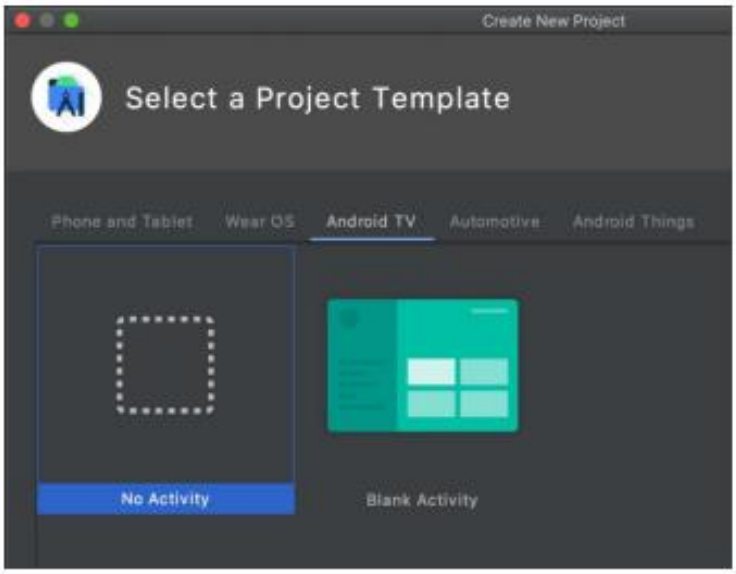
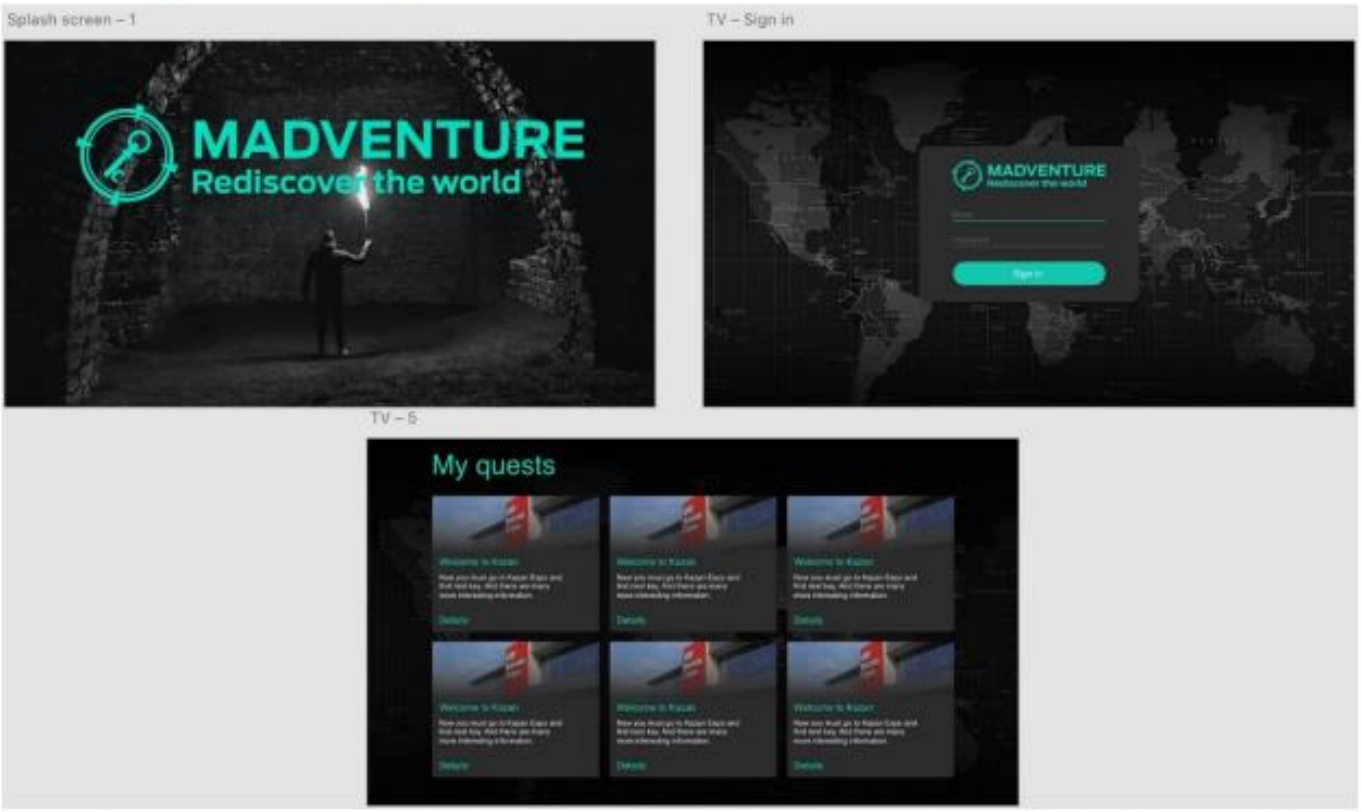
Для создания приложения на телевизор нам необходимо выбрать Android TV –> **No Activity**. И заполнить поля аналогично прошлым проектам.



Для реализации приложения мы будем ориентироваться на макет



Для начала необходимо создать новый файл под именем MainActivity, который будет содержать разметку для нашей Activity. Код разметки мы изменим позже.

```
import ...

class MainActivity : Activity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

Объявление Activity в файле манифеста приложения

Если вы попытаетесь запустить приложение на данном этапе, то оно естественно не запустится, так как в файле AndroidManifest.xml у нас не объявлено ни одной Activity.

В файл манифеста нужно добавить следующий код.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.myty">

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

    <uses-feature
        android:name="android.hardware.touchscreen"
        android:required="false" />
    <uses-feature
        android:name="android.software.leanback"
        android:required="true" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="MyTv"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyTv"
        android:banner="TODO">
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"

            android:screenOrientation="landscape">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LEANBACK_LAUNCHER" />
            </intent-filter>
        </activity>

    </application>

</manifest>
```

Теперь разберем некоторые моменты. Часть кода приведенная ниже отключает тач.

```
<uses-feature
   android:name="android.hardware.touchscreen"
   android:required="false" />
```

В этой части вы указываете, что приложение должно запускаться только на Android TV. Если вы разрабатываете приложение не только для TV, то вам следует установить значение false.

```
<uses-feature
   android:name="android.software.leanback"
   android:required="true" />
```

Объявляем созданную нами активити:

```
 <activity
        android:name=".MainActivity"
        android:label="@string/app_name"
        android:screenOrientation="landscape">
        <intent-filter>
           <action android:name="android.intent.action.MAIN" />
           <category android:name="android.intent.category.LEANBACK_LAUNCHER" />
        </intent-filter>
     </activity>
  </application>
```

При объявлении Activity мы указываем в intent-filter, что Activity должна запускаться на Android TV.

```
<category android:name="android.intent.category.LEANBACK_LAUNCHER/>
```

Так же в поле android:banner загружаем картинку логотипа из макета. Для этого создаем папку drawable и добавляем туда картинки.

После объявления баннера необходимо создать файл colors и заполнить всеми необходимыми нам цветами.

Используем созданную нами активность, как загрузочный экран. Сделаем верстку макета, а затем пропишем логику этого экрана. Изменим устройство для отображения макета на TV (1080). В качестве заднего фона будем использовать изображение.

Пример верстки загрузочного экрана:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout 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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background"
    tools:context=".MainActivity">
<ImageView
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="150dp"
    android:layout_marginEnd="150dp"
    android:layout_marginTop="50dp"
    android:src="@drawable/biglogo"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Как и на часах на телевизоре при создании новой активности необходимо изменять класс активности. Это действие необходимо повторять каждый раз при создании новой активности для телевизора.

```
class MainActivity : Activity()
```

Создадим новую активность и под нее добавим таймер с переходом, аналогично прошлым проектам.

```
class MainActivity : Activity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val timer = object: CountDownTimer( millisInFuture: 3000, countDownInterval: 1000){
            override fun onTick(millisUntilFinished: Long) {

            }

            override fun onFinish() {
                val intent = Intent( packageContext: this@MainActivity,SignInActivity::class.java)
                startActivity(intent)
                finish()
            }
        }
        timer.start()
}
```

Теперь сделаем верстку для нашего нового экрана. Она будет немного сложнее чем в прошлых работах, потому что само окно авторизации должно лежать в отдельном блоке. Т.е внутрь одного из видов разметки мы должны вложить еще один блок, мы будем использовать LinearLayout.

Результат реализации окна авторизации

```
<androidx.constraintlayout.widget.ConstraintLayout 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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background3"
    tools:context=".SignInActivity">
<LinearLayout
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:orientation="vertical"
    android:layout_marginTop="120dp"
    android:layout_marginEnd="300dp"
    android:layout_marginStart="300dp"
    android:layout_marginBottom="120dp"
    android:paddingStart="50dp"
    android:paddingEnd="50dp"
    android:paddingTop="20dp"
    android:paddingBottom="20dp"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    android:background="@drawable/back_login"
    >

    <ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:adjustViewBounds="true"
    android:src="@drawable/biglogo"/>

    <EditText
        android:id="@+id/email"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:backgroundTint="@drawable/edit"
        android:layout_marginTop="10dp"
        android:textCursorDrawable="@drawable/coursor"
        android:hint="Email"
        android:inputType="textEmailAddress"
        android:textColor="@color/aquamarine"
        android:textColorHighlight="@color/aquamarine"
        android:textColorHint="@color/hint" />
    <EditText
        android:id="@+id/password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:backgroundTint="@drawable/edit"
        android:layout_marginTop="10dp"
        android:textCursorDrawable="@drawable/coursor"
        android:hint="Password"
        android:inputType="textPassword"
        android:textColor="@color/aquamarine"
        android:textColorHighlight="@color/aquamarine"
        android:textColorHint="@color/hint" />
```

```
    <Button
        android:id="@+id/login"
        android:background="@drawable/bt"
        android:layout_width="match_parent"
        android:onClick="Login"
        android:layout_marginTop="30dp"
        android:textAllCaps="false"
        android:textSize="18sp"
        android:layout_height="wrap_content"
        android:text="Sign in" />

</LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Нам необходимо создать 4 стиля под наши элементы.

- Стили для EditText: смена цвета курсора и подчеркивания при использовании поля.
- Стиль для заднего фона поля ввода.
- Стиль для кнопки.

Стиль для смены цвета курсора у EditText должен выглядеть следующим образом:

```
<shape xmlns:android="http://schemas.android.com/apk/res/android">
 <size android:width="1dp"/>
 <solid android:color="@color/aquamarine"/>
</shape>
```

<size android:width="1dp"/> – выставление ширины курсора.

<solid android:color = "@color/aquamarine"/> – заливка курсора выбранным цветом

Стиль для изменения подчеркивания при выборе нужного поля:

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">
 <item android:state_focused="true" android:color = "@color/aquamarine"/>
    <item android:color = "@color/hint"/>
</selector>
```

<item android:state_focused = "true" android:color="@color/aquamarine" /> – при выборе этого поля подчеркивание будет изменять свой цвет.

<item android:color = "@color/hint"/> – стандартный цвет поля

Стиль для изменения заднего фона:

```
<shape xmlns:android="http://schemas.android.com/apk/res/android" android:shape="rectangle">
 <corners android:radius="30dp" />
    <solid android:color="@color/gray"/>
</shape>
```

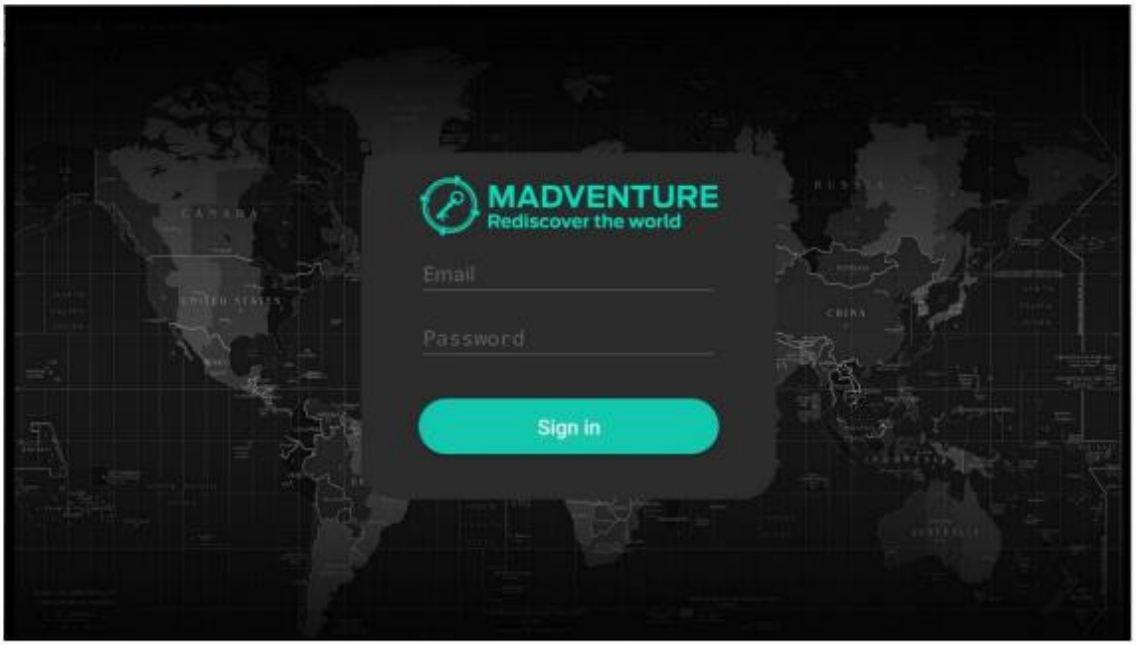
Подобные стили мы с вами делали ранее: закругляем углы объекта и заливаем его необходимым цветом. Такую же операцию производим с кнопкой, но заливаем другим цветом.

Логика экрана входа будет идентична прошлым работам, находим поля для ввода текста и проверяем их на пустоту:

```
import ...
    lateinit var email:EditText
    lateinit var password:EditText
class SignInActivity : Activity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_sign_in)
        email = findViewById(R.id.email)
        password = findViewById(R.id.password)
    }

    fun Login(view: View) {
        if (email.text.toString().isNotEmpty() && password.text.toString().isNotEmpty())
        {
            val intent = Intent( packageContext: this,QuestsActivity::class.java)
            startActivity(intent)
        }
        else
        {
            val alert = AlertDialog.Builder( context: this)
                .setTitle("Error")
                .setMessage("У вас есть пустые поля")
                .setPositiveButton( text: "Ok",  listener: null)
                .create()
                .show()
        }
    }
}
```

Результат верстки экрана авторизации



Сделаем верстку для следующего экрана, он будет состоять у нас из ресайклера и заголовка

```
<androidx.constraintlayout.widget.ConstraintLayout 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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background3"
    tools:context=".QuestsActivity">
    <View
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        android:background="@color/black"
        android:alpha="0.5"/>

    <TextView
        android:id="@+id/texts"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:fontFamily="sans-serif-medium"
        android:layout_marginTop="20dp"
        android:layout_marginStart="50dp"
        android:text="My quests"
        android:textColor="@color/aquamarine"
        android:textSize="50sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclers"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginTop="20dp"
        android:layout_marginStart="50dp"
        android:layout_marginEnd="50dp"
        android:layout_marginBottom="50dp"
        app:layout_constraintTop_toBottomOf="@id/texts"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Создадим адаптер для нашего ресайклера, но для начала необходимо создать градиент, который в дальнейшем мы будем использовать на картинке.

```
<shape xmlns:android="http://schemas.android.com/apk/res/android">
<gradient android:startColor="@color/gray" android:endColor="#00000000"  android:angle="270"/>
</shape>
```

Верстка адаптера элементов списка для ресайклера.

```xml
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_marginRight="10dp"
    android:layout_marginBottom="10dp"
    android:background="@drawable/back_card"
    android:layout_height="wrap_content"
    xmlns:app="http://schemas.android.com/apk/res-auto">
<ImageView
    android:id="@+id/image_quests"
    android:adjustViewBounds="true"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:src="@drawable/kazan"
    android:background="@drawable/image_gradient"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    />
    <TextView
        android:id="@+id/title_quests"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:layout_marginStart="20dp"
        android:textColor="@color/aquamarine"
        android:textSize="20sp"
        android:fontFamily="sans-serif-medium"
        app:layout_constraintTop_toBottomOf="@id/image_quests"/>
    <TextView
        android:id="@+id/descr"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textColor="@color/white"
        android:textSize="20sp"
        android:layout_marginTop="10dp"
        android:layout_marginStart="20dp"
        android:layout_marginEnd="20dp"
        app:layout_constraintTop_toBottomOf="@id/title_quests"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintBottom_toTopOf="@id/bt_details"/>
    <TextView
        android:id="@+id/bt_details"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Details"
        android:textSize="20sp"
        android:textColor="@color/aquamarine"
        android:layout_marginStart="20dp"
        android:layout_marginBottom="20dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Создадим массив с данными, которые будем использовать в нашем адаптере:

```kotlin
data class Quests(val image:Int,val title:String,val text:String)
class MyObj(val list = arrayListOf(Quests(R.drawable.kazan, title: "Welcome to Kazan", text: "Now you must go to Kazan Expo and find next key. And there are many more interesting information."),
    Quests(R.drawable.kazan, title: "Welcome to Kazan", text: "Now you must go to Kazan Expo and find next key. And there are many more interesting information."),
    Quests(R.drawable.kazan, title: "Welcome to Kazan", text: "Now you must go to Kazan Expo and find next key. And there are many more interesting information."),
    Quests(R.drawable.kazan, title: "Welcome to Kazan", text: "Now you must go to Kazan Expo and find next key. And there are many more interesting information."),
    Quests(R.drawable.kazan, title: "Welcome to Kazan", text: "Now you must go to Kazan Expo and find next key. And there are many more interesting information."),
    Quests(R.drawable.kazan, title: "Welcome to Kazan", text: "Now you must go to Kazan Expo and find next key. And there are many more interesting information.")))
```

После создания массива необходимо создать адаптер для соединения элементов списка и данных ресайкелера:

```kotlin
class QuestRecycler(val context: Context,val list: ArrayList<Quests>) : RecyclerView.Adapter<QuestRecycler.MyVH> () {


    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): QuestRecycler.MyVH {
        val root = LayoutInflater.from(context).inflate(R.layout.quests_adapter,parent, attachToRoot: false)
        return  MyVH(root)
    }
    class MyVH(val itemView: View): RecyclerView.ViewHolder(itemView) {
        val imageView:ImageView = itemView.findViewById(R.id.image_quests)
        val title:TextView = itemView.findViewById(R.id.title_quests)
        val descr:TextView = itemView.findViewById(R.id.descr)
    }
    override fun onBindViewHolder(holder: QuestRecycler.MyVH, position: Int) {
        holder.imageView.setImageResource(list[position].image)
        holder.title.setText(list[position].title)
        holder.descr.setText(list[position].text)
    }

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

Подключим наш адаптер к файлу, в котором у нас будет находиться ресайклер:

```
class QuestsActivity : Activity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_quests)
        val rec: RecyclerView = findViewById(R.id.recyclers)
        rec.layoutManager = GridLayoutManager( context: this, spanCount: 3)
        rec.adapter = QuestRecycler( context: this,MyObj().list)

    }
}
```

В этот раз мы будем использовать GridLayoutManager

Разметка относится к классу android.widget.GridLayout и имеет колонки, ряды, клетки как в TableLayout, но при этом элементы могут гибко настраиваться.

В GridLayout для любого компонента можно указать строку и колонку, и в этом месте таблицы он будет размещён. Указывать элемент для каждой ячейки не понадобится, это нужно делать только для тех ячеек, где действительно должны быть компоненты. Компоненты могут растягиваться на несколько ячеек таблицы. Более того, в одну ячейку можно поместить несколько компонентов.

Менеджер расположения GridBLayout подобно табличному менеджеру устанавливает компоненты в таблицу. Но он дает возможность определять для компонентов разную ширину и высоту колонок и строк таблицы. Фактически GridLayout позволяет получить абсолютно любое расположение компонентов.

Результат верстки экрана.