Чат-бот на основе микросервисов с самовосстановлением и поддержкой искусственного интеллекта для надежных диалоговых сервисов

Прешиоус Огенеоро Отуазохор

Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)

potuazokhor@stud.etu.ru

Адейе Адебусола Ияну

Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)

adebusolayeye@gmail.com

Аннотация. **Современные облачные приложения все чаще используют микросервисную архитектуру для создания масштабируемых и модульных программных систем, однако распределенный характер этих систем создает операционные проблемы, включая распространение ошибок, нестабильность сервисов и повышенную сложность мониторинга, особенно для разговорных платформ, где сбои в работе сервисов напрямую влияют на пользовательский опыт. В данной статье представлен проект и реализация самовосстанавливающейся микросервисной архитектуры на основе искусственного интеллекта для разговорного чат-бота о погоде, включающей шесть независимо развертываемых сервисов для понимания естественного языка, получения внешних данных о погоде, генерации ответов, взаимодействия с пользователем, централизованного логирования и обнаружения аномалий. Каждый сервис непрерывно передает телеметрические данные в централизованную базу данных PostgreSQL, в то время как модуль обнаружения аномалий анализирует эти данные с помощью модели Isolation Forest, обученной на поведенческих шаблонах каждого сервиса, включая задержку, частоту ошибок и шаблоны запросов. Когда модель выявляет отклонения от нормальной работы, она запускает автоматические действия по устранению неполадок через оркестратор на основе Docker, включая перезапуск контейнеров и адаптивное ограничение скорости, что позволяет системе восстанавливаться после сбоев без вмешательства человека. Прототипная реализация демонстрирует возможность объединения интеллектуального мониторинга с механизмами автономного восстановления, при этом ранние эксперименты показывают многообещающие возможности обнаружения и скорость восстановления при моделируемых сбоях сервисов. Эта работа вносит вклад в развивающуюся область отказоустойчивых микросервисных архитектур, демонстрируя, как машинное обучение может замкнуть цикл между обнаружением аномалий и автоматическим восстановлением, в конечном итоге повышая надежность приложений для диалогового взаимодействия в реальном времени.**

Ключевые слова: микросервисы, самовосстанавливающиеся системы, обнаружение аномалий, машинное обучение, разговорный ИИ, распределенные системы, отказоустойчивость, Isolation Forest

I. Введение

За последнее десятилетие разработка программного обеспечения претерпела фундаментальный архитектурный сдвиг, обусловленный широким распространением систем на основе микросервисов. В отличие от монолитных архитектур, где все функциональные компоненты интегрированы в единый развертываемый блок, микросервисы декомпозируют приложения на слабо связанные, независимо развертываемые сервисы. Эта парадигма обеспечивает улучшенную масштабируемость, более быстрые циклы разработки и лучшую изоляцию ошибок [1]. Однако она также вносит существенную операционную сложность, поскольку современная экосистема микросервисов может состоять из десятков взаимодействующих сервисов, каждый из которых генерирует свои собственные телеметрические данные, и сбои в одном сервисе могут быстро распространяться по всей системе [2].

Эти проблемы особенно критичны в разговорных системах и платформах чат-ботов, где перебои в работе сервиса или скачки задержки напрямую влияют на пользовательский опыт и доверие. Традиционные подходы к мониторингу на основе пороговых значений не адаптируются к изменяющимся рабочим нагрузкам и могут давать высокий уровень ложных срабатываний или игнорировать незначительные ухудшения производительности [3]. Поэтому возрастает потребность в интеллектуальных системах мониторинга, которые могут не только обнаруживать аномалии в режиме реального времени, но и автоматически инициировать корректирующие действия для поддержания непрерывности обслуживания.

Последние достижения в области машинного обучения сделали интеллектуальное обнаружение аномалий и автономное устранение проблем все более практичными. Самовосстанавливающиеся архитектуры объединяют непрерывный мониторинг, обнаружение аномалий, интеллектуальное принятие решений и автоматическое устранение проблем в единый цикл обратной связи [4]. Несмотря на растущий исследовательский интерес к самовосстанавливающимся микросервисным системам, ни одна из существующих работ не подтверждает эффективность таких подходов

на разговорных платформах, что является пробелом, мотивирующим данную работу.

В данной статье представлены проектирование, реализация и оценка самовосстанавливающегося чат-бота для прогноза погоды на основе искусственного интеллекта, построенного на шести контейнеризированных микросервисах, с централизованным сбором телеметрии, обнаружением аномалий в изолированном лесу для каждого сервиса и оркестратором восстановления на основе Docker, способным выполнять перезапуск контейнеров и адаптивное ограничение скорости. Эксперименты по контролируемому внедрению ошибок оценивают систему по четырем типам ошибок на трех сервисах, сообщая показатели обнаружения, MTTD, MTTR, точности, полноты и F1-меры.

II. Связанные работы

Самовосстанавливающиеся микросервисные архитектуры привлекают все большее внимание исследователей. Тутунджуоглу [4] предложил Phoenix Stack, развернув агенты мониторинга для каждого сервиса и глобальный уровень координации, достигнув сокращения MTTR на 68% с 92,1 секунды до 29,4 секунды и улучшения MTTD на 77% с 18,4 секунды до 4,2 секунды. Каул [9] продемонстрировал, что восстановление на основе обучения с подкреплением сокращает время простоя на 60% по сравнению с реактивными подходами за счет изучения оптимальных стратегий восстановления на основе прошлых инцидентов.

Что касается обнаружения, Найкаде [5] объединил Isolation Forest с прогнозированием LSTM, достигнув F1 = 0,863 в одномерных экспериментах. Джин и др. [6] применили Robust Principal Component Analysis с ансамблем Isolation Forest и One-Class SVM, получив 0,830 на бенчмарке AIOps. Нобре и др. [10] достигли 97% точности в обнаружении аномалий на уровне сервисов, используя многослойный персептрон. Раисзаде и др. [11] показали, что федеративное обучение с использованием графовых нейронных сетей улучшило F1 на 4% по сравнению с предыдущими методами, одновременно уменьшив накладные расходы на передачу данных в три-четыре раза.

Магаблех и Альмиани [7] продемонстрировали самовосстановление Docker Swarm с помощью выбора действий на основе полезности, достигнув точности обнаружения 97,1%. Несмотря на этот прогресс, большинство систем либо останавливаются на генерации оповещений, не замыкая цикл исправления, либо ограничивают восстановление базовыми перезапусками контейнеров, либо не имеют проверки на рабочих нагрузках с диалогами — ограничения, которые непосредственно устраняет данная работа.

III. Архитектура системы

A. Декомпозиция сервисов

Система включает шесть контейнеризированных сервисов, управляемых с помощью Docker Compose, как показано в табл. I. Каждый сервис отправляет каждый запрос в службу логирования по протоколу POST /log, фиксируя имя сервиса, конечную точку, метод, полезную нагрузку запроса и ответа, задержку в миллисекундах, статус и сообщения об ошибках. Эта унифицированная схема телеметрии позволяет проводить межсервисный анализ аномалий из единого источника данных PostgreSQL.

ТАБЛИЦА I. Архитектура системных сервисов

Услуга	Порт	Ответственность
внешний интерфейс	8000	HTML-интерфейс пользователя; организует конвейер обработки данных NLU-weather-responder
метеорологическая служба	8001	Оболочка REST API OpenWeatherMap
обнаружение аномалий	8002	Движок "Изоляционный лес"; запускает процесс исцеления.
logging_service	8004	Центральное хранилище телеметрии; исполнитель восстановления Docker.
nlu_service	8005	Извлечение города и даты из текста с помощью регулярных выражений
responder_service	8007	Генерация ответов LLM через API OpenRouter

B. Механизм обнаружения аномалий

Для каждой службы поддерживается независимая модель Isolation Forest. Алгоритм изолирует аномалии путем случайного разделения пространства признаков — аномальные наблюдения, будучи статистически редкими и удаленными от нормального распределения, требуют меньшего количества разделов для изоляции и получают более низкие оценки. Порог обнаружения установлен на уровне -0,15 при коэффициенте загрязнения = 0,05. Окно обучения охватывает логи за период от пяти до шестидесяти минут до начала обработки; окно обнаружения охватывает только последние пять минут, гарантируя, что обучение и обнаружение никогда не используют одни и те же данные. Из каждого лога извлекаются пять признаков: latency_ms, is_error (бинарный), response_size, hour_of_day и day_of_week. Модели переобучаются ежечасно и требуют не менее 30 обучающих выборок до начала обнаружения.

Аномалии классифицируются как service_error (is_error = 1), high_latency (задержка превышает среднее значение плюс три стандартных отклонения) или statistical_anomaly (многомерное отклонение). Каждая запись хранит упрощенное объяснение, содержащее оценку изоляции, значения признаков и удобочитаемые причины отклонения, обеспечивая основу для объяснимого ИИ в работе системы [8].

C. Организатор самоисцеления

Выбор действий осуществляется по принципу «тип прежде всего», где тип аномалии определяет класс ответа, а уровень серьезности повышается только в пределах этого класса. Ошибки сервиса с высокой степенью серьезности запускают команду restart_container через Docker SDK. Аномалии с высокой задержкой запускают команду rate_limit, снижая скорость пополнения хранилища токенов до 30% от обычных 10 запросов в секунду, что позволяет затронутому сервису восстановиться без дальнейшего увеличения нагрузки на трафик. Статистические аномалии запускают команду monitor — событие регистрируется, но никаких действий с контейнером не предпринимается, что позволяет избежать сбоев из-за некритических отклонений. Все действия по

восстановлению сохраняются с указанием ссылки на аномалию, типа действия, флага успеха и метки времени выполнения.

IV. Экспериментальная оценка

A. Экспериментальная установка

Эксперименты проводились на Windows 11 с использованием Docker Desktop. Перед каждым запуском через фронтенд выполнялось 60 реальных запросов к конвейеру обработки данных для создания аутентичных базовых показателей поведения для каждого сервиса. Создание таких базовых показателей на основе реального трафика имеет решающее значение: вызовы API LLM сервиса-ответчика естественным образом приводят к высокой задержке в 1-3 секунды, а модели, обученные на синтетических данных с задержкой 50-400 мс, систематически ошибочно классифицируют нормальные ответы LLM как аномальные, существенно завышая количество ложных срабатываний. В трех сервисах — weather_service, nlu_service и responder_service — в течение трех испытаний по 25 логов в каждом были внедрены четыре типа ошибок: высокая задержка (3000-8000 мс), ошибка сервиса (статус = ошибка), вспышка ошибок (первая половина — ошибка, вторая половина — нормальная) и смешанная (чередование высокой задержки и ошибки).

B. Результаты обнаружения и восстановления

В табл. II представлены показатели обнаружения, MTTD и MTTR для каждой службы. MTTD измеряется от момента внедрения до первой записи об аномалии; MTTR для restart_container измеряется от момента внедрения до подтвержденного восстановления конечной точки /health.

ТАБЛИЦА II. Показатели обнаружения неисправностей в зависимости от вида обслуживания (4 типа неисправностей по 3 попытки для каждого)

Услуга	Дет. Ставка	MTTD (с)	MTTR (с)	Аномалии
метеорологическая служба	91,7% (11/12)	6.8	7.8	93
nlu_service	50,0% (6/12)	7.9	7.6	31
responder_service	25,0% (3/12)	6.8	8.0	25
Общий	**55,6% (20/36)**	**7.1**	**7.8**	**149**

MTTR измеряет время от момента внедрения до подтвержденного восстановления работоспособности для действий перезапуска; для параметра rate_limit он измеряет время до начала устранения неполадок.

Метеорологическая служба показала наилучшие результаты с 91,7% обнаружений и самым низким средним временем задержки (MTTD) в 6,8 секунд, что отражает более стабильную и предсказуемую базовую задержку, точно смоделированную с помощью алгоритма Isolation Forest. Служба обработки естественного языка (NLU) обнаружила 50% внедренных ошибок, в то время как служба реагирования обнаружила 25%. Эти более низкие показатели объясняются двумя факторами: естественная изменчивость задержки LLM у службы реагирования затрудняет точную характеристику ее базовой линии, а механизм дедупликации идентификаторов журналов подавляет повторную пометку ранее оцененных журналов в последовательных испытаниях, иногда предотвращая вставку новых аномалий, когда окно обнаружения содержит данные предыдущих испытаний для той же службы. Во всех обнаруженных случаях MTTD варьировалось от 6,8 до 7,9 секунд, а MTTR — от 7,6 до 8,0 секунд, что указывает на стабильную скорость реагирования после обнаружения.

C. Показатели качества обнаружения

В табл. III представлены показатели точности, полноты и F1-меры, вычисленные по результатам всего эксперимента. Истинные положительные результаты — это записи об аномалиях, в журнале которых присутствует маркер синтетической инъекции; ложные положительные результаты — это обнаружения в журналах без инъекции; ложные отрицательные результаты — это журналы ошибок с инъекцией, не соответствующие ни одной записи об аномалии.

ТАБЛИЦА III. Качество обнаружения аномалий по видам услуг

Метрическая система	Метеорологическая служба	NLU svc	Служба реагирования	Общий
Количество испытаний (обнаружено / общее количество)	11 / 12	6 / 12	3 / 12	20 / 36
Пропущенные судебные процессы	1	6	9	16
Точность	0,831	0,789	0,714	0,776
Отзывать	0,917	0.500	0,250	0,556
F1-мера	0,872	0,612	0.370	0,645

Качество обнаружения значительно различалось в зависимости от службы. Метеорологическая служба показала лучшие результаты с точностью 0,831 и F1 0,872, что отражает более предсказуемый базовый уровень задержки. Служба NLU достигла F1 0,612, в то время как служба реагирования набрала 0,370 — более низкий показатель, обусловленный в основном недостатком полноты, а не чрезмерным количеством ложных срабатываний. Общая точность 0,776 отражает ложные срабатывания, возникающие из-за параметра загрязнения Isolation Forest (0,05), который по умолчанию помечает примерно 5% любых оцененных данных как аномальные; они сопоставляются с действиями монитора, а не с разрушительными перезапусками, что ограничивает их оперативное воздействие. Показатель F1 метеорологической службы 0,872 сопоставим с результатом ансамбля Найкаде 0,863 [5] и превосходит метод RPCA Джина и др. 0,830 [6].

D. Распределение лечебного действия

В табл. IV представлено распределение действий по восстановлению, выполненных за весь эксперимент. Всего было обнаружено 149 аномалий, каждая из которых запускала ровно одно действие по восстановлению, что обеспечивает соответствие 1:1 между записями об аномалиях и записями о восстановлении.

ТАБЛИЦА IV. РАСПРЕДЕЛЕНИЕ ЛЕЧЕБНОГО ДЕЙСТВИЯ

Действие	Считать	% от общего	Запущено по
перезапуск_контейнера	72	48,3%	service_error (высокая степень серьезности)
монитор	52	34,9%	статистическая аномалия (любой степени тяжести)
rate_limit	25	16,8%	высокая задержка (любой степени тяжести)
Общий	149	100%	—

Репертуар из трех действий демонстрирует, что система выходит за рамки простого перезапуска, напрямую устраняя пробел в ограниченном репертуаре действий в предыдущих работах [4]. Перезапуски контейнеров составляют 48,3% всех событий восстановления, что отражает тот факт, что аномалии service_error приводят к наиболее экстремальным отклонениям функций и наиболее надежно превышают порог высокой серьезности. Действие мониторинга, составляющее 34,9%, обрабатывает статистические аномалии, где обнаружено многомерное отклонение, но не применяется конкретная категория ошибки, регистрируя событие без прерывания работы. Ограничение скорости составляет 16,8% действий, снижая трафик до 30% от нормальной скорости без прерывания работы контейнера — соразмерный ответ на ошибки задержки, который сохраняет доступность сервиса, одновременно снижая нагрузку. Все 149 действий восстановления были инициированы без ошибок Docker или базы данных.

V. ОБСУЖДЕНИЕ

A. *Сравнение с предыдущими работами*

Phoenix Stack [4] сообщает о среднем времени обнаружения (MTTD) 4,2 секунды и среднем времени восстановления (MTTR) 29,4 секунды при использовании выделенных агентов мониторинга для каждого сервиса. Представленный здесь централизованный подход обеспечивает MTTD 6,8–7,9 секунды и MTTR 7,6–8,0 секунды для сервисов, где произошло обнаружение, при этом избегая накладных расходов на развертывание агентов для каждого сервиса. Разница в MTTR частично объясняется быстрыми локальными перезапусками контейнеров Docker Desktop; в производственных облачных средах с проблемами холодного запуска могут наблюдаться более высокие значения. Модель агентов для каждого сервиса обеспечивает меньшую задержку обнаружения за счет сложности инфраструктуры, в то время как централизованная телеметрия упрощает операции по мере роста числа сервисов.

По сравнению с системами, использующими только обнаружение, показатель F1-меры метеорологической службы, равный 0,872, сопоставим с ансамблевым подходом Найкаде (0,863) [5] и превосходит метод RPCA Джина и др. (0,830) [6]. Общий показатель F1, равный 0,645, отражает более низкую полноту в службах NLU и службе реагирования, что объясняется артефактом дедупликации, а не фундаментальным ограничением модели. Во всех обнаруженных неисправностях ложные срабатывания запускают только пассивное действие мониторинга, а не перезапуск контейнеров, что ограничивает сбои в работе.

B. *Ограничения и дальнейшая работа*

Необходимо отметить ряд ограничений. Стратегия дедупликации идентификаторов журналов может подавлять обнаружение повторяющихся ошибок в одной и той же записи журнала в течение последовательных циклов; окно дедупликации, основанное на времени, лучше сбалансировало бы корректность и скорость отклика. Оценка проводилась в условиях контролируемого внедрения синтетических ошибок; в реальных условиях развертывания используются более сложные и перекрывающиеся схемы ошибок. Параметр загрязнения 0,05 является фиксированным предположением, которое может не отражать фактическую частоту аномалий в производственной среде.

Ключевой практический вывод заключается в том, что модели для каждой службы должны обучаться на реальном рабочем трафике, а не на синтетических данных. Естественная высокая задержка LLM службы-ответчика систематически неправильно классифицировалась при обучении моделей на искусственно созданных данных с низкой задержкой, что существенно завышало количество ложных срабатываний. Это понимание в целом применимо к любой самовосстанавливающейся системе, работающей с генеративным ИИ или внешними API-зависимостями.

Дальнейшая работа будет сосредоточена на временной дедупликации, объяснимости решений по восстановлению на основе SHAP [8] и обучении с подкреплением для выбора действий — заменив существующую политику, основанную на правилах, на политику, основанную на обучении, которая может адаптироваться к меняющимся условиям эксплуатации и обобщаться на типы неисправностей, не встречавшиеся во время обучения.

VI. ЗАКЛЮЧЕНИЕ

В данной статье представлена самовосстанавливающаяся микросервисная архитектура на основе ИИ для разговорного чат-бота о погоде, демонстрирующая, что интеллектуальное обнаружение аномалий и автоматическое устранение могут быть интегрированы в распределенную систему из шести сервисов с минимальными инженерными затратами. Механизм обнаружения Isolation Forest, обученный на базовых поведенческих моделях для каждого сервиса из реального трафика, достиг общего показателя F1-меры 0,645, при этом показатель для сервиса погоды составил 0,872 — сопоставимо с передовыми ансамблевыми методами [5]. Оркестратор восстановления выполнил три различных типа действий — перезапуск контейнера, адаптивное ограничение скорости и мониторинг — в рамках 149 событий восстановления, демонстрируя более широкий репертуар устранения проблем, чем большинство предыдущих реализаций с одним действием.

Система обеспечивает среднее время обнаружения (MTTD) 6,8–7,9 секунд и среднее время восстановления (MTTR) 7,6–8,0 секунд для сервисов, где произошло обнаружение, показатели, в целом сопоставимые с

MTTD Phoenix Stack в 4,2 секунды [4], при этом не требуя инфраструктуры агентов для каждого сервиса. В этой работе впервые подтверждена работоспособность механизмов самовосстановления на платформе разговорных микросервисов, что подтверждает перенос моделей обнаружения и устранения проблем на разговорные рабочие нагрузки при условии, что базовые модели обучены на реальном операционном трафике — это особенно важно, когда сервисы зависят от внешних API генеративного ИИ с естественно высокой и переменной задержкой.

Список литературы

[1] Велепуча В., Флорес П. Обзор архитектуры микросервисов: принципы, шаблоны и проблемы миграции // IEEE Access, 2023, том 11, стр. 88339-88358.

[2] Parida RP, Singh STJ, Selvaraj A. Автоматизированное обнаружение аномалий в микросервисах в реальном времени с использованием передовых методов ИИ/машинного обучения // J. Artif. Intell. Res. Appl., 2023, vol. 3, no. 1.

[3] Ханахмади М. и др. Обнаружение программных аномалий на основе микросервисов с помощью OpenTracing в облаке // Softw. Pract. Exp., 2023, том 53, № 8, стр. 1681-1699.

[4] Тутунджуоглу ВТ Phoenix Stack: Самовосстанавливающаяся микросервисная архитектура для веб-приложений реального времени. 2025.

[5] Найкаде П.П. Автоматизированное обнаружение и локализация аномалий для облачных систем на основе микросервисов. 2020.

[6] Jin M. et al. An anomaly detection algorithm for microservice architecture based on RPCA // IEEE Access, 2020, vol. 8, pp. 226397-226408.

[7] Магаблех Б., Альмиани М. Самовосстанавливающаяся архитектура микросервисов: пример использования Docker Swarm. Adv. Inf. Netw. Appl., Springer, 2020, стр. 846-858.

[8] Гоял Б. Объяснимый ИИ (XAI) для корпоративных приложений на базе облачных вычислений. 2025.

[9] Каул Д. Механизмы обнаружения неисправностей и самовосстановления на основе ИИ в микросервисах для распределенных облачных сред. 2020.

[10] [10] Nobre J., Pires EJS, Reis A. Anomaly detection in microservice-based systems //Appl. Sci., 2023, vol. 13, no. 13, p. 7891.

[11] Raeiszadeh M. et al. Асинхронное федеративное обучение в реальном времени для обнаружения аномалий в облачных приложениях микросервисов // IEEE Trans. Mach. Learn. Commun. Netw., 2025, vol. 3, pp. 176-194