УДК 004.052.32

Разуваев Д.В.
студент магистратуры
2 курс, факультет компьютерных наук
Воронежский государственный университет
Россия, г. Воронеж

ПОСТРОЕНИЕ СИСТЕМЫ ДЛЯ АВТОМАТИЗАЦИИ ПРОВЕРОК РАБОТОСПОСОБНОСТИ БИЗНЕС-СЦЕНАРИЕВ В РАСПРЕДЕЛЕННЫХ СИСТЕМАХ

Аннотация: *В настоящее время большинство систем переходят из централизованного управления на распределенную модель. Однако, намного труднее понять и оценить свойства распределенных систем в целом, их сложнее проектировать, тестировать и обслуживать. Решением данной проблемы является разработка и использование системы мониторинга, которая позволяет оценить состояние распределенной среды с точки зрения бизнеса и бизнес-процессов. Основные проблемы разработки такой системы рассматриваются в данной работе.*

Ключевые слова: *распределенные системы, системы реального времени, мониторинг, Java, Spring.*

Annotation: *Currently, most systems are moving from centralized management to a distributed model. However, it is much more difficult to understand and evaluate the properties of distributed systems, they are more difficult to design, test and maintain. The solution of this problem is the development and using of a monitoring system that allows you to assess the state of the distributed environment in terms of business and business processes. The main problems of developing such system are considered in this paper.*

Key words: *distributed systems, real-time systems, monitoring, Java, Spring.*

Одним из вариантов решения проблемы усложняющихся приложений, требующих больших мощностей сервера, к которому прибегает все больше разработчиков, является создание распределенной системы.

Для распределённых систем характерно распределение функций между множеством узлов, которые взаимодействуют между собой по сети, и отсутствие единого управляющего центра. Так как все узлы системы считаются равнозначными, выход из строя одного из узлов не приводит к полной остановке всей системы.

Однако, намного труднее понять и оценить свойства распределенных систем. Их сложнее проектировать и обслуживать. Из-за сложности при разработке и тестировании распределенных систем, часто возникает проблема с трудоемким и долгим процессом идентификации ошибки, возникшей в реальном времени [1].

Из-за того, что ошибки в системе напрямую влияют на функционирование бизнес-процессов, для которых система разрабатывалась, важно обеспечивать надежность приложения.

Надежность обеспечивается системами мониторинга, позволяющими в реальном времени оценить состояние всей системы. Недостатком большинства существующих систем мониторинга является направленность исключительно на технические характеристики, не затрагивая при этом предметную область и бизнес.

Предлагаемая система позволит установить соответствие между бизнес-сценариями компании и средствами реализации системы. Это позволяет абстрагироваться от технических деталей и следить за исправностью бизнес-процессов.

Основными целями разработки данного приложения являются автоматизация проверки правильности работы бизнес-сценариев и мониторинг состояния системы и ее подсистем.

Также очевидно, что сама система должна быть распределенной, для обеспечения отказоустойчивости и минимизации нагрузки на отдельные узлы сети.

Для упрощения автоматизации проверки правильности работы бизнес-сценариев можно предоставить health check функциональность. Health check — это периодически выполняемая операция, по результатам которой можно определить состояние системы [2]. В качестве проверок по умолчанию планируется реализовать запросы к базам данных Oracle, NoSql хранилищам Apache Cassandra, кэшам Hazelcast и Apache Ignite, системам сетевого обнаружения Apache Zookeeper и Consul, брокерам сообщений Apache Kafka, серверам логов Graylog.

Предлагаемая к разработке система может помочь сконфигурировать стандартные проверки работоспособности различных бизнес кейсов, проверять их при необходимости решать возникшие ошибки, а также запускать стандартные автоматические скрипты для их устранения.

Механизм работы системы устроен следующим образом:

1. Производится описание бизнес-сценариев, а также условий, выполнение которых необходимо для их работоспособности.
2. Производится настройка источников данных системы.
3. Система разворачивается на окружении.
4. Осуществляется подключение к источникам данных.
5. Затем может производиться периодическое выполнение проверок работоспособности конкретных сценариев.

Для наглядности можно привести возможные примеры сценариев работы системы, для которых осуществляется проверка.

Рассмотрим как пример бизнес-сценария “Регистрация пользователя” в некоторой распределенной среде. Предположим, что система построена с использованием микросервисной архитектуры. Тогда можно выделить следующие подсистемы, участвующие в сценарии:

— API gateway – сервис, который производит маршрутизацию запросов клиентов на необходимый сервис;

— микросервис для управления пользователями - сервис, обрабатывающие данные пользователя и сохраняющий их в Oracle базу данных;

— Oracle база данных – хранилище пользовательских данных;

— микросервис для оповещения пользователей – сервис, производящий отправку email сообщений об успешной регистрации пользователя.

Можем предположить, что запрос пользователя о регистрации попадает в API gateway, который логирует информацию о полученном запросе, и отправляет запрос на микросервис для управления пользователями. Данный сервис обрабатывает данные пользователя, и в случае успешной регистрации логирует необходимую информацию, сохраняет нового пользователя в базе данных, после чего отправляет запрос на API gateway, который логирует запрос и перенаправляет его на сервис оповещения пользователей, который создает запись в логах об успешно отправленном сообщении.

Тогда можно выделить следующие шаги и проверки бизнес-сценария:

— шаг проверки API gateway № 1, содержащий проверки, что микросервис активен, а также что в логах микросервиса появилась необходимая информация о запросе;

— шаг проверки микросервиса для управления пользователями, содержащий проверки, что микросервис активен, что в логах нет ошибок связанных с регистрацией пользователя, а также что логи об успешной регистрации пользователя присутствуют;

— шаг проверки Oracle базы данных, содержащий проверку, что в базе данных появились новые строки, связанные с новым пользователем;

— шаг проверки API gateway № 2, содержащий проверки, что в логах микросервиса появилась необходимая информация о запросе с сервиса управления пользователями;

— шаг проверки микросервиса для оповещения пользователей, который содержит проверки, что сервис активен, а также что в логах микросервиса есть запись об успешно отправленном сообщении.

В качестве языка программирования был выбран Java, а также использовался популярный фреймворк Spring. На клиентской стороне использовался язык Typescript и фреймворк Angular.

В качестве способа хранения конфигурации было решено выбрать xml-файлы. XML представляет из себя расширяемый язык разметки. Данный язык обладает формальным синтаксисом, с которым одновременно удобно работать из Java кода, а также удобно работать пользователю.

Для обеспечения надежности систем и бизнес-процессов, необходимо постоянное отслеживание ошибок, с точки зрения бизнеса, и их устранение. Реализованное приложение позволяет упростить этот процесс, позволяя осуществить единоразовую настройку соответствия между бизнес-сценариями и средствами их технической реализации на предметно-ориентированном языке. Таким образом, данный проект позволяет автоматизировать проверки правильности работы системы с точки зрения бизнеса. Это в свою очередь важный аспект в повышении надежности проверяемой системы.

Использованные источники:

1. Страх и ненависть в распределенных системах. [Электронный ресурс]. URL: https://habr.com/ru/post/322876/ (дата обращения 14.05.2019)
2. Consul.io. Часть 2 [Электронный ресурс]. URL: https://habr.com/ru/post/278101/ (дата обращения 16.05.2019)