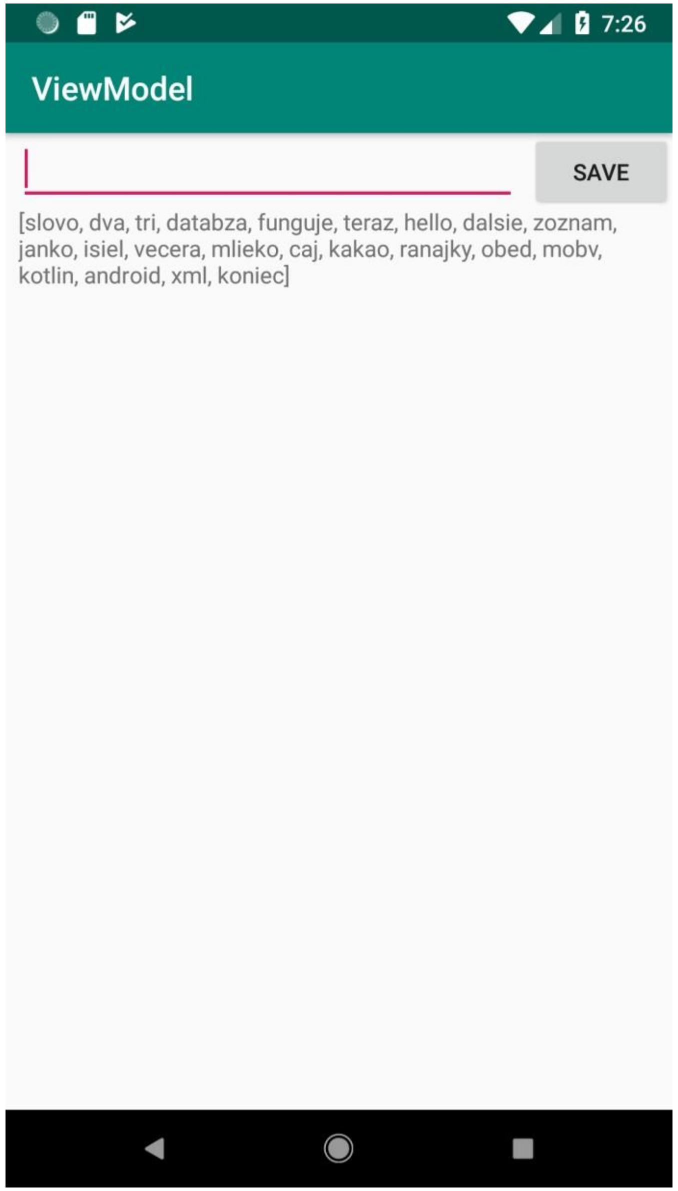
SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

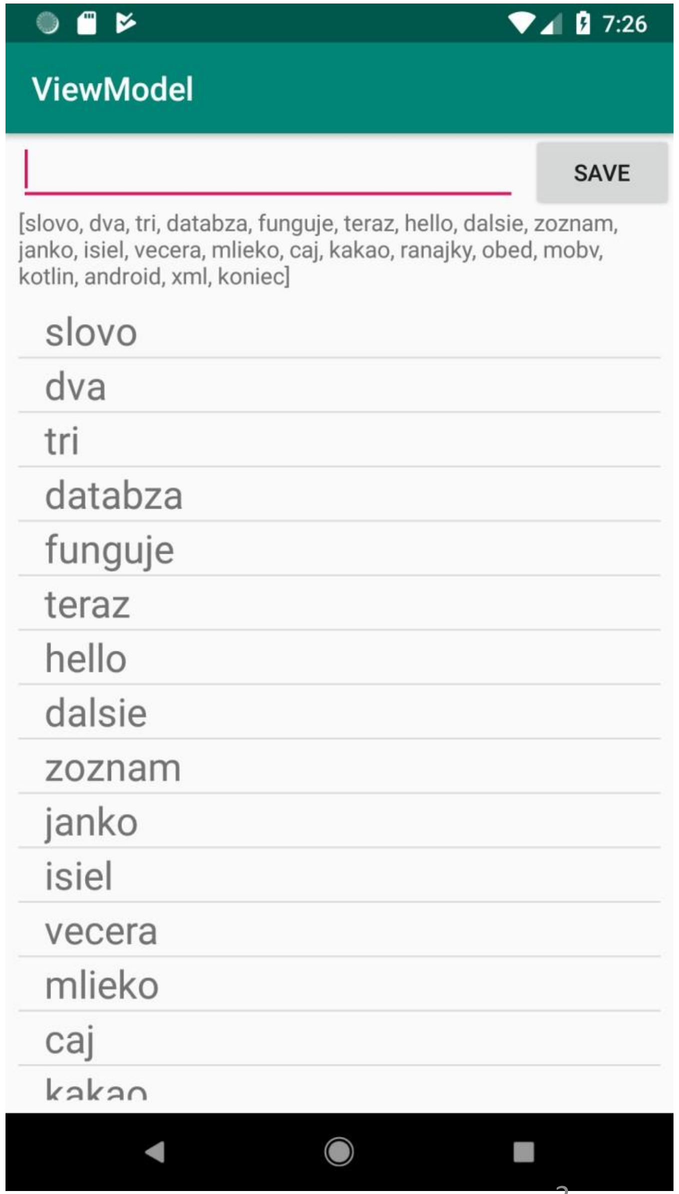
Mobilné výpočty

Ing. Maroš Čavojský, PhD.

RecyclerView



[slovo, dva, tri, databza, funguje, teraz, hello, dalsie, zoznam, janko, isiel, vecera, mlieko, caj, kakao, ranajky, obed, mobv, kotlin, android, xml, koniec]



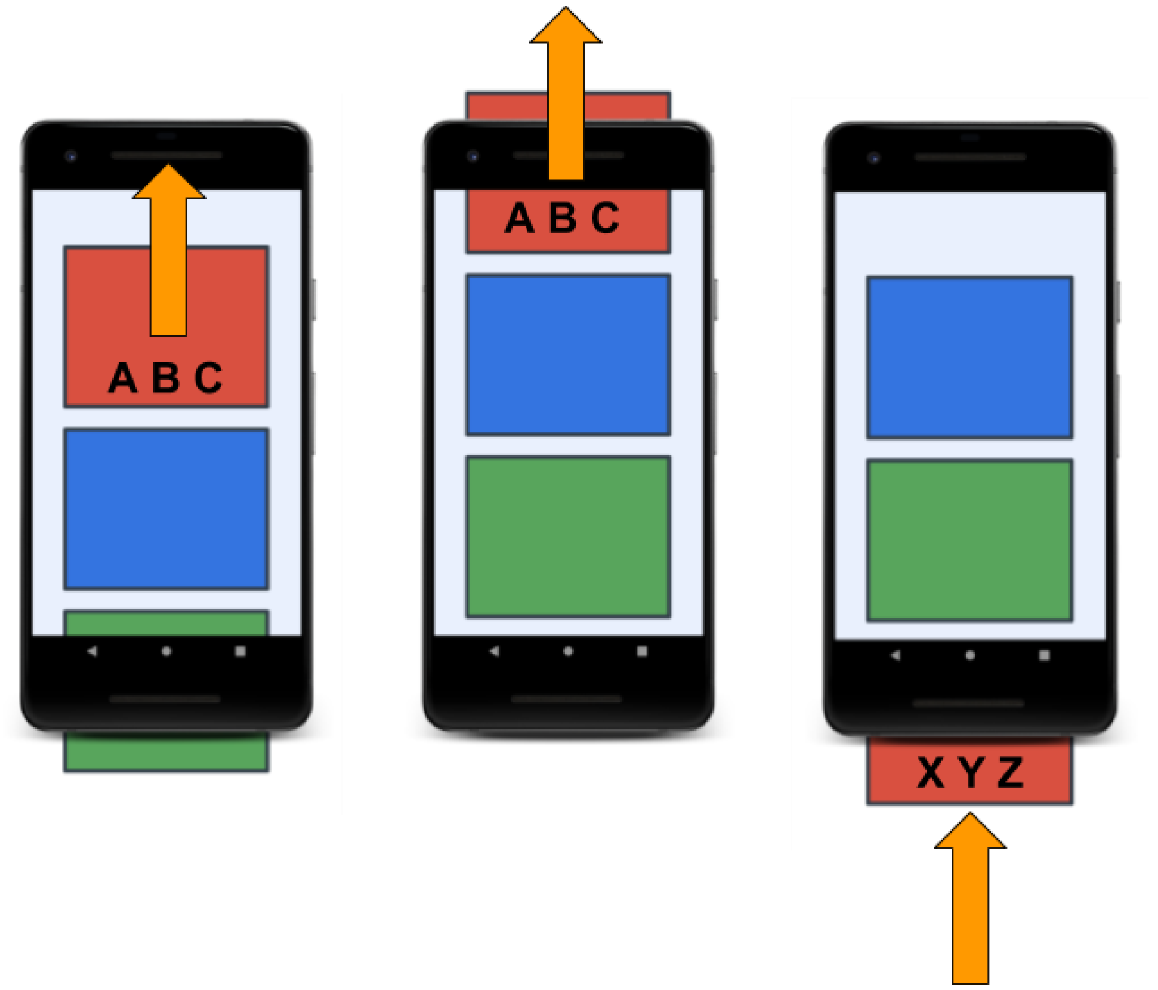
[slovo, dva, tri, databza, funguje, teraz, hello, dalsie, zoznam, janko, isiel, vecera, mlieko, caj, kakao, ranajky, obed, mobv, kotlin, android, xml, koniec]

slovo
dva
tri
databza
funguje
teraz
hello
dalsie
zoznam
janko
isiel
vecera
mlieko
caj
kakao

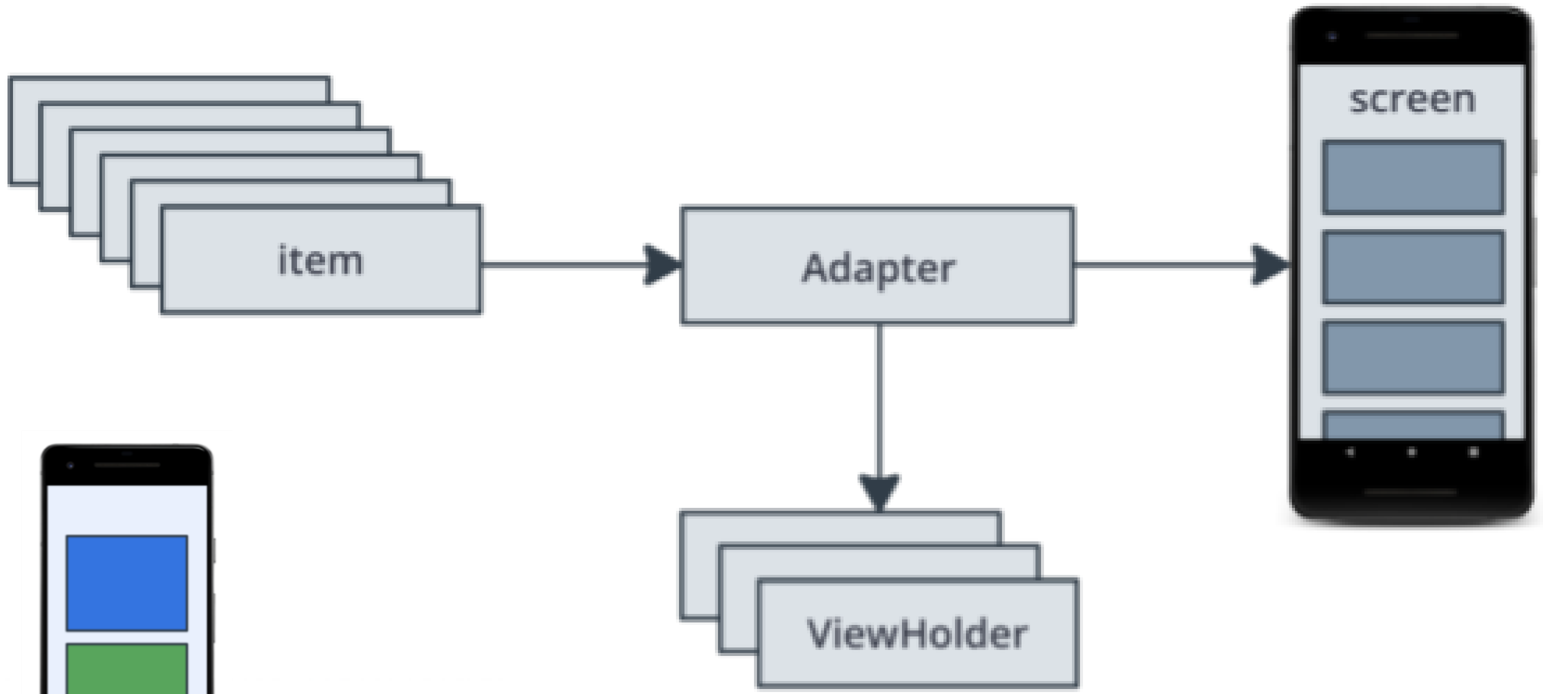
RecyclerView

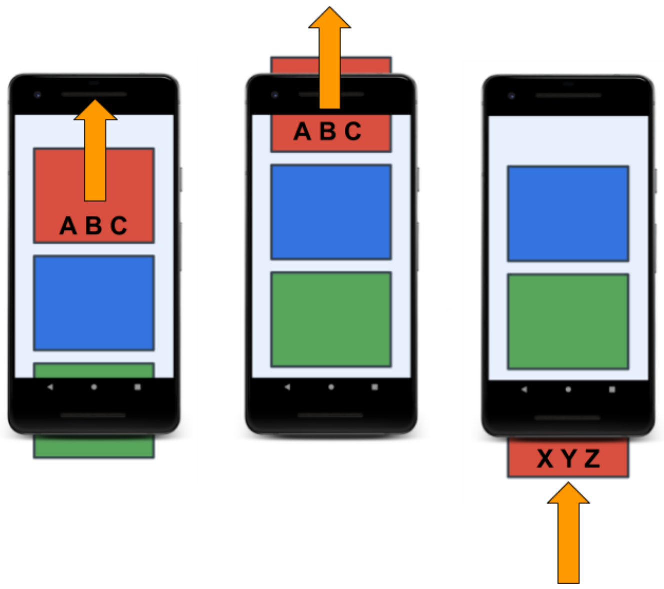
STU FEI SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE FAKULTA ELEKTROTECHNIKY A INFORMATIKY

RecyclerView



RecyclerView

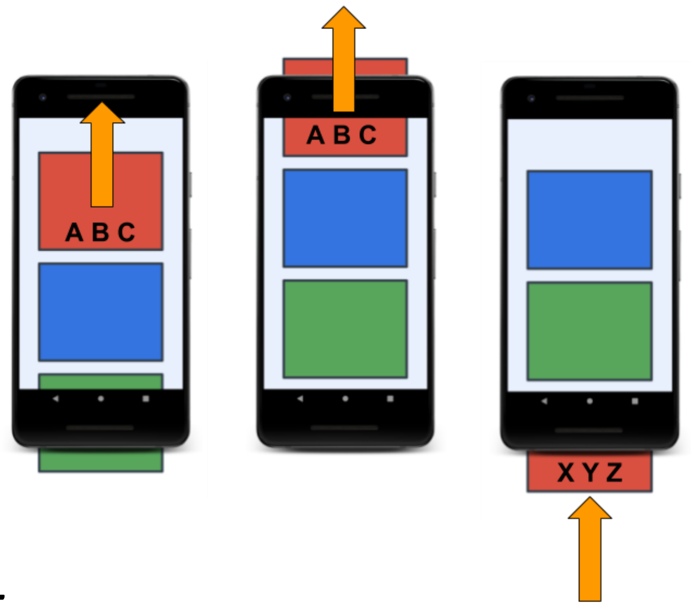




RecyclerView - checklist

- ❑ Dáta
- ❑ RecyclerView v layout-e
- ❑ Layout/y pre konkrétny prvok/prvky
- ❑ Layout manager
- ❑ ViewHolder
- ❑ Adapter

RecyclerView



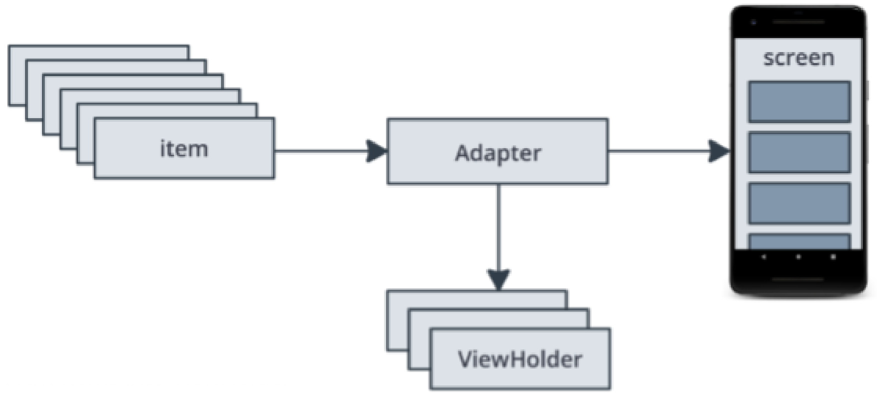
- *Adapter*
- *Position – v rámci adaptéra*
- *Index – v rámci child View ... getChildAt()*
- *Binding – proces pripájania dát na view*
- *Recycle (view) – view už použitý na zobrazenie dát*
- *Scrap (view) – child view dočasne odpojený z layout-u*

Môžu byť znovu použité bez toho aby boli úplne „detachnuté“ z rodiča RecyclerView, tiež bez úpravy ak nie je potrebné žiadne opätovné bindovanie alebo modifikované adaptérom ak je „dirty“

- *Dirty (view) – child view potrebný znova bound-núť pred zobrazením*

RecyclerView - checklist

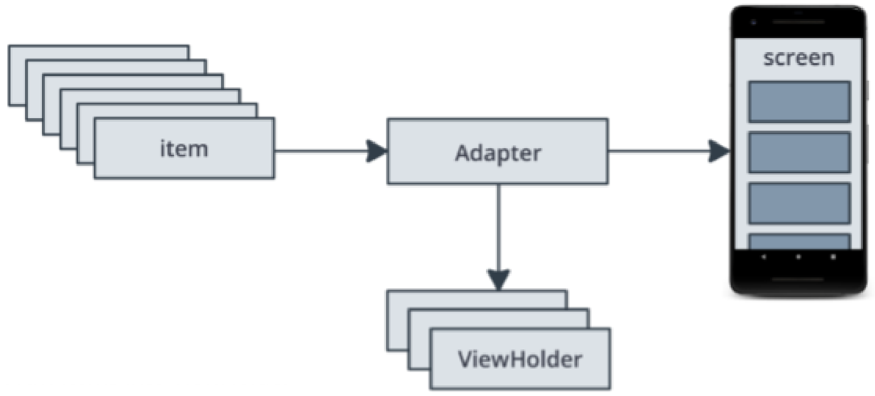
- ❑ Dáta
- ❑ RecyclerView v layout-e
- ❑ Layout/y pre konkrétny prvok/prvky
- ❑ Layout manager
- ❑ ViewHolder
- ❑ Adapter



```
implementation 'androidx.recyclerview:recyclerview:1.0.0'
```

RecyclerView

- ❑ Dáta
- ❑ RecyclerView v layout-e
- ❑ Layout/y pre konkrétny prvok/prvky
- ❑ Layout manager
- ❑ ViewHolder
- ❑ Adapter



```
implementation 'androidx.recyclerview:recyclerview:1.0.0'
```

Dáta

- Kolekcia – List<Any>
- LiveData – LiveData<List<Any>>

- ***notifyDataSetChanged()***
- notifyItemChanged(int position)
- notifyItemInserted(int position)
- notifyItemMoved(int fromPosition, int toPosition)
- notifyItemRangeChanged(int positionStart, int itemCount)
- notifyItemRangeInserted(int positionStart, int itemCount)
- notifyItemRangeRemoved(int positionStart, int itemCount)
- notifyItemRemoved(int position)

RecyclerView

```
<androidx.recyclerview.widget.RecyclerView
        android:id="@+id/words_list“
......
/>
```

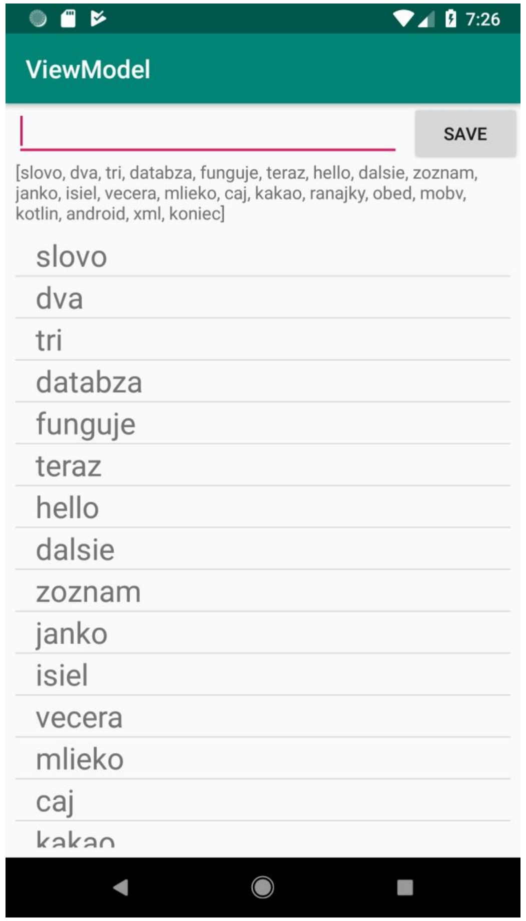
Layout/y pre konkrétny prvok/prvky

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
          android:textSize="24sp"
          android:padding="16dp"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"/>
```

Layout manager

LinearLayoutManager

GridLayoutManager

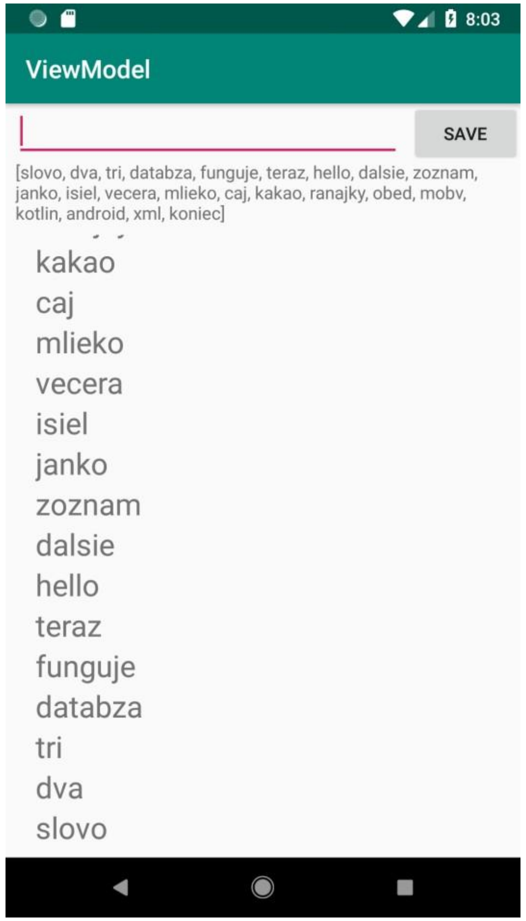

7:26

ViewModel

SAVE

[slovo, dva, tri, databza, funguje, teraz, hello, dalsie, zoznam, janko, isiel, vecera, mlieko, caj, kakao, ranajky, obed, mobv, kotlin, android, xml, koniec]

slovo
dva
tri
databza
funguje
teraz
hello
dalsie
zoznam
janko
isiel
vecera
mlieko
caj
kakao

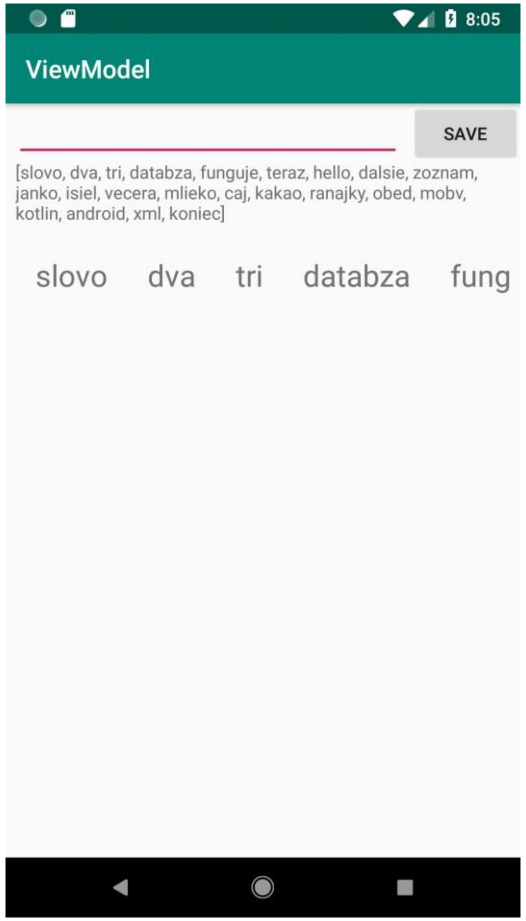

8:03

ViewModel

SAVE

[slovo, dva, tri, databza, funguje, teraz, hello, dalsie, zoznam, janko, isiel, vecera, mlieko, caj, kakao, ranajky, obed, mobv, kotlin, android, xml, koniec]

kakao
caj
mlieko
vecera
isiel
janko
zoznam
dalsie
hello
teraz
funguje
databza
tri
dva
slovo

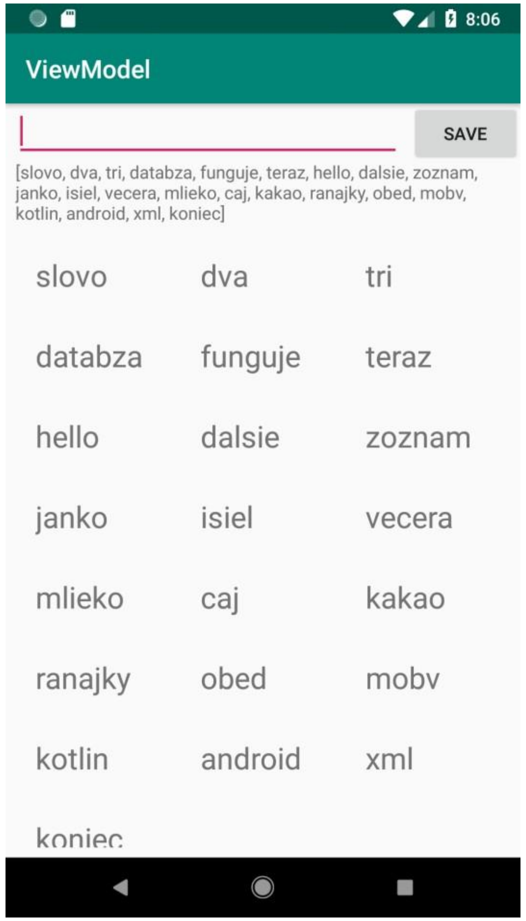

8:05

ViewModel

SAVE

[slovo, dva, tri, databza, funguje, teraz, hello, dalsie, zoznam, janko, isiel, vecera, mlieko, caj, kakao, ranajky, obed, mobv, kotlin, android, xml, koniec]

slovo dva tri databza fung


8:06

ViewModel

SAVE

[slovo, dva, tri, databza, funguje, teraz, hello, dalsie, zoznam, janko, isiel, vecera, mlieko, caj, kakao, ranajky, obed, mobv, kotlin, android, xml, koniec]

slovo	dva	tri
databza	funguje	teraz
hello	dalsie	zoznam
janko	isiel	vecera
mlieko	caj	kakao
ranajky	obed	mobv
kotlin	android	xml
koniec		

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

ViewHolder

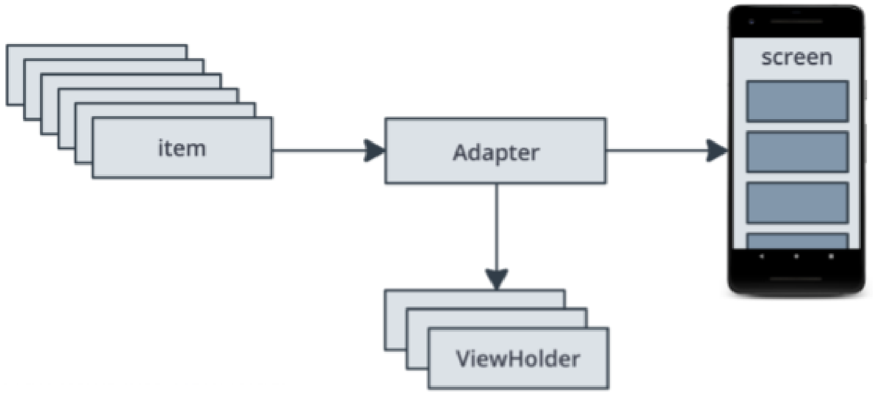
```
class ViewHolder private constructor(itemView: View) : RecyclerView.ViewHolder(itemView) {

    fun bind(item: WordItem) {
        (itemView as TextView).text = item.word
    }

    companion object {
        fun from(parent: ViewGroup): ViewHolder {
            val layoutInflater = LayoutInflater.from(parent.context)
            val view = layoutInflater
                .inflate(R.layout.text_item, parent, false)

            return ViewHolder(view)
        }
    }
}
```

Adapter

```
class MessagesAdapter : RecyclerView.Adapter<MessagesAdapter.ViewHolder>() {
     var data = listOf<WordItem>()
          set(value) {
               field = value
               notifyDataSetChanged()
          }

     override fun getItemCount() = data.size

     override fun onBindViewHolder(holder: ViewHolder, position: Int) {
          val item = data[position]
          holder.bind(item)
     }

     override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
              return ViewHolder.from(parent)
          }

     class ViewHolder … {}
}
```

RecyclerView - pozície

- Layout pozícia – [RecyclerView.LayoutManager]
- Adaptér pozícia - [RecyclerView.Adapter]



Rovnaké ???

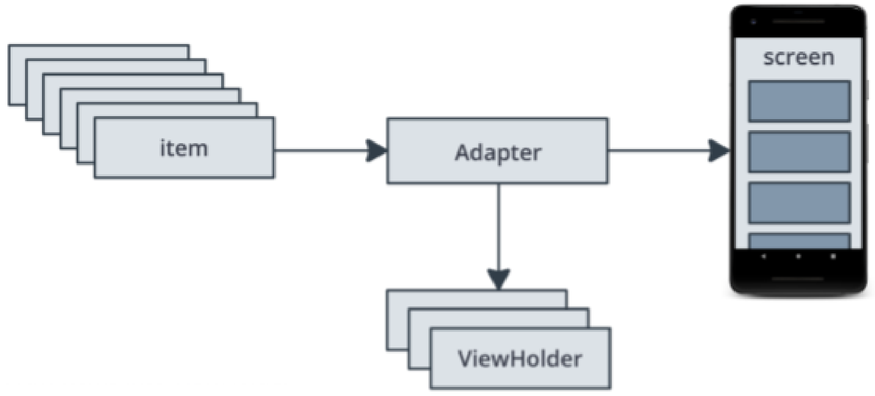
RecyclerView - pozície

❖ Layout pozícia – RecyclerView.LayoutManager

- getLayoutPosition(),
- findViewHolderForLayoutPosition(int)

❖ Adaptér pozícia - RecyclerView.Adapter

- getAdapterPosition(),
- findViewHolderForAdapterPosition(int)



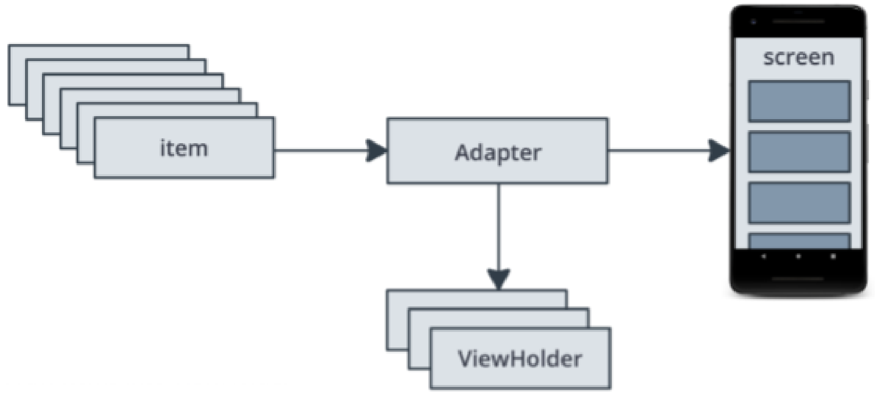
Rovnaké ???   Áno aj nie. Odlišné v čase medzi adapter.notify.   a úpravou layoutu*

RecyclerView - pozície

- onClick ? Akú pozíciu ?

Ak chceme pristúpiť k prvku na, ktorý sme

Klikli z eventu v našom ViewHolder-i



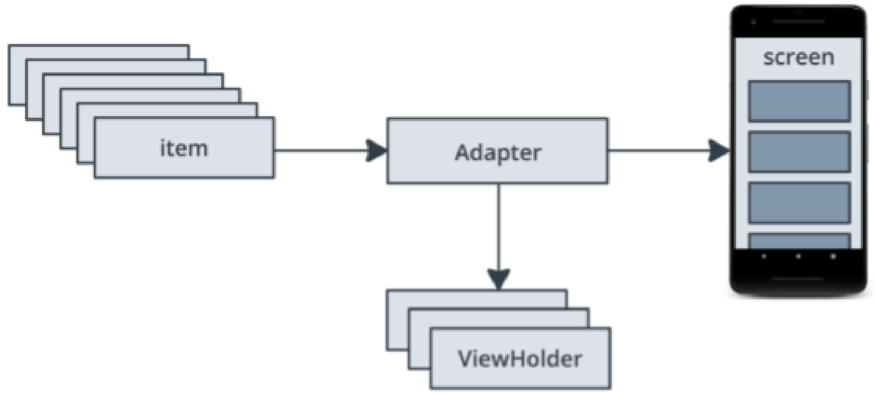
RecyclerView - pozície

- onClick ? Akú pozíciu ?

Ak chceme pristúpiť k prvku na, ktorý sme

Klikli z eventu v našom ViewHolder-i

Mali by sme použiť *getAdapterPosition()*



Pozor! Metódy na získanie pozície v adaptéri nedokážu získať

pozíciu medzi volaním adapter.notify. a kým nebol vypočítaný nový layout.*

Treba ošetriť NO_POSITION alebo null.

RecyclerView - binding

```
class ViewHolder(private var binding: ImageItemBinding) : RecyclerView.ViewHolder(binding.root) {


    fun bind(item: MarsItem) {
        binding.property = item
        binding.executePendingBindings()
    }

    companion object {
        fun from(parent: ViewGroup): ViewHolder {
            val view = ImageItemBinding.inflate(LayoutInflater.from(parent.context))

            return ViewHolder(view)
        }
    }
}
```

RecyclerView - binding

```
class MarsAdapter : ListAdapter<MarsItem, MarsAdapter.ViewHolder>(DiffCallback) {

    companion object DiffCallback : DiffUtil.ItemCallback<WordItem>() {
        override fun areItemsTheSame(oldItem: WordItem, newItem: WordItem): Boolean {
            return oldItem === newItem
        }

        override fun areContentsTheSame(oldItem: WordItem, newItem: WordItem): Boolean {
            return oldItem.compareTo(newItem)==0
        }
    }

    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        val item = getItem(position)
        holder.bind(item)
    }
```

RecyclerView - binding

```
<layout>
    <data>
        <variable name="property"  type="com.example.viewmodel.data.db.model.WordItem"/>
    </data>
    <TextView
            app:mojText="@{property}"
            android:textSize="24sp"
            android:paddingStart="16dp"
            android:paddingEnd="16dp"
            android:layout_width="match_parent" android:layout_height="wrap_content"/>
</layout>
```

RecyclerView - binding

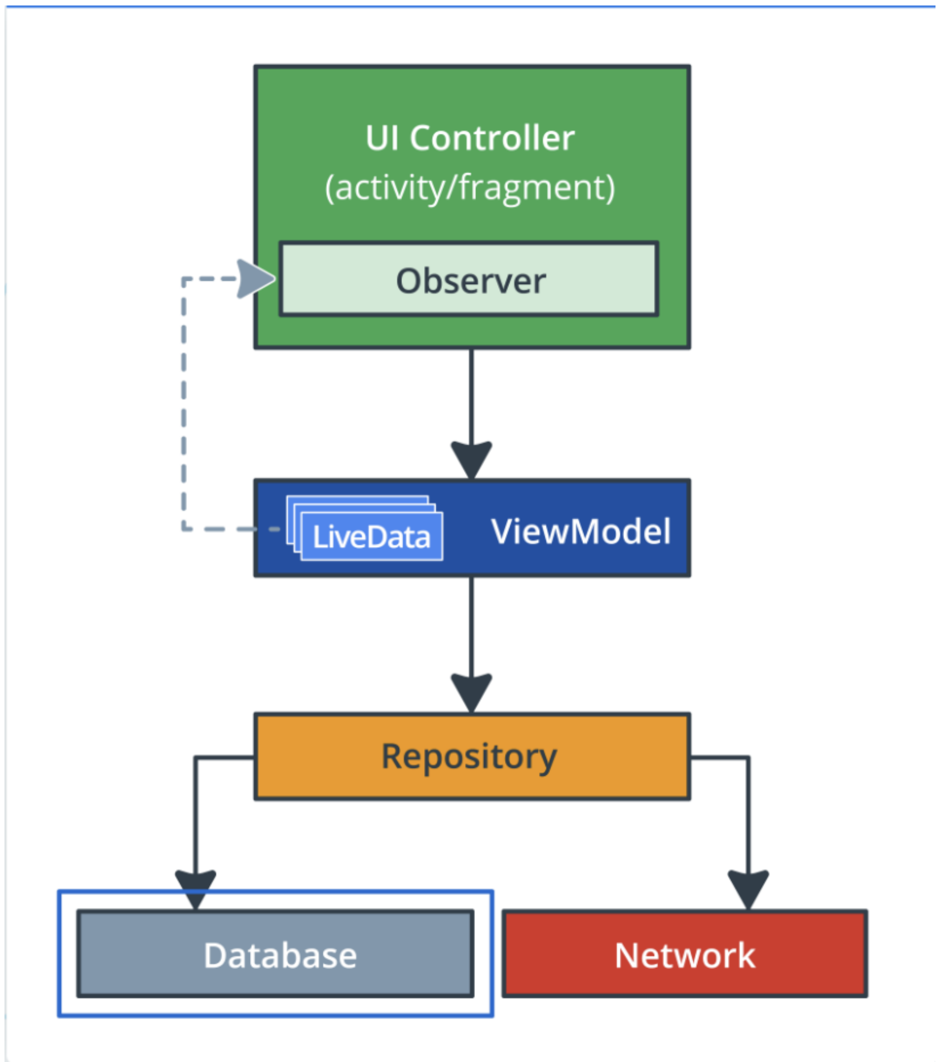
```
<layout>
    <data>
        <variable name="property" type="com.example.viewmodel.data.db.model.WordItem"/>
    </data>
    <TextView
            app:mojText="@{property}"
            android:textSize="24sp"
            android:paddingStart="16dp"
            android:paddingEnd="16dp"
            android:layout_width="match_parent" android:layout_height="wrap_content"/>
</layout>
```

```
@BindingAdapter("mojText")
fun TextView.setMojText(item: WordItem) {
    text = "Slovo je ${item.word}"
}
```

RecyclerView – Samoštúdium

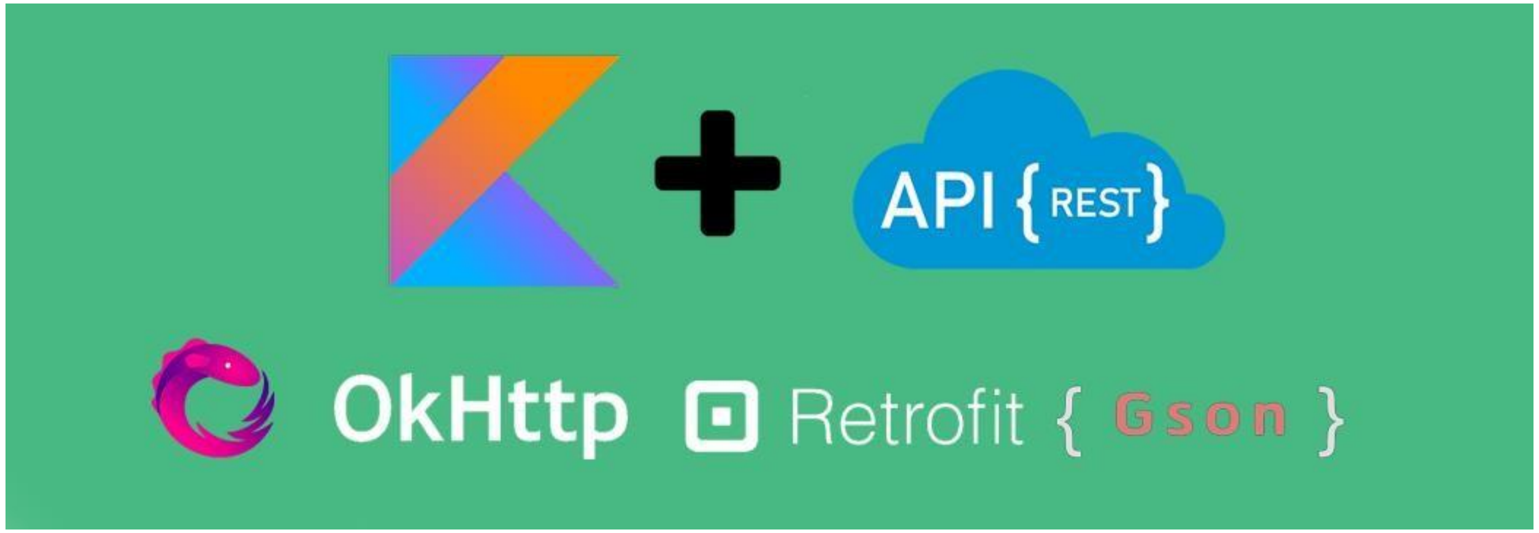
- 7.1 RecyclerView fundamentals
- 7.2 DiffUtil and data binding with RecyclerView
- 7.3 GridLayout with RecyclerView
- 7.4 Interacting with RecyclerView items
- 7.5 Headers in RecyclerView

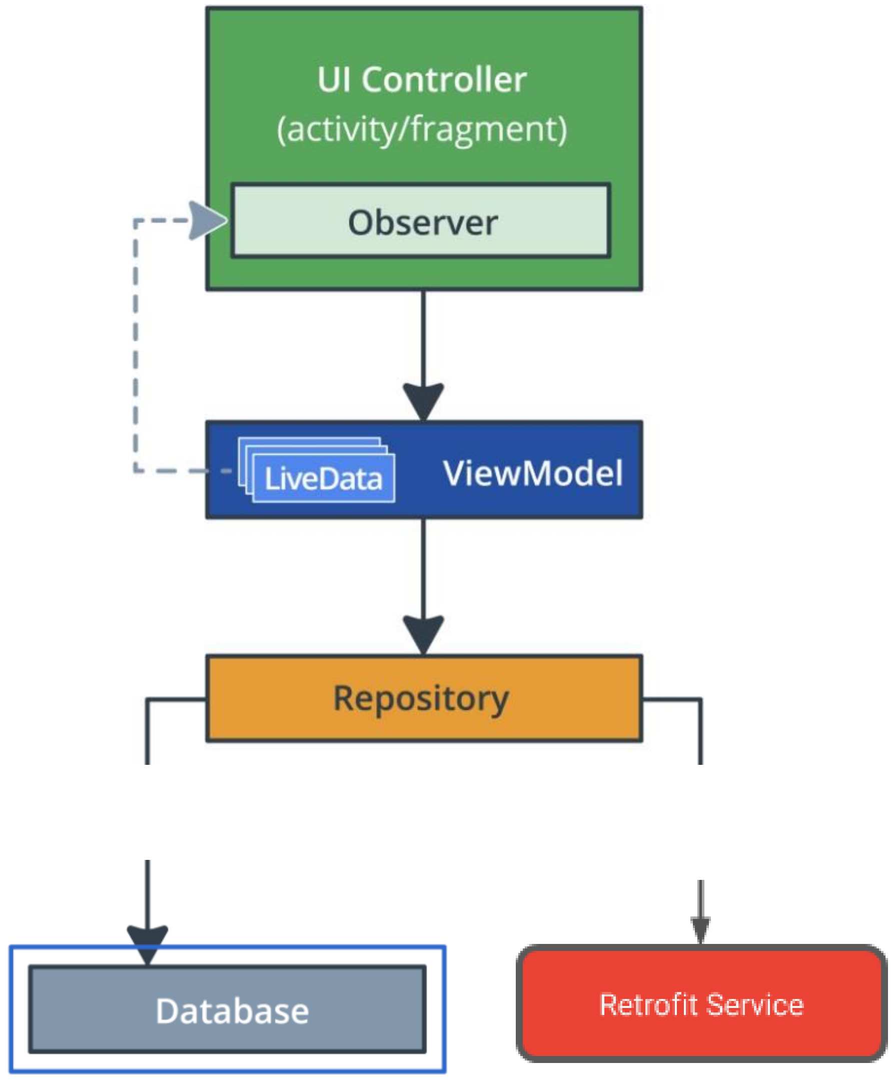
Získanie údajov z Internetu

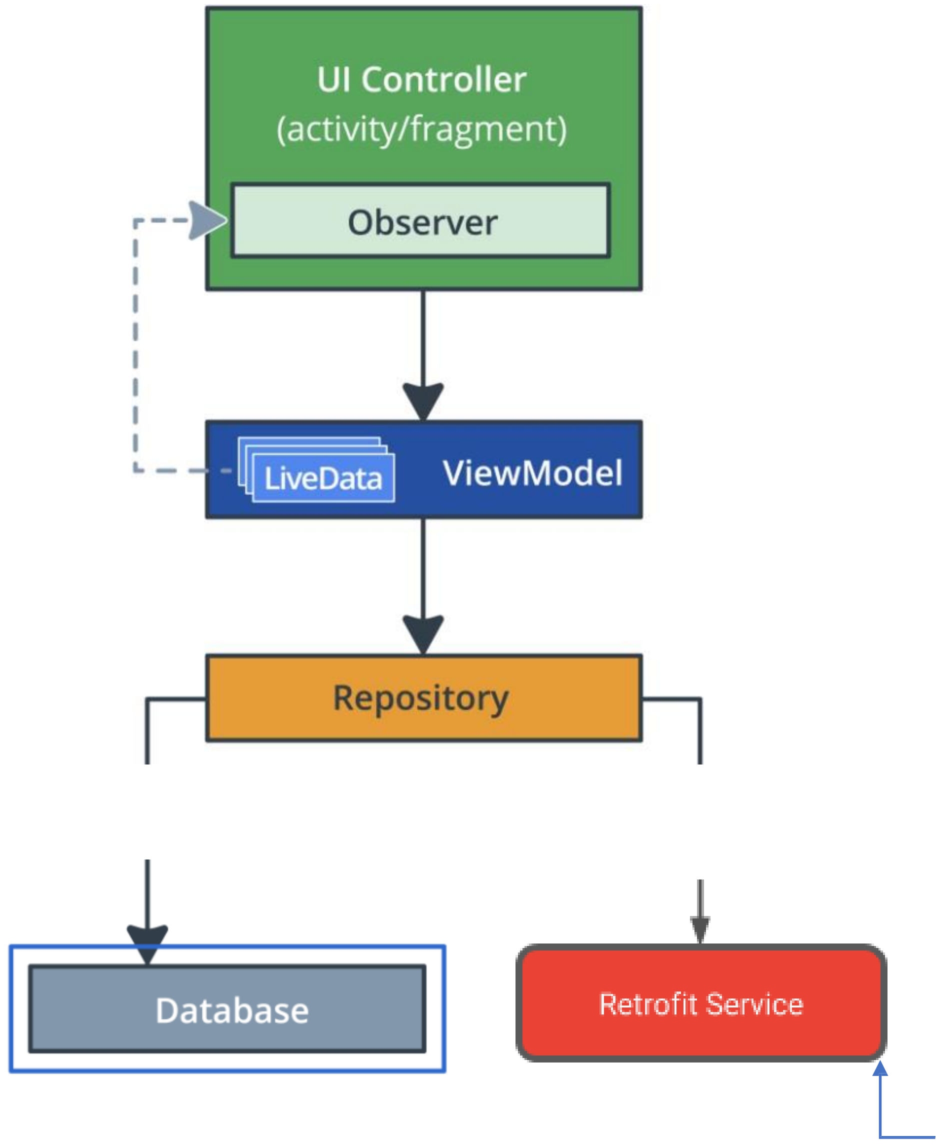


Získanie údajov z Internetu

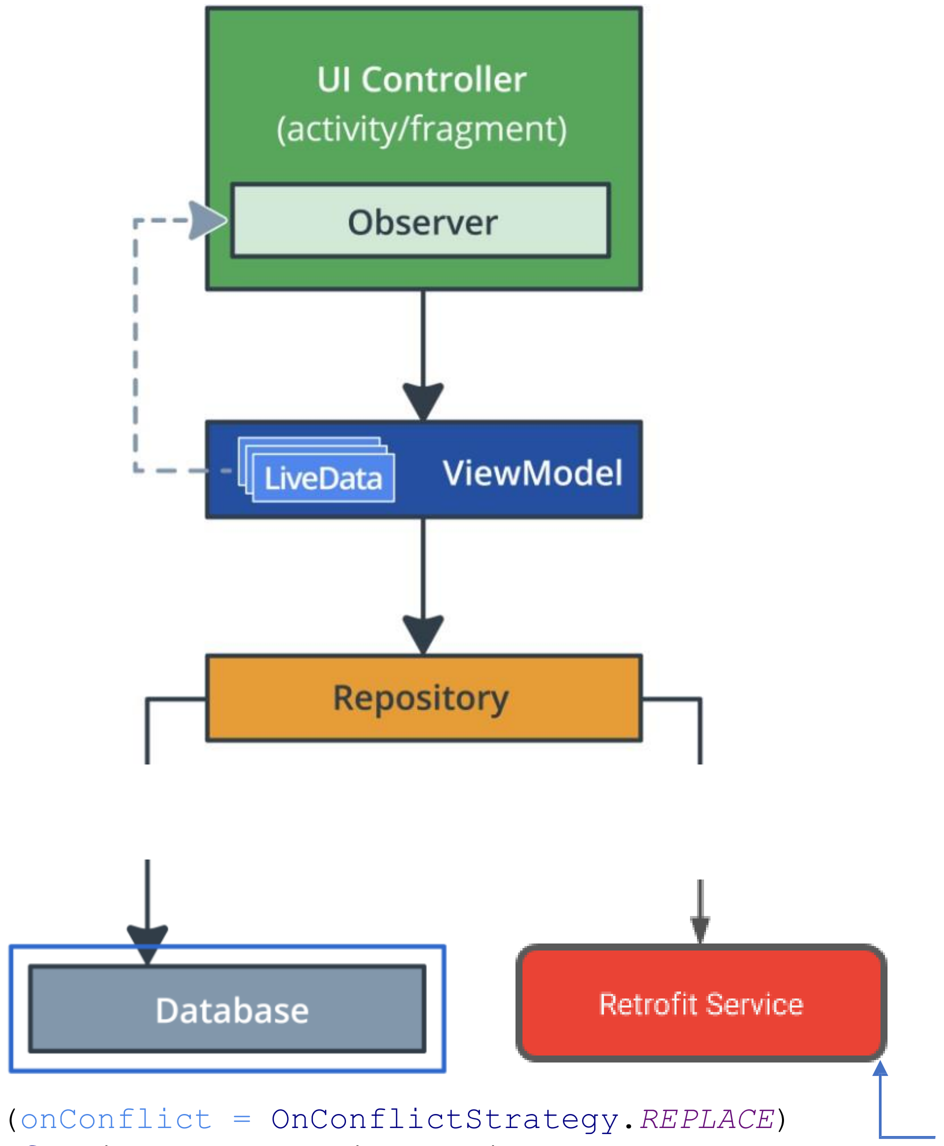
```
//Webservice
implementation 'com.squareup.retrofit2:retrofit:2.6.0'
implementation 'com.squareup.retrofit2:converter-gson:2.6.0'
implementation 'com.google.code.gson:gson:2.8.5'
```




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

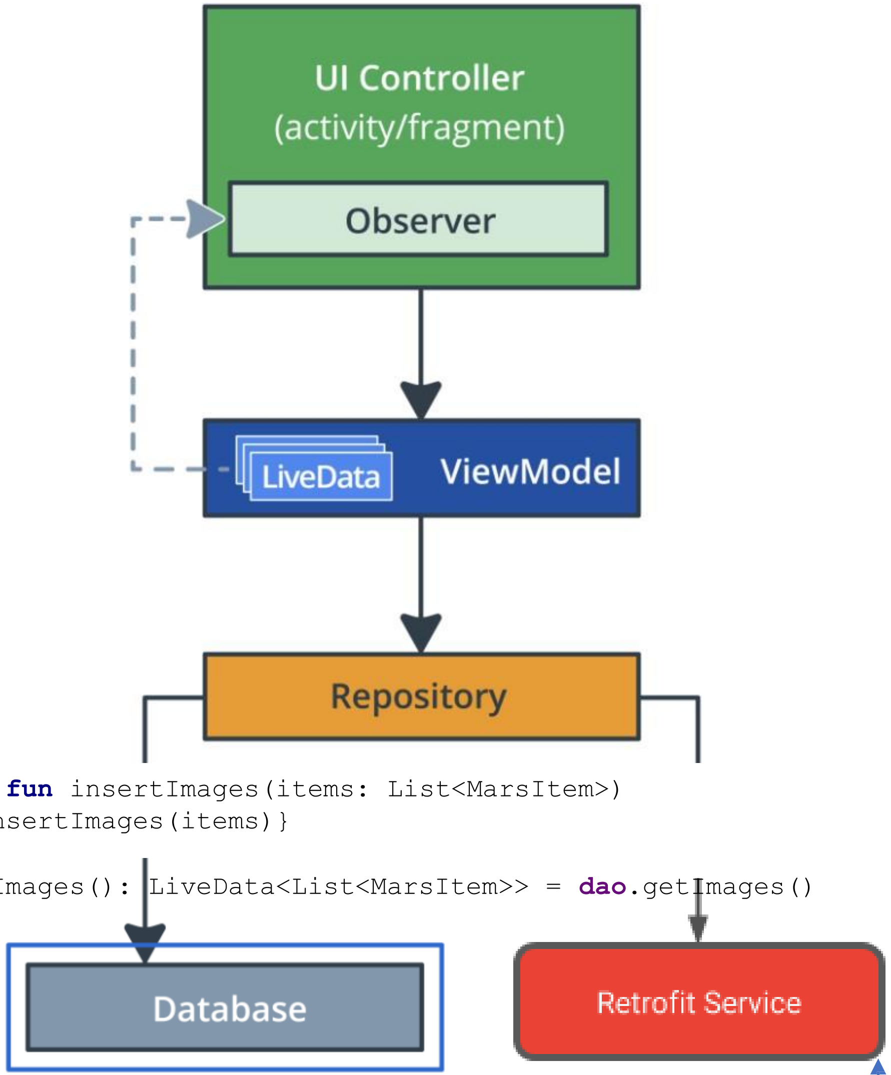


```
@GET("realestate")
    suspend fun getProperties():
Response<List<MarsResponse>>
```



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

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

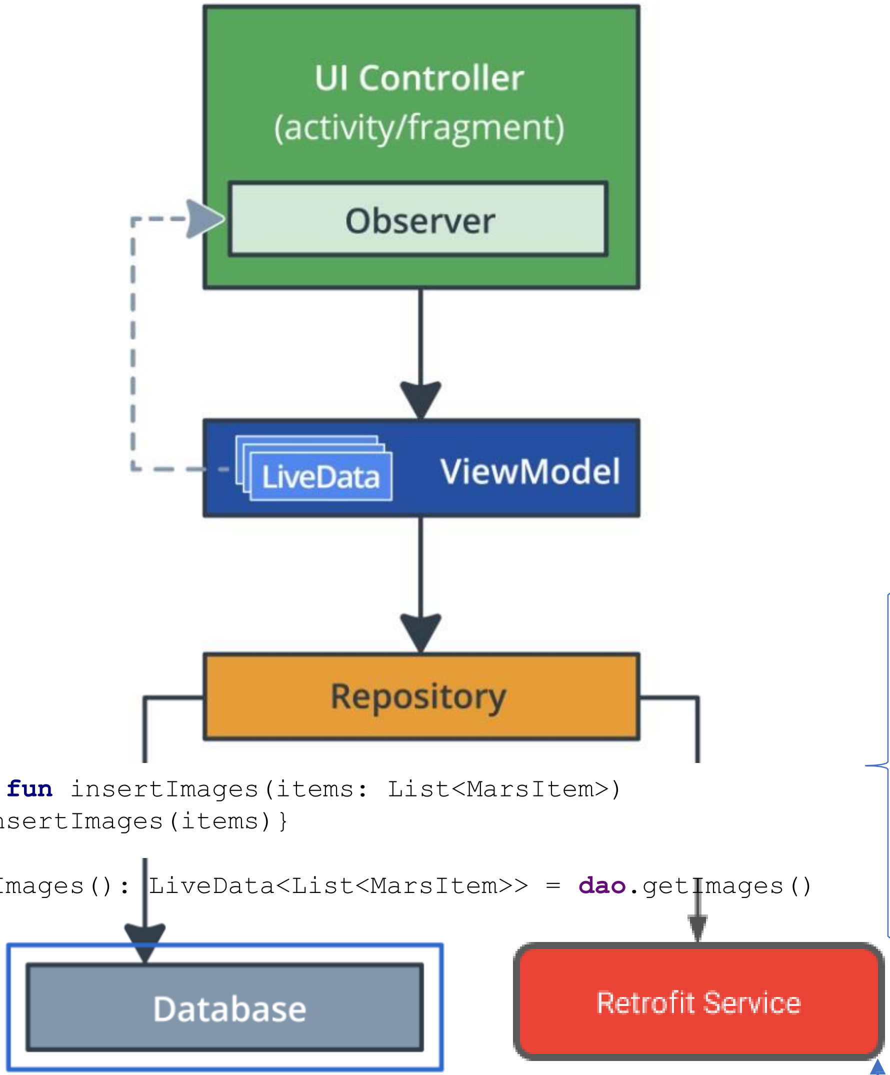
```
@GET("realestate")
    suspend fun getProperties():
Response<List<MarsResponse>>
```



```
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>>
```

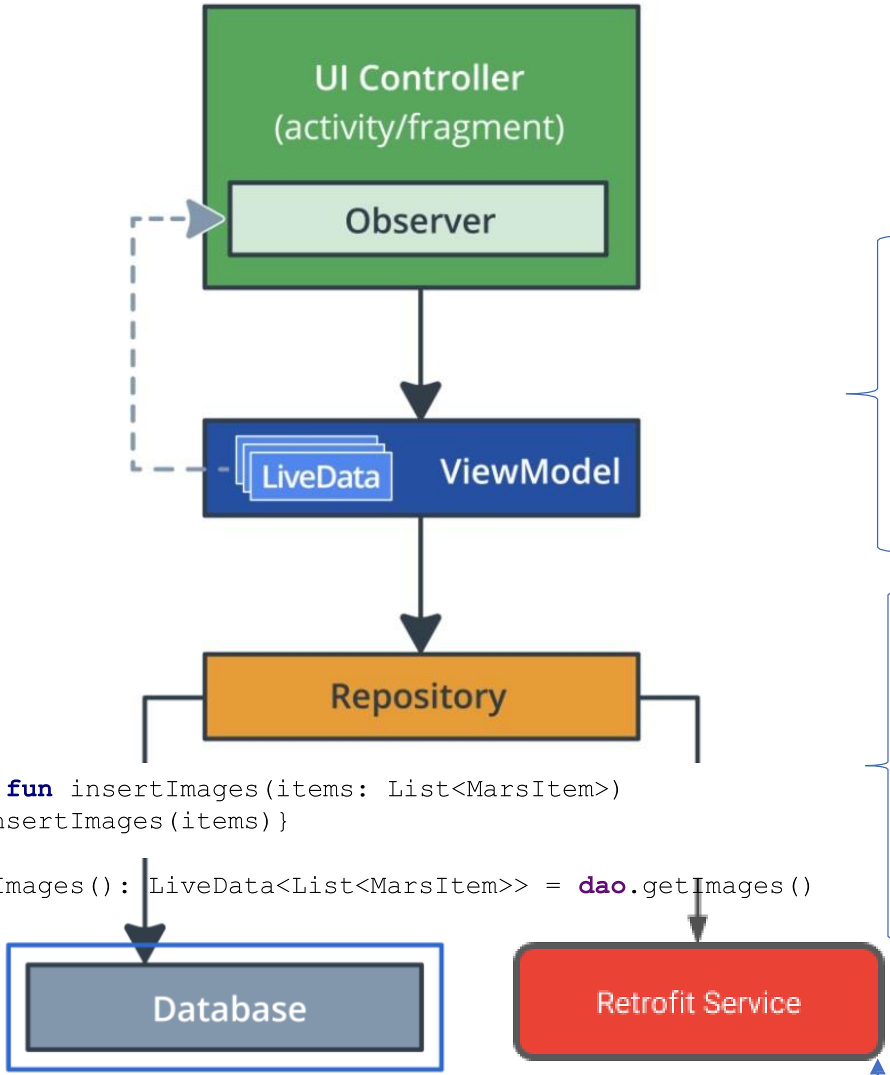


```
suspend fun insertImages(items: List<MarsItem>)
{ dao.insertImages(items)}

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

```
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>>
```



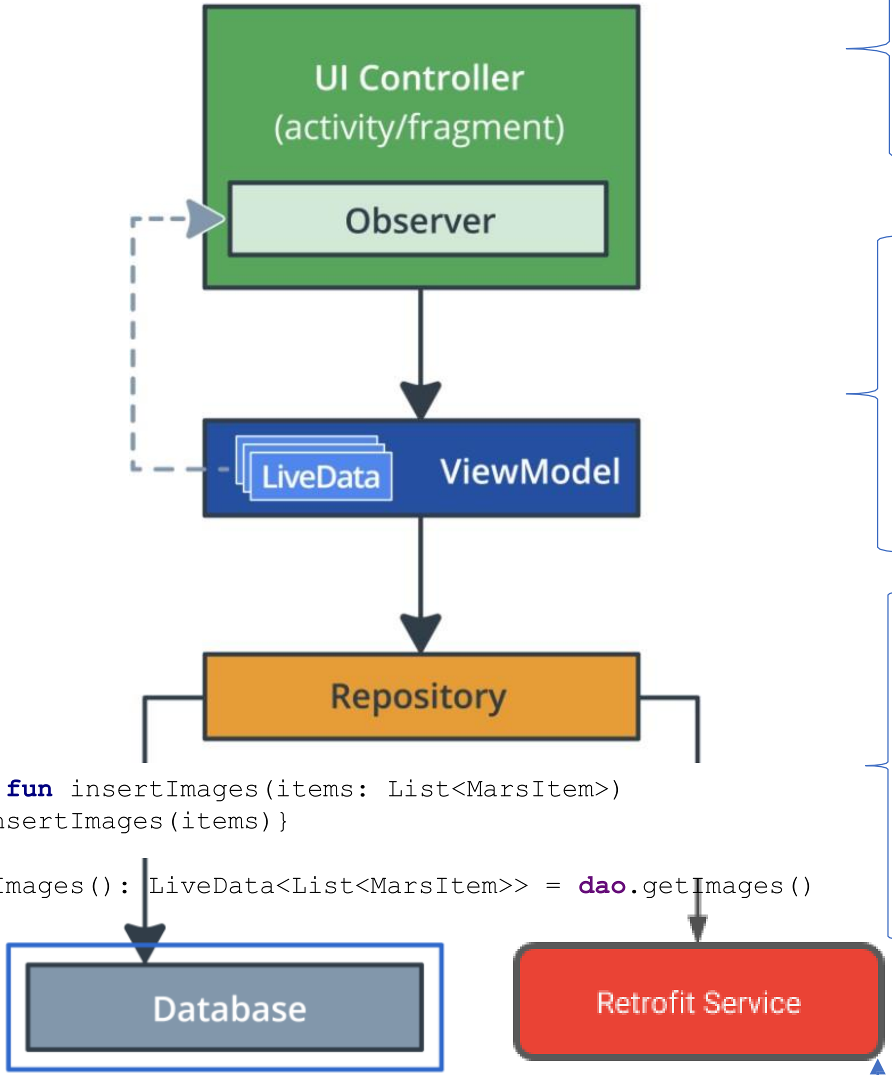
```
val images: LiveData<List<MarsItem>>
    get() = repository.getMars()

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()
```

```
@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>>
```



```
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) }
    }
}
```

```
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>>
```

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

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Mobilné výpočty

Ing. Maroš Čavojský, PhD.

maros.cavojsky@stuba.sk
C606