УДК 004.75+004.031.2+004.032.24

МЕХАНИЗМЫ ПРИМЕНЕНИЯ МОБИЛЬНЫХ УСТРОЙСТВ ДЛЯ ЗАДАЧ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ

Н. Р. Низамов[1], И. С. Шахова[2]

Высшая школа информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета

[1]nizamov.nurshat@gmail.com, [2]is@it.kfu.ru

Аннотация

Описана система, реализующая механизмы применения мобильных устройств для операционной системы Android в рамках решения задач, требующих использования распределенных вычислений. Особое внимание уделено компонентам данной системы, отвечающим за управление задачами и распределение ресурсов.

Ключевые слова: *распределенные вычисления, мобильные приложения, Android, мобильные устройства*

ВВЕДЕНИЕ

В основе распределенных вычислений лежит принцип декомпозиции исходного объемного набора данных на более мелкие и, соответственно, менее требовательные к ресурсам вычислительных систем пакеты данных. Данные пакеты распределяются по удаленным устройствам в сети, где производятся вычисления. После завершения вычислений на удаленных устройствах полученный результат отправляется на центральный компьютер, где производится объединение результатов и, при необходимости, запуск следующей итерации распределенного вычисления.

Одним из недостатков, наиболее препятствующих широкому распространению механизмов распределенных вычислений, является невозможность оперативной замены алгоритма вычисления на удаленных устройствах. Данная проблема особенно актуальна для случаев, когда для вычисления используются сторонние устройства, к которым у вычисляющего нет прямого доступа.

Согласно докладу ассоциации GSMA [1], представляющей интересы операторов мобильной связи по всему миру, в 2017 году 66% населения Земли пользовалось мобильными устройствами, где 57% приходилось на долю смартфонов. В то же время, согласно другому исследованию**,** проведенному аналитической компанией Counterpoint [2], только 26% людей пользуются смартфонами более 7 часов день. В свою очередь, наиболее распространенной мобильной операционной системой на апрель 2019 года остается операционная система Android. Согласно статистике, собранной компанией StatCounter GlobalStats [3], на долю Android-устройств приходится 75% устройств.

Исходя из этих данных, можно сделать вывод, что смартфоны имеют широкое распространение, однако, в то же время, большую часть суток они не активны, значит, могут быть использованы во время простоя для решения сторонних задач. Их растущая производительность (согласно аналитике, представленной на сайте Android Authority, только за период с 2011 по 2015 годы производительность смартфонов выросла чуть менее чем в 5 раз [4]) позволяет рассматривать мобильные устройства в качестве базовых устройств для задач распределенных вычислений.

Приняв во внимание потенциал систем, основанных на распределенных вычислениях, а также проблему, препятствующую их распространению, было принято решение разработать систему, основанную на мобильных устройствах, с возможностью изменения алгоритма вычисления без необходимости переустановки приложения.

ОБЩИЕ ТРЕБОВАНИЯ К СИСТЕМЕ

В качестве основных требований, выполнение которых необходимо для достижения поставленной цели, были выделены следующие:

1. В системе должен быть реализован интерфейс добавления заданий на вычисление. Задание должно состоять из алгоритма, представляющего собой скомпилированный java-класс, а также данных в формате JSON, относительно которых будет производиться вычисление.
2. Должна быть реализована система цепочек задач, при которой результирующие данные одного вычисления становятся входными данными следующего.

3. Система должна позволять изменять алгоритм вычисления без переустановки приложения.
4. В системе должен быть реализован функционал слежения за ходом вычисления. Такие события, как чрезмерное расходование ресурсов на стороне мобильного приложения, появление ошибок во время вычисления, недоступность устройств в течение определенного времени, должны отслеживаться.
5. Система должна уметь перераспределять данные при выбывании одного из вычисляющих устройств из задания.
6. Система должна оценивать производительность устройств и распределять данные в первую очередь по наиболее производительным устройствам.
7. Результат работы должен выводиться на экран пользователя в виде документа с возможностью загрузить его на устройство.

Общий принцип работы системы представлен на рисунке 1.

ПРИНЦИП РАБОТЫ МОБИЛЬНОГО ПРИЛОЖЕНИЯ

Одним из ключевых компонентов разрабатываемой системы является мобильное приложение. Оно представляет собой вычислительную единицу системы и позволяет производить параллельные вычисления. Упрощенная UML-диаграмма компонентов приложения представлена на рисунке 2.

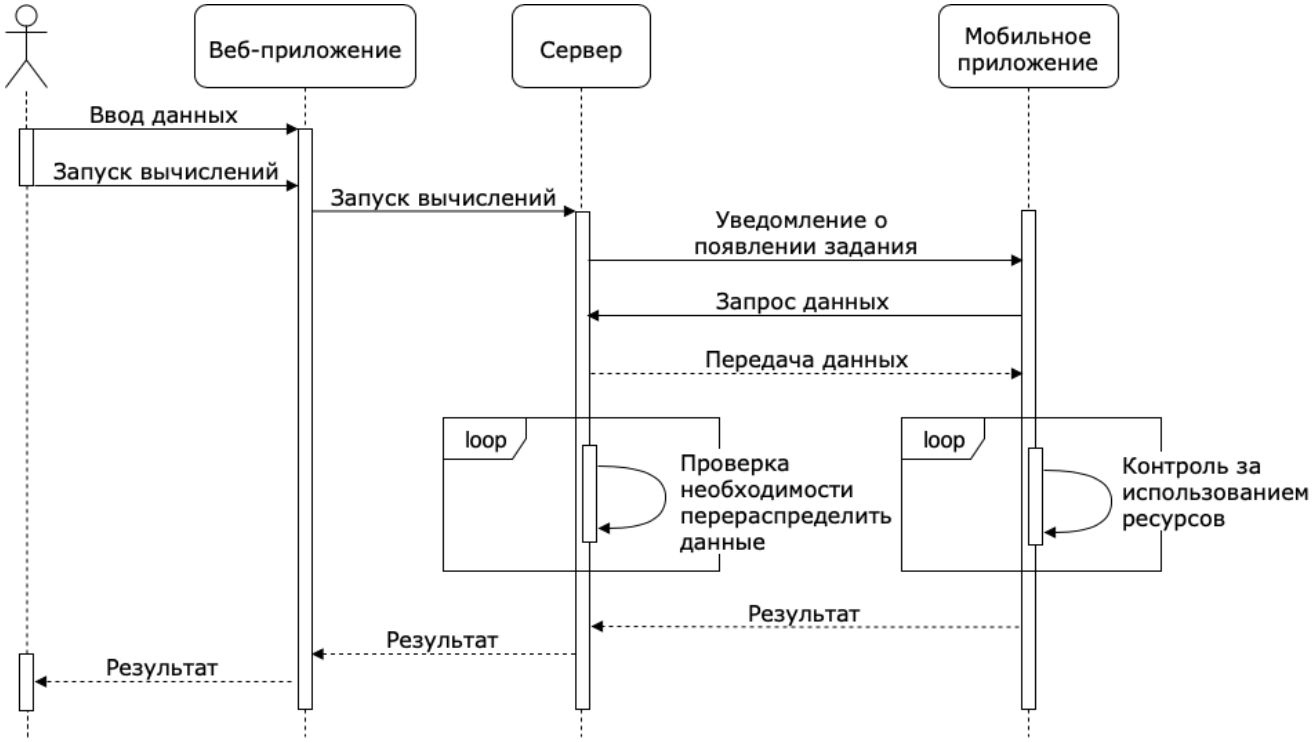

Рис. 1. Концептуальная схема работы системы

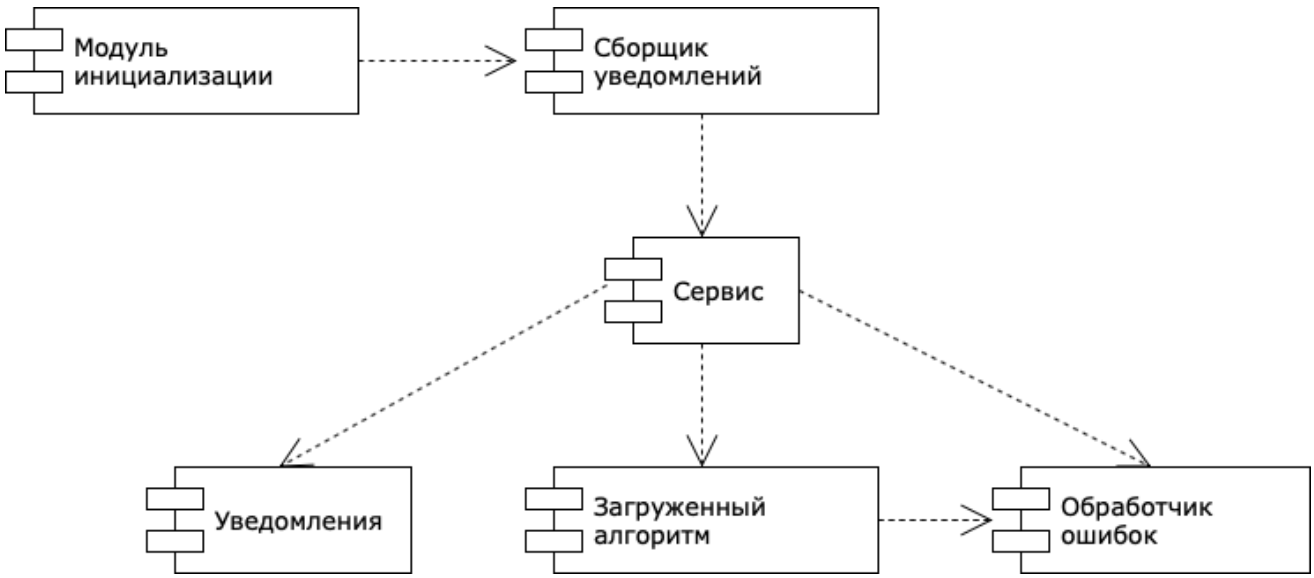

Рис. 2. Общее представление компонентов системы

Сразу после запуска приложение регистрирует несколько интент-фильтров. Интент-фильтр – это механизм, позволяющий приложению получать уведомления о системных событиях. В данном случае фильтры регистрируются на такие события, как:

1. Прием push-уведомлений – необходим для получения информации о появлении нового задания;
2. Событие включения устройства – необходим для запуска фонового сервиса, позволяющего уведомлять сервер о доступности устройства для вычисления на нем;
3. События изменения состояния сети – необходимы для своевременного реагирования на события отключения от интернета. Без этого события система продолжала бы вычисления даже при недоступности интернета, однако результаты вычисления были бы не востребованы, так как сервер настроен таким образом, что, распознав недоступное устройство, перераспределяет данные, а устройство переводит в состояние «недоступен».

Далее, после регистрации фильтров, система устанавливает соединение с платформой Firebase Cloud Messaging, обеспечивающей надежное и энергоэффективное соединение между сервером и устройствами и позволяющее получать уведомления. Успешное соединение с данной платформой сопровождается выдачей уникального токена для устройства, используя который, сервер способен отправлять уведомления конкретному устройству.

После успешного получения токена, приложение запрашивает идентификационный номер устройства. Полученный идентификатор уникален и статичен для каждого устройства, что позволит вести учет на сервере. После его получения все собранные данные отправляются на сервер, где происходит регистрация нового устройства.

После успешной регистрации запускается фоновый сервис, который с определенной периодичностью начинает уведомлять сервер о состоянии устройства – занят, свободен.

С момента инициализации приложение находится в режиме ожидания. Триггером для выхода из этого состояния служит push-уведомление, отправленное с сервера. Данное уведомление содержит в себе только одну строку данных – идентификационный номер задания. Полученный идентификатор передается в сервис, где происходит дальнейшая обработка.

Получив идентификационный номер, сервис в первую очередь проверяет

флаг, отвечающий за занятость. Данная проверка необходима по причине того, что уведомления могут приходить с опозданием. В результате проверки система отдаст предпочтение уже начатому вычислению и проигнорирует последующие.

В случае незанятости системы выполнение скрипта продолжается, и производится проверка наличия алгоритма для поступившей задачи. Система организована таким образом, что всегда хранит последний алгоритм вычисления. Это необходимо для ускорения работы в рамках сценариев, когда приложение получает новое задание с тем же идентификационным номером. В описанном случае повторная загрузка алгоритма не представляет необходимости. В случае же, когда задание с таким идентификационным номером пришло впервые, на сервер отправляются два параллельных запроса для получения алгоритма и данных для вычисления.

Если данные получены успешно, начинается загрузка байт-кода класса с реализацией алгоритма в систему. В случае успешного приведения загруженного класса к типу интерфейса в загруженный алгоритм передаются данные для вычисления, а также слушатель, через который данный класс может отправлять уведомления о своем состоянии. И, наконец, после передачи всех необходимых данных происходит запуск вычислений.

После завершения вычисления алгоритм передает результат слушателю, и, если данные не пусты, они отправляются на сервер, а приложение переводится в состояние «свободен».

Весь процесс, от начала загрузки данных до завершения вычислений, находится в блоке кода Rx, значит, информация о любой ошибке, случившаяся в этот период, будет передана в соответствующий блок.

При обнаружении таких ситуаций вычисление останавливается, а на сервер передается соответствующий сигнал, сообщающий, что во время вычисления задания с определенным идентификационным номером произошла ошибка. Сервер ведет учет сообщений об ошибках и в случае их массовости прекращает вычисления по данному заданию.

Для исключения чрезмерного потребления ресурсов было принято решение установить контроль за этим параметром. Контролируются два основных показателя – потребление оперативной памяти и загруженность центрального

процессора. Контроль производится циклично с интервалом в одну секунду. При обнаружении состояния нехватки памяти, если оно длится более пяти секунд, приложение фиксирует исключение и останавливает вычисление. Параллельно с контролем использования памяти контролируется и использование центрального процессора. Если среднее значение использования процессора превышает 90% в течение 5 секунд, вычисление останавливается и фиксируется соответствующее исключение.

СЕРВЕРНАЯ ЧАСТЬ ПРОГРАММНОГО РЕШЕНИЯ

Следующим ключевым компонентом рассматриваемой системы является сервер. Данный компонент принимает задания от пользователей, запускает вычисления, координирует устройства, а также распределяет и, при необходимости, перераспределяет данные.

Работа системы в целом и серверной части в частности начинается с добавления задания в список выполнения.

В первую очередь на сервер приходит запрос на добавление нового задания с информацией о названии и описании задания. После получения параметров метод проверяет их на заполненность, и, если хотя бы одна из переменных пуста, сервер возвращает ответ с кодом 400, обозначающим, что произошла ошибка на стороне клиента. Если же параметры переданы корректно, происходит проверка уникальности названия. В случае, если в базе данных уже имеется задание с таким названием, клиенту возвращается ответ с кодом 400, иначе переданные данные записываются в базу данных, и сервер возвращает клиенту идентификатор добавленного задания, присваивая ответу код 200.

С целью обеспечения минимального участия в работе системы владельца устройства, на котором оно установлено, была поставлена задача максимальной автоматизации всех этапов взаимодействия сервера с приложением. Первым этапом в рамках выполнения данной задачи стала автоматизация процесса регистрации. В случае разрабатываемой системы, когда личность владельца не имеет значения, удалось добиться полной автоматизации. Схематично данный процесс представлен на рисунке 3.

Для регистрации нового устройства был разработан метод, принимающий на вход следующий набор параметров:

- token – идентификатор Firebase, необходимый для дальнейшей отправки уведомлений на устройство;
- user_id – идентификатор устройства, статичный параметр, который служит для распознавания устройства;
- model – модель устройства, которая используется для определения мощности подключаемого устройства.

При вызове метода подключения устройства проверяется наличие вышеописанных параметров, и, если данные переданы корректно, начинается процесс регистрация устройства. В первую очередь проверяется наличие полученного идентификатора устройства в базе данных, и, в случае обнаружения записи, регистрация ограничивается заменой токена Firebase. Если же запись не обнаружена, запускается процесс определения производительности устройства: посылается запрос в платформу GeekBanch с указанием модели устройства. Данная платформа содержит в себе список результатов тестирования устройств на производительность, при этом тестируются мощность как отдельных компонентов (центральный процессор, видеокарта), так и общая производительность устройства. В качестве результата принимается среднее арифметическое 25 последних тестов. После получения оценки производительности устройства данные записываются в базу данных, и на устройство передается ответ с кодом 200, означающий, что регистрация прошла успешно.

Рис. 3. Блок-схема процесса подключения новых устройств

Одной из ключевых функций системы является распределение данных. Активация данной функции происходит после запроса клиентского приложения, а сами запросы могут быть спровоцированы двумя событиями:

1. Запуск вычисления новой задачи;
2. Система обнаружила недоступное устройство, и необходимо перераспределить освободившиеся данные.

В обоих случаях распределение данных происходит следующим образом:

1. В ответ на запрос приложения из базы данных запрашивается количество доступных устройств и нераспределенных данных;
2. Количество неиспользованных данных делится на текущее количество доступных устройств;
3. Полученное количество данных компонуется и преобразуется в json-массив;
4. Получившийся массив посылается на устройство, а данные помечаются в базе данных как использованные.

Весь список исключений, которые возможны при работе системы, можно разделить на две группы:

1. Ошибки, происходящие по вине автора задания: несоблюдение соглашений по структурированию данных для вычисления, ошибки, допущенные при написании алгоритма вычисления;
2. Ошибки на стороне приложения: перерасход ресурсов устройства, ошибки, выданные алгоритмом вычисления, невозможность приведения алгоритма к классу интерфейса, связанная с несоблюдением соглашения о написании алгоритма вычисления.

В первом случае ошибки обнаруживаются на стороне сервера, и ответом на них служат удаление всех данных, ранее загруженных от пользователя, и передача сообщения с описанием возникшей проблемы. Во втором случае сервер получает лишь уведомление от приложения и реагирует на него, исходя из полученных данных. Вне зависимости от типа ошибки, данные, ранее переданные устройству, переводятся в состояние «не использовано», а само устройство помечается в базе данных как непригодное для данного задания. Также в базу данных вносится запись о произошедшей ошибке, и, если количество ошибок превысит заданный лимит, задание будет исключено из вычисления и инициатору вычислений отобразится соответствующее сообщение.

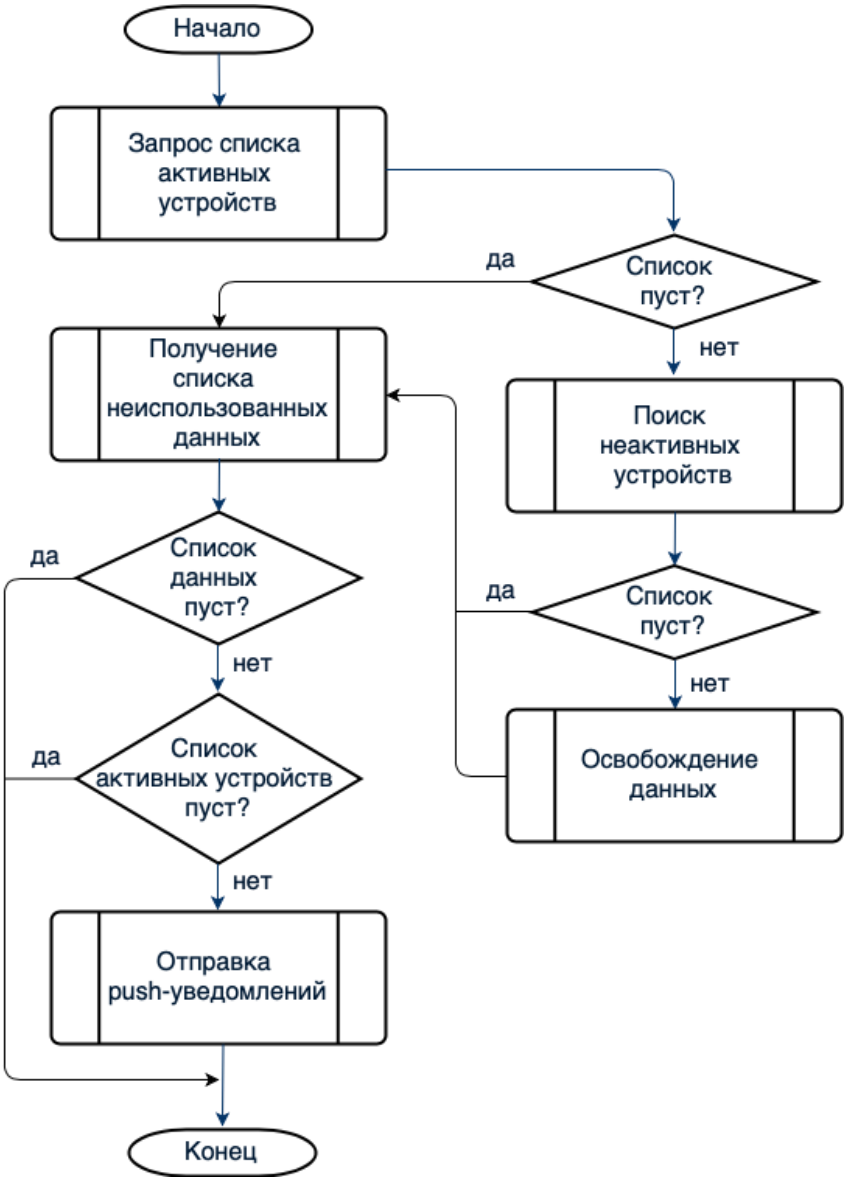

Рис. 4. Блок-схема отслеживания процесса вычисления

Во время вычисления заданий возможны такие случаи, когда устройство, не завершив вычисления, отключается от системы: у устройства сел аккумулятор, оно было внезапно отключено или же вышло за пределы зоны действия интернета. Такое поведение может стать причиной незавершенных вычислений по тому или иному заданию, что недопустимо. В целях избежания данной ситуации был реализован механизм слежения за вычислениями на протяжении всего процесса. Блок-схема данного процесса приведена на рисунке 4.

Процесс работы механизма может быть разделен на два подпроцесса: первый контролирует доступность устройств, второй – наличие неиспользован-

ных данных для вычисления. Сразу после запуска скрипт посылает запрос в базу данных для получения списка активных устройств: в случае, если данный список не пуст, происходит проверка времени их последней активности, и, если активность зафиксирована более полутора минут назад, устройство помечается как неактивное, а выделенные данные помечаются как неиспользованные. Далее в базу данных отправляется запрос на получение списка неиспользованных данных: в случае, если этот список не пуст и не пуст список ранее полученных активных устройств, устройствам отправляются пуш-уведомления с идентификатором задания, которому принадлежат обнаруженные данные. Далее работа системы протекает по стандартному алгоритму, который был описан ранее.

ЗАКЛЮЧЕНИЕ

В статье предложены механизмы применения мобильных устройств под управлением операционной системы Android для задач распределенных вычислений. Предложенные механизмы интегрированы в систему, состоящую из серверной части, реализующей алгоритмы управления задачами и распределения ресурсов, и мобильного приложения для операционной системы Android, ответственного за проведение вычислений. Приведенные механизмы могут быть применены для решения широкого спектра задач, требующих использования распределенных вычислений.

СПИСОК ЛИТЕРАТУРЫ

1. *The Mobile Economy 2018.* URL: https://www.gsma.com/mobile economy/wp-content/uploads/2018/05/The-Mobile-Economy-2018.pdf.

2. *Almost Half of Smartphone Users Spend More Than 5 Hours A Day on Their Mobile Device.* URL: https://www.counterpointresearch.com/ almost-half-of-smartphone-users-spend-more-than-5-hours-a-day-on-their-mobile-device.

3. *Mobile Operating System Market Share Worldwide.* URL: http://gs. statcounter.com/os-market-share/mobile/worldwide.

4. *How far we've come: a look at smartphone performance over the past 7 years*. URL: https://www.androidauthority.com/smartphone-performance-improvements- timeline-626109.

MECHANISMS FOR USING MOBILE DEVICES IN DISTRIBUTED COMPUTING

N. R. Nizamov[1], I. S. Shakhova[2]

Higher School of Information Technologies and Intelligent Systems, Kazan (Volga region) Federal University

[1]nizamov.nurshat@gmail.com, [2]is@it.kfu.ru

Abstract

The paper is aimed to describe a system with some mechanisms for using mobile devices in distributed computing. Emphasis is placed on components of the system which control tasks and distribute resources.

Keywords: *distributed computing, mobile applications, Android, mobile devices*

REFERENCES

1. *The Mobile Economy 2018.* URL: https://www.gsma.com/mobile economy/wp-content/uploads/2018/05/The-Mobile-Economy-2018.pdf.

2. *Almost Half of Smartphone Users Spend More Than 5 Hours A Day on Their Mobile Device.* URL: https://www.counterpointresearch.com/almost-half-of-smartphone-users-spend-more-than-5-hours-a-day-on-their-mobile-device.

3. *Mobile Operating System Market Share Worldwide.* URL: http://gs. statcounter. com/os-market-share/mobile/worldwide.

4. *How far we've come: a look at smartphone performance over the past 7 years*. URL: https://www.androidauthority.com/smartphone-performance-improvements- timeline-626109.

СВЕДЕНИЯ ОБ АВТОРАХ

НИЗАМОВ Нуршат Рушанович – магистр Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета по направлению «Программная инженерия».

Nurshat Rushanovich NIZAMOV, Master of Science in Software Engineering from the Higher School of Information Technologies and Intelligent Systems, Kazan (Volga region) Federal University.

email: nizamov.nurshat@gmail.com

ШАХОВА Ирина Сергеевна – ассистент кафедры программной инженерии Высшей школы информационных технологий и интеллектуальных систем Казанского федерального университета. Сфера научных интересов – мобильные приложения, цифровые образовательные системы, индивидуализация образования, мобильное обучение.

Irina Sergeevna SHAKHOVA – teacher of the Higher School of Information Technologies and Intelligent Systems, Kazan Federal University. Research interests include mobile applications, digital educational systems, individualization of education, mobile learning.

email: is@it.kfu.ru

Материал поступил в редакцию 3 июля 2019 года