Содержание

ВВЕДЕНИЕ..........................................................................................................................4

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ............................................................................5

2 ПОСТАНОВКА ЗАДАЧИ..............................................................................................10

3 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА......................................................... 11

3.1 Создание разметки для всех Activity..................................................................11

3.2 Подготовка сервера для запросов.......................................................................13

3.3 Создание базы данных.........................................................................................14

3.4 Создание «классов» регистрации и входа..........................................................15

3.5 Создание «класса» поиска поездки……………………………………………17

3.6 Языковая поддержка……………………………………………………………18

4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ.............................................................................20

4.1 Начало работы.......................................................................................................20

4.2 Регистрация........................................................................................................... 22

4.3 Вход в аккаунт…...................................................................................................23

4.4 Поиск попутчика………………………………………………………………...24

ЗАКЛЮЧЕНИЕ..................................................................................................................25

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ………………………………………..26

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ПРОГРАММЫ.................................................27

ПРИЛОЖЕНИЕ Б. БЛОК-СХЕМА КЛАССА makehttprequest

# ВВЕДЕНИЕ

На данный момент уровень развития вычислительной техники позволяет использовать её не только лишь в научной деятельности, где важны быстрые и точные машинные вычисления, но и в области развлечения. Компьютерные игры и всевозможные приложения со временем становятся всё более красочными и реалистичными, вместе с увеличением данного показателя растёт и количество тех, кто любит расслабиться с их помощью. Примеров таких приложений можно привести очень большое количество, однако остановимся на игровом приложении «Bomberman», более известное в русскоязычных кругах как «Бомбермэн».

Впервые на рынок данное игровое приложение вышло двадцать восьмого апреля тысяча девятьсот восемьдесят третьего года в Японии, а затем двадцатого ноября того же года и на европейский рынок.

Сценарий игры изначально отсутствовал, целью игры являлось либо уничтожение всех врагов, либо пройти уровень за определённое отведённое время. Игрок имел возможность ставить бомбы, которые взрываются через несколько секунд и способные разрушить некоторые блоки. За этими блоками могли скрываться различного рода бонусы: увеличение скорости, увеличение количества бомб либо увеличение скорости передвижения персонажа. Из данного функционала складывалось отличное и захватывающее игровое приложение.

Со временем данное игровое приложение обрело очень большое количество интерпретаций, которые пользуются успехом и по сегодняшний день.

Таким образом, данное игровое приложение до сих пор остаётся популярной и актуальной в широком кругу людей. Игровое приложение «Bomberman» может стать темой курсового проекта, целью которого будет создание новой интерпретации данного игрового приложения.

В курсовом проекте будет создан новый вариант данного игрового приложения в классическом стиле в 2D графике с простым и интуитивно понятным интерфейсом и достаточным функционалом, чтобы полностью передать суть данного игрового приложения и обеспечить приятное времяпрепровождение.

**1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

**1.1 Обзор аналогов**

Существует довольно много аналогов данного игрового приложения. В ходе их поиска и изучения наибольшее внимание уделяется программам для операционной системы Windows, онлайн-аналогам и приложениям под игровые консоли.

На рассмотрение представлено игровое приложение «Bomberman», которое было разработано под игровую консоль Nintendo Entertainment System (NES). Вид игрового окна приложения представлен на рисунке 1.1.

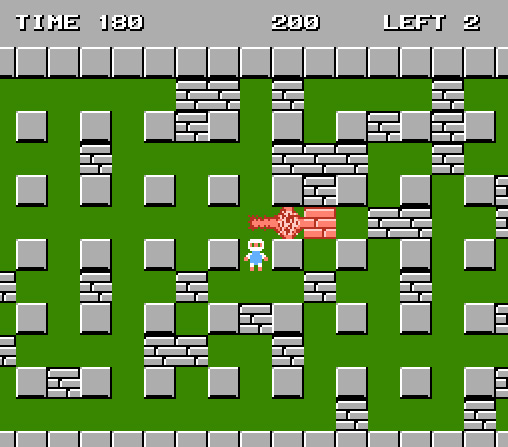


Рисунок 1.1 ­­– Основной экран программы «Bomberman» в NES

Данное приложение обладает примитивным, минималистичным интерфейсом, который не отвлекает от игрового процесса.

Для обеспечения игрового процесса в приложении присутствуют следующие бонусы:

* Огонь: Увеличивает мощность взрыва;
* Бомба: Дополнительная бомба;
* Бомба с фитилём: Возможность взрывать бомбы нажатием кнопки;
* Роликовые коньки: Увеличивает скорость;
* Бомбермэн и стенка: Способность проходить сквозь стены;
* Бомба и стенка: Способность проходить сквозь бомбы;
* Вопросительный знак: Неуязвимость на время.

Главная задача – за отведённое время уничтожить всех врагов на уровне и найти дверь, ведущую на следующий уровень.

Список врагов для обеспечения игрового процесса:

* Шарик: Глупый, движется медленно;
* Луковица: Движется быстро, может преследовать героя;
* Бочка: Глупый, движется быстро;
* Апельсин: Преследует героя, по скорости уступает медведю;
* Амёба: Очень медленный, может проходить сквозь стены;
* Призрак: Движется с нормальной скоростью, может проходить сквозь стены;
* Медведь: Умный, движется быстро, преследует героя;
* Копейка: Умный, движется быстро, может проходить сквозь стены.

Другим аналогом является игровое приложение «Atomic Bomberman». Данное приложение было разработано под платформу Windows для ПК. Вид игрового приложения представлен на рисунке 1.2.



Рисунок 1.2 ­­– Внешний вид окна приложения «Atomic Bomberman» в Windows

Для обеспечения игрового процесса в приложении присутствуют следующие бонусы:

* Дополнительные бомбы;
* Увеличение длины пламени, вплоть до бесконечности;
* Череп: Даёт болезни, например, минимального размера пламя, невозможность ставить бомбы и т.д.;
* Чёрный череп: Даёт до трёх болезней одновременно;
* Ускорение;
* Возможность двигать бомбы;
* Возможность поставить все имеющиеся бомбы в ряд;
* Возможность кидать бомбы;
* Возможность бросаться бомбами;
* Замедление;
* Превращение бомбы в «желе»;
* Установка триггера на бомбу;
* Случайный бонус.

Особенностью данной версии игрового приложения заключается в отсутствии класса врага, так как добавлена возможность многопользовательской игры и реализации искусственного интеллекта для оставшихся объектов бомбермэна, в основе которого лежит цель выжить самому, а не убить врага любой ценой, поэтому противники не столь агрессивны, как в предыдущих аналогах приложения.

Добавлена различного рода анимация, улучшающая восприятие игрового процесса.

Добавлена возможность выбора карты, из списка предложенных, вручную либо случайным образом.

Следующим аналогом является игровое приложение «Bomberman Online», разработанное под игровую консоль Sega Dreamcast. Вид окна игрового приложения представлен на рисунке 1.3.



Рисунок 1.2 ­­– Внешний вид окна приложения «Microsoft Minesweeper»

Для обеспечения игрового процесса в приложении присутствуют следующие бонусы:

* - Дополнительные бомбы;
* Увеличение длины пламени;
* Ускорение;
* Добавление и улучшение брони;
* И т.д.

В связи с введением многопользовательского режима игры были введены следующие стили игры:

* Survival Rule – «Электрические драконы» - это классический стиль, перенесённый из предыдущих аналогов приложения.
* Hyper Bomber Rule – «Красные фениксы» - это стиль, где враги и сам игрок может погибать неограниченное число раз, при этом теряя собранные предметы и призы, целью является сбор бонусов и улучшений;
* Submarine Rule – «Морские принцессы» - это стиль, где карта представляет собой арену, поделенную на 2 части, целью является уничтожение всех противников на арене;
* Ring Match Rule – «Штормовые гиганты» - это стиль, где участники не погибают, а вылетают с арены, целью является достижение максимального числа сброшенных противников;
* Panel Paint Rule – «Железные бульдозеры» - это стиль, где участники с помощью бомб переворачивают плиты на полу арены, целью является перевернуть на свою сторону как можно больше панелей.

Особенностью данного приложения является привлекательный графический интерфейс, хорошая анимация, многопользовательский режим и введение различных стилей игры.

В связи с большим количеством аналогов игрового приложения «Bomberman» были приведены лишь некоторые аналоги, наиболее выделяющиеся и известные пользователям.

**1.2 Постановка задачи**

В рамках данного курсового проекта должно быть разработано игровое приложение «Bomberman» с графическим представлением в 2D.

В процессе реализации программного средства должны быть разработаны алгоритмы корректного взаимодействия персонажа пользователя с окружающими его объектами.

В результате анализа аналогов данного игрового приложения принято решение выбрать минималитичный, простой интерфейс игрового приложения и создание многопользовательского режима на одном устройстве с использованием необходимого минимума дополнительных функций, с целью не загромождения игрового приложения лишним функционалом, обнаруженным в исследованных аналогах.

В программном средстве планируется реализовать следующие функции:

* Генерация поля бонусов;
* Решение возможных коллизий персонажа с элементами карты;
* Решение возможных коллизий персонажа с объектами бомб;
* Анимированние персонажа и карты;
* -Завершение игры после попадание в огонь от взрыва бомбы;

Для разработки программного средства будет использоваться язык программирования C++ и среда разработки Visual Studio 2015, а также использование сторонней библиотеки SFML для обеспечения графического интерфейса и удобства работы с ним.

**2 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА**

**2.1 Интерфейс и графические элементы программного средства**

Для создания и обработки элементов графического интерфейса используется графический редактор Adobe Photoshop CC 2015. Далее будет приведён список используемых графических ресурсов.

Для отображения информации о значении полей бонусов персонажа используется изображение, отображающее панель, где будет выводиться информация о персонаже. Данная панель представлена на рисунке 2.1.



Рисунок 2.1 – Изображение панели для отображения информации о персонаже

Для отображения текстуры персонажа и его движения используется набор спрайтов содержащихся в изображении, которое представлено на рисунке 2.2.

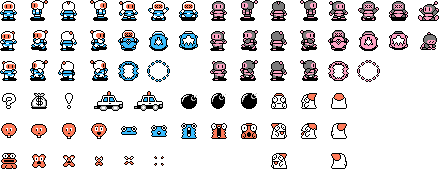
****

Рисунок 2.2 ­­– Изображение, содержащее текстуры персонажа и его движения

Для отображения текстур карты и бонусов используется набор спрайтов, содержащихся в изображении, которое представлено на рисунке 2.3.

****

Рисунок 2.3 ­­– Изображение, содержащее текстуры карты и бонусов

Для отображения текстур бомбы и её взрыва используется набор спрайтов, содержащихся в изображении, которое представлено на рисунке 2.4.

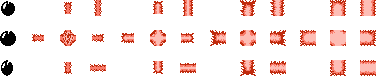
****

Рисунок 2.3 ­­– Изображение, содержащее текстуры карты и бонусов

Данные изображения были заимствованы из внешнего источника <http://randomhoohaas.flyingomelette.com/Sprites/spr-bomberman.php>. Данный источник содержит бесплатные изображения.

**2.2 Представление карты**

Для отображения полноценной карты был выбран способ хранения карты в виде массива символов. Вследствие последующего наложения символов карты на символы бонусов и взрывов было решено объявить три массива карты, непосредственно массив карты, массив бонусов и массив взрыва бомбы, которые имеют названия TileMap, TileBonusMap и TileBoomMap соответственно.

Обозначения для TileMap:

* ‘M’ – неразрушаемая часть карты;
* ‘S’ – разрушаемая стена;
* ‘ ’ – покрытие пола.

**2.3 Объявление классов**

Для отображения бомбы и возможности взаимодействия персонажа с бомбой объявим класс Bomb:

class Bomb

{

private:

float x, y, width, heigth;

float nativeTime, boomTime;

public:

int frame;

int fire;

sf::String File;

sf::Image image;

sf::Texture texture;

sf::Sprite sprite;

sf::Sound soundBoom;

bool visible, inPlayer, isBoom, enabled;

int fireState;

Bomb();

{

}

void Create(sf::String F, sf::Sound \*SoundBoom, float X, float Y, float Width, float Heigth);

void Reset();

void SetPosition(float X, float Y);

void SetTileDestroyStone(int positionY, int positionX);

void CheckBombOnFire(float x, float y)

void Update(float time)

float GetPositionX()

float GetPositionY()

float GetHeigth()

float GetWidth()

float GetNativeTime()

};

**3.3 Создание базы данных**

Для хранения пользователей, их профилей, логинов и паролей а так же для хранения поездок, добавленных пользователями необходима база данных, которая будет это выполнять.

Для создания базы данных использовался сервер, которые поддерживает базы данных MySQL. Каждые запросы, которые проходят через сервер напрямую обращаются к базе данных, которая даёт необходимую информацию для составления ответов в виде JSON описанные в пункте 3.2.



Рисунок 1 – «База данных поездок»

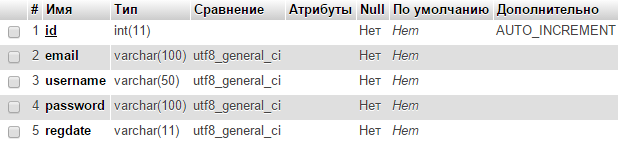


Рисунок 2 – «База данных пользователей»

**3.4 Создание «класса» регистрации и входа**

Для того, чтобы пользователь мог выполнять некоторые операции, например предложения своей поездки для поиска попутчиков, ему необходимо зарегистрироваться, либо выполнить вход непосредственно через Android приложение.

Процедура входа пользователя и его авторизация:

@Override  
**public void** onClick(View v) {  
 **switch**(v.getId()){  
 **case** R.id.***btnSignIn***:  
 **if** (!(**userEmail**.getText().toString().matches(**""**)) && !(**userPass**.getText().toString().matches(**""**))){  
 String email = **userEmail**.getText().toString();  
 String password = **userPass**.getText().toString();  
 **new** LoginUserClass(getActivity()).execute(email,password);  
 } **else** {  
 **textInfo**.setTextColor(Color.***RED***);  
 **textInfo**.setText(**"Заполнены не все поля!"**);  
 }  
 **break**;  
 **case** R.id.***signUpInLogin***:  
 **fragment** = **new** SignupScreen();  
 android.support.v4.app.FragmentManager fragmentManager = getFragmentManager();  
 fragmentManager.beginTransaction().replace(R.id.***content\_frame***,**fragment**).commit();  
 **break**;  
 }  
}



Блок-схема 1 - «Процедура входа пользователя»

В данном классе предусмотрена авторизация пользователя посредством заполнения пользователем полей для входа, в данном случае «логин» и «пароль», далее идет проверка на их заполнение и отправка запроса методом POST через класс LoginUserClass().

Процедура регистрации пользователя:

**public void** register(String userEmail, String userPassword, String userName) {  
 String urlSuffix = **"?email="** + userEmail + **"&password="** + userPassword + **"&username="** + userName; *// для GET запроса* **class** RegisterUser **extends** AsyncTask<String, Void, String> {  
 ProgressDialog **loading**;  
 RegisterUserClass **ruc** = **new** RegisterUserClass();  
  
 @Override  
 **protected void** onPreExecute() {  
 **super**.onPreExecute();  
 **loading** = ProgressDialog.*show*(getActivity(), **"Пожалуйста, подождите!"**, **null**, **true**, **true**);  
 }  
  
 @Override  
 **protected void** onPostExecute(String s) {  
 **super**.onPostExecute(s);  
 **loading**.dismiss();  
 String answerServer = **"Connection timeout"**;  
 **if** (s.matches(**"130"**)){  
 answerServer = **"Register successfull"**;  
 **fragment** = **new** SigninScreen();  
 android.support.v4.app.FragmentManager fragmentManager = getFragmentManager();  
 fragmentManager.beginTransaction().replace(R.id.***content\_frame***, **fragment**).commit();  
 } **else if** (s.matches(**"131"**)){  
 answerServer = **"This email is already in use"**;  
 } **else if** (s.matches(**"132"**)){  
 answerServer = **"Error register"**;  
 } **else if** (s.matches(**"133"**)){  
 answerServer = **"Not all fields was filled"**;  
 } **else if** (s.matches(**"134"**)){  
 answerServer = **"Oops! Try again!"**;  
 }  
 Toast toast = Toast.*makeText*(getActivity(), answerServer, Toast.***LENGTH\_LONG***);  
 toast.show();  
 }  
  
 @Override  
 **protected** String doInBackground(String... params) {  
 HashMap<String, String> data = **new** HashMap<String, String>();  
 data.put(**"email"**, params[0]);  
 data.put(**"password"**, params[1]);  
 data.put(**"username"**, params[2]);  
  
 String result = **ruc**.sedPostRequest(***REGISTER\_URL***,data);  
  
 **return** result;  
 }  
 }  
  
 RegisterUser ru = **new** RegisterUser();  
 ru.execute(userEmail,userPassword,userName);

}

В данном классе происходит регистрация пользователя так же посредством POST запроса на сервер.

**3.5 Создание «класса» поиска поездки**

Для того, чтобы пользователь мог выполнять некоторые операции, например предложения своей поездки для поиска попутчиков, ему необходимо зарегистрироваться, либо выполнить вход непосредственно через Android приложение.

**if** (**pointArrival**.length() != 0 && **pointDeparture**.length() != 0) {  
 **if** ((IsExist(**pointArrival**.getText().toString())) && (IsExist(**pointDeparture**.getText().toString()))) {  
 String inputPoint = String.*valueOf*(SearchIndex(**pointArrival**.getText().toString()));  
  
 String outputPoint = String.*valueOf*(SearchIndex(**pointDeparture**.getText().toString()));  
 Log.*d*(**"LOL"**, **"departure "** + inputPoint);  
 **listView**.setAdapter(**null**);  
 **new** JSONTask().execute(outputPoint, inputPoint);  
 } **else** {  
 Toast.*makeText*(getContext(),**"Ничего не найдено!"**,Toast.***LENGTH\_SHORT***).show();  
 **listView**.setAdapter(**null**);  
 }  
} **else** {  
 Toast.*makeText*(getContext(), **"Заполнены не все поля!"**, Toast.***LENGTH\_SHORT***).show();  
}

Данный класс описывает поиск поездки через POST запрос на сервер, который отправляет пользователь заполняя оба поля «Пункт отправления» и «Пункт пребытия».

Для того, чтобы показать все доступные поездки пользователю по заданным маршрутам использовался ListView с адаптером, созданным специально для данной разметки.

Класс кастомного адаптера:

public class CustomListAdapter extends ArrayAdapter<String>{

private final Activity context;

...

public CustomListAdapter(Activity context, String[] timeString, String[] priceString, String[] placesString, String[] dateString, String[] idInt, String[] freePlace, String[] totalPlace, String[] userName) {

super(context, R.layout.custom\_listview\_row, idInt);

this.context = context;

...

}

@Override

public View getView(int position, View convertView, ViewGroup parent) {

LayoutInflater inflater = context.getLayoutInflater();

View rowView = inflater.inflate(R.layout.custom\_listview\_row, null, true);

TextView textTime = (TextView) rowView.findViewById(R.id.ts\_time);

...

textTime.setText(timeString[position]);

...

this.notifyDataSetChanged();

return rowView;

}

}

**3.6 Языковая поддержка**

Исходя из того, что данным программным средством могут пользоваться люди из иноязычных стран, было решено ввести языковую поддержку для таких языков, как English и Russia. То есть, в зависимости от выбранного языка системы будет устанавливаться необходимый язык в самом программном средстве.

Для того чтобы внести такую поддержку, необходимо было использовать стандартные ресурсы в виде разметки strings.xml, которая хранит строки для заголовка программного средства, выводимых сообщений, различных TextView и так далее.

Разметка strings.xml для англоговорящих:

<**resources**>  
 <**string name="app\_name"**>Who is where?</**string**>  
 <**string name="search\_screen"**>Search trips</**string**>  
 <**string name="get\_screen"**>Get trips</**string**>  
 <**string name="signin\_screen"**>Sign In</**string**>  
 <**string name="signup\_screen"**>Sign Up</**string**>  
 <**string name="action\_settings"**>Settings</**string**>  
 <**string name="action\_search"**>Search</**string**>  
 <**string-array name="screen\_array"**>  
 <**item**>Search trips</**item**>  
 <**item**>Get trips</**item**>  
 <**item**>Sign In</**item**>  
 <**item**>Sign Up</**item**>  
 </**string-array**>  
 <**string-array name="screen\_array\_before"**>  
 <**item**>Search trips</**item**>  
 <**item**>Get trips</**item**>  
 <**item**>Log out</**item**>  
 </**string-array**>  
 <**string name="drawer\_open"**>Open navigation drawer</**string**>  
 <**string name="drawer\_close"**>Close navigation drawer</**string**>  
 <**string name="buttonSearch"**>Search trips</**string**>  
 <**string name="buttonGet"**>Get trips</**string**>  
 <**string name="auth"**>You don\'t authorized</**string**>  
 <**string name="userName"**>Name</**string**>  
 <**string name="email"**>E-Mail</**string**>  
 <**string name="pass"**>Password</**string**>  
 <**string name="rePass"**>Repeat password</**string**>  
 <**string name="forgotPass"**>Forgot password?</**string**>  
  
 <**string name="searchTrip"**>Find a trip</**string**>  
 <**string name="stringDepature"**>Point of depature</**string**>  
 <**string name="stringArrival"**>Point of arrival</**string**>  
 <**string name="stringStopping"**>Stopping places</**string**>  
  
 <**string name="getTrip"**>Get trip</**string**>  
 <**string name="stringTime"**>Time before trip</**string**>  
 <**string name="stringDate"**>Date to start trip</**string**>  
 <**string name="stringPrice"**>Trip price</**string**>  
 <**string name="stringFree"**>Free places</**string**>  
 <**string name="stringPlaces"**>Total places</**string**>  
  
  
 <**string-array name="country\_names"**>  
 <**item**></**item**>  
 </**string-array**>  
</**resources**>

Разметка strings.xml для русскоговорящих:

<**resources**>  
 <**string name="app\_name"**>Кто куда?</**string**>  
 <**string name="search\_screen"**>Поиск поездки</**string**>  
 <**string name="get\_screen"**>Предложение поездки</**string**>  
 <**string name="signin\_screen"**>Вход в аккаунт</**string**>  
 <**string name="signup\_screen"**>Регистрация</**string**>  
 <**string name="action\_settings"**>Настройки</**string**>  
 <**string name="action\_search"**>Поиск</**string**>  
 <**string-array name="screen\_array"**>  
 <**item**>Поиск поездки</**item**>  
 <**item**>Предложить поездку</**item**>  
 <**item**>Вход в аккаунт</**item**>  
 <**item**>Регистрация</**item**>  
 </**string-array**>  
 <**string-array name="screen\_array\_before"**>  
 <**item**>Поиск поездки</**item**>  
 <**item**>Предложить поездку</**item**>  
 <**item**>Выйти</**item**>  
 </**string-array**>  
 <**string name="drawer\_open"**>Панель навигации открыта</**string**>  
 <**string name="drawer\_close"**>Панель навигации закрыта</**string**>  
 <**string name="buttonSearch"**>Найти поездку</**string**>  
 <**string name="buttonGet"**>Предложить поездку</**string**>  
 <**string name="rePass"**>Повторите пароль</**string**>  
 <**string name="userName"**>Имя</**string**>  
 <**string name="auth"**>Вы не авторизованы</**string**>  
 <**string name="pass"**>Пароль</**string**>  
 <**string name="forgotPass"**>Забыли пароль?</**string**>  
  
 <**string name="searchTrip"**>Найти поездку</**string**>  
 <**string name="stringDepature"**>Пункт отправления</**string**>  
 <**string name="stringArrival"**>Пункт прибытия</**string**>  
 <**string name="stringStopping"**>Пункты остановок</**string**>  
  
 <**string name="getTrip"**>Предложить поездку</**string**>  
 <**string name="stringTime"**>Время отправления</**string**>  
 <**string name="stringDate"**>Дата отправления</**string**>  
 <**string name="stringPrice"**>Стоимость поездки</**string**>  
 <**string name="stringFree"**>Мест свободно</**string**>  
 <**string name="stringPlaces"**>Всего мест</**string**>  
  
</**resources**>

В данной разметке используются константные значения строк, которые указываются непосредственно при разметке для других Activity, которые используются для графического интерфейса.

Пример использования строковых констант из strings.xml, которые указываются в android:text.

<**Button  
 android:id="@+id/findTrip"  
 android:layout\_width="300dp"  
 android:layout\_height="40dp"  
 android:layout\_below="@id/pointOfArrival"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="10dp"  
 android:background="@drawable/hbutton"  
 android:text="@string/searchTrip"  
 android:textColor="#FFFFFF"** />

# **4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

# **4.1 Начало работы**

После запуска приложения открывается окно главного меню (Рисунок 2).



Рисунок 2 – «Главное меню»

В этом окне имеется две кнопки: «Найти поездку» и «Предложить поездку».

При нажатии на кнопку «Найти поездку» открывается активное окно «Поиск поездки» (Рисунок 3). Если нажать на кнопку «Предложить поездку», то откроется новое окно, в котором пользователь сможет предложить поездку для поиска попутчиков (Рисунок 4). Если пользователю нужно совершить выход из приложения, необходимо в главном меня нажать системную кнопку «Назад».

# 

Рисунок 3 – «Поиск попутчиков»

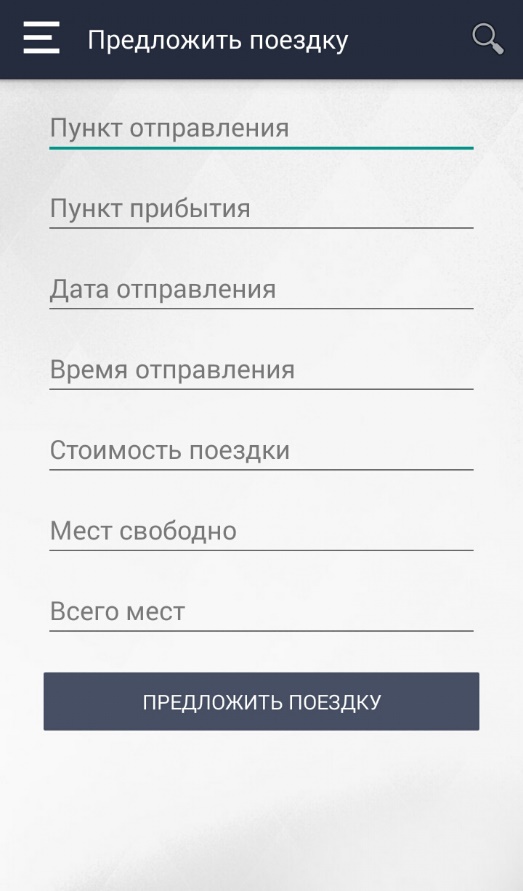


Рисунок 4 – «Предложить поездку»

**4.2 Регистрация**

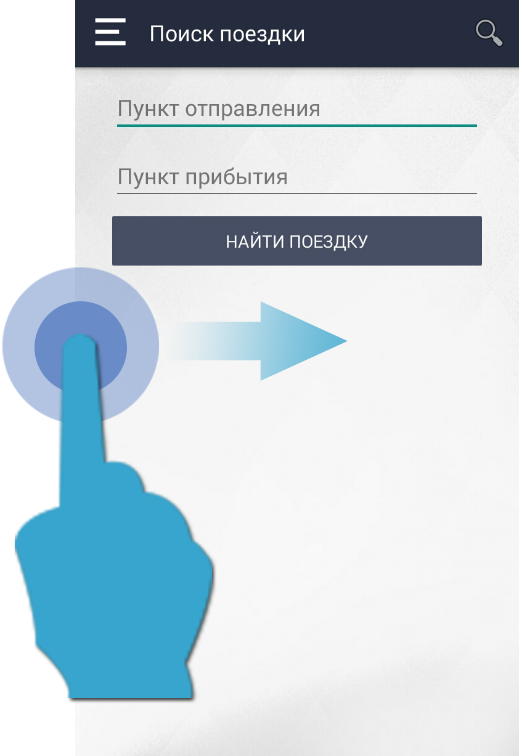
Для того, чтобы зарегистрироваться, необходимо открыть меню, для этого нужно провести пальцем от левого края телефона вправо (Рисунок 5) или нажать на кнопку сверху слева, обозначенная тремя полосками.

Рисунок 5 – «Жест»

Далее, в открытом меню необходимо нажать на кнопку «Регистрация» (Рисунок 6) далее необходимо заполнить все поля и нажать кнопку «Регистрация».

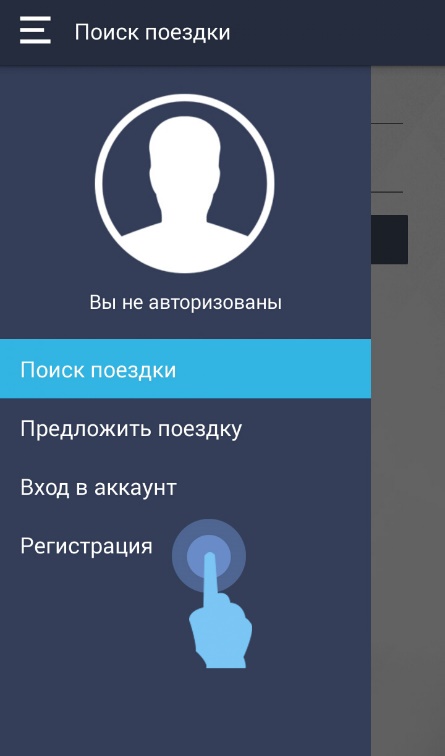


Рисунок 6 – «Кнопка регистрация»

**4.3 Вход в аккаунт**

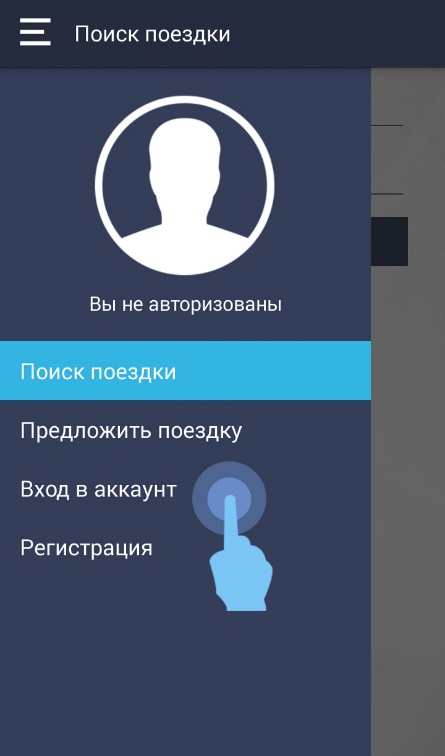
****Для того, чтобы войти в аккаунт, необходимо совершить аналогичные действия как и при регистрации, за исключением некоторым поправок. Вместо кнопки «Регистрация» необходимо выбирать кнопку «Вход в аккаунт» (Рисунок 7).

Рисунок 7 – «Кнопка вход в аккаунт»

После того, как пользователь выполнил все предписывающие требования, он может пользоваться всем функционалом программного обеспечения без ограничений.

**4.4 Поиск попутчика**

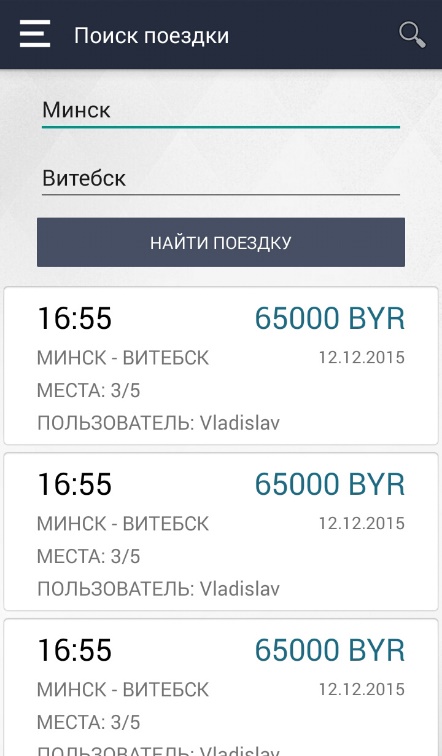
Чтобы воспользоваться услугой поиска попутчика, необходимо в меню перейти во вкладку «Поиск поездки», заполнить все необходимые поля и нажать кнопку «Найти поездку». Далее пользователь увидит все доступные и актуальные поездки на данный момент (Рисунок 8).

Рисунок 8 – «Найденные поездки»

# **ЗАКЛЮЧЕНИЕ**

Было разработано приложение для платформы Android – «Кто куда?», с помощью которого, пользователь может без труда найти попутчика по интересующему ему маршруту.

Приложение было реализовано с использованием среды для разработки приложений на базе Android – Android Studio. При разработке активно использовалось программное средство контроля версий – GitHub.

Для повышения качества разрабатываемого продукта использовался ручной метод тестирования.

Игровое приложение обладает интуитивным управлением, что упрощает его использование для рядового пользователя. Со временем данное приложение будет совершенствоваться: будет добавлена функция просмотра пользователей, оценка пользователей по поездкам, личные сообщения для обмена данными, просмотр личного профиля, редактирование и удаление уже существующих созданных поездок.

# **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Голощапов А.Л. «Google Android программирование для мобильных устройств». BHV Санкт-Петербург, 2011
2. Б. Эккель, «Философия Java»;
3. Роберт Мартин «Чистый Код». Питер, 2014;
4. ГОСТ 19.701–90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – Введ. 1992–01–01. – М. : Изд-во стандартов, 1991.
5. http://cyberforum.ru;
6. http://stackoverflow.com/

**ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ КОД**

**BasicActivity.java**

**public class** BasicActivity **extends** AppCompatActivity {  
  
 Button **searchTrip**, **getTrip**;  
  
 @Override  
 **public void** onCreate(Bundle savedInstanceState){  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***basic\_activity***);  
  
 **searchTrip** = (Button) findViewById(R.id.***buttonSearch***);  
 **getTrip** = (Button) findViewById(R.id.***buttonGet***);  
  
 **searchTrip**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 Intent intent = **new** Intent(BasicActivity.**this**, MainActivity.**class**);  
 intent.putExtra(**"fragmentName"**, **"0"**);  
 startActivity(intent);  
 }  
 });  
  
 **getTrip**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 Intent intent = **new** Intent(BasicActivity.**this**, MainActivity.**class**);  
 intent.putExtra(**"fragmentName"**, **"1"**);  
 startActivity(intent);  
 }  
 });  
  
 }  
  
}

**CustomListAdapter.java**

**public class** CustomListAdapter **extends** ArrayAdapter<String>{  
  
 **private final** Activity **context**;  
 **private final** String[] **timeString**;  
 **private final** String[] **priceString**;  
 **private final** String[] **placesString**;  
 **private final** String[] **dateString**;  
 **private final** String[] **idInt**;  
 **private final** String[] **freePlace**;  
 **private final** String[] **totalPlace**;  
 **private final** String[] **userName**;  
  
 **public** CustomListAdapter(Activity context, String[] timeString, String[] priceString, String[] placesString, String[] dateString, String[] idInt, String[] freePlace, String[] totalPlace, String[] userName) {  
 **super**(context, R.layout.***custom\_listview\_row***, idInt);  
 **this**.**context** = context;  
 **this**.**timeString** = timeString;  
 **this**.**priceString** = priceString;  
 **this**.**placesString** = placesString;  
 **this**.**dateString** = dateString;  
 **this**.**idInt** = idInt;  
 **this**.**freePlace** = freePlace;  
 **this**.**totalPlace** = totalPlace;  
 **this**.**userName** = userName;  
 }  
  
 @Override  
 **public** View getView(**int** position, View convertView, ViewGroup parent) {  
 LayoutInflater inflater = **context**.getLayoutInflater();  
 View rowView = inflater.inflate(R.layout.***custom\_listview\_row***, **null**, **true**);  
  
 TextView textTime = (TextView) rowView.findViewById(R.id.***ts\_time***);  
 TextView textPrice = (TextView) rowView.findViewById(R.id.***ts\_price***);  
 TextView textPlaces = (TextView) rowView.findViewById(R.id.***ts\_trip***);  
 TextView textDate = (TextView) rowView.findViewById(R.id.***ts\_date***);  
 TextView textId = (TextView) rowView.findViewById(R.id.***ts\_id***);  
 TextView textUser = (TextView) rowView.findViewById(R.id.***ts\_driver***);  
 TextView textTotalPlaces = (TextView) rowView.findViewById(R.id.***ts\_places***);  
  
 textTime.setText(**timeString**[position]);  
 textPrice.setText(**priceString**[position] + **" BYR"**);  
 textPlaces.setText(**placesString**[position]);  
 textDate.setText(**dateString**[position]);  
 textId.setText(**idInt**[position]);  
 textUser.setText(**"ПОЛЬЗОВАТЕЛЬ: "** + **userName**[position]);  
 textTotalPlaces.setText(**"МЕСТА: "** + **freePlace**[position] + **"/"** + **totalPlace**[position]);  
  
 **this**.notifyDataSetChanged();  
  
 **return** rowView;  
 }  
}

**GetScreen.java**

**public class** GetScreen **extends** Fragment {  
  
 AutoCompleteTextView **departurePoint**, **arrivalPoint**;  
 EditText **beginTrip**, **dayOfTrip**, **priceTrip**, **freePlaces**, **totalPlaces**;  
 Button **getTripButton**;  
  
 SharedPreferences **sharedPreferences**;  
 **public static final** String ***MyPREFERENCES*** = **"MyPrefs"**;  
 **public static final** String ***PrefId*** = **"idKey"**;  
  
 ProgressDialog **dialog**;  
 Fragment **fragment** = **null**;  
  
 String stringDeparture, **stringArrival**, **stringTime**, **stringDate**, **stringPrice**, **stringFreePlace**, **stringTotalPlace**, **stringUserId**;  
 String **countries**[] = {**"Минск"**,**"Борисов"**,**"Солигорск"**,**"Молодечно"**,**"Жодино"**,**"Слуцк"**,**"Вилейка"**,**"Дзержинск"**,**"Марьина Горка"**,**"Столбцы"**,**"Смолевичи"**,  
 **"Заславль"**,**"Несвиж"**,**"Фаниполь"**,**"Березино"**,**"Любань"**,**"Старые Дороги"**,**"Клецк"**,**"Логойск"**,**"Воложин"**,**"Червень"**,**"Копыль"**,**"Узда"**,**"Крупки"**,**"Мядель"**,  
 **"Витебск"**,**"Орша"**,**"Новополоцк"**,**"Полоцк"**,**"Поставы"**,**"Глубокое"**,**"Лепель"**,**"Новолукомль"**,**"Городок"**,**"Барань"**,**"Толочин"**,**"Браслав"**,**"Чашники"**,**"Миоры"**,  
 **"Дубровно"**,**"Сенно"**,**"Верхнедвинск"**,**"Докшицы"**,**"Дисна"**,**"Могилев"**,**"Осиповичи"**,**"Быхов"**,**"Климовичи"**,**"Шклов"**,**"Мстиславль"**,**"Чериков"**,**"Бобруйск"**,  
 **"Горки"**,**"Кличев"**,**"Костюковичи"**,**"Кричев"**,**"Славгород"**,**"Кировск"**,**"Чаусы"**,**"Гомель"**,**"Мозырь"**,**"Жлобин"**,**"Светлогорск"**,**"Речица"**,**"Калинковичи"**,  
 **"Рогачев"**,**"Добруш"**,**"Житковичи"**,**"Хойники"**,**"Петриков"**,**"Ельск"**,**"Буда-Кошелево"**,**"Наровля"**,**"Ветка"**,**"Чечерск"**,**"Василевичи"**,**"Туров"**,**"Брест"**,  
 **"Барановичи"**,**"Пинск"**,**"Кобрин"**,**"Береза"**,**"Лучинец"**,**"Ивацевичи"**,**"Пружаны"**,**"Иваново"**,**"Дрогичин"**,**"Ганцевичи"**,**"Жабинка"**,**"Микашевичи"**,  
 **"Белоозерск"**,**"Столин"**,**"Малорита"**,**"Ляховичи"**,**"Каменец"**,**"Давид-Городок"**,**"Высокое"**,**"Коссово"**,**"Гродно"**,**"Лида"**,**"Слоним"**,**"Волковыск"**,**"Сморгонь"**,  
 **"Новогрудок"**,**"Мосты"**,**"Щучин"**,**"Ошмяны"**,**"Скидель"**,**"Березовка"**,**"Островец"**,**"Ивье"**,**"Дятлово"**,**"Свислочь"**};  
  
 **public** GetScreen(){  
  
 }  
  
 @Override  
 **public** View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
  
 View rootView = inflater.inflate(R.layout.***get\_screen***, container, **false**);  
  
 **sharedPreferences** = getContext().getSharedPreferences(***MyPREFERENCES***, getContext().***MODE\_PRIVATE***);  
  
 **departurePoint** = (AutoCompleteTextView) rootView.findViewById(R.id.***autoCompleteTextView***);  
 **arrivalPoint** = (AutoCompleteTextView) rootView.findViewById(R.id.***autoCompleteTextView2***);  
 **beginTrip** = (EditText) rootView.findViewById(R.id.***timeString***);  
 **dayOfTrip** = (EditText) rootView.findViewById(R.id.***dateString***);  
 **priceTrip** = (EditText) rootView.findViewById(R.id.***priceString***);  
 **freePlaces** = (EditText) rootView.findViewById(R.id.***freePlaces***);  
 **totalPlaces** = (EditText) rootView.findViewById(R.id.***allPlaces***);  
 **getTripButton** = (Button) rootView.findViewById(R.id.***btnGetTrip***);  
  
 ArrayAdapter<String> adapter = **new** ArrayAdapter<String>(getContext(),R.layout.***autocomplete\_text***, R.id.***textOne***,**countries**);  
  
 **departurePoint**.setAdapter(adapter);  
 **departurePoint**.setThreshold(1);  
 **arrivalPoint**.setAdapter(adapter);  
 **arrivalPoint**.setThreshold(1);  
  
 **getTripButton**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **if** (**sharedPreferences**.contains(***PrefId***)){  
 stringDeparture = String.*valueOf*(SearchIndex(**departurePoint**.getText().toString()));  
 **stringArrival** = String.*valueOf*(SearchIndex(**arrivalPoint**.getText().toString()));  
 **stringTime** = **beginTrip**.getText().toString();  
 **stringDate** = **dayOfTrip**.getText().toString();  
 **stringPrice** = **priceTrip**.getText().toString();  
 **stringFreePlace** = **freePlaces**.getText().toString();  
 **stringTotalPlace** = **totalPlaces**.getText().toString();  
 **stringUserId** = String.*valueOf*(**sharedPreferences**.getInt(**"idKey"**, 0));  
  
 **if** ((stringDeparture.length() > 0) && (**stringArrival**.length() > 0) && (**stringTime**.length() > 0) && (**stringDate**.length() > 0)  
 && (**stringPrice**.length() > 0) && (**stringFreePlace**.length() > 0) && (**stringTotalPlace**.length() > 0))  
 **new** GetTrip().execute(stringDeparture, **stringArrival**, **stringTime**, **stringDate**, **stringPrice**, **stringFreePlace**, **stringTotalPlace**, **stringUserId**);  
 **else** {  
 Toast.*makeText*(getContext(),**"Заполнены не все поля!"**,Toast.***LENGTH\_SHORT***).show();  
 }  
 } **else** {  
 Toast.*makeText*(getContext(), **"Необходимо войти в аккаунт!"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
 });  
  
 **return** rootView;  
 }  
  
 **public int** SearchIndex(String thisText){  
 **int** i = 0, index = 0;  
 **while** ((i < **countries**.**length** - 1) && (index == 0)){  
 **if** (**countries**[i].equals(thisText)) {  
 index = i;  
 }  
 i++;  
 }  
 **return** index;  
 }  
  
 **public class** GetTrip **extends** AsyncTask<String, String, String>{  
  
 @Override  
 **protected void** onPreExecute() {  
 **dialog** = ProgressDialog.*show*(getContext(), **"Adding your trip"**,**"Please wait..."**, **true**);  
 **super**.onPreExecute();  
 }  
  
 @Override  
 **protected** String doInBackground(String... params) {  
  
 List<String> list = **new** ArrayList<String>();  
  
 String departure = params[0];  
 String arrival = params[1];  
 String time = params[2];  
 String date = params[3];  
 String price = params[4];  
 String freePlace = params[5];  
 String totalPlace = params[6];  
 String userId = params[7];  
  
 String link = **"http://192.168.0.104/get.php"**;  
 String data = **null**;  
  
 **try** {  
 data = URLEncoder.*encode*(**"departure"**, **"UTF-8"**) + **"="** + URLEncoder.*encode*(departure, **"UTF-8"**);  
 data += **"&"** + URLEncoder.*encode*(**"arrival"**, **"UTF-8"**) + **"="** + URLEncoder.*encode*(arrival, **"UTF-8"**);  
 data += **"&"** + URLEncoder.*encode*(**"time"**, **"UTF-8"**) + **"="** + URLEncoder.*encode*(time, **"UTF-8"**);  
 data += **"&"** + URLEncoder.*encode*(**"date"**, **"UTF-8"**) + **"="** + URLEncoder.*encode*(date, **"UTF-8"**);  
 data += **"&"** + URLEncoder.*encode*(**"price"**, **"UTF-8"**) + **"="** + URLEncoder.*encode*(price, **"UTF-8"**);  
 data += **"&"** + URLEncoder.*encode*(**"freePlace"**, **"UTF-8"**) + **"="** + URLEncoder.*encode*(freePlace, **"UTF-8"**);  
 data += **"&"** + URLEncoder.*encode*(**"totalPlace"**, **"UTF-8"**) + **"="** + URLEncoder.*encode*(totalPlace, **"UTF-8"**);  
 data += **"&"** + URLEncoder.*encode*(**"userId"**, **"UTF-8"**) + **"="** + URLEncoder.*encode*(userId, **"UTF-8"**);  
  
 URL url = **new** URL(link);  
 URLConnection connection = url.openConnection();  
 connection.setConnectTimeout(15000);  
 connection.setReadTimeout(15000);  
 connection.setDoOutput(**true**);  
  
 OutputStreamWriter wr = **new** OutputStreamWriter(connection.getOutputStream());  
  
 wr.write(data);  
 wr.flush();  
 wr.close();  
  
 BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(connection.getInputStream()));  
  
 StringBuilder sb = **new** StringBuilder();  
 String line = **null**;  
  
 **while** ((line = reader.readLine()) != **null**){  
 sb.append(line);  
 **break**;  
 }  
  
 String finalJson = sb.toString();  
  
 JSONObject parentObject = **new** JSONObject(finalJson);  
 JSONArray parentArray = parentObject.getJSONArray(**"trip"**);  
 JSONObject finalObject = parentArray.getJSONObject(0);  
  
 String success = finalObject.getString(**"success"**);  
 Log.*d*(**"LOL"**, **" "** + success);  
 **return** success;  
  
 } **catch** (IOException | JSONException e) {  
 e.printStackTrace();  
 }  
 **return null**;  
 }  
  
 @Override  
 **protected void** onPostExecute(String s) {  
 **dialog**.dismiss();  
 */\*if (s.equals("1")){  
 Toast.makeText(getContext(),"Your trip added!", Toast.LENGTH\_SHORT).show();  
 fragment = new GetScreen();  
 android.support.v4.app.FragmentManager fragmentManager = getFragmentManager();  
 fragmentManager.beginTransaction().replace(R.id.content\_frame, fragment).commit();  
 } else if (s.equals("0")){  
 Toast.makeText(getContext(),"Server error!",Toast.LENGTH\_SHORT).show();  
 } \*/* Toast.*makeText*(getContext(),**"Your trip added!"**, Toast.***LENGTH\_SHORT***).show();  
 **fragment** = **new** GetScreen();  
 android.support.v4.app.FragmentManager fragmentManager = getFragmentManager();  
 fragmentManager.beginTransaction().replace(R.id.***content\_frame***, **fragment**).commit();  
 Log.*d*(**"LOL"**, **" "** + s);  
 **super**.onPostExecute(s);  
 }  
 }  
  
  
}

**JSONParser.java**

**public class** JSONParser {  
 **static** InputStream *is* = **null**;  
 **static** JSONObject *jObj* = **null**;  
 **static** String *json* = **""**;  
  
 **public** JSONParser(){  
  
 }  
  
 **public** JSONObject makeHttpRequest(String url, String method, List<NameValuePair> params){  
 **if** (method == **"POST"**){  
 **try** {  
 DefaultHttpClient httpClient = **new** DefaultHttpClient();  
 HttpPost httpPost = **new** HttpPost(url);  
 httpPost.setEntity(**new** UrlEncodedFormEntity(params));  
  
 HttpResponse httpResponse = httpClient.execute(httpPost);  
 HttpEntity httpEntity = httpResponse.getEntity();  
 *is* = httpEntity.getContent();  
 } **catch** (Exception e){  
 e.printStackTrace();  
 }  
 }  
  
 **try**{  
 BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(*is*, **"iso-8859-1"**), 8);  
 StringBuilder sb = **new** StringBuilder();  
 String line = **null**;  
 **while** ((line = reader.readLine()) != **null**){  
 sb.append(line + **'\n'**);  
 }  
 *is*.close();  
 *json* = sb.toString();  
 } **catch** (Exception e){  
 e.printStackTrace();  
 }  
  
 **try**{  
 *jObj* = **new** JSONObject(*json*);  
 } **catch** (Exception e){  
 e.printStackTrace();  
 }  
  
 **return** *jObj*;  
 }  
  
}

**LoginUserClass.java**

**public class** LoginUserClass **extends** AsyncTask<String, Void, String> {  
  
 **private** TextView **statusField**, **roleField**;  
 **private** Context **context**;  
  
 **public static final** String ***MyPREFERENCES*** = **"MyPrefs"**;  
 **public static final** String ***PrefEmail*** = **"emailKey"**;  
 **public static final** String ***PrefId*** = **"idKey"**;  
 **public static final** String ***PrefName*** = **"nameKey"**;  
 **public static final** Boolean ***PrefLoginIn*** = **false**;  
 SharedPreferences **sharedPreferences**;  
  
 **public** LoginUserClass(Context context){  
 **this**.**context** = context;  
 }  
  
 **protected void** onPreExecute(){  
  
 }  
  
 @Override  
 **protected** String doInBackground(String... arg0) {  
 **try** {  
 String email = (String)arg0[0];  
 String password = (String)arg0[1];  
 String link = **"http://192.168.0.104/login\_in.php"**;  
 String data = **null**;  
 data = URLEncoder.*encode*(**"email"**, **"UTF-8"**) + **"="** + URLEncoder.*encode*(email,**"UTF-8"**);  
 data += **"&"** + URLEncoder.*encode*(**"password"**, **"UTF-8"**) + **"="** + URLEncoder.*encode*(password, **"UTF-8"**);  
  
 URL url = **new** URL(link);  
 URLConnection conn = url.openConnection();  
 conn.setConnectTimeout(15000);  
 conn.setReadTimeout(15000);  
  
 conn.setDoOutput(**true**);  
 OutputStreamWriter wr = **new** OutputStreamWriter(conn.getOutputStream());  
  
 wr.write(data);  
 wr.flush();  
 wr.close();  
  
 BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(conn.getInputStream()));  
  
 StringBuilder sb = **new** StringBuilder();  
 String line = **null**;  
  
 **while** ((line = reader.readLine()) != **null**){  
 sb.append(line);  
 **break**;  
 }  
  
 **return** sb.toString();  
  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 **return null**;  
 }  
  
 @Override  
 **protected void** onPostExecute(String result){  
  
 **sharedPreferences** = **context**.getSharedPreferences(***MyPREFERENCES***,Context.***MODE\_PRIVATE***);  
  
 String LOG\_TAG = **"my\_log"**;  
 String strJson = result;  
  
 Log.*d*(LOG\_TAG, **" "** + strJson);  
  
 *//Begin the parser JSON* JSONObject dataJsonObj = **null**;  
 String userName = **""**;  
 **int** userId = 0;  
 String userEmail = **""**;  
  
 **try**{  
 dataJsonObj = **new** JSONObject(strJson);  
 JSONArray user = dataJsonObj.getJSONArray(**"user"**);  
 JSONObject username = user.getJSONObject(0);  
 userName = username.getString(**"username"**);  
 userEmail = username.getString(**"email"**);  
 userId = username.getInt(**"id"**);  
  
 Log.*d*(LOG\_TAG,**"Имя: "** + userName + **"\nE-MaiL: "** + userEmail + **"\nId: "** + userId);  
 } **catch** (Exception e){  
 e.printStackTrace();  
 }  
 *//End parser JSON* **if** (userId > 0){  
 Boolean loginIn = **true**;  
 SharedPreferences.Editor editor = **sharedPreferences**.edit();  
 editor.clear();  
  
 editor.putBoolean(String.*valueOf*(***PrefLoginIn***), loginIn);  
 editor.putString(***PrefEmail***, userEmail);  
 editor.putInt(***PrefId***, userId);  
 editor.putString(***PrefName***, userName);  
 editor.commit();  
  
 String UserName = **sharedPreferences**.getString(***PrefName***,**""**);  
 **int** UserId = **sharedPreferences**.getInt(String.*valueOf*(***PrefId***), 0);  
 Log.*d*(LOG\_TAG, **"USERNAME: "** + UserName + **"ID: "** + UserId);  
  
 Intent intent = **context**.getPackageManager().getLaunchIntentForPackage(**context**.getPackageName());  
 intent.addFlags(Intent.***FLAG\_ACTIVITY\_CLEAR\_TOP*** | Intent.***FLAG\_ACTIVITY\_NEW\_TASK***);  
 **context**.startActivity(intent);  
 } **else** {  
 Boolean loginIn = **false**;  
 }  
 }  
  
}

**MainActivity.java**

**public class** MainActivity **extends** AppCompatActivity {  
  
 **private** String[] mScreenTitles;  
 **private** DrawerLayout mDrawerLayout;  
 **private** ListView mDrawerList;  
 **private** RelativeLayout mDrawerRelativeLayout;  
  
 **final** String USERNAME = **"username"**;  
  
 TextView userName;  
 **private** ActionBarDrawerToggle mDrawerToggle;  
 **private** CharSequence mDrawerTitle;  
 **private** CharSequence mTitle;  
  
 **public static final** String MyPREFERENCES = **"MyPrefs"**;  
 **public static final** String PrefEmail = **"emailKey"**;  
 **public static final** String PrefId = **"idKey"**;  
 **public static final** String PrefName = **"nameKey"**;  
 SharedPreferences sharedPreferences;  
  
 **public boolean** userLogginIn = **false**;  
  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
  
 sharedPreferences = getSharedPreferences(MyPREFERENCES, MODE\_PRIVATE);  
  
 String sharedUserName = sharedPreferences.getString(PrefName,**""**);  
 **int** userId = sharedPreferences.getInt(String.valueOf(PrefId), 0);  
  
 Log.d(**"my\_log"**, **"ID: "** + userId);  
  
 userName = (TextView) findViewById(R.id.userName);  
  
 **if** (userId > 0){  
 userLogginIn = **true**;  
 userName.setText(sharedUserName.toString());  
 } **else** {  
 userLogginIn = **false**;  
 userName.setText(**"Вы не авторизованы"**);  
 }  
  
 mTitle = mDrawerTitle = getTitle();  
 **if** (userLogginIn){  
 mScreenTitles = getResources().getStringArray(R.array.screen\_array\_before);  
 } **else** {  
 mScreenTitles = getResources().getStringArray(R.array.screen\_array);  
 }  
 mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer\_layout);  
 mDrawerList = (ListView) findViewById(R.id.left\_drawer);  
 mDrawerRelativeLayout = (RelativeLayout) findViewById(R.id.drawer\_left);  
  
 *//Установка адаптера для ListView* mDrawerList.setAdapter(**new** ArrayAdapter<String>(**this**, R.layout.drawer\_list\_item, mScreenTitles));  
 *//Установка листенеров для слушателей* mDrawerList.setOnItemClickListener(**new** DrawerItemClickListener());  
  
 String fragmentIndex = getIntent().getStringExtra(**"fragmentName"**);  
 **int** fIndex = Integer.parseInt(fragmentIndex);  
 selectItem(fIndex);  
  
 getSupportActionBar().setDisplayHomeAsUpEnabled(**true**);  
 getSupportActionBar().setHomeButtonEnabled(**true**);  
  
 mDrawerToggle = **new** ActionBarDrawerToggle(  
 **this**, */\* host Activity \*/* mDrawerLayout, */\* DrawerLayout object \*/* R.drawable.ic\_drawer, */\* nav drawer icon to replace 'Up' caret \*/* R.string.drawer\_open, */\* "open drawer" description \*/* R.string.drawer\_close */\* "close drawer" description \*/* ) {  
  
 */\*\* Called when a drawer has settled in a completely closed state. \*/* **public void** onDrawerClosed(View view) {  
 getSupportActionBar().setTitle(mTitle);  
 supportInvalidateOptionsMenu(); *// creates call to onPrepareOptionsMenu()* }  
  
 */\*\* Called when a drawer has settled in a completely open state. \*/* **public void** onDrawerOpened(View drawerView) {  
 getSupportActionBar().setTitle(mDrawerTitle);  
 supportInvalidateOptionsMenu(); *// creates call to onPrepareOptionsMenu()* }  
 };  
  
 *// Set the drawer toggle as the DrawerListener* mDrawerLayout.setDrawerListener(mDrawerToggle);  
  
 */\* Initialize the first fragment when the application first loads.  
 if (savedInstanceState == null) {  
 selectItem(0);  
 } \*/* }  
  
 @Override  
 **public boolean** onCreateOptionsMenu(Menu menu) {  
 *// Inflate the menu;* MenuInflater inflater = getMenuInflater();  
 inflater.inflate(R.menu.menu\_main, menu);  
 **return super**.onCreateOptionsMenu(menu);  
 }  
  
 @Override  
 **public boolean** onPrepareOptionsMenu(Menu menu) {  
 *// If the nav drawer is open, hide action items related to the content view* **boolean** drawerOpen = mDrawerLayout.isDrawerOpen(mDrawerRelativeLayout);  
 menu.findItem(R.id.action\_search).setVisible(!drawerOpen);  
 **return super**.onPrepareOptionsMenu(menu);  
 }  
  
 @Override  
 **public boolean** onOptionsItemSelected(MenuItem item) {  
 *// Pass the event to ActionBarDrawerToggle, if it returns  
 // true, then it has handled the app icon touch event* **if** (mDrawerToggle.onOptionsItemSelected(item)) {  
 **return true**;  
 }  
 *// Handle action buttons* **switch**(item.getItemId()) {  
 **case** R.id.action\_search:  
 *// Show toast about click.* Toast.makeText(**this**, R.string.action\_search, Toast.LENGTH\_SHORT).show();  
 **return true**;  
 **default**:  
 **return super**.onOptionsItemSelected(item);  
 }  
 }  
  
 */\* The click listener for ListView in the navigation drawer \*/* **private class** DrawerItemClickListener **implements** ListView.OnItemClickListener {  
 @Override  
 **public void** onItemClick(AdapterView<?> parent, View view, **int** position, **long** id) {  
 selectItem(position);  
 }  
 }  
  
 */\*\* Swaps fragments in the main content view \*/* **public void** selectItem(**int** position) {  
 *// Update the main content by replacing fragments* Fragment fragment = **null**;  
 **switch** (position) {  
 **case** 0:  
 fragment = **new** SearchScreen();  
 **break**;  
 **case** 1:  
 fragment = **new** GetScreen();  
 **break**;  
 **case** 2:  
 **if** (userLogginIn){  
 SharedPreferences.Editor editor = sharedPreferences.edit();  
 editor.clear();  
 editor.commit();  
 Intent intent = getIntent();  
 finish();  
 startActivity(intent);  
 } **else** {  
 fragment = **new** SigninScreen();  
 }  
 **break**;  
 **case** 3:  
 fragment = **new** SignupScreen();  
 **break**;  
 **default**:  
 **break**;  
 }  
  
 *// Insert the fragment by replacing any existing fragment* **if** (fragment != **null**) {  
 FragmentManager fragmentManager = getSupportFragmentManager();  
 fragmentManager.beginTransaction()  
 .replace(R.id.content\_frame, fragment).commit();  
  
 *// Highlight the selected item, update the title, and close the drawer* mDrawerList.setItemChecked(position, **true**);  
 setTitle(mScreenTitles[position]);  
 mDrawerLayout.closeDrawer(mDrawerRelativeLayout);  
 } **else** {  
 *// Error* Log.e(**this**.getClass().getName(), **"Error. Fragment is not created"**);  
 }  
 }  
  
 @Override  
 **public void** setTitle(CharSequence title) {  
 mTitle = title;  
 getSupportActionBar().setTitle(mTitle);  
 }  
  
 */\*\*  
 \* When using the ActionBarDrawerToggle, you must call it during  
 \* onPostCreate() and onConfigurationChanged()...  
 \*/* @Override  
 **protected void** onPostCreate(Bundle savedInstanceState) {  
 **super**.onPostCreate(savedInstanceState);  
 *// Sync the toggle state after onRestoreInstanceState has occurred.* mDrawerToggle.syncState();  
 }  
  
 @Override  
 **public void** onConfigurationChanged(Configuration newConfig) {  
 **super**.onConfigurationChanged(newConfig);  
 *// Pass any configuration change to the drawer toggles* mDrawerToggle.onConfigurationChanged(newConfig);  
 }  
  
}

**RegisterUserClass.java**

**public class** RegisterUserClass {  
  
 **public** String sedPostRequest(String requestURL, HashMap<String, String> postDataParams){  
 URL url;  
 String response = **""**;  
 **try**{  
 url = **new** URL(requestURL);  
  
 HttpURLConnection conn = (HttpURLConnection) url.openConnection();  
 conn.setReadTimeout(15000);  
 conn.setConnectTimeout(15000);  
 conn.setRequestMethod(**"POST"**);  
 conn.setDoOutput(**true**);  
 conn.setDoInput(**true**);  
  
 OutputStream os = conn.getOutputStream();  
 BufferedWriter writer = **new** BufferedWriter(**new** OutputStreamWriter(os, **"UTF-8"**));  
 writer.write(getPostDataString(postDataParams));  
  
 writer.flush();  
 writer.close();  
 os.close();  
 **int** responseCode = conn.getResponseCode();  
  
 **if** (responseCode == HttpsURLConnection.HTTP\_OK){  
 BufferedReader br = **new** BufferedReader(**new** InputStreamReader(conn.getInputStream()));  
 response = br.readLine();  
 } **else** {  
 response = **"132"**;  
 }  
  
 } **catch**(Exception e) {  
 e.printStackTrace();  
 }  
 **return** response;  
 }  
  
 **private** String getPostDataString(HashMap<String,String> params) **throws** UnsupportedEncodingException {  
 StringBuilder result = **new** StringBuilder();  
 **boolean** first = **true**;  
 **for** (Map.Entry<String,String> entry : params.entrySet()){  
 **if** (first){  
 first = **false**;  
 } **else** {  
 result.append(**"&"**);  
 }  
 result.append(URLEncoder.encode(entry.getKey(), **"UTF-8"**));  
 result.append(**"="**);  
 result.append(URLEncoder.encode(entry.getValue(), **"UTF-8"**));  
 }  
 **return** result.toString();  
 }  
  
}

**SearchScreen.java**

**public class** SearchScreen **extends** Fragment{  
  
 AutoCompleteTextView **pointDeparture**, **pointArrival**;  
 RelativeLayout **relativeLayout**;  
 Button **findTrip**;  
 String[] **priceString**,**timeString**, **placesString**, **dateString**, **idInt**, **freePlace**, **totalPlace**, **userName**;  
 List<String> **forPrice** = **new** ArrayList<String>();  
 List<String> **forTime** = **new** ArrayList<String>();  
 List<String> **forPlaces** = **new** ArrayList<String>();  
 List<String> **forDate** = **new** ArrayList<String>();  
 List<String> **forId** = **new** ArrayList<String>();  
 List<String> **forSpace** = **new** ArrayList<String>();  
 List<String> **forTotalSpace** = **new** ArrayList<>();  
 List<String> **forUserName** = **new** ArrayList<>();  
  
 CustomListAdapter **adapters**;  
 ProgressDialog **dialog**;  
 **public** String **forJson**;  
 String **countries**[] = {**"Минск"**,**"Борисов"**,**"Солигорск"**,**"Молодечно"**,**"Жодино"**,**"Слуцк"**,**"Вилейка"**,**"Дзержинск"**,**"Марьина Горка"**,**"Столбцы"**,**"Смолевичи"**,  
 **"Заславль"**,**"Несвиж"**,**"Фаниполь"**,**"Березино"**,**"Любань"**,**"Старые Дороги"**,**"Клецк"**,**"Логойск"**,**"Воложин"**,**"Червень"**,**"Копыль"**,**"Узда"**,**"Крупки"**,**"Мядель"**,  
 **"Витебск"**,**"Орша"**,**"Новополоцк"**,**"Полоцк"**,**"Поставы"**,**"Глубокое"**,**"Лепель"**,**"Новолукомль"**,**"Городок"**,**"Барань"**,**"Толочин"**,**"Браслав"**,**"Чашники"**,**"Миоры"**,  
 **"Дубровно"**,**"Сенно"**,**"Верхнедвинск"**,**"Докшицы"**,**"Дисна"**,**"Могилев"**,**"Осиповичи"**,**"Быхов"**,**"Климовичи"**,**"Шклов"**,**"Мстиславль"**,**"Чериков"**,**"Бобруйск"**,  
 **"Горки"**,**"Кличев"**,**"Костюковичи"**,**"Кричев"**,**"Славгород"**,**"Кировск"**,**"Чаусы"**,**"Гомель"**,**"Мозырь"**,**"Жлобин"**,**"Светлогорск"**,**"Речица"**,**"Калинковичи"**,  
 **"Рогачев"**,**"Добруш"**,**"Житковичи"**,**"Хойники"**,**"Петриков"**,**"Ельск"**,**"Буда-Кошелево"**,**"Наровля"**,**"Ветка"**,**"Чечерск"**,**"Василевичи"**,**"Туров"**,**"Брест"**,  
 **"Барановичи"**,**"Пинск"**,**"Кобрин"**,**"Береза"**,**"Лучинец"**,**"Ивацевичи"**,**"Пружаны"**,**"Иваново"**,**"Дрогичин"**,**"Ганцевичи"**,**"Жабинка"**,**"Микашевичи"**,  
 **"Белоозерск"**,**"Столин"**,**"Малорита"**,**"Ляховичи"**,**"Каменец"**,**"Давид-Городок"**,**"Высокое"**,**"Коссово"**,**"Гродно"**,**"Лида"**,**"Слоним"**,**"Волковыск"**,**"Сморгонь"**,  
 **"Новогрудок"**,**"Мосты"**,**"Щучин"**,**"Ошмяны"**,**"Скидель"**,**"Березовка"**,**"Островец"**,**"Ивье"**,**"Дятлово"**,**"Свислочь"**};  
  
 ListView **listView**;  
  
 **public** SearchScreen() {  
  
 }  
  
 @Override  
 **public** View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
  
 View rootView = inflater.inflate(R.layout.***search\_screen***, container, **false**);  
  
 **pointDeparture** = (AutoCompleteTextView) rootView.findViewById(R.id.***pointOfDepature***);  
 **pointArrival** = (AutoCompleteTextView) rootView.findViewById(R.id.***pointOfArrival***);  
 **relativeLayout** = (RelativeLayout) rootView.findViewById(R.id.***mainLayout***);  
 **findTrip** = (Button) rootView.findViewById(R.id.***findTrip***);  
  
 **listView** = (ListView) rootView.findViewById(R.id.***listView***);  
  
 **final** ArrayAdapter<String> adapter = **new** ArrayAdapter<String>(getContext(),R.layout.***autocomplete\_text***, R.id.***textOne***,**countries**);  
  
 **pointDeparture**.setAdapter(adapter);  
 **pointDeparture**.setThreshold(1);  
 **pointArrival**.setAdapter(adapter);  
 **pointArrival**.setThreshold(1);  
  
 **findTrip**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **if** (**pointArrival**.length() != 0 && **pointDeparture**.length() != 0) {  
 **if** ((IsExist(**pointArrival**.getText().toString())) && (IsExist(**pointDeparture**.getText().toString()))) {  
 String inputPoint = String.*valueOf*(SearchIndex(**pointArrival**.getText().toString()));  
  
 String outputPoint = String.*valueOf*(SearchIndex(**pointDeparture**.getText().toString()));  
 Log.*d*(**"LOL"**, **"departure "** + inputPoint);  
 **listView**.setAdapter(**null**);  
 **new** JSONTask().execute(outputPoint, inputPoint);  
 } **else** {  
 Toast.*makeText*(getContext(),**"Ничего не найдено!"**,Toast.***LENGTH\_SHORT***).show();  
 **listView**.setAdapter(**null**);  
 }  
 } **else** {  
 Toast.*makeText*(getContext(), **"Заполнены не все поля!"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
 });  
  
 **return** rootView;  
 }  
  
 **public int** SearchIndex(String thisText){  
 **int** i = 0, index = 0;  
 **while** ((i < **countries**.**length** - 1) && (index == 0)){  
 **if** (**countries**[i].toUpperCase().equals(thisText.toUpperCase())) {  
 index = i;  
 }  
 i++;  
 }  
 **return** index;  
 }  
  
 **public boolean** IsExist(String thisText){  
 **int** i = 0;  
 **boolean** exist = **false**;  
 **while** ((i < **countries**.**length** - 1) && (!exist)){  
 **if** (**countries**[i].toUpperCase().equals(thisText.toUpperCase())){  
 exist = **true**;  
 }  
 i++;  
 }  
 **return** exist;  
 }  
  
 **public class** JSONTask **extends** AsyncTask<String, String, String[]> {  
 String[] **allElements**;  
 String **tripPath**;  
  
 @Override  
 **protected void** onPreExecute() {  
 **dialog** = ProgressDialog.*show*(getContext(), **"Loading"**,**"Please wait..."**, **true**);  
 **super**.onPreExecute();  
 }  
  
 @Override  
 **protected** String[] doInBackground(String... params) {  
  
 List<String> list = **new** ArrayList<String>();  
 **try**{  
 String departure = params[0];  
 String arrival = params[1];  
 String link = **"http://192.168.0.104/search.php"**;  
 String data = **null**;  
 data = URLEncoder.*encode*(**"departure"**, **"UTF-8"**) + **"="** + URLEncoder.*encode*(departure, **"UTF-8"**);  
 data += **"&"** + URLEncoder.*encode*(**"arrival"**, **"UTF-8"**) + **"="** + URLEncoder.*encode*(arrival, **"UTF-8"**);  
  
 URL url = **new** URL(link);  
 URLConnection connection = url.openConnection();  
 connection.setConnectTimeout(15000);  
 connection.setReadTimeout(15000);  
  
 connection.setDoOutput(**true**);  
 OutputStreamWriter wr = **new** OutputStreamWriter(connection.getOutputStream());  
  
 wr.write(data);  
 wr.flush();  
 wr.close();  
  
 BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(connection.getInputStream()));  
  
 StringBuilder sb = **new** StringBuilder();  
 String line = **null**;  
  
 **while** ((line = reader.readLine()) != **null**){  
 sb.append(line);  
 **break**;  
 }  
  
 String finalJson = sb.toString();  
 String success;  
  
 JSONObject parentObject = **new** JSONObject(finalJson);  
 JSONArray parentArray = parentObject.getJSONArray(**"trip"**);  
  
 **for** (**int** i = 0; i < parentArray.length(); i++){  
 JSONObject finalObject = parentArray.getJSONObject(i);  
 list.add(finalObject.getString(**"id"**));  
 **tripPath** = **countries**[Integer.*parseInt*(finalObject.getString(**"departure"**))].toUpperCase() + **" - "** + **countries**[Integer.*parseInt*(finalObject.getString(**"arrival"**))].toUpperCase();  
 list.add(**tripPath**);  
 list.add(finalObject.getString(**"time"**));  
 list.add(finalObject.getString(**"date"**));  
 list.add(finalObject.getString(**"price"**));  
 list.add(finalObject.getString(**"username"**));  
 list.add(finalObject.getString(**"freeplace"**));  
 list.add(finalObject.getString(**"totalplace"**));  
 }  
  
 **allElements** = **new** String[list.size()];  
 list.toArray(**allElements**);  
 list.clear();  
  
 **return allElements**;  
  
 } **catch** (JSONException | IOException e) {  
 e.printStackTrace();  
 }  
 **return null**;  
 }  
  
 @Override  
 **protected void** onPostExecute(String[] result) {  
 **int** i = 0;  
 **while** (i < result.**length**){  
 **forId**.add(result[i]);  
 **forPlaces**.add(result[1 + i]);  
 **forTime**.add(result[2 + i]);  
 **forDate**.add(result[3 + i]);  
 **forPrice**.add(result[4 + i]);  
 **forUserName**.add(result[5 + i]);  
 **forSpace**.add(result[6 + i]);  
 **forTotalSpace**.add(result[7 + i]);  
 i = i + 8;  
 }  
  
 **idInt** = **new** String[**forId**.size()];  
 **placesString** = **new** String[**forPlaces**.size()];  
 **timeString** = **new** String[**forTime**.size()];  
 **dateString** = **new** String[**forDate**.size()];  
 **priceString** = **new** String[**forPrice**.size()];  
 **userName** = **new** String[**forUserName**.size()];  
 **freePlace** = **new** String[**forSpace**.size()];  
 **totalPlace** = **new** String[**forTotalSpace**.size()];  
  
 **forId**.toArray(**idInt**);  
 **forPlaces**.toArray(**placesString**);  
 **forTime**.toArray(**timeString**);  
 **forDate**.toArray(**dateString**);  
 **forPrice**.toArray(**priceString**);  
 **forUserName**.toArray(**userName**);  
 **forSpace**.toArray(**freePlace**);  
 **forTotalSpace**.toArray(**totalPlace**);  
  
 **forId**.clear();  
 **forPrice**.clear();  
 **forPlaces**.clear();  
 **forTime**.clear();  
 **forDate**.clear();  
 **forUserName**.clear();  
 **forSpace**.clear();  
 **forTotalSpace**.clear();  
 Log.*d*(**"LOL"**, **placesString**[0]);  
 Log.*d*(**"LOL"**, **timeString**[0]);  
 Log.*d*(**"LOL"**, **dateString**[0]);  
 Log.*d*(**"LOL"**, **priceString**[0]);  
  
 **adapters** = **new** CustomListAdapter((Activity) getContext(), **timeString**, **priceString**, **placesString**, **dateString**, **idInt**, **freePlace**, **totalPlace**, **userName**);  
 **adapters**.notifyDataSetChanged();  
 **listView**.setAdapter(**adapters**);  
 **dialog**.dismiss();  
 **super**.onPostExecute(result);  
 }  
 }  
  
}

**SigninScreen.java**

**public class** SigninScreen **extends** Fragment **implements** View.OnClickListener {  
  
 EditText **userEmail**, **userPass**;  
 Button **signIn**;  
 TextView **textInfo**, **signUp**;  
 Fragment **fragment** = **null**;  
  
 *//For "new user JSON parse"* **private** ProgressDialog **pDialog**;  
 JSONParser **jsonParser** = **new** JSONParser();  
 ArrayList<HashMap<String, String>> **userList**;  
 **private static** String *urlLogin* = **"http://192.168.0.104/login\_in.php"**;  
 **private static final** String ***TAG\_SUCCESS*** = **"success"**;  
 **private static final** String ***TAG\_USERNAME*** = **"username"**;  
 **private static final** String ***TAG\_ID*** = **"id"**;  
 JSONArray **user** = **null**;  
 *//End "new JSONParser"* **public** SigninScreen() {  
  
 }  
  
 @Override  
 **public** View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
  
 **final** View rootView = inflater.inflate(R.layout.***signin\_screen***, container, **false**);  
  
 **signIn** = (Button) rootView.findViewById(R.id.***btnSignIn***);  
 **userEmail** = (EditText) rootView.findViewById(R.id.***userEmail***);  
 **userPass** = (EditText) rootView.findViewById(R.id.***userPass***);  
 **textInfo** = (TextView) rootView.findViewById(R.id.***textInfo***);  
 **signUp** = (TextView) rootView.findViewById(R.id.***signUpInLogin***);  
  
 **signUp**.setOnClickListener(**this**);  
 **signIn**.setOnClickListener(**this**);  
  
 **return** rootView;  
 }  
  
 @Override  
 **public void** onClick(View v) {  
 **switch**(v.getId()){  
 **case** R.id.***btnSignIn***:  
 **if** (!(**userEmail**.getText().toString().matches(**""**)) && !(**userPass**.getText().toString().matches(**""**))){  
 String email = **userEmail**.getText().toString();  
 String password = **userPass**.getText().toString();  
 **new** LoginUserClass(getActivity()).execute(email,password);  
 } **else** {  
 **textInfo**.setTextColor(Color.***RED***);  
 **textInfo**.setText(**"Заполнены не все поля!"**);  
 }  
 **break**;  
 **case** R.id.***signUpInLogin***:  
 **fragment** = **new** SignupScreen();  
 android.support.v4.app.FragmentManager fragmentManager = getFragmentManager();  
 fragmentManager.beginTransaction().replace(R.id.***content\_frame***,**fragment**).commit();  
 **break**;  
 }  
 }  
  
}

**SignupScreen.java**

**public class** SignupScreen **extends** Fragment **implements** View.OnClickListener {  
  
 **public** SignupScreen() {  
 }  
 Button **signUp**;  
 EditText **email**, **pass**, **repass**, **name**;  
 TextView **infoText**;  
 String **user\_email**, **user\_name**, **user\_pass**;  
 Fragment **fragment** = **null**;  
 **private static final** String ***REGISTER\_URL*** = **"http://droidtest.com/register\_account.php"**;  
 @Override  
 **public** View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
  
 **final** View rootView = inflater.inflate(R.layout.***signup\_screen***, container, **false**);  
  
 **signUp** = (Button) rootView.findViewById(R.id.***btnSignUp***);  
 **email** = (EditText) rootView.findViewById(R.id.***userEmail***);  
 **pass** = (EditText) rootView.findViewById(R.id.***userPass***);  
 **repass** = (EditText) rootView.findViewById(R.id.***rePass***);  
 **name** = (EditText) rootView.findViewById(R.id.***userName***);  
 **infoText** = (TextView) rootView.findViewById(R.id.***textInfo***);  
 **signUp**.setOnClickListener(**this**);  
  
 **return** rootView;  
 }  
 @Override  
 **public void** onClick(View v) {  
 **switch**(v.getId()){  
 **case** R.id.***btnSignUp***:  
 **if** (!(**email**.getText().toString().matches(**""**)) && !(**pass**.getText().toString().matches(**""**)) && !(**repass**.getText().toString().matches(**""**)) && !(**name**.getText().toString().matches(**""**))){  
 **if** ((**pass**.getText().toString().equals(**repass**.getText().toString()))){  
 registerUser();  
 showMessage(1);  
 } **else** {  
 showMessage(2);  
 }  
 } **else** {  
 showMessage(0);  
 }  
 **break**;  
 }  
 }  
  
 **public void** showMessage(**int** indexAlert){  
 **if** (indexAlert == 0){  
 **infoText**.setTextColor(Color.***RED***);  
 **infoText**.setText(**"Заполнены не все поля"**);  
 **infoText**.setTextSize(16);  
 } **else if** (indexAlert == 1) {  
 **infoText**.setTextColor(Color.***BLACK***);  
 **infoText**.setText(**"Все поля заполнены!"**);  
 **infoText**.setTextSize(16);  
 } **else if** (indexAlert == 2){  
 **infoText**.setTextColor(Color.***RED***);  
 **infoText**.setText(**"Введенные пароли не совпадают!"**);  
 **infoText**.setTextSize(16);  
 }  
 }  
  
 **public void** register(String userEmail, String userPassword, String userName) {  
 String urlSuffix = **"?email="** + userEmail + **"&password="** + userPassword + **"&username="** + userName; *// для GET запроса* **class** RegisterUser **extends** AsyncTask<String, Void, String> {  
 ProgressDialog **loading**;  
 RegisterUserClass **ruc** = **new** RegisterUserClass();  
  
 @Override  
 **protected void** onPreExecute() {  
 **super**.onPreExecute();  
 **loading** = ProgressDialog.*show*(getActivity(), **"Пожалуйста, подождите!"**, **null**, **true**, **true**);  
 }  
  
 @Override  
 **protected void** onPostExecute(String s) {  
 **super**.onPostExecute(s);  
 **loading**.dismiss();  
 String answerServer = **"Connection timeout"**;  
 **if** (s.matches(**"130"**)){  
 answerServer = **"Register successfull"**;  
 **fragment** = **new** SigninScreen();  
 android.support.v4.app.FragmentManager fragmentManager = getFragmentManager();  
 fragmentManager.beginTransaction().replace(R.id.***content\_frame***, **fragment**).commit();  
 } **else if** (s.matches(**"131"**)){  
 answerServer = **"This email is already in use"**;  
 } **else if** (s.matches(**"132"**)){  
 answerServer = **"Error register"**;  
 } **else if** (s.matches(**"133"**)){  
 answerServer = **"Not all fields was filled"**;  
 } **else if** (s.matches(**"134"**)){  
 answerServer = **"Oops! Try again!"**;  
 }  
 Toast toast = Toast.*makeText*(getActivity(), answerServer, Toast.***LENGTH\_LONG***);  
 toast.show();  
 }  
 @Override  
 **protected** String doInBackground(String... params) {  
 HashMap<String, String> data = **new** HashMap<String, String>();  
 data.put(**"email"**, params[0]);  
 data.put(**"password"**, params[1]);  
 data.put(**"username"**, params[2]);  
  
 String result = **ruc**.sedPostRequest(***REGISTER\_URL***,data);  
  
 **return** result;  
 }  
 }  
 RegisterUser ru = **new** RegisterUser();  
 ru.execute(userEmail,userPassword,userName);  
 }  
}