

**Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Институт вычислительной математики и информационных технологий
Кафедра системного анализа и информационных технологий

Направление подготовки: 10.03.01 – Информационная безопасность
Профиль: Безопасность компьютерных систем

КУРСОВАЯ РАБОТА

**Распределенное приложение с мобильным клиентом на платформе
Android "Автошколы России"**

Студент 3 курса

группы 09-641

«___» _____ 2019 г. _____ Десятов А.Г.

Научный руководитель

доцент, к.н.

«___» _____ 2019 г. _____ Андрианова А.А.

Казань-2019

Содержание

ВВЕДЕНИЕ	3
1. АРХИТЕКТУРА И ТЕХНОЛОГИИ.....	6
1.1. Основные понятия	6
1.2. Формирование запроса.	8
1.3. Обработка ответа на запрос.....	9
2. ХРАНЕНИЕ ДАННЫХ.....	10
3. СИНТАКСИЧЕСКИЙ АНАЛИЗАТОР, ИЛИ ПАРСЕР	15
4. СЕРВЕРНАЯ ЧАСТЬ ПРИЛОЖЕНИЯ.....	21
5. КЛИЕНТСКАЯ ЧАСТЬ ПРИЛОЖЕНИЯ	23
5.1. Activity с выбором города	23
5.2. Activity с выбором категории	25
5.3. Activity с Google-картой	26
5.4. Инфо-окна маркеров	29
ЗАКЛЮЧЕНИЕ	31
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ.....	32
ПРИЛОЖЕНИЯ.....	34
1. Код парсера	34
2. Код клиентской части приложения.....	41
3. Код серверной части приложения	52

ВВЕДЕНИЕ

На сегодняшний день автомобили являются феноменально доступными транспортными средствами. Да и покупка автомобилей – дело не особо сложное. Очевидно, что с увеличением количества транспортных средств, растет и актуальность автошкол. Умение водить машину, а также правильно реагировать на ситуации на дорогах – незаменимы для водителей. [1]

Ежегодно в России более одного миллиона человек получают водительские права на управление транспортным средством. Кроме того, люди часто проходят переподготовку и обучаются на дополнительные категории прав. Перед сдачей экзамена будущим автолюбителям необходимо выбрать автошколу для обучения.

Разрабатываемая информационная система будет содержать клиентское мобильное приложение, которое поможет пользователю сделать этот непростой выбор. С помощью мобильного приложения он сможет найти на Google-картах автошколы, до которых ему будет легко добираться, а также пользователь сможет сравнить цены автошкол, расположенных рядом друг с другом.

Информационная система кроме клиентского приложения, будет содержать серверную часть, поэтому такая система будет являться распределенной. Распределенная информационная система должна решать различные проблемы.

К возможным проблемам разрабатываемой системы относятся возможные сбои в функционировании сервера и атаки, которым может быть подвержено как серверное, так и клиентское приложение.

Никакими секретными данными клиент и сервер не обмениваются, однако вопрос безопасности остается актуальным. Ведь злоумышленник может изменять ответы на запросы и дезинформировать клиента, может навязать какую-либо рекламу или передать пользователю другую ненужную информацию.

Также возможны атаки на сервер. Например, возможна перегрузка сервера, если на него злонамеренно посылать много случайных запросов. Из-за перегрузки сервера вся информационная система будет либо работать медленно, либо вовсе не работать, и вследствие этого может пострадать репутация информационной системы, и, соответственно, пользователи не захотят в дальнейшем пользоваться данным мобильным приложением, кроме того они могут написать негативный отзыв, который будет отталкивать новых пользователей даже после установления хорошей защиты от перегрузки сервера.

Для решения перечисленных проблем следует защищать и клиентскую, и серверную часть информационной системы.

Главной целью данной работы является разработка распределенного приложения "Автошколы России" с мобильным клиентом на платформе Android. Эта операционная система выбрана, потому что она является самой популярной для смартфонов на данный момент.

Перед началом разработки приложения были поставлены следующие задачи:

- 1) Найти источник данных.
- 2) Разработать приложение на языке программирования C#, которое возьмет все необходимые данные из источника и которое сохранит их в удобной форме для дальнейшей работы.
- 3) Подробно изучить работу с Google-картами на языке программирования Java, научиться создавать собственные маркеры и расставлять их на карте.
- 4) Разработать серверную часть приложения на языке программирования PHP, которая будет отправлять клиенту запрошенные данные.
- 5) Расположить все данные на сервере.
- 6) Реализовать Android-приложение, которое получает информацию с сервера, дает возможность пользователю выбрать город и категорию прав,

предоставляет карту с расставленными на ней маркерами – автошколами, также выдает подробную информацию о интересующей автошколе.

7) Проводить тестирование полученного приложения, исправлять найденные ошибки

1. АРХИТЕКТУРА И ТЕХНОЛОГИИ

1.1. Основные понятия

Android – это операционная система, которая используется не только в смартфонах, но и электронных книгах. Дополнительно ее можно встретить у некоторых игровых приставок и даже наручных часов. В дальнейшем указанная операционная система будет устанавливаться на автомобили [2]. Из-за такого широкого распространения этой системы именно эта платформа выбрана для реализации клиентской части данного проекта.

Activity, или Активность. Активность можно воспринимать как форму. Простые Android-приложения состоят из одной активности. Более сложные приложения могут иметь несколько окон, т.е. они состоят из нескольких активностей, которыми надо уметь управлять и которые могут взаимодействовать между собой [3].

Intent представляет собой объект обмена сообщениями [4]. С помощью Intent, например, можно передавать параметры между Activity.

Карты Google — набор приложений, построенных на основе платного картографического сервиса и технологии, предоставляемых компанией Google. [5]. Именно эти карты будут использоваться в приложении «Автошколы России»

Инфо-окно – всплывающее окно над картой, в определенном месте [6].

Распределённое приложение – это программа, состоящая из нескольких взаимодействующих частей, каждая из которых, как правило, выполняется на отдельном компьютере (или другом устройстве) сети (Рисунок 1) [7].



Рисунок 1. Распределенное приложение

Распределённая система – программно-аппаратное решение, состоящее из компонентов, функционирующих на физически удаленных и независимых друг от друга гетерогенных узлах, представляющее пользователям единой объединенной системой [7]. На рисунке 2 изображена архитектура клиент-сервер.

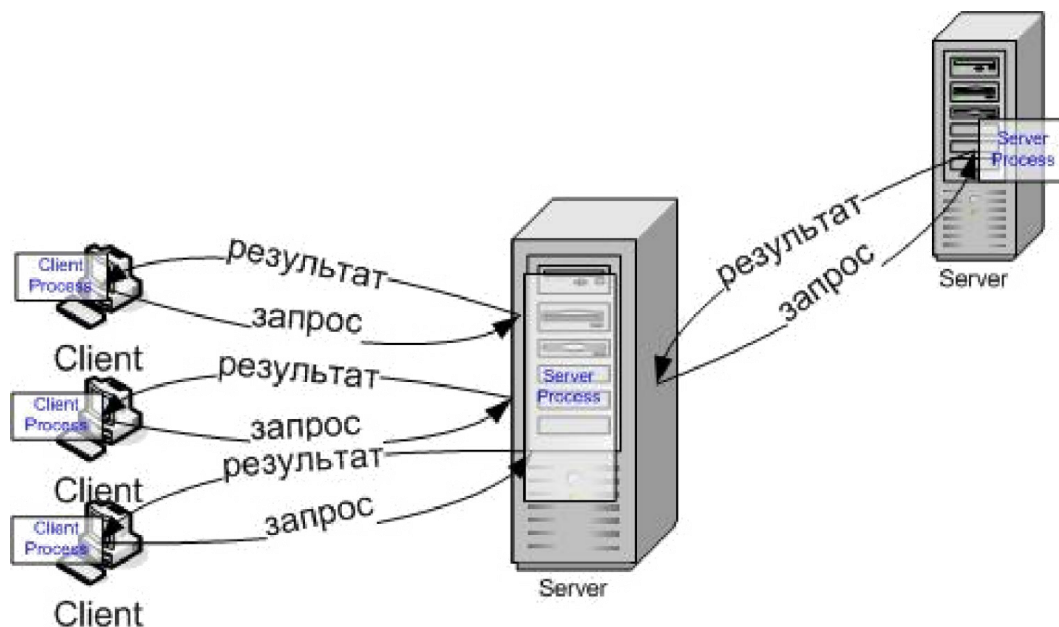


Рисунок 2. Архитектура клиент-сервер

GET запрос используется, чтобы получить данные, а **POST запрос** используется, чтобы отправить [8].

Парсер, или **синтаксический анализатор** — часть программы, преобразующей входные данные (как правило, текст) в структурированный формат.

Различают следующие виды парсеров:

- 1) Парсер, который выполняет преобразование текста, записанного на каком-либо языке программирования (или языке разметки) во внутреннее представление, удобное для дальнейшей работы.
- 2) Парсеры, которые применяются в разработке компьютерных игр при работе с текстовыми файлами, хранящими 3D графику.
- 3) Парсеры, которые применяются при разборе баз данных, сохранённых в текстовых форматах (таких, как CSV, XML и т. п.) [9].

Непосредственно в данном проекте будет реализован парсер первого вида, который будет выполнять преобразования текста, записанного на языке разметки HTML, во внутреннее представление.

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств [10].

JavaScript — это язык программирования, который добавляет интерактивность на веб-сайт [11].

Интерфейс (ударение на вторую «е») это, в широком смысле, набор инструментов для взаимодействия человека и компьютерной системы [12].

Класс **AsyncTask** предлагает простой и удобный механизм для перемещения трудоёмких операций в фоновый поток. Благодаря ему обеспечивается удобство синхронизации обработчиков событий с графическим потоком, что позволяет обновлять элементы пользовательского интерфейса для отчета о ходе выполнения задачи или для вывода результатов, когда задача завершена. Напрямую с классом **AsyncTask** работать нельзя, необходимо наследоваться от него [13].

1.2. Формирование запроса.

В клиентском приложении дважды реализовано формирование запроса. Одно для того, чтобы получить список городов и категорий, другое для того, чтобы получить список автошкол. Формирование этих запросов отличается лишь передаваемыми параметрами. Передаваемые параметры зависят от выбора пользователя.

Формирование запроса происходит по следующему алгоритму:

1. Запросы отправятся в фоновом потоке — в наследнике класса **AsyncTask**. Этот наследник содержит три метода, которые необходимо переопределить: **onPreExecute()** (метод, выполняемый перед запуском потока, в нем можно обращаться к UI), **doInBackground()** (главная функция потока, в ней нельзя обращаться к UI), **onPostExecute()** (метод выполнится, только если поток успешно завершится).

2. Записать параметры с помощью `HashMap<String, String>`.
3. Сформировать URL, состоящий из хостинга сервера и передаваемых параметров.
4. Установить соединение с помощью метода (`URLConnection`) `url.openConnection()`.
5. С помощью метода `URLConnection.getResponseCode()` определить, успешно ли был отправлен запрос.
6. В случае успеха получить ответ. Ответом на запрос является то, что возвращает метод `URLConnection.getInputStream()`.

Кроме этого, данные операции заключаются в конструкцию try-catch для того, чтобы программа отлавливала исключения, и чтобы она не выключалась сбоем.

1.3. Обработка ответа на запрос.

Ответ можно получить из входного потока, который имеет тип `InputStream`. Трансляция полученного ответа производится по следующему алгоритму:

1. Создается объект класса `InputStreamReader`, в конструктор которого в качестве аргумента передается входной поток.
2. Создается объект класса `BufferedReader`, в конструктор которого идет объект `InputStreamReader`.
3. Объект `BufferedReader` построчно считывает полученный ответ. Каждая такая строка добавляется к объекту `StringBuffer`.
4. Итоговый объект `StringBuffer` конвертируется в `String`.

Таким образом, происходит обработка ответа на запрос, чтобы затем им легко можно было пользоваться, потому что все текстовые поля принимают текстовые значения именно в типе данных `String`.

Полученный ответ приложение преподносит пользователю в удобном виде. В виде прокручиваемых списков предоставляются список городов и список категорий. В виде маркеров на карте ставятся точки-автошколы.

2. ХРАНЕНИЕ ДАННЫХ

Информационная система хранит различные данные:

- 1) Города России;
- 2) Категории прав на вождение различных транспортных средств, на которые можно обучаться в определенном городе из предыдущего пункта;
- 3) Автошколы, обучающие на определенную категорию прав из предыдущего пункта.
- 4) Подробная информация об автошколах из предыдущего пункта (координата X на карте, координата Y на карте, название, цена за обучение, номера телефонов, адрес, сайт, email).

Для хранения данных используется текстовые файлы вместо базы данных, поскольку нет необходимости постоянно формировать различные запросы к базе данных для выбора и группировки данных, чтобы не тратить на это дополнительное время. В текстовых файлах уже выбраны и сгруппированы все необходимые данные, которые нужно будет отправить пользователю, когда он их запросит. По запросу пользователя необходимо лишь быстро найти требуемый файл.

Итак, данные хранятся в текстовых файлах, которые расположены в директориях таким образом, чтобы их легко можно было найти (Рисунок 3).

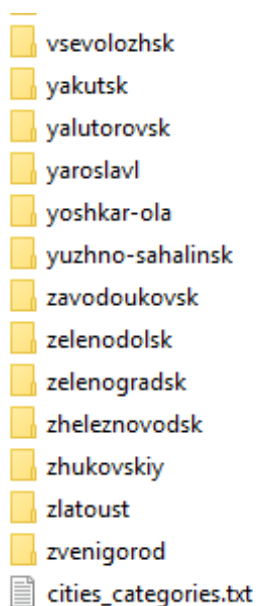
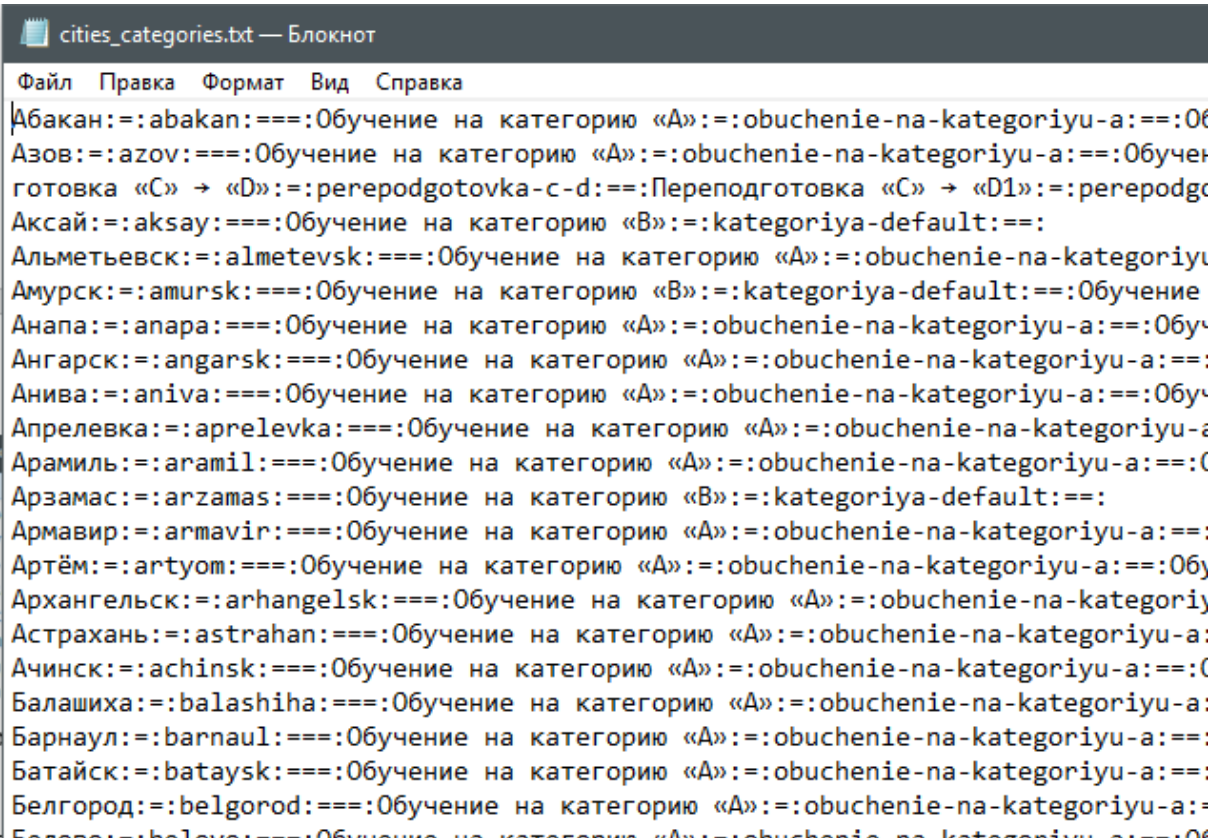


Рисунок 3. Содержимое основной директории с данными

Имеется текстовый файл со списком городов и списком категорий обучения для каждого города (Рисунок 4). В начале каждой строки файла расположены название города на русском языке и в английской раскладке, разделенные между собой сочетанием символов «:=:». После названия города ставятся разделяющие символы «:===:» и перечисляются категории так же: на русском языке и в английской раскладке, разделенные с помощью «:=:». Название города и название категории в английской раскладке необходимо, потому что параметры передаются с помощью английских букв и потому что именно так названы файлы в структуре хранения данных. Категории разделяются между собой с помощью «:=:».

Содержимое этого файла отправляется от сервера к клиенту при запуске Android-приложения пользователем.



```

cities_categories.txt — Блокнот
Файл  Правка  Формат  Вид  Справка
Абакан:=:abakan:===:Обучение на категорию «А»:=:obuchenie-na-kategoriyu-a:=:Обучение
Азов:=:azov:===:Обучение на категорию «А»:=:obuchenie-na-kategoriyu-a:=:Обучение
подготовка «С» → «D»:=:perepodgotovka-c-d:=:Переподготовка «С» → «D1»:=:perepodg
Аксай:=:aksay:===:Обучение на категорию «В»:=:kategoriya-default:=:
Альметьевск:=:almetevsk:===:Обучение на категорию «А»:=:obuchenie-na-kategoriyu
Амурск:=:amursk:===:Обучение на категорию «В»:=:kategoriya-default:=:Обучение
Анапа:=:anapa:===:Обучение на категорию «А»:=:obuchenie-na-kategoriyu-a:=:Обу
Ангарск:=:angarsk:===:Обучение на категорию «А»:=:obuchenie-na-kategoriyu-a:=:
Анива:=:aniva:===:Обучение на категорию «А»:=:obuchenie-na-kategoriyu-a:=:Обу
Апрелевка:=:aprelevka:===:Обучение на категорию «А»:=:obuchenie-na-kategoriyu-a
Арамил:=:aramil:===:Обучение на категорию «А»:=:obuchenie-na-kategoriyu-a:=:(
Арзамас:=:arzasmas:===:Обучение на категорию «В»:=:kategoriya-default:=:
Армавир:=:armavir:===:Обучение на категорию «А»:=:obuchenie-na-kategoriyu-a:=:
Артём:=:artyom:===:Обучение на категорию «А»:=:obuchenie-na-kategoriyu-a:=:Обу
Архангельск:=:arhangelsk:===:Обучение на категорию «А»:=:obuchenie-na-kategoriy
Астрахань:=:astrahan:===:Обучение на категорию «А»:=:obuchenie-na-kategoriyu-a:
Ачинск:=:achinsk:===:Обучение на категорию «А»:=:obuchenie-na-kategoriyu-a:=:(
Балашиха:=:balashiha:===:Обучение на категорию «А»:=:obuchenie-na-kategoriyu-a:
Барнаул:=:barnaul:===:Обучение на категорию «А»:=:obuchenie-na-kategoriyu-a:=:
Батайск:=:bataysk:===:Обучение на категорию «А»:=:obuchenie-na-kategoriyu-a:=:
Белгород:=:belgorod:===:Обучение на категорию «А»:=:obuchenie-na-kategoriyu-a:=
Беловодск:=:belovodsk:===:Обучение на категорию «А»:=:obuchenie-na-kategoriyu-a:=:Обу

```

Рисунок 4. Содержимое файла со списком городов и списком категорий для каждого города

Также есть директории для каждого города (пример содержимого директории можно наблюдать на рисунке 5).

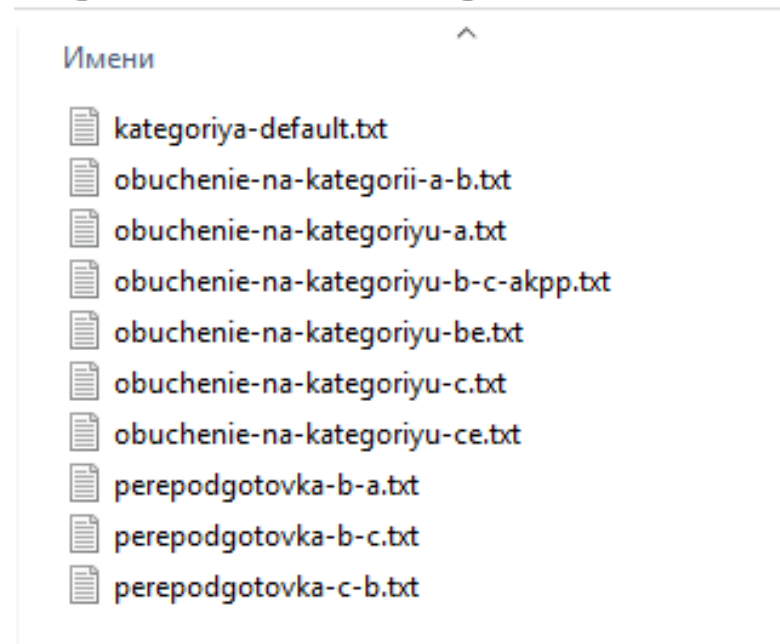


Рисунок 5. Список категорий обучения в Казани

Внутри таких директорий для каждой категории есть текстовый файл (Рисунок 6). В первой строке (выделено на рисунке 7) этот файл содержит информацию о том, где расположить камеру карты, и какой масштаб карты выбрать, если выберется данная категория данного города. Название атрибута (координаты X или Y или масштаб) и его значение разделяются с помощью «:=:», пары атрибут-значение разделяются с помощью «:=:=:».

```

kategoriya-default.txt — Блокнот
Файл  Правка  Формат  Вид  Справка
X_center:==:55.783463:==:Y_center:==:49.140313:==:Zoom:==:12
X:==:55.86454:==:Y:==:49.085923:==:Название:==:Драйв:==:Цена:==:не указано:==
X:==:55.830907:==:Y:==:49.104312:==:Название:==:Центральная автошкола:==:Цена:
X:==:55.827317:==:Y:==:49.059405:==:Название:==:Движение:==:Цена:==:26 000 руб.
X:==:55.728079:==:Y:==:49.173446:==:Название:==:Джек:==:Цена:==:32 000 руб.:==
X:==:55.835296:==:Y:==:49.070427:==:Название:==:КШВВМ:==:Цена:==:не указано:==
X:==:55.824814:==:Y:==:49.120293:==:Название:==:Элита:==:Цена:==:28 000 руб.:==
X:==:55.752683:==:Y:==:49.178567:==:Название:==:КАТТ им. А.П. Обыденнова:==:Це
X:==:55.770238:==:Y:==:49.239356:==:Название:==:Инфинити плюс:==:Цена:==:28 00
X:==:55.861499:==:Y:==:49.107061:==:Название:==:Успех Авто:==:Цена:==:28 000 р
X:==:55.75243:==:Y:==:49.202704:==:Название:==:НУР:==:Цена:==:27 000 руб.:==
X:==:55.790555:==:Y:==:49.169548:==:Название:==:Мастер:==:Цена:==:26 000 руб.:
X:==:55.81997:==:Y:==:49.114795:==:Название:==:Фиеста голд:==:Цена:==:28 900 р
X:==:55.830907:==:Y:==:49.104312:==:Название:==:Центральная автошкола:==:Цена:
X:==:55.83709:==:Y:==:49.11617:==:Название:==:Флагман:==:Цена:==:24 500 руб.:
X:==:55.755616:==:Y:==:49.242042:==:Название:==:Татарстан:==:Цена:==:24 990 ру
X:==:55.792817:==:Y:==:49.168829:==:Название:==:ДОСААФ:==:Цена:==:27 000 руб.:
X:==:55.78854:==:Y:==:49.223707:==:Название:==:ТЦО Казанские автошколы:==:Цена
X:==:55.83206:==:Y:==:49.080911:==:Название:==:Мотор:==:Цена:==:25 000 руб.:
X:==:55.816662:==:Y:==:49.091152:==:Название:==:Ягуарр:==:Цена:==:26 500 руб.:
X:==:55.796785:==:Y:==:49.0977:==:Название:==:Нур-Драйв:==:Цена:==:25 000 руб.
X:==:55.820268:==:Y:==:49.049775:==:Название:==:Грант:==:Цена:==:19 800 руб.:

```

Рисунок 6. Содержимое текстового файла для категории В города Казани

```

kategoriya-default.txt — Блокнот
Файл  Правка  Формат  Вид  Справка
X_center:==:55.783463:==:Y_center:==:49.140313:==:Zoom:==:12
X:==:55.86454:==:Y:==:49.085923:==:Название:==:Драйв:==:Цена
X:==:55.830907:==:Y:==:49.104312:==:Название:==:Центральная а
X:==:55.827317:==:Y:==:49.059405:==:Название:==:Движение:==:
X:==:55.728079:==:Y:==:49.173446:==:Название:==:Джек:==:Цена
X:==:55.835296:==:Y:==:49.070427:==:Название:==:КШВВМ:==:Цен
X:==:55.824814:==:Y:==:49.120293:==:Название:==:Элита:==:Цен
X:==:55.752683:==:Y:==:49.178567:==:Название:==:КАТТ им. А.П.
X:==:55.770238:==:Y:==:49.239356:==:Название:==:Инфинити плюс
X:==:55.861499:==:Y:==:49.107061:==:Название:==:Успех Авто:==
X:==:55.75243:==:Y:==:49.202704:==:Название:==:НУР:==:Цена:
X:==:55.790555:==:Y:==:49.169548:==:Название:==:Мастер:==:Це
X:==:55.81997:==:Y:==:49.114795:==:Название:==:Фиеста голд:==
X:==:55.830907:==:Y:==:49.104312:==:Название:==:Центральная а
X:==:55.83709:==:Y:==:49.11617:==:Название:==:Флагман:==:Цен

```

Рисунок 7. Координаты центра камеры и масштаб карты

Начиная со 2ой строки, построчно в текстовом файле категории записана информация о каждой точке (автошколе) на карте. Про точку-автошколу содержится следующая информация: координата X на карте,

координата Y на карте, название, цена за обучение, номера телефонов, адрес, сайт, email (Выделено на рисунок 8). Название атрибута и его значение разделяются с помощью «:=:», пары атрибут-значение разделяются с помощью «:=:=:». Так как атрибут «Номер(а)» может содержать несколько значений, то предусмотрено делении нескольких значений с помощью «:=:=:».

На рисунке 8 также заметно, что текстовый файл называется «kategoriya-default.txt». Такое особое название получила категория В. Потому что на сайте-источнике для нее не было отдельной директории, соответствующая html-страница открывалась по умолчанию.

```

kategoriya-default.txt — Блокнот
Файл  Правка  Формат  Вид  Справка
X_center:=:55.783463:=:=:Y_center:=:49.140313:=:=:Zoom:=:1
X:=:55.86454:=:=:Y:=:49.085923:=:=:Название:=:Драйв:=:=:Це
X:=:55.830907:=:=:Y:=:49.104312:=:=:Название:=:Центральная
X:=:55.827317:=:=:Y:=:49.059405:=:=:Название:=:Движение:=
X:=:55.728079:=:=:Y:=:49.173446:=:=:Название:=:Джек:=:=:Це
X:=:55.835296:=:=:Y:=:49.070427:=:=:Название:=:КШВВМ:=:=:Ц
X:=:55.824814:=:=:Y:=:49.120293:=:=:Название:=:Элита:=:=:Ц
X:=:55.752683:=:=:Y:=:49.178567:=:=:Название:=:КАТТ им. А.
X:=:55.770238:=:=:Y:=:49.239356:=:=:Название:=:Инфинити пл
X:=:55.861499:=:=:Y:=:49.107061:=:=:Название:=:Успех Авто:
X:=:55.75243:=:=:Y:=:49.202704:=:=:Название:=:НУР:=:=:Цена
X:=:55.790555:=:=:Y:=:49.169548:=:=:Название:=:Мастер:=:=:
X:=:55.81997:=:=:Y:=:49.114795:=:=:Название:=:Фиеста голд:
X:=:55.830907:=:=:Y:=:49.104312:=:=:Название:=:Центральная
  
```

Рисунок 8. Информация о конкретных точках - автошколах

3. СИНТАКСИЧЕСКИЙ АНАЛИЗАТОР, ИЛИ ПАРСЕР

Согласно первой поставленной задачи, прежде всего нужно найти источник таких данных, которые подробно описаны в разделе «Хранение данных». В результате был найден Интернет-ресурс «<http://avtoshkoli.ru/>», где вся информация находится в свободном доступе и разрешена для распространения.

Сначала были определены данные, которые пригодятся в распределенном приложении. Кроме этого, определен оптимальный порядок перехода по html-страницам, чтобы получать эти данные. Основной цикл идет по городам (Рисунок 9). Внутри него цикл, который идет по категориям (Рисунок 10). Еще глубже цикл по автошкалам (рисунок 11). Самый внутренний цикл проходит по имеющейся информации об автошколе.

Крупные города			
Барнаул	Казань	Новосибирск	Санкт-Петербург
Волгоград	Краснодар	Омск	Саратов
Воронеж	Красноярск	Пермь	Тюмень
Екатеринбург	Москва	Ростов-на-Дону	Уфа
Ижевск	Нижний Новгород	Самара	Челябинск
Все города			
А	Е	М	С
Абакан	Егорьевск	Магнитогорск	Салават
Азов	Екатеринбург	Махачкала	Самара
Аксай	Елабуга	Мегион	Санкт-Петербург
Альметьевск	Елизово	Минеральные Воды	Саратов
Амурск	Ессентуки	Минусинск	Светлый
Анапа		Михайловск	Северодвинск
Ангарск	Ж	Можайск	Семилуки
Анива	Железноводск	Москва	Сергиев Посад
Апрелевка	Жуковский	Мурманск	Сертолово

Рисунок 9. Список городов на сайте-источнике

АВТОШКОЛЫ ЕКАТЕРИНБУРГА

СПИСОК АВТОШКОЛ НА КАРТЕ ЦЕНЫ ОТЗЫВЫ

КАТЕГОРИИ ПРАВ

ВЫБРАТЬ МЕТРО

Выберите категорию прав

[Обучение на категорию «А»](#)

[Обучение на категорию «ВЕ»](#)

[Обучение на категории «А» + «В»](#)

[Обучение на категорию «А1»](#)

[Обучение на категорию «С»](#)

[Переподготовка «В» → «С»](#)

[Обучение на категорию «А1»](#)

[Обучение на категорию «СЕ»](#)

[Переподготовка «В» → «D»](#)

[Обучение на категорию «В»](#)

[Обучение на категорию «D»](#)

[Переподготовка «С» → «В»](#)

[Обучение на категорию «В» с АКПП](#)

[Обучение на категорию «DE»](#)

[Переподготовка «С» → «D»](#)

[Обучение на категорию «В1»](#)

[Обучение на категорию «М»](#)

[Переподготовка «D» → «В»](#)

Рисунок 10. Список категорий на сайте-источнике

Автошкола ДОСААФ Категория "В": 20 000 руб.	г. Екатеринбург, ул. Малышева 31 Д, офис 301, этаж 3	+7 (343) 371-16-62 +7 (922) 024-14-57
Автошкола ДОСААФ ЕАШ Категория "В": 20 000 руб.	г. Екатеринбург, ул. Малышева 31 Д, офис 114	+7 (343) 371-05-71 +7 (343) 371-06-53
Автошкола Свердловская автошкола Категория "В": 21 000 руб.	г. Екатеринбург, ул. Сибирский тракт, 8 Б, офис 428	+7 (343) 288-54-24 +7 (904) 162-38-61
Автошкола Курсант Категория "В": 19 000 руб.	г. Екатеринбург, ул. Хохрякова, 98, офис 219	+7 (343) 216-35-62 +7 (343) 201-33-01

Рисунок 11. Список автошкол на сайте-источнике

Перед написанием парсера также был выбран способ, как будут храниться данные, чтобы потом их можно было легко получить. Подробнее о хранении данных описано в предыдущем разделе.

Суть парсера заключается в том, что он из языка разметки (справа на рисунке 12) получает только данные (слева на рисунке 12) для дальнейшего использования информационной системой.

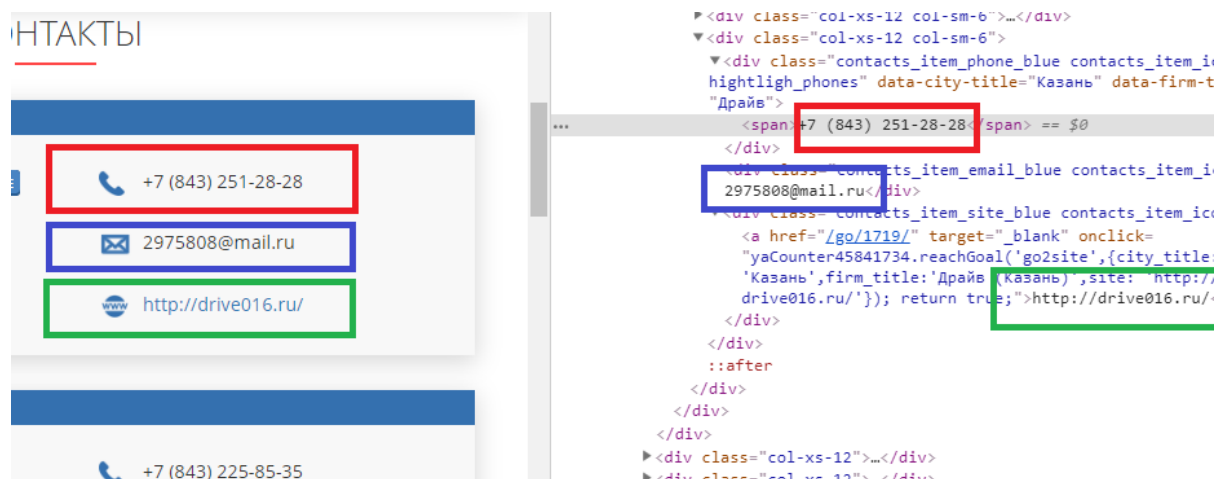


Рисунок 12. Соответствие данных и их расположения в коде разметки

Для написания синтаксического анализатора на языке программирования C# была подробно изучена библиотека «HtmlAgilityPack». Эта библиотека загружается в Microsoft Visual Studio IDE через Nuget.

Недостатком данной библиотеки являлось то, что html-страницы необходимо было загружать в коде. То есть проблема состояла в том, что сайт необходимо было сначала скачать. Для скачивания сайта была скачана и установлена программа «HTTrack Website Copier». Скачивание сайта заняло порядка 11 минут. Скачанный сайт представляет собой директорию и расположенные в ней html-страницы.

Парсер ходит по папкам и находит в скачанном сайте необходимые ему html-файлы. Программа запоминает пути к html-файлам, потому что в папке есть html-файл и другие папки с другими html-файлами (города, категории внутри папки города, автошколы внутри папки категорий). Синтаксический анализатор берет названия городов, узнает, какие есть категории в определенном городе. Программа создает свои директории для хранения

данных и записывает внутри них в текстовые файлы то, что считывает. Директории и текстовые файлы создаются согласно разработанной ранее структуре хранения данных.

При анализе каждой html-страницы считывается содержимое html-страницы с помощью метода `File.ReadAllText`, затем создается документ `HtmlAgilityPack.HtmlDocument`, в него загружается содержимое html-файла с помощью метода `HtmlAgilityPack.HtmlDocument.LoadHtml`. Из `HtmlDocument` парсер получает коллекцию узлов, или тэгов, `HtmlNodeCollection`. Переходы по узлам происходят аналогично переходам в дереве: от родительского узла к дочернему, от одного узла соседнему и так далее. От каждого узла можно получить атрибуты, которые являются атрибутами тэга. В `HtmlNode.InnerText` содержится текст данного узла. Именно таким образом парсер получает всю необходимую информацию.

Часть информации о каждой автошколе хранится в коде JavaScript, который записан внутри html-страницы (Рисунок 13).

```
myPlacemark1 = new
ymaps.Placemark([56.870494,60.520938], {
  balloonContent: '<div class="ymap_balloon">
<b>Автошкола &laquo;Экстрим&raquo;</b>
<i>Головной офис</i><i>г. Екатеринбург, ул.
Техническая, 67</i><i>+7 (343) 346-31-00<br
/>+7 (950) 654-90-34</i><br/><a
href="/ekaterinburg/avtoshkola-ekstrim/"
class="mybtn mybtn_blue mybtn_mini">
<span>Подробнее</span></a></div>',
  {iconImageHref:
'/images/map_point_red.png',iconImageSize:
[35, 47],iconImageOffset: [-17, -47]}});

myMap.geoObjects.add(myPlacemark1);
```

Рисунок 13. Фрагмент кода JavaScript, в котором содержится информация об автошколе

Эту информацию получить сложнее, нежели информацию из тэгов. Код JavaScript рассматривалась как одна строка, в парсере записан универсальный код для работы с такими строками, который работает правильно со всеми точками и берет всю информацию о них, содержащуюся в коде JavaScript. Поскольку работа с кодом JavaScript происходит в самом внутреннем цикле и переменная строки постоянно меняется, было решено использовать

переменные типа данных не string, а StringBuilder, так как второй тип данных является изменяемым. Это значительно экономит память и время работы парсера.

Парсер выполняет свою основную работу в отдельном потоке, чтобы компьютер не зависал во время парсинга. Также на сайте существует дублирование крупных городов. Например, город Санкт-Петербург (Рисунок 14). В коде предусмотрен этот случай, чтобы парсить каждый город не более одного раза.

Крупные города			
Барнаул	Казань	Новосибирск	Санкт-Петербург
Волгоград	Краснодар	Омск	Саратов
Воронеж	Красноярск	Пермь	Тюмень
Екатеринбург	Москва	Ростов-на-Дону	Уфа
Ижевск	Нижний Новгород	Самара	Челябинск
Все города			
А	Е	М	С
Абакан	Егорьевск	Магнитогорск	Салават
Азов	Екатеринбург	Махачкала	Самара
Аксай	Елабуга	Мегион	Санкт-Петербург
Альметьевск	Елизovo	Минеральные Воды	Саратов
Амурск	Ессентуки	Минусинск	Светлый

Рисунок 14. Пример дублирования крупного города

В парсере предусмотрено удаление старых и добавление новых данных с помощью проверки существования файла, с помощью новой записи файл или с помощью дописывания в конец файла. Синтаксический анализатор реализован в удобном интерфейсе, благодаря которому легко определить, на какой итерации он находится в определенный момент. Такой интерфейс сильно ускорял поиск ошибок во время разработки синтаксического анализатора. Интерфейс парсера представлен на рисунке 15.

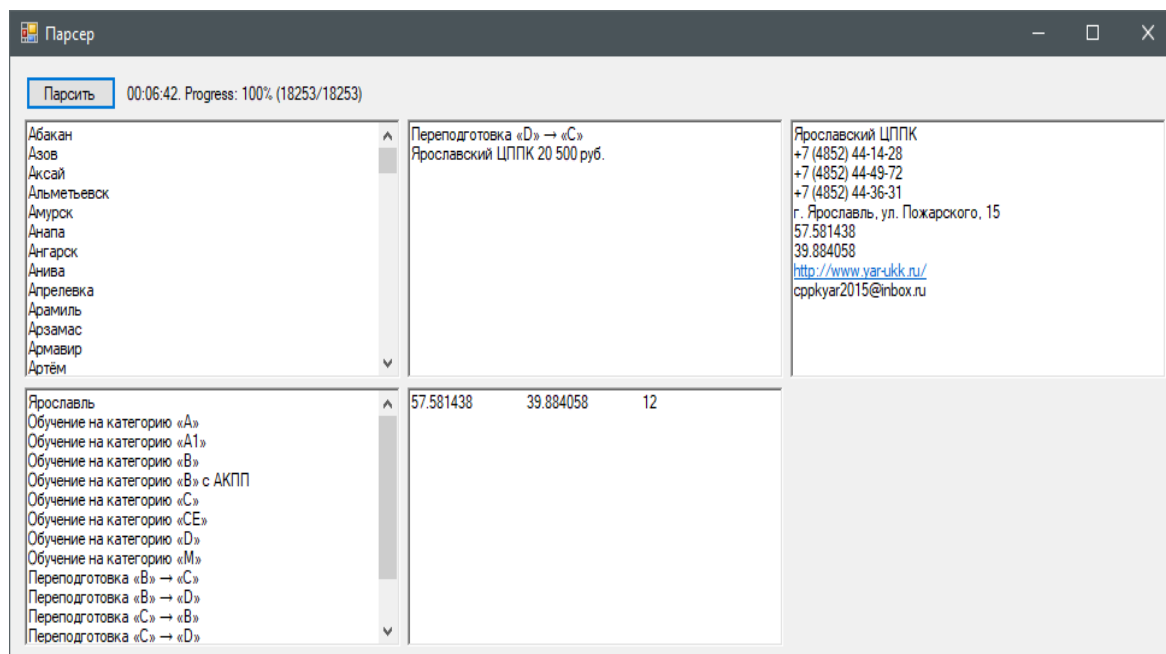


Рисунок 15. Интерфейс синтаксического анализатора

В первом окне представлен список городов, прошедших парсинг, во втором окне – список категорий определенного города, в третьем – список автошкол данной категории, в четвертом окне – координаты центра камеры и её масштаб, в пятом – вся информация про автошколу.

Весь парсинг сайта занимает порядка 7 минут.

4. СЕРВЕРНАЯ ЧАСТЬ ПРИЛОЖЕНИЯ

Как правило, пользователю достаточно лишь информация об автошколах одного города. То есть нет смысла хранить все данные внутри Android-приложения в виде ресурсов. Поэтому было принято решение создать распределенное приложение, в котором вся информация хранится на сервере.

Таким образом, серверная часть информационной системы состоит из логической части и информационной части.

Логическая серверная часть написана на языке программирования PHP, и она выступает посредником между клиентом и информационной частью системы.

Сначала она получает первый Get-запрос и посылает клиенту список городов и список категорий для каждого города. Затем получает второй Get-запрос с параметрами city (город, который выберет пользователь) и category (категория, которую выберет пользователь) и отвечает на этот запрос списком всех точек этой категории этого города. Второй ответ на запрос содержит подробную информацию про расположение камеры, про масштаб карты, данные про каждую точку: координата X на карте, координата Y на карте, название, цена за обучение, номера телефонов, адрес, сайт, email.

Тестировать серверное приложение можно при помощи любого браузера. Для этого необходимо написать Get-запрос в адресную строку (Рисунок 16).

Абакан:=:abakan:===:Обучение на категорию «А»:=:obuchenie-na-kategoriyu-a:===:Обучение
 default:=:Обучение на категорию «В» с АКПП:=:obuchenie-na-kategoriyu-b-c-akpp:===:Обуч
 kategoriyu-c:=:Обучение на категорию «СЕ»:=:obuchenie-na-kategoriyu-ce:=:Обучение на
 c:=:Переподготовка «В» → «D»:=:perepodgotovka-b-d:=:Переподготовка «С» → «D»:=:pe
 a:=:Обучение на категорию «А1»:=:obuchenie-na-kategoriyu-a1:=:Обучение на категорию
 категорию «ВЕ»:=:obuchenie-na-kategoriyu-be:=:Обучение на категорию «С»:=:obuchenie-na
 «СЕ»:=:obuchenie-na-kategoriyu-ce:=:Обучение на категорию «С1Е»:=:obuchenie-na-kategor
 «D1»:=:obuchenie-na-kategoriyu-d1:=:Обучение на категорию «ДЕ»:=:obuchenie-na-kategori
 «С»:=:perepodgotovka-b-c:=:Переподготовка «В» → «С1»:=:perepodgotovka-b-c1:=:Переп
 d1:=:Переподготовка «С» → «В»:=:perepodgotovka-c-b:=:Переподготовка «С» → «D»:=:pe
 «В»:=:perepodgotovka-d-b:=:Переподготовка «D» → «С»:=:perepodgotovka-d-c:=:Перепод
 «В»:=:kategoriya-default:=: Альметьевск:=:almetevsk:===:Обучение на категорию «А»:=:ob
 «С»:=:obuchenie-na-kategoriyu-c:=:Обучение на категорию «СЕ»:=:obuchenie-na-kategoriyu-
 «В»:=:perepodgotovka-c-b:=:Переподготовка «С» → «D»:=:perepodgotovka-c-d:=: Амурск:
 АКПП:=:obuchenie-na-kategoriyu-b-c-akpp:===: Анапа:=:anapa:===:Обучение на категорию «А
 a1:=:Обучение на категорию «В»:=:kategoriya-default:=: Ангарск:=:angarsk:===:Обучение
 kategoriyu-a1:=:Обучение на категорию «В»:=:kategoriya-default:=:Обучение на категорию
 c:=:Обучение на категорию «СЕ»:=:obuchenie-na-kategoriyu-ce:=:Обучение на категорию
 c:=:Переподготовка «В» → «D»:=:perepodgotovka-b-d:=:Переподготовка «С» → «В»:=:pe
 на категорию «А»:=:obuchenie-na-kategoriyu-a:=:Обучение на категорию «В»:=:kategoriya-c
 категорию «С»:=:obuchenie-na-kategoriyu-c:=: Апрелевка:=:aprelevka:===:Обучение на кате
 Арамил:=:aramil:===:Обучение на категорию «А»:=:obuchenie-na-kategoriyu-a:=:Обучени
 default:=:Обучение на категорию «СЕ»:=:obuchenie-na-kategoriyu-ce:=:Обучение на катег
 c:=:Переподготовка «С» → «В»:=:perepodgotovka-c-b:=:Переподготовка «С» → «D»:=:pe
 Армавир:=:armavir:===:Обучение на категорию «А»:=:obuchenie-na-kategoriyu-a:=:Обучени
 kategoriyu-b-c-akpp:=:Обучение на категорию «С»:=:obuchenie-na-kategoriyu-c:=:Обучени
 c:=:Переподготовка «В» → «D»:=:perepodgotovka-b-d:=:Переподготовка «С» → «В»:=:pe
 Артём:=:artyom:===:Обучение на категорию «А»:=:obuchenie-na-kategoriyu-a:=:Обучение
 c:=:Обучение на категорию «СЕ»:=:obuchenie-na-kategoriyu-ce:=:Обучение на категорию «
 «А»:=:obuchenie-na-kategoriyu-a:=:Обучение на категорию «А1»:=:obuchenie-na-kategoriyu-
 АКПП:=:obuchenie-na-kategoriyu-b-c-akpp:===:Обучение на категорию «ВЕ»:=:obuchenie-na-l
 «СЕ»:=:obuchenie-na-kategoriyu-ce:=:Обучение на категорию «D»:=:obuchenie-na-kategoriyu
 «М»:=:obuchenie-na-kategoriyu-m:=:Обучение на категории «В» + «С»:=:obuchenie-na-kateg
 «D»:=:perepodgotovka-b-d:=:Переподготовка «С» → «В»:=:perepodgotovka-c-b:=:Перепод

Рисунок 16. Тестирование серверной части приложения

Все данные хранятся в одной папке с php-файлом.

5. КЛИЕНТСКАЯ ЧАСТЬ ПРИЛОЖЕНИЯ

Клиентское Android-приложение написано на языке программирования Java. Оно состоит из нескольких Activity для удобства пользователя.

5.1. Activity с выбором города

При запуске Android-приложения открывается Activity для выбора города. Изначально списка городов в Activity нет. В фоновом потоке при помощи класса AsyncTask отправляется Get-запрос к серверной части приложения. Пока клиент ожидает ответ (при хорошем соединении это время составляет доли секунды, но иногда это время может затянуться на несколько секунд), приложение показывает свое состояние. Надпись: «Получение списка городов...» намекает пользователю, что нужно немного подождать (Рисунок 17).

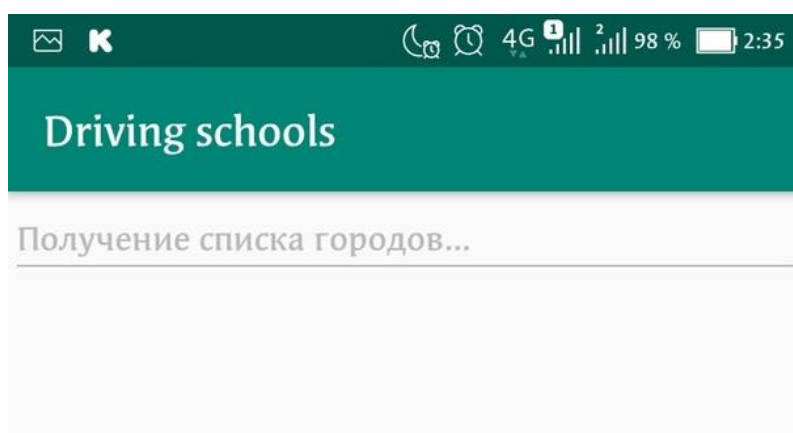


Рисунок 17. Ожидание от сервера ответа со списком городов

После получения ответа от сервера, этот ответ необходимо обработать. Полученную информацию необходимо представить в удобном виде для пользователя. Формируется список `ArrayList<String>` городов (Рисунок 18)

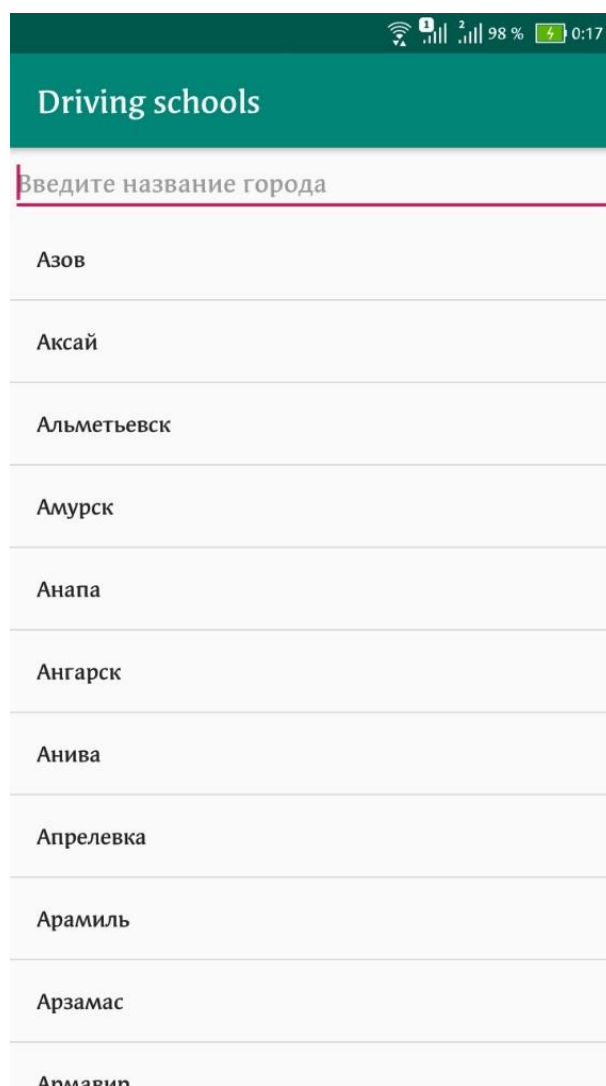


Рисунок 18. Activity со списком городов

Чтобы пользователю проще было найти интересующий его город, реализован умный поиск. Пользователю достаточно ввести часть названия города любым регистром, и число городов сократится, останутся лишь те, которые удовлетворяют критерию поиска (Рисунок 19)

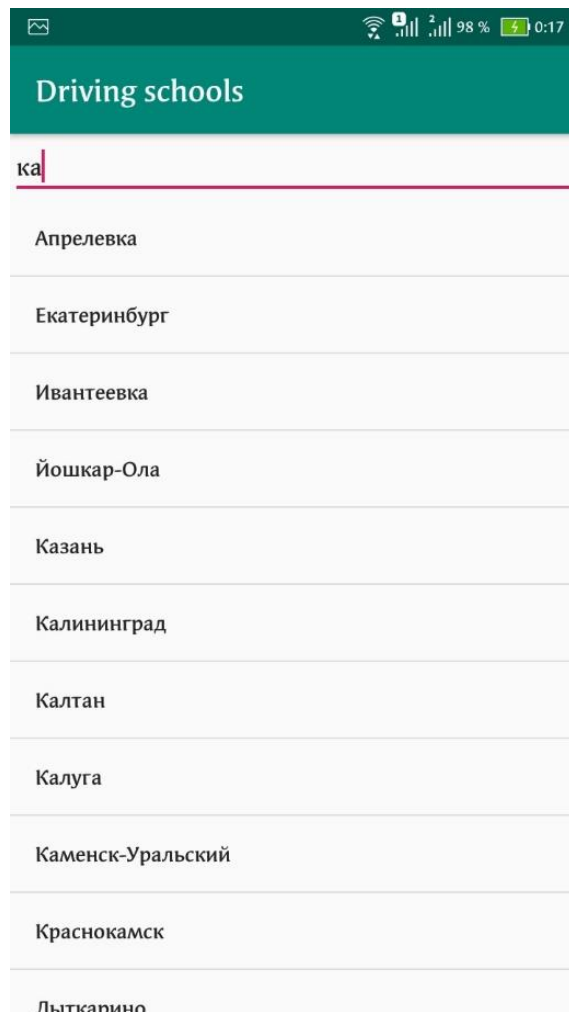


Рисунок 19. Поиск города

Когда пользователь нажимает на элемент списка, он попадает в Activity с выбором категории.

5.2. Activity с выбором категории

Activity содержит название выбранного города (выбранный город из предыдущего Activity передается с помощью параметров Intent) и список категорий. В зависимости от выбранного города в данном Activity могут быть различные вариации возможных категорий (Рисунок 20).

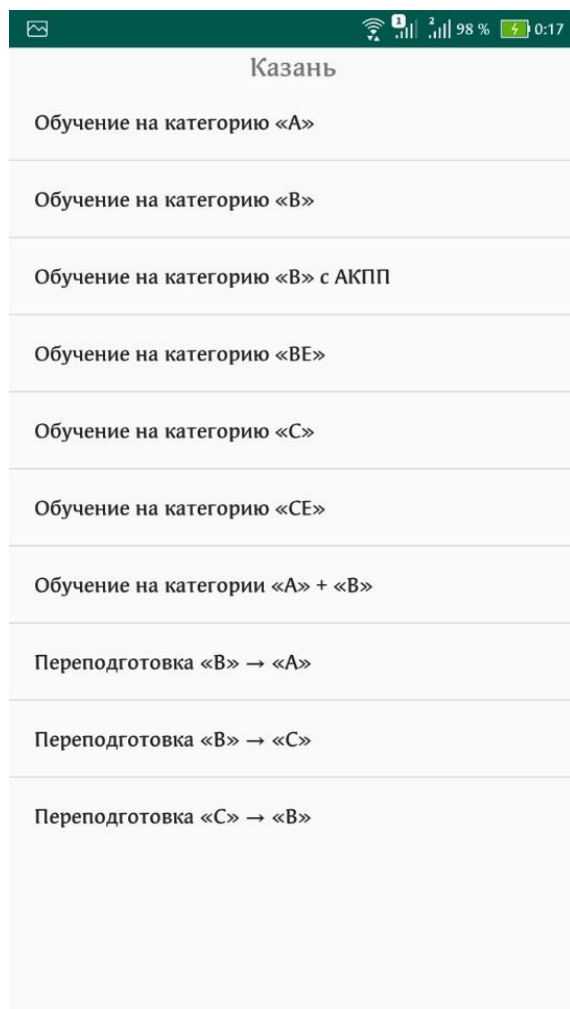


Рисунок 20. Activity с выбором категории

После нажатия на элемент списка. Пользователь переходит в Activity с Google-картой.

5.3. Activity с Google-картой

На этом этапе пользователь уже выбирал город и категорию, которые идут в параметры второго Get-запроса, который отправляется на сервер. В этот момент приложение снова показывает свое состояние «Получение списка автошкол...» (Рисунок 21).

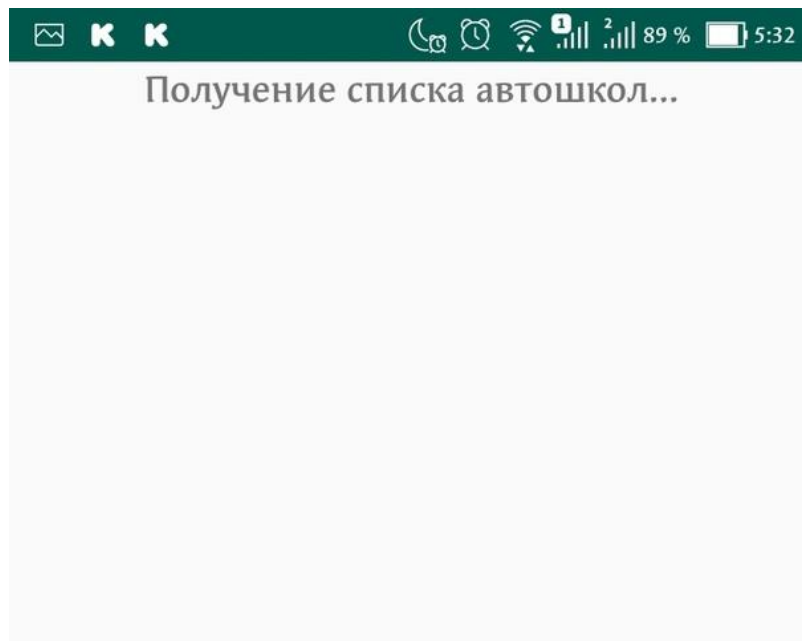


Рисунок 21. Ожидание от сервера ответа со списком автошкол

Затем клиентское приложение в виде ответа на запрос получает информацию про все точки выбранной категории выбранного города.

Для работы с Google-картами изначально был получен ключ API. Этот ключ добавляется в ресурсы таким образом, как показано на рисунке 22 (Ключ обрезан умышленно).

```
once you have your key (it starts with "AIza"), replace the "google_maps_key"
string in this file.
-->
<string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">AIzaSyBn
/resources>
```

Рисунок 22. Ключ API

Так же необходимо прописать строки в файле AndroidManifest.xml, как показано на рисунке 23.

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="@string/google_maps_key" />
```

Рисунок 23. Запись ключа API в манифест

Чтобы карта отобразилась в Activity, необходимо в layout-файле прописать необходимые строки (Рисунок 24).

```

<fragment
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_weight="1">
</fragment>

```

Рисунок 24. Описание карты в layout-файле

Загружается карта выбранного города с удобным масштабом. На карте расставлены точки, или маркеры, соответствующие автошколам выбранной категории обучения (Рисунок 25).



Рисунок 25. Activity с Google-картой

Отдельное внимание уделено маркерам. Каждый из них строится динамически и представляет собой полупрозрачную табличку с названием автошколы и с ценой за обучение на выбранную категорию прав вождения.

Место на карте с большой концентрацией автошкол можно увеличить, чтобы проще было разобраться в ценах и местонахождениях автошкол.

5.4. Инфо-окна маркеров

Чтобы получить больше информации про автошколу, пользователь может нажать на маркер. Появится инфо-окно, которое так же, как и маркер, создается динамически, с подробной информацией: с номерами телефонов, с точным адресом, с сайтом, с email (Рисунок 26).

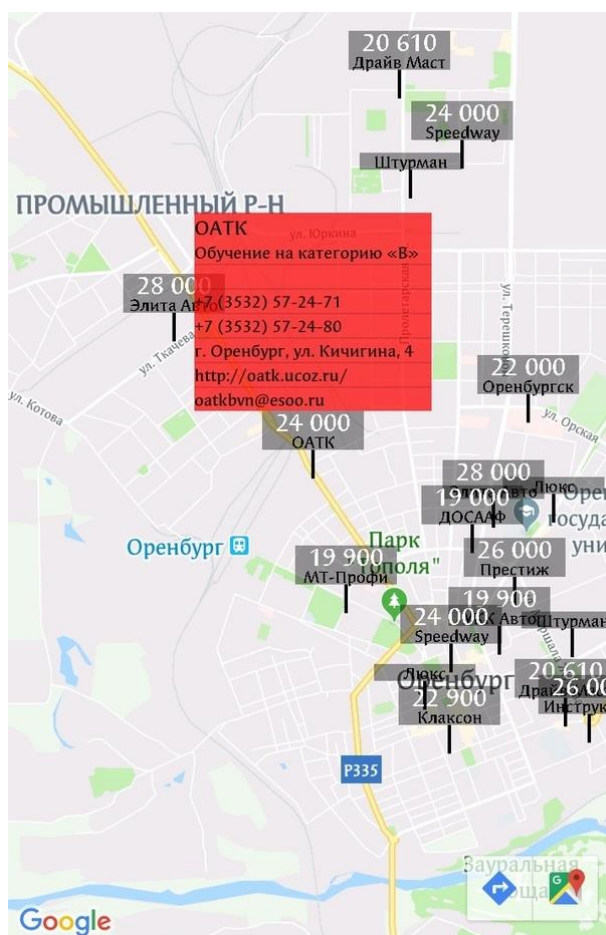


Рисунок 26. Инфо-окно с подробной информацией об автошколе

Составные части для создания такого инфо-окна:

- 1) Инфо-окно создается полностью с помощью метода `GoogleMap.setInfoWindowAdapter()`.
- 2) Этот метод получает аргумент — объект класса `GoogleMap.InfoWindowAdapter`, в котором переопределен метод `getInfoWindow()`, возвращающий `View` — созданное инфо-окно.

3) Инфо-окно состоит из текстового поля TextView (куда записывается название автошколы) и из списка ListView (где каждый элемент – часть информации об автошколе).

4) Структура инфо-окна задается с помощью xml-файла (Рисунок 27).

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:background="#BBFF0000">
    <TextView
        android:id="@+id/tv"
        android:layout_width="140dp"
        android:layout_height="wrap_content"
        android:textSize="14sp"
        android:textColor="#000"/>
    <ListView
        android:id="@+id/lv"
        android:layout_height="wrap_content"
        android:layout_width="140dp">
    </ListView>
</LinearLayout>
```

Рисунок 27. Структура инфо-окна

5) Каждый элемент списка формируется на основе написанного xml-файла (Рисунок 28).

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:gravity="center_horizontal"
    android:textColor="#000"
    android:textSize="12sp">
</TextView>
```

Рисунок 28. Структура элемента списка

ЗАКЛЮЧЕНИЕ

В ходе курсовой работы было реализовано распределенное приложение «Автошколы России» с мобильным клиентом на платформе Android.

- 1) Сначала был найден источник данных.
- 2) Затем был разработан синтаксический анализатор, который все необходимые данные из источника сохранил в удобной форме для дальнейшей работы.
- 3) Были исследованы маркеры и инфо-окна Google-карт.
- 4) Была разработана серверная часть приложения, которая отправляет клиенту запрошенные данные.
- 5) Было реализовано Android-приложение, которое получает информацию с сервера, дает возможность пользователю выбрать город и категорию прав, предоставляет карту с расставленными на ней маркерами – автошколами, также выдает подробную информацию о интересующей автошколе.

Тестирование проводилось на реальном Android-устройстве.

Таким образом, поставленные задачи успешно выполнены.

Приложение «Автошколы России» будет полезно любому человеку, захотевшему пройти обучение перед получением прав на вождение, который еще не определился с автошколой.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

- 1) «АКТУАЛЬНОСТЬ АВТОШКОЛ» // портал «driver-news». – URL: http://driver-news.ru/poleznie_soveti/10150 [Дата обращения: 10.10.2018]
- 2) «Андроид – наиболее популярная система для смартфонов»// портал «v-androide». – URL: <http://v-androide.com/system/harakteristyky/android-samaya-populyarnaya-sistema-dlya-smartfonov.html> [Дата обращения: 20.11.2018]
- 3) «Activity (Активность, Деятельность)». – URL: <http://developer.alexanderklimov.ru/android/theory/activity-theory.php> [Дата обращения: 22.09.2018]
- 4) «Объекты Intent и фильтры объектов Intent» // портал «developer.android». – URL: <https://developer.android.com/guide/components/intents-filters?hl=ru> [Дата обращения: 02.11.2018]
- 5) «Карты Google» // портал «Википедия». – URL: https://ru.wikipedia.org/wiki/Карты_Google [Дата обращения: 20.01.2019]
- 6) «Info Windows» // портал «developer.android». – URL: <https://developers.google.com/maps/documentation/javascript/infowindows> [Дата обращения: 12.02.2019]
- 7) «Распределённые приложения. Архитектура клиент-сервер» // портал «studfiles». – URL: <https://studfiles.net/preview/5911507/> [Дата обращения: 17.12.2018]
- 8) «Post и Get запросы, какая между ними разница и что лучше и для каких целей?» // портал «toster». – URL: <https://toster.ru/q/169915> [Дата обращения: 25.04.2019]
- 9) «Синтаксический анализатор» // портал «Википедия». – URL: https://ru.wikipedia.org/wiki/Синтаксический_анализатор [Дата обращения: 14.04.2019]

10) «Microsoft Visual Studio» // портал «Википедия». – URL: https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio [Дата обращения: 05.09.2018]

11) «Основы JavaScript» // портал «developer.mozilla». – URL: https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/JavaScript_basics [Дата обращения: 27.03.2019]

12) «Интерфейс — что это такое? Определение, значение, перевод» // портал «что-это-такое». – URL: <https://что-это-такое.ru/interface> [Дата обращения: 10.11.2018]

13) «Класс AsyncTask». – URL: <http://developer.alexanderklimov.ru/android/theory/asynctask.php> [Дата обращения: 22.04.2019]

ПРИЛОЖЕНИЯ

1. Код парсера

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Text;
using System.Windows.Forms;
using HtmlAgilityPack;
using System.Threading;

namespace Парсер
{
    public partial class Form1 : Form
    {
        // Открытая папка (в ней папки городов и index.html)
        string path_main = "Сайт avtoshkoli//Автошколы//avtoshkoli.ru";
        string path_for_write =
"driving_schools//DateBase_Driving_Schools";
        string delitel_3 = "===:";
        string delitel_2 = "==:";
        string delitel_1 = "=: ";
        int progress_current = 0;
        int progress_max = 18253; // это количество мелких итераций
        int number_point = 0;
        String time_str;
        int hours = 0, min = 0, sec = 0;
        public Form1()
        {
            InitializeComponent();
        }
        private void Parcing()
        {
            btn_parcng.Enabled = false;
            richTextBox1.Text = "";
            richTextBox2.Text = "";
            richTextBox3.Text = "";
            richTextBox4.Text = "";
            richTextBox5.Text = "";
            hours = 0; min = 0; sec = 0;
            label_time.Text = "00:00:00";
            // City->Category...
            if (!Directory.Exists(path_for_write))
            { // Создаем корневую папку
                Directory.CreateDirectory(path_for_write);
            }
            else
            {
                //Удаляем все содержимое
                DirectoryInfo di = new DirectoryInfo(path_for_write);
                foreach (FileInfo file in di.GetFiles())
                    file.Delete();
                foreach (DirectoryInfo dir in di.GetDirectories())
                    dir.Delete(true);
            }
            HtmlAgilityPack.HtmlDocument document_list_city = new
            HtmlAgilityPack.HtmlDocument();
            string text_html_list_city = File.ReadAllText(path_main +
            "//index.html");
            document_list_city.LoadHtml(text_html_list_city);
        }
    }
}
```

```

        HtmlNodeCollection links_city =
document_list_city.DocumentNode.SelectNodes("/html/body/div/div/div/div/div/div/div/div/ul/li");
        foreach (HtmlNode li in links_city)
        {
            if (li.GetAttributeValue("class", "-class-") ==
"cities_list_item" && li.ParentNode.ParentNode.GetAttributeValue("class", "-
class-") == "col-xs-12 col-sm-3 cities_list_items")
            {
                string city = li.InnerText;

                StringBuilder path_city = new
StringBuilder(li.ChildNodes.FindFirst("a").GetAttributeValue("href", "-href-
"));

                path_city.Replace("/", "\\");
                path_city.Insert(0, "\\");

                string path_city_without_index =
path_city.ToString().Replace("\\index.html", "");
                string city_en =
path_city_without_index.Replace("\\", "");
                //Запись города в файл:
                using (StreamWriter sw = new
StreamWriter(path_for_write + "\\cities_categories.txt", true,
Encoding.UTF8))
                {
                    sw.Write(city + delitel_1 + city_en +
delitel_3);

                    sw.Close();
                }

                if (!Directory.Exists(path_for_write +
path_city_without_index))
                { // Создаем папку-город
                    Directory.CreateDirectory(path_for_write +
path_city_without_index);
                }

                Categories_from_city(city, path_city,
path_city_without_index);
            }
        }
        timer_sec.Enabled = false;
        tick();
        btn_parsing.Enabled = true;
    }

    private void Categories_from_city(string city, StringBuilder
path_city, string path_city_without_index)
    {
        // Category -> Avtoskola
        richTextBox1.Text += city + "\n";
        richTextBox2.Text = city + "\n";
        HtmlAgilityPack.HtmlDocument document_list_category = new
HtmlAgilityPack.HtmlDocument();
        string text_html_list_category = File.ReadAllText(path_main
+ path_city);

        document_list_category.LoadHtml(text_html_list_category);
        HtmlNodeCollection links_categorys =
document_list_category.DocumentNode.SelectNodes("/html/body/div/div/div/div/div/div/div/div/ul/li/a");
        if (links_categorys != null)
        {
            foreach (HtmlNode a in links_categorys)

```

```

        {
            if
(a.ParentNode.ParentNode.ParentNode.GetAttributeValue("class", "-class-") ==
"col-xs-12 col-sm-4")
            {
                StringBuilder path_category = new
StringBuilder(path_city.ToString());
                string category_en =
a.GetAttributeValue("href", "-href-");
                path_category.Replace("index.html",
category_en);
                path_category.Replace("/", "\\");
                string category = a.InnerText;
                // Дописываем категорию в файл городов-
категорий
                using (StreamWriter sw = new
StreamWriter(path_for_write + "\\cities_categories.txt", true,
Encoding.UTF8))
                {
                    category_en =
category_en.Replace("/index.html", "");
                    if(category_en == "index.html")
                        category_en = "kategoriya-
default";
                    sw.Write(category + delitel_1 +
category_en + delitel_2);
                    sw.Close();
                }
                Avtoskoli_from_Category(category,
path_category, path_city, path_city_without_index, category_en);
            }
        }
    else
    {
        // Если нет выбора категорий...
        // Дописываем категорию в файл городов-категорий
        string category = "Категория...";
        string category_en = "kategoriya-default";
        using (StreamWriter sw = new
StreamWriter(path_for_write + "\\cities_categories.txt", true,
Encoding.UTF8))
        {
            sw.Write(category + delitel_1 + "kategoriya-
default" + delitel_2);
            sw.Close();
        }
        Avtoskoli_from_Category(category, path_city,
path_city, path_city_without_index, category_en);
    }

    // Переход на новую строку для записи нового города
    using (StreamWriter sw = new StreamWriter(path_for_write +
"\\cities_categories.txt", true, Encoding.UTF8))
    {
        sw.WriteLine();
        sw.Close();
    }
}

private void Avtoskoli_from_Category(string category,
StringBuilder path_category, StringBuilder path_city, string
path_city_without_index, string category_en)

```

```

        {
            richTextBox2.Text += category + "\n";
            richTextBox3.Text = category + "\n";
            string path_category_without_index =
path_category.ToString().Replace("\\\\index.html", "");

            if (path_category_without_index == path_city_without_index)
            { //Для категории "В" создаем папку сами (т.к. она по
умолчанию)
                path_category_without_index += "\\\" + category_en;
            }
            Dictionary<string, StringBuilder> dict_avtoshkola_path = new
Dictionary<string, StringBuilder>();
            Dictionary<string, string> dict_avtoshkola_price = new
Dictionary<string, string>();
            Dictionary<string, string> dict_avtoshkola_site = new
Dictionary<string, string>();
            Dictionary<string, string> dict_avtoshkola_email = new
Dictionary<string, string>();

            HtmlAgilityPack.HtmlDocument document_list_avtoshkoli = new
HtmlAgilityPack.HtmlDocument();
            string text_html_list_avtoshkoli =
File.ReadAllText(path_main + path_category);

            document_list_avtoshkoli.LoadHtml(text_html_list_avtoshkoli);

            // Взять центральную точку карты отсюда.
            HtmlNodeCollection link_koord =
document_list_avtoshkoli.DocumentNode.SelectNodes("/html/body/div/script/text
()");

            string kod_js = link_koord [1].InnerText;

            string koord_X_and_Y_c =
kod_js.Substring(kod_js.IndexOf('[') + 1, kod_js.IndexOf(']') -
kod_js.IndexOf('[') - 1);
            string [ ] koord_c = koord_X_and_Y_c.Split(new char [ ] {
',' });

            string zoom = kod_js.Substring(kod_js.IndexOf("zoom:") + 5,
kod_js.IndexOf("// масштаб") - kod_js.IndexOf("zoom:") - 5).Trim();

            // Записываем центр карты и масштаб
            using (StreamWriter sw = new StreamWriter(path_for_write +
path_city_without_index + "\\\" + category_en + ".txt", false,
Encoding.UTF8))
            {
                sw.WriteLine("X_center" + delitel_2 + koord_c [0] +
delitel_3 + "Y_center" + delitel_2 + koord_c [1] + delitel_3 + "Zoom" +
delitel_2 + zoom);
                sw.Close();
            }

            richTextBox4.Text = koord_c [0] + "\t" + koord_c [1] + "\t"
+ zoom + "\n";

            HtmlNodeCollection links_categorys =
document_list_avtoshkoli.DocumentNode.SelectNodes("/html/body/div/div/div/div
/div/ul/li/div/a/span/span/span/b");
            foreach(HtmlNode b in links_categorys)
            {
                string avtoskola_name = b.InnerText;
                if (!dict_avtoshkola_path.ContainsKey(avtoskola_name))

```

```

        {
            StringBuilder path_avtoshkoli = new
StringBuilder(path_city.ToString());
            path_avtoshkoli.Replace("index.html",
b.ParentNode.ParentNode.ParentNode.ParentNode.GetAttributeValue("h
ref", "-href-"));
            path_avtoshkoli.Replace("/", "\\");
            path_avtoshkoli.Replace("..\\", "");

            dict_avtoshkola_path.Add(avtoshkola_name,
path_avtoshkoli);

            HtmlNode i_u = b.ParentNode.ChildNodes
[5].ChildNodes.FindFirst("u");
            string price = i_u.InnerText;
            dict_avtoshkola_price.Add(avtoshkola_name,
price);

            richTextBox3.Text += avtoshkola_name + " " +
price + "\n";

            // Одновременно взять сайт и email...
            Info_from_Avtoshkola(avtoshkola_name,
path_avtoshkoli, dict_avtoshkola_site, dict_avtoshkola_email);
            progress_current++;
        }

    }
    #region По карте
    int left = kod_js.IndexOf("// масштаб");
    int right = kod_js.IndexOf("d(myPlacemark)");
    // left и right в коде js - границы текущей рассматриваемой
точки
    while (left != -1 || right != -1) // == -1 если не найден в
строке
    {
        int count = right - left;
        string koord_X_and_Y =
kod_js.Substring(kod_js.IndexOf('[', left, count) + 1, kod_js.IndexOf(']',
left, count) - kod_js.IndexOf('[', left, count) - 1);
        string [] koord = koord_X_and_Y.Split(new char [ ] {
',' });

        string avtoshkola_name_k = new
StringBuilder(kod_js.Substring(kod_js.IndexOf("&laquo;", left, count) + 7,
kod_js.IndexOf("&raquo;", left, count) - kod_js.IndexOf("&laquo;", left,
count) - 7)).Replace("&quot;", "\\").ToString();
        StringBuilder adress = new StringBuilder
(kod_js.Substring(kod_js.IndexOf("</u><i>", left, count) + 7,
kod_js.IndexOf("</i><i>", left, count) - kod_js.IndexOf("</u><i>", left,
count) - 7));
        adress.Replace("&quot;", "\\");

        List<string> nomera = new List<string>();

        int beg_nom = kod_js.IndexOf("+7", left, count);

        while (beg_nom != -1)
        {
            string nom = kod_js.Substring(beg_nom,
kod_js.IndexOf("<", beg_nom) - beg_nom);
            nomera.Add(nom);
            beg_nom = kod_js.IndexOf("+7", beg_nom + 1,
right - beg_nom - 1);
        }
    }
}

```

```

    }

    richTextBox5.Text = avtoskola_name_k + "\n";
    for(int i=0; i<nomera.Count; i++)
    {
        richTextBox5.Text += nomera [i] + "\n";
    }
    richTextBox5.Text += adress + "\n" + koord [0] + "\n"
+ koord [1] + "\n";

    progress_current++;
    number_point++;
    // Записываем центр карты и масштаб
    using (StreamWriter sw = new
StreamWriter(path_for_write + path_city_without_index + "\\\\" + category_en
+ ".txt", true, Encoding.UTF8))
    {
        sw.Write("X" + delitel_2 + koord [0] + delitel_3
+ "Y" + delitel_2 + koord [1] + delitel_3 + "Название" + delitel_2 +
avtoskola_name_k + delitel_3);
        if
(dict_avtoshkola_price.ContainsKey(avtoskola_name_k))
        {
            sw.Write("Цена" + delitel_2 +
dict_avtoshkola_price [avtoskola_name_k]);
        }
        else
        {
            sw.Write("Цена" + delitel_2 + "не
указана");
        }

        sw.Write(delitel_3 + "Homep(a)" + delitel_2);
        for (int i = 0; i < nomera.Count; i++)
        {
            sw.Write(nomera [i] + delitel_1);
        }

        sw.Write(delitel_3 + "Адрес" + delitel_2 +
adress);

        if
(dict_avtoshkola_site.ContainsKey(avtoskola_name_k))
        {
            sw.Write(delitel_3 + "Сайт" + delitel_2 +
dict_avtoshkola_site [avtoskola_name_k]);
            richTextBox5.Text += dict_avtoshkola_site
[avtoskola_name_k] + "\n";
        }
        else
        {
            sw.Write(delitel_3 + "Сайт" + delitel_2 +
"не указан");
        }
        if
(dict_avtoshkola_email.ContainsKey(avtoskola_name_k))
        {
            sw.Write(delitel_3 + "Email" + delitel_2 +
dict_avtoshkola_email [avtoskola_name_k]);
            richTextBox5.Text += dict_avtoshkola_email
[avtoskola_name_k];
        }
        else
        {

```

```

        sw.Write(delitel_3 + "Email" + delitel_2 +
"не указан");
    }
    sw.WriteLine();
    sw.Close();
}

left = kod_js.IndexOf(".Placemark(", right);
right = kod_js.IndexOf("d(myPlacemark", right + 1);

}
#endregion
}

private void Info_from_Avtoshkola(string avtoskola_name,
StringBuilder path_avtoshkoli, Dictionary<string,string>
dict_avtoshkola_site, Dictionary<string, string> dict_avtoshkola_email)
{
    // Avtoskola -> site, email

    HtmlAgilityPack.HtmlDocument document_info = new
HtmlAgilityPack.HtmlDocument();
    string text_html_info = File.ReadAllText(path_main +
path_avtoshkoli);
    document_info.LoadHtml(text_html_info);
    HtmlNodeCollection nodes =
document_info.DocumentNode.SelectNodes("/html/body/div/div/div/div/div/div/div/div/div/div");
    foreach(HtmlNode node in nodes)
    {
        if (node.GetAttributeValue("class", "-class-") ==
"contacts_item_site_blue contacts_item_icon_label")
        {
            string site = node.InnerText;
            if
(!dict_avtoshkola_site.ContainsKey(avtoskola_name))
            {
                dict_avtoshkola_site.Add(avtoskola_name,
site);
            }
        }
        if (node.GetAttributeValue("class", "-class-") ==
"contacts_item_email_blue contacts_item_icon_label")
        {
            string email = node.InnerText;
            if
(!dict_avtoshkola_email.ContainsKey(avtoskola_name))
            {
                dict_avtoshkola_email.Add(avtoskola_name,
email);
            }
        }
    }
}

private void timer1_Tick(object sender, EventArgs e)
{
    tick();
}

private void tick()
{
    sec++;
}

```



```

        if (sec > 59)
        {
            sec = 0;
            min++;
            if (min > 59)
            {
                min = 0;
                hours++;
            }
        }
        time_str = ( hours > 9 ? hours.ToString() : "0" +
hours.ToString() ) + ":" + ( min > 9 ? min.ToString() : "0" + min.ToString()
) + ":" + ( sec > 9 ? sec.ToString() : "0" + sec.ToString() );
        time_str += ". Progress: " + ( (int) ( (double)
progress_current / progress_max * 100 ) ).ToString() + "% (" +
progress_current + "/" + progress_max + ")";
        if (progress_current == 0)
            time_str += " ...Удаление старых данных...";
        label_time.Text = time_str;
    }

    private void btn_parsing_Click(object sender, EventArgs e)
    {
        timer_sec.Enabled = true;
        Thread thread = new Thread(Parsing);
        thread.Start();
    }
}

```

2. Код клиентской части приложения

```

package ru.desyatovalexander10.drivingschools;

import android.content.Intent;
import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterViewAdapter;
import android.widget.EditText;
import android.widget.ListView;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.nio.charset.Charset;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashMap;
import java.util.Map;

public class CityActivity extends AppCompatActivity {
    EditText etFind;
    ListView lvCities;
}

```

```

HashMap<String, String> city_categories; // пара: город - категории
ArrayList<String> cities_in_list;
String delitel_1 = ":=:";
String delitel_3 = ":=::=";
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_city);

    etFind = findViewById(R.id.etFind);
    etFind.setEnabled(false);
    etFind.setText(R.string.get_cities);
    etFind.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int
count, int after) {

        }

        @Override
        public void onTextChanged(CharSequence s, int start, int before,
int count) {
            list_formation();
        }

        @Override
        public void afterTextChanged(Editable s) {

        }
    });
    lvCities = findViewById(R.id.lvCities);
    lvCities.setOnItemClickListener(new AdapterView.OnItemClickListener()
{
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
        try {

            String city_ru = cities_in_list.get(position);
            Intent intent = new Intent(CityActivity.this,
CategoryActivity.class);

            for (Map.Entry<String, String> entry :
city_categories.entrySet())
                if (entry.getKey().contains(city_ru)) {

                    intent.putExtra("city", entry.getKey());
                    intent.putExtra("categories", entry.getValue());

                    break;
                }
            startActivity(intent);
        } catch (Exception e) {
            Log.d("myTag", e.toString());
        }
    }
});
    new MyAsyncTask_getting_cities().execute();
}

class MyAsyncTask_getting_cities extends AsyncTask<Void, Void,
HashMap<String,String>> {
    String server = "http://192.168.0.110/driving_schools/";//дом
    //String server = "http://191985xt.beget.tech/driving_schools/";

```

```

@Override
protected void onPreExecute() {
    super.onPreExecute();
}

@Override
protected HashMap<String, String> doInBackground(Void... voids) {
    HashMap<String, String> postDataParams = new HashMap<String,
String>();
    postDataParams.put("all", "all");
    return performGetCall(server, postDataParams);
}

@Override
protected void onPostExecute(HashMap<String,String> result) {
    super.onPostExecute(result);
    city_categories = new HashMap<String, String> (result);

    etFind.setEnabled(true);
    etFind.setText("");
    etFind.setHint(R.string.searching);
}
}

private HashMap<String,String> performGetCall(String requestURL,
HashMap<String, String> DataParams) {
    HashMap<String, String> response = new HashMap<>();
    URL url;
    HttpURLConnection urlConnection = null;
    try {
        url = new URL(requestURL + getDataString(DataParams));
        urlConnection = (HttpURLConnection) url.openConnection();
        int responseCode = urlConnection.getResponseCode();
        if(responseCode == HttpURLConnection.HTTP_OK){
            response =
convertInputStreamToHashMap(urlConnection.getInputStream());
        }
        else {
            Log.d("myTag", "performGetCall: try... responseCode == " +
String.valueOf(responseCode) + " " + requestURL + getDataString(DataParams));
        }
    } catch (Exception e) {
        Log.d("myTag", e.toString());
        e.printStackTrace();
    } finally {
        if (urlConnection != null)
            urlConnection.disconnect();
    }
    return response;
}

private HashMap<String,String> convertInputStreamToHashMap(InputStream
in) {
    BufferedReader reader = null;
    HashMap<String, String> response = new HashMap<>();
    try {
        reader = new BufferedReader(new InputStreamReader(in,
Charset.forName("UTF-8")));
        String line = "";
        while ((line = reader.readLine()) != null) {
            String[] city_0_categories_1 = line.split(delitel_3);
            response.put(city_0_categories_1[0], city_0_categories_1[1]);
        }
    } catch (IOException e) {

```

```

        e.printStackTrace();
    } finally {
        if (reader != null) {
            try {
                reader.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
    return response;
}

private String getDataString(HashMap<String, String> params) throws
UnsupportedEncodingException {
    StringBuilder result = new StringBuilder();
    result.append("?");
    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();
}

// Формируем списка, в зависимости от того, что в строке поиска
private void list_formation(){
    // В cities_in_list загоняем только русскую запись названия города
    // если в поиске есть часть этого города
    cities_in_list = new ArrayList<String>();
    // То, что в поисковой строке
    String find_city = etFind.getText().toString().toLowerCase();
    if(find_city.equals(""))
        for (Map.Entry<String, String> entry :
city_categories.entrySet())
            cities_in_list.add(entry.getKey().split(delitel_1)[0]);
    else {
        for (Map.Entry<String, String> entry :
city_categories.entrySet()) {
            String city = entry.getKey().split(delitel_1)[0];
            if(city.toLowerCase().contains(find_city))
                cities_in_list.add(city);
        }
    }
    if(cities_in_list.size() == 0) {
        lvCities.setAdapter(null);
        return;
    }
    else{
        Collections.sort(cities_in_list);

        // Только самый первый элемент ставит в конец почему-то. (не
всегда)
        // Последний элемент ставим в начало
        // cities_in_list.add(0,
cities_in_list.remove(cities_in_list.size() - 1));
    }
}

```

```

        ArrayAdapter<String> adapter_cities = new ArrayAdapter<>(this,
android.R.layout.simple_list_item_1, cities_in_list);
        lvCities.setAdapter(adapter_cities);

    }

}

}

package ru.desyatovalexander10.drivingschools;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterViewAdapter;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;

import java.util.ArrayList;
import android.util.Log;

public class CategoryActivity extends Activity {
    TextView tv_city;
    ListView lvCities;
    String delitel_1 = ":=:";
    String delitel_2 = "===:";
    String city_intent;
    String categories_intent;
    @Override
    protected void onResume()
    {
        tv_city.setText(city_intent.split(delitel_1)[0]);
        String[] categories_ru_en = categories_intent.split(delitel_2);
        final ArrayList<String> categories_in_list = new ArrayList<String>();

        for (String str : categories_ru_en)
            categories_in_list.add(str.split(delitel_1)[0]);

        ArrayAdapter<String> adapter_categories = new ArrayAdapter<>(this,
android.R.layout.simple_list_item_1, categories_in_list);
        lvCities.setAdapter(adapter_categories);
        super.onResume();
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_category);

        final Intent intent_from_city = getIntent();
        city_intent = intent_from_city.getStringExtra("city");
        categories_intent = intent_from_city.getStringExtra("categories");

        tv_city = findViewById(R.id.tv_city);
        tv_city.setText(city_intent.split(delitel_1)[0]);

        lvCities = findViewById(R.id.lvCategories);

        final String[] categories_ru_en = categories_intent.split(delitel_2);
        final ArrayList<String> categories_in_list = new ArrayList<String>();

        for (String str : categories_ru_en)
            categories_in_list.add(str.split(delitel_1)[0]);
    }
}

```

```

        final ArrayAdapter<String> adapter_categories = new
ArrayAdapter<>(this, android.R.layout.simple_list_item_1,
categories_in_list);
        lvCities.setAdapter(adapter_categories);
        lvCities.setOnItemClickListener(new AdapterView.OnItemClickListener()
{
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
        try {
            lvCities.setAdapter(null);
            String category_ru = categories_in_list.get(position);
            String categ_ru_en = "";
            for (String categ : categories_ru_en)
                if(categ.contains(category_ru)) {
                    categ_ru_en = categ;
                    break;
                }
            Intent intent_to_map = new Intent(CategoryActivity.this,
MapsActivity.class);
            intent_to_map.putExtra("city", city_intent);
            intent_to_map.putExtra("category", categ_ru_en);
            tv_city.setText(R.string.get_points);
            startActivity(intent_to_map);
        }
        catch (Exception e){
            Log.d("myTag", e.toString());
        }
    }
});
}

}

package ru.desyatovalexander10.drivingschools;

import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.os.AsyncTask;
import android.support.v4.app.FragmentActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.TextView;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptor;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;

```

```

import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.nio.charset.Charset;
import java.util.HashMap;
import java.util.Map;

public class MapsActivity extends FragmentActivity implements
OnMapReadyCallback {
    TextView tv_city_and_category;
    private GoogleMap map;
    String city_ru_en;
    String category_ru_en;
    String answerHTTP;
    String delitel_1 = ":=:";
    String delitel_2 = "===:";
    String delitel_3 = "====:";
    String myTag = "myTag";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);

        final Intent intent_from_category = getIntent();
        city_ru_en = intent_from_category.getStringExtra("city");
        category_ru_en = intent_from_category.getStringExtra("category");

        tv_city_and_category = findViewById(R.id.tv_city_and_category);
        tv_city_and_category.setText(city_ru_en.split(delitel_1)[0] + ", " +
category_ru_en.split(delitel_1)[0]);

        new MyAsyncTask_getting_points().execute();
    }

    @Override
    public void onMapReady(GoogleMap googleMap) {
        map = googleMap;
        // Обработать этот случай, мб инета нет, или нет ответа от сервера
        if(answerHTTP.equals(""))
            return;
        String[] all_points_str = answerHTTP.split("\n");
        // Центр, масштаб
        float x_center = 0f, y_center = 0f;
        float zoom = 10f;

        String[] x_y_z = all_points_str[0].split(delitel_3);

        x_center = Float.parseFloat(x_y_z[0].split(delitel_2)[1]);
        y_center = Float.parseFloat(x_y_z[1].split(delitel_2)[1]);
        zoom = Float.parseFloat(x_y_z[2].split(delitel_2)[1]);

        map.setMinZoomPreference(zoom);
        //map.setMaxZoomPreference(zoom+1);
        LatLng latLng = new LatLng(x_center, y_center); // центр камеры сюда
        map.moveCamera(CameraUpdateFactory.newLatLng(latLng));

        // Инфоокно
        map.setInfoWindowAdapter(new GoogleMap.InfoWindowAdapter() {
            @Override //Замена всего инфоокна
            public View getInfoWindow(Marker marker) {
                // если не сделает это, останется title и snippet по
умолчанию)
                try {

```

```

        // layout берем за основу инфоокна
        //Отменить растягивание по всей ширине
        View view =
getLayoutInflater().inflate(R.layout.info_window, null);

        // Название автошколы сюда
        TextView textView = view.findViewById(R.id.tv);
        textView.setText(marker.getTitle());

        // Каждая часть информации будет элементом списка
        String[] inform = marker.getSnippet().split("\n");
        // создаем адаптер
        ArrayAdapter<String> adapter = new
ArrayAdapter<String>(MapsActivity.this,
                    R.layout.list_point_for_info_window, inform);
        /*ArrayAdapter<String> adapter = new
ArrayAdapter<String>(MapsActivity.this,
                    R.layout.support_simple_spinner_dropdown_item,
inform);*/

        // находим список
        ListView listView = view.findViewById(R.id.lv);
        // присваиваем адаптер списку
        listView.setAdapter(adapter);

        return view;
    } catch (Exception e) {
        return null;
    }
}

@Override
public View getInfoContents(Marker marker) {
    return null;
}

});

// Открываем файлы-точки ( В отдельном потоке мб?)
try {
    for(int i = 1; i < all_points_str.length; i++) {
        if(!all_points_str[i].equals("")){
            MyMarker myMarker = new MyMarker(all_points_str[i]);
        }
    }
} catch (RuntimeException e) {
    Log.d(myTag, "ИСКЛЮЧЕНИЕ " + e.toString());
}

}

class MyMarker {
    // Расставить потоки
    Marker marker;

    float x, y;
    String name;
    String price_string;
    float price_float;
    String category;
    String[] numbers;
    String address;
    String site;
    String email; // Бывает несколько? чего еще бывает несколько?
    // цена, email мб "не указано"

    MyMarker(String all_infa)

```



```

    {
        process_all_info(all_infa);
        build_marker();
        // Мб, сделать это, когда все маркеры построятся ??
        add_info_window();
    }
    void process_all_info(String main_line){
        String[] infa_otdelno = main_line.split(delitel_3);
        for(int i = 0; i < infa_otdelno.length; i++){

            if (infa_otdelno[i].contains("X")) {
                x =
Float.parseFloat(infa_otdelno[i].split(delitel_2)[1]);
            } else if (infa_otdelno[i].contains("Y")) {
                y =
Float.parseFloat(infa_otdelno[i].split(delitel_2)[1]);
            } else if (infa_otdelno[i].contains("Название")) {
                name = infa_otdelno[i].split(delitel_2)[1];
            } else if (infa_otdelno[i].contains("Цена")) {
                price_string = infa_otdelno[i].split(delitel_2)[1]
                    .replace(" руб.", "");
            } else if (infa_otdelno[i].contains("Номер(a)")) {
                numbers =
infa_otdelno[i].split(delitel_2)[1].split(delitel_1);
            } else if (infa_otdelno[i].contains("Адрес")) {
                address = infa_otdelno[i].split(delitel_2)[1];
            } else if (infa_otdelno[i].contains("Сайт")) {
                site = infa_otdelno[i].split(delitel_2)[1];
            } else if (infa_otdelno[i].contains("Email")) {
                email = infa_otdelno[i].split(delitel_2)[1];
            }
        }
    }
    // Рисуем маркер
    void build_marker()
    {
        // Общее для всего маркера
        int bitmapWidth = 120;
        int bitmapHeight = 80;
        Bitmap bitmap = Bitmap.createBitmap(bitmapWidth, bitmapHeight,
Bitmap.Config.ARGB_8888);
        //bitmap.eraseColor(Color.BLUE); // Сделать прозрачным или просто
убрать

        Canvas canvas = new Canvas(bitmap);
        Paint paint = new Paint();

        // rectangle_name
        paint.setStrokeWidth(3);
        paint.setColor(Color.argb(100,1,1,1));
        canvas.drawRect(0, bitmapHeight/3, bitmapWidth,
7*bitmapHeight/12, paint);

        // Название
        paint.setTextAlign(Paint.Align.CENTER);
        paint.setColor(Color.BLACK);
        paint.setTextSize(20);
        // Пока так
        if(name.length() > 10)
            canvas.drawText(name.substring(0, 10), bitmapWidth/2,
7*bitmapHeight/12 - 2, paint);
        else
            canvas.drawText(name, bitmapWidth/2, 7*bitmapHeight/12 - 2,
paint);

        if(!price_string.equals("не указано")) {

```

```

        // rectangle_price
        paint.setStrokeWidth(3);
        paint.setColor(Color.argb(100,1,1,1));
        canvas.drawRect(0, 0, bitmapWidth, bitmapHeight/3, paint);
        // Цена
        paint.setColor(Color.WHITE);
        paint.setTextSize(30);
        canvas.drawText(price_string, bitmapWidth / 2, bitmapHeight /
3 - 2, paint);
    }
    // rectangle_post
    paint.setStrokeWidth(3);
    paint.setColor(Color.BLACK);
    canvas.drawRect(bitmapWidth/2 - 2, 7*bitmapHeight/12,
bitmapWidth/2 + 2, bitmapHeight, paint);

    BitmapDescriptor bitmapDescriptor =
BitmapDescriptorFactory.fromBitmap(bitmap);

    LatLng latLng = new LatLng(x, y);
    MarkerOptions markerOptions = new MarkerOptions()
        .position(latLng)
        .icon(bitmapDescriptor)
        .anchor(0.5f, 1f);
    marker = map.addMarker(markerOptions); // мар пока глобально (а
если класс вынести отдельно)
}
// Добавление инфоокна
void add_info_window()
{
    marker.setTitle(name);

    String detailed_info = "";

    // Нужно ли проверять на null ?
    if(numbers != null)
        for (String num : numbers)
            detailed_info += num + "\n";
    // А если не указано?
    if(address != null)
        detailed_info += address + "\n";
    if(site != null)
        detailed_info += site + "\n";
    if(email != null)
        detailed_info += email;

    marker.setSnippet(detailed_info);

    // Title и Snippet потом будут использоваться при формировании
своего инфоокна
}
// Удалять маркеры при переходе к другой карте в отдельном потоке?
void delete и память
}

class MyAsyncTask_getting_points extends AsyncTask<Void, Void, String> {
    String city_en;
    String category_en;
    String server = "http://192.168.0.110/driving_schools/";//дом

    @Override
    protected void onPreExecute() {
        city_en = city_ru_en.split(delitel_1)[1];

```

```

        category_en = category_ru_en.split(delitel_1)[1];
        super.onPreExecute();
    }
    @Override
    protected String doInBackground(Void... voids) {
        HashMap<String, String> postDataParams = new HashMap<String,
String>();
        postDataParams.put("city", city_en);
        postDataParams.put("category", category_en);
        answerHTTP = performGetCall(server, postDataParams);
        return answerHTTP;
    }
    @Override
    protected void onPostExecute(String result) {
        super.onPostExecute(result);
        answerHTTP = result;

        // Обрабатываем ответ!
        SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.map);
        mapFragment.getMapAsync(MapsActivity.this);
    }
}

    public String performGetCall(String requestURL, HashMap<String, String>
DataParams) {
        String response = "";
        URL url;
        HttpURLConnection urlConnection = null;
        try {
            url = new URL(requestURL + "?" + getDataString(DataParams));

            urlConnection = (HttpURLConnection) url.openConnection();

            int responseCode = urlConnection.getResponseCode();

            if(responseCode == HttpURLConnection.HTTP_OK){
                response =
convertInputStreamToString(urlConnection.getInputStream());
            }
            else
                Log.d("myTag", "performGetCall: try... responseCode == " +
String.valueOf(responseCode));
        } catch (Exception e) {
            Log.d("myTag", "performGetCall: catch (Exception e)");
            e.printStackTrace();
        } finally {
            if (urlConnection != null) {
                urlConnection.disconnect();
            }
        }
        return response;
    }

    private String getDataString(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();
}

private String convertInputStreamToString(InputStream in) {
    BufferedReader reader = null;
    StringBuffer response = new StringBuffer();
    try {
        reader = new BufferedReader(new InputStreamReader(in,
Charset.forName("UTF-8")));
        String line = "";
        while ((line = reader.readLine()) != null) {
            response.append(line + "\n");
        }
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (reader != null) {
            try {
                reader.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
    return response.toString();
}
}
}

```

3. Код серверной части приложения

```

<?php

if(isset($_GET['city']))
    $string_city = htmlentities($_GET['city']);
if(isset($_GET['category']))
    $string_category = htmlentities($_GET['category']);
if(isset($_GET['all']))
    $string_all = htmlentities($_GET['all']);

if(!empty($string_all)) {
    $path = "DateBase_Driving_Schools/cities_categories.txt";
    echo file_get_contents($path);
}
else if(!empty($string_city) && !empty($string_category)) {
    $path = "DateBase_Driving_Schools/$string_city/$string_category.txt";
    echo file_get_contents($path);
}

```