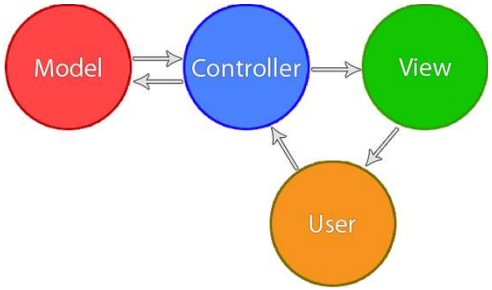
Jetpack

Peter Borovanský
KAI, I-18

borovan 'at' ii.fmph.uniba.sk

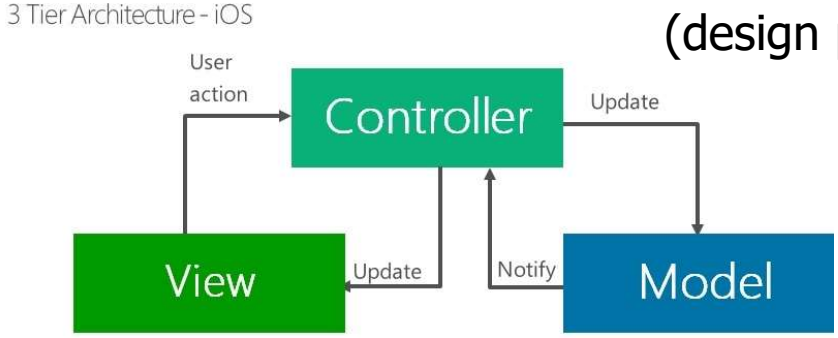


Architektonický *mess*
(MVC)

- vzniká, ak vizuálne komponenty (Views) sú zviazané s dátovými objektami a opačne

```
prev.setOnClickListener(new OnClickListener() {
  @Override
  public void onClick(View v) {
    i++;
    i %= imgs.length;
    iv.setImageDrawable(imgs[i]);
  }
});
```

- preto sa pri návrhu GUI používajú návrhové vzory, Model-View-Controller (design patterns)



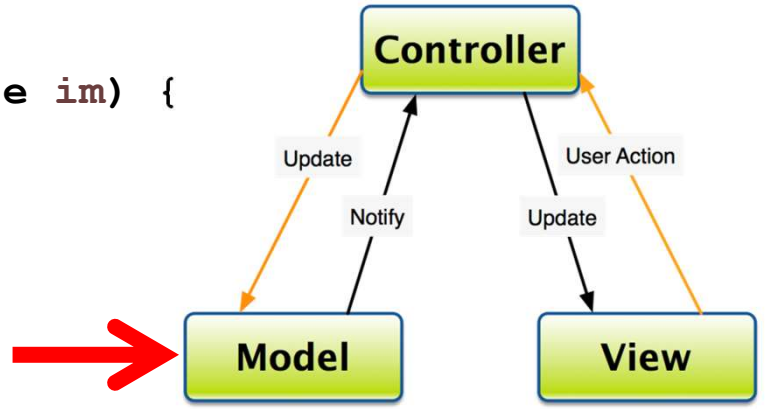
- motto: the architecture of most Android-apps is a mess.

http://doridori.github.io/Android-Architecture-MV%3F/#sthash.SiE5eude.IQq3XhmU.dpbs

Model View Controller (MVC)

(model – len data, netuší nič o ich prezentácii)

```
public class Model extends Observable {
int indx = 0;           // actual picture on the screen
ArrayList<Drawable> list = new ArrayList<Drawable>(); // all pics

public void addDrawableImage(Drawable im) {
   list.add(im);
}
public Drawable getDrawable() {
   return list.get(indx);
}
public void nextValue() {
   indx++;
   indx %= list.size();
   setChanged();
   notifyObservers();
}
```



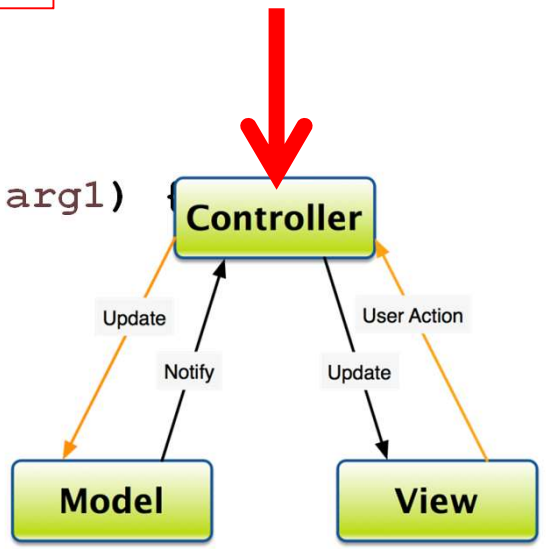

```
public void prevValue() {
  indx--;
  if (indx < 0)
    indx = list.size()-1;
  setChanged();
  notifyObservers();
}
```

PikatchuMVC.zip

Model View Controller (MVC)

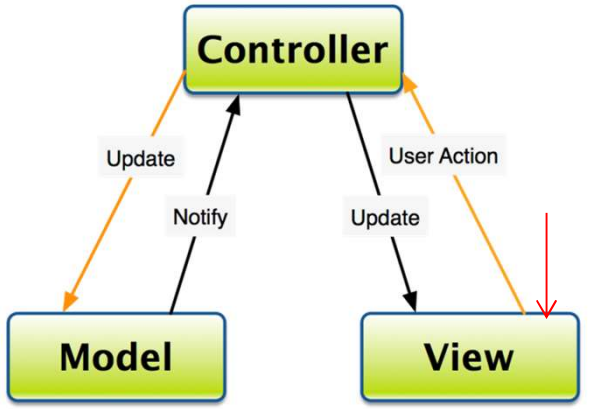
(controller – komunikuje medzi modelom a view)

```
public class Controller extends … implements Observer {

mModel = new Model();
mModel.addObserver(this);
mModel.addDrawableImage(getResources().getDrawable(R.drawable.pok0));
mModel.addDrawableImage(getResources().getDrawable(R.drawable.pok1));

mView = new myView(this);

@Override
public void update(Observable arg0, Object arg1) {
   mView.update(mModel.getDrawable());
}
```



PikatchuMVC.zip

Model View Controller (MVC) (view)

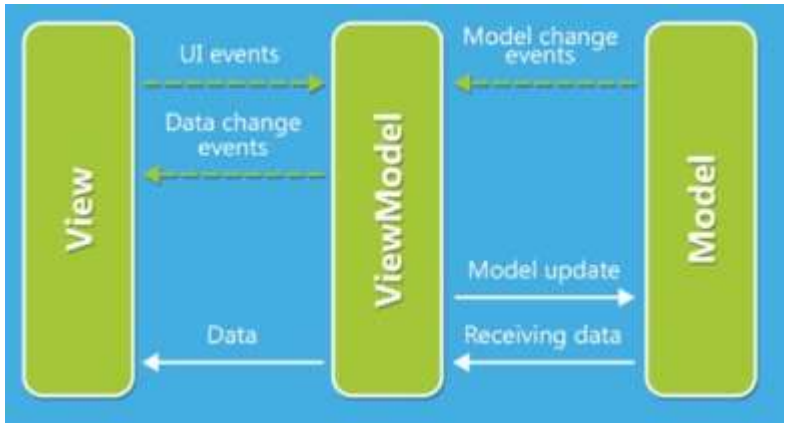



```
public class myView {
    final Controller controller;
    ImageView iv;
    Button prev, next;

public myView(Controller c) {
    this.controller = c;
    iv = (ImageView)mainActivity.findViewById(R.id.imageView1);
    Button prev = (Button)mainActivity.findViewById(R.id.prevBtn);
    prev.setOnClickListener(new OnClickListener() {
    @Override
         public void onClick(android.view.View v) {
           controller.mModel.prevValue(); }
    });
public void update(android.graphics.drawable.Drawable im) {
    iv.setImageDrawable(im);
}
```

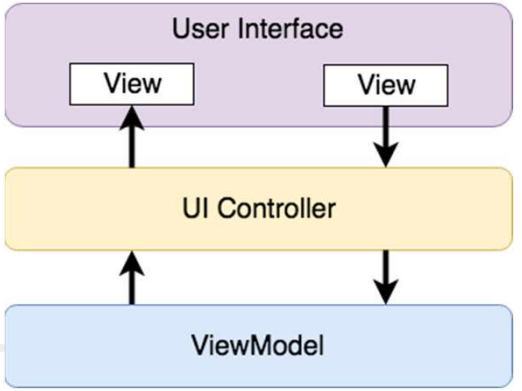
PikatchuMVC.zip

Čo je JetPack

- celý iOS vývoj postavený na Swifte je striknte založený na Model-View-Controller vzore (MVC)
- na mnohých príkladoch single activity apps sme videli, že sa mieša kód pre GUI s business logikou aplikácie
- Google si to uvedomil 2017 a navrhol JetPack pre multi-activity apps
- cieľom:
  - je oddeliť kód pre GUI od kódu s logikou
  - problémy so životným cyklom, napr. pri rotácii displaya
  - perzistenciu dát
- architektúra separácie GUI a logic kódu založená na ViewModel, nie MVC



ViewModel

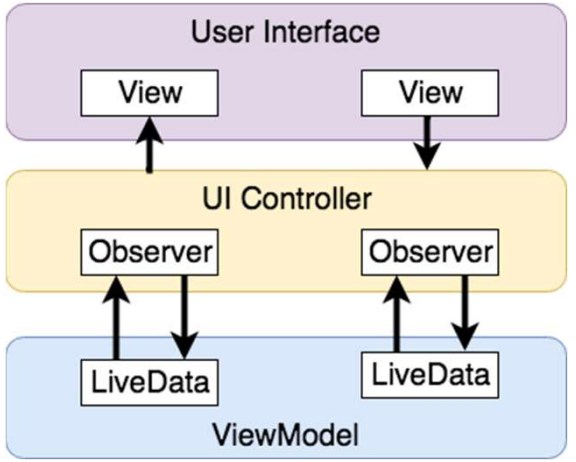


- ViewModel je jediný, čo vie o dátach a ich logike
- keď zmeníme GUI, ViewModel zostáva nezmenený
- ak sa zmení napr. orientácia, tak ViewModel stále drží pôvodné data

- dáta sa ale môžu meniť nezávisle a často, napr. realtime data
- kedy sa má GUI dopytovať, či nemá dáta prekresliť, či sa náhodou nezmenili
- argesívne „spojité" poolovanie dát je náročné, tak sa to nerobí

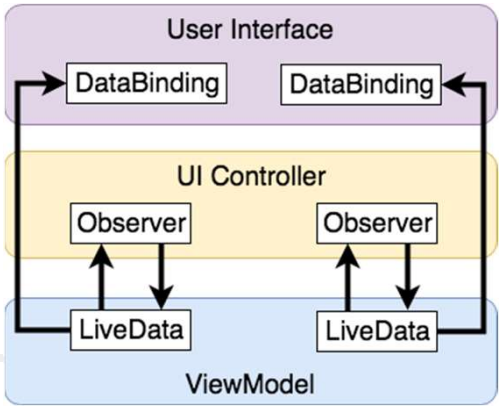
LiveData – Observer

- observer dostane info, ak sa dáta zmenia



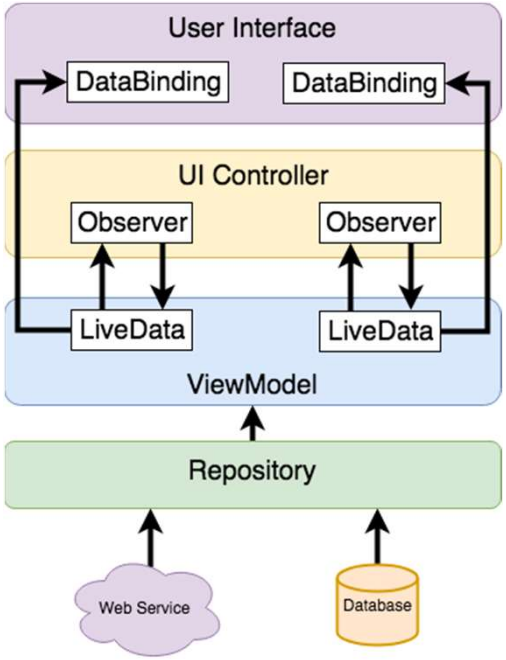
https://www.techotopia.com/index.php/Modern_Android_App_Architecture_with_Jetpack

Data Binding



- ako zabezpečiť, aby sa dáta v observeri správne zobrazili v GUI
- ViewModel má priamo informáciu o konktrétnom view v .xml layout file, kde sa majú dáta zobraziť

- Repository slúži na dáta extených zdrojov



https://www.techotopia.com/index.php/Modern_Android_App_Architecture_with_Jetpack

Projekt Fragmet+ViewModel

```
class MainFragment : Fragment() {
    companion object {  // statická metoda
        fun newInstance() = MainFragment()
    }
    private lateinit var viewModel: MainViewModel

    override fun onCreateView(inflater: LayoutInflater,
                                container: ViewGroup?,
                              savedInstanceState: Bundle?): View {
        return inflater.inflate(R.layout.main_fragment, container,
                                false)
    }
    override fun onActivityCreated(savedInstanceState: Bundle?) {
        super.onActivityCreated(savedInstanceState)
        viewModel = ViewModelProvider(this).get(MainViewModel::class.java)
        // TODO: Use the ViewModel
    }
}
```

```
import androidx.lifecycle.ViewModel

class MainViewModel : ViewModel() {
    // TODO: Implement the ViewModel
}
```

JetPack1.zip

Projekt Fragmet+ViewModel

```
class MainFragment : Fragment() {
  override fun onActivityCreated(savedInstanceState: Bundle?) {
     super.onActivityCreated(savedInstanceState)
     viewModel = ViewModelProvider(this).get(MainViewModel::class.java)
     convertBtn.setOnClickListener {
        if (inputAmount.text.isNotEmpty()) {
          viewModel.setInputCurrencyAmount(inputAmount.text.toString())
          viewModel.convertUSD2EURO = usd2euro.isChecked
          outputAmount.setText("%.2f".format(viewModel.outputCurrencyAmount))
} }    }
```

```
class MainViewModel : ViewModel() {
    val dolar2euroRate = 1.1f
    var convertUSD2EURO = true
    var inputCurrencyAmount = 0f
    var outputCurrencyAmount = 0f

   fun setInputCurrencyAmount(value : String) {
        inputCurrencyAmount = value.toFloat()
        outputCurrencyAmount =
           if   (convertUSD2EURO) inputCurrencyAmount * dolar2euroRate
          else inputCurrencyAmount / dolar2euroRate
}  }
```

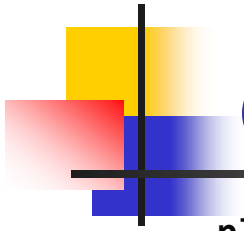
JetPack1.zip

LiveData

```
class MainFragment : Fragment() {
  override fun onActivityCreated(savedInstanceState: Bundle?) {
     super.onActivityCreated(savedInstanceState)
     viewModel = ViewModelProvider(this).get(MainViewModel::class.java)
     var resultObserver = Observer<Float> {
        result -> outputAmount.setText("%.2f".format(result))
     }
      viewModel.outputCurrencyAmount.observe(this, resultObserver)
     convertBtn.setOnClickListener {
        if (inputAmount.text.isNotEmpty()) {
          viewModel.setInputCurrencyAmount(inputAmount.text.toString())
          viewModel.convertUSD2EURO = usd2euro.isChecked
```

```
class MainViewModel : ViewModel() {
    val dolar2euroRate = 1.1f
    var convertUSD2EURO = true
    var inputCurrencyAmount = 0f
    var outputCurrencyAmount : MutableLiveData<Float> = MutableLiveData()
    fun setInputCurrencyAmount(value : String) {
       inputCurrencyAmount = value.toFloat()
       outputCurrencyAmount.value =
          if  (convertUSD2EURO) inputCurrencyAmount * dolar2euroRate
                                else inputCurrencyAmount / dolar2euroRate
```

JetPack2.zip

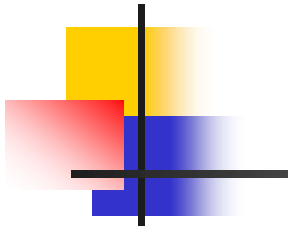


DataBinding

(build.gradle)

```
plugins {
    . . .
    id 'kotlin-android-extensions'
    id 'kotlin-kapt'
}
android {
    buildFeatures {
        dataBinding = true
    }
}
dependencies {
    annotationProcessor "com.android.databinding:compiler:$kotlin_version"
          ...
}
kapt {
    generateStubs = true
}
```

JetPack3.zip



DataBinding

(fragment.xml)

```
<?xml version="1.0" encoding="utf-8"?>

<layout xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools">
    <data>
        <variable
            name="myViewModel"
            type="com.example.jetpack3.ui.main.MainViewModel" />
    </data>
    <androidx.constraintlayout.widget.ConstraintLayout
      android:id="@+id/main"
      tools:context=".ui.main.MainFragment">

      <EditText
         android:text="@={myViewModel.inputCurrencyAmount}"
         android:hint="@string/input_currency_amount"/>

      <EditText
          android:id="@+id/outputAmount"
          android:text="@{String.valueOf(myViewModel.outputCurrencyAmount)}"
          android:text='@{safeUnbox(myViewModel.outputCurrencyAmount) == 0.0 ?"":
                                          String.valueOf(safeUnbox(myViewModel.outputCurrencyAmount))}’/>

      <Button
          android:id="@+id/convertBtn"
          android:onClick="@{() -> myViewModel.convertValue()}“ />

      <RadioGroup">
          <RadioButton
             android:id="@+id/usd2euro"
             android:checked="@={myViewModel.usd2euroChecked}"/>
          <RadioButton
             android:id="@+id/euro2usd"
             android:checked="@={myViewModel.euro2usdChecked}"/>
      </RadioGroup>
  </androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```

JetPack3.zip

DataBinding

```
class MainFragment : Fragment() {
    private lateinit var viewModel: MainViewModel
    lateinit var binding : MainFragmentBinding
    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
                              savedInstanceState: Bundle?): View {

        binding = DataBindingUtil.inflate(inflater,
                              R.layout.main_fragment, container, false)
        binding.setLifecycleOwner(this)
        return binding.root
    }

    override fun onActivityCreated(savedInstanceState: Bundle?) {
        super.onActivityCreated(savedInstanceState)
        viewModel = ViewModelProvider(this).get(MainViewModel::class.java)
        binding.setVariable(myViewModel, viewModel)
    }
}
```

JetPack3.zip

DataBinding

```
class MainViewModel : ViewModel() {
    val dolar2euroRate = 1.1f
    var usd2euroChecked : MutableLiveData<Boolean> = MutableLiveData()
    var euro2usdChecked : MutableLiveData<Boolean> = MutableLiveData()
    var inputCurrencyAmount : MutableLiveData<String> = MutableLiveData()
    var outputCurrencyAmount : MutableLiveData<Float> = MutableLiveData()

    fun convertValue() {
      inputCurrencyAmount.let {
        if ((it.value?:"").isNotEmpty()) {
          if (usd2euroChecked.value?:false)
           //outputCurrencyAmount.value=it.value?.toFloat()?.times(dolar2euroRate)
            outputCurrencyAmount.value = (it.value?:"0").toFloat() *
dolar2euroRate
          else
           //outputCurrencyAmount.value=it.value?.toFloat()?.div(dolar2euroRate)
           outputCurrencyAmount.value = (it.value?:"0").toFloat() / dolar2euroRate
        } else {
           outputCurrencyAmount.value = 0f
        }
}  } }
```

JetPack3.zip