УДК 004.415.2+004.353+004.75

ФРЕЙМВОРК ДЛЯ ОБЛАЧНОГО ВИДЕОМОНИТОРИНГА ЧЕРЕЗ IP-КАМЕРЫ С ИНТУИТИВНО-ПОНЯТНЫМ ИНТЕРФЕЙСОМ

А.С. Гришина[1], В.В. Кугуракова[2]

[1,2] Казанский (Приволжский) федеральный университет

[1]sinerysmile@gmail.com, [2]vlada.kugurakova@gmail.com

Аннотация

Описаны основные моменты процесса создания системы, которая позволяет управлять несколькими камерами одновременно, сохраняя данные на сервере. Система имеет возможность подключать IP-камеры и камеры на мобильных устройствах, предоставлять доступ другим пользователям, а также позволяет осуществлять просмотр видео в онлайн-режиме. Выявлены и описаны «горячие точки» в архитектуре системы. Разработан отдельный Angular-модуль, использованы паттерны проектирования. Описано взаимодействие системы с пользователем. Предложены этапы дальнейшего развития облачного видеомониторинга через IP-камеры с легкодоступным управлением для конечного пользователя.

***Ключевые слова**: IP-камера, система, сервер, приложение, видеонаблюдение.*

ВВЕДЕНИЕ

Инновации не стоят на месте, изо дня в день появляются новые идеи и наработки. С развитием технологии видеозаписи становятся всё более востребованы системы, которые могут выполнять многие функции, такие, как подключение камеры, просмотр и скачивание видео, предоставление доступа к камере и многие другие. Пользователи ищут более удобное и доступное приложение.

Наиболее доступным решением в качестве источника трансляции может стать любая IP-камера. Это бюджетное устройство, которое сможет использовать любой пользователь. Если используется множество устройств, поддерживающих запись видео как источников транслируемого сигнала, а пользователей много, то необходима удобная система управления пользователями, их камерами, их файлами и доступом к ним. Такими камерами могут выступать как веб-камеры,

видеокамеры смартфонов, так и сетевые стримеры и медиа серверы. Просматривать трансляции было бы удобно как с компьютера, так и с мобильных устройств: смартфонов, планшетов. Эта система управления автономными устройствами видеонаблюдения должна решать также такие задачи, как настройка и управление камер, распределение доступа между пользователями с разными ролями, с разграничением возможностей с помощью введения групп доступа.

ДОСТАВКА ВИДЕОПОТОКОВ С ГЕОГРАФИЧЕСКИ УДАЛЕННЫХ КАМЕР

Разработка облачных систем обработки видеопотоков актуальна для сегодняшнего времени, такие системы активно разрабатываются сейчас и будут продолжать разрабатываться, обрастая новыми возможностями, так как, по мере увеличения числа пользователей, увеличиваются и требования самих пользователей к задачам по обработке данных с видео.

Самой важной проблемой доставки видео является защита информации и увеличение ресурсов для сохранения видео на сервер. В статье [1] описан один из методов для обеспечения конфиденциальности и защиты видео в облачных сервисах: представлена POC-реализация распределенного CVSS в облаке, основанная на сети.

Другая проблема – это масштабируемость, для обеспечения надежного получения видео одновременно с множества камер через ненадежные сети, в [2] для ее решения предложены эффективные методы.

Следующая проблема – обеспечение качества обслуживания – освещена в [3], где предложено использование контроллера OpenQoS на основе подхода SDN для доставки мультимедиа.

Исследования [1–3] не охватывают пользовательских проблем по обеспечению комфортного использования большого количества разнородных камер в личных целях, отводя решению этой проблемы менее значительное место. Мы же считаем проблему простого, комфортного, интуитивного, понятного управления пользовательскими трансляциями довольно важной.

НАША КОНЦЕПЦИЯ ПРОСТОГО УПРАВЛЕНИЯ КАМЕРАМИ

Чтобы можно было подключить любую камеру любому пользователю с его мобильного устройства, необходимо создать такую систему управления его ка-

мерами, которая может удовлетворить логичные потребности пользователя по работе со своими видеофайлами. В качестве основного функционала можно выделить следующие задачи:

- управление несколькими камерами одновременно («easily» добавление / удаление доступа к камере, удаленное включение / выключение);
- хранение данных на сервере;
- возможность подключения камер и на мобильных устройствах;
- предоставление доступа другим пользователям.

Система состоит из клиентской и серверной частей, тем самым реализуя технологию «клиент-сервер». Клиентская часть системы реализует пользовательский интерфейс, формирует запросы к серверу и обрабатывает ответы от него. Серверная часть системы получает запрос от клиента, выполняет вычисления, после этого формирует веб-страницу и отправляет её клиенту по сети с использованием протокола HTTP[1]. Рисунок 1 иллюстрирует работу системы.

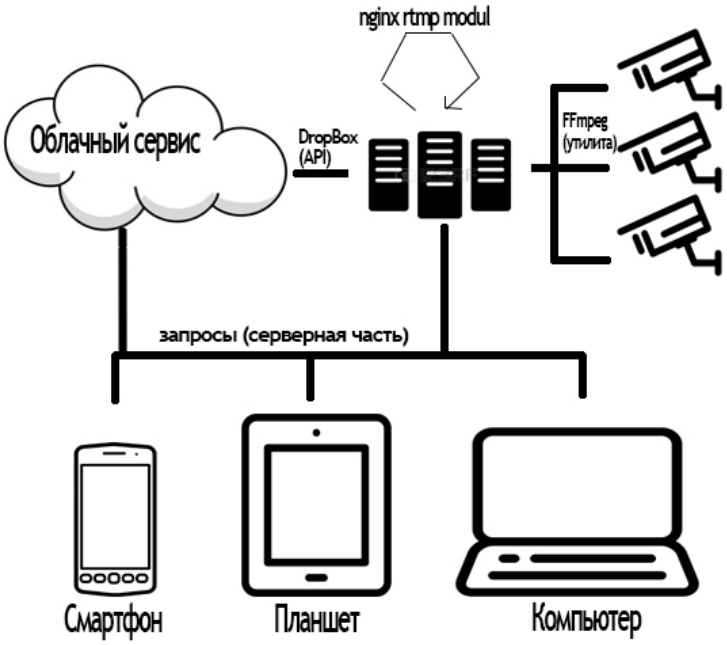

Рис. 1. Схема работы системы

При подключении новой камеры записывается URL-адрес, являющийся для каждой камеры уникальным. После подключения камеры запускается циклическая консольная команда в фоновом режиме, использующая библиотеки

[1] HTTP (англ. HyperText Transfer Protocol — «протокол передачи гипертекста») — протокол прикладного уровня передачи данных (изначально — в виде гипертекстовых документов в формате «HTML», в настоящий момент используется для передачи произвольных данных)

FFMPEG (набор свободных библиотек с открытым исходным кодом [4]), которая записывает на сервер видео указанными промежутками времени. Записанное на сервер видео направляется в облако [2], например, в Dropbox[2]. В облаке создается папка с идентификатором пользователя, и сохраняется видео, которое закрепилось за новой камерой. После того, как видео направлено в облако, на сервере консольная команда удаляет этот фрагмент видео. Для того чтобы просмотреть видео в онлайн режиме, был использован модуль Nginx rtmp [5] и js-библиотека JWplayer. Видео можно просмотреть как из приложения, так из веб-версии. Однако, чтобы скачать видео, стоит использовать веб-версию, потому что в большинстве случаев смартфон не сможет его воспроизвести.

Для организации данного сервиса главной технологией служат докеры[3], они используются для запуска таких процессов, как Nginx и его rtmp-модуль. Новый докер запускается, когда пользователь добавляет новую камеру. Nginx запускается с настройками, в которых переданы hash-данные[4] пользователя, который записан в базу данных, сгенерированных при регистрации. Докер запускается посредством командного интерфейса Laraver. Также в докер-контейнере запускается процесс записи видео с помощью утилиты FFmpeg с последующей записью в облако DropBox.

Таким образом, каждый процесс обработки камеры является изолированным друг от друга и никак не влияет на параллельные процессы. При добавлении камеры в базу данных PostgreSQL также записывается имя докер-контейнера, который создан под данную камеру, а при удалении останавливается докер-контейнер с заданным именем.

Категории, являющиеся «горячими точками»[5] в архитектуре приложения (рис. 2):

- аутентификация и авторизация (Authentication and Authorization);
- композиция (Composition);

[2] Dropbox — файловый хостинг компании Dropbox Inc., включающий персональное облачное хранилище, синхронизацию файлов и программу-клиент

[3] Докер — это открытая платформа для разработки, доставки и эксплуатации приложений

[4] Hash — результат обработки неких данных хеш-функцией (функция, реализующая алгоритм и выполняющая преобразование)

[5] «Горячие точки» (хот-спот, или англ. hotspot) — участок кода в программе, на который приходится большая часть исполняемых инструкций процессора или на исполнение которого процессор затрачивает очень много времени

- управление конфигурацией (Configuration Management);
- связанность и сцепление;
- доступ к данным (Data Access);
- взаимодействие с пользователем (User Experience).

Горячие точки соответствуют разным сквозным функционалам, которые используются при создании приложений, и, соответственно, разным наборам паттернов и практик. Таблица 1 перечисляет ключевые вопросы для каждой горячей точки.

Аутентификация. Для аутентификации разработан отдельный Angular-модуль, работающий со стандартным функционалом Laravel [6]. Авторизовать запросы можно с помощью PHP-сессий, которые передаются из запроса в запрос каждого пользователя. Между запросами передаются данные пользователя, которые изначально при авторизации кладутся в сессию. При обновлении данных синхронизируются данные сессии с уже обновленными данными.

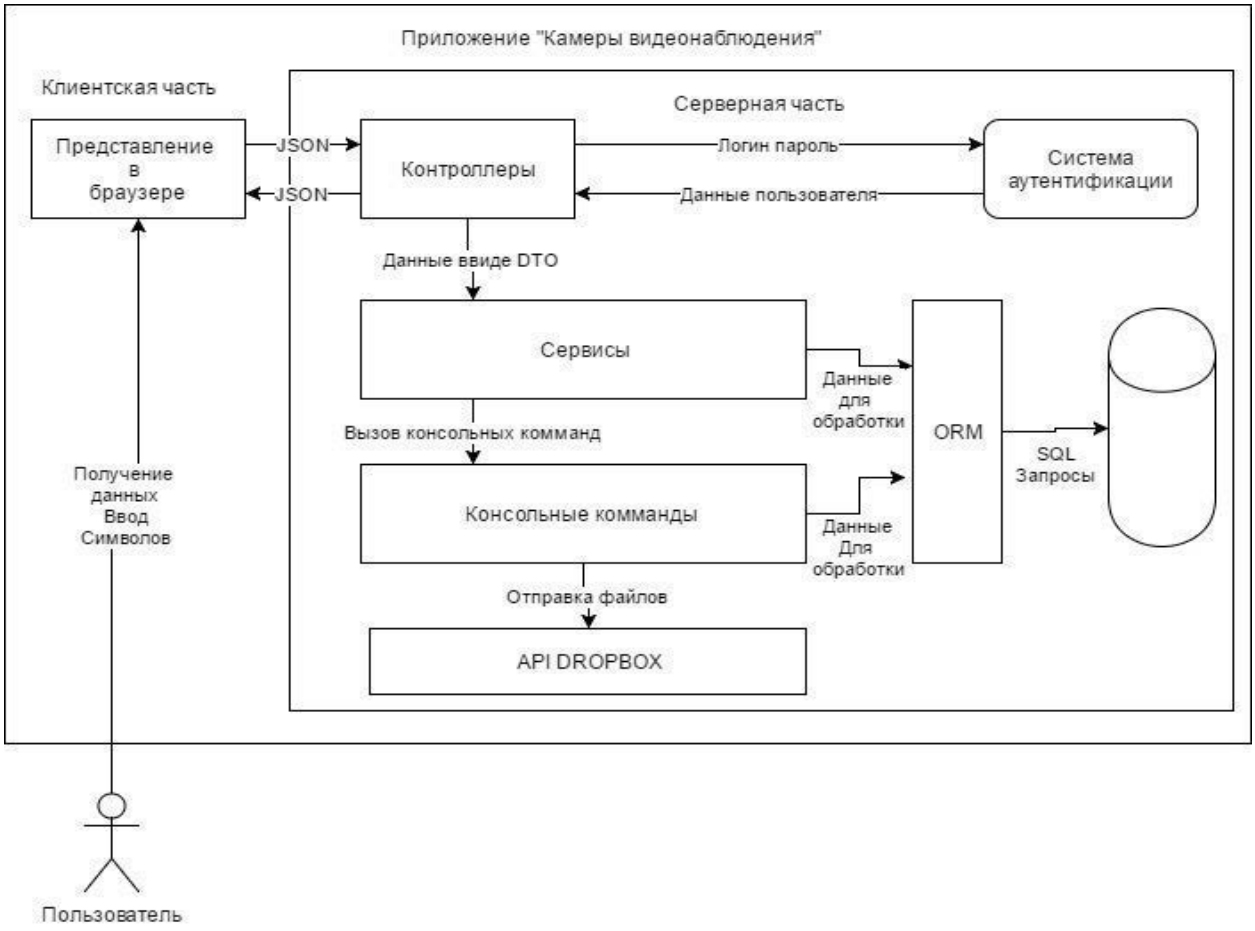

Рис. 2. Архитектура системы

Композиция. Для слабой связанности компонентов приложения используются три вида паттерна проектирования: MVC; Service; Repository. Паттерн проектирования предназначен для того, чтобы отделить представление от логики приложения.

Для разработки с использованием архитектурной модели MVC (англ. Model View Controller – модель-представление-контроллер) используется Laravel framework (с версей php 7.0) [7].

Service – предназначен для того, чтобы отделить входные данные от их обработки. Для реализации данного паттерна использовалась технология сервис контейнеров фреймворка Laravel для более гибкой реализации логики. Данная технология позволяет внедрять реализацию определенных ранее интерфейсов, которые представляют собой набор функций приложения.

Паттерн Repository предназначен для отделения работы с базой данных от логики приложения. Для его реализации использован Eloquent – инструмент, встроенный в Laravel. Также используется Artisan [8] для разработки команд для cmd.

Таблица 1. Ключевые вопросы «горячих точек»

Категория	**Ключевые вопросы**
Аутентификация	- Как хранить идентификационные данные зарегистрированных пользователей? - Как аутентифицировать запросы? - Как передавать идентификационные данные пользователей между слоями приложения?
Композиция	- Как спроектировать структуру приложения? - Как спроектировать слабую связанность между модулями? - Как работать с зависимостями в слабосвязанном режиме?
Управление конфигурацией	- Как определить какая информация подлежит конфигурированию? - Как определить где и каким образом сохранять конфигурационную информацию? - Как работать с секретной информацией?

	- Как работать с конфигурационной информацией в кластере?
Связанность и сцепление	- Как разделить функциональности? - Как структурировать приложение? - Как выбрать подходящее разбиение на слои? - Как задать границы между частями системы?
Доступ к данным	- Как управлять соединениями БД?
Взаимодействие с пользователем	- Как повысить эффективность выполнения задач? - Как улучшить отзывчивость интерфейса? - Как улучшить возможности приложения для пользователя?

Связанность и сцепление. Структура приложения обоснована структурой фреймворка Laravel. Корневой каталог свежеустановленного Laravel содержит ряд папок:

- папка app, содержит код ядра приложения (далее будет рассмотрена подробнее);
- папка bootstrap содержит несколько файлов, которые загружают фреймворк и настраивают автозагрузку, папка cache содержит несколько файлов для оптимизации процесса автозагрузки, сгенерированных фреймворком;
- папка config содержит все конфигурационные файлы приложений;
- папка database содержит миграции и классы для наполнения начальными данными БД; при необходимости эту папку можно использовать для хранения базы данных SQLite;
- папка public содержит фронт-контроллер и ресурсы (изображения, JavaScript, CSS и т. д.);
- папка resources содержит представления, сырые ресурсы (LESS, SASS, CoffeeScript) и «языковые» файлы;
- папка storage содержит скомпилированные Blade-шаблоны, файл-сессии, кэши файлов и другие файлы, создаваемые фреймворком; эта папка делится на подпапки app, framework и logs; в папке app можно хранить любые файлы, используемые вашим приложением; в папке framework хранятся создаваемые

фреймворком файлы и кэш, а в папке logs находятся файлы журналов приложения;

- папка tests содержит автотесты; изначально там уже есть пример PHPUnit;.
- папка vendor содержит Composer-зависимости.

Доступ к данным. За основу был взят PostgreSQL, свободная объектно-реляционная система управления базами данных [9]. Для общения приложения и базы данных используется система объектно-реляционного отображения Eloquent – красивая и простая реализация шаблона ActiveRecord в Laravel для работы с базами данных. Каждая таблица имеет соответствующий класс-модель, который используется для работы с этой таблицей. Модели позволяют запрашивать данные из таблиц, а также вставлять в них новые записи.

Взаимодействие с пользователем.

1. Форма для регистрации. Данная страница предназначена для новых пользователей, прежде всего необходимо зарегистрироваться, чтобы осуществить вход в аккаунт.

2. Форма входа предназначена для пользователей, которые зарегистрированы в системе.

3. Профиль пользователя. На данной странице пользователь может обновить информацию о себе, такую, как Имя, Фамилия, а также почтовый адрес.

4. Страница добавления камер, а также просмотр текущего статуса камеры, Stream-потока камеры.

5. Страница с видеозаписями пользователя. На этой странице пользователь может посмотреть видеозаписи камер, список камер, количество свободного пространства, а также скачать видеозаписи.

6. Меню приложения. Пользователь может выбрать свои дальнейшие действия, а также изменить информацию.

РЕЗУЛЬТАТЫ

Было проведено нагрузочное тестирование, на основании результатов которого можно сделать вывод, что система способна подключать одновременно и получать данные транслирования с достаточно большого количества разных устройств видеонаблюдения. В зависимости от разных конфигураций серверов, на которых расположена серверная часть, это может быть от 100 до 1000 камер одновременно. При увеличении количества пользователей и используемых ими

камер необходимо переходить на другую серверную архитектуру, предлагаемую в [1, 2].

Использовались мобильные устройства и различные камеры – уличные камеры с открытым доступом, камеры с мобильных устройств, а также IP-камеры.

Разработанная система предлагает как простоту шаринга видеопотока (совместного использования карты условного доступа) для заинтересованных пользователей без увеличения нагрузки на сеть, так и использование веб-камер абсолютно разного типа.

Устройства, которые были использованы при тестировании системы:

- Мобильные устройства: Xiaomi Redmi 3S; Samsung Galaxy S5; Sony Xperia Z3; LG G5; HTC One M8; Asus ZenFone 3;
- Камеры видеонаблюдения: CMD IP1080-WB3,6IR V2; VStarcam C7815WIP; VStarcam T7892WIP; IPC-HFW1300S-0360B;
- Онлайн камеры.

На рисунке 3 показаны страница добавления новой камеры и страница подключенных устройств камер видеонаблюдения.

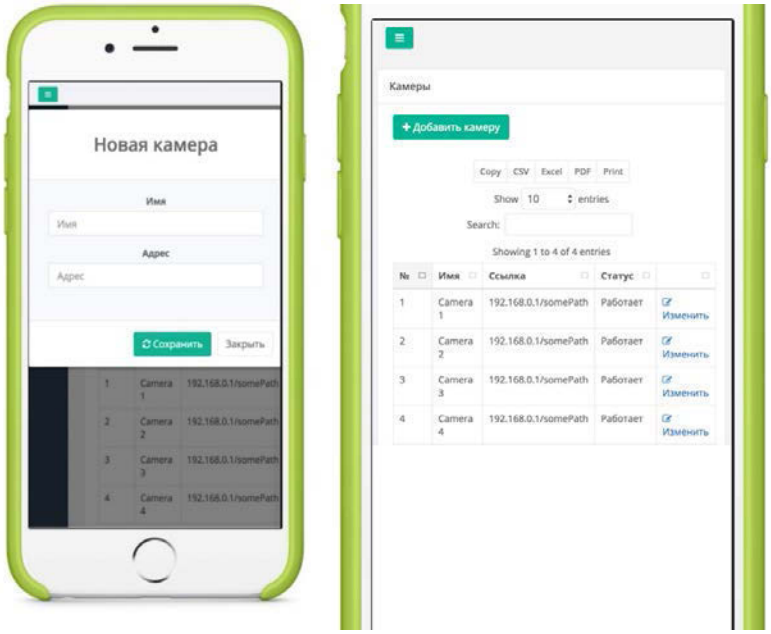

Рис. 3. Добавление новой камеры. Информация о камерах

В интерфейсе для мобильных приложений пользователю доступны: просмотр видеозаписей, доступных камер, информации о них, информация о свободном дисковом пространстве в его персональном облаке, работа с его спис-

ками групп: «друзья», «соседи», «работа» и т. д. Например, на рисунке 4 приведен список видео, записанных с данных камер, которые можно как просмотреть, так и скачать для просмотра. Пользователю доступны все записи с камер, а также свободное дисковое пространство.

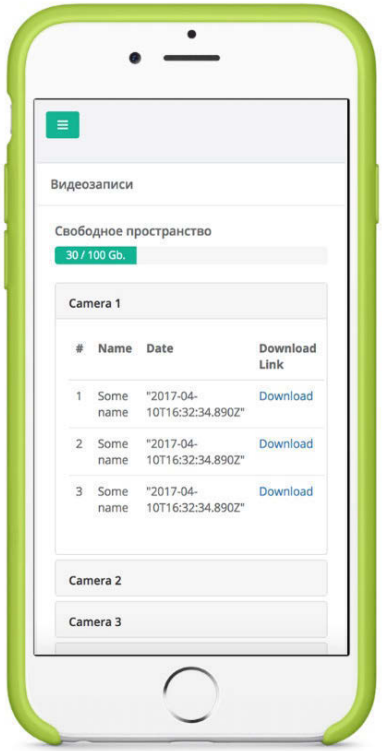

Рис. 4. Страница с видеозаписями пользователя

Проведенные работы положили начало минимально жизнеспособной версии, на основе которой в дальнейшем можно реализовать, например:

- на уровне облачных сервисов такие функции, как:

1. создание службы настройки и обработки видео [10];

2. создание службы распознавания человеческих лиц [11], животных [12], номерных знаков [13], погодных явлений, стихийных бедствий [14];

3. создание функции резервного копирования [10];

4. протоколирование событий по расписанию (праздники, ночное время);

- на клиентском уровне [2]:

1. создание приложения монитора (использовать в качестве мониторинга и инструментов для обнаружения объектов и событий в реальном времени);

2. приложение тревоги (обнаруживается подозрительный человек в наблюдаемой области, о чем отправляется тревожное сообщение);

3. приложение отчета (может быть использовано для составления сводной информации о видеопотоках, камерах или использовании системы).

ЗАКЛЮЧЕНИЕ

Технологии, которые были выбраны в начале разработки системы, отвечающие за получение данных в онлайн режиме, были изменены на другие, более удобные, примером является фреймворк Laravel, который имеет легковесную библиотеку. К такому решение пришли в результате тестирования, которое показало, что эти процессы влияют на отклик приложения.

Разработанная система содержит открытый API[6], на основе которого можно расширять функционал для выполнения новых требований пользователей. Примерами такого функционала могут быть: создание базы биометрических данных на основании обработки изображений лиц; создание групп камер, которые связаны между собой местом наблюдения или другим тегированием; обучение системы распознаванию движения и эмоций; оповещения об энергии камер; интеграция с мессенджерами типа telegram и др.

СПИСОК ЛИТЕРАТУРЫ

1. *Choi K.I., Lee J.H., Lee B.C.* A Distributed Cloud Based Video Storage System with Privacy Protection // Int. Conf. on Advanced Communication Technology – ICACT 2017. P. 830–835.

2. *Sandar N.M., Chaisiri S., Yongchareon S., Liesaputra V.* Cloud-based Video Monitoring Framework: An Approach Based on Software-defined Networking for Addressing Scalability Problems // Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2015. V. 9051.

3. *Egilmez H.E., Dane S.T., Bagci K.T., Tekalp A.M., Koc Univ*. In Signal & Information Processing Association Annual Summit and Conference // APSIPA ASC, 2012 Asia-Pacific – Istanbul, Turkey, 2012. P. 1–8.

4. FFMPEG. https://ffmpeg.org/.

[6] API (программный интерфейс приложения, интерфейс прикладного программирования) (англ. *Application programming interface*, *API* [эй-пи-ай]) — набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) или операционной системой для использования во внешних программных продуктах. Используется программистами при написании всевозможных приложений

5. Nginx-rtmp-module. https://github.com/arut/nginx-rtmp-module.
6. AngularJS. https://angularjs.org/.
7. Laravel. https://github.com/laravel/framework.
8. Artisan. http://laravel.su/docs/5.0/artisan.
9. PostgreSQL. https://www.postgresql.org/.
10. *Dasu A., Panchanathan S.* A Survey of Media Processing Approaches. Circuits and Systems for Video Technology // IEEE Transactions on. 2002. V. 12, No 8. P. 633–645.
11. *Connolly J. F., Granger E., Sabourin R.* An Adaptive Classification System for Video-based Face Recognition // Information Sciences. 2012. V. 192. P. 50–70.
12. *Burghardt T., Cali´c J*. Analysing Animal Behaviour in Wildlife Videos Using ´ Face Detection and Tracking // IEE Proceedings-Vision, Image and Signal Processing. 2006. V. 153, No 3. P. 305–312.
13. *Du S., Ibrahim M., Shehata M., Badawy W.* Automatic License Plate Recognition (ALPR): A State-of-the-art review // Circuits and Systems for Video Technology. IEEE Transactions on. 2013. V. 23, No 2. P. 311–325.
14. *Lai C.L., Yang J.C., Chen Y.H*. A Real Time Video Processing Based Surveillance System for Early Fire and Flood Detection // Instrumentation and Measurement Technology Conference Proceedings. 2007. IMTC 2007. IEEE. P. 1–6.

FRAMEWORK FOR CLOUD-VIDEO MONITORING VIA IP-CAMERAS WITH EASILY ACCESSIBLE CONTROL FOR USERS

A.S. Grishina[1], V.V. Kugurakova[2].

[1,2] Higher School ITIS. Kazan Federal University
[1] sinerysmile@gmail.com, [2] vlada.kugurakova@gmail.com

Abstract

The article describes the main points of the process of creating a system that allows you to manage several cameras simultaneously while saving data on the server. The system has the ability to connect IP cameras and cameras on mobile devices,

provide access to other users, and also allows you to watch video online. Hotspots in the architecture of the system have been identified and described. A separate Angular module was developed and design patterns were used. The interaction of the system with the user is described. The stages of further development of cloud video monitoring through ip-cameras with easily accessible control for the end-user are offered.

Keywords: *IP-camera, system, server, application, video surveillance.*

REFERENCES

1. *Choi K.I., Lee J.H., Lee B.C.* A Distributed Cloud Based Video Storage System with Privacy Protection // Int. Conf. on Advanced Communication Technology – ICACT 2017. P. 830–835.

2. *Sandar N.M., Chaisiri S., Yongchareon S., Liesaputra V.* Cloud-based Video Monitoring Framework: An Approach Based on Software-defined Networking for Addressing Scalability Problems // Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2015. V. 9051.

3. *Egilmez H.E., Dane S.T., Bagci K.T., Tekalp A.M., Koc Univ*. In Signal & Information Processing Association Annual Summit and Conference // APSIPA ASC, 2012 Asia-Pacific – Istanbul, Turkey, 2012. P. 1–8.

4. FFMPEG. https://ffmpeg.org/.

5. Nginx-rtmp-module. https://github.com/arut/nginx-rtmp-module.

6. AngularJS. https://angularjs.org/.

7. Laravel. https://github.com/laravel/framework.

8. Artisan. http://laravel.su/docs/5.0/artisan.

9. PostgreSQL. https://www.postgresql.org/.

10. *Dasu A., Panchanathan S.* A Survey of Media Processing Approaches. Circuits and Systems for Video Technology // IEEE Transactions on. 2002. V. 12, No 8. P. 633–645.

11. *Connolly J. F., Granger E., Sabourin R.* An Adaptive Classification System for Video-based Face Recognition // Information Sciences. 2012. V. 192. P. 50–70.

12. *Burghardt T., Cali´c J*. Analysing Animal Behaviour in Wildlife Videos Using ´ Face Detection and Tracking // IEE Proceedings-Vision, Image and Signal Processing. 2006. V. 153, No 3. P. 305–312.

13. *Du S., Ibrahim M., Shehata M., Badawy W.* Automatic License Plate Recognition (ALPR): A State-of-the-art review // Circuits and Systems for Video Technology. IEEE Transactions on. 2013. V. 23, No 2. P. 311–325.

14. *Lai C.L., Yang J.C., Chen Y.H*. A Real Time Video Processing Based Surveillance System for Early Fire and Flood Detection // Instrumentation and Measurement Technology Conference Proceedings. 2007. IMTC 2007. IEEE. P. 1–6.

СВЕДЕНИЯ ОБ АВТОРАХ

ГРИШИНА Анастасия Сергеевна – бакалавр Высшей школы ИТИС Казанского (Приволжского) Федерального университета. Сфера научных интересов – обработка видеопотока, облачный видеомониторинг, распознавание объектов и другой информации в видеопоследовательностях.

Anastasia Sergeevna GRISHINA, has bachelor's degree of the Higher School of ITIS Kazan (Privolzhsky) Federal University. Research interests include video stream processing, cloud video monitoring, recognition of objects and other information in video sequences.

email: sinerysmile@gmail.com.

КУГУРАКОВА Влада Владимировна – старший преподаватель Высшей школы информационных технологий и информационных систем, руководитель лаборатории «Виртуальные и симуляционные технологии в биомедицине». Сфера научных интересов – реалистичность визуализации и симуляций, иммерсивность VR.

Vlada Vladimirovna KUGURAKOVA, Senior Lecturer of Higher School of Information Technology and Information Systems, Head of Laboratory "Virtual and simulation technologies in biomedicine". Research interests include realism of visualization and simulation, immersion VR.

email: vlada.kugurakova@gmail.com.

Материал поступил в редакцию 2 июня 2017 года