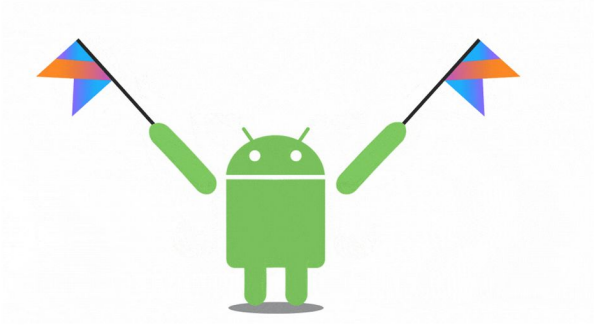
Gangame App

Curso de desarrollo de aplicaciones Android en Kotlin

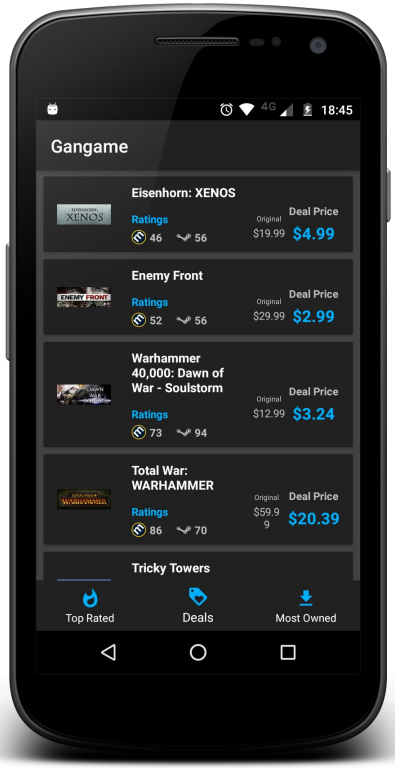


Pedro Antonio Hernández López
Software Developer at Bunsan.io
@silmood

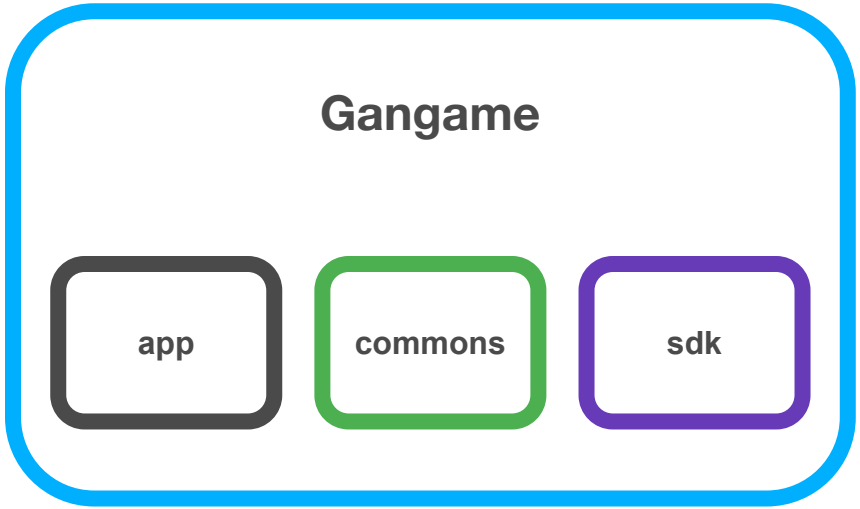
Pedro Antonio Hernández López
Software Developer at Bunsan.io
@silmood

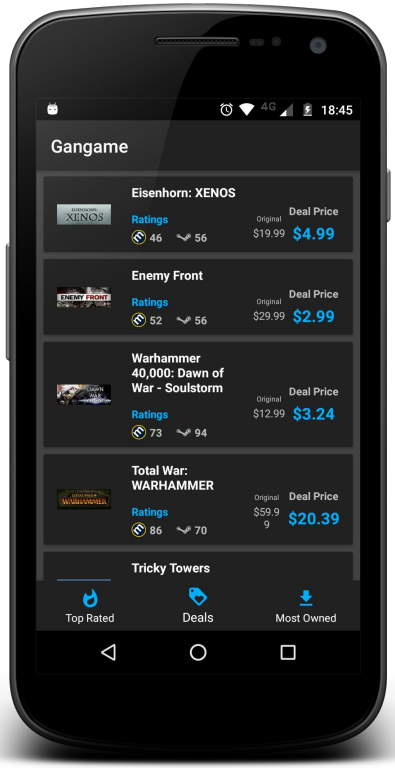
Gangame App

- Ofertas de videojuegos en Steam
- Top 100 en 2 semanas
- Juegos más adquiridos



Gangame App

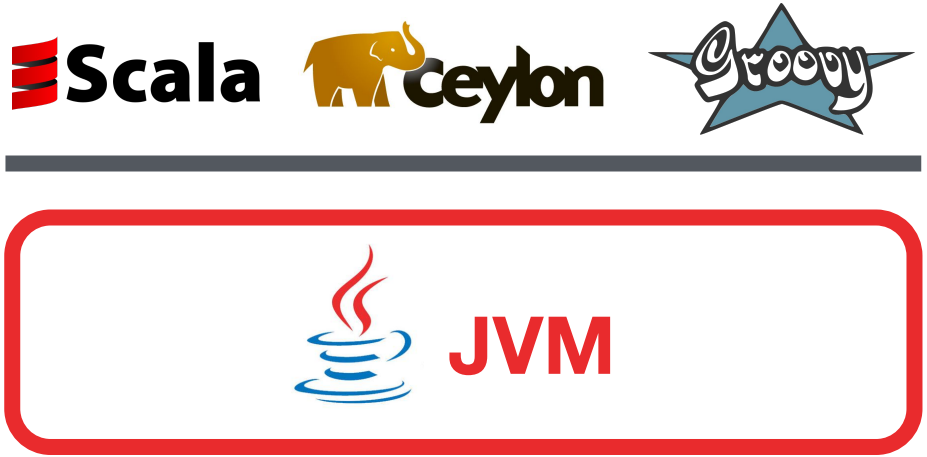




Kotlin

Un breve tour histórico

Kotlin



Kotlin

- Tipado inferido
- Multiparadigma
- Null Safety
- Ligero
- Configuración rápida

Gangame App

Creando nuestro proyecto

Configuración

```
apply plugin: 'kotlin-android'

compile "org.jetbrains.kotlin:kotlin-stdlib-jre7:$kotlin_version"
```

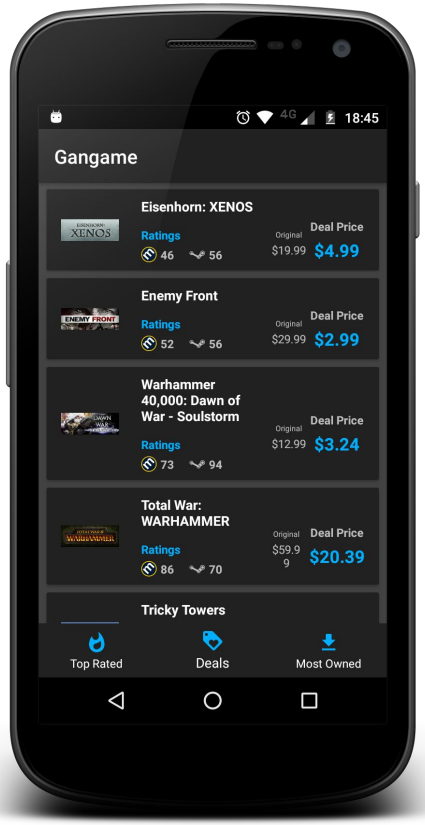
build.gradle (app)

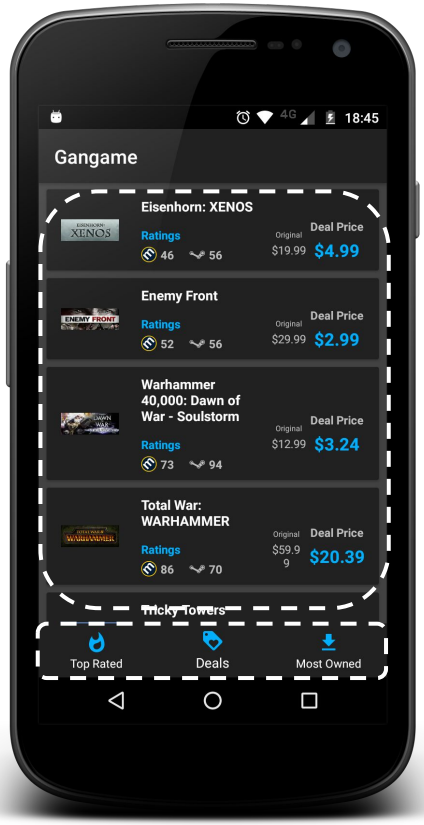
Gangame App

Definición del modelo de navegación

Navigation Bottom Bar

```
compile 'com.android.support:design:$support_ver'
```





activity_main.xml

- LinearLayout
  - FrameLayout
  - BottomNavigationView

Sintaxis básica

```
class DealsFragment: Fragment() {
    //...
}
```

Sintaxis básica

Constructores

```
class Deal constructor(price: Float){
    //...
}


class Deal(price: Float){
    //...
}
```

Sintaxis básica

Constructores

```
class Deal constructor(price: Float){
    val price: Float = price
}


class Deal constructor(val price: Float){
    //..
}
```

Sintaxis básica

Constructores

```
class Deal constructor(price: Float){
    val price: Float

    init {
        this.price = price
    }
}
```

Sintaxis básica

Definición de una función

```
fun greeting(name: String): String {
    return "Hello ${name}!"
}
```

Sintaxis básica

Null Safety

```
fun greeting(name: String?): String? {
    return if(name != null)
        "Hello ${name}!"
    else
        null
}
```

Sintaxis básica

Elvis operator y safe calls

```
fun wordCounter(name: String?): String {
    val count = name?.length ?: 0
    return "${count} letters!"
}
```

Creación de un nuevo módulo

gangame.commons

build.gradle (app)

```
compile project(':gangame.commons')
```

Sintaxis básica

Herencia

```
open class Base {
    open fun v() {}
    fun nv() {}
}
class Derived() : Base() {
    override fun v() {}
}
```

Las clases son finales por defecto

Sintaxis básica

Clases abstractas

```
abstract class BaseFragment : Fragment(){
    abstract fun getLayoutRes(): Int
}
```

Sintaxis básica

Extensiones

```
fun ViewGroup.inflate(viewId: Int, attachToRoot: Boolean): View {
   val inflater = LayoutInflater.from(context)
   return inflater.inflate(viewId, this, attachToRoot)
}
```

Sintaxis básica

Parámetros nombrados y por defecto

Definición

```
fun reformat(str: String,
             normalizeCase: Boolean = true,
             upperCaseFirstLetter: Boolean = true,
             divideByCamelHumps: Boolean = false,
             wordSeparator: Char = ' ') {
    ...
}
```

Llamada a función

```
reformat(str, true, false, '_')
```

```
reformat(str,
     normalizeCase = true,
     divideByCamelHumps = false,
     wordSeparator = '_'
)
```

Kotlin android extensions

```kotlin
findViewById(R.id.navigationView) as BottomNavigationView
```

```kotlin
import kotlinx.android.synthetic.main.activity_main.*

navigationView.selectedItemId = R.id.action_deals

// Behind the scenes
navigationView = findViewById(R.id.navigationView) as BottomNavigationView
```

build.gradle (app)

```gradle
apply plugin: 'kotlin-android-extensions'
```

Sintaxis básica

val / var

```
var price: Float = 0F

val title: String = ""
```

Sintaxis básica

val / var - setters, getter e inferencia de tipos

```
val isFree get() = price == 0.0F
```

Sintaxis básica

val / var - Backing fields

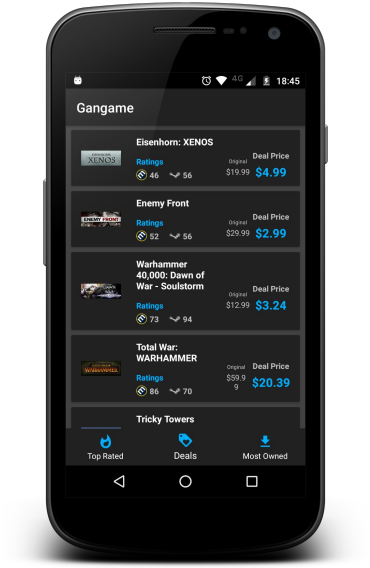
```
var counter: Int = 0
set(value) {
    if (value >= 0) field = value
}
```

Sintaxis básica

val / var - Constantes en tiempo de compilación

```
var counter: Int = 0
set(value) {
    if (value >= 0) field = value
}
```

Abstracción de fragmentos



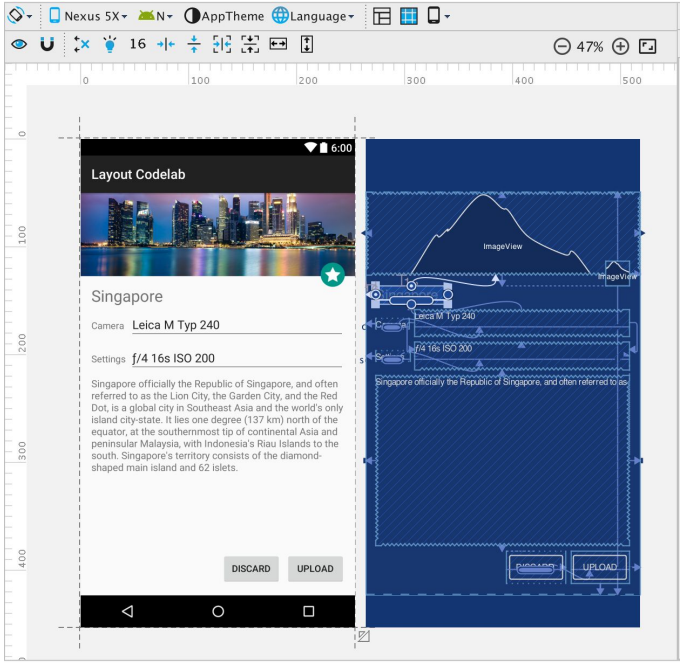
RecyclerView

```
BaseListFragment : BaseFragment
```

- `getAdapter(): RecyclerView.Adapter<*>`
- `setupList(list: RecyclerView)`

ConstraintLayout

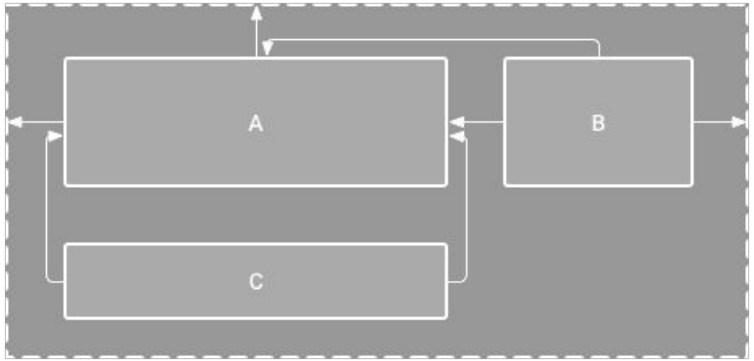
- Layouts anidados
- Múltiples resoluciones
- Constructor de layouts



ConstraintLayout

¿Cómo funciona?

- Constraints
- 2 ejes
- En cada vista un constraint por cada eje (min)



ConstraintLayout

Configuración

build.gradle (app)

```
compile 'com.android.support.constraint:constraint-layout:$version'
```

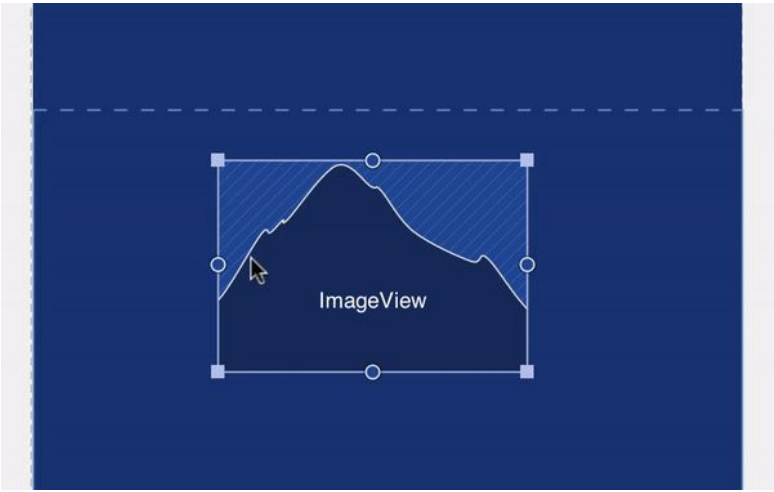
ConstraintLayout

Añadiendo constraints

resize

side

baseline



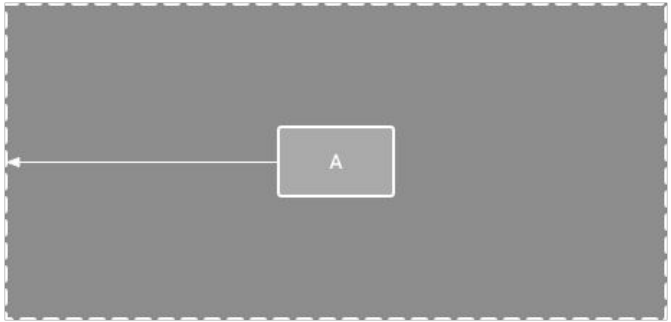
ConstraintLayout

Reglas básicas para añadir constraints

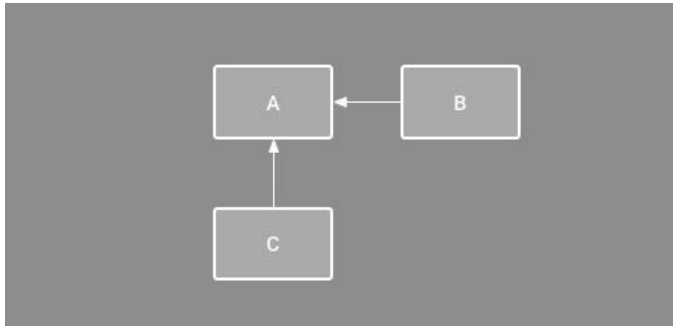
- Todas las vistas deben tener al menos un constraint vertical y un constraint horizontal
- Cada handler solo puede estar asociado a un constraint
- Los constraints solo pueden crearse entre anclas del mismo eje

ConstraintLayout

Tipos de Constraint



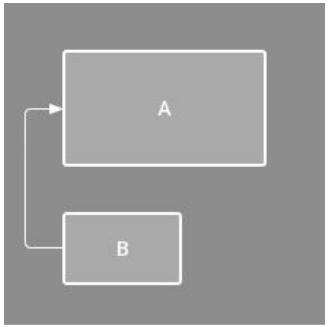
Parent constraint



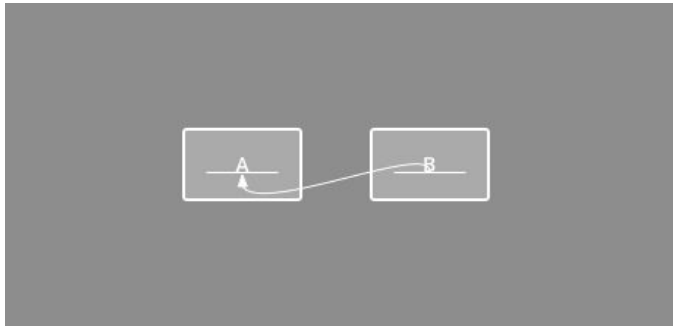
Position constraint

ConstraintLayout

Tipos de Constraint



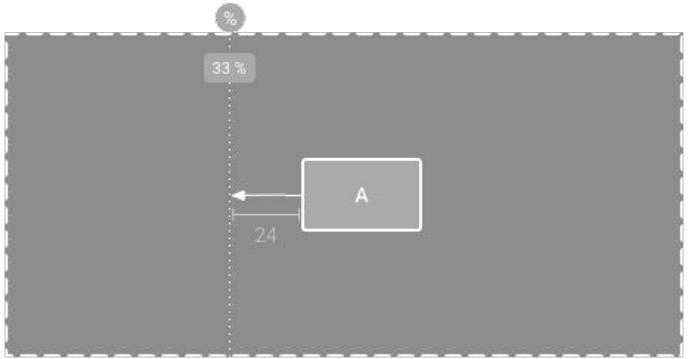
Alignment constraint



Baseline constraint

ConstraintLayout

Tipos de Constraint



Guideline constraint

ConstraintLayout

Ajustes de tamaño

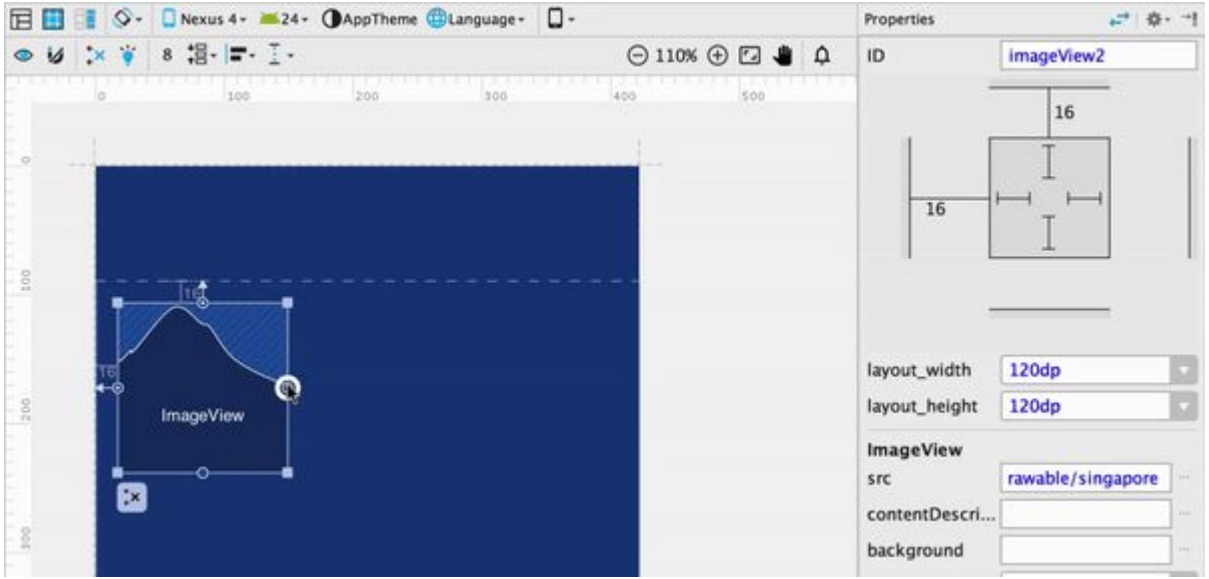
Wrap content

Match Constraints

Fixed

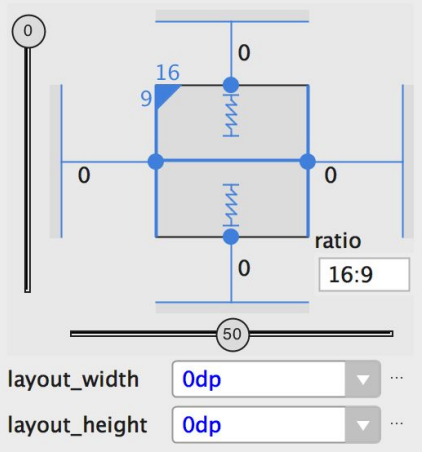
ConstraintLayout

Bias



ConstraintLayout

Aspect ratio

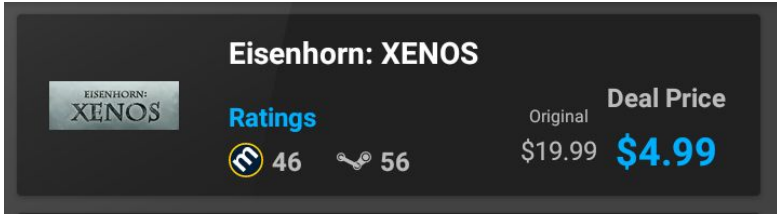


Definición de aspect ratio

`ancho:alto`

Definición de vistas

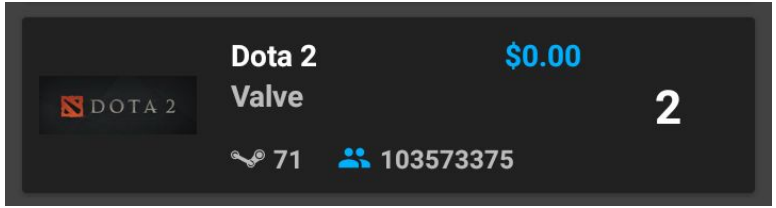
Deal



- CardView
  - ConstraintLayout
    - ImageView:thumb
    - TextView:labelTitle
    - TextView:labelRatings
    - TextView:labelMetacriticScore
    - TextView:labelSteamRating
    - TextView:labelOriginal
    - TextView:labelOriginalPrice
    - TextView:labelPrice
    - TextView:labelDealPrice

Definición de vistas

Top Game



- CardView
  - ConstraintLayout
    - ImageView:thumb
    - TextView:labelTitle
    - TextView:labelPublisher
    - TextView:labelOwners
    - TextView:labelPrice
    - TextView:labelPosition

VectorDrawables

```xml
<?xml version="1.0" encoding="utf-8"?>
<vector xmlns:android="http://schemas.android.com/apk/res/android"
   android:width="8dp"
   android:height="8dp"
   android:viewportHeight="100"
   android:viewportWidth="100">

   <path
       android:fillColor="#100"
       android:pathData="M 0 0 L 100 0 L 100 100 L 0 100 z" />

</vector>
```

M : Move to
L : Line to
z : Close path

Definición de modelos

Data classes

```
data class Deal(var title: String,var salePrice: Float)
```

- Getters
- Setters
- toString
- hasCode/equals
- copy

Definición de modelos

Data classes - copy

```
val superDeal = Deal("SuperGame", 11.0F)
superDeal.copy(salePrice = 10.9F)
```

Definición de modelos

Deconstrucción (Pattern Matching)

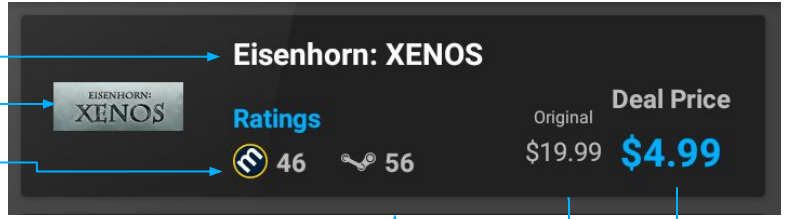
```
val superDeal = Deal("SuperGame", 11.0F)
val (title, price) = superDeal
```

RecyclerView.Adapter + DataBinding

No más adaptadores

```
onBindViewHolder()
```

```
data class Deal(var title: String,
                var salePrice: Float,
                var normalPrice: Float,
                var metacriticScore: Int,
                var steamRating: Int,
                var thumb: String)
```



DataBinding

The basics

```
@{user.firstName}
```

```
data class User(val firstName: String,
                val lastName: String)
```

```
val binding = DataBindingUtil.setContentView(this,
                                R.layout.main_activity)
val user = User("Test", "User")
binding.setUser(user)
```

```
<?xml version="1.0" encoding="utf-8"?>
<layout
   xmlns:android="http://schemas.android.com/apk/res/android">
   <data>
       <variable name="user" type="com.example.User"/>
   </data>
   <LinearLayout
       android:orientation="vertical"
       android:layout_width="match_parent"
       android:layout_height="match_parent">
       <TextView android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:text="@{user.firstName}"/>
       <TextView android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:text="@{user.lastName}"/>
   </LinearLayout>
</layout>
```

DataBinding

Listeners y Handlers

```
<layout
xmlns:android="http://schemas.android.com/apk/res/android">
<data>
   <variable name="handlers" type="com.example.Handlers"/>
   <variable name="user" type="com.example.User"/>
</data>
<LinearLayout
   android:orientation="vertical"
   android:layout_width="match_parent"
   android:layout_height="match_parent">
   <TextView android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="@{user.firstName}"
       android:onClick="@{handlers::onClickFriend}"/>
</LinearLayout>
</layout>
```

```
class MyHandlers {
   fun onClickFriend(view: View) { ... }
}
```

DataBinding

Listeners y Handlers

```
<layout
xmlns:android="http://schemas.android.com/apk/res/android">
<data>
   <variable name="presenter" type="com.example.Presenter"/>
   <variable name="user" type="com.example.User"/>
</data>
<LinearLayout
   android:orientation="vertical"
   android:layout_width="match_parent"
   android:layout_height="match_parent">
   <TextView android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="@{user.firstName}"
       android:onClick="@{() -> showUserProfile(user)}"/>
</LinearLayout>
</layout>
```

```
class Presenter {
   fun showUserProfile(user: User) { ... }
}
```

DataBinding

Setup

build.gradle (commons)
build.gradle (app)

```
dataBinding {
    enabled = true
}
```

```
apply plugin: 'kotlin-kapt'

kapt 'com.android.databinding:compiler:2.3.0'
```

Abstracción de ViewHolder

```kotlin
class DealViewHolder(view: View) :
RecyclerView.ViewHolder(view){
    fun bindItem(deal: Deal){
        //...
    }
}
```

```kotlin
class TopGameViewHolder(view: View) :
RecyclerView.ViewHolder(view){
    fun bindItem(topGame: topGame){
        //...
    }
}
```

```kotlin
class DataBindingViewHolder<MODEL>(view:
View) : RecyclerView.ViewHolder(view){
    fun bindItem(model: MODEL){
        //...
    }
}
```

Abstracción de Adapter

```
class DealAdapter: RecyclerView.Adapter<DataBindingViewHolder<Deal>>() {
   val items: MutableList<Deal> = mutableListOf()

}
```

```
class DataBindingRecyclerAdapter<MODEL>: RecyclerView.Adapter<DataBindingViewHolder<MODEL>>() {
   val items: MutableList<MODEL> = mutableListOf()

}
```

Abstracción de Adapter

```
override fun onCreateViewHolder(parent: ViewGroup?, viewType: Int): DataBindingViewHolder<Deal> {
      val view = parent.inflate(R.layout.item_deal)
      return DataBindingViewHolder(view)
   }

class DataBindingRecyclerAdapter<MODEL>(val viewItemResId: Int):
                RecyclerView.Adapter<DataBindingViewHolder<MODEL>>() {

 override fun onCreateViewHolder(parent: ViewGroup?, viewType: Int): DataBindingViewHolder<Deal> {
      val view = parent.inflate(viewItemResId)
      return DataBindingViewHolder(view)
   }
}
```

Abstracción de Adapter

```
override fun onBindViewHolder(holder: DataBindingViewHolder<Deal>, position: Int) {
      val item = items[position]
      holder.bindItem(item)
   }
```



```
override fun onBindViewHolder(holder: DataBindingViewHolder<MODEL>, position: Int) {
      val item = items[position]
      holder.bindItem(item)
   }
```

Abstracción de ViewHolder

Binding dinámico

```
val binding = DataBindingUtil.setContentView(this,
                                R.layout.main_activity)
val user = User("Test", "User")
binding.setUser(user)
```

```
val binding: ViewDataBinding =
      DataBindingUtil.inflate(LayoutInflater.from(parent.context),
            R.layout.main_activity,
            parent,
            false)

binding.setVariable(BR.User, user)
binding.executePendingBindings()
```

DataBinding

BindingAdapters

```
@BindingAdapter("imageUrl")
fun loadImage(imageView: ImageView, url: String)
{
   Glide.with(imageView)
           .load(url)
           .into(imageView)
}
```

```
app:imageUrl="@{deal.thumb}"
```

```
compile 'com.github.bumptech.glide:glide:$version'
```

Gangame.SDK

Product flavors

mock

Peticiones dummy

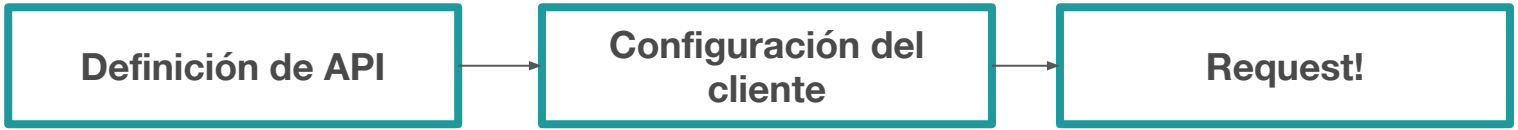
prod

Peticiones al servidor

```
publishNonDefault true
productFlavors {
   prod{}
   mock{}
}
```

Retrofit

Funcionamiento de Retrofit



Definición de API

Retrofit

@GET
@POST
@PUT

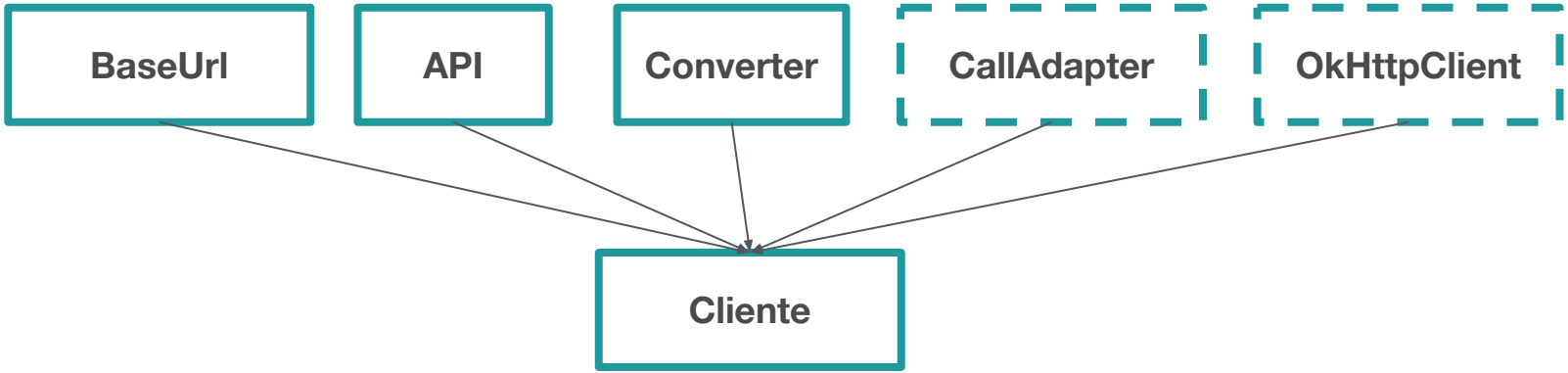
https://steamspy.com/api.php?request=topOwned

```
interface RetrofitGangameApi {

  @GET(Routes.GET_TOP_100_GAMES)
  fun getTop100Games():
  Call<ArrayList<TopGame>>

  @GET(Routes.GET_MOST_OWNED_GAMES)
  fun getMostOwnedGames():
  Call<ArrayList<TopGame>>

  @GET(Routes.GET_DEALS)
  fun getDeals(): Call<ArrayList<Deal>>
}
```

```
compile 'com.squareup.retrofit2:retrofit:$version'
```

Configuración del cliente

Retrofit



Configuración del cliente

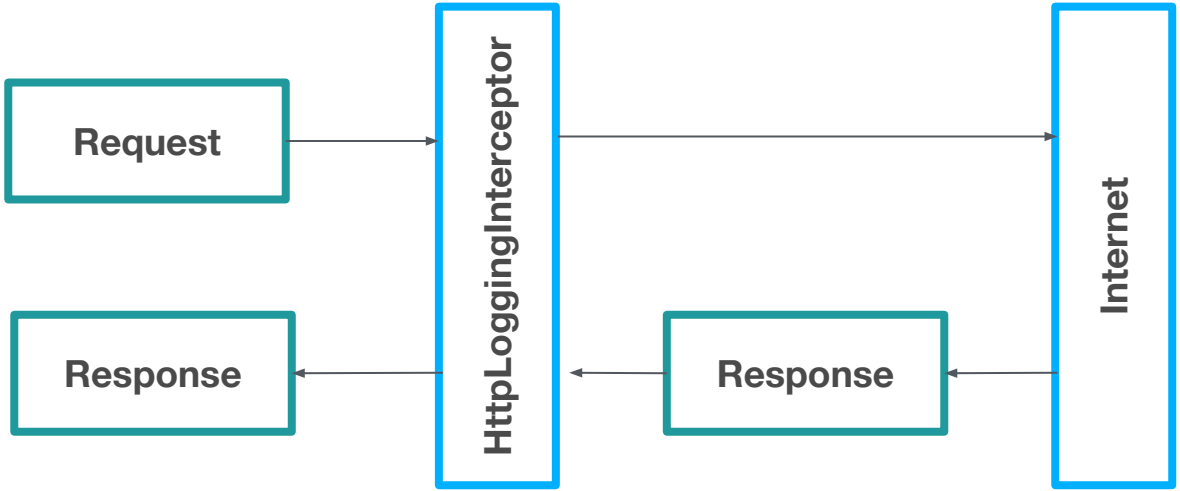
Converter / Gson

```
{
    "appid":730,
    "name":"Counter-Strike",
    "developer":"Valve",
    "publisher":"Valve",
    "score_rank":77,
    "owners":30404242
}
```

```
data class TopGame(@SerializedName("name") val title: String,
                   val publisher: String,
                   @SerializedName("score_rank") val steamRating: Int,
                   val owners: Int,
                   val price: Float,
                   val thumb: String)
```

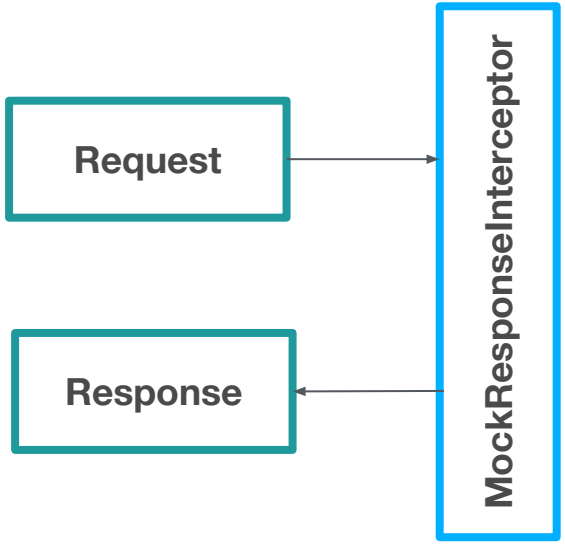
Mock responses

OkHttpInterceptors



Mock responses

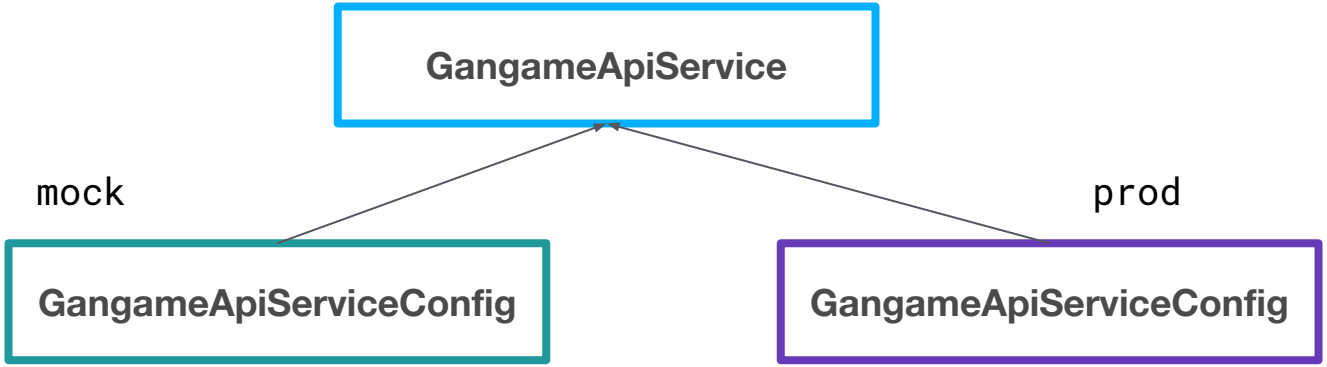
MockResponseInterceptor




```
fun getResponseFor(url: String): String
```

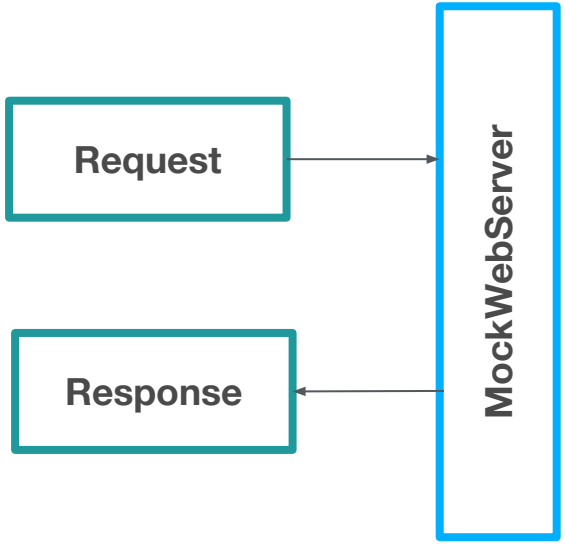
Configuración de cliente por flavor

MockResponseInterceptor



Testing para flavor prod

okhttp MockWebServer




```
val mockWebServer = MockWebServer()
mockWebServer.enqueue(MockResponses
                .dealsSuccessResponse())
```

```
fun dealsSuccessResponse(): MockResponse =
            MockResponse()
            .setBody(DEALS_RESPONSE)
            .setResponseCode(200)
```

Llamadas a API

Callbacks

```
gangameApiService.apiClient.getDeals()
      .enqueue(object : Callback<ArrayList<Deal>> {
   override fun onFailure(call: Call<ArrayList<Deal>>?, t: Throwable?) {
   }

   override fun onResponse(call: Call<ArrayList<Deal>>?,
                           response: Response<ArrayList<Deal>>?) {
   }

})
```

Llamadas a API

RxJava

```
gangameApiService.apiClient.getDealsObservable()
        .observeOn(AndroidSchedulers.mainThread())
        .subscribeOn(Schedulers.io())
```

RxJava

¿Qué es?

```
gangameApiService.apiClient.getDealsObservable()
        .observeOn(AndroidSchedulers.mainThread())
        .subscribeOn(Schedulers.io())
```

RxJava

Observables

```
gangameApiService.apiClient.getDealsObservable()
        .observeOn(AndroidSchedulers.mainThread())
        .subscribeOn(Schedulers.io())
```

RxJava

Observers

```
gangameApiService.apiClient.getDealsObservable()
        .observeOn(AndroidSchedulers.mainThread())
        .subscribeOn(Schedulers.io())
```

RxJava

Operators

```
gangameApiService.apiClient.getDealsObservable()
        .observeOn(AndroidSchedulers.mainThread())
        .subscribeOn(Schedulers.io())
```