6. Jin, Q. Voice Converging: Speaker De-identification by Voice Transformation / Q. Jin, A.R. Toth, T. Schultz, A. W. Black // 2009 IEEE International Conference on Acoustics, Speech and Signal Processing. – 2009. – P.3909-3912. – DOI: 10.1109/icassp.2009.4960482.

7. Jin, Q. Speaker De-identification via Voice Transformation / Q. Jin, A.R. Toth, T. Schultz, A. W. Black // Proceedings of the 2009 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2009. – 2009. – P. 529-533. – DOI: 10.1109/ASRU.2009.5373356.

8. Lal Srivastava, B.M. Evaluating Voice Conversion-Based Privacy Protection against Informed Attackers / B. M. Lal Srivastava, N. Vauquier, M. Sahidullah, A. Bellet, M. Tommasi, E. Vincent // 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). – 2020. – P.2802-2806. – DOI: 10.1109/ICASSP40776.2020.9053868.

9. Mukhopadhyay, D. All Your Voices are Belong to Us: Stealing Voices to Fool Humans and Machines / D. Mukhopadhyay, M. Shirvanian, N. Saxena // Lecture Notes in Computer Science – 2015. – Vol. 9327. – P. 599-621. – DOI: 10.1007/978-3-319-24177-7_30.

10. Neupane, A. The Crux of Voice (In)Security: A Brain Study of Speaker Legitimacy Detection / A. Neupane, N. Saxena, L. Hirshfield, S. Bratt // Network and Distributed Systems Security Symposium (NDSS 2019). – 2019. – DOI: 10.14722/ndss.2019.23206.

Р.Ш. Шарипов, Л.С. Зеленко

СИСТЕМА МОНИТОРИНГА СОСТОЯНИЙ ОТКАЗОУСТОЙЧИВОЙ РАСПРЕДЕЛЕННОЙ АРХИТЕКТУРЫ ПРОГРАММНОГО КОМПЛЕКСА «КОНТРОЛЬ ОХРАНЫ ТРУДА»

(Самарский университет)

В настоящее время многие большие программные системы являются распределенными, они состоят из нескольких автономных компьютеров, но при этом обработка информации сосредоточена не на одной вычислительной машине, а распределена между несколькими. Важным преимуществом таких систем является то, что они упрощают интеграцию различных приложений, работающих на разных компьютерах, в единую систему и хорошо масштабируются. Их размер ограничивается только размером базовой сети. Платой за эти преимущества часто является очень сложное программное обеспечение, падение производительности и особенно проблемы с безопасностью. Тем не менее, заинтересованность в построении и внедрении распределенных систем наблюдается повсеместно [1].

К числу распределенных систем относится программный комплекс (ПК) «Контроль охраны труда (КОТ)», разработанный ООО «СМС-Информационные технологии». Для того чтобы поддерживать работу такой

сложной системы, необходимо отслеживать состояние ее основных компонентов. Для этой цели разрабатывается система мониторинга состояний распределённой отказоустойчивой архитектуры.

Распределенная архитектура ПК «КОТ» представляет собой балансировщик и четыре ЭВМ (см. рисунок 1). Две ЭВМ используются как серверы приложения, две – как серверы базы данных.

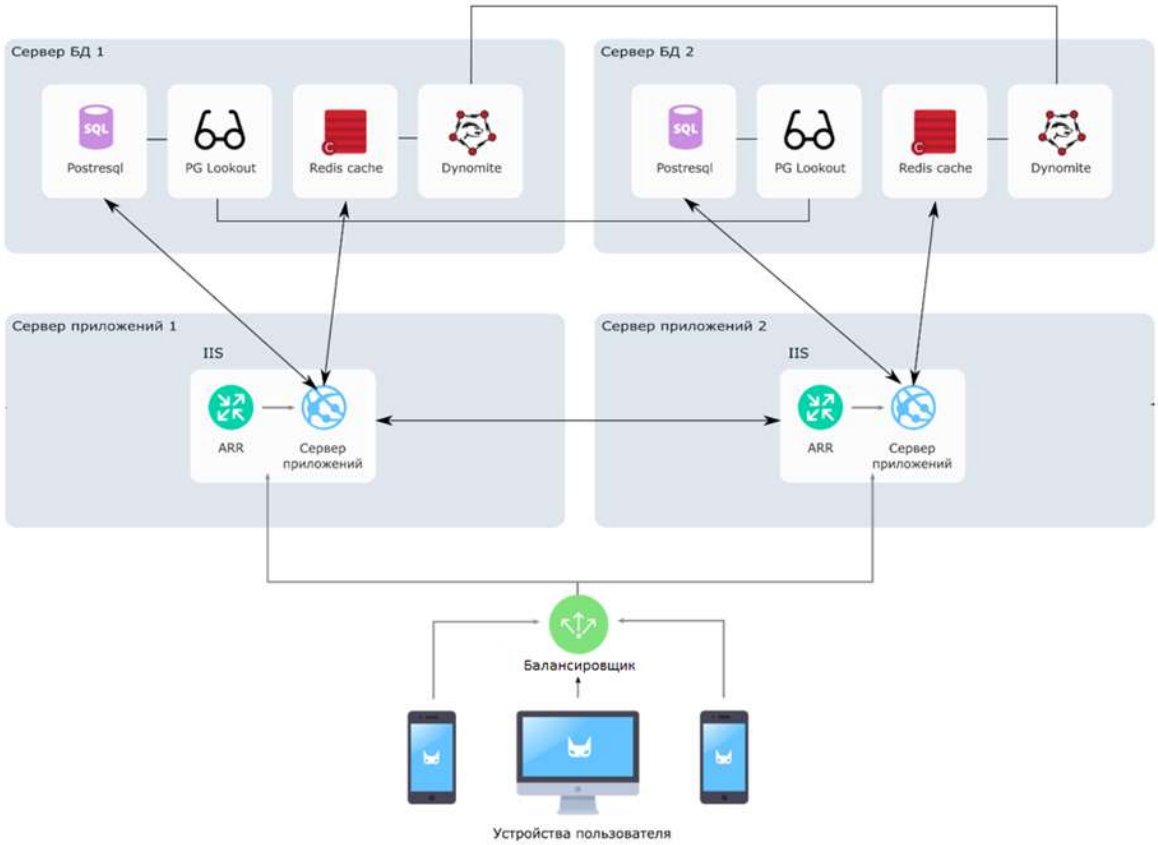

Рисунок 1 – Схема распределенной архитектуры ПК «Комплекс охраны труда»

Балансировщик NLB предназначен для проверки работоспособности двух серверов приложений и содержит в себе информацию об их состоянии. При необходимости он перенаправляет все запросы на оставшийся в рабочем состоянии сервер.

На каждом из серверов приложений установлен веб-сервер IIS (Internet Information Services), они будут связаны друг с другом. IIS позволяет настроить балансировку запросов. Маршрутизатор запросов ARR (Application Request Routing) подвергает запросы, поступившие на IIS, балансировке, запросы распределяются равномерно между двумя ЭВМ.

На двух ЭВМ сервера БД для синхронизации оперативной памяти применяется резидентная база данных Redis, в ней хранится вся оперативная информация, которую необходимо перераспределить между ЭВМ. Так же на серверах БД установлена утилита Dynomite, которая обеспечивает передачу данных между базами данных Redis.

На каждом сервере БД установлена утилита PgLookout, она связывается с другим сервером для проверки его работоспособности и при необходимости переключает приоритеты баз данных.

Система мониторинга представляет собой web-приложение на базе двухзвенной клиент-серверной архитектуры «клиент-сервер», взаимодействие между сервером и клиентами осуществляется по протоколу HTTP. На рисунке 2 представлена структурная схема системы мониторинга. На серверной части системы расположены подсистемы отправки и обработки запросов на локальные и удаленные ЭВМ, а также подсистема отправки сообщений на электронную почту в случае возникновения нештатных ситуаций, на клиентской части системы расположены подсистема вывода информации о компонентах системы и подсистема, с помощью которой можно было бы добавлять детекторы, определяющие работоспособность компонентов системы.

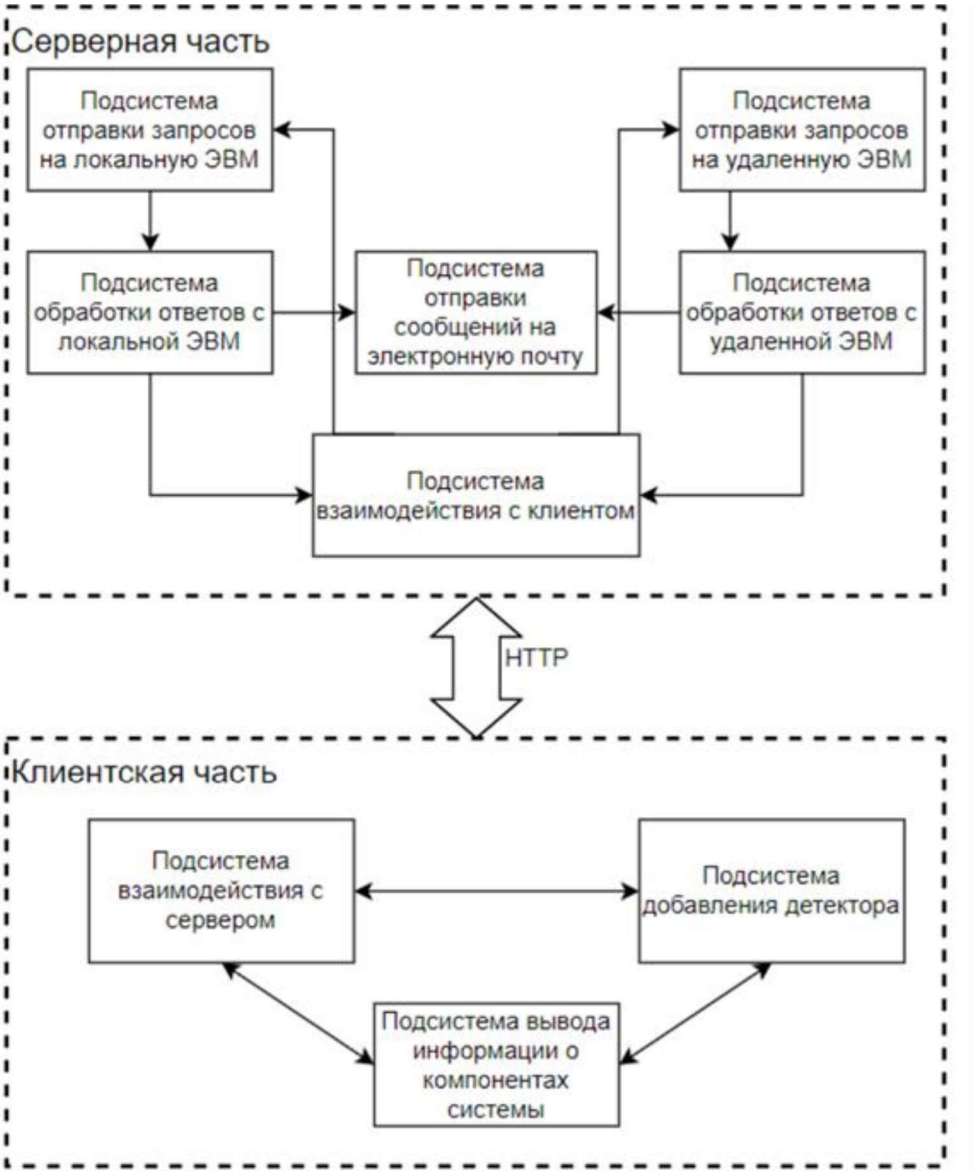

Рисунок 2 – Структурная схема системы мониторинга

Для разработки были выбраны языки C# и TypeScript, в качестве среды разработки выбрана среда JetBrains Rider.

На рисунке 3 приведена главная страница приложения, здесь отображается информация о состоянии компонентах системы (если компонент находится в рабочем состоянии, то в соответствующей строке стоит символ "+", если не в рабочем состоянии, то стоит символ «–»), время последний проверки.

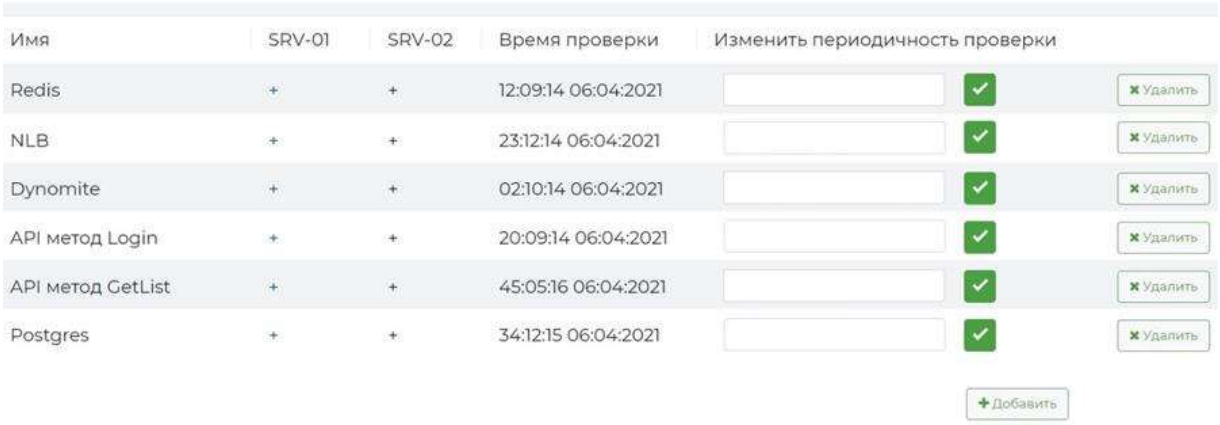

Рисунок 3 – Главная страница приложения

При добавлении детектора пользователь должен задать (рисунок 4): имя детектора, выбрать его тип (Redis, NLB, Dynomite, API-методы, БД postgres), задать IP-адрес, задать номер порта, задать частоту проверки в секундах, установить флаг отправки электронного письма.

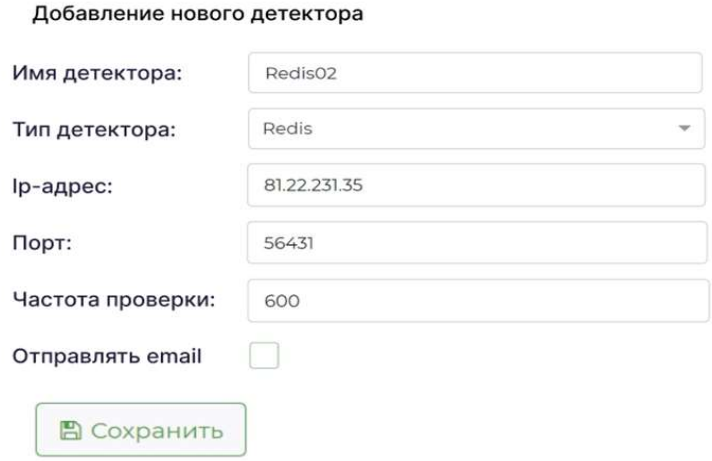

Рисунок 4 – Страница добавления детектора

Если все поля заполнены и данные корректные, то происходит сохранение детектора в файл, в противном случае выдается сообщение о некорректности заполнения полей.

Разработанная система находится в опытной эксплуатации.

Литература

1 Востокин С. В. Архитектура современных распределённых систем: электрон. учеб.-метод. комплекс по дисциплине в LMS Moodle / Мин-во образования и науки РФ, Самар. гос. аэрокосм. ун-т им. С. П. Королева (нац. исслед. ун-т). Электрон. текстовые и граф. дан. Самара, 2013. 1 эл. опт. диск (CD-ROM).