СДЕЛАТЬ ПРИЛОЖЕНИЕ ДЛЯ МНОГИХ ПОЛЬЗОВАТЕЛЕЙ

1. диалог ввода веса, пола, расписание физической активности пользователя

2. расчет дневной нормы

3. сделать сбор данных о погоде

4. сделать умные уведомления:

• когда пользователь больше 3 часов не делает записи;

• сканирование данных о погоде;

• сканирование данных пользователя о физической активности, которую он ввел;

•

5. сделать так чтобы в настройках можно было поменять вес, пол, расписание физической активности  
  
СДЕЛАТЬ ТАК ЧТОБЫ ДАЖЕ КОГДА ДНЕВНАЯ НОРМА ДОСТИГНУТА ПОЛЬЗОВАТЕЛЬ МОГ ДОБАВЛЯТЬ ЗАПИСИ И СЧЕТЧИК МЛ ОБНОВЛЯЛСЯ  
  
Нужно сделать так, чтобы если пользователь удалил приложение на очень длительное время, или на случай, если в приложении произойдёт сбой, или при переустановке приложения, а потом заново установил его – он мог восстановить свой аккаунт по ID (должен храниться в базе данных всегда). нужно, чтобы при восстановлении аккаунта по ID, все личные данные пользователя (ID, пол, вес, возраст), записи о воде, записи о физических активностях – все было восстановлено.  
на Java в Android Studio.  
  
ДОБАВИТЬ НА МАКЕТ ВВОДА ДАННЫХ ЕДИНИЦУ ИЗМЕРЕНИЯ ВЕСА

ДОБАВИТЬ НА МАКЕТ СТРАНИЦЫ НАСТРОЕК ВЫБОР ЕДИНИЦЫ ИЗМЕРЕНИЯ ВЕСА   
  
Добавить на HomePage расписание физической активности

package com.example.waterly3;

import android.annotation.SuppressLint;

import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

import java.util.ArrayList;

import java.util.List;

public class DatabaseHelper extends SQLiteOpenHelper {

// ... остальной код класса остается без изменений ...

private void insertInitialRecommendations(SQLiteDatabase db) {

String[] recommendations = {

"Начинайте утро со стакана тёплой воды: это помогает выводить токсины и запускает работу печени",

"Пейте воду небольшими порциями, чтобы не создавать резкой нагрузки на сердечно-сосудистую систему",

"Не ждите чувства жажды — регулярно пейте воду в течение дня",

"Носите воду с собой — это поможет не забывать о питьевом режиме",

"Избегайте ледяной воды во время еды, чтобы не замедлять выработку пищеварительных ферментов",

"Пейте воду перед сдачей экзаменов — это улучшит концентрацию и снизит тревожность",

"За 30 минут до еды выпивайте стакан воды — это улучшит пищеварение и подготовит желудок к приёму пищи",

"Пейте воду перед утренней растяжкой — это улучшит эластичность мышц",

"Следите за качеством воды — выбирайте очищенную или фильтрованную воду",

"Пейте воду перед утренней зарядкой — это поможет разогнать метаболизм и зарядит энергией",

"Заменяйте сладкие напитки водой — это снизит потребление сахара и калорий",

"Регулярное употребление чистой воды улучшает эластичность и снижает сухость кожи",

"Не пейте залпом — медленное употребление воды предотвращает нагрузку на почки",

"Пейте воду комнатной температуры с долькой лимона — это стимулирует отток желчи в желчном пузыре",

"Следите за цветом мочи — светлый оттенок говорит о достаточном потреблении воды",

"Пейте воду с имбирем — это улучшит пищеварение и укрепит иммунитет",

"Заменяйте 1 кофе на 2 стакана воды — кофеин обезвоживает",

"После еды подождите 30-40 минут, прежде чем пить воду, чтобы не разбавлять пищеварительные соки",

"Пейте воду в течение дня, даже если не хочется — обезвоживание часто маскируется под голод",

"Пейте воду после алкоголя — это снизит последствия обезвоживания",

"Стакан воды с щепоткой корицы утром — разгоняет обмен веществ и снижает тягу к сладкому",

"Не пейте воду из бутылок, которые долго стояли на солнце — это может привести к выделению вредных веществ из пластика",

"Пейте воду перед приемом витаминов — это улучшит их усвоение",

"Тёплая вода с фенхелем перед сном — Помогает печени ночью активнее очищать кровь",

"Используйте стеклянную или металлическую бутылку — это экологично и безопасно",

"Следите за температурой воды в жару — прохладная, но не ледяная вода лучше утоляет жажду",

"Пейте воду с куркумой — это добавит антиоксидантов и улучшит иммунитет",

"Пейте воду после пробуждения от дневного сна — это поможет взбодриться",

"Пейте воду комнатной температуры — холодная вода замедляет метаболизм и сужает сосуды",

"Вода с розмарином улучшает память (настаивайте веточку в воде 2 часа)",

"Пейте через трубочку — это поможет употреблять воду медленнее и избежать заглатывания воздуха",

"Пейте воду перед сном — небольшое количество (100–150 мл) поможет поддерживать баланс жидкости ночью",

"«Водяное правило 5-3-2»: 5 мл на кг веса за 5 часов до тренировки, 3 мл — за 2 часа, 2 мл — каждые 20 минут во время",

"Кокосовая вода восстанавливает электролиты лучше спортивных напитков",

"Не пейте во время подхода — только между упражнениями",

"Пейте воду перед приемом лекарств — это улучшит их усвоение и снизит риск раздражения желудка",

"Используйте воду для борьбы с усталостью — обезвоживание часто вызывает чувство слабости",

"Вода с алоэ вера ускоряет рост волос и ногтей",

"2 глотка воды перед важным решением — улучшает когнитивные функции на 14%",

"Используйте воду для профилактики головной боли — часто ее причиной является недостаток жидкости",

"Пейте воду перед сдачей анализов — это улучшит точность результатов",

"Используйте воду для улучшения настроения — даже небольшое обезвоживание может влиять на эмоциональное состояние",

"Добавляйте в воду кусочки свежего граната — это придаст воде яркий вкус и обогатит антиоксидантами",

"Добавляйте в воду кусочки свежего киви — это придаст воде освежающий вкус и обогатит витамином С",

"Газированная вода лучше очищает кожу (но без газа на ночь)",

"Чередуйте обычную воду с минеральной — для восполнения микроэлементов",

"Пейте воду с щепоткой морской соли утром для баланса электролитов",

"Пейте воду перед дневным сном — это улучшит качество отдыха",

"Чередуйте воду с кокосовой водой в жаркие дни",

"Пейте воду перед проверкой почты — это снизит стресс от информации",

"«Водяной ритуал самураев» — перед питьем 3 раза перелейте воду из стакана в стакан для насыщения кислородом",

"Добавляйте лимон или мяту – улучшает вкус и стимулирует пищеварение",

"«Правило двух глотков» — каждые 15 минут делайте 2-3 глотка — так вода усваивается идеально",

"«1 стакан на час активности» — за компьютером, за рулём, при ходьбе — восполняйте потери",

"«Правило пустого стакана» — поставили рядом — выпили до конца часа",

"«Правило 1:1 для кофе» — на каждую чашку кофе — стакан воды",

"«Техника 4-4-4» — 4 больших глотка каждые 40 минут при сидячей работе",

"«Терморегуляция» — зимой увеличивайте потребление на 15-20% (отопление сушит воздух)",

"После 50 лет увеличивайте норму на 10-15% — с возрастом чувство жажды притупляется",

"«Гидро-ритуал перед сном» — за 1 час до сна выпивайте 100 мл теплой воды с медом — улучшает качество сна",

"1 стакан за 30 мин до душа — снижает артериальное давление",

"Пейте сидя — улучшает усвоение по сравнению с положением стоя",

"Пейте больше при стрессе — стресс увеличивает потребность в жидкости",

"Не пейте через силу — гипергидрация снижает уровень натрия (риск гипонатриемии)",

"Используйте правило «1 стакан в час» — так легче поддерживать баланс без перегрузки почек",

"Пейте больше в первой половине дня — почки активнее работают до 17:00",

"В горах пейте на 30% больше — разреженный воздух усиливает испарение через легкие",

"Температурный контраст — чередуйте теплую и прохладную воду для стимуляции метаболизма",

"14:00 — пиковое время гидратации — в этот период вода максимально эффективно распределяется",

"При тепловом ударе — вода с мятой и солью (1/4 ч.л. на 0,5 л) — экстренно восстанавливает баланс",

"Пейте больше в самолётах — воздух в салоне очень сухой, обезвоживание наступает быстрее",

"Не пейте слишком много за раз — это создаёт нагрузку на почки и может привести к гипонатриемии",

"Ограничьте воду во время еды — большое количество жидкости разбавляет желудочный сок",

"Пейте осознанно — делайте паузу, вдыхайте аромат (если с добавками) и наслаждайтесь процессом"

};

ContentValues values = new ContentValues();

for (String recommendation : recommendations) {

values.clear();

values.put(COLUMN\_DESCRIPTION, recommendation);

db.insert(TABLE\_RECOMMENDATIONS, null, values);

}

}

// ... остальной код класса остается без изменений ...

}

СОВМЕСТИ ТРИ КОДА В ОДИН

рассчитать дневную норму пользователя:

если ввёл данные и если оставил пустыми;

ввод возраста. если человеку за 50 увеличивать дневную норму

на главный экран добавить ввод информации о тренировках:

время тренировки или количество калорий. чтобы записывались в БД.

в настройках добавить смену языка интерфейса и других полей.

добавить светлую и темную тему. также добавить возможность самостоятельной настройки темы интерфейса пользователю.

добавить возможность убрать советы;

добавить возможность убрать расписание тренировок

проверить адаптивность элементов всех макетов под все устройства.

настроить цвет всех уведомлений. сделать так, чтобы при выборе темы интерфейса, уведомления настраивались под цвет интерфейса

сделай так, чтобы когда пользователь указывал данные в макете, они сохранялись в базу данных. создай таблицу.

сделай так, чтобы в расписании физической активности пользователь указывал только время тренировки. базовый выбор: 30, 1 час, 2 часа.

Также сделай так, чтобы эти параметры можно было изменять в настройках.

на Java Android Studio.

НА ДОБАВЛЕНИЕ ТАЛИЦЫ ФИЗ АКТИВНОСТЬ:

package com.example.waterly3;

import android.annotation.SuppressLint;

import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

import java.util.ArrayList;

import java.util.List;

public class DatabaseHelper extends SQLiteOpenHelper {

// Общие настройки базы данных

private static final String DATABASE\_NAME = "WaterlyDB";

private static final int DATABASE\_VERSION = 4;

// Таблица для рекомендаций

public static final String TABLE\_RECOMMENDATIONS = "Recommendations";

public static final String COLUMN\_REC\_ID = "ID";

public static final String COLUMN\_DESCRIPTION = "Description";

// Таблица для записей о воде

public static final String TABLE\_WATER\_ENTRIES = "water\_entries";

public static final String COLUMN\_ID = "id";

public static final String COLUMN\_USER\_ID = "user\_id";

public static final String COLUMN\_AMOUNT = "amount";

public static final String COLUMN\_DATE\_TIME = "date\_time";

// Таблица для профиля пользователя

public static final String TABLE\_USER\_PROFILE = "user\_profile";

public static final String COLUMN\_GENDER = "gender";

public static final String COLUMN\_WEIGHT = "weight";

public static final String COLUMN\_AGE = "age";

// SQL запросы для создания таблиц

private static final String SQL\_CREATE\_RECOMMENDATIONS\_TABLE =

"CREATE TABLE " + TABLE\_RECOMMENDATIONS + " (" +

COLUMN\_REC\_ID + " INTEGER PRIMARY KEY AUTOINCREMENT," +

COLUMN\_DESCRIPTION + " TEXT NOT NULL)";

private static final String SQL\_CREATE\_WATER\_ENTRIES\_TABLE =

"CREATE TABLE " + TABLE\_WATER\_ENTRIES + "(" +

COLUMN\_ID + " INTEGER PRIMARY KEY AUTOINCREMENT," +

COLUMN\_USER\_ID + " TEXT NOT NULL," +

COLUMN\_AMOUNT + " INTEGER NOT NULL," +

COLUMN\_DATE\_TIME + " TEXT NOT NULL" + ")";

private static final String SQL\_CREATE\_USER\_PROFILE\_TABLE =

"CREATE TABLE " + TABLE\_USER\_PROFILE + " (" +

COLUMN\_USER\_ID + " TEXT PRIMARY KEY," +

COLUMN\_GENDER + " TEXT NOT NULL," +

COLUMN\_WEIGHT + " INTEGER NOT NULL," +

COLUMN\_AGE + " INTEGER NOT NULL)";

public DatabaseHelper(Context context) {

super(context, DATABASE\_NAME, null, DATABASE\_VERSION);

}

@Override

public void onCreate(SQLiteDatabase db) {

db.execSQL(SQL\_CREATE\_RECOMMENDATIONS\_TABLE);

db.execSQL(SQL\_CREATE\_WATER\_ENTRIES\_TABLE);

db.execSQL(SQL\_CREATE\_USER\_PROFILE\_TABLE);

insertInitialRecommendations(db);

}

@Override

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

db.execSQL("DROP TABLE IF EXISTS " + TABLE\_RECOMMENDATIONS);

db.execSQL("DROP TABLE IF EXISTS " + TABLE\_WATER\_ENTRIES);

db.execSQL("DROP TABLE IF EXISTS " + TABLE\_USER\_PROFILE);

onCreate(db);

}

private void insertInitialRecommendations(SQLiteDatabase db) {

String[] recommendations = {

"Начинайте день со стакана воды — это помогает запустить обмен веществ и пробудить организм",

// ... остальные рекомендации ...

};

ContentValues values = new ContentValues();

for (String recommendation : recommendations) {

values.clear();

values.put(COLUMN\_DESCRIPTION, recommendation);

db.insert(TABLE\_RECOMMENDATIONS, null, values);

}

}

// Методы для работы с рекомендациями

public Cursor getAllRecommendations() {

SQLiteDatabase db = this.getReadableDatabase();

return db.query(TABLE\_RECOMMENDATIONS,

new String[]{COLUMN\_REC\_ID, COLUMN\_DESCRIPTION},

null, null, null, null, null);

}

public String getRecommendationById(int id) {

SQLiteDatabase db = this.getReadableDatabase();

Cursor cursor = db.query(TABLE\_RECOMMENDATIONS,

new String[]{COLUMN\_DESCRIPTION},

COLUMN\_REC\_ID + "=?",

new String[]{String.valueOf(id)},

null, null, null);

if (cursor != null && cursor.moveToFirst()) {

String recommendation = cursor.getString(0);

cursor.close();

return recommendation;

}

return null;

}

public String getRandomRecommendation() {

SQLiteDatabase db = this.getReadableDatabase();

Cursor cursor = db.rawQuery("SELECT " + COLUMN\_DESCRIPTION + " FROM " +

TABLE\_RECOMMENDATIONS + " ORDER BY RANDOM() LIMIT 1", null);

if (cursor != null && cursor.moveToFirst()) {

String recommendation = cursor.getString(0);

cursor.close();

return recommendation;

}

return null;

}

// Методы для работы с записями о воде

public long addWaterEntry(WaterEntry entry) {

SQLiteDatabase db = this.getWritableDatabase();

ContentValues values = new ContentValues();

values.put(COLUMN\_USER\_ID, entry.getUserId());

values.put(COLUMN\_AMOUNT, entry.getAmount());

values.put(COLUMN\_DATE\_TIME, entry.getDateTime());

long id = db.insert(TABLE\_WATER\_ENTRIES, null, values);

db.close();

return id;

}

@SuppressLint("Range")

public List<WaterEntry> getUserWaterEntries(String userId) {

List<WaterEntry> entries = new ArrayList<>();

String query = "SELECT \* FROM " + TABLE\_WATER\_ENTRIES +

" WHERE " + COLUMN\_USER\_ID + " = ?" +

" ORDER BY " + COLUMN\_DATE\_TIME + " DESC";

SQLiteDatabase db = this.getReadableDatabase();

Cursor cursor = db.rawQuery(query, new String[]{userId});

if (cursor.moveToFirst()) {

do {

WaterEntry entry = new WaterEntry();

entry.setId(cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)));

entry.setUserId(cursor.getString(cursor.getColumnIndex(COLUMN\_USER\_ID)));

entry.setAmount(cursor.getInt(cursor.getColumnIndex(COLUMN\_AMOUNT)));

entry.setDateTime(cursor.getString(cursor.getColumnIndex(COLUMN\_DATE\_TIME)));

entries.add(entry);

} while (cursor.moveToNext());

}

cursor.close();

db.close();

return entries;

}

@SuppressLint("Range")

public int getUserTotalWater(String userId) {

int total = 0;

String query = "SELECT SUM(" + COLUMN\_AMOUNT + ") as total FROM " +

TABLE\_WATER\_ENTRIES + " WHERE " + COLUMN\_USER\_ID + " = ?";

SQLiteDatabase db = this.getReadableDatabase();

Cursor cursor = db.rawQuery(query, new String[]{userId});

if (cursor.moveToFirst()) {

total = cursor.getInt(cursor.getColumnIndex("total"));

}

cursor.close();

db.close();

return total;

}

public boolean deleteWaterEntry(int id) {

SQLiteDatabase db = this.getWritableDatabase();

int result = db.delete(TABLE\_WATER\_ENTRIES, COLUMN\_ID + " = ?",

new String[]{String.valueOf(id)});

db.close();

return result > 0;

}

public boolean updateWaterEntry(WaterEntry entry) {

SQLiteDatabase db = this.getWritableDatabase();

ContentValues values = new ContentValues();

values.put(COLUMN\_AMOUNT, entry.getAmount());

values.put(COLUMN\_DATE\_TIME, entry.getDateTime());

int result = db.update(TABLE\_WATER\_ENTRIES, values,

COLUMN\_ID + " = ?",

new String[]{String.valueOf(entry.getId())});

db.close();

return result > 0;

}

// Методы для работы с профилем пользователя

public void saveUserProfile(String userId, String gender, int weight, int age) {

SQLiteDatabase db = this.getWritableDatabase();

ContentValues values = new ContentValues();

values.put(COLUMN\_USER\_ID, userId);

values.put(COLUMN\_GENDER, gender);

values.put(COLUMN\_WEIGHT, weight);

values.put(COLUMN\_AGE, age);

int rowsAffected = db.update(TABLE\_USER\_PROFILE, values,

COLUMN\_USER\_ID + " = ?", new String[]{userId});

if (rowsAffected == 0) {

db.insert(TABLE\_USER\_PROFILE, null, values);

}

db.close();

}

public UserProfile getUserProfile(String userId) {

SQLiteDatabase db = this.getReadableDatabase();

UserProfile profile = null;

Cursor cursor = db.query(TABLE\_USER\_PROFILE,

new String[]{COLUMN\_GENDER, COLUMN\_WEIGHT, COLUMN\_AGE},

COLUMN\_USER\_ID + " = ?",

new String[]{userId},

null, null, null);

if (cursor.moveToFirst()) {

profile = new UserProfile(

cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_GENDER)),

cursor.getInt(cursor.getColumnIndexOrThrow(COLUMN\_WEIGHT)),

cursor.getInt(cursor.getColumnIndexOrThrow(COLUMN\_AGE))

);

cursor.close();

}

db.close();

return profile;

}

}

ИЗМЕНИ ТАБЛИЦУ user\_profile: добавь туда столбец “Единица измерения”, ПОСЛЕ СТОЛБЦА weight.

ДОБАВЬ ТАБЛИЦУ "ФИЗИЧЕСКАЯ АКТИВНОСТЬ", СО СТОЛБЦАМИ:

- ID АККАУНТА

- ДАТА И ВРЕМЯ ЗАПИСИ

- ДЛИТЕЛЬНОСТЬ ТРЕНИРОВКИ (В МИНУТАХ)

- ЕДИНИЦА ИЗМЕРЕНИЯ ПОТРАЧЕННОЙ ЭНЕРГИИ

- КОЛИЧЕСТВО ПОТРАЧЕННОЙ ЭНЕРГИИ (ККАЛ ИЛИ КДЖ)

база данных:

package com.example.waterly3;

import android.annotation.SuppressLint;

import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

import java.util.ArrayList;

import java.util.List;

public class DatabaseHelper extends SQLiteOpenHelper {

// Общие настройки базы данных

private static final String DATABASE\_NAME = "WaterlyDB";

private static final int DATABASE\_VERSION = 3;

// Таблица для рекомендаций

public static final String TABLE\_RECOMMENDATIONS = "Recommendations";

public static final String COLUMN\_REC\_ID = "ID";

public static final String COLUMN\_DESCRIPTION = "Description";

// Таблица для записей о воде

public static final String TABLE\_WATER\_ENTRIES = "water\_entries";

public static final String COLUMN\_ID = "id";

public static final String COLUMN\_USER\_ID = "user\_id";

public static final String COLUMN\_AMOUNT = "amount";

public static final String COLUMN\_DATE\_TIME = "date\_time";

// SQL запросы для создания таблиц

private static final String SQL\_CREATE\_RECOMMENDATIONS\_TABLE =

"CREATE TABLE " + TABLE\_RECOMMENDATIONS + " (" +

COLUMN\_REC\_ID + " INTEGER PRIMARY KEY AUTOINCREMENT," +

COLUMN\_DESCRIPTION + " TEXT NOT NULL)";

private static final String SQL\_CREATE\_WATER\_ENTRIES\_TABLE =

"CREATE TABLE " + TABLE\_WATER\_ENTRIES + "(" +

COLUMN\_ID + " INTEGER PRIMARY KEY AUTOINCREMENT," +

COLUMN\_USER\_ID + " TEXT NOT NULL," +

COLUMN\_AMOUNT + " INTEGER NOT NULL," +

COLUMN\_DATE\_TIME + " TEXT NOT NULL" + ")";

public DatabaseHelper(Context context) {

super(context, DATABASE\_NAME, null, DATABASE\_VERSION);

}

@Override

public void onCreate(SQLiteDatabase db) {

// Создаем обе таблицы

db.execSQL(SQL\_CREATE\_RECOMMENDATIONS\_TABLE);

db.execSQL(SQL\_CREATE\_WATER\_ENTRIES\_TABLE);

// Заполняем таблицу рекомендаций начальными данными

insertInitialRecommendations(db);

}

@Override

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

// При обновлении версии пересоздаем обе таблицы

db.execSQL("DROP TABLE IF EXISTS " + TABLE\_RECOMMENDATIONS);

db.execSQL("DROP TABLE IF EXISTS " + TABLE\_WATER\_ENTRIES);

onCreate(db);

}

// Метод для заполнения таблицы рекомендаций начальными данными

private void insertInitialRecommendations(SQLiteDatabase db) {

String[] recommendations = {

"Начинайте день со стакана воды — это помогает запустить обмен веществ и пробудить организм",

"Пейте воду небольшими глотками — так она лучше усваивается",

"Не ждите чувства жажды — регулярно пейте воду в течение дня",

"Носите воду с собой — это поможет не забывать о питьевом режиме",

"Пейте воду комнатной температуры — холодная вода может замедлить пищеварение",

"Пейте воду перед сдачей экзаменов — это улучшит концентрацию и снизит тревожность",

"Пейте до и после еды — за 20–30 минут до приема пищи и через час после",

"Пейте воду перед утренней растяжкой — это улучшит эластичность мышц",

"Следите за качеством воды — выбирайте очищенную или фильтрованную воду",

"Пейте воду перед утренней зарядкой — это поможет разогнать метаболизм и зарядит энергией",

"Заменяйте сладкие напитки водой — это снизит потребление сахара и калорий",

"Используйте мерную бутылку — так легче контролировать объем выпитой воды",

"Не пейте залпом — медленное употребление воды предотвращает нагрузку на почки",

"Пейте воду с электролитами — особенно после тренировок или в жаркую погоду",

"Следите за цветом мочи — светлый оттенок говорит о достаточном потреблении воды",

"Пейте воду с имбирем — это улучшит пищеварение и укрепит иммунитет",

"Пейте воду перед кофе или чаем — это поможет избежать обезвоживания от кофеина",

"Добавьте ягоды или фрукты — это придаст воде вкус и обогатит витаминами",

"Используйте воду для перекусов — иногда жажда маскируется под голод",

"Пейте воду после алкоголя — это снизит последствия обезвоживания",

"Пейте воду с мятой или базиликом — это освежает и улучшает пищеварение",

"Не пейте воду из бутылок, которые долго стояли на солнце — это может привести к выделению вредных веществ из пластика",

"Пейте воду перед приемом витаминов — это улучшит их усвоение",

"Пейте воду перед важными задачами — это улучшит концентрацию и продуктивность",

"Используйте стеклянную или металлическую бутылку — это экологично и безопасно",

"Следите за температурой воды в жару — прохладная, но не ледяная вода лучше утоляет жажду",

"Пейте воду с куркумой — это добавит антиоксидантов и улучшит иммунитет",

"Пейте воду после пробуждения от дневного сна — это поможет взбодриться",

"Следите за качеством воды в путешествиях — выбирайте бутилированную или кипяченую воду",

"Добавляйте в воду дольку лимона или огурца — это улучшит вкус и добавит витаминов",

"Пейте через трубочку — это поможет употреблять воду медленнее и избежать заглатывания воздуха",

"Пейте воду перед сном — небольшое количество (100–150 мл) поможет поддерживать баланс жидкости ночью",

"Избегайте газированной воды с сахаром — она может вызвать обезвоживание и вредна для здоровья",

"Не заменяйте воду чаем или кофе — они обладают мочегонным эффектом и могут способствовать обезвоживанию",

"Пейте воду перед физической активностью — это подготовит организм к нагрузкам и улучшит выносливость",

"Пейте воду перед приемом лекарств — это улучшит их усвоение и снизит риск раздражения желудка",

"Используйте воду для борьбы с усталостью — обезвоживание часто вызывает чувство слабости",

"Пейте воду перед утренним душем — это поможет избежать резкого падения давления",

"Пейте воду перед важными переговорами — это снизит нервозность и улучшит ясность мысли",

"Используйте воду для профилактики головной боли — часто ее причиной является недостаток жидкости",

"Пейте воду перед сдачей анализов — это улучшит точность результатов",

"Используйте воду для улучшения настроения — даже небольшое обезвоживание может влиять на эмоциональное состояние",

"Добавляйте в воду кусочки свежего граната — это придаст воде яркий вкус и обогатит антиоксидантами",

"Добавляйте в воду кусочки свежего киви — это придаст воде освежающий вкус и обогатит витамином С",

"Добавляйте в воду кусочки свежей груши — это придаст воде мягкий сладкий вкус и обогатит клетчаткой",

"Чередуйте обычную воду с минеральной — для восполнения микроэлементов",

"Пейте воду с щепоткой морской соли утром для баланса электролитов",

"Пейте воду перед дневным сном — это улучшит качество отдыха",

"Чередуйте воду с кокосовой водой в жаркие дни",

"Пейте воду перед проверкой почты — это снизит стресс от информации"

};

ContentValues values = new ContentValues();

for (String recommendation : recommendations) {

values.clear();

values.put(COLUMN\_DESCRIPTION, recommendation);

db.insert(TABLE\_RECOMMENDATIONS, null, values);

}

}

// ===== Методы для работы с рекомендациями =====

public Cursor getAllRecommendations() {

SQLiteDatabase db = this.getReadableDatabase();

return db.query(TABLE\_RECOMMENDATIONS,

new String[]{COLUMN\_REC\_ID, COLUMN\_DESCRIPTION},

null, null, null, null, null);

}

public String getRecommendationById(int id) {

SQLiteDatabase db = this.getReadableDatabase();

Cursor cursor = db.query(TABLE\_RECOMMENDATIONS,

new String[]{COLUMN\_DESCRIPTION},

COLUMN\_REC\_ID + "=?",

new String[]{String.valueOf(id)},

null, null, null);

if (cursor != null && cursor.moveToFirst()) {

String recommendation = cursor.getString(0);

cursor.close();

return recommendation;

}

return null;

}

public String getRandomRecommendation() {

SQLiteDatabase db = this.getReadableDatabase();

Cursor cursor = db.rawQuery("SELECT " + COLUMN\_DESCRIPTION + " FROM " +

TABLE\_RECOMMENDATIONS + " ORDER BY RANDOM() LIMIT 1", null);

if (cursor != null && cursor.moveToFirst()) {

String recommendation = cursor.getString(0);

cursor.close();

return recommendation;

}

return null;

}

// ===== Методы для работы с записями о воде =====

public long addWaterEntry(WaterEntry entry) {

SQLiteDatabase db = this.getWritableDatabase();

ContentValues values = new ContentValues();

values.put(COLUMN\_USER\_ID, entry.getUserId());

values.put(COLUMN\_AMOUNT, entry.getAmount());

values.put(COLUMN\_DATE\_TIME, entry.getDateTime());

long id = db.insert(TABLE\_WATER\_ENTRIES, null, values);

db.close();

return id;

}

@SuppressLint("Range")

public List<WaterEntry> getUserWaterEntries(String userId) {

List<WaterEntry> entries = new ArrayList<>();

String query = "SELECT \* FROM " + TABLE\_WATER\_ENTRIES +

" WHERE " + COLUMN\_USER\_ID + " = ?" +

" ORDER BY " + COLUMN\_DATE\_TIME + " DESC";

SQLiteDatabase db = this.getReadableDatabase();

Cursor cursor = db.rawQuery(query, new String[]{userId});

if (cursor.moveToFirst()) {

do {

WaterEntry entry = new WaterEntry();

entry.setId(cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)));

entry.setUserId(cursor.getString(cursor.getColumnIndex(COLUMN\_USER\_ID)));

entry.setAmount(cursor.getInt(cursor.getColumnIndex(COLUMN\_AMOUNT)));

entry.setDateTime(cursor.getString(cursor.getColumnIndex(COLUMN\_DATE\_TIME)));

entries.add(entry);

} while (cursor.moveToNext());

}

cursor.close();

db.close();

return entries;

}

@SuppressLint("Range")

public int getUserTotalWater(String userId) {

int total = 0;

String query = "SELECT SUM(" + COLUMN\_AMOUNT + ") as total FROM " +

TABLE\_WATER\_ENTRIES + " WHERE " + COLUMN\_USER\_ID + " = ?";

SQLiteDatabase db = this.getReadableDatabase();

Cursor cursor = db.rawQuery(query, new String[]{userId});

if (cursor.moveToFirst()) {

total = cursor.getInt(cursor.getColumnIndex("total"));

}

cursor.close();

db.close();

return total;

}

// Метод для удаления записи о воде

public boolean deleteWaterEntry(int id) {

SQLiteDatabase db = this.getWritableDatabase();

int result = db.delete(TABLE\_WATER\_ENTRIES, COLUMN\_ID + " = ?",

new String[]{String.valueOf(id)});

db.close();

return result > 0;

}

// Метод для обновления записи о воде

public boolean updateWaterEntry(WaterEntry entry) {

SQLiteDatabase db = this.getWritableDatabase();

ContentValues values = new ContentValues();

values.put(COLUMN\_AMOUNT, entry.getAmount());

values.put(COLUMN\_DATE\_TIME, entry.getDateTime());

int result = db.update(TABLE\_WATER\_ENTRIES, values,

COLUMN\_ID + " = ?",

new String[]{String.valueOf(entry.getId())});

db.close();

return result > 0;

}

}

package com.example.waterly3;

import android.annotation.SuppressLint;

import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

import java.util.ArrayList;

import java.util.List;

public class DatabaseHelper extends SQLiteOpenHelper {

// Общие настройки базы данных

private static final String DATABASE\_NAME = "WaterlyDB";

private static final int DATABASE\_VERSION = 3;

// Таблица для рекомендаций

public static final String TABLE\_RECOMMENDATIONS = "Recommendations";

public static final String COLUMN\_REC\_ID = "ID";

public static final String COLUMN\_DESCRIPTION = "Description";

// Таблица для записей о воде

public static final String TABLE\_WATER\_ENTRIES = "water\_entries";

public static final String COLUMN\_ID = "id";

public static final String COLUMN\_USER\_ID = "user\_id";

public static final String COLUMN\_AMOUNT = "amount";

public static final String COLUMN\_DATE\_TIME = "date\_time";

// SQL запросы для создания таблиц

private static final String SQL\_CREATE\_RECOMMENDATIONS\_TABLE =

"CREATE TABLE " + TABLE\_RECOMMENDATIONS + " (" +

COLUMN\_REC\_ID + " INTEGER PRIMARY KEY AUTOINCREMENT," +

COLUMN\_DESCRIPTION + " TEXT NOT NULL)";

private static final String SQL\_CREATE\_WATER\_ENTRIES\_TABLE =

"CREATE TABLE " + TABLE\_WATER\_ENTRIES + "(" +

COLUMN\_ID + " INTEGER PRIMARY KEY AUTOINCREMENT," +

COLUMN\_USER\_ID + " TEXT NOT NULL," +

COLUMN\_AMOUNT + " INTEGER NOT NULL," +

COLUMN\_DATE\_TIME + " TEXT NOT NULL" + ")";

public DatabaseHelper(Context context) {

super(context, DATABASE\_NAME, null, DATABASE\_VERSION);

}

@Override

public void onCreate(SQLiteDatabase db) {

// Создаем обе таблицы

db.execSQL(SQL\_CREATE\_RECOMMENDATIONS\_TABLE);

db.execSQL(SQL\_CREATE\_WATER\_ENTRIES\_TABLE);

// Заполняем таблицу рекомендаций начальными данными

insertInitialRecommendations(db);

}

@Override

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

// При обновлении версии пересоздаем обе таблицы

db.execSQL("DROP TABLE IF EXISTS " + TABLE\_RECOMMENDATIONS);

db.execSQL("DROP TABLE IF EXISTS " + TABLE\_WATER\_ENTRIES);

onCreate(db);

}

// Метод для заполнения таблицы рекомендаций начальными данными

private void insertInitialRecommendations(SQLiteDatabase db) {

String[] recommendations = {

"Начинайте день со стакана воды — это помогает запустить обмен веществ и пробудить организм",

"Пейте воду небольшими глотками — так она лучше усваивается",

"Не ждите чувства жажды — регулярно пейте воду в течение дня",

"Носите воду с собой — это поможет не забывать о питьевом режиме",

"Пейте воду комнатной температуры — холодная вода может замедлить пищеварение",

"Пейте воду перед сдачей экзаменов — это улучшит концентрацию и снизит тревожность",

"Пейте до и после еды — за 20–30 минут до приема пищи и через час после",

"Пейте воду перед утренней растяжкой — это улучшит эластичность мышц",

"Следите за качеством воды — выбирайте очищенную или фильтрованную воду",

"Пейте воду перед утренней зарядкой — это поможет разогнать метаболизм и зарядит энергией",

"Заменяйте сладкие напитки водой — это снизит потребление сахара и калорий",

"Используйте мерную бутылку — так легче контролировать объем выпитой воды",

"Не пейте залпом — медленное употребление воды предотвращает нагрузку на почки",

"Пейте воду с электролитами — особенно после тренировок или в жаркую погоду",

"Следите за цветом мочи — светлый оттенок говорит о достаточном потреблении воды",

"Пейте воду с имбирем — это улучшит пищеварение и укрепит иммунитет",

"Пейте воду перед кофе или чаем — это поможет избежать обезвоживания от кофеина",

"Добавьте ягоды или фрукты — это придаст воде вкус и обогатит витаминами",

"Используйте воду для перекусов — иногда жажда маскируется под голод",

"Пейте воду после алкоголя — это снизит последствия обезвоживания",

"Пейте воду с мятой или базиликом — это освежает и улучшает пищеварение",

"Не пейте воду из бутылок, которые долго стояли на солнце — это может привести к выделению вредных веществ из пластика",

"Пейте воду перед приемом витаминов — это улучшит их усвоение",

"Пейте воду перед важными задачами — это улучшит концентрацию и продуктивность",

"Используйте стеклянную или металлическую бутылку — это экологично и безопасно",

"Следите за температурой воды в жару — прохладная, но не ледяная вода лучше утоляет жажду",

"Пейте воду с куркумой — это добавит антиоксидантов и улучшит иммунитет",

"Пейте воду после пробуждения от дневного сна — это поможет взбодриться",

"Следите за качеством воды в путешествиях — выбирайте бутилированную или кипяченую воду",

"Добавляйте в воду дольку лимона или огурца — это улучшит вкус и добавит витаминов",

"Пейте через трубочку — это поможет употреблять воду медленнее и избежать заглатывания воздуха",

"Пейте воду перед сном — небольшое количество (100–150 мл) поможет поддерживать баланс жидкости ночью",

"Избегайте газированной воды с сахаром — она может вызвать обезвоживание и вредна для здоровья",

"Не заменяйте воду чаем или кофе — они обладают мочегонным эффектом и могут способствовать обезвоживанию",

"Пейте воду перед физической активностью — это подготовит организм к нагрузкам и улучшит выносливость",

"Пейте воду перед приемом лекарств — это улучшит их усвоение и снизит риск раздражения желудка",

"Используйте воду для борьбы с усталостью — обезвоживание часто вызывает чувство слабости",

"Пейте воду перед утренним душем — это поможет избежать резкого падения давления",

"Пейте воду перед важными переговорами — это снизит нервозность и улучшит ясность мысли",

"Используйте воду для профилактики головной боли — часто ее причиной является недостаток жидкости",

"Пейте воду перед сдачей анализов — это улучшит точность результатов",

"Используйте воду для улучшения настроения — даже небольшое обезвоживание может влиять на эмоциональное состояние",

"Добавляйте в воду кусочки свежего граната — это придаст воде яркий вкус и обогатит антиоксидантами",

"Добавляйте в воду кусочки свежего киви — это придаст воде освежающий вкус и обогатит витамином С",

"Добавляйте в воду кусочки свежей груши — это придаст воде мягкий сладкий вкус и обогатит клетчаткой",

"Чередуйте обычную воду с минеральной — для восполнения микроэлементов",

"Пейте воду с щепоткой морской соли утром для баланса электролитов",

"Пейте воду перед дневным сном — это улучшит качество отдыха",

"Чередуйте воду с кокосовой водой в жаркие дни",

"Пейте воду перед проверкой почты — это снизит стресс от информации"

};

ContentValues values = new ContentValues();

for (String recommendation : recommendations) {

values.clear();

values.put(COLUMN\_DESCRIPTION, recommendation);

db.insert(TABLE\_RECOMMENDATIONS, null, values);

}

}

// ===== Методы для работы с рекомендациями =====

public Cursor getAllRecommendations() {

SQLiteDatabase db = this.getReadableDatabase();

return db.query(TABLE\_RECOMMENDATIONS,

new String[]{COLUMN\_REC\_ID, COLUMN\_DESCRIPTION},

null, null, null, null, null);

}

public String getRecommendationById(int id) {

SQLiteDatabase db = this.getReadableDatabase();

Cursor cursor = db.query(TABLE\_RECOMMENDATIONS,

new String[]{COLUMN\_DESCRIPTION},

COLUMN\_REC\_ID + "=?",

new String[]{String.valueOf(id)},

null, null, null);

if (cursor != null && cursor.moveToFirst()) {

String recommendation = cursor.getString(0);

cursor.close();

return recommendation;

}

return null;

}

public String getRandomRecommendation() {

SQLiteDatabase db = this.getReadableDatabase();

Cursor cursor = db.rawQuery("SELECT " + COLUMN\_DESCRIPTION + " FROM " +

TABLE\_RECOMMENDATIONS + " ORDER BY RANDOM() LIMIT 1", null);

if (cursor != null && cursor.moveToFirst()) {

String recommendation = cursor.getString(0);

cursor.close();

return recommendation;

}

return null;

}

// ===== Методы для работы с записями о воде =====

public long addWaterEntry(WaterEntry entry) {

SQLiteDatabase db = this.getWritableDatabase();

ContentValues values = new ContentValues();

values.put(COLUMN\_USER\_ID, entry.getUserId());

values.put(COLUMN\_AMOUNT, entry.getAmount());

values.put(COLUMN\_DATE\_TIME, entry.getDateTime());

long id = db.insert(TABLE\_WATER\_ENTRIES, null, values);

db.close();

return id;

}

@SuppressLint("Range")

public List<WaterEntry> getUserWaterEntries(String userId) {

List<WaterEntry> entries = new ArrayList<>();

String query = "SELECT \* FROM " + TABLE\_WATER\_ENTRIES +

" WHERE " + COLUMN\_USER\_ID + " = ?" +

" ORDER BY " + COLUMN\_DATE\_TIME + " DESC";

SQLiteDatabase db = this.getReadableDatabase();

Cursor cursor = db.rawQuery(query, new String[]{userId});

if (cursor.moveToFirst()) {

do {

WaterEntry entry = new WaterEntry();

entry.setId(cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)));

entry.setUserId(cursor.getString(cursor.getColumnIndex(COLUMN\_USER\_ID)));

entry.setAmount(cursor.getInt(cursor.getColumnIndex(COLUMN\_AMOUNT)));

entry.setDateTime(cursor.getString(cursor.getColumnIndex(COLUMN\_DATE\_TIME)));

entries.add(entry);

} while (cursor.moveToNext());

}

cursor.close();

db.close();

return entries;

}

@SuppressLint("Range")

public int getUserTotalWater(String userId) {

int total = 0;

String query = "SELECT SUM(" + COLUMN\_AMOUNT + ") as total FROM " +

TABLE\_WATER\_ENTRIES + " WHERE " + COLUMN\_USER\_ID + " = ?";

SQLiteDatabase db = this.getReadableDatabase();

Cursor cursor = db.rawQuery(query, new String[]{userId});

if (cursor.moveToFirst()) {

total = cursor.getInt(cursor.getColumnIndex("total"));

}

cursor.close();

db.close();

return total;

}

// Метод для удаления записи о воде

public boolean deleteWaterEntry(int id) {

SQLiteDatabase db = this.getWritableDatabase();

int result = db.delete(TABLE\_WATER\_ENTRIES, COLUMN\_ID + " = ?",

new String[]{String.valueOf(id)});

db.close();

return result > 0;

}

// Метод для обновления записи о воде

public boolean updateWaterEntry(WaterEntry entry) {

SQLiteDatabase db = this.getWritableDatabase();

ContentValues values = new ContentValues();

values.put(COLUMN\_AMOUNT, entry.getAmount());

values.put(COLUMN\_DATE\_TIME, entry.getDateTime());

int result = db.update(TABLE\_WATER\_ENTRIES, values,

COLUMN\_ID + " = ?",

new String[]{String.valueOf(entry.getId())});

db.close();

return result > 0;

}

}

// Добавьте эти константы в класс DatabaseHelper

public static final String TABLE\_USER\_PROFILE = "user\_profile";

public static final String COLUMN\_USER\_ID = "user\_id";

public static final String COLUMN\_GENDER = "gender";

public static final String COLUMN\_WEIGHT = "weight";

// Добавьте этот SQL запрос для создания таблицы

private static final String SQL\_CREATE\_USER\_PROFILE\_TABLE =

"CREATE TABLE " + TABLE\_USER\_PROFILE + " (" +

COLUMN\_USER\_ID + " TEXT PRIMARY KEY," +

COLUMN\_GENDER + " TEXT NOT NULL," +

COLUMN\_WEIGHT + " INTEGER NOT NULL)";

// В методе onCreate() добавьте создание новой таблицы

@Override

public void onCreate(SQLiteDatabase db) {

db.execSQL(SQL\_CREATE\_RECOMMENDATIONS\_TABLE);

db.execSQL(SQL\_CREATE\_WATER\_ENTRIES\_TABLE);

db.execSQL(SQL\_CREATE\_USER\_PROFILE\_TABLE);

insertInitialRecommendations(db);

}

// В методе onUpgrade() добавьте удаление новой таблицы

@Override

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

db.execSQL("DROP TABLE IF EXISTS " + TABLE\_RECOMMENDATIONS);

db.execSQL("DROP TABLE IF EXISTS " + TABLE\_WATER\_ENTRIES);

db.execSQL("DROP TABLE IF EXISTS " + TABLE\_USER\_PROFILE);

onCreate(db);

}

// Метод для сохранения или обновления профиля пользователя

public void saveUserProfile(String userId, String gender, int weight) {

SQLiteDatabase db = this.getWritableDatabase();

ContentValues values = new ContentValues();

values.put(COLUMN\_USER\_ID, userId);

values.put(COLUMN\_GENDER, gender);

values.put(COLUMN\_WEIGHT, weight);

// Пытаемся обновить запись, если она уже существует

int rowsAffected = db.update(TABLE\_USER\_PROFILE, values,

COLUMN\_USER\_ID + " = ?", new String[]{userId});

// Если запись не существует, вставляем новую

if (rowsAffected == 0) {

db.insert(TABLE\_USER\_PROFILE, null, values);

}

db.close();

}

// Метод для получения профиля пользователя

public UserProfile getUserProfile(String userId) {

SQLiteDatabase db = this.getReadableDatabase();

UserProfile profile = null;

Cursor cursor = db.query(TABLE\_USER\_PROFILE,

new String[]{COLUMN\_GENDER, COLUMN\_WEIGHT},

COLUMN\_USER\_ID + " = ?",

new String[]{userId},

null, null, null);

if (cursor != null && cursor.moveToFirst()) {

profile = new UserProfile(

cursor.getString(cursor.getColumnIndexOrThrow(COLUMN\_GENDER)),

cursor.getInt(cursor.getColumnIndexOrThrow(COLUMN\_WEIGHT))

);

cursor.close();

}

db.close();

return profile;

}

совмести коды в один

сделай так, чтобы когда пользователь нажимал на кнопку "Создать новый" ему показывался диалог ввода веса, пола и расписание физической активности.

Также сделай так, чтобы эти параметры можно было изменять в настройках.

И сделай так, чтобы при восстановлении аккаунта, данные о весе, поле и физической активности были сохранены в настройках.

с помощью Firebase Firestore.

на Java Android Studio.

Диалог восстановления данных:

package com.example.waterly3;

import android.app.Dialog;

import android.content.Context;

import android.os.Bundle;

import android.view.LayoutInflater;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import androidx.annotation.NonNull;

import androidx.annotation.Nullable;

import androidx.fragment.app.DialogFragment;

import com.google.android.material.dialog.MaterialAlertDialogBuilder;

public class RestoreAccountDialog extends DialogFragment {

// Интерфейсы для обработки действий

public interface OnRestoreListener {

void onRestore(String backupId);

}

public interface OnCreateListener {

void onCreateNew();

}

private OnRestoreListener restoreListener;

private OnCreateListener createListener;

// Метод для создания нового экземпляра диалога

public static RestoreAccountDialog newInstance(OnRestoreListener restoreListener,

OnCreateListener createListener) {

RestoreAccountDialog dialog = new RestoreAccountDialog();

dialog.restoreListener = restoreListener;

dialog.createListener = createListener;

return dialog;

}

// Проверяем, что активность реализует наши интерфейсы

@Override

public void onAttach(@NonNull Context context) {

super.onAttach(context);

if (context instanceof OnRestoreListener) {

restoreListener = (OnRestoreListener) context;

}

if (context instanceof OnCreateListener) {

createListener = (OnCreateListener) context;

}

}

@NonNull

@Override

public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {

MaterialAlertDialogBuilder builder = new MaterialAlertDialogBuilder(requireContext(), R.style.AlertDialogTheme);

LayoutInflater inflater = requireActivity().getLayoutInflater();

View dialogView = inflater.inflate(R.layout.dialog\_restore\_account, null);

final EditText idInput = dialogView.findViewById(R.id.backupIdEditText);

Button btnRestore = dialogView.findViewById(R.id.btnRestore);

Button btnCreateNew = dialogView.findViewById(R.id.btnCreateNew);

String backupId = UserManager.getBackupId(getContext());

if (backupId != null) {

idInput.setText(backupId);

idInput.setSelection(backupId.length());

}

builder.setView(dialogView)

.setTitle("Восстановление данных");

androidx.appcompat.app.AlertDialog dialog = builder.create();

btnRestore.setOnClickListener(v -> {

String id = idInput.getText().toString().trim();

if (!id.isEmpty() && restoreListener != null) {

restoreListener.onRestore(id);

}

dialog.dismiss();

});

btnCreateNew.setOnClickListener(v -> {

if (createListener != null) {

createListener.onCreateNew();

}

dialog.dismiss();

});

return dialog;

}

}

Макет страницы настроек:

<?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:padding="16dp">

<ScrollView

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintEnd\_toEndOf="parent">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical">

<!-- Заголовок "Настройки напоминаний" -->

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Настройки напоминаний"

android:textSize="20sp"

android:textColor="@color/blue"

android:textStyle="bold"

android:layout\_marginBottom="8dp"/>

<!-- Голубая линия -->

<View

android:layout\_width="match\_parent"

android:layout\_height="2dp"

android:background="@color/blue"

android:layout\_marginBottom="16dp"/>

<!-- Пункты настроек напоминаний -->

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Расписание напоминаний"

android:textSize="16sp"

android:textColor="@android:color/black"

android:layout\_marginBottom="16dp"/>

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Звук напоминания"

android:textSize="16sp"

android:textColor="@android:color/black"

android:layout\_marginBottom="16dp"/>

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Режим напоминания"

android:textSize="16sp"

android:textColor="@android:color/black"

android:layout\_marginBottom="16dp"/>

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:layout\_marginBottom="24dp"

android:gravity="center\_vertical">

<TextView

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="Дальнейшее напоминание"

android:textSize="16sp"

android:textColor="@android:color/black"/>

<Switch

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"/>

</LinearLayout>

<!-- Заголовок "Общие настройки" -->

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Общие настройки"

android:textSize="20sp"

android:textColor="@color/blue"

android:textStyle="bold"

android:layout\_marginBottom="8dp"/>

<!-- Голубая линия -->

<View

android:layout\_width="match\_parent"

android:layout\_height="2dp"

android:background="@color/blue"

android:layout\_marginBottom="16dp"/>

<!-- Пункты общих настроек -->

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Стиль интерфейса"

android:textSize="16sp"

android:textColor="@android:color/black"

android:layout\_marginBottom="16dp"/>

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Цель приема"

android:textSize="16sp"

android:textColor="@android:color/black"

android:layout\_marginBottom="16dp"/>

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Язык"

android:textSize="16sp"

android:textColor="@android:color/black"

android:layout\_marginBottom="24dp"/>

<!-- Заголовок "Личные данные" -->

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Личные данные"

android:textSize="20sp"

android:textColor="@color/blue"

android:textStyle="bold"

android:layout\_marginBottom="8dp"/>

<!-- Голубая линия -->

<View

android:layout\_width="match\_parent"

android:layout\_height="2dp"

android:background="@color/blue"

android:textStyle="bold"

android:layout\_marginBottom="16dp"/>

<!-- Пункты личных данных -->

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Пол"

android:textSize="16sp"

android:textColor="@android:color/black"

android:layout\_marginBottom="16dp"/>

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Вес"

android:textSize="16sp"

android:textColor="@android:color/black"

android:layout\_marginBottom="16dp"/>

<!-- Блок с ID пользователя -->

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical"

android:background="@drawable/rounded\_bg"

android:padding="16dp"

android:layout\_marginBottom="16dp">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textColor="@android:color/black"

android:text="ID аккаунта:"

android:textStyle="bold"

android:textSize="16sp"/>

<TextView

android:id="@+id/userIdTextView"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:textSize="18sp"

android:textColor="@android:color/black"

android:layout\_marginTop="8dp"

android:textStyle="bold"/>

</LinearLayout>

<!-- Кнопка показа ID -->

<Button

android:id="@+id/showIdButton"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Скопировать ID"

android:backgroundTint="@color/blue"

android:textColor="@android:color/white"

android:layout\_marginTop="-10dp"/>

</LinearLayout>

</ScrollView>

<!-- Временное уведомление (появляется при нажатии кнопки) -->

<TextView

android:id="@+id/idNotification"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textSize="16sp"

android:background="@drawable/rounded\_bg\_notification"

android:padding="12dp"

android:visibility="gone"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

android:textColor="@android:color/white"

android:layout\_marginBottom="16dp"/>

</androidx.constraintlayout.widget.ConstraintLayout>