Санкт-Петербургский государственный университет

Кафедра информатики

Фундаментальная информатика и информационные системы

Хохлова Дарья Дмитриевна

РАЗРАБОТКА И РЕАЛИЗАЦИЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ИДЕНТИФИКАЦИИ АБОНЕНТОВ

Бакалаврская работа

Научный руководитель:
доц. каф. инф., к. ф.-м. н. Григорьев Д. А.

Рецензент:
инженер по тестированию ПО ООО «Алкатель-Лусент Энтерпрайз СиАйЭс» Кузьмина Т. И.

Санкт-Петербург
2016

Saint-Petersburg State University

Computer Science Department

Fundamental Informatics and Information Technology

Daria Khokhlova

DEVELOPMENT AND IMPLEMENTATION OF A MOBILE APPLICATION FOR CALLER IDENTIFICATION

Bachelor's Thesis

Scientific supervisor:
Ph. D. of Sc., associate professor Dmitrii Grigoriev

Reviewer:
Soft. test eng. at Alcatel-Lucent Enterprise CIS Tatiana Kuzmina

Saint-Petersburg
2016

Содержание

Введение

Определение имени и цели звонящего абонента по его телефонному номеру — актуальная задача, которая на сегодняшний момент в ИТ-индустрии не решается комплексно [44]. Возможностью приобрести подробные телефонные базы, доступные любому желающему, пользуются коллекторские и рекламные агентства, телефонные мошенники и другие люди и организации, досаждая владельцам смартфонов нежелательными звонками. Не зная, кто ему звонит, абонент вынужден тратить время на бесполезные беседы. Другая проблема, существующая в современном обществе — информационная перегрузка. У абонента нет времени отвечать на все входящие вызовы. Большинство людей хотели бы сэкономить своё время и другие ресурсы, отказываясь от разговоров, которые не принесут важной информации. В этом свете актуальной представляется задача разработки технологической системы, которая могла бы обеспечить пользователей мгновенным доступом к важной информации о входящих вызовах — кто именно звонит и с какой целью.

Существующие решения — онлайн-справочники (например, «Жёлтые страницы» [9]), мобильные приложения (TrueCaller [30], Contactive [20], Whoscall [32]), — решают задачу лишь частично, обеспечивая либо высокую скорость взаимодействия с системой (за счёт небольшого размера базы данных, содержащей только те номера, звонки с которых совершаются чаще всего, и чаще всего хранящейся прямо на мобильном устройстве), либо точность подачи информации и её актуальность (как, правило, точную и актуальную информацию по большинству запросов пользователя предоставляют объёмные онлайн-базы, доступ к которой осуществляется при помощи веб-браузера). Ещё одна сложность в разработке полезной системы заключается в быстром устаревании информации — и коллекторы, и рекламные агентства, и обычные пользователи время от времени ме-

няют телефонные номера, и номер, который несколько дней или месяцев принадлежал одному абоненту, уже сейчас может принадлежать другому человеку. Решением этой проблемы могла бы стать рейтинговая система, обновление которой происходит на основе полученной от пользователей обратной связи. Многие из перечисленных решений обеспечивают механизмы получения и обработки обратной связи, но скорость работы этих механизмов может оказаться слишком мала, чтобы оперативно реагировать на изменения в телефонной базе.

Представленная в данной работе система учитывает недостатки упомянутых выше решений. Теоретическая ценность такой системы состоит в демонстрации возможности уточнять, добавлять и ранжировать информацию об абонентах самим пользователям. Практическая цель данной работы — создать такую технологическую платформу, которая могла бы предоставлять пользователю максимально полную информацию о характере входящего вызова — кто именно звонит и с какой целью. При этом система должна обрабатывать обратную связь от пользователей, а также предоставлять информацию в сжатые сроки (а именно, обработка запроса не должна превышать 1–2 секунды: именно столько обычно требуется человеку, чтобы оценить ситуацию и принять решение — в данном случае, ответить на звонок), а также работать в облегченном виде без доступа к сети Интернет.

Наиболее комфортной для пользователя представляется ситуация, когда информация о входящем вызове отображается мгновенно на самом мобильном устройстве. Именно поэтому для разработки системы, представленной в данной квалификационной работе, в качестве платформы была выбрана операционная система Android [14]. На базе этой операционной системы к началу 2016 года работает 80,7% [51] смартфонов по всему миру. Основной IDE (средой разработки) была выбрана система Xamarin [34] — она обеспечивает средства кроссплатформенной разработки и портирова-

ния мобильных приложений, так что созданное решение можно будет перенести и на другие мобильные операционные системы. Непрерывную работу системы и её масштабируемость обеспечивает облачный бэкенд, реализованный на платформе Microsoft Azure [24].

Представленное решение состоит из мобильного приложения на базе Android, названного CallDialer (реализовано на платформе .NET [63] во фреймворке Xamarin), двух веб-приложений (реализованных на платформе ASP.NET) и СУБД [5]. И СУБД, и веб-приложения на базе ASP.NET размещены в облачном хранилище Microsoft Azure [18].

В СУБД хранятся телефонные номера и варианты описаний, которые им присваивают клиенты системы, а также дата и время последнего звонка с каждого номера. Кроме того, в ней ведётся учёт рейтинга каждого из вариантов и каждого из клиентов системы. Рейтинг конкретного имени определяется тем, какое количество клиентов подтвердили его состоятельность и каков рейтинг этих клиентов. Рейтинг клиентов зависит от количества добавленных вариантов имён и их рейтинга. Одно из разработанных веб-приложений отвечает за обработку рейтингов — как пользователей, так и добавленных ими вариантов имён. Второе веб-приложение выступает в качестве анализатора, который определяет владельца телефонного номера и цель его звонков, обрабатывая записи пользователей сети Интернет, собранных с различных специализированных сайтов.

В мобильном приложении пользователь может просмотреть историю входящих вызовов, дополнительную информацию по каждому телефонному номеру и предложить собственные варианты имён. При каждом звонке с неизвестного пользователю номера приложение выводит на главный экран имя с самым высоким рейтингом на текущий момент.

Для удобства изложения сначала раскрывается часть данной работы, касающаяся разработки рейтинговой системы ранжирования вариантов описаний (глава 4), а затем — разработки самой технологической системы (гла-

ва 5). Все используемые определения вынесены в главу 1.

1 Определения и обозначения

В этом разделе даны определения основным терминам, которые используются в квалификационной работе.

Облачные вычисления [59] — технологическая концепция предоставления сетевого доступа к набору настраиваемых вычислительных ресурсов (например, системам управления базами данных, серверам, хранилищам данных, приложениям и сервисам). Позволяет [38] удешевить доступ и использование различных вычислительных ресурсов.

Облачная платформа — обособленная платформа для облачных вычислений.

SaaS [62] — (Software as a Service, «программное обеспечение как сервис») модель обслуживания облачной платформы, при которой пользователю предоставляется онлайн-доступ к программному обеспечению, разработанному и контролируемому его поставщиком.

PaaS [61] — (Platform as a Service, «платформа как сервис») модель обслуживания облачной платформы, при которой пользователю предоставляется онлайн-доступ к использованию технологических платформ: различных операционных систем, СУБД, средствам разработки и тестирования.

IaaS [8]— (Infrastructure as a Service, «платформа как сервис») модель обслуживания облачной платформы, при которой пользователю предоставляется онлайн-доступ к вычислительной инфраструктуре для развёртывания собственного программного обеспечения: сервера, хранилища данных, компьютерные сети и так далее.

Microsoft Azure — облачная платформа, разработанная корпорацией Microsoft. Реализует модели обслуживания PaaS и IaaS.

ASP.NET — технология для создания веб-приложений и веб-сервисов, разработанная корпорацией Microsoft.

API [58] — (application programming interface, интерфейс прикладного программирования) набор классов, методов, структур и констант, предоставляемых веб-приложением для использования во внешних приложениях.

Android — операционная система для смартфонов, интернет-планшетов, электронных книг, цифровых проигрывателей, наручных часов, игровых приставок, нетбуков, смартбуков, очков Google, телевизоров и других устройств. Основана на ядре Linux и собственной реализации виртуальной машины Java от Google. Изначально разрабатывалась компанией Android Inc., которую в июле 2005 года выкупила [52] корпорация Google. По данным исследовательской компании Gartner, к началу 2016 года мировая доля рынка мобильных устройств на базе Android составила 80,7% [51].

Xamarin — фреймворк для кроссплатформенной разработки мобильных приложений (iOS [23], Android, Windows Phone [33]) с использованием языка C#. Основан на платформе Mono [25] — открытой реализации платформы .NET. Mono включает [46] собственный компилятор для языка C#, среду выполнения и основные библиотеки платформы .NET.

REST [42] — (Representational State Transfer — «передача состояния представления») архитектурный стиль взаимодействия компонентов распределённого приложения в сети. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой системы.

RESTful — веб-сервис, удовлетворяющий ограничениям архитектуры REST.

ORM (Object-Relational Mapping, объектно-реляционное отображение) [60] — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «вир-

туальную объектную базу данных».

ADO.NET Entity Framework [57] — объектно-ориентированная технология доступа к данным, является ORM-решением для .NET от Microsoft. Предоставляет возможность взаимодействия с объектами с использованием Entity SQL [26]. Использует инструменты доступа к базам данных ADO.NET [6, 56].

ADO.NET построена с использованием паттерна проектирования Unit of Work [41]. Технология нацелена на автономную работу с помощью объектов DataSet. Эти типы представляют локальные копии любого количества взаимосвязанных таблиц данных, каждая из которых содержит набор строк и столбцов. Объекты DataSet позволяют вызывающей сборке работать с содержимым DataSet, изменять его, не требуя подключения к источнику данных, и отправлять обратно блоки измененных данных для обработки с помощью соответствующего адаптера данных. ADO.NET является управляемой кодовой библиотекой и подчиняется тем же правилам, что и любая управляемая библиотека.

2 Постановка целей

Теоретическая цель представленной работы состоит в разработке рейтинговой системы, обрабатывающей и учитывающей обратную связь от пользователей и позволяющей уточнять, добавлять и ранжировать информацию об абонентах самим пользователям.

Практическая цель данной работы — создать такую технологическую платформу, которая могла бы предоставлять пользователю максимально полную информацию о характере входящего вызова — кто именно звонит и с какой целью, — основываясь на иформации, полученной из различных электронных справочников, отзывов пользователей в сети Интернет и обратной связи от пользователей, поступающих в саму систему. При этом

решение должно предоставлять нужную пользователю информацию в сжатые сроки (а именно, обработка запроса не должна превышать 1–2 секунды: именно столько обычно требуется человеку, чтобы оценить ситуацию и принять решение — в данном случае, ответить на звонок), а также работать в облегченном виде без доступа к Интернет. Система должна быть облачной — это обеспечит её отказоустойчивость и масштабируемость.

3 Обзор существующих технологий для идентификации абонента

3.1 Электронные справочники

Часто для идентификации абонента используются различные электронные справочники. В них может содержаться различная информация — как релевантная для пользователя, так и нерелевантная. Например, онлайн-база GSM-Inform [21] способна предоставить пользователю данные о том, в каком регионе страны был зарегистрирован абонент. Более подробную информацию можно получить, обратившись к ресурсу «Жёлтые страницы» — в нём содержится информация о том, какой организации принадлежит тот или иной номер. В этом разделе рассмотрены самые популярные электронные справочники и их основные преимущества и недостатки.

3.1.1 Справочники телефонных кодов

Справочники телефонных кодов предоставляют информацию о том, в каком регионе был зарегистрирован абонент, которого требуется идентифицировать. В зависимости от того, какой именно справочник используется, можно получить данные разной степени точности: от региона до города или села. Самые популярные справочники — GSM-Info, «Код Телефона» [10], Indexmain [22] (по данным аналитической платформы SimilarWeb [27]).

Основные недостатки подобных справочников заключаются в неполноте предоставляемой информации, а также в скорости взаимодействия. К таким системам пользователь вынужден обращаться с помощью веб-браузера, это требует дополнительных действий и зачастую неудобно. Механизмы получения обратной связи от пользователей в системах не предусмотрены.

3.1.2 Справочники типа «Жёлтых страниц» и 2ГИС

«Жёлтые страницы» [9] и 2ГИС [36] — электронные справочники, в которых содержится информация о различных организациях на территории России (в том числе, телефонные номера, которые используют их сотрудники). Информацию создателям подобных справочников предоставляют руководители организаций — поэтому она, как правило, достаточно точна. В «Жёлтых страницах» содержится информация о более чем 1,6 миллиона различных компаний [9]. В 2ГИС — о более чем 2,8 миллиона компаний и их филиалов.

И «Жёлтые страницы», и 2ГИС обладают собственными мобильными приложениями на базе операционных систем Android и iOS, которые обеспечивают быстрый доступ к информации владельцам мобильных устройств. Оба справочника предоставляют возможность поиска организации по номеру телефона.

Основное преимущество подобных справочников перед другими системами — точность предоставляемой информации. Основные недостатки: с помощью таких систем практически невозможно идентифицировать абонента, звонящего с личного мобильного устройства. Для того, чтобы идентифицировать организацию по номеру телефона, требуется открыть соответствующую программу. Технология не может идентифицировать абонента непосредственно во время звонка. Механизмы для быстрого получения и обработки обратной связи от пользователей не предусмотрены.

3.1.3 Электронные справочники для идентификации рекламных звонков типа Tellows

Цель разработки подобных справочников [29] — предоставить пользователям возможность идентифицировать нежелательные звонки. Справочник представляет собой список номеров, к каждому из которых оставляют комментарии пользователи системы. Система анализирует полученные комментарии и на их основе определяет цель звонящего (продажа продуктов, реклама, помощь пользователю и так далее).

Среди преимуществ таких систем — точность и актуальность предоставляемой информации, а также продвинутые механизмы обработки обратной связи от пользователей. В то же время, как и другие электронные справочники, подобные технологии не могут обеспечить мобильность и мгновенную идентификацию абонента. В основном справочники этого типа наполнены информацией об абонентах, которые звонят другим пользователям с нежелательными целями, а информации об остальных абонентах в таких системах не существует. В то же время зачастую пользователю важно знать, кто именно ему звонит, даже если цели у звонящего положительные — например, некоторые абоненты принципиально не отвечают на входящие звонки, если не могут идентифицировать входящего абонента. Так они могут пропустить важный вызов.

3.2 Мобильные приложения и сервисы для мгновенной идентификации абонента

Существует несколько популярных мобильных технологий для мгновенной идентификации абонента на мобильных устройствах. Они рассмотрены в данном разделе.

3.2.1 Встроенная система идентификации абонента в мобильной операционной системе iOS

Встроенная система идентификации абонента в мобильной операционной системе iOS впервые появилась [50] в июне 2016 года в версии 9.0. Для идентификации входящего звонка технология использует данные, полученные пользователем по электронной почты, в SMS-сообщениях и с помощью других мобильных приложений, установленных на мобильном устройстве.

Основные недостатки технологии заключаются в её неуниверсальности (аналогичной системы для других операционных систем — Android, Windows Phone [33], Blackberry OS [19] — не существует), ограниченности данных (система неспособна получать информацию из сети Интернет) и отсутствии механизмов для получения и обработки обратной связи от пользователей. Преимущество технологии заключается в быстроте обработки информации и простоте использования.

3.2.2 Мобильные приложения для идентификации абонента

3.2.2.1 Truecaller

Truecaller — система, разработанная компанией True Software Scandinavia AB, офис которой располагается в Стокгольме, Швеция. Общее количество пользователей приложения на начало 2016 года составляет [64] более 200 млн человек. Количество зарегистрированных в системе номеров на начало 2016 года составляет более 2 млрд. Мобильное приложение Truecaller адаптировано для мобильных операционных систем Android, iOS и Windows Phone.

Данные о каждом абоненте система получает различными способами. Технология Truecaller анализирует [1] записи и данные на мобильном устройстве пользователя, чтобы получить информацию об абонентах. Компания-разработчик Truecaller заключает партнёрские соглашения [2] с создателя-

ми и владельцами электронных справочников (раздел 4.1) в разных странах, получая данные о номерах организаций.

У системы существуют недостатки. Мобильное приложение Truecaller способно идентифицировать входящий звонок только в том случае, если мобильное устройство подключено к сети Интернет. Система не располагает механизмами для быстрого получения и обработки обратной связи от пользователей. Технология не способна определять цель звонка, если телефонного номера звонящего абонента нет в базе программы. Система разработана в Швеции и ориентирована на западный рынок, база номеров и абонентов для России проработана не так хорошо, как для других стран. Основные преимущества системы Truecaller заключаются в мгновенной идентификации абонента и обширной базе данных.

3.2.2.2 2GIS Dialer

2GIS Dialer [37] — система для идентификации абонента, разработанная российской компанией 2ГИС. Данные о телефонных номерах мобильное приложение получает из электронного справочника 2ГИС. Система обладает теми же недостатками, что и электронный справочник 2ГИС, а именно: с помощью неё невозможно идентифицировать абонента, звонящего с личного мобильного устройства, а механизмы для быстрого получения и обработки обратной связи от пользователей не предусмотрены. Её основное преимущество заключается в способности мгновенно идентифицировать абонента во время получения пользователем входящего вызова.

4 Разработка рейтинговой системы обработки обратной связи от пользователей

4.1 Основные проблемы и задачи

Перед автором данной выпускной квалификационной работы стояла задача разработать такую систему ранжирования вариантов описаний, предоставляемых клиентами, которая учитывала бы не только количество упоминаний того или иного варианта, но и количество пользователей, отметивших конкретный вариант как правильный, а также собственный рейтинг (вес) пользователей.

Одна из проблем, с которой пришлось столкнуться при разработке системы — накрутки рейтингов описаний. Она может возникнуть в случае, если в системе регистрируется множество пользователей, целью которых является выведение на верхние строчки рейтинга одного из вариантов описаний. Все они голосуют за один из вариантов описаний, таким образом увеличивая его рейтинг.

К конечной системе были предъявлены следующие требования:

1) система должна быть непрозрачной — клиенты разработанного предложения не должны знать о том, как именно учитываются их оценки;

2) система должна предлагать методы защиты от накруток и спама;

3) значения оценок в системе должен варьироваться от 0 до 1.

4.2 Разработка рейтинговой системы

За основу рейтинговой системы автором была взята средневзвешенная система голосования [48], разработанная создателем сайта для фотографов «Фотогорький» [12].

$$Weight = th\left(\frac{N}{C}\right) * \frac{1}{5} * \frac{\sum mValue * mWeight}{\sum mWeight} \quad (1)$$

Формула (1), которая используется для расчёта веса голоса пользователя на сайте «Фотогорький», не подходит для рассмотренного в работе случая. При использовании такой средневзвешенной системы голосования предполагается, что значения оценок варьируются от 0 до 5. Представленная в данной квалификационной работе система идентификации абонента предусматривает лишь одно возможное значение выставленной пользователем оценки — 1. В таком случае последняя часть формулы (1) обратится в единицу.

Для рассмотренной в данной работе систему было решено использовать сокращённую формулу (2) вычисления веса оценки пользователя вида:

$$Weight = th\left(\frac{N}{C} * \frac{1}{5} * \sum mWeight\right), \quad (2)$$

где гиперболический тангенс точно так же учитывает активность пользователя — чем больше вариантов описаний он отправил, тем больше оценок от других пользователей он может получить. При этом гиперболический тангенс также нормирует график значения веса оценки пользователя — он может варьироваться от 0 до 1. Выражение не может принять отрицательное значение, так как все использующиеся в нём переменные неотрицательны.

Нормирующая контанта вычисляется как среднее количество оценок, выставленных одним пользователем системы.

$\sum mWeight$ — сумма весов оценок, которые получили варианты описаний, предоставленные пользователем. Хранится в базе данных системы как рейтинг самого пользователя.

Рейтинг одного варианта описания в таком случае можно определять следующим произведением (3):

$$Rate = N * \sum mWeight, \tag{3}$$

где $\sum mWeight$ — это сумма весов оценок, поставленных этому варианту, а N — их количество.

Таким образом легко отсечь спам — если один вариант описания получил большое количество оценок, но все они поставлены пользователями с нулевым или близким к нулевому весом оценки (только что установивших приложение или ранее не добавлявших «хороших» вариантов), становится очевидно, что имеет место накрутка рейтинга. Вариант описания получает низкий рейтинг и не демонстрируется пользователю. Стоит также отметить, что система не учитывает при расчёте рейтинга варианта описания оценки с весом ниже 0,2.

После добавления 5 вариантов имен и до тех пор, пока его рейтинг не сдвинется выше 0,3, пользователь блокируется. Блокировка не позволяет пользователю добавлять совершенно новые варианты, но позволяет оценивать уже имеющиеся в системе варианты. Система никак не информирует клиента о том, что возможность добавления новых вариантов для него заблокирована. Таким образом, если в дальнейшем пользователь решит изменить модель поведения и начнёт оставлять оценки вариантам описаний, которые сочли верными большинство пользователей, система сможет разблокировать для него возможность вносить в базу новые варианты описаний телефонных номеров.

При этом если пользователь системы оценивает один из самых популярных вариантов или добавляет новый вариант описания для номера, у которого пока нет ни одного варианта описания, его рейтинг увеличивается на 0,025. Это небольшое значение и оно не позволит пользователя сразу после регистрации достичь нужного порога, чтобы избежать блокировки после добавления первых 5 вариантов, но даст необходимый задел для эффективной работы системы.

Оценка любого добавленного варианта производится следующим образом: с помощью мобильного приложения пользователь разработанной системы указывает нужный вариант описания. Веб-приложение сравнивает добавленный пользователем вариант с уже имеющимися в системе. В случае, если этот вариант совпадает с каким-либо из уже добавленных, считается, что пользователь оценил соответствующий вариант. Система пересчитывает рейтинг клиента и рейтинг предложенного им варианта.

В случае, если соответствующего варианта описания пока не содержится в базе, он добавляется в систему, рейтинг самого пользователя также пересчитывается.

4.3 Отличия от существующих систем и обзор существующих вариантов

Большинство из существующих рейтинговых систем реализованы по двум принципам — «плюс–минус» или «от 0 до N».

В системах типа «плюс-минус» предполагается наличие возможности оставить как отрицательную, так и положительную оценку. В системах второго типа пользователь может поставить любую оценку значением от 0 до N.

В системах обоих типов предполагается, что клиент системы осведомлён о наличии рейтинговой системы, она реализована явно — так, что пользователь вытавляет оценки осознанно. В разрабатываимой технологии было решено реализовать «скрытую» рейтинговую систему — то есть такую, что при оценивании какого-либо из вариантов описаний пользователь не осознаёт, что он оценивает запись, а полагает, что добавляет в базу новый вариант описания. В таком случае логичнее было бы реализовать систему, которая предполагает лишь один возможный вариант оценки каждой записи.

4.4 Обзор альтернативных вариантов

Для обзора в данной работе были выбраны несколько рейтинговых систем, построенных на наиболее близких идеях.

4.4.1 Рейтинговая система сайта «Хабрахабр»

На сайте проекта «Хабрахабр» [13] используется система «Кармы» типа «плюс-минус». Любой пользователь сайта может публиковать заметки на определённую тематику. Эти заметки получают оценки от других клиентов системы. Предусмотрены оценки двух типов: «плюс» или «минус». В зависимости от количества полученных оценок и их типа устанавливается рейтинг пользователя, отправившего заметку на сайт. Рейтинг начисляется и за комментарии к заметкам — по аналогичному принципу.

Чем выше рейтинг клиента в описанной системе, тем больше у него возможностей на сайте. Клиенты с самым высоким рейтингом («кармой») могут не только публиковать материалы на сайте, но и рассылать приглашения для регистрации в системе своим знакомым, а также публиковать материалы в специализированных разделах.

Такая система наиболее близка к рейтинговой системе, реализованной в приложении CallDialer. Она обладает всеми теми же недостатками, что и другие системы типа «плюс-минус»:

1) клиенты системы осведомлены о её наличии на используемом ресурсе;

2) клиенты системы обладают возможностью присваивать публикациям и положительные, и отрицательные оценки.

Ещё одна особенность, предусмотренная реализованной на ресурсе «Хабрахабр» системой — обнуление полученного рейтинга через заданный промежуток времени. Если после публикации одного из материалов рейтинг клиента системы увеличился на число X, спустя определённое время он

снизится — на то же число X. Такая особенность позволяет влиять на рейтинги других пользователей даже новым клиентам системы и вынуждает пользователей публиковать новые материалы. В реализованной в данной квалификационной работе системе такой особенности не предусматривается — для того, чтобы не дать новым пользователям влиять на показатели рейтингов других клиентов.

Существенное отличие рейтинговой системы «Хабрахабра» от рейтинговой системы, представленной в данной работе, заключается в том, что при подсчёте показателя рейтинга пользователя на сайте «Хабрахабр» все перечисленные показатели (оценки, полученные за публикацию материалов, и оценки, полученные за публикацию комментариев к материалам) учитываются независимо [7]. В реализованной в данной работе системе предполагается, что оценки и публикации (оставленные клиентом варианты) неразличимы с точки зрения пользователя системы, и при подсчёте рейтинга каждого варианта или показателя рейтинга самого пользователя они учитываются вместе.

Для расчёта рейтинга пользователя на сайте проекта предположительно [7] используется формула, близкая к выражению (4)

$$Rate = \frac{k}{10} + \frac{4t}{5} + \frac{c}{100}, \tag{4}$$

где переменные k, t и c характеризуют количество и значение оценок, который пользователь получает за публикацию заметок и комментариев на сайте.

4.4.2 Рейтинговая система сайта «Фотогорький»

Система, реализованная на онлайн-платформе «Фотогорький», была взята за основу при реализации рейтинговой системы для настоящей работы. Её основные преимущества и недостатки подробно описаны в разделе 5.2. Ещё раз перечислим основные из них:

1) клиенты системы осведомлены о её наличии на используемом ресурсе;

2) клиенты системы обладают возможностью присваивать публикациям оценки со значением от 0 до 5.

4.4.3 Рейтинговая система сайта «Фотобратск»

Система, реализованная на сайте фотосервиса «Фотобратск» [11], предусматривает три типа оценок: «+1», «-1» и «0». Клиенты системы осведомлены об использовании рейтинговой системы.

Вес голоса пользователя в представленной системе вычисляется по формуле (5):

$$Weight_k = \begin{cases} s, Rate_k < s \\ \frac{Rate_k}{s} + s, Rate_k \geq s \end{cases}, \quad (5)$$

где s — средний балл публикаций по всем пользователям сайта, а $Rate_k$ — показатель рейтинга клиента.

4.5 Результаты

В результате была разработана рейтинговая система, удоблетворяющая всем заданным условиям.

5 Реализация мобильного приложения для идентификации абонента

В качестве языка программирования для реализации системы был выбран C#. Этот язык позволяет разрабатывать масштабируемые кроссплатформенные мобильные приложения. Для разработки использовалось три

платформы: Microsoft Visual Studio 2013 Ultimate [31], Microsoft Azure и Xamarin. Все три платформы принадлежат корпорации Microsoft.

Схема работы системы продемонстрирована на рис. 1. Пользователь запускает приложение на своём мобильном телефоне. Приложение получает список последних входящих вызовов (с помощью классов *CallLog* [16] и *CallType* [35]) и запрашивает информацию о них у веб-сервера (при наличии подключения к сети Интернет). Веб-сервер обращается к кэшу или базе данных (в зависимости от того, содержится ли нужная информация в закэшированной версии БД) и отправляет полученные данные клиенту. Приложение выводит на экран полученную информацию и переходит в режим ожидания. После возможны два сценария: получение входящего звонка или запись нового варианта описания в базу данных. В первом случае приложение точно так же обращается к веб-серверу с запросом на получение информации о характере входящего звонка. Получив нужные данные, приложение выводит их на экран. Во втором случае клиент обращается к веб-серверу с запросом на запись данных. Веб-сервер передаёт данные базе данных и отправляет киенту ответ с информацией об успешности мероприятия. В обоих случаях после выполнения всех опреаций приложение возвращается в режим ожидание до тех пор, пока его работа не будет прервана пользователем или системой.

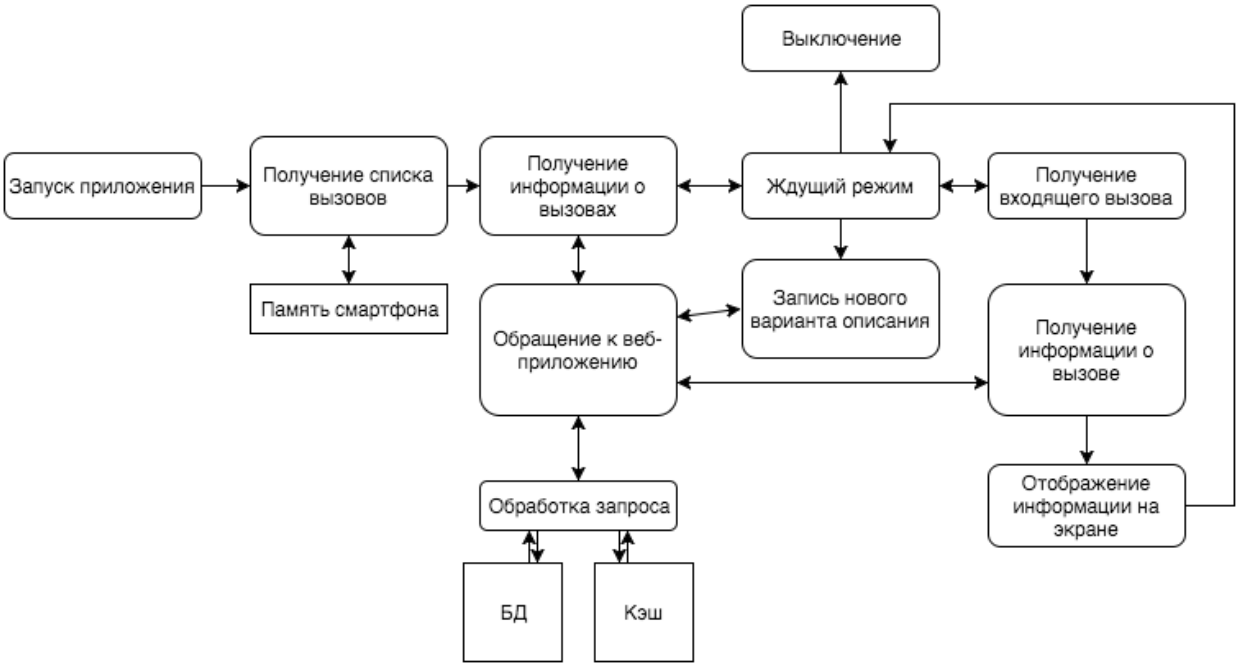

Рис. 1: Схема работы системы

5.1 База данных

База данных, использующаяся в разработанном приложении, состоит из четырёх таблиц: *Regions*, *Users* и *Names*. В таблице *Regions* хранятся телефонные коды городов и регионов длиной от 3 до 7 символов в формате *String*. В таблице *Users* хранятся телефонные номера абонентов, зарегистрированных в системе. Особенность этой таблицы заключается в том, что в ней содержатся записи не только об абонентах, установивших систему на своё мобильное устройство, но и об абонентах, совершавших звонки пользователям системы, но сами её не использовавшие. В таблице также содержится информация о дате и времени последнего звонка, совершённого с этого телефонного номера, рейтинге пользователя и принадлежности номера к определённому региону. Рейтинг и идентификационный номер региона, к которому принадлежит номер, хранятся как вещественные числа в формате *Int*, номер телефона абонента хранится в строковом формате.

В таблице Names хранятся варианты описаний абонентов, присвоенные им другими пользователями, их рейтинги и идентификационный номер абонента, добавившего запись в базу (это позволяет увеличивать его рейтинг в случае, если вариант оказывается популярным среди других пользователей). Варианты описаний хранятся в строковом формате, а рейтинги и идентификационные номера — в числовом.

Для работы с СУБД используется технология ADO.NET Entity Framework, реализующая принципы технологии ORM (object-relational mapping). Это набор классов .NET, позволяющих работать с системами управления базами данных, используя привычный синтаксис языка C# и не применяя язык запросов SQL. Сама база данных организована на платформе Microsoft Azure.

Основные классы, использующиеся для работы с базой данных — *Name*, *Number*, *Region*. Они хранят информацию о добавленных вариантах, пользователях системы и регионах соответственно.

Листинг 1: User

```
public partial class User
    {
        public int Id { get; set; }
        public string PhoneNumber { get; set; }
        public Nullable<int> RegionID { get; set; }
        public Nullable<System.DateTime> LastCall {
          get; set; }
        public int UserRating { get; set; }
    }

...

public partial class Region
```

```
    {
        public int Id { get; set; }
        public string Prefix { get; set; }
        public string RegionName { get; set; }
    }

...

public partial class Name
    {
        public int Id { get; set; }
        public string NickName { get; set; }
        public int NumberUserID { get; set; }
        public int NickNameRating { get; set; }
        public int AddedByUserID { get; set; }
    }
```

5.1.1 Масштабирование базы данных в случае высокой нагрузки на сервер

Для размещения базы данных на облачной платформе при реализации системы использовались мощности платформы Microsoft Azure. Microsoft предоставляет разработчикам доступ к масштабируемым ресурсам — при возникновении такой необходимости можно быстро увеличить объём хранилища и число подключений, оплатив дополнительное пространство.

Платформа Microsoft Azure также предлагает инструменты Elastic Scale [53] для создания пулов баз данных [43]. Разделение базы данных на пулы позволяет повысить производительность системы в целом. Система Microsoft Azure позволяет управлять пулами эластичных баз данных так же, как и другими ресурсами. Создание пула баз данных может быть

экономически выгодно [40] в ситуации, когда созданная разработчиком система использует несколько экземпляров баз данных, при этом количество обращений к ним в большую часть времени характеризуется небольшим показателем, но время от времени возникают «пики» — кратковременный рост количества запросов. Важно, что для каждого экземпляра базы данных такие «пики» приходятся на различные моменты времени — за счёт этой особенности совокупное число обращений к пулу баз данных в единицу времени не превышает заданного разработчиком предела.

Модель Elastic Scale невыгодна для использования на начальных этапах существования CallDialer. В системе действительно возникают «пики» использования (согласно данным панели управления Microsoft Azure, такие всплески активности возникают с 10:00 до 14:00 и с 19:00 до 22:00), однако в перспективе за счёт расширения географии использования системы количество обращений к базе данных окажется распределено во времени более равномерно.

5.2 Веб-приложение на базе ASP.NET

Веб-приложение на базе ASP.NET Web API организовано на платформе Microsoft Azure. Оно отвечает за взаимодействие мобильного приложения на базе Android с системой управления базой данных. Обмен данными между мобильным приложением и веб-приложением осуществляется по протоколу HTTP [3].

Веб-приложение обрабатывает запросы от мобильного приложения (они могут быть двух типов: записать данные в СУБД или запросить данные от СУБД). Кроме того, оно выполняет роль веб-кэша — приложение хранит копию ответа на конкретный запрос в течение 12 часов. Если в это время системе поступает аналогичный запрос, она отправляет ответ, сохранённый в веб-приложении. Это экономит время и уменьшает стоимость работы системы (обращение к базе данных обойдётся системе дороже [47], чем

обращение к закэшированной версии ответа — и во временном, и в денежном измерении). Веб-приложения также отвечает за обработку рейтингов пользователей и вариантов описаний. Все процессы, связанные с обработкой рейтингов, происходят в этом веб-приложении. Это снимает нагрузку с мобильного приложения и ограждает систему от взлома злоумышленниками.

По умолчанию контроллеры ASP.NET Web API поддерживают архитектуру REST. Основные принципы [49] такой архитектуры:

1) независимость от состояния (веб-сервер не должен хранить или использовать информацию о текущем состоянии клиента);

2) кэшируемая архитектура (приложение-клиент может кэшировать информацию на самом устройстве или на промежуточном узле и обращаться к ней, не отправляя запрос на веб-сервер);

3) клиент-серверная архитектура;

4) единый интерфейс (приложение-клиент не в состоянии обратить к источнику данных — например, к базе данных, — самостоятельно. Любое взаимодействие организовано через веб-сервер).

Реализованная система удовлетворяет всем названным требованиям. Действительно, веб-сервер не хранит и не использует информацию о текущем состоянии клиента. Единственные данные, которые приложение-клиент предоставляет веб-серверу с каждым запросом — номер телефона пользователя. Он применяется для регистрации и идентификации пользователя в системе. Кроме того, приложение-клиент кэширует часть информации, получаемой от веб-сервера, в локальном экземпляре базы данных, построенном с использованием технологии SQLite [28]. Клиент обращается к данному экземпляру перед непосредственного обращения к веб-серверу. В системе также реализован единый интерфейс связи с базой данных.

Приложение-клиент не имеет доступа к основной базе данных, хранящейся в «облаке», и любое взаимодействие с ней осуществляется посредством веб-сервера на базе ASP.NET Web API. Таким образом, веб-сервер реализует принципы RESTful.

Краткая схема работы веб-приложения продемонстрирована на рис. 2. Веб-сервер, размещённый в «облаке» функционирует постоянно. После получения запроса сервер определяет, к какому типу относится данный запрос. В случае, если получен запрос на запись информации о телефонном номере в базу данных, веб-сервер извлекает данные, подлежащие изменению, из БД и пересчитывает нужные показатели. Если приложение прислало запрос на получение информации, сначала веб-сервер обращается к закэшированной версии БД. Если нужно информации в ней не содержится, веб-сервер обращается к основной базе данных. После в случае необходимости производится пересчёт рейтинга в базе данных, а затем ответ на запрос отправляется клиенту.

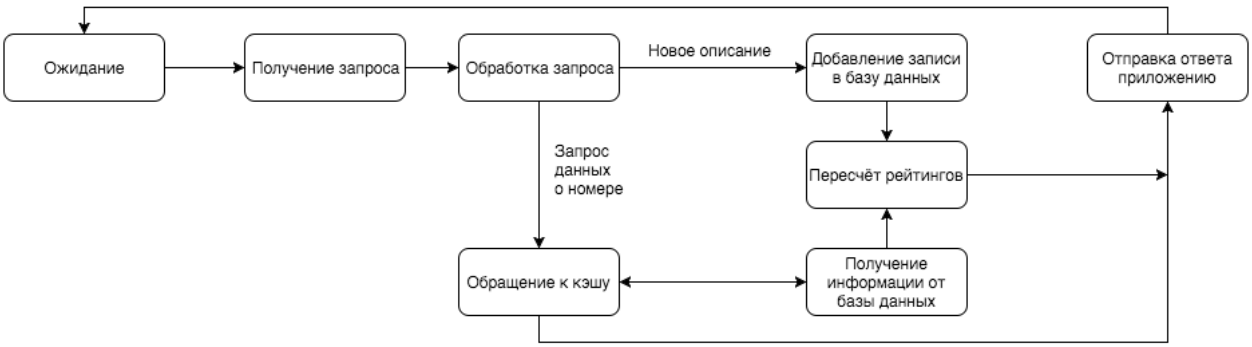

Рис. 2: Схема работы серверной части приложения

5.2.1 Обработка входящего запроса

Веб-приложение на базе Microsoft Azure способно обабатывать запросы двух типов — на получение и запись информации. Связь между мобильным приложением CallDialer и веб-сервером осуществляется по протоколу

HTTP. За обработку запросов отвечает класс-контроллер *NameController*.

Листинг 2: NameController

```
public class NamesController : ApiController {
        ...

        public IEnumerable<CachedName> Get(){...}

        public IHttpActionResult Get(string id) {...}

        public HttpResponseMessage Post(PutName
          name) {...}
    }
```

5.2.2 Пересчёт рейтинга пользователя

Расчёт рейтинга пользователя производится, исходя из рейтингов добавленных им имён. Рейтинг пересчитывается каждый раз, когда клиент системы отправляет со своего мобильного устройства новый вариант описания для конкретного телефонного номера. Рассмотрим несколько ситуаций.

5.2.2.1 Добавление описания для телефонного номера, у которого пока нет ни одного варианта описания

В случае, если пользователь системы отправляет вариант описания для телефонного номера, у которого в пока нет ни одного описания, его рейтинг увеличивается на $0,025$.

5.2.2.2 Добавление описания для телефонного номера, у которого уже есть несколько вариантов описания

Рассмотрим ситуацию, когда пользователь добавляет описание для телефонного номера, у которого уже есть несколько вариантов описания.

В случае, если добавленный пользователем вариант уже содержится в базе данных, и при этом рейтинг этого варианта — выше, чем у всех остальных вариантов, рейтинг пользователя увеличивается на $0,25$. Рейтинг самого варианта при этом пересчитывается в соответствии с формулой (3), определённой в разделе 4.2.

В случае, если для конкретного телефонного номера представлено не менее двух вариантов описаний, и пользователь указал вариант с наименьшим рейтингом, его собственный рейтинг упадёт на $0,25$. Рейтинг сама варианта при этом пересчитывается в соответствии с упомянутой формулой.

В остальных случаях рейтинг клиента системы не изменяется, а рейтинг указанного им варианта пересчитывается в соответствии с упомянутой формулой.

Возможна ситуация, когда телефонный номер переходит от одного абонента к другому. В таком случае пользователь, узнавший о смене первым и указавший новый вариант описания, не получит баллы рейтинга. Однако рейтинг самого описания определяется в том числе и количеством пользователей, приславших аналогичные описания. Таким образом, рейтинг нового описания будет постепенно расти, и в конце концов окажется среди наиболее популярных вариантов. Когда описание попадёт в список пяти самых популярных вариантов описаний, рейтинг добавившего его пользователя увеличится на $0,5$.

5.2.2.3 Сравнение нескольких вариантов описаний

Система определяет схожесть описаний с помощью фонетического алгоритма [39] анализа строк Metaphone [4].

5.2.3 Кэширование базы данных и работа с основным экземпляром базы данных

За работу с закэшированной версией базы данных отвечает класс *CacheController*, реализованный с применением паттерна проектирования Singleton [45]:

Листинг 3: Класс *CacheController*

```
public class CacheController {
        private const int
          TimeInHoursBeforeUpdateNumberName = 12;
        private const int
          TimeInHoursBeforeUpdateRegion = 100;
        private const int
          TimeInHoursBeforeClearCache = 24;
        private DateTime _lastClearCacheTime;

        private static CacheController _instance;

        public static CacheController Instance =>
          _instance ?? (_instance = new
          CacheController());

        public LinkedList<CachedName> CachedNames {
          get; }
```

```
        public LinkedList<CachedRegion>
          CachedRegions { get; }

        private CacheController() {...}

        private void InitRegions() {...}

        public void AddName(string number, string
          name) {...}

        public CachedRegion GetRegion(string number)
          {...}

        public CachedName GetName(string number)
          {...}

        private void ClearCache() {...}
    }
```

За работу с основным экземпляром базы данных отвечает класс *SQLhelper*:

Листинг 4: Класс *SQLhelper*

```
public static class SqLhelper {
        private static readonly myDBEntities Db =
          new myDBEntities();
        public const int DefaultValue = -1;

        private static int? GetNameId(string number)
          {...}
```

```
        private static IEnumerable<Name>
            GetAllNames(string number) {...}

        public static Name GetMostPopularName(string
            number) {...}

        public static Region
            GetRegionByNumber(string number) {...}

        private static int?
            GetRegionIdByNumber(string number) {...}

        public static void AddNewName(string number,
            string name, string numberAdded) {...}

        private static int AddNewUser(string
            phoneNumber) {...}

        public static IEnumerable<Region>
            GetAllRegions() {...}

        public static void UpNewNameRaiting(int
            nameId, int value) {...}

        public static void UpUserRaiting(int userId,
            int value) {...}
    }
```

5.3 Мобильное приложение на базе Android

Мобильное приложение на базе операционной системы Android реализовано в среде Xamarin с использованием .NET-подобной платформы Mono. Приложение состоит из трёх основных экранов (рис. 3): стартовый экран со списком входящих вызовов, экран с информацией о каждом номере и экран для добавления новой записи в базу данных.

5.3.1 Получение информации о входящих вызовах

Информацию о последних входящих вызовах приложение получает из телефонной книжки смартфона с помощью класса *CallLog*, хранящего информацию обо всех совершённых и полученных вызовах.

Получив список последних вызовов, в первую очередь мобильное приложение обращается к локальной базе данных, хранящейся на мобильном устройстве. Её размер в экстремальных случаях не превышает 30 МБ. Эта база данных, реализованная с помощью технологии *SQLite*, хранит информацию о номерах, принадлежащих тому же региону, в котором находится пользователь. Информация о них обновляется каждые 12 часов при наличии подключения к сети Интернет. Если размер локальной базы данных превышает допустимые размеры, из неё исключается информация об абонентах, не совершавших звонки в течение длительного времени (информация о дате и времени последнего звонка хранится в свойстве *LastCall* класса *User*).

Если в локальной версии базы данных не содержится нужной информации, приложение запрашивает данные о соответствующих номерах у веб-сервера. За связь с веб-сервером отвечает вспомогательный класс *WebAPIHelper*. Обмен данными с сервером осуществляется по протоколу HTTP с помощью класса *HttpClient* [54] (библиотека *System.Net.Http* [55]).

Листинг 5: Класс *WebAPIHelper*

```
public class WebAPIHelper
        {
                private string url;
                HttpClient client = new HttpClient();

                public WebAPIHelper(string url) {...}

                public PutName Get(string productId)
                  {...}

                public string Post(PutName obj) {...}
        }
```

Веб-сервер определяет, присутствует ли нужный номер в закэшированной версии базы данных. При его отсутствии веб-приложение обращается к самой базе данных. Закэшированная версия базы данных хранится в течение 12 часов.

5.3.2 Отслеживание входящих вызовов

Отслеживание входящих вызовов осуществляется при помощи компонента *Broadcast Receiver* [15]. За обработку выходящих вызовов отвечает класс *Telefon*, наследующий класс *Broadcast Receiver*.

5.3.3 Визуализация результатов

Разработанная система предусматривает несколько вариантов визуализации полученных результатов. Рассмотрим два случая.

Когда пользователь открывает приложение, то видит список, состоящий из 20 последних звонков. В каждой строке содержится информация

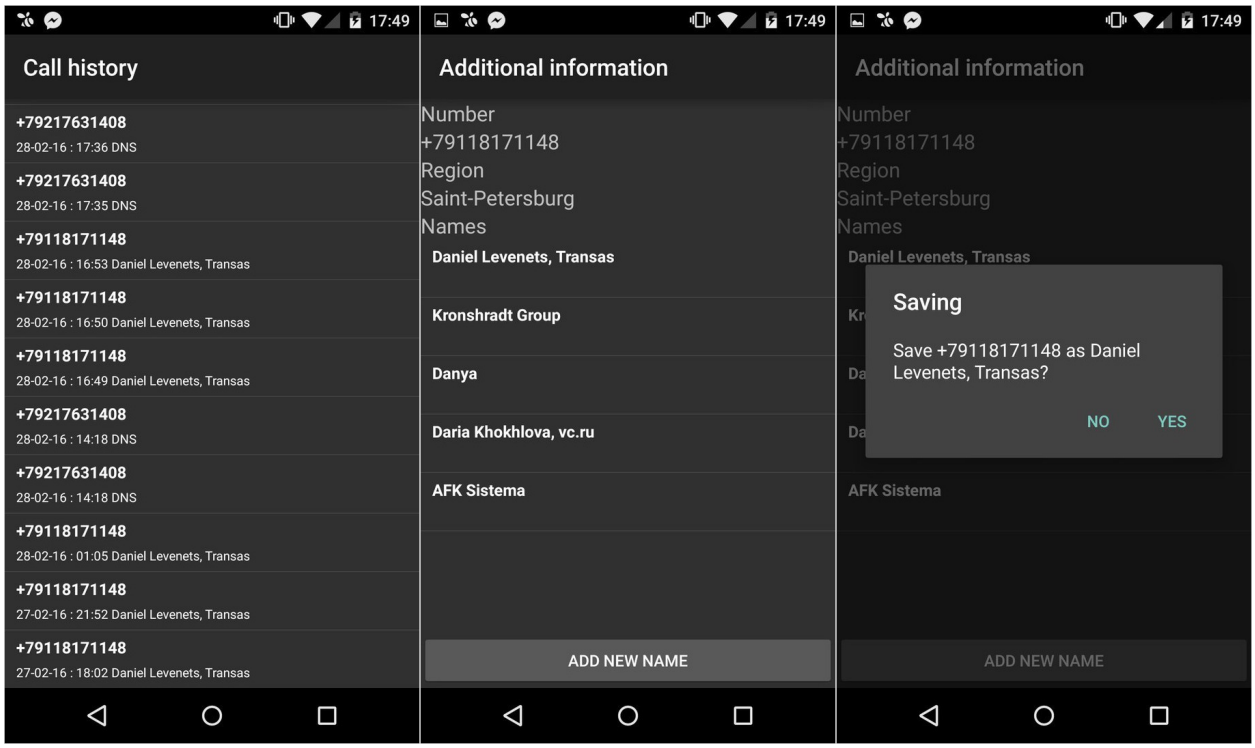

Рис. 3: Основные экраны мобильного приложения

о дате и времени звонка, номер телефона звонившего, а также наиболее популярный вариант описания, предложенный пользователями системы. При нажатии на любой из пунктов приложение открывает новый экран, на котором пользователь может просмотреть список из пяти наиболее популярных вариантов описаний, оставленных другими клиентами (при их наличии), а также отправить собственный вариант описания. Общий вид приложения продемонстрирован на рис. 3.

В момент получения входящего вызова на экран выводится Toast-уведомление [17], содержащее информацию о характере входящего вызова (первый вариант описания из базы данных). Пример уведомления продемонстрирован на рис. 4.

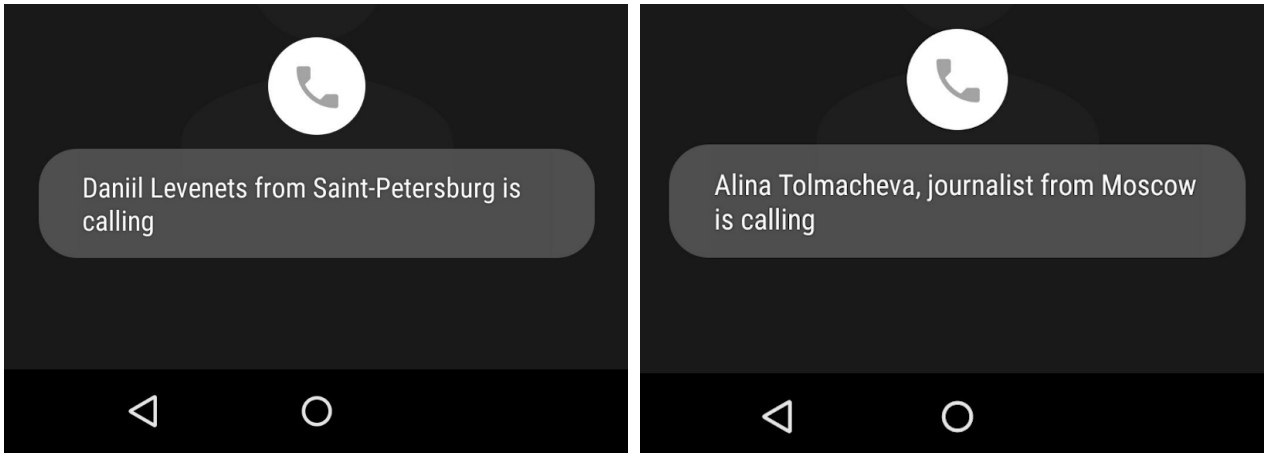

Рис. 4: Примеры Toast-уведомлений

5.4 Результаты

В рамках данной квалификационной работы была разработана система, предоставляющая информацию о характере и цели входящего звонка и удовлетворяющая всем заданным требованиям. Система способна работать в облегчённом режиме без доступа к сети Интернет (за счёт использования локального источника данных). Кроме того, она позволяет уточнять и ранжировать информацию, основываясь на поступающей от пользователей информации — за счёт разработанной рейтинговой системы.

Технология также позволяет получить информацию о входящем звонке за короткий промежуток времени. Для подтверждения этого тезиса автором работы была проведена серия экспериментов. Сначала на смартфон участника эксперимента поступал звонок с телефонного номера, варианты описаний для которого содержались в локальной базе данных. Среднее время вывода информации на экран для 100 звонков составило $0,58$ секунды. Затем на смартфон поступали звонки с телефонного номера, содержащегося в основном экземпляре базы данных и отсутствующего в локальном экземпляре базы данных на устройстве пользователя. Среднее время вывода информации на экран для 100 звонков составило $1,64$ секунды. После

на смартфон поступали звонки с телефонного номера, не содержащегося ни в одном из экземпляров баз данных. Среднее время вывода информации на экран составило $1,97$ секунды.

С расширением базы данных время вывода информации может меняться. Дальнейшее исследование выходит за рамки задач настоящей выпускной квалификационной работы.

Заключение

В данной работе рассмотрен алгоритм ранжирования данных, учитывающий обратную связь от пользователей системы. Также представлена технологическая платформа, позволяющая идентифицировать абонента и цель его звонка по номеру телефона звонящего, удовлетворяющая всем заданным требованиям.

Результаты работы были представлены на нескольких конференциях «Наука настоящего и будущего 2016», «СПИСОК 2016», «Современные технологии в теори практике программирования 2016».

Все цели работы достигнуты.

Список литературы

[1] Андере, К. Truecaller: How it works and what you need to know about it. Режим доступа [проверено 7.05.2016]. URL: https://www.linkedin.com/pulse/truecaller-how-works-what-you-need-know-keith-andere

[2] Баранвал, В. Where does Truecaller get its data. Режим доступа [проверено 7.05.2016]. URL: https://www.quora.com/Where-does-Truecaller-get-its-data-It-doesnt-seem-to-work-for-me-but-if-it-did-work-wouldnt-that-raise-privacy-concerns/answer/Vipul-Baranwal

[3] Бернерс-Ли, Т., Филдинг, Р., Фристик, Г. Hypertext Transfer Protocol — HTTP/1.0. Режим доступа [проверено 7.05.2016]. URL: https://tools.ietf.org/html/rfc1945

[4] Лоуренс, Ф. Hanging on the Metaphone. Computer Language, Vol. 7, No. 12, 1990.

[5] Майер, Д. Theory of Relational Databases. Rockville, MD: Computer Science Press, 1983. P. 637.

[6] Нейгел, К., Ивьен, Б., Глинн, Д., Уотсон, К., Скиннер, М., Джонс, А. C# 5.0 и платформа .NET 4.5 для профессионалов. М.: Издательство Вильямс, 2014.

[7] Парамонов, С. Расшифровываем формулу Хабра-рейтинга или восстановление функциональных зависимостей по эмпирическим данным. Режим доступа [проверено 7.05.2016]. URL: https://habrahabr.ru/post/249375/

[8] Просто о корпоративном IaaS: что это, для кого, и как оплачивается. Режим доступа [проверено 7.05.2016]. URL: https://habrahabr.ru/company/it-grad/blog/257295/

[9] Сайт проекта «Жёлтые страницы». Режим доступа [проверено 7.05.2016]. URL: http://www.yp.ru/

[10] Сайт проекта «Код телефона». Режим доступа [проверено 7.05.2016]. URL: http://www.kodtelefona.ru/region_ru

[11] Сайт проекта «Фотобратск». Режим доступа [проверено 7.05.2016]. URL: http://photobratsk.ru/

[12] Сайт проекта «Фотогорький». Режим доступа [проверено 7.05.2016]. URL: http://www.photogorky.ru/

[13] Сайт проекта «Хабрахабр». Режим доступа [проверено 7.05.2016]. URL: https://habrahabr.ru/

[14] Сайт проекта Android. Режим доступа [проверено 7.05.2016]. URL: https://www.android.com/intl/ru_ru/

[15] Сайт проекта Android Developers. BroadcastReceiver. Режим доступа [проверено 7.05.2016]. URL: http://developer.android.com/reference/android/content/BroadcastReceiver.html

[16] Сайт проекта Android Developers. CallLog. Режим доступа [проверено 7.05.2016]. URL: http://developer.android.com/reference/android/provider/CallLog.html

[17] Сайт проекта Android Developers. Toast. Режим доступа [проверено 7.05.2016]. URL: http://developer.android.com/intl/ru/guide/topics/ui/notifiers/toasts.html

[18] Сайт проекта ASP.NET. Режим доступа [проверено 7.05.2016]. URL: http://www.asp.net/

[19] Сайт проекта Blackberry OS. Режим доступа [проверено 7.05.2016]. URL: http://global.blackberry.com/en/software/smartphones/blackberry-10-os.html

[20] Сайт проекта Google Play. Contactive. Режим доступа [проверено 7.05.2016]. URL: https://play.google.com/store/apps/details?id=com.contactive&hl=ru

[21] Сайт проекта GSM Inform. Режим доступа [проверено 7.05.2016]. URL: https://gsm-inform.ru/info/

[22] Сайт проекта Indexmain. Режим доступа [проверено 7.05.2016]. URL: http://indexmain.ru/phone/ru

[23] Сайт проекта iOS. Режим доступа [проверено 7.05.2016]. URL: http://www.apple.com/ru/ios/

[24] Сайт проекта Microsoft Azure. Режим доступа [проверено 7.05.2016]. URL: https://azure.microsoft.com/ru-ru/

[25] Сайт проекта Mono Develop. Режим доступа [проверено 7.05.2016]. URL: http://www.mono-project.com/

[26] Сайт проекта MSDN. Язык Entity SQL. Режим доступа [проверено 7.05.2016]. URL: https://msdn.microsoft.com/ru-ru/library/bb399560(v=vs.110).aspx

[27] Сайт проекта SimilarWeb. Режим доступа [проверено 7.05.2016]. URL: https://www.similarweb.com/

[28] Сайт проекта SQLite. Режим доступа [проверено 7.05.2016]. URL: https://www.sqlite.org/

[29] Сайт проекта Tellows. Режим доступа [проверено 7.05.2016]. URL: http://www.tellows.ru/

[30] Сайт проекта Truecaller. Режим доступа [проверено 7.05.2016]. URL: https://www.truecaller.com/ru

[31] Сайт проекта Visual Studio 2013. Режим доступа [проверено 7.05.2016]. URL: https://www.microsoft.com/ru-ru/SoftMicrosoft/VisualStudio2013.aspx

[32] Сайт проекта Whoscall. Режим доступа [проверено 7.05.2016]. URL: https://whoscall.com/en-US/

[33] Сайт проекта Windows Phone. Режим доступа [проверено 7.05.2016]. URL: https://www.microsoft.com/ru-ru/windows/phones

[34] Сайт проекта Xamarin. Режим доступа [проверено 7.05.2016]. URL: https://www.xamarin.com/

[35] Сайт проекта Xamarin Developers. CallType. Режим доступа [проверено 7.05.2016]. URL: https://developer.xamarin.com/api/type/Android.Provider.CallType/

[36] Сайт проекта 2ГИС. Режим доступа [проверено 7.05.2016]. URL: https://2gis.ru/

[37] Сайт проекта 2GIS Dialer. Режим доступа [проверено 7.05.2016]. URL: https://apps.2gis.ru/dialer

[38] Скарлетт, К. Облачные стандарты: средства взаимодействия приложений в облаке. Режим доступа [проверено 7.05.2016]. URL: https://www.ibm.com/developerworks/ru/library/cl-tools-to-ensure-cloud-application-interoperability/

[39] Сметанин, Н. Фонетические алгоритмы. Режим доступа [проверено 7.05.2016]. URL: https://habrahabr.ru/post/114947/

[40] Стайн, С. Когда следует использовать пул эластичных баз данных. Режим доступа [проверено 7.05.2016]. URL: https://azure.microsoft.com/ru-ru/documentation/articles/sql-database-elastic-pool-guidance/

[41] Фаулер, М. Unit of Work. Режим доступа [проверено 7.05.2016]. URL: http://martinfowler.com/eaaCatalog/unitOfWork.html

[42] Филдинг, Р. Representational State Transfer (REST). Режим доступа [проверено 7.05.2016]. URL: http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[43] Хига, С. Создание пула эластичных баз данных на портале Azure. Режим доступа [проверено 7.05.2016]. URL: https://azure.microsoft.com/ru-ru/documentation/articles/sql-database-elastic-pool-create-portal/

[44] Хохлова Д. Д., Григорьев Д. А. Разработка системы идентификации абонентов по номеру телефона. Материалы научно-практической конференции студентов, аспирантов и молодых учёных Северо-Запада «Современные технологии в теории и практике программирования». — 2016. — С. 28–29.

[45] Шаллоуей А., Тротт Д. Шаблоны проектирования. Новый подход к объектно-ориентированному анализу и проектированию. М.: Издательство Вильямс, 2002.

[46] Шелёхин, А. Подробно о Xamarin. Режим доступа [проверено 7.05.2016]. URL: https://habrahabr.ru/post/188130/

[47] Шинкаренко, В.И. Зависимость временной эффективности алгоритмов и программ обработки больших объемов данных от их кэширования. Математические машины и системы. 2007. №2. С. 43–55.

[48] Шторкин, С. Средневзвешенная система голосования. Режим доступа [проверено 7.05.2016]. URL: https://habrahabr.ru/post/63664/

[49] Щербин, И. RESTful API для сервера — делаем правильно. Режим доступа [проверено 7.05.2016]. URL: https://habrahabr.ru/post/144011/

[50] iOS 9 tells you who that unknown caller is. Режим доступа [проверено 7.05.2016]. URL: http://venturebeat.com/2015/09/16/ios-9-tells-you-who-that-unknown-caller-is/

[51] Gartner Says Worldwide Smartphone Sales Grew 9.7 Percent in Fourth Quarter of 2015. Режим доступа [проверено 7.05.2016]. URL: http://www.gartner.com/newsroom/id/3215217

[52] Google buys secretive Silicon Valley start-up, Android. Режим доступа [проверено 7.05.2016]. URL: http://www.siliconbeat.com/entries/2005/08/17/google_buys_secretive_silicon_v

[53] Microsoft Azure: Get started with Elastic Database tools. Режим доступа [проверено 7.05.2016]. URL: https://azure.microsoft.com/en-us/documentation/articles/sql-database-elastic-scale-get-started/

[54] MSDN: Класс HttpClient. Режим доступа [проверено 7.05.2016]. URL: https://msdn.microsoft.com/ru-ru/library/system.net.http.httpclient.aspx

[55] MSDN: System.Net.Http — пространство имен. Режим доступа [проверено 7.05.2016]. URL: https://msdn.microsoft.com/ru-ru/library/system.net.http.aspx

[56] Wikipedia: ADO.NET. Режим доступа [проверено 7.05.2016]. URL: https://ru.wikipedia.org/wiki/ADO.NET

[57] Wikipedia: ADO.NET Entity Framework. Режим доступа [проверено 7.05.2016]. URL: https://ru.wikipedia.org/wiki/ADO.NET_Entity_Framework

[58] Wikipedia: API. Режим доступа [проверено 7.05.2016]. URL: https://ru.wikipedia.org/wiki/API

[59] Wikipedia: Cloud Computing. Режим доступа [проверено 7.05.2016]. URL: https://en.wikipedia.org/wiki/Cloud_computing

[60] Wikipedia: ORM. Режим доступа [проверено 7.05.2016]. URL: https://ru.wikipedia.org/wiki/ORM

[61] Wikipedia: Platform as a Service. Режим доступа [проверено 7.05.2016]. URL: https://ru.wikipedia.org/wiki/Platform_as_a_service

[62] Wikipedia: SaaS. Режим доступа [проверено 7.05.2016]. URL: https://ru.wikipedia.org/wiki/SaaS

[63] .NET Framework. Режим доступа [проверено 7.05.2016]. URL: https://www.microsoft.com/net/default.aspx

[64] 200 million users strong and new tagging feature. Сайт проекта Truecaller. Режим доступа [проверено 7.05.2016]. URL: http://blog.truecaller.com/2015/11/05/200-million-users-strong-and-new-tagging-feature/