ОГЛАВЛЕНИЕ

УДК 004.514+004.42

АВТОМАТИЗАЦИЯ ПРОЦЕССОВ СБОРА И АНАЛИЗА ДАННЫХ О ВЗАИМОДЕЙСТВИИ С ИНТЕРАКТИВНЫМИ ПРОТОТИПАМИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

А. Р. Динмухаметов[1], И. С. Шахова[2]

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

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

Аннотация

Представлено описание разработанной программной платформы для сбора и автоматического анализа данных о взаимодействии пользователей с интерактивными прототипами, позволяющей организовать непрерывную и оперативную связь между целевой аудиторией и проектировщиками интерфейсов мобильных приложений. Программная платформа включает в себя десктопное и мобильное приложения, а также серверную часть для осуществления анализа данных, хранения информации и организации взаимодействия между клиентскими приложениями.

Ключевые слова: *UI, UX, пользовательский интерфейс, мобильные приложения, прототипирование*

ВВЕДЕНИЕ

Пользовательский опыт (UX) включает в себя все эмоции, убеждения, предпочтения, ощущения, физические и психологические реакции пользователя, поведение и достижения, которые возникают до, во время и после использования системы [1].

Положительный пользовательский опыт повышает качество продукта и заинтересованность пользователей, однако негативный пользовательский опыт может привести к таким последствиям, как снижение продаж, недовольство клиентов, отрицательные отзывы и низкие позиции в рейтинге, негативное влияние на бренд, повышенная потребность в документации и обучении, увеличение запросов и затрат на поддержку [2].

Включение действий по проектированию пользовательского опыта в стандартный процесс разработки программного обеспечения увеличивают временные и финансовые затраты на создание программного продукта. Однако, в то же время, данные работы на начальном этапе позволяют снизить риски внесения более ресурсозатратных изменений на поздних этапах разработки.

Ключом к положительному пользовательскому опыту является вовлечение пользователей в процесс проектирования UX, анализ их поведения, проектирование на основе принципов человеческого фактора и лучших методик проектирования, тестирование проекта с ними в итеративном процессе проектирования.

Таким образом, целью работы стала реализация системы, позволяющей производить сбор данных о взаимодействии пользователей с прототипами, их анализ и находить проблемы с UX на основе эмоций при взаимодействии. Данная система позволит находить и устранять проблемы с UX на этапе прототипирования мобильных приложений.

КОНЦЕПЦИЯ ПРОГРАММНОГО РЕШЕНИЯ

Интерактивные прототипы приложения состоят из страниц и элементов, которые при взаимодействии реагируют так же, как и финальный продукт. Такие прототипы используют, чтобы показать, каким образом реальные процессы отражены в приложении и для проведения юзабилити-тестирований. Качественно проработанные интерактивные прототипы позволяют обнаружить проблемы или удачные решения до выхода продукта в релиз.

Для создания интерактивных прототипов используются специальные вебсервисы и инструменты, такие, как Invision [3], Marvel [4], Atomic.io [5], Framer.js [6], также эти сервисы основываются на создании прототипов мобильных приложений и имеют адаптированную под мобильные устройства версию, что позволяет использовать прототипы для тестирования на мобильных устройствах.

Одними из самых важных измерений UX являются юзабилити и эстетика [7]. Оба эти параметра должны учитываться при разработке прототипов приложений, а выявление проблем в UX на этапе тестирования прототипа позволяет сделать конечный продукт качественнее и уменьшить затраты на разработку продукта [8].

Эффективным инструментом сбора и анализа юзабилити является тепло-

вая карта, которая позволяет выявить наиболее часто используемые элементы и компоненты, определить, какие элементы дизайна кажутся пользователям неявными или не используются ими вовсе [9].

При взаимодействии человека с компьютером эмоции пользователей признаются важными в процессах проектирования и оценки. Анализ эмоций позволяет оценить и сделать выводы относительно как юзабилити, так и эстетической части прототипа [10].

Для решения поставленных задач было принято решение реализовать систему для сбора и анализа данных, состоящую из следующих компонентов:

1. Android-приложение для сбора статистики;
2. Десктопное приложение для отображения статистики в виде тепловых карт, видеозаписей взаимодействия, графиков и результатов анализа данных;
3. Сервер для хранения информации и организации взаимодействия мобильного и десктопного приложений.

Процесс взаимодействия владельца прототипа с тестировщиком интерактивного прототипа отображен на рисунке 1 и состоит из следующих основных этапов:

1. Владелец прототипа создаёт интерактивный прототип в одном из сервисов прототипирования (Marvel, Invision и т.д.);
2. Он добавляет прототип в десктопное приложение и отправляет ссылку на прототип фокус-группе или тестировщикам;
3. Тестировщики открывают прототип в мобильном приложении и тестируют прототип. Во время взаимодействия происходит сбор данных и отправка их на сервер.

Собранные данные отображаются в десктопном приложении в виде графиков, статистики и тепловых карт, а также происходит их анализ для прогнозирования проблем с UX.

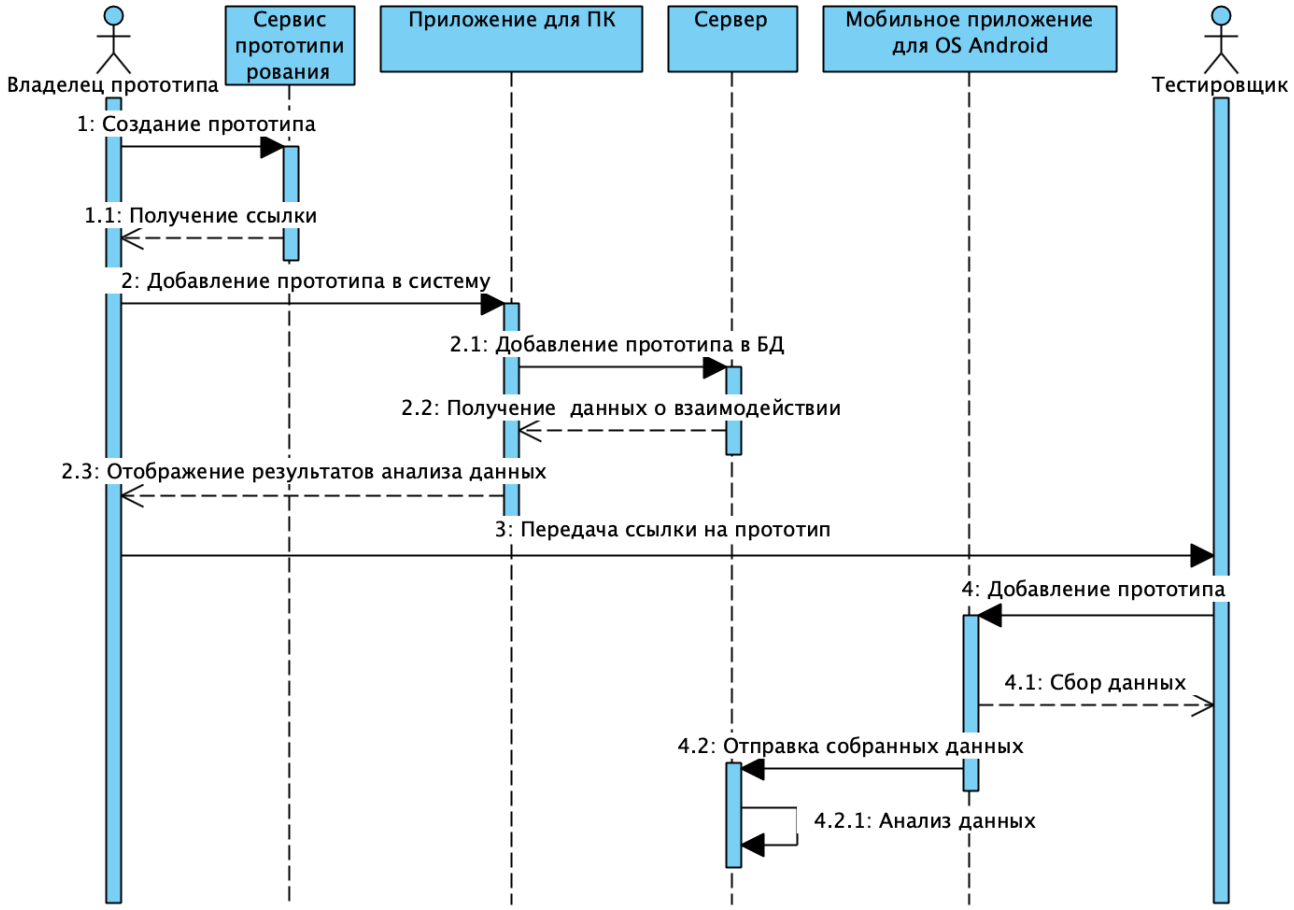


Рисунок 1 – Диаграмма последовательности

АЛГОРИТМ ВЫЯВЛЕНИЯ ПРОБЛЕМ С UX НА ОСНОВЕ ЭМОЦИЙ

UX является важным фактором качества пользовательского интерфейса, который напрямую влияет на ощущения пользователя во время взаимодействия. В статье [11] рассматривается категоризация эмоций в зависимости от возраста и пола пользователя и двух аспектов пользовательского опыта (эстетика и удобство использования) и описывается исследование, в котором приняли участие 45 человек, из которых 53% являлись мужчинами в возрасте от 19 до 67 лет, а 47% – женщины в возрасте от 23 до 69 лет. Каждому участнику был выдан прототип веб-приложения, содержащего явные проблемы с юзабилити или эстетикой. В ходе эксперимента каждые 10 секунд происходил анализ эмоций пользователя на основе записи с камеры.

Результатом эксперимента является определение порогов эмоций, которые могут быть использованы для выявления юзабилити и эстетических про-

блем в зависимости от возраста и пола пользователя. Эти результаты используются для создания алгоритма определения негативных эмоций и связанных с ними проблем пользовательского интерфейса.

Помимо удобства использования, пользовательский опыт (UX) в настоящее время признается важным фактором качества, обеспечивающим успех систем или программного обеспечения с точки зрения восприятия пользователем и частоты использования. Например, согласно [12], UX зависит от таких аспектов, как эмоции, эстетика или внешний вид, значение/ценность или даже веселье, наслаждение или удовольствие.

В упомянутой статье представлены следующие результаты эксперимента:

- эмоции могут быть классифицированы в зависимости от возраста и пола пользователя;
- некоторые пороги эмоций могут быть определены для выявления юзабилити и эстетических проблем в зависимости от возраста и пола пользователя.

Полученные в статье пороговые значения позволяют с 95% точностью определить существующие проблемы с UX на основании данных об эмоциях пользователя во время взаимодействия в зависимости от пола:

1) Мужчины:

anger – от 39% до 64%, disgust – от 6% до 10%, fear – от 17% до 30%, contempt – от 3.3% до 6.5%, neutral – от 58% до 69%, happiness – от 0.6% до 5%.

2) Женщины

anger – от 20% до 50%, disgust – от 25% до 32%, fear – от 4% до 11%, contempt – от 23% до 34%, neutral – от 50% до 64%, happiness – от 5% до 8%.

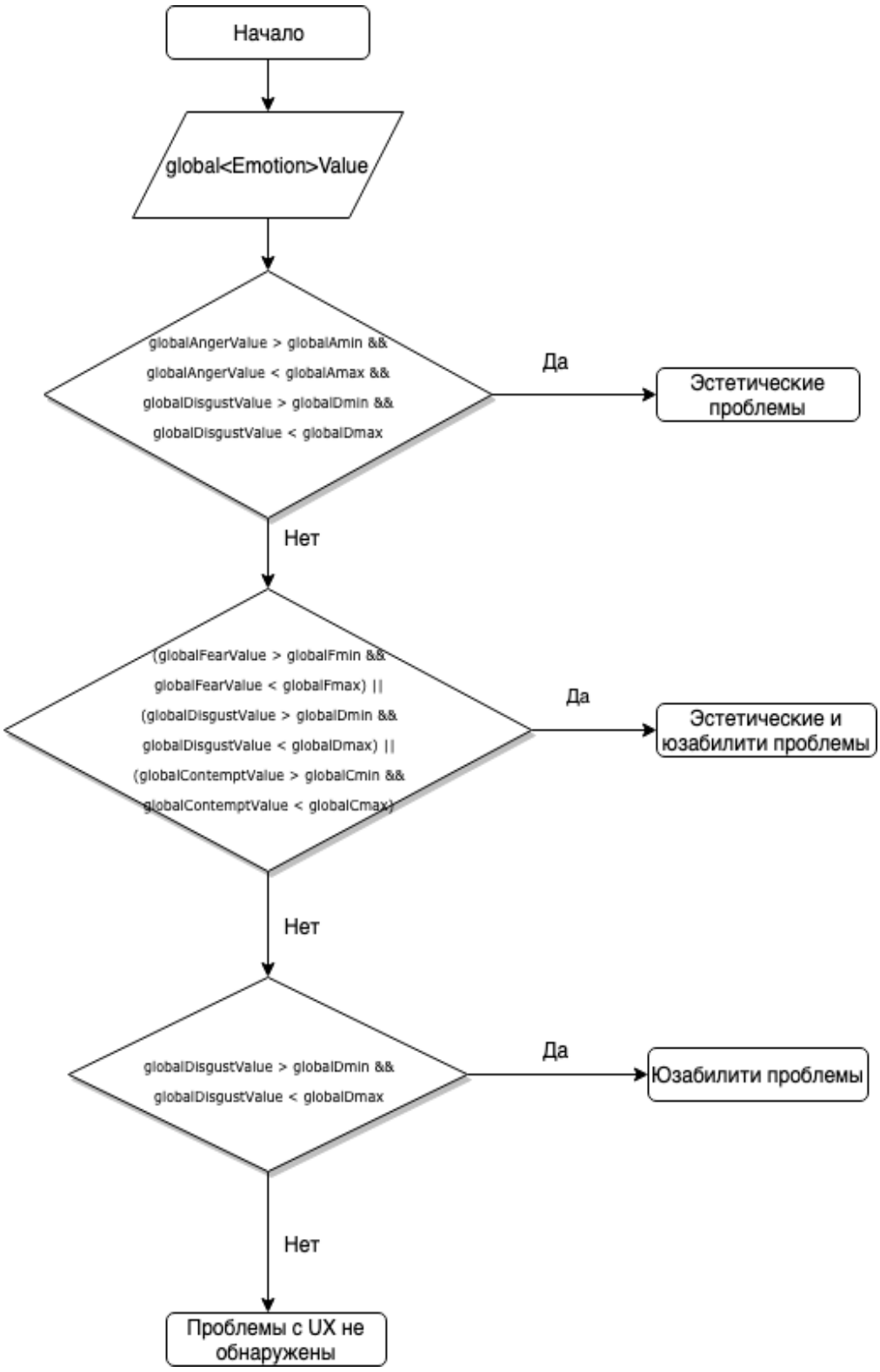


Рисунок 2 – Схема алгоритма обнаружения проблем с UX

Эти данные необходимы для реализации алгоритма поиска существующих проблем с пользовательским опытом (рис. 2).

МОБИЛЬНЫЙ ИНСТРУМЕНТ ДЛЯ СБОРА ДАННЫХ

Мобильный инструмент представляет собой приложение для операционной системы Android. На рис. 3 изображена диаграмма сценариев использования данного мобильного приложения.

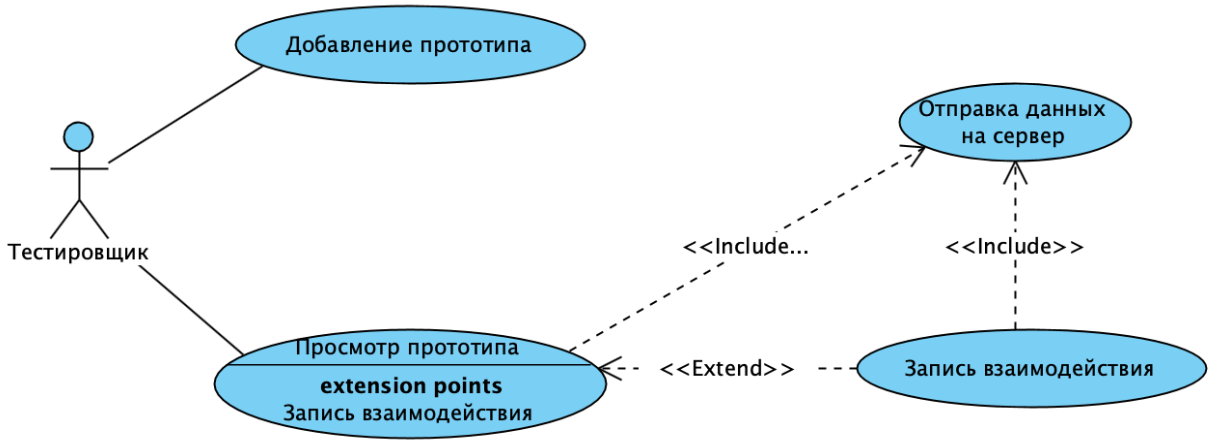


Рисунок 3 – Сценарии использования мобильного приложения

Мобильный инструмент обладает следующими функциональными возможностями:

1) Добавление прототипа в приложение: для начала тестирования и сбора данных необходимо добавить прототип в приложение, указав название и описание прототипа, а также ссылку на него;

2) Тестирование прототипа и сбор данных: интерактивный прототип отображается с помощью webview, во время взаимодействия происходит сбор данных, затем после нажатия на “SAVE HEATMAP” собранные данные отправляются на сервер;

3) Видеозапись взаимодействия и эмоций с фронтальной камеры, отправка видеозаписи и собранных данных на сервер.

ДЕСКТОПНОЕ ПРИЛОЖЕНИЕ

Реализованное десктопное приложение является кроссплатформенным и поддерживает macOS, Windows и Linux. На рис. 4 изображена диаграмма вариантов использования десктопного приложения для владельца прототипа.

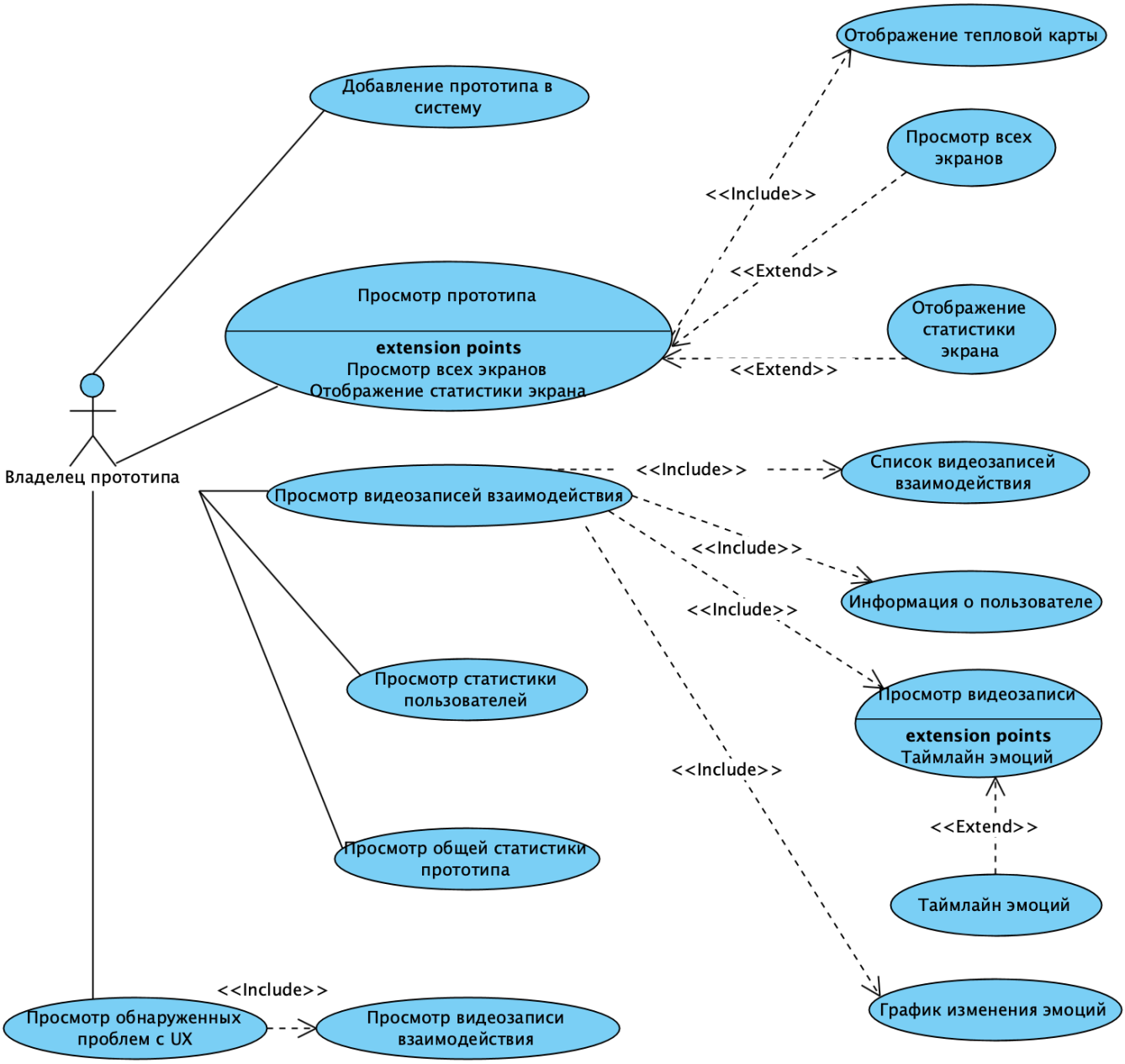


Рисунок 4 – Сценарии использования десктопного приложения

Приложение состоит из следующих экранов:

1. Интерактивный просмотр прототипа и информации по нему с возможностью перехода по ссылкам и страницам прототипа, отображение тепло-

вой карты прототипа (рис. 5);

2. Отображение видеозаписей взаимодействия и эмоций, соответствующих видеозаписи. Экран содержит следующие элементы (рис. 6):

1) список видеозаписей взаимодействия;

2) информация о пользователе: возраст; пол; этническая принадлежность;

3) таймлайн эмоций с возможностью перехода на нужный отрезок времени;

4) просмотр видеозаписи;

5) график изменения эмоций во время взаимодействия, где OX – время, OY – испытываемая эмоция в момент времени;

3. Статистика пользователей по критериям: возраст; пол; этническая принадлежность;

4. Общая статистика прототипа: лучшие или худшие экраны, общий % довольных пользователей;

5. Обнаруженные проблемы с UX – отображение результатов анализа UX в виде таблицы с полями: видеозапись взаимодействия и обнаруженные проблемы (рис. 7).

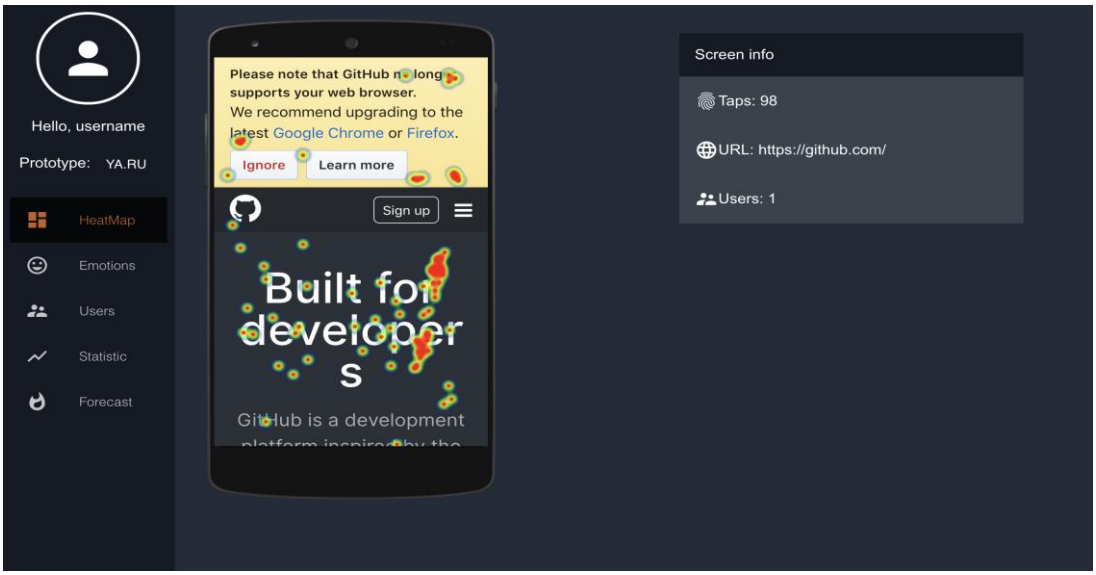


Рисунок 5 – Интерактивный просмотр прототипа

Рисунок 6 – Отображение видеозаписей взаимодействия и эмоций

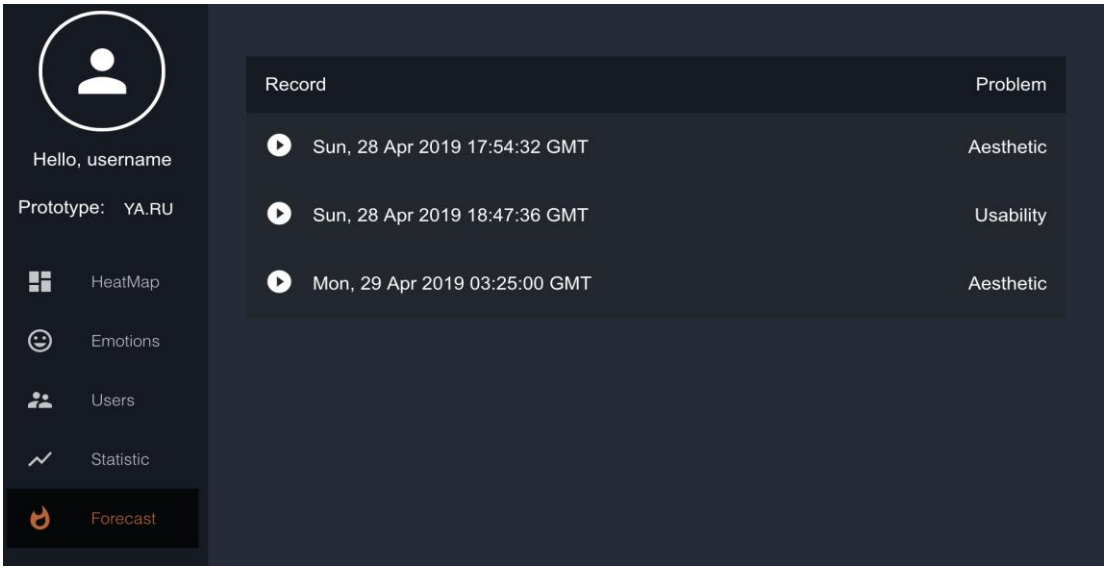


Рисунок 7 – Обнаружение проблем с UX

ЗАКЛЮЧЕНИЕ

В статье приведено описание системы, разработанной с целью снижения трудозатрат на сбор и анализ данных о взаимодействии пользователей с интерактивными прототипами мобильных приложений. Разработанная система позволяет строить тепловые карты взаимодействия пользователя с элементами интерфейсов, анализировать эмоции и выявлять проблемы в интерфейсах на основе данного анализа. Описанные функциональные возможности позволяют обеспечить перманентную обратную связь и вносить необходимые изменения на этапе проектирования.

СПИСОК ЛИТЕРАТУРЫ

1. *Garrett J.* The Elements of User Experience: User-Centered Design for the Web // New Riders, 2011. URL: http://ptgmedia.pearsoncmg.com/images/9780321683687/samplepages/0321683684.pdf (дата обращения: 01.06.2019).

2. *Ross J.* The Business Value of User Experience // INFRAGISTICS, 2014. URL: http://www.infragistics.com/media/335732/the_business_value_of_user_ experience-3. pdf (дата обращения: 01.06.2019).

3. *Invision*. URL: https://www.invisionapp.com/ (дата обращения: 01.06.2019).

4. *MarvelApp*. URL: https://marvelapp.com/ (дата обращения: 01.06.2019).

5. *Atomic.io*. URL: https://atomic.io/ (дата обращения: 01.06.2019).

6. *Framer.js*. URL: https://www.framer.com/ (дата обращения: 01.06.2019).

7. *Tuch A., Roth P., Opwis K., Bargas-Avila A.* Is beautiful really usable? Toward understanding the relation between usability, aesthetics, and affect in HCI // Computers in Human Behavior, 2012. URL: https://www.sciencedirect.com/ science/article/pii/S0747563212000908 (дата обращения: 01.06.2019).

8. *Gilb T., Finzi S.* Principles of Software Engineering Management // Addison-Wesley Professional, 1988. URL: https://dl.acm.org/ citation.cfm?id=59124 (дата обращения: 01.06.2019).

9. *Jacko A.* Human-Computer Interaction. New Trends // Human-Computer Interaction. New Trends, 2009. URL: http://citeseerx.ist.psu.edu/viewdoc/ down-

load;jsessionid=92992AA0F69F7AA52C84EC732A44028C?doi=10.1.1.458.7291&rep=rep1&type=pdf (дата обращения: 01.06.2019).

10. *Hook K.* Affective Computing // Idea Group Reference, 2005. URL: https://www.interaction-design.org/literature/book/the- encyclopedia-of-human-computer-interaction-2nd-ed/affective-computing (дата обращения: 01.06.2019).

11. *Galindo J.* Using user emotions to trigger UI adaptation // RCIS, 2018. URL: http://iihm.imag.fr/publs/2018/rcis18juliangalindo.pdf (дата обращения: 01.06.2019).

12. *Bernhaupt R.* User experience evaluation in entertainment // Springer, 2010. URL: https://www.researchgate.net/publication/ 225995853_User_ Experience_Evaluation_in_Entertainment (дата обращения: 01.06.2019).

AUTOMATION OF COLLECTION AND DATA ANALYSIS OF USER EXPERIENCE IN INTERACTIVE MOBILE PROTOTYPES USAGE

A. R. Dinmukhametov[1], I. S. Shakhova[2]

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

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

Abstract

The paper describes the developed platform for data collection and data analysis of user experience in interactive prototypes usage. The platform allows to organise permanent and fast feedback between the target audience and interface designers. The system includes desktop and mobile applications, and the web-server for data analysis, information storage and connection between the client applications.

Keywords: *UI, UX, user interface, user experience, mobile applications, prototyping*

REFERENCES

1. *Garrett J.* The Elements of User Experience: User-Centered Design for the Web // New Riders, 2011. URL: http://ptgmedia.pearsoncmg.com/images/9780321683687/samplepages/0321683684.pdf.
2. *Ross J.* The Business Value of User Experience // INFRAGISTICS, 2014. URL: http://www.infragistics.com/media/335732/the_business_value_of_user_ experience-3. pdf.
3. *Invision*. URL: https://www.invisionapp.com/.
4. *MarvelApp*. URL: https://marvelapp.com/.
5. *Atomic.io*. URL: https://atomic.io/.
6. *Framer js*. URL: https://www.framer.com/.
7. *Tuch A., Roth P., Opwis K., Bargas-Avila A.* Is beautiful really usable? Toward understanding the relation between usability, aesthetics, and affect in HCI // Computers in Human Behavior, 2012. URL: https://www.sciencedirect.com/science/

article/pii/S0747563212000908.

8. *Gilb T., Finzi S.* Principles of Software Engineering Management // Addison-Wesley Professional, 1988. URL: https://dl.acm.org/ citation.cfm?id=59124.

9. *Jacko A.* Human-Computer Interaction. New Trends // Human-Computer Interaction. New Trends, 2009. URL: http://citeseerx.ist.psu.edu/viewdoc/ download;jsessionid=92992AA0F69F7AA52C84EC732A44028C?doi=10.1.1.458.7291&rep=rep1&type=pdf.

10. *Hook K.* Affective Computing // Idea Group Reference, 2005. URL: https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/affective-computing.

11. *Galindo J.* Using user emotions to trigger UI adaptation // RCIS, 2018. URL: http://iihm.imag.fr/publs/2018/rcis18juliangalindo.pdf.

12. *Bernhaupt R.* User experience evaluation in entertainment // Springer, 2010. URL: https://www.researchgate.net/publication/225995853_User_ Experience_Evaluation_in_Entertainment.

СВЕДЕНИЯ ОБ АВТОРАХ

ДИНМУХАМЕТОВ Айнур Ринатович – магистр Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета по направлению «Программная инженерия».

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

email: aynurdinm@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 in education, mobile learning.

email: is@it.kfu.ru

Материал поступил в редакцию 7 июля 2019 года

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.

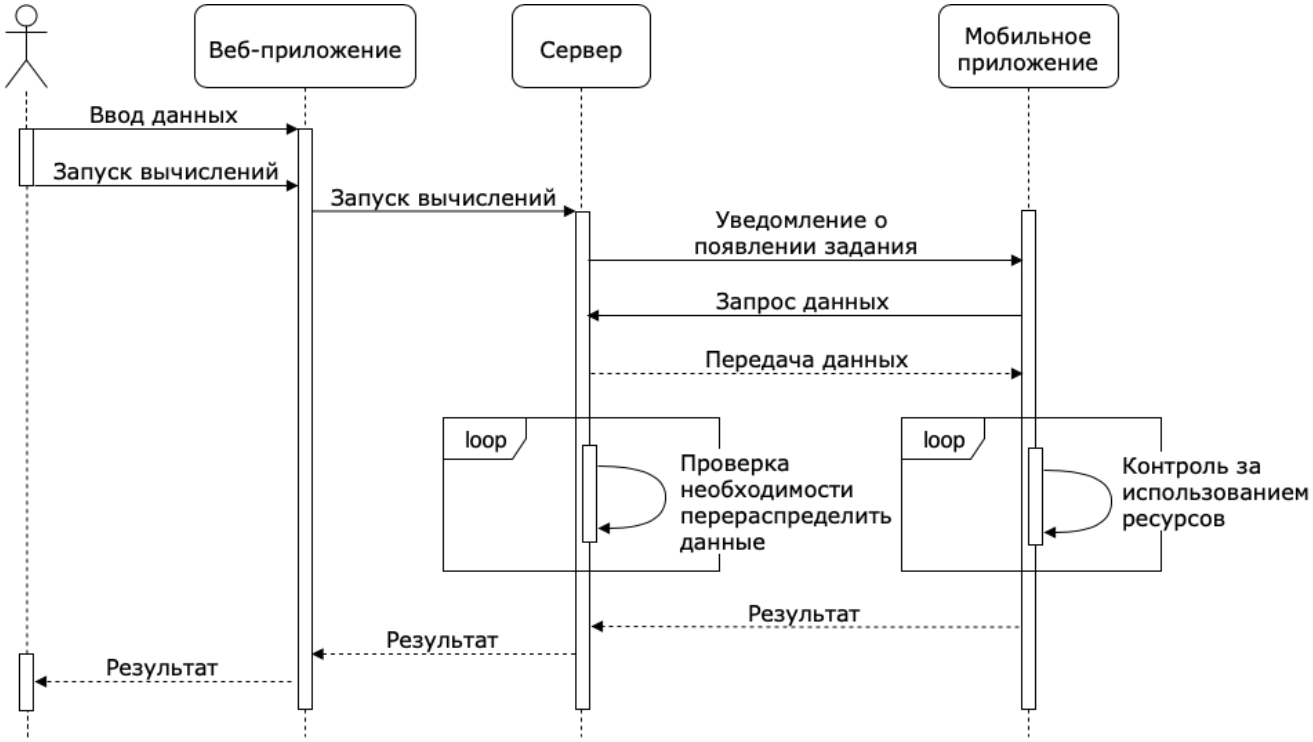


Рис. 1. Концептуальная схема работы системы

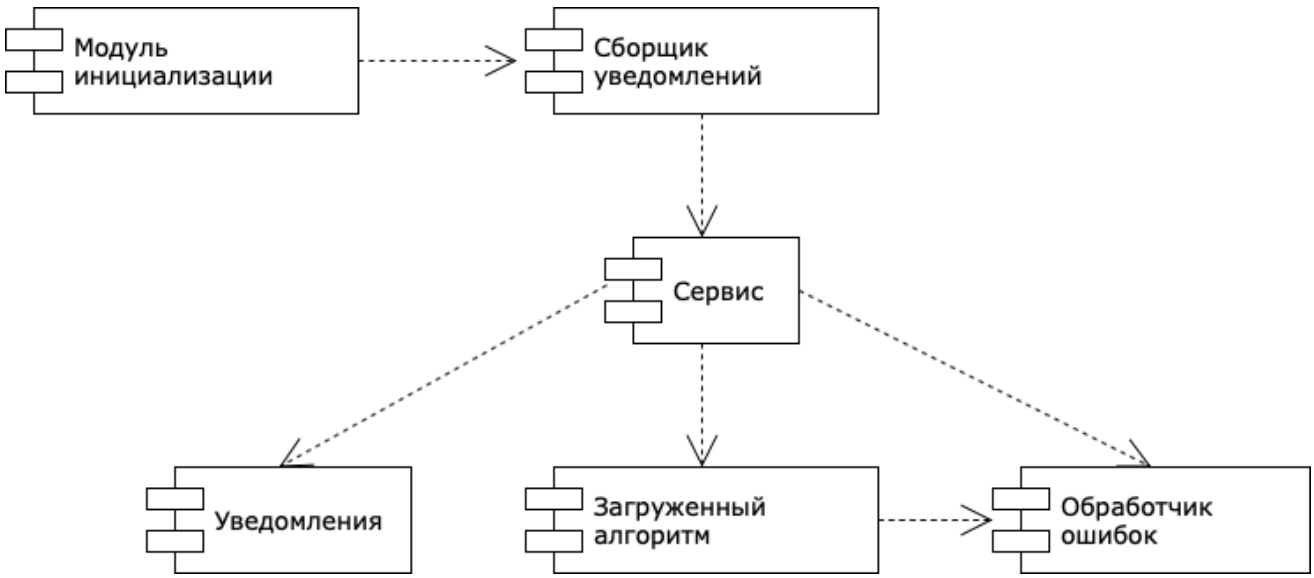


Рис. 2. Общее представление компонентов системы

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.

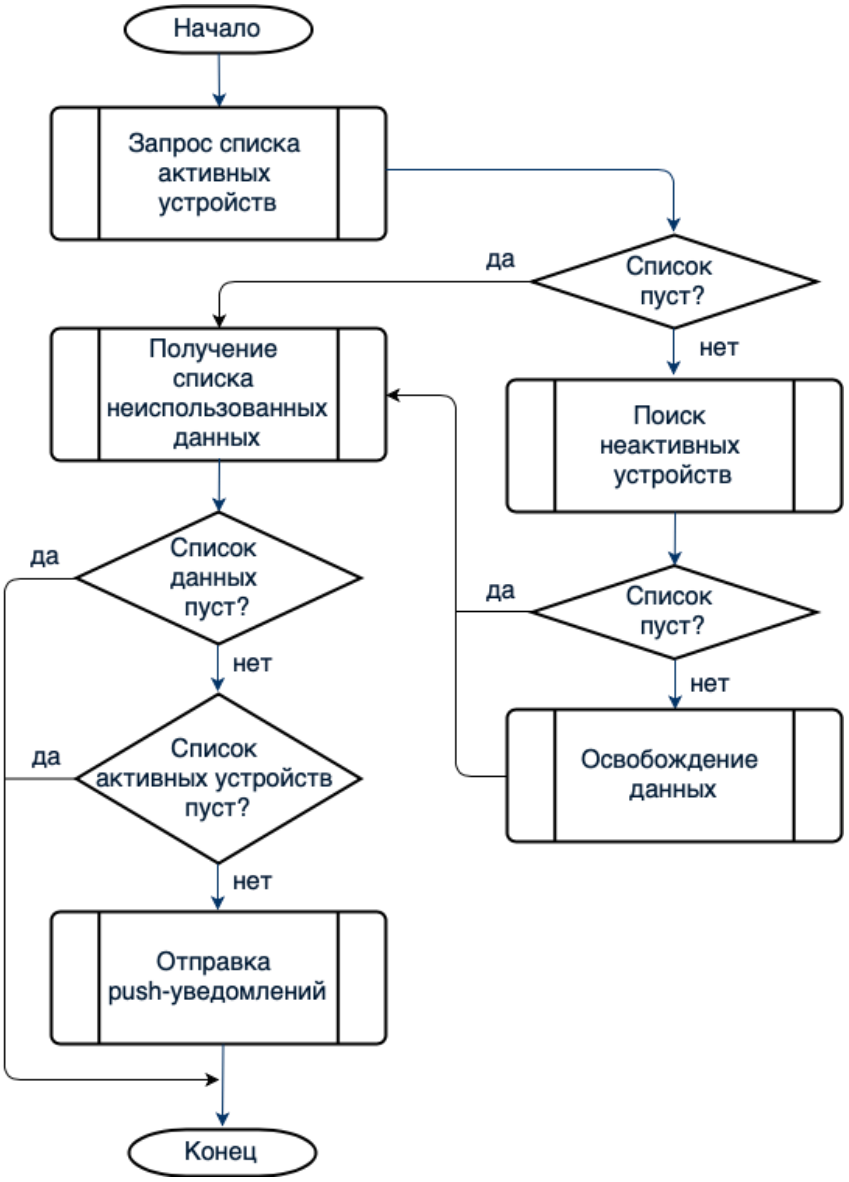


Рис. 4. Блок-схема отслеживания процесса вычисления

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

УДК 004.021 + 004.623 + 004.421

РАЗРАБОТКА СИСТЕМЫ ДЕТЕКТИРОВАНИЯ ДУБЛИКАТОВ ВИДЕОФАЙЛОВ НА ОСНОВЕ ИХ ЦВЕТОВЫХ КАРТ

Г. А. Нуриева[1], А. А. Ференец[2]

[1,2] Высшая школа информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета

[1] nurievag97@gmail.com, [2]ist.kazan@gmail.com

Аннотация

Описана разработка системы детектирования дубликатов видеофайлов на основе методики получения и анализа цветовой карты видеоряда как подхода, не требующего больших вычислительных ресурсов, применимого к широкому спектру типов видеороликов и имеющего малые искажения относительно оригинала.

***Ключевые слова**: видеофайл, цветовая карта, детектирование дубликатов*

ВВЕДЕНИЕ

Согласно модели Web 2.0 [1], содержимое сайтов должно пополняться самими пользователями. Этот подход обеспечивает гораздо более быстрое развитие сайтов, но поддержка сайта всё равно необходима для модерации содержимого, в том числе, исключения дублирования материалов. У крупных веб-порталов невозможно проанализировать каждый ролик вручную. Например, в 2018 году на Youtube каждую минуту загружалось порядка 300 часов видеоматериалов [2]. Поэтому есть необходимость полной или частичной автоматизации этого процесса.

Анализ видеофайлов осложняется тем, что иногда искажается видео- или аудиоряд: используется другой способ кодирования, специально применяются незначительные искажения (изменение разрешения, зеркальное отражение, добавление артефактов, удаление некоторых кадров и прочие изменения). Это исключает точное сравнение последовательностей байт и требует анализа непосредственно видеоданных. Для этого можно анализировать низкоуровневые

признаки: частота кадров, разрешение, глубина цвета. При этом их точные замеры не дают однозначного определения соответствия, так как изменение этих параметров за счёт перекодирования видеофайла сохраняет узнаваемые образы, но изменяет числовые показатели указанных. Соответственно можно анализировать образы, которые передаёт видео. С точки зрения детектирования дублирования видеофайлов это наиболее прогрессивный метод, но требующий использования технологий компьютерного зрения для распознавания образов, значит, обучающих наборов данных для разных тематических категорий видео и определённых вычислительных ресурсов. С другой стороны, возможно выделение некоторых параметров видеоряда, которые связаны не с техническими особенностями представления, а именно с содержимым, и при этом достаточно абстрактными, чтобы их замерять напрямую или без больших вычислительных цепочек. Примером такого параметра является информация о цветах, использованных в видеоряде, или, иначе говоря, о его цветовой карте: непосредственное количество пикселей того или иного цвета или иная информация, следующая из замера количества этих пикселей. Была выдвинута гипотеза, что на основе сравнения цветовых карт видеорядов можно сделать вывод о дублировании одного из них другим даже при незначительных модификациях повторяющегося фрагмента.

1. СОЗДАНИЕ ЦВЕТОВОЙ КАРТЫ

Для получения цветовой карты видеоряда было решено вычислять средний цвет избранных кадров (получать усреднённые показатели цветовых параметров всех пикселей кадров) и далее собирать эту информацию во временные ряды. Кадры для анализа берутся из исходного видеоряда с определённым периодом, который должен быть подобран в зависимости от способа замера цвета.

Были рассмотрены две модели представления цвета: RGB и HSV. В цветовой модели RGB каждый цвет может быть описан как числовое соотношение трех базовых цветов – красного (Red), зеленого (Green) и синего (Blue). Обычно эти параметры варьируются в диапазоне 0–255. HSV (англ. Hue, Saturation, Value) – цветовая модель, в которой координатами цвета являются Hue – тон, Saturation – насыщенность и Value – значение (Brightness – яркость). Тон варьи-

руется от 0 до 360° (но часто приводится к диапазону 0–1), насыщенность и яркость – от 0 до 1. Модель HSV является нелинейным преобразованием модели RGB. Поэтому легко выполнить преобразование RGB в HSV и в обратную сторону. При этом использование того или иного представления может дать более наглядные результаты в зависимости от параметров видеоряда.

Для получения среднего цвета кадра было решено проанализировать все пиксели, вычленения значения RGB и их преобразование в HSV при необходимости, а далее проводить вычисление среднего арифметического как общего показателя для массива данных.

2. АЛГОРИТМ ДЕТЕКТИРОВАНИЯ ДУБЛИРОВАНИЯ ВИДЕОРЯДОВ

Для детектирования необходимо произвести поиск похожих последовательностей в списках, содержащих средние цвета извлеченных кадров видеофайлов. Перед выполнением проверки необходимо подготовить цветовые последовательности. Похожесть определяется максимальным количеством кадров со средними цветами, отличающихся более, чем заданное значение. Как было сказано выше, разность цветов определяется в зависимости от цветовой модели.

Таким образом, алгоритм может быть описан кодом на Java, представленным ниже. В нём осуществлено полное поэлементное сравнение двух списков list1 и list2, содержащих значения средних цветов кадров.

```
for (int i = 0; i <= list1.size() - minFragmentDuration; i++) {
  for (int j = 0; j <= list2.size() - minFragmentDuration; j++) {
    distance = getRGBorHSVDistance(list1.get(i), list2.get(j), colorModel);
    if (distance < averageDifference) {
      VideoFragment vf = new VideoFragment(i, j, distance , 1);
      vf.setPossibleDuration(list1.size(), list2.size());
      checkFragmentSimilarity(vf);
     }
   }
}
```

В коде используются переменные и функции:

- minFragmentDuration – количество кадров в последовательности, при которой считается, что найден дублирующийся видеофрагмент;
- getRGBorHSVDistance – метод, возвращающий различие значений средних цветов в зависимости от используемой цветовой модели; для цветовой модели RGB различие значений определяется как евклидово расстояние: $\mathrm{distance} = \mathrm{red}2 + \mathrm{green}2 + \mathrm{blue}2$. Для цветовой модели HSV вычисляется модуль разности для каждого значения Hue, Saturation, Value: $\mathrm{distance} = \Delta\mathrm{Hue}$, $\mathrm{distance} = \Delta\mathrm{Saturation}$, $\mathrm{distance} = \Delta\mathrm{Value}$;
- averageDifference – параметр различия средних цветов кадров; значения параметра различаются для цветовых моделей RGB и HSV; если для кадров distance < averageDifference, то они считаются похожими.

Пусть даны списки средних цветов кадров двух видео. Списки обрабатываются по вложенному циклу – вычисляется разность значений distance и сравнивается с установленным параметром averageDifference.

Если для сравниваемых кадров distance>averageDifference, то алгоритм продолжает поиск по циклу (Рис. 1).

Если встречается пара элементов из двух списков с разностью, меньшей указанного параметра distance<averageDifference, то предполагается, что найден дублирующийся видеофрагмент (Рис. 2). Создается экземпляр класса Fragment, в конструкторе которого указывается информация о найденном фрагменте: индексы первых кадров для обоих видео, параметр сходства, длительность. Каждая следующая пара элементов из двух списков сравнивается между собой, и, если значения их средних цветов близки, обновляются данные о фрагменте: длина, значение сходства.

Рисунок 1 – Схема работы алгоритма при distance<averageDifference

Рисунок 2 – Схема работы алгоритма при distance > averageDifference

Если встречается пара кадров, для которых distance>averageDifference, то попарное сравнение завершается, в ранее созданный объект класса Fragment записываются индексы конца видеофрагмента. Найденный видеофрагмент считается дублирующимся и заносится в список результатов, если его длина duration больше или равна установленному значению minFragmentDuration.

```
if (ed < averageDifference) {
  vf.setDuration(duration + 1);
  vf.setSimilarity(ed);
  checkFragmentSimilarity(vf);
} else {
  if (duration >= minFragmentDuration) {
     vf.setEndFrameOne();
     vf.setEndFrameTwo();
     fragments.add(vf);
  }
}
```

После этого алгоритм возвращается к вложенному циклу, и продолжается поиск.

3. ОПРЕДЕЛЕНИЕ ОПТИМАЛЬНЫХ ПАРАМЕТРОВ РАБОТЫ АЛГОРИТМА

Эффективность работы алгоритма поиска дубликатов видео зависит от выбора цветовой модели и параметров seconds_between_frames, average_difference, min_fragment_duration.

В тестовой выборке видеофайлов использованы дубликаты с небольшими искажениями, экранные снимки, видео с низким качеством.

average_difference определяет величину расстояния между значениями средних цветов двух кадров. Необходимо задать такое значение параметра, при котором алгоритм будет считать дубликатами извлеченные кадры видео и экранные снимки, сделанные в один и тот же короткий промежуток времени.

Два кадра, извлеченные из видео с промежутком в 1–3 секунды, можно считать похожими. Из видеофайлов извлечены 10 таких пар кадров и определено различие средних цветов для каждой пары в разных цветовых моделях. Результаты представлены в Таблице 1.

На одном и том же моменте видеоролика извлекается кадр и производится экранный снимок. Это моделирует ситуацию распознавания «экранок» видео. Сравнение среднего цвета производилось на 10 парах таких изображений. Результаты представлены в Таблице 2.

Таблица 1 – Различие средних цветов похожих кадров

Пары кадров	Параметр сравнения кадров			
	RGB	HSV (Hue), 10-4	HSV (Saturation), 10-4	HSV (Value), 10-4
1	2.1	3.7	11	3.28
2	0.13	4.4	8.9	3.72
3	0.08	3.1	9.7	2.27
4	0.12	4.1	10.7	4.05
5	0.19	5.5	11.9	3.36
6	0.18	2.3	6.3	3.48
7	0.16	3.8	7.25	3.26
8	0.09	2.5	3.46	2.97
9	2.08	1.7	7.85	2.53
10	1.13	2.9	9.2	3.78

Таблица 2 – Разница средних цветов кадра и экранного снимка

Пары кадров	Параметр сравнения кадров			
	RGB	HSV (Hue), 10-2	HSV (Saturation), 10-2	HSV (Value), 10-2
1	23	3	0.54	2.64
2	45	10	0.77	1.78
3	39	8	1.22	1.55
4	17	22	0.93	2.93
5	28	15	0.57	3.15
6	53	19	0.71	1.69
7	48	9	1.24	1.05
8	25	30	0.73	1.89
9	34	7	1.51	2.48
10	39	14	0.86	1.15

Также оценено различие среднего цвета изображений, не являющихся дубликатами. Для пространств HSV и RGB полученные результаты представлены в Таблице 3.

Таблица 3 – Различие средних цветов различных изображений

Параметр сравнения кадров	Среднее значение разницы цветов	Минимальное значение разницы средних цветов
RGB	38.35	26.48
HSV (Hue)	0.034	0.0165
HSV (Saturation)	0.0413	0.0328
HSV (Value)	0.0529	0.0356

Для цветовой модели RGB и параметра Hue пространства HSV вычисленная разность для экранных снимков и изображений, не являющихся дубликатами, очень близка. Из этого следует, что поиск по данным параметрам будет давать ошибочные результаты для экранных съемок. Сравнение изображений по параметру Saturation является оптимальным при определении экранных снимков. С учетом погрешностей измерений установлены следующие значения average_difference при различных параметрах сравнения: Hue – 0.02 (не применимо для поиска экранных съемок), Saturation – 0.03, Value – 0.03.

Параметр seconds_between_frames устанавливает число секунд между извлечением кадров.

Значение параметра должно определяться с учетом следующих условий:

1. частота извлечения должна быть достаточной для обнаружения дубликатов кадров;

2. при увеличении значения seconds_between_frames уменьшается число извлеченных кадров, что способствует увеличению производительности.

На Рис. 3 представлен график зависимости разницы средних цветов от интервала извлечения seconds_between_frames.

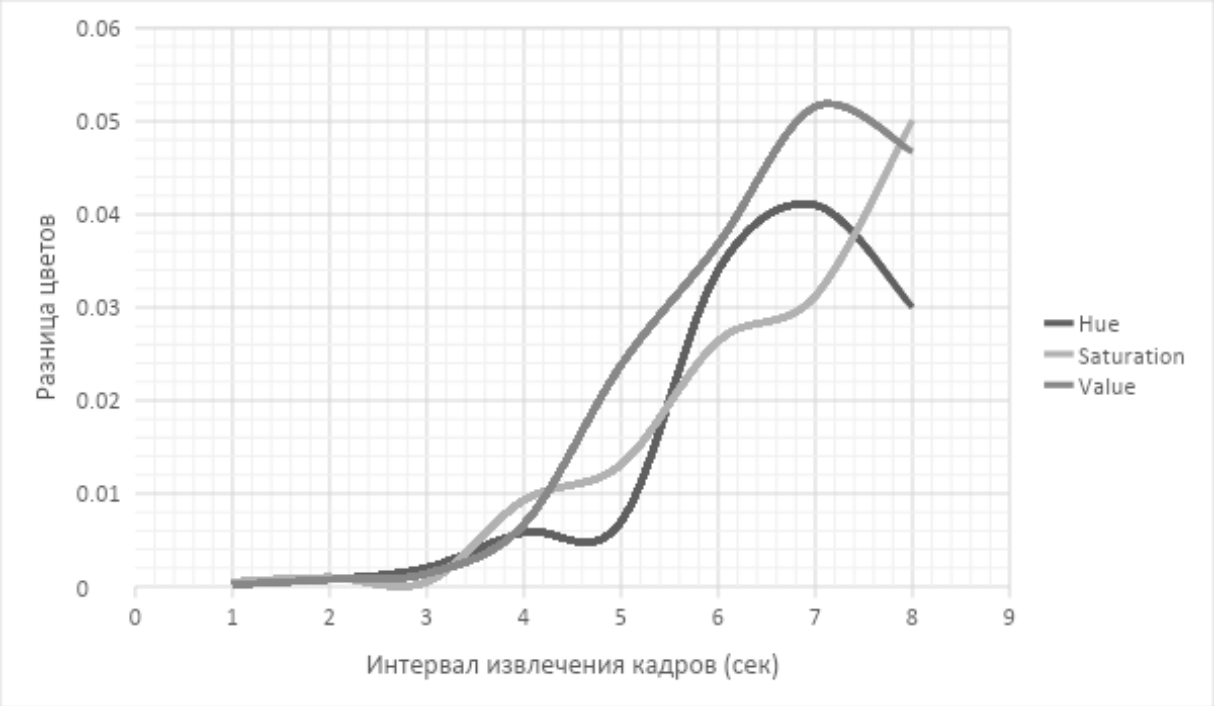


Рисунок 3 – Зависимость разницы среднего цвета кадров от интервала извлечения из видео

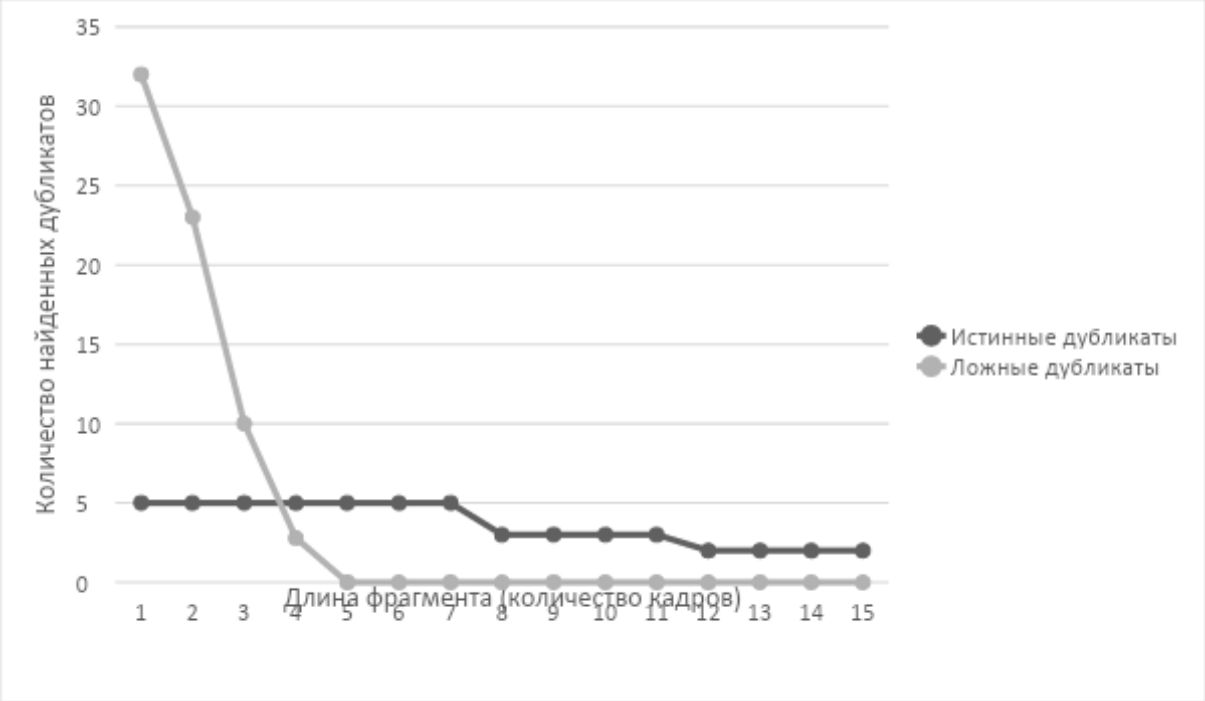


Рисунок 4 – Зависимость разницы эффективности поиска дубликатов от длины искомого фрагмента

На Рис. 4 представлен график зависимости эффективности поиска дубликатов от длины искомого фрагмента min_fragment_duration.

Согласно полученным результатам, оптимальное значение для параметров seconds_between_frames и min_fragment_duration соответственно равно 4 и 5.

4. ИСКЛЮЧИТЕЛЬНЫЕ СИТУАЦИИ В РАБОТЕ АЛГОРИТМА

Существуют различные методы обхода алгоритмов поиска дубликатов видео. Основной способ состоит в искажении исходного видео. При этом следует учитывать, что искаженное видео является нечетким дубликатом только в том случае, если оно сохраняет визуальное содержимое оригинального видео. Поэтому видео, подвергшиеся сильным модификациям и потерявшие целостность содержания, не представляют информационной ценности как дубликаты и не попадают под действие алгоритма поиска нечетких дубликатов.

Возможны следующие варианты искажений:

1. добавление рамки в видео;
2. поворот и растягивание видео в кадре;
3. наложение фильтров на видео.

Для обнаружения рамок, цветовых вставок, яркостных, геометрических искажений или их комбинаций следует добавить предобработку кадров после их извлечения из видеоряда. Это позволит удалять либо игнорировать такого рода искажения.

Существует метод предварительной обработки изображений в рамках подхода, основанного на вычислении хэш-значений в скользящем окне [3]. Предварительная обработка заключается в препарировании исходного изображения при помощи одного из преобразований: снижение яркостного диапазона, вычисление градиента, разложение поля яркости по ортонормированному базису, адаптивное линейное контрастирование, локальный бинарный шаблон. Исследования, проведенные в работе [4], показали устойчивость большинства преобразований к яркостному сдвигу.

5. ОПРЕДЕЛЕНИЕ ПАРАМЕТРОВ ДУБЛИРОВАНИЯ ВИДЕОФАЙЛОВ

Результат работы алгоритма представляет собой информацию о наличии дублирующихся последовательностей для каждого видео: длина (количество кадров), индексы начала и конца фрагмента, значения сходства тона, насыщенности и яркости. На основе этих данных составляется описание параметров дублирования видеофайлов и формируется отчет.

По длине фрагмента определяется процент дублирования от общей длительности видео. Информация о величине параметра seconds_between_frames позволяет найти длину фрагмента в секундах и временный интервал дублирования по индексам начала и конца фрагмента в списке кадров.

По полученным параметрам производится оценка характера дублирования:

1. полный дубликат – более 90% дублирования;

2. частичный дубликат – найден один или более фрагментов дублирования, процент общего дублирования менее 90%;

3. экранная съемка – определяется по среднему значению сходства видео.

В результате анализа параметров дублирования видеофайлов генерируется отчет о проверке, имеющий следующую структуру:

1. длина дублирования,

2. процент дублирования от общей длительности,

3. интервал(ы) дублирования,

4. характер дублирования (полный дубликат, частичный дубликат, экранная съемка).

6. ТЕСТИРОВАНИЕ

Для тестирования системы использовался набор из 50 видео, взятых с видеосервиса YouTube, содержащий дубликаты видео, созданные с помощью обрезки, искажения, экранной съемки оригинальных видео. Скорость подготовки цветовых последовательностей в зависимости от длительности видео приведена в Таблице 4.

Таблица 4 – Зависимость времени обработки от длительности видео

Длительность видео (min)	0.5	1	2	5	10	20	60
Время извлечения кадров (s)	2	5	12	31	84	218	718
Время вычисления среднего цвета (ms)	8	19	27	74	168	385	1384

Для оценки времени работы алгоритма поиска дубликатов был проведён поиск дубликатов в базе видео методом полного перебора, то есть производилось сравнение каждого фильма с каждым (меньшее по длине видео искалось в большем) за исключением сравнения с самим собой. Таким образом, для базы из N фильмов выполнялось N(N–1)/2 сравнений. Время обработки видео не учитывалось.

Для оценки качества работы алгоритма использовались величины Полнота (Recall) и Точность (Precision): Recall=M/R, Precision=M/F, где M – количество правильно найденных дубликатов, R – действительное количество дубликатов, F – количество дубликатов, найденное алгоритмом.

При помощи изменения порога сравнения меры схожести сцен average_difference для HSV(Saturation) был получен график Recall/Precision, представленный на Рис. 5. Максимальная величина F-меры равна 0,839.

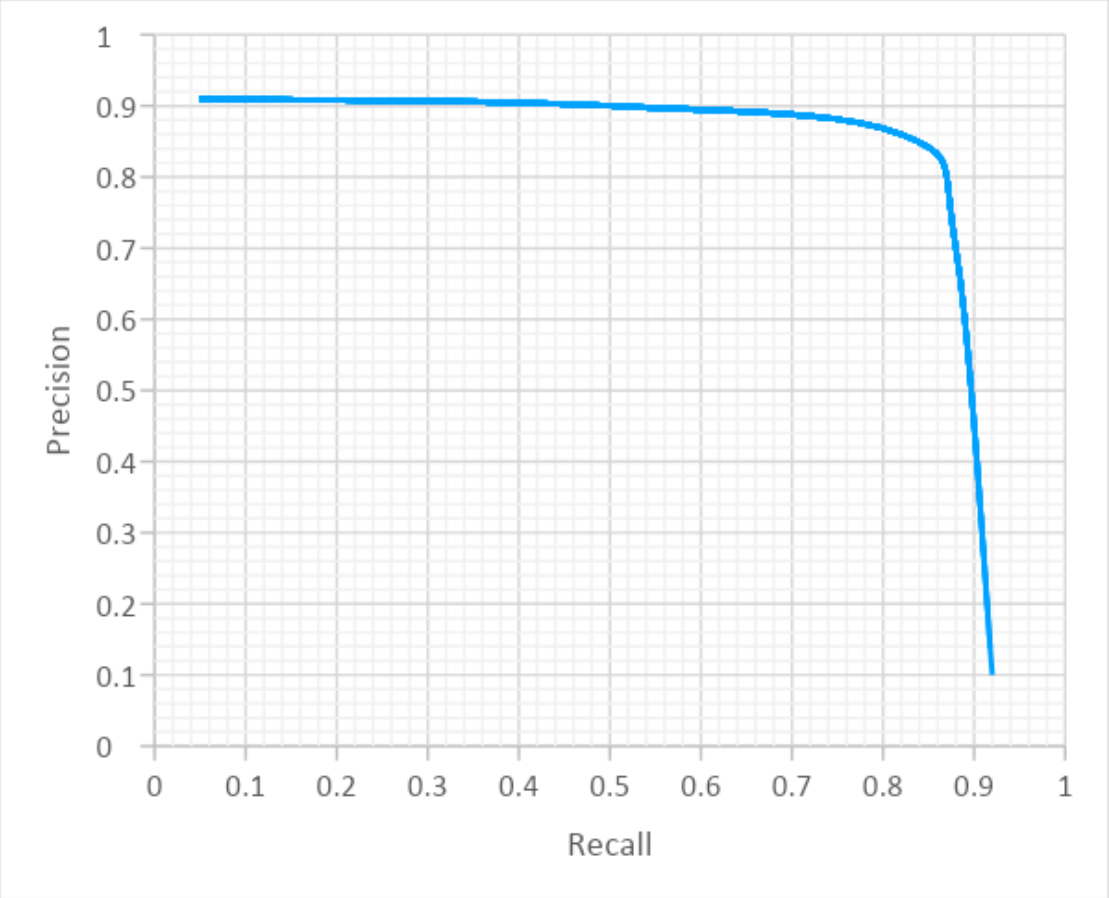


Рисунок 5 – Отношение Recall/Precision

ЗАКЛЮЧЕНИЕ

Программа обнаружения дублирования видеофайлов способна распознавать экранную съемку, частичное дублирование, зеркальные копии и устойчива к различным искажениям цифрового видео, таким, как изменение качества изображения, размеров, ориентации видео (повороты), добавление субтитров и небольших логотипов и т. д.

Модуль определения параметров дублирования обрабатывает данные, полученные в результате обнаружения дубликатов, и формирует отчет, содержащий информацию о параметрах и характере дублирования.

Таким образом, система позволяет не только детектировать наличие дублирования видеофайлов, но и измерить количественные и качественные показателей дублирования, за счет чего можно построить индивидуальную политику в отношении дубликатов видео.

СПИСОК ЛИТЕРАТУРЫ

1. What Is Web 2.0 [Электронный ресурс] URL: https://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html
2. Youtube [Электронный ресурс] URL: https://www.youtube.com/
3. Мясников В.В., Глумов Н.И. Поиск дубликатов на цифровых изображениях // Компьютерная оптика, 2013, Т. 37, № 3, С. 360–367.
4. Кузнецов А.В., Мясников В.В. Исследование методов предварительной обработки изображений в задаче обнаружения дубликатов на изображении// Сборник трудов III международной конференции и молодежной школы «Информационные технологии и нанотехнологии», Самара. 2017. С. 904–911.

DEVELOPMENT OF A SYSTEM FOR DETECTING VIDEO DUPLICATES BASED ON THEIR COLOR MAPS

G. A. Nurieva[1], A. A. Ferenetz[2]

[1-2] Higher School for Information Technologies and Intelligent Systems of Kazan (Volga Region) Federal University

[1]nurievag97@gmail.com, [2]ist.kazan@gmail.com

Abstract

This article describes the development of a system for detecting duplicate video files based on the method of obtaining and analyzing the color map of a video sequence, as an approach that does not require large computational resources, is applicable to a wide range of types of video clips and has small distortions relatively to the original.

Keywords*: video file, color map, duplicate detection*

REFERENCES

1. What Is Web 2.0 https://www.oreilly.com/pub/a/web2/archive/ what-is-web-20.html
2. Youtube https://www.youtube.com/
3. Myasnikov V.V., Glumov N.I. Poisk dublikatov na cifrovyh izobrazhe-niyah // Komp'yuternaya optika, 2013, T. 37, № 3, S. 360–367.

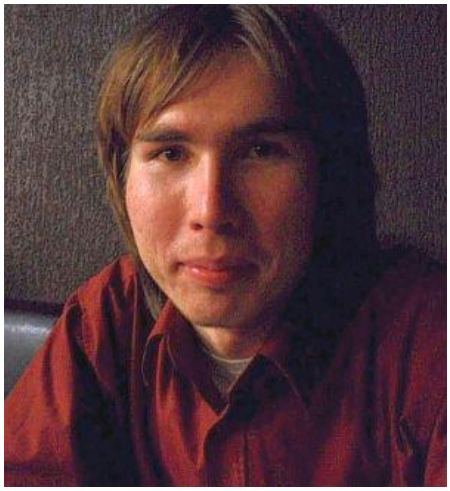
4. Kuznecov A.V., Myasnikov V.V. Issledovanie metodov predvaritel'-noj obrabotki izobrazhenij v zadache obnaruzheniya dublikatov na izobrazhenii// Sbornik trudov III mezhdunarodnoj konferencii i molodezhnoj shkoly «Infor-macionnye tekhnologii i nanotekhnologii», Samara. 2017. S. 904–911.

СВЕДЕНИЯ ОБ АВТОРАХ

НУРИЕВА Гульшат Аталасовна – выпускник бакалавриата Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Gulshat Atlasovna NURIEVA – graduate bachelor of Higher School of ITIS KFU.

email: nurievag97@gmail.com



ФЕРЕНЕЦ Александр Андреевич – ассистент, преподаватель кафедры программной инженерии Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

***Alexande*r Andreevich FERENETS** – assistant at Department of Software Engineering of Higher School of ITIS KFU

email: ist.kazan@gmail.com

Материал поступил в редакцию 15 августа 2019 года