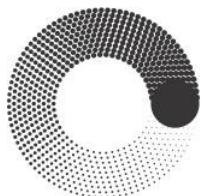


**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

**Факультет информационных технологий
Кафедра Информатики и информационных технологий**

**направление подготовки 09.03.02 «Информационные системы и
технологии»**

ИТОГОВЫЙ ПРОЕКТ

Дисциплина: Программирование для мобильных устройств

Тема: Трекер для подсчета шагов

Выполнил: студент группы 221-3710

Мелихов Владимир Семенович

Проверил:

(Оценка)

Дата, подпись _____

(Дата)

(Подпись)

Замечания:

Москва

2025

Оглавление

Задание	2
1. Система подсчета шагов.....	2
2. Главный экран	4
3. Экран статистики	5
4. Подключение БД Google Firebase.....	5
5. Экран настроек	6
6. Отправка уведомлений	7
7. Верстка горизонтального разрешения	7
8. Активити экрана загрузки	8
9. Результат	8
10. Исходный код приложения:	13

Видео:

<https://drive.google.com/file/d/1qh0mKnNF8sjKC1lm8TmN92NVeX3gQv54/view?usp=sharing>

Задание

В рамках итогового проектирования необходимо разработать в Android Studio на языке Java или Kotlin мобильное Android-приложение.

Проект состоит из нескольких активити, имеет продуманный дизайн и контент, в приложении используются уведомления в строке состояния, осуществляется подключение к серверу БД для извлечения и записи данных. Приложение протестировано на разных версиях Android и на устройствах с разными размерами и разрешением экрана.

1. Система подсчета шагов

Для начала были запрошены все разрешение для корректной работы приложения:

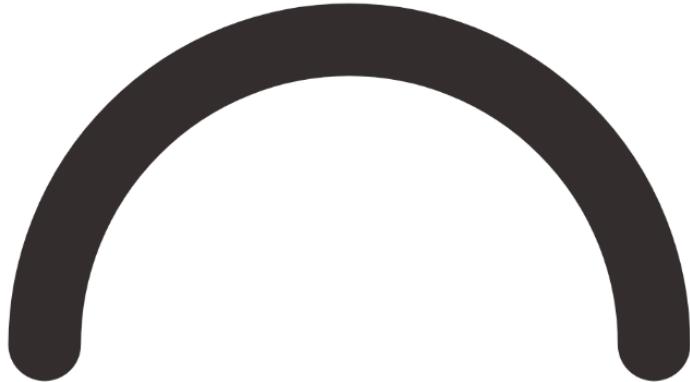
- Отслеживание активности устройства (шагомер)

- Отправка уведомлений
- Синхронизация данных в фоне
- Использование трекинга шагов на фоне

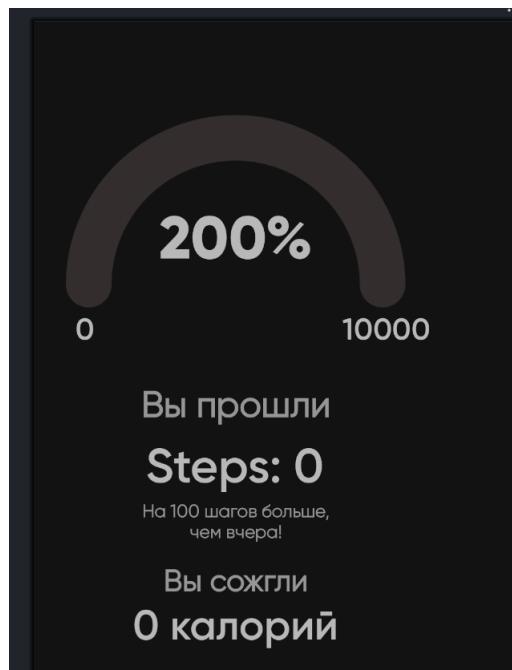
```
<uses-permission android:name="android.permission.ACTIVITY_RECOGNITION" />
<uses-permission android:name="android.permission.POST_NOTIFICATIONS" />
<uses-permission
    android:name="android.permission.FOREGROUND_SERVICE_DATA_SYNC" />
<uses-permission
    android:name="android.permission.FOREGROUND_SERVICE"
    tools:ignore="ForegroundServicesPolicy" />
<uses-permission
    android:name="android.permission.FOREGROUND_SERVICE_LOCATION" />
```

Затем был создан класс StepCounterService, в котором будут отслеживаться все шаги, сделанные пользователем, а также сохраняться в БД. А при достижении определенного порога шагов (50%, 100%), пользователю будет отправлено уведомление об успехе.

Затем был создан при помощи кода закругленный progress bar, который можно гибко менять в дальнейшем.



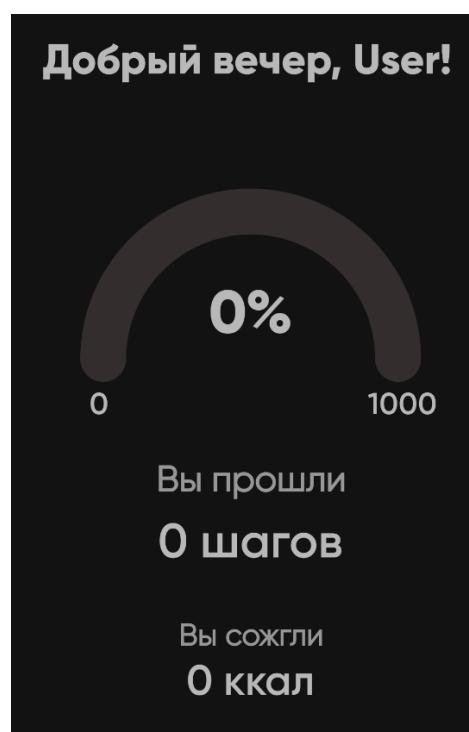
Затем был сверстан фрагмент для красивого отображения статистики по шагам, пройденным за сегодняшний день



2. Главный экран

Главный экран состоит из:

- Приветствия с никнеймом, которое зависит от текущего часа (вечером пишет «Добрый вечер» итп).
- Статистики по шагам, пройденным за сегодняшний день.



3. Экран статистики

Экран статистики состоит из:

- Общей статистики по шагам и калориям
- Удобному графику пройденных шагов за месяц

Для создания графика была использована библиотека библиотека junit, а также кастомный класс для изменения графика под свой дизайн



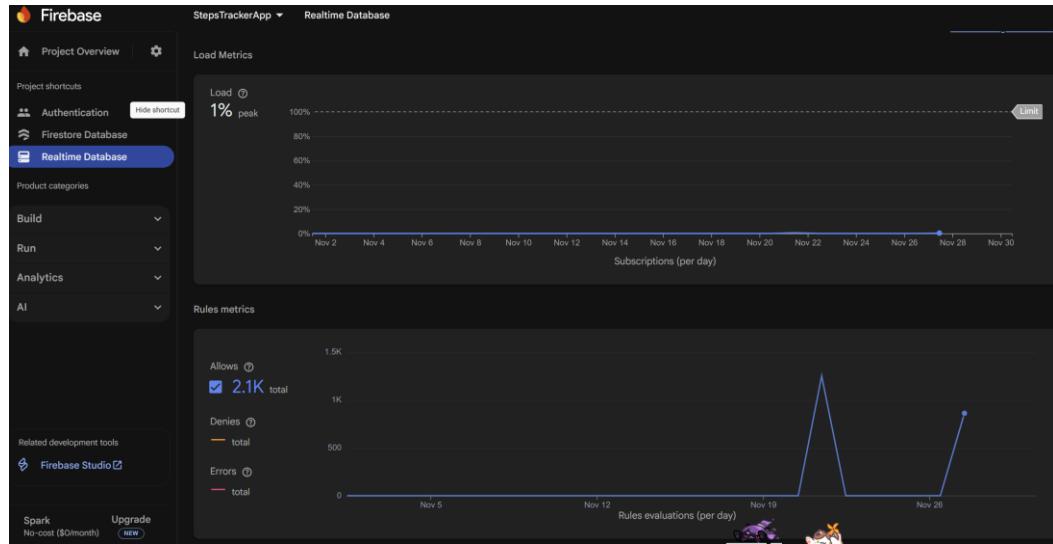
4. Подключение БД Google Firebase

В качестве БД выбрана Google Firebase – наиболее удобная для подключения база данных, которая предоставляет бесплатный сервер.

Был создан класс FirebaseRepository, который отвечает за сохранение и получение данных с сервера

```
fun saveSetting(key: String, value: String) {
    db.child(uid).child("settings").child(key).setValue(value)
}
fun loadSetting(key: String, default: String, callback: (String) -> Unit) {
    db.child(uid).child("settings").child(key)
        .get()
        .addOnSuccessListener {
            callback(it.getValue(String::class.java) ?: default)
        }
}
```

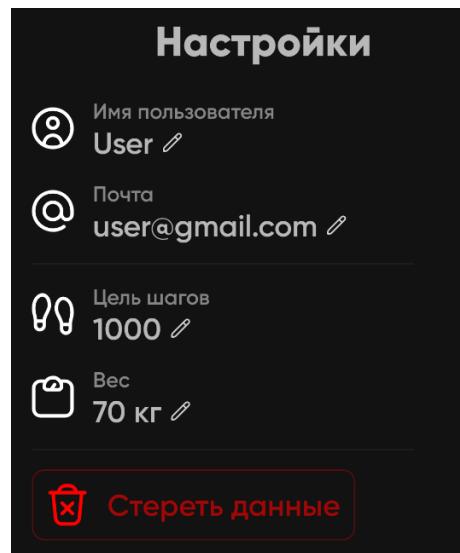
```
        }
    }
fun deleteAllUserData() {
    db.child(uid).removeValue()
}
```



5. Экран настроек

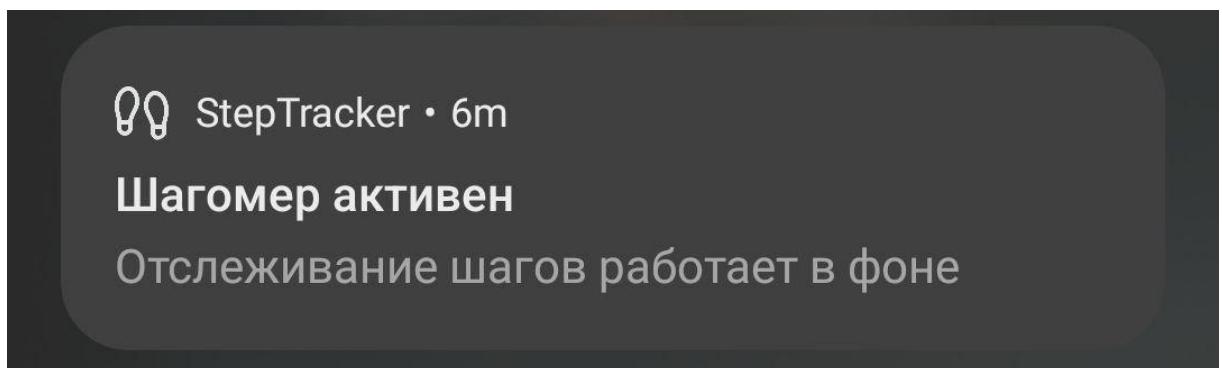
На экране настроек есть:

- Имя пользователя
- Почта
- Цель по шагам за день
- Текущий вес (для подсчета калорий)
- Кнопка стирания данных о пользователе с БД



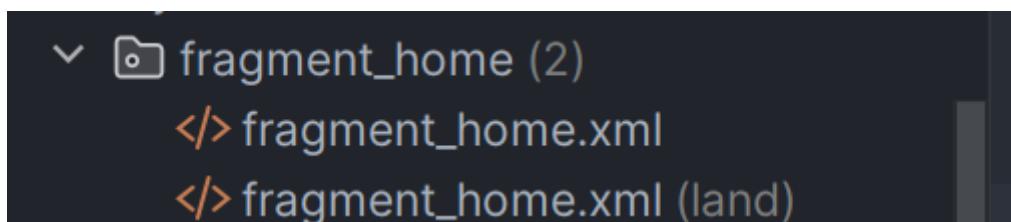
6. Отправка уведомлений

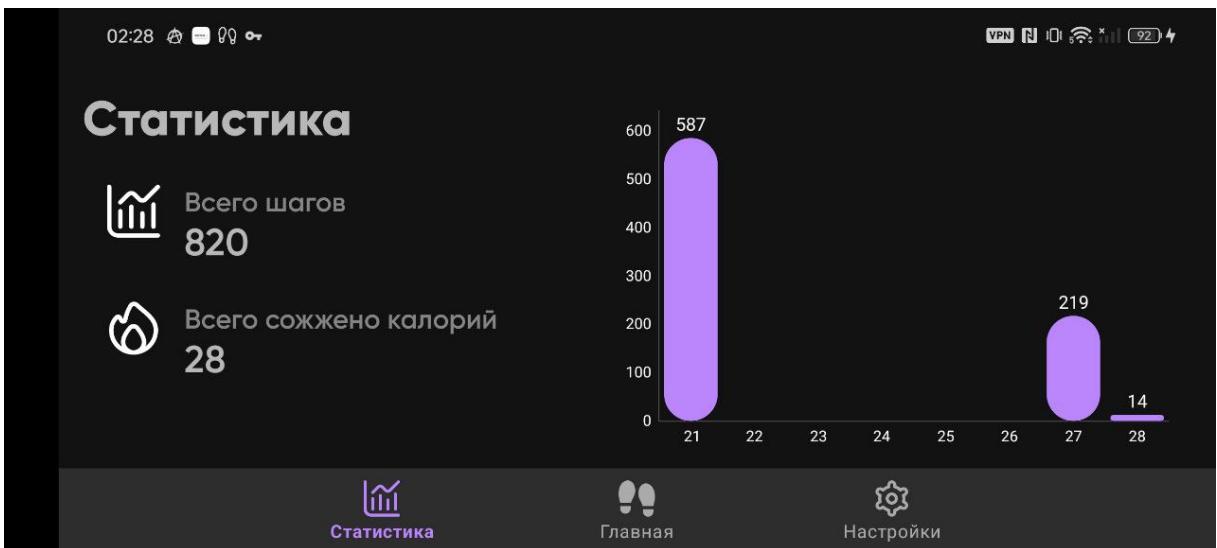
Отправка уведомлений реализована через отдельный класс StepNotifications, который представляет собой сервис, который работает в фоне и отправляет уведомление, когда произошло событие



7. Верстка горизонтального разрешения

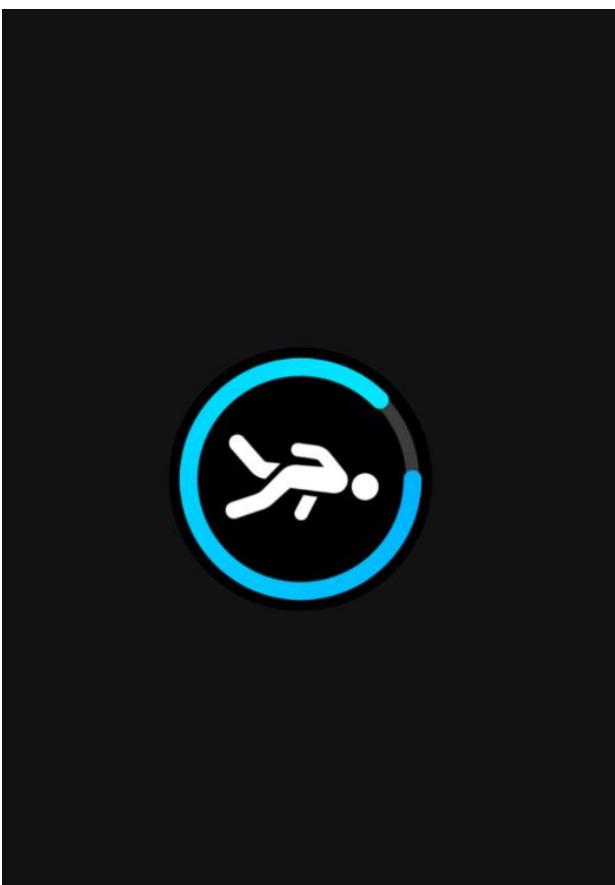
Для каждого экрана, требующего горизонтальную версию, был создан такой же файл в папке land





8. Активити экрана загрузки

Было создано второе активити для экрана загрузки: в центре логотип приложения вращается по кругу



9. Результат

Главный экран

Добрый вечер, User!



Вы прошли
0 шагов

Вы сожгли
0 ккал



Статистика



Главная



Настройки

Настройки

Настройки

Имя пользователя
User 

Почта
user@gmail.com 

Цель шагов
1000 

Вес
70 кг 

 Стереть данные

Статистика

Статистика



Всего шагов

0



Всего сожжено калорий

0

Шаги в этом месяце



10. Исходный код приложения:

Домашний экран

```
package com.example.steptracker.fragments

import android.content.SharedPreferences
import android.icu.util.Calendar
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.fragment.app.Fragment
import com.example.steptracker.R
import com.example.steptracker.data.FirebaseRepository

class HomeFragment : Fragment() {
    private lateinit var sharedpreferences: SharedPreferences

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val root = inflater.inflate(R.layout.fragment_home, container, false)
        val textView = root.findViewById<TextView>(R.id.textView)

        FirebaseRepository.loadSetting("nickname", "User") { nickname ->
            textView.text = sayHelloWithTime(nickname)
        }

        return root
    }

    private fun sayHelloWithTime(nickname: String): String {
        val hour = Calendar.getInstance().get(Calendar.HOUR_OF_DAY)
        val timeOfDay = when (hour) {
            in 6..11 -> "Доброе утро"
            in 12..17 -> "Добрый день"
            in 18..23 -> "Добрый вечер"
            else -> "Доброй ночи"
        }
        return "$timeOfDay, $nickname!"
    }
}
```

Экран настроек

```
package com.example.steptracker.fragments

import android.content.Context
import android.content.SharedPreferences
import android.os.Bundle
import android.view.LayoutInflater
```

```
import android.view.View
import android.view.ViewGroup
import android.widget.EditText
import android.widget.TextView
import androidx.appcompat.app.AlertDialog
import androidx.constraintlayout.widget.ConstraintLayout
import androidx.fragment.app.Fragment
import com.example.steptracker.R
import com.example.steptracker.data.FirebaseRepository
import com.example.steptracker.logic.EventBus

class SettingsFragment : Fragment() {
    private lateinit var nicknameTextView: TextView
    private lateinit var mailTextView: TextView
    private lateinit var stepsTextView: TextView
    private lateinit var weightTextView: TextView

    private lateinit var changeNicknameLayout: ConstraintLayout
    private lateinit var changeMailLayout: ConstraintLayout
    private lateinit var changeStepsLayout: ConstraintLayout
    private lateinit var changeWeightLayout: ConstraintLayout
    private lateinit var removeDataLayout: ConstraintLayout

    private lateinit var sharedPreferences: SharedPreferences

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val root = inflater.inflate(R.layout.fragment_settings, container, false)

        nicknameTextView = root.findViewById(R.id.nicknameTextView)
        mailTextView = root.findViewById(R.id.mailTextView)
        stepsTextView = root.findViewById(R.id.stepsTextView)
        weightTextView = root.findViewById(R.id.weightTextView)

        changeNicknameLayout = root.findViewById(R.id.changeNickname)
        changeMailLayout = root.findViewById(R.id.changeMail)
        changeStepsLayout = root.findViewById(R.id.changeSteps)
        changeWeightLayout = root.findViewById(R.id.changeWeight)
        removeDataLayout = root.findViewById(R.id.removeData)

        sharedpreferences =
            requireContext().getSharedPreferences("StepTrackerPreferences", Context.MODE_PRIVATE)

        changeNicknameLayout.setOnClickListener {
            showEditNameDialog()
        }

        changeMailLayout.setOnClickListener {
            showEditMailDialog()
        }

        changeStepsLayout.setOnClickListener {
            showEditStepsDialog()
        }

        changeWeightLayout.setOnClickListener {
            showEditWeightDialog()
        }
    }
}
```

```
removeDataLayout.setOnClickListener {
    showRemoveDataDialog()
}

loadData()

return root
}

private fun showEditNameDialog() {
    val dialogView =
        LayoutInflater.from(requireContext()).inflate(R.layout.dialog_edit_name, null)
    val editText = dialogView.findViewById<EditText>(R.id.editNameEditText)

    AlertDialog.Builder(requireContext())
        .setTitle("Изменить никнейм")
        .setView(dialogView)
        .setPositiveButton("Сохранить") { _, _ ->
            val newName = editText.text.toString().trim()
            if (newName.isNotEmpty()) {
                nicknameTextView.text = newName
                saveData("nickname", newName)
            }
        }
        .setNegativeButton("Отмена", null)
        .create()
        .show()
}

private fun showEditMailDialog() {
    val dialogView =
        LayoutInflater.from(requireContext()).inflate(R.layout.dialog_edit_mail, null)
    val editText = dialogView.findViewById<EditText>(R.id.editMailEditText)

    AlertDialog.Builder(requireContext())
        .setTitle("Изменить почту")
        .setView(dialogView)
        .setPositiveButton("Сохранить") { _, _ ->
            val newMail = editText.text.toString().trim()
            if (newMail.isNotEmpty() && newMail.contains("@") && newMail.contains(".")) {
                mailTextView.text = newMail
                saveData("mail", newMail)
            } else {
                AlertDialog.Builder(requireContext())
                    .setTitle("Ошибка")
                    .setMessage("Некорректный адрес электронной почты")
                    .setPositiveButton("Ok") { _, _ -> showEditMailDialog() }
                    .create()
                    .show()
            }
        }
        .setNegativeButton("Отмена", null)
        .create()
        .show()
}

private fun showEditStepsDialog() {
    val dialogView =
        LayoutInflater.from(requireContext()).inflate(R.layout.dialog_edit_steps, null)
    val editText = dialogView.findViewById<EditText>(R.id.editStepsEditText)
```

```

AlertDialog.Builder(requireContext())
    .setTitle("Изменить цель")
    .setView(dialogView)
    .setPositiveButton("Сохранить") { _, _ ->
        val newSteps = editText.text.toString().trim()
        if (newSteps.isNotEmpty() && newSteps.toInt() > 0) {
            stepsTextView.text = newSteps
            saveData("stepsGoal", newSteps)

            EventBus.notify("stepsGoalChanged")
        } else {
            AlertDialog.Builder(requireContext())
                .setTitle("Ошибка")
                .setMessage("Некорректное количество шагов")
                .setPositiveButton("Ok") { _, _ -> showEditStepsDialog() }
                .create()
                .show()
        }
    }
    .setNegativeButton("Отмена", null)
    .create()
    .show()
}

private fun showEditWeightDialog() {
    val dialogView =
        LayoutInflater.from(requireContext()).inflate(R.layout.dialog_edit_weight, null)
    val editText = dialogView.findViewById<EditText>(R.id.editWeightEditText)

    AlertDialog.Builder(requireContext())
        .setTitle("Изменить вес")
        .setView(dialogView)
        .setPositiveButton("Сохранить") { _, _ ->
            val newWeight = editText.text.toString().trim()
            if (newWeight.isNotEmpty() && newWeight.toInt() > 0) {
                weightTextView.text = "$newWeight кг"
                saveData("weight", newWeight)
            } else {
                AlertDialog.Builder(requireContext())
                    .setTitle("Ошибка")
                    .setMessage("Некорректный вес")
                    .setPositiveButton("Ok") { _, _ -> showEditWeightDialog() }
                    .create()
                    .show()
            }
        }
        .setNegativeButton("Отмена", null)
        .create()
        .show()
    }

private fun showRemoveDataDialog() {
    AlertDialog.Builder(requireContext())
        .setTitle("Удалить данные")
        .setMessage("Вы уверены?")
        .setPositiveButton("Да") { _, _ ->
            FirebaseRepository.deleteAllUserData()
            loadData()
        }
        .setNegativeButton("Нет", null)
        .show()
}

```

```
}

private fun saveData(key: String, value: String) {
    FirebaseRepository.saveSetting(key, value)
}

private fun loadData() {
    FirebaseRepository.loadAllSettings { nickname, mail, steps, weight ->
        nicknameTextView.text = nickname
        mailTextView.text = mail
        stepsTextView.text = steps
        weightTextView.text = "$weight кг"
    }
}
```

Экран статистики

```
package com.example.steptracker.fragments

import android.graphics.Color
import com.example.steptracker.ui.RoundedBarChartRenderer
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.fragment.app.Fragment
import com.example.steptracker.R
import com.example.steptracker.data.FirebaseRepository
import com.github.mikephil.charting.charts.BarChart
import com.github.mikephil.charting.components.AxisBase
import com.github.mikephil.charting.components.XAxis
import com.github.mikephil.charting.data.BarData
import com.github.mikephil.charting.data.BarDataSet
import com.github.mikephil.charting.data.BarEntry
import com.github.mikephil.charting.formatter.ValueFormatter
import java.util.Calendar

class StatsFragment : Fragment() {

    private lateinit var barChart: BarChart
    private lateinit var totalSteps: TextView
    private lateinit var totalCalories: TextView

    private lateinit var valueFormatter: ValueFormatter
    private lateinit var barEntryFormatter: ValueFormatter

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val root = inflater.inflate(R.layout.fragment_stats, container, false)

        barChart = root.findViewById(R.id.barChart)
        totalSteps = root.findViewById(R.id.totalSteps)
```

```

totalCalories = root.findViewById(R.id.totalCalories)

setupBarChart()
loadHistoryAndUpdateUI()

return root
}

private fun loadHistoryAndUpdateUI() {
    FirebaseRepository.loadStepsHistory { history ->

        val total = history.values.sum()
        totalSteps.text = total.toString()
        totalCalories.text = calculateCalories(total).toInt().toString()

        updateBarChart(history)
    }
}

private fun setupBarChart() {
    barChart.apply {
        description.isEnabled = false
        axisRight.isEnabled = false

        axisLeft.apply {
            axisMinimum = 0f
            setDrawGridLines(false)
            textColor = getTextColorTheme()
        }

        xAxis.apply {
            position = XAxis.XAxisPosition.BOTTOM
            granularity = 1f
            setDrawGridLines(false)
            textColor = getTextColorTheme()
        }

        valueFormatter = object : ValueFormatter() {
            override fun getAxisLabel(value: Float, axis: AxisBase?): String {
                return value.toInt().toString()
            }
        }
    }

    barEntryFormatter = object : ValueFormatter() {
        override fun getBarLabel(e: BarEntry?): String =
            e?.y?.toInt()?.toString()
    }
}

legend.isEnabled = false
renderer = RoundedBarChartRenderer(this, animator, viewPortHandler)
}

private fun updateBarChart(history: Map<String, Int>) {
    val stepsForMonth = filterCurrentMonth(history)

    val entries = stepsForMonth.map { (day, steps) ->
        BarEntry(day.toFloat(), steps.toFloat())
    }

    val dataSet = BarDataSet(entries, "Steps")
    dataSet.color = resources.getColor(R.color.red, null)
}

```

```

        dataSet.valueTextSize = 12f
        dataSet.valueTextColor = getTextColorTheme()
        dataSet.valueFormatter = barEntryFormatter

        barChart.data = BarData(dataSet)
        barChart.invalidate()
    }

private fun filterCurrentMonth(history: Map<String, Int>): Map<Int, Int> {
    val cal = Calendar.getInstance()
    val month = cal.get(Calendar.MONTH)
    val year = cal.get(Calendar.YEAR)
    val today = cal.get(Calendar.DAY_OF_MONTH)

    val result = mutableMapOf<Int, Int>()

    history.forEach { (date, steps) ->
        val parts = date.split("-")
        if (parts.size == 3) {
            val y = parts[0].toInt()
            val m = parts[1].toInt() - 1
            val d = parts[2].toInt()

            if (y == year && m == month) {
                result[d] = steps
            }
        }
    }

    // ↴ Добавляем сегодняшний день, если его нет
    if (!result.containsKey(today)) {
        result[today] = 0
    }
}

return result
}

private fun calculateCalories(steps: Int): Float {
    return steps * 0.035f
}

private fun getTextColorTheme(): Int {
    val isDark = resources.configuration.uiMode and
        android.content.res.Configuration.UI_MODE_NIGHT_MASK ==
        android.content.res.Configuration.UI_MODE_NIGHT_YES

    return if (isDark)
        resources.getColor(R.color.white, null)
    else
        resources.getColor(R.color.black, null)
}

```

Фрагмент прогресс-бара шагов

```
package com.example.steptracker.fragments

import android.content.Context
import android.hardware.Sensor
import android.hardware.SensorEvent
import android.hardware.SensorEventListener
import android.hardware.SensorManager
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import com.example.steptracker.R
import com.example.steptracker.StepNotifications
import com.example.steptracker.data.FirebaseRepository
import com.example.steptracker.databinding.FragmentTotalStepsBinding
import java.text.SimpleDateFormat
import java.util.*

class TotalStepsFragment : Fragment(), SensorEventListener {

    private var binding: FragmentTotalStepsBinding? = null

    private lateinit var sensorManager: SensorManager
    private var stepSensor: Sensor? = null

    private var stepGoal = 10000
    private var currentSteps = 0
    private var currentOffset = 0
    private var offsetLoaded = false

    private var notified50 = false
    private var notified100 = false

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = FragmentTotalStepsBinding.inflate(inflater, container, false)

        sensorManager =
            requireContext().getSystemService(Context.SENSOR_SERVICE) as SensorManager
        stepSensor = sensorManager.getDefaultSensor(Sensor.TYPE_STEP_COUNTER)

        loadGoal()
        loadOffsetAndTodaySteps()

        return binding!!.root
    }

    override fun onResume() {
        super.onResume()
        stepSensor?.also {
            sensorManager.registerListener(this, it, SensorManagerSENSOR_DELAY_NORMAL)
        }
    }

    override fun onPause() {
        super.onPause()
        sensorManager.unregisterListener(this)
    }
}
```

```
private fun loadGoal() {
    FirebaseRepository.loadSetting("stepsGoal", "10000") { goal ->
        stepGoal = goal.toInt()
        binding?.maxStepGoalView?.text = goal
    }
}

private fun loadOffsetAndTodaySteps() {
    val date = today()

    // 1) Загружаем OFFSET
    FirebaseRepository.loadOffsetForDate(date) { offset ->
        currentOffset = offset
        offsetLoaded = true
    }

    // 2) Загружаем шаги за сегодня из БД
    FirebaseRepository.loadStepsForDate(date) { steps ->
        currentSteps = steps
        updateUI(steps)
    }
}

override fun onSensorChanged(event: SensorEvent) {
    if (!offsetLoaded) return
    if (event.sensor.type != Sensor.TYPE_STEP_COUNTER) return

    val totalSteps = event.values[0].toInt()
    val date = today()

    if (currentOffset == 0) {
        currentOffset = totalSteps
        FirebaseRepository.saveOffsetForDate(date, currentOffset)
    }

    val dailySteps = totalSteps - currentOffset
    currentSteps = dailySteps

    FirebaseRepository.saveStepsForDate(date, dailySteps)

    updateUI(dailySteps)
}

private fun updateUI(steps: Int) {
    binding?.apply {
        stepCountView.text = "$steps шагов"
        calorieCountView.text = "${calculateCalories(steps)} ккал"

        stepProgressBar.init(0, stepGoal)
        stepProgressBar.setProgress(steps)

        val percent = if (stepGoal > 0) (steps * 100 / stepGoal) else 0
        percentStepGoalView.text = "$percent%"
    }
}

// checkGoalNotifications(percent)
}

private fun calculateCalories(steps: Int): Int {
```

```
        return (steps * 0.04).toInt()
    }

private fun today(): String {
    val f = SimpleDateFormat("yyyy-MM-dd", Locale.getDefault())
    return f.format(Date())
}

override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {}

private fun checkGoalNotifications(percent: Int) {
    val ctx = requireContext()

    if (percent >= 50 && !notified50) {
        notified50 = true
        StepNotifications.send(
            ctx,
            "Отличный прогресс!",
            "Вы прошли 50% ежедневной цели!",
            50
        )
    }

    if (percent >= 100 && !notified100) {
        notified100 = true
        StepNotifications.send(
            ctx,
            "Цель достигнута!",
            "Вы выполнили 100% шаги цели на сегодня 🎉",
            100
        )
    }
}
```