<MarsItem>> = dao.getImages()
```

```
fun getMars(): LiveData<List<MarsItem>> = cache.getImages()
suspend fun loadMars(
        onError: (error: String) -> Unit)

{...
cache.insertImages(it.map { item ->
     MarsItem(item.price, item.id, item.type, item.img_src)
    })
 ...}
```




```
@Insert(onConflict = OnConflictStrategy.REPLACE)
suspend fun insertImages(items:List<MarsItem>)

@Query("SELECT * FROM images")
fun getImages(): LiveData<List<MarsItem>>
```

```
@GET("realestate")
    suspend fun getProperties():
Response<List<MarsResponse>>
```

Získanie údajov z Internetu

```
interface WebApi {
    @GET("realestate")
    suspend fun getProperties(): Response<List<MarsResponse>>

    companion object {
        private const val BASE_URL =
            "https://android-kotlin-fun-mars-server.appspot.com"

        fun create(context: Context): WebApi {

            val client = OkHttpClient.Builder()
                .build()

            val retrofit = Retrofit.Builder()
                .baseUrl(BASE_URL)
                .client(client)
                .addConverterFactory(GsonConverterFactory.create())
                .build()

            return retrofit.create(WebApi::class.java)
        }
    }
}
```

Získanie údajov z Internetu

```
suspend fun loadMars(onError: (error: String) -> Unit) {

    try {
        val response = api.getProperties()
        if (response.isSuccessful) {
            response.body()?.let {
                return cache.insertImages(it.map { item ->
                        MarsItem(item.price, item.id, item.type, item.img_src)
                        })
            }
        }
        onError("Load images failed. Try again later please.")
    } catch (ex: ConnectException) {
        onError("Off-line. Check internet connection.")
        ex.printStackTrace()
        return
    } catch (ex: Exception) {
        onError("Oops...Change failed. Try again later please.")
        ex.printStackTrace()
        return
    }
}
```

Získanie údajov z Internetu

```
class MarsViewModel(private val repository: DataRepository) : ViewModel() {

    val error: MutableLiveData<String> = MutableLiveData()

    val images: LiveData<List<MarsItem>>
        get() = repository.getMars()

    init {
      loadMars()
    }

    fun loadMars() {
        viewModelScope.launch {
            repository.loadMars { error.postValue(it) }
        }
    }
}
```

Internet – Samoštúdium

- 8.1 Getting data from the internet
- 8.2 Loading and displaying images from the internet
- 8.3 Filtering and detail views with internet data

STU FEI
SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Mobilné výpočty

Ing. Maroš Čavojský, PhD.

maros.cavojsky@stuba.sk
C606