Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 23.М07-мм

Логистический портал Cargotime.ru: модификация внутренней системы мониторинга процесса сбора данных из открытых источников

СЧЕРЕВСКИЙ Виктор Максимович

Отчёт по учебной практике
в форме «Производственное задание»

Научный руководитель:
доцент кафедры информатики СПбГУ, к. т. н., Абрамов М.В.

Консультант:
м. н. с. лаб. ТиМПИ СПб ФИЦ РАН, Есин М.С.

Санкт-Петербург
2024

Оглавление

Введение

Логистика – это ключевой элемент современной экономики. От эффективной доставки товаров до управления запасами и складской логистики, она играет важную роль в обеспечении бесперебойного функционирования бизнеса. В условиях глобализации и нестабильности мировых рынков, такие аспекты как гибкость и адаптивность логистических систем становятся ключевыми для поддержания конкурентоспособности. Современная логистика не только удовлетворяет текущие потребности бизнеса, но и активно формирует будущее глобальных цепочек поставок.

Автоматизация и цифровизация процессов позволяют компаниям оперативно реагировать на изменения спроса, оптимизировать маршруты транспортировки, управлять запасами в режиме реального времени и снижать издержки. Также с целью облегчения доступа к своим услугам, многие логистические компании создают так называемые автоматизированные калькуляторы доставки.

Логистический портал Cargotime.ru [10], развивающийся на базе лаборатории теоретических и междисциплинарных проблем информатики (ТиМПИ) СПб ФИЦ РАН, предоставляет возможность упрощенного и автоматизированного доступа к информации о доставке грузов и других коммерческих предложениях для более, чем 40 различных компаний. В том числе портал предлагает сервисы отслеживания контейнеров, посылок и калькулятора доставки. Эти сервисы работают на базе парсеров веб-сайтов компаний, которые по различным причинам могут выходить из строя. В связи с этим для сервиса отслеживания контейнеров была разработана система мониторинга. Однако первичная реализация системы имеет ряд недостатков: результаты проверок недостаточно информативны, избыточно нагружаются работающие парсеры, а также нередки случаи их ложного отключения. В этой работе предлагается модифицировать существующую систему мониторинга и встроить ее в сервис калькулятора доставки грузов.

1. Постановка задачи

Целью работы является повышение стабильности работы сервиса оценки стоимости доставки путем модификации и адаптации существующей системы мониторинга. Для её выполнения были поставлены следующие задачи:

- Разработать расширенную классификацию ошибок парсеров, позволяющую уточнить состояние парсера;
- Модифицировать алгоритм принятия решений об отключении парсеров с целью повысить информативность отчёта;
- Оптимизировать использование ресурсов ядра калькуляторов путем внедрения нового типа запросов;
- Внедрить обновленную систему мониторинга в сервис расчёта стоимости доставки.

2. Обзор сервиса

В данном разделе будут приведены высокоуровневое описание архитектуры сервиса калькулятора доставки, а также внутреннее устройство системы мониторинга.

2.1. Общая архитектура

Часть общей архитектуры портала, связанная с сервисом калькулятора доставки, выглядит следующим образом:

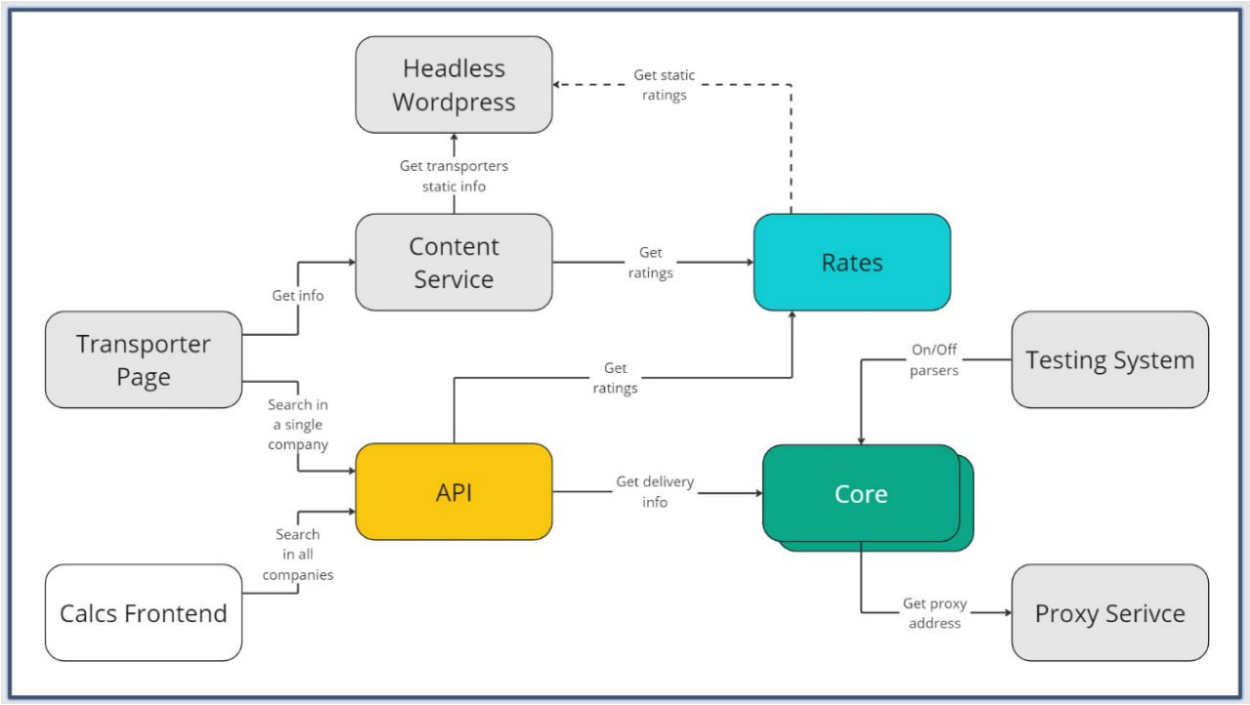

Рис. 1: Архитектура сервиса калькулятора доставки

Описание компонентов:

- **Core** – ядро парсеров, в котором обрабатываются все запросы к источникам. Данный сервис реплицируется.
- **API** – промежуточный сервис между веб-интерфейсом и парсерами. В нём агрегируется и кэшируется метаинформация о компаниях и направляется клиенту вместе с ответами **Core**.

- **Rates** – данный сервис считывает рейтинги компаний, агрегируя отзывы и другие показатели. Основное назначение сервиса заключается в ранжировании результатов для отображения клиенту.

- **Proxy Service** – оптимально распределяет динамические и статические порты для парсинга.

- **Headless Wordpress, Content Service** – сервисы, предназначенные для хранения и преобразования данных о компаниях. Используются преимущественно для создания каталога перевозчиков и публикации постов.

- **Transporter Page, Calcs Frontend** – веб-интерфейс сервиса калькулятора доставки.

- **Testing System** – сервис, который должен осуществлять мониторинг калькулятора доставки: своевременно обнаруживать неработающие парсеры, а также автоматически их выключать и включать.

Основная часть данной работы касается именно разработки сервиса **Testing System**.

2.2. Используемые технологии

Ниже перечислены основные программные технологии, используемые в сервисе калькулятора доставки:

- **Node.js** [6] – программная платформа с открытым исходным кодом, предназначенная для разработки серверных приложений на языке JavaScript.

- **TypeScript** [9] – строго типизированный язык программирования с открытым исходным кодом, компилирующийся в JavaScript и расширяющий его возможности. Повсеместно применяется в проекте для построения серверных приложений.

- **NestJS** [5] – современный фреймворк с открытым исходным кодом для создания производительных и масштабируемых серверных приложений на Node.js. Поддерживает разработку на языках JavaScript и TypeScript. С использованием данного фреймворка были построены сервисы ядра парсеров и мониторинговой системы.

- **RabbitMQ** [7] – открытая система обмена сообщениями (брокер сообщений), реализующая протокол AMQP. С помощью RabbitMQ производится обмен данными между ядром парсеров и мониторинговой системой.

- **Redis** [8] – высокопроизводительная in-memory (в оперативной памяти) база данных с доступным исходным кодом (source-available). Используется для кэширования данных в сервисе ядра парсеров.

- **ELK (Elasticsearch, Logstash, Kibana)** [3] – стек технологий для поиска, анализа и визуализации логов. В системе мониторинга Elasticsearch используется для поиска информации о поломках парсеров.

- **MySQL** [4] – свободная реляционная СУБД. Используется в сервисе ядра парсеров для хранения информации о компаниях.

- **Docker** [2] – открытая платформа для разработки, доставки и запуска приложений в контейнерах. В сервисах ядра парсеров и системы мониторинга также используется Docker Compose для автоматизированного развертывания контейнеров в единой сети.

2.3. Архитектура системы мониторинга

Сервис системы мониторинга состоит из нескольких модулей, схемы которых изображены на рисунках ниже:

- **AppModule** – корневой модуль, служащий точкой входа в приложение.

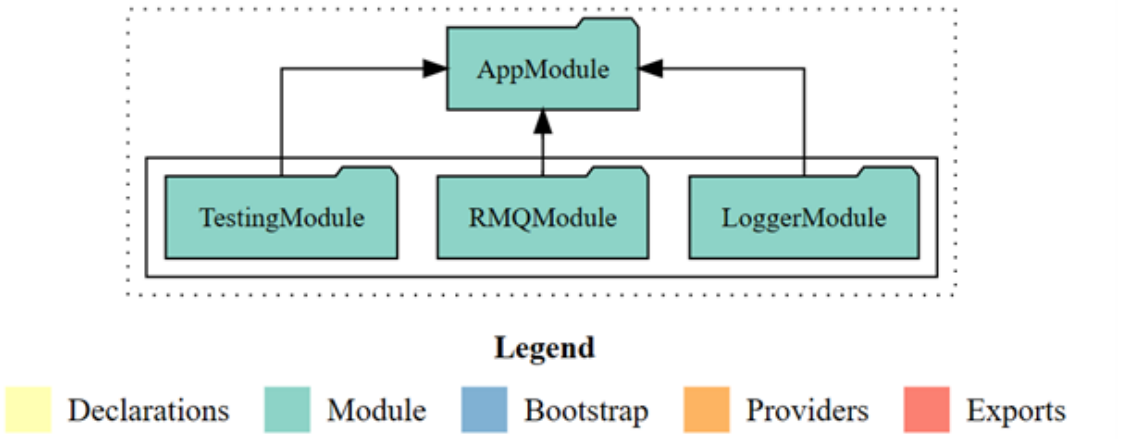

Рис. 2: AppModule

- **RMQModule** – инкапсулирует логику для работы с RabbitMQ.

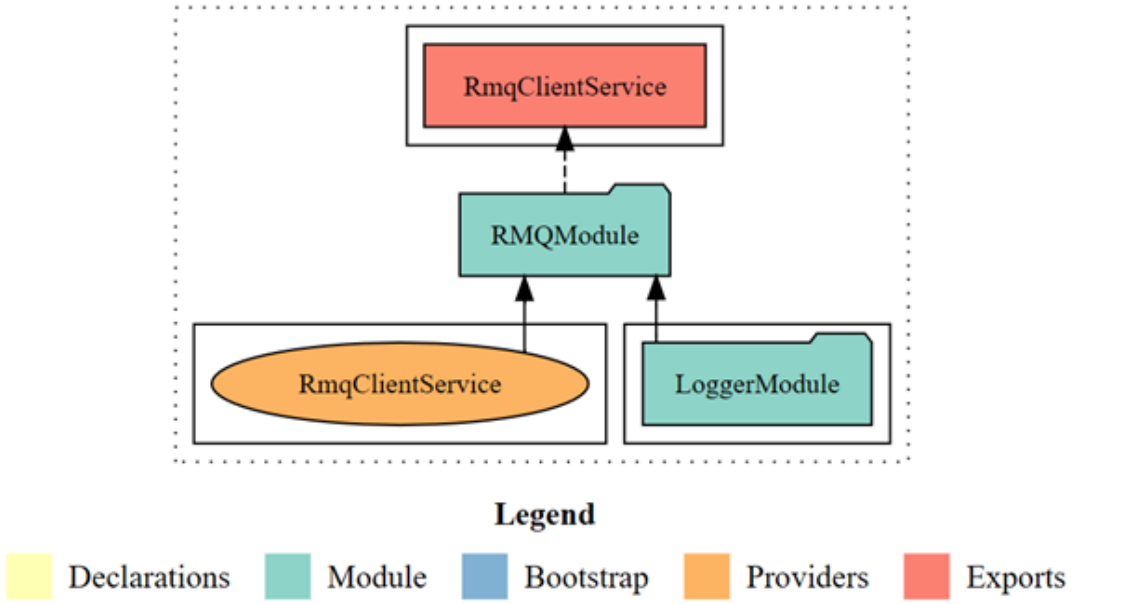

Рис. 3: RMQModule

- **ElasticClientModule** – содержит методы для запросов информации о парсерах в ElasticSearch.

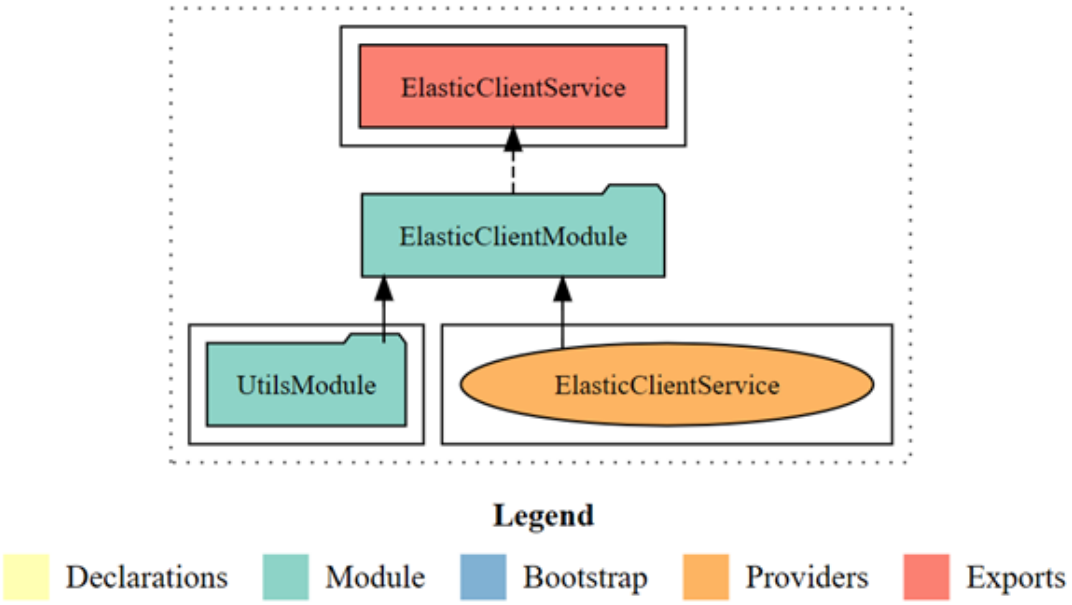

Рис. 4: ElasticClientModule

- **LoggerModule** – включает инструменты для единообразного логирования.

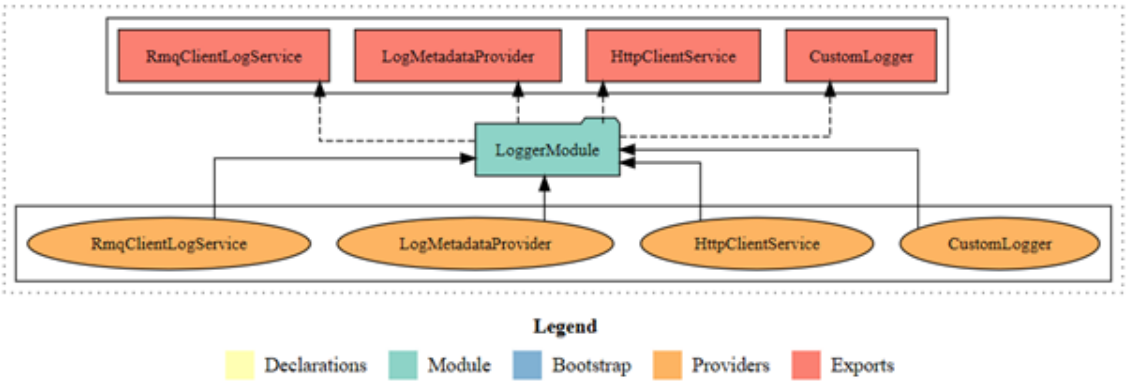

Рис. 5: LoggerModule (упрощённая схема)

- **UtilsModule** – набор вспомогательных утилит для взаимодействия с ядром парсеров.

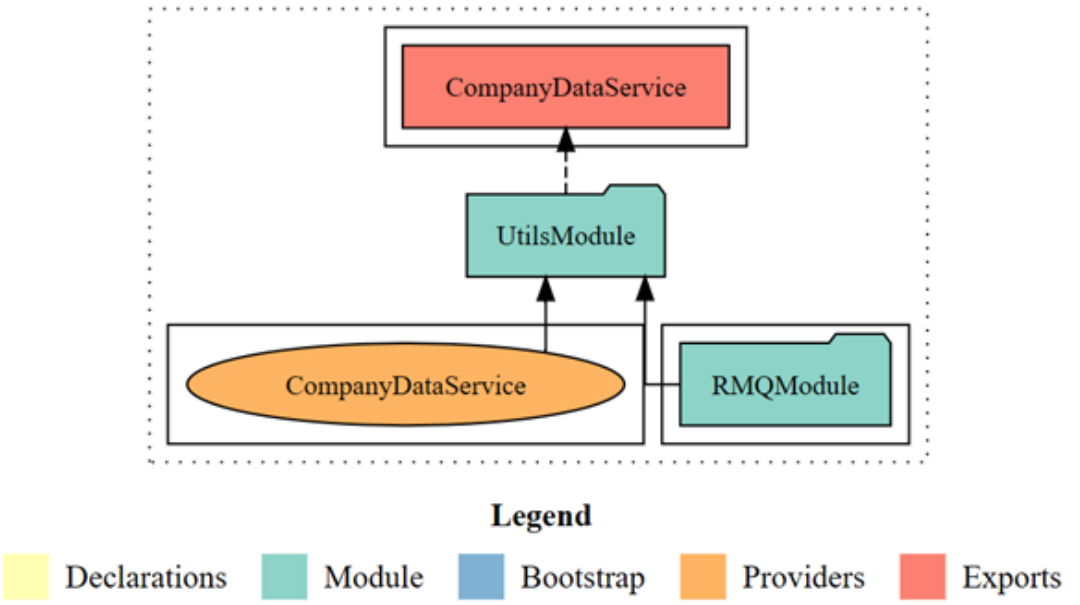

Рис. 6: UtilsModule

- **TestingModule** – содержит алгоритмы мониторинга парсеров.

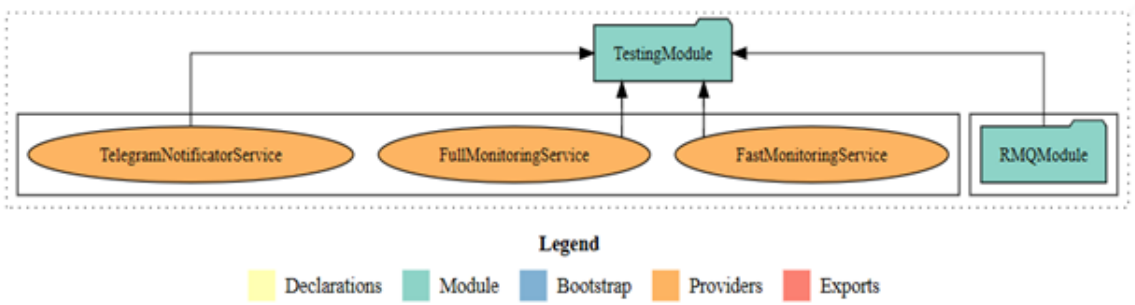

Рис. 7: TestingModule (упрощённая схема)

Для отрисовки схем зависимостей использовался инструмент Compodoc [1].

3. Реализация проекта

В этой главе будут описаны изменения, касающиеся непосредственно доработок в алгоритме мониторинга парсеров.

3.1. Классификация ошибок

С целью повысить информативность ответов ядра парсеров была разработана расширенная классификация ошибок. Все ошибки, влияющие на включение и отключение парсера, были разбиты на 4 группы:

- ошибки на стороне источника
  - SOURCE_ERROR – пустой ответ источника или любой статус-код, отличный от 200;
  - USER_TIMEOUT_ERROR – истечение времени на обработку запроса;
- ошибки во время парсинга ответа
  - UNEXPECTED_ERROR – случайная ошибка в парсере (например, обращение к undefined-полю или к элементу пустого списка);
- ошибки типизации ответа
  - RESPONSE_VALIDATION_ERROR – ошибка валидации выходных данных парсера (например, отправка строки вместо числа);
- внутренние ошибки
  - PACKING_ERROR – внутренняя ошибка алгоритма упаковки;
  - COMPANY_IS_UNDER_MAINTENANCE - компания отключена.

Получение любой из этих ошибок сигнализирует о некорректной работе парсера, что делает его кандидатом на отключение. Также был введён ряд информативных статусов, которые не являются причиной для отключения парсера и игнорируются мониторинговой системой:

- INFO_NOT_FOUND – всё отработало штатно, но у компании нет предложений;
- CARGO_LIMITS_EXCEEDED – превышены лимиты груза или партии;
- UNSUPPORTED_LOCATION – указанная локация не поддерживается компанией;
- UNSUPPORTED_SERVICES – указанные дополнительные услуги не поддерживаются компанией.

3.2. Модификация алгоритма

Первая версия алгоритма проверки парсеров была реализована в системе мониторинга трекинга контейнеров. Схема этого алгоритма представлена на рисунке ниже.

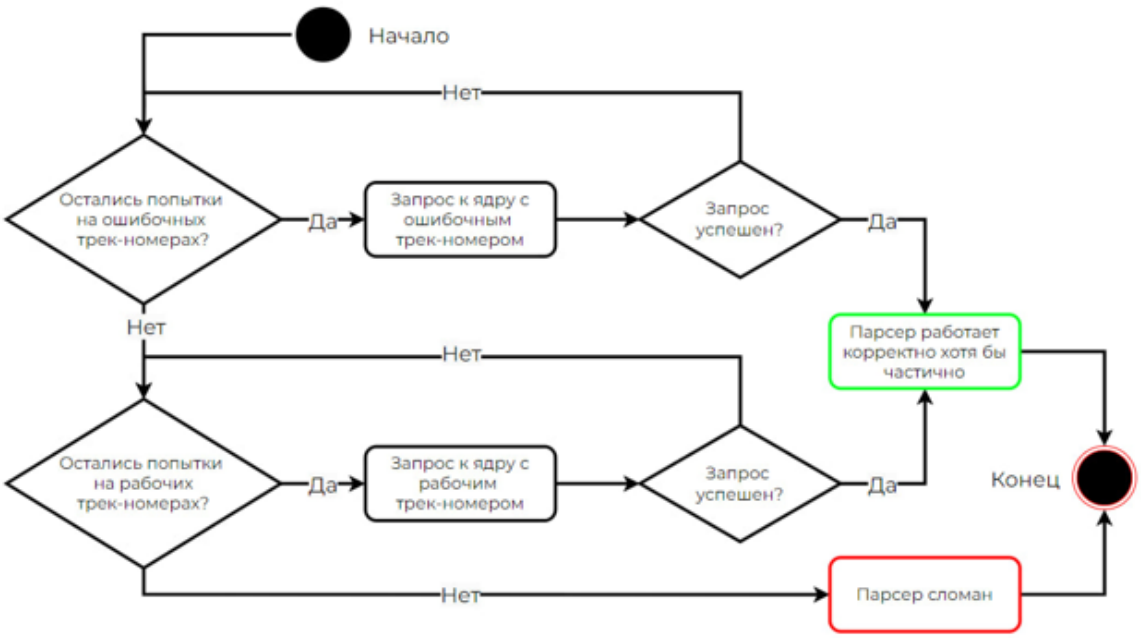

Рис. 8: Первая версия алгоритма

Данный алгоритм принимает на вход некоторое количество ошибочных и рабочих трек-номеров. Далее он посылает запросы к ядру, проверяя сначала ошибочные трек-номера, а затем рабочие. Если хотя бы один запрос прошел успешно, то парсер помечается, как “рабочий хотя бы частично”, и алгоритм завершается. Если же все трек-номера были проверены, и ни один запрос не прошел успешно, то парсер считается сломанным и подлежит отключению.

При такой реализации алгоритм имеет ряд недостатков: проверка является неполной, т.к. при одном успешном запросе пропускаются оставшиеся трек-номера, а также не учитывается текущее состояние парсера. С целью их устранения была предложена модификация этого алгоритма. Его новая версия представлена на рисунке ниже.

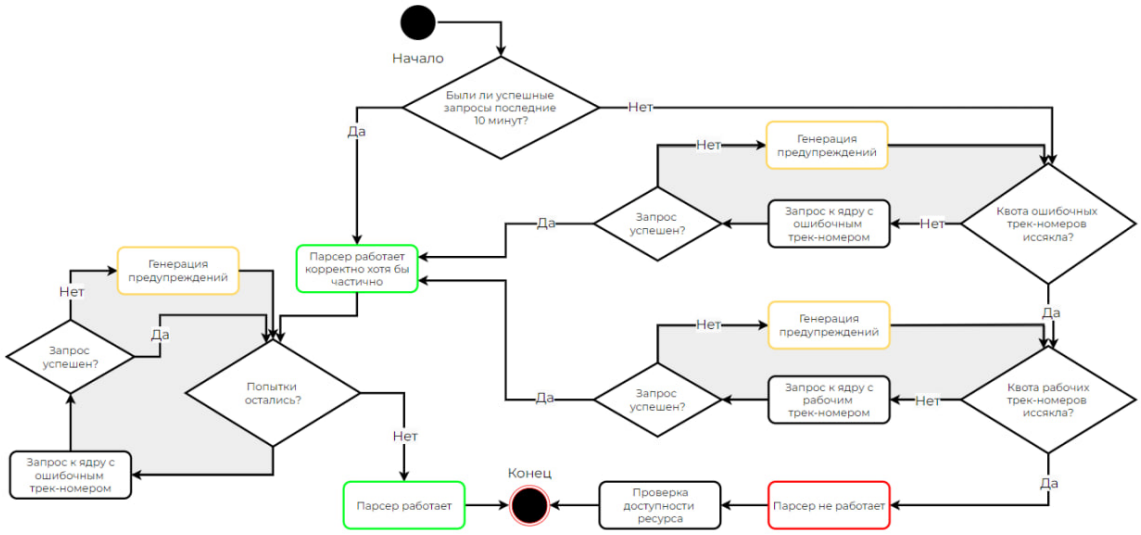

Рис. 9: Вторая версия алгоритма

В отличие от предыдущей реализации, здесь при одном успешном запросе алгоритм не завершается, а попадает в цикл, где проверяются оставшиеся трек-номера. В этот же цикл сразу попадают парсеры, у которых были успешные запросы за последние 10 минут. Также дополнительно генерируются предупреждения в случае неуспешных запросов.

Алгоритм мониторинга в сервисе калькулятора доставки грузов работает аналогичным образом.

3.3. Оптимизация использования ресурсов

Изначально сервис ядра парсеров поддерживал 2 типа клиентских запросов:

- запросы, направленные сразу ко всем компаниям, в которых поступает только информация об одном грузе;
- запросы, направленные только к одной компании, в которых поступает информация об одном грузе и об одной компании.

Первый тип запросов является довольно тяжелым, поскольку нагружает сразу все парсеры. В системе мониторинга его использовать не оптимально, поскольку нет необходимости проверять те парсеры, про которые достоверно известно, что они рабочие.

Второй тип запросов более гибкий, т.к. с его помощью можно точечно проверить конкретный парсер с конкретным грузом. Также запросы этого типа равнозначны по трудоемкости, что позволяет балансировать нагрузку на парсеры. Однако при активной работе мониторинговой системы может возникнуть такая ситуация, что в очереди скапливается слишком большое число запросов. Для того, чтобы это оптимизировать, был предложен новый тип запросов.

Запрос нового типа позволяет отсылать несколько грузов к различным компаниям, но при условии, что в записях каждая компания присутствует не более, чем 1 раз. Грузы, в свою очередь, могут повторяться. Пример такого запроса:

(груз А, компания А)
(груз А, компания Б)
(груз Б, компания В)
(груз В, компания Г)
(груз А, компания Д)
(груз Б, компания Е)

При таком подходе нагрузка на парсеры будет распределяться равномерно, а количество запросов в очереди кратно сократится.

3.4. Внедрение

Система мониторинга сервиса калькуляторов разрабатывалась, как отдельный сервис, расширяющий функциональность тестирующей системы трекинга контейнеров, а также исправляющий её недочеты. Система была внедрена в сервис калькуляторов и показала свою работоспособность на тестовых данных.

Однако стоит отметить, что разработка велась параллельно с разработкой ядра парсеров, и на данный момент ещё не собрано достаточное количество данных, позволяющих провести более подробный анализ. В случае, если разработанная система покажет свою эффективность и на реальных данных, то она также будет внедрена в сервис трекинга контейнеров.

Заключение

Целью работы являлось повышение стабильности работы сервиса оценки стоимости доставки путем модификации и адаптации существующей системы мониторинга.

Для достижения цели в ходе работы были выполнены следующие задачи:

- Разработана расширенная классификация ошибок парсеров, позволяющая уточнить состояние парсера;
- Модифицирован алгоритм принятия решений об отключении парсеров с целью повысить информативность отчёта;
- Внедрён новый тип запросов, оптимизирующий использование ресурсов ядра калькуляторов;
- Обновленная система мониторинга внедрена в сервис расчета стоимости доставки.

Стабильность работы сервиса оценки стоимости доставки была повышена за счёт своевременного включения и отключения парсеров, уменьшения нагрузки на ядро парсеров в процессе работы мониторинговой системы, а также сокращения числа ошибочных отключений парсеров.

В дальнейшем планируется провести более подробный анализ работы тестирующей системы сервиса калькулятора доставки на реальных данных, а также внедрить её в другие сервисы портала Cargotime.ru.

Список литературы

[1] Compodoc. — URL: https://compodoc.app/ (дата обращения: 11 июня 2024 г.).

[2] Docker. — URL: https://www.docker.com/ (дата обращения: 11 июня 2024 г.).

[3] ELK Stack. — URL: https://www.elastic.co/elastic-stack (дата обращения: 11 июня 2024 г.).

[4] MySQL. — URL: https://www.mysql.com/ (дата обращения: 11 июня 2024 г.).

[5] NestJS. — URL: https://nestjs.com/ (дата обращения: 11 июня 2024 г.).

[6] Node.js. — URL: https://nodejs.org/en (дата обращения: 11 июня 2024 г.).

[7] RabbitMQ. — URL: https://www.rabbitmq.com/ (дата обращения: 11 июня 2024 г.).

[8] Redis. — URL: https://redis.io/ (дата обращения: 11 июня 2024 г.).

[9] TypeScript. — URL: https://www.typescriptlang.org/ (дата обращения: 11 июня 2024 г.).

[10] Логистический портал Cargotime. — URL: https://cargotime.ru/ (дата обращения: 11 июня 2024 г.).