ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc96692011)

[ДНЕВНИК ПРАКТИКИ 4](#_Toc96692012)

[ЦЕЛИ И ЗАДАЧИ ПРАКТИКИ 5](#_Toc96692013)

[АНАЛИЗ И ПРОЕКТИРОВАНИЕ ИНТЕРФЕЙСА ПРИЛОЖЕНИЯ 6](#_Toc96692014)

[РЕАЛИЗАЦИЯ ЛОГИКИ ПРИЛОЖЕНИЯ 7](#_Toc96692015)

[ЗАКЛЮЧЕНИЕ 8](#_Toc96692016)

[СПИСОК ЛИТЕРАТУРЫ 9](#_Toc96692017)

# ВВЕДЕНИЕ

Практика ПП.02.02 Разработка модулей доступа к данным проводилась в период с 21.02.2022 г. по 26.02.2022 на базе мастерской «Разработка мобильных приложений» в БПОУ ОО «Омский авиационный колледж имени Н.Е. Жуковского»

Целью учебной практики ПП.02.02 Разработка модулей доступа к данным является комплексное освоение студентами вида профессиональной деятельности «Осуществление интеграции программных модулей», формирование общих и профессиональных компетенций, а также приобретение необходимых умений и опыта практической работы в будущей профессиональной деятельности студентами по специальности 09.02.07 «Информационные системы и программирование».

В период выполнения ПП.02.02 необходимо разработать приложение для работы на устройствах под управлением ОС Android с возможностью обработки запрошенных данных с Удаленной Базы Данных на основании технического задания и макета экранов.

За время практики должны быть выполнены следующие виды работ:

* анализ требований к приложению
* разработка программного кода приложения
* разработка серверной части
* разработка модулей для доступа к данным
* разработка модулей для взаимодействия с памятью устройства
* оформление программного кода в соответствии с установленными требованиями
* формирование итоговой отчетности по проекту, документирование и презентация решений

Результатом практики является готовое приложение, а так же презентационный материал, демонстрирующий процесс создания и функциональные возможности приложения.

# ДНЕВНИК ПРАКТИКИ

|  |  |  |
| --- | --- | --- |
| ***Дата*** | ***Виды выполненных работ*** | ***Подпись руководителя практики*** |
| 23.11.23 | Вводный инструктаж, Проектирование структуры хранения данных |  |
| 24.11.23 | Верстка элементов приложения |  |
| 25.11.23 | Разработка логики работы приложения |  |
| 26.11.23 | Разработка удаленной базы данных |  |
| 27.11.23 | Настройка ролей и безопасности в серверной части приложения |  |
| 28.11.23 | Разработка взаимодействия с серверной частью приложения |  |
| 30.11.23 | Разработка модулей авторизации и регистрации пользователей в базе данных |  |
| 31.11.23 | Осуществление интеграции программных модулей |  |
| 01.10.23 | Разработка модулей взаимодействия с данными пользователя из памяти устройства |  |
| 02.10.23 | Структурирование исходного программного кода |  |
| 03.10.23 | Проверка и отладка программного кода, Документирование и защита проекта. |  |

# ЦЕЛИ И ЗАДАЧИ ПРАКТИКИ

**Реализуйте мобильное приложение для смартфона согласно следующим требованиям:**

* дизайн приложения должен строго следовать макету;
* в работе необходимо использовать систему контроля версий Git, и ежедневно сохранять разработанные элементы проекта;
* проект приложения должен быть структурирован по экранам, то есть исходные файлы конкретного экрана должны быть в соответствующей папке.
* необходимо корректно обрабатывать запросы к базе данных. В случае получения ошибки от сервера или отсутствия соединения с сетью Интернет необходимо отобразить соответствующий текст ошибки в диалоговом окне, которое должно закрываться только пользователем.

Макет приложения доступен по адресу: https://www.figma.com/file/SERP11LP0KA39IKz8sKbB3/SmartHouse?type=design&node-id=0-1&mode=design&t=d73SMbaGMyw3IZ4I-0

Для создания базы данных используйте удаленную СУБД supabase

Ссылка на документацию: Supabase Kotlin Client - Initializing

# АНАЛИЗ И ПРОЕКТИРОВАНИЕ ИНТЕРФЕЙСА ПРИЛОЖЕНИЯ

Окно авторизации в приложении:

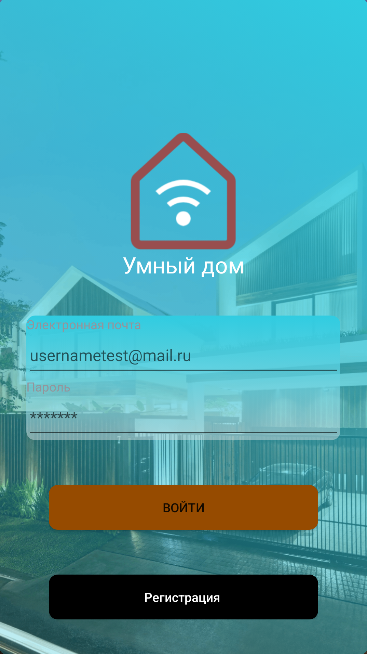


Рисунок 1 – “Окно авторизации”

Листинг подключения базы данных к проекту:

…

val client = createSupabaseClient(  
 supabaseUrl = "https://frwzdxnvfejlwekzzhat.supabase.co",  
 supabaseKey = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6ImZyd3pkeG52ZmVqbHdla3p6aGF0Iiwicm9sZSI6ImFub24iLCJpYXQiOjE2OTgzMTIzMTIsImV4cCI6MjAxMzg4ODMxMn0.Hcftjvx9ruC4rxSZhzjB79iU9sWVOfmkMhp87W4vBiA"  
) {  
 install(GoTrue)  
 install(Postgrest)  
}

Листинг для валидации вводимых данных:

override fun onTextChanged(p0: CharSequence?,  
 p1: Int, p2: Int, p3: Int) {  
 if (p0.isValidEmail()){  
 edit\_email.error = null  
 btnLog.isEnabled=true  
 }else{  
 edit\_email.error = "Введите E-mail."  
 btnLog.isEnabled=false  
 }  
 }

Листинг для авторизации пользователя:  
 lifecycleScope.launch {  
 client.gotrue.loginWith(Email) {  
 email = mail.toString()//"test@mail.ru"//mail.toString()  
 password = pass.toString()//"123456"//pass.toString()  
 }}

Окно регистрации в приложении:

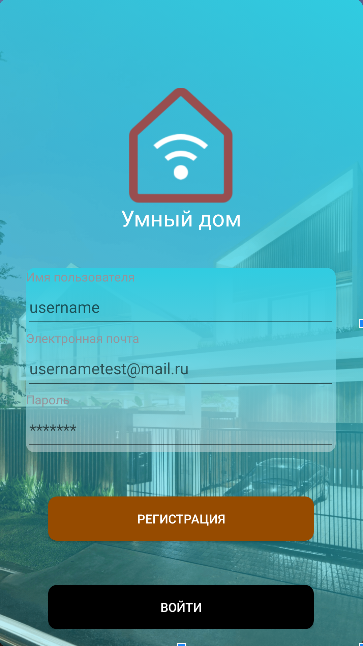


Рисунок 2 – “Окно регистрации”

Листинг окна регистрации:

…

val client = createSupabaseClient(  
 supabaseUrl = "https://frwzdxnvfejlwekzzhat.supabase.co",  
 supabaseKey = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6ImZyd3pkeG52ZmVqbHdla3p6aGF0Iiwicm9sZSI6ImFub24iLCJpYXQiOjE2OTgzMTIzMTIsImV4cCI6MjAxMzg4ODMxMn0.Hcftjvx9ruC4rxSZhzjB79iU9sWVOfmkMhp87W4vBiA" ) {  
 install(GoTrue)  
 install(Postgrest) }  
 email\_edit.addTextChangedListener(object: TextWatcher {  
 override fun beforeTextChanged(p0: CharSequence?,  
 p1: Int, p2: Int, p3: Int) { }  
 override fun onTextChanged(p0: CharSequence?,  
 p1: Int, p2: Int, p3: Int) {  
 if (p0.isValidEmail()){  
 email\_edit.error = null  
 btnReg.isEnabled = true  
 }else{  
 email\_edit.error = "Введите E-mail." }}  
 override fun afterTextChanged(p0: Editable?) {mail= email\_edit.text.toString()  
 }})  
 name\_edit.addTextChangedListener(object: TextWatcher{  
 override fun beforeTextChanged(p0: CharSequence?,  
 p1: Int, p2: Int, p3: Int) {  
 }  
 override fun onTextChanged(p0: CharSequence?,  
 p1: Int, p2: Int, p3: Int) {  
 if (p0 != null) {  
 if (p0.trim().length>0){  
 name\_edit.error = null  
 btnReg.isEnabled = true  
 }else{  
 name\_edit.error = "Нужно ввести пароль."  
 }} }  
 override fun afterTextChanged(p0: Editable?) {  
 name = name\_edit.text.toString()}  
 })  
 btnReg.setOnClickListener(object : View.OnClickListener{  
 override fun onClick(view: View?) {  
 lifecycleScope.launch {  
 val user = client.gotrue.signUpWith(Email) {  
 email = mail.toString()  
 password = pass.toString()  
 }  
 val user\_id = client.gotrue.retrieveUserForCurrentSession(updateSession = true)  
 var profile = profile(user\_id.id.toString(),username=name.toString())  
 client.postgrest["profile"].insert(profile)  
 }  
 val intent1 = Intent(this@Registration\_class, address\_class::class.java)  
 intent.putExtra("name", name)  
 startActivity(intent1)  
 val intent = Intent(applicationContext, PIN\_reg\_class::class.java)  
 startActivity(intent)}})

Окно профиля пользователя:

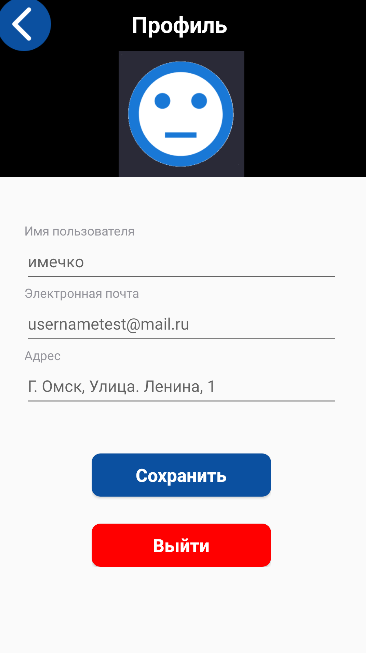


Рисунок 3 – “Окно профиля пользователя”

Листнинг окна профиля пользователя:

…

val client = createSupabaseClient(  
 supabaseUrl = "https://frwzdxnvfejlwekzzhat.supabase.co",  
 supabaseKey = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6ImZyd3pkeG52ZmVqbHdla3p6aGF0Iiwicm9sZSI6ImFub24iLCJpYXQiOjE2OTgzMTIzMTIsImV4cCI6MjAxMzg4ODMxMn0.Hcftjvx9ruC4rxSZhzjB79iU9sWVOfmkMhp87W4vBiA"  
) {  
 install(GoTrue)  
 install(Postgrest)  
}  
lifecycleScope.launch {  
 var username = client.postgrest["profile"].select(){  
 eq("id" , id!!)  
 }  
  
 val buf = StringBuilder()  
 buf.append(username.body.toString()).append("\n");  
 val itemObj: JSONObject = JSONArray(buf.toString()).getJSONObject(0) //TODO Разбитие массива в объект  
 name = itemObj.getString("username")  
 val adress = itemObj.getString("adress")  
 address\_edit.setText(adress.toString())  
 edit\_name.setText(name.toString())  
 edit\_email.setText(email)  
}  
  
edit\_email.addTextChangedListener(object: TextWatcher {  
 override fun beforeTextChanged(p0: CharSequence?,  
 p1: Int, p2: Int, p3: Int) {  
 }  
 override fun onTextChanged(p0: CharSequence?,  
 p1: Int, p2: Int, p3: Int) {  
 if (p0.isValidEmail()){  
 edit\_email.error = null  
 }else{  
 edit\_email.error = "Введите E-mail."  
 }  
 }  
 override fun afterTextChanged(p0: Editable?) {  
 mail= edit\_email.text.toString()  
 }  
})  
address\_edit.addTextChangedListener(object: TextWatcher{  
 override fun beforeTextChanged(p0: CharSequence?,  
 p1: Int, p2: Int, p3: Int) {  
 }  
 override fun onTextChanged(p0: CharSequence?,  
 p1: Int, p2: Int, p3: Int) {  
 if (p0 != null) {  
 if (p0.trim().length>0){  
 address\_edit.error = null  
 }else{  
 address\_edit.error = "Нужно ввести адрес."  
 }  
 }  
 }  
 override fun afterTextChanged(p0: Editable?) {  
 adress\_edit= address\_edit.text.toString()  
 }  
})  
edit\_name.addTextChangedListener(object: TextWatcher{  
 override fun beforeTextChanged(p0: CharSequence?,  
 p1: Int, p2: Int, p3: Int) {  
 }  
 override fun onTextChanged(p0: CharSequence?,  
 p1: Int, p2: Int, p3: Int) {  
 if (p0 != null) {  
 if (p0.trim().length>0){  
 edit\_name.error = null  
 }else{  
 edit\_name.error = "Нужно ввести имя."  
 }  
 }  
 }  
 override fun afterTextChanged(p0: Editable?) {  
 name = edit\_name.text.toString()  
 }  
})  
btn\_save.setOnClickListener(object : View.OnClickListener {  
 override fun onClick(view: View?) {  
 lifecycleScope.launch {  
  
 val user = client.gotrue.retrieveUserForCurrentSession()  
 Log.e("!", user.id)  
 client.postgrest["profile"].update({  
 set("adress",adress\_edit.toString() )  
 })  
 {  
 eq("id", user.id.toString())  
 }  
 client.postgrest["profile"].update({  
 set("username",name.toString() )  
 })  
 {  
 eq("id", user.id.toString())  
 }  
 client.postgrest["auth.users.id"].update({  
 set("email",mail.toString() )  
 })  
 {  
 eq("id", user.id.toString())  
 }  
 intent.putExtra("ID", user.id)  
 intent.putExtra("EMAIL", user.email.toString())  
 startActivity(intent)  
  
 }

Окно добавления адреса:

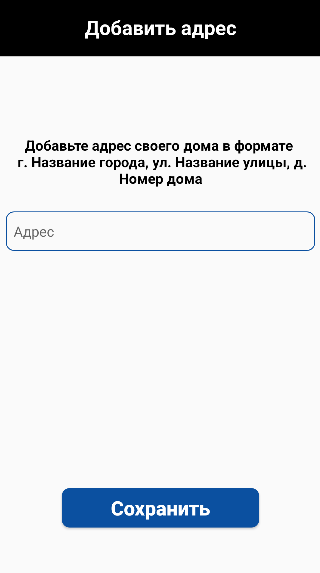


Рисунок 4 – “ Окно добавления адреса ”

Листнинг окна профиля пользователя:

…

val client = createSupabaseClient(  
 supabaseUrl = "https://frwzdxnvfejlwekzzhat.supabase.co",  
 supabaseKey = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6ImZyd3pkeG52ZmVqbHdla3p6aGF0Iiwicm9sZSI6ImFub24iLCJpYXQiOjE2OTgzMTIzMTIsImV4cCI6MjAxMzg4ODMxMn0.Hcftjvx9ruC4rxSZhzjB79iU9sWVOfmkMhp87W4vBiA"  
 ) {  
 install(GoTrue)  
 install(Postgrest)  
 }  
  
 val AddressPattern =  
 Regex("city. " + "[a-zA-Z0-9\\\\+\\\\.\\\\\_\\\\%\\\\-\\\\+]{1,256}" + ", " + "street. " + "[a-zA-Z0-9][a-zA-Z0-9\\\\-]{0,64}" + ", " + "[1-100]")  
 val edit\_address = findViewById(R.id.address\_edit) as EditText  
 edit\_address.addTextChangedListener(object : TextWatcher {  
 override fun beforeTextChanged(  
 p0: CharSequence?,  
 p1: Int, p2: Int, p3: Int  
 ) {  
 }  
  
 override fun onTextChanged(  
 p0: CharSequence?,  
 p1: Int, p2: Int, p3: Int  
 ) {  
 if (p0 != null) {  
 if (p0.matches(AddressPattern)) {  
 edit\_address.error = null  
 } else {  
 edit\_address.error = "Неправильный адрес."}} }  
 override fun afterTextChanged(p0: Editable?) {  
 adress\_p = edit\_address.text.toString()}})  
 btnSave.setOnClickListener(object : View.OnClickListener {  
 override fun onClick(view: View?) {  
 val intent = Intent(applicationContext, room\_available\_Class::class.java)  
  
 lifecycleScope.launch {  
  
 val user = client.gotrue.retrieveUserForCurrentSession()  
 Log.e("!", user.id)  
 client.postgrest["profile"].update({  
 set("adress",adress\_p.toString() )  
 })  
 {  
 eq("id", user.id.toString())}  
 intent.putExtra("ID", user.id)  
 intent.putExtra("EMAIL", user.email.toString())  
 startActivity(intent)}}}}}

# ЗАКЛЮЧЕНИЕ

В ходе выполнение практического задания я научились делать верстку приложения, клиент-серверное взаимодействие приложения, научились взаимодействовать с аппаратными расширениями устройства, производить тестирование, работать с API-серверами, делать парсинг приложения.

# СПИСОК ЛИТЕРАТУРЫ

1. Barry, Burd Android Application Development All–in–One For Dummies® / Barry Burd. - Москва: **Машиностроение**, 2011. - 816 c.

2. Биллиг, В. А. Основы объектного программирования на C# (C# 3.0, Visual Studio 2008) / В.А. Биллиг. - М.: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2010. - 584 c.

3. Гарнаев, Андрей WEB-программирование на Java и JavaScript / Андрей Гарнаев , Сергей Гарнаев. - М.: БХВ-Петербург, **2012. – 179** c.

4. Голощапов, Алексей Google Android. Программирование для мобильных устройств (+ CD-ROM) / Алексей Голощапов. - М.: БХВ-Петербург, 2011. - 438 c.

5. Голощапов, Алексей Google Android. Программирование для мобильных устройств / Алексей Голощапов. - М.: БХВ-Петербург, 2012. - 448 c.

6. Давыдов, Станислав IntelliJ IDEA. Профессиональное программирование на Java. Наиболее полное руководство (+ CD-ROM) / Станислав Давыдов, Алексей Ефимов. - М.: БХВ-Петербург, **2011**. - 800 c.

7. Дэрси, Лорен Android за 24 часа. Программирование приложений под операционную систему Google / Лорен Дэрси, Шейн Кондер. - М.: Рид Групп, 2011. - 464 c.

8. Майер, Рето Android 2. Программирование приложений для планшетных компьютеров и смартфонов / Рето Майер. - М.: "Издательство "Эксмо", 2011. - 672 c.

9. Майер, Рето Android 4. Программирование приложений для планшетных компьютеров и смартфонов / Рето Майер. - М.: Эксмо, 2013. - 816 c.

10. Мартин, К. Соломон Oracle. Программирование на языке Java / Мартин К. Соломон, Нирва Мориссо-Леруа, Джули Басу. - М.: ЛОРИ, 2010. - 512 c.