МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1-98 01 03 “Программное обеспечение информационной

безопасности мобильных систем”

Специализация Инженер-программист

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Vikkru»

Выполнил студент Халипов Евгений Сергеевич

(Ф.И.О.)

Руководитель проекта ассистент Северинчик Н.А. (учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В .

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант: ассистент Северинчик Н.А. (учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: ассистент Северинчик Н.А. (учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2020

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc59620318)

[1 Обзор аналогов приложений «Vikkru» 4](#_Toc59620319)

[1.1 Приложение «Поиск героев» 4](#_Toc59620320)

[1.3 Выводы по главе «Обзор аналогов приложений «Поиск героев» 5](#_Toc59620321)

[2 Проектирование программного продукта 6](#_Toc59620322)

[2.1 Проектирование мобильного приложения 6](#_Toc59620323)

[2.2 Проектирование базы данных 7](#_Toc59620324)

[3 Программная реализация приложений 10](#_Toc59620325)

[3.1 Технические средства для разработки 10](#_Toc59620326)

[3.2 Разработка баз данных 10](#_Toc59620327)

[3.3 Разработка мобильного приложения 10](#_Toc59620328)

[3.4 Вывод по главе «Программная реализация приложений» 13](#_Toc59620329)

[4 Руководство пользователя 14](#_Toc59620330)

[4.1 Выводы по главе «Руководство пользователя» 20](#_Toc59620331)

[ЗАКЛЮЧЕНИЕ 21](#_Toc59620332)

[Список использованных источников 22](#_Toc59620333)

[ПРИЛОЖЕНИЕ А 23](#_Toc59620334)

[ПРИЛОЖЕНИЕ Б 24](#_Toc59620335)

[ПРИЛОЖЕНИЕ В 27](#_Toc59620336)

# **ВВЕДЕНИЕ**

Цель курсового проекта: предоставление возможности безопасного заполнения анкеты о раскопке

Задачи курсового проекта:

1. Изучить аналогичные мобильные приложения.

Выполнение данной задачи поможет нам сформировать требования к разрабатываемому продукту, определить, какие функциональные возможности предоставит пользователям мобильное приложение, а также выявить недостатки конкурентов, которые мы можем исправить.

1. Спроектировать приложение.

Проектирование приложения позволит нам увидеть картину предстоящей работы целиком, четко разграничить функции и взаимосвязи сущностей. В результате мы получим некоторый план действий, отталкиваясь от которого нам будет проще реализовать проект.

1. Реализовать приложение.

В рамках курсового проекта будет разработано программное средство, обеспечивающее пользователям быстрый доступ к магазину и статусам заказов.

1. Разработать руководство пользователя.

В результате выполнения этой задачи будет составлено руководство пользователя, предназначенное для описания основных возможностей, а также их ограничений. Это руководство нацелено на пользователей данного приложения, для облегчения знакомства с интерфейсом и функционалом.

# **1 Обзор аналогов приложений «Vikkru»**

# **1.1 Приложение «Поиск героев»**

Это приложение, имеющее приятный минималистический дизайн, содержит ряд основных функций, включающих заполнения анкеты. Приложение предоставлено скриншоте ниже (рис 1.1.1).

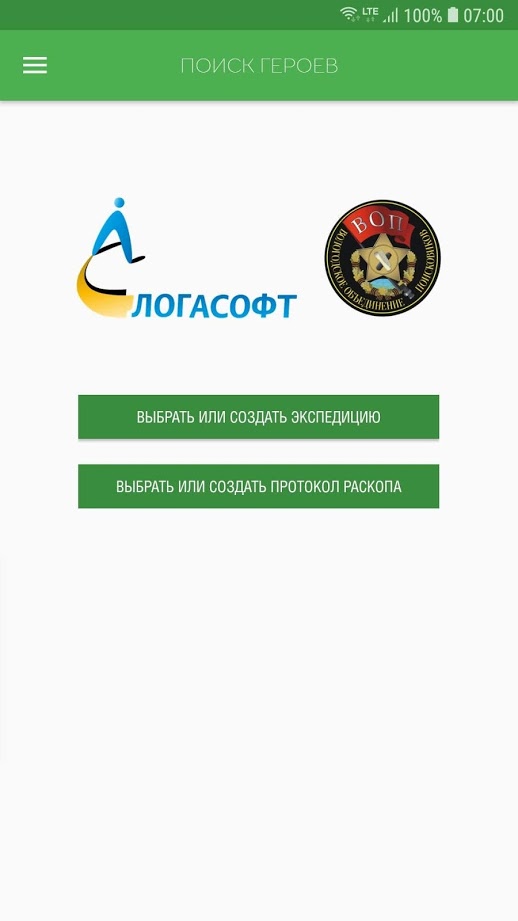
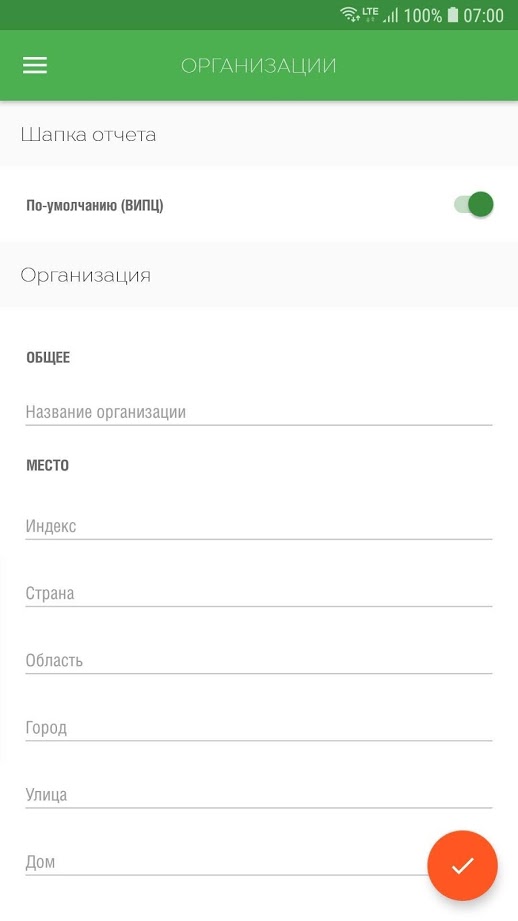
 

Рисунок 1.1.1 – Каталог приложения Поиск Героев

Здесь вы сможете создать экспедицию, а также протокол раскопа, заполнить вполне привлекательную и информативную анкету для информатизации раскопа.

# **1.3 Выводы по главе «Обзор аналогов приложений «Поиск героев»**

В данной главе были рассмотрено приложение «Поиск Героев», которое является аналогом для темы курсового проекта. В результате были сделаны следующие выводы.

Дизайн у приложения «Поиск Героев» более приятный и удобный, интерфейс интуитивно понятный. Структура приложения является более привычной для пользователей, где он может добавлять новые протоколы в одно нажатие. К недостаткам можно отнести то, что нет отправки на сервер

# **2 Проектирование программного продукта**

# **2.1 Проектирование мобильного приложения**

В рамках курсового проекта необходимо разработать мобильное приложение позволяющего пользователям просматривать и безопасно составлять информативную анкету, а также сохранять ее в pdf и отправлять на сервер.

При запуске приложения перед пользователем появляется окно входа и регистрации. После успешной регистрации или авторизации пользователь должен попадать на главный экран. На этом экране отображается выбор формата сохранения. После выбора можно начинать информативно заполнять нашу анкету.

Так же должно быть правое боковое меню, в котором можно будет выбрать текущую анкету и формат.

На первом экране происходит заполнение таких важных деталей как дата эксгумации, фамилия ответственного за раскоп, количество человек на раскопе, а также фамилии всех учувствовавших.

На втором экране мы указываем место обнаружения останков, район, населенный пункт, место, также мы можем получить координаты нашего местоположения по нажатию кнопки «Geolocation».

Далее мы указываем подробности места обнаружения останков. Это включает в себя захоронение по состоянию, тип залегания останков, количество останков, количество захороненных, количество медальонов, количество именных вещей

# **2.2 Проектирование базы данных**

В базе данных данного приложения должны быть реализованы шесть таблиц и четыре представления. Схема базы данных представлена на рисунке 2.2.1.

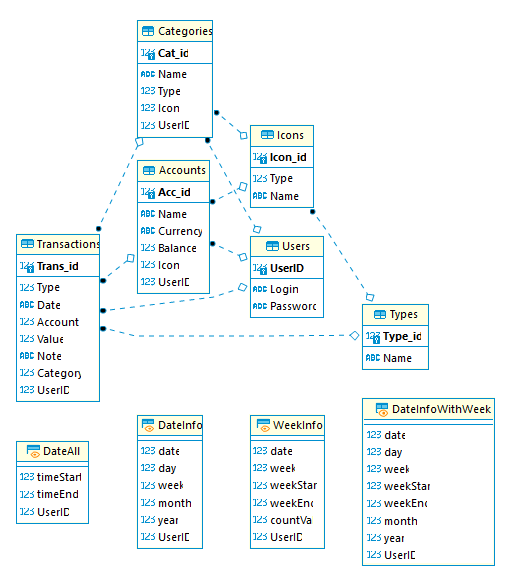


Рисунок 2.2.1 – Схема базы данных

В таблице Categories содержится информация о категориях.

* Cat\_id – первичный ключ, представленный целочисленным значением
* Name – название категории
* Type – тип категории, для расходов или доходов, является вторичным ключом таблицы Types
* Icon – вторичный ключ таблицы Icons, иконка категория
* UserID – вторичный ключ таблицы Users, показывает к какому пользователю относится категория

В таблице Icons содержится информация о иконках.

* Icon\_id – первичный ключ, представленный целочисленным значением
* Type – тип иконки, является вторичным ключом таблицы Types
* Name– значение имя иконки

В таблице Users содержится информация о пользователях.

* UserID – первичный ключ, представленный целочисленным значением
* Login – логин пользователя
* Password – пароль пользователя

В таблице Types содержится информация о типах.

* Type\_id – первичный ключ, представленный целочисленным значением
* Name – имя типа

**2.3 Выводы по главе «Проектирование программного продукта»**

В данной главе было рассмотрено проектирование всех компонентов приложения, был определен основной функционал приложения.

Была спроектирована база данных, где были определены необходимые таблицы и поля для каждой из них. Также были установлены связи между таблицами, которые описаны на схеме базы данных.

# **3 Программная реализация приложений**

# **3.1 Технические средства для разработки**

Для разработки приложений курсового проекта были использованы следующие технические средства:

SQLite – компактная встраиваемая СУБД представляющая собой библиотеку.

Android Studio – интегрированная среда разработки для работы с платформой Android.

Flutter – SDK с открытым исходным кодом для создания мобильных приложений от компании Google. Он используется для разработки приложений под Android и iOS, а также это пока единственный способ разработки приложений под Google Fuchsia.

Dart – язык программирования, созданный Google. Dart позиционируется в качестве замены/альтернативы JavaScript. Один из разработчиков языка Марк Миллер написал, что JavaScript «имеет фундаментальные изъяны», которые невозможно исправить. Поэтому и был создан Dart.

SQLCipher – библиотека с открытым исходным кодом, созданная компанией Zetetic, для прозрачного 256-битного AES шифрования баз данных SQLite.

# **3.2 Разработка баз данных**

Для хранения данных на телефоне будет использоваться встраиваемая СУБД SQLite, а для обеспечения безопасности будет использована библиотека SQLCipher. Код создания структуры мобильной базы данных представлен в Приложении А.

# **3.3 Разработка мобильного приложения**

Мобильное приложение было реализовано в среде разработки – Android Studio. Для написания приложения был использован язык программирования Flutter/Dart.

Графическая структура проекта представлена на рисунке 3.3.1.

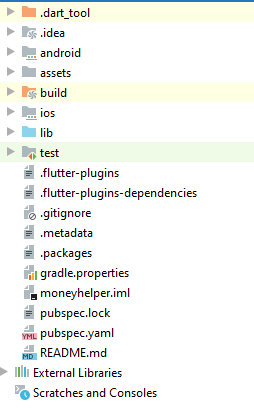


Рисунок 3.3.1 – Графическая структура проекта

Пакет data содержит классы для работы с базой данных и классы моделей, используемых в приложении.

Пакет screens содержит классы всех экранов приложения

Пакет values содержит классы для хранения данных необходимых для работы приложения.

Пакет widgets содержит класс для работы с виджетами.

В основном пакете классы для работы приложения.

Теперь рассмотрим подробнее основные пакеты и классы.

В методе ininDB класса DBProvider предназначен для инициализации базы данных уже подготовленной, если базы данных нет, либо замена существующей. Код представлен на рисунке 3.3.2.

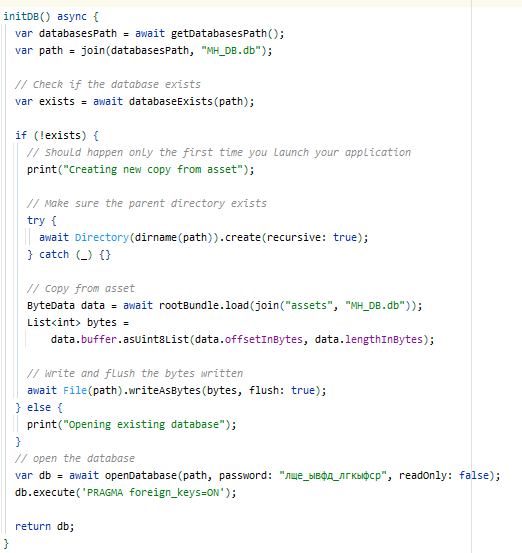


Рисунок 3.3.2 – Пример метода initDB

Пример код модели класса Category. В нем содержатся поля класса, конструкторы для создания экземпляра класса, а также методы для cериализации в json. Код представлен на рисунке 3.3.3.

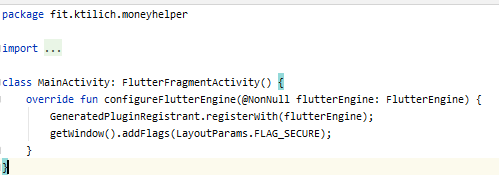


Рисунок 3.3.3 – Код класса MainActivity.kt

# **3.4 Вывод по главе «Программная реализация приложений»**

В данной главе было рассмотрена разработка каждого отдельного компонента приложения.

На первоначальном этапе были рассмотрены технологии, которые применены для разработки курсового проекта.

Вторым этапом была разработка структуры базы данных на стороне клиента.

Третий этапом стала разработка мобильного приложения, где была представлена графическая структура приложения, рассмотрен каждый класс и их методы, в частности.

# **4 Руководство пользователя**

Для того чтобы установить приложения на телефон необходимо иметь версию Android минимум 8.0 и версию API не ниже 28. Версию Android и API вашего телефона можно узнать в настройках телефона.

Если версия Android подходит, тогда можно устанавливать приложение на мобильный телефон.

В меню ваших приложений или на главном экране смартфона появится новая иконка с именем «Yin Wallet». Запустим приложение нажатием на иконку приложения. Откроется страница входа или регистрации. Для того, чтобы зайти в приложении необходимо зарегистрироваться и/или войти в уже созданный профиль. Рисунок 5.1

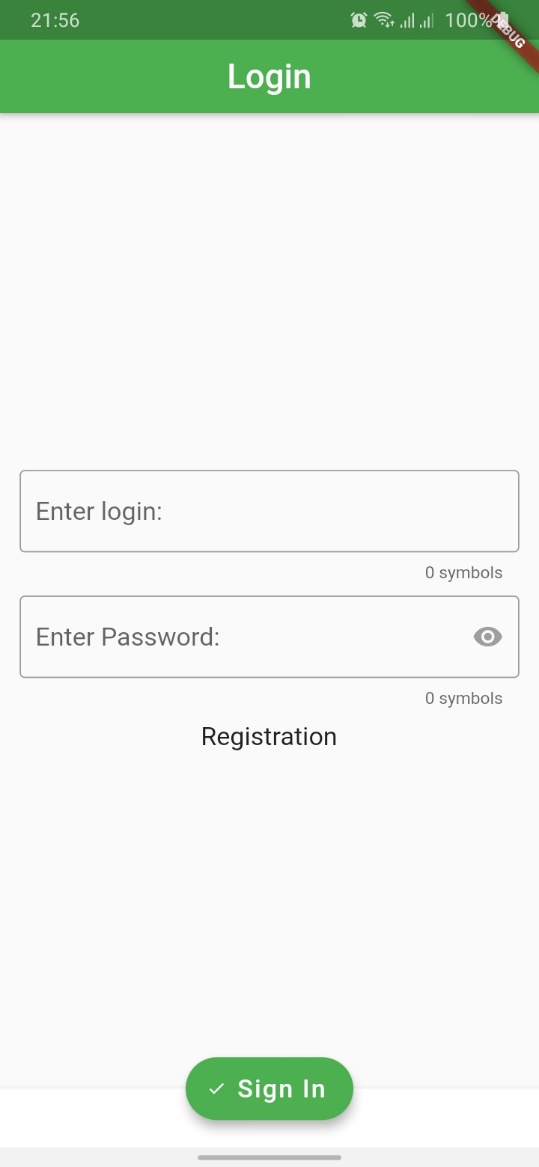
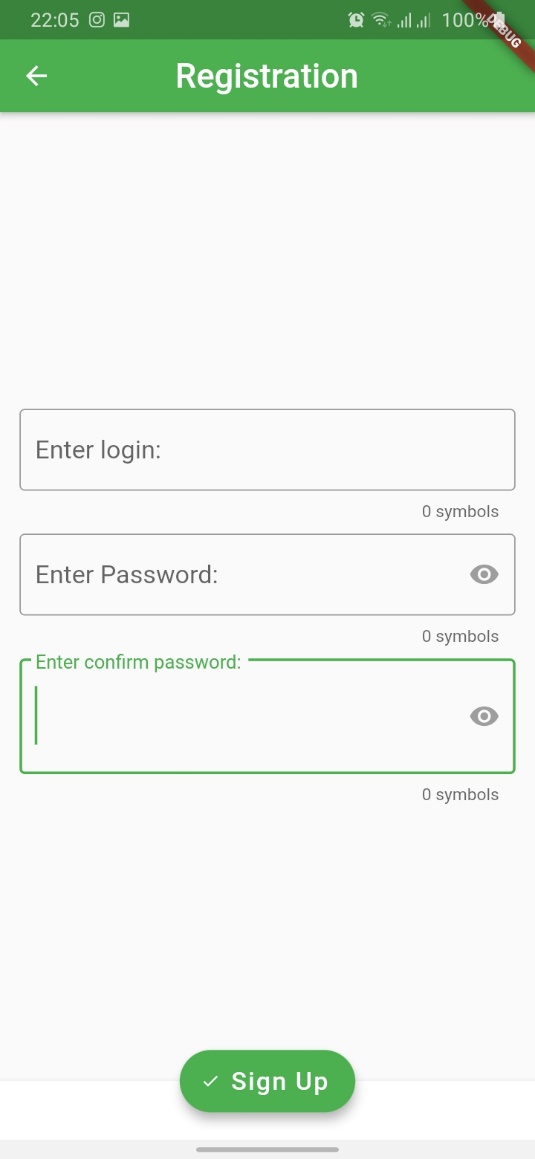
 

Рисунок 5.1 – Экран входа и регистрации

При нажатии кнопки Sing In, а также при открытии свернутого приложения в дальнейшем, появится окно с запросом отпечатка пальца (рис 5.2).

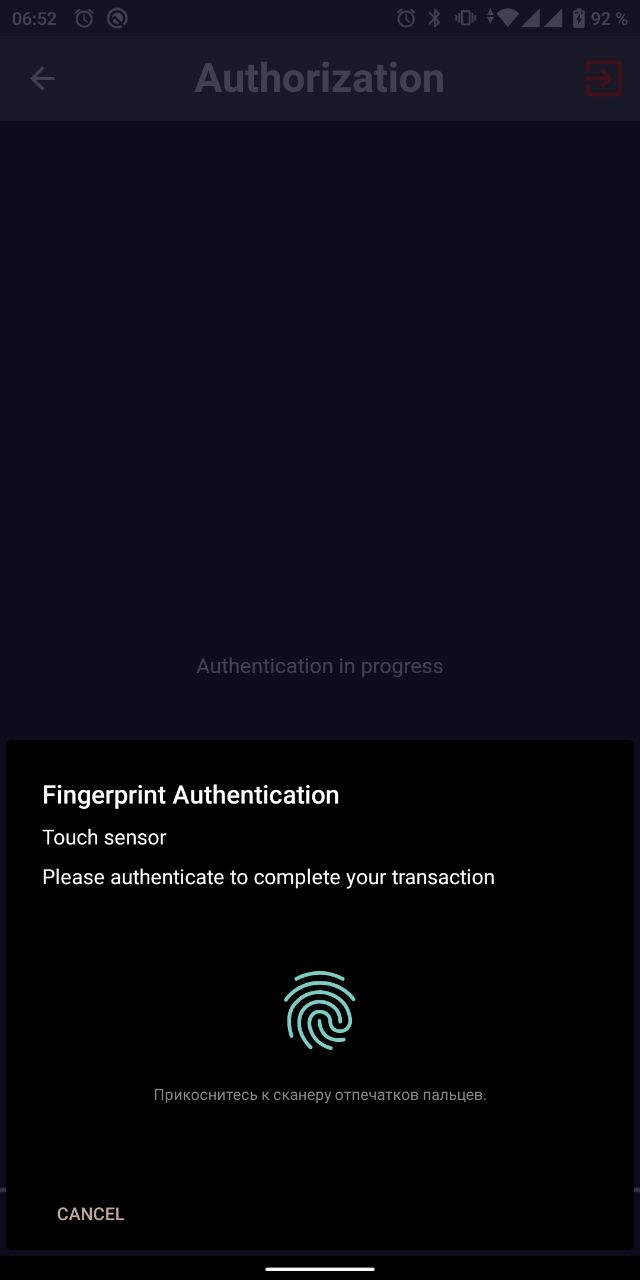


Рисунок 5.2 – Фрагмент аутентификации по отпечатку пальца

После того, как вы войдете или зарегистрируетесь и пройдете аутентификацию по отпечатку пальца, вам откроется домашняя страница приложения (рис. 5.3). Далее у нас откроется начало заполнения нашей анкеты.

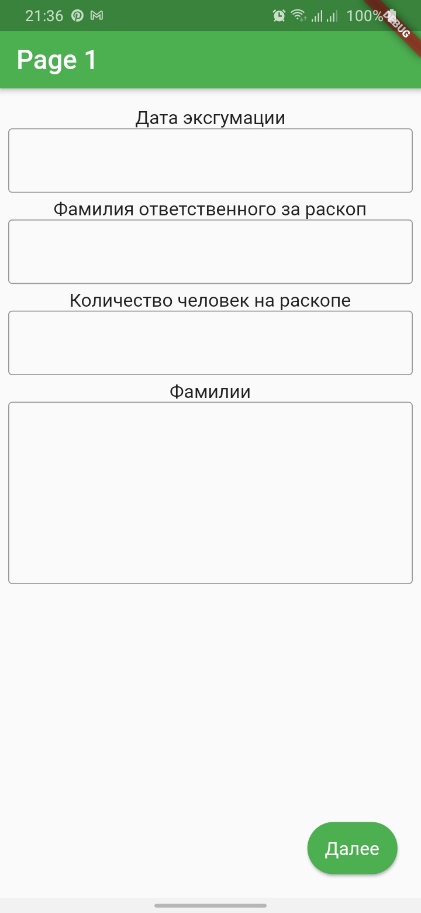


Рисунок 5.3 – Первый экран заполнения

На первом экране происходит заполнение таких важных деталей как дата эксгумации, фамилия ответственного за раскоп, количество человек на раскопе, а также фамилии всех учувствовавших. По нажатию кнопки далее происходит переход на следующую страницу. Демонстрация на рисунок 5.4.



Рисунок 5.4 – Второй экран заполнения

На втором экране мы указываем место обнаружения останков, район, населенный пункт, место, также мы можем получить координаты нашего местоположения по нажатию кнопки «Geolocation». Также мы можем нажать на кнопку назад и перейти на предыдущую страницу. По нажатию кнопки далее происходит переход на следующую страницу. Демонстрация на рисунок 5.5.

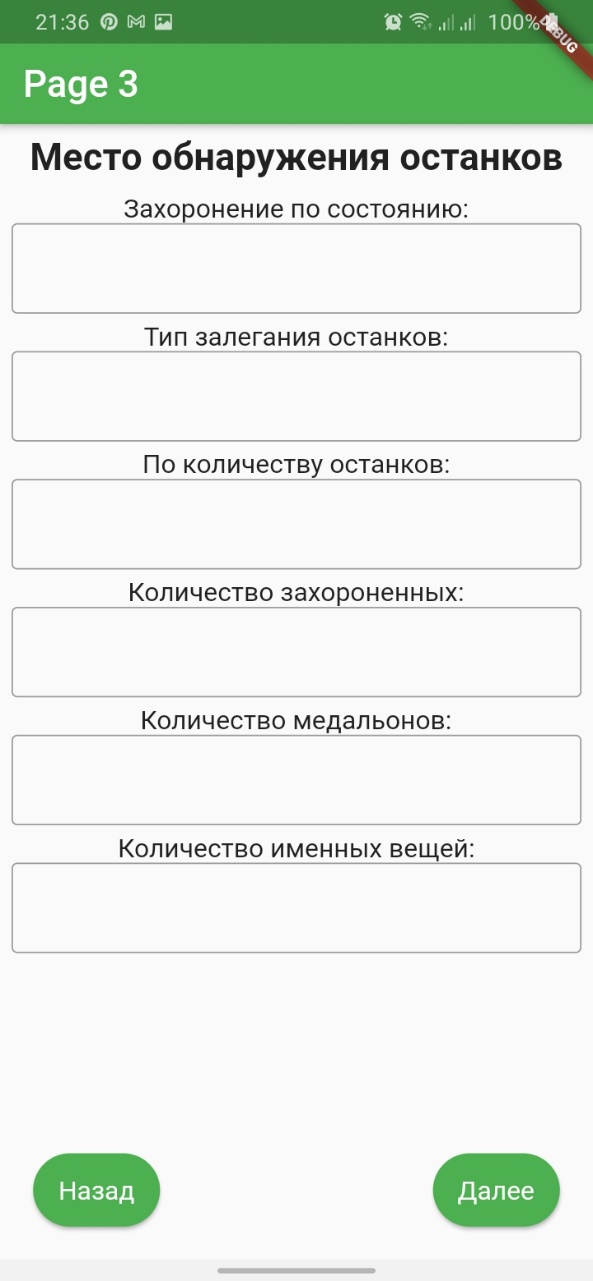
. 

Рисунок 5.5 – Третий экран заполнения

Далее мы указываем подробности места обнаружения останков. Это включает в себя захоронение по состоянию, тип залегания останков, количество останков, количество захороненных, количество медальонов, количество именных вещей. По нажатию кнопки далее происходит переход на следующую страницу. Демонстрация на рисунок 5.6

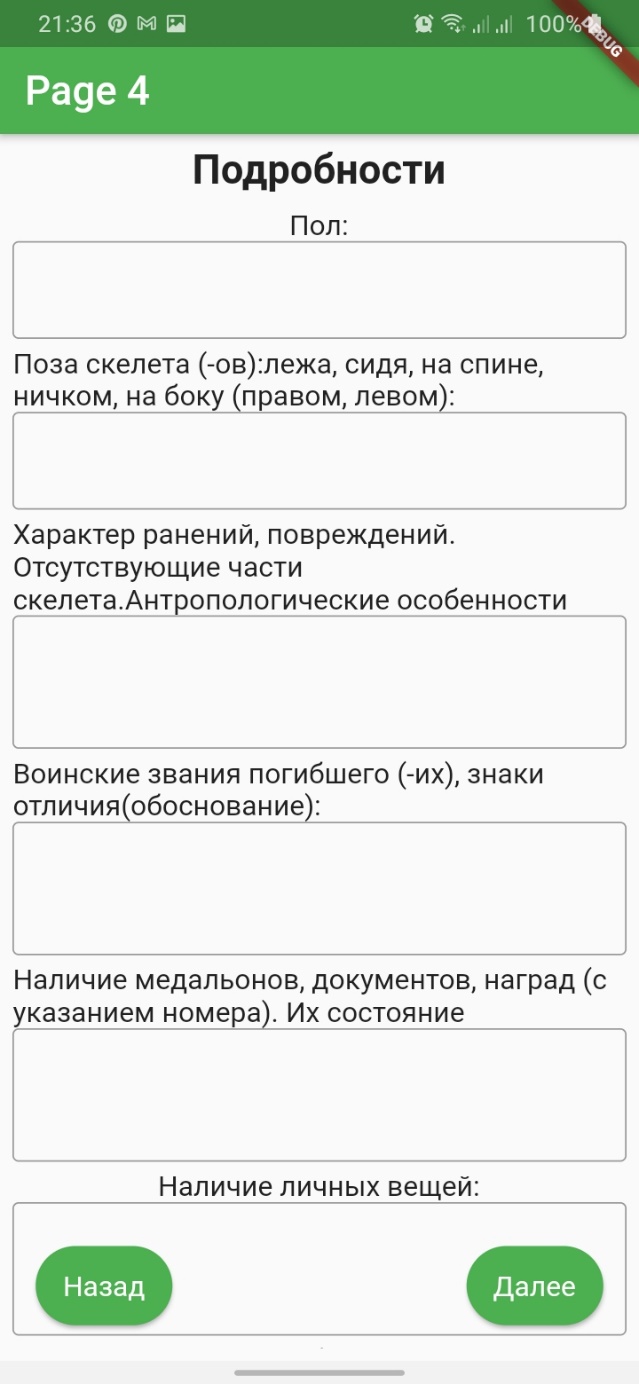
. 

Рисунок 5.6 – Четвертый экран заполнения

Далее мы указываем подробности места обнаружения останков. Это включает в себя захоронение по состоянию, тип залегания останков, количество останков, количество захороненных, количество медальонов, количество именных вещей. По нажатию кнопки далее происходит переход на следующую страницу. Демонстрация на рисунок 5.7



Рисунок 5.7 – Четвертый экран заполнения

На этом экране мы указываем примечания и особенности. По нажатию на плюс, открывается галлерея, где мы можем загрузить фотографии, который прикладываются к нашей анкете.

# **4.1 Выводы по главе «Руководство пользователя»**

В данной главе было описано руководство пользователя.

Данное руководство позволяет пользователю узнать требования для установки приложения на телефон, а также как использовать все функции приложения. Были предоставлены скриншоты самого приложения для большей наглядности использования.

# **ЗАКЛЮЧЕНИЕ**

Целью моего курсового проекта было предоставление возможности учета расходов и доходов для людей, который хотят контролировать свой бюджет.

Были решены следующие поставленные задачи:

* обзор аналогов;

В результате было рассмотрено одно приложение и были выявлении его положительные и отрицательные стороны. Эта информация была использована для улучшения функционала приложения и пользовательского интерфейса.

* проектирование приложения;

Был спроектирован каждый отдельный компонент приложения. В приложении был определен основной функционал, также были разработаны необходимые диаграммы. При проектировании базы данных, были определены необходимые таблицы и разработана схема базы данных.

* реализация приложения;

Данная задача была реализована с помощью следующих технических средств: СУБД SQLite и MySQL, сред разработки Android Studio, языков программирования Dart/Flutter.

Была реализована зашифрованная база данных на клиенте.

При решении данной задачи была достигнута поставленная цель, и был создан проект «Vikkru», который включает в себя мобильное приложение.

* разработка руководства пользователя.

Разработано руководство, позволяющее пользователю узнать требования для установки приложения на телефон, а также как использовать все функции мобильного приложения.

В соответствии с поставленными задачами и полученным результатом можно сделать вывод, что задачи были выполнены в полном объеме.

# **Список использованных источников**

1. Stackoverflow - https://stackoverflow.com;

2. Cyberforum - https://cyberforum.ru;

3. Github - https://github.com;

4. Android documentation - https://developer.android.com/docs.

5. Flutter documentation - https://flutter.dev/

6. ASP.NET documentation - https://docs.microsoft.com/en-us/aspnet/core/

7. SQLServer documentation - https://docs.microsoft.com/en-us/sql/sql-server/

**ПРИЛОЖЕНИЕ А**

Код создания структуры мобильной базы данных

PRAGMA foreign\_keys = ON

drop TABLE Icons

CREATE TABLE Types(

Type\_id INTEGER,

Name TEXT,

PRIMARY KEY (Type\_id)

);

CREATE TABLE Icons (

Icon\_id INTEGER,

Type INTEGER,

Name TEXT,

PRIMARY KEY (Icon\_id),

FOREIGN KEY (Type) REFERENCES Types(Type\_id)

);

CREATE TABLE Categories (

Cat\_id INTEGER,

Name TEXT,

Type INTEGER,

Icon INTEGER,

PRIMARY KEY (Cat\_id),

FOREIGN KEY (Icon) REFERENCES Icons(Icon\_id)

);

drop table Transactions

CREATE TABLE Transactions (

Trans\_id INTEGER,

Type INTEGER,

Date TEXT,

Account INTEGER,

Value REAL,

Note TEXT,

Category INTEGER,

PRIMARY KEY (Trans\_id),

FOREIGN KEY (Type) REFERENCES Types(Type\_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (Account) REFERENCES Accounts(Acc\_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (Category) REFERENCES Categories(Cat\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

drop view WeekInfo

create view WeekInfo as

select

Date(t.Date) date,

strftime('%W', t.Date ) week,

max(date(t.Date, 'weekday 0', '-7 day')) weekStart,

max(date(t.Date, 'weekday 0', '-1 day')) weekEnd,

count(\*) as countVal

from Transactions t

group by strftime('%Y', t.Date ), week

order by date;

select \* from Transactions t

drop view DateInfo

CREATE VIEW DateInfo as SELECT DISTINCT

DATE(t.Date) as date,

strftime('%j', t.Date) as day,

strftime('%W', t.Date) as week,

strftime('%m', t.Date) as month,

strftime('%Y', t.Date) as year

FROM Transactions t order by date

drop view DateInfoWithWeek

create view DateInfoWithWeek as

select wi.date, di."day", wi.week, wi.weekStart, wi.weekEnd, di."month", di."year"

from WeekInfo wi

join DateInfo di

on wi.date = di.date

select \* from DateInfoWithWeek diww

create view DateAll as

select

min(date(t.Date)) timeStart,

max(date(t.Date)) timeEnd

from Transactions t;

select

Date,

strftime('%W', Date) WeekNumber,

max(date(Date, 'weekday 0', '-7 day')) WeekStart,

max(date(Date, 'weekday 0', '-1 day')) WeekEnd,

count(\*) as GroupedValues

from Transactions

group by WeekNumber;

SELECT \* FROM Transactions t where strftime('%W', t.Date) = strftime('%W', DATE('2020-05-12 00:00:00.000')) and strftime('%Y', t.Date) = strftime('%Y', DATE('2020-05-12 00:00:00.000')) order by t.Date

**ПРИЛОЖЕНИЕ Б**

Код класса Database

PRAGMA foreign\_keys = ON

drop TABLE Icons

CREATE TABLE Types(

Type\_id INTEGER,

Name TEXT,

PRIMARY KEY (Type\_id)

);

CREATE TABLE Icons (

Icon\_id INTEGER,

Type INTEGER,

Name TEXT,

PRIMARY KEY (Icon\_id),

FOREIGN KEY (Type) REFERENCES Types(Type\_id)

);

CREATE TABLE Accounts (

Acc\_id INTEGER,

Name TEXT,

Currency TEXT,

Balance REAL,

Icon INTEGER,

PRIMARY KEY (Acc\_id),

FOREIGN KEY (Icon) REFERENCES Icons(Icon\_id)

);

CREATE TABLE Categories (

Cat\_id INTEGER,

Name TEXT,

Type INTEGER,

Icon INTEGER,

PRIMARY KEY (Cat\_id),

FOREIGN KEY (Icon) REFERENCES Icons(Icon\_id)

);

drop table Transactions

CREATE TABLE Transactions (

Trans\_id INTEGER,

Type INTEGER,

Date TEXT,

Account INTEGER,

Value REAL,

Note TEXT,

Category INTEGER,

PRIMARY KEY (Trans\_id),

FOREIGN KEY (Type) REFERENCES Types(Type\_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (Account) REFERENCES Accounts(Acc\_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (Category) REFERENCES Categories(Cat\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

drop view WeekInfo

create view WeekInfo as

select

Date(t.Date) date,

strftime('%W', t.Date ) week,

max(date(t.Date, 'weekday 0', '-7 day')) weekStart,

max(date(t.Date, 'weekday 0', '-1 day')) weekEnd,

count(\*) as countVal

from Transactions t

group by strftime('%Y', t.Date ), week

order by date;

select \* from Transactions t

drop view DateInfo

CREATE VIEW DateInfo as SELECT DISTINCT

DATE(t.Date) as date,

strftime('%j', t.Date) as day,

strftime('%W', t.Date) as week,

strftime('%m', t.Date) as month,

strftime('%Y', t.Date) as year

FROM Transactions t order by date

drop view DateInfoWithWeek

create view DateInfoWithWeek as

select wi.date, di."day", wi.week, wi.weekStart, wi.weekEnd, di."month", di."year"

from WeekInfo wi

join DateInfo di

on wi.date = di.date

select \* from DateInfoWithWeek diww

create view DateAll as

select

min(date(t.Date)) timeStart,

max(date(t.Date)) timeEnd

from Transactions t;

select

Date,

strftime('%W', Date) WeekNumber,

max(date(Date, 'weekday 0', '-7 day')) WeekStart,

max(date(Date, 'weekday 0', '-1 day')) WeekEnd,

count(\*) as GroupedValues

from Transactions

group by WeekNumber;

SELECT \* FROM Transactions t where strftime('%W', t.Date) = strftime('%W', DATE('2020-05-12 00:00:00.000')) and strftime('%Y', t.Date) = strftime('%Y', DATE('2020-05-12 00:00:00.000')) order by t.Date

**ПРИЛОЖЕНИЕ В**

Код класса MyApp

import 'package:flutter/material.dart';

import 'package:flutter/services.dart';

import 'package:shared\_preferences/shared\_preferences.dart';

import 'data/DataModel.dart' as model;

import 'screens/accounts\_screen.dart';

import 'screens/add\_edit\_account\_screen.dart';

import 'screens/add\_edit\_category\_screen.dart';

import 'screens/add\_edit\_transaction\_screen.dart';

import 'screens/categories\_screen.dart';

import 'screens/home\_screen.dart';

import 'screens/login\_screen.dart';

import 'screens/registration\_screen.dart';

import 'screens/settings\_screen.dart';

import 'screens/auth\_screen.dart';

import 'values/theme.dart';

import 'package:local\_auth/local\_auth.dart';

class MyApp extends StatefulWidget {

const MyApp({Key key}) : super(key: key);

final String appName = 'Money Helper';

static final String routeHome = '/';

static final String routeCategories = '/category';

static final String routeAddEditCategory = '/category/add\_edit';

static final String routeAccounts = '/account';

static final String routAddEditAccount = '/account/add\_edit';

static final String routeAddEditTransaction = '/transaction/add\_edit';

static final String routeSettings = '/settings';

static final String routeLogin = '/login';

static final String routeRegistration = '/registration';

static final String authScreen = '/auth';

static final String userPrefKey = 'user';

static int curUser = -1;

@override

\_MyAppState createState() => \_MyAppState();

}

class \_MyAppState extends State<MyApp> {

SharedPreferences \_preferences;

DateTime start, end;

bool isInit = true;

@override

void initState() {

super.initState();

start = DateTime.now();

end = start.add(Duration(milliseconds: 300));

}

init() {

if (isInit) {

SharedPreferences.getInstance()

..then((prefs) {

setState(() {

this.\_preferences = prefs;

\_loadUserPref();

});

});

}

}

@override

Widget build(BuildContext context) {

SystemChrome.setPreferredOrientations([

DeviceOrientation.portraitUp,

]);

return FutureBuilder<void>(

future: init(),

builder: (context, snapshot) {

if (DateTime.now().millisecondsSinceEpoch <=

end.millisecondsSinceEpoch) {

return Center(

child: CircularProgressIndicator(

backgroundColor: appTheme.accentColor,

),

);

}

return MaterialApp(

initialRoute:

MyApp.curUser == -1 ? MyApp.routeLogin : MyApp.authScreen,

routes: {

MyApp.routeHome: (BuildContext context) => HomeScreen(),

MyApp.routeCategories: (BuildContext context) => CategoriesScreen(),

MyApp.routeAddEditCategory: (BuildContext context) =>

AddEditCategoryScreen(),

MyApp.routeAccounts: (BuildContext context) => AccountsScreen(),

MyApp.routAddEditAccount: (BuildContext context) =>

AddEditAccountScreen(),

MyApp.routeAddEditTransaction: (BuildContext context) =>

AddEditTransactionScreen(),

MyApp.routeSettings: (BuildContext context) => SettingsScreen(),

MyApp.routeLogin: (BuildContext context) => LoginScreen(),

MyApp.routeRegistration: (BuildContext context) =>

RegistrationScreen(),

MyApp.authScreen: (BuildContext context) =>

AuthScreen(),

},

debugShowCheckedModeBanner: false,

theme: appTheme,

);

},

);

}

void \_loadUserPref() {

setState(() {

MyApp.curUser = this.\_preferences.getInt(MyApp.userPrefKey) ?? -1;

});

}

}

// You can pass any object to the arguments parameter. In this example,

// create a class that contains both a customizable title and message.

class ScreensArguments {

final TypeOperation operation;

Type type;

model.Category category;

model.Category category\_1;

model.Account account;

model.Account account\_1;

model.Transaction transaction;

model.Transaction transaction\_1;

ScreensArguments(

{@required this.operation,

this.type,

this.category,

this.category\_1,

this.account,

this.account\_1,

this.transaction,

this.transaction\_1});

}

class DateFromDB {}

class DateInfo implements DateFromDB {

DateTime date;

int day;

int week;

int month;

int year;

DateInfo({this.date, this.day, this.week, this.month, this.year});

DateInfo.fromMap(Map<String, dynamic> map) {

date = DateTime.parse(map[model.vColDDate]);

day = int.parse(map[model.vColDDay]);

week = int.parse(map[model.vColDDay]);

month = int.parse(map[model.vColDMonth]);

year = int.parse(map[model.vColDYear]);

}

Map<String, dynamic> toMap() {

var map = <String, dynamic>{

model.vColDDate: date,

model.vColDDay: day,

model.vColDWeek: week,

model.vColDMonth: month,

model.vColDYear: year,

};

return map;

}

}

class DateInfoWithWeek implements DateInfo {

DateTime date;

int day;

int week;

int month;

int year;

DateTime weekStart;

DateTime weekEnd;

DateInfoWithWeek({

this.date,

this.day,

this.week,

this.month,

this.year,

this.weekStart,

this.weekEnd,

});

DateInfoWithWeek.fromMap(Map<String, dynamic> map) {

date = DateTime.parse(map[model.vColWIDate]);

day = int.parse(map[model.vColDDay]);

week = int.parse(map[model.vColWIWeek]);

month = int.parse(map[model.vColDMonth]);

year = int.parse(map[model.vColDYear]);

weekStart = DateTime.parse(map[model.vColWIStart]);

weekEnd = DateTime.parse(map[model.vColWIEnd]);

}

Map<String, dynamic> toMap() {

var map = <String, dynamic>{

model.vColWIDate: date,

model.vColDDay: day,

model.vColWIWeek: week,

model.vColDMonth: month,

model.vColDYear: year,

model.vColWIStart: weekStart,

model.vColWIEnd: weekEnd,

};

return map;

}

}

class DateAll implements DateFromDB {

DateTime timeStart;

DateTime timeEnd;

DateAll({this.timeStart, this.timeEnd});

DateAll.fromMap(Map<String, dynamic> map) {

timeStart = DateTime.parse(map[model.vColDateAllStart]);

timeEnd = DateTime.parse(map[model.vColDateAllEnd]);

}

Map<String, dynamic> toMap() {

var map = <String, dynamic>{

model.vColDateAllStart: timeStart,

model.vColDateAllEnd: timeEnd,

};

return map;

}

}

// удалить

class WeekInfo implements DateFromDB {

int weekNUmber;

DateTime start;

DateTime end;

int count;

WeekInfo({this.weekNUmber, this.start, this.end, this.count});

WeekInfo.fromMap(Map<String, dynamic> map) {

weekNUmber = int.parse(map[model.vColWIWeek]);

start = DateTime.parse(map[model.vColWIStart]);

end = DateTime.parse(map[model.vColWIEnd]);

weekNUmber = int.parse(map[model.vColWIWeek]);

}

Map<String, dynamic> toMap() {

var map = <String, dynamic>{

model.vColWIWeek: weekNUmber,

model.vColWIStart: start,

model.vColWIEnd: end,

model.vColWICount: count

};

return map;

}

}

enum TypeOperation { Add, Edit, Remove }

enum Type { Income, Expense, Transfer }

enum Period { Day, Week, Month, Year, All, ExactDate }