СОДЕРЖАНИЕ

[Введение 4](#_Toc3233859)

[1 Экономическая характеристика объекта 5](#_Toc3233860)

[2 Постановка задачи 6](#_Toc3233861)

[2.1 Экономическое содержание задачи 6](#_Toc3233862)

[2.2 Входная информация 6](#_Toc3233863)

[3 Технические и инструментальные средства для решения задачи 7](#_Toc3233864)

[3.1 Обоснование выбора языка программирования 7](#_Toc3233865)

[3.2 Технические требования к компьютеру для решения задачи 9](#_Toc3233866)

[4 Информационное обеспечение задачи 10](#_Toc3233867)

[4.1 Классификаторы, используемые для решения задачи 10](#_Toc3233868)

[5 Программное обеспечение задачи 13](#_Toc3233869)

[5.1 Описание методов 13](#_Toc3233870)

[5.2 Алгоритм решения задачи 15](#_Toc3233871)

[5.3 Описание блок – схемы 15](#_Toc3233872)

[Заключение 19](#_Toc3233873)

[Список использованных источников 20](#_Toc3233874)

[Приложение А 21](#_Toc3233875)

[Приложение Б 22](#_Toc3233876)

# Введение

Смартфоны давно превзошли по количеству традиционные мобильные телефоны. Рынок современных смартфонов делится на 3 основных операционных системы: Apple iOS, Android и Microsoft Windows Phone. Каждый из производителей определяет собой платформу для разработки мобильных приложений. Рынок мобильных приложений является одним из самых быстрорастущих.

В настоящее же время смартфоны прочно укоренились в нашей повседневной жизни благодаря широкому спектру функций. С помощью смартфонов мы можем не только позвонить, но и выйти в интернет, зайти в социальную сеть, просмотреть или отредактировать какие-либо документы.

Большое предприятие предполагает больше количество различной документации. Большая часть из них хранится в цифровом формате в базах данных. Эти документы можно так же просматривать и редактировать, проследить на каком этапе они находятся, кто сейчас ими занимается при помощи смартфона. Это особенно полезно для инжиниринговой компании «Атоммашэкспорт».

Актуальность темы обусловлена тем, что она позволяет управлять документами дистанционно с помощью смартфона.

Цель курсового проекта заключается в создании Программного «Документооборот» на базе ОС Android.

В курсовом проекте решаются задачи:

* формирование базы данных;
* фильтрация документов по типу;
* подписание, согласование документов с помощью смартфона.

# 1 Экономическая характеристика объекта

В отеле 257 [**номеров**](https://spborbita.ru/rooms/) различных категорий: [**экономы**](https://spborbita.ru/rooms/economy/), [**стандарты**](https://spborbita.ru/rooms/standard/), [**двухкомнатные стандарты**](https://spborbita.ru/rooms/double/), [**3-местные экономы**](https://spborbita.ru/rooms/three-room/), [**4-х местные экономы**](https://spborbita.ru/economy-four-room/), [**комфорты**](https://spborbita.ru/rooms/comfort/), [**полулюксы**](https://spborbita.ru/rooms/suite/) и [**люксы**](https://spborbita.ru/rooms/suite-one-room/). Каждый сможет подобрать тип размещения согласно своим предпочтениям: однокомнатный или двухкомнатный номер, с балконом или без, с видом на улицу или во двор, с общей или раздельными кроватями, с дополнительным местом или без. Все номера независимо от категории оснащены согласно современным стандартам сервиса: в каждом имеется санузел с ванной или душем, телевизор с плоским экраном, телефон, холодильник и банные принадлежности.

На шестом этаже гостиницы работает [**кафе**](https://spborbita.ru/cafe/) «Арбуз», где с 7:00 до 11:00 часов проходит [**завтрак**](https://spborbita.ru/cafe/breakfast/) «шведский стол», с 13:00 до 17:00 гостям предлагают [**комплексный обед**](https://spborbita.ru/cafe/package/), а с 17:00 до 20:30 — [**ужин**](https://spborbita.ru/cafe/package/).

Под бронированием понимается заказ номера или места в гостинице, осуществляемый предварительно, то есть за несколько часов, дней или даже недель до даты въезда. Данную услугу предоставляют практически все отели – именно с нее, как правило, начинается обслуживание клиента. Приемом заявок занимается специально созданный отдел бронирования или служба приема гостей.

Существует два основных вида бронирования:

* гарантированное. В этом случае отель получает с клиента предоплату и обязуется предоставить ему забронированный номер даже в случае более позднего приезда, но не дольше расчетного времени (обычно – 12.00) дня, следующего за планируемой датой приезда. Фактически, гостиница закрепляет на сутки номер за предполагаемым гостем;
* негарантированное. Данный вид бронирования встречается реже. Он не предполагает предварительной оплаты, но при этом определяется время аннуляции брони (чаще всего – 18.00 планируемой даты заселения).

Бронирование номеров является системой, крайне выгодной и клиенту, и самому отелю. Первый получает возможность гарантированного заезда в удобное ему время без опасности обнаружить отсутствие свободных номеров, а гостиница может быть уверенной в получении очередного гостя с высокой степенью вероятности.

SS

Инжиниринговая компания «Атоммашэкспорт» выполняет комплексные конструкторские разработки и изготовление оборудования для атомных, тепловых станций и нефтегазохимии. По требованию заказчика возможна поставка оборудования с обеспечением работ по его монтажу, наладке и вводу в эксплуатацию.

Конструкторское подразделение компании состоит из специалистов, имеющих многолетний опыт в проектировании оборудования для АЭС, а также для нефтегазового комплекса [1].

# 2 Постановка задачи

## 2.1 Экономическое содержание задачи

Цель курсового проекта – создание приложения «Резервирования в отеле» на базе Microsoft для ООО «Отель Элеон»

Разработанный модуль позволяет производить различные действия добавления: клиента(удаление, изменение),

Добавления комнат(удаление, изменение),

Добавление резервирования(удаление, изменение).

Программа создана на языке C#, графический интерфейс приложения дружественный и понятный, сама программа проста в использовании и не требует специального образования или прохождение курсов.

## 2.2 Входная информация

Входная информация представляет собой информацию различного вида, хранящиеся в базе данных.

2.3 Выходная информация

Выходной информацией является уже обновленная информация над которым были произведены какие-либо из доступных действий.

# 3 Технические и инструментальные средства для решения задачи

## 3.1 Обоснование выбора языка программирования

C# — современный объектно-ориентированный и типобезопасный язык программирования. C# позволяет разработчикам создавать разные типы безопасных и надежных приложений, выполняющихся в .NET. C# относится к широко известному семейству языков C, и покажется хорошо знакомым любому, кто работал с C, C++, Java или JavaScript. Здесь представлен обзор основных компонентов языка C# 8 и более ранних версий. Если вы хотите изучить язык с помощью интерактивных примеров, рекомендуем поработать с [вводными руководствами по C#](https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/tutorials/).

C# — объектно-ориентированный, **ориентированный на компоненты** язык программирования. C# предоставляет языковые конструкции для непосредственной поддержки такой концепции работы. Благодаря этому C# подходит для создания и применения программных компонентов. С момента создания язык C# обогатился функциями для поддержки новых рабочих нагрузок и современными рекомендациями по разработке ПО. В основном C# — **объектно-ориентированный** язык. Вы определяете типы и их поведение.

Вот лишь несколько функций языка C#, которые позволяют создавать надежные и устойчивые приложения. [**Сборка мусора**](https://docs.microsoft.com/ru-ru/dotnet/standard/garbage-collection/) автоматически освобождает память, занятую недостижимыми неиспользуемыми объектами. [**Типы, допускающие значение null,**](https://docs.microsoft.com/ru-ru/dotnet/csharp/nullable-references) обеспечивают защиту от переменных, которые не ссылаются на выделенные объекты. [**Обработка исключений**](https://docs.microsoft.com/ru-ru/dotnet/csharp/fundamentals/exceptions/) предоставляет структурированный и расширяемый подход к обнаружению ошибок и восстановлению после них. [**Лямбда-выражения**](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/lambda-expressions) поддерживают приемы функционального программирования. [**Синтаксис LINQ**](https://docs.microsoft.com/ru-ru/dotnet/csharp/linq/) создает общий шаблон для работы с данными из любого источника. Поддержка языков для [**асинхронных операций**](https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/concepts/async/) предоставляет синтаксис для создания распределенных систем. В C# имеется [**Единая система типов**](https://docs.microsoft.com/ru-ru/dotnet/csharp/fundamentals/types/). Все типы C#, включая типы-примитивы, такие как int и double, наследуют от одного корневого типа object. Все типы используют общий набор операций, а значения любого типа можно хранить, передавать и обрабатывать схожим образом. Более того, C# поддерживает как определяемые пользователями [ссылочные типы](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/reference-types), так и [типы значений](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/value-types). C# позволяет динамически выделять объекты и хранить упрощенные структуры в стеке. C# поддерживает универсальные методы и типы, обеспечивающие повышенную безопасность типов и производительность. C# предоставляет итераторы, которые позволяют разработчикам классов коллекций определять пользовательские варианты поведения для клиентского кода.

C# подчеркивает **Управление версиями** , чтобы обеспечить совместимость программ и библиотек с течением времени. Вопросы управления версиями существенно повлияли на такие аспекты разработки C#, как раздельные модификаторы virtual и override, правила разрешения перегрузки методов и поддержка явного объявления членов интерфейса.

Главные возможности среды разработки Visual Studio:

* включены все «интеллектуальные» возможности по редактированию кода;
* есть возможность визуального просмотра будущего приложения;
* сборка проекта работает быстрее, чем в альтернативных средах разработки;
* удобный конструктор интерфейсов;
* встроенный SDK Manager;
* удобное и интуитивно понятное логирование проекта;
* указанные цвета и рисунки, использованные в layout'e отображаются на границе в виде небольших превью, которые легко помогают понять какой конкретно ресурс используется;
* среда разработки является технологиями компании Google;
* при выборе ресурса, его содержимое отображается во всплывающих JavaDoc;
* возможность создания одним кликом layout, сразу привязанным к activity;
* управление локализацией и переводами;
* предпросмотр приложений на разных разрешениях, языках, и версиях API во встроенном эмуляторе;
* мониторинг используемой памяти [3],[4].
* [Поиск в Visual Studio](https://docs.microsoft.com/ru-ru/visualstudio/ide/visual-studio-search?view=vs-2022)
* [Live Share](https://docs.microsoft.com/ru-ru/visualstudio/liveshare/)
* [Иерархия вызовов](https://docs.microsoft.com/ru-ru/visualstudio/ide/reference/call-hierarchy?view=vs-2022)
* [CodeLens](https://docs.microsoft.com/ru-ru/visualstudio/ide/find-code-changes-and-other-history-with-codelens?view=vs-2022)
* [Перейти к определению](https://docs.microsoft.com/ru-ru/visualstudio/ide/go-to-and-peek-definition?view=vs-2022)
* [Показать определение](https://docs.microsoft.com/ru-ru/visualstudio/ide/how-to-view-and-edit-code-by-using-peek-definition-alt-plus-f12?view=vs-2022)

## 3.2 Технические требования к компьютеру для решения задачи

Рекомендуемые системные требования к компьютеру на которой будет установлена IDE Visual Studio следующие:

* Операционная система – Microsoft® Windows® 7/8/10/11 (32- или 64-bit);
* оперативная память – 3 ГБ минимум, рекомендуется 8 ГБ;
* пространство на жестком диске – рекомендуется 5 ГБ;
* монитор – 1280 х 800 минимальное разрешение экрана.

# 4 Информационное обеспечение задачи

## 4.1 Классификаторы, используемые для решения задачи

Классификатор представляет собой набор таблиц (справочников).

Объект – некоторая сущность в цифровом пространстве, обладающая определённым состоянием и поведением в данной предметной области.

Атрибут – элемент, описывающий любую характеристику объекта, важную для данной предметной области. Объектом в данной базе данных будет: информация о работнике приведенная в таблице 4.1, документе показанных на таблице 4.2 и местонахождении документа на таблице 4.3.

Таблица 4.1 – Атрибуты Клиент

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Клиент | Идентификационный код  Фамилия  Имя  Отчество  Телефон  Паспорт  Дата рождения  Пол |

Таблица 4.2 – Атрибуты Комнаты

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Комнаты | Идентификационный код  Номер  Емкость  Резервирование  Тип комфорта  Этаж  Цена |

Таблица 4.3 – Атрибуты Резервирование

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Резервирование | Идентификационный код  Дата вселения  Дата выселения  Идентификационный код Клиента  Идентификационный код Комнаты  Статус резервирования  Тип оплаты |

Для выполнения проекта была выбрана СУБД SQL Server. Схема БД представлена на рисунке 4.1.

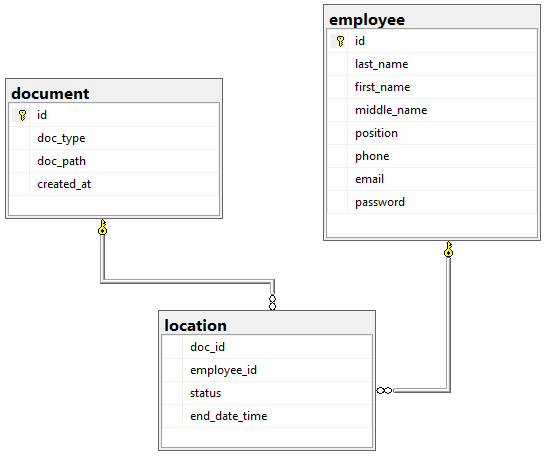


Рисунок 4.1 – Связи таблиц в БД

Структура таблиц документов, местонахождения и работников базы данных представлена на рисунках 4.2 – 4.4 соответственно.

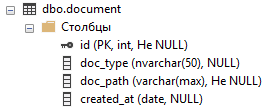


Рисунок 4.2 – Таблица «Документ»

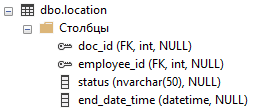


Рисунок 4.3 – Таблица «Местонахождение»

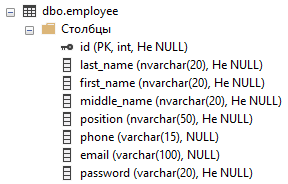


Рисунок 4.4 – Таблица «Работник»

# 5 Программное обеспечение задачи

## 5.1 Описание методов

Для примера взята часть кода для входа в приложение под своей учетной записью. При нажатии на кнопку вызывается активити, для дальнейшей работы с приложением.

package ru.atomexp.android;  
  
import android.content.Intent;  
import android.content.SharedPreferences;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.ImageButton;  
import android.widget.Toast;  
import com.google.gson.Gson;  
import com.google.gson.GsonBuilder;  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;  
  
public class LoginActivity extends AppCompatActivity {  
  
 private Button btnSignIn;  
 private ImageButton btnQrCode;  
 private EditText etLogin;  
 private EditText etPass;  
 private User user;  
 private Gson GSON;  
 public static final String *APP\_PREFERENCES* = "authorizedUser";  
 public static final String *SAVED\_USER* = "savedUser";  
 public static SharedPreferences *spUser*;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_login*);  
  
 *spUser* = getSharedPreferences(*APP\_PREFERENCES*, *MODE\_PRIVATE*);  
 GSON = new GsonBuilder().create();  
  
 btnSignIn = findViewById(R.id.*signInButton*);  
 btnQrCode = findViewById(R.id.*qrCodeButton*);  
 etLogin = findViewById(R.id.*loginText*);  
 etPass = findViewById(R.id.*passwordText*);  
 }  
  
 public void signIn(View view) {  
 Toast.*makeText*(this, "Нажатие", Toast.*LENGTH\_SHORT*).show();  
 try {  
 try {  
 Class.*forName*("net.sourceforge.jtds.jdbc.Driver");  
 Toast.*makeText*(this, "Зарегистрирован", Toast.*LENGTH\_LONG*).show();  
 } catch(Exception ex){  
 Toast.*makeText*(this, "Не зарегистрирован" + ex.getMessage(), Toast.*LENGTH\_LONG*).show(); }  
 final String DB\_URL = "jdbc:jtds:sqlserver://192.168.31.80:1433/documents";  
 final String DB\_USER = "p1onerw0w";  
 final String DB\_PASSWORD = "4321193";  
 final Gson GSON = new GsonBuilder().create();  
 String login = etLogin.getText().toString();  
 String password = etPass.getText().toString();  
 String query =  
 "select \* from users " +  
 "where (phone = '" + login + "' or email = '" + login + "') " +  
 "and [password] = '" + password + "'";  
 try {  
 Connection connection = DriverManager.*getConnection*(DB\_URL, DB\_USER, DB\_PASSWORD);  
 Toast.*makeText*(this, "Подключение создано", Toast.*LENGTH\_LONG*).show();  
 } catch (Exception ex) {  
 Toast.*makeText*(this, "Не подключен" + ex.getLocalizedMessage(), Toast.*LENGTH\_LONG*).show();  
 }  
 if(!*spUser*.contains(*SAVED\_USER*)) {  
 try {  
 user = User.*userBuild*(SqlServerService.*getResultSet*(query));  
  
 String gsonUser = GSON.toJson(user);  
  
 SharedPreferences.Editor editor = *spUser*.edit();  
 editor.putString(*SAVED\_USER*, gsonUser);  
 editor.apply();  
  
 Intent intent = new Intent(this, MainActivity.class);  
 intent.putExtra("user", gsonUser);  
 Toast.*makeText*(this, "Авторизация", Toast.*LENGTH\_SHORT*).show();  
 startActivity(intent);  
  
 } catch(SQLException sqlex) {  
 Toast.*makeText*(this, "Ошибка SQL", Toast.*LENGTH\_LONG*).show(); } catch(Exception ex) {  
 Toast.*makeText*(this, "NON-SQL ошибка", Toast.*LENGTH\_LONG*).show();  
 }  
  
 } else {  
  
 }  
 } catch(Exception ex) {   
 Log.*e*(this.getClass().getSimpleName(), "Получено исключение", ex);  
 }  
 }  
}

## 5.2 Алгоритм решения задачи

Для проведения каких-либо действий с программой сначала необходимо выбрать раздел: добавить клиента, сделать резервирован.

## 5.3 Описание блок – схемы

Последовательное описание блок – схемы приложения приведено ниже:

Шаг 1. Запуск программы.

Шаг 2. Вопрос: Верно ли указан логин и пароль? Если да, то переходим к следующему шагу, иначе остаемся на 2 шаге.

Шаг 3. Вход в приложение под Администратором учетной записью.

Шаг 4. Вопрос: Выбрано ли действие? Если да, то переходим к шагу 5, если нет, то переходим к 7 шагу.

Шаг 5. Выполнение

Шаг 6. Возвращаемся к 4 шагу.

Шаг 7. Закрытие приложения.

Блок-схема приведена в приложении А.

Листинг программы приведен в приложении Б.

5.4 Описание интерфейса программы

В Окне входа в приложение, представленном на рисунке 5.1, находятся два текстовых поля логин и пароль, кнопка Войти. Для дальнейшей работы нужно верно указать логин и пароль. Если логин и пароль верны, то откроется основное Окно, представленное на рисунке 5.2 с возможностью выбора действия над документами. При нажатии на кнопку происходит открытие нужного нам Окно которая показана на рисунке 5.3.



Рисунок 5.1 – Активити входа

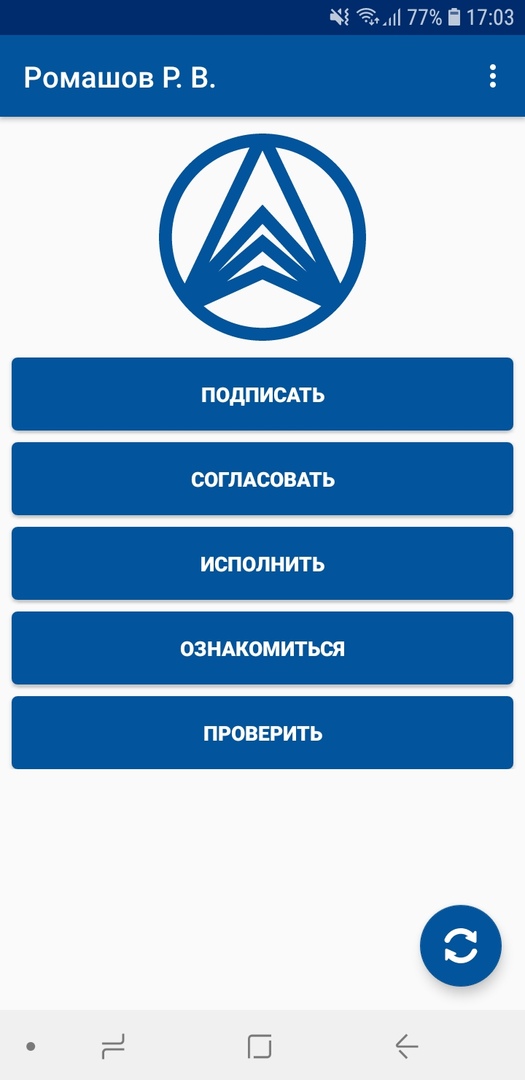


Рисунок 5.2 – Основное активити

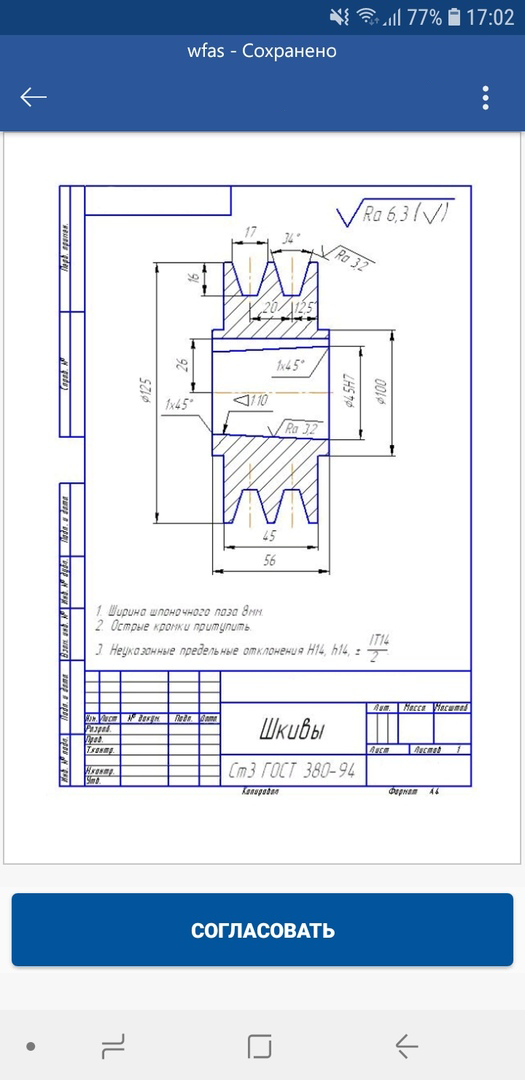


Рисунок 5.3 – Активити согласования

Для успешной работы с программой необходимо:

1. ОС Система
2. Подключение к базе данных.
3. Учетная запись.

# Заключение

Разработанное мобильное приложение резервирования для многократного использования представляет собой простое приложение с лаконичным дизайном в стиле Material Design.

Программа разработана на языке С# в интегрированной среде разработки Visual Studio с использованием СУБД SQL Server. Приложение интуитивно понятно даже неопытному пользователю и не затрудняет его работу программой с помощью OC Система

В дальнейшем в приложение будет обновляться и будут добавляться новые функции необходимые для работы с Резервирования.

В процессе написания кода для данной программы были приобретены практические навыки в работе в СУБД SQL Server и языке программирования С.

Цель курсового проекта достигнута. Задачи выполнены.

# Список использованных источников

Интернет ресурсы:

* 1. АО “Атоммашэкспорт”. О компании [Электронный ресурс] – Режим доступа: http://www.atomexp.ru/page/about
  2. Microsoft. Язык программирования C# его особенности [Электронный ресурс] – Режим доступа:https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide
  3. Microsoft. Visual Studio [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/
  4. Habr. Российский блог информационных технологий [Электронный ресурс] – Режим доступа: https://habr.com/ru/

# Приложение А

(обязательное)

Блок – схема программы

Запуск программы

Введены верные логин и пароль?

Нет

Да

Вход в приложение

Выбрано действие?

Нет

Выполнение

Да

Сохранение данных

Рисунок А.1–Блок схема программы

# Приложение Б

(обязательное)

Листинг программы

create database TermPaper

go

use TermPaper

go

create table Clients

(

ClientsID int primary key identity(1,1) not null,

[Name] varchar(255) not null,

LastName varchar(255) not null,

DateOfBirth date not null,

Gender varchar(255) not null,

PhoneNumber varchar(255) not null,

Passport varchar(255) not null,

);

create table Rooms

(

RoomsID int primary key identity(1,1) not null,

Number varchar(255) not null,

[Floor] int not null,

[Type] varchar(255) not null,

Capfcity int not null,

[Status] varchar(255) not null,

Price varchar(255) not null,

);

create table Reservations

(

ReservationsID int primary key iDenTITy(1,1) not null,

CheckInDate date not null,

CheckOutDate date not null,

RoomID int not null,

ReservationStatus varchar(255) not null,

typePayment varchar(255) not null,

ClientID int not null,

foreign key (ClientID) references Clients(ClientsID),

foreign key (RoomID) references Rooms(RoomsID)

);

import android.content.Intent;  
import android.content.SharedPreferences;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.ImageButton;  
import android.widget.Toast;  
  
import com.google.gson.Gson;  
import com.google.gson.GsonBuilder;  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;  
  
public class LoginActivity extends AppCompatActivity {  
  
 private Button btnSignIn;  
 private ImageButton btnQrCode;  
 private EditText etLogin;  
 private EditText etPass;  
 private User user;  
 private Gson GSON;  
 public static final String *APP\_PREFERENCES* = "authorizedUser";  
 public static final String *SAVED\_USER* = "savedUser";  
 public static SharedPreferences *spUser*;  
  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_login*);  
 *spUser* = getSharedPreferences(*APP\_PREFERENCES*, *MODE\_PRIVATE*);  
 GSON = new GsonBuilder().create();  
 btnSignIn = findViewById(R.id.*signInButton*);  
 btnQrCode = findViewById(R.id.*qrCodeButton*);  
 etLogin = findViewById(R.id.*loginText*);  
 etPass = findViewById(R.id.*passwordText*);  
 }  
  
 public void signIn(View view) {  
 Toast.*makeText*(this, "Нажатие", Toast.*LENGTH\_SHORT*).show();  
 try {  
 try {  
 Class.*forName*("net.sourceforge.jtds.jdbc.Driver");  
 Toast.*makeText*(this, "Зарегистрирован", Toast.*LENGTH\_LONG*).show();  
 } catch(Exception ex){  
 Toast.*makeText*(this, "Не зарегистрирован" + ex.getMessage(), Toast.*LENGTH\_LONG*).show(); }  
 final String DB\_URL = "jdbc:jtds:sqlserver://192.168.31.80:1433/documents";  
 final String DB\_USER = "p1onerw0w";  
 final String DB\_PASSWORD = "4321193";  
 final Gson GSON = new GsonBuilder().create();  
 String login = etLogin.getText().toString();  
 String password = etPass.getText().toString();  
 String query =  
 "select \* from users " +  
 "where (phone = '" + login + "' or email = '" + login + "') " +  
 "and [password] = '" + password + "'";  
 try {  
 Connection connection = DriverManager.*getConnection*(DB\_URL, DB\_USER, DB\_PASSWORD);  
 Toast.*makeText*(this, "Подключение создано", Toast.*LENGTH\_LONG*).show();  
 } catch (Exception ex) {  
 Toast.*makeText*(this, "Не подключен" + ex.getLocalizedMessage(), Toast.*LENGTH\_LONG*).show();  
 }  
 if(!*spUser*.contains(*SAVED\_USER*)) {  
 try {  
 user = User.*userBuild*(SqlServerService.*getResultSet*(query));  
 String gsonUser = GSON.toJson(user);  
  
 SharedPreferences.Editor editor = *spUser*.edit();  
 editor.putString(*SAVED\_USER*, gsonUser);  
 editor.apply();  
 Intent intent = new Intent(this, MainActivity.class);  
 intent.putExtra("user", gsonUser);  
 Toast.*makeText*(this, "Авторизация", Toast.*LENGTH\_SHORT*).show();  
 startActivity(intent);  
 } catch(SQLException sqlex) {  
 Toast.*makeText*(this, "Ошибка SQL", Toast.*LENGTH\_LONG*).show(); } catch(Exception ex) {  
 Toast.*makeText*(this, "Иная ошибка", Toast.*LENGTH\_LONG*).show();  
 }  
 }   
 } catch(Exception ex) {  
 Toast.*makeText*(this, "Снаружи " + ex.getMessage(), Toast.*LENGTH\_LONG*).show();  
 Log.*e*(this.getClass().getSimpleName(), "Получено исключение", ex);  
 }  
 }  
}

package ru.atomexp.android;  
  
import android.os.Bundle;  
import android.support.design.widget.NavigationView;  
import android.support.v4.view.GravityCompat;  
import android.support.v4.widget.DrawerLayout;  
import android.support.v7.app.ActionBarDrawerToggle;  
import android.support.v7.app.AppCompatActivity;  
import android.support.v7.widget.Toolbar;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.widget.TextView;  
import android.widget.Toast;  
import com.google.gson.Gson;  
import com.google.gson.GsonBuilder;  
import com.google.gson.JsonObject;  
import org.json.JSONObject;  
  
public class MainActivity extends AppCompatActivity  
 implements NavigationView.OnNavigationItemSelectedListener {  
 private TextView navHeaderTitle;  
 private TextView navHeaderSubTitle;  
 private User thisUser;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 Toolbar toolbar = findViewById(R.id.*toolbar*);  
 setSupportActionBar(toolbar);  
 DrawerLayout drawer = findViewById(R.id.*drawer\_layout*);  
 ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(  
 this, drawer, toolbar, R.string.*navigation\_drawer\_open*, R.string.*navigation\_drawer\_close*);  
 drawer.addDrawerListener(toggle);  
 toggle.syncState();  
 NavigationView navigationView = findViewById(R.id.*nav\_view*);  
 navigationView.setNavigationItemSelectedListener(this);  
 if(savedInstanceState == null) { navigationView.setCheckedItem(R.id.*m\_messages*);  
 getSupportActionBar().setTitle(R.string.*messages*);  
 }  
 navHeaderTitle = findViewById(R.id.*headerTitle*);  
 navHeaderSubTitle = findViewById(R.id.*headerSubTitle*);  
 Gson GSON = new GsonBuilder().create();  
 thisUser = GSON.fromJson((String)getIntent().getSerializableExtra("user"), User.class);  
 String headerTitle = thisUser.getLastName() + thisUser.getFirstName();  
 navHeaderTitle.setText(headerTitle);  
 String headerSubTitle = thisUser.getPosition();  
 navHeaderSubTitle.setText(headerSubTitle);  
 }  
  
 @Override  
 public void onBackPressed() {  
 DrawerLayout drawer = findViewById(R.id.*drawer\_layout*);  
 if (drawer.isDrawerOpen(GravityCompat.*START*)) {  
 drawer.closeDrawer(GravityCompat.*START*);  
 } else {  
 *super.onBackPressed();* }  
 }  
  
 @Override  
 public boolean onCreateOptionsMenu(Menu menu) { getMenuInflater().inflate(R.menu.*main*, menu);  
 return true;  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) { int id = item.getItemId(); if (id == R.id.*action\_settings*) {  
 Toast.*makeText*(MainActivity.this, "Settings", Toast.*LENGTH\_LONG*).show();  
 }  
 return super.onOptionsItemSelected(item);  
 }