Министерство науки и высшего образования Российской Федерации

Федерального государственного бюджетное образовательное учреждение высшего образования

«Российской экономический университет имени Г. В. Плеханова»

**МОСКОВСКИЙ ПРИБОРОСТРОИТЕЛЬНЫЙ ТЕХНИКУМ.**

ПРАКТИЧЕСКИЕ РАБОТЫ № 10

ДИСЦИПЛИНА: «РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ»

Тема: Работа с датчиками и создание гироскопа.

Специальность: 09.02.07 Информационные системы и программирование

Квалификация: Программист

Листов:

|  |  |  |
| --- | --- | --- |
| Выполнил студент |  | Проверил преподаватель |
| Группа П50-3-18 |  | \_\_\_\_\_\_\_\_\_\_\_А.О.Лясников |
| Пахомов Даниил Александрович |  | «\_\_\_» \_\_\_\_\_\_\_\_2020 года |

Москва 2021

ПРАКТИЧЕСКАЯ РАБОТА №10

Работа с партером MVVM

Цель работы: научится создавать приложения, которые могут взаимодействовать с партером MVVM и сохранять данные в базу данных после валидации.

1. Build.gradle

|  |
| --- |
| plugins **{** id 'com.android.application'  id 'kotlin-android'  id 'kotlin-android-extensions'  id 'kotlin-kapt'  id 'com.google.gms.google-services' **}** |
| buildFeatures  **{** dataBinding= true  **}** |
| dependencies **{** implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin\_version"  implementation 'androidx.core:core-ktx:1.3.2'  implementation 'androidx.appcompat:appcompat:1.2.0'  implementation 'com.google.android.material:material:1.2.1'  implementation 'androidx.constraintlayout:constraintlayout:2.0.4'  implementation 'com.google.firebase:firebase-auth:20.0.1'  testImplementation 'junit:junit:4.+'  androidTestImplementation 'androidx.test.ext:junit:1.1.2'  androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'   kapt 'com.android.databinding:compiler:4.1.1'  implementation 'android.arch.lifecycle:extensions:1.1.1' **}** |

1. User

|  |
| --- |
| import android.text.TextUtils import android.util.Patterns import androidx.databinding.BaseObservable  class User(private var email: String, private var password: String) : BaseObservable() {  val isDataValid: Boolean  get() = (!TextUtils.isEmpty(email)) && Patterns.*EMAIL\_ADDRESS*.matcher(email)  .matches() && (password.length > 6)   fun getPassword(): String {  return password  }   fun getEmail(): String {  return email  }   fun setEmail(email: String) {  this.email = email  }   fun setPassword(password: String) {  this.password = password  }  } |

1. LoginResultCallBacks

|  |
| --- |
| interface LoginResultCallBacks {  fun onSuccess(message: String)  fun onError(message: String) } |

1. LoginViewModel

|  |
| --- |
| import android.app.Activity import android.text.Editable import android.text.TextWatcher import android.view.View import android.widget.Toast import androidx.appcompat.app.AppCompatActivity import androidx.lifecycle.ViewModel import com.example.mvvmvalidation.LoginResultCallBacks import com.example.mvvmvalidation.MainActivity import com.example.mvvmvalidation.model.DataBaseHelper import com.example.mvvmvalidation.model.User import com.google.firebase.auth.FirebaseAuth  class LoginViewModel(private val listener: LoginResultCallBacks) : ViewModel(){   private val user: User  private lateinit var auth: FirebaseAuth  var DB: DataBaseHelper? = null   init {  this.user = User("", "");  }   val emailTextWatcher: TextWatcher  get() = object : TextWatcher {  override fun beforeTextChanged(p0: CharSequence?, p1: Int, p2: Int, p3: Int) {  }   override fun onTextChanged(p0: CharSequence?, p1: Int, p2: Int, p3: Int) {  }   override fun afterTextChanged(p0: Editable?) {  user.setEmail(p0.*toString*())  }  }   val passwordTextWatcher: TextWatcher  get() = object : TextWatcher {  override fun beforeTextChanged(p0: CharSequence?, p1: Int, p2: Int, p3: Int) {  }   override fun onTextChanged(p0: CharSequence?, p1: Int, p2: Int, p3: Int) {  }   override fun afterTextChanged(p0: Editable?) {  user.setPassword(p0.*toString*())  }  }   fun onLoginClicked(v: View) {  if (user.isDataValid) {  var main = MainActivity()  auth = FirebaseAuth.getInstance()  auth.createUserWithEmailAndPassword(user.getEmail(),user.getPassword()).addOnCompleteListener(main)**{** task **->** if (task.*isSuccessful*){  listener.onSuccess("Пользователь заргистрирован")  }else  {  listener.onError("ошибка Авторизации")  }  **}***// Добавление данных в базу данных firebase* } else {  listener.onError("Ошибка")  }  }  } |

1. LoginViewModelFactory

|  |
| --- |
| import androidx.annotation.MainThread import androidx.lifecycle.ViewModel import androidx.lifecycle.ViewModelProvider import com.example.mvvmvalidation.LoginResultCallBacks import com.example.mvvmvalidation.MainActivity  class LoginViewModelFactory(private val listener: LoginResultCallBacks) :  ViewModelProvider.NewInstanceFactory() {  override fun <T : ViewModel?> create(modelClass: Class<T>): T {  return LoginViewModel(listener)as T  } } |

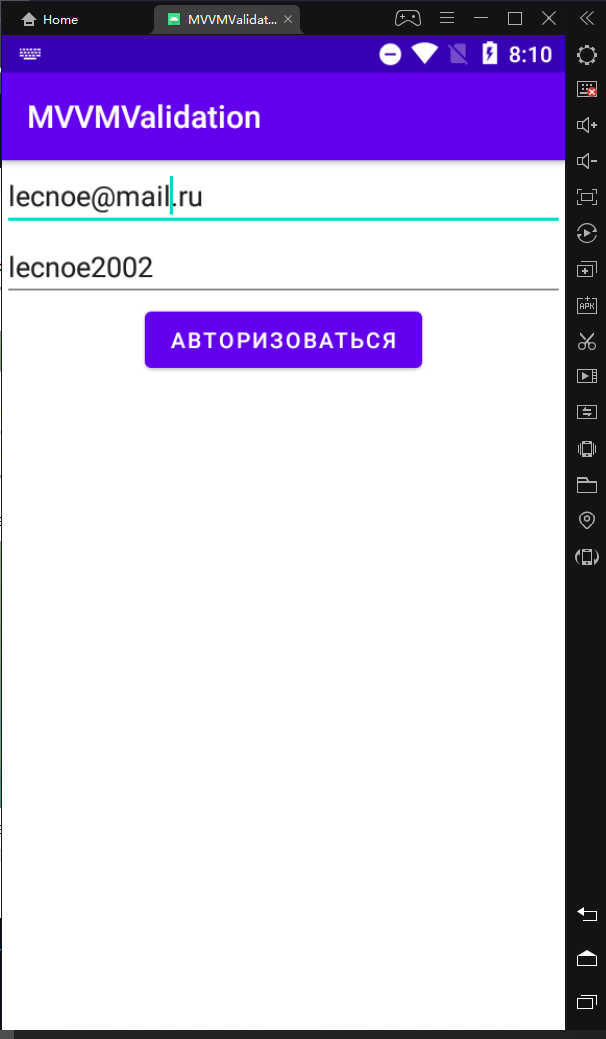
1. MainActivity

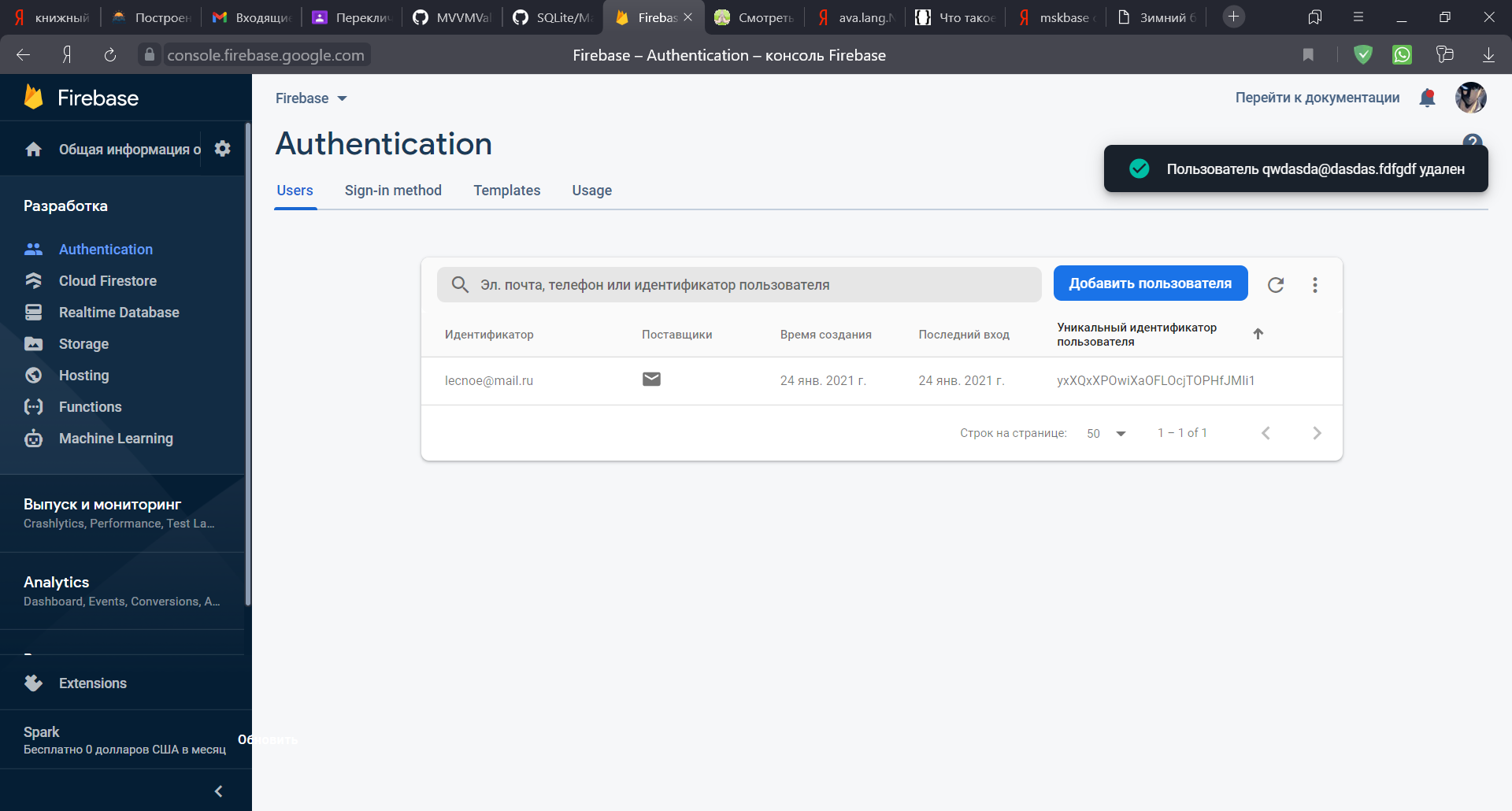
|  |
| --- |
| import androidx.appcompat.app.AppCompatActivity import android.os.Bundle import android.widget.Toast import androidx.databinding.DataBindingUtil import androidx.lifecycle.ViewModelProviders import com.example.mvvmvalidation.databinding.ActivityMainBinding import com.example.mvvmvalidation.viewmodel.LoginViewModel import com.example.mvvmvalidation.viewmodel.LoginViewModelFactory  class MainActivity : AppCompatActivity(),LoginResultCallBacks {  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.*activity\_main*)   val activityMainBinding =  DataBindingUtil.setContentView<ActivityMainBinding>(this, R.layout.*activity\_main*)  activityMainBinding.*viewModel* =  ViewModelProviders.of(this, LoginViewModelFactory(this)).get(LoginViewModel::class.*java*)  }   override fun onSuccess(message: String) {  Toast.makeText(this, message, Toast.*LENGTH\_SHORT*).show()  }   override fun onError(message: String) {  Toast.makeText(this, message, Toast.*LENGTH\_SHORT*).show()  } } |

1. activity\_main

|  |
| --- |
| *<?*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">   <data>   <variable  name="viewModel"  type="com.example.mvvmvalidation.viewmodel.LoginViewModel" />  </data>   <LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="vertical">   <EditText  android:id="@+id/Login"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:hint="Логин"  app:addTextChangedListener="@{viewModel.emailTextWatcher}" />   <EditText  android:id="@+id/Password"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:hint="Пароль"  app:addTextChangedListener="@{viewModel.passwordTextWatcher}" />   <Button  android:id="@+id/btnLogin"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:onClick="@{viewModel.onLoginClicked}"  android:text="Авторизоваться"  android:layout\_gravity="center"/>  </LinearLayout>  </layout> |

1. Результат работы программы





Вывод: научился создавать приложения, которые могут взаимодействовать с партером MVVM и сохранять данные в базу данных после валидации.