МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра дискретной математики и алгоритмики

ЖИДКОВ

Александр Геннадьевич

МЕТОДЫ И ТЕХНОЛОГИИ МОНИТОРИНГА КРУПНОМАСШТАБНЫХ ГЕТЕРОГЕННЫХ СИСТЕМ

Магистерская диссертация

Специальность 1-31 81 09 «Алгоритмы и системы обработки больших объемов информации»

Научный руководитель:
Александр Николаевич Вальвачев
кандидат технических наук, доцент

Допущена к защите

«___» _____ 2020 г.

Зав. кафедрой дискретной математики и алгоритмики
доктор физико-математических наук, профессор

_____ В. М. Котов

Минск, 2020

РЕФЕРАТ

Магистерская диссертация, 49 страниц, 10 рисунков, 53 источника, 1 приложение.

Ключевые слова: МОНИТОРИНГ, КРУПНОМАСШТАБНЫЕ СИСТЕМЫ, СВЕРХБОЛЬШИЕ СИСТЕМЫ, СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ.

Объект исследования – теория систем, методы и технологии мониторинга систем, крупномасштабные гетерогенные системы.

Предмет исследования – применение теории систем в целях реализации системы, выполняющей мониторинг крупномасштабных гетерогенных систем.

Цель работы – выполнить анализ литературы по теме теории систем, определить актуальные проблемы мониторинга крупномасштабных гетерогенных систем, классифицировать объекты мониторинга, разработать модели и алгоритмы для мониторинга, на базе которых реализовать систему мониторинга крупномасштабных систем, которая будет обеспечивать поддержку принятия решений в аварийных ситуациях.

Результатами являются модели и алгоритмы мониторинга крупномасштабных гетерогенных систем, реализованная и корректно функционирующая система мониторинга и поддержки принятия решений.

Практическая значимость работы заключается в том, что результаты исследования могут использоваться для решения широкого спектра задач мониторинга, разработанная система мониторинга позволит отслеживать текущее состояние системы и сокращать убытки из-за задержек или ошибок во время принятия решений в аварийных ситуациях.

РЭФЕРАТ

Магістарская дысертацыя, 49 старонак, 10 малюнкаў, 53 крыніцы, 1 дадатак.

Ключавыя словы: МАНІТОРЫНГ, БУЙНАМАШТАБНЫЯ СІСТЭМЫ, ЗВЫШВЯЛІКІЯ СІСТЭМЫ, СІСТЭМЫ ПАДТРЫМКІ ПРЫНЯЦЦЯ РАШЭННЯЎ.

Аб'ект даследавання – тэорыя сістэм, метады і тэхналогіі маніторынгу сістэм, буйнамаштабныя гетэрагенныя сістэмы.

Прадмет даследавання – прымяненне тэорыі сістэм у мэтах рэалізацыі сістэмы, якая выконвае маніторынг буйнамаштабных гетэрагенных сістэм.

Мэта працы – выканаць аналіз літаратуры па тэме тэорыі сістэм, вызначыць актуальныя праблемы маніторынгу буйнамаштабных гетэрагенных сістэм, класіфікаваць аб'екты маніторынгу, распрацаваць мадэлі і алгарытмы для маніторынгу, на базе якіх рэалізаваць сістэму маніторынгу буйнамаштабных сістэм, якая будзе забяспечваць падтрымку прыняцця рашэнняў у аварыйных сітуацыях.

Вынікамі з'яўляюцца мадэлі і алгарытмы маніторынгу буйнамаштабных гетэрагенных сістэм, рэалізаваная і карэктна функцыянуючая сістэма маніторынгу і падтрымкі прыняцця рашэнняў.

Практычная значнасць работы заключаецца ў тым, што вынікі даследавання могуць выкарыстоўвацца для вырашэння шырокага спектру задач маніторынгу, распрацаваная сістэма маніторынгу дазволіць адсочваць бягучы стан сістэмы і скарачаць страты з-за затрымак або памылак падчас прыняцця рашэнняў у аварыйных сітуацыях.

ABSTRACT

Master thesis, 49 pages, 10 figures, 53 sources, 1 appendix.

Keywords: MONITORING, LARGE-SCALE SYSTEMS, EXTRA BIG SYSTEMS, DECISION SUPPORT SYSTEMS.

The object of the research is systems theory, systems monitoring methods and technologies, large-scale heterogeneous systems.

The subject of the research is to use systems theory to implement an application that monitors large-scale heterogeneous systems.

The purpose of the research is to perform literature analysis on the systems theory, identify current monitoring problems of large-scale heterogeneous systems, classify monitoring objects, develop models and algorithms for monitoring, implement a monitoring system for large-scale systems based on that models and algorithms, the developed system would provide decision support in emergency situations.

The results are models and algorithms for monitoring large-scale heterogeneous systems, an implemented and correctly functioning monitoring and decision support system.

The practical significance of the work is that the research results can be used to solve a wide range of monitoring tasks, the developed monitoring system will allow to monitor the current state of the system and reduce losses due to delays or errors during decision-making in emergency situations.

ОГЛАВЛЕНИЕ

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ

БД – база данных
ИИ – искусственный интеллект
ИКТ – информационные и коммуникационные технологии
КМС – крупномасштабные системы
ЛПР – лицо, принимающее решение
ОТС – организационно-техническая система
ПО – программное обеспечение

ВВЕДЕНИЕ

В XXI веке в результате глобализации мира, компьютеризации и интеллектуализации бизнес-процессов возник ряд новых типов организационно-технических систем [1]. К ним относятся: цифровые государства, умные города, структуры типа «интернет вещей», «киберфизические системы» и др. Они отличаются большими масштабами и гигантским количеством составляющих их разнородных или гетерогенных (разнородных) элементов. Научная литература по управлению новыми структурами начала формироваться в последние двадцать лет и в настоящее время находится в стадии становления [2, 6, 10, 11]. Отслеживать состояние и управлять такими системами чрезвычайно трудно, а ошибки или опоздание в принятии решений обходятся очень дорого. Поэтому актуальной является задача мониторинга крупномасштабных систем с поддержкой принятия оперативных решений для устранения различных аварийных ситуаций производственного, социального и бытового характера [2,3].

Для разработки такого рода систем мониторинга важно определить, какую понятийную базу, какие структуры данных и математические методы следует использовать для моделирования сложных и масштабных иерархий, как объективно оценить состояние крупномасштабных организационно-технических систем с миллионами элементов, как и кому получать от них информацию, обработать и правильно интерпретировать результат, а также оперативно принимать наиболее эффективные решения [4, 5, 6].

В данной работе представлен один из возможных вариантов комплексного решения этих задач на основе системного подхода, современной сетевой инфраструктуры Internet и ранее полученных результатов автора [22].

В качестве решения проблемы мониторинга крупномасштабных организационно-технических систем разработаны методы и программная система, ориентированная на применение в городах. Она позволит улучшить качество жизни населения за счет быстрой фиксации органами управления города (городским советом, мэрией) возникновения чрезвычайных ситуаций различного плана, назначения (автоматически или вручную) организации-исполнителя для устранения ситуации и оценки мнения жителей о качестве проделанной работы.

Система сможет собирать и обрабатывать информацию от различного вида источников. Это могут быть датчики, измеряющие уровень загрязнения воздуха или воды в конкретных местах, или сообщения жителей о их проблемах в различных сферах жизни, например: водо- и электроснабжение,

уборка общественных дворов, работа транспортных и дорожных служб и многое другое.

В первой главе диссертации выполнен анализ литературы и поставлена задача исследования. Во второй главе разработаны модели и алгоритмы для мониторинга крупномасштабных гетерогенных систем. В третьей главе выполнено исследование инструментов для программной реализации приложения и предложен подход к построению микросервисной архитектуры. В четвертой главе представлено разработанное программное обеспечение и результаты его применения для решения прикладной задачи.

ГЛАВА 1. ПРОБЛЕМА МОНИТОРИНГА КРУПНОМАСШТАБНЫХ СИСТЕМ

1.1 Анализ литературы

Исторически человеческое сообщество складывалось как множество организационно-технических систем (ОТС), которые эволюционировали от семьи, рода и племени до международных корпораций и экономических кластеров [1]. В XXI веке в результате глобализации и компьютеризации появились типы ОТС, включая цифровые государства, умные города, цифровая промышленность, умная энергетика, цифровое сельское хозяйство, цифровая медицина, киберфизические системы, интернет вещей и другие [1, 19]. В новых ОТС особый интерес представляют малоизученные крупномасштабные системы (КМС), которые быстро формируются из существующих и новых социальных, производственных и электронных компонентов. К ним относятся содружества государств (например, ЕС), экономические кластеры (деревообрабатывающий кластер Финляндии), цифровые государства (Эстония) и др. В процессе становления КМС выявляется множество сложных ранее неисследованных проблем. Особую актуальность имеет проблема гомеостаза КМС, т.е. обеспечения стабильности жизненного цикла КМС. Ее решение невозможно без разработки новых типов систем мониторинга, использующих новые подходы для сбора, хранения и обработки гигантских объемов данных о тысячах составляющих КМС объектах. Особенно хочется отметить актуальность систем сбора данных и мониторинга, направленных на улучшение качества повседневной жизни обычных людей, в последние годы они становятся всё более востребованными. Системы данного вида могут иметь совершенно различный функционал, но общая их черта заключается в том, что они позволяют выполнять мониторинг различных структур и объектов на уровне отдельного города, а также некоторые из них осуществляют поддержку принятия решений в зависимости от конкретной ситуации. В литературе такие ОТС называют большими [6, 20], сверхбольшими [4], сложными [7, 40] и крупномасштабными [23, 43]. При их создании и эксплуатации обнаружилось множество сложных проблем, опыт решения которых у человечества отсутствует в принципе [11, 19, 36]. Прежде всего возникли вопросы: какую понятийную базу и математическую структуру принять для моделирования столь сложных и масштабных иерархий, как объективно оценить состояние крупномасштабных ОТС с миллионами элементов, как и кому получать от них информацию, обработать, правильно интерпретировать результат и

принимать правильные решения в реальном масштабе времени [52]. Для ответа на эти вопросы целесообразно использовать аппарат общей теории систем [15, 29], так как любая из ОТС изначально является иерархической открытой системой [4]. Рассмотрим кратко теори. систем, текущую ситуацию и возможности применения системного подхода к анализу новых ОТС.

Предпосылки теории систем формировались, начиная с конца XIX-начала XX в. в работах Г.Ч.Брауна, Р.В.Селларса, А.Б.Новикова, А.А.Богданова. В 1930-1950-х г.г. Л. фон Берталанфи заложил основы общей теории систем (General System Theory) под влиянием биологии, которую в рамках различных подходов развивали У.Р.Эшби, Н.Винер, С.Бир, Д. фон Нейман, А.Холл, Г.Греневский, А.И.Берг, А.Ляпунов, Л.Жерар, М.Месарович, Р.Акофф, Г.Саймон, П.К. Анохин и другие. Вопросами проектирования больших компьютерных систем занимались Л.Гуд, Р.Макол, С.Янг, Д.Клир. В XXI в. различные направления теории систем развивали: В.Н. Бурков – по большим системам; В.Б.Тарасов – по большим системам, состоящих из интеллектуальных программных агентов; М.Ю.Охтилев – по большим системам сложных техногенных объектов; Д.А.Новиков – по активным системам; В.Н.Воронин – по оптимизации структур больших систем и т.д.

Несмотря на значительные успехи, общая теория систем окончательно не сформирована. Соответственно, имеется несколько вариантов базового понятия – системы. Приведем наиболее известные:

- По Берталанфи система – это комплекс элементов, находящихся во взаимодействии;
- По Акоффу система – это любая сущность, состоящая из взаимосвязанных частей;
- Фейджин и Холл утверждали, что система – это множество элементов с отношениями между ними и между их атрибутами;
- По Биру система – это взаимосвязь самых различных элементов или все, состоящее из связанных друг с другом частей;
- Месарович утверждал, что система - отображение входов и состояний объекта в выходах объекта;
- По Мыльнику система – это целенаправленный комплекс взаимосвязанных элементов любой природы и отношений между ними;
- По Анохину система – это совокупность взаимосвязанных элементов, направленная на достижение общей цели.

Разнообразие определений вызвало разнообразие классификации систем на абстрактные и материальные, дедуктивные и индуктивные, локальные и распределенные и т.д.

В данной работе в качестве основы взяты определения и классификация систем академика Берга [4, с.100-116], так как они носят упорядоченный комплексный характер и аккумулируют толкование базовых понятий различных отечественных и зарубежных школ:

1) Система – это некоторый объект, обладающий целостностью или рассматриваемый как целое.
2) Система имеет некоторый системообразующий параметр.
3) Система является частью каких-либо больших систем.
4) Возможность выделение из системы частей, меньших подсистем.
5) Большие системы – системы с большим числом элементов, состояний, связей между элементами, которые могут иметь связь с биологическими системами. С кибернетической точки зрения характерной чертой большой системы является иерархическая структура управления и большое количество источников информации из внешней среды.
6) Сверхбольшие или крупномасштабные системы – совокупность «индуктивно» связанных между собой больших систем. Их основной особенностью является повышенная стабильность (системный гомеостаз) входящих в них больших подсистем.

На основании определения из пункта 5 перечисленные выше новые ОТС можно рассматривать как крупномасштабные системы (КМС).

1.2 Актуальные проблемы КМС

На основании анализа литературы [19, 36] можно выделить важную проблему современных КМС: неразвитость моделей и методов для решения важнейших задач поддержки жизненного цикла (ЖЦ) и гомеостаза КМС. Прежде всего, это методы имитационного моделирования [5, 45], так как ставить физические эксперименты над системами такого масштаба невозможно. Разработка и применение таких методов позволит лицу, принимающему решение (ЛПР), видеть состояние и все составляющие на своем уровне в реальном масштабе и времени, получать рекомендации и оперативно принимать соответствующие решения при возникновении аварийных ситуаций. К сожалению, такие методы и системы только начали появляться на уровне больших систем (подсистем КМС). В результате решения часто запаздывают, что затрудняет устранение неблагополучных и аварийных ситуаций в различных областях: от мировых финансовых кризисов до загрязнения океанов. Яркий пример – финансовый кризис 2008 года,

вызванный запаздыванием реакции правительства США на крах крупнейших инвестиционных банков, включая Lehman Brothers.

Для решения проблемы запаздывания и неэффективности решений в КМС необходимо ответить, как минимум, на следующие вопросы:

- какую математическую структуру (в смысле Бурбаки) выбрать в качестве основы для моделирования КМС с учетом иерархии реальных ОТС;
- какие уровни иерархии (управления) необходимы в КМС;
- как нужно организовать потоки информации в КМС для того, чтобы обеспечить их обработки в реальном масштабе и времени на каждом уровне КМС;
- как получить, воспринять, оценить и принять обоснованное решение в отношении миллионов объектов КМС;
- как объективно оценить качество управления ЛПР каждого уровня;
- как унифицировать: показания датчиков различной природы (механические, электронные, химические и т.д.), трудносовместимые единицы измерения (температура, давление, скорость, вибрация) и шкалы оценок (от -1 до +1, от 0 до 42, от 0 до 5) и т.д.;
- как унифицировать архитектуру системы поддержки принятия решений для различных типов КМС;
- как унифицировать структуру программных модулей для разработки интероперабельных программных модулей.
- В целом, перечисленные вопросы вызывают две основные проблемы: рост неопределенности и запаздывание принятие решений.

1.3 Постановка задачи

В качестве основы используем классическую постановку задачи принятия решений [1, 25, 35]. Для простоты понимания будем считать, что КМС включает три уровня иерархии: центр (руководство), подсистемы, объекты. Трехуровневые структуры довольно часто встречаются на практике, например: президент, министерство машиностроения, заводы и фабрики.

Пусть имеется КМС, функционирующая в среде Env. Структура КМС включает центр (руководство) C и n гетерогенных подсистем $P_1 \ldots P_n$, каждая из которых может содержать большое количество производственных, сервисных, образовательных, социальных и других объектов $q_{11} \ldots q_{nr}, r \to \infty$.

ЛПР каждого уровня (C, P, Q) используют системы поддержки принятия решений (СППР), связанные корпоративными и глобальными коммуникациями:

Деятельность объектов q характеризуется множеством параметров $X_{q1},X_{qm},\ldots,X_{qr}$. В зависимости от их значений уровни иерархии КМС могут находиться в одном из возможных состояний $V=V_1,V_2,\ldots,V_z$ каждому из которых соответствует релевантное управляющее решение $U = U_1,U_2,\ldots,U_z$. Решение U_j может быть поддержано ЛПР Q, P, C или заменено на собственный вариант U_{jC}.

Требуется разработать систему мониторинга, обеспечивающую оценку всех составляющих КМС в реальном масштабе и времени.

Для достижения этой цели необходимо разработать:

- непротиворечивый понятийный каркас, ориентированный на практическое применение и отражающий реалии современных КМС;
- комплекс моделей, структурно и семантически релевантных специфике современных КМС и условиям решения задач в условиях неопределенности;
- комплекс алгоритмов, инвариантных количеству уровней КМС;
- архитектуру системы мониторинга, инвариантную типу КМС;
- унифицированную структуру программных модулей, обеспечивающих разработку релевантных модулей независимо от языка программирования;
- библиотеку программных модулей для реализации архитектуры.

Для проверки работоспособности программ целевой системы необходимо решить практическую задачу с большим количеством объектов.

Решение поставленных задач встречает ряд трудностей:

- большое количество объектов q и их гетерогенная природа затрудняют унификацию алгоритма оценки и построение интероперабельных программных модулей для различных компонентов;
- значения параметров объектов X носят как формальный (измерение), так и субъективный (мнение) характер и представлены различными типам данных (int, double, str, bool и другие);
- подходы к оценке состояния объекта, подсистемы и КМС могут отличаться, что приводит к неоднозначности алгоритма оценки;
- объем потока данных от объектов q к ЛПР P и C может превысить возможности человека воспринимать информацию, оперативно оценивать ситуацию и принимать адекватные решения;

- испытание целевой системы в рабочем режиме чревато отрицательными последствиям, поэтому необходим синтез продукта в двух вариантах: для имитационного моделирования и для практического внедрения.

В рамках описанной структуры актуальной является разработка КМС для улучшения качества жизни обычных людей в рамках отдельного города. Данная категория систем имеет абстрактный характер, поэтому определим некоторые функциональные особенности, которые требуется реализовать в разрабатываемой системе:

- Входной информацией могут являться данные с датчиков, а также сообщения жителей города об их проблемах;
- Возможность отследить ход решения проблем;
- Органы власти с помощью данной системы будут иметь возможность определить организацию-исполнителя для решения конкретной проблемы;
- Жители могут оценить качество и скорость решения проблемы, а также удовлетворенность работой государственных органов;
- При возникновении аварийных ситуаций система мониторинга может осуществлять поддержку принятия решений.

1.4 Подход к решению

Поставленная задача носит междисциплинарный характер (как и общая теория систем), поэтому будут использованы несколько подходов.

В качестве фундамента для решения предлагается использовать общую теорию систем [15, 29] и преимущества глобальной web-инфраструктуры.

Для моделирования центра, подсистем и объектов целесообразно использовать концептуальный (онтологический или топологический по Бурбаки) подход, так как он позволяет отображать любые сущности и отношения и уточнять их с учетом ограничений до уровня программного кода [17].

Для алгоритмической части существуют два основных подхода:

- наблюдательный, предложен в классической работе Ю.А.Израэля [13] и в его различных модификациях [22]. Предназначен для организации регулярных наблюдений за стационарными (постоянные координаты) медленно изменяющимися природными объектами (заповедниками, лесами, озерами и так далее). Решение принимается после окончания, как правило, длительного процесса наблюдений;
- проактивный, предложен в новейших работах О.В.Майдановича [26], М.Ю.Охтилева, Б.В.Соколова, Р.М.Юсупова [33], А.И.Кузьмича [24].

Применяется при наблюдении техногенных объектов, заключается в наблюдении параметров быстротекущих процессов и немедленное вмешательство при возникновении аварийных ситуаций без прерывания процесса наблюдений (часто дистанционных).

Согласно формулировке задачи, КМС могут включать как естественные (люди, животные, ландшафты), так и искусственные (города, заводы, электростанции, поезда) объекты. Возникновение в них аварийных ситуаций в некоторых случаях может привести к необратимым политическим, экологическим и социальным последствиям. Поэтому для мониторинга КМС следует применить проактивный подход, адаптировав его для применения к гетерогенным (разнородным) объектам с учетом их современной специфики (компьютеризация, глобальный коммуникации, искусственный интеллект.

Для распознавания состояний, в которых находятся составляющие КМС и выработки решений следует применять теорию распознавания образов [8, 44, 50] и теорию принятия решений [25, 35, 42].

Требование к моделям и алгоритмам: унификация, обеспечивающая создание интероперабельных программных модулей и единообразное применение для ЛПР всех уровней КМС, что позволит сократить время разработки целевой системы, снизить расходы на внедрение, обучение персонала и адаптацию к различным типам КМС.

1.5 Выводы

В первой главе получены следующие основные результаты:

- выполнен анализ литературы по тематике крупномасштабных систем;
- выделена проблема оценки и синтеза управляющих решений для большого количества объектов и уровней КМС в реальном масштабе и времени;
- сформулирована общая постановка задачи на разработку методов и технологии, обеспечивающих решение проблемы запаздывания принятия решений;
- выполнена декомпозиция общей задачи на теоретические и технологические подзадачи;
- обоснован подход к решению на основе адаптации методологии проактивного мониторинга сложных техническим систем к проблематике гетерогенных КМС.
- определена категория КМС для реализации приложения.
- описаны функциональные требования к разрабатываемой КМС.

ГЛАВА 2. МЕТОДЫ, МОДЕЛИ И АЛГОРИТМЫ МОНИТОРИНГА КРУПНОМАСШТАБНЫХ СИСТЕМ

2.1 Основные понятия и определения

При моделировании сложных систем необходимо учитывать, что в основе модели должен лежать комплекс понятий, дающий начало действительно междисциплинарным исследованиям [1, с.22]. В высшей степени это утверждение относится к задаче мониторинга КМС, изначально носящий междисциплинарный характер.

Анализ работ по мониторингу Т.В.Аксенова, С.К.Андрюшкевича [2], В.Л.Венгеровича, С.П.Ковалева [19] и других позволяет сделать вывод, что предлагаемые определения носят частный узконаправленный характер, что затрудняет формирование понятийного каркаса для концептуального моделирования системы мониторинга КМС и противоречит требованию Акоффа. Для решения этой проблемы предлагается следующий комплекс определений, которые носят прагматический характер и ориентированы на практическое использование при разработке системы мониторинга КМС.

КМС – структура, включающая большое количество объектов различной природы, параметры которых измеряются датчиками или могут быть получены от пользователей, а оцениваются и управляются системой мониторинга КМС. Управление КМС осуществляет ЛПР КМС на основе решений, предлагаемых системой мониторинга КМС.

Объект – материальная или виртуальная сущность, параметры которой могут измеряться, оцениваться и изменяться в зависимости от решений системы мониторинга КМС. Управление объектом осуществляет ЛПР объекта.

Сцена мониторинга КМС – система наблюдения, включающая субъекта, объекты и коммуникации. Параметры объектов доступны измерению и могут изменяться решениями субъекта.

2.2 Классификация объектов наблюдения

Как показано в главе 1, модели объектов мониторинга носят узко специализированный характер, что затрудняет их классификацию, унификацию и, соответственно, исключает возможность построения интероперабельных программных модулей для снижения стоимости систем мониторинга КМС. Для устранения этого недостатка в рамках предметной области мониторинга выполним четыре классификации.

2.2.1 Классификация по признаку «видимости» объекта

Для решения проблемы классификации на макроуровне предлагается группировать объекты в зависимости от их «видимости» (нахождения в поле зрения) системы мониторинга.

Видимые объекты: объекты, сведения о которых попадают в сферу, наблюдаемую, программой мониторинга КМС, которую определим, как aiM.

Невидимые объекты: объекты, физически существующие, но не входящие в сферу видимости aiM.

Сфера видимости aiM: объекты, параметры которых фиксируются локальными сенсорами всех типов и передаются по каналам связи в aiM.

На рисунке 2.1 представлена схема, описывающая сферы видимости КМС.

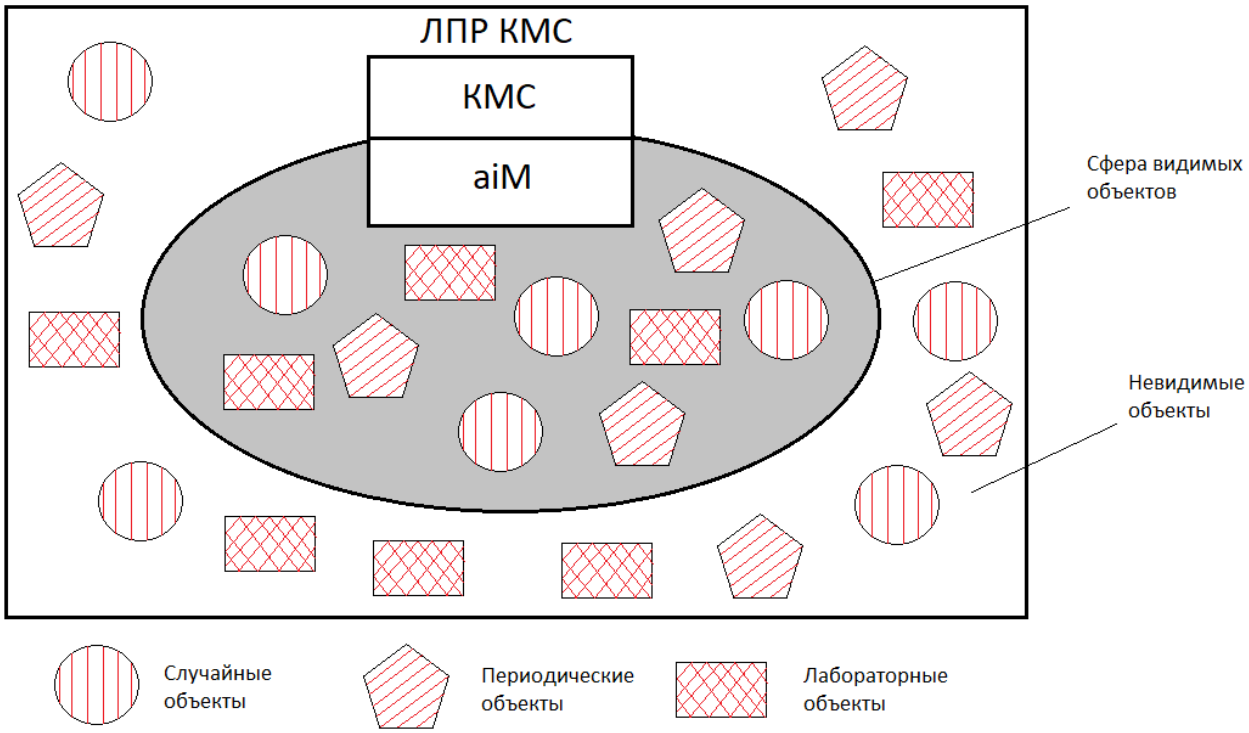

Рисунок 2.1 - Схема сферы видимости КМС

Объектом наблюдения может являться: завод, растение, человек, автомобиль, город, процесс и все другие сущности или параметры, которые можно измерить.

Объекты вне поля зрения для КМС не существуют, но могут войти в сферу видимости после установления коммуникаций, по которым в aiM будут поступать сигналы, характеризующие объект.

Классификация на видимые и невидимые полезна для ЛПР КМС, но недостаточна, так как не готовит о том, какую природу имеют наблюдения.

2.2.2 Классификация по признаку природы объекта

В результате анализа существующих в настоящее время сущностей можно выделить три типа объектов: случайные, периодические, лабораторные.

Случайные объекты – объекты внешней среды, данные о которых случайно попали в сферу видимости системы мониторинга. Например, это может быть случайная фотография объекта, присланная из любой точки Земли.

Периодические объекты – объектов внешней среды, данные о которых периодически поступают в систему мониторинга, анализируются и по ним принимаются соответствующие решения.

Лабораторные объекты – подсистема, состоящая из объектов локальной среды, данные о которых регулярно автоматически фиксируются датчиками в определенное время и отсылаются для анализа и принятия соответствующих решений в локальную систему мониторинга.

На основе этой классификации следует классифицировать среду.

2.2.3 Классификация по признаку типа среды

При рассмотрении практических задач можно определить, что в одних случаях объект может быть размещен рядом с aiM и подключен напрямую (локально), в других – находиться в любой точке Земли и подключаться через коммуникации Internet (внешне). Выполним соответствующую классификацию.

Локальная среда – видимые объекты, подключенные к aiM через локальную сеть. Отличается низким уровнем неопределенности и высоким уровнем надежности процесса мониторинга.

Внешняя среда – видимые объекты, подключенные к aiM через глобальную сеть. Отличается высоким уровнем неопределенности и низким уровнем надежности процесса мониторинга.

2.2.4 Классификация по признаку типа входных данных

Важнейшее значение для выбора метода обработки входных данных имеет тип входных данных. Для чисел (вектор) – методы численного анализа;

для строк (текст) – лингвистические методы; для графики (фото) – методы распознавания образов; для аудио (голос) – методы анализа гармоник; для видео (запись с камер) – распознавание динамических образов. Нужно отметить, что для обработки каждого типа данных используется специализированное ПО, а также каждый тип данных предполагает использование широкого спектра специализированных методов и представляет отдельную научную дисциплину.

Таким образом, выделим пять типов подсистем aiM:

- численный;
- строковый;
- графический;
- аудио;
- видео.

Предложенный комплекс классификаций охватывает все материальные и виртуальные сущности, что позволяет готовить о возможности построения на его основе концептуальных междисплинарных моделей, что отвечает требованию Акоффа и требованиям задачи данного исследования.

2.3 Концептуальные модели

2.3.1 Понятие концептуальной модели

Концептуальная модель – это модель, которая определяется набором понятий и связей между этими понятиями, без которых невозможно выделить смысловую структуру исследуемой области или объекта.

Для построения концептуальной модели необходимо четко поставить задачу, выбрать критерии оценки качества работы системы, определить переменные модели и зависимости между ними, выявить требования к модели, определить, какую информацию о модели нужно собирать и как организовать её хранение, выдвинуть гипотезы и принять предположения.

2.3.2 Модель КМС

Исходя из определений КМС состоит из объектов, которые случайно, периодически или постоянно находятся в сфере зрения системы мониторинга.

Соответственно, концептуальную модель КМС можно описать кортежем:

$$ULS = (P, aiM, O_{sl}, O_{pr}, O_{lb}) \quad (2.1)$$

где: P – лицо, принимающее решение (субъект); aiM – система мониторинга; O_{sl} – случайные объекты; O_{pr} – периодические объекты; O_{lb} – лабораторные объекты.

Модель ULS носит общий характер, для практического применения необходимо уточнить ее составляющие.

2.3.3 Модель aiM для мониторинга объектов КМС

Модель aiM отражает концепцию подхода, состав, функциональные и информационные составляющие:

$$aiM = (Master, f1(O_{sl}), f2(O_{pr}), f3(O_{lb})) \rightarrow (V_1/U_1(O_{sl}), V_2/U_2(O_{pr}), V_3/U_3(O_{lb})), DB, KB) \quad (2.2)$$

где: Master – программа управления; f1, f3, f3 – функции наблюдения (фиксации параметров объектов), оценки состояния и синтеза управляющего решения; V/U – состояния и соответствующие управляющие решения для объектов каждой группы; DB – база данных с поступающими от различных источников данными; KB – база знаний с достоверными историческими знаниями.

2.3.4 Модель случайного объекта

Модель случайного объекта инвариантна к ЛПР объекта, месту его нахождения и времени фиксации датчиками (например, фотоаппаратом) его параметров:

$$O_{sl} = (adr, id, dt, type, crd, P, t, X, V, U) \quad (2.3)$$

где: adr – адрес объекта; id – идентификатор; dt – устройство фиксации значений параметров; type – тип; crd – координаты месторасположения; P – лицо, фиксирующие параметры; t – время фиксации параметров; X – параметры; V – состояние; U – управляющее решение.

Средства фиксации параметров могут быть различными и не известны заранее, тип dt можно определить по типу X.

2.3.5 Модель периодического объекта

Модель периодического объекта жёстко привязана к ЛПР объекта, месту его нахождения и времени фиксации его параметров:

$$O_{pr} = (adr, id, dt, type, crd, P, \Delta, t, X, V, U) \quad (2.4)$$

где: Δ – период времени, через который проводится фиксация параметров объекта. Средства фиксации параметров известны.

2.3.6 Модель лабораторного объекта

Модель лабораторного объекта жёстко инвариантна к ЛПР объекта (так как обычно его заменяет ИИ), жестко привязана к месту его нахождения и периоду фиксации датчиками его параметров:

$$O_{lb} = (adr, id, dt, type, crd, P, \Delta, t, X, V, U) \quad (2.5)$$

где: dt – средство фиксации параметров, например, цифровой фотоаппарат.

Средства фиксации параметров dt известны и неизменны (фотоаппараты, влагомеры, термометры и другие).

2.4 Алгоритмы мониторинга КМС

2.4.1 Общая схема решения

Наиболее эффективные современные системы мониторинга и поддержки принятия решений используют в качестве основы принципы работы человеческого мозга. Используя этот подход, выделим:

- объект O, генерирующий сигналы X;
- датчики, фиксирующие значения сигналов X объекта dt;
- управляющую программу Master, которая получает на входе сигналы X и управляет выбором функций принятия решений f для их обработки;
- базу достоверных знаний (соответствует древней памяти мозга, KB);
- базу горячих данных, поступающих от сенсоров (соответствует молодой памяти мозга, DB);
- функции оценивания и принятия решений f1, f2, f3 для разных типов объектов.

Схема синтеза решений в рамках данного подхода показана на рисунке 2.2.

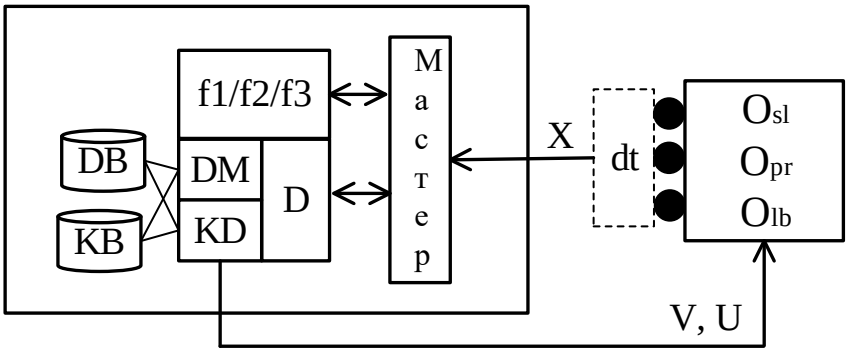

Рисунок 2.2 - Схема работы aiM КМС

Алгоритмы реализации данной схемы представлены ниже.

2.4.2 Алгоритм построения БЗ эксперта

Вход: формализованные знания эксперта.

Шаг 01. Выбор объекта исследования O.

Шаг 02. Анализ доступных экземпляров объекта в различных состояниях.

Шаг 03. Выбор диагностических параметров X.

Шаг 04. Классификация возможных состояний V.

Шаг 05. Формирование образа каждого состояния E.

Шаг 06. Формирование управляющих решений U для каждого состояния.

Шаг 07. Запись в базу знаний БЗ (исторические знания).

Выход: БЗ

Для реализации указанных шагов необходима разработка рабочего места эксперта, снабженного средствами автоматического математического и спектрального анализа параметров различного типа (графики, аудио, видео) для каждого экземпляра объекта.

2.4.3 Алгоритм работы управляющей программы

Вход: сообщение-запрос из среды (заказчика)

Шаг 01. Обращение к aiM по каналам от заказчика (ЛПР объекта).

Шаг 02. Считывание сообщения.

Шаг 03. Парсинг сообщения, выделение объекта наблюдения.

Шаг 04. Определение типа объекта.

Шаг 05. Поиск аналога в БЗ.

Шаг 06. Если аналог не найден, сообщение об отказе, завершение сеанса.

Шаг 07. Вызов в зависимости от типа объекта f1 или f2 или f3.

Шаг 08. Передача в f1 или f2 или f3 образа объекта.

Шаг 09. Анализ образа по релевантному алгоритму.

Шаг 10. Получение Vj, Uj от f1 или f2 или f3.

Шаг 11. Отправка Vj, Uj заказчику.

Шаг 12. Завершение сеанса.

Выход: сообщение-ответ заказчику (ЛПР объекта)

2.4.4 Унифицированный алгоритм мониторинга случайного объекта

Вход: Сообщение от модуля Master.

Шаг 01. Активизация в зависимости от типа объекта функции (f1/f2/f3).

Шаг 02. Поиск аналога в БЗ.

Шаг 03. Сравнение внешнего образа и образов-аналогов из БЗ.

Шаг 04. Распознавание наиболее похожего аналога.

Шаг 05. Выбор реквизитов Vj, Uj наиболее похожего аналога.

Шаг 06. Передача Vj, Uj модулю Мастер.

Выход: сообщение Vj, Uj модулю Master.

2.5 Выводы

Во второй главе получены следующие основные результаты:

- сформирован понятийный каркас, отвечающий требованиям актуальных задач мониторинга и современному уровню развития компьютерных технологий;
- выполнена классификация объектов наблюдения, включая случайные, периодические и стационарные;
- построены концептуальные модели КМС, специализированного ИИ, рабочего места эксперта (создание БЗ эксперта), а также случайного, периодического и стационарного объектов наблюдения;
- разработан алгоритм общения управляющей программы со средой для обеспечения жизненного цикла системы мониторинга КМС, включая фиксацию изменения параметров среды и синтез соответствующих решений;
- разработан алгоритм распознавания состояния и синтеза управляющих решений для случайных, периодических и стационарных объектов.

ГЛАВА 3. ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

3.1 Исследование программного инструментария

Современные приложения используют так называемую микросервисную архитектуру. Такой вид архитектуры представляет собой набор сервисов, которые почти не зависят друг от друга, каждый из которых может быть легко изменен без изменений в других сервисах. При этом каждый отдельный модуль может представлять собой сложную систему, которая обращается к другим системам посредствам различных стандартных протоколов, например: SOAP, XML-RPC, REST.

Для разработки КМС будет использован высокоуровневый язык программирования Java, а для реализации микросервисной архитектуры фреймворки Spring и Spring Boot.

Spring Framework – универсальный фреймворк, используется для разработки приложений на языке Java. Он позволяет решать различные задачи, и его можно рассматривать как коллекцию меньших фреймворков, каждый из которых предназначен для решения определенной задачи. Основные из них:

- Фреймворк Spring Core

  Представляет собой ядро платформы, которое обеспечивает разработчика необходимыми средствами для создания приложений. Занимается управлением компонентами, соединением с базой данных, отвечает за управление транзакциями и выполняет множество других функций. Можно сказать, что он неявно используется другими компонентами Spring Framework.
- Фреймворк MVC

  Обеспечивает поддержку архитектуры Model-View-Controller. Model инкапсулирует данные приложения. View отображает данные. Controller принимает запросы, создает необходимую модель и передает вид для отображения.
- Inversion of Control контейнер

  Управляет жизненным циклом объектов.
- Spring Security - фреймворк для авторизации и аутентификации

  Позволяет производить авторизацию и аутентификацию, использует такие протоколы, как OAuth, LDAP и другие.
- Spring Data - фреймворк для доступа к данным

  Обеспечивает доступ к данным в базе данных. Работает с различными реляционными и нереляционными базами данных.

База данных – это сохраненная в определенном порядке информация. Для управления базой данных необходима система управления базами данных, в частности для разработки системы мониторинга КМС была выбрана PostgreSQL.

Достоинства PostgreSQL:

- Кроссплатформенность;
- Гибкость;
- Надежность и целостность данных;
- Наличие API для различных языков программирования;
- Существование различных библиотек для разных платформ для работы с данной СУБД;
- Постоянное совершенствование функционала;
- Возможность создания сложных пользовательских процедур;
- Простая интеграция с другими системами управления базами данных

Недостатки PostgreSQL:

- Могут возникнуть трудности с первоначальной настройкой;
- При неправильной настройке, данная СУБД проигрывает своим конкурентам в быстродействии

Для реализации клиентской части приложения использовался фреймворк Angular. Этот фреймворк отличается гибкостью, надежностью и обладает высокой масштабируемостью. Также положительными фактором является поддержка Google, огромное количество библиотек и готовых инструментов, а также обилие справочной информации.

3.2 Проектирование архитектуры приложения

В приложении будет использована микросервисная архитектура. Это означает, что будут выделены несколько независимых модулей, каждый из которых представляет собой отдельный сервис.

Модули будут разделены на две части:

- Модули, взаимодействующие с базой данных (Inner Modules)
- Модули, взаимодействующие с клиентской частью приложения (External Modules)

Между собой взаимодействие сервисов будет осуществляться посредствам REST запросов. Модули, взаимодействующие с клиентской частью приложения, будут защищены авторизацией, возможности каждого отдельного пользователя будут определяться его ролью.

На рисунке 3.1 представлена компонентная схема приложения:

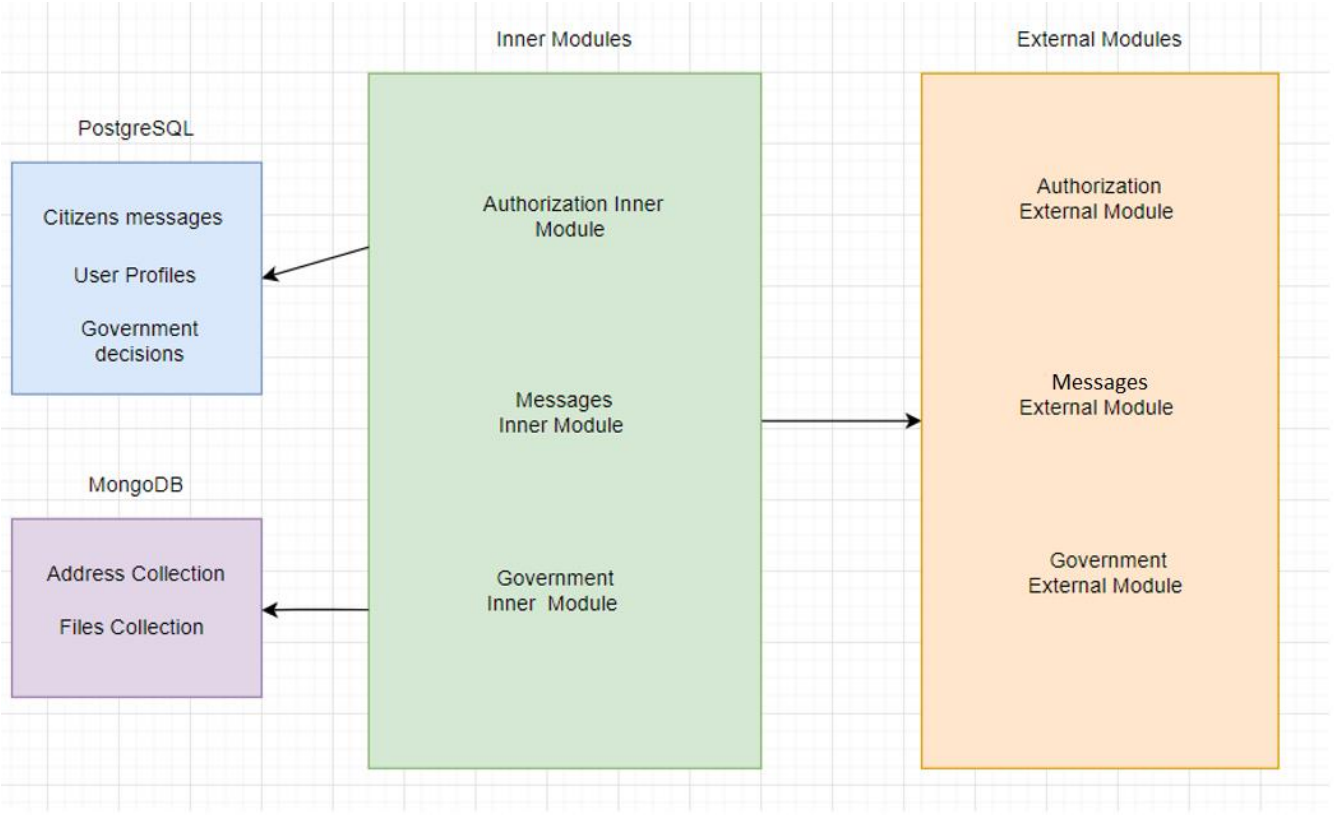

Рисунок 3.1 - Компонентная схема приложения

Такой подход к проектированию архитектуры приложения является наиболее безопасным, так как у пользователей есть доступ только ко внешним модулям. Внутренние модули, имеющие связь с базой данных, взаимодействуют только со внешними модулями, но не с пользователями.

3.3 Проектирование архитектуры базы данных

При проектировании архитектуры базы данных были выделены основные сущности, которые будут присутствовать в системе и атрибуты этих сущностей, а также определены взаимосвязи между ними.

На рисунке 3.2 представлены несколько основных таблиц, которые определяют сущности пользователя (User), сообщения пользователя о проблеме (Problem), а также организацию-исполнителя (Executor), которая назначается для решения конкретной проблемы.

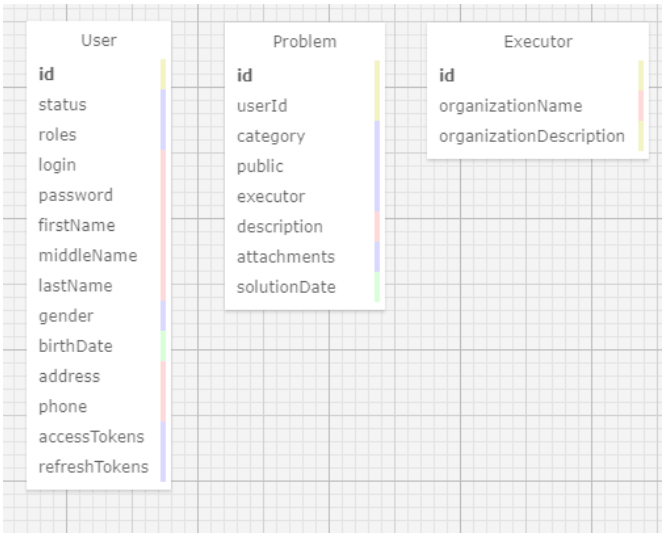

Рисунок 3.2 - Основные таблицы базы данных с их атрибутами. Пользователь (User), Сообщение о проблеме (Problem), исполнитель (Executor)

Сущности пользователя соответствует определенная роль, которая определяет его права доступа, например: администратор системы, простой житель, представитель местных органов власти или ответственное лицо от организации-исполнителя. Пользователь при регистрации указывает логин и пароль, а также информацию о своей личности. Пароли пользователей хранятся в зашифрованном виде. Для того, чтобы система могла функционировать в реальных условиях на этапе регистрации необходимо идентифицировать пользователя как определенного гражданина, так как его сообщения о проблемах являются официальными заявками к городским органам власти. Вопрос идентификации может быть решен путем предоставления паспортных данных, а также указания своего настоящего номера мобильного телефона, который у сетевых операторов обязательно закреплен за конкретным человеком. Это не единственный вариант идентификации, в реальных условиях такого рода требования должны быть согласованы на законодательном уровне.

Чтобы добавить к базе данных новые таблицы, необходимо написать скрипт миграции и выполнить его. Для этого будет использован инструмент Flyway. Он позволяет с помощью языка SQL работать с миграциями. Для этого необходимо создать в каталоге /src/main/resources подкаталог db/migration и разместить в нем все необходимые скрипты. Обычно скрипты именуют с указанием следующих атрибутов:

- Версия миграции
- Дата применения миграции к базе данных
- Краткое описание компонентов, затронутых миграцией

Такой подход к именованию позволяет отслеживать, какие изменения происходили с базой данных, даже не открывая отдельно взятые скрипты.

Еще одной особенностью Flyway является удобная интеграция с различными фреймворками для автоматизации сборки проекта, такими как Maven или Ant. В то же время Flyway можно использовать просто из командной строки. Чтобы применить миграции в созданном подкаталоге db/migration, нужно всего лишь выполнить команду flyway migrate.

Вот так может выглядеть SQL скрипт для добавления в базу данных таблицы Problem, которое представляет из себя сообщение о проблеме:

```
create table if not exists problem_entity
(
   id bigserial
      constraint problem_entity_pkey
         primary key,
   public boolean not null,
   category varchar(255),
   description varchar(255),
   attachments varchar(255),
   solution_date timestamp,
   user_id bigint
         references user_entity,
   executor_id bigint,
         references executor_entity
);
```

3.4 Выводы

В третьей главе получены следующие основные результаты:

- исследованы инструменты разработки современных приложений, проанализированы их достоинства и недостатки;
- для реализации приложения выбрана микросервисная архитектура, которая позволяет добиться почти полной независимости модулей, которые могут общаться между собой посредствам определенных протоколов связи. Каждый отдельный модуль в то же время является отдельной системой со своим набором сущностей и сервисов;
- выбраны наиболее надежные инструменты для разработки системы; мониторинга КМС и поддержки принятия решений;
- выполнено проектирование схемы базы данных.

ГЛАВА 4. РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

4.1 Реализация серверной части приложения

Для того, чтобы реализовать отдельный модуль с использованием Spring Boot, необходимо подключить соответствующие зависимости в pom файл модуля. Это зависимости от следующих библиотек: spring-boot-starter-log4j2, spring-boot-starter-data-jpa, postgresql, spring-boot-starter-test. Они необходимы для записи в журнал системных сообщений, по которым можно отследить возникающие ошибки в программе, корректно подключаться к базе данных и выполнять различные запросы к ней, а также для выполнения автоматического тестирования программы перед её запуском. Разумеется, можно подключать и другие библиотеки, в зависимости от того, какой функционал необходим в отдельном разрабатываемом модуле в рамках микросервисной архитектуры приложения. Для каждой библиотеки указываются параметры groupId и artifactId, опционально можно указать параметр version, если нужна определенная версия некоторой библиотеки. Это позволяет фреймворку Apache Maven автоматически выполнять сборку проекта и однозначно определять необходимые библиотеки и их версии среди огромного их количества в репозитории.

Вот пример того, как можно подключить библиотеки в pom-файле:

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-log4j2</artifactId>
</dependency>
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
   <groupId>org.postgresql</groupId>
   <artifactId>postgresql</artifactId>
   <version>42.2.5</version>
</dependency>
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-test</artifactId>
<dependency>
```

После подключения библиотек к модулю необходимо создать главный класс, который будет выполнять запуск этого модуля. Фреймворк SpringBoot позволяет это сделать всего лишь добавлением одной аннотации @SpringBootApplication перед объявлением главного класса. В этой аннотации можно указать параметр scanBasePackages. Он определяет, в каких пакетах необходимо выполнить сканирование компонентов.

Ниже представлен пример того, как может выглядеть главный класс SpringBoot приложения:

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication(scanBasePackages = "by.bsu")
public class ProblemInnerApplication {
   public static void main(String[] args) {
      SpringApplication.run(ProblemInnerApplication.class, args);
   }
}
```

Во время разработки очень важно правильно создать структура пакетов, это необходимо для обеспечения логического разделения классов. Использование пакетов исключает в том числе и проблему конфликта наименований классов. Для системы мониторинга КМС были выделены следующие основные пакеты:

- apicontroller

  В данном пакете размещаются классы, которые позволяют принимать REST запросы от клиентской части приложения.
- dbcontroller

  Пакет используется для классов, в которых задана конфигурация базы данных, описаны все доступные запросы определенным сервисам запросы к базе данных, а также в этом пакете расположены все миграции.
- dto

  Dto расшифровывается как Data Transfer Object, классы указанного типа используются для конвертации сущностей таким образом, чтобы их можно было дальше использовать в REST запросах.
- endpoints

Пакет endpoints содержит перечисление всех определенных в модуле точек доступа, на которые клиентские приложения могут присылать запросы.

- entities

  Данный пакет содержит классы-сущности, которые используются во время запросов к базе данных.

- exceptions

  В этом пакете хранятся классы, представляющие собой исключения, которые могут быть вызваны во время выполнения программы в ситуации различных ошибок.

- rest

  В данном пакете расположены классы, описывающие запросы и ответы REST контроллера.

- services

  В классах пакета services присутствует реализация тех функций, которые классы из apicontroller вызывают, когда получают очередной запрос от клиентского приложения.

- utils

  Пакет utils содержит классы, которые могут быть полезны в приложении и выполняют некоторые специфические функции. Например, это классы для работы с временными зонами и т.д.

На рисунке 4.1 можно увидеть, как выглядит иерархия пакетов у одного из разработанных модулей. Похожая структура использовалась и во всех остальных модулях, небольшие отклонения от неё допускались только когда в модуле был нужен какой-то специфический функционал.

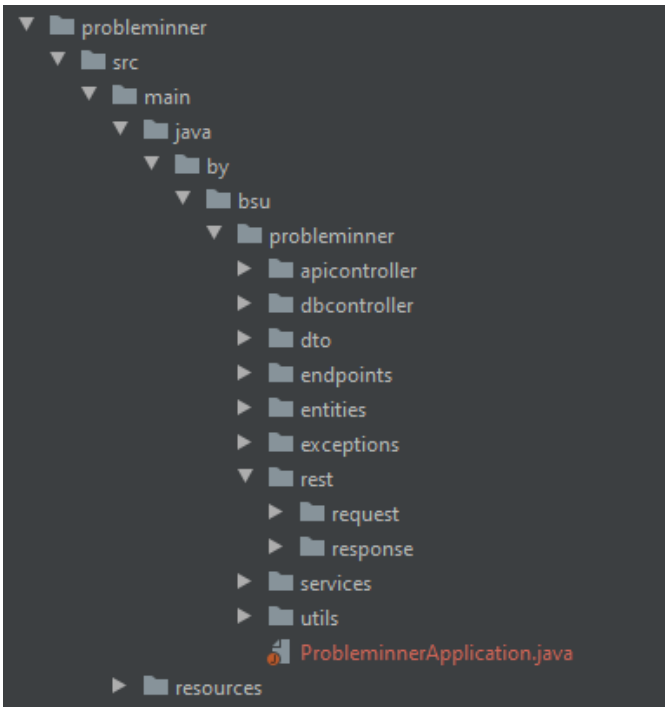

Рисунок 4.1 - Структура пакетов

Ниже на примере реализации функции получения списка проблем жителей можно увидеть, как классы из разных пакетов взаимодействуют между собой и позволяют выполнять REST запросы:

```
@RestController
@Api
public class ProblemInnerController {

   private final ProblemInnerService problemInnerService;

   @Autowired
   public ProblemInnerController (ProblemInnerService problemInnerService) {
      this. problemInnerService = problemInnerService;
   }

   @ApiOperation(value = "Получение списка проблем жителей.")
   @PostMapping(value = ProblemInnerEndpoints.LIST_PROBLEMS, produces
= MediaType.APPLICATION_JSON_VALUE)
  public RestResponse<ProblemInnerListResponse>> getProblemList
(@RequestBody ProblemInnerProblemListRequest request)
throws RestApiException {
      return new RestResponse<>( problemInnerService.getProblemList(request));
   }
}
```

4.2 Реализация клиентской части приложения

На рисунке 4.2 представлена страница со списком сообщений жителей о возникших проблемах. Аналогично выглядит страница с показаниями датчиков, на ней присутствуют только сообщения от датчиков, которые наблюдают некоторое отклонение от нормы. Пороговое значение состояний отклонений определяется отдельно для каждого датчика экспертом.

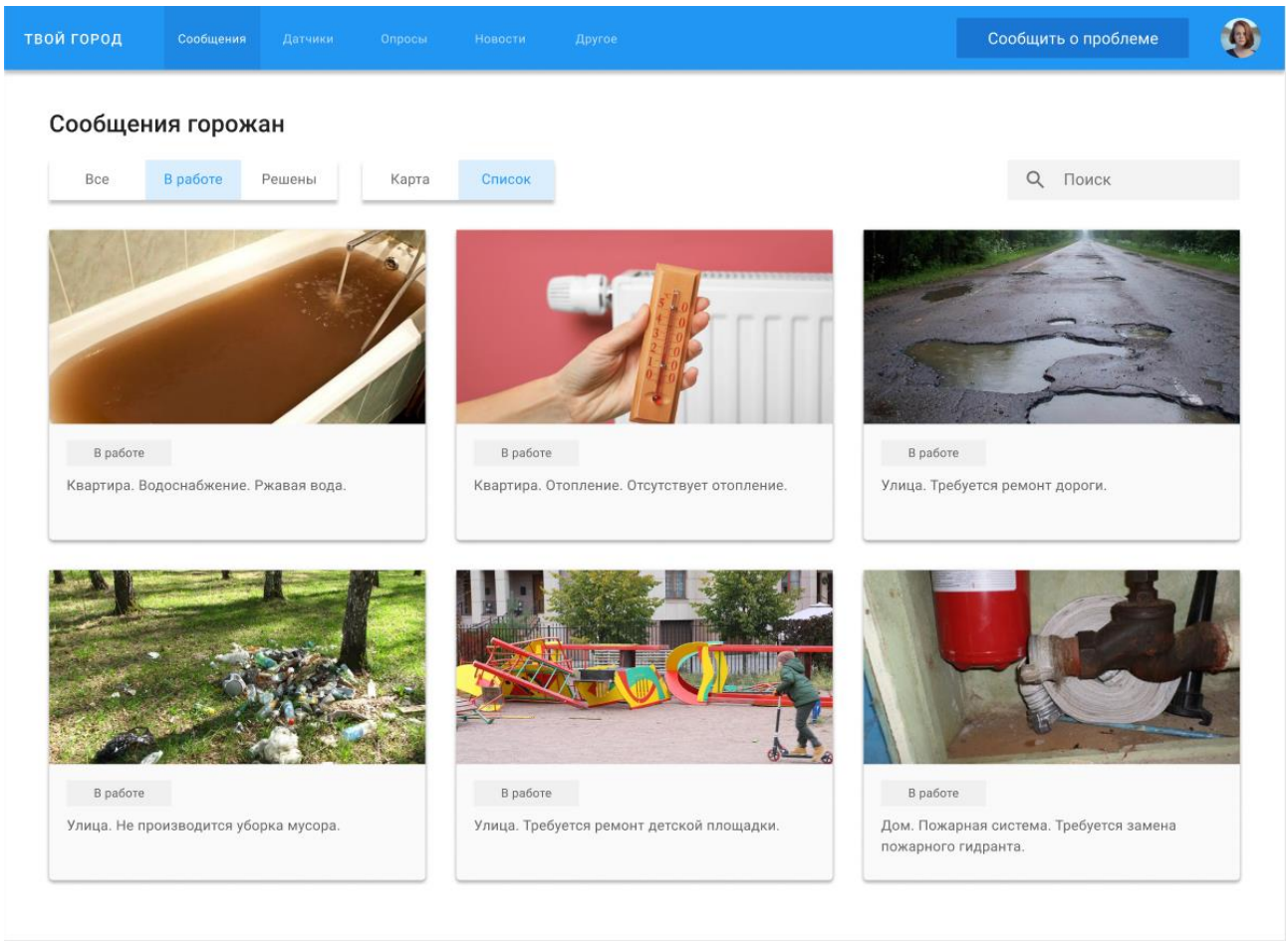

Рисунок 4.2 - Страница с собранными сообщениями о проблемах, отображение проблем в виде отдельных карточек

На ней можно выполнить фильтрацию проблем в зависимости от того, находятся они в работе или уже решены. Также пользователям доступен функционал поиска проблемы по полному или частичному совпадению описания проблемы.

Все сообщения о проблемах обязательно соответствуют конкретному адресу на карте города. Поэтому для большей наглядности можно просмотреть их в виде карты. Такой подход позволяет оценить картину в целом, выделить наиболее проблемные районы города, а также использовать информацию о местоположении для выбора подходящих организаций-исполнителей, закрепленных за каким-либо конкретным районом города.

На рисунке 4.3 представлена страница с картой, на которой красными метками отображаются все места, в которых возникли проблемы. В качестве карты в приложение посредствам API были интегрирован сервис Google Maps.

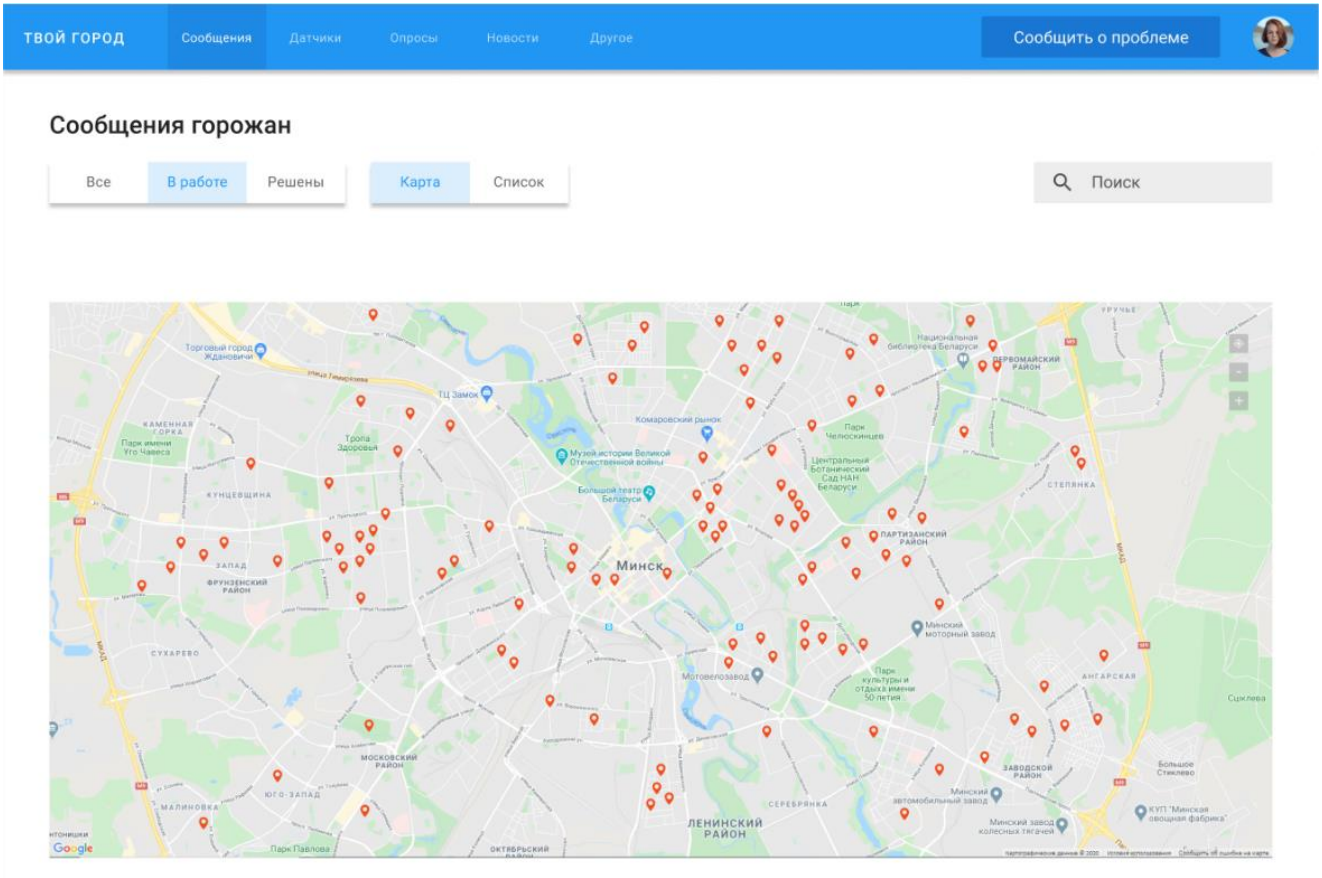

Рисунок 4.3 - Страница с собранными сообщениями о проблемах, отображение проблем в виде карты

У пользователей, которые выполняют авторизацию в системе с правами жителя, есть возможность нажать на кнопку «Сообщить о проблеме», после чего пользователь перенаправляется на страницу, где ему предлагается указать категорию проблемы. Список категорий проблем заранее определен и может в случае необходимости быть дополнен или изменен администратором системы. Такой подход применяется, чтобы закрепить за отдельными видами проблем определенный список доступных организаций-исполнителей. Например, категориями могут являться следующие: водоснабжение, отопление, обеспечение электроэнергией, уборка общественных территорий, качество воздуха и многие другие. Сообщения о проблемах, предоставленные датчиками, создаются автоматически, когда показатели измерений датчика достигают предварительно заданных пороговых значений. Они, как и сообщения жителей, содержат адрес, потому что каждый датчик имеет строго зафиксированное положение, а также описание проблемы, в котором присутствует информация с датчика.

На рисунке 4.4 представлена страница создания нового сообщения о проблеме, выбор категории проблемы. Пользователь выбирает категорию и

подкатегорию проблемы, указывает адрес, описание проблемы и прикрепляет фотографию.

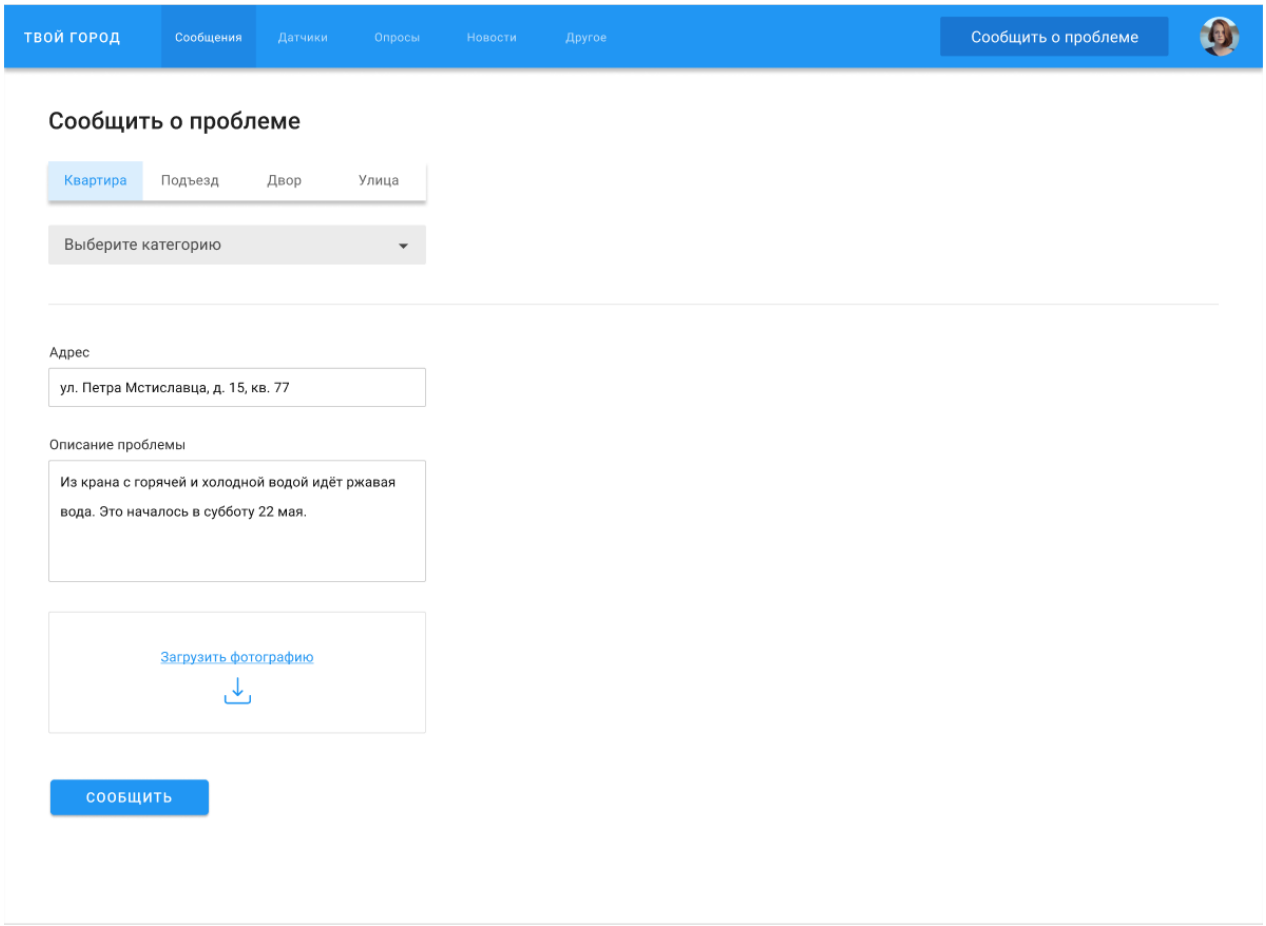

Рисунок 4.4 - Страница создания пользователем с правами жителя нового сообщения о проблеме

У всех горожан есть возможность просмотреть детали любой проблемы в их городе, увидеть ход её решения. А у пользователя, который является автором данного сообщения, также есть возможность оценить удовлетворенность решением органами власти этой проблемы.

На рисунке 4.5 представлена страница процесса решения проблемы. На ней присутствует фотография и другие детали описания проблемы. Так как проблема является решенной, то пользователю предлагается оценить удовлетворенность решением проблемы. В правой части экрана представлена временная линия, на которой отображаются основные этапы решения проблемы, а именно:

- Когда было получено сообщено о проблеме.
- Когда был назначен исполнитель для решения проблемы. Очевидно, что это может происходить автоматически, если администратор системы мониторинга задал организацию-исполнителя, закрепленную

за конкретным районом города для решения определенных категорий проблем. Также задать организацию-исполнителя можно вручную, если авторизоваться в системе с правами администратора.

- Когда исполнитель взял данную проблему в работу.
- Когда исполнитель окончил работу по данной проблеме.
- Когда пользователем-автором сообщения о проблеме была поставлена оценка качеству и скорости решения его проблемы.

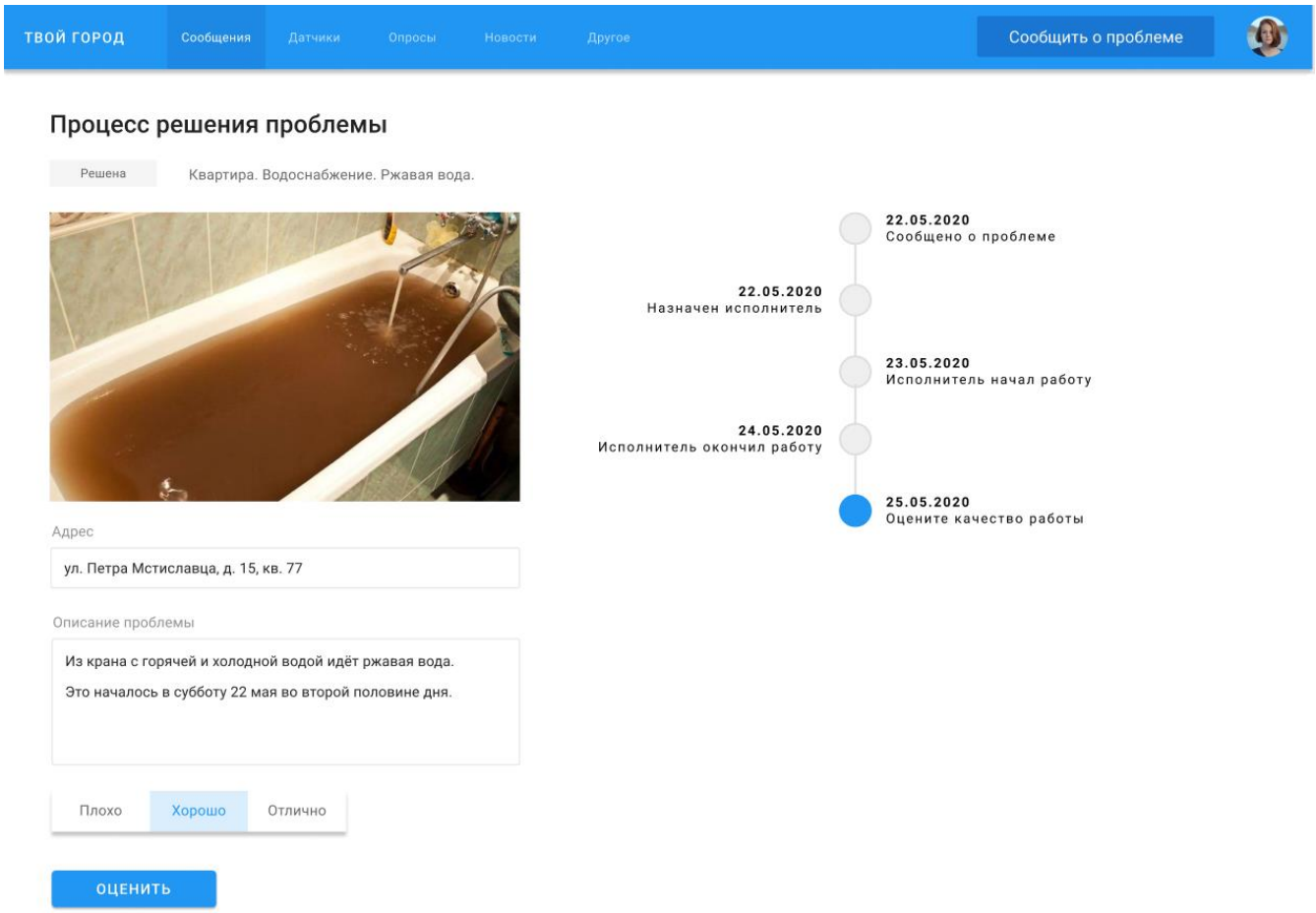

Рисунок 4.5 - Страница процесса решения проблемы

На рисунке 4.6 представлена страница, доступная представителям органов власти, на которой отображается ситуация в городе. Можно выбрать категорию проблем, по которым будет произведена фильтрация. Если в некотором районе города количество сообщений о проблемах в рамках выбранной категории превосходят заданные пороговые значения, то эта область будет выделена на карте красным цветом, а в предлагаемых действиях будет отображено предлагаемое системой поддержки принятия решений действие. Эти действия предварительно должны внести в систему эксперты в отдельно взятых областях. Они определяют возможные состояния системы и действия, которые в этом случае должны быть предприняты. Поэтому во

время аварийных ситуаций ответственное лицо не будет тратить время на поиск решения, а сможет сразу же приступить к действиям. Например, связаться по заранее заданным телефонам с ответственными за этот участок лицами, передать им должностные инструкции, обозначить необходимые меры и сроки.

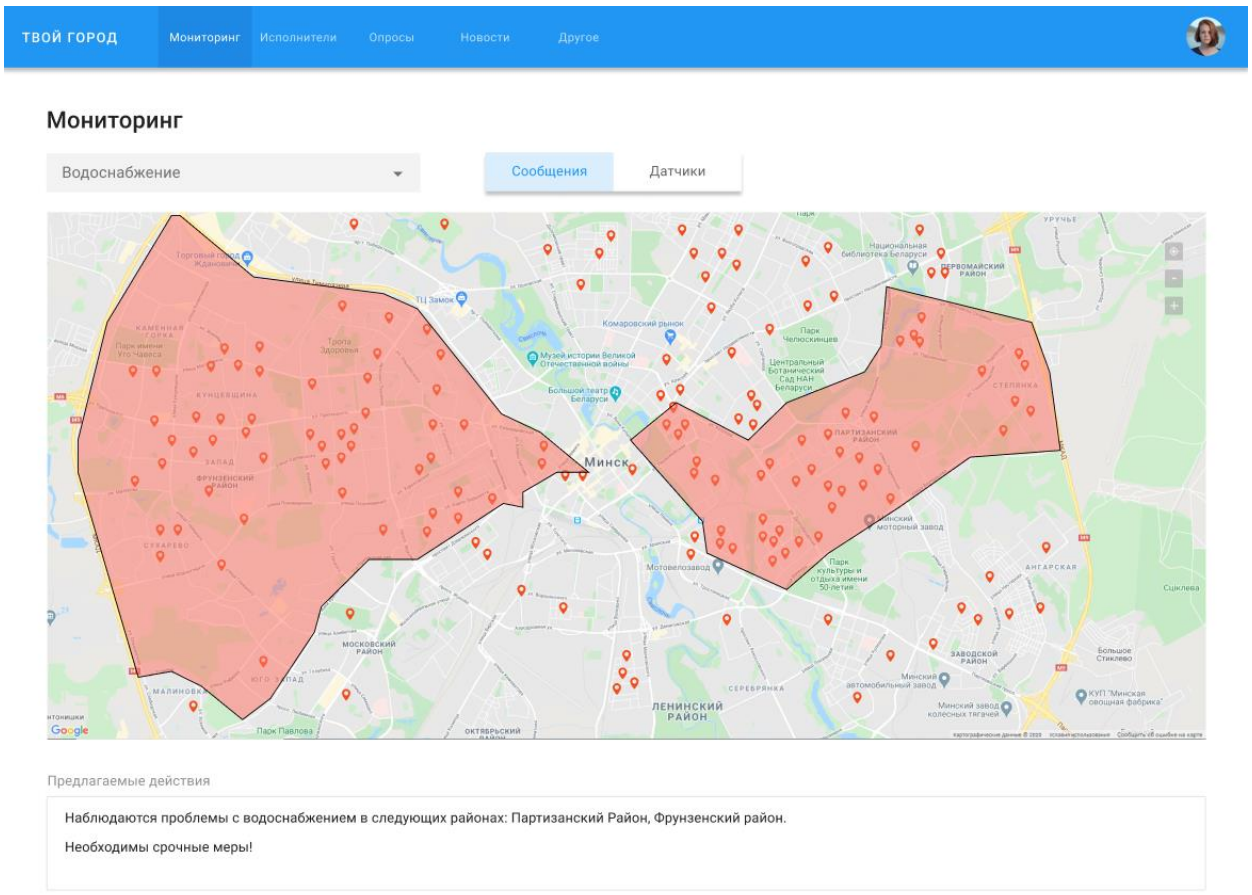

Рисунок 4.6 - Страница результатами мониторинга и выделением аварийных участков

Таким образом, разработанное приложение позволяет производить непрерывный мониторинг крупномасштабной гетерогенной системы, а именно отдельного города. Полученные результаты могут быть использованы в городах, независимо от их размера и количества населения. В данный момент рассматриваются возможности внедрения представленной системы.

4.3 Выводы

В четвертой главе получены следующие основные результаты:

- разработаны серверная и клиентская части приложения, представляющего собой реализацию системы мониторинга КМС. Для разработки были использованы современные программные инструменты и подходы к построению архитектуры приложения;
- выполнено тестирование разработанного программного кода. Для тестирования использовалась система Postman, которая предоставляет удобный интерфейс для отправки различных веб-запросов, а также удобно отображает ответы сервера;
- разработанное программное обеспечение носит открытый характер и может быть расширено новыми интероперабельными модулями. Например, модулем для голосований и опросов, новостным модулем и др.;
- в настоящее время автор ведет поиск компании для внедрения разработанного программного обеспечения в проекты типа «умный город» «цифровое государство». Применение такой системы позволит сократить риски при возникновении нештатных или аварийных ситуаций.

ЗАКЛЮЧЕНИЕ

В процессе написания диссертации был выполнен анализ литературы по тематике крупномасштабных гетерогенных систем, выявлена актуальность проблемы мониторинга их компонентов, построены концептуальные модели КМС, специализированного ИИ, рабочего места эксперта, а также случайного, периодического и стационарного объектов наблюдения. Также был разработан алгоритм мониторинга и оценивания состояния объектов наблюдения на основе зафиксированных изменений их параметров и синтез соответствующих управляющих решений для административных органов.

На базе разработанных моделей и алгоритмов в рамках микросервисного подхода была реализована система мониторинга КМС для отслеживания ситуации и принятия решений для устранения аварийных ситуаций на уровне отдельного города. Для программной реализации были изучены инструменты разработки современных приложений с микросервисной архитектурой и выбраны наиболее надежные из них.

Разработанная система мониторинга КМС может быть полезна городским органам власти, так как они значительно с её помощью будет возможно более оперативное реагирование в условиях постоянно изменяющейся среды и принятие правильных решений при возникновении аварийных ситуаций. В то же время разработанная система позволит обычным горожанам улучшать качество своей жизни и объективно оценивать работу чиновников и государственных организаций.

Дальнейшее развитие разработанной системы мониторинга и поддержки принятия решений возможно и планируется в направлении построения умного города:

- поиск возможности внедрения системы сначала в небольших городах Республики Беларусь с целю выявления недостатков при массовой эксплуатации в реальных условиях;
- расширение функционала системы, путем добавления новых модулей (в частности, модуля голосования и опросов, ленты городских новостей и т.д.);
- разработка мобильных приложений под Android и iOS;
- интеграция с ранее разработанной автором данной работы системой мониторинга антропогенной нагрузки, которая была опубликована в журнале «Журнал Белорусского государственного университета. Математика. Информатика» №3 от 2017 года.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Виссия, Х. Принятие решений в информационном обществе / Х.Виссия, В.В. Краснопрошин, А.Н. Вальвачев – СПб: ЛАНЬ, 2019. – 227 с.
2. Андрюшкевич, С.К. Интеллектуальный мониторинг распределенных технологических объектов с использованием моделей состояния / С.К.Андрюшкевич, С.П.Ковалев // Управление, вычислительная техника и информатика. Вестник ТПУ. – 2010. – № 5. – С.35-39.
3. Батищева, О.М. Анализ методов формирования баз эталонов в системах оперативного контроля состояния технологического оборудования / О.М.Батищева, Б.Л.Штриков // Инновации в условиях развития информационно-коммуника-ционных технологий: Мат.научно-практ. конференции. – М.: МИЭМ, 2006. – С. 225-227.
4. Берг, А.И. Управление, информация, интеллект / А.И. Берг. – М.: Мысль, 1976. – 384 с.
5. Бродский, Ю.И. Распределенное имитационное моделирование сложных систем / Ю.И. Бродский. – М.: ВЦ РАН, 2010. – 156 с.
6. Бурков, В.Н. Большие системы: моделирование организационных механизмов / В.Н.Бурков, Б.Данев Б., А.К.Еналеев. – М.: Наука, 1989. – 246 с.
7. Бусленко Н.П. Моделирование сложных систем /Н.П. Бусленко. – М.: "Наука", 1978. – 400 с.
8. Вапник, В.II. Теория распознавания образов / В.П. Вапник, А.Я. Червоненскис. – М.: Наука, 1974. – 415 с.
9. Виртуальные гетерогенные коллективы, поддерживающие принятие решений / И. А. Кириков, А. В. Колесников, С. В. Листопад, С. Б. Румовская // Системы и средства информатики. – 2015. – № 3. – С. 126–149.
10. Воронин, А.А. Оптимальные иерархические структуры /А.А. Воронин, С.П. Мишин. – М.: ИПУ РАН, 2003. – 210 с.
11. Гэлбрейт, Д. Новое индустриальное общество / Д.Гэлбрейт. – М.: Экономика, 2004. – 602 с.
12. Ивахенко, А.Г. Моделирование сложных систем по экспериментальным данным / А.Г.Ивахенко, Ю.П. Юрачковский. – М.: Радио и связь. 1987. – 119 с.
13. Израэль, Ю.А. Экология и контроль состояния природной среды / Ю.А. Израэль. – Л.: Гидрометеоиздат, 1979. – 376 с.
14. Калашников, В.В. Теория сложных систем и методы их моделирования / В.В.Калашников. – М.: ВНИИСИ, 1982. – 120с.
15. Калужский, М.Л. Общая теория систем / М.Л.Калужский. – М.: Директ-Медиа. – 177 с.

16. Касти, Д. Большие системы. Связность, сложность и катастрофы /Д.Касти. – М: Мир, 1982. – 216 с.

17. Кириллов, Н.П. Концептуальные модели технических систем с управляемыми состояниями / Н.П.Кириллов // Искусственный интеллект и принятие решений. – М.: РАСН, 2011. – № 4. – С. 81-91.

18. Клир, Д. Системология / Д.Клир. М.: радио и связь, 1990. – 539 с.

19. Ковалев, М.М. Цифровая экономика / М.М.Ковалев, Г.Г.Головенчик – Минск, БГУ, 2018. – 327 с.

20. Контроль функционирования больших систем. Сборник трудов. – М.: Машиностроение, 1977. – 356 с.

21. Краснопрошин, В.В. Проблема принятия решений по прецедентности: разрешимость и выбор алгоритмов / В.В. Краснопрошин, В.А. Образцов // Выбраныя навуковыя працы Беларускага Дзяржаунага універсітэта. – Том 6: Матэматыка. – 2001. – С. 285–312.

22. Краснопрошин, В.В. Система мониторинга антропогенной нагрузки на природно-территориальные комплексы / В.В.Краснопрошин, А.Г.Жидков, А.Н.Вальвачев // Журнал БГУ. Математика. Информатика. – 2017. – № 3. – С. 100-105.

23. Крупномасштабные системы. Сборник нучн. трудов ИПУ. – М.: ИПУ, 1999. – 103с.

24. Кузьмич, А.И. Методы построения и анализа дескриптора для систем мониторинга мобильных гетерогенных объектов /А.И.Кузьмич, В.В.Краснопрошин, А.Н.Вальвачев // Доклады БГУИР. – Минск, 2014. – № 7. – С. 61-66.

25. Ларичев, О.И. Теория и методы принятия решений / О.И.Ларичев. – М.: Логос, 2002. – 327 с.

26. Майданович, О.В. Перспективные направления развития информационных технологий мониторинга и управления состояниями сложных технических объектов в реальном масштабе времени / О.В.Майданович, М.Ю.Охтилев, Б.В.Соколов // Information Models and Analyses. – 2012. – №1. – С. 318-335.

27. Малышев, М.Л. Мониторинг социально-трудовой сферы / М.Л.Малышев. – Перспектива, 2007. – 280 с.

28. Меерович, Г.А. Эффект больших систем. / Г.А.Меерович. – М.: Знание, 1985. – 192.

29. Месарович, М. Общая теория систем / М.Месарович, И.Такахара. – М.: Мир, 1978. – 312 с.

30. Месарович, М. Теория иерархических многоуровневых систем. / М.Месарович, Д.Мако, И.Такахара. – М.: Мир, 1973. – 344 с.

31. Негойцэ К. Применение теории систем к проблемам управления. / К.Негойце. – М.: Мир, 1981. – 180 с.

32. Новиков, Д.А. Теория управления организационными системами / Д.А.Новиков. – М.: МПСИ, 2005. – 584 с.

33. Охтилев, М.Ю. Теоретические и технологические основы концепции проактивного мониторинга и управления сложными объектами / М.Ю. Охтилев, Б.В. Соколов, Р.М. Юсупов // Известия ЮФУ. – 2015. – № 1. – С. 162-174.

34. Павлов, В.И. Алгоритмы контроля и управления состоянием сложной технической системы / В.И.Павлов // Антенны. – 2010. – № 11. – С. 65-67.

35. Петровский, А.Б. Теория принятия решений / А.Б.Петровский. – М.: ИЦ Академия, 2009. – 400 с.

36. Пригожин, И. Конец определенности / И. Пригожин. – Ижевск: РХД, 2001. – 208 с.

37. Рязанов В.В. О построении оптимальных алгоритмов распознавания и таксономии (классификации) при решении прикладных задач / В.В. Рязанов // Распознавание, классификация, прогноз. Математические методы и их применение. Ежегодник. – М.: Наука, 1988. – Вып.1. – С. 229–279.

38. Саймон, Г. Науки об искусственном / Г.Саймон. – М.: Едиториал, 2004. – 144 с.

39. Системные исследования. Сборник трудов. – М.: Наука,1969. – 203 с.

40. Соловьёв, И.В. Основы управления сложной организационно-технической системой. / И.В.Соловьёв, А.Н.Тихонов, А.Д.Иванников, В.Я.Цветков. – М.: МАКС Пресс, 2010. – 208 с.

41. Тарасов, В.Б. От многоагентных систем к интеллектуальным организациям: философия, психология, информатика / В.Б. Тарасов. – М.: Едиториал, 2002. – 352 с.

42. Трухаев, Р.И. Модели принятия решений в условиях неопределенности / Р.И.Трухаев. – М.: Наука, 1981. – 258 с.

43. Управление развитием крупномасштабных систем (MLSD’2018): материалы Одиннадцатой межд. конф., 1 - 3 окт. 2018 г., Москва: в 2-х т. – М.: ИПУ РАН, 2018. – 498 с.

44. Фу, К. Структурные методы и распознавание образов / К. Фу. – М.: Мир, 1977. – 319 с.

45. Шеннон, Р. Имитационное моделирование систем – искусство и наука / Р.Шеннон. – М.: Мир, 1978. – 418 с.

46. Alur, R. Principles of Cyber-Physical Systems / R.Alur. – Cambridge: The MIT Press, 2015. – 464 p.

47. Greengard, S. The Internet of Things / S.Greengard. – Cambridge: The MIT Press, 2015. – 232 p.

48. Lee, E.A. The Past, Present and Future of Cyber-Physical Systems: A Focus on Models / E.A.Lee // Sensors. – 2015. – № 15. – PP.4837-4869.

49. McCrady, S. Designing SCADA Application Software: A Practical Approach / S.McCrady. – Elsevier, 2013. – 246 p.

50. Murty, M. Pattern Recognition: An Algorithmic Approach / M.Murty. – Springer, 2011. – 275 p.

51. Pasco, A. Heterogeneous objects modelling and applications / A.Pasco, V.Adziev, P.Comninos. – Springer, 2008. – 285 p.

52. Power, D. J. Decision Support Systems: Concepts and Resources for Managers / D.J. Power. – Greenwood Publishing Group, 2002. – 272 p.

53. Zadeh, L. Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems / L.Zadeh, G.Klir, B.Yuan. – World Scientific Pub. Co. Inc., 1996. – 840 p.

ПРИЛОЖЕНИЕ А

Ключевые фрагменты кода программы

```
@RestController
@Api
public class ProblemInnerController {

    private final ProblemInnerService problemInnerService;

    @Autowired
    public ProblemInnerController (ProblemInnerService problemInnerService) {
        this. problemInnerService = problemInnerService;
    }

    @ApiOperation(value = "Получение списка проблем жителей.")
    @PostMapping(value = ProblemInnerEndpoints.LIST_PROBLEMS, produces
= MediaType.APPLICATION_JSON_VALUE)
   public RestResponse<ProblemInnerListResponse>> getProblemList
(@RequestBody ProblemInnerProblemListRequest request)
throws RestApiException {
        return new RestResponse<>(
problemInnerService.getProblemList(requestDto));
   }

@ApiOperation(value = "Получение  проблемы жителей по идентификатору.")
    @PostMapping(value = ProblemInnerEndpoints.PROBLEM_BY_ID, produces
= MediaType.APPLICATION_JSON_VALUE)
   public RestResponse<ProblemInnerByIdResponse>> getProblemById
(@RequestBody ProblemInnerByIdRequest request)
throws RestApiException {
      return new RestResponse<>(
problemInnerService.getProblemById(requestDto));
   }


@ApiOperation(value = "Необходимая роль: Administrator. Изменение статуса
проблемы, доступно только администратору системы.", produces =
MediaType.APPLICATION_JSON_VALUE,
```

```
authorizations = {@Authorization(value =
SwaggerConfig.securitySchemaOAuth2)})
   @PreAuthorize("hasRole('AdministratorRole')")
   @PostMapping(value = ProblemInnerEndpoints.CHANGE_STATUS, produces
= MediaType.APPLICATION_JSON_VALUE)
   public RestResponse<HttpResponseState>
changeProblemStatus(@RequestBody ProblemInnerChangeStatusDto requestDto)
throws RestApiException {return new RestResponse<>(
problemInnerService.changeProblemStatus(requestDto));
   }

@ApiOperation(value = "Необходимая роль: Administrator. Редактирование
атрибутов проблемы, доступно только администратору системы.", produces =
MediaType.APPLICATION_JSON_VALUE,
            authorizations = {@Authorization(value =
    SwaggerConfig.securitySchemaOAuth2)})
      @PreAuthorize("hasRole('AdministratorRole')")
      @PostMapping(value = ProblemInnerEndpoints.EDIT_PROBLEM,
    produces = MediaType.APPLICATION_JSON_VALUE)
      public RestResponse<HttpResponseState> editProblem(@RequestBody
    ProblemInnerEditProblemDto requestDto) throws RestApiException {
    return new RestResponse<>( problemInnerService.editProblem(requestDto));
      }

@ApiOperation(value = "Необходимая роль: Citizen. Создание новой
проблемы, доступно только жителям.", produces =
MediaType.APPLICATION_JSON_VALUE,
            authorizations = {@Authorization(value =
    SwaggerConfig.securitySchemaOAuth2)})
      @PreAuthorize("hasRole('CitizenRole')")
      @PostMapping(value = ProblemInnerEndpoints.CREATE_PROBLEM,
    produces = MediaType.APPLICATION_JSON_VALUE)
      public RestResponse<HttpResponseState> createProblem(@RequestBody
    ProblemInnerCreateProblemDto requestDto) throws RestApiException {
    return new RestResponse<>( problemInnerService.createProblem(requestDto));
      }
    }
```

```
public class ProblemInnerEndpoints {
   private static final String ROOT = "/problemInner";
   public static final String PROBLEM_BY_ID = ROOT + "ById/" + "{id}";
   public static final String CHANGE_STATUS = ROOT + "Status";
   public static final String EDIT_PROBLEM = ROOT + "Edit
   public static final String CREATE_PROBLEM= ROOT + "Create";
}

@RestController
@Api
public class ProblemExternalController {

   private final ProblemExternalService problemExternalService;

   @Autowired
   public ProblemExternalController (ProblemExternalService
problemExternalService) {
       this. problemExternalService = problemExternalService;
   }

   @ApiOperation(value = "Получение списка проблем жителей.")
   @PostMapping(value = ProblemExternalEndpoints.LIST_PROBLEMS,
produces = MediaType.APPLICATION_JSON_VALUE)
  public RestResponse<ProblemExternalListResponse>> getProblemList
(@RequestBody ProblemExternalProblemListRequest request)
throws RestApiException {
       return new RestResponse<>(
problemExternalService.getProblemList(requestDto));
  }

@ApiOperation(value = "Получение  проблемы жителей по
идентификатору.")
   @PostMapping(value = ProblemExternalEndpoints.PROBLEM_BY_ID,
produces = MediaType.APPLICATION_JSON_VALUE)
  public RestResponse<ProblemExternalByIdResponse>> getProblemById
(@RequestBody ProblemExternalByIdRequest request)
throws RestApiException {
     return new RestResponse<>(
problemExternalService.getProblemById(requestDto));
```

```
    }


@ApiOperation(value = "Необходимая роль: Administrator. Изменение
статуса проблемы, доступно только администратору системы.", produces =
MediaType.APPLICATION_JSON_VALUE,
authorizations = {@Authorization(value =
SwaggerConfig.securitySchemaOAuth2)})
    @PreAuthorize("hasRole('AdministratorRole')")
    @PostMapping(value = ProblemExternalEndpoints.CHANGE_STATUS,
produces = MediaType.APPLICATION_JSON_VALUE)
    public RestResponse<HttpResponseState>
changeProblemStatus(@RequestBody ProblemExternalChangeStatusDto
requestDto) throws RestApiException {return new RestResponse<>(
problemExternalService.changeProblemStatus(requestDto));
    }

@ApiOperation(value = "Необходимая роль: Administrator. Редактирование
атрибутов проблемы, доступно только администратору системы.",
produces = MediaType.APPLICATION_JSON_VALUE,
            authorizations = {@Authorization(value =
SwaggerConfig.securitySchemaOAuth2)})
    @PreAuthorize("hasRole('AdministratorRole')")
    @PostMapping(value = ProblemExternalEndpoints.EDIT_PROBLEM,
produces = MediaType.APPLICATION_JSON_VALUE)
    public RestResponse<HttpResponseState> editProblem(@RequestBody
ProblemExternalEditProblemDto requestDto) throws RestApiException {
return new RestResponse<>(
problemExternalService.editProblem(requestDto));
    }

@ApiOperation(value = "Необходимая роль: Citizen. Создание новой
проблемы, доступно только жителям.", produces =
MediaType.APPLICATION_JSON_VALUE,
            authorizations = {@Authorization(value =
SwaggerConfig.securitySchemaOAuth2)})
    @PreAuthorize("hasRole('CitizenRole')")
    @PostMapping(value = ProblemExternalEndpoints.CREATE_PROBLEM,
produces = MediaType.APPLICATION_JSON_VALUE)
```

```
    public RestResponse<HttpResponseState> createProblem(@RequestBody
ProblemExternalCreateProblemDto requestDto) throws RestApiException {
return new RestResponse<>(
problemExternalService.createProblem(requestDto));
    }
}

public class ProblemExternalEndpoints {
    private static final String ROOT = "/problemExternal";
    public static final String PROBLEM_BY_ID = ROOT + "ById/" + "{id}";
    public static final String CHANGE_STATUS = ROOT + "Status";
    public static final String EDIT_PROBLEM = ROOT + "Edit
    public static final String CREATE_PROBLEM= ROOT + "Create";
}
```