**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Сибирский государственный университет науки и технологий**

**имени академика М.Ф. Решетнева»**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

институт/факультет/подразделение

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

кафедры/цикловая комиссии

**КУРСОВОЙ ПРОЕКТ**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

тема проекта (работы)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

|  |  |  |  |
| --- | --- | --- | --- |
| Руководитель |  |  |  |
|  | подпись, дата |  | инициалы и фамилия |
|  |  |  |  |
| Обучающийся\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |  |  |
| номер группы, зачетной книжки | подпись |  | инициалы и фамилия |

Красноярск 2019 г.

**Содержание**

[Введение 3](#_Toc27649767)

[1. Структура данных 4](#_Toc27649768)

[2. Разработка интерфейса программного обеспечения 9](#_Toc27649769)

[3. Структура интерфейса 21](#_Toc27649770)

[Заключение 25](#_Toc27649771)

[Список использованных источников 26](#_Toc27649772)

# **Введение**

В современном мире трудно представить любую деятельность без использования средств вычислительной техники и сети интернет. Ежедневно приходиться сталкиваться с различными проблемами, поломками, которые ведут к неисправностям в вычислительные техники. Зачастую проблема решается перезагрузкой устройства или проверкой соединения.

В настоящее время существует широкий спектр специализированных программных продуктов и экспертных систем для применения в данной предметной области, среди которых можно выделить такие программные системы как Fusion Server Tools, Fusion Server Tools Info Collect,

Данные экспертные системы являются системным программным обеспечением (ПО), что увеличивает время работы инженера и снижает объем выполненных работ по устранению неисправностей из-за отсутствия своей мобильности.

Главная цель проектируемого мобильного приложения – упростить процесс диагностирования для технического - инженера, за счет автоматизации и систематизации накопленного специалистами опыта в виде базы знаний экспертной системы.

Разработанная экспертная система должна следующие функции:

* выявление неисправностей в работе вычислительной техники;
* поиск неисправностей вычислительной техники в мануалах;

Разработанный продукт должен быть ориентирован на технических специалистов, имеющих опыт по использованию вычислительной техники и профессионеально разбирающихся в данной предметной области.

1. **Структура данных**

В разрабатываемом приложении для диагностики неисправностей вычислительной техники использовалась база данных SQLite.

SQLite - встраиваемая реляционная база данных. SQLite является совершенно бесплатной для использования в любых проектах. SQLite состоит из библиотеки, которая добавляется к разрабатываемой программе и таким образом становится её составной частью. Таким образом, в качестве протокола обмена используются вызовы функций библиотеки SQLite.

Такой подход уменьшает накладные расходы, время отклика и упрощает программу. SQLite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том устройстве, на котором исполняется программа.

Таблица - объект, который позволяет хранить данные.

Для каждой таблицы можно определить первичный ключ (одно или несколько полей, имеющих уникальные значения в каждой записи) и один или несколько индексов, позволяющих ускорить доступ к данным.

Таблицы базы данных предназначены для хранения данных, необходимых для выполнения некоторых задач приложения. Любая таблица состоит из столбцов или полей, каждое из которых содержит определенный род сведений (например, наименование неисправности), и строк или записей, содержащих наборы проблем, наводящих вопросов и путей решения различного рода неисправностей. Таблицы базы данных приведены на рисунках 1,2.

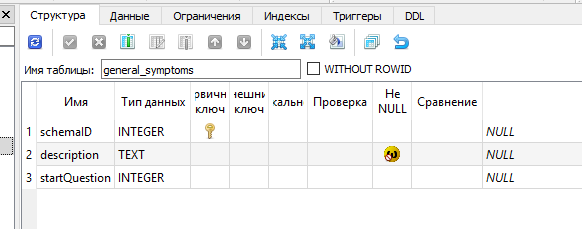


Рисунок 1 – Таблица «Главные симптомы»

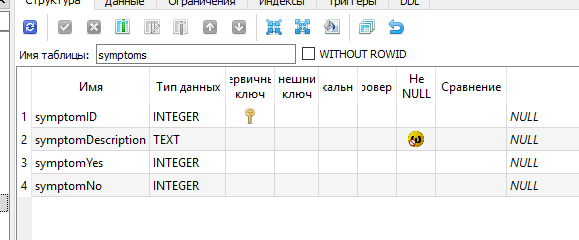


Рисунок 2 – Таблица «Наводящие вопросы»

Работа с базой данных осуществляется через класс DatabaseHelper.java.

Данный класс позволяет связать приложение с базой данных SQLite, формирует строку для создания таблиц в базе данных, а также проверяет существование базы данных с указанным именем.

Ниже приведен фрагмент кода, отвечающий за работу приложения с базой данных:

//Класс для работы с базой данных

public class DatabaseHelper extends SQLiteOpenHelper implements BaseColumns{

public static String DATABASE\_PATH = "data/data/ovsyankin.com/databases/"; //Путь к файлу БД

public static final String DATABASE\_NAME = "diagnostic2.db"; //Имя файла БД

public static final int DATABASE\_VERSION = 1; //Версия БД

public static final String DB\_MAIN\_TABLE = "general\_symptoms"; //Имя основной таблицы в БД

public static final String SCHEMA\_ID = "schemaID";

public static final String SCHEMA\_DESCRIPTION = "description";

public static final String START\_QUESTION = "startQuestion";

public static final String DATABASE\_TABLE = "symptoms"; //Имя таблицы с симптомами в БД

public static final String SYMPTOM\_ID = "symptomID";

public static final String SYMPTOM\_DESCRIPTION = "symptomDescription";

public static final String SYMPTOM\_YES = "symptomYes";

public static final String SYMPTOM\_NO = "symptomNo";

private SQLiteDatabase myDataBase;

private final Context myContext;

//Формируем строку запроса для создания таблицы в базе данных

public static final String DATABASE\_CREATE\_SCRIPT = "create table "

+ DATABASE\_TABLE + " (" + BaseColumns.\_ID

+ " integer primary key autoincrement, " + SYMPTOM\_ID

+ " integer, " + SYMPTOM\_DESCRIPTION + " text not null, "

+ SYMPTOM\_YES + " integer, " + SYMPTOM\_NO + " integer);";

public DatabaseHelper(Context context){

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

this.myContext = context;

}

public DatabaseHelper(Context context, String name, SQLiteDatabase.CursorFactory factory,

int version){

super(context, name, factory, version);

this.myContext = context;

}

@Override

public void onCreate(SQLiteDatabase db) {

db.execSQL(DATABASE\_CREATE\_SCRIPT);

}

@Override

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

Log.w("SQLite", "Обновляемся с версии " + oldVersion + " на версию " + newVersion);

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

db.execSQL("DROP TABLE IF EXISTS " + DB\_MAIN\_TABLE);

onCreate(db);

}

//Создаем базу

public void createDataBase()throws IOException{

boolean dbExist = checkDataBase();

if(dbExist){

//Если база существует то ничего не делать

}else {

this.getReadableDatabase();

try{

copyDataBase();

}catch (IOException e){

throw new Error("Ошибка при копировании базы");

}

}

}

//Проверяем существование БД с указанным именем по указанному пути

private boolean checkDataBase(){

SQLiteDatabase checkDB = null;

try {

String myPath = DATABASE\_PATH + DATABASE\_NAME;

checkDB = SQLiteDatabase.openDatabase(myPath, null, SQLiteDatabase.OPEN\_READONLY);

}catch (SQLiteException e){

}

if(checkDB != null){

checkDB.close();

}

return checkDB != null ? true : false;

}

1. **Разработка интерфейса программного обеспечения**

Для начала работы с программой необходимо запустить приложение «Диагностика ПК». Далее нужно зарегистрироваться или осуществить вход в аккаунт для входа в приложение (Рисунок 3).

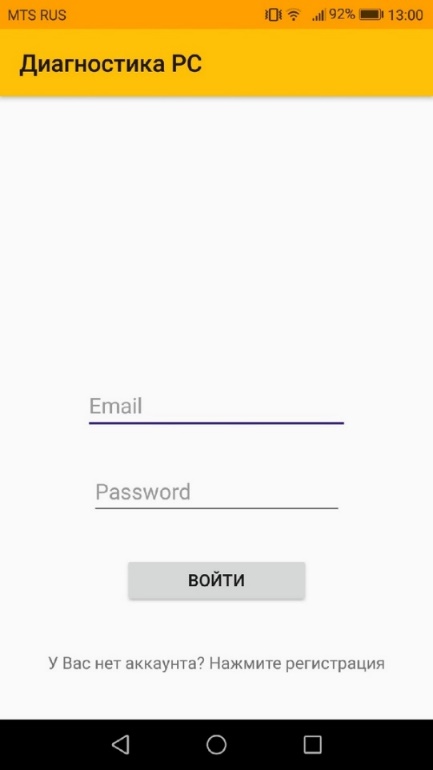
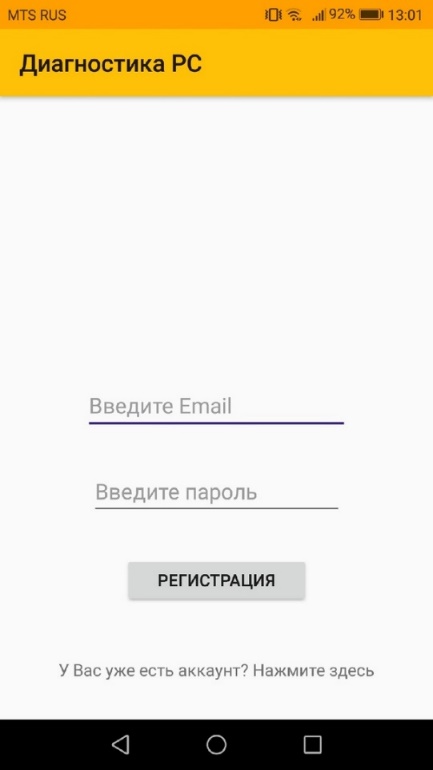


Рисунок 3 – Окно аутентификации

Регистрация аккаунта или вход в аккаунт осуществляется через сервис Firebase.

Firebase - это кроссплатформенная платформа мобильной базы данных в режиме реального времен. Firebase, опираясь на Google, берет на себя сложность в работе с базами данных реального времени, аутентификацией пользователей и работой с автономными рабочими процессами синхронизации.

Firebase Authentication предоставляет базовые службы, простые в использовании SDK и готовые библиотеки пользовательского интерфейса для аутентификации пользователей в приложении. Он поддерживает аутентификацию с использованием паролей, телефонных номеров, популярных провайдеров идентификации, таких как Google, Facebook, Twitter и др. (Рисунок 4).

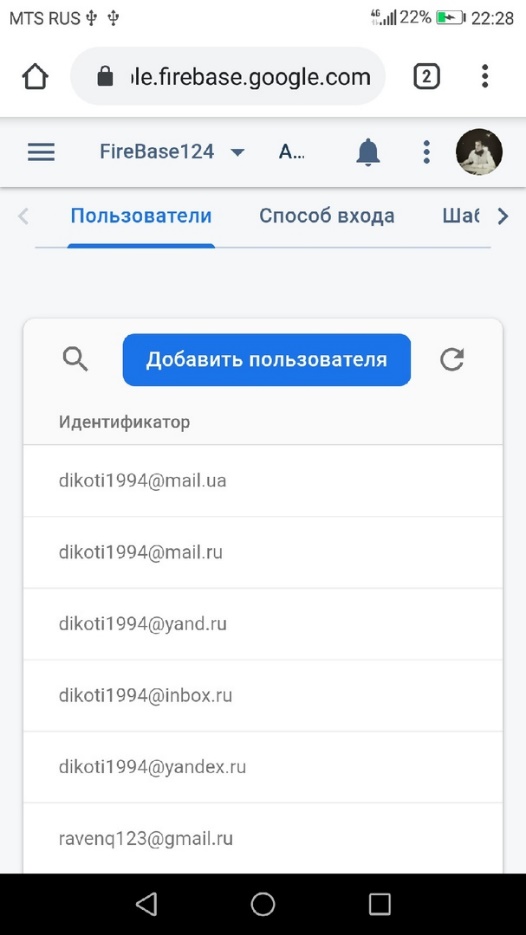


Рисунок 4 – Список зарегистрированных пользователей в Firebase

Регистрация пользователей осуществляется при помощи класса Registr.java. Ниже приведен фрагмент кода, отвечающий за регистрацию пользователей в сервисе Firebase:

public class Registr extends AppCompatActivity {

EditText emailId, password;

Button btnSignUp;

TextView tvSignIn;

FirebaseAuth mFirebaseAuth;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_registr);

mFirebaseAuth = FirebaseAuth.getInstance();

emailId = findViewById(R.id.editText);

password = findViewById(R.id.editText2);

btnSignUp = findViewById(R.id.button);

tvSignIn = findViewById(R.id.textView);

btnSignUp.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

String email = emailId.getText().toString();

String pwd = password.getText().toString();

if (email.isEmpty()) {

emailId.setError("Пожалуйста введите логин");

emailId.requestFocus();

}

else if (pwd.isEmpty()) {

password.setError("Пожалуйста введите пароль");

password.requestFocus();

}

else if (email.isEmpty() && pwd.isEmpty()) {

Toast.makeText(Registr.this, "Пустые поля", Toast.LENGTH\_SHORT).show();

}

else if (!(email.isEmpty() && pwd.isEmpty())) {

FirebaseAuth.createUserWithEmailAndPassword(email,pwd).addOnCompleteListener(Registr.this, new OnCompleteListener<AuthResult>() {

@Override

public void onComplete(@NonNull Task<AuthResult> task)

if (!task.isSuccessful()) {

Toast.makeText(Registr.this, "Зарегистрироваться не удалось, попробуйте еще раз!", Toast.LENGTH\_SHORT).show();

}

else {

startActivity(new Intent(Registr.this,MainActivity.class));

}

}

});

}

else {

Toast.makeText(Registr.this, "Произошла ошибка!", Toast.LENGTH\_SHORT).show();

}

}

});

tvSignIn.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent i = new Intent(Registr.this,LoginActivity.class);

startActivity(i);

}

});

Работа с базой пользователей осуществляется через подключаемый в проекте файл google-services.json. Ниже приведен фрагмент кода файла google-services.json:

"project\_info": {

"project\_number": "347664860781",

"firebase\_url": "https://fir-124-8a1c1.firebaseio.com",

"project\_id": "fir-124-8a1c1",

"storage\_bucket": "fir-124-8a1c1.appspot.com"

},

"client": [

{

"client\_info": {

"mobilesdk\_app\_id": "1:347664860781:android:a905e49de5ef191e256164",

"android\_client\_info": {

"package\_name": "ovsyankin.com"

}

},

"oauth\_client": [

{

"client\_id": "347664860781-n8oj99ep8kv8q10q0rrhvpdr2mkon6ao.apps.googleusercontent.com",

"client\_type": 1,

"android\_info": {

"package\_name": "ovsyankin.com",

"certificate\_hash": "39820397e1acaffd187c63507f31510d999743e0"

}

},

{

"client\_id": "347664860781-r51ni043acocj23fvjshhfv4fv4taetr.apps.googleusercontent.com",

"client\_type": 1,

"android\_info": {

"package\_name": "ovsyankin.com",

"certificate\_hash": "d494ce04560f5bc9ada7ec96a779d4db5510b9e8"

}

},

{

"client\_id": "347664860781-7np17dtqq4ckcocnbhqr7u1892d9ekfs.apps.googleusercontent.com",

"client\_type": 3

}

],

"api\_key": [

{

"current\_key": "AIzaSyAGgpUV3-44yPlJKeVENKFfcPD2sPKXqeM"

}

],

"services": {

"appinvite\_service": {

"other\_platform\_oauth\_client": [

{

"client\_id": "347664860781-7np17dtqq4ckcocnbhqr7u1892d9ekfs.apps.googleusercontent.com",

"client\_type": 3

}

]

}

}

}

],

"configuration\_version": "1"

}

Также в приложении реализован раздел обратной связи при помощи которого можно отправлять почтовые сообщения (Рисунок 5). Ниже приведен код класса Directory.java отвечающий за отправку e-mail:

public class Directory extends AppCompatActivity {

private EditText mEditTextTo;

private EditText mEditTextSubject;

private EditText mEditTextMessage;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_directory);

mEditTextTo = (EditText) findViewById(R.id.edit\_text\_to);

mEditTextSubject = (EditText) findViewById(R.id.edit\_text\_subject);

mEditTextMessage = (EditText) findViewById(R.id.edit\_text\_message);

Button buttonSend = (Button) findViewById(R.id.button\_send);

buttonSend.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

sendMail();

}

});

}

private void sendMail() {

String recipientList = mEditTextTo.getText().toString();

String[] recipients = recipientList.split(",");

String subject = mEditTextSubject.getText().toString();

String message = mEditTextMessage.getText().toString();

Intent intent = new Intent(Intent.ACTION\_SEND);

intent.putExtra(Intent.EXTRA\_EMAIL, recipients);

intent.putExtra(Intent.EXTRA\_SUBJECT, subject);

intent.putExtra(Intent.EXTRA\_TEXT, message);

intent.setType("message/rfc822");

startActivity(Intent.createChooser(intent, "Choose an email client"));

}

}

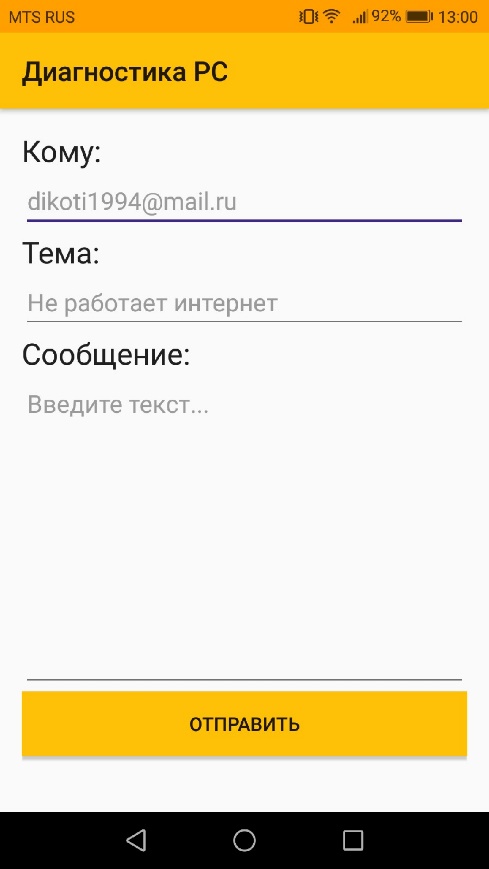


Рисунок 5 – Форма отправки электронного сообщения

В разрабатываемом приложении также реализована диагностика неисправностей вычислительной техники.

В соответствии с диагностической схемой, выбранной в главном окне информационной системы, на опросное окно выводится блок с названием диагностической схемы и фиксируется в верхней части опросного окна. Ниже данного блока пользователю выводится начальный наводящий вопрос, соответствующий данной диагностической схеме с вариантами ответов в виде кнопок «Да» и «Нет». При выборе пользователем, какого-либо варианта экспертная система обрабатывает результат и принимает решение в соответствии с диагностической схемой предоставить пользователю следующий наводящий вопрос или же вывести возможный диагноз. Блок вывода возможно диагноза визуально выделен в красную рамку с целью привлечения внимания для упрощения взаимодействия с пользователем (Рисунок 6).

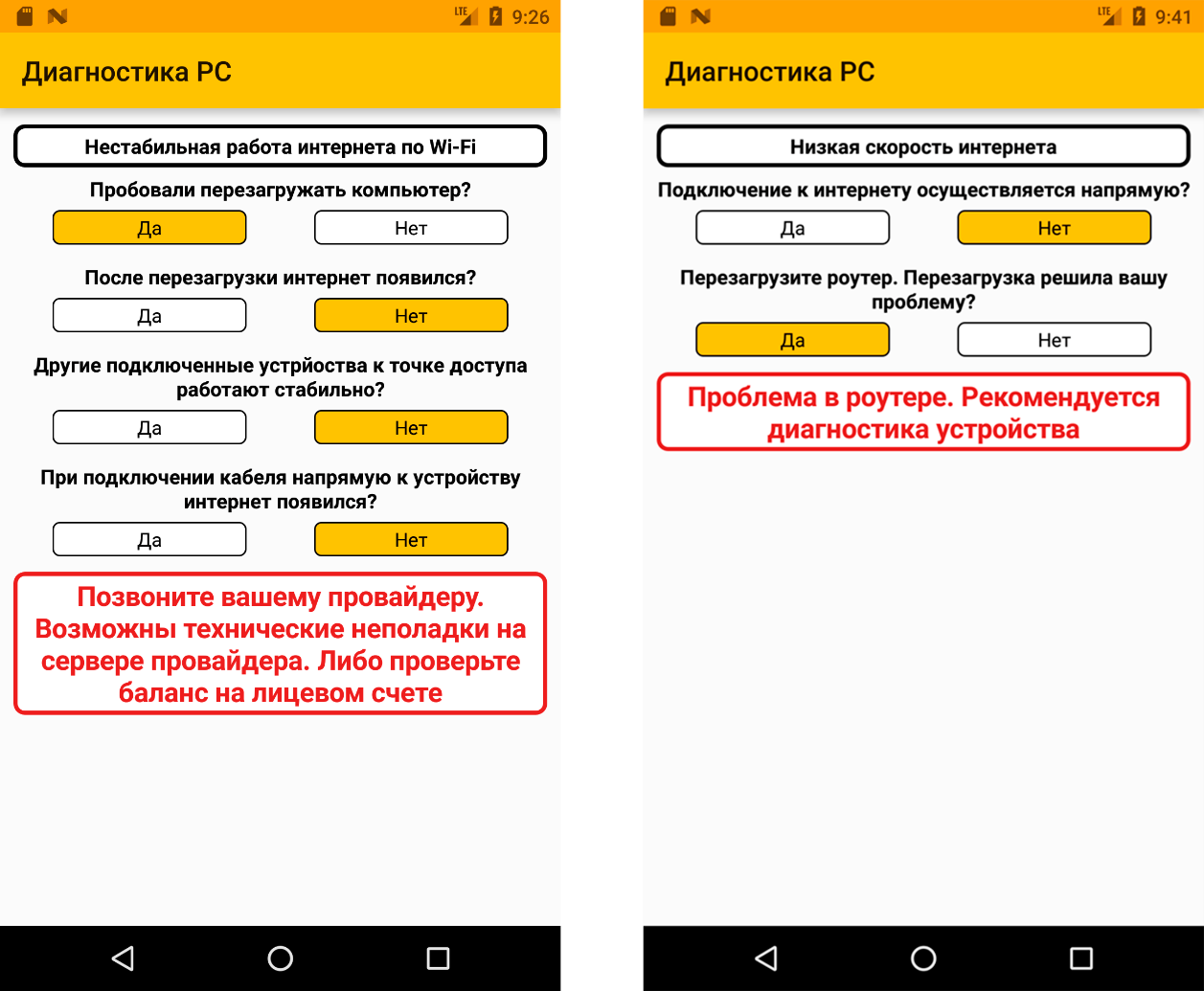


Рисунок 6 – Представления опросного окна

Ниже приведен фрагмент кода класса Main.java, отвечающий за работу модуля диагностики:

public class Main extends AppCompatActivity implements View.OnClickListener {

private static final String DATABASE\_NAME = "diagnostic2.db";

private static final int DB\_VERSION = 1;

private DatabaseHelper mDatabaseHelper;

private SQLiteDatabase mSQLiteDatabase;

int startQuestion, schemaID;

String description;

LinearLayout mainLayout, generalSymptoms;

ScrollView scrollContent;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

mDatabaseHelper = new DatabaseHelper(this);

try {

mDatabaseHelper.createDataBase();

} catch (IOException ioe) {

throw new Error("Невозможно создать базу");

}

mainLayout = (LinearLayout)findViewById(R.id.mainLayout);

scrollContent = new ScrollView(this);

scrollContent.setLayoutParams(new ViewGroup.LayoutParams(

ViewGroup.LayoutParams.MATCH\_PARENT,

ViewGroup.LayoutParams.WRAP\_CONTENT

));

scrollContent.setVerticalScrollBarEnabled(true);

generalSymptoms = new LinearLayout(this);

generalSymptoms.setLayoutParams(new ViewGroup.LayoutParams(

ViewGroup.LayoutParams.MATCH\_PARENT,

ViewGroup.LayoutParams.WRAP\_CONTENT

));

generalSymptoms.setOrientation(LinearLayout.VERTICAL);

generalSymptoms.setPadding(16, 16, 16, 16);

getGeneralSymptomsList();

scrollContent.addView(generalSymptoms);

mainLayout.addView(scrollContent);

setContentView(mainLayout);

}

@Override

public boolean onCreateOptionsMenu(Menu menu){

getMenuInflater().inflate(R.menu.about\_menu, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item){

int id = item.getItemId();

switch (id) {

case R.id.about\_therapist:

createAboutBox();

break;

}

return super.onOptionsItemSelected(item);

}

private void createAboutBox() {

Dialog aboutBox = new Dialog(this);

aboutBox.setCanceledOnTouchOutside(true);

aboutBox.requestWindowFeature(Window.FEATURE\_NO\_TITLE);

aboutBox.getWindow().setBackgroundDrawableResource(R.drawable.about\_box);

aboutBox.setContentView(R.layout.about);

aboutBox.show();

}

//Устанавливаем выбранный симптом и запускаем тестирование

public void onClick(View v){

Intent intent = new Intent(this, QuestionBox.class);

if(v instanceof TextView) {

String qText = ((TextView)v).getText().toString();

intent.putExtra("qDescription", qText);

}

intent.putExtra("qID", v.getId()-50000);

startActivity(intent);

}

//Возвращаем список основных симптомов из БД с первого по последний

private void getGeneralSymptomsList(){

mDatabaseHelper = new DatabaseHelper(this, DATABASE\_NAME, null, DB\_VERSION);

mSQLiteDatabase = mDatabaseHelper.getReadableDatabase();

Cursor cursor = mSQLiteDatabase.query(DatabaseHelper.DB\_MAIN\_TABLE, new String[]{

DatabaseHelper.SCHEMA\_ID,

DatabaseHelper.SCHEMA\_DESCRIPTION,

DatabaseHelper.START\_QUESTION},

null, null,

null, null, null, null);

cursor.moveToFirst();

do {

this.schemaID = cursor.getInt(cursor.getColumnIndex(DatabaseHelper.SCHEMA\_ID));

this.description = cursor.getString(cursor.getColumnIndex(DatabaseHelper.SCHEMA\_DESCRIPTION));

this.startQuestion = cursor.getInt(cursor.getColumnIndex(DatabaseHelper.START\_QUESTION));

createGeneralQuestion(this.description, this.startQuestion, this.generalSymptoms);

cursor.moveToNext();

} while (!cursor.isAfterLast());

}

//Создаем кликабельный блок с основным симптомом и добавляем его в конец лэйаута

private void createGeneralQuestion(String description, int id, LinearLayout gSHome){

TextView questionText = new TextView(this);

LinearLayout.LayoutParams qtParams = new LinearLayout.LayoutParams(

ViewGroup.LayoutParams.MATCH\_PARENT,

ViewGroup.LayoutParams.WRAP\_CONTENT

);

qtParams.setMargins(0, 5, 0, 5);

questionText.setLayoutParams(qtParams);

questionText.setId(id + 50000);

questionText.setText(description);

questionText.setGravity(Gravity.LEFT);

questionText.setPadding(15, 15, 15, 15);

questionText.setTextSize(15);

questionText.setTextColor(Color.parseColor("#000000"));

questionText.setTypeface(Typeface.DEFAULT\_BOLD);

questionText.setClickable(true);

questionText.setOnClickListener(this);

questionText.setBackgroundResource(R.drawable.general\_questions\_button);

gSHome.addView(questionText);

}

}

1. **Структура интерфейса**

Макет графического интерфейса хранится в файле activity\_main.xml. Каждый пакет содержит файл style.xml, который хранит в себе цветовые

настройки для отображения графических элементов. Ниже приведен фрагмент кода, отвечающий за структуру формы отображения основного раздела меню (Рисунок 7):

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity"

android:orientation="vertical"

android:background="#000000"

android:id="@+id/mainLayout">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="25dp"

android:background="#000"

android:baselineAligned="false"

android:gravity="center"

android:orientation="vertical">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_margin="1dp"

android:text="ОСНОВНЫЕ КАТЕГОРИИ"

android:textAllCaps="false"

android:textColor="@color/colorPrimary"

android:textSize="16dp"

android:textStyle="normal" />

</LinearLayout>

<LinearLayout

android:layout\_marginTop="3dp"

android:background="@color/colorPrimary"

android:orientation="horizontal"

android:layout\_width="match\_parent"

android:layout\_height="0dp"

android:layout\_weight="2" >

<LinearLayout

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_marginTop="65dp"

android:layout\_weight="1"

android:gravity="center\_horizontal"

android:orientation="vertical">

<ImageButton

android:id="@+id/imgbtn1"

android:layout\_width="100dp"

android:layout\_height="100dp"

android:background="@drawable/button\_stuts"

android:contentDescription="TODO"

android:src="@mipmap/dg3" />

<TextView

android:text="Диагностика"

android:textAlignment="center"

android:textColor="#000"

android:layout\_width="wrap\_content"

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

<ImageButton

android:id="@+id/logOut"

android:layout\_marginTop="30dp"

android:src="@mipmap/baseline\_exit\_to\_app\_black\_48"

android:layout\_width="100dp"

android:layout\_height="100dp"

android:background="@drawable/button\_stuts"

android:contentDescription="TODO" />

<TextView

android:text="Выход"

android:textColor="#000"

android:textAlignment="center"

android:layout\_width="wrap\_content"

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

</LinearLayout>

<LinearLayout

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_marginTop="65dp"

android:layout\_weight="1"

android:gravity="center\_horizontal"

android:orientation="vertical">

<ImageButton

android:id="@+id/imgbtn2"

android:layout\_width="100dp"

android:layout\_height="100dp"

android:background="@drawable/button\_stuts"

android:contentDescription="TODO"

android:src="@mipmap/baseline\_email\_black\_48"/>

<TextView

android:text="Обратная связь"

android:textColor="#000"

android:textAlignment="center"

android:layout\_width="wrap\_content"

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

<ImageButton

android:id="@+id/imgbtn4"

android:layout\_width="100dp"

android:layout\_height="100dp"

android:background="@drawable/button\_stuts"

android:contentDescription="TODO"

android:layout\_marginTop="30dp"

android:src="@mipmap/ic\_launcher\_foreground" />

<TextView

android:text="Настройки"

android:textColor="#000"

android:textAlignment="center"

android:layout\_width="wrap\_content"

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

</LinearLayout>

</LinearLayout>

</LinearLayout>

</RelativeLayout>

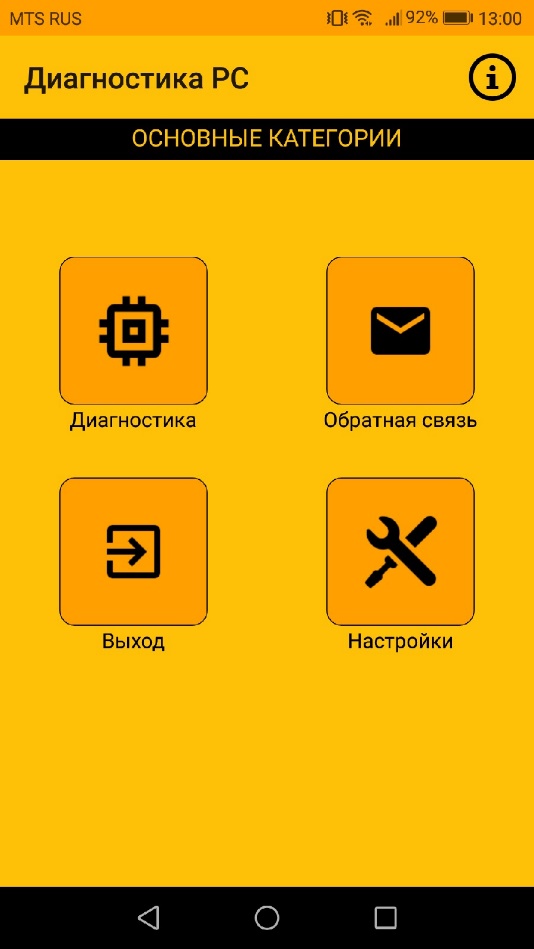


Рисунок 7 – Главное меню программы «Диагностика»

# **Заключение**

В ходе разработки курсового проекта, была спроектирована и создана экспертная система в виде мобильного приложения для диагностики неисправностей в работе вычислительной техники.

Система реализует функции поиска неисправностей вычислительной техники производителя по оборудованию, базам данных типовых проблем и методам их устранения.

В дальнейшем планируется реализовать автоматическую синхронизацию технической документации с базой знаний, усовершенствовать механизм вывода, организовать работу в офлайн режиме.

Второе направление развития разработанного решения – реализация дистанционного мониторинга и автоопроса о состоянии оборудования для прогнозирования и выявлении будущих неисправностей в вычислительные техники.

Большое внимание было уделено методам выбора, разработки, оценки качества и анализа эффективности программного обеспечения для решения задач в различных предметных областях. Получили представление о проблемах и направлениях развития технологии разработки программного обеспечения, об основных методологиях и средствах автоматизации по фазам жизненного цикла программного обеспечения.

**Список использованных источников**

1. Экспертная система. URL: https://clck.ru/FSXiS (дата обращения 18.11.2018).
2. NetAdapter Repair. URL: https://clck.ru/Gi6L7 (дата обращения 12.05.2019).
3. Малая экспертная система. URL: https://clck.ru/Gi6PA (дата обращения 12.05.2019).
4. Иванов И.Д. Информатика. База данных. – Москва, 2006. – 47 с.
5. Якубов С.Х. Алгоритмические методы построения баз знаний в интеллектуальных системах // Современные материалы, техника и технологии. – 2016. – № 1 (4). – С. 238-243.
6. Android Developers. URL: http://developer.android.com/ (дата обращения 15.05.2019).
7. Основы работы с SQLite в Андроид. URL: http://vlad8.com/tech/android-sqlite-database/ (дата обращения 18.05.2019).
8. Болотова Л.С. Системы искусственного интеллекта: модели и технологии, основанные на знаниях. – Москва, 2012. – 664 с.
9. Гаврилова Т.А. Базы знаний интеллектуальных систем. – Питер, 2000.
10. Elizabeth M 2016 Maintenance strategies [Online] Available: https://clck.ru/FPMu92019 [Accessed 11 may 2019].
11. Terry Q 2016 Visual modeling with Rational Rose [Online]

Available: https://clck.ru/Gi93c [Accessed 13 may 2019].

1. Arbour G 2019 Maintenance strategies [Online] Available: https://www.fiixsoftware.com/blog/evaluating-maintenance-strategies-select-model-asset-management/ [Accessed 12 may 2019].
2. Enge E 2018 Mobile vs desktop usage in 2018: mobile takes the lead [Online] Available: https://www.stonetemple.com/mobile-vs-desktop-usage-study/ [Accessed 10 may 2019].